



# AN85951 - PSoC<sup>®</sup> 4 CapSense<sup>®</sup> Design Guide

Doc. No. 001-85951 Rev. \*N

Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709  
Phone (USA): 800.858.1810  
Phone (Intl): 408.943.2600  
[www.cypress.com](http://www.cypress.com)

**Copyrights**

© Cypress Semiconductor Corporation, 2013-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, lifesaving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

**Trademarks**

PSoC Designer™, PSoC Creator™ and Programmable System-on-Chip™ are trademarks and PSoC® and CapSense® are registered trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

**Source Code**

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

**Disclaimer**

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

# Contents



<b>1. Introduction.....</b>	<b>6</b>
1.1 Abstract .....	6
1.2 Introduction.....	6
1.3 PSoC 4 and PProC BLE CapSense Features .....	7
1.3.1 Additional CapSense Features in PSoC 4-S Series .....	7
1.4 PSoC 4 and PProC BLE CapSense Plus Features .....	7
1.5 CapSense Design Flow .....	9
<b>2. CapSense Technology .....</b>	<b>11</b>
2.1 CapSense Fundamentals .....	11
2.1.1 Self-Capacitance Sensing .....	13
2.1.2 Mutual-Capacitance Sensing.....	14
2.2 Capacitive Touch Sensing Method .....	15
2.2.1 CapSense Sigma Delta (CSD) .....	15
2.2.2 CapSense Crosspoint (CSX) .....	16
2.3 Signal-to-Noise Ratio.....	17
2.4 CapSense Widgets.....	17
2.4.1 Buttons (Zero-Dimensional).....	18
2.4.2 Sliders (One-Dimensional).....	19
2.4.3 Touchpads / Trackpads (Two-Dimensional) .....	20
2.4.4 Proximity (Three-Dimensional) .....	20
2.5 Liquid Tolerance .....	21
2.5.1 Effect of Liquid Droplets and Liquid Stream on a CapSense Sensor.....	22
2.5.2 Driven-Shield Signal and Shield Electrode .....	24
2.5.3 Guard Sensor .....	24
<b>3. PSoC 4 and PProC BLE CapSense .....</b>	<b>27</b>
3.1 CapSense Sensing.....	27
3.1.1 CapSense Architecture in PSoC 4 and PProC BLE.....	27
3.1.2 CapSense Architecture in PSoC 4 S-Series .....	38
3.2 CapSense in PSoC 4xxxM/4xxxL-Series.....	41
<b>4. CapSense Design and Development Tools.....</b>	<b>43</b>
4.1 PSoC Creator .....	43
4.1.1 CapSense Component .....	43
4.1.2 CapSense_ADC Component.....	44
4.1.3 Tuner Helper.....	44
4.1.4 Example Projects.....	45
4.2 Hardware Kits .....	45

<b>5. CapSense Performance Tuning .....</b>	<b>47</b>
5.1 Selecting between SmartSense and Manual Tuning .....	47
5.2 SmartSense .....	47
5.2.1 Component Configuration for SmartSense .....	48
5.3 Manual Tuning .....	64
5.3.1 Overview .....	64
5.3.2 CSD sensing method .....	65
5.3.7 CSX sensing method .....	97
5.3.8 Manual Tuning Trade-offs .....	104
5.3.9 Tuning Debug FAQs .....	106
<b>6. Design Considerations .....</b>	<b>109</b>
6.1 Firmware .....	109
6.1.1 Low-Power Design .....	110
6.2 Sensor Construction .....	111
6.3 Overlay Selection .....	113
6.3.1 Overlay Material .....	113
6.3.2 Overlay Thickness .....	113
6.3.3 Overlay Adhesives .....	114
6.4 PCB Layout Guidelines .....	114
6.4.1 Parasitic Capacitance, $C_p$ .....	114
6.4.2 Board Layers .....	114
6.4.3 Slider Design .....	117
6.4.4 Sensor and Device Placement .....	124
6.4.5 Trace Length and Width .....	124
6.4.6 Trace Routing .....	124
6.4.7 Crosstalk Solutions .....	125
6.4.8 Vias .....	126
6.4.9 Ground Plane .....	126
6.4.10 Power Supply Layout Recommendations .....	130
6.4.11 Layout Guidelines for Liquid Tolerance .....	131
6.4.12 Schematic Rule Checklist .....	133
6.4.13 Layout Rule Checklist .....	134
6.5 ESD Protection .....	136
6.5.1 Preventing ESD Discharge .....	137
6.5.2 Redirect .....	137
6.5.3 ESD Protection Devices .....	138
6.6 Electromagnetic Compatibility (EMC) Considerations .....	138
6.6.1 Radiated Interference and Emissions .....	138
6.6.2 Conducted RF Noise .....	147
<b>7. CapSense Plus .....</b>	<b>148</b>
<b>8. Resources .....</b>	<b>151</b>
8.1 Website .....	151
8.2 Datasheet .....	151
8.3 Technical Reference Manual .....	151

8.4	Development Kits .....	151
8.5	PSoC Creator .....	152
8.6	Application Notes.....	152
8.7	Design Support.....	152
<b>Glossary.....</b>		<b>153</b>
<b>Revision History.....</b>		<b>158</b>

# 1. Introduction



## 1.1 Abstract

The PSoC® 4 CapSense® Design Guide shows how to design capacitive touch sensing applications with the CapSense feature in PSoC 4 and PRoC™ Bluetooth Low Energy (BLE) device families. The CapSense feature in PSoC 4 and PRoC BLE offers unprecedented signal-to-noise ratio (SNR), best-in-class liquid tolerance, and a wide variety of sensors such as buttons, sliders, trackpads, and proximity sensors. This guide explains the CapSense operation, CapSense design tools, performance tuning of the PSoC Creator™ CapSense Component and design considerations.

Cypress provides different device families with the CapSense feature. If you have not chosen a particular device, or are new to capacitive sensing, refer to the [Getting Started with CapSense Design Guide](#). It helps you understand the advantages of CapSense over mechanical buttons, CapSense technology fundamentals, and selecting the right device for your application. It also directs you to the right documentation, kits, or tools to help with your design.

## 1.2 Introduction

Capacitive touch sensors are user interface devices that use human body capacitance to detect the presence of a finger on or near a sensor. Cypress CapSense solutions bring elegant, reliable, and easy-to-use capacitive touch sensing functionality to your product.

This design guide focuses on the CapSense feature in the PSoC 4 and PRoC BLE families of devices. These are true programmable embedded system-on-chip, integrating configurable analog and digital peripheral functions, memory, radio, and a microcontroller on a single chip. These devices are highly flexible and can implement many functions such as ADC, DAC, and BLE in addition to CapSense, which accelerates time-to-market, integrates critical system functions, and reduces overall system cost.

This guide assumes that you are familiar with developing applications for PSoC 4 or PRoC BLE using the Cypress PSoC Creator IDE. If you are new to PSoC 4, an introduction can be found in [AN79953, Getting Started with PSoC 4](#) or [AN92167, Getting Started with PSoC 4 BLE](#). If you are new to PRoC BLE, take a look at [AN94020, Getting Started with PRoC BLE](#). If you are new to PSoC Creator, see the [PSoC Creator home page](#).

This design guide helps you understand:

- [CapSense technology in PSoC 4 and PRoC BLE](#)
- [Design and development tools available for PSoC 4 and PRoC BLE CapSense](#)
- [CapSense PCB layout guidelines for PSoC 4 and PRoC BLE](#)
- [Performance tuning of PSoC 4 and PRoC BLE CapSense Component](#)
- [Applications using CapSense Plus™ features such as Motor Control Systems and Induction Cookers](#)

## 1.3 PSoC 4 and PRoC BLE CapSense Features

CapSense in PSoC 4 and PRoC BLE has the following features:

- Supports self-capacitance and mutual-capacitance based touch sensing
- Robust [CapSense Sigma Delta \(CSD\)](#) and [CapSense Crosspoint \(CSX\)](#) sensing technologies that provides best-in-class [Signal-to-Noise Ratio](#) for self-capacitance and mutual-capacitance based touch sensing respectively
- High-performance sensing across a variety of overlay materials and varied thickness (see [CapSense Fundamentals](#), [Overlay Material](#), and [Overlay Thickness](#))
- [SmartSense™](#) Auto-tuning technology
- High-range proximity sensing (up to a 30-cm proximity-sensing distance)
- Liquid-tolerant operation (see [Liquid Tolerance](#))
- Pseudo random sequence (PRS) clock source for lower electromagnetic interference (EMI)
- Low power consumption with as low as 1.71-V operation and as low as 150-nA current consumption in Hibernate mode
- Supports Capacitive Sensing and Shielding on all GPIO pins

### 1.3.1 Additional CapSense Features in PSoC 4-S Series

The PSoC 4 S family supports the fourth-generation CapSense, which is an updated version of CapSense in PSoC 4 and PRoC BLE devices. PSoC 4 S Series family has following additional features to previous PSoC 4 and PRoC BLE families:

- Allows CapSense block re-configuration as an ADC, and supports ADC input on any GPIO pin
- Superior SNR with programmable voltage reference ( $V_{REF}$ )
- Supports spread spectrum and programmable resistance switches for lower electromagnetic interference (EMI)
- Reduced overhead on CPU during CapSense scanning by offloading initialization and configuration process to the CapSense sequencer

## 1.4 PSoC 4 and PRoC BLE CapSense Plus Features

You can create PSoC 4 or PRoC BLE [CapSense Plus applications](#) that feature capacitive touch sensing and additional system functionality. The main features of these devices, in addition to CapSense are:

- ARM® Cortex™-M0 CPU with single cycle multiply delivering up to 43 DMIPS at 48 MHz
- 1.71-V – 5.5-V operation over –40 to 85 °C ambient
- Up to 128 KB of flash (Cortex-M0 has >2X code density over 8-bit solutions)
- Up to 16 KB of SRAM
- Up to 94 programmable GPIOs
- Independent center-aligned PWMs with complementary dead-band programmable outputs, synchronized ADC operation (ability to trigger the ADC at a customer-specifiable time in the PWM cycle), and synchronous refresh (ability to synchronize PWM duty cycle changes across all PWMs to avoid anomalous waveforms)
- Comparator-based triggering of PWM Kill signals (to terminate motor-driving when an over-current condition is detected)
- 12-bit 1 Msps ADC including sample-and-hold (S&H) capability with zero-overhead sequencing allowing the entire ADC bandwidth to be used for signal conversion and none used for sequencer overhead
- Opamps with comparator mode and SAR input buffering capability
- Segment LCD direct drive that supports up to four commons
- SPI / UART / I<sup>2</sup>C serial communication channels
- Bluetooth Low Energy (BLE) communication compliant with version 4.0 and multiple features of version 4.1

- Programmable logic blocks, each having eight macrocells and a cascadable data path, called universal digital blocks (UDBs) for efficient implementation of programmable peripherals (such as I<sup>2</sup>S)
- Controller area network (CAN)
- Fully-supported PSoC Creator design entry, development, and debug environment providing:
  - Design entry and build (comprehending analog routing)
  - Components for all fixed-function peripherals and common programmable peripherals
  - Documentation and training modules

Support for porting builds to MDK ARM environment (previously known as RealView) and others

Refer to [AN64846, Getting Started with CapSense Design Guide](#) to select an appropriate CapSense device based on your requirements.



## 1.5 CapSense Design Flow

Figure 1-1 shows the typical flow of a product design cycle with capacitive sensing; the information in this guide is highlighted in green. Table 1-1 on page 10 provides links to the supporting documents for each of the numbered tasks in Figure 1-1.

Figure 1-1. CapSense Design Flow

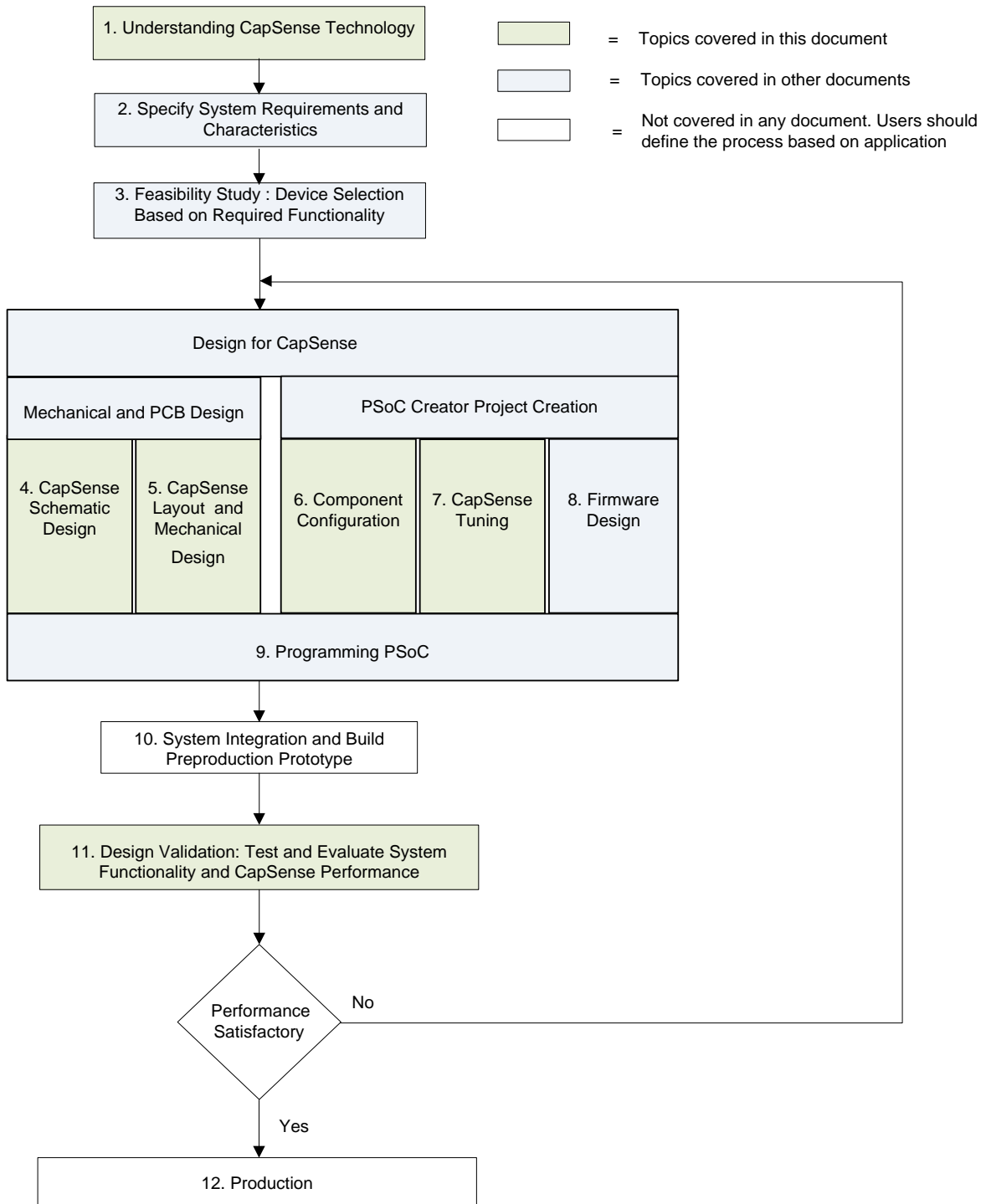


Table 1-1. Supporting Documentation

Steps in Flowchart	Supporting Cypress Documentation	
	Name	Chapter
1. Understanding CapSense	PSoC 4 CapSense Design Guide (This document) <a href="#">Getting Started with CapSense Design Guide</a>	<a href="#">Chapter 2</a> and <a href="#">Chapter 3</a> –
2. Specify requirements	<a href="#">Getting Started with CapSense Design Guide</a>	–
3. Feasibility study	PSoC 4 Datasheet PSoC 4 BLE Datasheet PRoC BLE Datasheet	–
	<a href="#">AN64846</a> , <a href="#">Getting Started with CapSense Design Guide</a> <a href="#">AN79953</a> , <a href="#">Getting Started with PSoC 4</a> <a href="#">AN91267</a> , <a href="#">Getting Started with PSoC 4 BLE</a> <a href="#">AN94020</a> , <a href="#">Getting Started with PRoC BLE</a>	–
4. Schematic design	PSoC 4 CapSense Design Guide (This document)	<a href="#">Chapter 6</a>
5. Layout design	PSoC 4 CapSense Design Guide (This document)	<a href="#">Chapter 6</a>
6. Component configuration	CapSense <a href="#">Component Datasheet</a>	–
	PSoC 4 CapSense Design Guide (This document)	<a href="#">Chapter 5</a>
7. Performance tuning	PSoC 4 CapSense Design Guide (This document)	<a href="#">Chapter 5</a>
8. Firmware design	PSoC Creator CapSense <a href="#">Component Datasheet</a>	–
	PSoC Creator <a href="#">Example Projects</a>	–
9. Programming PSoC	<a href="#">PSoC Creator User Guide</a> for in-IDE programming <a href="#">PSoC Programmer home page</a> and <a href="#">MiniProg3 User Guide</a> for Standalone programming	–
10. Prototype	–	–
11. Design validation	PSoC 4 CapSense Design Guide (This document)	<a href="#">Chapter 5</a>
12. Production	–	–

## 2. CapSense Technology

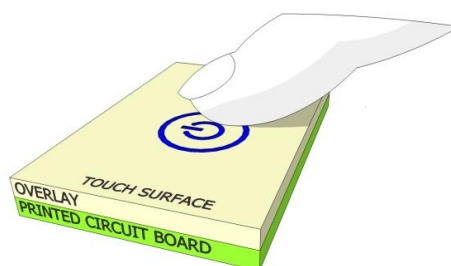


Capacitive touch sensing technology measures changes in capacitance between a plate (the sensor) and its environment to detect the presence of a finger on or near a touch surface.

### 2.1 CapSense Fundamentals

A typical CapSense sensor consists of a copper pad of proper shape and size etched on the surface of a PCB. A nonconductive overlay serves as the touch surface for the button, as [Figure 2-1](#) shows.

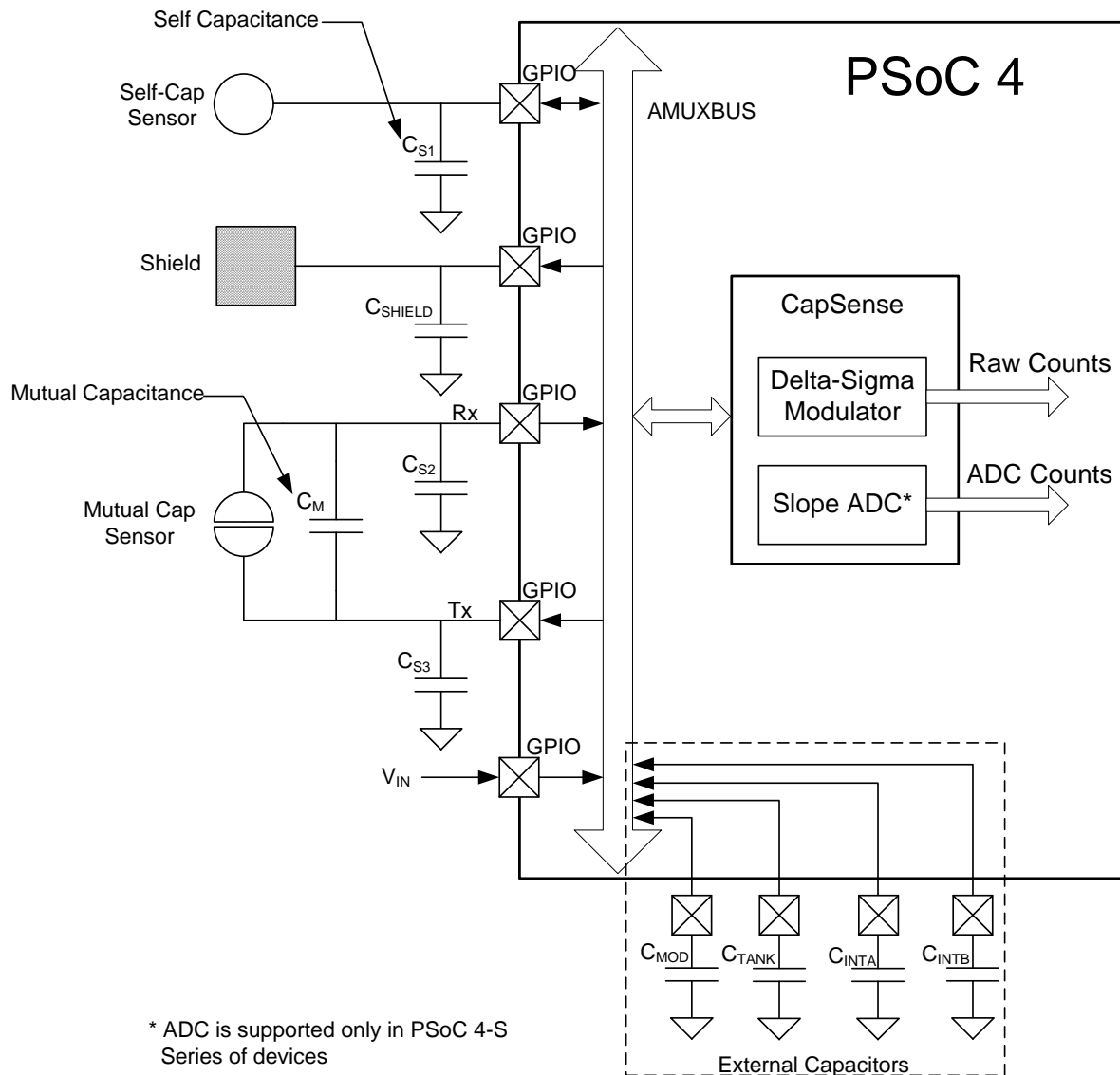
Figure 2-1. Capacitive Touch Sensor



PCB traces and vias connect the sensor pads to PSoC GPIOs that are configured as CapSense sensor pins. As [Figure 2-2](#) shows, the self-capacitance of each electrode is modeled as  $C_{SX}$  and the mutual capacitance between electrodes is modeled as  $C_{MXY}$ . CapSense circuitry internal to the PSoC converts these capacitance values into equivalent digital counts (see [Chapter 3](#) for details). These digital counts are then processed by the CPU to detect touches.

CapSense also requires external capacitor  $C_{MOD}$  for self-capacitance sensing and  $C_{INTA}$  and  $C_{INTB}$  capacitors for mutual-capacitance sensing. These external capacitors are connected between a dedicated GPIO pin and ground. If shield electrode is implemented for liquid tolerance, or for large proximity sensing distance, an additional  $C_{TANK}$  capacitor may be required. The recommended values of the external capacitors are listed in [Table 6-6](#).

Figure 2-2. PSoC Device, Sensors, and External Capacitors

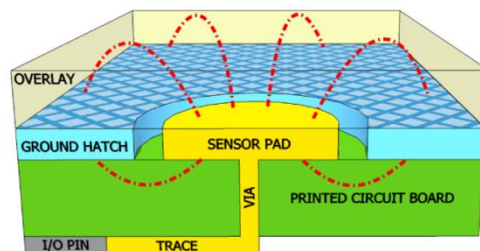


The capacitance of the sensor in the absence of a touch is called the parasitic capacitance,  $C_P$ . Parasitic capacitance results from the electric field between the sensor (including the sensor pad, traces, and vias) and other conductors in the system such as the ground planes, traces, and any metal in the product's chassis or enclosure. The GPIO and internal capacitances of PSoC also contribute to the parasitic capacitance. However, these internal capacitances are typically very small compared to the sensor capacitance.

## 2.1.1 Self-Capacitance Sensing

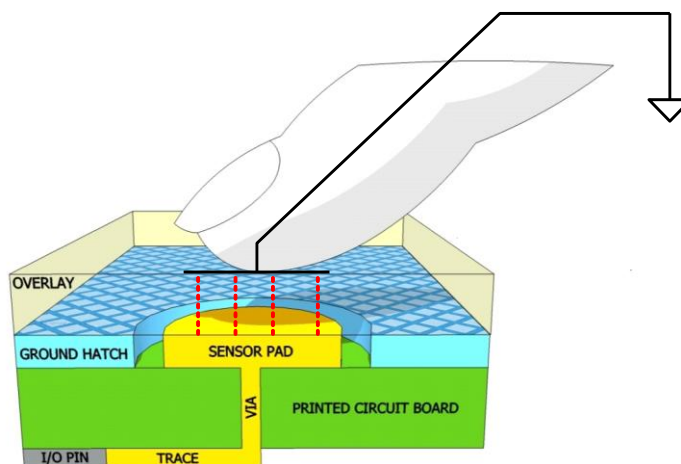
Figure 2-3 shows how a GPIO pin is connected to a sensor pad by traces and vias for self-capacitance sensing. Typically, a ground hatch surrounds the sensor pad to isolate it from other sensors and traces. Although Figure 2-3 shows some field lines around the sensor pad, the actual electric field distribution is very complex.

Figure 2-3. Parasitic Capacitance



When a finger is present on the overlay, the conductive nature and large mass of the human body forms a grounded, conductive plane parallel to the sensor pad, as Figure 2-4 shows.

Figure 2-4. Finger Capacitance



This arrangement forms a parallel plate capacitor. The capacitance between the sensor pad and the finger is:

Equation 2-1: Finger Capacitance

$$C_F = \frac{\epsilon_0 \epsilon_r A}{d}$$

Where:

$\epsilon_0$  = Free space permittivity

$\epsilon_r$  = Relative permittivity of overlay

A = Area of finger and sensor pad overlap

d = Thickness of the overlay

$C_F$  is known as the finger capacitance. The parasitic capacitance  $C_P$  and finger capacitance  $C_F$  are parallel to each other because both represent the capacitance between the sensor pin and ground. Therefore, the total capacitance  $C_S$  of the sensor, when the finger is present on the sensor, is the sum of  $C_P$  and  $C_F$ .

Equation 2-2: Total Sense Capacitance when finger is present on sensor

$$C_S = C_P + C_F$$

In the absence of touch,  $C_S$  is equal to  $C_P$ .

PSoC converts the capacitance  $C_S$  into equivalent digital counts called raw counts. Because a finger touch increases the total capacitance of the sensor pin, an increase in the raw counts indicates a finger touch.

PSoC 4 CapSense supports parasitic capacitance values as high as 65 pF for 0.3-pF finger capacitance, and as high as 35 pF for 0.1-pF finger capacitance.

## 2.1.2 Mutual-Capacitance Sensing

Figure 2-5 shows the button sensor layout for mutual-capacitance sensing. Mutual-capacitance sensing measures the capacitance between two electrodes, which are called transmit (Tx) and receive (Rx) electrodes.

In a mutual-capacitance sensing system, a digital voltage signal switching between VDDIO<sup>a</sup> or VDDD<sup>b</sup> (if VDDIO is not supported by the device) and GND is applied to the Tx pin and the amount of charge received on the Rx pin is measured. The amount of charge received on the Rx electrode is directly proportional to the mutual capacitance ( $C_M$ ) between the two electrodes.

When a finger is placed between the Tx and Rx electrodes, the mutual-capacitance decreases to  $C_M^1$ , as shown in Figure 2-6. Because of the reduction in the mutual-capacitance, the charge received on the Rx electrode also decreases. The CapSense system measures the amount of charge received on the Rx electrode to detect a touch/no touch condition.

Figure 2-5. Mutual-Capacitance Sensing Working

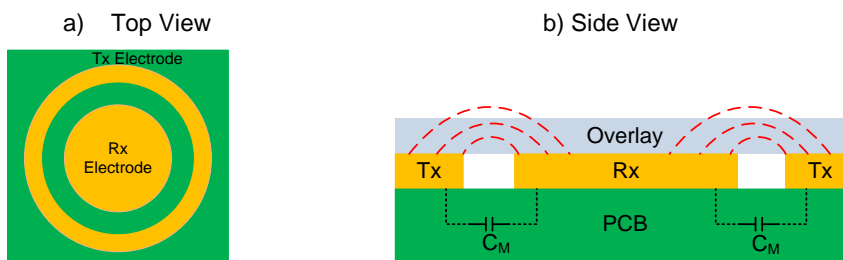
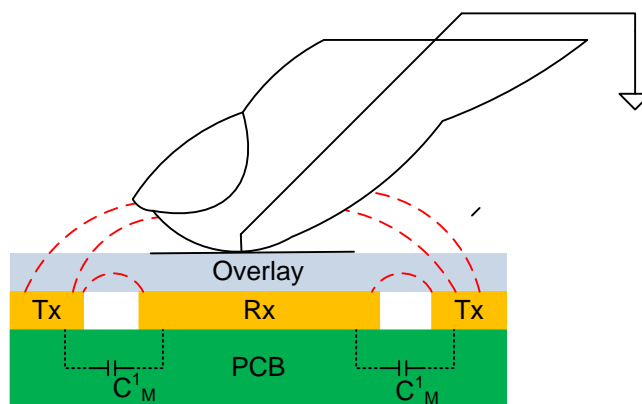


Figure 2-6. Mutual-Capacitance with Finger Touch



<sup>a</sup> VDDIO is the power supply for I/O pin

<sup>b</sup> VDDD is the device power supply for digital section

## 2.2 Capacitive Touch Sensing Method

PSoC 4 uses Cypress's patented capacitive touch sensing methods known as CapSense Sigma Delta (CSD) for self-capacitance sensing and CapSense Crosspoint (CSX) for mutual-capacitance scanning. The CSD and CSX touch sensing methods provide the industry's best-in-class [Signal-to-Noise Ratio](#). These sensing methods are a combination of hardware and firmware techniques.

### 2.2.1 CapSense Sigma Delta (CSD)

Figure 2-7 shows a simplified block diagram of the CSD method.

In CSD, each GPIO has a switched-capacitance circuit that converts the sensor capacitance into an equivalent current. An analog multiplexer then selects one of the currents and feeds it into the current to digital converter. The current to digital converter is similar to a sigma delta ADC. The output count of the current to digital converter, known as **raw count**, is a digital value that is proportional to the self-capacitance or mutual-capacitance between the electrodes.

Equation 2-3: Raw Count and Sensor Capacitance Relationship in CSD

$$\text{raw count} = G_C C_S$$

Where  $G_C$  is the capacitance to digital conversion gain of CSD, and

$C_S$  is the self-capacitance of the electrode

Figure 2-7. Simplified Diagram of CapSense Sigma Delta Method

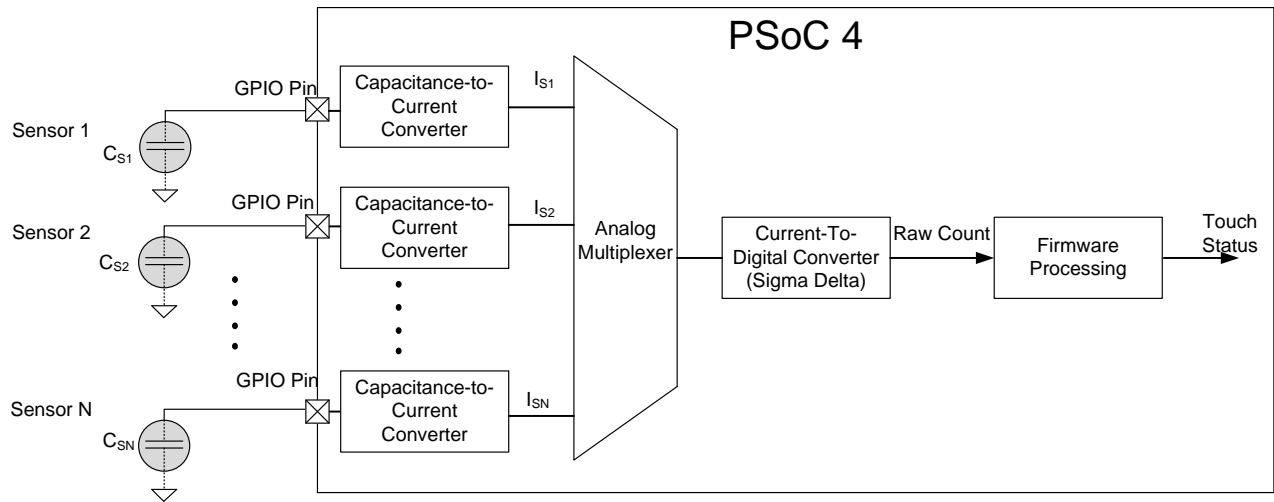
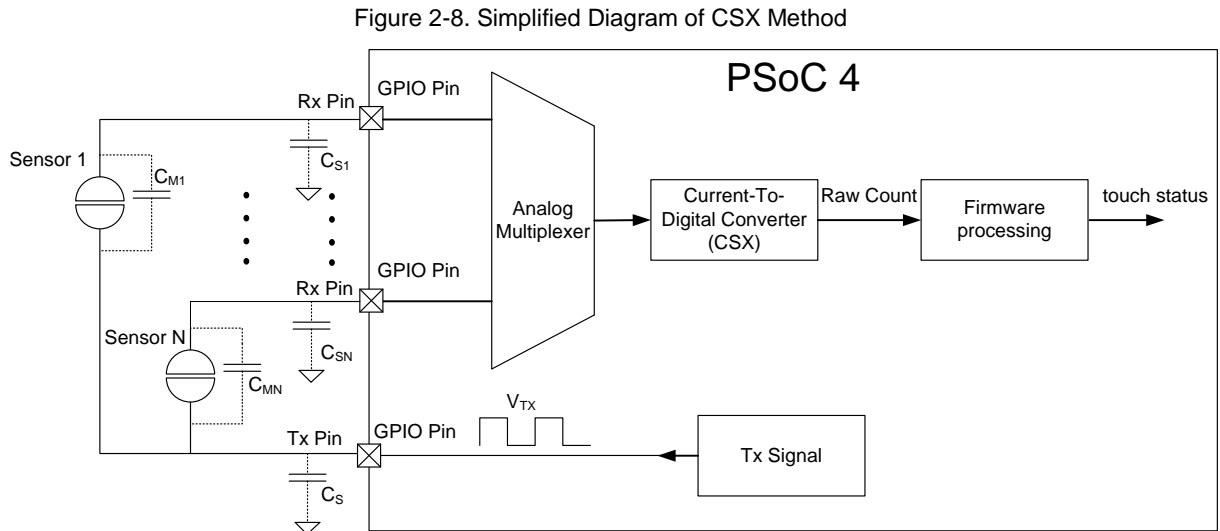


Figure 2-9 shows a plot of raw count over time. When a finger touches the sensor, the  $C_S$  increases from  $C_P$  to  $C_P + C_F$ , and the raw count increases. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

## 2.2.2 CapSense Crosspoint (CSX)

Figure 2-8 shows the simplified block diagram of the CSX method.



With CSX, a voltage on the Tx pin (or Tx electrode) couples charge on to the RX pin. This charge is proportional to the mutual capacitance between the Tx and Rx electrodes. An analog multiplexer then selects one of the Rx channel and feeds it into the current to digital converter.

The output count of the current to digital converter, known as **raw count**, is a digital value that is proportional to the self-capacitance or mutual-capacitance between the electrodes.

Equation 2-4: Raw Count and Sensor Capacitance Relationship in CSX

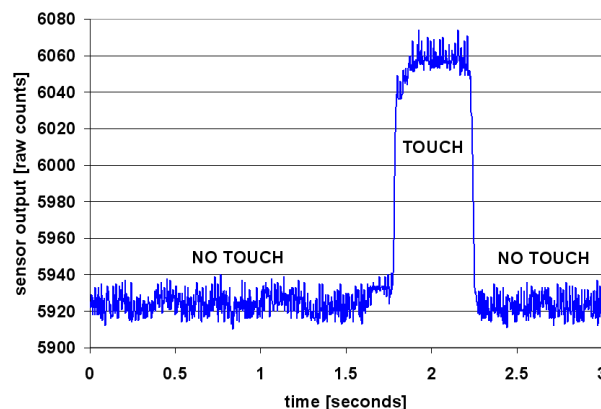
$$\text{raw count} = G_{CM} C_M$$

Where  $G_{CM}$  is the capacitance to digital conversion gain of Mutual Capacitance method, and

$C_M$  is the mutual-capacitance between two electrodes.

Figure 2-9 shows a plot of raw count over time. When a finger touches the sensor,  $C_M$  decreases from  $C_M$  to  $C_M$ , hence the counter output decreases. The firmware normalizes the raw count such that the raw counts go high when  $C_M$  decreases. This is to maintain the same visual representation of raw count between CSD and CSX methods. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

Figure 2-9. Raw Count versus Time



For an in-depth discussion of the PSoC 4 CapSense CSD and CSX blocks, see [PSoC 4 CapSense](#).

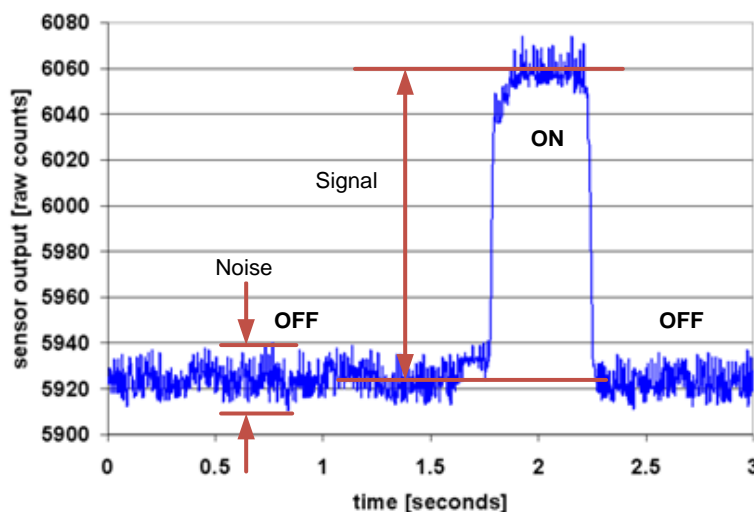


## 2.3 Signal-to-Noise Ratio

In practice, the raw counts vary due to inherent noise in the system. CapSense noise is the peak-to-peak variation in raw counts in the absence of a touch, as [Figure 2-10](#) shows.

A well-tuned CapSense system reliably discriminates between the ON and OFF states of the sensors. To achieve good performance, the CapSense signal must be significantly larger than the CapSense noise. Signal-to-noise Ratio (SNR), which is defined as the ratio of CapSense signal to CapSense noise is the most important performance parameter of a CapSense sensor.

Figure 2-10. SNR



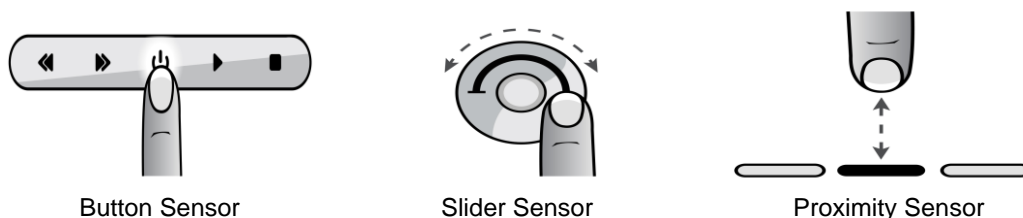
In this example, the average level of raw count in the absence of a touch is 5925 counts. When a finger is placed on the sensor, the average raw count increases to 6060 counts, which means the signal is  $6060 - 5925 = 135$  counts. The minimum value of the raw count in the OFF state is 5912 and the maximum value is 5938 counts. Therefore, the CapSense noise is  $5938 - 5912 = 26$  counts. This results in an SNR of  $135 / 26 = 5.2$ .

The minimum SNR recommended for a CapSense sensor is 5. This 5:1 ratio comes from best practice threshold settings, which enable enough margin between signal and noise in order to provide reliable ON/OFF operation.

## 2.4 CapSense Widgets

CapSense widgets consist of one or more CapSense sensors, which as a unit represent a certain type of user interface. CapSense widgets are broadly classified into four categories – Buttons (Zero-Dimensional), Sliders (One-Dimensional), Touchpads/Trackpads (Two-Dimensional), and Proximity sensors (Three-Dimensional). [Figure 2-11](#) shows button, slider, and proximity sensor widgets. This section explains the basic concepts of different CapSense widgets. For a detailed explanation of sensor construction, see [Sensor Construction](#).

Figure 2-11. Several Types of Widgets

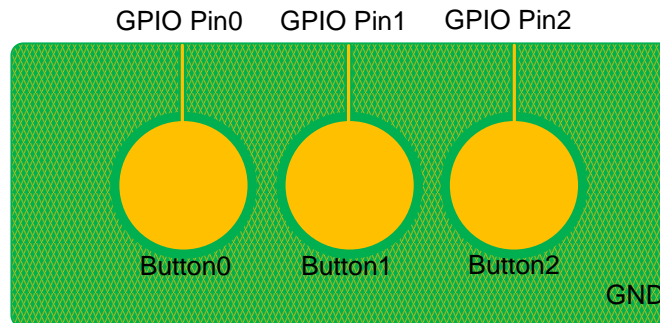


## 2.4.1 Buttons (Zero-Dimensional)

CapSense buttons replace mechanical buttons in a wide variety of applications such as home appliances, medical devices, white goods, lighting controls, and many other products. It is the simplest type of CapSense widget, consisting of a single sensor. A CapSense button gives one of two possible output states: active (finger is present) or inactive (finger is not present). These two states are also called ON and OFF states, respectively.

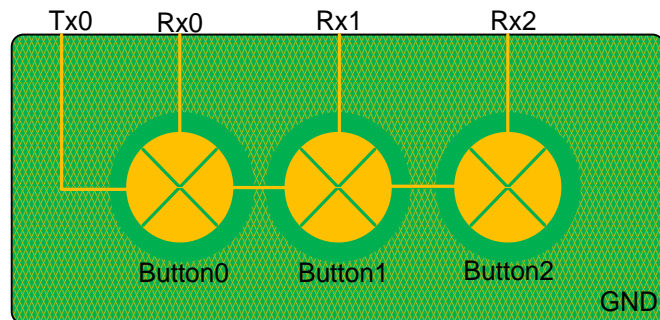
For the self-capacitance based i.e. CSD sensing method, a simple CapSense button consists of a circular copper pad connected to a PSoC GPIO with a PCB trace. The button is surrounded by grounded copper hatch to isolate it from other buttons and traces. A circular gap separates the button pad and the ground hatch. Each button requires one PSoC GPIO.

Figure 2-12. Simple CapSense Buttons



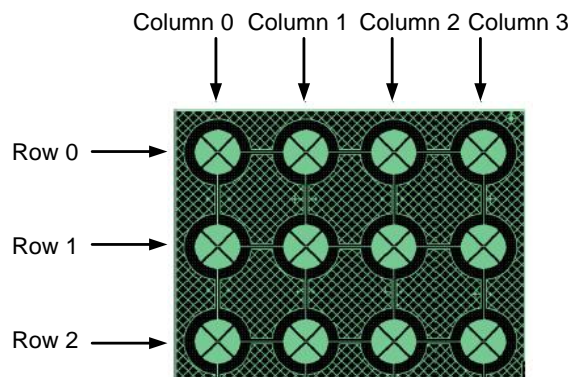
For the mutual-capacitance based i.e. CSX sensing method, each button requires one GPIO pin configured as Tx electrode and one GPIO pin configured as Rx electrode. The Tx pin can be shared across multiple buttons, as shown in [Figure 2-13](#).

Figure 2-13. Simple CapSense Buttons for Mutual-Capacitance Sensing Method



If the application requires a large number of buttons, such as in a calculator keypad or a QWERTY keyboard, you can arrange the CapSense buttons in a matrix, as [Figure 2-14](#) shows. This allows a design to have multiple buttons per GPIO. For example, the 12-button design in [Figure 2-14](#) requires only seven GPIOs.

Figure 2-14. Matrix Buttons



A matrix button design has two groups of capacitive sensors: row sensors and column sensors. The matrix button architecture can be used for both self-capacitance and mutual-capacitance methods. Mutual-capacitance method has advantages in that it can detect multiple fingers at the same time.

In self-capacitance mode, each button consists of a row sensor and a column sensor, as [Figure 2-14](#) shows. When a button is touched, both row and column sensors of that button become active. The number of buttons supported by the matrix is equal to the product of the number of rows and the number of columns. In the self-capacitance mode, matrix buttons can only be sensed one at a time. If more than one row or column sensor is in the active state, the finger location cannot be resolved, which is considered an invalid condition. Some applications require simultaneous sensing of multiple buttons, such as a keyboard with Shift, Ctrl, and Alt keys. In this case, you can use mutual-capacitance sensing method or you should design the Shift, Ctrl, and Alt keys as individual buttons.

## 2.4.2 Sliders (One-Dimensional)

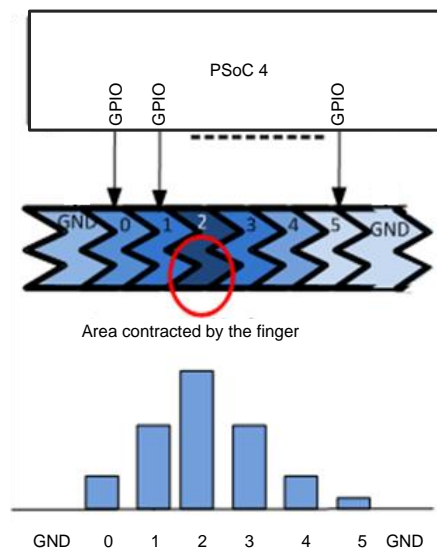
Sliders are used when the required input is in the form of a gradual increment or decrement. Examples include lighting control (dimmer), volume control, graphic equalizer, and speed control. Currently, the CapSense Component in PSoC Creator supports only self-capacitance-based sliders. Mutual capacitance based sliders will be supported in future version of component.

A slider consists of a one-dimensional array of capacitive sensors called segments, which are placed adjacent to one another. Touching one segment also results in partial activation of adjacent segments. The firmware processes the raw counts from the touched segment and the nearby segments to calculate the position of the geometric center of the finger touch, which is known as the **centroid position**.

The actual resolution of the calculated centroid position is much higher than the number of segments in a slider. For example, a slider with five segments can resolve at least 100 physical finger positions. This high resolution gives smooth transitions of the centroid position as the finger glides across a slider.

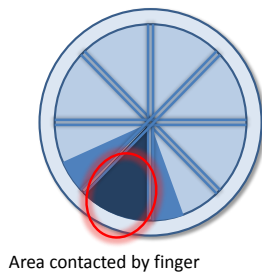
In a linear slider, the segments are arranged inline, as [Figure 2-15](#) shows. Each slider segment connects to a PSoC GPIO. A zigzag pattern (double chevron) is recommended for slider segments. This layout ensures that when a segment is touched, the adjacent segments are also partially touched, which aids estimation of the centroid position.

Figure 2-15. Linear Slider



Radial sliders are similar to linear sliders except that radial sliders are continuous. [Figure 2-16](#) shows a typical radial slider.

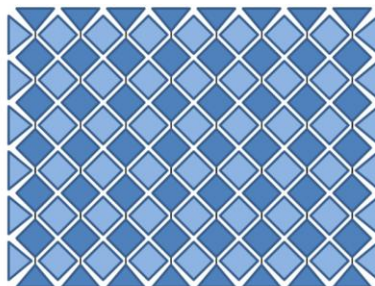
Figure 2-16. Radial Slider



### 2.4.3 Touchpads / Trackpads (Two-Dimensional)

A trackpad (also known as touchpad) has two linear sliders arranged in an X and Y pattern, enabling it to locate a finger's position in both X and Y dimensions. [Figure 2-17](#) shows a typical arrangement of a trackpad sensor. Currently, the CapSense Component in PSoC Creator supports only self-capacitance-based touchpads. Mutual capacitance based touchpads will be supported in future version of Component.

Figure 2-17. Trackpad Sensor Arrangement

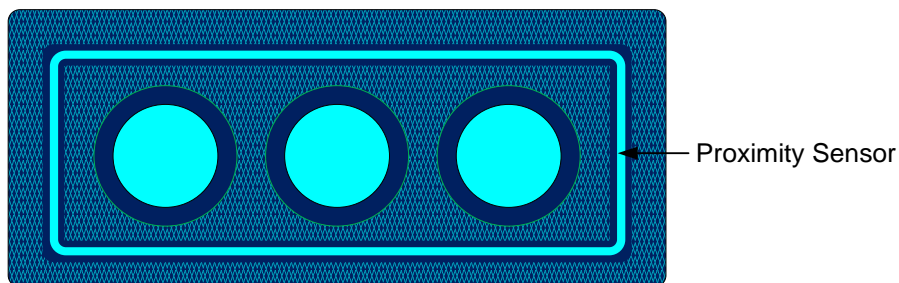


### 2.4.4 Proximity (Three-Dimensional)

Proximity sensors detect the presence of a hand in the three-dimensional space around the sensor. However, the actual output of the proximity sensor is an ON/OFF state similar to a CapSense button. Proximity sensing can detect a hand at a distance of several centimeters to tens of centimeters depending on the sensor construction. Currently, the CapSense Component in PSoC Creator supports only self-capacitance-based proximity sensors. Mutual capacitance based proximity sensor will be supported in future version of Component.

Proximity sensing requires electric fields that are projected to much larger distances than buttons and sliders. This demands a large sensor area. However, a large sensor area also results in a large parasitic capacitance  $C_P$ , and detection becomes more difficult. This requires a sensor with high electric field strength at large distances while also having a small area. Use a trace with a thickness of 2-3 mm surrounding the other sensors, as [Figure 2-18](#) shows.

Figure 2-18. Proximity Sensor



You can also implement a proximity sensor by ganging other sensors together. This is accomplished by combining multiple sensor pads into one large sensor using firmware. The disadvantage of this method is high parasitic capacitance. See the [CapSense Component Datasheet](#) for details.

Refer to [AN92239 Proximity Sensing with CapSense](#) and the proximity sensing section in [Getting Started with CapSense Design Guide](#) to learn more about proximity sensors.

## 2.5 Liquid Tolerance

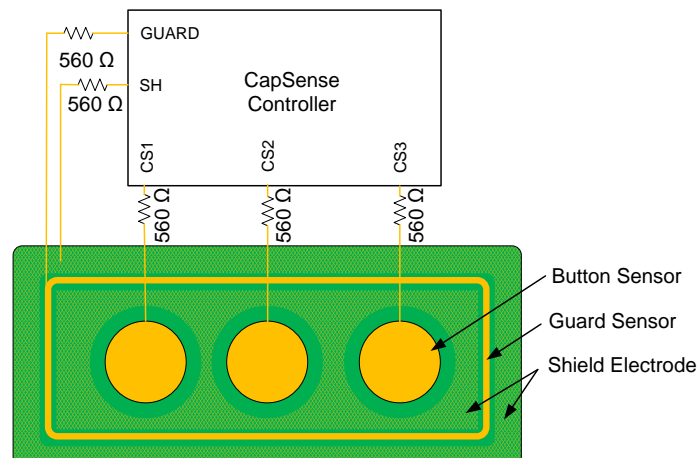
Capacitive sensing is used in a variety of applications such as home appliances, automotive, and industrial applications. These applications require robust capacitive-sensing operation even in the presence of mist, moisture, water, ice, and humidity changes. In a capacitive-sensing application design, false sensing of touch or proximity detection may happen due to the presence of a film of liquid or liquid droplets on the sensor surface, due to the conductive nature of some liquids. Cypress's CSD sensing method can compensate for variation in raw count due to these causes and provide a robust, reliable, capacitive sensing application operation.

Figure 2-19 Liquid-Tolerant CapSense-Based Touch User Interface in a Washing Machine



To compensate for changes in raw count due to mist, moisture, and humidity changes, the CapSense sensing method continuously adjusts the baseline of the sensor to prevent false triggers. To compensate for changes in raw count due to a liquid droplet or liquid flow, you should implement a [Shield Electrode](#) and [Guard Sensor](#) to provide robust touch sensing, as [Figure 2-20](#) shows. CapSense reliably works and reports the sensor ON/OFF status when a shield electrode is implemented and liquid droplets are present on the sensor surface. When there is a liquid flow, [Guard Sensor](#) will detect the presence of a streaming liquid and the sensors will not be scanned. Therefore, the sensor ON/OFF status will not be reported.

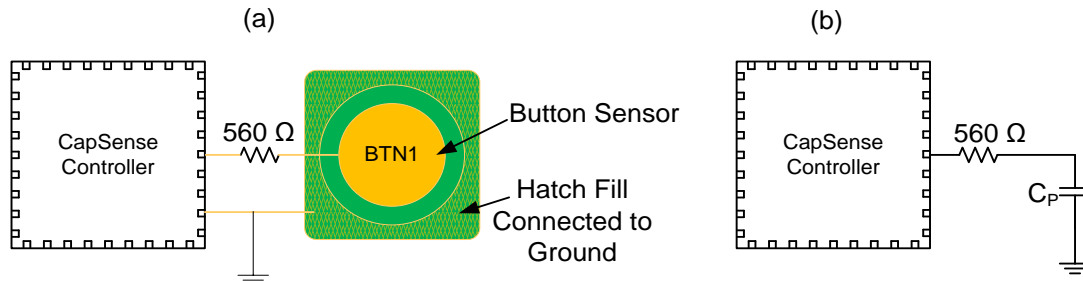
Figure 2-20. Shield Electrode (SH) and Guard Sensor (GUARD) Connected to CapSense Controller



## 2.5.1 Effect of Liquid Droplets and Liquid Stream on a CapSense Sensor

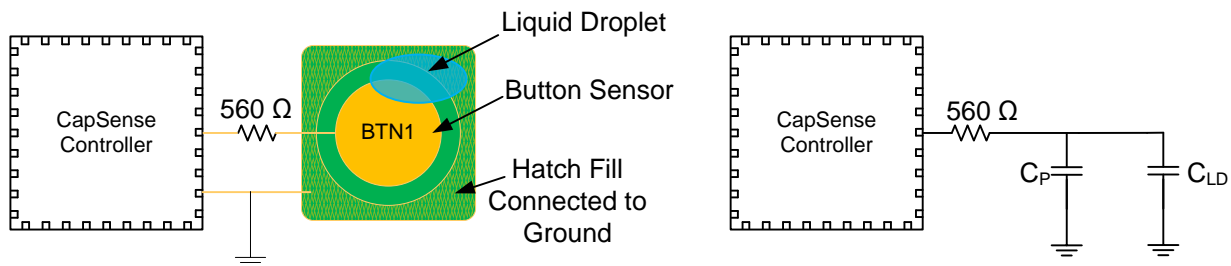
To understand the effect of liquids on a CapSense sensor, consider a CapSense system in which the hatch fill around the sensor is connected to ground, as [Figure 2-21\(a\)](#) shows. The hatch fill when connected to a ground improves the noise immunity of the sensor. Parasitic capacitance of the sensor is denoted as  $C_P$  in [Figure 2-21 \(b\)](#).

Figure 2-21. Typical CapSense System Layout



As shown in [Figure 2-22](#), when a liquid droplet falls on the sensor surface, due to its conductive nature it provides a strong coupling path for the electric field lines to return to ground; this adds a capacitance  $C_{LD}$  in parallel to  $C_P$ . This added capacitance draws an additional charge from the AMUX bus as explained in [GPIO Cell Capacitance to Current Converter](#), resulting in an increase in the sensor raw count. In some cases (such as salty water or water containing minerals), the increase in raw count when a liquid droplet falls on the sensor surface may be equal to the increase in raw count due to a finger touch, as [Figure 2-23](#) shows. In such a situation, sensor false triggers might occur.

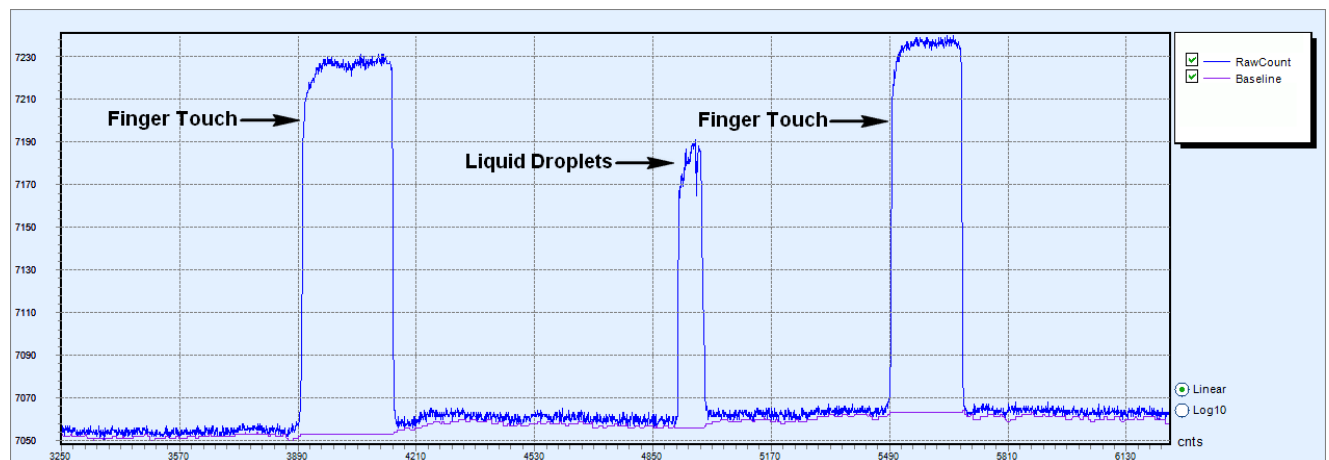
Figure 2-22. Capacitance Added by Liquid Droplet when the Hatch Fill is Connected to Ground



$C_P$  – Sensor parasitic capacitance

$C_{LD}$  – Capacitance added by the liquid droplet

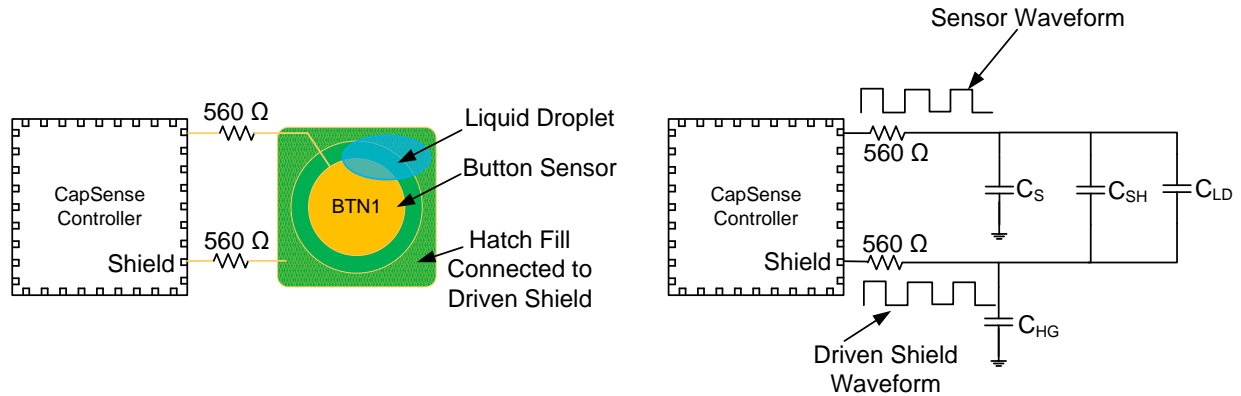
Figure 2-23. Effect of Liquid Droplet when the Hatch Fill around the Sensor is Connected to Ground



To nullify the effect of capacitance added by the liquid droplet to the CapSense circuitry, you should drive the hatch fill around the sensor with the [driven-shield](#) signal.

As Figure 2-24 shows, when the hatch fill around the sensor is connected to the driven-shield signal and when a liquid droplet falls on the touch interface, the voltage on both sides of the liquid droplet remains at the same potential. Because of this, the capacitance,  $C_{LD}$ , added by the liquid droplet does not draw any additional charge from the AMUX bus and hence the effect of capacitance  $C_{LD}$  is nullified. Therefore, the increase in raw count when a water droplet falls on the sensor will be very small, as Figure 2-25 shows.

Figure 2-24. Capacitance Added by Liquid Droplet when the Hatch Fill around the Sensor Is Connected to Shield



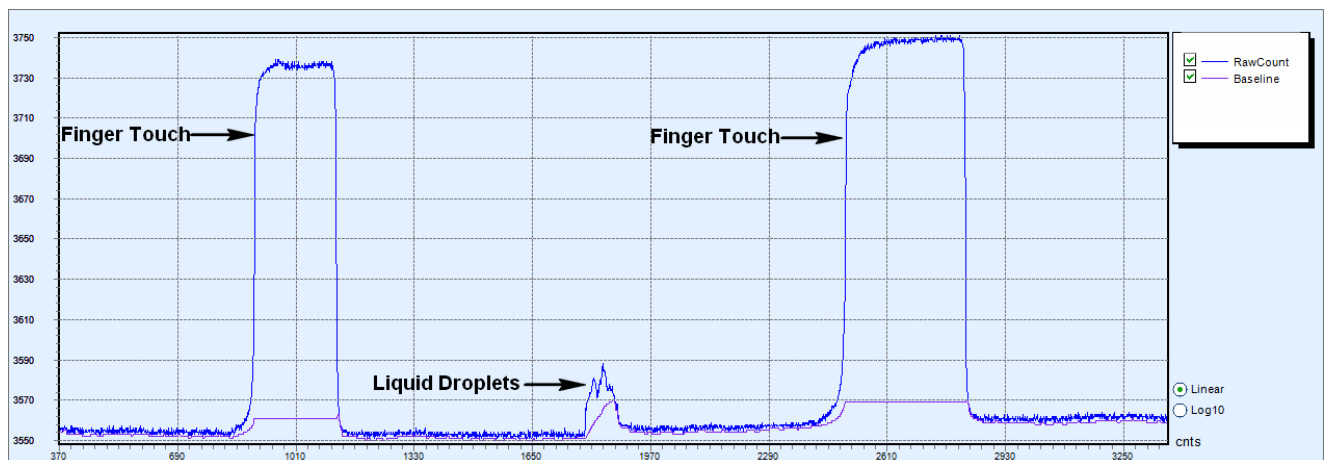
$C_S$  – Sensor parasitic capacitance

$C_{SH}$  – Capacitance between the sensor and the hatch fill

$C_{HG}$  – Capacitance between the hatch fill and ground

$C_{LD}$  – Capacitance added by the liquid droplet

Figure 2-25. Effect of Liquid Droplet when the Hatch Fill around the Sensor is Connected to the Driven-Shield

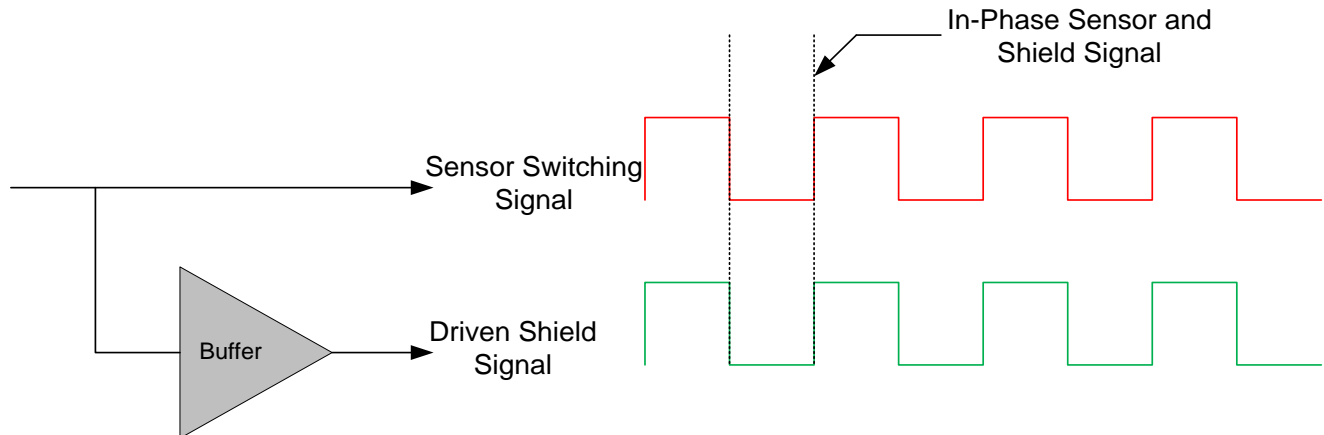




## 2.5.2 Driven-Shield Signal and Shield Electrode

The driven-shield signal is a buffered version of the sensor-switching signal, as [Figure 2-26](#) shows. The driven-shield signal has the same amplitude, frequency, and phase as that of sensor switching signal. The buffer provides sufficient current for the driven-shield signal to drive the high parasitic capacitance of the hatch fill. When the hatch fill around the sensor is connected to the driven shield signal, it is referred as shield electrode.

Figure 2-26. Driven Shield Signal



Shield electrode is used for the following purposes:

- To implement liquid-tolerant CapSense designs: Shield electrode helps in making CapSense designs liquid-tolerant as explained above.
- To improve the proximity sensing distance in the presence of floating or grounded conductive objects: A shield electrode, when placed between the proximity sensor and a floating or a grounded conductive object, reduces the effect of these objects on the proximity-sensing distance and helps in achieving large proximity-sensing distance. See the “Proximity Sensing” section in the [Getting Started with CapSense Design Guide](#) for more details.
- To reduce the parasitic capacitance of the sensor: When a CapSense sensor has a long trace, the  $C_P$  of the sensor will be very high because of the increased coupling of sensor electric field lines from the sensor trace to the surrounding ground. By implementing a shield electrode, the coupling of electric field lines to ground is reduced, which results in reducing the  $C_P$  of the sensor.

See [Layout Guidelines for Liquid Tolerance](#) for layout guidelines of shield electrode.

## 2.5.3 Guard Sensor

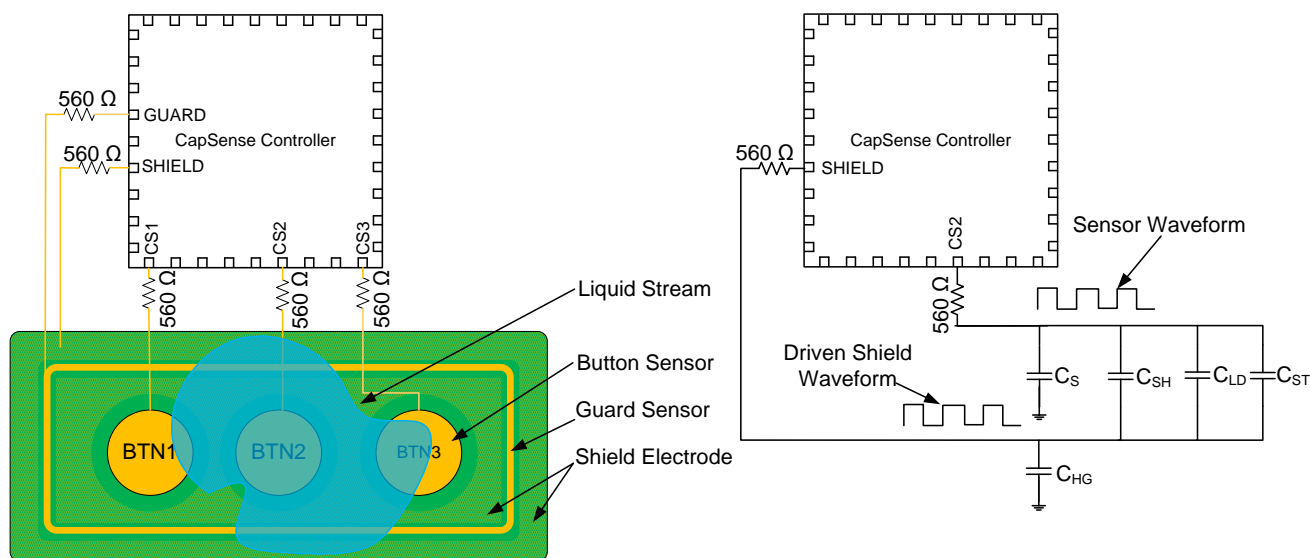
When a continuous liquid stream is present on the sensor surface, the liquid stream adds a large capacitance ( $C_{ST}$ ) to the CapSense sensor. This capacitance may be several times larger than  $C_{LD}$ . Because of this, the effect of the shield electrode is completely masked and the sensor raw counts will be same as or even higher than a finger touch. In such situations, a guard sensor is useful to prevent sensor false triggers.

A guard sensor is a copper trace that surrounds all the sensors on the PCB, as [Figure 2-27](#) shows. A guard sensor is similar to a button sensor and is used to detect the presence of streaming liquids. When a guard sensor is triggered, the firmware disables the scanning of all other sensors except the guard sensor to prevent sensor false triggers.

**Note** Because the sensors are not scanned when the guard sensor is triggered, touch cannot be detected when there is a liquid stream on the touch surface.



Figure 2-27. Capacitance Measurement with a Liquid Stream

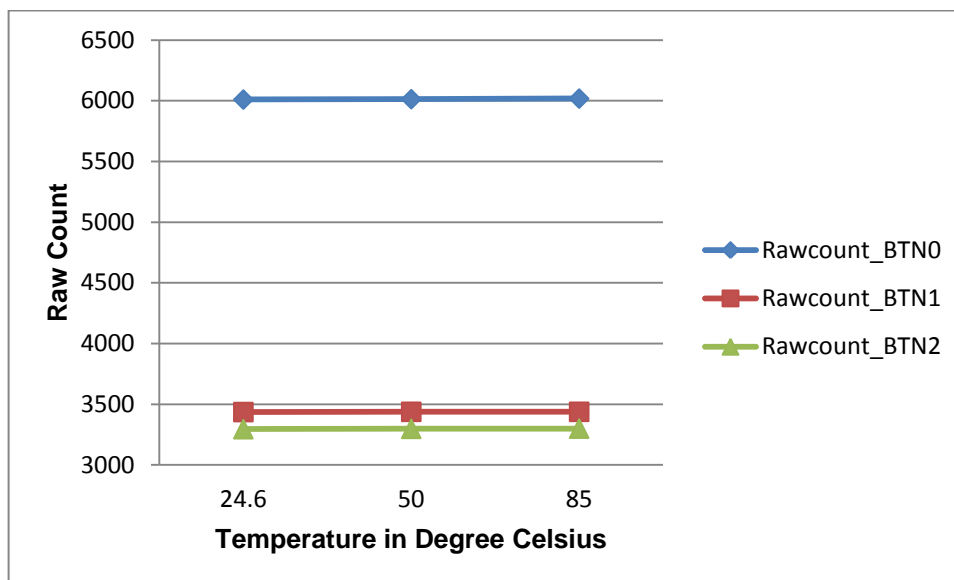


### 2.5.3.1 Effect of Liquid Properties on Liquid-Tolerance Performance

In certain applications, the CapSense system has to work in the presence of a variety of liquids such as soap water, sea water, and mineral water. In such applications, it is always recommended to tune the CapSense parameters for sensors by considering the worst-case signal due to liquid droplets. To simulate the worst-case conditions, it is recommended that you test the liquid-tolerance performance of the sensors with salty water by dissolving 40 grams of cooking salt (NaCl) in one liter of water. Tests were done using soapy water; the results show that the effect of soapy water is similar to the effect of salty water. Therefore, if the tuning is done to reject salty water, the CapSense system will work even in the presence of soapy water.

In applications such as induction cooktops, there are chances of hot water spilling on to the CapSense touch surface. To determine the impact of the temperature of a liquid droplet on CapSense performance, droplets of water at different temperatures were poured on a sensor and the corresponding change in raw counts was monitored. Experiment shows that the effect of hot liquid droplets is same as that of the liquid at room temperature as Figure 2-28 shows. This is because the hot liquid droplet cools down immediately to room temperature when it falls on the touch surface. If hot water continuously falls on the sensor and the temperature of the overlay rises because of the hot water, the increase in raw count due to the increase in temperature is compensated by the baseline algorithm, thereby preventing any false triggering of the sensors.

Figure 2-28. Raw Count Variation versus Water Temperature



To make your design liquid-tolerant, follow these steps:

1. If your application requires tolerance to liquid droplets, implement a shield electrode. If your application requires tolerance to streaming liquids along with liquid droplets, implement a shield electrode and a guard sensor. Follow the schematic and layout guidelines explained in the [Layout Guidelines for Liquid Tolerance](#) section to construct the shield electrode and guard sensor respectively.
2. In the CapSense Component, enable the driven-shield signal and specify the “Inactive sensor connection” option as “Shield”.
3. If the SmartSense algorithm is used, set the sensitivity of the guard sensor (if implemented) such that it will be triggered only when there is a liquid stream on the touch surface.

If manual tuning is used, set the resolution of the guard sensor such that it will be triggered only when there is a liquid stream on the touch surface.

## 3. PSoC 4 and PRoC BLE CapSense



This chapter explains how CapSense CSD and CSX is implemented in the PSoC 4 and PRoC BLE devices. See [Capacitive Touch Sensing Method](#) to understand the basic principles of CapSense. A basic knowledge of the PSoC 4 or PRoC BLE device architecture is a prerequisite for this chapter. If you are new to PSoC 4, refer to [AN79953, Getting Started with PSoC 4](#) or [AN91267, Getting Started with PSoC 4 BLE](#). If you are new to PRoC BLE, refer to [AN94020, Getting Started with PRoC BLE](#).

You can skip this chapter if you are using the automatic tuning feature (SmartSense) of the Component. See the [CapSense Performance Tuning](#) chapter for details.

The PSoC 4 family of devices has two different CapSense architectures. Section [3.1.1](#) explains the CapSense architecture in PSoC 4000, PSoC 4200, PSoC 4200 BLE, PRoC BLE, PSoC 4200M, and PSoC 4200L devices and Section [0](#) explains the CapSense architecture in the PSoC 4 S-Series family of devices.

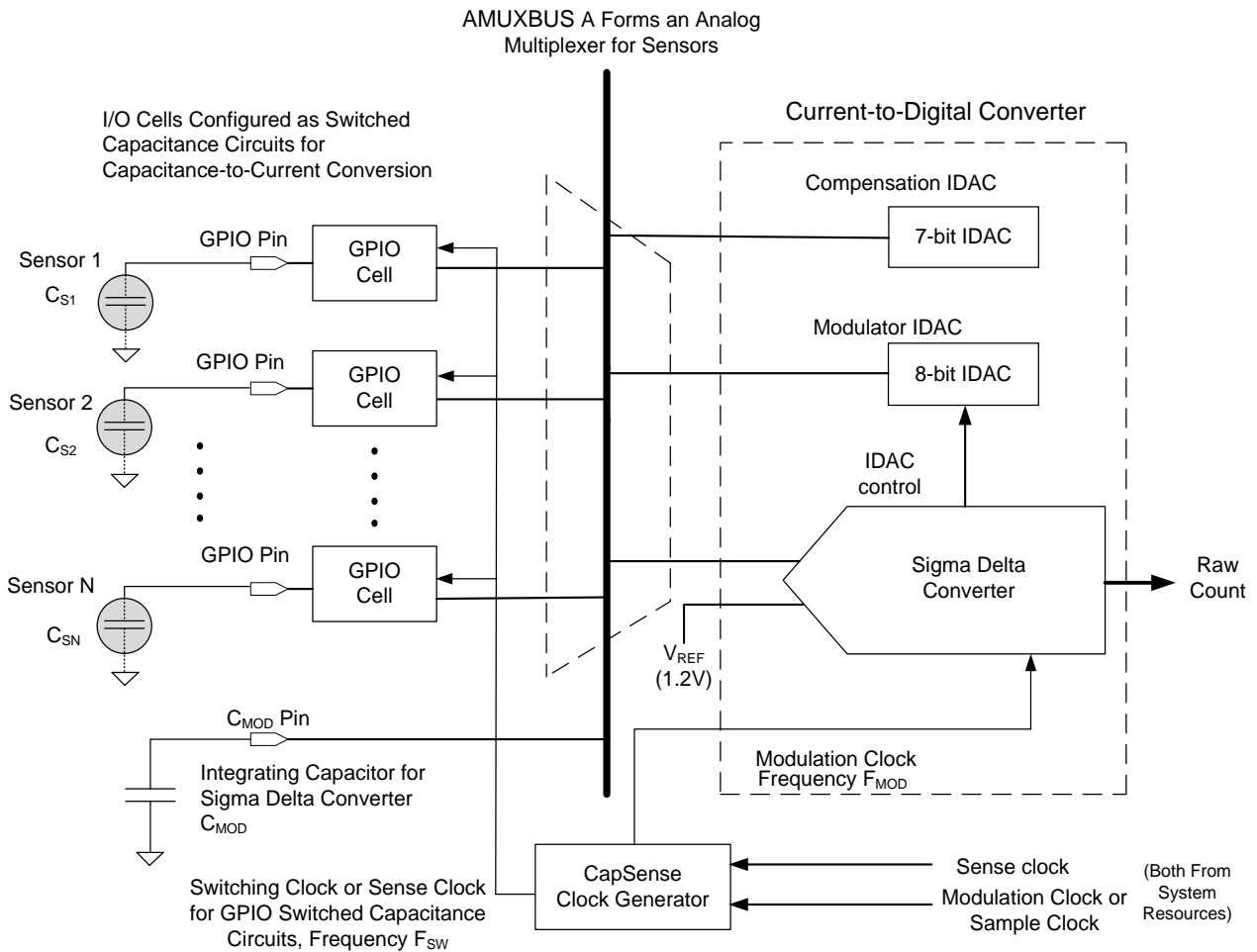
### 3.1 CapSense Sensing

#### 3.1.1 CapSense Architecture in PSoC 4 and PRoC BLE

##### 3.1.1.1 CSD Sensing Method

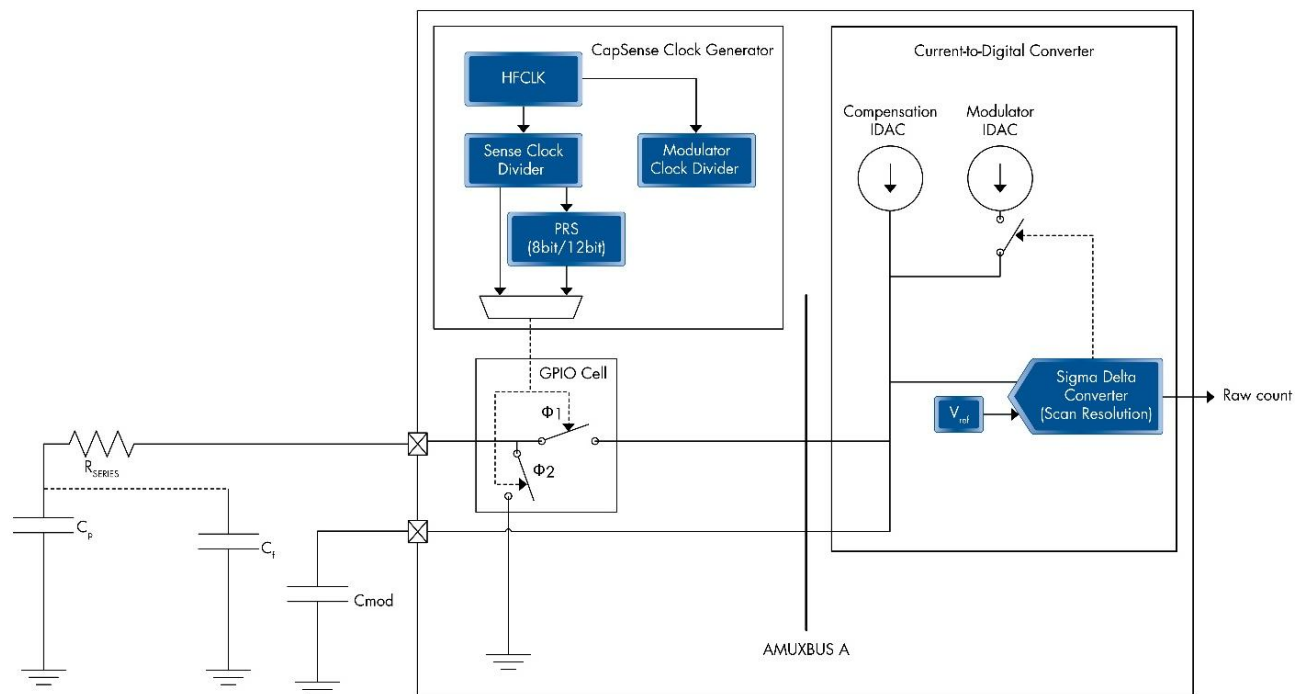
[Figure 3-1](#) illustrates the CapSense block that scans CapSense sensors in CSD sensing mode.

Figure 3-1. PSoC 4 and PSoC BLE CapSense CSD Sensing



As explained in [Capacitive Touch Sensing Method](#), this block works by first converting the sensor capacitance into an equivalent current. An analog multiplexer then selects one of the currents and feeds it into the current-to-digital converter. This current-to-digital converter consists of a sigma-delta converter, which controls the modulation IDAC such that for a specific period of time, the total current sourced or sunk by the IDACs is the same as the total current sunk or sourced by the sensor capacitance. The digital count output of the sigma-delta converter is an indicator of the sensor capacitance and is called a raw count. This block can be configured in either IDAC Sourcing mode or IDAC Sinking mode. In the IDAC Sourcing mode, the IDACs source current to AMUXBUS while the GPIO cells sink current from AMUXBUS. In the IDAC Sinking mode, the IDACs sink current from AMUXBUS while the GPIO cells source current to AMUXBUS. [Figure 3-2](#) shows the IDAC Sourcing mode configuration. The IDAC Sourcing mode is recommended for all applications because, unlike the IDAC Sinking mode, it is free from power supply noise.

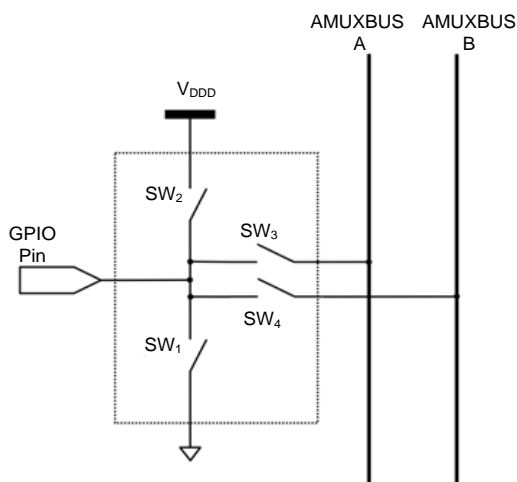
Figure 3-2 CapSense CSD Block in IDAC Sourcing Mode



### 3.1.1.1.1 GPIO Cell Capacitance to Current Converter

In the CapSense CSD system, the GPIO cells are configured as switched-capacitance circuits that convert sensor capacitances into equivalent currents. [Figure 3-3](#) shows a simplified diagram of the GPIO cell structure.

Figure 3-3. GPIO Cell Structure

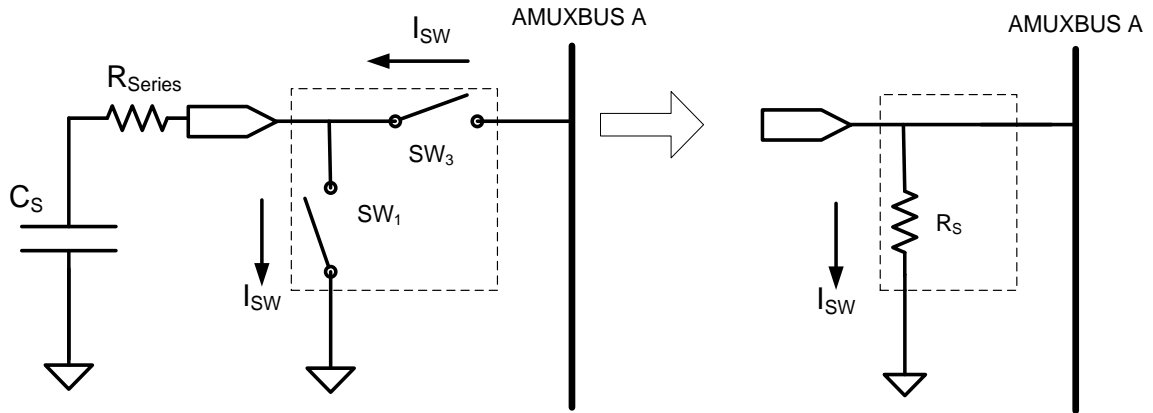


PSoC 4 and PRoC BLE devices have two analog multiplexer buses: AMUXBUS A is used for CSD sensing and AMUXBUS B is used for [CapSense CSD Shielding](#). The GPIO switched-capacitance circuit has two possible configurations: source current to AMUXBUS A or sink current from AMUXBUS A.

### 3.1.1.1.2 IDAC Sourcing Mode (Recommended)

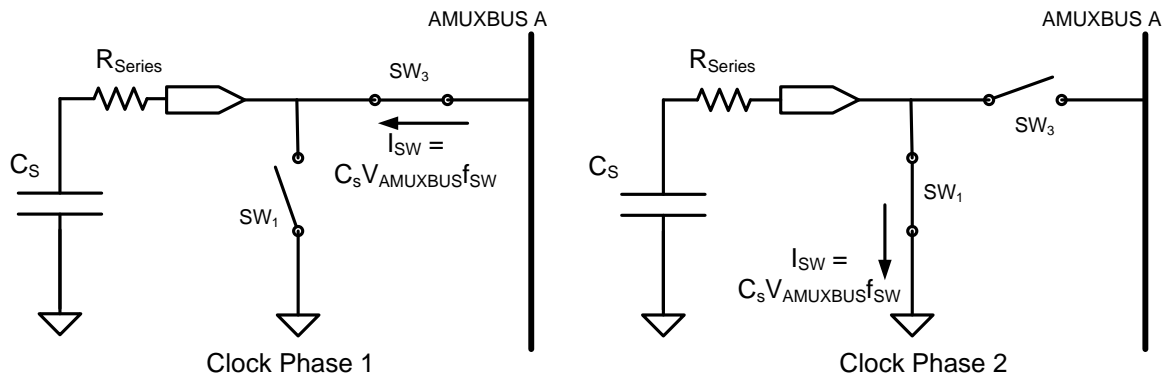
In the IDAC Sourcing mode, the GPIO cell sinks current from the AMUXBUS A through a switched capacitor circuit as Figure 3-4 shows.

Figure 3-4. GPIO Cell Sinking Current from AMUXBUS A



Two non-overlapping, out-of-phase clocks of frequency  $F_{SW}$  control the switches  $SW_1$  and  $SW_3$  as Figure 3-5 shows. The continuous switching of  $SW_1$  and  $SW_3$  forms an equivalent resistance  $R_S$ , as Figure 3-4 shows.

Figure 3-5.  $SW_1$  and  $SW_3$  Switch in a Non-overlapping Manner



If the switches operate at a sufficiently low frequency  $f_{SW}$ , such that time  $T_{SW}/2$  is sufficient to fully charge the sensor to  $V_{REF}$  and fully discharge it to ground, as Figure 3-5 shows, the value of the equivalent resistance  $R_S$  is given by Equation 3-1:

Equation 3-1: Sensor Equivalent Resistance

$$R_S = \frac{1}{C_S F_{SW}}$$

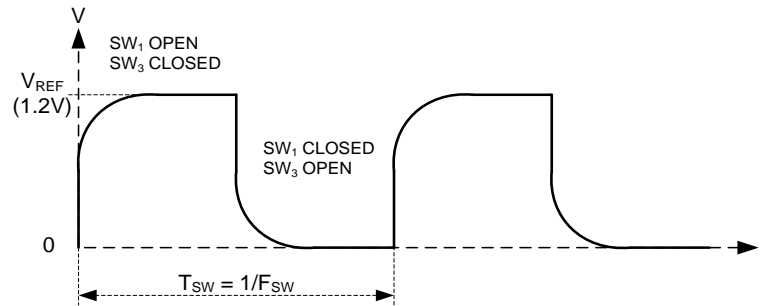
Where:

$C_S$  = Sensor capacitance

$F_{SW}$  = Frequency of the sense clock

The sigma-delta converter maintains the voltage of AMUXBUS A at a constant  $V_{REF}$  (this process is explained in Sigma Delta Converter). Figure 3-6 shows the resulting voltage waveform across  $C_S$ .

Figure 3-6. Voltage across Sensor Capacitance



Equation 3-2 gives the value of average current taken from AMUXBUS A.

Equation 3-2: Average Current Sunk from AMUXBUS A to GPIO through CapSense Sensor ( $I_{CS}$ )

$$I_{CS} = C_S F_{SW} V_{REF}$$

### 3.1.1.1.3 IDAC Sinking Mode (Not Recommended for New Designs)

In the IDAC Sinking mode, the GPIO cell sources current to the AMUXBUS A through a switched capacitor circuit as Figure 3-7 shows. Figure 3-8 shows the voltage waveform across the sensor capacitance.

Because this mode charges the AMUXBUS A directly through V<sub>DDD</sub>, it is more susceptible to power supply noise compared to the IDAC Sourcing mode. Hence, this mode is not recommended for new designs.

Figure 3-7. GPIO Cell Sourcing Current to AMUXBUS A

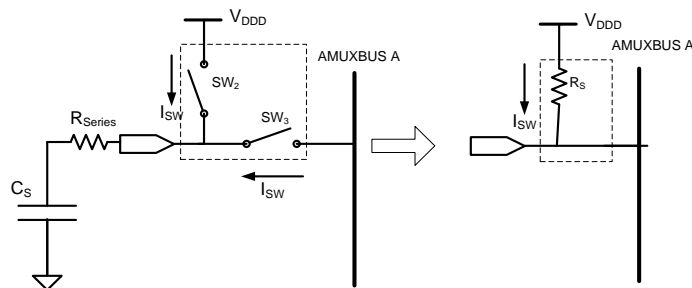
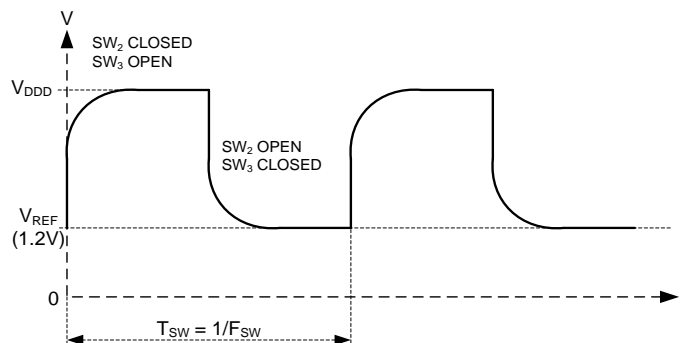


Figure 3-8. Voltage across Sensor Capacitance



Equation 3-3 gives the value of average current supplied to AMUXBUS A.

Equation 3-3: Average Current Sourced to AMUXBUS A from GPIO through CapSense Sensor ( $I_{CS}$ )

$$I_{CS} = C_S F_{SW} (V_{DDD} - V_{REF})$$

### 3.1.1.1.4 CapSense Clock Generator

This block generates the sense clock,  $F_{SW}$ , and the modulation clock,  $F_{MOD}$ , from the high-frequency system resource clock (HFCLK), as [Figure 3-1](#) and [Figure 3-2](#) show.

#### 3.1.1.1.4.1 Sense Clock

The sense clock, also referred to as the switching clock, drives the non-overlapping clocks to the GPIO cell switched capacitor circuits for the [GPIO Cell Capacitance to Current Converter](#). This clock output has three options: direct, 8-bit pseudo random sequence (PRS), and 12-bit PRS.

Direct clock implies that the sense clock is directly driven from the sense clock divider and hence it is a constant frequency clock. In this case, the sense clock frequency is given by the following equation:

Equation 3-4: Direct Clock Frequency

$$f_{SW} = \text{HFCLK} / (\text{Sense Clock Divider} * 2)$$

PRS clock implies that the sense clock is driven from a PRS block, which can generate either 8-bit or 12-bit PRS. Use of the PRS clock spreads the sense clock frequency over a wide frequency range. The maximum frequency of PRS output is given in the following equation:

Equation 3-5: Maximum PRS Clock Frequency

$$f_{SW\_max} = \text{HFCLK} / (\text{Sense Clock Divider} * 2)$$

The following equation gives the average sense clock frequency when either an 8-bit or a 12-bit PRS is used

Equation 3-6: Average PRS Clock Frequency

$$f_{SW\_avg} = \text{HFCLK} / (\text{Sense Clock Divider} * 4)$$

#### 3.1.1.1.4.2 Modulation Clock

The modulation clock is used by the sigma-delta converter. This clock configures the sensor scan time based on the following equation:

Equation 3-7: Sensor Scan Time

$$\text{Sensor scan time} = \frac{(2^{\text{Resolution}} - 1)}{\text{Modulator Clock Frequency}}$$

Here, “Resolution” is the scan resolution; that is, the resolution of the sigma-delta converter.

The modulator clock frequency is given in the following equation:

Equation 3-8: Modulator Clock Frequency

$$\text{Modulator Clock frequency} = \text{HFCLK} / \text{Modulator Clock Divider}$$

### 3.1.1.1.5 Sigma Delta Converter

The sigma-delta converter converts the input current to a corresponding digital count. It consists of a sigma-delta converter, a clock generator known as a modulator clock divider, and two current sourcing/sinking digital-to-analog converters (IDACs), as [Figure 3-1](#) on page 28 shows.

The sigma-delta modulator controls the current of the 8-bit IDAC in an on/off manner. This IDAC is known as the modulation IDAC. The 7-bit IDAC, known as the compensation IDAC, is either always ON or always OFF.

The sigma-delta converter can operate in either single IDAC mode or dual IDAC mode:

- In the **single IDAC mode**, the modulation IDAC (8-bit IDAC) is controlled by the sigma-delta modulator; the compensation IDAC (7 bit IDAC) is always OFF.



- In the **dual IDAC mode**, the modulation IDAC (8-bit IDAC) is controlled by the sigma-delta modulator; the compensation IDAC (7-bit IDAC) is always ON.

The sigma-delta converter also requires an external integrating capacitor, called modulator capacitor  $C_{MOD}$ , as [Figure 3-1](#) on page 28 shows. The recommended value of  $C_{MOD}$  is 2.2 nF.

The sigma delta modulator maintains the voltage across  $C_{MOD}$  at  $V_{REF}$ . It works in one of the following modes:

- **IDAC Sourcing Mode (Recommended)**: In this mode, the switched-capacitor circuit sinks current from  $C_{MOD}$  through AMUXBUS A, and the IDACs then source current to AMUXBUS A to balance its voltage.
- **IDAC Sinking Mode (Not Recommended for New Designs)**: In this mode, the IDACs sink current from  $C_{MOD}$  through AMUXBUS A, and the switched-capacitor circuit sources current to AMUXBUS A to balance its voltage.

[Figure 3-2](#) shows the configuration in IDAC Sourcing mode.

In both cases, the modulation IDAC current is switched ON and OFF corresponding to the small voltage variations across  $C_{MOD}$  to maintain the  $C_{MOD}$  voltage at  $V_{REF}$ .

The sigma-delta converter can operate from 8-bit to 16-bit resolutions. In the **single IDAC mode**, the raw count is proportional to the sensor capacitance. If 'N' is the resolution of the sigma-delta converter and  $I_{MOD}$  is the value of the modulation IDAC current, the approximate value of raw count in the IDAC Sourcing mode is given by [Equation 3-9](#).

Equation 3-9: Single IDAC Sourcing Raw Count

$$\text{raw count} = (2^N - 1) \frac{V_{REF} F_{SW}}{I_{MOD}} C_S$$

Similarly, the approximate value of raw count in the IDAC Sinking mode is:

Equation 3-10: Single IDAC Sinking Raw Count

$$\text{raw count} = (2^N - 1) \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S$$

In both cases, the raw count is proportional to sensor capacitance  $C_S$ . The raw count is then processed by the CapSense CSD Component firmware to detect touches. The hardware parameters such as  $I_{MOD}$ ,  $I_{COMP}$ , and  $F_{SW}$ , and the firmware parameters, should be tuned to optimum values for reliable touch detection. For an in-depth discussion of the tuning, see [CapSense Performance Tuning](#).

In the **dual IDAC mode**, the compensation IDAC is always ON. If  $I_{COMP}$  is the compensation IDAC current, the equation for the raw count in the IDAC Sourcing mode is:

Equation 3-11: Dual IDAC Sourcing Raw Count

$$\text{raw count} = (2^N - 1) \frac{V_{REF} F_{SW}}{I_{MOD}} C_S - (2^N - 1) \frac{I_{COMP}}{I_{MOD}}$$

Raw count in the IDAC Sinking mode is given by [Equation 3-12](#).

Equation 3-12: Dual IDAC Sinking Raw Count

$$\text{raw count} = (2^N - 1) \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S - (2^N - 1) \frac{I_{COMP}}{I_{MOD}}$$

Note that raw count values are always positive. It is thus imperative to ensure that  $I_{COMP}$  is less than  $(V_{DD} - V_{REF}) C_S F_{SW}$  for the IDAC Sinking mode and  $I_{COMP}$  is less than  $C_S F_{SW} V_{REF}$  for the IDAC Sourcing mode. [Equation 3-11](#) does not hold true if  $I_{COMP} > V_{REF} C_S F_{SW}$  and [Equation 3-12](#) does not hold true if  $I_{COMP} > (V_{DD} - V_{REF}) C_S F_{SW}$ ; in these cases, raw counts will be zero.

The relation between the parameters shown in the above equation to the CapSense Component parameters is listed in [Table 3-1](#).

Table 3-1. Relation between CapSense Raw Count Equation and CapSense CSD Hardware Parameters

Sl. No.	Parameter	CapSense v3.0 Component Parameter	CapSense v2.40 Component Parameter	Comments
1	N	Scan Resolution	Scan Resolution	Scan resolution is configurable from 8-bit to 16-bit
2	V <sub>REF</sub>	N/A	N/A	The V <sub>REF</sub> value is 1.2 V
3	F <sub>SW</sub>	Sense Clock Frequency	Sense clock divider	Sense clock frequency and sense clock source decides the frequency at which sensor is switching
		Sense Clock Source	Analog switch drive source	Refer section "Sense clock" for relation between FSW and sense clock divider
4	I <sub>MOD</sub>	Modulator IDAC	Modulation IDAC	I <sub>MOD</sub> = Modulation IDAC * 1.2uA/bit (4x range) I <sub>MOD</sub> = Modulation IDAC * 2.4uA/bit (8x range) <sup>[1]</sup>
5	I <sub>COMP</sub>	Compensation IDAC	Compensation IDAC	I <sub>COMP</sub> = Compensation IDAC * 1.2uA/bit (4x range) I <sub>COMP</sub> = Compensation IDAC * 2.4uA/bit (8x range) <sup>a</sup>
6	V <sub>DD</sub>	N/A	N/A	This parameter is the device supply voltage
7	C <sub>S</sub>	N/A	N/A	This parameter is the sensor parasitic capacitance
8	N/A	Modulator Clock Frequency	Modulator clock divider	Modulator clock divider does not impact raw count equation Refer section "Modulation clock" for relation between modulator clock frequency and modulator clock divider

\* The CapSense v3.0 Component does not support 8x mode.

### 3.1.1.1.6 Analog Multiplexer

The sigma delta converter scans one sensor at a time. An analog multiplexer selects one of the GPIO cells and connects it to the input of the sigma delta converter, as [Figure 3-1](#) on page 28 shows. The AMUXBUS A and the GPIO cell switches (see SW<sub>3</sub> in [Figure 3-7](#) on page 31) form this analog multiplexer. AMUXBUS A connects to all GPIOs that support CapSense. See your corresponding device [Datasheet](#) for a list of port pins that support CapSense. AMUXBUS A also connects the integrating capacitor C<sub>MOD</sub> to the sigma-delta converter circuit.

### 3.1.1.1.7 CapSense CSD Shielding

PSoC 4 and PRoC BLE CapSense supports shield electrodes for liquid tolerance and proximity sensing. See the [Liquid Tolerance](#) section for details. The shield electrode is always kept at the same potential as the sensors. CapSense has a shielding circuit that drives the shield electrode with a replica of the sensor switching signal (see [GPIO Cell Capacitance to Current Converter](#)) to nullify the potential difference between sensors and shield electrode.

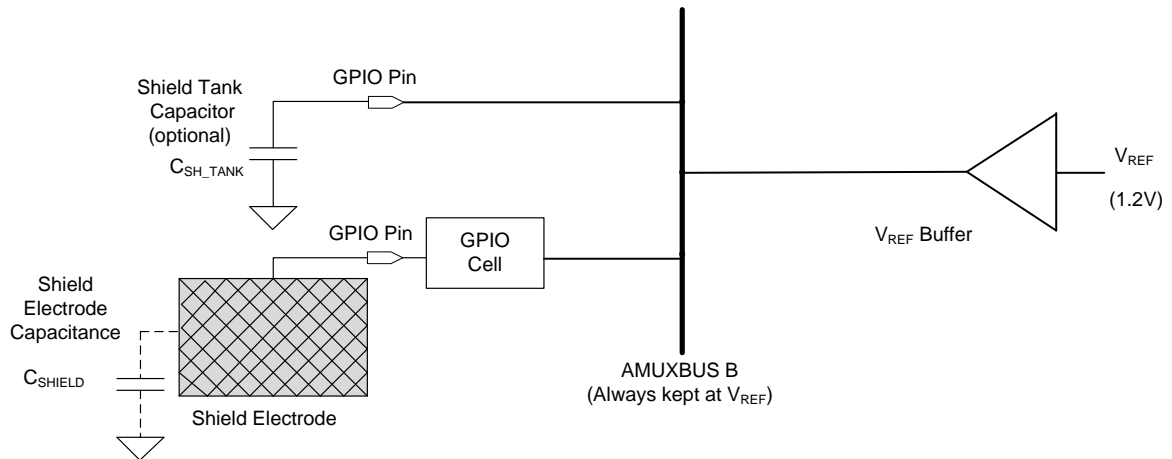
In the sensing circuit, the sigma delta converter keeps the AMUXBUS A at V<sub>REF</sub> (see [Sigma Delta Converter](#)). The GPIO cells generate the sensor waveforms by [switching the sensor](#) between AMUXBUS A and a supply rail (either V<sub>DD</sub> or ground, depending on the configuration). The shielding circuit works in a similar way; AMUXBUS B is always kept at V<sub>REF</sub>. The GPIO cell switches the shield between AMUXBUS B and a supply rail (either V<sub>DD</sub> or ground, the same configuration as the sensor). This process generates a replica of the sensor switching waveform on the shield electrode.

<sup>[1]</sup> To simplify CapSense tuning, the CapSense v3.0 Component does not support 8x range.

Depending on how AMUXBUS B is kept at  $V_{REF}$ , two different configurations are possible:

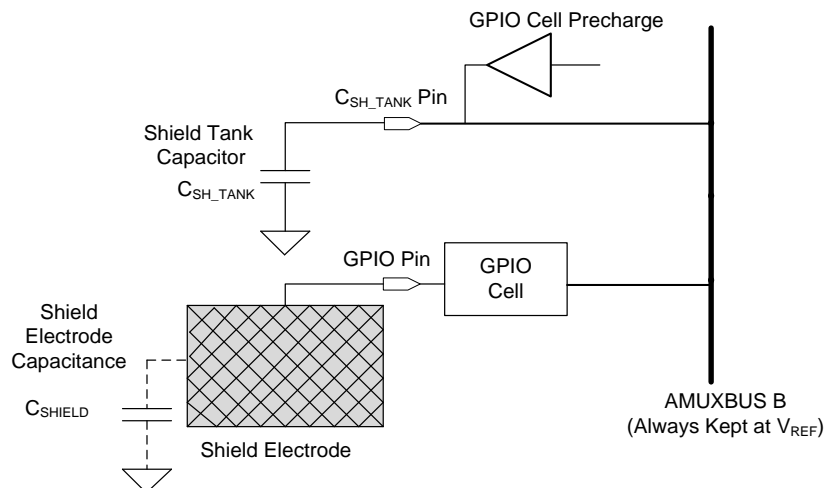
- Shield driving using  $V_{REF}$  buffer: In this configuration, a voltage buffer is used to drive AMUXBUS B to  $V_{REF}$ , as Figure 3-9 shows. An external 10-nF  $C_{SH\_TANK}$  capacitor is recommended to reduce switching transients.

Figure 3-9. Shield Driving Using  $V_{REF}$  Buffer



- Shield driving using GPIO cell precharge: This configuration requires an external 10 nF  $C_{SH\_TANK}$  capacitor, as Figure 3-10 shows. A special GPIO cell charges the  $C_{SH\_TANK}$  capacitor and hence the AMUXBUS B to  $V_{REF}$ . This is the recommended method for precharging of shield drive as the noise on raw counts is less in this case compared to that in the shield driving using the  $V_{REF}$  buffer.

Figure 3-10. Shield Driving Using GPIO Precharge



This GPIO cell precharge capability is available only on a fixed  $C_{SH\_TANK}$  pin. See the device pinout in your corresponding device [Datasheet](#) for details.

Shield drive can be configured through the “Shield tank capacitor” option in the CSD Settings tab of the CapSense Component.

#### 3.1.1.2 CSX Sensing Method

Figure 3-11 shows the simple representation of a CapSense system configured for CSX sensing method in PSoC 4.

**Note** PSoC 4100 does not support the CSX sensing method.

Figure 3-11. CapSense CSX Sensing Method Configuration

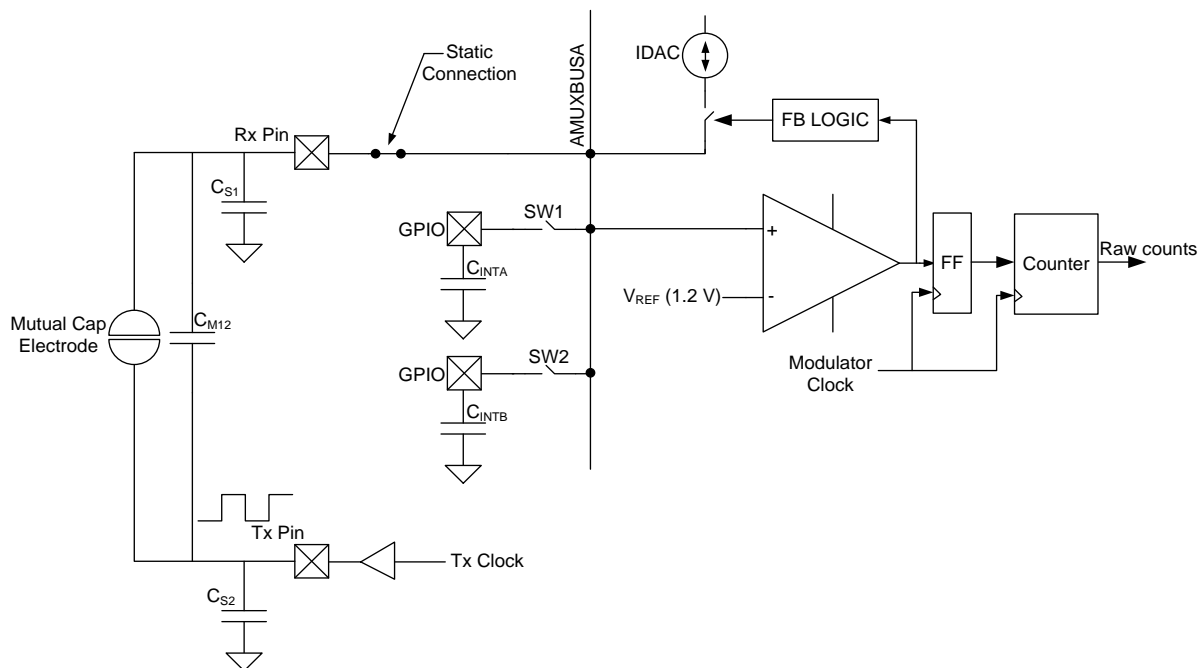
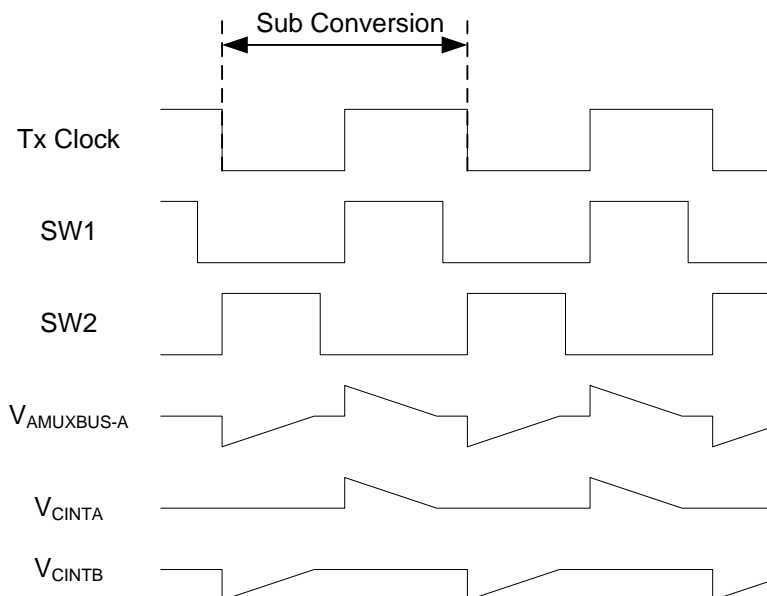


Figure 3-12. CSX Sensing Waveforms



The CSX sensing method measures the mutual capacitance between the Tx electrode and Rx electrode, as shown in [Figure 3-11](#). The Tx electrode is excited by a digital waveform (Tx clock), which switches between VDDIO (VDDD if VDDIO is not available) and ground. The Rx electrode is statically connected to AMUXBUS A. The CSX method requires two external integration capacitors, C<sub>INTA</sub> and C<sub>INTB</sub>. The value of these capacitors is listed in [Table 6-6](#).

Figure 3-12 shows the waveforms on the Tx electrode and C<sub>INTA</sub> and C<sub>INTB</sub> capacitors. The sampling – a process of producing a “sample” – is started by the firmware by initializing the voltage on both external capacitors to V<sub>REF</sub> and performing a series of sub-conversions. The sum of all sub-conversions in a sample is the result, and is referred to as “raw count”. A sub-conversion is a capacitance to count conversions performed within a Tx clock cycle.

During a sub-conversion, both Sw1 and Sw2 switches are operated in phase with the Tx clock. On the rising edge of the Tx clock, SW1 is closed (SW2 is open during this time) and charge flows from the Tx electrode to the Rx electrode. This charge is integrated onto the C<sub>INTA</sub> capacitor, which increases the voltage on C<sub>INTA</sub>. The IDAC is configured in sink mode to discharge the C<sub>INTA</sub> capacitor back to voltage V<sub>REF</sub>.

On the falling edge of the Tx clock, SW2 is closed (SW1 is open during this time) and the charge flows from the Rx electrode to the Tx electrode. This causes the voltage on C<sub>INTB</sub> to go below V<sub>REF</sub>. The IDAC is configured in source mode to bring the voltage on C<sub>INTB</sub> back to V<sub>REF</sub>. The change transferred between Tx and Rx electrodes in both the cycles is proportional to mutual capacitance, C<sub>M</sub>, between the electrodes. The comparator output enables the counter while the IDAC is charging or discharging the external capacitors. The counter counts in terms of modulator clock cycles during a sub-conversion. Multiple sub-conversions are performed and the result is accumulated to the same counter to produce “raw count” for a sensor.

The modulator clock is used to measure the time taken to charge/discharge external capacitors within a Tx clock cycle. For this reason, modulator clock frequency must be always greater than Tx clock frequency; higher modulator clock frequency leads to better accuracy. For proper operation, the IDAC current should be set such that the C<sub>INTA</sub> and C<sub>INTB</sub> capacitors are charged/discharged within one Tx clock cycle. The CapSense Component provides an option to automatically calibrate the IDAC. It is recommended to enable this option.

Equation 3-13. Raw Count Relationship for Mutual Capacitance Sensing

$$\text{Raw count} = \frac{2 * V_{TX} * F_{SW} * C_M * (2^{16} - 1)}{IDAC}$$

Where,

IDAC – IDAC current

C<sub>M</sub> – Mutual capacitance between Tx and Rx electrodes

V<sub>TX</sub> – Amplitude of the Tx signal

F<sub>SW</sub> – Tx clock frequency

The Number of Conversions parameter, specified in the CapSense Component controls the duration for which a sensor is scanned. The number of conversions is related to Tx Clock and Modulator Clock according to Equation 3-14.

Equation 3-14. Relationship between Number of Conversions, Tx Clock Frequency, and Modulator Clock Frequency

$$\text{Number of Conversions} < \frac{2^{16} * \text{Tx Clock Frequency}}{\text{Modulator Clock Frequency}}$$

### 3.1.2 CapSense Architecture in PSoC 4 S-Series

The fourth-generation CapSense architecture in PSoC 4 S-Series is an improved version of previous generation CapSense architecture. The main differences in the CapSense architecture between the PSoC 4 devices are listed in [Table 3-2](#).

Table 3-2. Comparison of CapSense Architecture

Feature	Third-Generation CapSense (PSoC 4, 4-M, 4-BLE and 4-L)	Fourth-Generation CapSense (PSoC 4-S)	Advantages of Fourth-Generation over Third Generation CapSense
<b>Sensing Modes</b>	Self-Cap and Mutual-Cap modes <sup>a</sup>	Self-Cap, Mutual-Cap, and ADC modes	Additional ADC functionality with the same hardware
<b>Sensor Parasitic Capacitance (C<sub>P</sub>) Range</b>	5 pF – 60 pF	5 pF – 200 pF	Supports high-C <sub>P</sub> design applications
<b>V<sub>REF</sub></b>	1.2	0.6 V to VDDA-0.6 V <sup>b</sup>	Improved SNR
<b>IDAC LSB Size</b>	1.2 uA, 2.4 uA	37.5 nA, 300 nA, 2.4 uA	Improved sensitivity
<b>Split IDAC Capability</b>	Requires two IDACs	Requires one IDAC <sup>c</sup>	Requires less resource to achieve same performance and frees up one IDAC for general purpose use
<b>EMI Reduction - Digital</b>	-	Spread Spectrum - CSD controlled	Spread Spectrum clock is generated by hardware, and CPU is completely free
<b>10-bit ADC</b>	No	Yes	The same CSD hardware can be used as an ADC when CapSense Scanning is not in progress.
<b>Hardware State Machine<sup>d</sup></b>	No	Yes	CPU is no longer required for Initialization or Spread Spectrum SenseCk generation

The CapSense hardware in the PSoC 4 S-Series supports self-capacitance (CSD) and mutual-capacitance (CSX)-based capacitive sensing. The hardware also supports input voltage measurement when CapSense scanning is not in progress.

This section explains how the CapSense hardware is used for performing self-capacitance and mutual-capacitance sensing. Refer to the **CapSense** chapter in the PSoC 4 S-Series device [Technical Reference Manual](#) for a detailed explanation of the CapSense hardware in the PSoC 4 S-Series of devices. A basic knowledge of CapSense architecture in PSoC 4 S-Series is required to understand the self-capacitance and mutual-capacitance working.

**Self-Capacitance Sensing** [Figure 3-13](#) shows the circuit diagram of the CapSense block configured for self-capacitance mode. In this mode, the sensor connected to the GPIO pin is alternately connected to AMUXBUS A and to ground by a non-overlapping clock (ph1, ph2). This clock is called the Sense Clock (SenseCk) and is generated by the CapSense block. The V<sub>REF</sub> voltage is programmable<sup>b</sup> and can range from 0.6 V to VDDA-0.6 V.

During phase ph1, the sensor (C<sub>S</sub>) is connected to ground and is completely discharged. During phase ph2, the sensor capacitor is connected to AMUXBUS A and hence the voltage on the C<sub>MOD</sub> capacitor drops below V<sub>REF</sub>. The CSD comparator (csdcomp) compares the voltage across its terminals and controls the IDAC ON/OFF condition. When the voltage on C<sub>MOD</sub> is below V<sub>REF</sub>, the IDAC/A/B sources current to the C<sub>MOD</sub> capacitor to maintain the voltage across the capacitor at ~V<sub>REF</sub>.

<sup>a</sup> PSoC 4100 family does not support CSX sensing method since it does not have UDB resources which is required to implement CSX for the this family

<sup>b</sup> The CapSense component automatically selects the V<sub>REF</sub> voltage depending on the VDDA volatage specified in the cydwr window

<sup>c</sup> Require one IDAC if compensation and modulation IDAC split is 50-50; if it is not 50-50, it requires two IDACs.

<sup>d</sup> The hardware state machine is a logic which controls the CapSense block and sensor scanning.

The comparator output is also fed to the CSD sequencer and counter. The CSD sequencer generates the control signals required to control the switches in the CapSense block and the counter counts the duration for which the voltage on  $C_{MOD}$  was less than  $V_{REF}$ . This digital count value is called raw count.

When you touch the sensor, the total sensor capacitance ( $C_S + C_F$ ) increases; hence, the voltage on  $C_{MOD}$  drops below  $V_{REF}$  by a larger amount. In this case, the IDAC takes longer to charge it back to  $V_{REF}$ . This results in the counter requiring to count for a longer duration. The firmware interprets this change in raw counts to detect touch/no-touch condition.

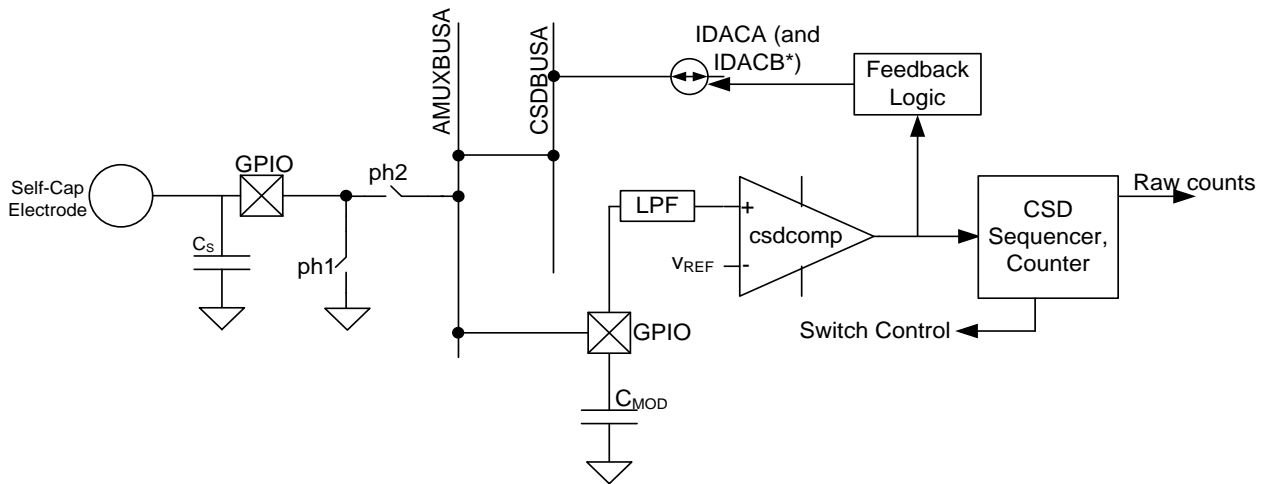
The relation between the raw count and CapSense hardware parameters is given by:

Equation 3-15. Raw Count Equation in Self-Capacitance Sensing Mode

For compensation IDAC disabled: 
$$\text{raw count} = (2^N - 1) \frac{V_{REF} F_{SW}}{I_{MOD}} C_S$$

For compensation IDAC enabled: 
$$\text{raw count} = (2^N - 1) \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S - (2^N - 1) \frac{I_{COMP}}{I_{MOD}}$$

Figure 3-13. CapSense Configuration in Self-Capacitance Measurement Mode



\* The IDACB is used only when the IDAC split is not 50-50.

### 3.1.2.1 Mutual Capacitance Sensing

Figure 3-14 shows the CapSense system configuration for mutual-capacitance measurement. The Tx electrode is excited by a signal (Tx clock), which switches between VDDIO or VDDD (if VDDIO is not available) and GND. This clock is generated by the CapSense block. The Rx electrode is statically connected to AMUXBUS A. The  $C_{INTA}$  and  $C_{INTB}$  capacitors are statically connected (alternately) to AMUXBUS A on the ph1 and ph2 phases respectively. The input to the csdcmp is switched between  $C_{INTA}$  and  $C_{INTB}$  during CapSense scanning.

Figure 3-14. CapSense Configuration in Mutual-Capacitance Measurement Mode

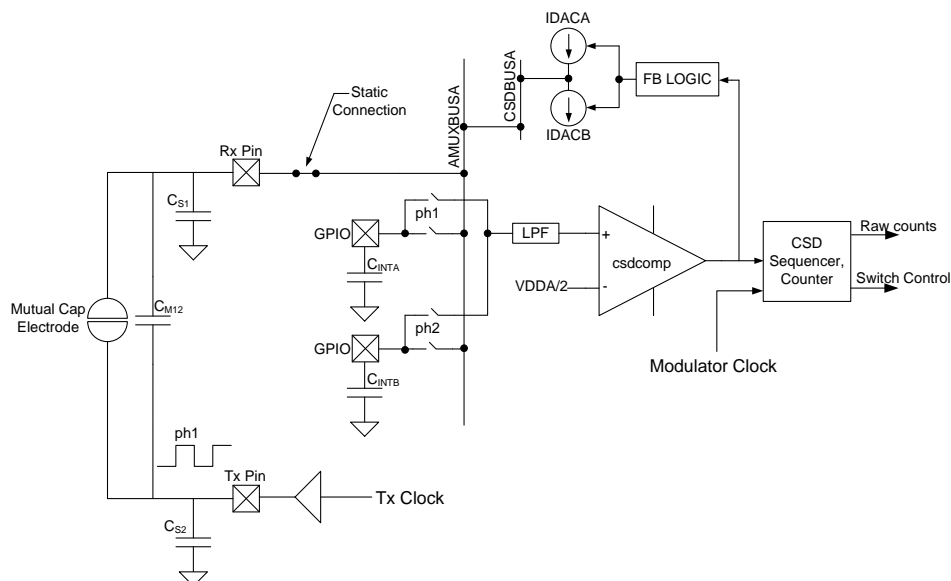
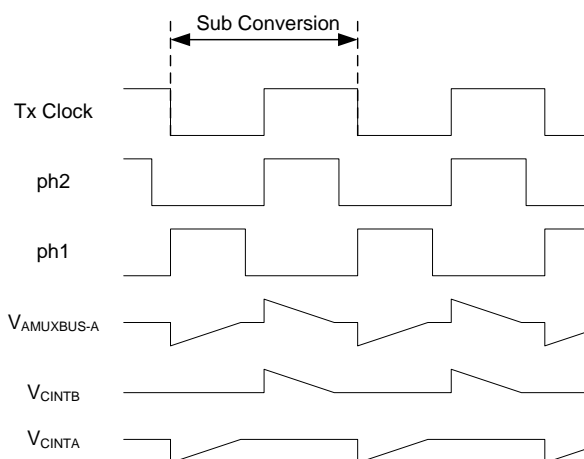


Figure 3-15. CSX Waveforms



The  $C_{INTA}$  and  $C_{INTB}$  capacitors are initialized to  $V_{DDA}/2$  (reference voltage of comparator) before scanning begins. The voltage on the  $C_{INTA}$  and  $C_{INTB}$  capacitors are maintained at  $V_{DDA}/2$  using IDACA/B.

When Tx clock is high (ph2), positive charge from  $C_{M12}$  is coupled onto the  $C_{INTB}$  capacitor. When positive charge couples to the  $C_{INTB}$  capacitor, the voltage on the capacitor increases above  $V_{DDA}/2$ . This causes the comparator to trip and the IDACB (configured in Sinking mode) is turned on to pull the voltage on the  $C_{INTB}$  capacitor back to  $V_{DDA}/2$ . At the same time, the counter is started. The counter counts until the voltage on the  $C_{INTB}$  capacitor reaches  $V_{DDA}/2$ .

Similarly, when the Tx clock is low (ph1), charge gets coupled from the Rx electrode to Tx electrode. This causes voltage on the C<sub>INTA</sub> capacitor to drop below V<sub>DDA</sub>/2. At the same time, the counter is started and the IDACA (configured in Sourcing mode) is turned on to bring the voltage on the C<sub>INTA</sub> capacitor back to V<sub>DDA</sub>/2. When the voltage on C<sub>INTA</sub> reaches V<sub>DDA</sub>/2, the counter is stopped. A sub-conversion is a capacitance-to-counts conversion performed within a Tx clock cycle. The sum of all sub-conversions in a sample is the result and is referred as “raw count”. The raw count equation for CSX sensing method is the same as [Equation 3-13](#).

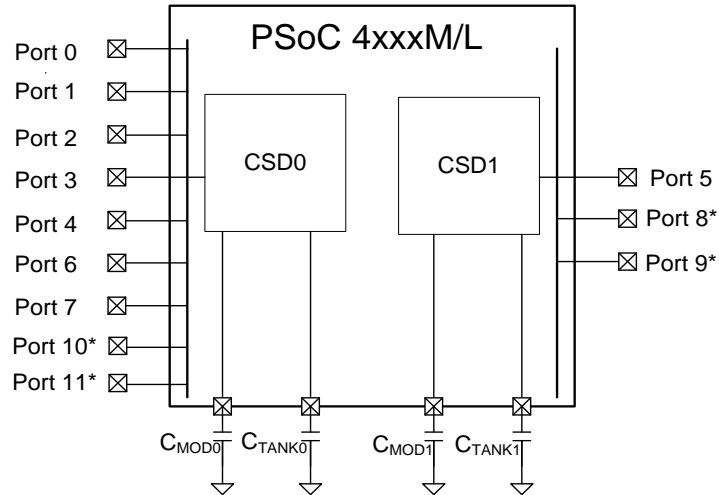
When a finger touches the sensor, the mutual capacitance between the electrodes decreases and hence the dip in voltage on  $C_{INTA}$  and  $C_{INTB}$  capacitors is reduced. This results in decrease in duration for which IDAC is ON and hence the counter counts for a smaller duration. The firmware interprets this reduction in raw counts to detect touch/no-touch condition.



### 3.2 CapSense in PSoC 4xxxM/4xxxL-Series

The PSoC 4xxxM/4xxxL series of devices support two CapSense blocks – CSD0 and CSD1. Each block has the same functionality and performance as explained in the [CSD Sensing Method](#) section. The main difference between the CSD0 and CSD1 blocks in PSoC 4xxxM is that the CSD0 block can scan CapSense sensors on all GPIOs except Port 5 pins and the CSD1 block can scan CapSense sensors on only Port 5 pins as shown in [Figure 3-16](#).

Figure 3-16. CapSense in PSoC 4 M-Series



\*Port 8, 9, 10, and 11 are available only on the PSoC 4xxxL family of devices. Port 12 in the PSoC 4xxxL family cannot be used for CapSense.

Each CSD block requires a separate  $C_{MOD}$  and  $C_{SH\_TANK}$  capacitor. The summary of differences between CSD0 and CSD1 blocks is listed in [Table 3-3](#).

Table 3-3. Difference between CSD0 and CSD1 Blocks in PSoC 4xxxM/L-Series

	CSD0	CSD1
$C_{MOD}$	Refer to <a href="#">Table 6-7</a> for recommended pins.	
$C_{SH\_TANK}$		
$C_{INTA/B}$		
CapSense Pin	Any pin except PORT5 pins (for PSoC 4xxxM). Any pin except PORT5, 8, and 9 pins (for PSoC 4xxxL)	Any pin in PORT5 (for PSoC 4xxxM). Any pin in PORT5, 8, and 9 (for PSoC 4xxxL)
Shield Pin	Any pin except PORT5 pins (for PSoC 4xxxM). Any pin except PORT5, 8, and 9 pins (for PSoC 4xxxL)	Any pin in PORT5 (for PSoC 4xxxM). Any pin in PORT5, 8, and 9 (for PSoC 4xxxL)
Max Number of CapSense Pins	47* (for PSoC 4xxxM) 68 (for PSoC 4xxxL)	4* (for PSoC 4xxxM) 22 (for PSoC 4xxxL)

\* Maximum number of pins are specified, excluding two pins used for  $C_{MOD}$  and  $C_{SH\_TANK}$  in the design.

**Note** Because the CSD0 and CSD1 blocks use different shield pins, isolate the shield hatch of the CSD0 sensors from the shield hatch of the CSD1 sensors.

To select a specific CSD block, follow this procedure:

- Place the CapSense CSD Component in the PSoC Creator schematic.  
**Note** The CapSense v3.0 Component does not support sensing using the CSD1 block for the PSoC 4-M series. If you need to use both CapSense blocks, you should use the CapSense\_CSD v2.40 Component.
- In the PSoC Creator cydwr pins tab, assign the  $C_{MOD}$  pin depending on the required CSD block, as shown in [Figure 3-17](#). For example, if you want to use the CSD0 block, select  $C_{MOD}$  pin as P4.2.

Figure 3-17. Selecting CSD0 or CSD1 Block in PSoC 4xxxM/L-Series

Alias	Name	Port	Pin	Lock
Cmod	\CapSense_1:Cmod\			
Button0_BTN	\CapSense_1:Sns[0]\	P4[2] CSD0:c_mod, SCB0:uart_cts,		
Button1_BTN	\CapSense_1:Sns[1]\	P5[0] OA2:vplus, CSD1:c_mod, TCPW		

- To use CapSense on the ports allocated for the CSD0 and CSD1 blocks in the same project, place two instances of the CSD Component.

The following is an example code snippet to use both the CSD blocks in the same project:

```

/* Start CapSense Component */
CapSense_1_Start();
CapSense_2_Start();

/* Initialize all baselines */
CapSense_1_InitializeAllBaselines();
CapSense_2_InitializeAllBaselines();

for (;;)
{
    /* Check that scanning is completed */
    if (0u == CapSense_1_IsBusy() && 0u == CapSense_2_IsBusy())
    {
        /* Update all enabled baselines */
        CapSense_1_UpdateEnabledBaselines();
        CapSense_2_UpdateEnabledBaselines();

        /* Start scanning all enabled sensors */
        CapSense_1_ScanEnabledWidgets();
        CapSense_2_ScanEnabledWidgets();
    }
}

```

## 4. CapSense Design and Development Tools



Cypress provides a complete set of hardware and software tools to develop your CapSense application.

### 4.1 PSoC Creator

PSoC Creator is a state-of-the-art, easy-to-use integrated development environment. It offers a unique combination of hardware configuration and software development based on classical schematic entry. You can develop applications in a drag-and-drop design environment using a library of Components. For details, see the [PSoC Creator home page](#).

#### 4.1.1 CapSense Component

PSoC Creator provides a CapSense Component, which is used to create a capacitive touch system in PSoC or PSoC 4 BLE by simply configuring this Component. The Component also provides an application programming interface (API) to simplify firmware development. Some PSoC 4 BLE and PSoC 4 BLE devices also support a CapSense Gesture Component (refer to your corresponding device [Datasheet](#) to see if your device supports this Component). Gesture Component is identical to the CapSense Component, with an additional software feature that supports trackpad gestures.

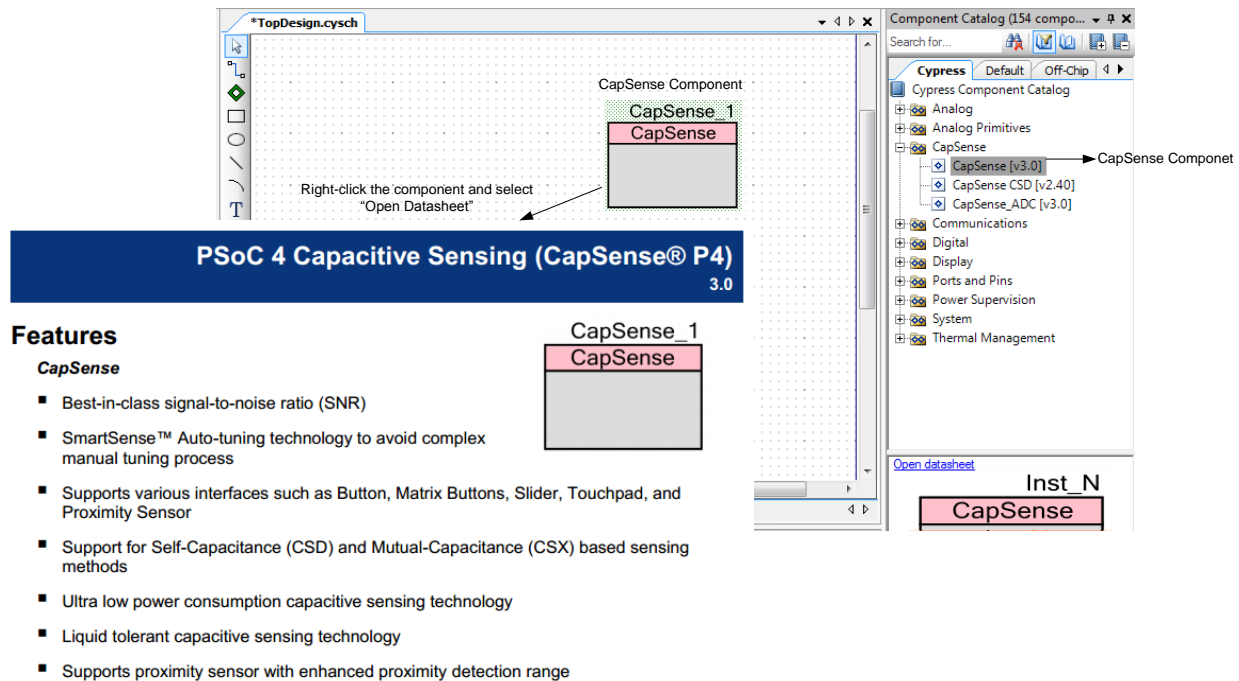
Starting with the PSoC Creator 3.3 SP2 release, two versions of CapSense Component are supported – CapSense CSD v2.40 and CapSense v3.0.

CapSense v3.0 is the latest version and supports existing PSoC 4 devices in addition to the new PSoC 4 S-Series of devices. CapSense v3.0 has several improvements in terms of code density and code execution when compared to CapSense CSD v2.40. It also supports CSX sensing for all PSoC 4 devices. [Figure 4-1](#) shows an example of the PSoC Creator schematic entry with a CapSense Component dragged from the Component catalog and placed on the schematic page.

It is highly recommended to use CapSense v3.0 for all new designs. See the [CapSense v3.0 Component datasheet](#) for the complete list of changes from CapSense CSD v2.40 and guidelines for migrating your projects from CapSense CSD v2.4 to CapSense v3.0.

**Note** The CapSense v3.0 Component does not support touch gestures. This feature will be supported in future versions of the Component. If you want to implement touch gestures, use the [CapSense Gesture v2.4 Component](#).

Figure 4-1. PSoC Creator Component Placement



Each Component has an associated datasheet that explains details about the Component. To open the Component datasheet, right-click on the Component and select “Open Datasheet”.

The CapSense Component also has a Tuner GUI, called the [Tuner Helper](#), to help with the tuning process.

#### 4.1.2 CapSense\_ADC Component

The CapSense\_ADC Component is only applicable for the PSoC 4 S-Series devices. This Component should be used when both CapSense and ADC operations are required. It is similar to CapSense v3.0 and supports using the CapSense block for ADC operation in a time-multiplexed manner.

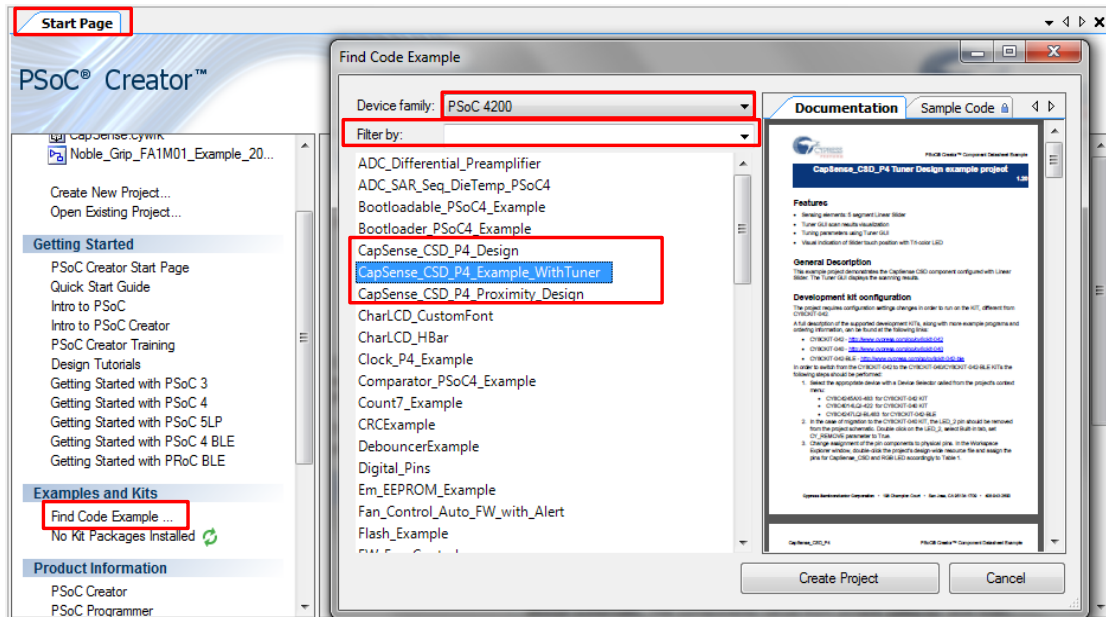
#### 4.1.3 Tuner Helper

Tuner Helper is included with the [CapSense](#) Component and assists in tuning CapSense parameters and monitoring sensor data such as raw count, baseline, and difference count. The procedure to use the tuner is slightly different in CapSense v3.0 when compared to previous versions of the Component. Refer to the respective [CapSense Component datasheet](#) for the detailed procedure on how to use Tuner Helper.

## 4.1.4 Example Projects

You can use the CapSense example projects provided in PSoC Creator to learn schematic entry and firmware development. To find a CapSense example project, go to the PSoC Creator Start Page, click “Find Code Example ...”, and select the appropriate architecture, as Figure 4-2 shows. You can also filter for a project by writing a partial or complete project name in “Filter by” box.

Figure 4-2. PSoC Creator Example Project



## 4.2 Hardware Kits

Table 4-1 lists the development kits that support evaluation of PSoC 4 CapSense.

Table 4-1. PSoC 4 CapSense Development Kits

Development Kit	Supported CapSense Features
PSoC 4000 Pioneer Kit (CY8CKIT-040)	A 5x6 CapSense touchpad and a wire proximity sensor
PSoC 4 S-Series Pioneer Kit (CY8CKIT-041)	Two self- or mutual-capacitive sensing buttons A 7x7 self- or mutual-capacitive sensing touchpad
PSoC 4 S-Series Prototyping Kit (CY8CKIT-145)	Three self- or mutual-capacitive sensing buttons A five-segment self- or mutual-capacitive sensing linear slider
PSoC 4 Pioneer Kit (CY8CKIT-042)	A five-segment linear slider
PSoC 4 BLE Bluetooth Low Energy Pioneer Kit (CY8CKIT-042-BLE)	A five-segment linear slider and a wire proximity sensor
PSoC 4200-M Pioneer Kit (CY8CKIT-044)	A five-element gesture detection and two proximity wire sensors
PSoC 4200-L Pioneer Kit (CY8CKIT-046)	A five-element gesture detection, two proximity wire sensors, and an eight-element radial slider
CapSense Proximity Shield (CY8CKIT-024)	A four-element gesture detection and one proximity loop sensor
CapSense® Liquid Level Sensing Shield (CY8CKIT-022)	A two-element flexible PCB and 12-element flexible PCB
PSoC 4 Processor Module (CY8CKIT-038), with PSoC Development Kit (CY8CKIT-001)	A five-segment linear slider and two buttons
CapSense Expansion Board Kit (CY8CKIT-031), to be used with CY8CKIT-038 and CY8CKIT-001	A 10-segment slider, five buttons and a 4x4 matrix button with LED indication.
MiniProg3 Program and Debug Kit (CY8CKIT-002)	CapSense performance tuning in CY8CKIT-038

Table 4-2 lists the Reference Design Kits that support evaluation of PRoC BLE CapSense.

Table 4-2. PRoC BLE CapSense Reference Design Kits

Reference Design Kit	Supported CapSense Features
PRoC BLE Remote Control Reference Design Kit (CY5672)	An 8 x 8 trackpad with these gestures: <ul style="list-style-type: none"> <li>• Top-edge swipe gesture</li> <li>• One-finger click gesture</li> <li>• Two-finger click gesture</li> <li>• Pinch-zoom gesture</li> <li>• One-finger horizontal scroll gesture</li> <li>• One-finger vertical scroll gesture</li> </ul>
PRoC BLE Touch Mouse Reference Design Kit (CY5682)	A 9 x 3 trackpad with these gestures: <ul style="list-style-type: none"> <li>• One-finger horizontal scroll gesture</li> <li>• One-finger vertical scroll gesture</li> </ul>

# 5. CapSense Performance Tuning



After you have completed the sensor layout (see [PCB Layout Guidelines](#)), the next step is to implement the firmware and tune the CapSense parameters for the sensor to achieve optimum performance. The CapSense sensing method is a combination of hardware and firmware techniques. Therefore, it has several hardware and firmware parameters required for proper operation. These parameters should be tuned to optimum values for reliable touch detection and fast response. Most of the capacitive touch solutions in the market must be manually tuned. Cypress provides a unique feature called SmartSense (also known as Auto-tuning) for PSoC 4 and PRoC BLE CapSense. SmartSense is a firmware algorithm that automatically sets all<sup>a</sup> parameters to optimum values.

## 5.1 Selecting between SmartSense and Manual Tuning

SmartSense auto-tuning reduces design cycle time and provides stable performance across PCB variations, but requires additional RAM and CPU resources, as indicated in the [Component datasheet](#), to allow runtime tuning of CapSense parameters. SmartSense is recommended mainly for conventional CapSense applications involving simple button and slider widgets, and is currently supported only for [self-capacitance sensing](#) and not for [mutual-capacitance sensing](#).

On the other hand, manual tuning requires effort to tune optimum CapSense parameters, but allows strict control over characteristics of capacitive sensing system, such as response time and power consumption. It also allows use of CapSense beyond the conventional button and slider applications such as proximity and liquid-level-sensing.

It is recommended to use SmartSense tuning for conventional CapSense applications involving buttons and slider widgets provided the parasitic capacitance ( $C_P$ ) of these widgets is within the SmartSense-supported range.

For button widgets, SmartSense supports a sensor parasitic capacitance range of 5 pF to 45 pF if the expected finger capacitance is higher than 0.2 pF. If the expected finger capacitance is lower than 0.2 pF, but higher than or equal to 0.1 pF, the supported parasitic capacitance range is 5 pF to 35 pF.

For slider widgets, each individual slider segment should fall in the same  $C_P$  range as supported for button widgets. In addition,  $C_P$  of any slider-segment should be greater than 75 percent of the  $C_P$  of the maximum  $C_P$  segment in the slider. For example, in a slider, if 30 pF is the  $C_P$  of the maximum  $C_P$  segment, the  $C_P$  of other segments should be greater than 22.5 pF.

Use manual tuning for CapSense applications with sensor parasitic capacitance not following the above criteria. You can also use [Manual Tuning](#) where strict control is needed over the sensor-scan-time or other CapSense parameters. In such cases, you can initially use SmartSense to find the optimum tuning parameters and then change the tuning mode to manual tuning.

Note that manual tuning requires I<sup>2</sup>C or UART communication with a host PC.

## 5.2 SmartSense

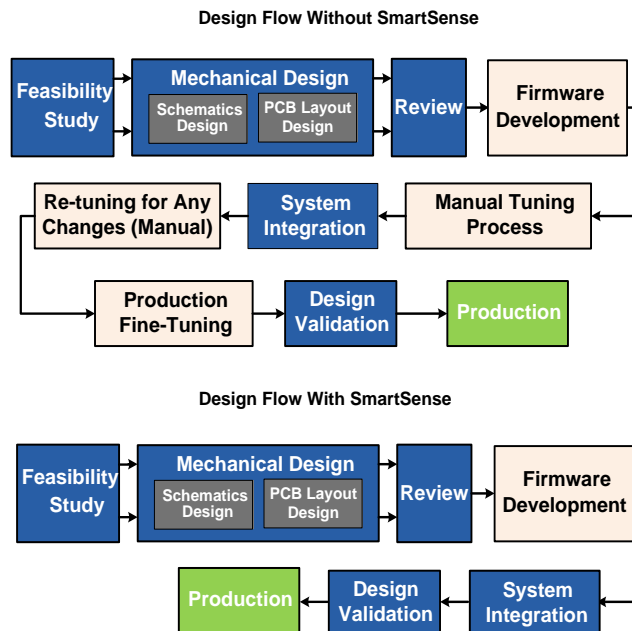
SmartSense<sup>a</sup> is a firmware algorithm that automatically sets all CapSense tuning parameters to optimum values. Some advantages of SmartSense, as opposed to Manual Tuning are:

- **Reduced Design Cycle Time:** The design flow for capacitive touch applications involves tuning all of the sensors. This step can be time consuming if there are many sensors in your design. In addition, you must repeat the tuning when there is a change in the design, PCB layout, or mechanical design. Auto-tuning solves these problems by setting all of the parameters automatically. [Figure 5-1](#) shows the design flow for a typical CapSense application with and without SmartSense.

---

<sup>a</sup> SmartSense is currently applicable only for self-capacitance sensing (CSD) widgets.

Figure 5-1. Design Flow With and Without SmartSense



- **Performance is independent of PCB variations:** The parasitic capacitance,  $C_P$ , of individual sensors can vary due to process variations in PCB manufacturing, or vendor-to-vendor variation in a multi-sourced supply chain. If there is significant variation in  $C_P$  across product batches, the CapSense parameters must be re-tuned for each batch. SmartSense sets parameters for each device automatically, hence taking care of variations in  $C_P$ .
- **Ease of use:** SmartSense is faster and easier to use because only a basic knowledge of CapSense is needed.

Note that SmartSense can be used in multiple ways:

1. **SmartSense (Full Auto-Tune)** – This is the quickest way to tune. This method calibrates all of the CapSense hardware and firmware tuning parameters automatically at runtime. This is the recommended method for most designs.
2. **SmartSense (Hardware parameters only)** – This method auto-tunes all hardware parameters of CapSense automatically, but allows to set user-defined threshold values. This method consumes less flash/RAM resources than SmartSense (Full Auto-Tune). Also, this method avoids the extra processing needed for automatic threshold calculation and hence allows lower power consumption for a given scan rate. Use this method for low-power or noisy designs or in cases with constrained memory requirements.
3. **SmartSense for initial tuning** – You may also use SmartSense for initial tuning, to quickly find the best settings for a CapSense board and then change to manual tuning. This method is useful for cases with strict requirements on response time or power consumption. This is a quick method to find the best settings, instead of starting manual tuning from scratch.

### 5.2.1 Component Configuration for SmartSense

This section explains the Component configuration for the SmartSense mode. For details on manual tuning, see [Manual Tuning](#).

As mentioned in Section 4.1.1 [CapSense Component](#), starting with PSoC Creator 3.3 SP2 release, two versions of CapSense Component are supported – CapSense CSD v2.40 and CapSense v3.0.

CapSense v3.0 is the latest version and supports existing PSoC 4 devices in addition to the new PSoC 4 S-Series of devices. CapSense v3.0 has several improvements in terms of code density and code execution when compared to CapSense CSD v2.40. CapSense v3.0 also supports CSX sensing for all<sup>a</sup> PSoC 4 family of devices.

This section provides details for both CapSense v3.0 and CapSense v2.40 Components in PSoC Creator. It is recommended to use CapSense v3.0 for all new designs; however, if you are using an older version of the CapSense Component for a legacy project, skip to [SmartSense Configuration for CapSense v2.40 and Earlier](#).

<sup>a</sup> CSX is not supported for PSoC 4100 family of devices



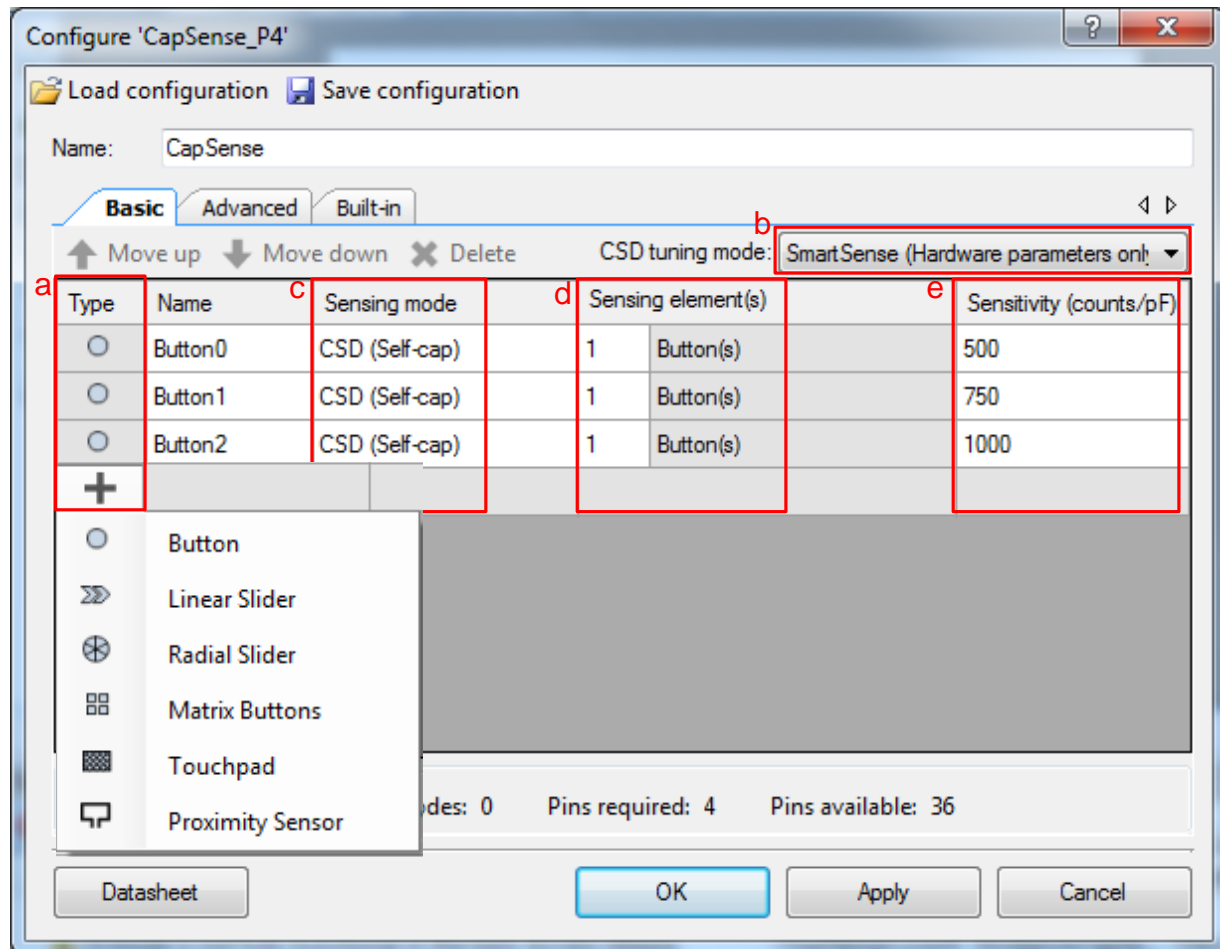
### 5.2.1.1 SmartSense Configuration for CapSense v3.0

To open the CapSense Component configuration window (Figure 5-2), either double-click the Component, or right-click the Component and select **Configure**.

#### 5.2.1.1.1 Basic Settings

Set the Basic tab configuration according to Figure 5-2 and as follows:

Figure 5-2. CapSense Component Basic Tab



- Type:** Specify the widget type by clicking the + symbol and then selecting the widget (such as buttons, liner slider, and radial slider) in the drop-down list. Repeat this process until all the widgets are added to the Component.
- CSD tuning mode:** Select the **SmartSense (Full Auto-Tune)** tuning method if you want to automatically tune all the hardware and software parameters; otherwise, select the **SmartSense (Hardware parameters only)** option. Table 5-1 lists the parameters that are automatically set by SmartSense, depending on the mode selected.

**Note** Currently, SmartSense only supports widgets with **CSD (Self-cap)** Sensing mode. **CSX (Mutual-cap)** widgets must be tuned manually.

Table 5-1 CapSense Parameters Auto-tuned in SmartSense

Parameter	Calculation frequency	
	Full Auto-Tune Mode	Hardware Parameters Only Mode
Finger Threshold	Calculated after each sensor scan.	Manual selection
Noise Threshold		Calculated once on CapSense startup based on manually selected finger threshold
Hysteresis		
Negative Noise Threshold		
Scan Resolution	Calculated once on CapSense startup.	
Compensation IDAC		
Modulator IDAC		
Sense Clock Frequency		
Modulator Clock Frequency		
Low Baseline Reset		

- c. **Sensing mode:** Select the CapSense sensing mode - **CSD (Self-cap)** or **CSX (Mutual-cap)** for each widget based on your hardware design. SmartSense currently supports only CSD widgets and not CSX widgets. CSX widgets must be tuned manually.
- d. **Sensing element(s):** This field allows you to specify the number of sensors in each CSD widget. This option is useful in applications where there are multiple sensors that are identical, that is, have the same  $C_F$  and sensor area. The SmartSense method will tune all the sensors under a single widget for same sensitivity. Having multiple sensors under a single widget saves processing time and reduces memory footprint. You can configure the number of sensors depending on the hardware design.
- e. **Sensitivity:** This parameter sets the minimum expected signal (difference count) for 1 pF of finger capacitance  $C_F$ . This parameter can range from 1 to 2500. See [CapSense Fundamentals](#) for details on  $C_F$ . For example, a sensitivity value of 500 ensures a difference counts of 500 will be obtained when  $C_F = 1$  pF.

If you do not know the value of  $C_F$  ( $C_F$  can be estimated based on [Equation 2-1](#)), set the sensitivity as follows:

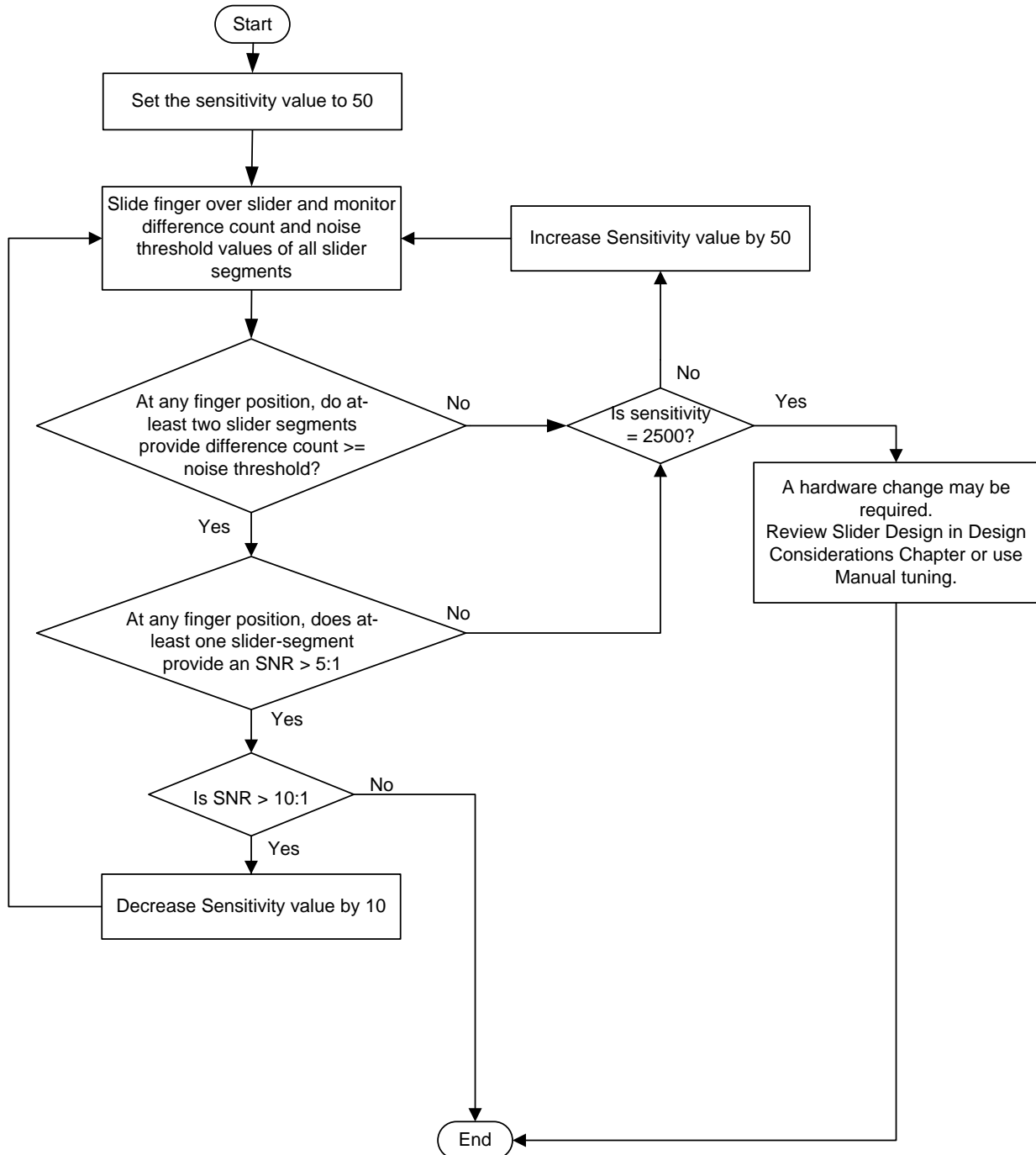
- For button sensors, set the sensitivity at 2500 and measure the SNR ([Signal-to-Noise Ratio](#)). If the SNR is less than 5:1, revise your PCB according to [PCB Layout Guidelines](#) or use [Manual Tuning](#). If the SNR for all or some sensors is greater than 5:1, check the sensor performance and set the sensitivity to an optimal value as follows:
  - If the sensor becomes active even before the finger touches it, decrease the sensitivity value by 50<sup>a</sup> counts and repeat the step until the sensor performance is optimum.
- For sliders, set the sensitivity to 50 initially. Slide your finger on the slider. If, at any slider position, at least two slider segments do not report a difference count value greater than or equal to the noise threshold value, increase the sensitivity value by 50 until the difference count value is greater than or equal to the noise threshold value.

If the SNR is not greater than 5:1 even after setting maximum sensitivity (2500), use [Manual Tuning](#) or revise the hardware according to [Slider Design](#) considerations. The following flow chart explains this process.

If the current sensitivity value results in an SNR of greater than 5:1 and less than 10:1 on at least one sensor on the slider, irrespective of where the finger is placed, use this sensitivity value. If SNR is greater than 10:1, reduce the sensitivity value in steps of 10 until the SNR fall in the range of 5:1 to 10:1. This helps in achieving optimum touch response and reducing device power consumption.

<sup>a</sup> This value is chosen to reduce the number of steps to achieve optimum value. You can increase or decrease the step size.

Figure 5-3. Setting Sensitivity Value for Sliders



#### 5.2.1.1.2 Advanced Settings

The Advanced tab allows you to specify the CapSense clocks and set threshold parameters. In SmartSense auto-tuning mode, most of these advanced parameters are automatically tuned by the algorithm; therefore, you do not need to set values for these parameters using the manual tuning process.

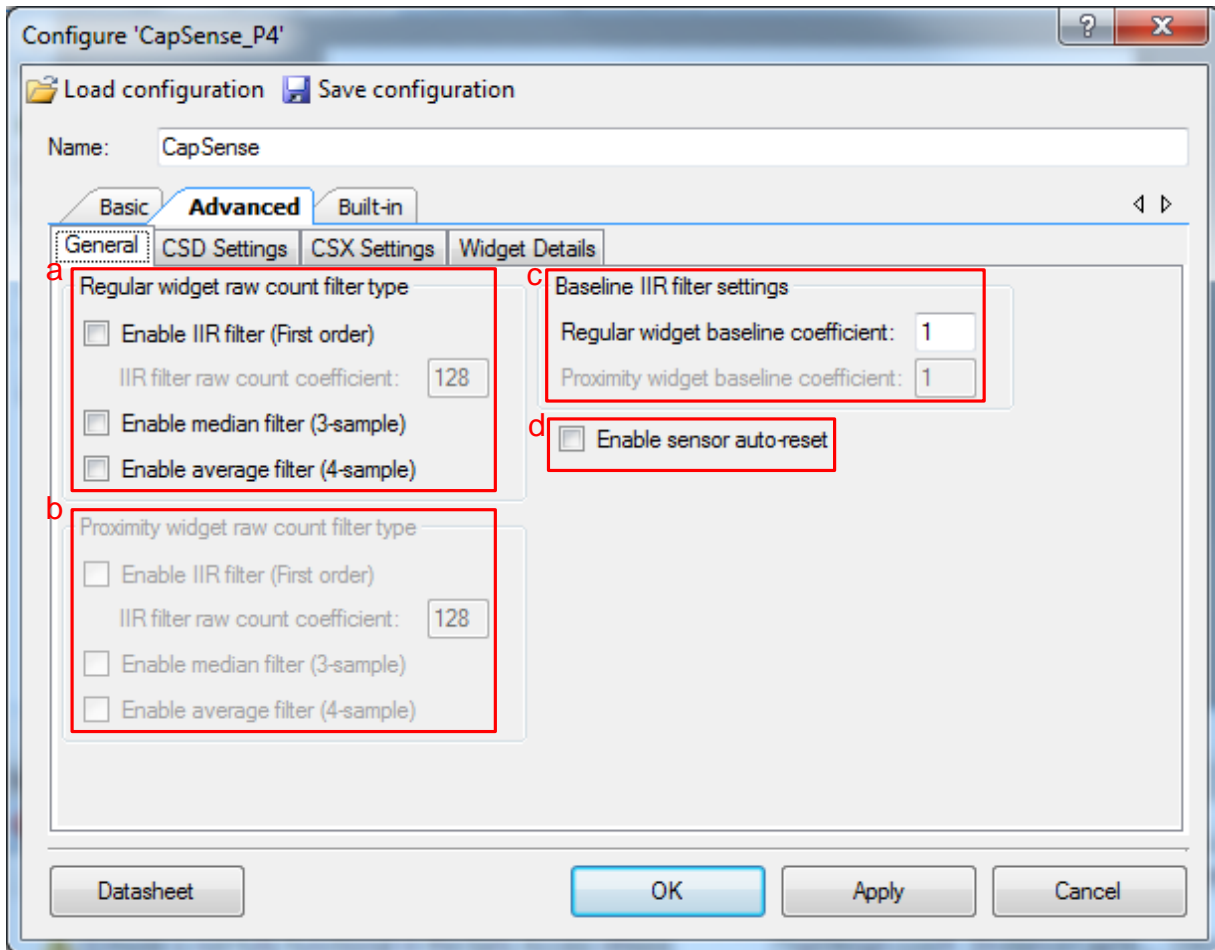
The Advanced tab has the following sub-tabs.

#### 5.2.1.1.2.1 General Tab

Figure 5-4 shows the General tab. This tab allows you to configure the following features:

- Enable and configure filters for all widgets
- Configure the baseline update rate
- Enable/disable sensor auto-reset feature

Figure 5-4. General Tab under Advanced Settings



- a. **Regular widget raw count filter type:** This parameter allows you to select a firmware filter for all widgets except the proximity widget to reduce the noise in raw counts. [Table 5-2](#) explains the available filters and their applications. Do not enable the filters in the beginning; when you use the Tuner to observe the noise in raw counts, select the appropriate filter according to [Table 5-2](#). If you choose to use an IIR filter, begin by selecting a filter with a higher value of filter coefficient and keep decreasing filter coefficient until you achieve an SNR greater than or equal to 5:1. CapSense 3.0 component allows user to enable more than one filter.
- b. **Proximity widget raw count filter type:** This parameters is similar to the **Regular widget rawcount filter type**, except that it applies only to proximity widgets. Proximity sensors have high noise when compared to button or slider sensors and hence require different settings for filters. Similar to the Regular widget rawcount filter type, you need not enable the filters in the beginning. When you observe the raw counts, you can enable the filter to achieve an SNR greater than or equal to 5:1
- c. **Baseline IIR filter settings:** This parameter controls the [Baseline](#) update rate. The baseline is an IIR-filtered version of raw count. The baseline coefficient value is the parameter 'N' in the IIR filter equation listed in [Table](#)

5-2. This parameter controls the rate at which the baseline is updated. You can start with a lower value of N (N=1) and keep increasing the value until the baseline tracks slow variations in raw counts.

- d. **Enable sensor auto-reset:** Enabling sensor auto-reset limits the maximum time duration for which the sensor stays ON (typically 5 to 10 seconds). This setting prevents the sensors from permanently turning ON when the raw count accidentally rises because of a large power supply voltage fluctuation, or a sudden change in noise conditions. Use this setting to avoid latch-up of sensors in high-noise conditions.

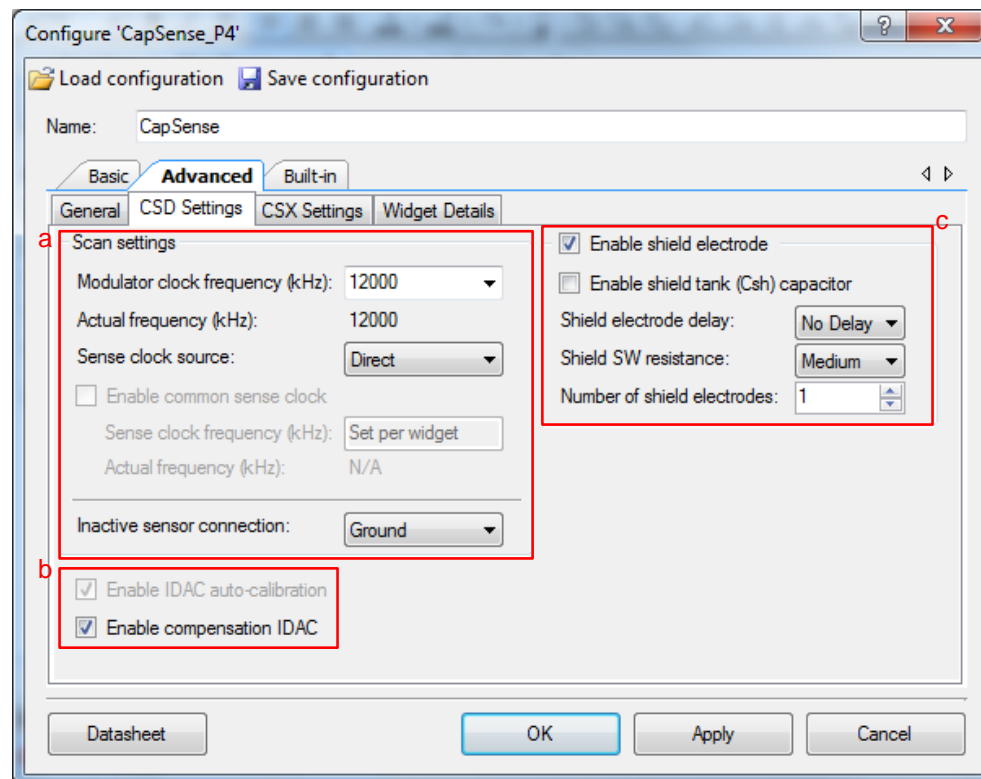
Table 5-2. Raw Data Noise Filters in CapSense v3.0

Filter	Description	Mathematical Description	Application
Median	Nonlinear filter that takes the three most recent samples and computes the median value.	$y[i] = \text{median}(x[i], x[i - 1], x[i - 2])$	Eliminates noise spikes from motors and switching power supplies
Average	Finite impulse response filter (no feedback) with equally weighted coefficients. It takes the four most recent samples and computes their average.	$y[i] = \frac{1}{4} * (x[i] + x[i - 1] + x[i - 2] + x[i - 3])$	Eliminates periodic noise (for example, from power supplies)
First order IIR	Infinite impulse response filter (feedback) with a step response similar to an RC low pass filter, thereby passing the low-frequency signals (finger touch responses). K value is fixed to 256. N is the <b>IIR filter rawcount coefficient</b> selectable in the <a href="#">General Tab under Advanced Settings</a> . A lower N value results in lower noise, but slows down the response.	$y[i] = \frac{1}{K} * \{N * x[i] + (K - N) * y[i - 1]\}$	Eliminates high-frequency noise.

#### 5.2.1.1.2.2 CSD Settings

The CSD Settings tab allows you to configure settings related to CSD widgets as shown in [Figure 5-5](#).

Figure 5-5. CSD Settings in CapSense Component



a. **Scan settings:** The scan settings parameter allows you to specify the following parameters:

- **Modulator clock frequency (kHz):** This parameter should be set to maximum value as it results in shorter scan time and hence helps in reducing average power consumption. Refer to the [Modulator Clock Related Parameters](#) section for details on the recommended values of this parameter.
- **Sense clock source:** This parameter is used to specify the source for the sense clock. The options available are Direct, 8-bit pseudo random sequence (PRS), 12-bit PRS, Auto, and SSCx (SSC – Spread Spectrum, applicable only for PSoC 4 S-Series). The “Auto” option automatically selects the length of the PRS sequence (for all PSoC 4 family of devices) or SSCx value (only for PSoC 4 S-Series). You should use PRS or SSCx if your design has strict electromagnetic compatibility requirements, as it reduces the electromagnetic emission. The other option is to select “Direct” because it provides higher sensitivity compared to PRS. Refer to the [Sense Clock Related Parameters](#) section for details on the clock source selection guidelines.
- **Inactive sensor connection:** CapSense scans one sensor at a time. This option determines the connection of sensors when they are not being scanned. The “Ground” option is recommended because it reduces noise on the scanned sensors. For liquid-tolerant designs, this option should be specified as “Shield”.

**Note** You should select the **Enable shield electrode** option to specify the “Inactive sensor connection” option to “Shield”.

b. **Enable compensation IDAC:** This parameter enables or disables compensation IDAC. Enabling the compensation IDAC selects the dual-IDAC mode of CSD operation. SmartSense with Dual-IDAC helps in reducing sensor scan time, which results in lower average device power consumption.

c. **Enable shield electrode:** This parameter enables or disables the shield electrode. You should enable the shield electrode if your board has a shield electrode for proximity sensing, liquid tolerance or to reduce  $C_P$  of sensors. See [Driven-Shield Signal and Shield Electrode](#) for details.

- **Enable shield tank capacitor:** Enable this option if you are using a  $C_{SH\_TANK}$  capacitor; see [CapSense CSD Shielding](#) for details.
- **Shield signal delay:** For proper operation of the shield electrode, the shield signal should match the sensor signal in phase. You can use an oscilloscope to view both sensor and shield signals to verify this condition. If they are not aligned, use this option to add delay to the shield signal to align the two signals. The available delays vary depending on the device selected.
- **Shield SW resistance:** This parameter controls the shield signal rise and fall times to reduce EMI. This parameter is valid only for PSoC 4 S-Series of devices. It is recommended to start with the “Low EMI” setting because it dynamically controls the slew rate of shield signal. If there is electromagnetic emissions due to shield switching, you can set this parameter to either “medium” or “high” value.
- **Number of shield electrode:** This parameter specifies the number of shield electrodes required in the design. Most designs works with one dedicated shield electrode; however, some designs require multiple dedicated shield electrodes for ease of PCB layout routing or to minimize the PCB real-estate used for the shield layer.

#### 5.2.1.1.2.3 CSX Settings

The CSX Settings tab allows you to configure settings related to the CSX widgets. Currently, the SmartSense algorithm does not support CSX widgets and CSX widgets must be tuned manually. Refer to the [Manual Tuning](#) section for manual tuning procedure.

#### 5.2.1.1.2.4 Widget Details

The Widget Details tab allows you to configure the thresholds ([Figure 5-6](#)) and specify the ganged sensor elements ([Figure 5-7](#)). If the **CSD tuning mode** is selected as **SmartSense (Full Auto-Tune)** in the [CapSense Component Basic Tab](#), thresholds are auto-tuned by the CapSense component and hence appear as **Set by SmartSense** in the Widget Details tab.

The recommended values for thresholds when the **CSD tuning method** is **SmartSense (Hardware parameters only)** is specified in [Selecting CapSense Software Parameters](#).

Figure 5-6. Widget Details Tab Settings

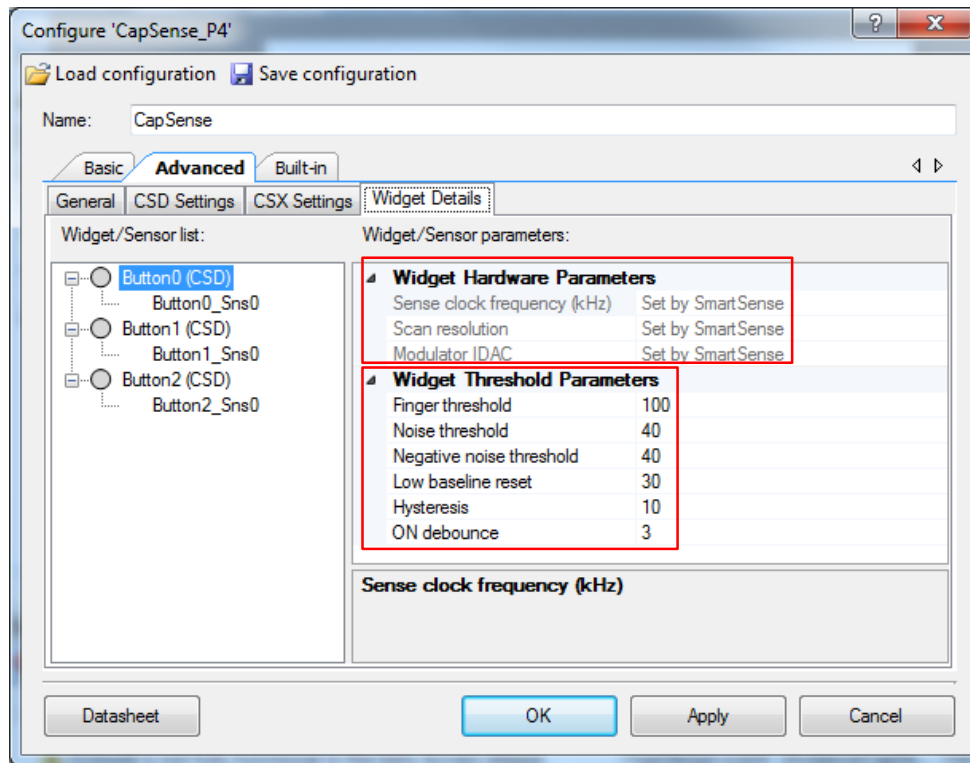
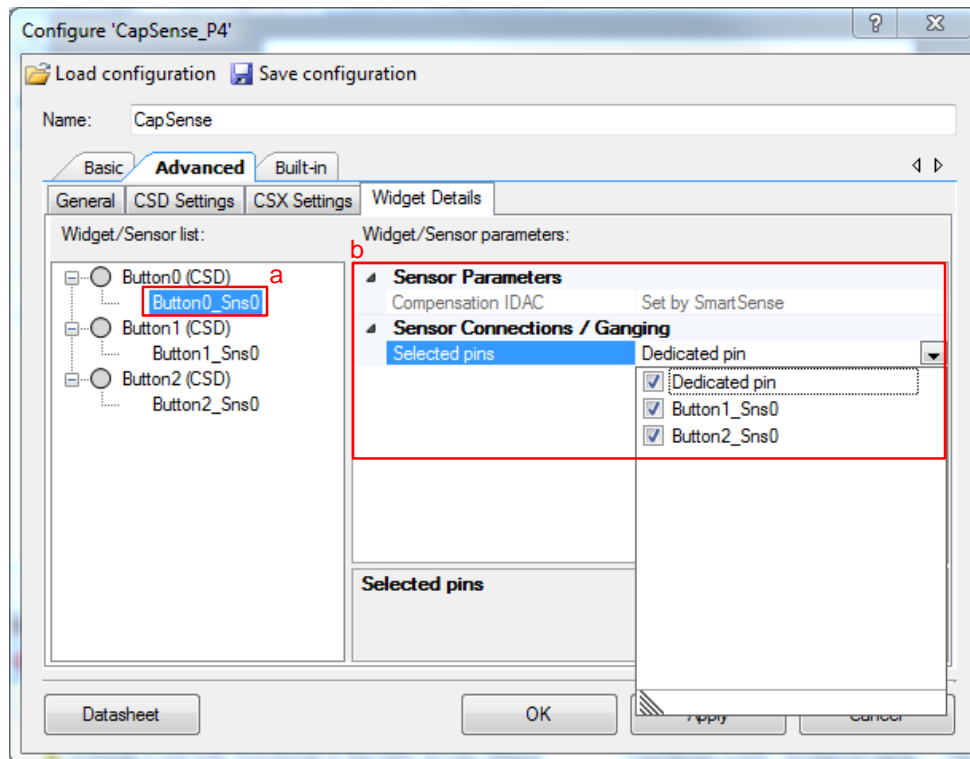


Figure 5-7. Specifying Ganged Sensor Elements in SmartSense Mode

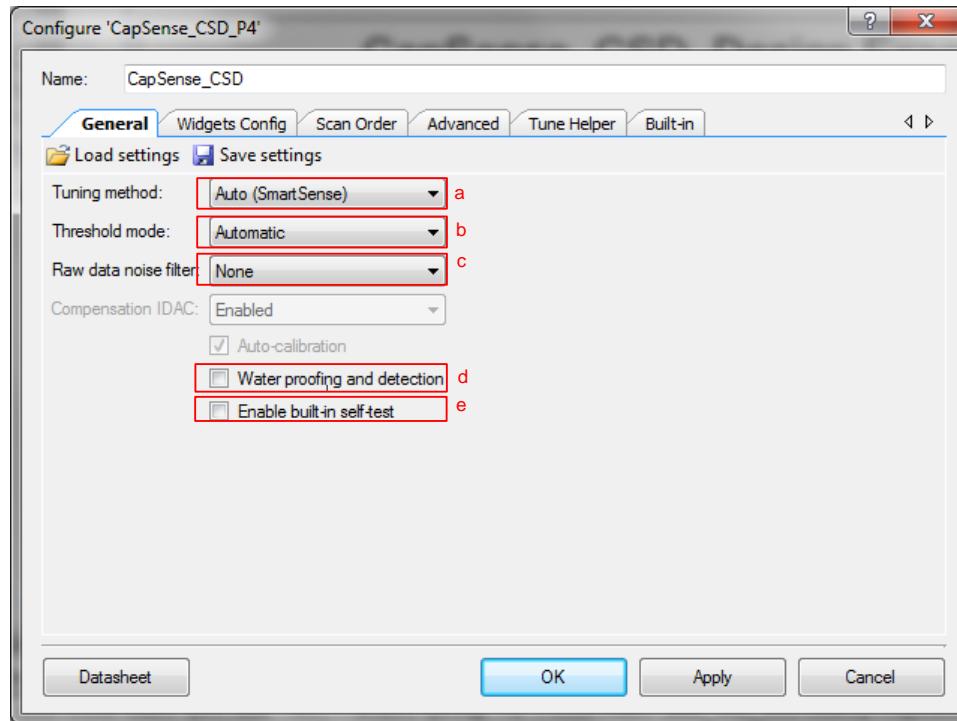




### 5.2.1.2 SmartSense Configuration for CapSense v2.40 and Earlier

To open the CapSense Component configuration window (Figure 5-8), double-click the Component or right-click the Component and select “Configure”.

Figure 5-8. CapSense Component General Tab



#### 5.2.1.2.1 General Settings

Set the General tab configurations according to Figure 5-8 and as follows:

- Tuning method:** Select the **Auto (SmartSense)** tuning method. This method automatically tunes the CapSense Component parameters mentioned in Table 5-3 based on the sensors'  $C_p$ .

Table 5-3 CapSense Parameters Auto-tuned in SmartSense Tuning Method

Parameter	Calculation Frequency	
	Automatic Threshold Mode	Flexible Threshold mode
Finger Threshold	Calculated for each sensor scan.	Manual selection
Noise Threshold		Calculated once on CapSense startup based on manually selected finger threshold
Hysteresis		
Negative Noise Threshold		
Scan Resolution	Calculated once on CapSense startup.	
Compensation IDAC		
Modulation IDAC		
Sense Clock Divider		
Modulator Clock Divider		
Low Baseline Reset		

- Threshold mode:** This can be chosen to be “Automatic” or “Flexible”. Automatic thresholds imply that all thresholds will be calibrated automatically during runtime i.e. allows you to use SmartSense in **Full (auto-tune)** mode. “Flexible” thresholds allow manual selection of **finger threshold** in the **Widget Configuration** tab of the



CapSense configuration window i.e. allows you to use SmartSense in [Hardware parameters only](#) mode. Selecting the “Automatic” option is recommended and works for most designs. However, as explained [above](#), if you need strict control over [finger threshold](#), or want to reduce power consumption in the SmartSense mode, you can select the “flexible” option.

- c. **Raw data noise filter:** This parameter allows you to select a firmware filter to reduce the noise in raw counts. [Table 5-4](#) explains the available filters and their applications. Select the option “None” in the beginning. Later, when you use the tuner to observe the noise in raw counts, select the appropriate filter according to [Table 5-4](#). If you chose to use an IIR filter, begin by selecting a filter with a lower value of k and keep increasing k until you achieve a 5:1 SNR.
- d. **Waterproofing and detection:** Enable this option if you are using a shield electrode or guard sensor for liquid tolerance. Disable otherwise. See the [Liquid Tolerance](#) section for details.
- e. **Enable built-in self-test:** Enable this option if you are using the Component APIs to perform a built-in self-test of CapSense. See the [Component Datasheet](#) for more details on the available built-in self-test APIs. Disabling this option reduces the flash memory usage.

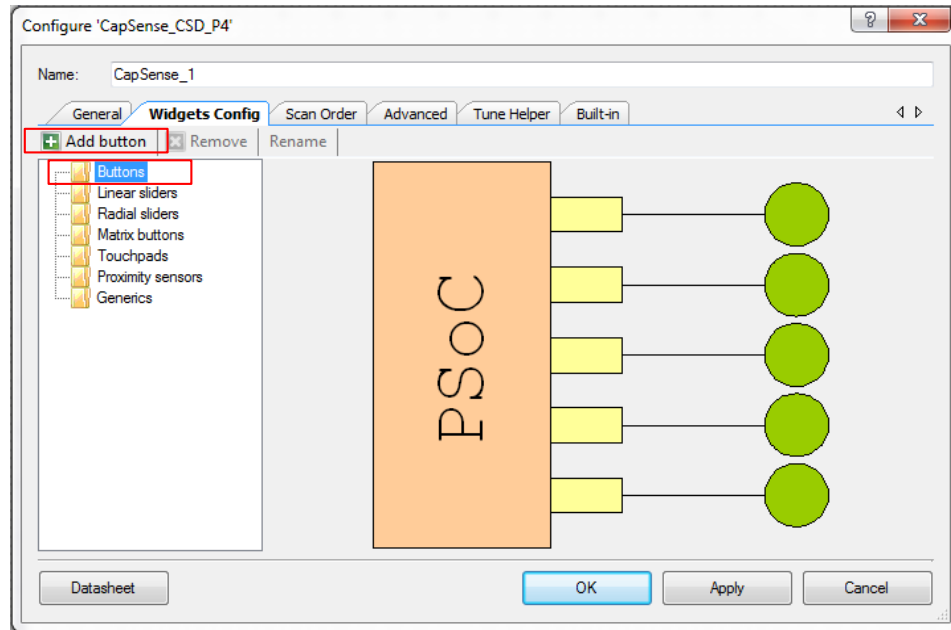
Table 5-4 Raw Data Noise Filters in CapSense v2.40

Filter	Description	Mathematical Description	Application
<b>Median</b>	Nonlinear filter that takes the three most recent samples and computes the median value.	$y[i] = \text{median}(x[i], x[i - 1], x[i - 2])$	Eliminates noise spikes from motors and switching power supplies
<b>Average</b>	Finite impulse response filter (no feedback) with equally weighted coefficients. It takes the three most recent samples and computes their average.	$y[i] = \frac{1}{3} * (x[i] + x[i - 1] + x[i - 2])$	Eliminates periodic noise (for example, from power supplies)
<b>First order IIR 1/k</b>	Infinite impulse response filter (feedback) with a step response similar to an RC low pass filter, thereby passing the low-frequency signals (finger touch responses). A higher k-value results in lower noise, but slows down the response	$y[i] = \frac{1}{k} * \{x[i] + (k - 1)y[i - 1]\}$	Eliminates high-frequency noise.
<b>Jitter</b>	Jitter filter eliminates noise by comparing the present input value with its previous output value. If the difference is greater than $\pm 1$ , then the output is changed by $\pm 1$ .	$y[i] = \begin{cases} x[i] - 1, & x[i] - y[i - 1] > 1 \\ x[i] + 1, & x[i] - y[i - 1] < -1 \\ y[i - 1], & \text{otherwise} \end{cases}$	Noise due to thick overlay. Especially useful for slider centroid data.

### 5.2.1.2.2 Widget Configuration

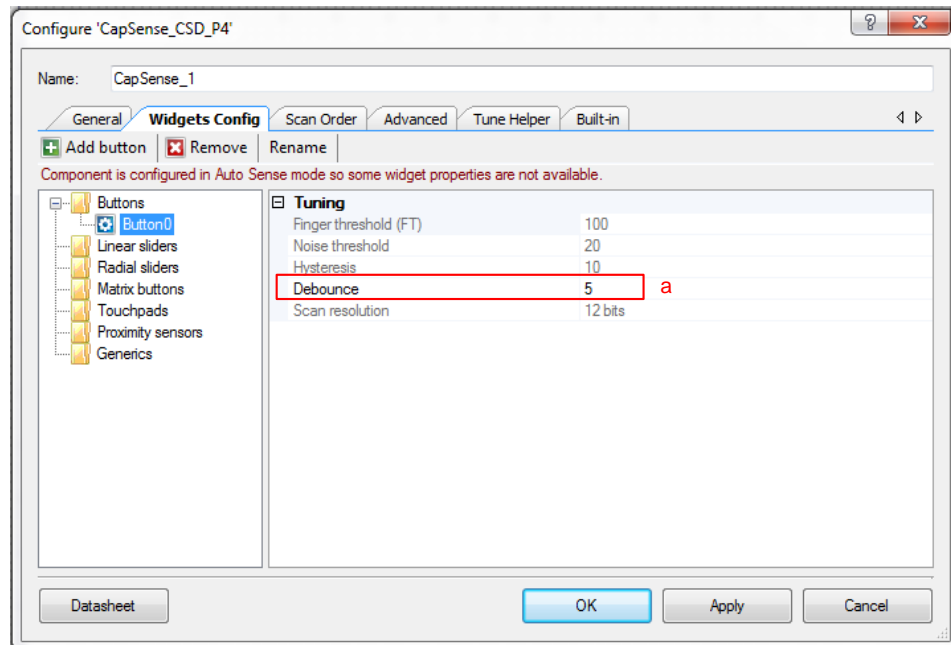
The “Widgets Config” tab allows you to add and configure CapSense widgets, as [Figure 5-9](#) shows. See [CapSense Widgets](#) for details on each widget type. To add a widget, select the type of widget and click “Add”.

Figure 5-9. Adding Widgets



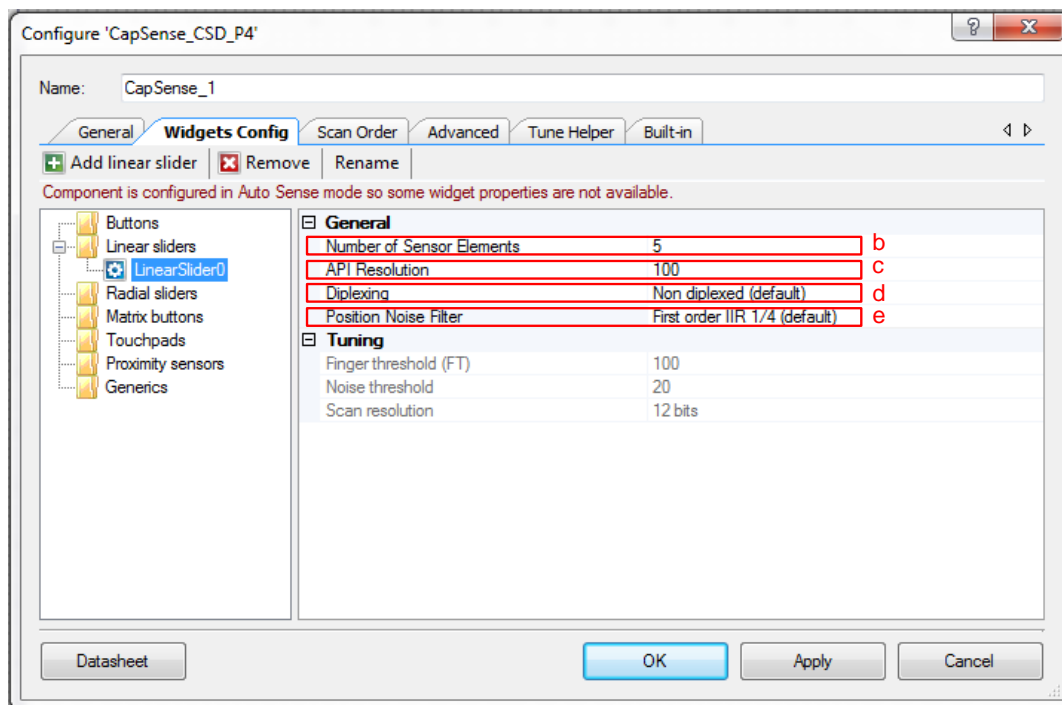
Click a widget to configure its parameters. SmartSense automatically sets some of the widget parameters. These parameters are grayed out in the configuration window, as [Figure 5-10](#) shows. See also [Figure 5-11](#) and [Figure 5-12](#).

Figure 5-10. Configuration of a Button



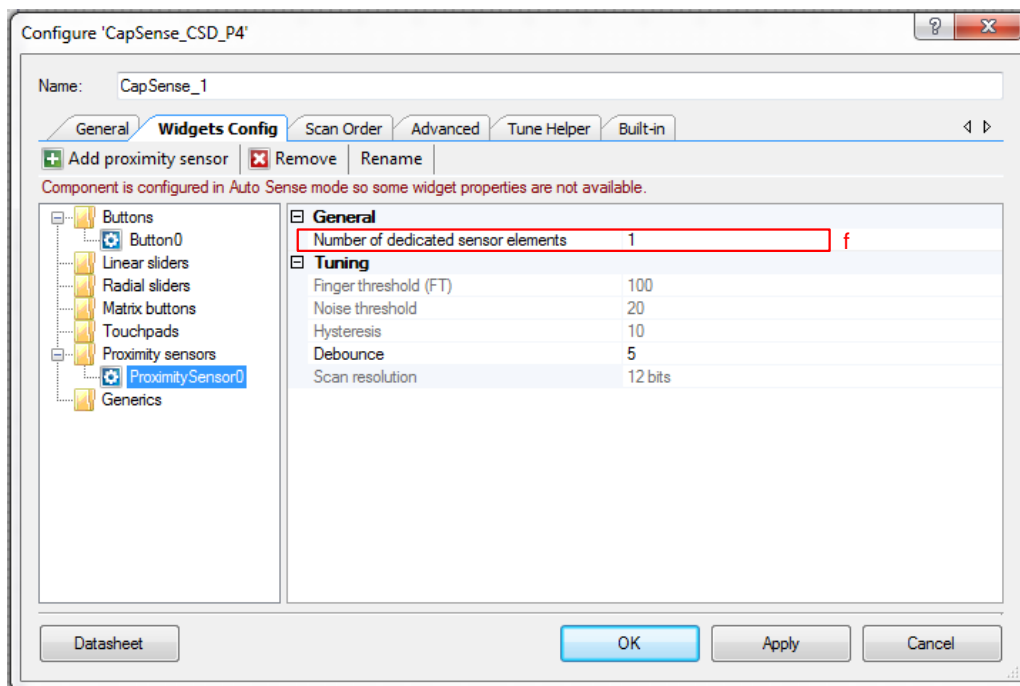
- a. **Debounce:** (see [Figure 5-10](#); available for buttons, matrix buttons, and proximity sensors). This parameter selects the number of consecutive CapSense scans during which a sensor must be active to generate an ON state from the Component. Debounce ensures that high-frequency, high-amplitude noise does not cause false detection. You can increase the debounce value if the CapSense detects false touches. If not, decrease the debounce value to improve the response time. For more details, see [Debounce](#) in Manual Tuning section.
- b. **Number of Sensor Elements:** (see [Figure 5-11](#); available for Linear Sliders, Radial Sliders, and Touchpads). This parameter defines the number of segments within the slider (See [Sliders](#) for details). The number of sensor elements depends on the required API resolution of the slider. A good ratio of API resolution to sensor elements is 20:1. Increasing the ratio of API resolution to sensor elements beyond 20:1 may result in noisy finger position.
- c. **API resolution:** (see [Figure 5-11](#); available for Linear Sliders, Radial Sliders, and Touchpads). This parameter defines the number of discrete finger positions that a slider or a touchpad must resolve. This parameter is different from the resolution of the [Sigma Delta converter](#). A good ratio of API resolution to the number of slider segments is 20:1. Increasing the ratio of API resolution to sensor elements beyond 20:1 can result in increased noise on the calculated finger position.  
For example, if your design has five slider segments, set the API resolution to any value between 1 and 100 according to your application requirement.
- d. **Diplexing:** (see [Figure 5-11](#); available for Linear Sliders and Radial Sliders). Use diplexing to reduce the number of GPIOs required. Diplexing allows a design to have two slider segments per GPIO pin. For example, a diplexed 16-segment slider requires only eight GPIO pins. See the CapSense [Component Datasheet](#) for details.
- e. **Position Noise filter:** (see [Figure 5-11](#); available for Linear Sliders, Radial Sliders, and Touchpads). This parameter selects the type of firmware noise filter used to remove noise from the calculated finger position. The available filters and their applications are explained in [Table 5-4](#).

Figure 5-11. Configuration of a Slider



- f. **Number of dedicated sensor elements:** (see [Figure 5-12](#); available for proximity sensors.) Select '1' if you are using a dedicated proximity sensor. Set to '0' if you are ganging other sensors together to form a proximity sensor. See [Proximity Sensor](#) for details.

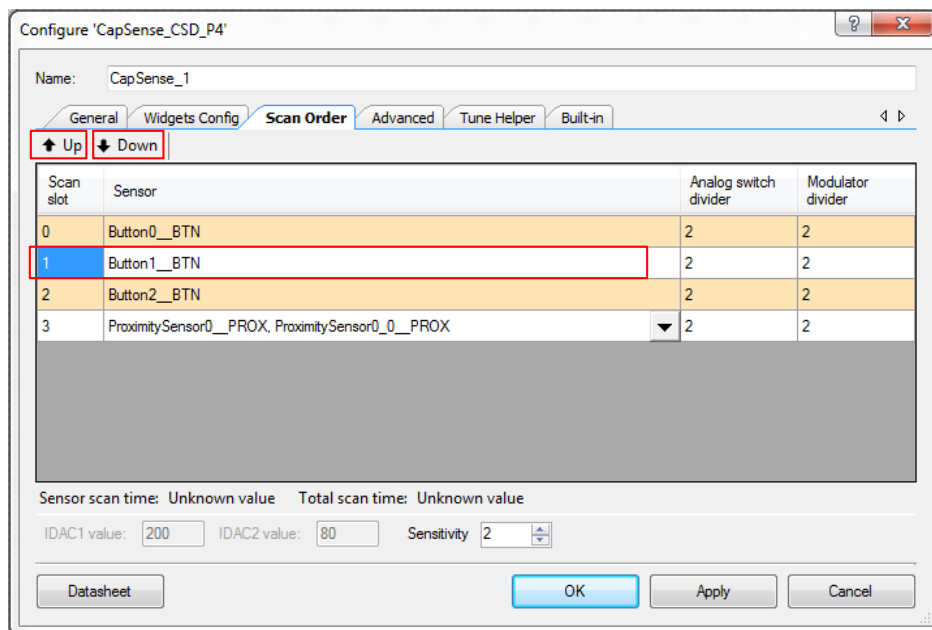
Figure 5-12. Configuration of a Proximity Sensor



### 5.2.1.2.3 Scan Order

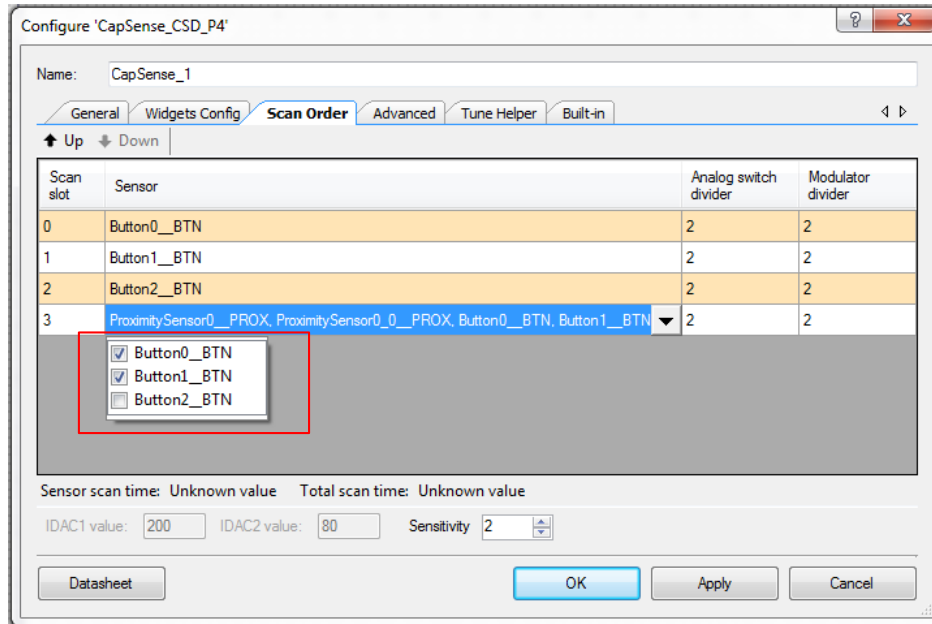
Generally, CapSense performance is not impacted by the order in which the sensors are scanned. However, if you want to change scan order, you can use this tab, as [Figure 5-13](#) shows. To change the scan order, select a sensor and click "Up" or "Down".

Figure 5-13. Scan Order



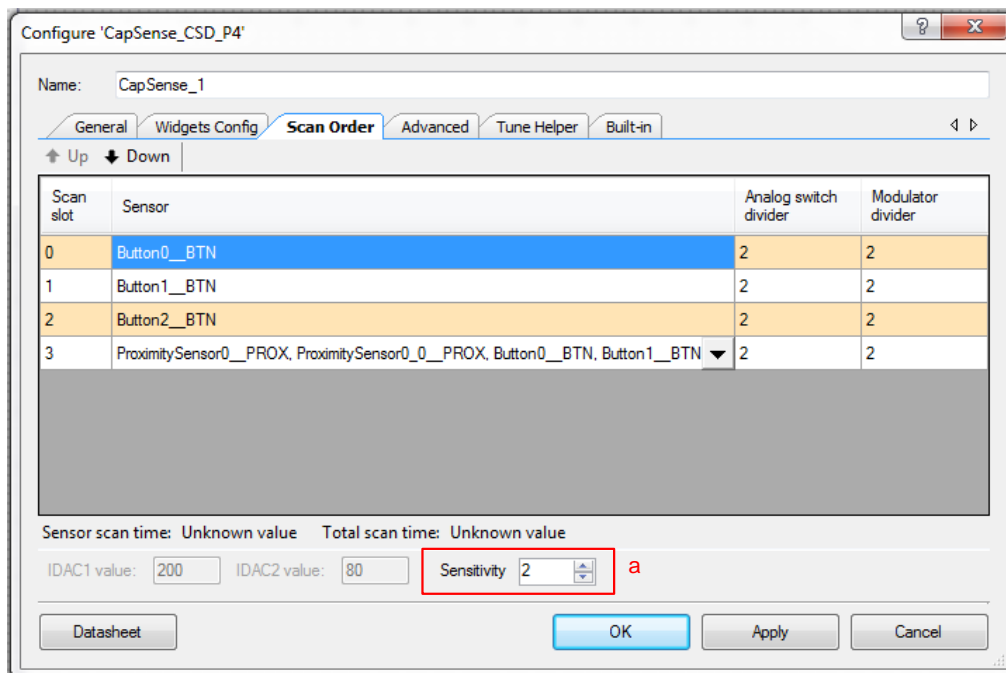
You can combine other sensors into a proximity sensor by using the drop-down menu on the Scan Order tab, as [Figure 5-14](#) shows. Combining other sensors with the proximity sensor does not affect their operation, however, the resulting proximity sensor may have a high  $C_P$  and hence a high scan time. Therefore, a proximity sensor should be scanned at a lower rate than the other sensors to avoid long scanning intervals. Proximity sensors are disabled by default. You must enable them using firmware before scanning them. Refer to the [Component Datasheet](#) for details.

Figure 5-14. Combining Sensors to Form a Proximity Sensor



Click on a sensor to change its sensitivity, as [Figure 5-15](#) shows.

Figure 5-15. Sensitivity



- a. **Sensitivity:** This parameter sets the minimum expected value of finger capacitance  $C_F$ . See [CapSense Fundamentals](#) for details on  $C_F$ . SmartSense multiplies the sensitivity setting by 0.1 pF to get  $C_F$ . For example, a sensitivity value of 1 represents  $C_F = 0.1$  pF and a setting of 4 represents  $C_F = 0.4$  pF.

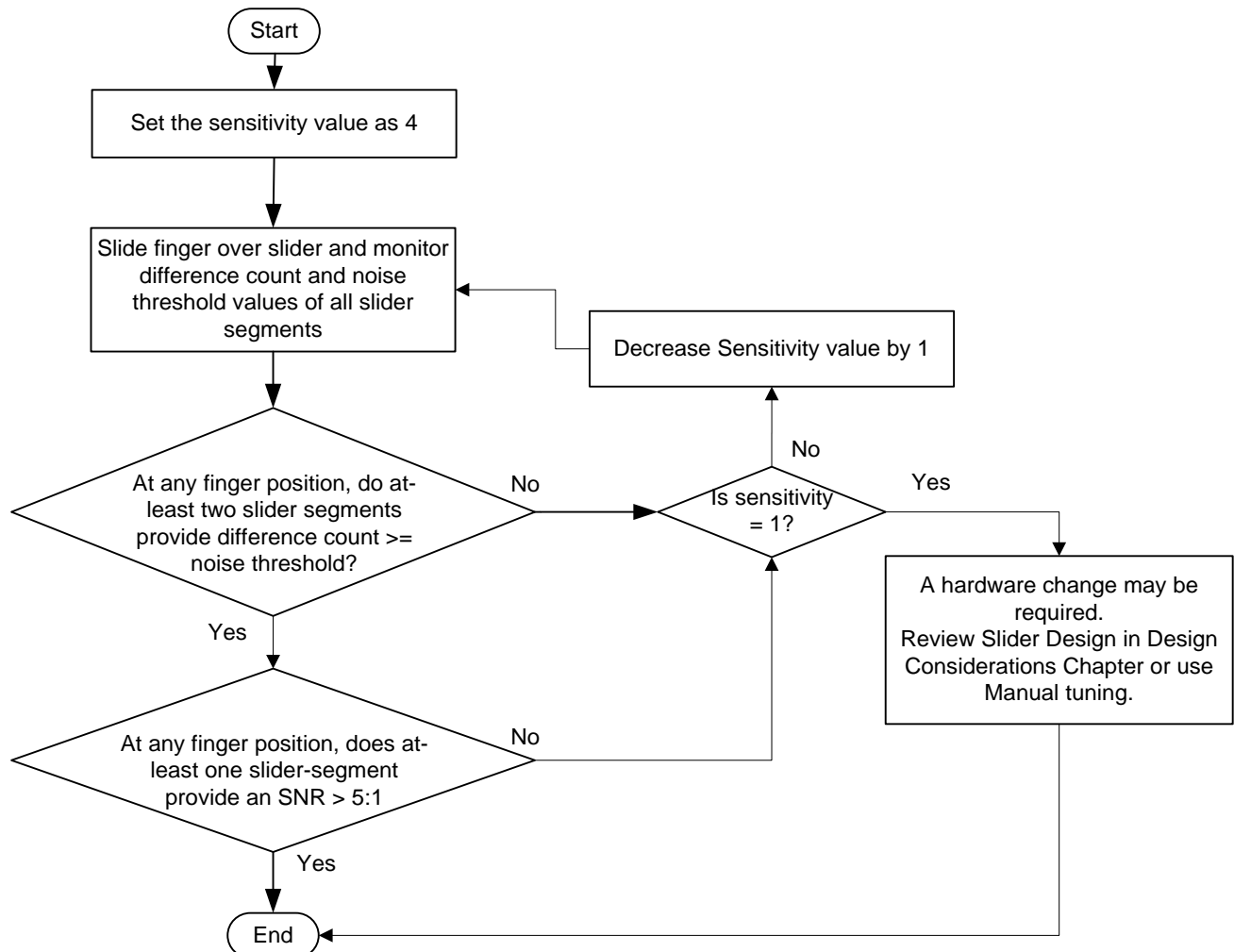
Equation 5-1: Sensitivity and Finger Capacitance

$$C_F = 0.1 \text{ pF} \times \text{sensitivity}$$

If you do not know the value of  $C_F$  ( $C_F$  can be estimated based on [Equation 2-1](#)), set the sensitivity as follows:

- For button sensors, set the sensitivity at 1 and measure the SNR ([Signal-to-Noise Ratio](#)). If the SNR is less than 5:1, revise your PCB according to [PCB Layout Guidelines](#) or use. If the SNR for all sensors is greater than 5:1, check the sensor performance and set the sensitivity to an optimal value as follows:
  - If the sensor becomes active even before the finger touches it, increase the sensitivity value.
- For sliders, set the sensitivity at 4. Slide your finger on the slider. If, at any slider position, at-least two slider segments do not report a difference count value greater than or equal to the noise threshold value, decrease the sensitivity value. If the current sensitivity value results in an SNR of 5:1 on at least one sensor on the slider, irrespective of where the finger is placed, use this sensitivity value. If none of the sensitivity values meet both the above conditions, use [Manual Tuning](#) or revise the hardware according to [Slider Design](#) considerations. The following flowchart explains this process.

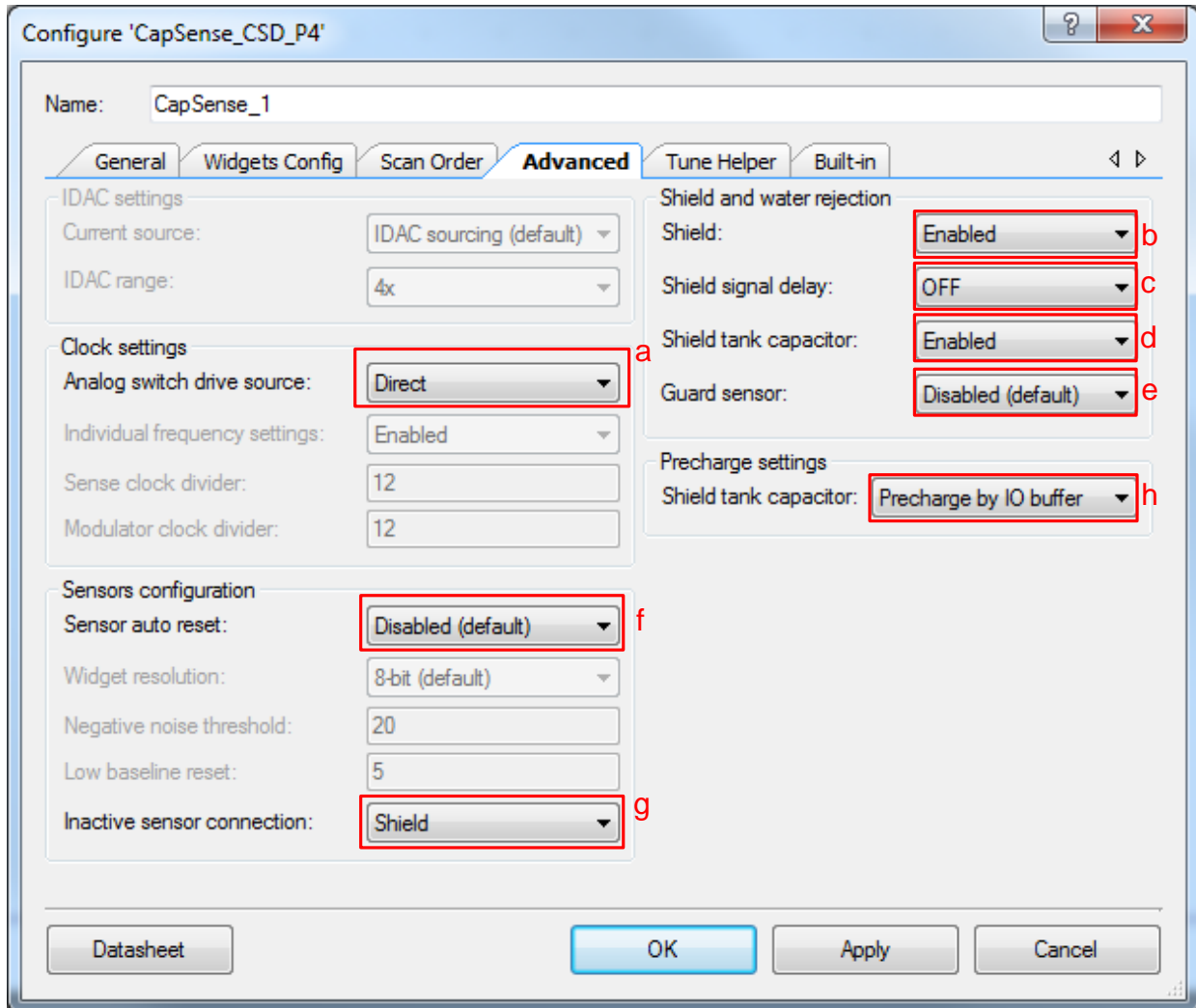
Figure 5-16. Setting Sensitivity Value for Sliders



#### 5.2.1.2.4 Advanced Settings

These settings are available on the Advanced tab, as Figure 5-17 shows.

Figure 5-17. Advanced Settings



Configure 'CapSense\_CSD\_P4'

Name: CapSense\_1

General Widgets Config Scan Order **Advanced** Tune Helper Built-in

IDAC settings

Current source: IDAC sourcing (default)

IDAC range: 4x

Clock settings

Analog switch drive source: Direct

Individual frequency settings: Enabled

Sense clock divider: 12

Modulator clock divider: 12

Sensors configuration

Sensor auto reset: Disabled (default)

Widget resolution: 8-bit (default)

Negative noise threshold: 20

Low baseline reset: 5

Inactive sensor connection: Shield

Shield and water rejection

Shield: Enabled

Shield signal delay: OFF

Shield tank capacitor: Enabled

Guard sensor: Disabled (default)

Precharge settings

Shield tank capacitor: Precharge by IO buffer

Datasheet OK Apply Cancel

- Analog switch drive source:** The options available are Direct, 8-bit pseudo random sequence (PRS), 12-bit PRS, and PRS Auto. The PRS Auto option automatically selects the length of the sequence. You should use PRS if your design has strict [electromagnetic compatibility](#) requirements, as it reduces the electromagnetic emission. The other option is to select "Direct" because it provides higher sensitivity compared to PRS. See [CapSense Clock Generator](#) for details.
- Shield:** You should enable the shield electrode if your board has a shield electrode to reduce  $C_P$  of sensors or for proximity sensing or liquid tolerance. See [Driven-Shield Signal and Shield Electrode](#) for details.
- Shield signal delay:** For proper operation of the shield electrode, the shield signal should match the sensor signal in phase. You can use an oscilloscope to view both sensor and shield signals to verify this condition. If they are not aligned, use this option to add delay to the shield signal to align the two signals. The available delays are 50 ns and 10 ns.
- Shield tank capacitor enable:** Enable this option if you are using a  $C_{SH\_TANK}$  capacitor; see [CapSense CSD Shielding](#) for details.
- Guard Sensor:** A guard sensor is used for liquid tolerance. See [Guard Sensor](#) for details. Enable this option if you are using a guard sensor in your design.

- f. **Sensor auto reset:** Enabling sensor auto reset limits the maximum time duration for which the sensor stays ON (typically 5 to 10 seconds). This setting prevents the sensors from permanently turning ON when the raw count accidentally rises because of a large power supply voltage fluctuation, or a sudden change in noise conditions. Use this setting to avoid latch-up of sensors in high-noise conditions.
- g. **Inactive sensor connection:** CapSense scans one sensor at a time. This option determines the connection of sensors when they are not being scanned. The “Ground” option is recommended because it reduces noise on the scanned sensors. For liquid-tolerant designs, this option should be specified as “Shield”.  
**Note** You should enable the “Shield” option to specify the “Inactive sensor connection” option to “Shield”.
- h. **Shield tank capacitor:** This option selects the precharge configuration of  $C_{SH\_TANK}$ . It is recommended to select “Precharge by IO buffer”. See [CapSense CSD Shielding](#) for details.

## 5.3 Manual Tuning

### 5.3.1 Overview

Cypress SmartSense technology allows a device to calibrate itself for optimal performance and complete the entire tuning process automatically. This technology will meet the needs of most designs, but in cases where SmartSense does not work or there are specific SNR or power requirements, the CapSense parameters can be adjusted to meet system requirements. This is called manual tuning.

Some advantages of manual tuning, as opposed to [SmartSense](#) auto-tuning are:

- Strict control over parameter settings: SmartSense sets all of the parameters automatically. However, there may be situations where you need to have strict control over the parameters. For example, use manual tuning if you need to strictly control the time PSoC 4 and PProC BLE takes to scan a group of sensors or strictly control the sense clock frequency of each sensor (this can be done to reduce EMI in systems).
- Supports higher parasitic capacitances: SmartSense supports parasitic capacitances as high as 45 pF for a 0.2-pF finger capacitance, and as high as 35 pF for a 0.1-pF finger capacitance. If the parasitic capacitance is higher than the value supported by SmartSense, you should use manual tuning.

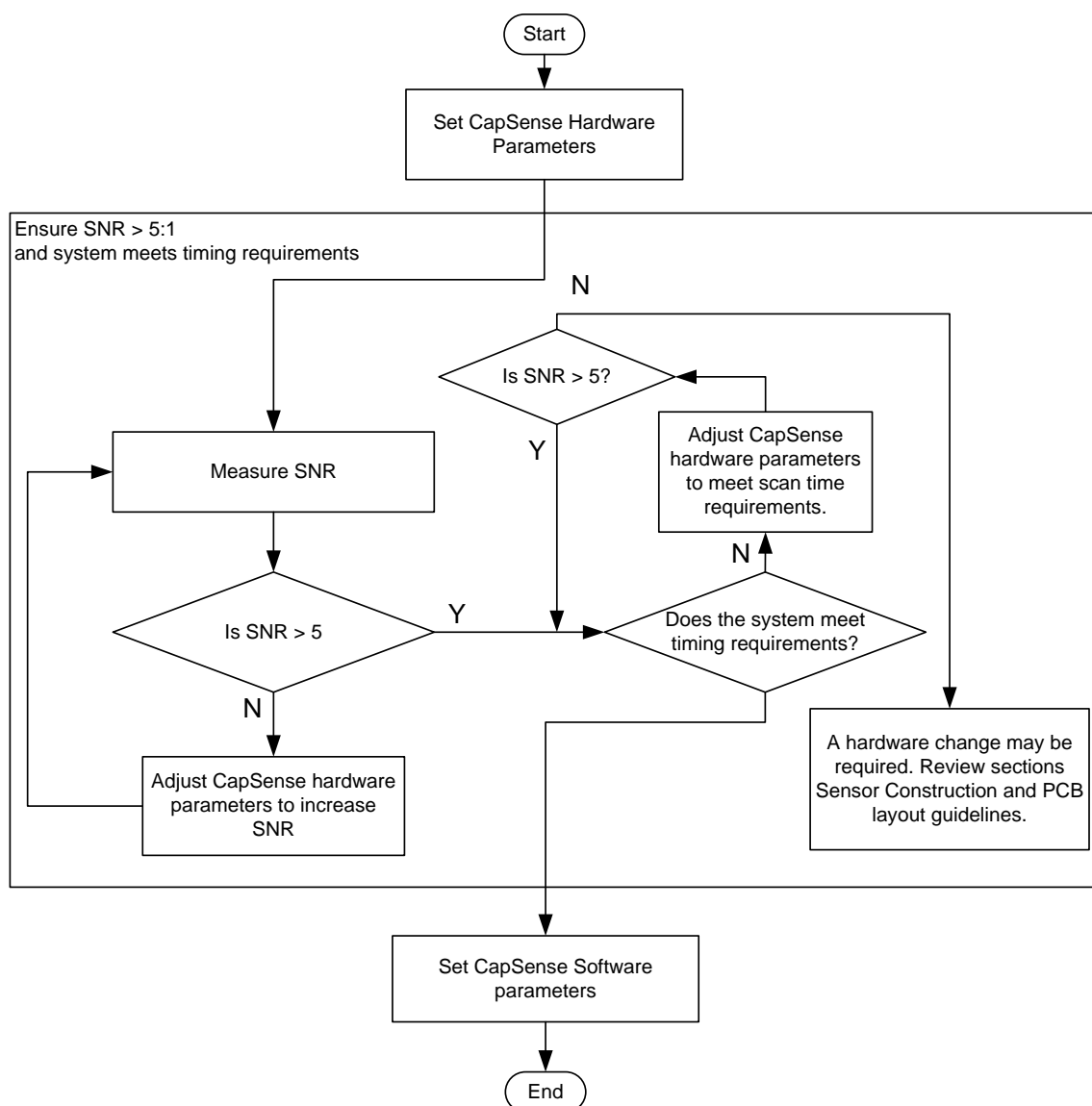
The manual tuning process can be summarized in the following three steps and is shown in [Figure 5-18](#).

1. Set initial values of [CapSense Component Hardware Parameters](#)
2. Tune CapSense component hardware parameters to ensure [Signal-to-Noise Ratio](#) is greater than 5:1 and the system meets timing requirements.
3. Set optimum values of [CapSense Component Software Parameters](#) (threshold parameters)

The following sections describe the fundamentals of manual tuning and the above three steps in detail. Knowledge of the PSoC 4 and PProC BLE CapSense architecture is a prerequisite for these sections. See [Capacitive Touch Sensing Method](#) and [CapSense Sensing](#) to become familiar with PSoC 4 and PProC BLE CapSense architecture. Depending upon the sensing method selected, the manual tuning procedure will differ. Refer to [CSD sensing method](#) section for manual tuning using CSD for self-capacitance and refer to [CSX sensing method](#) section for manual tuning using CSX for mutual-capacitance. You can skip these sections if you are not planning to use manual tuning in your design.



Figure 5-18. Manual Tuning Process Overview



### 5.3.2 CSD sensing method

This section explains the basics of manual tuning using CSD sensing method. It also explains the hardware and software parameters that influence a manual tuning procedure. The section ends with three examples on manual tuning of button, slider and proximity widgets.

### 5.3.2.1 Basics

#### 5.3.2.1.1 Conversion Gain and CapSense Signal

Conversion gain will influence how much signal the system sees. If there is more gain, the signal is higher and a higher signal means a higher achievable [Signal-to-Noise Ratio](#).

#### 5.3.2.1.1.1 Conversion Gain in Single IDAC Mode

In the **single IDAC mode**, the raw count is directly proportional to the sensor capacitance.

Equation 5-2: Raw Count Relationship to Sensor Capacitance

$$\text{raw count} = G_C C_S$$

Where  $G_C$  is the capacitance to digital conversion gain of CapSense CSD. The approximate value of this conversion gain using the recommended IDAC Sourcing mode, according to [Equation 3-9](#) and [Equation 5-2](#) is:

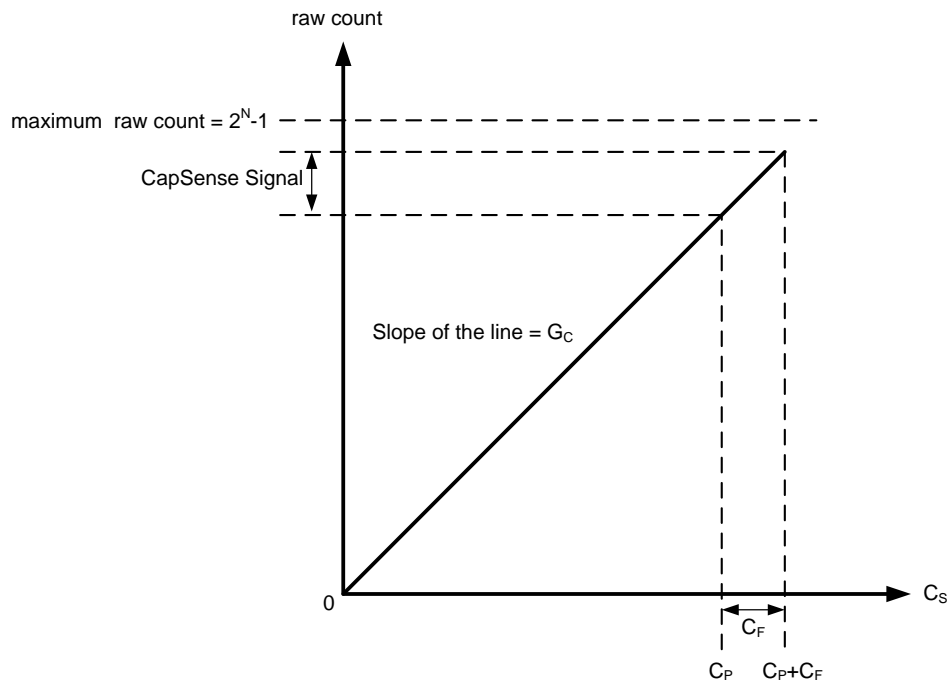
Equation 5-3: Capacitance to Digital Converter Gain

$$G_C = (2^N - 1) \frac{V_{REF} F_{SW}}{I_{MOD}}$$

The value of  $V_{REF}$  is fixed and is equal to 1.2 volts for all the PSoC 4 family of devices except the PSoC 4 S-Series. For the PSoC 4 S-Series,  $V_{REF}$  can vary from 0.6 V to  $V_{DDA} - 0.6$  V. The tunable parameters of the conversion gain are  $V_{REF}$  (for PSoC 4 S-Series only),  $F_{SW}$ ,  $I_{MOD}$ , and  $N$ . [Figure 5-19](#) shows a plot of raw count versus sensor capacitance.

**Note** The CapSense Component automatically select the  $V_{REF}$  value depending on the  $V_{DDA}$  voltage.

Figure 5-19. Raw Count versus Sensor Capacitance



The change in raw counts when a finger is placed on the sensor is called CapSense signal. [Figure 5-20](#) shows how the value of the signal changes with respect to the conversion gain.

Figure 5-20. Signal Values for Different Conversion Gains

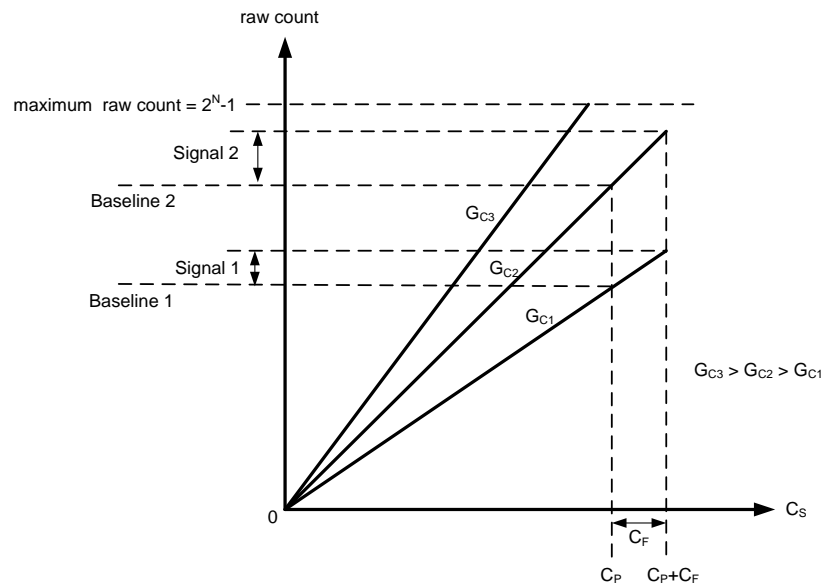
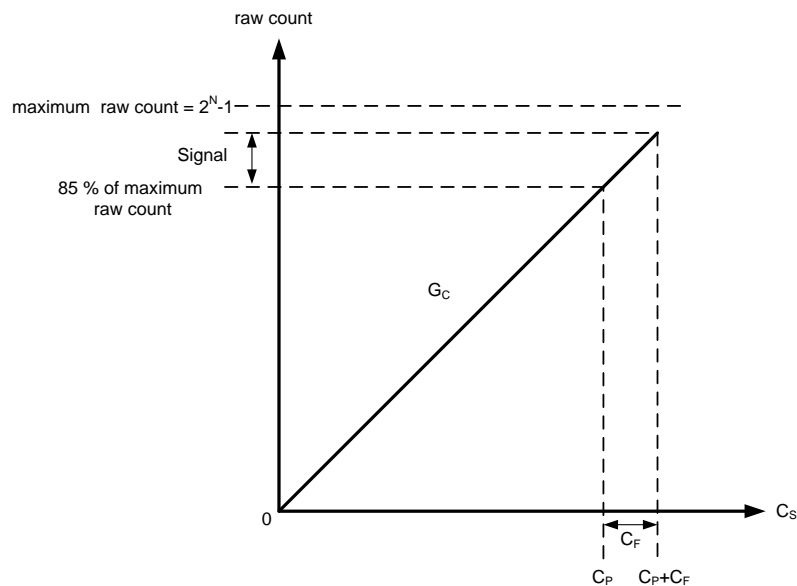


Figure 5-20 shows three plots corresponding to three conversion gain values  $G_{C3}$ ,  $G_{C2}$ , and  $G_{C1}$ . An increase in the conversion gain results in higher signal value. However, this increase in the conversion gain also moves the raw count corresponding to  $C_P$  towards the maximum value of raw count ( $2^N-1$ ). For very high gain values, the raw count saturates as the plot of  $G_{C3}$  shows. Therefore, you should tune the conversion gain to get a good signal value while avoiding saturation of raw count. Tune the gain in such a way that the raw count corresponding to  $C_P$  is 85 percent of the maximum raw count, as Figure 5-21 shows. This ensures maximum gain, with enough margin for the raw count to grow because of environmental changes, and not saturate on finger touches.

Figure 5-21. Recommended Tuning



### 5.3.2.1.1.2 Conversion Gain in Dual IDAC Mode

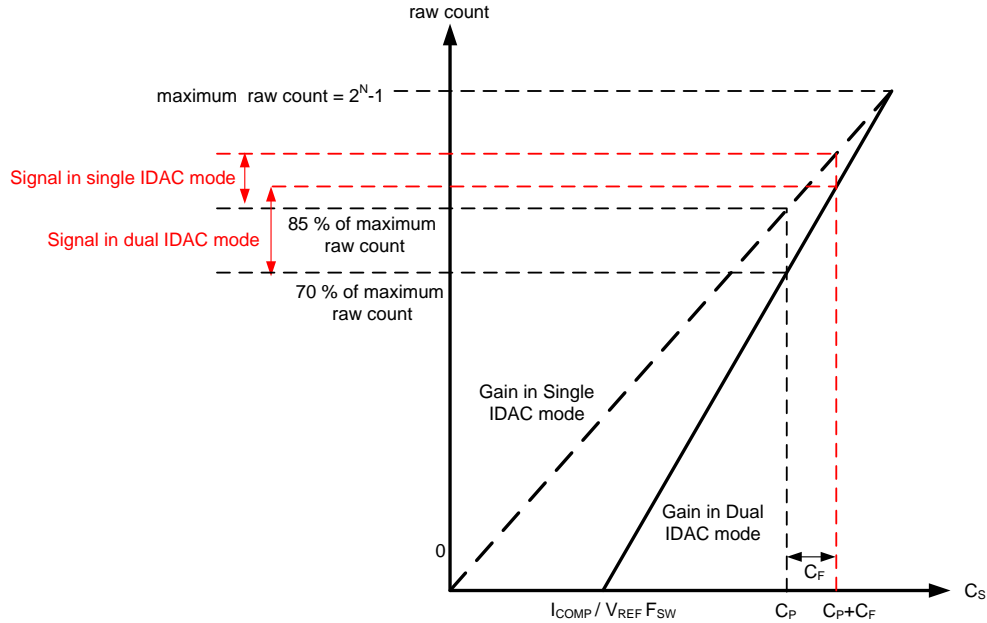
The equation for raw count in the [dual IDAC mode](#), according to [Equation 5-3](#) and [Equation 3-11](#) is:

Equation 5-4. Dual IDAC Mode Raw Counts

$$\text{raw count} = G_C C_S - (2^N - 1) \frac{I_{\text{COMP}}}{I_{\text{MOD}}}$$

[Figure 5-22](#) shows a plot of raw count versus sensor capacitance in dual IDAC mode. It shows that dual IDAC mode yields higher signal than the single IDAC mode.

Figure 5-22. Recommended Tuning in Dual IDAC Mode



The recommended way of tuning in the dual IDAC mode is to first find the value of  $I_{\text{MOD}}$  in the single IDAC mode, such that raw count is equal to 85 percent of the maximum raw count value for a given  $C_P$ . Let this  $I_{\text{MOD}}$  value be  $I_{\text{MOD\_Single\_IDAC}}$  and then set the IDAC value of both modulation and compensation IDACs to be  $I_{\text{MOD\_Single\_IDAC}}/2$ . When this is done, the raw count becomes equal to 70 percent of maximum raw count for the given  $C_P$  as [Figure 5-22](#) shows.

Using dual IDAC mode this way brings the following changes to the Raw Count versus  $C_P$  graph:

- Use of compensation IDAC introduces a non-zero intercept on the  $C_S$  axis as given by the following equation:

Equation 5-5.  $C_S$  Axis Intercept with Regards to  $I_{\text{COMP}}$

$$C_S \text{ axis intercept} = \left( \frac{I_{\text{COMP}}}{V_{\text{REF}} F_{\text{SW}}} \right)$$

- Because the value of  $I_{\text{MOD}}$  in the dual IDAC mode is half compared to the value of  $I_{\text{MOD}}$  in the single IDAC mode (all other parameters remaining the same), the gain  $G_C$  in the dual IDAC mode is double the gain in the single IDAC mode according to [Equation 5-3](#). Thus, the signal in the dual IDAC mode is double the signal in the single IDAC mode for a given resolution  $N$ .

While manually tuning a sensor, keep [Equation 5-3](#) and [Equation 5-4](#) as well as the following points in mind:

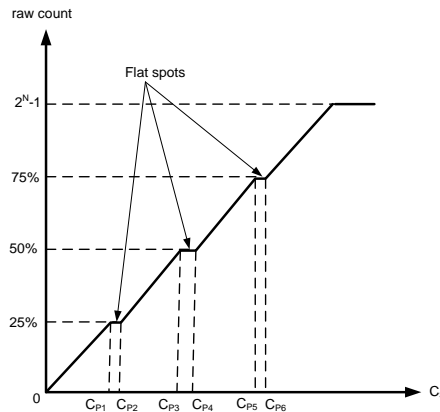
- Higher gain leads to increased sensitivity and better overall system performance. However, do not set the gain such that raw counts saturate, as the plot of gain  $G_{C3}$  shows in [Figure 5-20](#). Thus, it is recommended to set the gain in such a way that the raw count corresponding to  $C_P$  is 85 percent of the maximum raw count for the single IDAC mode and 70 percent of maximum raw count for the dual-IDAC mode.

2. The sense clock frequency ( $F_{SW}$ ) should be set carefully; higher the frequency, higher the gain, but the frequency needs to be low enough to fully charge and discharge the sensor as the figure [Voltage across Sensor Capacitance](#) indicates.
3. Enabling the Compensation IDAC plays a huge role in increasing the gain; it will double the gain if set as recommended [above](#). Always enable the Compensation IDAC when it is not being used for general-purpose applications.
4. Lower the modulation IDAC current, higher the gain. Adjust your IDAC to achieve the highest gain, but make sure that the raw counts corresponding to  $C_P$  have enough margin for environmental changes such as temperature shifts, as indicated in [Figure 5-21](#) and [Figure 5-22](#).
5. Increasing the number of bits of resolution used for scanning increases gain. An increase in resolution by one bit will double the gain of the system, but also double the scan time according to [Equation 3-7](#). A balance of scan time and gain needs to be achieved using resolution.

#### 5.3.2.1.2 Flat Spots

Ideally, raw counts should have a linear relationship with sensor capacitance as [Figure 5-19](#) and [Figure 5-22](#) show. However, in practice, sigma delta modulators have non-sensitivity zones, also called flat-spots or dead-zones – for a range of sensor capacitance values, the sigma delta modulator may produce the same raw count value as [Figure 5-23](#) shows. This range is known as a dead-zone or a flat-spot.

Figure 5-23. Flat Spots in Raw Counts versus Sensor Capacitance when Direct Clock is Used



In the case of CapSense CSD, these flat spots occur near 25%, 50%, and 75 percent of the maximum raw count value (that is, near 25%, 50%, and 75% of  $2^N-1$ , where  $N$  is the [Scan Resolution](#)). These flat spots are prominent when direct clock is used as [Sense Clock](#) source. Flat spots do not occur if PRS is used as the Sense Clock source (see also section [Sense Clock Related Parameters](#)).

For almost all systems, we recommend using PRS as the Sense Clock source because it limits the impact of flat spots and also provides EMI/EMC benefits as indicated in [Analog Switch Drive Source](#). If your system requires a direct clock, ensure that you use [auto-calibration](#) or avoid this raw count range when using manual calibration.

#### 5.3.2.2 Selecting CapSense Hardware Parameters

CapSense hardware parameters govern the conversion gain and CapSense signal. [Table 5-5](#) lists the CapSense hardware parameters that apply to CSD sensing method. The following section gives guidance on how to adjust these parameters to achieve optimal performance for your particular system.

Table 5-5. CapSense Component Hardware Parameters

SI. No.	CapSense Component Parameter Name	
	CapSense v3.0	CapSense v2.40
1	Sense Clock Frequency	Sense Clock Divider
2	Sense Clock Source	Analog Switch Drive Source
3	Modulator Clock Frequency	Modulator Clock Divider
4	Modulator IDAC	Modulation IDAC
5	Compensation IDAC	Compensation IDAC
6	Scan Resolution	Scan Resolution

### 5.3.2.2.1 Sense Clock Related Parameters

For a given resolution  $N$ , you can vary both  $F_{SW}$  and IDACs to make the raw count corresponding to  $C_P$  equal to 85 percent of the maximum raw count value. However, selecting an improper sense clock frequency ( $F_{SW}$ ) can affect the operation of CapSense CSD. From [Equation 5-3](#), it is best to use the maximum clock frequency but ensure a full charge and discharge of the sensor capacitor as indicated in [Figure 3-6. Voltage across Sensor Capacitance](#).

The simplest way to select the optimum clock frequency is to enable SmartSense on the device, and read back the values as outlined in [Button Widget Example](#). SmartSense will automatically read the sensor parasitic capacitance and set the maximum possible sense clock frequency to ensure proper charge and discharge. Alternatively, you can manually find the sense clock frequency using the following information:

The charge and discharge paths of the sensor capacitor include series resistances that slow down the charging and discharging process as indicated by the  $R_{Series}$  and GPIO cell switch in [Figure 3-2](#), [Figure 3-4](#), and [Figure 3-6](#).

If  $R_{SeriesTotal}$  is the sum of the GPIO resistance and the external series resistance and  $C_P$  is the parasitic capacitance of the sensor, you should select a switching frequency that is low enough to allow the sensor capacitance to fully charge and discharge. The rule of thumb is to allow a period of  $5R_{SeriesTotal}C_P$  for charging and discharging cycles. The equations for minimum time period and maximum frequency for sense clock follow:

Equation 5-6. Sense Clock Minimum Time Period

$$T_{SW}(\text{minimum}) = 10R_{SeriesTotal}C_P$$

Equation 5-7: Sense Clock Maximum Frequency

$$F_{SW}(\text{maximum}) = \frac{1}{10R_{SeriesTotal}C_P}$$

The typical value of the GPIO resistance is 500  $\Omega$  and the recommended external resistance is 560  $\Omega$  (see [Series Resistors on CapSense Pins](#) for details). Therefore, take the value of  $R_S$  as 1.06 k $\Omega$  when calculating the maximum switching frequency. The value for  $C_P$  can be estimated using the CSD Built-in-Self-test API; `CapSense_GetSensorCp()` provided with CapSense component v2.40. More information on this API can be found in the [Component Datasheet](#).

The source clock that drives  $F_{SW}$  is called as **Sense Clock Source** in CapSense v3.0 and **Analog Switch Drive Source** in CapSense v2.40.

By default, the system will use a direct clock as sense clock source that drives  $F_{SW}$ . In most cases because of [Flat Spots](#), and especially if your system is susceptible to EMI/EMC noise, it is recommended to enable the PRS clock option. In addition to Direct and PRS clock sources, the PSoC 4 S-Series of devices supports spread spectrum clock generation. In this case, the frequency of the sense clock is spread over a predetermined range to reduce EMI. The frequency spread range can be specified by selecting the clock divider as SSC2 to SSC5. Refer to the [PSoC 4000S TRM](#) for details on the spread spectrum architecture.

**Note** SSC should be selected such that  $2^{SSCn}$  is less than or equal to 10 percent of the ratio between the modulator clock and sense clock.

There are two PRS options: PRS-8 and PRS-12. In general, PRS-12 is better than PRS-8 but it is only true for the long scan periods when the whole PRS sequence fits into one scan. To determine which one to use, you should calculate the scan period and the PRS sequence period.

Equation 5-8. PRS Scan Period

$$T_{SCAN} = \frac{2^{Resolution} - 1}{\text{Modulator Clock Frequency}}$$

Equation 5-9: PRS Sequence Period

$$T_{PRS} = \frac{2^{PRS\_Resolution} - 1}{\text{Sense Clock Frequency}}$$

here  $N$  is either 8 or 12

After enabling PRS, make sure that  $T_{SCAN} \geq T_{PRS}$ . By choosing PRS-Auto, this process will be automatically done for you.

In CapSense v3.0, you can directly specify the **sense clock frequency** in the CapSense component configuration window. However, in CapSense v2.40, you need to specify the **sense clock divider**. Based on whether you chose direct clock or PRS clock as the sense clock frequency, use [Equation 3-4](#) or [Equation 3-5](#), with [Equation 5-9](#) to determine the value of sense clock divider.

For a ready reference on  $C_P$  versus sense clock frequency for 560 ohm series resistance, you can also see the [CapSense Component Datasheet](#).

#### 5.3.2.2.2 Modulator Clock Related Parameters

The modulator clock will impact scan time ([Equation 3-7](#)) as well as [Signal-to-Noise Ratio](#). In general, it is recommended to choose the highest modulator clock frequency. This will give the best possible scan times as well as reduce measurement error as much as possible.

In CapSense v3.0, you can directly specify the **modulator clock frequency** in the CapSense component configuration window. However, in CapSense v2.40, you need to specify the **modulator clock divider**. Choose the highest modulator clock frequency amongst the available options for CapSense v3.0. For CapSense v2.40, choose the lowest divider value to ensure the highest modulator clock frequency.

#### 5.3.2.2.3 Modulator and Compensation IDACs

To ensure the largest possible gain, you must first try to enable the compensation IDAC and then keep the modulation IDAC value as low as possible (see [Equation 5-4](#)).

The PSoC Creator CapSense Component includes a feature called **IDAC Auto-calibration** (called IDAC Auto-calibration in CapSense v3.0, **Auto-calibration** in CapSense v2.40), which automatically sets your IDAC values to achieve the optimal 85 percent raw count for single-IDAC mode or 70 percent for dual-IDAC mode.

If your design environment includes large temperature variation, you may find that the 85 percent calibration target is too high, and that over time your sensor saturates easily, leading to lower SNR. If this is the case you can adjust the calibration target percent lower, manually. When adjusting manually, your modulation IDAC will change 1.2-uA/bit in 4x mode and 2.4-uA/bit in 8x mode. You may use [Equation 3-9](#), [Equation 3-10](#), [Equation 3-11](#), or [Equation 3-12](#) to select IDAC values that will achieve your new target, or you can use a search algorithm.

**Note** The CapSense v3.0 Component does not support 8x mode.

When using a search algorithm, it is recommended to set the modulation IDAC to the lowest value (that is, 1) and slowly adjust it higher until the preferred calibration target is found. Increasing the  $I_{MOD}$  value will decrease the raw counts. If the raw counts remain saturated even with the highest value of  $I_{MOD}$ , increase the IDAC range to 8x and start over. Avoid adjusting raw counts to 25, 50, and 75 percent calibration targets if a direct clock used as the [Sense Clock](#) source, or use PRS as the Sense Clock source to avoid [Flat Spots](#).

When using a compensation IDAC, the recommended way of tuning, as explained in [Conversion Gain and CapSense Signal](#), is to first keep  $I_{COMP}$  to 0 and find the value of  $I_{MOD}$  such that the raw count is equal to 85 percent of the maximum raw count value for a given  $C_P$ , and then divide this  $I_{MOD}$  value by two and apply to both  $I_{MOD}$  and  $I_{COMP}$  (that is, split the current between  $I_{MOD}$  and  $I_{COMP}$  in a 50/50 ratio). When this is done, the raw count becomes equal to 70 percent of maximum raw count for the given  $C_P$ , as [Figure 5-22](#) shows.

In some cases, for example in very high  $C_P$  designs, after tuning based on the above guidelines, you may want to adjust the 50/50 split and decrease the modulation IDAC value to increase gain ([Equation 5-3](#)). This will cause the raw count to increase to a higher calibration percentage or even saturate. To bring down the raw count to the same calibration percentage as earlier and avoid saturation, increase the compensation IDAC value. Decreasing the Modulation IDAC increases the gain, and hence the signal. Note that an increase in signal may not correspondingly result in an increased [Signal-to-Noise Ratio](#) as increasing the gain may result in increase in both signal and noise. However, if required, you can use firmware filters to decrease noise. For information on available firmware filters, see [Table 5-2](#) or [Table 5-4](#) depending on the CapSense component version you are using.

#### 5.3.2.2.4 Scan Resolution

Scan resolution needs to be selected to maintain a balance between the signal and scan time. As resolution increases the signal will increase ([Equation 5-3](#)) but so will the scan time (according to [Equation 3-7](#)). For each bit, both will double. For most sensor designs, it is recommended to begin with the lowest possible resolution and slowly increase this value to increase signal. If there is room in the timing budget, the resolution can be increased to achieve an SNR beyond the minimum 5:1 recommendation to increase system robustness.



### 5.3.2.3 Selecting CapSense Software Parameters

CapSense software parameters govern the sensor status based on the raw count of a sensor. [Table 5-6](#) provides a list of CapSense software parameters. These parameters apply to both CSD and CSX sensing methods. This section defines these parameters with the help of [Baseline](#), and provides guidance on how to adjust these parameters for optimal performance of your design.

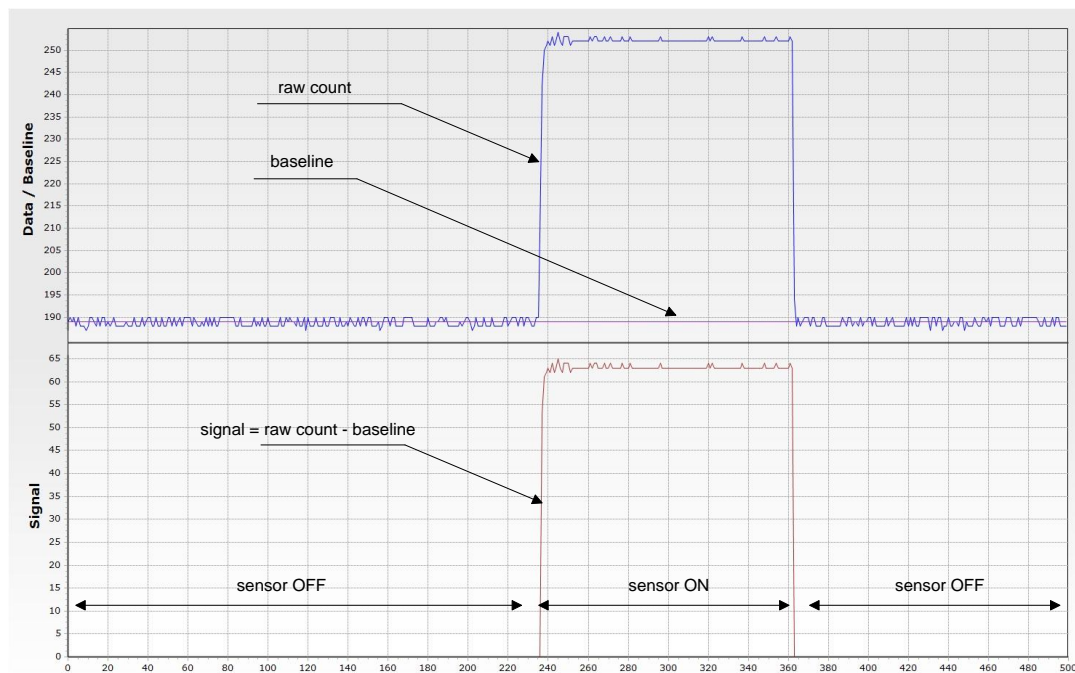
Table 5-6. CapSense Component Software Parameters

Sl. No.	CapSense Component Parameter Name	
	CapSense v3.0	CapSense v2.40
1.	Finger Threshold	Finger Threshold
2.	Noise Threshold	Noise Threshold
3.	Hysteresis	Hysteresis
4.	ON Debounce	Debounce
5.	Sensor Auto-Reset	Sensor Auto Reset
6.	Low Baseline Reset	Low Baseline Reset
7.	Negative Noise Threshold	Negative Noise Threshold

#### 5.3.2.3.1 Baseline

After tuning the CapSense Component for a given  $C_P$ , the raw count value of a sensor may vary gradually due to changes in the environment such as temperature and humidity. Therefore, the CapSense Component creates a new count value known as baseline by low-pass filtering the raw counts. **Baseline** keeps track of, and compensates for, the gradual changes in raw count. The baseline is less sensitive to sudden changes in the raw count caused by a touch. Therefore, the baseline value provides a reference level for computing signal. [Figure 5-24](#) shows the concept of raw count, baseline, and signal.

Figure 5-24. Raw Count, Baseline, and Signal



Baseline needs to be updated after every scan of the CapSense sensors by calling the `CapSense_UpdateEnabledBaselines()` API.

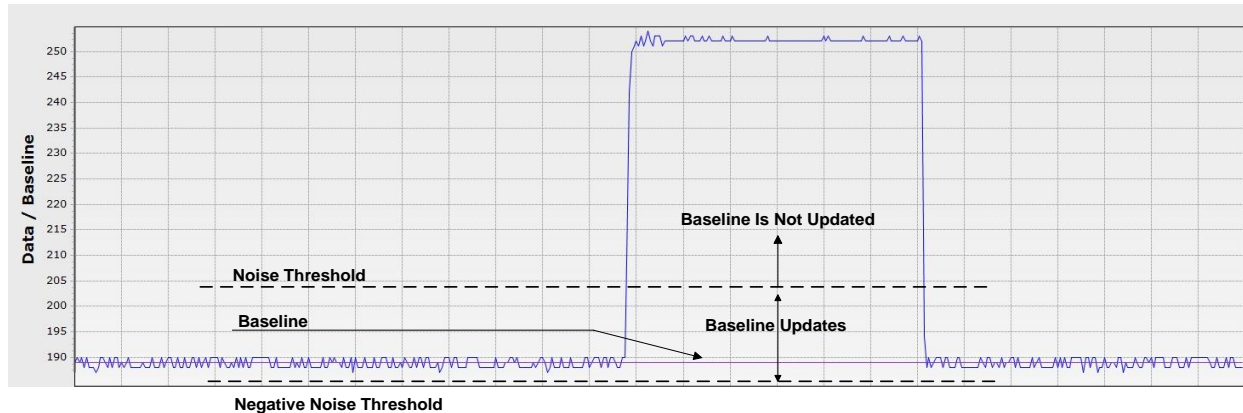


### 5.3.2.3.2 Baseline Update Algorithm

To properly tune the CapSense software, that is, the threshold parameters, it is important to understand how baseline is calculated by the CapSense\_UpdateEnabledBaselines() API and how the threshold parameters affect the baseline update.

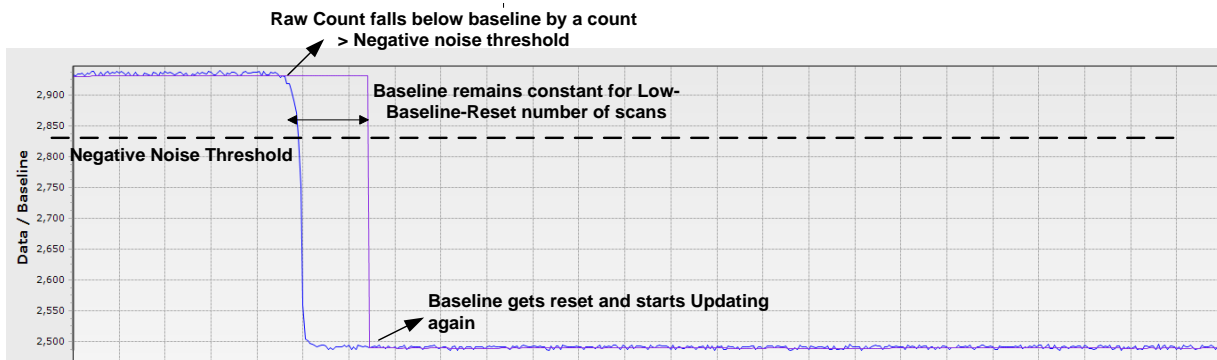
Baseline is basically a low-pass-filtered version of raw counts. As Figure 5-25 shows, baseline is updated by low-pass-filtering raw counts if the current raw count is within a range of (Baseline – Negative Noise Threshold) to (Baseline + Noise Threshold). If the current raw count is higher than baseline by a value greater than noise threshold, baseline remains at a constant value equal to prior baseline value.

Figure 5-25. Baseline Update Algorithm



If the current raw count is below Baseline minus Negative Noise Threshold, baseline again remains constant at a value equal to prior baseline value for Low Baseline Reset number of sensor scans. If the raw count continuously remains lower than Baseline minus Noise Threshold for low baseline reset number of scans, the baseline is reset to the current raw count value and starts getting updated again, as Figure 5-26 shows.

Figure 5-26. Low Baseline Reset



### 5.3.2.3.3 Finger Threshold

The finger threshold parameter is used along with the hysteresis parameter to determine the sensor state, as Equation 5-10 shows.

Equation 5-10: Sensor State

$$\text{Sensor State} = \begin{cases} \text{ON} & \text{if (Signal} \geq \text{Finger Threshold} + \text{Hysteresis)} \\ \text{OFF} & \text{if (Signal} \leq \text{Finger Threshold} - \text{Hysteresis)} \end{cases}$$

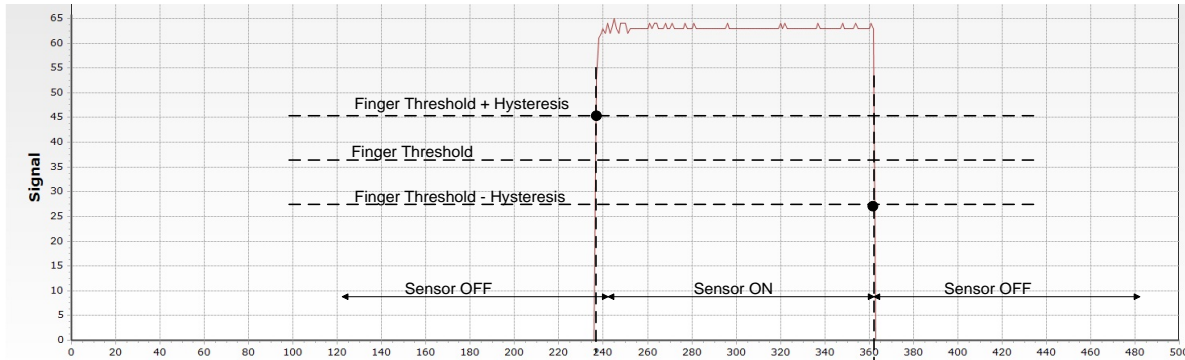
Note that signal in the above equation refers to the difference: Raw Count – Baseline, when the sensor is touched, as Figure 5-24 shows.

It is recommended to set Finger Threshold to 80 percent of the signal. This setting allows enough margin to reliably detect sensor ON/OFF status over signal variations across multiple PCBs.

#### 5.3.2.3.4 Hysteresis

The hysteresis parameter is used along with the finger threshold parameter to determine the sensor state, as [Equation 5-10](#) and [Figure 5-27](#) show. Hysteresis provides immunity against noisy transitions of sensor state. The hysteresis parameter setting must be lower than the finger threshold parameter setting.

Figure 5-27. Hysteresis



It is recommended to set hysteresis to 10 percent of the signal.

#### 5.3.2.3.5 Noise Threshold

For single-sensor widgets, such as buttons and proximity sensors, the noise threshold parameter sets the raw count limit above which the baseline is not updated, as [Figure 5-25](#) shows. In other words, the baseline remains constant as long as the raw count is above *baseline + noise threshold*. This prevents the baseline from following raw counts during a finger touch.

The noise threshold value should always be lower than the *finger threshold – hysteresis*. It is recommended to set noise threshold to 40 percent of the signal.

If the noise threshold is set to a low value, the baseline will remain constant if raw counts suddenly increase by a small amount, say because of small shifts in power supply or shifts in ground voltage because of high GPIO sink current and so on.

On the other hand, if the noise threshold is set to a value close to *finger threshold – hysteresis*, the baseline may keep updating even when the sensor is touched. This will lead to reduced signal (note that *signal = raw count – baseline*) and the sensor state may not be reported as ON.

#### 5.3.2.3.6 Negative Noise Threshold

The negative noise threshold parameter sets the raw count limit below which the baseline is not updated for the number of samples specified by the low baseline reset parameter as [Figure 2-26](#) shows.

Negative noise threshold ensures that the baseline does not fall low because of any high amplitude repeated negative noise spikes on raw count caused by different noise sources such as ESD events.

It is recommended to set the negative noise threshold parameter value to be equal to the noise threshold parameter value.

#### 5.3.2.3.7 Low Baseline Reset

This parameter is used along with the negative noise threshold parameter. It counts the number of abnormally low raw counts required to reset the baseline as [Figure 2-26](#) shows.

If a finger is placed on the sensor during device startup, the baseline is initialized to the high raw count value at startup. When the finger is removed, raw counts fall to a lower value. In this case, the baseline should track the low raw counts. The Low Baseline Reset parameter helps to handle this event. It resets the baseline to the low raw count value when the number of low samples reaches the low baseline reset number. Note that in this case, when the finger is removed from the sensor, the sensor will not respond to finger touches for a low baseline reset time given by [Equation 5-11](#).

Equation 5-11. Low Baseline Reset Time

$$\text{Low Baseline Reset Time} = \frac{\text{Low Baseline Reset parameter value}}{\text{Scan Rate}}$$

The low baseline reset parameter should be set to meet following conditions on [Low Baseline Reset](#):

- Low Baseline Reset Time is greater than the time for which negative noise (due to noise sources such as ESD events) is expected to last
- Low Baseline Reset Time is lower than the time in which a sensor is expected to start responding again after the finger kept on sensor during device startup is removed from the sensor.

The low baseline reset parameter is generally set to a value of 30.

#### 5.3.2.3.8 Debounce

This parameter, called **Debounce** in CapSense v2.4 and **ON Debounce** in CapSense v3.0, selects the number of consecutive CapSense scans during which a sensor must be active to generate an ON state from the Component. Debounce ensures that high-frequency, high-amplitude noise does not cause false detection.

Equation 5-12. Sensor State with Debounce

$$\text{Sensor State} = \begin{cases} \text{ON} & \text{if (Signal} \geq \text{Finger Threshold} + \text{Hysteresis) for scans} \geq \text{debounce} \\ \text{OFF} & \text{if (Signal} \leq \text{Finger Threshold} - \text{Hysteresis)} \\ \text{OFF} & \text{if (Signal} \geq \text{Finger Threshold} + \text{Hysteresis) for scans} < \text{debounce} \end{cases}$$

The Debounce parameter impacts the response time of a CapSense system. The time it takes for a sensor to report ON after the raw counts value have increased above finger threshold + hysteresis because of finger presence, is given by the following equation:

Equation 5-13 Relationship between Debounce and Sensor Response Time

$$\text{Sensor response time} = \frac{\text{Debounce}}{\text{Scan Rate}}$$

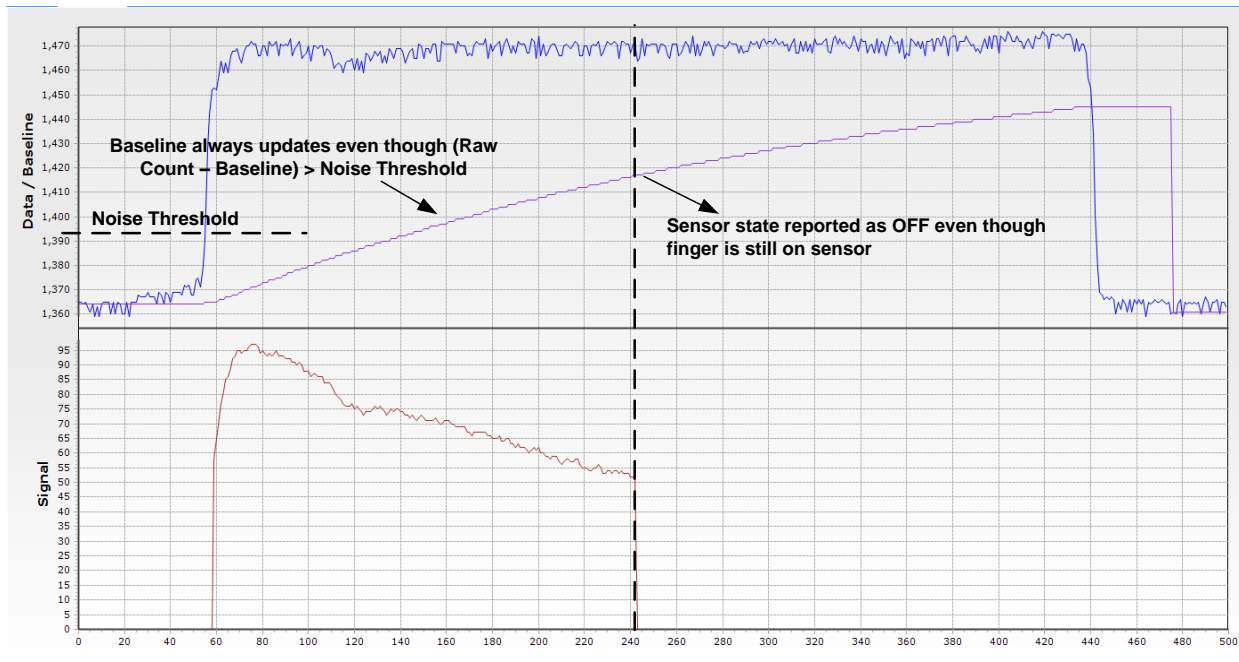
The Debounce parameter is generally set to a value of '3' for reliable sensor status detection. It can be raised or lowered based on the noise aspects of the end user system

#### 5.3.2.3.9 Sensor Auto Reset

Enabling the Sensor Auto Reset parameter causes the baseline to always update regardless of whether the signal is above or below the noise threshold.

When auto reset is disabled, the baseline only updates if the current raw count is within a range of  $(\text{Baseline} - \text{Negative Noise Threshold})$  to  $(\text{Baseline} + \text{Noise Threshold})$  as Figure 5-25 shows and the Baseline Update Algorithm describes. However, when Auto Reset is enabled, baseline is always updated if the current raw count is higher than  $(\text{Baseline} - \text{Negative Noise Threshold})$  as Figure 5-28 shows.

Figure 5-28. Baseline update with Sensor Auto Reset Enabled



Because the baseline is always updated when sensor auto reset is enabled, this setting limits the maximum time duration for which the sensor will be reported as pressed. However, enabling this parameter prevents the sensors from permanently turning on if the raw count suddenly rises without anything touching the sensor. This sudden rise can be caused by a large power supply voltage fluctuation, a high-energy RF noise source, or a very quick temperature change.

Enable this option if you have a problem with sensors permanently turning on when the raw count suddenly rises without anything touching the sensor.

#### 5.3.2.4 Button Widget Example

The following examples explain tuning of self-capacitance based button widgets in a PSoC Creator CapSense Component using the [Tuner Helper](#). For details on the Component and all related parameters, refer to the [Component Datasheet](#).

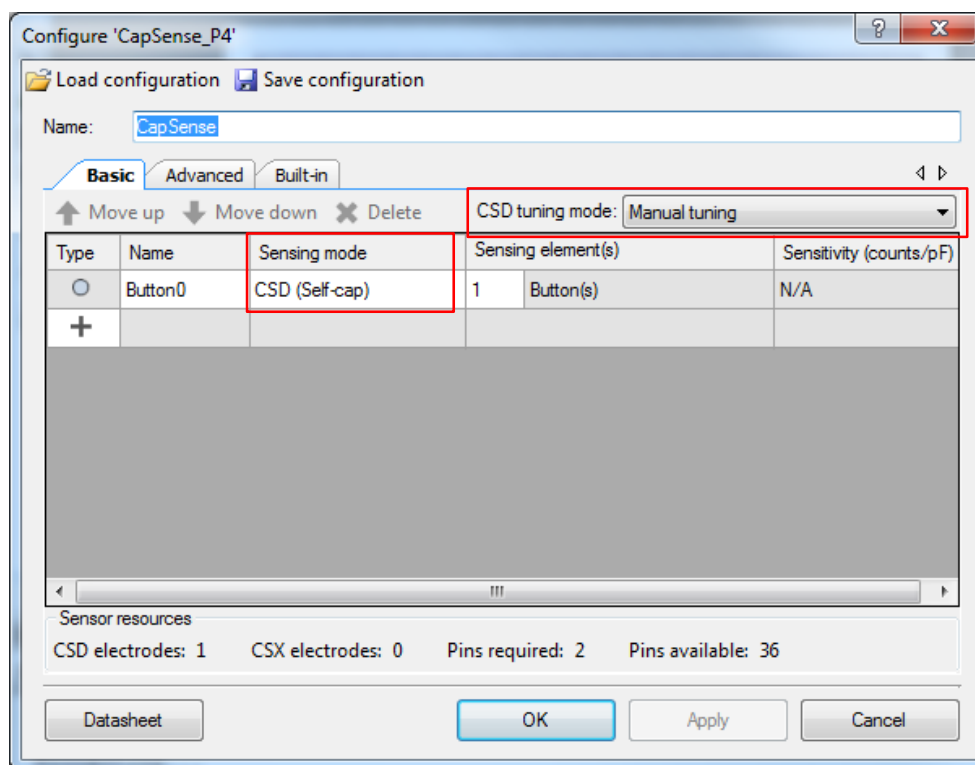
Self-capacitance-based sensing is supported in CapSense v3.0 and earlier versions. This section provides details for both CapSense v3.0 and CapSense v2.40 Components in PSoC Creator. It is recommended to use CapSense v3.0 for all new designs; however, if you are using an older version of the CapSense Component for a legacy project, skip to [Using CapSense v2.40](#).

##### 5.3.2.4.1 Using CapSense v3.0

Follow the detailed process listed in the following steps to manually set all the tuning parameters.

1. Open the CapSense Component configuration window by double-clicking the Component or right-clicking on the Component and selecting **Configure**.
2. In the Basic tab, add the button widget by clicking on the **+** symbol. Select the **Sensing mode** as **CSD (Self-cap)** and **CSD tuning mode** as **Manual tuning**, as [Figure 5-29](#) shows.

Figure 5-29. Selecting Manual Tuning Mode



3. In the **Advanced** tab > **General** settings window, leave all the filter parameters at their default settings. Filters will be enabled depending on the SNR and response time requirements as explained in [step 11](#).
4. In the **Advanced** tab > **CSD Settings** window, specify the parameters as shown in [Figure 5-30](#) and explained in [Table 5-7](#).

Figure 5-30. CSD Widget Settings

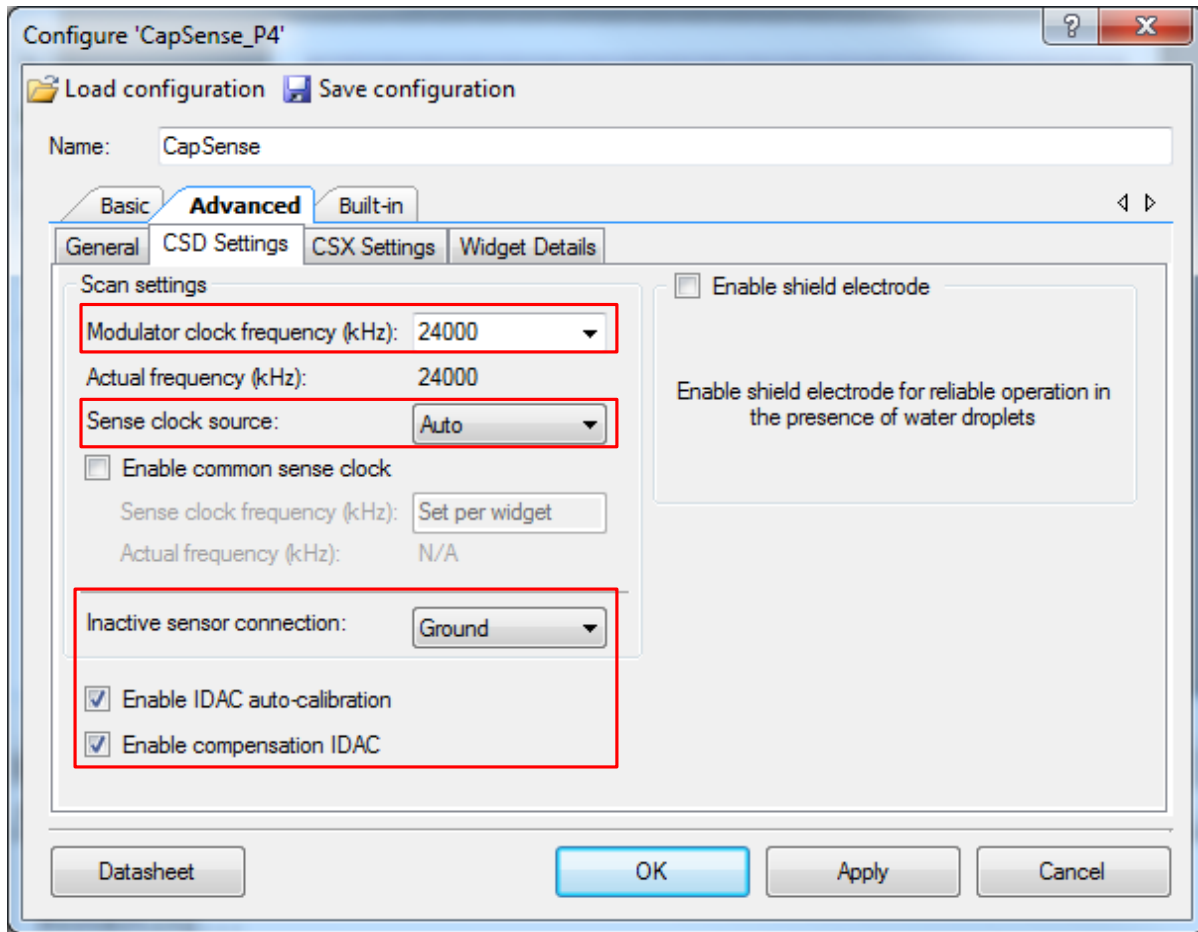


Table 5-7. CapSense Component CSD Configuration Window

Parameter	Value	Rationale
Modulator clock frequency	Maximum available option	Higher modulator clock frequency reduces sensor scan time, which results in lower power and reduces the noise in the raw counts; hence, it is recommended to use the highest possible frequency.
Sense clock source	Auto	Enabling Spread Spectrum (SSC) helps to deal with EMC/EMI or <a href="#">Flat Spots</a> issues. Refer to <a href="#">Sense Clock</a> , <a href="#">Sense Clock Related Parameters</a> sections for more information on choosing direct or SSC clocks.
Inactive sensor connection	Ground(default)	Inactive sensors are connected to ground to provide good shielding from noise sources. Use inactive sensor connection as shield for liquid-tolerant designs or if your design contains a proximity sensor. For additional information see the <a href="#">Liquid Tolerance</a> section and <a href="#">AN92239 Proximity Sensing with CapSense</a>
Enable IDAC auto-calibration	Enabled	Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (85 percent in single-IDAC mode and 70 percent for dual-IDAC mode). For systems that may need a different calibration point because of environmental factors, see <a href="#">Modulator and Compensation IDACs</a> .
Enable compensation IDAC	Enabled	Enabling the compensation IDAC selects the dual-IDAC mode operation of the CSD. Dual-IDAC mode gives higher signal values compared to single-IDAC mode as explained in <a href="#">Modulator and Compensation IDACs</a> .
Enable shield electrode	Per PCB design	Enable shield if your design requires large proximity sensing distance, <a href="#">Liquid Tolerance</a> , or if the shield is being used to reduce $C_P$ of sensors.

5. In the **Advanced** tab > **Widget Details** window, specify the settings as shown in Figure 5-31 and Figure 5-32 and explained in Table 5-8 and Table 5-9.

Figure 5-31. CapSense Component Widget Details Window

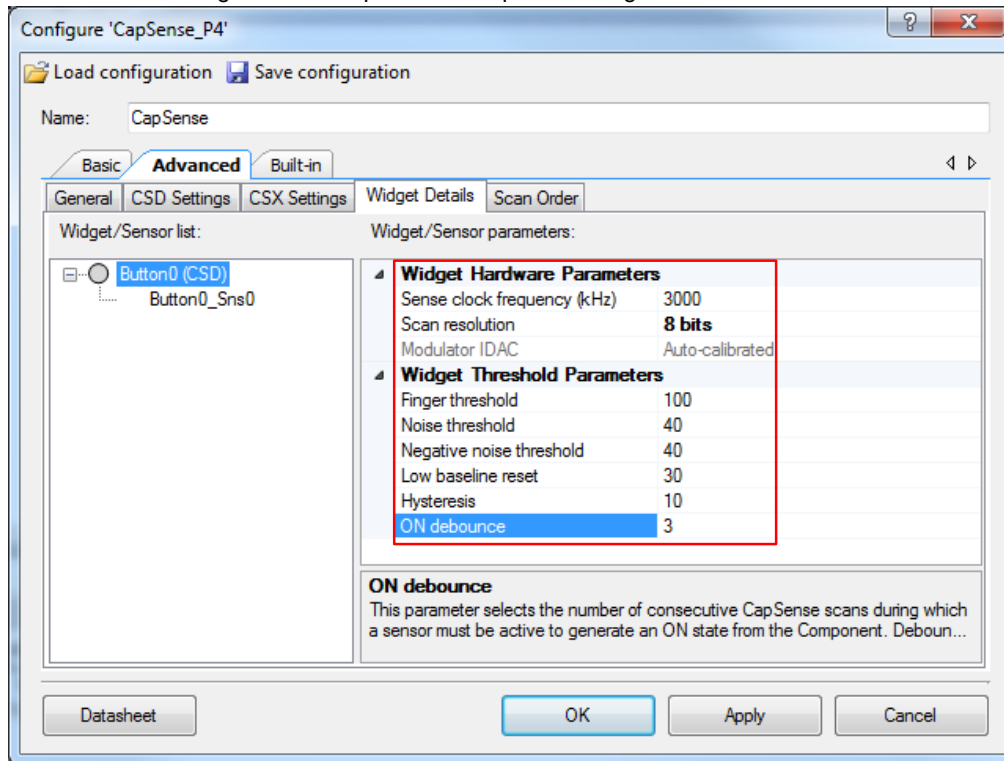


Figure 5-32. CapSense Component Widget Details Window – Sensor Settings

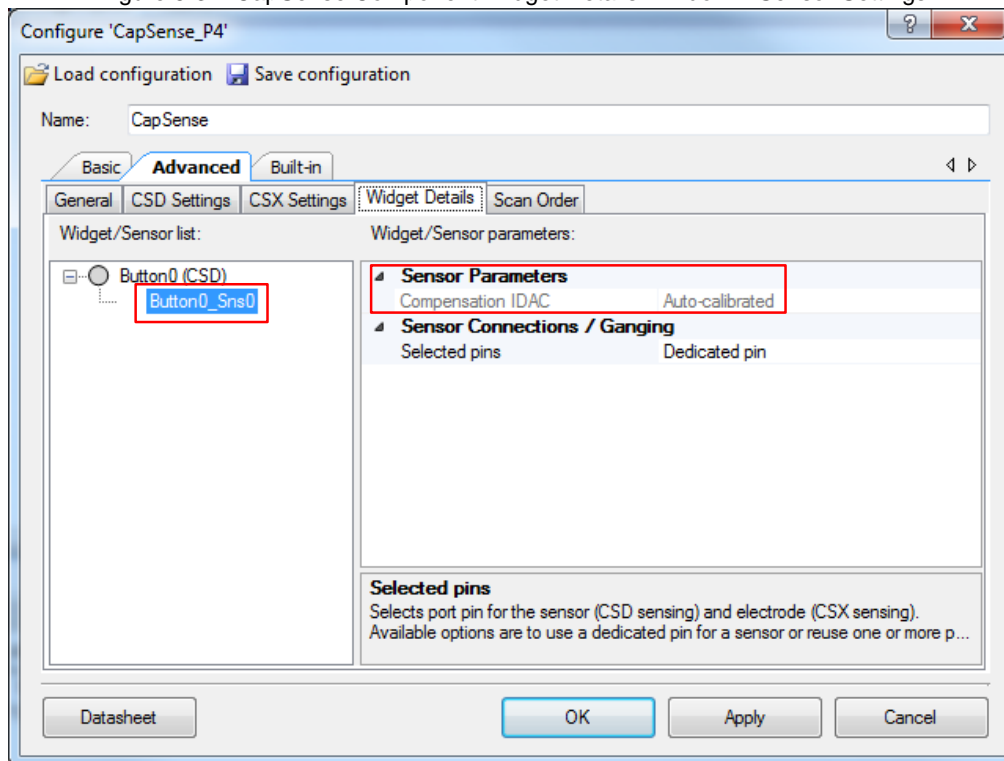




Table 5-8. CapSense Component Widget Details Window

Parameter	Value	Rationale
Sense clock frequency	$\frac{1}{10R_{SeriesTotal}C_P}$	See <a href="#">Sense Clock Related Parameters</a> to choose the appropriate frequency.
Scan resolution	8-bit	8-bits is a good starting point to ensure fast scan time, and some signal. This value will be adjusted as needed in step 8.
Modulator IDAC	NA	With auto-calibration enabled, the device automatically chooses this value. To choose a different IDAC value, see <a href="#">Modulator and Compensation IDACs</a> .
Finger threshold (FT)	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Noise threshold	5.3.3 Default	5.3.4 Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Negative noise threshold	5.3.5 Default	5.3.6 Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Low baseline reset	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Hysteresis	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
ON Debounce	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.

Table 5-9. CapSense Component Widget Details Window – Sensor Settings

Parameter	Value	Rationale
Compensation IDAC	NA	With auto-calibration enabled, the device automatically chooses this value. To choose a different IDAC value, see <a href="#">Modulator and Compensation IDACs</a> .
Selected Pins	Default	This parameters allows you to gang multiple sensors and scan as a single sensor.

- Next, use the tuner to observe raw counts in the Graphing tab in Tuner GUI and calculate the [Signal-to-Noise Ratio](#) of the sensor. Refer to [CapSense Component datasheet](#) for the detailed procedure on how to add tuner to your project. To calculate SNR, measure the raw count value when a finger is present and divide it by the peak-to-peak noise of the raw counts with no touch present. 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 – 9 mm.
- If the initial SNR is greater than 5, you can move to step 9. Otherwise, move to step 8.
- When the SNR is less than 5, increase it to achieve proper performance. The main parameters that influence SNR are resolution and filters. Select an appropriate filter for your application based on [Table 5-2. Raw Data Noise Filters](#).

Scan Resolution – Can be increased to increase signal at a disproportionate rate to noise to improve overall SNR. Increasing resolution adds to the overall scan time based on [Equation 3-7](#).

Filters – Filters help to reduce noise, without increasing the signal. Based on your noise type you can enable a filter to improve SNR. Each filter will add additional processing time as well as memory use.

It is best to find a balance between the resolution 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 that resolution can be updated directly in the Widget/Sensor Parameter window, as [Figure 5-33](#) shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as [Figure 5-34](#) shows, and reprogram the device to update filter settings. For details on filters, see the [CapSense Component Datasheet](#).

Figure 5-33. Update Scan Resolution in Tuner GUI

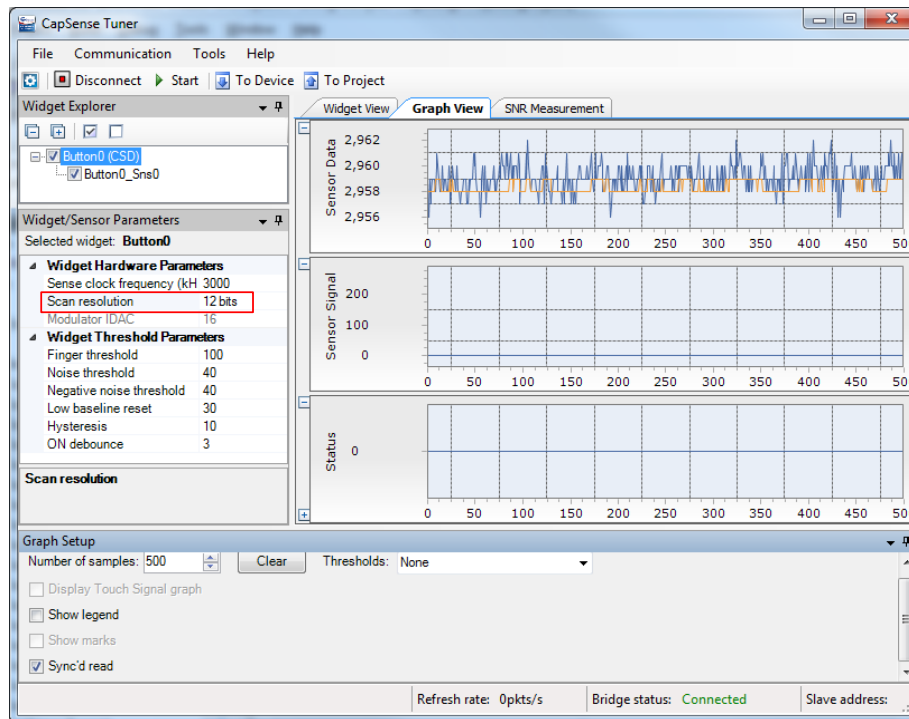
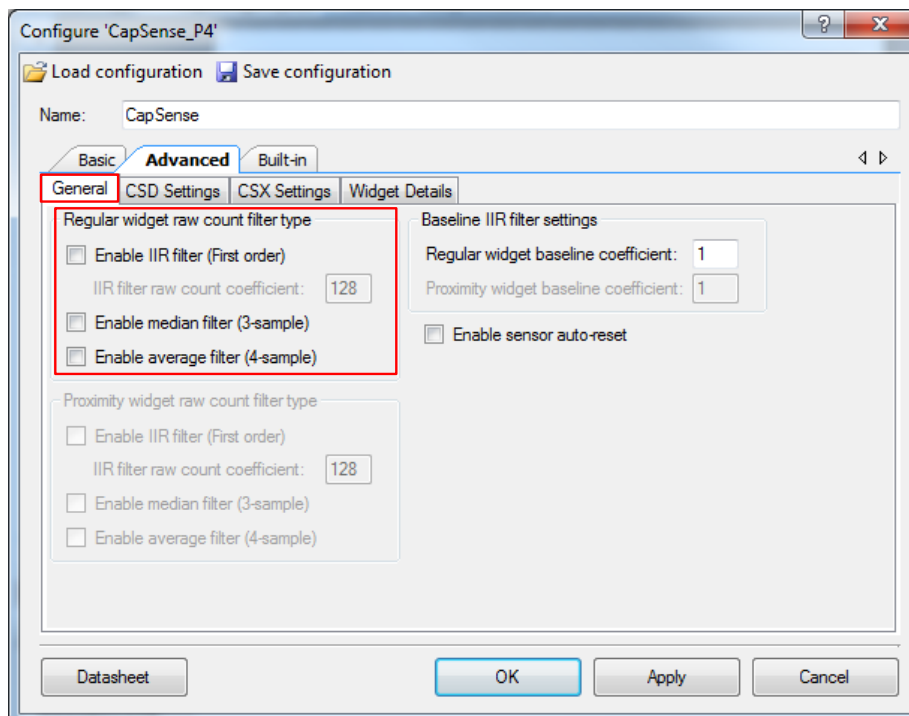


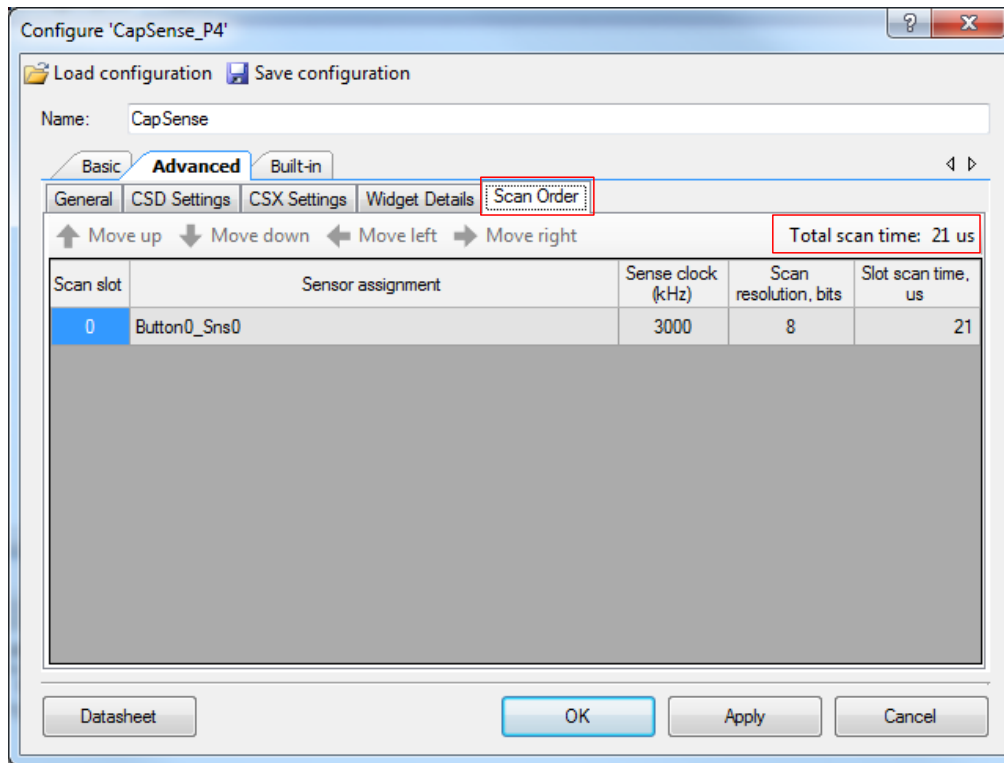
Figure 5-34. Update Filter Settings in Configure 'CapSense\_CSD\_P4' Dialog



9. Check the total scan time (see [Figure 5-35](#)) to determine if it meets the system requirements. This timing will impact the response time and is an important factor in the overall power consumption of the device in CapSense applications, as indicated in [Power Consumption and Response Time](#).



Figure 5-35. Sensor Scan Time in Scan Order Tab



10. If you meet the timing requirement of the system, skip to step 11. Otherwise, adjust the tuning to speed up the scan time. If SNR is greater than 10 on any sensor, then you can lower your resolution or remove filters to decrease scan time, but keep your SNR greater than 5. If you are unable to meet your timing requirements and maintain SNR greater than 5, you should see step 12.
11. After you have confirmed that your design meets the timing parameters, and the SNR is greater than 5, set the widget threshold parameters for your design as follows. Ensure that you observe the difference count (**signal** output) in the Graph View tab in Tuner GUI, *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 - 9 mm. Consider testing with smaller sizes that should be "rejected" by the system to ensure they do not reach the finger threshold

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

Again, these settings can be first set in the tuner GUI, as [Figure 5-36](#) shows, or they can be input directly in the CapSense Component customizer, as [Figure 5-37](#) shows.

For more information on these settings, refer to [Selecting CapSense Software Parameters](#).

12. If you are not able to achieve an SNR greater than 5 or cannot meet your timing requirements, refer to [Tuning Debug FAQs](#) or Manual Tuning [Basics](#) for more information on how to tune your system. You may need to modify the advanced parameters of the CapSense Component and/or adjust your hardware design to meet the end system requirements.

Figure 5-36. Updating Threshold Parameters in Tuner GUI

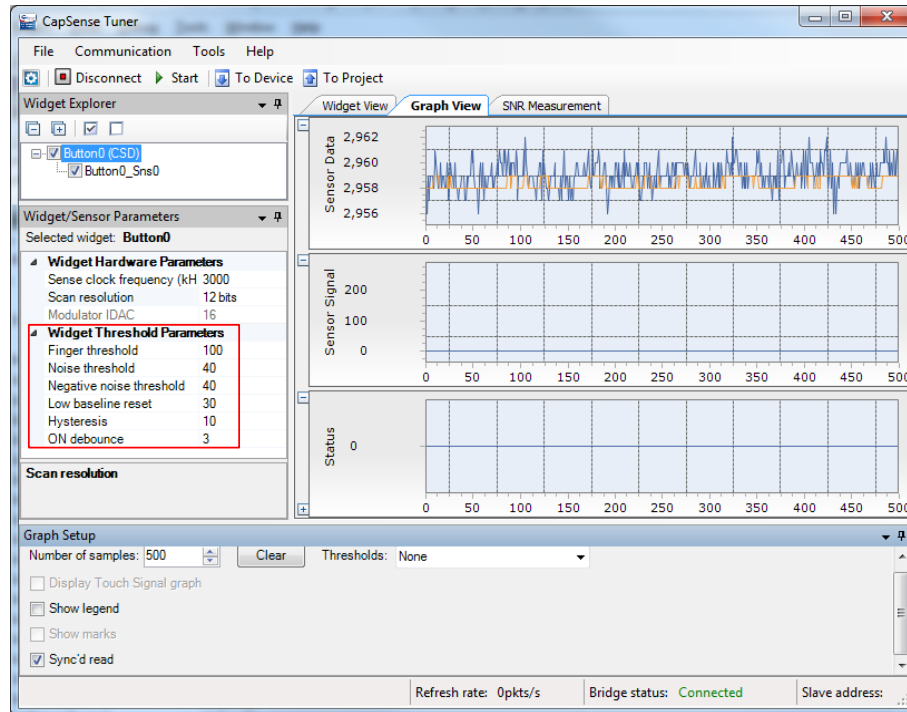


Figure 5-37. Updating Threshold Parameters in Configure 'CapSense\_CSD\_P4' Dialog

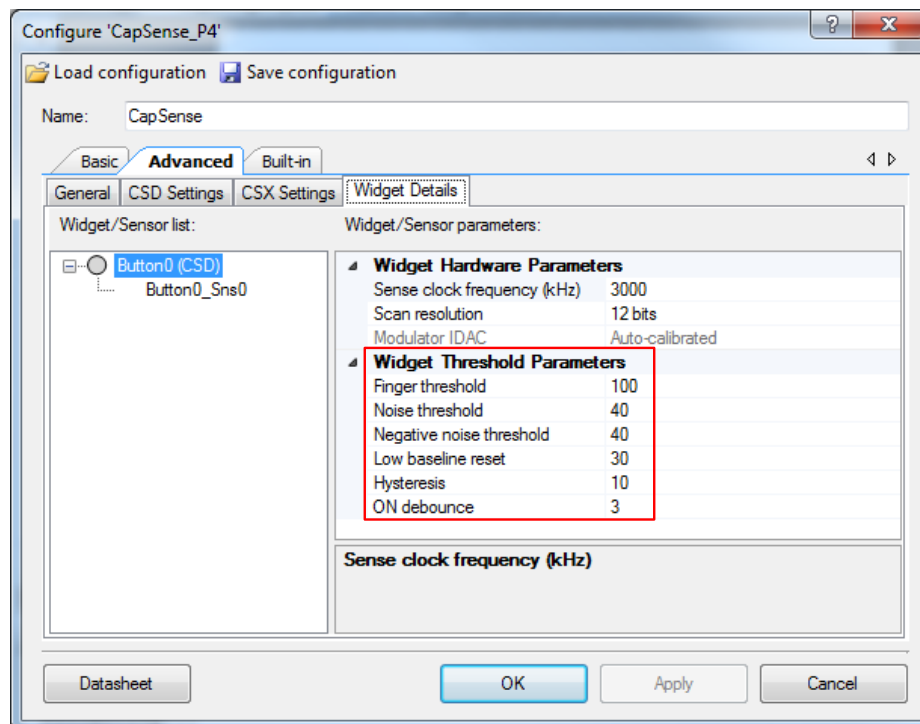
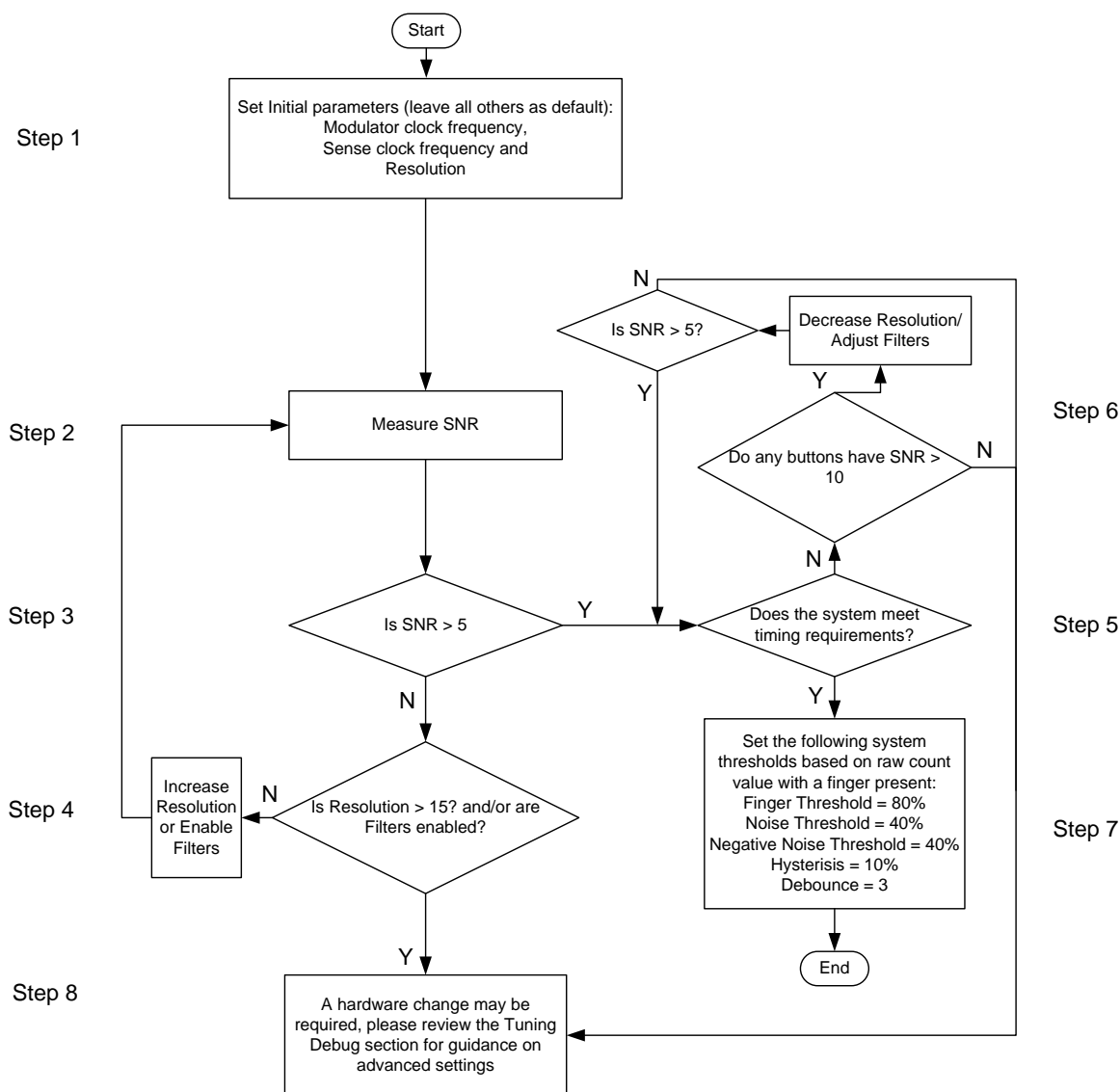


Figure 5-38. CSD Button Widget Tuning Flow for CapSense v3.0



#### 5.3.6.1.1 Using CapSense v2.40

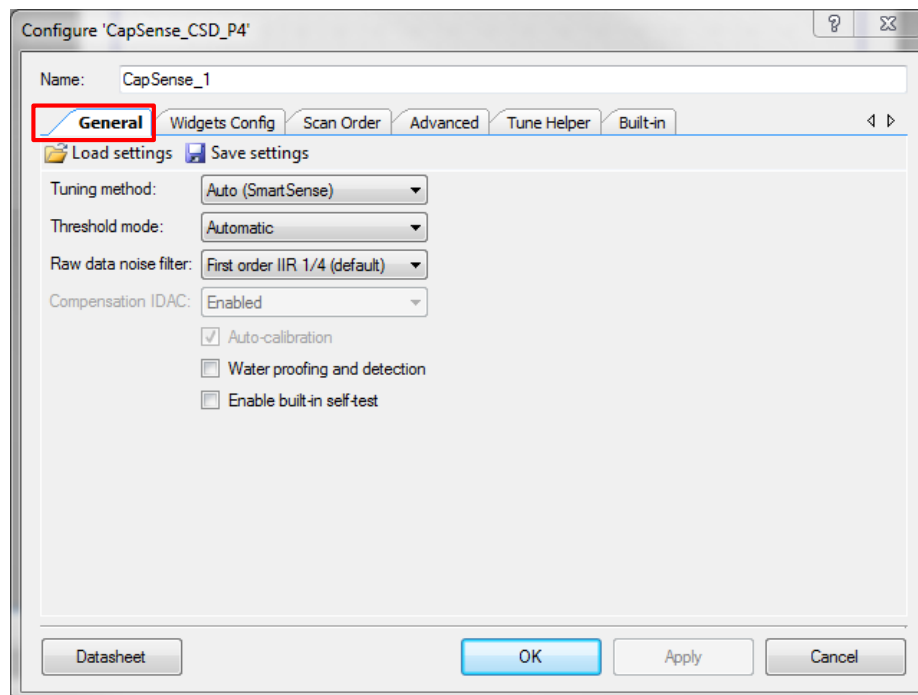
Follow the detailed process listed here to manually set all the tuning parameters. See [Figure 5-54](#) for a quick reference flow chart.

- The first step in the manual tuning process is to determine the values for [Sense Clock Related Parameters](#) and [Modulator Clock Related Parameters](#). These can be determined either manually, as explained in [Selecting CapSense Hardware Parameters](#), or by reading back the values set by SmartSense. Using SmartSense is the fastest, and hence, the recommended way.

Follow these steps to capture the clock divider values set by SmartSense:

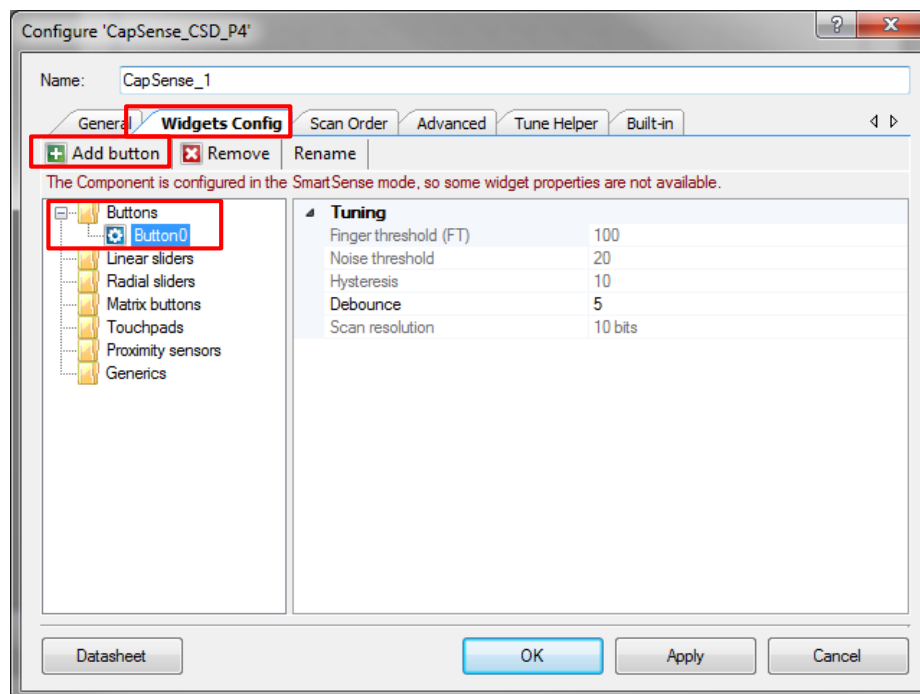
- Double-click on the **CapSense CSD** Component to open the **Configure 'CapSense\_CSD\_P4'** dialog.
- In the **General** tab, leave all the parameters to their default values, as [Figure 5-39](#) shows, to use the SmartSense tuning method for capturing divider values. These values will be changed in step 2 and later steps for manually tuning the buttons.

Figure 5-39. CapSense General Tab Configuration to Capture SmartSense Clock Divider Values



- c) In the **Widgets Config** tab, add the required button widgets by clicking **Add button**, as [Figure 5-40](#) shows. Leave all other parameters in this tab and the **Scan Order** tab at their default values; these will be changed in later steps for manually tuning the buttons.

Figure 5-40. Adding Button Widgets in Widgets Config Tab



- d) In the **Advanced** tab, set the **Analog switch drive source** as **PRS-Auto**. Enabling PRS provides EMC/EMI benefits and also reduces **Flat Spots**. See the [Sense Clock](#) and [Sense Clock Related](#)

[Parameters](#) sections for more information on selecting between **Direct**, **PRS-8b**, **PRS-12b**, and **PRS-Auto**. If your PCB design does not contain a shield electrode and/or guard sensor (see [Driven-Shield Signal and Shield Electrode](#) and [CapSense CSD Shielding](#) for details on shield electrode), leave all other parameters as default, as [Figure 5-41](#) shows.

However, if your PCB design contains a shield electrode and/or guard sensor, enable the same in the **Advanced** tab, as [Figure 5-42](#) shows.

Note that it is important to enable the shield electrode (if it is being used in the design) while using SmartSense to capture clock divider values. Shield electrode reduces the sensor  $C_P$  and the SmartSense algorithm first calculates sensor  $C_P$  and then uses this data to calculate clock divider values.

Figure 5-41. Advanced Tab Configuration to Capture SmartSense Clock Divider Values

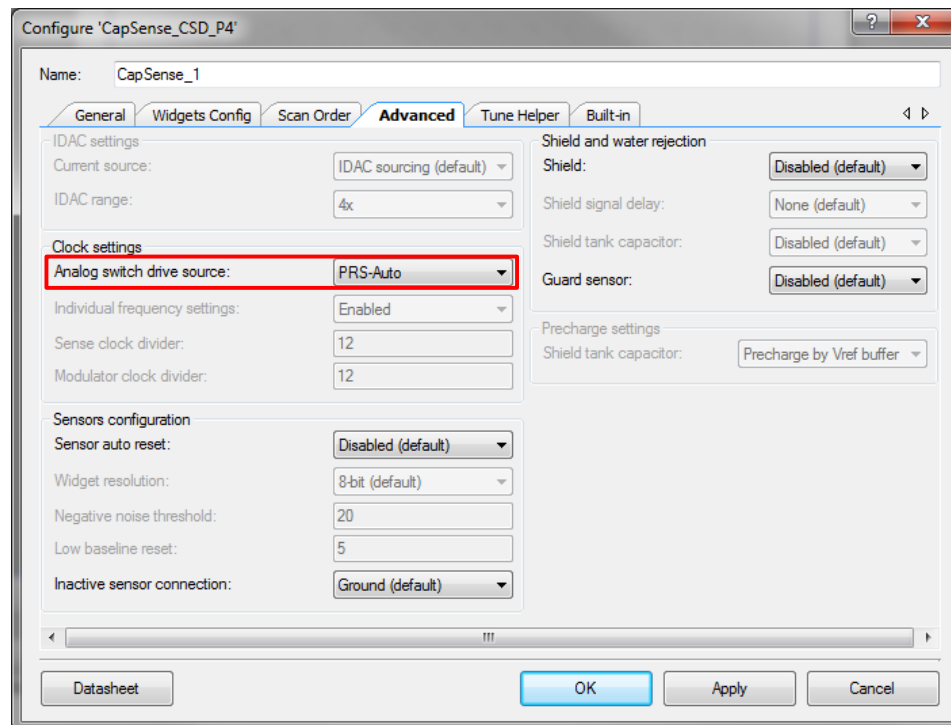
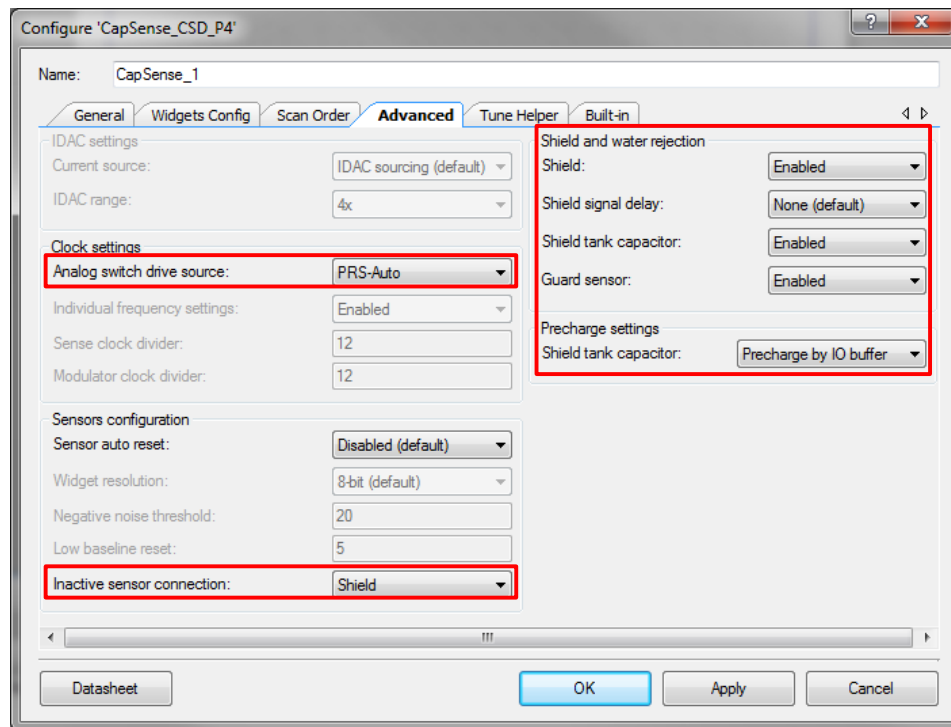
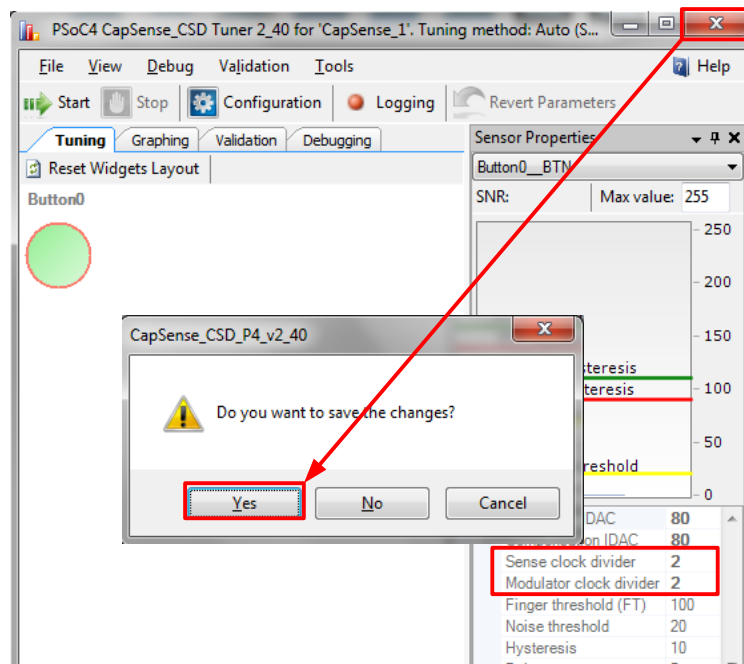


Figure 5-42. Advanced Tab Configuration for Designs using Shield Electrode



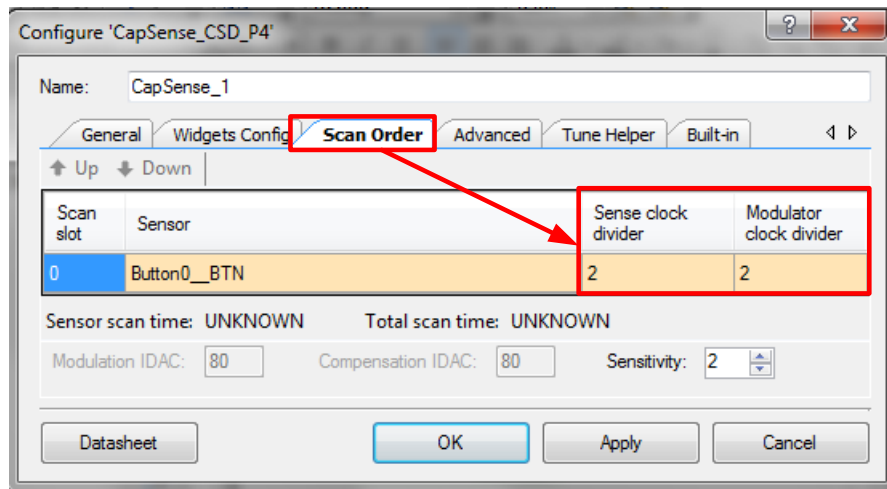
- e) In the **Tune Helper** tab, select the **Enable tune helper** check box and follow steps in “Using the Tuner GUI” section in the [Component datasheet](#) to use the tuner to capture the SmartSense clock values.
- f) Close the tuner and select **Yes** to save the changes, as [Figure 5-43](#) shows.

Figure 5-43. Closing Tuner and Saving Clock Divider Values to CapSense Component



- g) Open the Component and ensure the clock settings match the tuner settings as [Figure 5-44](#) shows.

Figure 5-44. Clock Divider Values Updated in CapSense Component



It is recommended to use SmartSense to set clock dividers as this will set the maximum possible sense clock frequencies to fully charge and discharge a sensor. If you face problems with EMI/EMC because of the chosen frequencies, go to [Electromagnetic Compatibility \(EMC\) Considerations](#) and [Sense Clock Related Parameters](#).

- The next step in the manual tuning process is to change the tuning method to **Manual with run-time tuning** and configure the Component with initial values. Use the following initial values for tuning:

Figure 5-45. General Settings

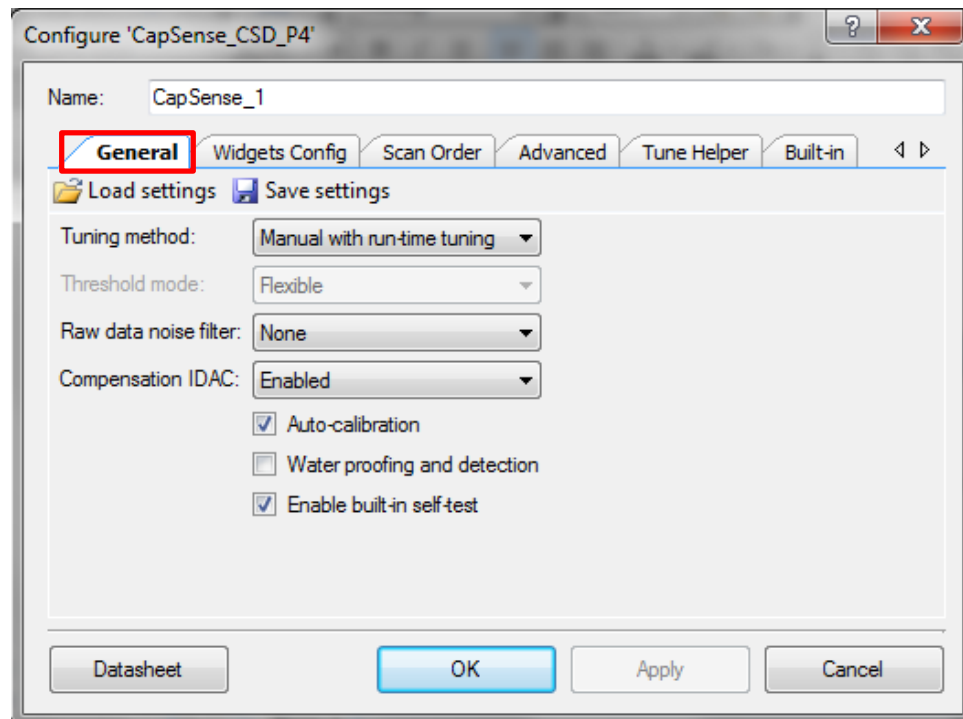


Table 5-10. CapSense Component General Configuration Window

Parameter	Value	Rationale
Tuning method	Manual with run-time tuning	The manual method with the runtime tuning option allows you to change the tuning parameters during runtime through Tuner Helper or APIs.
Raw-data noise filter	None	Filters will be enabled in step 5 depending on the SNR of the sensor.
Compensation IDAC	Enabled	Enabling the compensation IDAC selects the dual-IDAC mode operation of the CSD. Dual-IDAC mode gives higher signal values compared to single-IDAC mode as explained in <a href="#">Conversion Gain and CapSense Signal</a> .
Auto-calibration	Enabled	Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (85 percent in single-IDAC mode and 70 percent for dual-IDAC mode). For systems that may need a different calibration point because of environmental factors, see <a href="#">Modulator and Compensation IDACs</a> .
Waterproofing and detection	Unchecked	This parameter should be enabled when your design has a <a href="#">Guard Sensor</a> or <a href="#">Driven-Shield Signal and Shield Electrode</a> for <a href="#">Liquid Tolerance</a>
Enable built-in self-test	Enabled	This parameter is enabled to expose the C <sub>P</sub> measurement API. This can be used to manually set the sense and modulator clock values when needed.

Figure 5-46. CapSense Component Widgets Config Window

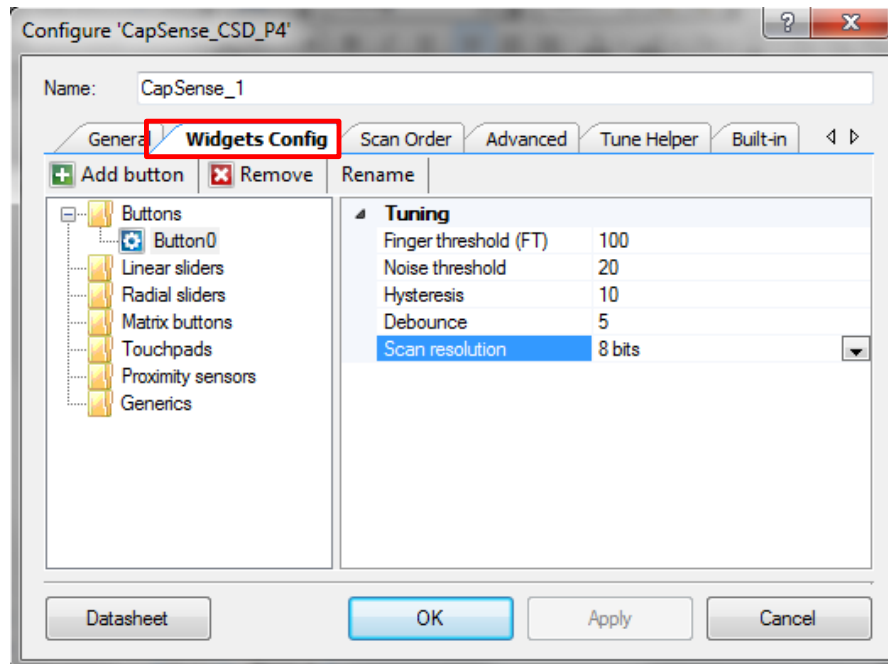


Table 5-11. CapSense Component Widgets Config Window

Parameter	Value	Rationale
Finger threshold (FT)	Default	Threshold settings will be adjusted in Step 8 of the tuning process.
Noise Threshold	Default	Threshold settings will be adjusted in Step 8 of the tuning process.
Hysteresis	Default	Threshold settings will be adjusted in Step 8 of the tuning process.
Debounce	Default	Threshold settings will be adjusted in Step 8 of the tuning process.
Scan resolution	8-bit	8-bits is a good starting point to ensure fast scan times, and some signal. This value will get adjusted as needed in step 5.



Figure 5-47. CapSense Component Scan Order Configuration Window

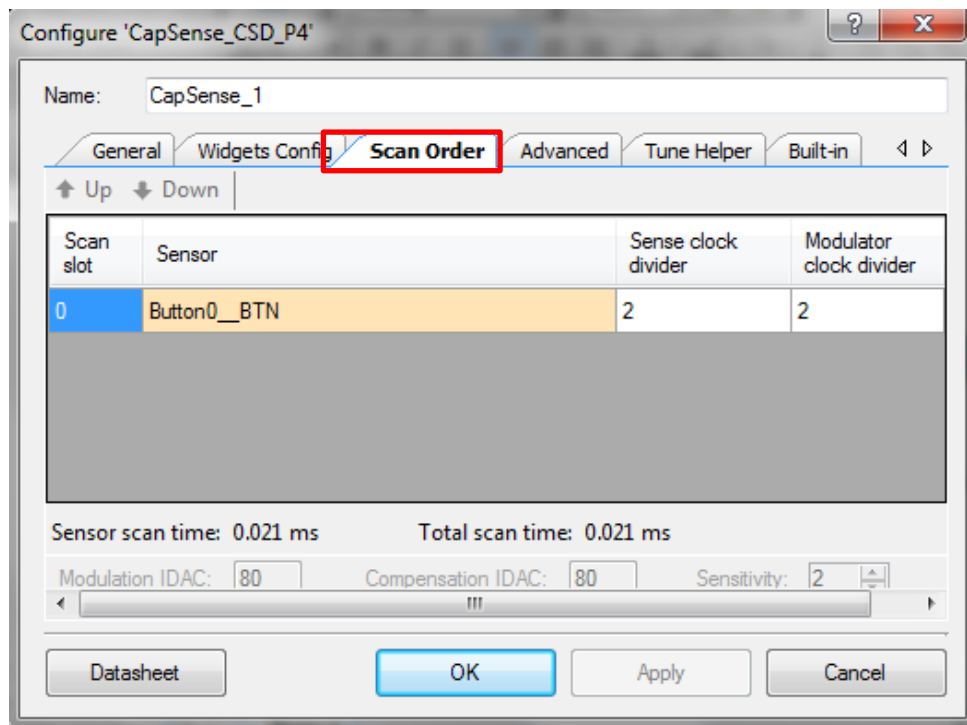


Table 5-12. CapSense Component Scan Order Configuration Window

Parameter	Value	Rationale
Sense clock divider	SmartSense**	Enter the value that was captured in step 1.
Modulator clock divider	SmartSense**	Enter the value that was captured in step 1.
Modulation IDAC	NA	With auto-calibration is enabled, the device automatically chooses this value. To choose a different IDAC value, see <a href="#">Modulator and Compensation IDACs</a> .
Compensation IDAC	NA	With auto-calibration is enabled, the device automatically chooses this value. To choose a different IDAC value, see <a href="#">Modulator and Compensation IDACs</a> .

Figure 5-48. CapSense Component Advanced Configuration Window

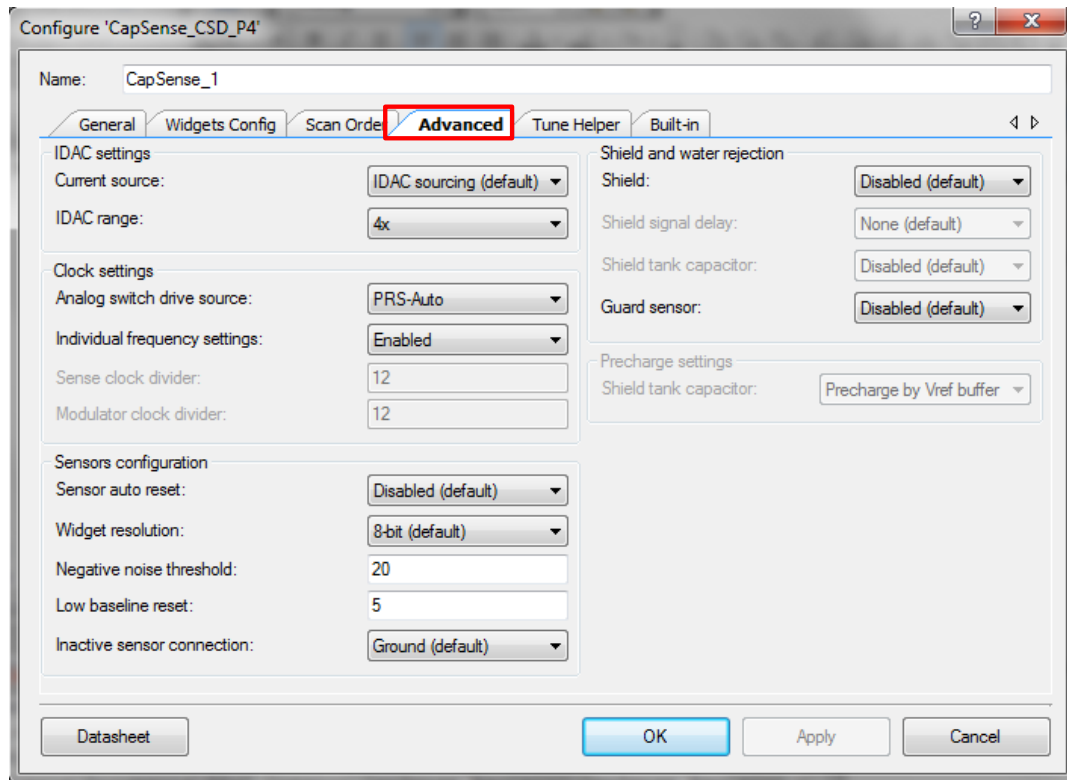


Table 5-13. CapSense Component Advanced Configuration Window

Parameter	Value	Rationale
Current source	IDAC sourcing (default)	The IDAC Sourcing mode is recommended for most applications, as it is free from power supply noise compared to the IDAC Sinking mode.
IDAC range	4x	The 4x range is sufficient for most applications. You can use the 8x range if Raw counts remain saturated at the max 4x IDAC value. For more information see the <a href="#">Modulator and Compensation IDACs</a> section.
Analog switch drive source	PRS-Auto	Enabling PRS helps to deal with EMC/EMI or <a href="#">Flat Spots</a> issues. Refer to <a href="#">Sense Clock</a> and <a href="#">Sense Clock Related Parameters</a> for more information on choosing Direct or PRS clocks.
Individual frequency settings	Enabled	This option is enabled to specify separate sense clock divider and modulator clock divider for each sensor in the <b>Scan order</b> tab. Disable this option if sense clock divider and modulator clock divider are found to be same for all sensors in step 1; disable this option otherwise.
Sensor auto reset	Disabled (default)	This option is set to “Disabled” to ensure that the baseline is not updated when a finger is touching the sensor. Refer to <a href="#">Sensor Auto Reset</a> section for more details. Enable this option if you have problems with sensors permanently turning on when the raw count suddenly rises without anything touching the sensor
Widget resolution	8-bit (default)	Widget resolution defines the byte-size of difference counts (or signal). For most designs, 8-bit will work fine. Details on when to enable 16-bit can be found in the Tuner Debug item (see <a href="#">5.3.9.6</a> ).
Shield	Per PCB design	Enable shield if your design requires <a href="#">Liquid Tolerance</a> , or if the shield is being used to reduce $C_p$ of sensors.
Guard sensor	Per PCB design	The guard sensor should be enabled when you need to eliminate false triggers due to liquid flow over the sensor. It can be disabled otherwise.
Inactive sensor connection	Ground(default)	Inactive sensors are connected ground to provide good shielding from noise sources. Use inactive sensor connection as shield for liquid tolerant designs or if your design contains a proximity sensor For additional information see the <a href="#">Liquid Tolerance</a> section and <a href="#">AN92239 Proximity Sensing with CapSense</a>

3. Next, use the tuner to observe raw counts in the Graphing tab in Tuner GUI and calculate the [Signal-to-Noise Ratio](#) of the sensor. To do this, measure the raw count value when a finger is present and divide it by the peak-to-peak noise of the raw counts with no touch present. 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 – 9 mm.
4. If the initial SNR is greater than 5, you can move to step 6. Otherwise, move to step 5.
5. When the SNR is less than 5, increase it to achieve proper performance. The main parameters that influence SNR are resolution and filters. Select an appropriate filter for your application based on [Table 5-4 Raw Data Noise Filters in CapSense v2.40](#).

Scan Resolution – Can be increased to increase signal at a disproportionate rate to noise to improve overall SNR. Adjusting resolution adds to the overall scan time based on [Equation 3-7](#).

Filters – Filters help to reduce noise, without increasing the signal. Based on your noise type you can enable a filter to improve SNR. Each filter will add additional processing time as well as memory use.

It is best to find a balance between the resolution 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 that resolution can be updated directly in the Sensor properties tab in Tuner GUI as [Figure 5-49](#) shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as [Figure 5-50](#) shows, and reprogram the device to update filter settings. For details on filters, see the [CapSense Component Datasheet](#).

Figure 5-49. Update Scan Resolution in Tuner GUI

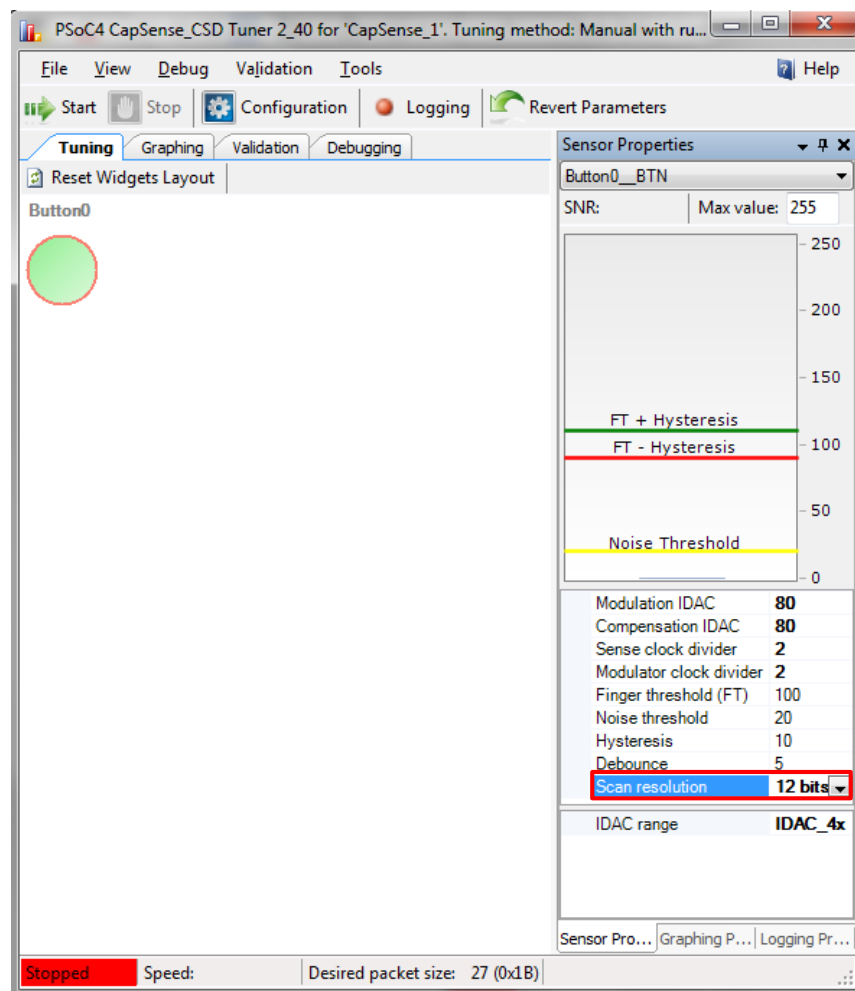
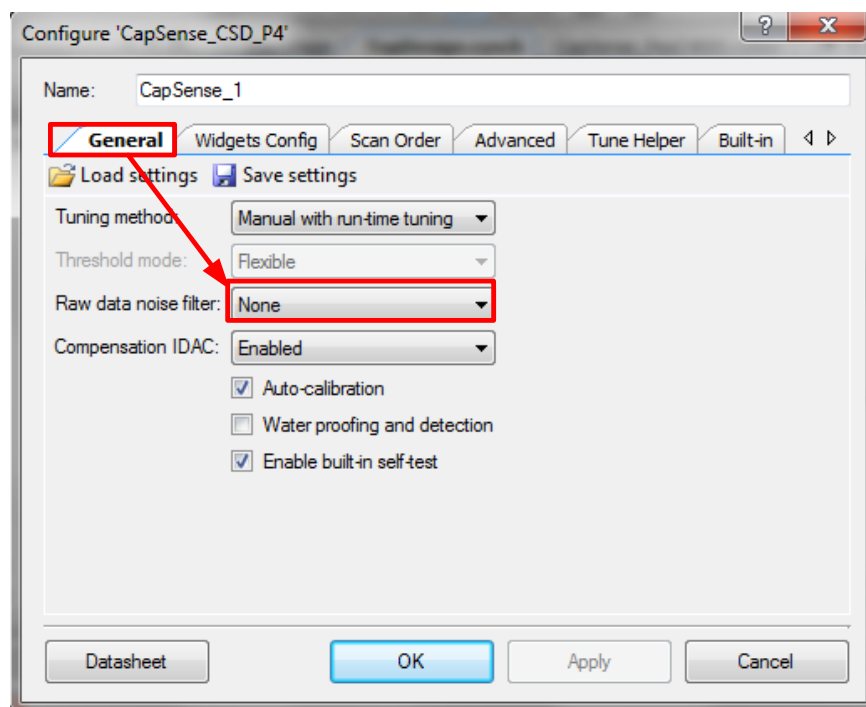
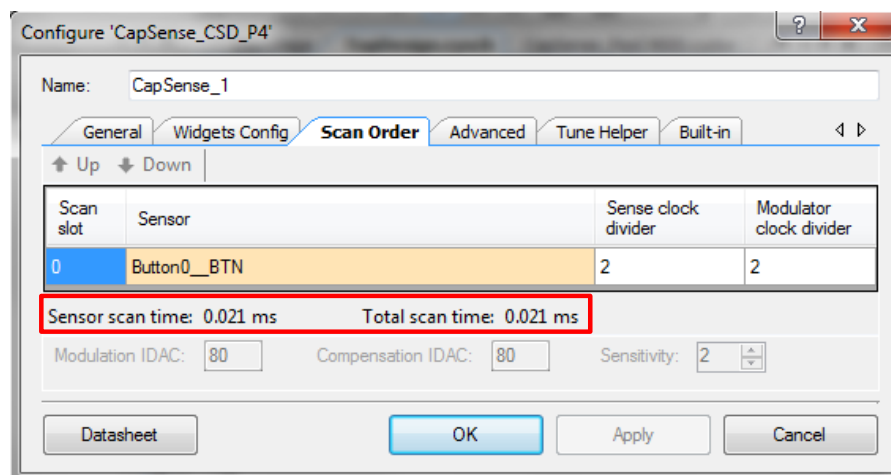


Figure 5-50. Update Filter Settings in Configure 'CapSense\_CSD\_P4' Dialog



6. Check the total scan time (see Figure 5-51) to determine if it meets the system requirements. This timing will impact the response time and be an important factor in the overall power consumption of the device in CapSense applications, as indicated in the section [Power Consumption and Response Time](#).

Figure 5-51. Sensor Scan Time in Scan Order Tab



7. If you meet the timing requirement of the system, skip to step 8. Otherwise, adjust the tuning to speed up the scan time. If SNR is greater than 10 on any sensor, then you can lower your resolution or remove filters to decrease scan time, but keep your SNR greater than 5. If you are unable to meet your timing requirements and maintain SNR > 5, you should see step 9.
8. After you have confirmed that your design meets the timing parameters, and the SNR is greater than 5, set your design thresholds as follows. Ensure that you observe the difference count (**signal** output) in graphing tab in Tuner GUI, *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 – 9 mm. Consider testing with smaller sizes that should be “rejected” by the system to ensure they do not reach the finger threshold

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

Again, these settings can be first set in the tuner GUI, as [Figure 5-52](#) shows, and saved when closing, or they can be input directly in the CapSense Component customizer as [Figure 5-53](#) shows.

For more information on these settings, refer to [Selecting CapSense Software Parameters](#).

Figure 5-52. Updating Threshold Parameters in Tuner GUI

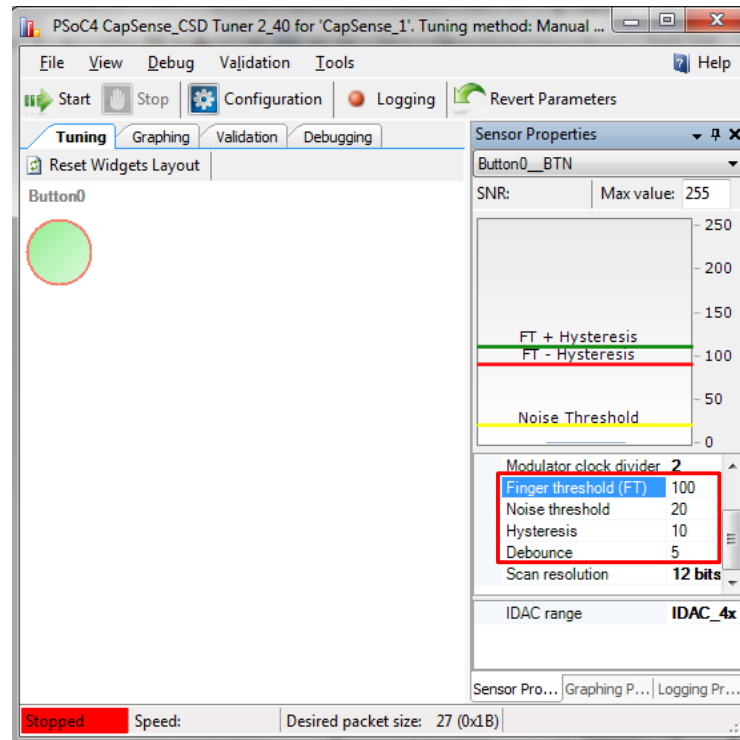
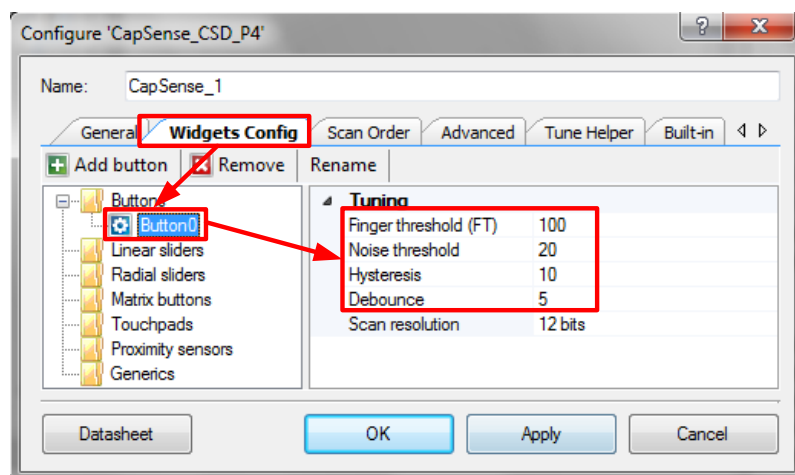
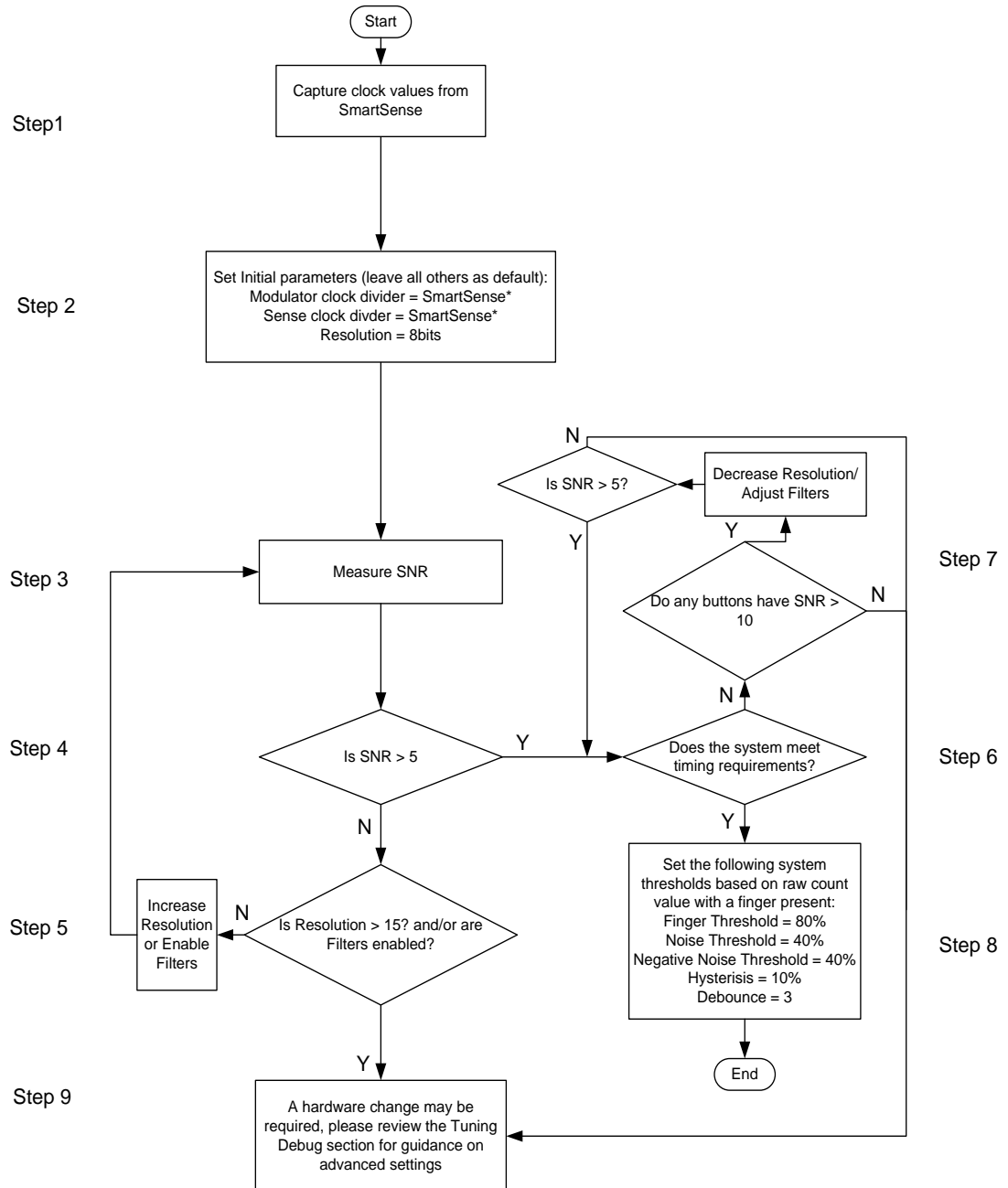


Figure 5-53. Updating Threshold Parameters in Configure 'CapSense\_CSD\_P4' Dialog



9. If you are not able to achieve an SNR of greater than 5 or cannot meet your timing requirements, refer to [Tuning Debug FAQs](#) or [Overview](#) for more information on how to tune your system. You may need to modify the advanced parameters of the CapSense Component and/or adjust your hardware design to meet the end system requirements.

Figure 5-54. Button Widget Tuning Flow for CapSense v2.40

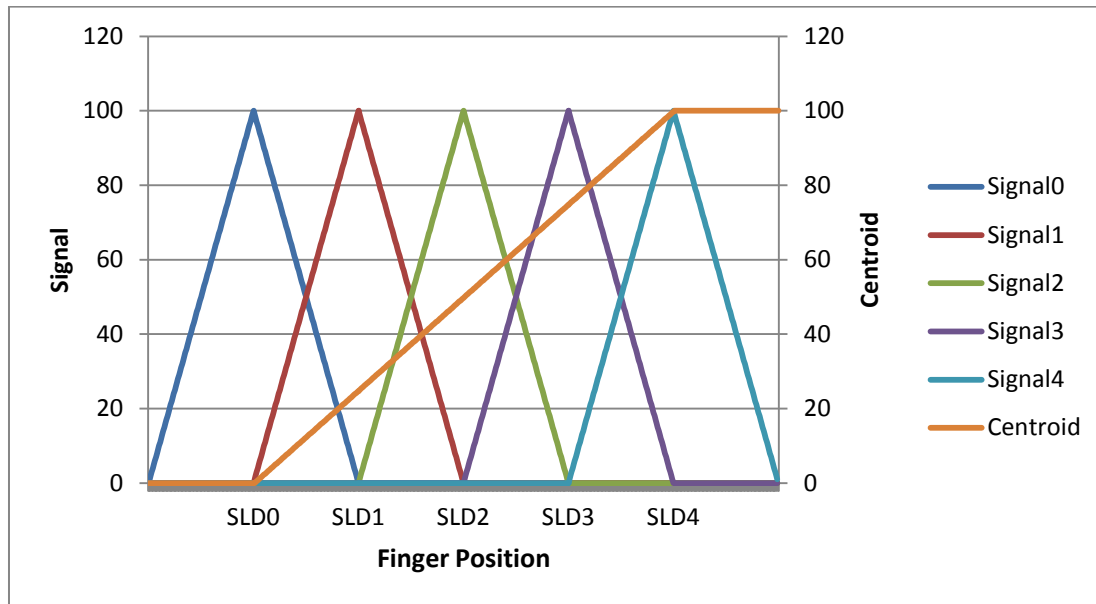


### 5.3.6.2 Slider Widget Example

A slider has many segments, each of which is connected to the CapSense input pins of the PSoC 4 and PSoC BLE devices. Unlike the simple on/off operation of a button widget sensor, slider widget sensors work together to track the location of a finger or other conductive object. Because of this, the slider layout design should ensure that the  $C_P$  of all the segments in a slider remain as close as possible. Keeping similar  $C_P$  values between sensors will help minimize the tuning effort and ensure an even response across the entire slider.

1. To avoid nonlinearity in the centroid, ensure that the signal from all the slider segments is equal, as [Figure 5-55](#) shows. Here, the signal for each slider segments is the shift in Raw Count minus the [Noise Threshold](#), when a finger is placed at the center of the slider segment. If the signal of the slider segments is different, then the centroid will be nonlinear, as [Figure 5-56](#) shows.

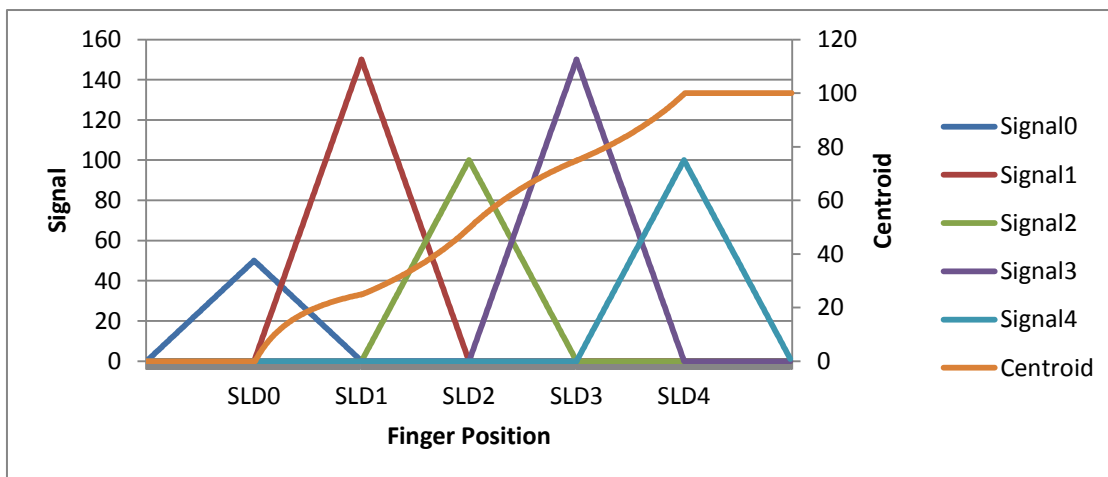
Figure 5-55. Response of Centroid versus Finger Location when Signals of All Slider Elements Are Equal



**Note** Signal = Difference count – Noise Threshold

Difference count = Raw Count - Baseline

Figure 5-56. Response of Centroid versus Finger Location when the Signal of All Slider Elements Are Different



Both CapSense v2.40 and CapSense v3.0 Components support CSD-based slider widgets. Use the following steps to manually tune the slider segments to achieve an equal signal for all slider segments:

1. Use of IDAC auto-calibration (as recommended in later steps) requires that the  $C_P$  of other segments in the slider should be greater than 75 percent of the  $C_P$  of the segment with the maximum  $C_P$  value in that slider. For example, in a slider, if 30 pF is the maximum  $C_P$  of a segment, the  $C_P$  of other segments should be greater than 22.5 pF. Ensure that your hardware meets this requirement.

You can measure the  $C_P$  of the slider segments by using the CSD Built-in-Self-test (BIST) API; `CapSense_GetSensorCp()` provided with CapSense Component v2.40. More information on this API can be found in the [Component Datasheet](#).

2. Follow the manual tuning procedure shown in [Figure 5-38](#) for CapSense v3.0 or [Figure 5-54](#) for CapSense v2.40 and tune the slider segment, which has the maximum  $C_P$  value among all the slider segments.
3. In CapSense v2.40, for the remaining segments in the slider, set the same value as that of the maximum  $C_P$  slider segment for the following parameters:
  - Scan resolution
  - Sense clock divider
  - Modulator clock divider
  - Modulation IDAC
  - Compensation IDAC

Setting the same values for these parameters for all segments will ensure that the sensitivity (change in the raw count for a given change in  $C_F$ ) is the same for all sensors.

In CapSense v3.0, the above parameters are widget-level parameters – they need not be set individually for each segment as is the case with CapSense v2.40.

4. Measure the [Signal-to-Noise Ratio](#) for each slider segment and ensure that it is greater than 5:1 for all the slider segments. If the SNR is not greater than 5:1 for any slider segment, increase the resolution by one bit for all slider segments, until the SNR is greater than 5:1 for all segments.
5. After confirming that the SNR for all slider segments is greater than 5:1, set the following thresholds to the value listed in [Table 5-14](#).

Table 5-14. Threshold Parameter Values

Threshold Parameter	Recommendation
Finger Threshold	80 percent of the signal
Noise Threshold	40 percent of the signal
Negative Noise Threshold	40 percent of the signal
Low-Baseline Reset	Set to 30

### 5.3.6.3 Proximity Widget Example

For tuning a proximity sensor, refer to [AN92239 - Proximity Sensing with CapSense](#).



## 5.3.7 CSX sensing method

This section explains the basics of manual tuning using CSX sensing method. It also explains the hardware parameters that influence a manual tuning procedure. The section ends with an example on manual tuning of a button widget.

### 5.3.7.1 Basics

CapSense component v3.0 and later support Mutual-Capacitance sensing as well. In a mutual-capacitance sensing system, the raw count is directly proportional to the mutual capacitance between the Tx and Rx sensors, as [Equation 5-14](#) shows.

Equation 5-14. Raw Count Relationship to Sensor Capacitance

$$\text{raw count} = G_{\text{CSX}} C_{\text{M}}$$

Where  $G_{\text{CSX}}$  is the capacitance to digital conversion gain of CapSense CSX and  $C_{\text{M}}$  is the mutual capacitance between the Tx and Rx electrodes. The approximate value of this conversion gain is:

Equation 5-15. Capacitance to Digital Converter Gain

$$G_{\text{CSX}} = 2 * (2^N - 1) \frac{V_{\text{TX}} F_{\text{SW}}}{I_{\text{DAC}}}$$

The value of  $V_{\text{TX}}$  is equal to the  $V_{\text{DDIO}}$  or  $V_{\text{DDD}}$  (If  $V_{\text{DDIO}}$  is not available). The tunable parameters of the conversion gain are,  $F_{\text{SW}}$ ,  $I_{\text{DAC}}$  and  $N$ .

Where,  $F_{\text{SW}}$  is the Tx clock frequency,  $I_{\text{DAC}}$  is the current charging and discharging the  $C_{\text{INT}}$  capacitors and  $N$  is the counter resolution.

**Note** The value of  $N$  is controlled by the Number of Conversions parameter, as explained in the [CSX Sensing Method](#) section.

The change in raw counts when a finger is placed on the sensor is called CapSense signal. The signal can be increased by increasing  $N$  and  $F_{\text{SW}}$  and decreasing  $I_{\text{DAC}}$  value.

### 5.3.7.2 Selecting CapSense Hardware Parameters

CapSense hardware parameters govern the conversion gain and CapSense signal. [Table 5-15](#) lists the CapSense hardware parameters that apply to CSX sensing method. The following section gives guidance on how to adjust these parameters to achieve optimal performance for your particular system.

Table 5-15. CapSense Component Hardware Parameters

Sl. No.	CapSense Component Parameter Name	
	CapSense v3.0	CapSense v2.40
1	Modulator Clock Frequency	Not supported by CapSense 2.40
2	Tx Clock Source	
3	Tx Clock Frequency	
4	Number of Conversions	

#### 5.3.7.2.1 Modulator Clock Frequency

The modulator clock frequency will impact sensor scan time as well as [Signal-to-Noise Ratio](#). In general, it is recommended to choose the highest modulator clock frequency. This will give the best possible scan times as well as reduce measurement error as much as possible.

The relation between the modulator clock and duration of sensor scan is given by equation [Equation 3-14](#).

### 5.3.7.2.2 Tx Clock Source

The Tx clock source parameter is only applicable for the PSoC 4 S-Series of devices. The following clock sources are available for Tx clock:

- Direct – Clock signal with 50 percent duty cycle.
- Spread spectrum clock – The clock signal frequency is dynamically spread over a predetermined range to reduce EMI.
- Auto – The spread spectrum range is automatically selected such that the maximum clock dither is limited to  $\pm 10$  percent and ratio of 160 or more is maintained between HFCLK and SenseClk, and at least one full spread spectrum polynomial executed within a sensor scan interval.

The recommended setting is “Auto” for robust tuning and performance.

### 5.3.7.2.3 Tx Clock Frequency

The Tx clock frequency determines the duration of each sub-conversion as explained in the [CSX Sensing Method](#) section. According to [Equation 3-14](#), it is recommended to use the maximum clock frequency available in the component drop-down list to achieve maximum sensitivity.

### 5.3.7.2.4 Number of Conversions

The number of conversions parameter decides the sensitivity of the sensor and is related to the raw count counter resolution as:

$$\left( \frac{\text{Modulator Clock}}{\text{Tx Clock}} * \text{Number of Conversions} \right) < 2^{16}$$

Where,  $2^{16}$  is the maximum counter value

Increasing the number of conversions increases the signal and scan time of the sensor i.e. for a fixed modulator clock and Tx clock, the above equation shows that the resolution of the CapSense raw counter is directly proportional to the number of conversions parameter. Reducing the number of conversions by half, reduces the counter resolution by 1-bit and reduces the scan time by half which in turn reduces the sensor signal as shown in [Equation 3-13](#). Refer to [Button Widget Example](#) section for details on how to configure optimum value for “number of conversions” parameter.

### 5.3.7.3 Selecting CapSense Software Parameters

CapSense software parameters for mutual capacitance are the same as that for self-capacitance, and hence these can be selected in the same way. Refer to [Selecting CapSense Software Parameters](#) in CSD sensing method for more details on these parameters.

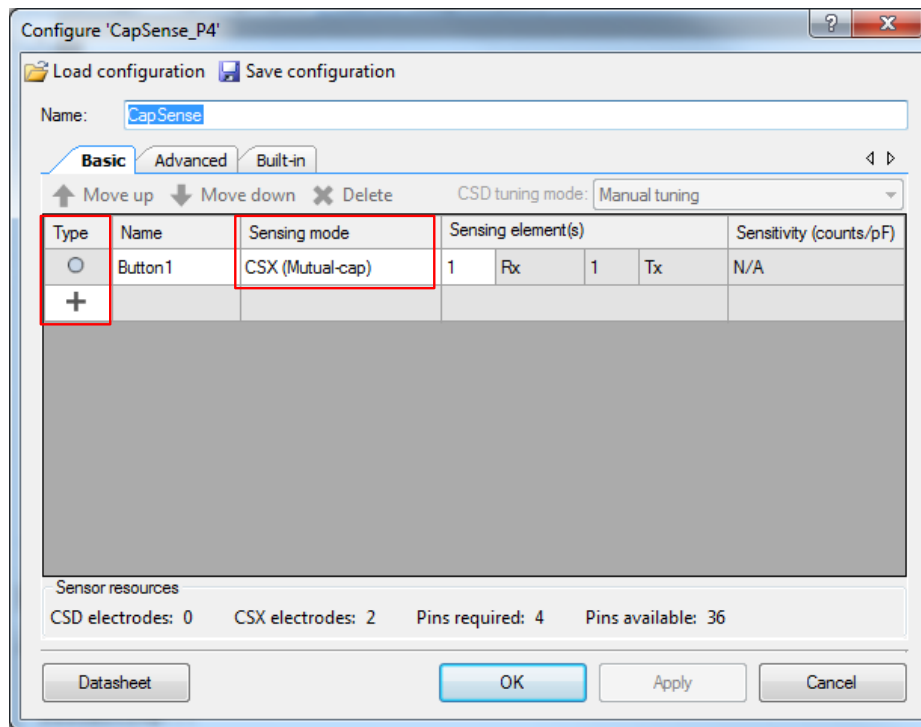
### 5.3.7.4 Button Widget Example

This example explains tuning of mutual-capacitance-based button widget in a PSoC Creator CapSense Component using [Tuner Helper](#). For details on the Component and all related parameters, refer to the [Component Datasheet](#).

Follow the detailed process listed in the following steps to manually set all the tuning parameters. See [Figure 5-64](#) for a quick reference flow chart.

1. Open the CapSense Component configuration window by double-clicking the Component or right-clicking on the Component and selecting the “Configure” option.
2. In the Basic tab, add the button widget by clicking on the + symbol and select the **Sensing mode** as **CSX (Mutual-cap)**, as [Figure 5-57](#) shows.

Figure 5-57. Adding CSX Button Widget



3. In the **Advanced** tab - **General** settings windows, leave all the filter parameters at their default settings. Filters will be enabled depending on the SNR and response time requirements.
4. In the **Advanced** tab - **CSX Settings** window, specify the parameters settings as shown in Figure 5-58.
5. In the **Advanced** tab - **Widget Details** window, specify the parameter settings as shown in Figure 5-59.

Figure 5-58. CSX Widget Settings

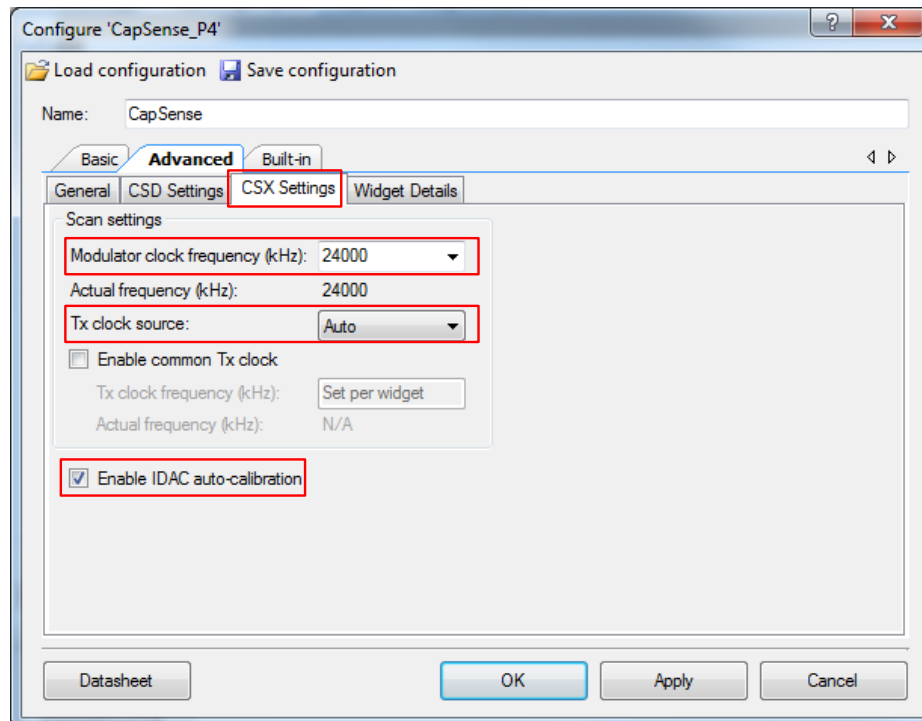


Table 5-16. CapSense Component General Configuration Window

Parameter	Value	Rationale
Modulator clock frequency	Maximum available option	Higher modulator clock frequency reduces sensor scan time and therefore results in lower power, and lower noise in the raw counts; hence it is recommended to use the highest possible frequency.
Tx clock source	Auto	This option is available only for PSoC 4 S-Series of devices. For other devices, the Tx clock is always direct. Enabling Auto (chooses optimum value of spread spectrum clock spread) helps to deal with EMC/EMI. Refer to the <a href="#">Tx Clock Source</a> section for more information on choosing direct or SSC clocks.
Enable IDAC auto-calibration	Enabled	Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (for CSX this is 40 percent of maximum value).

Figure 5-59. CapSense Component Widget Details Window

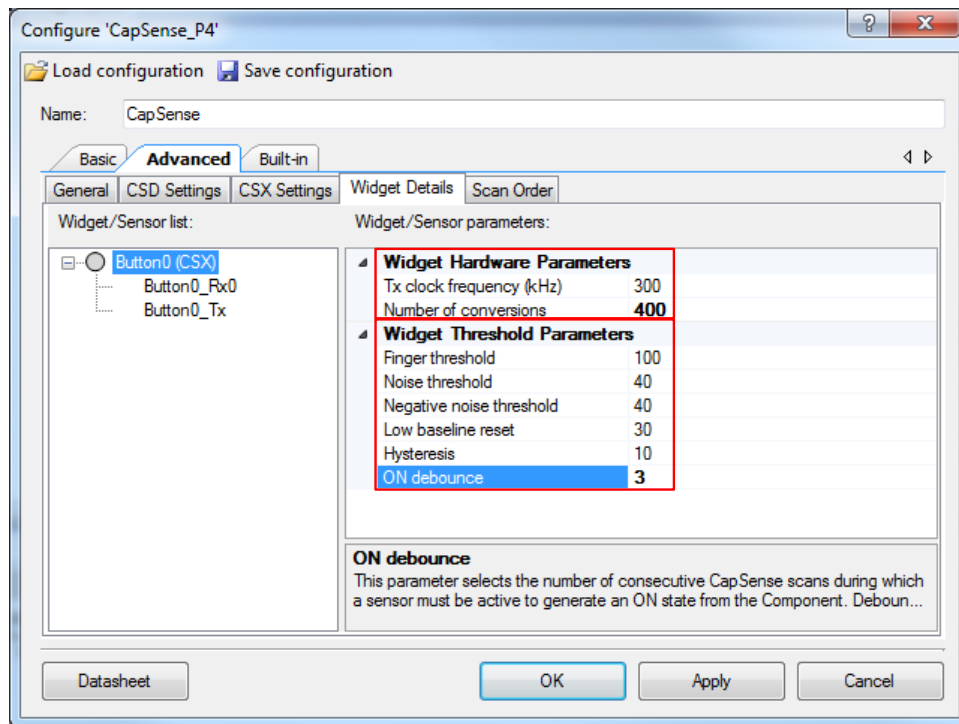


Table 5-17. CapSense Component Widget Details Window

Parameter	Value	Rationale
Tx clock frequency	$\frac{1}{10R_{SeriesTotal}C_p}$	It is recommended to set the highest clock frequency that satisfies the condition $Tx\ Clock\ Freq < \frac{1}{10R_{SeriesTotal}C_p}$ The Tx pin can be probed using oscilloscope to check if the sensor is completely charged and discharged.
Number of conversions	$< \frac{1}{2} \left\{ \frac{2^{16} * Tx\ Clock}{Modulator\ Clock} \right\}$	It is good to start with half the maximum achievable sensitivity. This value will be adjusted as needed in Step 8.
Finger threshold (FT)	Default	Threshold settings will be adjusted in Step 10 of the tuning process.
Noise Threshold	Default	Threshold settings will be adjusted in Step 10 of the tuning process.
Negative Noise Threshold	Default	Threshold settings will be adjusted in Step 10 of the tuning process.

Parameter	Value	Rationale
Low baseline reset	Default	Threshold settings will be adjusted in Step 10 of the tuning process.
Hysteresis	Default	Threshold settings will be adjusted in Step 10 of the tuning process.
ON Debounce	Default	Threshold settings will be adjusted in Step 10 of the tuning process.

- Next, use the tuner to observe raw counts in the Graph View tab in Tuner GUI and calculate the [Signal-to-Noise Ratio](#) of the sensor. To do this, measure the raw count value when a finger is present and divide it by the peak-to-peak noise of the raw counts with no touch present. 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 – 9 mm.
- If the initial SNR is greater than 5, you can move to step 9. Otherwise, move to step 8.
- When the SNR is less than 5, increase it to achieve proper performance. The main parameters that influence SNR are number of conversions and filters. Select an appropriate filter for your application based on [Table 5-2](#).

Number of conversions – Can be increased to increase signal at a disproportionate rate to noise to improve overall SNR. Increasing number of conversions increases the resolution of the counter and increases overall scan time.

Filters – Filters help to reduce noise, without increasing the signal. Based on your noise type you can enable a filter to improve SNR. Each filter will add additional processing time as well as memory use.

It is best to find a balance between the number of 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 that number of conversions can be updated directly in the Widget/Sensor Parameters tab, as [Figure 5-60](#) shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as [Figure 5-61](#) shows, and reprogram the device to update filter settings. For details on filters, see the [CapSense Component Datasheet](#).

Figure 5-60. Update Number of Conversions in Tuner GUI

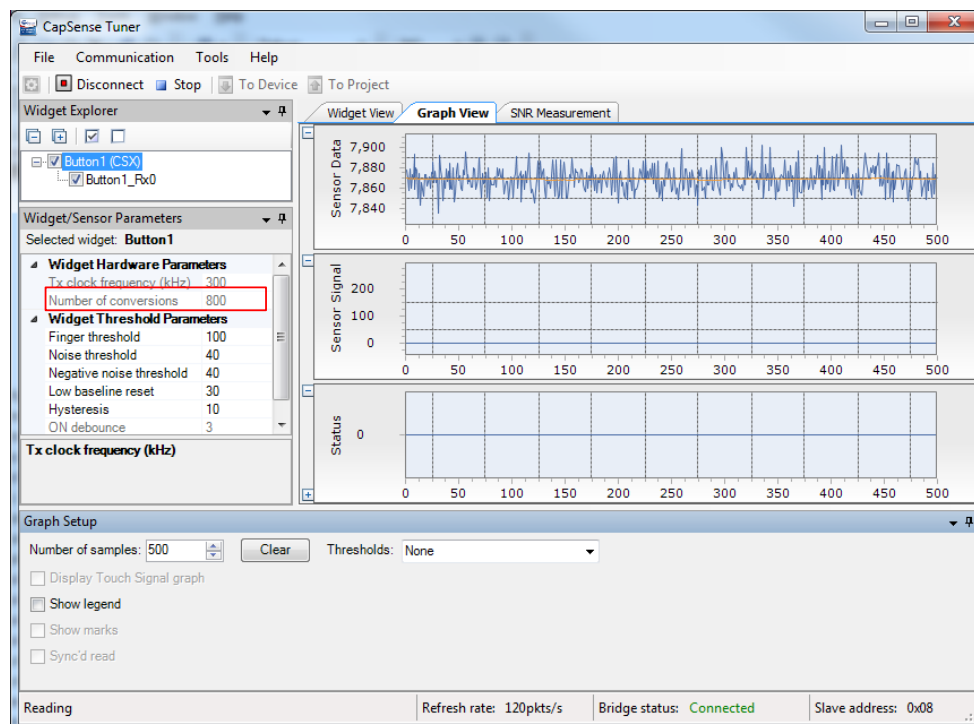
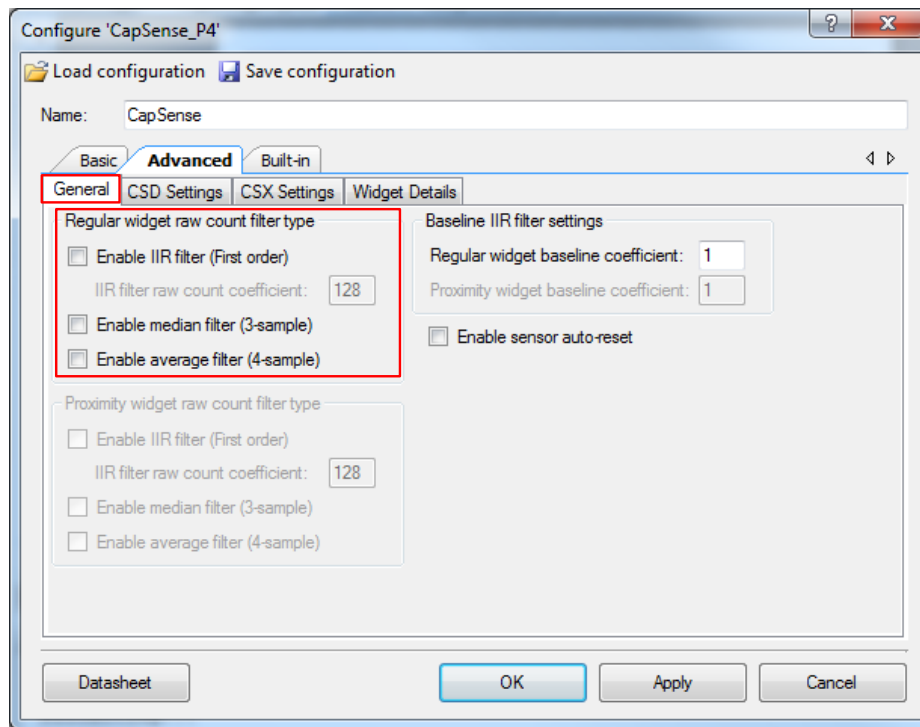


Figure 5-61. Update Filter Settings in Configure 'CapSense\_P4' Dialog



9. If the total sensor scan time meets your requirements, skip to step 10. Otherwise, adjust the tuning to speed up the scan time. If SNR is greater than 10 on any sensor, then you can lower your number of conversions or remove filters to decrease scan time, but keep your SNR greater than 5.
10. After you have confirmed that your design meets the timing parameters, and the SNR is greater than 5, set your design thresholds as follows. Ensure that you observe the difference count (that is **signal** output) in Graph View tab in Tuner GUI, *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 ~8-9 mm. Consider testing with smaller sizes that should be “rejected” by the system to ensure they do not reach the finger threshold. 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

These settings can be first set in the tuner GUI, as [Figure 5-62](#) shows, and saved by clicking on **Apply to Project** in the **File** menu of Tuner; or they can be input directly in the CapSense Component customizer as [Figure 5-63](#) shows. For more information on these settings, refer to [Selecting CapSense Software Parameters](#).

Figure 5-62. Updating Threshold Parameters in Tuner GUI

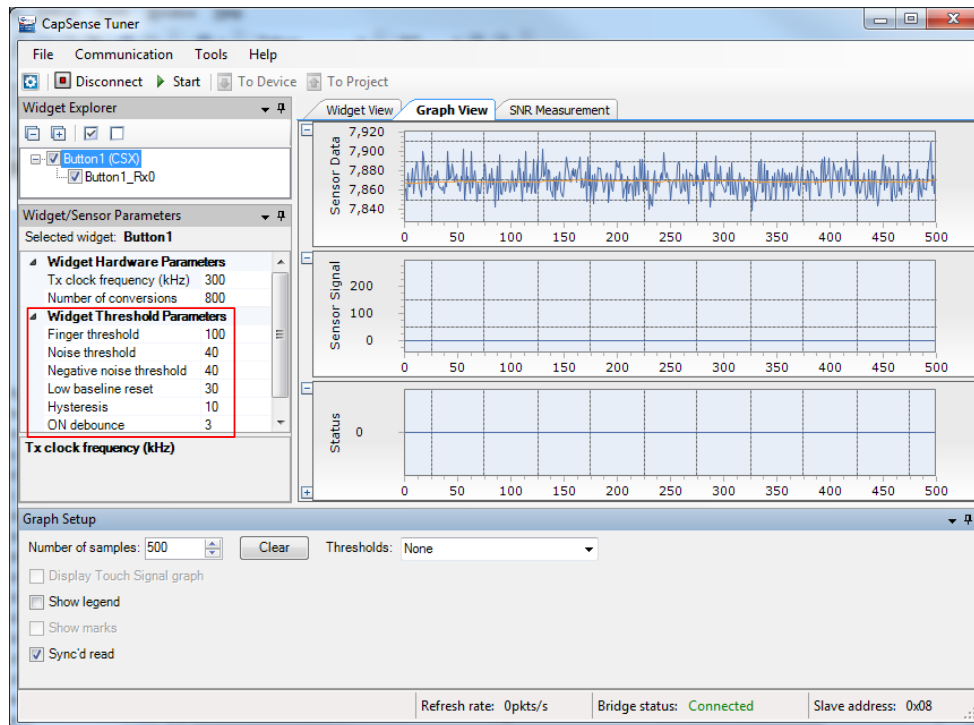


Figure 5-63 Updating Threshold Parameters in Configure 'CapSense\_CSD\_P4' Dialog

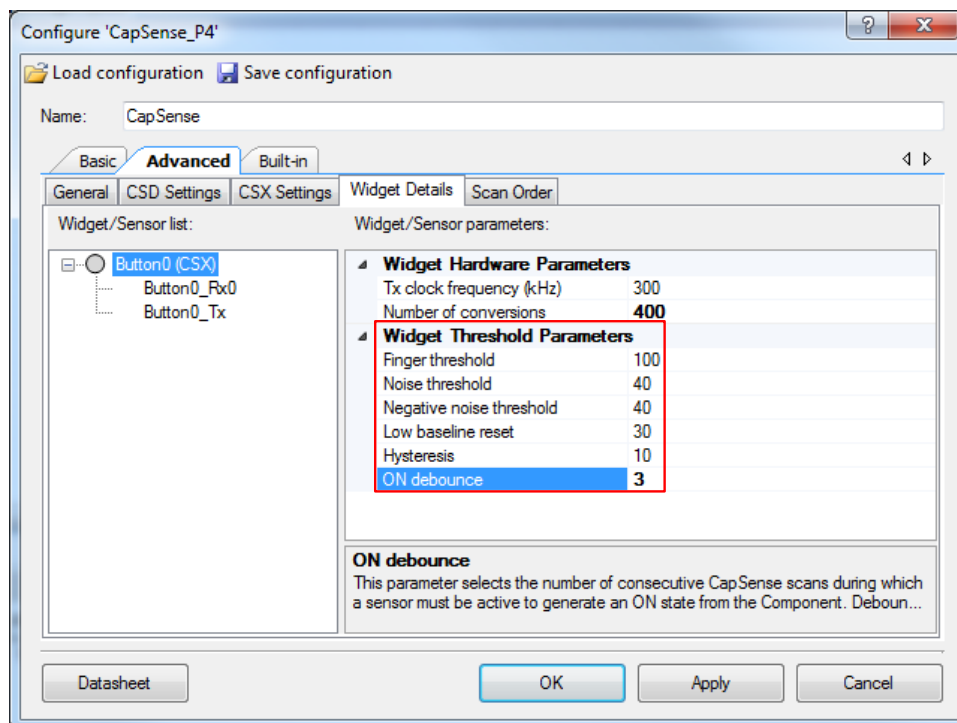
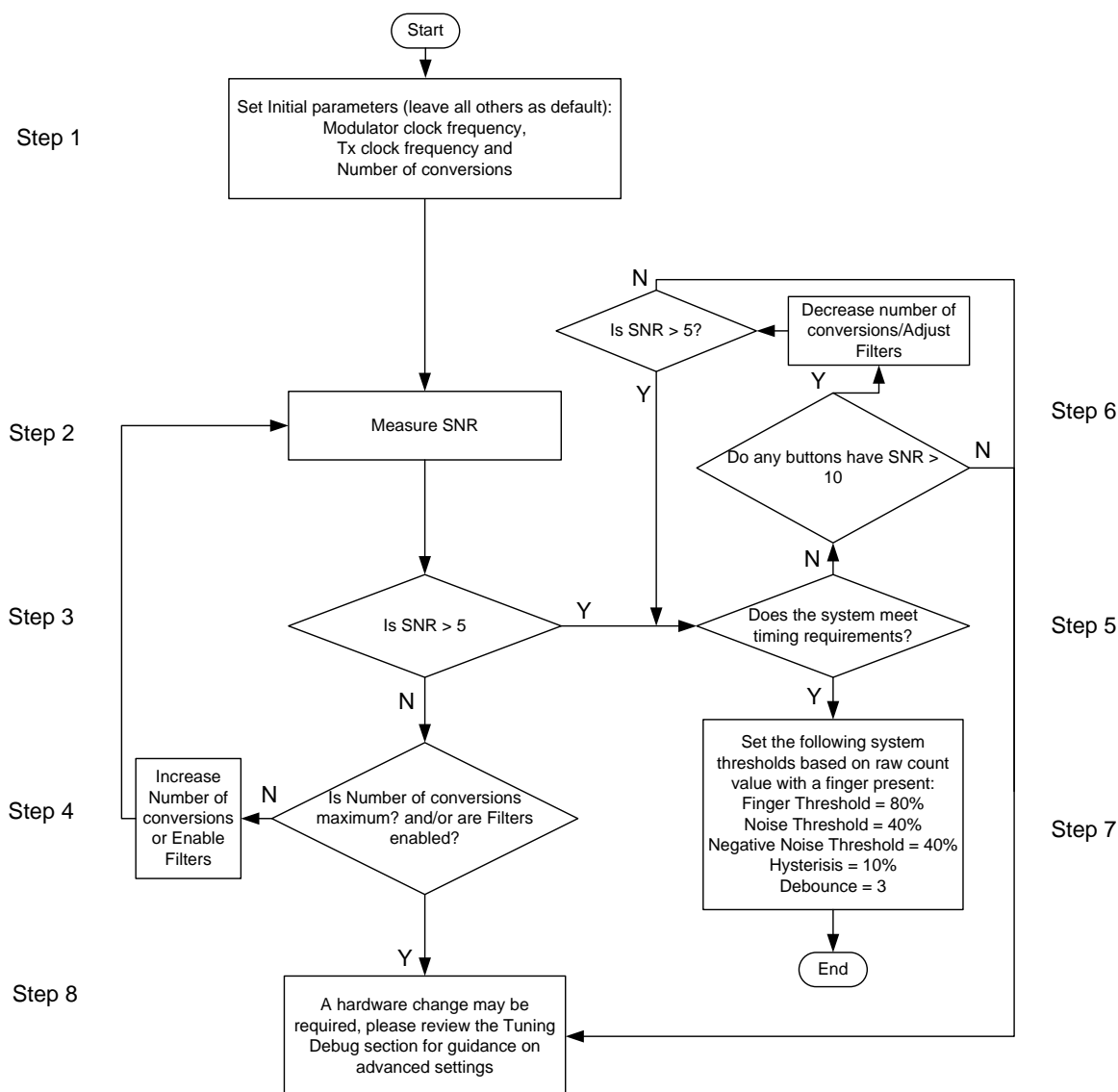


Figure 5-64. CSX Button Widget Tuning Example



### 5.3.8 Manual Tuning Trade-offs

When manually tuning a design, it is important to understand how the settings impact the characteristics of the capacitive sensing system. Any CapSense design has three major performance characteristics: reliability, power consumption, and response time.

- **Reliability** defines how CapSense systems behave in adverse conditions such as a noisy environment or in the presence of water. High-reliability designs will avoid triggering false touches, and ensure that all intended touches are registered in these adverse conditions.

- **Power Consumption** is defined as the average power drawn by the device, which includes, scanning, processing, and low-power mode transitions as explained in [Low-Power Design](#). Quicker scanning and processing of the sensors ensures that the device spends less time in a higher power state and maximizes the time it can spend in a lower power sleep state.

- **Response Time** defines how much time it takes from the moment a finger touches the sensor until there is a response from the system. Because the lowest response time is limited by the scan and processing time of the sensors, it is important to properly define and follow a timing budget. A good target for total response time is below 100 ms.

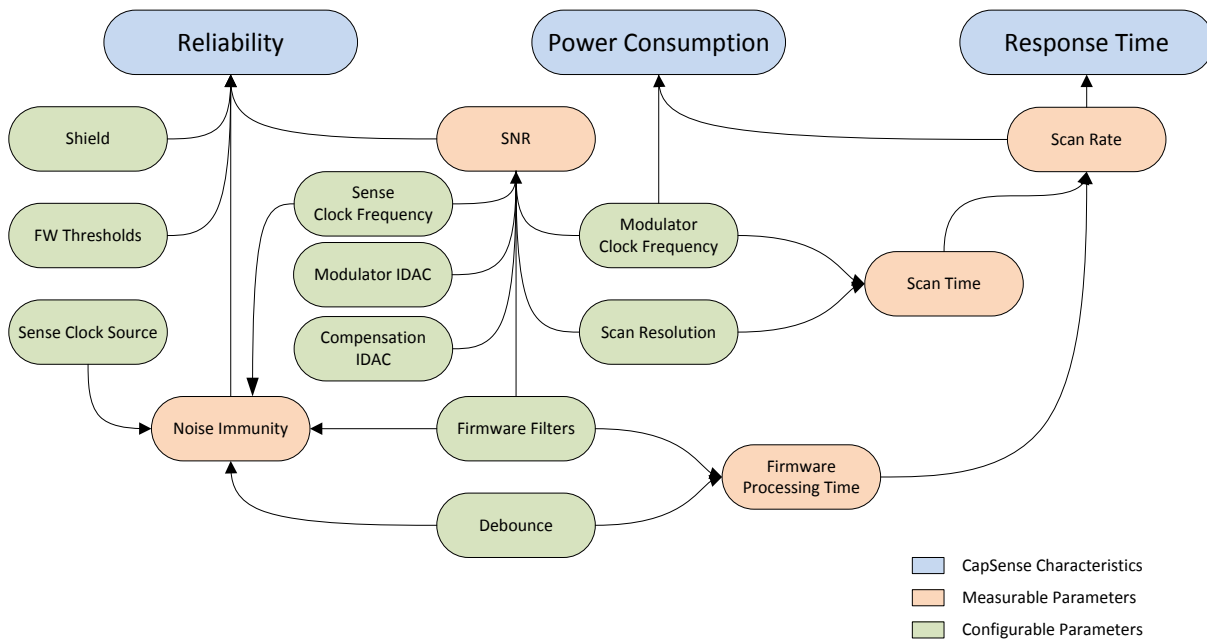


All of these characteristics depend on each other. The purpose of the tuning process is to find an optimal ratio that satisfies the project's specific requirements. When planning a design, it is important to note that these characteristics usually have an inverse relationship. If you take action to improve one characteristic, the others will degrade.

For example, if you want to use CapSense in a toy, it is more important to have a quick response time and low power consumption. In a different example, such as a "Start/Stop" button for an oven, reliability is the most important characteristic and the response time and power consumption are secondary.

Now let us consider the factors that affect each characteristic. The following figure shows dependencies between CapSense characteristics, measurable parameters, and actual CapSense configurable parameters.

Figure 5-65. CapSense Parameter Relationships



### 5.3.8.1 Reliability

The following factors affect reliability:

#### 1. Signal-to-Noise Ratio (SNR):

SNR gives a measure of confidence in a valid touch signal. For reliable CapSense operation it should be greater than 5. Manual tuning can ensure optimal SNR in specific designs.

#### 2. Noise Immunity:

It is the ability of the system to resist external or internal noise. Typical examples of external noise are ESD events, RF transmitters such as BLE, switching relays, power supply and so on. The internal noise source could be an LED driven by PWM, or I2C or SPI communications for example. Even designs with good SNR may suffer from poor performance because of poor noise immunity. Manual tuning allows to tune frequencies and parameters to help avoid noise interference by allowing more control over selection of different parameters.

### 5.3.8.2 Power Consumption and Response Time

The following factors affect the power consumption and response time:

#### 1. Scan Rate

Scan rate can be defined as the frequency at which you scan the sensor. Scan rate decides the minimal possible time from the finger touch until it is reported. The maximum scan rate will be limited by the [Sensor Scan Time](#).

## 2. Scan Time

It is the time taken to scan and process a particular sensor. It affects power consumption as indicated in [Low-Power Design](#) and scan rate as indicated above. Manual tuning can achieve specific scan durations while maintaining a minimum SNR.

## 3. Firmware Touch Delay

This can be caused by the [Debounce](#) procedure or use of Raw Data Noise Filters (see [Table 5-2. Raw Data Noise Filters in CapSense v3.0](#) or [Table 5-4 Raw Data Noise Filters in CapSense v2.40](#) depending on the CapSense component version you are using). Both of these affect scan time by adding to the processing time of a sensor and also delay the touch reporting until a certain number of samples in a row show the touch signal.

In both cases response rate is reduced, but reliability is usually improved.

The following sections provide typical examples for how to tune the CapSense CSD parameters in PSoC Creator. These can be used along with sections [Overview](#), [Selecting CapSense Hardware Parameters](#), AND [Selecting CapSense Software Parameters](#) to achieve optimal manual tuning for your design.

## 5.3.9 Tuning Debug FAQs

This section lists the general debugging questions on CapSense Component tuning. Jump to the question you have, for quick information on various causes and solution for your debugging topic.

### 5.3.9.1 The tuner does not communicate with the device

**Cause 1:** Your device is not programmed.

Solution 1: Make sure to [program](#) your device with your latest project updates before launching the tuner.

**Cause 2:** The tuner configuration setting does not match the SCB Component setting.

Solution 2: Open the EzI2C slave component configuration window, that is, the Configure 'SCB\_P4' dialog and verify that the settings match the configuration of the Tuner Communication Setup dialog. Refer to the [CapSense Component datasheet](#) for details on tuner usage.

**Cause 3:** Your I2C pins are not configured correctly.

Solution 3: Open the .cydwr file in Workspace Explorer and ensure the pin assignment matches what is physically connected on the board.

**Cause 4:** You did not include the CapSense TunerStart API or other required tuner code.

Solution 4: Add the tuner code listed in [CapSense Component](#) datasheet to your main.c and reprogram the device.

### 5.3.9.2 I am unable to update parameters on my device through the tuner

**Cause 1:** Your communications settings on the device are incorrect.

Solution 1: Review and make sure the settings in the Configure SCB\_P4 dialog and Tuner Communication Setup dialog match. In particular make sure that the sub-address size is equal.

### 5.3.9.3 I can connect to the device but I do not see any raw counts

**Cause 1:** You did not add the tuner code to your project.

Solution 1: Review the [Tuner Helper](#) section and add the tuner code to your main.c and reprogram the device.

### 5.3.9.4 Difference counts only change slightly (10 to 20 counts) when a finger is placed on the sensor

**Cause 1:** The gain of your system is too low.

Solution 1: Review the [Tuner Helper](#) section of the document.

**Cause 2:** Your sensor parasitic capacitance is very high.

Solution 2: To confirm this issue, use the Built-in Self-Test (BIST) APIs documented in the [Component Datasheet](#). These functions allow you to read out an estimate of the sensor parasitic capacitance. You can also confirm this reading independently with an LCR meter.

If your hardware has an option to enable [Driven-Shield Signal and Shield Electrode](#), use this option in the advanced settings of the CapSense Component configuration window. A driven shield around the sensors

helps reduce the parasitic capacitance. When you enable this option, you may want to enable driving the shield to unused sensors by also changing the “Inactive Sensor connection” setting to “shield” in the advanced settings. If after enabling the shield, your  $C_P$  remains greater than 45 pF, contact Cypress [Technical Support](#) to review your layout.

**Cause 3:** Your overlay may be too thick.

Solution 3: Review your [Overlay Thickness](#) with respect to your [Overlay Material](#).

**Cause 4:** Raw counts may be too close to saturation and hence, saturating when sensor is touched.

Solution 4: Tune IDAC to ensure that raw counts are tuned to ~85 percent of the resolution for a given sensor according to the [Electromagnetic Compatibility \(EMC\) Considerations](#) section.

#### 5.3.9.5 After tuning the system, I see large amount of radiated noise during testing

**Cause 1:** The sense clock frequency is causing radiated noise in your system.

Solution 1: Reduce the sense clock frequency or enable PRS for your sensor based on section [Electromagnetic Compatibility \(EMC\) Considerations](#). If it is already enabled, refer to the [Electromagnetic Compatibility \(EMC\) Considerations](#) section.

**Cause 2:** Large shield electrode may be contributing to a large radiated noise.

Solution 2: Reduce the size of shield electrode based on [Layout Guidelines for Liquid Tolerance](#).

#### 5.3.9.6 My difference count always max out at 255 with the slightest touch

**Note:** This tuning debug item is only applicable to CapSense v2.40 as the difference count is always a 16-bit number in CapSense v3.0.

**Cause 1:** Your sensor is too sensitive.

Solution 1: If your sensor is too sensitive, that is, the [Signal-to-Noise Ratio](#) is greater than 5:1, try reducing the resolution.

**Cause 2:** You need more than 8-bit widget resolution for your sensor signal.

Solution 2: Change you widget resolution from 8-bit to 16-bit in the CapSense Component Advanced Configuration Window. This will allow your maximum difference count to go beyond 255.

#### 5.3.9.7 My scan time no longer meets system requirements after manual tuning

**Cause:** The noise and  $C_P$  of your system are high, which requires more scan time and filtering to achieve reliable operation.

Solution:  $C_P$  needs to be reduced. First, enable the [Driven-Shield Signal and Shield Electrode](#) in the advanced settings of the CapSense Component configuration window and ensure gain is set as high as possible by reviewing the [PCB Layout Guidelines](#). If your system still cannot meet final requirements, you may need to change your board layout to reduce  $C_P$  further, review the [PCB Layout Guidelines](#) for the same.

#### 5.3.9.8 I am unable to calibrate my system to 85 percent

**Cause:** Your sensor may have a short to ground or a very high  $C_P$ .

Solution: First, use a multimeter to check if there is a short between your sensor and ground. If it is present, review your schematic and layout for errors. If no short is present, you can adjust from 4x IDAC to 8x IDAC and restart the tuning process. If a higher range IDAC results in similar results, review your hardware design with a Cypress engineer by creating a [Technical Support](#) case.

**Note** The CapSense v3.0 Component does not support the 8x mode.

#### 5.3.9.9 My slider centroid response is non-linear.

**Cause:** Layout may not meet hardware design guidelines to ensure proper linearity.

Solution: Check the  $C_P$  of the sensors using the built-in self-test option in the General tab of the CapSense configuration window (the BIST option is only available in CapSense v2.40) and update the layout according to the [Slider Design](#) section. Refer to the [CapSense Component datasheet](#) for details on BIST API.

#### 5.3.9.10 My slider segments have a large variation of $C_P$

**Cause:** Your layout design caused your sensors to have an unbalanced  $C_P$ .

Solution: Your layout needs to be updated. Review [Slider Design](#) and update your layout as required. If this is not immediately possible, you should re-tune every sensor to have a similar response. This will be a long iterative process and the preferred method is to update the hardware, if possible.

## 6. Design Considerations



This chapter explains firmware and hardware design considerations for CapSense.

### 6.1 Firmware

The [PSoC 4 and PSoC BLE CapSense Component](#) provides multiple application programming interfaces to simplify firmware development. The [CapSense Component Datasheet](#) provides a detailed list and explanation of the available APIs. You can use the [CapSense Example Projects](#) provided in PSoC Creator to learn schematic entry and firmware development. See [Chapter 4](#) for more details.

The CapSense scan is non-blocking in nature. The CPU intervention is not required between the start and the end of a CapSense scan. Therefore, you can use CPU to perform other tasks while a CapSense scan is in progress. However, note that CapSense is a high-sensitive analog system. Therefore, sudden changes in the device current may increase the noise present in the raw counts. If you are using widgets that require high sensitivity such as proximity sensors, or buttons with thick overlay, you should use a blocking scan. Example firmware for a non-blocking scan is shown below:

```
/* Enable global interrupts */
CyGlobalIntEnable;

/* Start EZI2C component */
EZI2C_Start();

/*
 * Set up communication data buffer to CapSense data structure to be
 * exposed to I2C master at primary slave address request.
 */
EZI2C_EzI2CSetBuffer1(sizeof(CapSense_dsRam),
sizeof(CapSense_dsRam),
(uint8 *) &CapSense_dsRam);

/* Initialize CapSense component */
CapSense_Start();
/* Scan all widgets */
CapSense_ScanAllWidgets();

for (;;)
{
    /* Do this only when a scan is done */
    if(CapSense_NOT_BUSY == CapSense_IsBusy())
    { /* Process all widgets */
        CapSense_ProcessAllWidgets();
        /* Scan result verification */
        if (CapSense_IsAnyWidgetActive())
        {
            /* Add any required functionality
```

```

        based on scanning result */
    }
    /* Include Tuner */
    CapSense_RunTuner();
    /* Start next scan */
    CapSense_ScanAllWidgets();
}
/* CPU Sleep */
CySysPmSleep();
}
}

```

You should avoid interrupted code, power mode transitions, and switching ON/OFF peripherals while a high-sensitivity CapSense scan is in progress. However, if you are not using high-sensitivity widgets, you can use CPU to perform other tasks. You can also use low-power modes of PSoC 4 and PProC BLE to reduce the average power consumption of the CapSense system, as explained in the next section. **Monitoring and verifying the raw counts and SNR using the tuner GUI is recommended if you are using a non-blocking code.**

### 6.1.1 Low-Power Design

PSoC 4 and PProC BLE's low-power modes allow you to reduce overall power consumption while retaining essential functionality. See [AN86233, PSoC 4 Low-Power Modes and Power Reduction Techniques](#), for a basic knowledge of PSoC 4's low-power modes and [AN210998, PSoC® 4 Low-Power CapSense® Design](#), for design a low-power CapSense application.

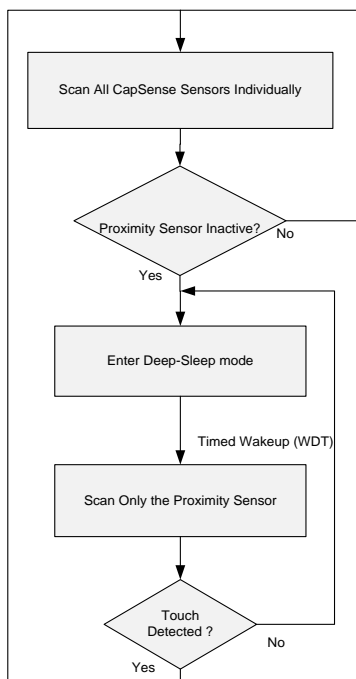
The CPU intervention is not required between the start and the end of a CapSense scan. If the firmware does not have any additional task other than waiting for the scan to finish, you can put the device to Sleep mode after initiating a scan to save power. When the CSD hardware completes the scan, it generates an interrupt to return the device to the Active mode.

If you use APIs that scan multiple sensors together, the device returns to Active mode after finishing the scan of a single sensor. Therefore, if you have multiple sensors in your design, you should scan each one individually.

You can use the Deep-Sleep mode of PSoC 4 or PProC BLE to considerably reduce the power consumption of a CapSense design. However, the CapSense hardware is disabled in the Deep-Sleep mode. Therefore, the device must wake up frequently to scan for touches. You can use the watchdog timer (WDT) in PSoC 4 and PProC BLE to wake up the device from the Deep-Sleep mode at frequent intervals. Increasing the frequency of the scans improves the response of the CapSense system, but it also increases the average power consumption.

As the number of sensors in the design increases, the device has to spend more time in the Active mode to scan all sensors. This, in turn, increases the average power consumption. If a design has many sensors, you should include a separate proximity sensor loop that surrounds the sensors. When the device wakes up from the Deep-Sleep mode, only scan this proximity sensor. If the proximity sensor is active, the device must stay in the Active mode and scan other sensors. If the proximity sensor is inactive, the device can return to the Deep-Sleep mode. [Figure 6-1](#) illustrates this process.

Figure 6-1. Low Power CapSense Design



To further reduce the power consumption, you can make the device enter the Sleep mode when the CPU activity is not required, but other high-speed peripherals such as system timers and I<sup>2</sup>C are required.

**Note** In PSoC 4000 devices, it is not recommended to enter Sleep mode if a CapSense scan is in progress.

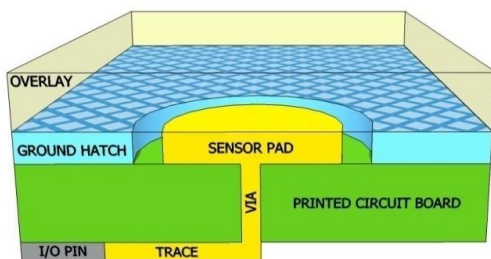
You can also add a shield hatch in the design, as explained in Section 2.5.2 to reduce the parasitic capacitance and hence the scan time. You can achieve very low system currents while maintaining a good touch response, by properly tuning CapSense and the wakeup interval.

## 6.2 Sensor Construction

A capacitive sensor can be constructed using different materials depending on the application requirement. In a typical sensor construction, a conductive pad, or surface that senses a touch is connected to the pin of the capacitive controller using a conductive trace or link. This whole arrangement is placed below a non-conductive overlay material and the user interacts on top of the overlay.

Figure 6-2 shows the most common CapSense sensor construction.

Figure 6-2. CapSense Sensor Construction

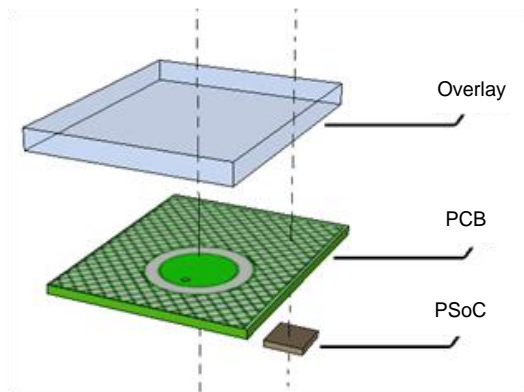


The copper pads etched on the surface of the PCB act as CapSense sensors. A nonconductive overlay serves as the touch surface. The overlay also protects the sensor from the environment and prevents direct finger contact. A ground hatch surrounding the sensor pad isolates the sensor from other sensors and PCB traces.

If liquid tolerance is required, you should use a shield hatch instead of the ground hatch. In this case, drive the hatch with a shield signal instead of connecting to ground. See [Liquid Tolerance](#) for details.

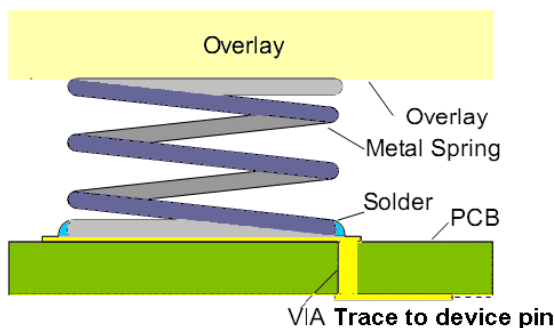
The simplest CapSense PCB design is a two layer board with sensor pads and hatched ground plane on the top, and the electrical components on the bottom. [Figure 6-3](#) shows an exploded view of the CapSense hardware.

Figure 6-3. CapSense Hardware



Sensors may also be constructed by using materials other than copper, such as Indium Tin Oxide (ITO) or printed ink on substrates such as glass or a flex PCB. In some cases, springs can also be used as CapSense sensors as [Figure 6-4](#) shows, in order to create elevated sensors that allow overlay to be placed at an elevated distance from the PCB. Refer to [Getting Started with CapSense Design Guide](#) for PCB design considerations specific to spring sensors and other non-copper sensors such as ITO and printed ink.

Figure 6-4 Sensor Construction using Springs as Sensors





## 6.3 Overlay Selection

### 6.3.1 Overlay Material

The overlay is an important part of CapSense hardware as it determines the magnitude of finger capacitance. The finger capacitance is directly proportional to the relative permittivity of the overlay material. See [finger capacitance](#) for details.

[Table 6-1](#) shows the relative permittivity of some common overlay materials. Materials with relative permittivity between 2.0 and 8.0 are well suited for CapSense overlay.

Table 6-1. Relative Permittivity of Overlay Materials

Material	$\epsilon_r$
Air	1.0
Formica	4.6 – 4.9
Glass (Standard)	7.6 – 8.0
Glass (Ceramic)	6.0
PET Film (Mylar®)	3.2
Polycarbonate (Lexan®)	2.9 – 3.0
Acrylic (Plexiglas®)	2.8
ABS	2.4 – 4.1
Wood Table and Desktop	1.2 – 2.5
Gypsum (Drywall)	2.5 – 6.0

**Note** Conductive materials interfere with the electric field pattern. Therefore, you should not use conductive materials for overlay. You should also avoid using conductive paints on the overlay.

### 6.3.2 Overlay Thickness

Finger capacitance is inversely proportional to the overlay thickness. Therefore, a thin overlay gives more signal than a thick overlay. See [finger capacitance](#) for details.

[Table 6-2](#) lists the recommended maximum thickness of acrylic overlay for different CapSense widgets.

Table 6-2. Maximum Thickness of Acrylic Overlay

Widget	Maximum thickness (mm)
Button	5
Slider	5 <sup>1</sup>
Touchpad	0.5

Because [finger capacitance](#) also depends on the dielectric constant of the overlay, the dielectric constant also plays a role in the guideline for the maximum thickness of the overlay. Common glass has a dielectric constant of approximately  $\epsilon_r = 8$ , while acrylic has approximately  $\epsilon_r = 2.5$ . The ratio of  $\epsilon_r/2.5$  is an estimate of the overlay thickness relative to plastic for the same level of sensitivity. Using this rule of thumb, a common glass overlay can be about three times as thick as a plastic overlay while maintaining the same level of sensitivity.

For CSX sensing, it is recommended to have minimum overlay thickness of 0.5 mm.

<sup>1</sup> For a 5-mm acrylic overlay, the SmartSense Component requires a minimum of 9-mm finger diameter for slider operation. If the finger diameter is less than 9 mm, Manual Tuning should be used.

### 6.3.3 Overlay Adhesives

The overlay must have a good mechanical contact with the PCB. You should use a nonconductive adhesive film for bonding the overlay and the PCB. This film increases the sensitivity of the system by eliminating the air gap between the overlay and the sensor pads. 3M™ makes a high-performance acrylic adhesive called 200MP that is widely used in CapSense applications. It is available in the form of adhesive transfer tapes; example product numbers are 467MP and 468MP.

## 6.4 PCB Layout Guidelines

PCB layout guidelines help you to design a CapSense system with good sensitivity and high [Signal-to-Noise Ratio](#).

### 6.4.1 Parasitic Capacitance, $C_P$

The main components of  $C_P$  are trace capacitance and sensor capacitance. See [CapSense Fundamentals](#) for details. The relationship between  $C_P$  and the PCB layout features is not simple.  $C_P$  increases when:

- sensor pad size increases
- trace length and width increases
- gap between the sensor pad and the ground hatch decreases

You should decrease the trace length and width as much as possible to reduce  $C_P$ . Reducing trace length increases noise immunity. Reducing the sensor pad size is not recommended as it also reduces the finger capacitance.

Another way to reduce  $C_P$  is to increase the gap between the sensor pad and ground hatch. However, widening the gap also decreases noise immunity. You can also reduce  $C_P$  by driving the hatch with a shield signal.

See [Driven-Shield Signal and Shield Electrode](#) for details.

If the sensor  $C_P$  is very high due to long traces or because of a nearby ground, you can use the mutual-capacitance sensing method so that the sensitivity is not degraded because of high  $C_P$  value. The sensitivity of the CapSense sensor in a mutual-capacitance sensing method is independent of sensor  $C_P$ .

### 6.4.2 Board Layers

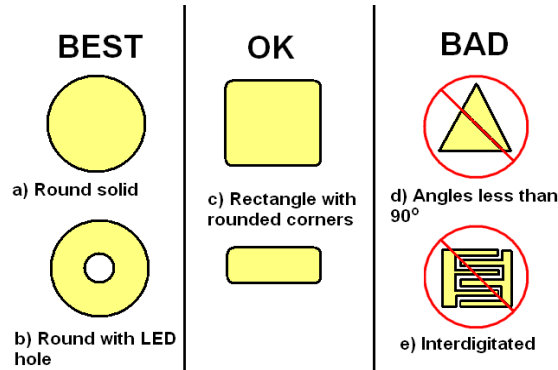
Most applications use a two-layer board with the sensor pads and the hatched ground planes on the top side and all other components on the bottom side. PCBs that are more complex use four layers. FR4-based PCB designs perform well with board thickness ranging from 0.020 inches (0.5 mm) to 0.063 inches (1.6 mm).

Flex circuits work well with CapSense. You should use flex circuits for curved surfaces. All PCB guidelines in this document also apply to flex. You should use flex circuits with thickness 0.01 inches (0.25 mm) or higher for CapSense. The high breakdown voltage of the Kapton® material (290 kV/mm) used in flex circuits provides built in ESD protection for the CapSense sensors.

### 6.4.2.1 Self-Capacitance Button Design

You should use circular sensor pads for CapSense buttons. Rectangular shapes with rounded corners are also acceptable. However, you should avoid sharp corners (less than 90°) since they concentrate electric fields. [Figure 6-5](#) shows recommended button shapes.

Figure 6-5. Recommended Button Shapes

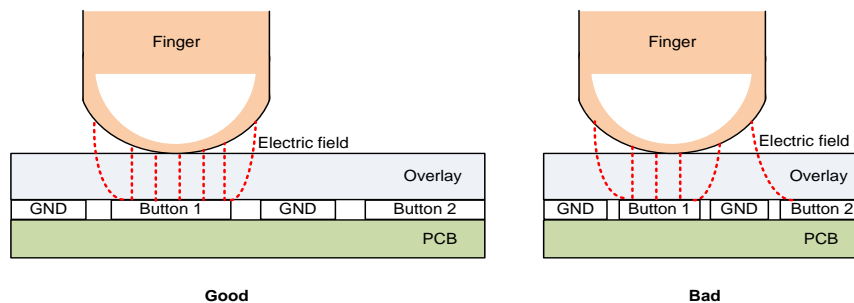


Button diameter should be 5 mm to 15 mm, with 10 mm suitable for most applications. A larger diameter is appropriate for thicker overlays.

The width of the gap between the sensor pad and the ground hatch should be equal to the overlay thickness, and range from 0.5 mm to 2 mm. For example, if the overlay thickness is 1 mm, you should use a 1-mm gap. However, for a 3-mm overlay, you should use a 2-mm gap.

Select the spacing between two adjacent buttons such that when touching a button, the finger is not near the gap between the other button and the ground hatch, to prevent false touch detection on the adjacent buttons, as [Figure 6-6](#) shows.

Figure 6-6. Spacing between Buttons



### 6.4.2.2 Mutual-Capacitance Button

Mutual capacitance sensing measures the change in capacitive coupling between two electrodes. The sensor pattern should be designed in such a way that the finger disturbs the electric field between the electrodes to a maximum extent. [Figure 6-7](#) shows the mutual-cap button design for overlays with thickness less than 1.5 mm and [Figure 6-8](#) shows the mutual-cap button design for overlays with a thickness greater than 1.5 mm.

Sensor area is a critical parameter for the mutual-capacitance sensors. For an overlay thickness of (t), the minimum sensor area (A) is given by:

$$A = \frac{0.25pF \times t}{\epsilon_0 \times \epsilon_r}$$

Where  $\epsilon_0$  is the permittivity of free space ( $8.85 \times 10^{-15}$  F/mm) and  $\epsilon_r$  is the dielectric constant (also known as relative permittivity) of the overlay.

The minimum electrode size for patterns shown in [Figure 6-8](#) and [Figure 6-9](#) should be twice the overlay thickness to ensure good signal.

Figure 6-7. Mutual-Capacitance Button Design for Thin Overlays

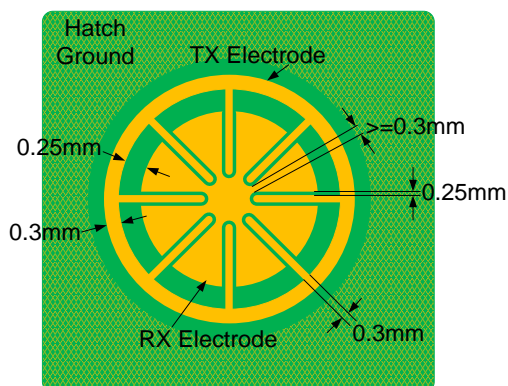
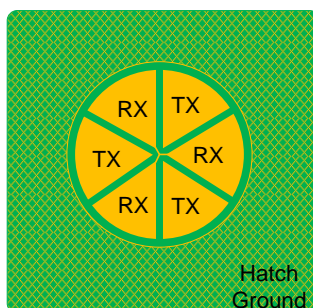


Figure 6-8. Mutual-Capacitance Button Design for Thick Overlays



For thicker overlay ( $> 1.5$  mm), the button size will be large and in such cases, it is recommended to increase the number of Tx and Rx segments as shown in [Figure 6-9](#). This will increase the coupling between the electrodes and increase the signal when a finger touches the sensor.

Figure 6-9. Mutual-Capacitance Button Design with Multiple Tx and Rx Segments



The hatch ground around the sensor reduces the noise in the sensor raw counts and hence improves SNR. The hatch ground can also be placed beneath the sensor to avoid coupling and reduce noise from the bottom side. The hatch fill percentage can be same as the one listed in [Table 6-8](#) for self-capacitance buttons.

### 6.4.3 Slider Design

Figure 6-10 shows the recommended slider pattern for a linear slider and Table 6-3 shows the recommended values for each of the linear slider dimensions. A detailed explanation on the recommended layout guidelines is provided in the following sections.

Figure 6-10. Typical Linear Slider Pattern

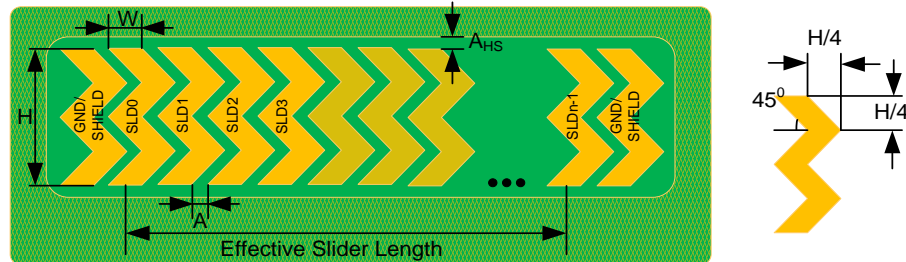


Table 6-3. Linear Slider Dimensions

Parameter	Acrylic Overlay Thickness	Minimum	Maximum	Recommended
Width of the segment (W)	1 mm	2 mm	-	8 mm <sup>1</sup>
	3 mm	4 mm	-	
	4 mm	6 mm	-	
Height of the segment (H)	-	7 mm <sup>2</sup>	15 mm	12 mm
Air gap between segments (A)	-	0.5 mm	2 mm	0.5 mm
Air gap between the hatch and the slider (A <sub>HS</sub> )	-	0.5 mm	2 mm	Equal to overlay thickness

<sup>1</sup> The recommended slider-segment width is based on an average human finger diameter of 9 mm. Refer to section 6.4.3.1, “Slider-Segment Shape, Width, and Air Gap” for more details.

<sup>2</sup> The minimum slider segment height of 7 mm is recommended based on a minimum human finger diameter of 7 mm. Slider height may be kept lower than 7 mm if the overlay thickness and CapSense tuning is such that an [Signal-to-Noise Ratio](#)  $\geq 5:1$  is achieved when the finger is placed in the middle of any segment.

### 6.4.3.1 Slider-Segment Shape, Width, and Air Gap

A linear response of the reported finger position (that is, the Centroid position) versus the actual finger position on a slider requires that the slider design is such that whenever a finger is placed anywhere between the middle of the segment SLD0 and middle of segment SLDn-1, other than the exact middle of slider segments, exactly two sensors report a valid signal<sup>1</sup>. If a finger is placed at the exact middle of any slider segment, the adjacent sensors should report a difference count = noise threshold. Therefore, it is recommended that you use a double-chevron shape as Figure 6-10 shows. This shape helps in achieving a centroid response close to the ideal response, as Figure 6-11 and Figure 6-12 show. For the same reason, the slider-segment width and air gap (dimensions “W” and “A” respectively, as marked in Figure 6-10) should follow the relationship mentioned in Equation 6-1.

Figure 6-11. Ideal Slider Segment Signals and Centroid Response

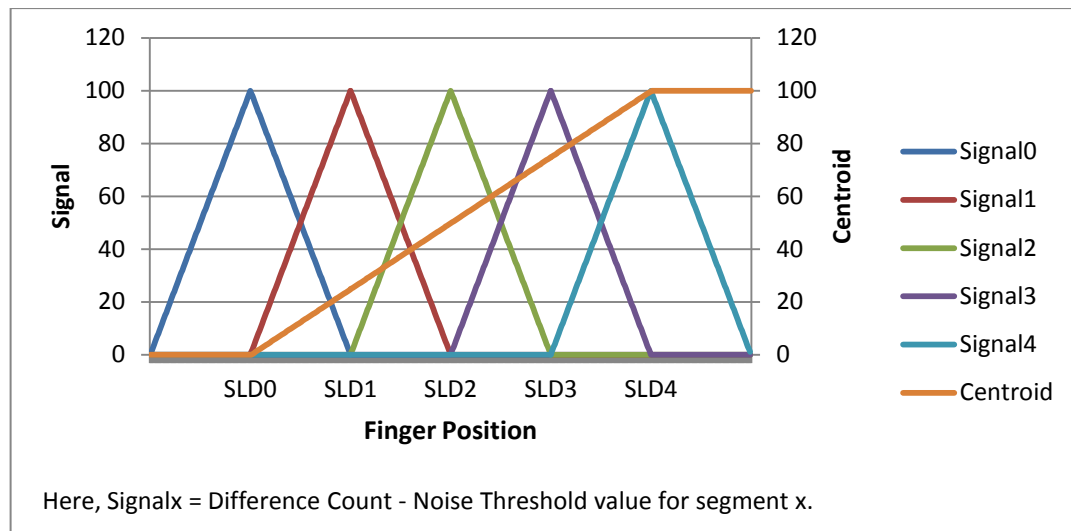
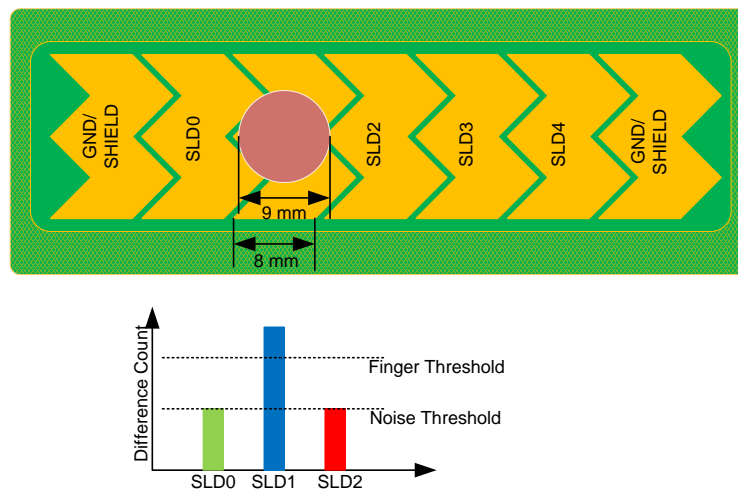


Figure 6-12. Ideal Slider Signals



Equation 6-1 Segment width and air gap relation with the finger diameter

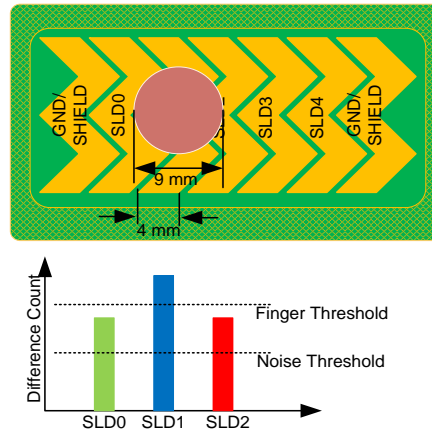
$$W + 2A = \text{finger diameter}$$

<sup>1</sup> Here, a valid signal means that the difference count of the given slider segment is greater than or equal to the noise threshold value.

Typically, an average human finger diameter is approximately 9 mm. Based on this average finger diameter and [Equation 6-1](#), the recommended slider-segment-width and air-gap is 8 mm and 0.5 mm respectively.

If the *slider-segment-width + 2 \* air-gap* is lesser than *finger diameter*, as required according to [Equation 6-1](#), the centroid response will be non-linear. This is because, in this case, a finger placed on the slider will add capacitance, and hence valid signal to more than two slider-segments at some given position, as [Figure 6-13](#) shows. Thus, calculated centroid position in [Equation 6-2](#) will be non-linear as [Figure 6-14](#) shows.

Figure 6-13. Finger Causes Valid Signal on More Than Two Segments When Slider Segment Width Is Lower Than Recommended



Equation 6-2. Centroid Algorithm used by CapSense Component in PSoC Creator

$$\text{Centroid position} = \left( \frac{S_{x+1} - S_{x-1}}{S_{x+1} + S_{x0} + S_{x-1}} + \text{maximum} \right) * \frac{\text{Resolution}}{(n - 1)}$$

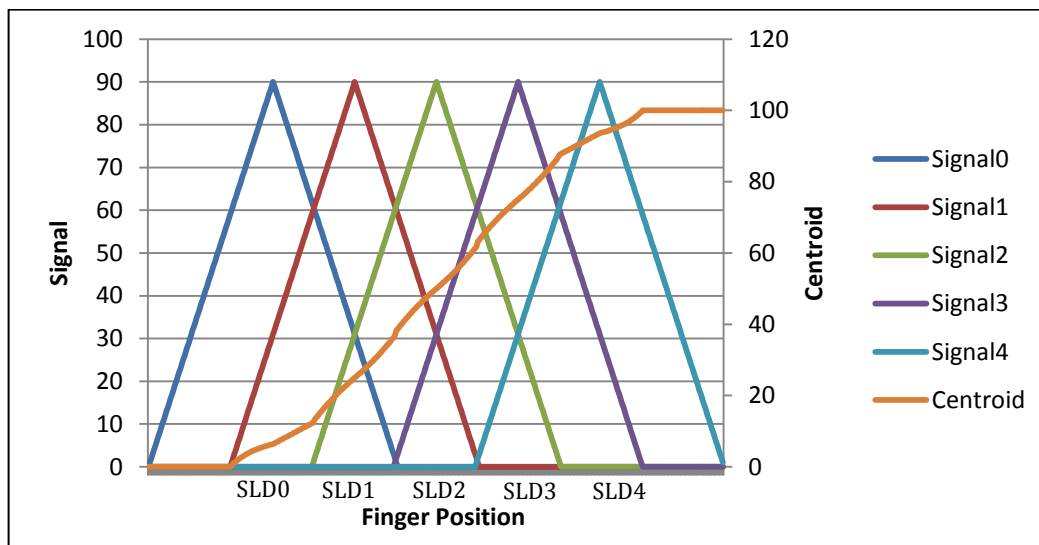
Resolution – API resolution set in the CapSense Component Customizer

n – Number of sensor elements in the CapSense Component Customizer

maximum – Index of element which gives maximum signal

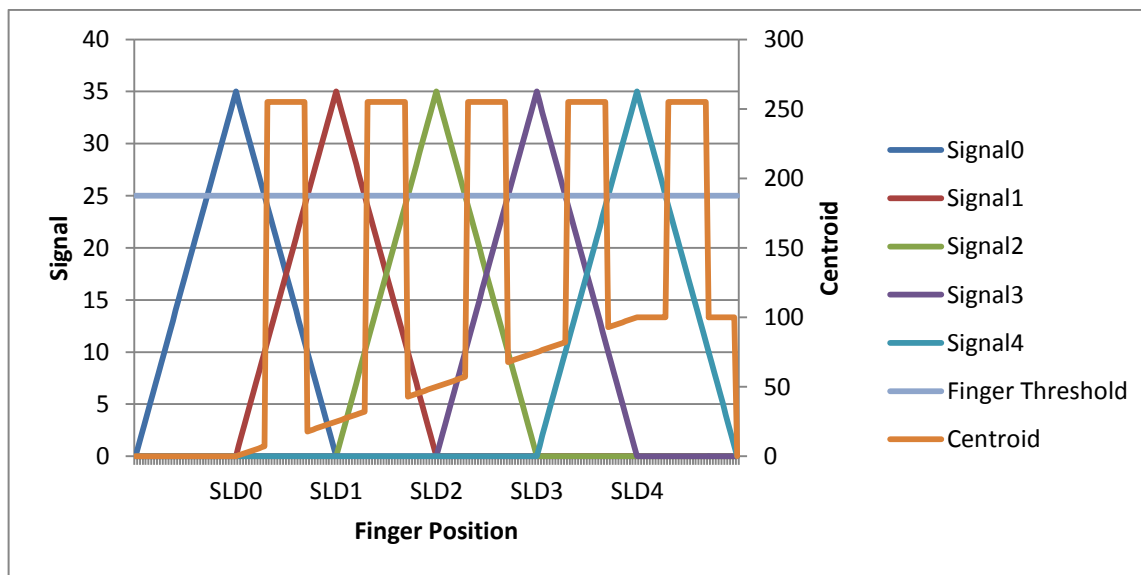
Si – different counts (with subtracted noise threshold value) near the maximum position

Figure 6-14. Nonlinear Centroid Response when Slider Segment Width Is Lower Than Recommended



Note that even though a *slider-segment-width* value of less than *finger diameter* - 2 \* *air-gap* provides a non-linear centroid response, as Figure 6-14 shows; it may still be used in an end application where the linearity of reported centroid versus actual finger position does not play a significant role. However, a minimum value of slider-segment-width must be maintained, based on overlay thickness, such that, at any position on the effective slider length, at least one slider-segment provides a *Signal-to-Noise Ratio* of  $\geq 5:1$  (that is signal greater than or equal to the finger threshold parameter) at that position. If the slider-segment width is too low, a finger may not be able to couple enough capacitance, and hence, none of the slider-segments will have a 5:1 SNR, resulting in a reported centroid value of 0xFF<sup>1</sup>, as Figure 6-15 shows.

Figure 6-15. Incorrect Centroid Reported when Slider-Segment-Width Is Too Low

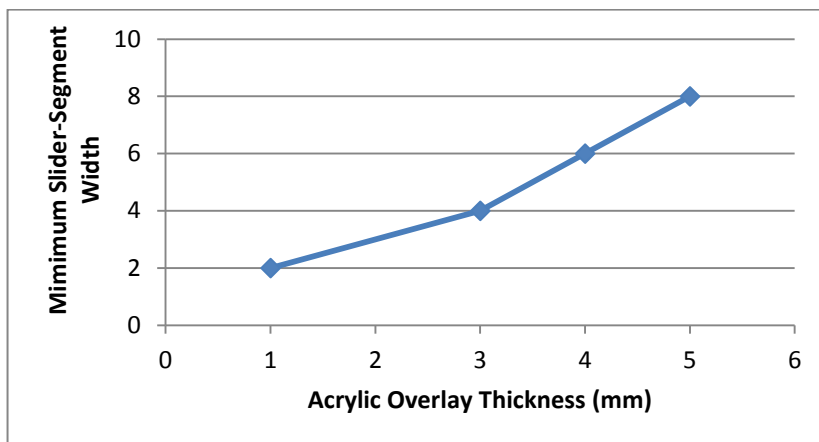


The minimum value of slider-segment width for certain overlay thickness values for an acrylic overlay are provided in Table 6-3. For thickness values of acrylic overlays, which are not specified in Table 6-3, Figure 6-16 may be used to estimate the minimum slider-segment width.

<sup>1</sup> The CapSense Component in PSoC Creator reports a centroid of 0xFF when there is no finger detected on the slider, or when none of the slider segments reports a difference count value greater than the Finger Threshold parameter.



Figure 6-16. Minimum Slider-Segment Width w.r.t. Overlay Thickness for an Acrylic Overlay



If the *slider-segment-width* + 2 \* *air-gap* is higher than the *finger diameter* value as required in Equation 6-1, the centroid response will have flat spots; that is, if the finger is moved towards the middle of any segment, the reported centroid position will remain constant as Figure 6-17 shows. This is because, as Figure 6-18 shows, when the finger is placed in the middle of a slider segment, it will add a valid signal only to that segment even if the finger is moved a little towards adjacent segments.

Figure 6-17. Flat Spots (Nonresponsive Centroid) when Slider-Segment Width Is Higher than Recommended

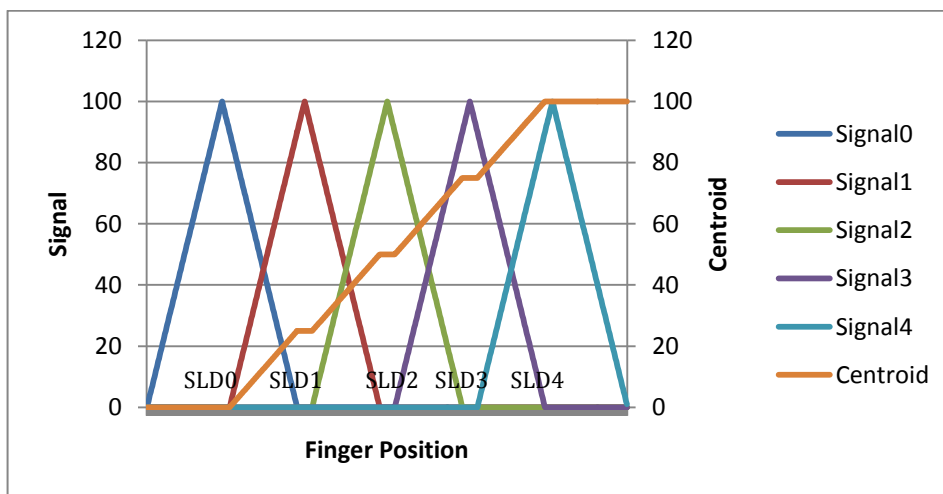
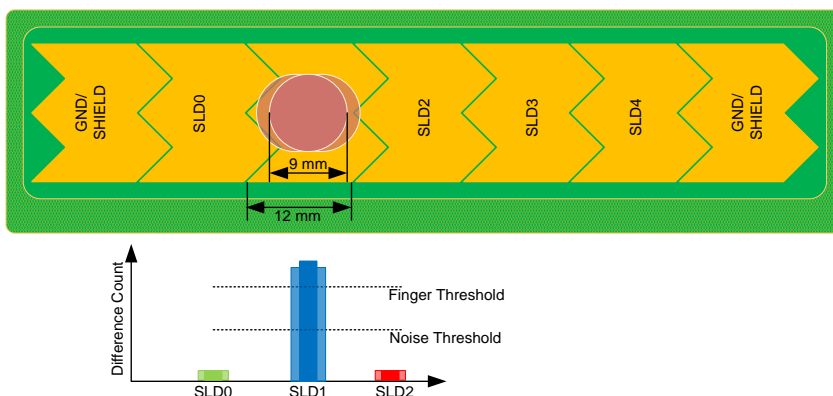


Figure 6-18. Signal on Slider Segments when Slider-Segment Width is Higher than Recommended



Note that if the value of *slider-segment-width* + 2 \* *air-gap* is higher than the *finger diameter*, it may be possible to increase and adjust the sensitivity of all slider segments such that even if the finger is placed in the middle of a slider segment, adjacent sensors report a difference count value equal to the noise threshold value (see [Figure 6-11](#)); however, this will result in the hover effect – the slider may report a centroid position even if the finger is hovering above the slider and not touching the slider.

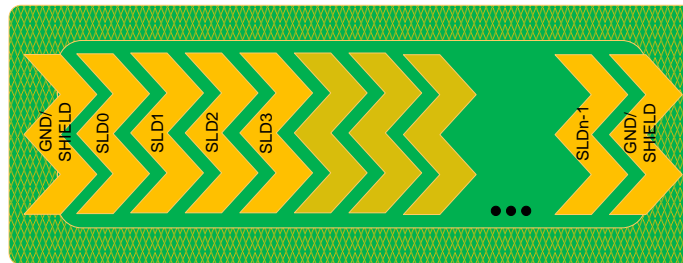
#### 6.4.3.2 Dummy Segments at the Ends of a Slider

In a CapSense design, when one segment is scanned, adjacent segments are connected to either ground or to the driven-shield signal based on the option specified in the “Inactive sensor connection” parameter in the CapSense CSD Component. For a linear centroid response, the slider requires all the segments to have the same sensitivity, that is, the increase in the raw count (signal) when a finger is placed on the slider segment should be the same for all segments. To maintain a uniform signal level from all slider segments, it is recommended that you physically connect the two segments at both ends of a slider to either ground or driven shield signal. The connection to ground or to the driven-shield signal depends on the value specified in the “Inactive sensor connection” parameter. Therefore, if your application requires an ‘n’ segment slider, it is recommended that you create n + 2 physical segments, as [Figure 6-10](#) shows.

If it is not possible to have two segments at both ends of a slider due to space constraints, you can implement these segments in the top hatch fill, as [Figure 6-19](#) shows. Also, if the total available space is still constrained, the width of these segments may be kept lesser than the width of segments SLD0 through SLDn-1, or these dummy segments may even be removed.

If the two segments at the both ends of a slider are connected to the top hatch fill, you should connect the top hatch fill to the signal specified in the “Inactive sensor connection” parameter. If liquid tolerance is required for the slider, the hatch fill around the slider, the last two segments, and the inactive slider segments should be connected to the driven-shield signal. See the [Effect of Liquid Droplets and Liquid Stream on a CapSense Sensor](#) section for more details.

Figure 6-19. Linear Slider Pattern when First and Last Segments are Connected to Top Hatch Fill



#### 6.4.3.3 Deciding Slider Dimensions

Slider dimensions for a given design can be chosen based on following considerations:

- Decide the required length of the slider (L) based on application requirements. This is same as the “effective slider length” as [Figure 6-10](#) shows.
- Decide the height of the segment based on the available space on the board. Use the maximum allowed segment height (15 mm) if the board space permits; if not, use a lesser height but ensure that the height is greater than the minimum specified in [Table 6-3](#).
- The slider-segment width and the air gap between slider segments should be as recommended in [Table 6-3](#). The recommended slider-segment-width and air-gap for an average finger diameter of 9 mm is 8 mm and 0.5 mm respectively.
- For a given slider length L, calculate the number of segments required by using the following formula:

Equation 6-3: Number of Segments Required for a Slider

$$\text{Number of segments} = \frac{\text{slider length}}{\text{slider segment width} + \text{air gap}} + 1$$

Note that a minimum of two slider segments are required to implement a slider.

If the available number of CapSense pins is slightly less than the number of segments calculated for a certain application, you may increase the segment width to achieve the required slider length with the available number of pins. For example, a 10.2-cm slider requires 13 segments. However, if only 10 pins are available, the segment width may be increased to 10.6 cm. This will either result in a nonlinear response as [Figure 6-17](#) shows, or a hover effect; however, this layout may be used if the end application does not need a high linearity.

Note that the PCB length is higher than the required slider length as [Figure 6-10](#) shows. PCB length can be related to the slider length as follows:

Equation 6-4. Relationship between Minimum PCB Length and Slider Length

$$PCB\ length = Slider\ Length + 3 * slider\ segment\ width + 2 * air\ gap$$

If the available PCB area is less than that required per this equation, you can remove the dummy segments. In this case, the minimum PCB length required will be as follows:

$$PCB\ length = Slider\ Length + slider\ segment\ width$$

#### 6.4.3.4 Routing Slider Segment Trace

A slider has many segments, each of which is connected separately to the CapSense input pin of the device. Each segment is separately scanned and the centroid algorithm is applied finally on the signal values of all the segments to calculate the centroid position. The SmartSense algorithm implements a specific tuning method for sliders to avoid nonlinearity in the centroid that could occur due to the difference of  $C_P$  in the segments. However, the following layout conditions need to be met for the slider to work:

1.  $C_P$  of any segment should always be within the supported range of 5-45 pF.
2.  $C_P$  of other segments should be greater than 75 percent of the  $C_P$  of the segment with the maximum value in that slider. For example, in a slider, if 30 pF is the maximum  $C_P$  of a segment, the  $C_P$  of other segments should be greater than 22.5 pF.

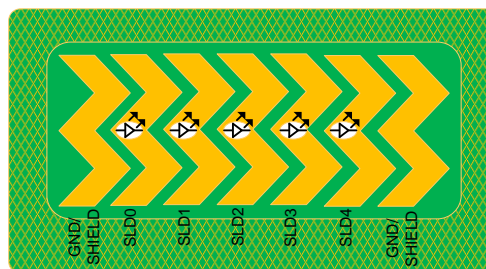
Implement the following layout design rules to meet this condition:

- Design the shape of all segments to be as uniform as possible.
- Ensure that the length and the width of the traces connecting the segments to the device are same for all the segments.
- Maintain the same air gap between the sensors or traces to ground plane or hatch fill.

#### 6.4.3.5 Slider Design with LEDs

In some applications, it may be required to display the finger position by driving LEDs. You can either place the LEDs just above the slider segments or drill a hole in the middle of a slider segment for LED backlighting, as [Figure 6-20](#) shows. When a hole is drilled for placing an LED, the effective area of the slider segment reduces. To achieve an  $SNR > 5:1$ , you need to have a slider segment with a width larger than the LED hole size. Refer to [Table 6-3](#) for the minimum slider width required to achieve an  $SNR > 5:1$  for a given overlay thickness. Follow the guidelines provided in the [Crosstalk Solutions](#) section to route the LED traces.

Figure 6-20. Slider Design with LED Backlighting



## 6.4.4 Sensor and Device Placement

Follow these guidelines while placing the sensor and the device in your PCB design:

- Minimize the trace length from the device pins to the sensor pad.
- Mount series resistors within 10 mm of the device pins to reduce RF interference and provide ESD protection. See [Series Resistors on CapSense Pins](#) for details.
- Mount the device and the other components on the bottom layer of the PCB.
- Isolate switching signals, such as PWM, I<sup>2</sup>C communication lines, and LEDs, from the sensor and sensor traces. You should place them at least 4 mm apart and fill a hatched ground between the CapSense traces and the switching signals to avoid crosstalk.
- DC loads such as LEDs and I<sup>2</sup>C pins should be physically separate from the CapSense pins by a full port wherever possible. For example, if there are LED pins in port P1, it is recommended to avoid having a CapSense pin in the same port. Also, it is recommended to limit the total source or sink current through GPIOs to less than 40 mA while the CapSense block is scanning the sensor. Sinking a current greater than 40 mA during the CapSense sensor scanning may result in excessive noise in the sensor raw count.
- Avoid connectors between the sensor and the device pins because connectors increase C<sub>p</sub> and noise pickup.

## 6.4.5 Trace Length and Width

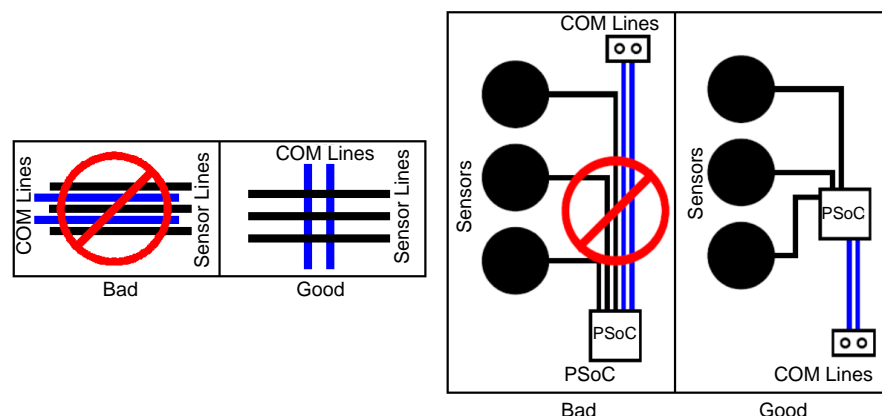
Use short and narrow PCB traces to minimize the parasitic capacitance of the sensor. The maximum recommended trace length is 12 inches (300 mm) for a standard PCB and 2 inches (50 mm) for flex circuits. The maximum recommended trace width is 7 mil (0.18 mm). You should surround the CapSense traces with a hatched ground or hatched shield with trace-to-hatch clearance of 10 mil to 20 mil (0.25 mm to 0.51 mm).

## 6.4.6 Trace Routing

You should route the sensor traces on the bottom layer of the PCB, so that the finger does not interact with the traces. Do not route traces directly under any sensor pad unless the trace is connected to that sensor.

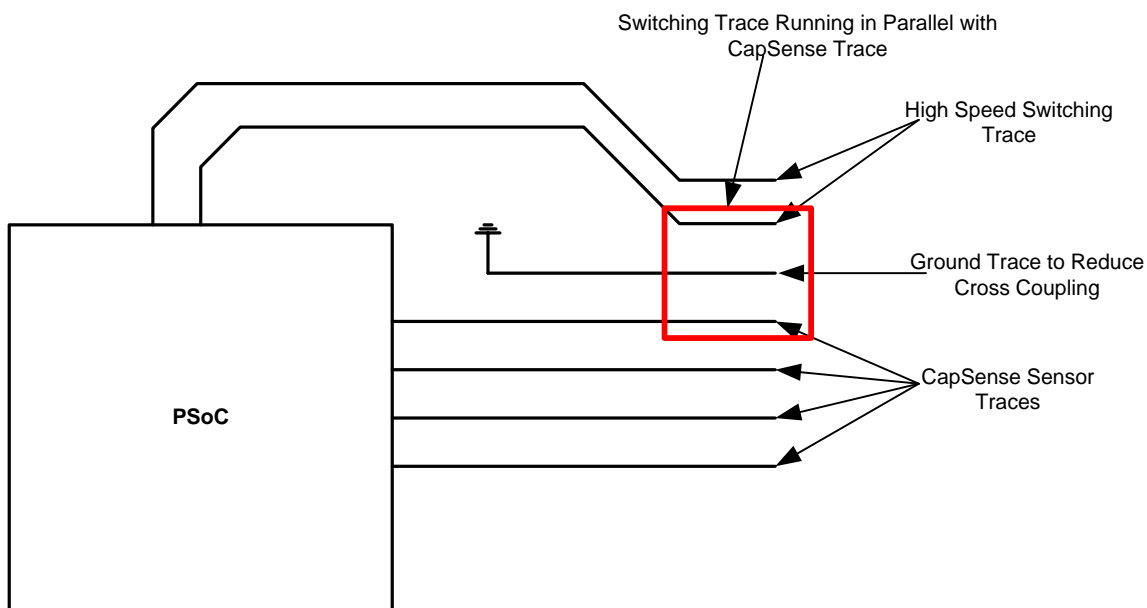
Do not run capacitive sensing traces closer than 0.25 mm to switching signals or communication lines. Increasing the distance between the sensing traces and other signals increases the noise immunity. If it is necessary to cross communication lines with sensor pins, make sure that the intersection is at right angles, as [Figure 6-21](#) shows.

Figure 6-21. Routing of Sensor and Communication Lines



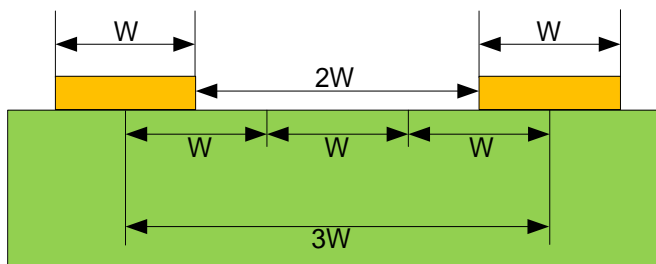
If, due to spacing constraints, sensor traces run in parallel with high-speed traces such as I<sup>2</sup>C communication lines or BLE antenna traces, it is recommended to place a ground trace between the sensor trace and the high-speed trace as shown in [Figure 6-22](#). This guideline also applies to the cross talk caused by CapSense sensor trace with precision analog trace such as a traces from temperature sensor to the PSoC device. The thickness of the ground trace can be 7 mils and the spacing from sensor trace to ground trace should be equal to minimum of 10 mils to reduce the C<sub>P</sub> of the CapSense sensor.

Figure 6-22. Reducing Cross Talk between High Speed Switching Trace and CapSense Trace



If a ground trace cannot be placed in between the switching trace and the CapSense trace, the 3W rule can be followed to reduce the cross talk between the traces. The 3W rule states that “to reduce cross talk from adjacent traces, a minimum spacing of two trace widths should be maintained from edge to edge” as shown in [Figure 6-23](#).

Figure 6-23. 3W Trace Spacing to Minimize Cross Talk

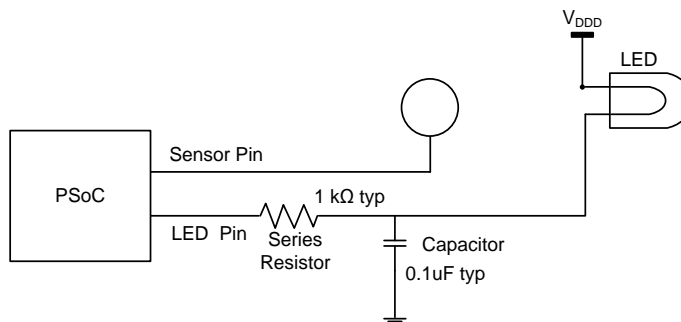


## 6.4.7 Crosstalk Solutions

A common backlighting technique for panels is an LED mounted under the sensor pad so that it is visible through a hole in the middle of the sensor pad. When the LED is switched ON or OFF, voltage transitions on the LED trace can create crosstalk in the capacitive sensor input, creating noisy sensor data. To prevent this crosstalk, isolate CapSense and the LED traces from one another as [section 6.3.7](#) explains.

You can also reduce crosstalk by removing the rapid transitions in the LED drive voltage, by using a filter as [Figure 6-24](#) shows. Design the filter based on the required LED response speed.

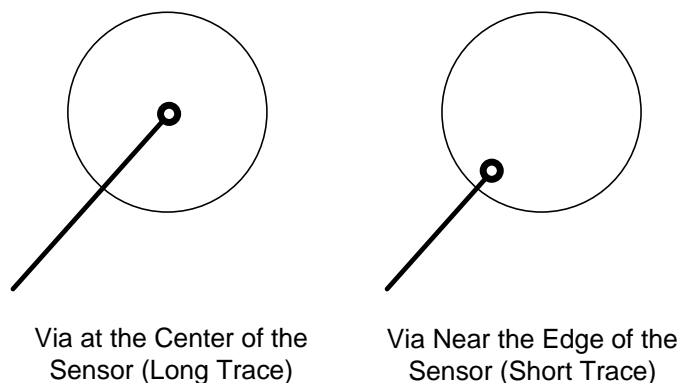
Figure 6-24. Reducing Crosstalk



## 6.4.8 Vias

Use the minimum number of vias possible to route CapSense signals, to minimize parasitic capacitance. Place the vias on the edge of the sensor pad to reduce trace length, as [Figure 6-25](#) shows.

Figure 6-25. Via Placement on the Sensor Pad



## 6.4.9 Ground Plane

When designing the ground plane, follow these guidelines:

- Ground surrounding the sensors should be in a hatch pattern. If you are using ground planes in both top and bottom layers of the PCB, you should use a 25 percent hatching on the top layer (7-mil line, 45-mil spacing), and 17 percent on the bottom layer (7-mil line, 70-mil spacing). Use the same hatching fill on both the top and bottom layers if you are using driven shield instead of ground.
- For the other parts of the board not related to CapSense, solid ground should be present as much as possible.
- The ground planes on different layers should be stitched together as much as possible, depending on the PCB manufacturing costs. Higher amount of stitching results in lower ground inductance, and brings the chip ground closer to the supply ground. This is important especially when there is high current sinking through the ground, such as when the radio is operational.
- Every ground plane used for CapSense should be star-connected to a central point, and this central point should be the sole return path to the supply ground. Specifically:
  - The hatch ground for all sensors must terminate at the central point
  - The ground plane for  $C_{MOD}$  must terminate at the central point
  - The ground plane for  $C_{SH\_TANK}$  must terminate at the central point

[Figure 6-26](#) explains the star connection. The central point for different families is mentioned in [Table 6-4](#).

Figure 6-26. Star Connection for Ground

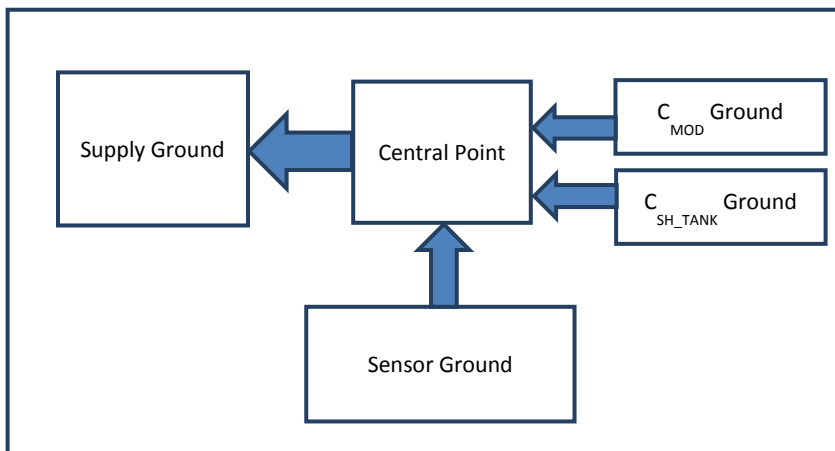


Table 6-4. Central Point for Star Connection

Family	Central Point
PSoC 4000	VSS pin
PSoC 4100/4100M	VSS pin
PSoC 4200/4200M/4200L/PSoC 4-S	VSS pin
PSoC 4100-BL	E-pad
PSoC 4200-BL	E-pad
PRoC BLE	E-pad

- All the ground planes for CapSense should have an inductance of less than 0.2 nH from the central point. To achieve this, place the  $C_{MOD}$ ,  $C_{INTX}$ , and  $C_{SH\_TANK}$  capacitor pads close to the chip, and keep their ground planes thick enough.

#### 6.4.9.1 Using Packages without E-pad

When not using the E-pad, the VSS pin should be the central point and the sole return path to the supply ground.

High-level layout diagrams of the top and bottom layers of a board when using a chip without the E-pad are shown in [Figure 6-27](#) and [Figure 6-28](#).



Figure 6-27. PCB Top Layer Layout Using a Chip without E-pad

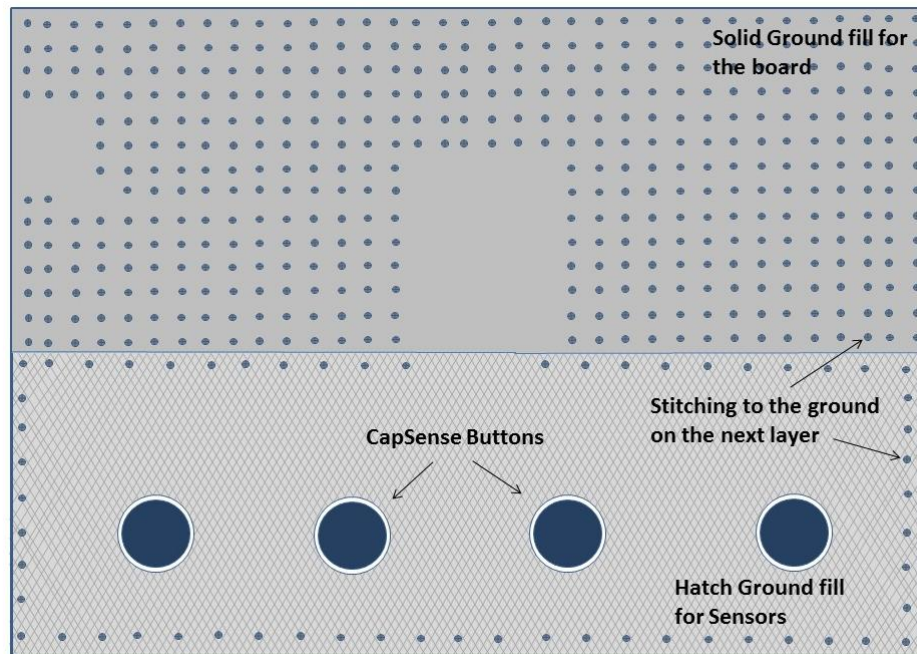
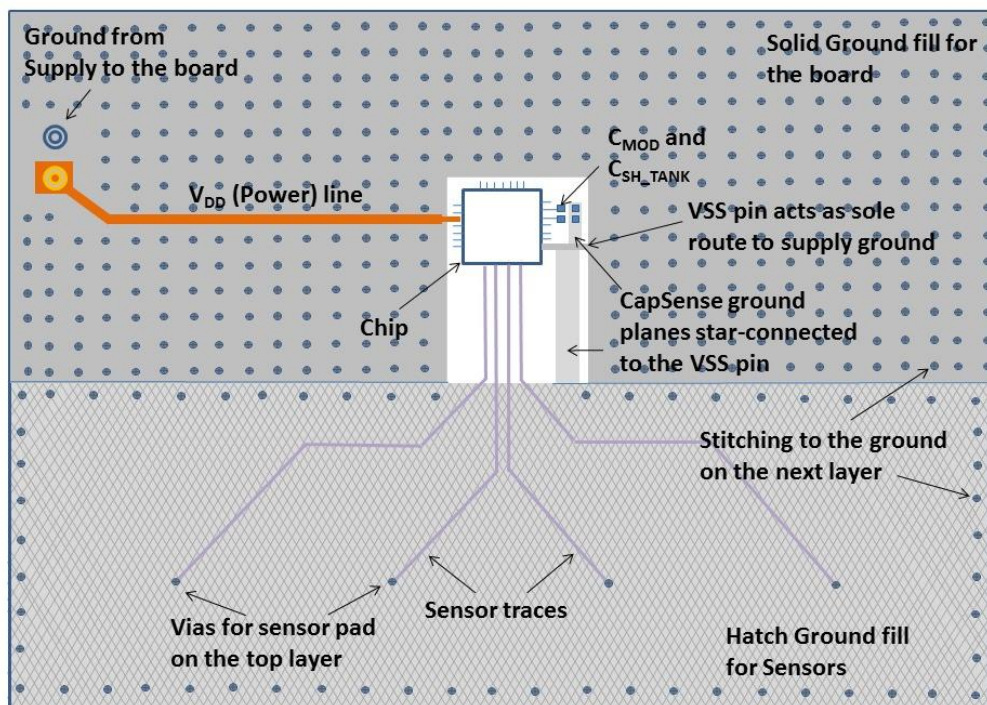


Figure 6-28. PCB Bottom Layer Layout Using a Chip without E-pad





### 6.4.9.2 Using Packages with E-pad

If you are using packages with E-pad, the following guidelines must be followed:

- The E-pad must be the central point and the sole return path to the supply ground.
- The E-pad must have vias underneath to connect to the next layers for additional grounding. Usually unfilled vias are used in a design for cost purposes, but silver-epoxy filled vias are recommended for the best performance as they result in the lowest inductance in the ground path.

### 6.4.9.3 Using PSoC 4 BLE or PProC BLE Chips

In the case of PSoC 4 BLE or PProC BLE chips in the QFN package (with E-pad):

- The general guidelines of ground plane (discussed above) apply.
- The E-pad usage guidelines of [Section 6.4.9.2](#) apply.
- The VSSA pin should be connected to the E-pad below the chip itself.
- The vias underneath the E-pad are recommended to be 5 x 5 vias of 10-mil size.

High-level layout diagrams of the top and bottom layers of a board when using PSoC 4 BLE or PProC BLE chips are shown in [Figure 6-29](#) and [Figure 6-30](#).

Figure 6-29. PCB Top Layer Layout with PSoC 4 BLE / PProC BLE (with E-pad)

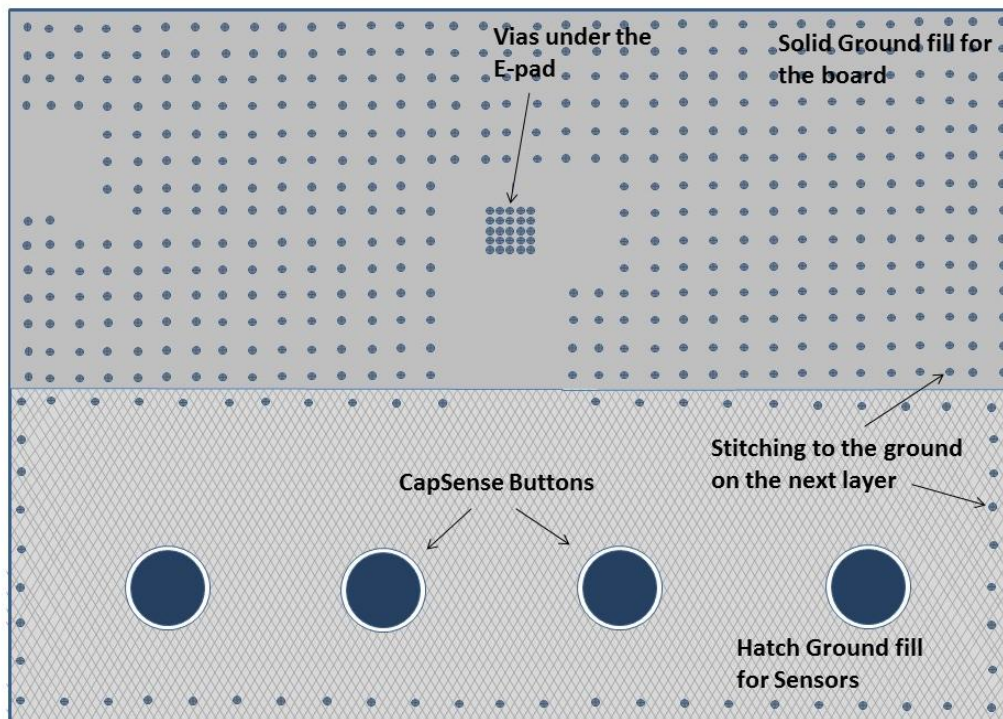
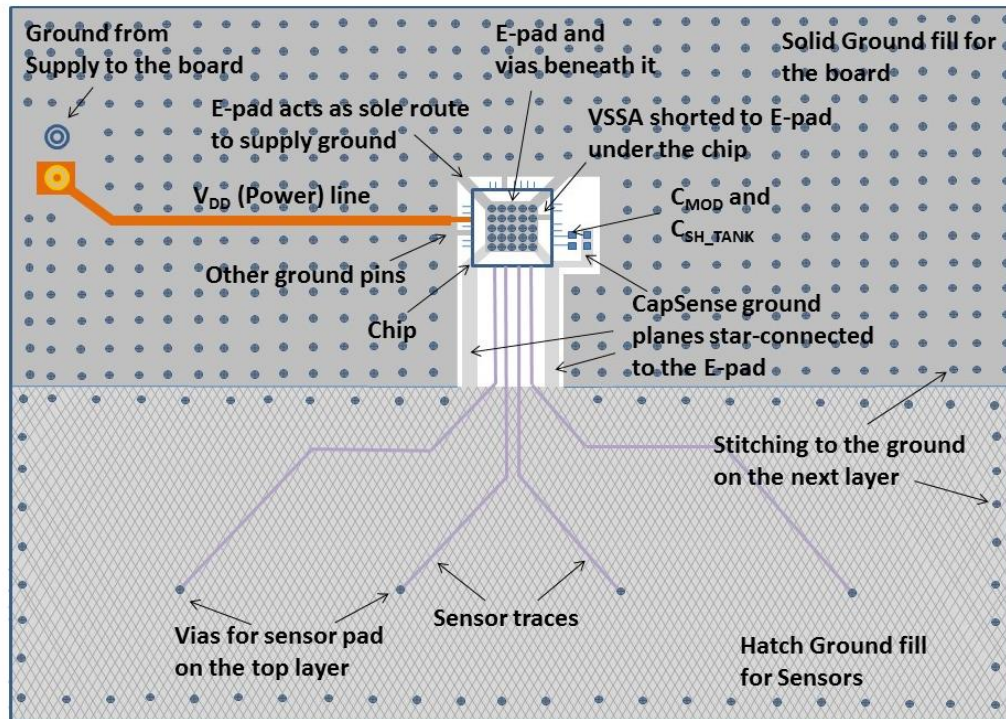


Figure 6-30. PCB Bottom Layer Layout with PSoC 4 BLE / PProC BLE (with E-pad)



#### 6.4.10 Power Supply Layout Recommendations

CapSense is a high-sensitivity analog system. Therefore, a poor PCB layout introduces noise in high-sensitivity sensor configurations such as proximity sensors and buttons with thick overlays (>1 mm). To achieve low noise in a high-sensitivity CapSense design, the PCB layout should have decoupling capacitors on the power lines, as listed in [Table 6-5](#).

Table 6-5. Decoupling Capacitors on Power Lines

Power Line	Decoupling Capacitors	Corresponding Ground Terminal	Applicable Device Family
VDD	0.1 $\mu$ F and 1 $\mu$ F	VSS	PSoC 4000
VDDIO	0.1 $\mu$ F	VSS	PSoC 4000
VDDD	0.1 $\mu$ F and 1 $\mu$ F	VSS	PSoC 4100, PSoC 4200
	0.1 $\mu$ F and 1 $\mu$ F	VSSD	PSoC 4100-BL, PSoC 4200-BL, PProC BLE, PSoC 4200 L, PSoC 4-S
VDDA <sup>1</sup>	0.1 $\mu$ F and 1 $\mu$ F (Battery powered supply)	VSSA	PSoC 4100, PSoC 4200, PSoC 4100-BL, PSoC 4200-BL, PProC BLE, PSoC 4200 L, PSoC 4-S
	0.1 $\mu$ F and 10 $\mu$ F (Mains Powered supply)	VSSA	PSoC 4-S
VDDR	0.1 $\mu$ F and 1 $\mu$ F	VSSD	PSoC 4100-BL, PSoC 4200-BL, PProC BLE
VCCD	Refer device datasheet	VSS (PSoC 4000) or VSSD (all others)	All device families

<sup>1</sup> The VDDA pin on PSoC 4 S-Series requires different values of bulk capacitor depending on the power supply source. If the device is battery powered, it is recommended to use 0.1- $\mu$ F and 1- $\mu$ F capacitors in parallel and if the device is mains powered, it is recommended to use 0.1  $\mu$ F and 10  $\mu$ F in parallel. This is to improve the power supply rejection ratio of reference generator (REFGEN) used in the CapSense block.

The decoupling capacitors and  $C_{MOD}$  capacitor must be placed as close to the chip as possible to keep ground impedance and supply trace length as low as possible.

For further details on bypass capacitors, refer to the Power section in the device [Datasheet](#).

### 6.4.11 Layout Guidelines for Liquid Tolerance

As explained in the [Liquid Tolerance](#) section, by implementing a shield electrode and a guard sensor, a liquid-tolerant CapSense system can be implemented. This section shows how to implement a shield electrode and a guard sensor.

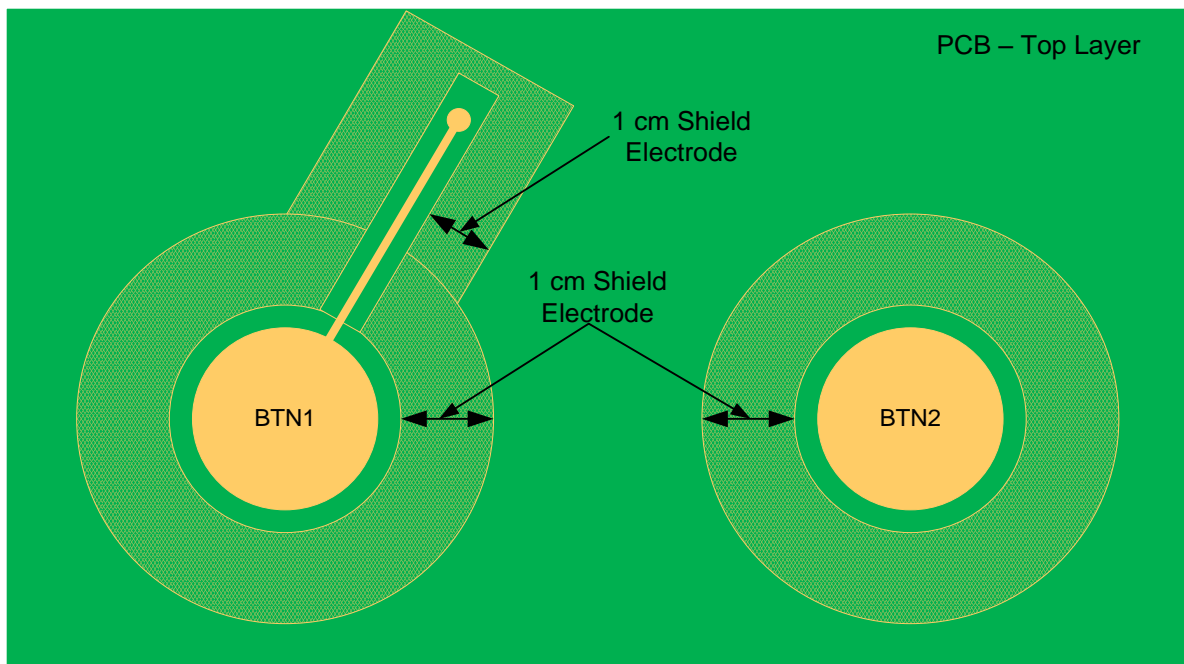
The area of the shield electrode depends on the size of the liquid droplet and the area available on the board for implementing the shield electrode. The shield electrode should surround the sensor pads and traces, and spread no further than 1 cm from these features. Spreading the shield electrode beyond 1 cm has negligible effect on system performance.

Also, having a large shield electrode may increase radiated emissions. If the board area is very large, the area outside the 1-cm shield electrode should be left empty, as [Figure 6-31](#) shows. For improved liquid tolerance, there should not be any hatch fill or a trace connected to ground in the top and bottom layers of the PCB.

When there is a grounded hatch fill or a trace then, when a liquid droplet falls on the touch surface, it may cause sensor false triggers. Even if there is a shield electrode in between the sensor and ground, the effect of the shield electrode will be totally masked out and sensors may false trigger.

In some applications, there may not be sufficient area available on the PCB for shield electrode implementation. In such cases, the shield electrode can spread less than 1 cm; the minimum area for shield electrode can be the area remaining on the board after implementing the sensor.

Figure 6-31. Shield Electrode Placement when Sensor Trace is Routed in Top and Bottom Layer



Follow these guidelines to implement the shield electrode in two-layer and four-layer PCBs:

#### Two-Layer PCB:

- Top layer: Hatch fill with 7-mil trace and 45-mil grid (25 percent fill). Hatch fill should be connected to the driven-shield signal.
- Bottom layer: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to the driven-shield signal.

#### Four (or More)-Layer PCB:

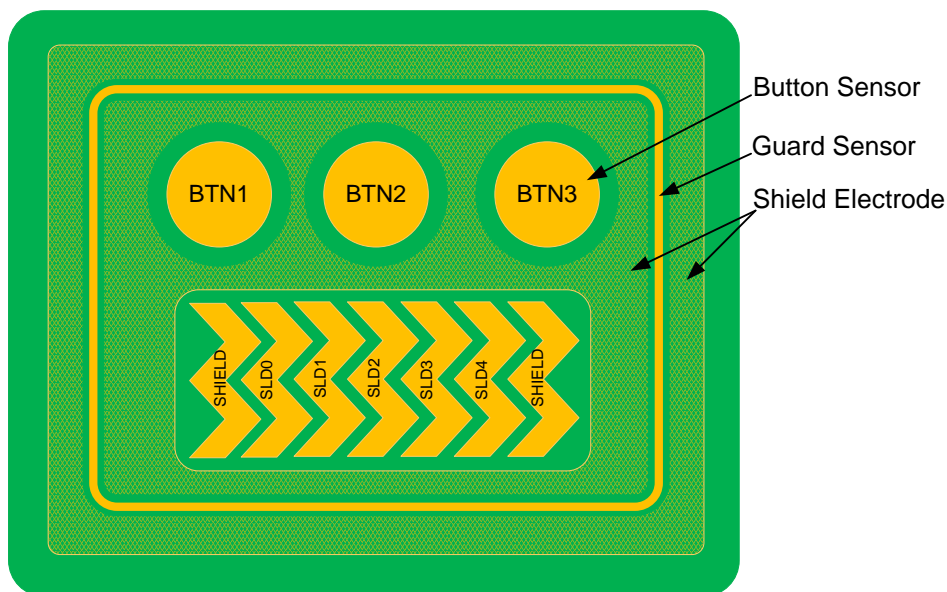
- Top layer: Hatch fill with 7-mil trace and 45-mil grid (25 percent fill). Hatch fill should be connected to the driven-shield signal.
- Layer-2: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to the driven-shield signal.
- Layer-3:  $V_{DD}$  Plane
- Bottom layer: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to ground.

The recommended air gap between the sensor and the shield electrode is 1 mm.

#### 6.4.11.1 Guard Sensor

As explained in the [Guard Sensor](#) section, the guard sensor is a copper trace that surrounds all of the sensors, as [Figure 6-32](#) shows.

Figure 6-32. PCB Layout with Shield Electrode and Guard Sensor



The guard sensor should be triggered only when there is a liquid stream on the touch surface. Make sure that the shield electrode pattern surrounds the guard sensor to prevent it from turning on due to liquid droplets. The guard sensor should be placed such that it meets the following conditions:

- It should be the first sensor to turn on when there is a liquid stream on the touch surface. To accomplish this, the guard sensor is usually placed such that it surrounds all sensors.
- It should not be accidentally touched while pressing a button or slider sensor. Otherwise, the button sensors and slider sensor scanning will be disabled and the CapSense system will become nonoperational until the guard sensor is turned off. To ensure the guard sensor is not accidentally triggered, place the guard sensor at a distance greater than 1 cm from the sensors.

Follow these guidelines to implement the guard sensor:

- The guard sensor should be in the shape of a rectangle with curved edges and should surround all the sensors.
- The recommended thickness for a guard sensor is 2 mm.
- The recommended clearance between the guard sensor and the shield electrode is 1 mm.

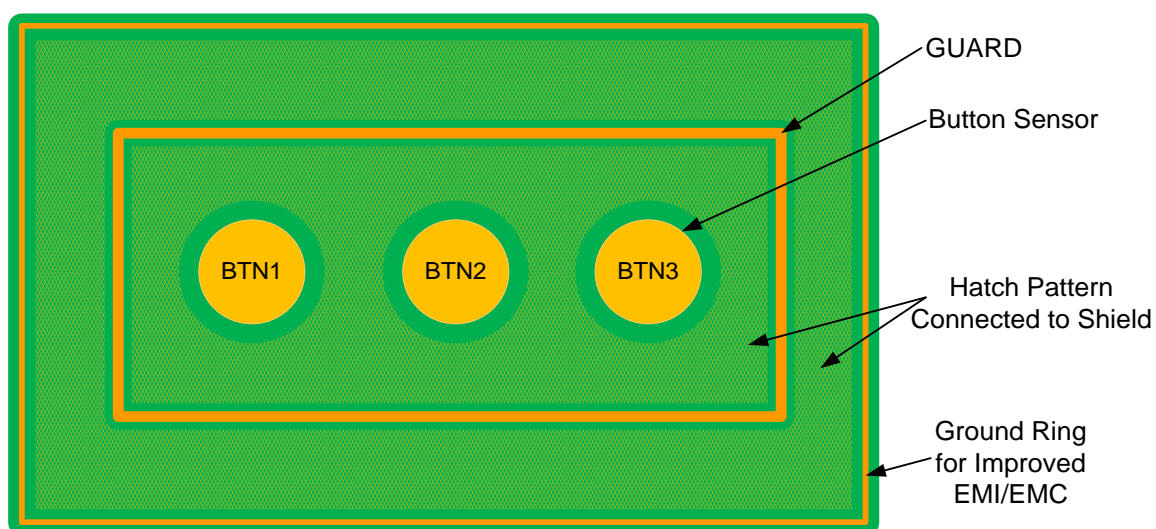
If there is no space on the PCB for implementing a guard sensor, the guard sensor functionality can be implemented in the firmware. For example, you can use the ON/OFF status of different sensors to detect a liquid stream. The following conditions can be used to detect a liquid stream on the touch surface:

- When there is a liquid stream, more than one button sensor will be active at a time. If your design does not require multi-touch sensing, you can detect this and reject the sensor status of all the button sensors to prevent false triggering.
- In a slider, if the slider segments which are turned ON are not adjacent segments, you can reset the slider segments status or reject the slider centroid value that is calculated.

#### 6.4.11.2 Liquid Tolerance with Ground Ring

In some applications, it is required to have a ground ring (solid trace or a hatch fill) around the periphery of the board for improved ESD and EMI/EMC performance, as shown in [Figure 6-33](#). Having a ground ring around the board may result in sensor false triggers when liquid droplets fall in between the sensor and the ground sensor. Therefore it is recommended not to have any ground in the top layer. If the design must have a ground ring in the top layer, use a ground ring with the minimum thickness (8 mils).

Figure 6-33. CapSense Design with a Ground Ring for Improved ESD and EMI/EMC Performance



#### 6.4.12 Schematic Rule Checklist

You can use the checklist provided to verify your CapSense schematic.

Table 6-6. Schematic Rule Checklist

No.	Category	Recommendations/Remarks
1	C <sub>MOD</sub>	2.2 nF. Refer to <a href="#">Table 6-7</a> for pin selection.
2	C <sub>SH_TANK</sub>	10 nF if shield electrode is being used, NA otherwise. Refer to <a href="#">Driven-Shield Signal and Shield Electrode</a> and <a href="#">CapSense CSD Shielding</a> for details on shield electrode and use of C <sub>SH_TANK</sub> respectively. Refer to <a href="#">Table 6-7</a> for pin selection.
3	C <sub>INTA</sub> /C <sub>INTB</sub>	470 pF. Refer to <a href="#">Table 6-7</a> for pin selection.
3	Series resistance on input lines	560 Ω. See <a href="#">Series Resistors on CapSense Pins</a> for details.
4	Sensor pin selection	If possible, avoid pins that are close to the GPIOs carrying switching/communication signals. Physically separate DC loads such as LEDs and I <sup>2</sup> C pins from the CapSense pins by a full port wherever possible. See <a href="#">Sensor and Device Placement</a> for details.
5	GPIO Source/Sink Current	Ensure that the total sink current through GPIOs is not greater than 40 mA when the CapSense block is scanning the sensors.



### 6.4.12.1 External Capacitors Pin Selection

**Table 6-7** lists the recommended pins for  $C_{MOD}$ ,  $C_{INTX}$ , and  $C_{SH\_TANK}$ . Note that pins other than recommended pins may be used for  $C_{MOD}$  with the following constraints:

1. The recommended  $C_{MOD}$  pin is always used because it is directly connected to the sigma delta converter. If a pin other than the recommended pin is selected for  $C_{MOD}$ , the recommended  $C_{MOD}$  pin will not be available for any other function. For example, if you try routing  $C_{MOD}$  to P2[0] in PSoC Creator for a PSoC 4200 device, it uses both P2[0] and P4[2].
2. If a pin other than the recommended pin is used for  $C_{MOD}$ , the shield cannot be used. This is because the  $C_{MOD}$  connection to a pin other than the recommended pin requires the use of AMUXBUS B. The shield signal requires the same AMUXBUS B as mentioned in the [CapSense CSD Shielding](#) section. If you want to use the shield,  $C_{MOD}$  should be placed on a recommended pin only.

Table 6-7. Recommended Pins for External Capacitors

No.	Type	Recommended Pin					
		PSoC 4000	PSoC 4100, PSoC 4200	PSoC 4200M/PSoC 4200L <sup>1</sup>	PSoC 4100-BL, PSoC 4200-BL	CYBL10X6X	PSoC 4-S
1	$C_{MOD}$	P0[4]	P4[2]	P4[2] – CSD0 P5[0] – CSD1	P4[0]	P4[0]	P4[2], P4[3] or P4[1]
2	$C_{SH\_TANK}$	P0[2]	P4[3]	P4[3] – CSD0 P5[1] – CSD1	P4[1]	P4[1]	Any GPIO Pin
3	$C_{INTA}$	P0[4]	P4[2]	P4[2]	P4[0]	P4[0]	P4[2]
4	$C_{INTB}$	P0[2]	P4[3]	P4[3]	P4[1]	P4[1]	P4[3]

### 6.4.13 Layout Rule Checklist

You can use the checklist provided in [Table 6-8](#) to help verify your layout design.

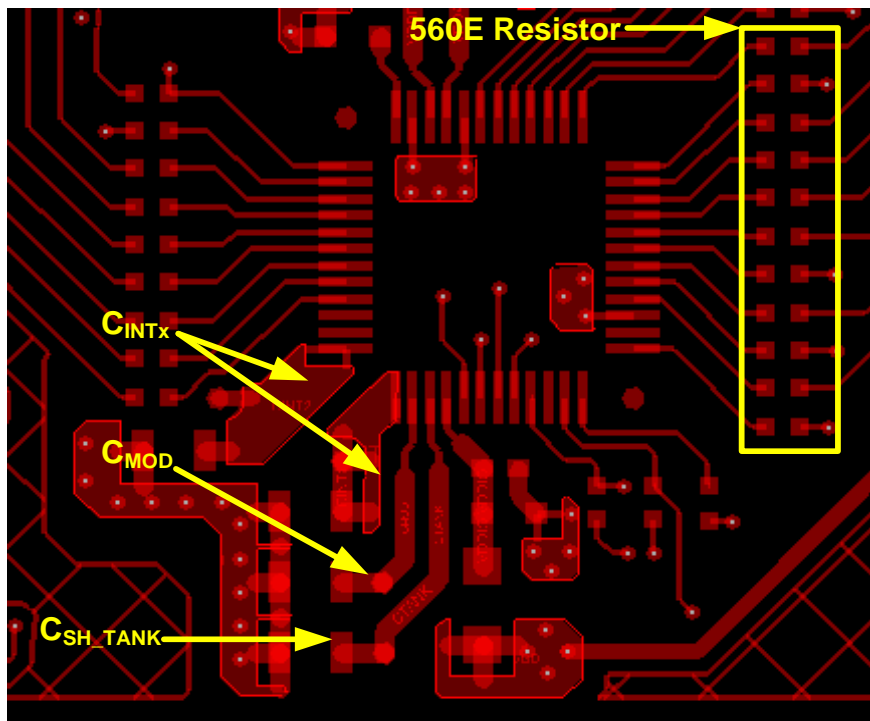
Table 6-8. Layout Rule Checklist

No.	Category		Minimum Value	Maximum Value	Recommendations / Remarks
1	Button	Shape	N/A	N/A	Circle or rectangular with curved edges
		Size	5 mm	15 mm	10 mm
		Clearance to ground hatch	0.5 mm	2 mm	Should be equal to overlay thickness
2	Slider	Width of segment	1.5 mm	8 mm	8 mm
		Clearance between segments	0.5 mm	2 mm	0.5 mm
		Height of segment	7 mm	15 mm	12 mm
3	Overlay	Type	N/A	N/A	Material with high relative permittivity (except conductors) Remove any air gap between sensor board and overlay / front panel of the casing.
		Thickness for buttons	N/A	5 mm	
		Thickness for sliders	N/A	5 mm	
		Thickness for touchpads	N/A	0.5 mm	
4	Sensor Traces	Width	N/A	7 mil	Use the minimum width possible with the PCB technology that you use.

<sup>1</sup> See [CapSense in PSoC 4xxxM/4xxxL-Series](#) section for details on CSD0 and CSD1 block.

No.	Category		Minimum Value	Maximum Value	Recommendations / Remarks
		Length	N/A	300 mm for a standard (FR4) PCB 50 mm for flex PCB	Keep as low as possible
		Clearance to ground and other traces	0.25 mm	N/A	Use maximum clearance while keeping the trace length as low as possible
		Routing	N/A	N/A	Route on the opposite side of the sensor layer. Isolate from other traces. If any non-CapSense trace crosses the CapSense trace, ensure that intersection is orthogonal. Do not use sharp turns.
5	Via	Number of vias	1	2	At least one via is required to route the traces on the opposite side of the sensor layer
		Hole size	N/A	N/A	10 mil
6	Ground	Hatch Fill Percentage	N/A	N/A	Use hatch ground to reduce parasitic capacitance. Typical hatching: 25% on the top layer (7-mil line, 45-mil spacing) 17% on the bottom layer (7-mil line, 70-mil spacing)
7	Series resistor	Placement	N/A	N/A	Place the resistor within 10 mm of the PSoC pin. Refer to <a href="#">Figure 6-34</a> for an example placement of series resistance on board.
8	Shield electrode	Spread	N/A	1 cm	If you have PCB space, use 1-cm spread.
9	Guard sensor (for water tolerance)	Shape	N/A	N/A	Rectangle with curved edges
		Thickness	N/A	N/A	Recommended thickness of guard trace is 2 mm and distance of guard trace to shield electrode is 1 mm.
10	C <sub>MOD</sub>	Placement	N/A	N/A	Place close to the PSoC pin. Refer to <a href="#">Figure 6-34</a> for an example placement of C <sub>MOD</sub> on printed circuit board.
11	C <sub>SH_TANK</sub>	Placement	N/A	N/A	Place close to the PSoC pin. Refer to <a href="#">Figure 6-34</a> for an example placement of C <sub>SH_TANK</sub> on board.
12	C <sub>INTA</sub>	Placement	N/A	N/A	Place close to the PSoC pin. Refer to <a href="#">Figure 6-34</a> for an example placement of C <sub>INTA</sub> on the PCB.
13	C <sub>INTB</sub>	Placement	N/A	N/A	Place close to the PSoC pin. Refer to <a href="#">Figure 6-34</a> for an example placement of C <sub>INTA</sub> on the PCB.

Figure 6-34. Example Placement for  $C_{MOD}$ ,  $C_{INTx}$ ,  $C_{SH\_TANK}$ , and Series Resistance on Input Lines in PSoC 4200M Device



## 6.5 ESD Protection

The nonconductive overlay material used in CapSense provides inherent protection against ESD. [Table 6-9](#) lists the thickness of various overlay materials, required to protect the CapSense sensors from a 12-kV discharge (according to the IEC 61000 - 4 - 2 specification).

Table 6-9. Overlay Thickness for ESD Protection

Material	Breakdown Voltage (V/mm)	Minimum Overlay Thickness for Protection Against 12 kV ESD (mm)
Air	1200 – 2800	10
Wood – dry	3900	3
Glass – common	7900	1.5
Glass – Borosilicate (Pyrex <sup>®</sup> )	13,000	0.9
PMMA Plastic (Plexiglas <sup>®</sup> )	13,000	0.9
ABS	16,000	0.8
Polycarbonate (Lexan <sup>®</sup> )	16,000	0.8
Formica	18,000	0.7
FR-4	28,000	0.4
PET Film (Mylar <sup>®</sup> )	280,000	0.04
Polyimide film (Kapton <sup>®</sup> )	290,000	0.04

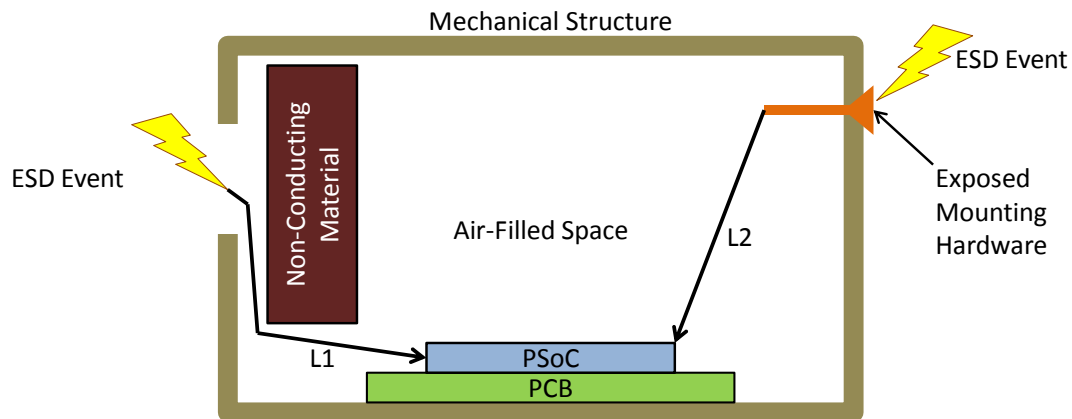
If the overlay material does not provide sufficient protection (for example, ESD from other directions), you can apply other ESD counter-measures, in the following order: [Prevent](#), [Redirect](#), [ESD protection devices](#).



### 6.5.1 Preventing ESD Discharge

Preventing the ESD discharge from reaching the PSoC is the best countermeasure you can take. Make sure that all paths to PSoC have a breakdown voltage greater than the maximum ESD voltage possible at the surface of the equipment. You should also maintain an appropriate distance between the PSoC and possible ESD sources. In the example illustrated in [Figure 6-35](#), if L1 and L2 are greater than 10 mm, the system can withstand a 12-kV ESD.

Figure 6-35. ESD Paths

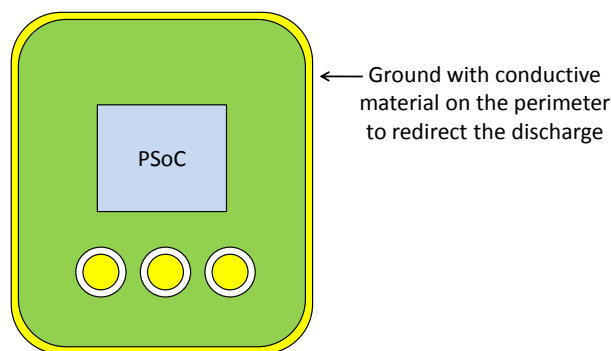


If it is not possible to maintain adequate distance, place a protective layer of nonconductive material with a high breakdown voltage between the possible ESD source and PSoC. One layer of 5-mil thick Kapton<sup>®</sup> tape can withstand 18 kV. See [Table 6-9](#) for other material dielectric strengths.

### 6.5.2 Redirect

If your product is densely packed, preventing the discharge event may not be possible. In such cases, you can protect the PSoC from ESD by redirecting the ESD. A standard practice is to place a ground ring on the perimeter of the circuit board, as [Figure 6-36](#) shows. The ground ring should connect to the chassis ground. Using a hatched ground plane around the button or slider sensor can also redirect the ESD event away from the sensor and PSoC.

Figure 6-36. Ground Ring



### 6.5.3 ESD Protection Devices

You can use ESD protection devices on vulnerable traces. Select ESD protection devices with a low input capacitance to avoid reduction in CapSense sensitivity. [Table 6-10](#) lists the recommended ESD protection devices.

Table 6-10. ESD Protection Devices

ESD Protection Device		Input Capacitance	Leakage Current	Contact Maximum ESD Limit	Air Discharge Maximum ESD Limit
Manufacturer	Part Number				
Littelfuse	SP723	5 pF	2 nA	8 kV	15 kV
Vishay	VBUS05L1-DD1	0.3 pF	0.1 $\mu$ A	$\pm$ 15 kV	$\pm$ 16 kV
NXP	NUP1301	0.75 pF	30 nA	8 kV	15 kV

## 6.6 Electromagnetic Compatibility (EMC) Considerations

EMC is related to the generation, transmission, and reception of electromagnetic energy that can affect the working of an electronic system. Electronic devices are required to comply with specific limits for emitted energy and susceptibility to external events. Several regulatory bodies worldwide set regional regulations to help ensure that electronic devices do not interfere with each other.

CMOS analog and digital circuits have very high input impedance. As a result, they are sensitive to external electric fields. Therefore, you should take adequate precautions to ensure their proper operation in the presence of radiated and conducted noise.

Computing devices are regulated in the US by the FCC under Part 15, Sub-Part B for unintentional radiators. The standards for Europe and the rest of the world are adapted from CENELEC. These are covered under CISPR standards (dual-labeled as ENxxxx standards) for emissions, and under IEC standards (also dual labeled as ENxxxx standards) for immunity and safety concerns.

The general emission specification is EN55022 for computing devices. This standard covers both radiated and conducted emissions. Medical devices in the US are not regulated by the FCC, but rather are regulated by FDA rules, which include requirements of EN55011, the European norm for medical devices. Devices that include motor controls are covered under EN55014 and lighting devices are covered under EN50015.

These specifications have essentially similar performance limitations for radiated and conducted emissions. Radiated and conducted immunity (susceptibility) performance requirements are specified by several sections of EN61000-4. Line voltage transients, electrostatic discharge (ESD) and some safety issues are also covered in this standard.

### 6.6.1 Radiated Interference and Emissions

While PSoC 4 and PRoC BLE offer a robust CapSense performance, radiated electrical energy can influence system measurements and potentially influence the operation of the CapSense processor core. Interference enters the CapSense device at the PCB level through sensor traces and through other digital and analog inputs. CapSense devices can also contribute to electromagnetic compatibility (EMC) issues in the form of radiated emissions.

Use the following techniques to minimize the radiated interference and emissions.

#### 6.6.1.1 Hardware Considerations

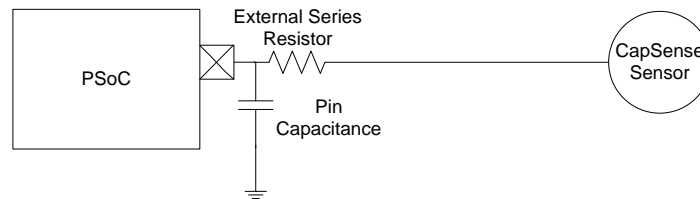
##### 6.6.1.1.1 Ground Plane

In general, proper ground plane on the PCB reduces both RF emissions and interference. However, solid grounds near CapSense sensors or traces connecting these sensors to PSoC pins increase the parasitic capacitance of the sensors. It is thus recommended to use hatched ground planes surrounding the sensor and on the bottom layer of the PCBs, below the sensors, as explained in the [Ground Plane](#) section in [PCB Layout Guidelines](#). Solid ground may be used below the device and other circuitry on the PCB which is farther from CapSense sensors and traces. A solid ground flood is not recommended within 1 cm of CapSense sensors or traces.

##### 6.6.1.1.2 Series Resistors on CapSense Pins

Every CapSense controller pin has some parasitic capacitance,  $C_p$ , associated with it. As [Figure 6-37](#) shows, adding an external resistor forms a low-pass RC filter that attenuates the RF noise amplitude coupled to the pin. This resistance also forms a low-pass filter with the parasitic capacitance of the CapSense sensor that significantly reduces the RF emissions.

Figure 6-37. RC Filter



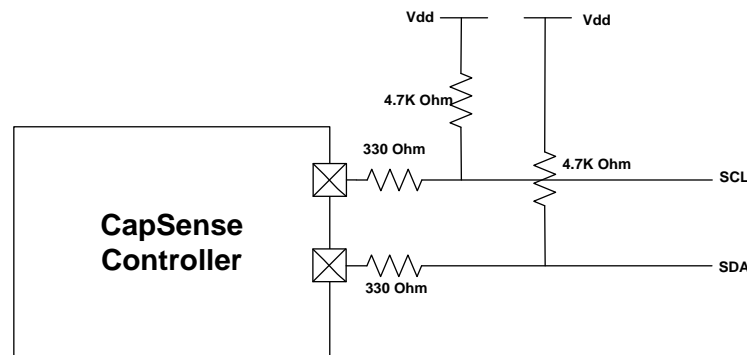
Series resistors should be placed close to the device pins so that the radiated noise picked by the traces gets filtered at the input of the device. Thus, it is recommended to place series resistors within 10 mm of the pins.

For CapSense designs using copper on PCBs, the recommended series resistance for CapSense input lines is 560  $\Omega$ . Adding resistance increases the time constant of the switched-capacitor circuit that converts  $C_P$  into an equivalent resistor; see [GPIO Cell Capacitance to Current Converter](#). If the series resistance value is larger than 560  $\Omega$ , the slower time constant of the switching circuit suppresses the emissions and interference, but limits the amount of charge that can transfer. This lowers the signal level, which in turn lowers the SNR. Smaller values are better in terms of SNR, but are less effective at blocking RF.

#### 6.6.1.1.3 Series Resistors on Digital Communication Lines

Communication lines, such as I<sup>2</sup>C and SPI, also benefit from series resistance; 330  $\Omega$  is the recommended value for series resistance on communication lines. Communication lines have long traces that act as antennae similar to the CapSense traces. The recommended pull-up resistor value for I<sup>2</sup>C communication lines is 4.7 k $\Omega$ . So, if more than 330  $\Omega$  is placed in series on these lines, the  $V_{IL}$  and  $V_{IH}$  voltage levels may fall out of specifications. 330  $\Omega$  will not affect I<sup>2</sup>C operation as the  $V_{IL}$  level still remains within the I<sup>2</sup>C specification limit of 0.3  $V_{DD}$  when PSoC outputs a LOW.

Figure 6-38. Series Resistors on Communication Lines



#### 6.6.1.1.4 Trace Length

Long traces can pick up more noise than short traces. Long traces also add to  $C_P$ . Minimize the trace length whenever possible.

#### 6.6.1.1.5 Current Loop Area

Another important layout consideration is to minimize the return path for currents. This is important as the current flows in loops. Unless there is a proper return path for high-speed signals, the return current will flow through a longer return path forming a larger loop, thus leading to increased emissions and interference.

If you isolate the CapSense ground hatch and the ground fill around the device, the sensor-switching current may take a longer return path, as [Figure 6-39](#) shows. As the CapSense sensors are switched at a high frequency, the return current may cause serious EMC issues. Therefore, you should use a single ground hatch, as [Figure 6-40](#) shows.

Figure 6-39. Improper Current Loop Layout

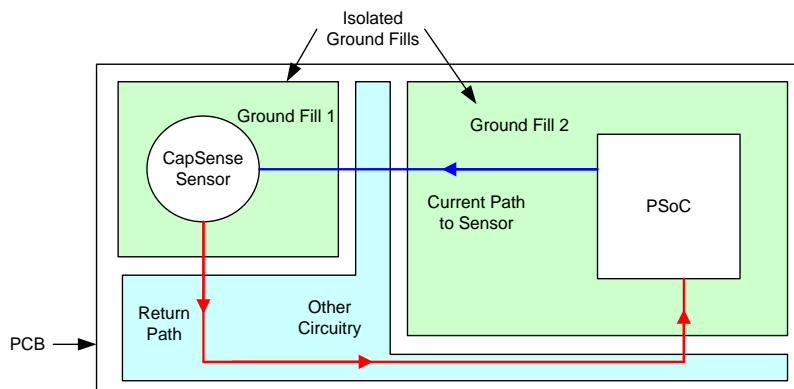
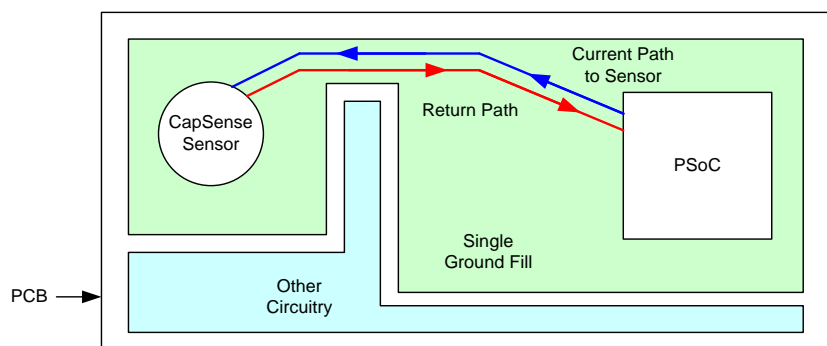


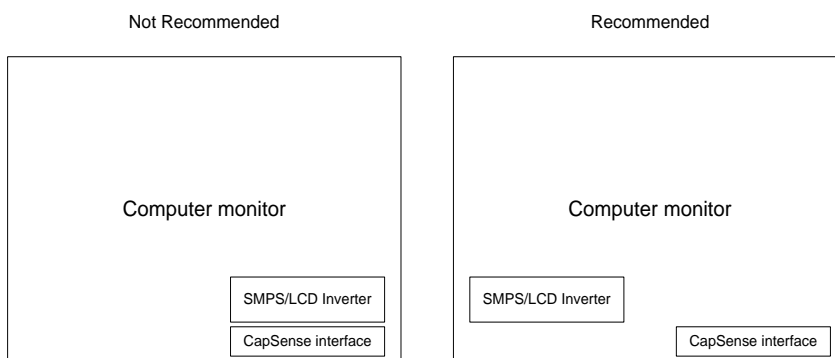
Figure 6-40. Proper Current Loop Layout



#### 6.6.1.1.6 RF Source Location

If your system has a circuit that generates RF noise, such as a switched-mode power supply (SMPS) or an inverter, you should place these circuits away from the CapSense interface. You should also shield such circuits to reduce the emitted RF. Figure 6-41 shows an example of separating the RF noise source from the CapSense interface.

Figure 6-41. Separating Noise Sources



#### 6.6.1.2 Firmware Considerations

The following parameters affect Radiated Emissions (RE) in a CapSense system:

- Device operating voltage
- Device operation frequency
- Sensor switching frequency
- Shield signal

- Sensor scan time
- Analog switch drive source
- Inactive sensor termination

The following sections explain the effect of each parameter.

#### 6.6.1.2.1 Device Operating Voltage

The emission is directly proportional to the voltage levels at which switching happens. Reducing the operating voltage helps to reduce the emissions as the amplitude of the switching signal at any output pin directly depends on the operating voltage of the device.

PSoC allows you to operate at lower operating voltages, thereby reducing the emissions. [Figure 6-42](#) and [Figure 6-43](#) show the impact of operating voltage on radiated emissions. Because  $IMO = 24\text{ MHz}$  and the other spikes are caused by different hardware and firmware operations of the device.

Figure 6-42. Effect of  $V_{DD}$  on Radiated Emissions (150 kHz – 30 MHz)

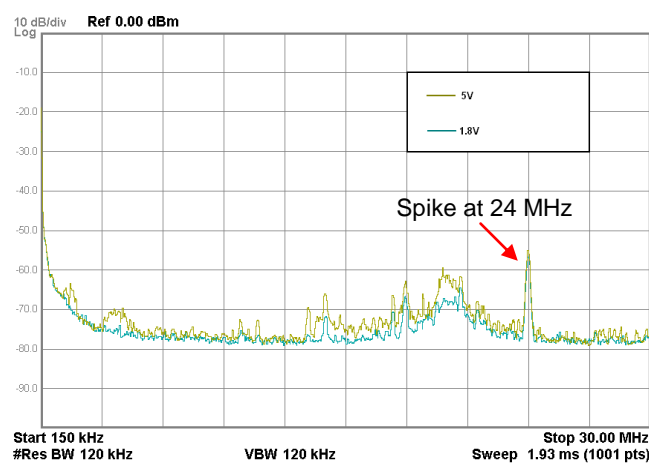
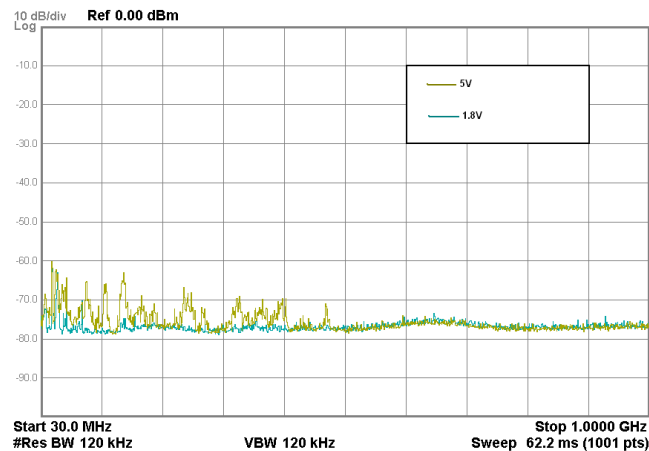


Figure 6-43. Effect of  $V_{DD}$  on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale

#### 6.6.1.2.2 Device Operating Frequency

Reducing the system clock frequency ( $IMO$  frequency) reduces radiated emissions. However, reducing the  $IMO$  frequency may not be feasible in all applications because the  $IMO$  frequency impacts the CPU clock and all other system timings. Choose a suitable  $IMO$  frequency based on your application.

### 6.6.1.2.3 Sensor-Switching Frequency

Reducing the sensor-switching frequency (see [Sense Clock](#)) also helps to reduce radiated emissions. See [Figure 6-44](#) and [Figure 6-45](#). Because  $IMO = 24\text{ MHz}$  and the other spikes are caused by different hardware and firmware operations of the device.

Figure 6-44. Effect of Sensor-Switching Frequency on Radiated Emissions (150 kHz – 30 MHz)

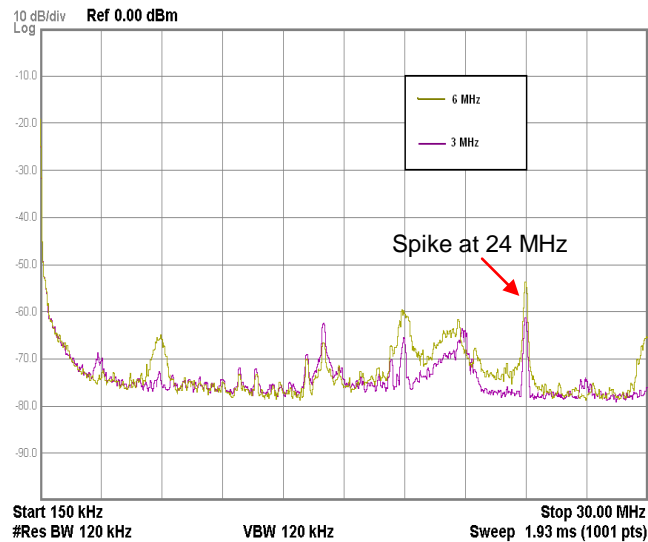
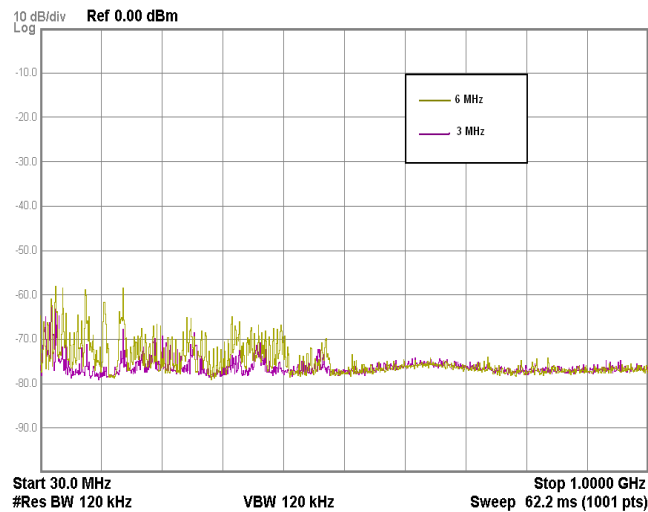


Figure 6-45. Effect of Sensor-Switching Frequency on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale

#### 6.6.1.2.3.1 Pseudo Random Sense Clock

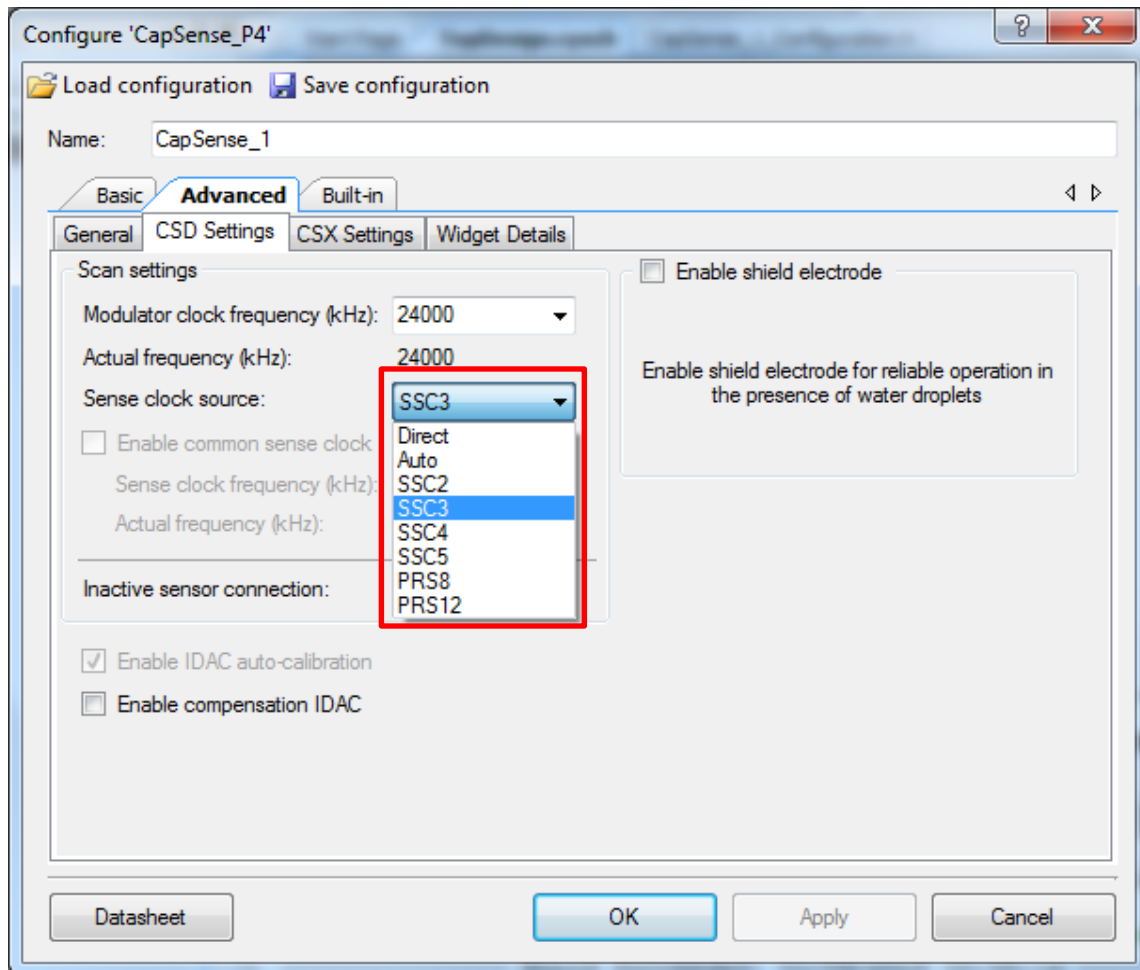
The PSoC 4 device supports PRS-based sense clock generation. A PRS is used instead of a fixed clock source to attenuate emitted noise on the CapSense pins by reducing the amount of EMI created by a fixed-frequency source and to increase EMI immunity from other sources and their harmonics.

#### 6.6.1.2.3.2 Spread Spectrum Sense Clock

In addition to the PRS-based clock generation, the PSoC 4 S-Series family of devices supports a unique feature called spread spectrum sense clock generation, in which the sense clock frequency is spread over a desired range. This method will help to reduce the peaks and spread out the emissions over a range of frequencies. The spread spectrum clock can be enabled by selecting the “Sense Clock Source” as “SSCn”. The range of frequency spread is decided by the length of the

register. For more details on the spread spectrum clock generation in the PSoC 4 S-Series, refer to the Spread Spectrum Clock section in the CapSense chapter of the respective device [Technical Reference Manual](#).

Figure 6-46. Sense Clock Sources in PSoC 4 S-Series



#### 6.6.1.2.4 Shield Signal

Enabling the shield signal (see [Driven-Shield Signal and Shield Electrode](#)) on the hatch pattern increases the radiated emissions. Enable the driven-shield signal only for liquid-tolerant, proximity-sensing, or high-parasitic-capacitance designs. Also, if the shield must be used, ensure that the shield electrode area is limited to a width of 1 cm from the sensors, as [Figure 6-31](#) shows.

[Figure 6-47](#) and [Figure 6-48](#) show the impact of enabling the driven-shield signal on the hatch pattern surrounding the sensors on radiated emissions. Note that in these figures, the hatch pattern is grounded when the driven-shield signal is disabled. Because  $f_{IMO} = 24 \text{ MHz}$ , there is a spike at 24 MHz and the other spikes are caused by different hardware and firmware operations of the device.

Figure 6-47. Effect of Shield Electrode on Radiated Emissions (150 kHz – 30 MHz)

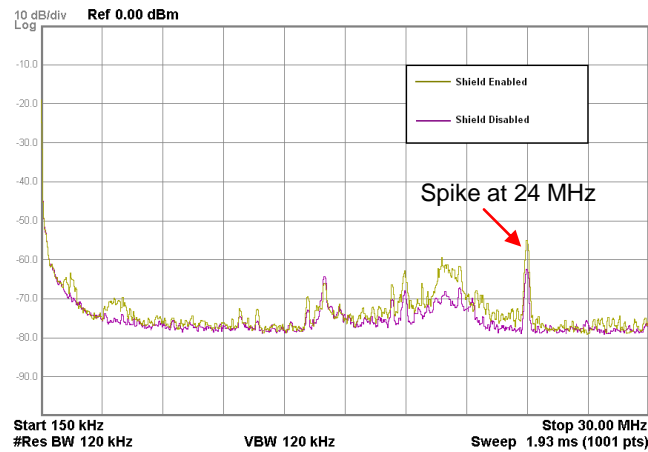
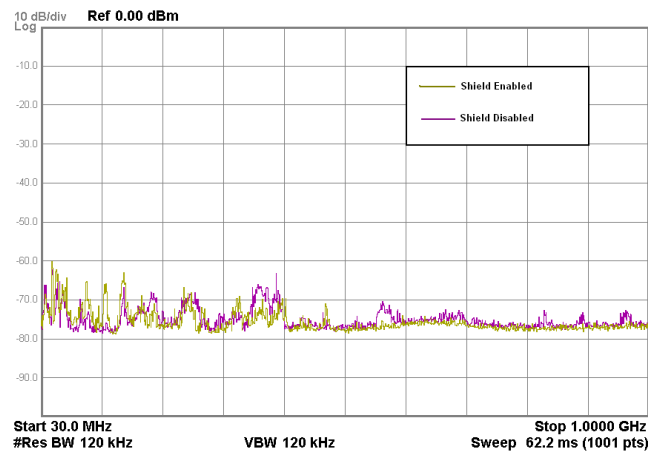


Figure 6-48. Effect of Shield Electrode on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale

#### 6.6.1.2.5 Sensor Scan Time

Reducing the sensor scan time reduces the average radiated emissions. The sensor-scan time depends on the scan resolution and modulator clock divider (See [Equation 3-7: Sensor Scan Time](#)). Increasing the scan resolution or modulator clock divider increases the scan time. [Figure 6-49](#) and [Figure 6-50](#) show the impact of sensor scan time on radiated emissions. Note that, here, the sensor scan time was varied by changing the scan resolution. Because IMO = 24 MHz, there is a spike at 24 MHz and the other spikes are caused by different hardware and firmware operations of the device.

Table 6-11. Sensor Scan Time

Parameter	Total Scan time for 5 Buttons	
	0.426 ms	0.106 ms
Modulation Clock Divider	2	2
Scan Resolution	10 bits	8 bits
Individual Sensor Scan Time	0.085 ms	0.021 ms



Figure 6-49. Effect of Scan Time on Radiated Emissions (150 kHz – 30 MHz)

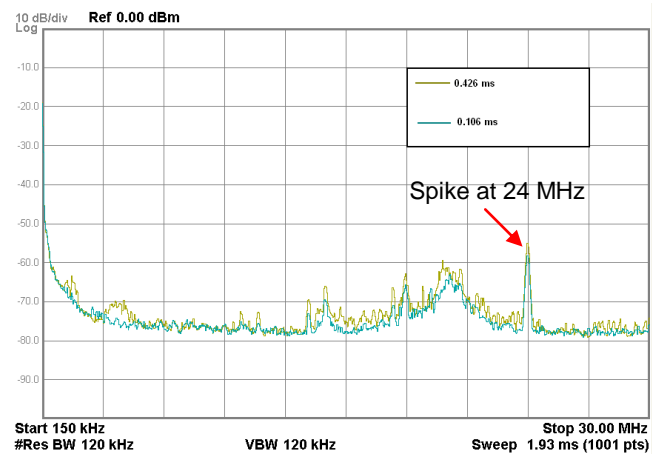
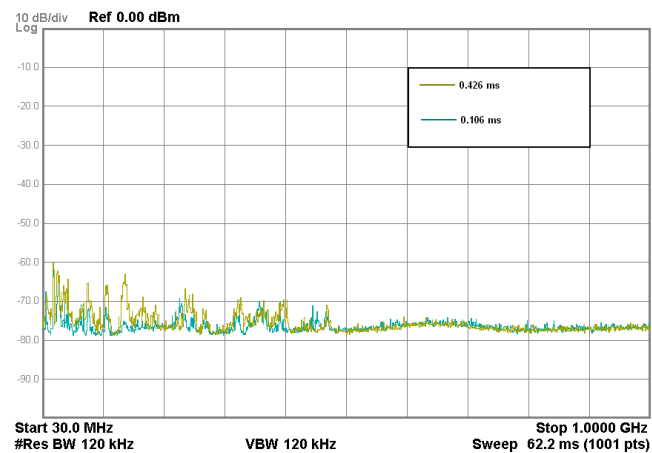


Figure 6-50. Effect of Scan Time on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale

### 6.6.1.2.6 Analog Switch Drive Source

Using PRS instead of direct clock drive as [analog switch drive source](#) spreads the radiated spectrum and hence reduces the average radiated emissions. See [Figure 6-51](#) and [Figure 6-52](#). Because IMO = 24 MHz, there is a spike at 24 MHz and the other spikes are caused by different hardware and firmware operations of the device.

Figure 6-51. Effect of Analog Switch Drive Source on Radiated Emissions (150 kHz – 30 MHz)

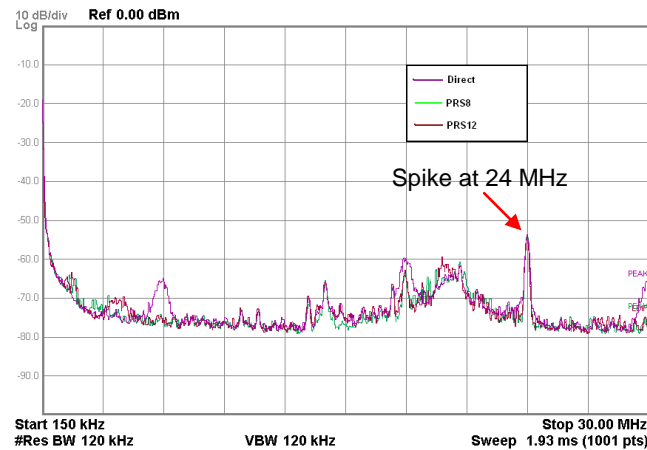
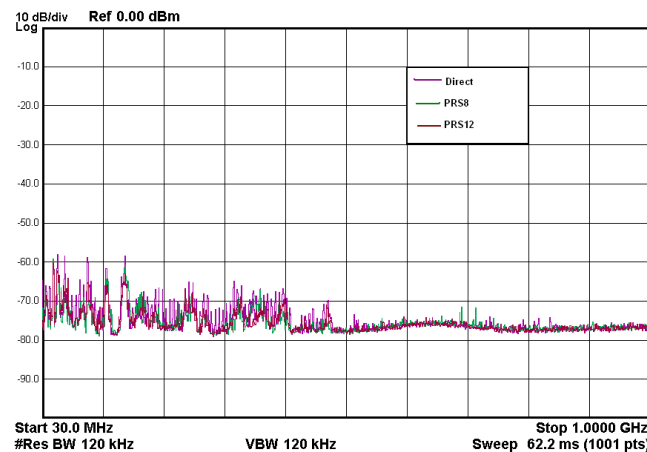


Figure 6-52. Effect of Analog Switch Drive Source on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale.

### 6.6.1.2.7 Inactive Sensor Termination

Connecting inactive sensors to ground reduces the radiated emission by a greater degree than connecting them to the shield. [Figure 6-53](#) and [Figure 6-54](#) show the impact of different inactive sensor terminations on radiated emission. Because  $f_{IMO} = 24 \text{ MHz}$ , there is a spike at 24 MHz and the other spikes are caused by different hardware and firmware operations of the device.

Figure 6-53. Effect of Inactive Sensor Termination on Radiated Emissions (150 kHz – 30 MHz)

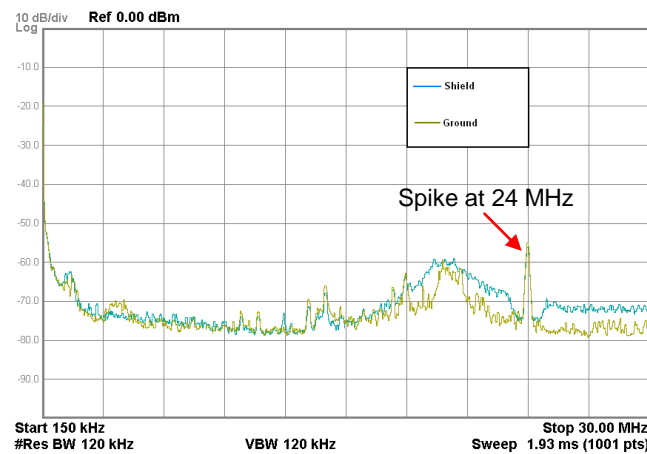
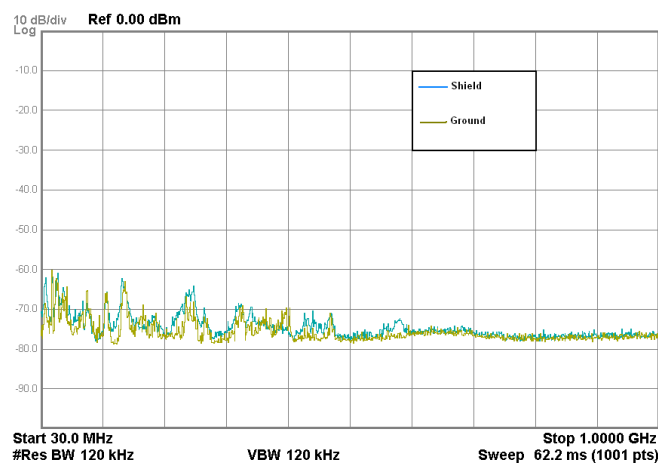


Figure 6-54. Effect of Inactive Sensor Termination on Radiated Emissions (30 MHz – 1 GHz)



**Note** Frequency axis is in log scale.

## 6.6.2 Conducted RF Noise

The noise current that enters the CapSense system through the power and communication lines is called conducted noise. You can use the following techniques to reduce the conducted RF noise.

- Use decoupling capacitors on the power supply pins to reduce the conducted noise from the power supply. See [section 6.4.10](#) and the device [Datasheet](#) for details.
- Provide GND and  $V_{DD}$  planes on the PCB to reduce current loops.
- If the PSoC PCB is connected to the power supply using a cable, minimize the cable length and consider using a shielded cable.

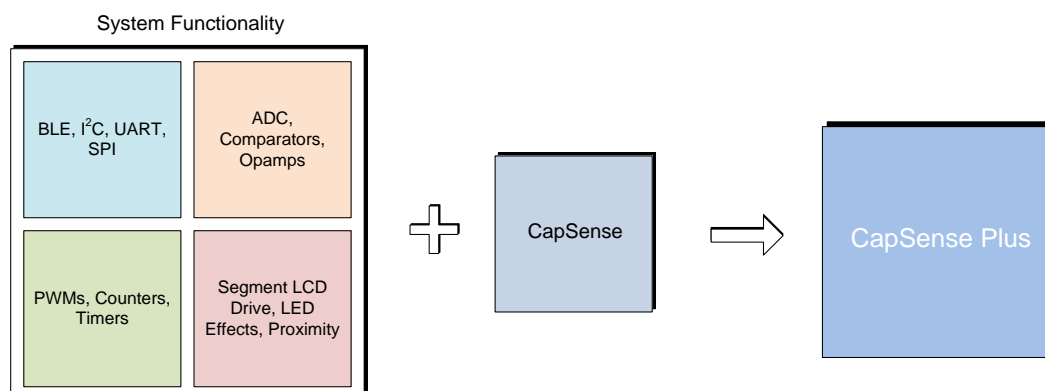
To reduce high-frequency noise, place a ferrite bead around power supply or communication lines.

## 7. CapSense Plus



PSoC 4 and PSoC BLE can perform many additional functions along with CapSense. The wide variety of features offered by this device allows you to integrate various system functions in a single chip, as [Figure 7-1](#) shows. Such applications are known as CapSense Plus applications.

Figure 7-1. CapSense Plus



The additional features available in a PSoC 4 or PSoC BLE device include:

- Communication: BLE, I<sup>2</sup>C, UART, SPI, CAN, and LIN
- Analog functions: ADC, comparators, and opamps
- Digital functions: PWMs, counters, timers, and UDBs
- Segment LCD drive
- Bootloaders
- Different power modes: Active, Sleep, Deep Sleep, Hibernate, and Stop

For more information on PSoC 4, refer to [AN79953, Getting Started with PSoC 4](#), or [AN91267, Getting Started with PSoC 4 BLE](#). For more information on PSoC BLE, refer to [AN94020, Getting Started with PSoC BLE](#).

The flexibility of the PSoC 4 and PSoC BLE and the unique PSoC Creator IDE allow you to quickly make changes to your design, which accelerates time-to-market. Integrating other system functions significantly reduces overall system cost. Table 7-1 shows a list of example applications, where using CapSense Plus can result in significant cost savings.

Table 7-1. Examples of CapSense Plus

Application	CapSense	Opamp	ADC	Comp	PWM, Counter, Timer, UDBs	Comm (BLE, I <sup>2</sup> C, SPI, UART)	LCD drive	GPIOs
Heart rate monitor (wrist band)	User interface: buttons, linear sliders	TIA, Buffer	Heart Rate Measurement, Battery voltage measurement		LED Driving	BLE	Segment LCD	LED indication
LED bulb	User interface: buttons, radial sliders	Amplifier	LED current measurement	Short Circuit Protection	LED color control (PrISM*)	BLE		LED indication
Washing machine	User interface: buttons, radial sliders		Temperature sensor	Water level monitor	Buzzer, FOC** motor control	I <sup>2</sup> C LCD display, UART network interface	Segment LCD	LED indication
Water heater	User interface: buttons, linear sliders		Temperature sensor, water flux sensor	Water level monitor	Buzzer	I <sup>2</sup> C LCD display, UART Network Interface	Segment LCD	LED indication
IR remote controllers	User interface: buttons, linear and radial sliders, touchpads				Manchester encoder			LED indication
Induction cookers	User interface: buttons, linear sliders		Temperature sensor				Segment LCD	LED indication
Motor control systems	User interface: buttons, linear sliders				BLDC*** and FOC motor control			LED indication
Gaming / simulation controllers	User interface: buttons, touchpads		Reading analog joysticks			I <sup>2</sup> C/SPI/UART communication interface	Segment LCD	LED indication
Thermal printers	User interface: buttons		Overheat protection, paper sensor		Stepper motor control	SPI communication interface		LED indication

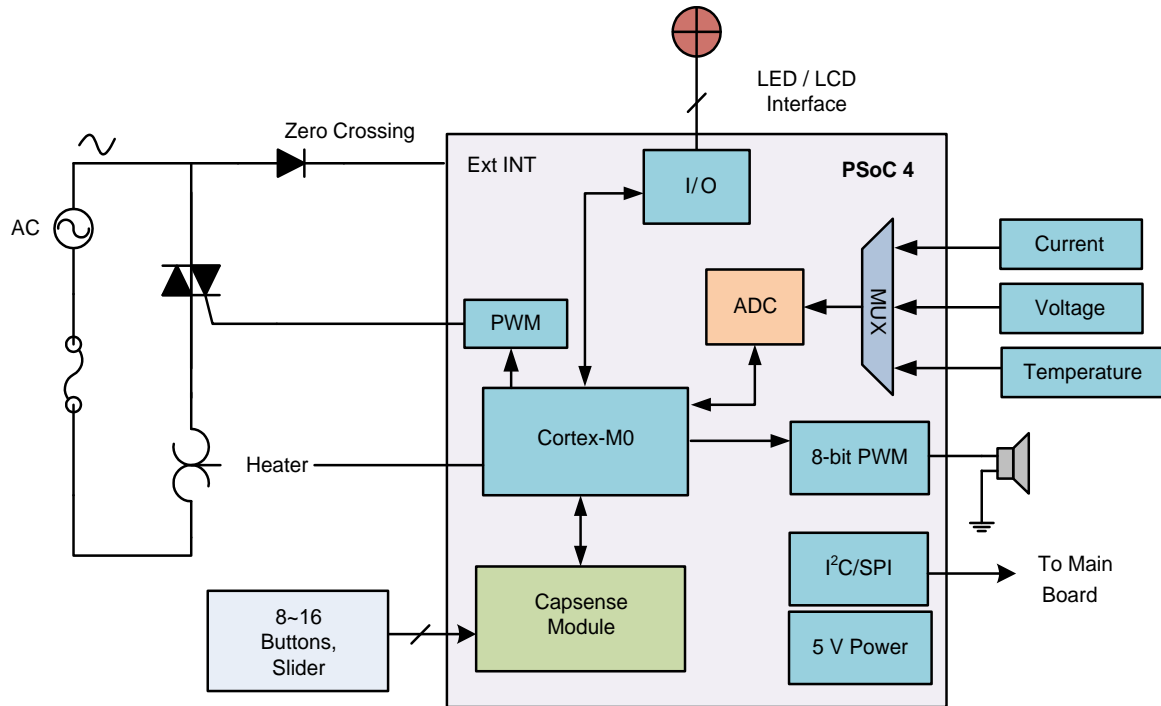
\* PrISM stands for Precision Illumination Signal Modulation

\*\* FOC stands for Field Oriented Control

\*\*\* BLDC stands for Brushless DC Motor

Figure 7-2 shows a general block diagram of a CapSense Plus application, such as an induction cooker or a microwave oven.

Figure 7-2. CapSense Plus System with PSoC 4



In this application, the 12-bit 1-Msps SAR ADC in the PSoC 4 detects over-current, overvoltage, and high temperature conditions. The PWM output drives the speaker for status and alarm tones. Another PWM controls the heating element in the system. The CapSense buttons and slider constitute the user interface. PSoC 4 can also drive a segment LCD for visual outputs. PSoC 4 has a serial communication block that can connect to the main board of the system.

CapSense Plus systems, such as this example, allow you to reduce your board size, BOM cost, and power consumption.

## 8. Resources



### 8.1 Website

Visit the [Getting Started with PSoC 4](#), [Getting Started with PSoC 4 BLE](#), or [Getting Started with PRoC BLE](#) website to understand the PSoC 4 and PRoC BLE devices.

### 8.2 Datasheet

[PSoC 4 Datasheet](#)

[PSoC 4 BLE Datasheet](#)

[PRoC BLE Datasheet](#)

### 8.3 Technical Reference Manual

The [PSoC 4 Technical Reference Manual \(TRM\)](#) provides quick and easy access to information on PSoC 4 architecture including top-level architectural diagrams, register summaries, and timing diagrams.

### 8.4 Development Kits

[Table 8-1](#) lists Cypress development kits that support PSoC 4 CapSense.

Table 8-1. PSoC 4 CapSense Development Kits

Development Kit	Supported CapSense Features
PSoC 4000 Pioneer Kit ( <a href="#">CY8CKIT-040</a> )	A 5x6 CapSense touchpad and a wire proximity sensor
PSoC 4 S-Series Pioneer Kit ( <a href="#">CY8CKIT-041</a> )	Two self- or mutual-capacitive sensing buttons A 7x7 self- or mutual-capacitive sensing touchpad
PSoC 4 S-Series Prototyping Kit ( <a href="#">CY8CKIT-145</a> )	Three self- or mutual-capacitive sensing buttons A five-segment self- or mutual-capacitive sensing linear slider
PSoC 4 Pioneer Kit ( <a href="#">CY8CKIT-042</a> )	A five-segment linear slider
PSoC 4 BLE Bluetooth Low Energy Pioneer Kit ( <a href="#">CY8CKIT-042-BLE</a> )	A five-segment linear slider and a wire proximity sensor
PSoC 4200-M Pioneer Kit( <a href="#">CY8CKIT-044</a> )	A five-element gesture detection and two proximity wire sensors
PSoC 4200-L Pioneer Kit( <a href="#">CY8CKIT-046</a> )	A five-element gesture detection, two proximity wire sensors, and an eight-element radial slider
CapSense Proximity Shield ( <a href="#">CY8CKIT-024</a> )	A four-element gesture detection and one proximity loop sensor
CapSense® Liquid Level Sensing Shield ( <a href="#">CY8CKIT-022</a> )	A two-element flexible PCB and 12-element flexible PCB
PSoC 4 Processor Module ( <a href="#">CY8CKIT-038</a> ), with PSoC Development Kit ( <a href="#">CY8CKIT-001</a> )	A five-segment linear slider and two buttons

Development Kit	Supported CapSense Features
CapSense Expansion Board Kit (CY8CKIT-031), to be used with CY8CKIT-038 and CY8CKIT-001	A 10-segment slider, five buttons, and a 4x4 matrix button with LED indication.
MiniProg3 Program and Debug Kit (CY8CKIT-002)	CapSense performance tuning in CY8CKIT-038

Table 8-2 lists Cypress Reference Design kits that support PSoC BLE CapSense.

Table 8-2. PSoC BLE CapSense Reference Design Kits

Reference Design Kit	Supported CapSense Features
PSoC BLE Remote Control Reference Design Kit (CY5672)	An 8 x 8 trackpad with these gestures: Top edge swipe gesture One finger click gesture Two finger click gesture Pinch zoom gesture One finger horizontal scroll gesture One finger vertical scroll gesture
PSoC BLE Touch Mouse Reference Design Kit (CY5682)	A 9 x 3 trackpad with these gestures: One-finger horizontal scroll gesture One-finger vertical scroll gesture

## 8.5 PSoC Creator

PSoC Creator is a state-of-the-art, easy-to-use integrated development environment. For details, see the [PSoC Creator home page](#).

## 8.6 Application Notes

Cypress offers a large collection of application notes to get your design up and running fast. See [PSoC 4 Application Notes](#), [PSoC 4 BLE Application Notes](#) or [PSoC BLE Application Notes, CapSense Application Notes and Design Guides](#).

## 8.7 Design Support

Cypress has a variety of design support channels to ensure the success of your CapSense solutions.

- [Knowledge Base Articles](#) – Browse technical articles by product family or perform a search on CapSense topics.
- [White Papers](#) – Learn about advanced capacitive-touch interface topics.
- [Cypress Developer Community](#) – Connect with the Cypress technical community and exchange information.
- [Video Library](#) – Quickly get up to speed with tutorial videos.
- [Quality & Reliability](#) – Cypress is committed to complete customer satisfaction. At our Quality website, you can find reliability and product qualification reports.
- [Technical Support](#) – Submit your design for review by creating a Cypress Support Case. You need to register and login at Cypress website to be able to contact [technical support](#). Cypress recommends PDF prints for the schematic and Gerber files with layer information for the layout.



# Glossary



## **AMUXBUS**

Analog multiplexer bus available inside PSoC that helps to connect I/O pins with multiple internal analog signals.

## **SmartSense™ Auto-Tuning**

A CapSense algorithm that automatically sets sensing parameters for optimal performance after the design phase and continuously compensates for system, manufacturing, and environmental changes.

## **Baseline**

A value resulting from a firmware algorithm that estimates a trend in the Raw Count when there is no human finger present on the sensor. The Baseline is less sensitive to sudden changes in the Raw Count and provides a reference point for computing the Difference Count.

## **Button or Button Widget**

A widget with an associated sensor that can report the active or inactive state (that is, only two states) of the sensor. For example, it can detect the touch or no-touch state of a finger on the sensor.

## **Difference Count**

The difference between Raw Count and Baseline. If the difference is negative, or if it is below Noise Threshold, the Difference Count is always set to zero.

## **Capacitive Sensor**

A conductor and substrate, such as a copper button on a printed circuit board (PCB), which reacts to a touch or an approaching object with a change in capacitance.

## **CapSense®**

Cypress's touch-sensing user interface solution. The industry's No. 1 solution in sales by 4x over No. 2.

## **CapSense Mechanical Button Replacement (MBR)**

Cypress's configurable solution to upgrade mechanical buttons to capacitive buttons, requires minimal engineering effort to configure the sensor parameters and does not require firmware development. These devices include the CY8CMBR3XXX and CY8CMBR2XXX families.

## **Centroid or Centroid Position**

A number indicating the finger position on a slider within the range given by the Slider Resolution. This number is calculated by the CapSense centroid calculation algorithm.

## **Compensation IDAC**

A programmable constant current source, which is used by CSD to compensate for excess sensor  $C_p$ . This IDAC is not controlled by the Sigma-Delta Modulator in the CSD block unlike the Modulation IDAC.

## **CSD**

CapSense Sigma Delta (CSD) is a Cypress-patented method of performing self-capacitance (also called self-cap) measurements for capacitive sensing applications.

In CSD mode, the sensing system measures the self-capacitance of an electrode, and a change in the self-capacitance is detected to identify the presence or absence of a finger.

**Debounce**

A parameter that defines the number of consecutive scan samples for which the touch should be present for it to become valid. This parameter helps to reject spurious touch signals.

A finger touch is reported only if the Difference Count is greater than Finger Threshold + Hysteresis for a consecutive Debounce number of scan samples.

**Driven-Shield**

A technique used by CSD for enabling liquid tolerance in which the Shield Electrode is driven by a signal that is equal to the sensor switching signal in phase and amplitude.

**Electrode**

A conductive material such as a pad or a layer on PCB, ITO, or FPCB. The electrode is connected to a port pin on a CapSense device and is used as a CapSense sensor or to drive specific signals associated with CapSense functionality.

**Finger Threshold**

A parameter used with Hysteresis to determine the state of the sensor. Sensor state is reported ON if the Difference Count is higher than Finger Threshold + Hysteresis, and it is reported OFF if the Difference Count is below Finger Threshold – Hysteresis.

**Ganged Sensors**

The method of connecting multiple sensors together and scanning them as a single sensor. Used for increasing the sensor area for proximity sensing and to reduce power consumption.

To reduce power when the system is in low-power mode, all the sensors can be ganged together and scanned as a single sensor taking less time instead of scanning all the sensors individually. When you touch any of the sensors, the system can transition into active mode where it scans all the sensors individually to detect which sensor is activated.

PSoC supports sensor-ganging in firmware, that is, multiple sensors can be connected simultaneously to AMUXBUS for scanning.

**Gesture**

Gesture is an action, such as swiping and pinch-zoom, performed by the user. CapSense has a gesture detection feature that identifies the different gestures based on predefined touch patterns. In the CapSense Component, the Gesture feature is supported only by the Touchpad Widget.

**Guard Sensor**

Copper trace that surrounds all the sensors on the PCB, similar to a button sensor and is used to detect a liquid stream. When the Guard Sensor is triggered, firmware can disable scanning of all other sensors to prevent false touches.

**Hatch Fill or Hatch Ground or Hatched Ground**

While designing a PCB for capacitive sensing, a grounded copper plane should be placed surrounding the sensors for good noise immunity. But a solid ground increases the parasitic capacitance of the sensor which is not desired. Therefore, the ground should be filled in a special hatch pattern. A hatch pattern has closely-placed, crisscrossed lines looking like a mesh and the line width and the spacing between two lines determine the fill percentage. In case of liquid tolerance, this hatch fill referred as a shield electrode is driven with a shield signal instead of ground.

**Hysteresis**

A parameter used to prevent the sensor status output from random toggling due to system noise, used in conjunction with the Finger Threshold to determine the sensor state. See [Finger Threshold](#).

**IDAC (Current-Output Digital-to-Analog Converter)**

Programmable constant current source available inside PSoC, used for CapSense and ADC operations.

**Liquid Tolerance**

The ability of a capacitive sensing system to work reliably in the presence of liquid droplets, streaming liquids or mist.

**Linear Slider**

A widget consisting of more than one sensor arranged in a specific linear fashion to detect the physical position (in single axis) of a finger.

**Low Baseline Reset**

A parameter that represents the maximum number of scan samples where the Raw Count is abnormally below the Negative Noise Threshold. If the Low Baseline Reset value is exceeded, the Baseline is reset to the current Raw Count.

**Manual-Tuning**

The manual process of setting (or tuning) the CapSense parameters.

**Matrix Buttons**

A widget consisting of more than two sensors arranged in a matrix fashion, used to detect the presence or absence of a human finger (a touch) on the intersections of vertically and horizontally arranged sensors.

If M is the number of sensors on the horizontal axis and N is the number of sensors on the vertical axis, the Matrix Buttons Widget can monitor a total of M x N intersections using ONLY M + N port pins.

When using the CSD sensing method (self-capacitance), this Widget can detect a valid touch on only one intersection position at a time.

**Modulation Capacitor (CMOD)**

An external capacitor required for the operation of a CSD block in Self-Capacitance sensing mode.

**Modulator Clock**

A clock source that is used to sample the modulator output from a CSD block during a sensor scan. This clock is also fed to the Raw Count counter. The scan time (excluding pre and post processing times) is given by  $(2^N - 1)/\text{Modulator Clock Frequency}$ , where N is the Scan Resolution.

**Modulation IDAC**

Modulation IDAC is a programmable constant current source, whose output is controlled (ON/OFF) by the sigma-delta modulator output in a CSD block to maintain the AMUXBUS voltage at  $V_{REF}$ . The average current supplied by this IDAC is equal to the average current drawn out by the sensor capacitor.

**Mutual-Capacitance**

Capacitance associated with an electrode (say TX) with respect to another electrode (say RX) is known as mutual capacitance.

**Negative Noise Threshold**

A threshold used to differentiate usual noise from the spurious signals appearing in negative direction. This parameter is used in conjunction with the Low Baseline Reset parameter.

Baseline is updated to track the change in the Raw Count as long as the Raw Count stays within Negative Noise Threshold, that is, the difference between Baseline and Raw count ( $\text{Baseline} - \text{Raw count}$ ) is less than Negative Noise Threshold.

Scenarios that may trigger such spurious signals in a negative direction include: a finger on the sensor on power-up, removal of a metal object placed near the sensor, removing a liquid-tolerant CapSense-enabled product from the water; and other sudden environmental changes.

**Noise (CapSense Noise)**

The variation in the Raw Count when a sensor is in the OFF state (no touch), measured as peak-to-peak counts.

**Noise Threshold**

A parameter used to differentiate signal from noise for a sensor. If  $\text{Raw Count} - \text{Baseline}$  is greater than Noise Threshold, it indicates a likely valid signal. If the difference is less than Noise Threshold, Raw Count contains nothing but noise.

**Overlay**

A non-conductive material, such as plastic and glass, which covers the capacitive sensors and acts as a touch-surface. The PCB with the sensors is directly placed under the overlay or is connected through springs. The casing for a product often becomes the overlay.

**Parasitic Capacitance ( $C_P$ )**

Parasitic capacitance is the intrinsic capacitance of the sensor electrode contributed by PCB trace, sensor pad, vias, and air gap. It is unwanted because it reduces the sensitivity of CSD.

**Proximity Sensor**

A sensor that can detect the presence of nearby objects without any physical contact.

**Radial Slider**

A widget consisting of more than one sensor arranged in a specific circular fashion to detect the physical position of a finger.

**Raw Count**

The unprocessed digital count output of the CapSense hardware block that represents the physical capacitance of the sensor.

**Refresh Interval**

The time between two consecutive scans of a sensor.

**Scan Resolution**

Resolution (in bits) of the Raw Count produced by the CSD block.

**Scan Time**

Time taken for completing the scan of a sensor.

**Self-Capacitance**

The capacitance associated with an electrode with respect to circuit ground.

**Sensitivity**

The change in Raw Count corresponding to the change in sensor capacitance, expressed in counts/pF. Sensitivity of a sensor is dependent on the board layout, overlay properties, sensing method, and tuning parameters.

**Sense Clock**

A clock source used to implement a switched-capacitor front-end for the CSD sensing method.

**Sensor**

See [Capacitive Sensor](#).

**Sensor Auto Reset**

A setting to prevent a sensor from reporting false touch status indefinitely due to system failure, or when a metal object is continuously present near the sensor.

When Sensor Auto Reset is enabled, the Baseline is always updated even if the Difference Count is greater than the Noise Threshold. This prevents the sensor from reporting the ON status for an indefinite period of time. When Sensor Auto Reset is disabled, the Baseline is updated only when the Difference Count is less than the Noise Threshold.

**Sensor Ganging**

See [Ganged Sensors](#).

**Shield Electrode**

Copper fill around sensors to prevent false touches due to the presence of water or other liquids. Shield Electrode is driven by the shield signal output from the CSD block. See [Driven-Shield](#).

**Shield Tank Capacitor ( $C_{SH}$ )**

An optional external capacitor ( $C_{SH}$  Tank Capacitor) used to enhance the drive capability of the CSD shield, when there is a large shield layer with high parasitic capacitance.

**Signal (CapSense Signal)**

Difference Count is also called Signal. See Difference Count.

**Signal-to-Noise Ratio (SNR)**

The ratio of the sensor signal, when touched, to the noise signal of an untouched sensor.

**Slider Resolution**

A parameter indicating the total number of finger positions to be resolved on a slider.

**Touchpad**

A Widget consisting of multiple sensors arranged in a specific horizontal and vertical fashion to detect the X and Y position of a touch.

**Trackpad**

See [Touchpad](#).

**Tuning**

The process of finding the optimum values for various hardware and software or threshold parameters required for CapSense operation.

**V<sub>REF</sub>**

Programmable reference voltage block available inside PSoC used for CapSense and ADC operation.

**Widget**

A user-interface element in the CapSense Component that consists of one sensor or a group of similar sensors. Button, proximity sensor, linear slider, radial slider, matrix buttons, and touchpad are the supported widgets.

# Revision History



## Document Revision History

Document Title: AN85951 – PSoC® 4 CapSense® Design Guide				
Document Number: 001-85951				
Revision	ECN#	Issue Date	Origin of Change	Description of Change
**	3973432	04/19/2013	NIDH	New Design Guide
*A	4059171	07/29/2013	NIDH	Added dual IDAC support. Updated some schematics in chapter 6. Other minor changes to chapters 3, 5, and 6.
*B	4189700	11/13/2013	NIDH	Added support of CY8C4000 devices. Minor fixes throughout the document.
*C	4289925	02/24/2014	NIDH	Updated the table of device features. Changed IDAC names to sync with new PSoC Creator Component terms. Added a schematic checklist. Changed screenshots to match the new Component version.
*D	4293476	02/27/2014	NIDH	Updated Table 1-1 per PSoC 4000 datasheet.
*E	4314223	03/20/2014	NIDH	Added firmware design considerations to Chapter 6. Added power supply layout and schematic considerations to Chapter 6. Updated the IMO range for PSoC 4000
*F	4339713	04/15/2014	NIDH	Updated to support PSoC 4000 and PSoC Creator 3.0 SP1.
*G	4494249	08/29/2014	DCHE	<p>Added Reference to Getting Started with CapSense in Section <a href="#">2.4.4 Proximity (Three-Dimensional)</a></p> <p>Renamed Section 2.5 to <a href="#">Liquid Tolerance</a> and re-wrote this section.</p> <p>Updated the recommendations for Shield drive i.e. Csh_tank precharge and Cmod precharge in Section <a href="#">3.1.1.1.7CapSense CSD Shielding</a></p> <p>Added recommendation for setting "API resolution" in Section</p> <p>Added guidelines on how to select value of "Sensitivity" parameter in Section</p> <p>Updated recommended values of threshold and hysteresis parameters in Section <a href="#">Manual Tuning Trade-offs</a>.</p> <p>Added Section <a href="#">Manual Tuning Slider Example</a></p> <p>Updated maximum overlay thickness value for sliders in <a href="#">Table 6-2</a></p> <p>Added guideline on maximum thickness for overlays of materials other than acrylic in Section <a href="#">6.3.2 Overlay Thickness</a></p> <p>Re-wrote Section <a href="#">Slider Design</a></p> <p>Added recommendations on DC loads in <a href="#">Section 6.3.5</a></p> <p>Renamed and rewrote section <a href="#">6.4.11</a> to <a href="#">Layout Guidelines for Liquid Tolerance</a></p> <p>Added Section <a href="#">6.4.12.1 External Capacitors Pin Selection</a></p> <p>Updated slider related recommendations in <a href="#">Table 6-8. Layout Rule Checklist</a></p> <p>Updated Section 6.5 Electromagnetic Compatibility (EMC) Considerations, added extensive data on hardware and firmware considerations.</p>
*H	4602375	12/19/2014	UDYG	<p>Added information for the PSoC 4 BLE family of devices.</p> <p>Added information for the PProC BLE family of devices.</p>

<b>Document Title: AN85951 – PSoC® 4 CapSense® Design Guide</b> <b>Document Number: 001-85951</b>				
Revision	ECN#	Issue Date	Origin of Change	Description of Change
				Updated ground and power layout guidelines in Section 6.4.9 and Section 6.4.10 .
*I	4624027	01/21/2015	NIDH	Added information for PSoC 4200-M family of devices Added footnote in section <a href="#">Slider Design</a> Added GPIO source/sink current limit in <a href="#">Table 6-6</a> . Changed document title to PSoC® 4 CapSense® Design Guide – AN85951
*J	4771699	06/02/2015	DCHE	Changed Document Title to “AN85951 – PSoC® 4 CapSense® Design Guide”. Updated <a href="#">Design Considerations</a> : Updated <a href="#">ESD Protection</a> : Updated <a href="#">Preventing ESD Discharge</a> : Updated <a href="#">Figure 6-35</a> . Updated <a href="#">Redirect</a> : Replaced "Guard Ring" with "Ground Ring". Updated <a href="#">Figure 6-36</a> .
*K	4891423	08/20/2015	DCHE	Added Table 3-1. Removed section 3.2.1 CMOD Precharge. Added section <a href="#">CapSense in PSoC 4xxxM/4xxxL-Series</a> . Updated section <a href="#">Trace Routing</a> . Added reference of AN2397. Added recommendation for modulator clock divider in section <a href="#">Manual Tuning Trade-offs</a> . Added <a href="#">Figure 6-34</a> .
*L	4905591	09/16/2015	DIMA	Updated Section <a href="#">3.2</a> . Updated <a href="#">Figure 3-16</a> . Updated <a href="#">Table 3-3</a> , <a href="#">Table 4-1</a> , <a href="#">Table 6-4</a> , <a href="#">Table 6-5</a> , <a href="#">Table 6-7</a> and <a href="#">Table 8-1</a> .
*M	5076590	01/19/2016	PRIA	Updated <a href="#">Introduction</a> Moved <a href="#">Signal-to-Noise Ratio</a> to Chapter 2 Updated Chapters <a href="#">PSoC 4</a> and <a href="#">PSoC BLE CapSense</a> and <a href="#">CapSense Performance Tuning</a> for details. Added section to Chapter 4 Added <a href="#">Glossary</a>
*N	5131335	02/23/2016	DCHE	Added information on mutual-capacitance sensing in PSoC 4 device series. Added information on CapSense 3.0 changes. Added following sections: <ul style="list-style-type: none"> <li>- <a href="#">Mutual-Capacitance Sensing</a></li> <li>- <a href="#">CapSense Architecture in PSoC 4 S-Series</a></li> </ul> Updated following sections: <ul style="list-style-type: none"> <li>- <a href="#">Introduction</a></li> <li>- <a href="#">CapSense Widgets</a></li> <li>- <a href="#">CapSense Design and Development Tools</a></li> <li>- <a href="#">CapSense Performance Tuning</a></li> </ul>