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

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

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

AN85951

PSoC® 4 and PSoC 6 MCU CapSense® Design Guide

Document Number: 001-85951 Rev. *Z

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com
www.infineon.com

Contents

1 Introduction

1.1 Abstract

The CapSense® Design Guide shows how to design capacitive touch sensing applications with the CapSense feature in PSoC® 4 and PSoC 6 MCU device families. The CapSense feature in these devices offer 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. This guide also introduces Cypress' new ModusToolbox® design tool for CapSense evaluation.

Cypress provides different device families with the CapSense feature. If you have not chosen a particular device, or are new to capacitive sensing, see the Getting Started with CapSense Design Guide. It helps you understand the advantages of CapSense over mechanical buttons, CapSense technology fundamentals, and to select 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 PSoC 6 MCU 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 and PSoC 6 MCU using the Cypress PSoC Creator™ integrated design environment (IDE). If you are new to PSoC 4, see AN79953 - Getting Started with PSoC 4 or AN92167 - Getting Started with PSoC 4 BLE. If you are new to PSoC 6 MCU, see AN221774 – Getting Started with PSoC 6 MCU and AN210781 - Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity. If you are new to PSoC Creator, see the PSoC Creator home page.

If you are new to ModusToolbox, see ModusToolbox® IDE Quick Start Guide.

This design guide helps you understand:

- CapSense technology in PSoC 4 and PSoC 6 MCU
- Design and development tools available for PSoC 4 and PSoC 6 MCU CapSense
- CapSense PCB layout guidelines for PSoC 4 and PSoC 6 MCU
- Performance tuning of PSoC 4 and PSoC 6 MCU CapSense Component
- Applications using CapSense Plus™ features such as Motor Control Systems and Induction Cookers

1.3 CapSense Features

CapSense in PSoC 4 and PSoC 6 MCU 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]
- Allows CapSense block re-configuration as an ADC, and supports ADC input on any GPIO pin[1]
- Provides superior SNR with programmable voltage reference (V_{REF})
- Supports spread spectrum and programmable resistance switches for lower electromagnetic interference (EMI)
- Provides reduced overhead on CPU during CapSense scanning by offloading initialization and configuration process to the CapSense sequencer

The PSoC 4100S Plus devices have the following additional features when compared to the PSoC 4100S devices:

- Offers larger flash memory and more I/Os
- Provides one Control Area Network (CAN) block
- Provides a true random number generator for secure key generation for cryptography applications
- Accepts additional external clock source of 4- to 33-MHz crystal oscillator (ECO)

1.4 PSoC 4 and PSoC 6 MCU CapSense Plus Features

You can create PSoC 4 CapSense Plus applications that feature capacitive touch sensing and additional system functionality. The key features of these devices, in addition to CapSense are:

- Arm® Cortex®-M0/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 (CM0+ 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.

[1] To achieve the best CapSense performance, follow the recommendations in Sensor Pin Selection section.

- Opamps with comparator mode and SAR input buffering capability
- Segment LCD direct drive that supports up to four commons
- SPI/UART/I2C serial communication channels
- 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 I2S)
- 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

The main features of PSoC 6 MCU device, in addition to CapSense are:

- Single CPU devices (Arm Cortex-M4), dual CPU devices (Arm Cortex-M4 and Cortex-M0+). Support for Inter-processor communication in hardware.
- 1.71 V - 3.6 V device operating voltage with user selectable core logic operation at either 1.1 V or 0.9 V
- Up to 2 MB of flash memory and up to 1 MB of SRAM
- Up to 78 GPIOs that can be used for analog, digital, CapSense, or segment LCD functions
- Programmable Analog Blocks: Two opamps, configurable PGAs, comparators, 12-bit 1 Msps SAR ADC, 12-bit voltage mode DAC
- Programmable Digital Blocks, Communication Interfaces
- 12 UDBs, 32 TCPWMs configurable as 16-bit/32-bit timer, counter, PWM, or quadrature decoder
- Up to 13 serial communication block (SCB) configurable as I2C, SPI, or UART interfaces. See the Device Datasheet for more details.
- Audio subsystem with one I2S interface and two PDM channels
- SMIF interface with support for execute-in-place from external quad SPI flash memory and on-the-fly encryption and decryption.
- Bluetooth Smart connectivity with BLE 5.0 (applicable only to PSoC 6 MCU with BLE family of devices)

See AN64846 - Getting Started with CapSense 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 provides links to the supporting documents for each of the numbered tasks in Figure 1-1.

Figure 1-1. CapSense Design Flow

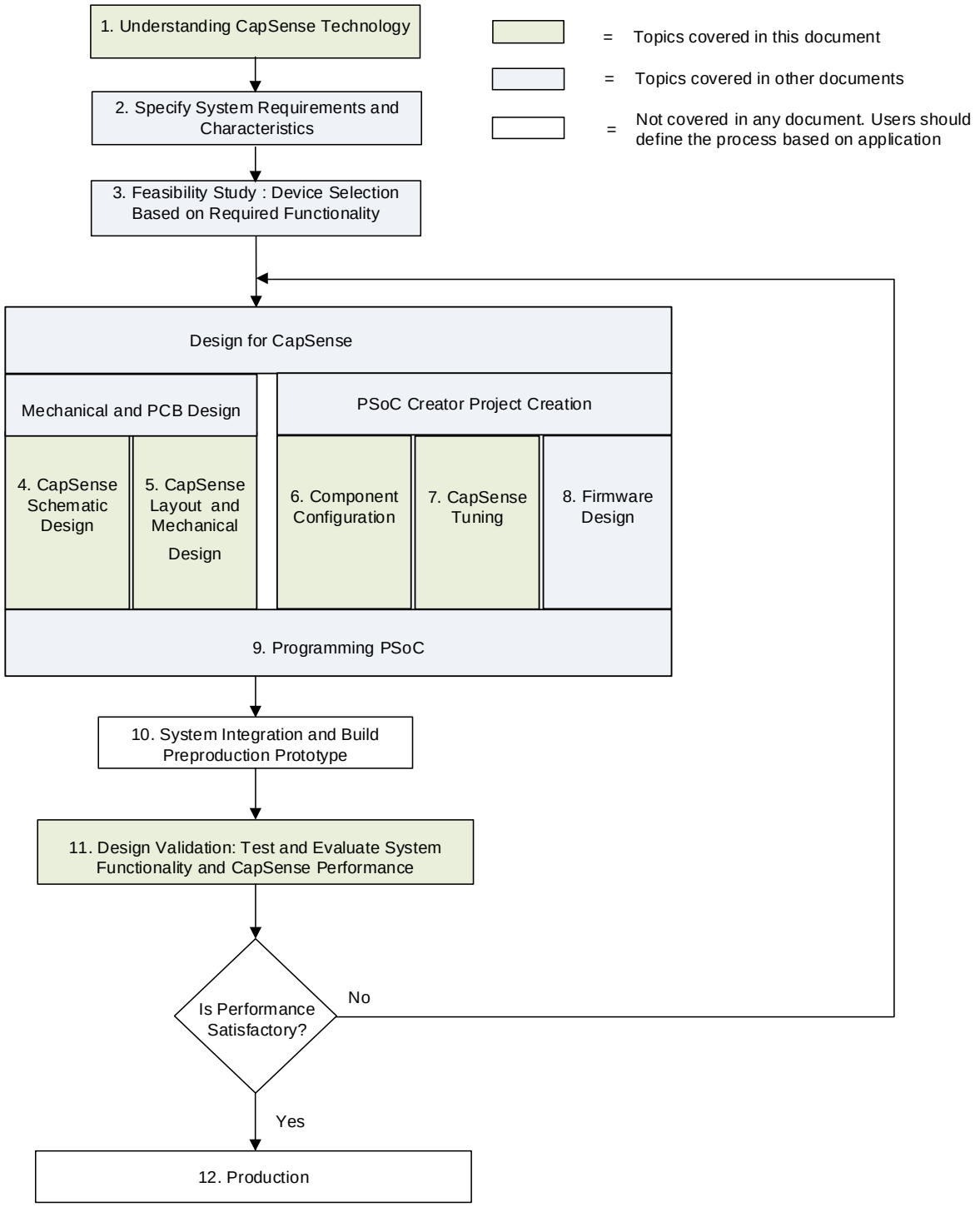


Table 1-1. Supporting Documentation

Steps in Flowchart	Supporting Cypress Documentation	
	Name	Chapter
1. Understanding CapSense	CapSense Design Guide (This document) Getting Started with CapSense	Chapter 2 and Chapter 3 –
2. Specify requirements	Getting Started with CapSense	–
3. Feasibility study	PSoC 4 Datasheet PSoC 4 BLE Datasheet PSoC 6 MCU Datasheet	–
	AN64846 – Getting Started with CapSense Design Guide AN79953 – Getting Started with PSoC 4 AN91267 – Getting Started with PSoC 4 BLE AN221774 – Getting Started with PSoC 6 MCU	–
4. Schematic design	CapSense Design Guide (This document)	Chapter 6
5. Layout design	CapSense Design Guide (This document)	Chapter 6
6. Component configuration	PSoC CapSense Component Datasheet / Middleware Document	–
	CapSense Design Guide (This document)	Chapter 5
7. Performance tuning	CapSense Design Guide (This document)	Chapter 5
8. Firmware design	PSoC Component Datasheet / Middleware Document	–
	PSoC Creator Example Projects	–
	Download ModusToolbox® here. See the ModusToolbox® related documents: ModusToolbox Release Notes ModusToolbox® User Guide ModusToolbox® Quick Start Guide ModusToolbox® CapSense® Configurator Guide ModusToolbox® CapSense® Tuner Guide PSoC® Creator™ to ModusToolbox® Porting Guide	
9. Programming PSoC	PSoC Creator User Guide for in-IDE programming PSoC Programmer home page and MiniProg3 User Guide for Standalone programming	–
10. Prototype	–	–
11. Design validation	CapSense Design Guide (This document)	Chapter 5
12. Production	–	–

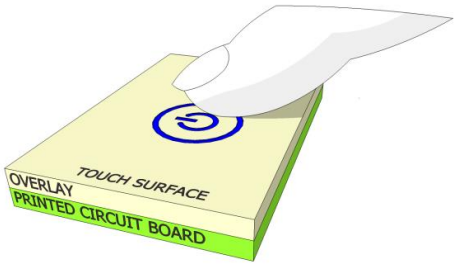
2 CapSense Technology

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.

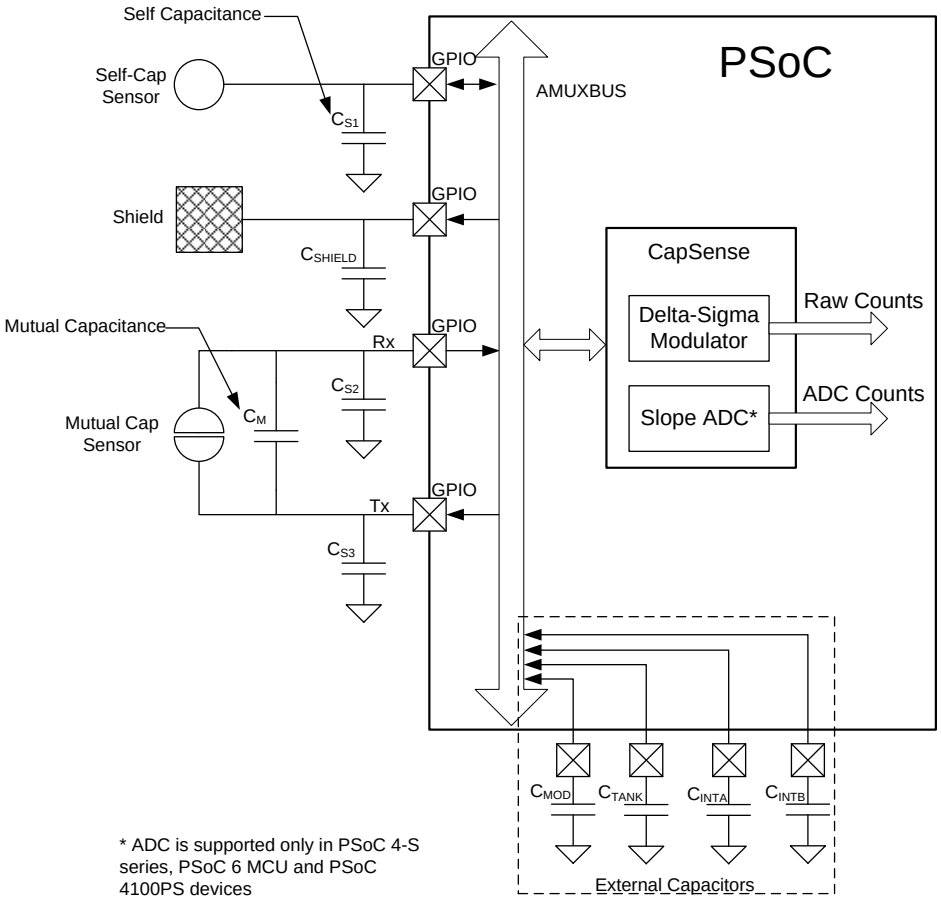
Figure 2-1. Capacitive Touch Sensor



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_{MX}. 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 7-9.

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

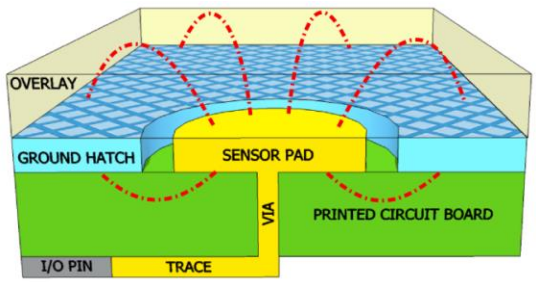


The capacitance of the sensor in the absence of a touch is called the parasitic capacitance, C_P. Parasitic capacitance results from the electric field between the sensor (including the sensor pad, traces, and vias) and other conductors in the system such as the ground planes, traces, and any metal in 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

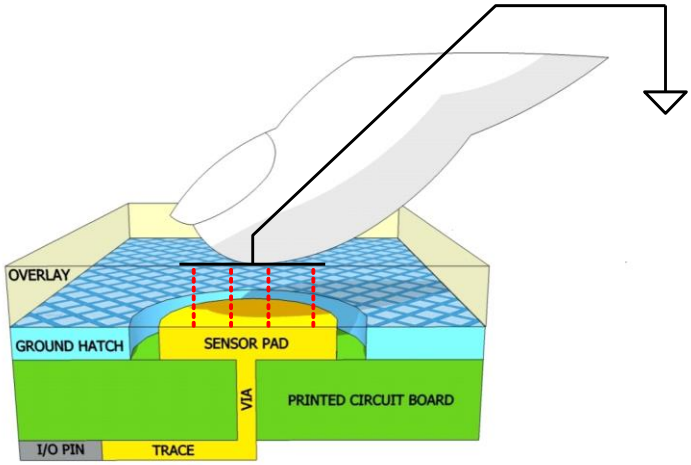
Figure 2-3 shows how a GPIO pin is connected to a sensor pad by traces and vias for self-capacitance sensing. Typically, a ground hatch surrounds the sensor pad to isolate it from other sensors and traces. Although Figure 2-3 shows some field lines around the sensor pad, the actual electric field distribution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

$$C_F = \frac{\varepsilon_0 \varepsilon_r A}{d}$$

Where:

ε_0 = Free space permittivity

ε_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. Refer to the CSD specification in Device Datasheet / Component Datasheet / Middleware Document to learn about the supported C_P range for a given device with which the recommended SNR can be achieved.

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[2] or VDDD[3] (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.

[3] VDDD is the device power supply for digital section.

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 $C^1_{M,}$ 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

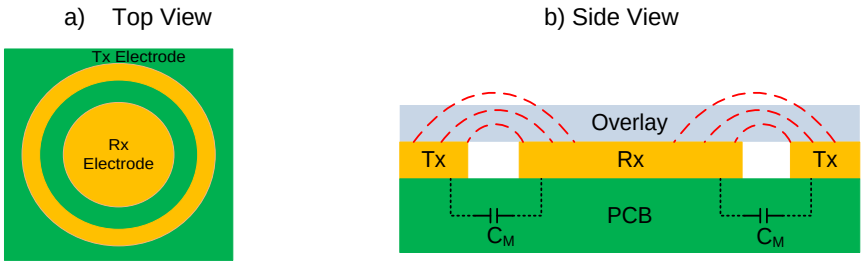
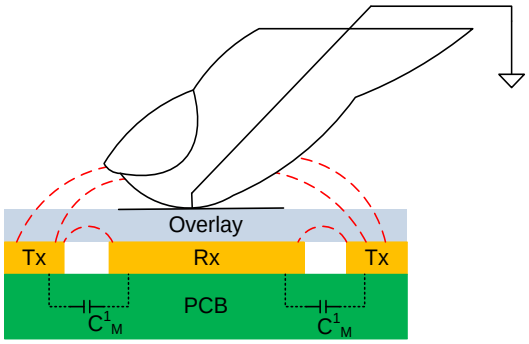


Figure 2-6. Mutual-Capacitance with Finger Touch



2.2 Capacitive Touch Sensing Method

PSoC uses Cypress 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 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

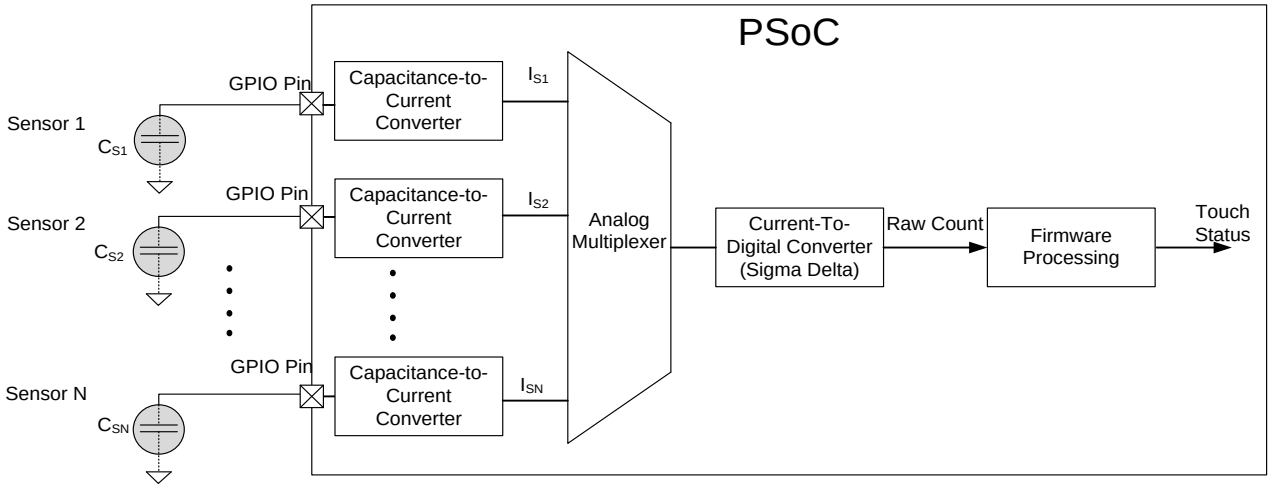
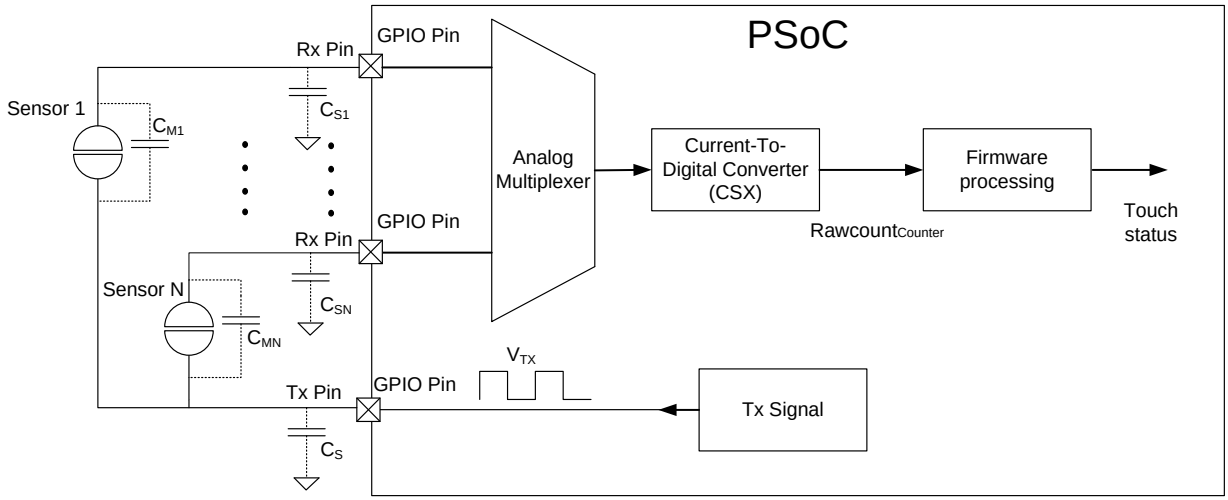


Figure 2-9 shows a plot of raw count over time. When a finger touches the sensor, the C_S increases from C_P to $C_P + C_F$, and the raw count increases. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

2.2.2 CapSense Crosspoint (CSX)

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

Figure 2-8. Simplified Diagram of 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 $\mathbf{Rawcount_{Counter}}$, is a digital value that is proportional to the mutual-capacitance between the Rx and Tx electrodes as shown by Equation 2-4.

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

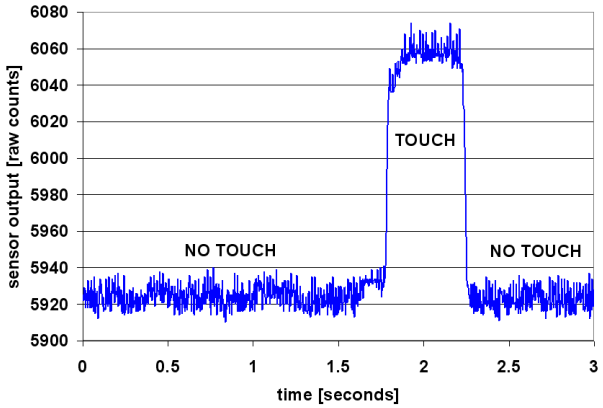
$$\mathrm{Rawcount_{Counter}} = \mathrm{G_{CM}\, C_M}$$

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^1_M (see Figure 2-6) 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). The normalized inverted raw count is computed using Equation 3-11.

Figure 2-9. Raw Count versus Time



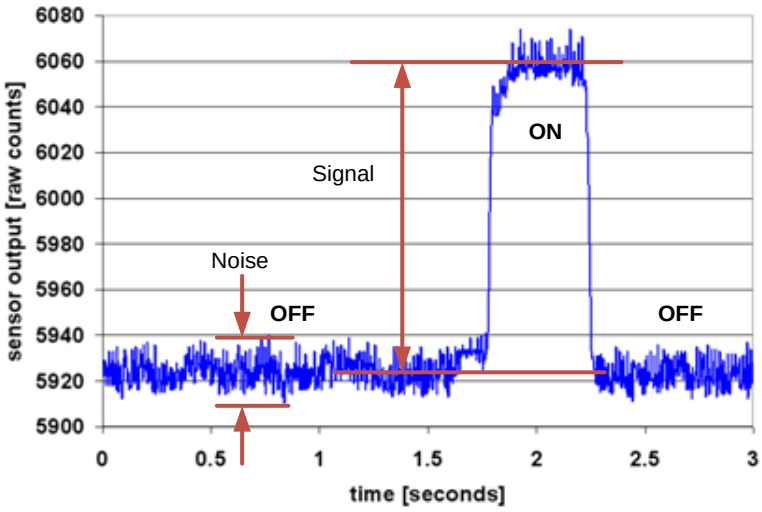
For an in-depth discussion of the PSoC 4 and PSoC 6 CapSense CSD and CSX blocks, see chapter PSoC 4 and PSoC 6 MCU CapSense.

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.

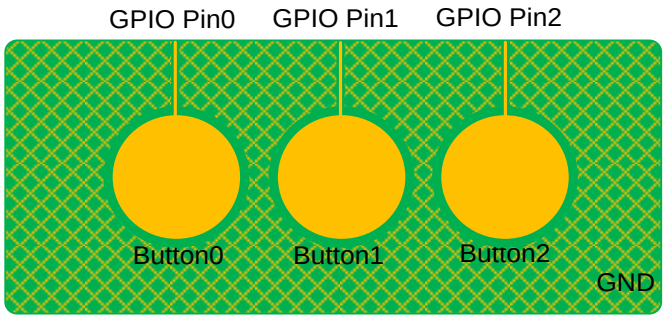
Figure 2-11. Several Types of Widgets

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.

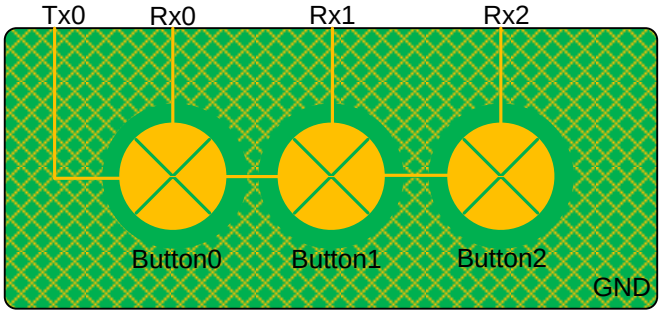
For the self-capacitance based i.e. CSD sensing method, a simple CapSense button consists of a circular copper pad connected to a PSoC GPIO with a PCB trace. The button is surrounded by grounded copper hatch to isolate it from other buttons and traces. A circular gap separates the button pad and the ground hatch. Each button requires one PSoC GPIO. These buttons can be constructed using any conductive material on a non-conductive substrate; for example indium Tin Oxide on a glass substrate, or silver ink on a non-conductive film. Even metallic springs can be used as button sensors; see Sensor Construction for more details.

Figure 2-12. Simple CapSense Buttons



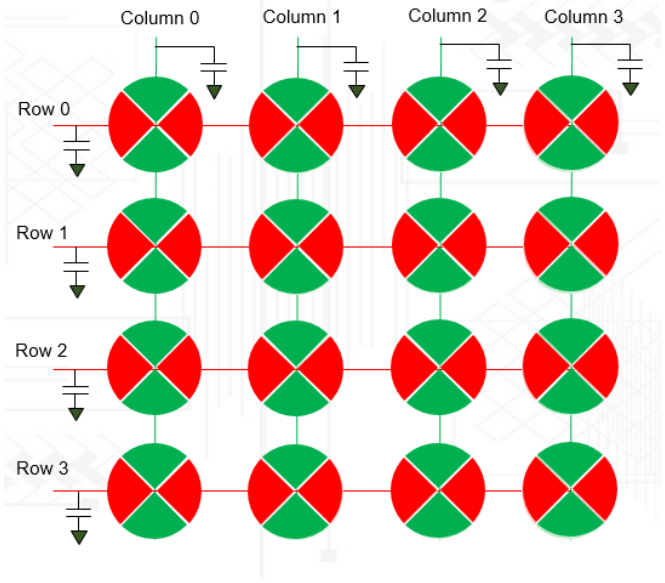
For the mutual-capacitance based i.e. CSX sensing method, each button requires one GPIO pin configured as Tx electrode and one GPIO pin configured as Rx electrode. The Tx pin can be shared across multiple buttons, as shown in Figure 2-13.

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



If the application requires many 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 eight GPIOs.

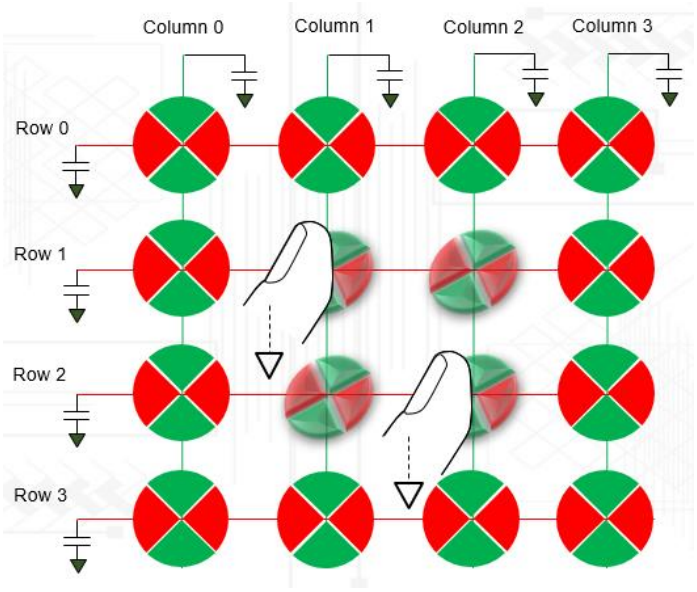
Figure 2-14. Matrix Buttons Based on CSD



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 (CSD) and mutual-capacitance (CSX) methods.

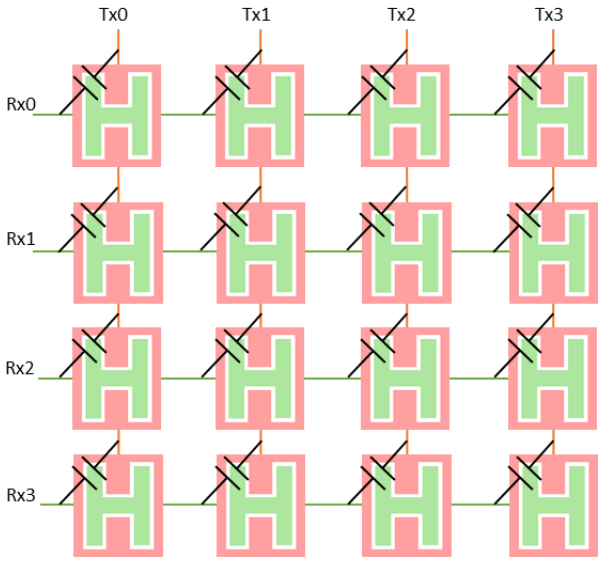
In self-capacitance mode, each button consists of a row sensor and a column sensor, as Figure 2-14 shows. When a button is touched, both row and column sensors of that button become active. The CSD-based matrix button should be used only if the user is expected to touch one button at a time. If the user touches more than one diagonally opposite buttons, the finger location cannot be resolved as Figure 2-15 shows. This effect is called as ghost effect, which is considered an invalid condition.

Figure 2-15. Ghost Effect in Matrix Button Based on CSD



Mutual capacitance is the recommended sensing method for matrix buttons because it doesn't suffer from ghost touch and provides better SNR for high Cp sensors. This is because it senses mutual capacitance formed at each intersection rather than sensing rows and columns as shown in Figure 2-16. Applications that require simultaneous sensing of multiple buttons, such as a keyboard with Shift, Ctrl, and Alt keys can use mutual-capacitance sensing method or you should design the Shift, Ctrl, and Alt keys as individual CSD buttons.

Figure 2-16. Matrix Button Based on CSX



Note however that scanning a matrix keypad using CSX sensing method may require a longer overall scan time than the CSD sensing method. This is because the CSD sensing method scans rows and columns as sensors, while the CSX sensing method scans each intersection as a sensor.

2.4.2 Sliders (One-Dimensional)

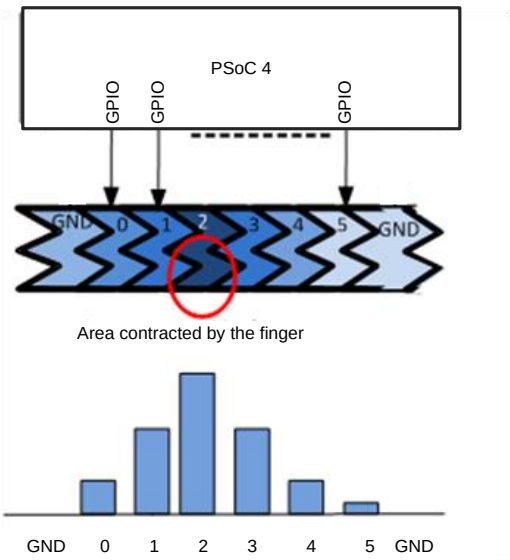
Sliders are used when the required input is in the form of a gradual increment or decrement. Examples include lighting control (dimmer), volume control, graphic equalizer, and speed control. Currently, the CapSense Component in PSoC Creator and ModusToolbox supports only self-capacitance-based sliders.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

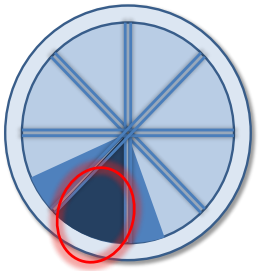
In a linear slider, the segments are arranged inline, as Figure 2-17 shows. Each slider segment connects to a PSoC GPIO. A zigzag pattern (double chevron) is recommended for slider segments. This layout ensures that when a segment is touched, the adjacent segments are also partially touched, which aids estimation of the centroid position.

Figure 2-17. Linear Slider



Radial sliders are similar to linear sliders except that radial sliders are continuous. Figure 2-18 shows a typical radial slider.

Figure 2-18. Radial Slider



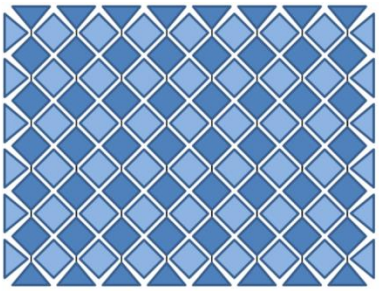
2.4.3 Touchpads / Trackpads (Two-Dimensional)

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 2-19 shows a typical arrangement of a trackpad sensor. Like matrix buttons, trackpads can also be sensed using either CSD or CSX sensing method.

CSD-based touchpads suffer from ghost touches, so it supports only single-point touch applications.

CSX touchpads can support multi-point touch applications, but these may need more scanning time compared to CSD touchpad because this method scans each intersection rather than rows and columns.

Figure 2-19. Trackpad Sensor Arrangement

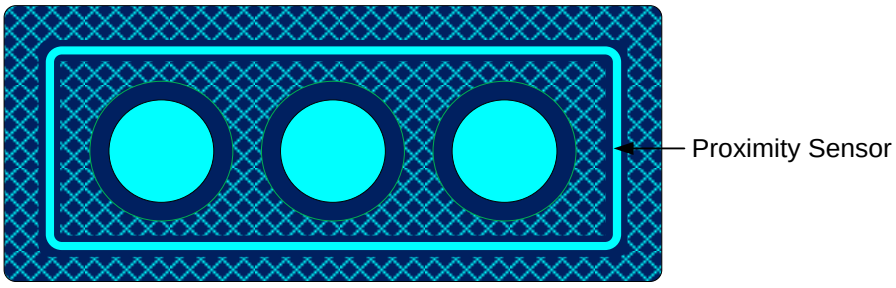


2.4.4 Proximity (Three-Dimensional)

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. Self capacitance is the recommended method of sensing for a proximity application.

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. Figure 2-20 shows a proximity sensor using a trace with a thickness of 2-3 mm surrounding the other sensors.

Figure 2-20. 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 Component Datasheet / Middleware Document for details on maximum parasitic capacitance supported by a given device.

See 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

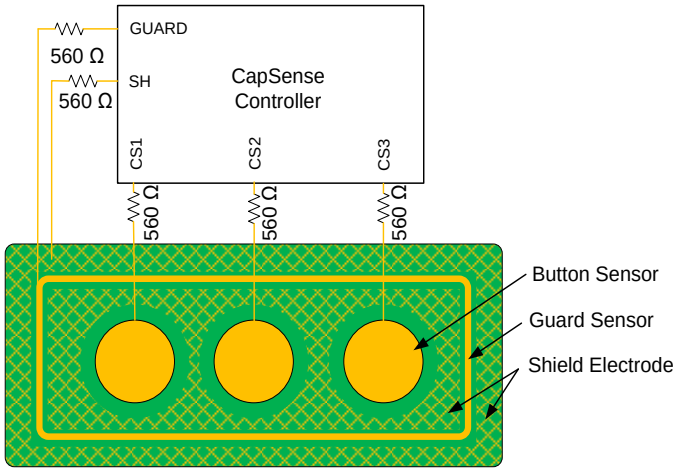
Capacitive sensing is used in a variety of applications such as home appliances, automotive, and industrial applications. These applications require robust capacitive-sensing operation even in the presence of mist, moisture, water, ice, humidity, or other liquids. In a capacitive-sensing application 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 2-21 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 prevent sensor false triggers due to a liquid flow, you should implement a Guard Sensor as Figure 2-22 shows. This Guard Sensor can be used to detect the presence of a streaming liquid and ignore the status or stop the sensing from rest of the sensors as long as the liquid flow is present.
- Note that the Guard sensor itself is just another self-capacitance sensor; even though you could implement it around mutual-capacitance sensors also for liquid flow tolerance. PSoC devices allow implementation of such self-capacitance sensors and mutual-capacitance sensors together in the same design.
- To compensate for changes in raw count due to liquid droplets for self-capacitance sensing, you can implement a Shield Electrode as Figure 2-22 shows. When a shield electrode is implemented, CapSense reliably works and reports the sensor ON/OFF status correctly, even when liquid droplets are present on the sensor surface. To prevent sensor false triggers due to liquid droplets for mutual-capacitance sensing, you can use both the sensing methods i.e., mutual capacitance and self-capacitance with Shield Electrode on the same set of sensors as Using Self-Capacitance Sensing for Liquid Tolerance of Mutual Capacitance Sensors explains.

In summary, 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 as shown in Figure 2-22. Follow the schematic and layout guidelines explained in the Layout Guidelines for Liquid Tolerance section to construct the shield electrode and guard sensor respectively.

Figure 2-22. Shield Electrode (SH) and Guard Sensor (GUARD) Connected to CapSense Controller

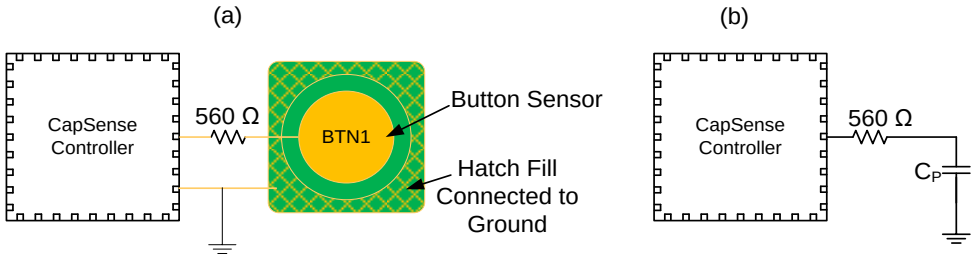


2.5.1 Liquid Tolerance for Self-Capacitance Sensing

2.5.1.1 Effect of Liquid Droplets and Liquid Stream on a Self-Capacitance Sensor

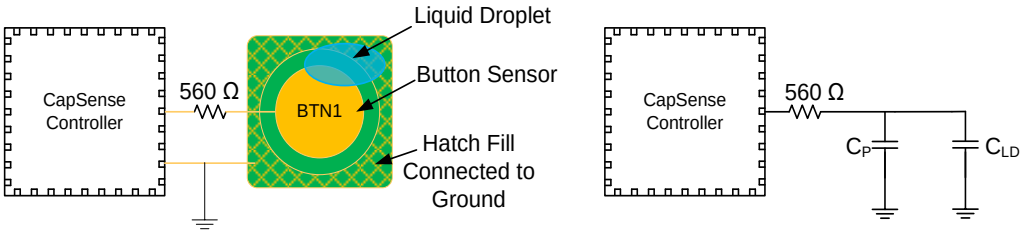
To understand the effect of liquids on a CapSense sensor, consider a CapSense system in which the hatch fill around the sensor is connected to ground, as Figure 2-23(a) shows. The hatch fill when connected to a ground improves the noise immunity of the sensor. Parasitic capacitance of the sensor is denoted as C_P in Figure 2-23 (b).

Figure 2-23. Typical CapSense System Layout



As shown in Figure 2-24, 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 2-25 shows. In such a situation, sensor false triggers might occur.

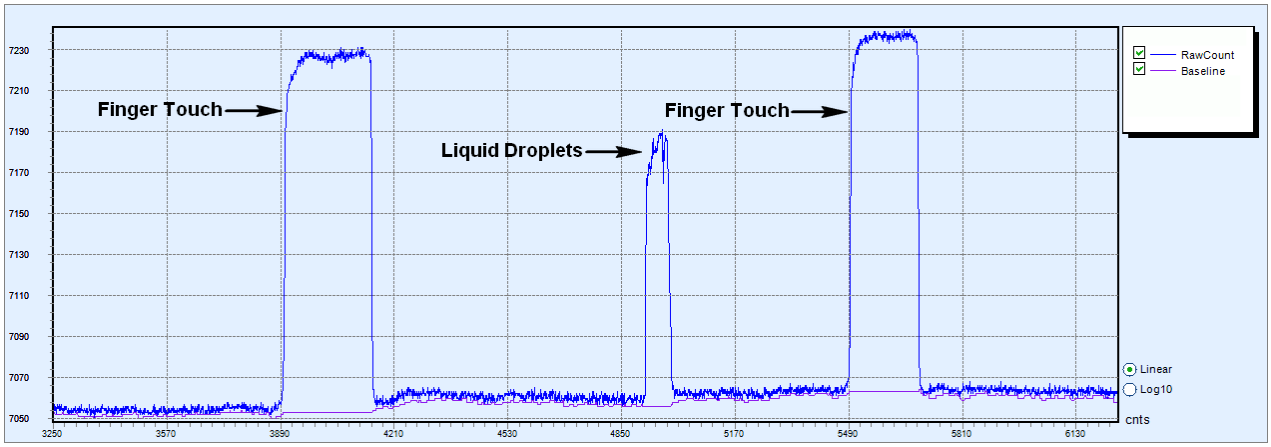
Figure 2-24. Capacitance Added by Liquid Droplet when the Hatch Fill is Connected to Ground



C_P – Sensor parasitic capacitance

C_{LD} – Capacitance added by the liquid droplet

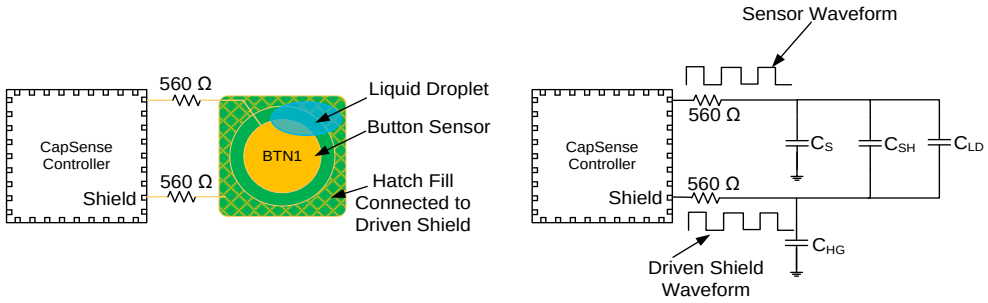
Figure 2-25. Effect of Liquid Droplet when the Hatch Fill around the Sensor is Connected to Ground



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 2-26 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 2-27 shows.

Figure 2-26. 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 2-27. Effect of Liquid Droplet when the Hatch Fill around the Sensor is Connected to the Driven-Shield

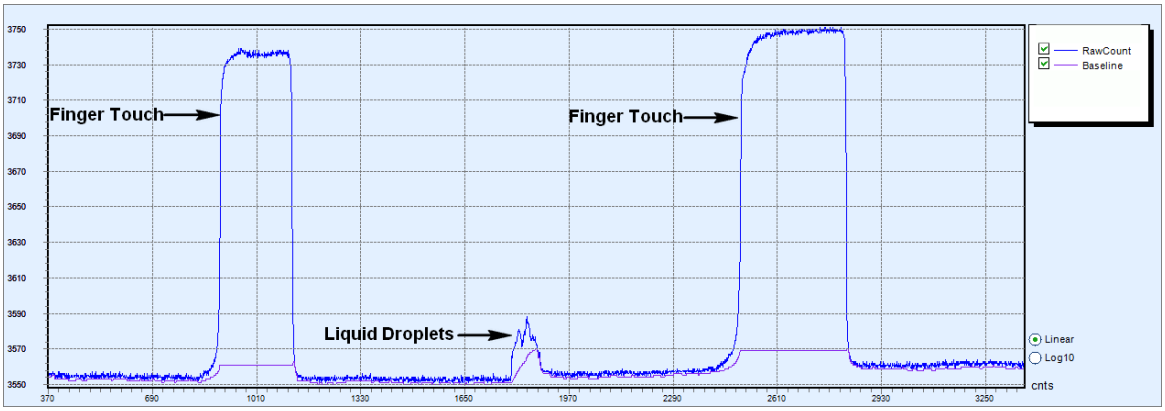
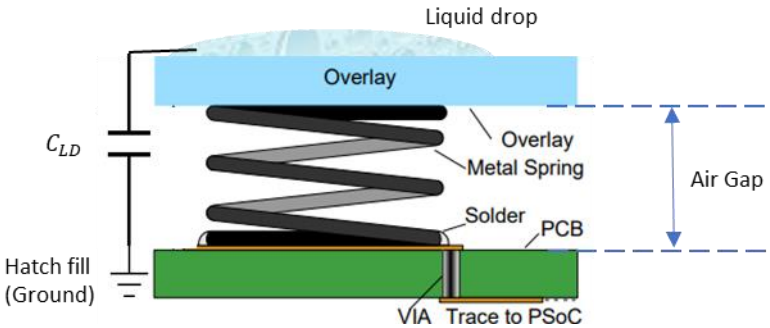


Figure 2-25 shows how a sensor may false trigger in presence of a liquid, if hatch fill is connected to ground. Note however, that the same is not true for all cases. For example, spring sensors, which are inherently more liquid tolerant than sensors etched on PCB surface. As Figure 2-28 shows, due to the large airgap between the liquid drop and the hatch fill, the capacitance C_{LD} between the liquid drop and grounded hatch pattern on the PCB would be very low so as not to cause any false triggers. If required, the hatched pattern on the PCB can still be connected to a driven shield electrode to further nullify the effect of C_{LD} and have an improved liquid tolerance.

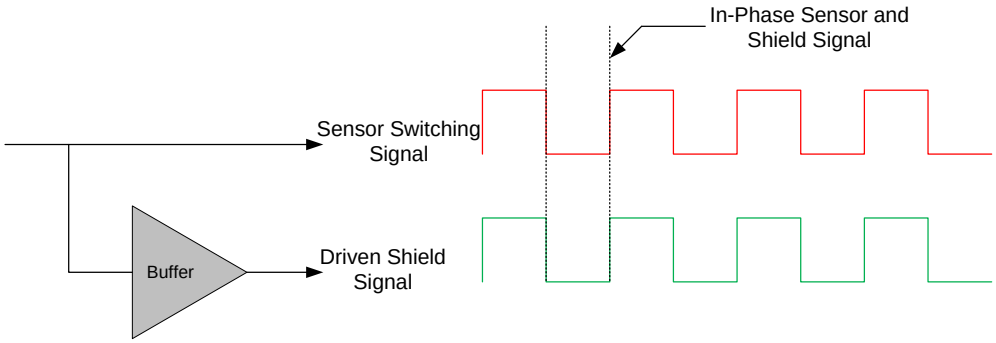
Figure 2-28. Capacitance Added by Liquid Droplet in Spring Sensor



2.5.1.2 Driven-Shield Signal and Shield Electrode

The driven-shield signal is a buffered version of the sensor-switching signal, as Figure 2-29 shows. The driven-shield signal has the same amplitude, frequency, and phase as that of sensor switching signal. When the hatch fill around the sensor is connected to the driven shield signal, it is referred as shield electrode.

Figure 2-29. Driven Shield Signal



Shield electrode can be used for following purposes:

- *To implement liquid-tolerant CapSense designs:* Shield electrode helps in making CapSense designs liquid-tolerant as explained above.
- *To improve proximity sensing distance in 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 Shield Electrode for layout guidelines of shield electrode.

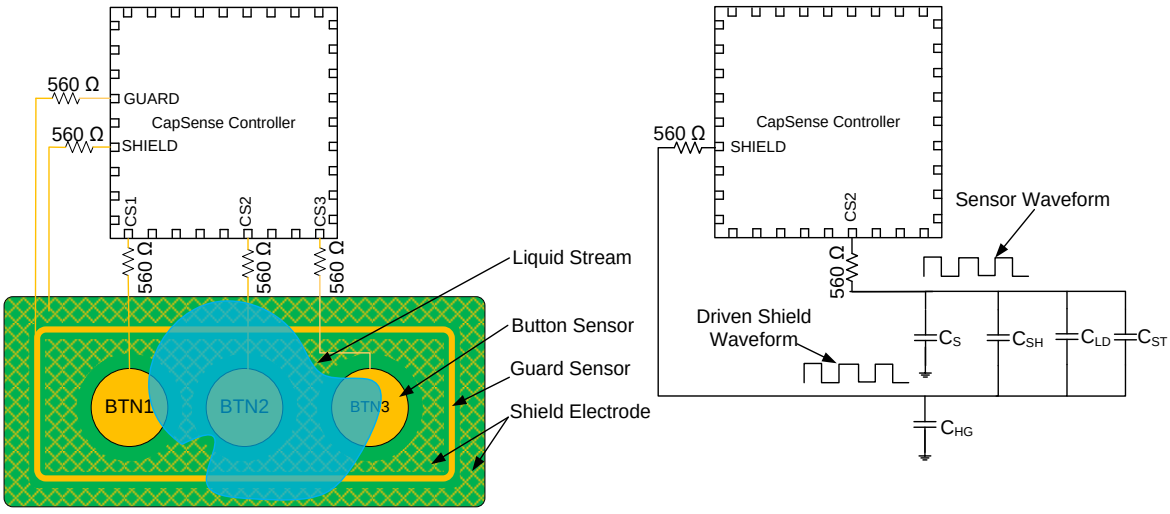
2.5.1.3 Guard Sensor

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 2-30 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 should disable the scanning of all other sensors except the guard sensor to prevent sensor false triggers.

Note: The sensors are not scanned, or the sensor status is ignored when the guard sensor is triggered; therefore, touch cannot be detected when there is a liquid stream on the touch surface.

Figure 2-30. Capacitance Measurement with a Liquid Stream



See Layout guidelines for Guard Sensor for PCB layout guidelines for implementing a guard sensor.

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 depending on the use case, such as follows:

- 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 ignore 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 ignore the slider centroid value that is calculated.
- Likewise, you could create your own custom algorithm to detect the presence of streaming liquids and ignore the sensor status during the time a liquid is present on the touch surface.

Note: The sensors are not scanned, or the sensor status is ignored when the guard sensor is triggered; therefore, touch cannot be detected when there is a liquid stream on the touch surface.

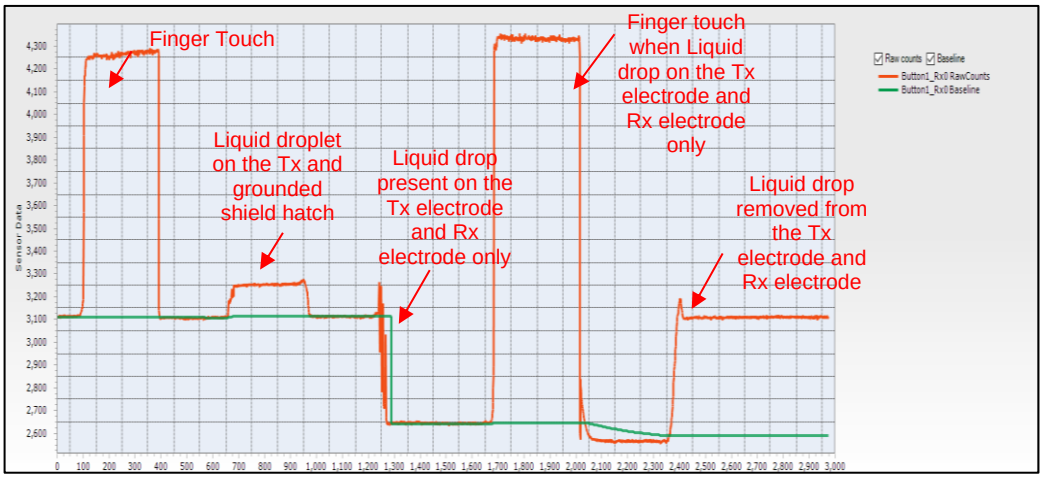
2.5.2 Liquid Tolerance for Mutual-Capacitance Sensing

2.5.2.1 Effect of Liquid Droplets and Liquid Stream on a Mutual-Capacitance Sensor

Mutual capacitance buttons often have a grounded hatch fill around the sensors for improved noise immunity. If a liquid droplet falls over the sensor while covering some part of the grounded hatch, the mutual capacitance decreases similar to the effect of placing a finger on the sensor. This decrease in mutual capacitance causes an increase in raw count as explained in CapSense CSX Sensing Method and as shown in the Figure 2-31. The amount of increase in the raw count depends on the size and characteristics of the liquid drop.

However, mutual capacitance increases if the liquid droplet covers just the Tx and Rx electrode and does not spread over the grounded hatch. This causes a decrease in raw count as shown in Figure 2-31. This decrease in raw count may cause the baseline reset due to Low Baseline Reset. Once the liquid drop is removed, the raw count would rise while the baseline may remain at the lower value, resulting in a difference signal which may cause the sensor to false trigger.

Figure 2-31. Effect of Liquid Droplet on CSX Sensor when the Hatch Fill Around the Sensor Is Connected to Ground



2.5.2.2 Using Self-Capacitance Sensing for Liquid Tolerance of Mutual Capacitance Sensors

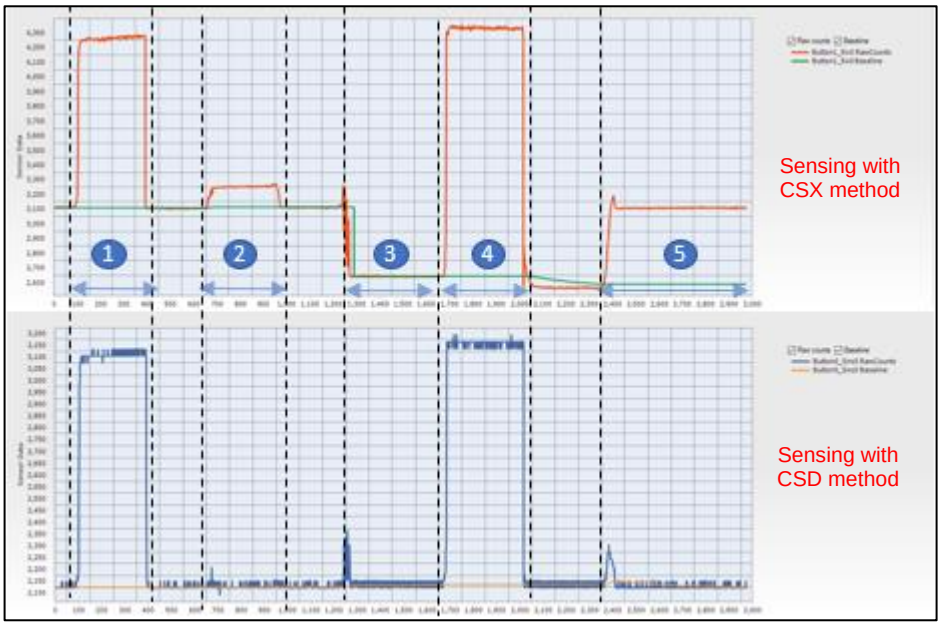
CapSense senses the self-capacitance of Tx and Rx nodes of a mutual-capacitance sensor. This ability of scanning the sensor using both CSD and CSX modes could be used to avoid false triggers due to the presence of liquid drops on a mutual capacitance sensor. See the code example PSoC 4 Hybrid Sensing Using CapSense to understand how to sense a mutual capacitance button with both CSD as well as CSX sensing method.

To achieve liquid tolerance, you need to scan the Rx electrode of the sensor with the CSD sense method. While scanning the Rx electrode as a CSD sensor, ensure that you enable the shield electrode, and connect the Tx pin of the mutual capacitance sensor to the driven shield signal. You can use the low-level API function `CapSense_SetPinState()` to connect the Tx pin of the mutual capacitance sensor to the shield electrode before calling the `CapSense_ScanAllWidgets()` API function that scans the Rx electrode as a CSD sensor as shown below:

```
CapSense_SetPinState(CapSense_BUTTON1_WDGT_ID,CapSense_BUTTON1_TX0_ID,CapSense_SHIELD);
CapSense_ScanAllWidgets();
```

From sections 2.5.1 and 2.5.2 you understood the effect of liquid drop on the CSD and CSX button respectively. By utilizing the difference in their response to the liquid drop, you can create a firmware logic to achieve a liquid-tolerant mutual capacitance sensor. The effect of presence of the liquid drop on the CSD and CSX scan results is summarized in Figure 2-32.

Figure 2-32. Effect of Water Drop on the CSX Sensor Pattern Scanned with CSD and CSX Methods



Where Figure 2-32 shows the effect of the water drop on the CSX sensor pattern surrounded by hatch fill when scanned using this method. The regions in Figure 2-32 represents the following:

1. Finger touch
2. Liquid droplet on the Tx line and grounded shield hatch
3. Liquid drop present on the Tx and Rx electrodes only
4. Finger touch when a liquid drop is on the Tx and Rx electrodes only
5. Liquid drop removed from the Tx and Rx electrodes

The changes in raw count as shown in Figure 2-32 could be used in the firmware to reset the baseline of the CSX sensor to nullify the effect of liquid drops. The button status should be ON state for Region 1, 4, and OFF state in other regions; additionally, the baseline of the CSX button must be re-initialized in Region 3 and Region 5. The baseline of the sensor could be reset by using the `CapSense_InitializeWidgetBaseline()` API function as shown below:

```
CapSense_InitializeWidgetBaseline(CapSense_CSX_BUTTON_WDGT_ID);
```

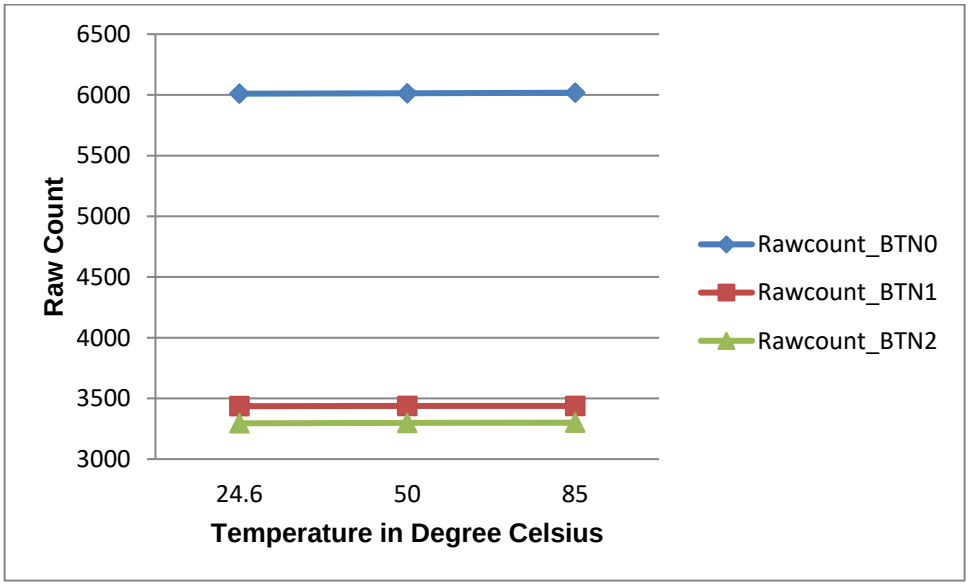
See the Component Datasheet / Middleware Document for more details on using this API; see Selecting CapSense Software ParametersBaseline to learn about the baseline of the sensor.

2.5.3 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 2-33 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 Update Algorithm, thereby preventing any false triggering of the sensors.

Figure 2-33. Raw Count Variation versus Water Temperature



3 PSoC 4 and PSoC 6 MCU CapSense

This chapter explains how CapSense CSD and CSX is implemented in the PSoC 4 and PSoC 6 MCU. See Capacitive Touch Sensing Method to understand the basic principles of CapSense. A basic knowledge of the PSoC device architecture is a prerequisite for this chapter. If you are new to PSoC 4, see AN79953 - Getting Started with PSoC 4 or AN91267 - Getting Started with PSoC 4 BLE; for PSoC 6 MCU, see AN221774 - Getting Started with PSoC 6 MCU.

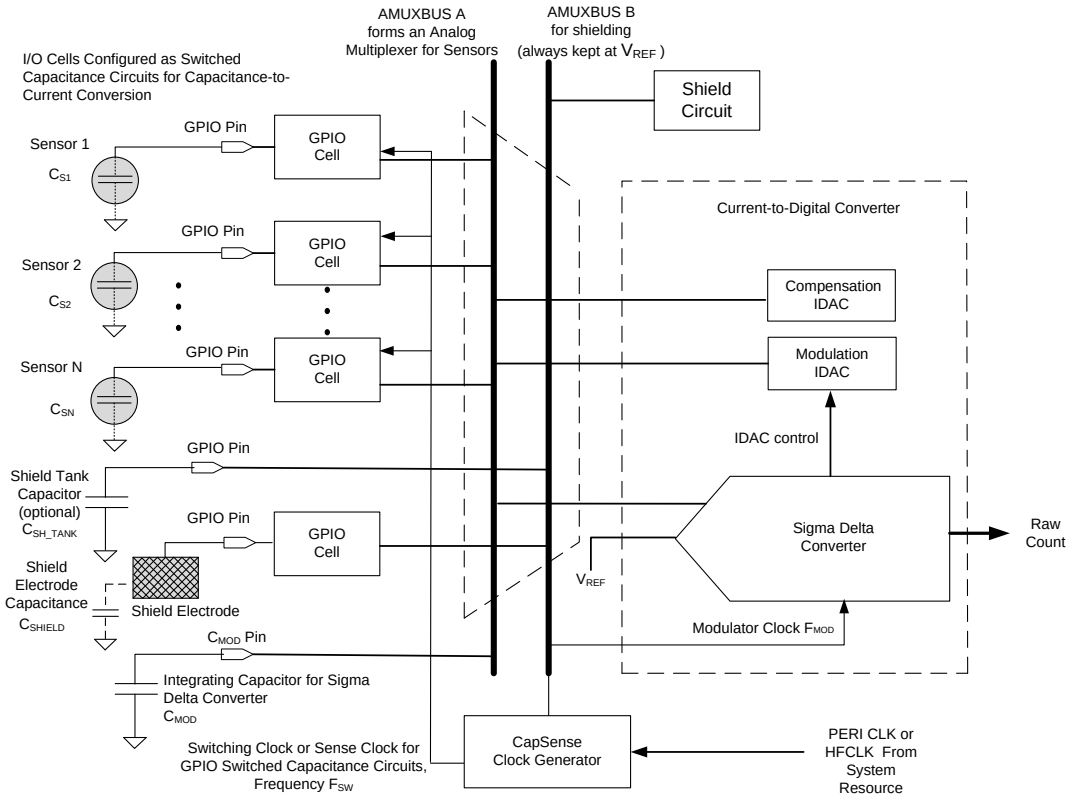
You can skip this chapter if you are using the automatic tuning feature (SmartSense) of the Component. See the CapSense Performance Tuning chapter for details.

The PSoC 4 family of devices has two different CapSense architectures. Section 3.1 explains the third-generation CapSense architecture, which is present in PSoC 4000, PSoC 4200, PSoC 4200 BLE, PSoC 4200M, and PSoC 4200L devices, and Section 3.3 explains the differences between the third-generation and the fourth-generation CapSense architecture. The fourth generation CapSense is present in PSoC 4000S, PSoC 4100S, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices.

3.1 CapSense CSD Sensing Method

Figure 3-1 illustrates the CapSense block that scans CapSense sensors in CSD sensing mode.

Figure 3-1. CapSense CSD Sensing

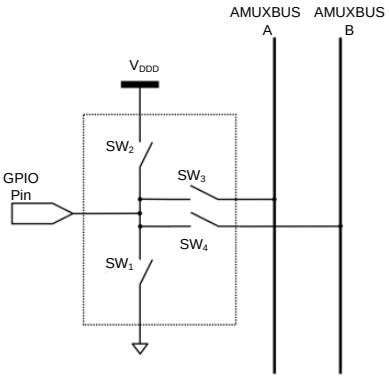


As explained in Capacitive Touch Sensing Method, this block works by first converting the sensor capacitance into an equivalent current. An analog multiplexer then selects one of the currents and feeds it into the current-to-digital converter. This current-to-digital converter consists of a sigma-delta converter, which controls the modulation IDAC for a specific period, the total current sourced or sinked by the IDACs is the same as the total current sinked 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.

3.1.1 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 3-2 shows a simplified diagram of the GPIO cell structure.

Figure 3-2. GPIO Cell Structure

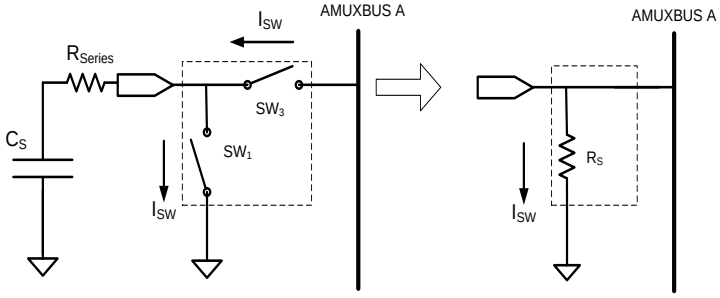


PSoC 4 and PSoC 6 devices have two analog multiplexer buses: AMUXBUS A is used for CSD sensing and AMUXBUS B is used for CapSense CSD Shielding. The GPIO switched-capacitance circuit has two possible configurations: source current to AMUXBUS A or sink current from AMUXBUS A.

3.1.2 IDAC Sourcing Mode

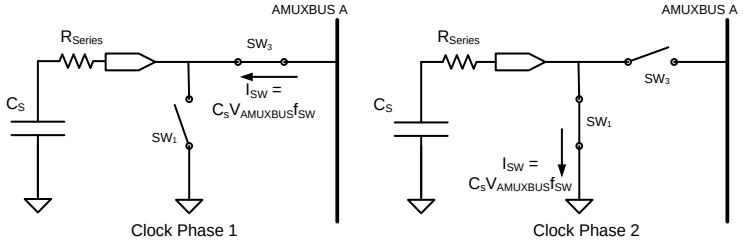
In the IDAC Sourcing mode, the GPIO cell sinks current from the AMUXBUS A through a switched capacitor circuit as Figure 3-3. shows.

Figure 3-3. 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 3-4. shows. The continuous switching of SW_1 and SW_3 forms an equivalent resistance R_S, as Figure 3-3. shows.

Figure 3-4. SW_1 and SW_3 Switch in 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 3-4. shows, the value of the equivalent resistance R_S is given by Equation 3-1.

Equation 3-1. Sensor Equivalent Resistance

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

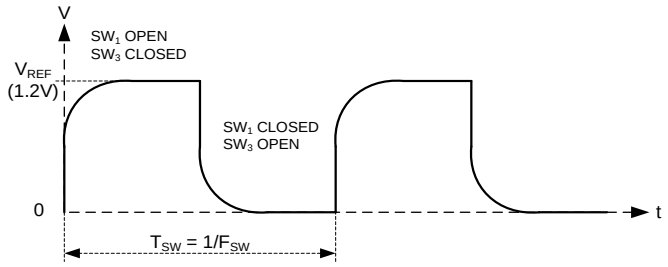
Where:

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 3-5. shows the resulting voltage waveform across C_S.

Figure 3-5. Voltage across Sensor Capacitance



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

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

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

3.1.3 IDAC Sinking Mode

In the IDAC Sinking mode, the GPIO cell sources current to the AMUXBUS A through a switched capacitor circuit as Figure 3-6. shows. Figure 3-7. shows the voltage waveform across the sensor capacitance.

Because this mode charges the AMUXBUS A directly through VDDD, it is more susceptible to power supply noise compared to the IDAC Sourcing mode. Hence, it is recommended to use this mode with an LDO or a very stable and quiet VDDD.

Figure 3-6. GPIO Cell Sourcing Current to AMUXBUS A

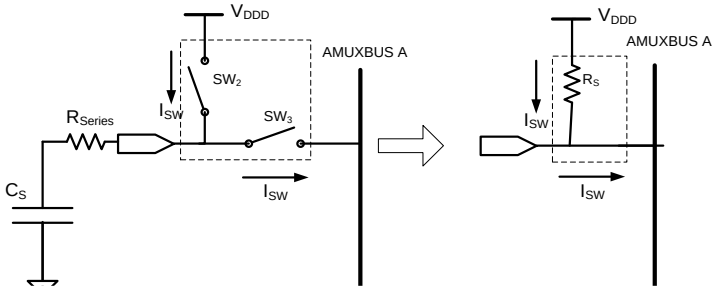
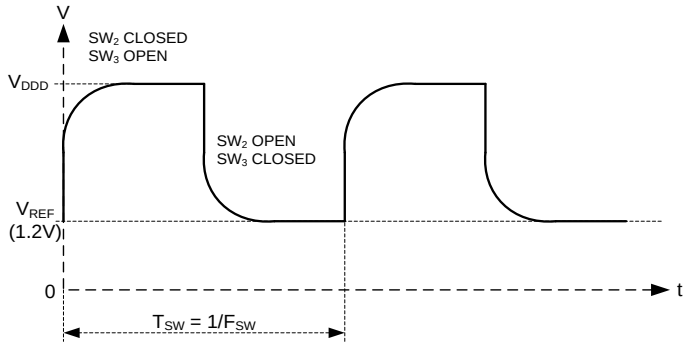


Figure 3-7. Voltage across Sensor Capacitance



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})

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

3.1.4 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) or peripheral clock (PERI) depending on the PSoC device family as shown in Figure 3-1.

3.1.4.1 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.

Sense clock can be sourced from three options: direct, 8-bit PRS, and 12-bit PRS. Some PSoC 4 and PSoC 6 MCU parts also support additional Spread Spectrum Clock (SSCx) modes. For more details on the supported modes for PSoC device, see the Component Datasheet / Middleware Document.

Direct clock is a constant frequency sense clock source. When you chose this option, the sensor pin switches with a constant frequency clock with frequency as specified in the CapSense Component configuration window.

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 by dividing the input clock using a PRS.

SSCx also spreads the sense clock frequency. It provides better noise immunity and reduces radiated electromagnetic emissions.

See Manually Tuning Hardware Parameters for details on the clock source and frequency selection guidelines.

3.1.4.2 Modulator Clock

The modulation clock is used by the Sigma Delta Converter. This clock determines the sensor scan time based on the following equations:

Equation 3-4. Sensor Scan Time

$$Sensor\ \text{scan time} = \text{Hardware scan time} + \text{Sensor Initialization time}$$

Equation 3-5. Hardware Scan Time

$$Hardware\ \text{scan time} = {\left(2^{\text{Resolution}} - 1\right)}/{\text{Modulator Clock Frequency}}$$

Here, “Resolution” is the scan resolution and Sensor Initialization time is the time taken by the sensor to write to the internal registers and initiate a scan.

3.1.5 Sigma Delta Converter

The Sigma Delta Converter converts the input current to a corresponding digital count. It consists of a sigma-delta converter and two current sourcing/sinking digital-to-analog converters (IDACs) called Modulation IDAC and Compensation IDAC as Figure 3-1 shows.

The sigma-delta converter uses an external integrating capacitor, called modulator capacitor C_{MOD}, as Figure 3-1 shows. Sigma-delta converter controls the modulation IDAC current by switching it ON or OFF corresponding to the small voltage variations across C_{MOD} to maintain the C_{MOD} voltage at V_{REF}. The recommended value of C_{MOD} is listed in Table 7-9.

The sigma-delta converter can operate in either IDAC sourcing mode or IDAC sinking mode.

- IDAC Sourcing Mode: In this mode, the GPIO Cell Capacitance to Current Converter sinks current from CMOD through AMUXBUS A, and the IDACs then source current to AMUXBUS A to balance its voltage.
- IDAC Sinking Mode : In this mode, the GPIO Cell Capacitance to Current Converter sources current from CMOD to AMUXBUS A and the IDACs sink current through AMUXBUS A to balance its voltage.

In both the above-mentioned modes, the sigma delta converter can operate in either single IDAC mode or dual IDAC mode:

- In the single IDAC mode, the modulation IDAC is controlled by the sigma-delta converter; the compensation IDAC is always OFF.
- In the dual IDAC mode, the modulation IDAC is controlled by the sigma-delta converter; the compensation IDAC is always ON.

In the single IDAC mode, 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 3-6.

Equation 3-6. 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 3-7. 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 software 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 3-8. 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 3-9.

Equation 3-9. 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 3-8 does not hold true if $I_{COMP} > V_{REF}\, C_S\, F_{SW}$ and Equation 3-9 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. Relationship Between CapSense Raw Count and CapSense Hardware Parameters

Sl. No.	Parameter	Description	Comments
1	N	Scan Resolution	Scan resolution is configurable from 6-bit to 16-bit. See Component Datasheet / Middleware Document for details.
2	V_{REF}	N/A	The V_{REF} value is 1.2 V or configurable between 0.6 V to V_{DDA}-0.6 V depending on the PSoC device family. See Component Datasheet / Middleware Document for details.
3	F_{SW}	Sense Clock Frequency	Sense clock frequency and sense clock source decide the frequency at which the sensor is switching. See Sense Clock for details.
		Sense Clock Source	
4	I_{MOD}	Modulator IDAC	I_{MOD} = Modulation IDAC current
5	I_{COMP}	Compensation IDAC	I_{COMP} = Compensation IDAC current
6	V_{DD}	N/A	This parameter is the device supply voltage.
7	C_S	N/A	This parameter is the sensor parasitic capacitance.
8	N/A	Modulator Clock Frequency	Modulator clock divider does not impact raw count. See the Modulator Clock section for more details

3.1.6 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 shows. The AMUXBUS A and the GPIO cell switches (see SW_3 in Figure 3-6.) form this analog multiplexer. AMUXBUS A connects to all GPIOs that support CapSense. See your corresponding device Device Datasheet for a list of port pins that support CapSense. AMUXBUS A also connects the integrating capacitor C_{MOD} to the sigma-delta converter circuit. AMUXBUS B is used for shielding and is kept at V_{REF} when shield is enabled.

3.1.7 CapSense CSD Shielding

PSoC 4 and PSoC 6 MCU CapSense supports shield electrodes for liquid tolerance and proximity sensing. CapSense has a shielding circuit that drives the shield electrode with a replica of the sensor switching signal to nullify the potential difference between sensors and shield electrode. See Driven-Shield Signal and Shield Electrode and Effect of Liquid Droplets and Liquid Stream on a Self-Capacitance Sensor for details on how this is useful for liquid tolerance.

In the sensing circuit, the sigma delta converter keeps the AMUXBUS A at V_{REF} (see Sigma Delta Converter). The GPIO cells generate the sensor waveforms by switching the sensor between AMUXBUS A and a supply rail (either V_{DD} or ground, depending on the configuration). The shielding circuit works in a similar way; AMUXBUS B is always kept at V_{REF}. The GPIO

cell switches the shield between AMUXBUS B and a supply rail (either V_{DDD} or ground, the same configuration as the sensor). This process generates a replica of the sensor switching waveform on the shield electrode.

For a large shield layer with high parasitic capacitance, an external capacitor (Csh Tank Capacitor) is used to enhance the drive capacity of the shield electrode driver.

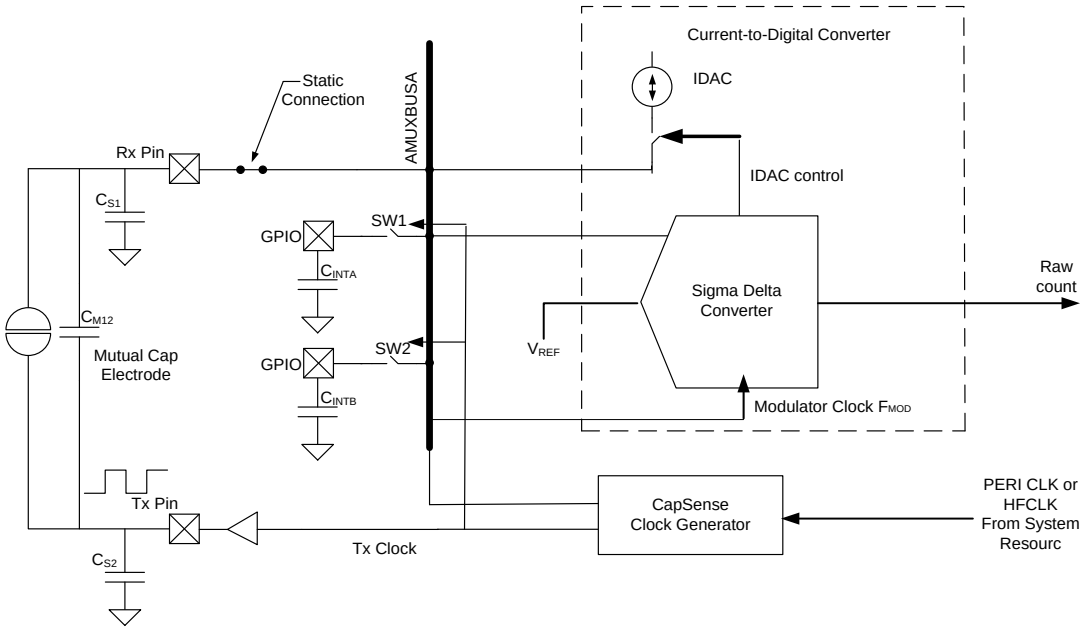
3.2 CapSense CSX Sensing Method

Figure 3-8 is a simple representation of the CSX sensing circuit. The implementation uses the following hardware sub-blocks from CSD HW.

- An 8-bit IDAC and the Sigma Delta Converter
- AMUXBUS A
- CapSense Clock generator for Tx clock and Modulator clock
- V_{REF} and port pins for Tx and Rx electrodes and external caps
- Two external capacitors (CINTA and CINTB). (See Table 7-9 for recommended value of these capacitors).

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

Figure 3-8. CapSense CSX Sensing Method Configuration



The CSX sensing method measures the mutual capacitance between the Tx electrode and Rx electrode, as shown in Figure 3-8. The Tx electrode is excited by a digital waveform (Tx clock), which switches between VDDIO (or VDDD if VDDIO is not available in the given part number) and ground. The Rx electrode is statically connected to AMUXBUS A. The CSX method requires two external integration capacitors, C_{INTA} and C_{INTB}. The value of these capacitors is listed in Table 7-9.

Figure 3-9. CSX Sensing Waveforms

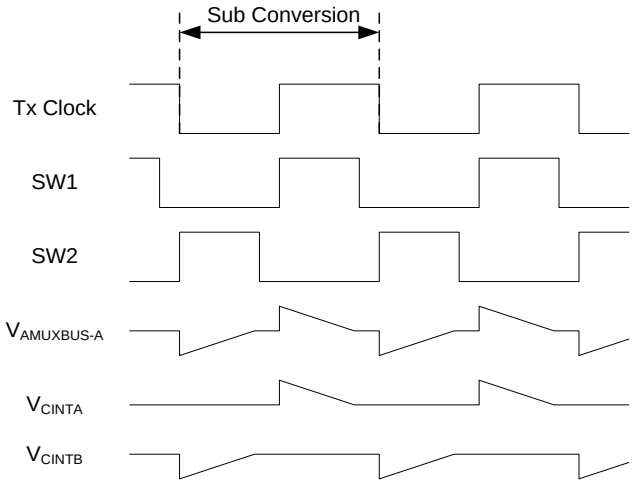


Figure 3-9 shows the voltage waveforms on the Tx electrode and C_{INTA} and C_{INTB} capacitors. The sampling – a process of producing a "sample" – is started by the firmware by initializing the voltage on both external capacitors to V_{REF} and performing a series of sub-conversions. A sub-conversion is a capacitance to count conversions performed within a Tx clock cycle. The sum of results of all sub-conversions in a sample is referred to as "raw count".

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_{INTA} capacitor, which increases the voltage on C_{INTA}. The IDAC is configured in sink mode to discharge the C_{INTA} capacitor back to voltage V_{REF}. 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_{INTB} to go below V_{REF}. The IDAC is configured in source mode to bring the voltage on C_{INTB} back to V_{REF}.

The charge transferred between Tx and Rx electrodes in both the cycles is proportional to mutual capacitance, C_M, between the electrodes. The Sigma Delta Converter controls IDAC for charging or discharging the external capacitors and also it measures the charging and discharging time in terms of modulator clock cycles for a sub-conversion. Multiple sub-conversions are performed during the CSX scanning and the result of each sub-conversion is accumulated 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_{INTA} and C_{INTB} capacitors are charged/discharged within one Tx clock cycle. The CapSense Component / middleware provides an option to automatically calibrate the IDAC. It is recommended to enable this option.

Equation 3-10. Raw Count Relationship for Mutual Capacitance Sensing

$$\mathrm{Rawcount}_{\mathrm{Counter}} = \frac{2\ V_{TX}\ F_{TX}\ C_M\ \mathrm{MaxCount}}{\mathrm{IDAC}}$$

$$\mathrm{MaxCount} = \frac{F_{Mod}\ N_{Sub}}{F_{TX}}$$

Where,

IDAC – IDAC current

C_M – Mutual capacitance between Tx and Rx electrodes

V_{TX} – Amplitude of the Tx signal

F_{TX} – Tx clock frequency

F_{Mod} – Modulator clock frequency

N_{Sub} – Number of Sub-Conversions

When you place a finger on the CSX button, the mutual capacitance between Rx and Tx electrodes decreases, which decreases the raw count. This decrease in raw count from the hardware is inverted by the CapSense Component to make it similar to the raw count change in CSD for a finger touch. The final resulting inverted raw count is given by Equation 3-11.

Equation 3-11. Formula to Determine $Rawcount_{Component}$

$$Rawcount_{Component} = MaxCount - Rawcount_{Counter}$$

See CSX Sensing Method for more details of CSX hardware parameters.

3.3 CapSense Architecture in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU

The fourth-generation CapSense architecture in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU is an improved version of the previous generation CapSense architecture. The main differences in the CapSense architecture are listed in Table 3-2.

Table 3-2. Comparison of CapSense Architecture for CSD and CSX

Feature	Third-Generation CapSense	Fourth-Generation CapSense	Advantages of Fourth-Generation over Third Generation CapSense	
			For CSD	For CSX
Sensor Parasitic Capacitance (C_P) Range	5 pF – 60 pF	5 pF – 200 pF	Supports high-C_P design applications	
Sensing modes	Self-Cap and Mutual-Cap modes[4]	Self-Cap, Mutual-Cap, and ADC modes	The CapSense hardware block can be used as a 10-bit ADC when CapSense sensor scanning is not in progress. Refer CSDADC component/middleware datasheet for detailed ADC specifications.	
V_{REF}	1.2 V	0.6 V to VDDA-0.6 V[5]	Higher Vref allows improved SNR	NA
IDAC LSB Size	1.2 µA, 2.4 µA	37.5 nA, 300 nA, 2.4 µA	Improved sensitivity, small IDAC for improved tuning	
Split IDAC Capability	Requires two IDACs	Requires one IDAC[6]	Requires fewer resources to achieve the same performance and frees up one IDAC for general-purpose use.	NA
EMI Reduction - Digital	Supports only PRS method	Supports additional spread Spectrum clock (SSC) method	More options to control the sense/Tx clock frequency spread for EMI reduction	
Modulator Clock Frequency Range	Lower	Higher	Higher modulator-clock frequency implies faster scans.	Higher modulator-clock frequency implies increased sensitivity and accuracy
Hardware State Machine[7]	No	Yes	Initiation of sensor scanning is less dependent on CPU; there are fewer critical sections during scan initialization.	
Tx Clock Frequency	Supports up-to 300 kHz	Supports much higher clock frequencies (up-to 3 MHz)	NA	Higher Tx clock results in shorter scan time.

[4] PSoC 4100 family does not support CSX sensing method since it does not have UDB resources which is required to implement CSX for this family

[5] The CapSense component automatically selects the V_{REF} voltage depending on the VDDA voltage specified in the cydwr window

[6] Require one IDAC if compensation and modulation IDAC split is 50-50; if it is not 50-50, it requires two IDACs.

[7] The hardware state machine is a logic which controls the CapSense block and sensor scanning.

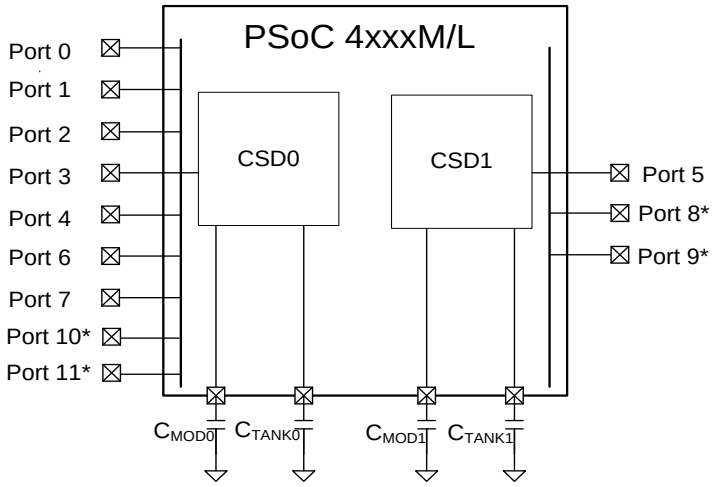
Table 3-3. PSoC Device Family and CapSense Architecture

PSoC Device Family	CapSense Architecture
PSoC 4	Third-Generation CapSense
PSoC 4-M	
PSoC 4100-BLE	
PSoC 4-L	
PSoC 4 S-Series	Fourth-Generation CapSense
PSoC 4100S Plus	
PSoC 4100PS	
PSoC 6 MCU	

3.4 CapSense in PSoC 4xxxM/4xxxL-Series

The PSoC 4xxxM/4xxxL series of devices support two third-generation CapSense blocks – CSD0 and CSD1. Each block has the same functionality and performance as explained in the CapSense 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 3-10.

Figure 3-10. CapSense in PSoC 4 M-Series



*Ports 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 3-4.

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

	CSD0	CSD1
C_{MOD}	See Table 7-10 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.

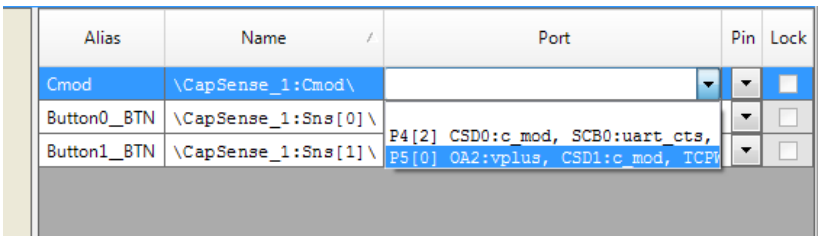
To select a specific CSD block, follow this procedure:

1. Place the CapSense CSD Component in the PSoC Creator schematic.

   Note: The CapSense v6.0 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 v2.60 Component.

2. In the PSoC Creator cydwr pins tab, assign the CMOD pin depending on the required CSD block, as shown in Figure 3-11. For example, if you want to use the CSD0 block, select CMOD pin as P4.2.

Figure 3-11. Selecting CSD0 or CSD1 Block in PSoC 4xxxM/L-Series



3. 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_ ScanAllWidgets();
    CapSense_2_ ScanAllWidgets();


    for(;;)
    {
            /* Check that scanning is completed */
            if (0u == CapSense_1_IsBusy() && 0u == CapSense_2_IsBusy())
            {
                    /* Update all enabled baselines */
                    CapSense_1_ProcessAllWidgets();
                    CapSense_2_ProcessAllWidgets();

                    /* Start scanning all enabled sensors */
                    CapSense_1_ ScanAllWidgets();
                    CapSense_2_ ScanAllWidgets();
            }

     }
```

4 CapSense Design and Development Tools

This chapter introduces the available software tools from Cypress, such as PSoC Creator and ModusToolbox, to develop your CapSense application. For more details, see the user manual of the respective IDE. Table 4-1 shows the supported devices and the CapSense Component/middleware version in PSoC Creator and ModusToolbox.

Table 4-1 Tools and Supported devices

Device Family	Software Tool	CapSense Library
PSoC 4	PSoC Creator	CapSense Component
PSoC 6 MCU	ModusToolbox	CapSense Middleware

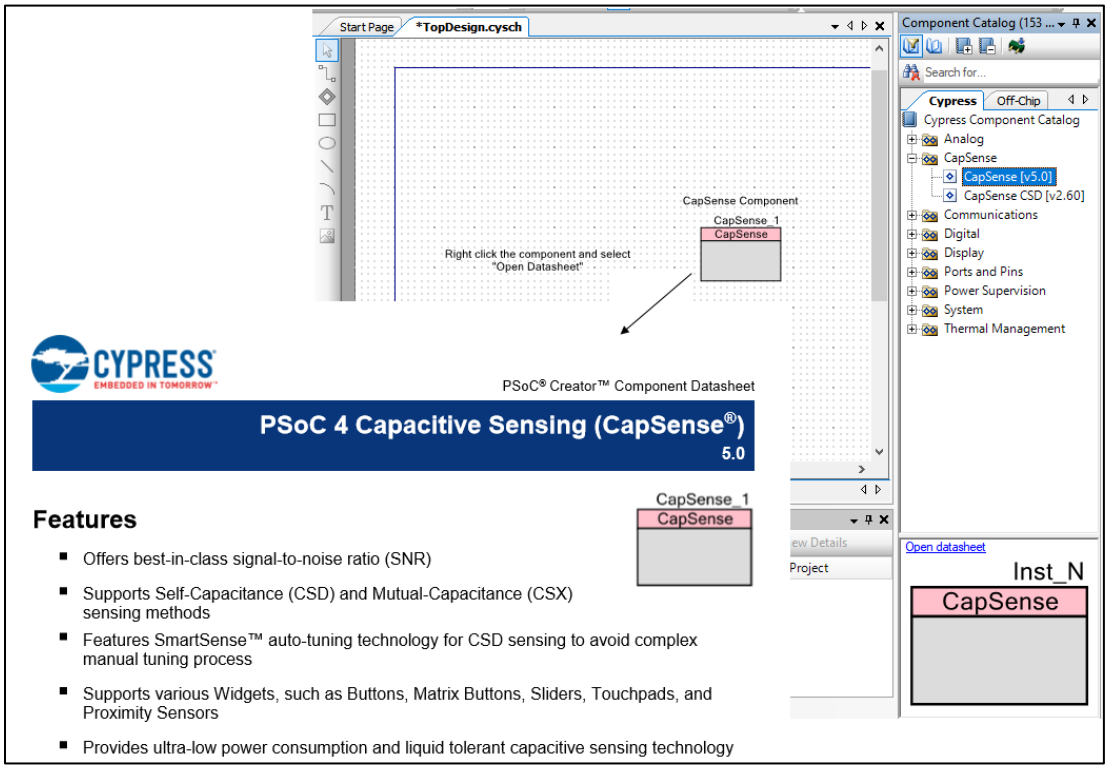
4.1 PSoC Creator

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 by simply configuring this Component. The Component also provides an application programming interface (API) to simplify firmware development. Some PSoC 4 BLE and PSoC 6 MCU devices also support a CapSense Gesture Component (see the corresponding device Device Datasheet to see if your device supports this 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 the Component and select **Open Datasheet**.

The CapSense Component also has a Tuner GUI, called the Tuner , 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, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU devices. This Component should be used when both CapSense and ADC operations are required. This component allows using the CapSense block for ADC operation and touch functionality in a time-multiplexed manner.

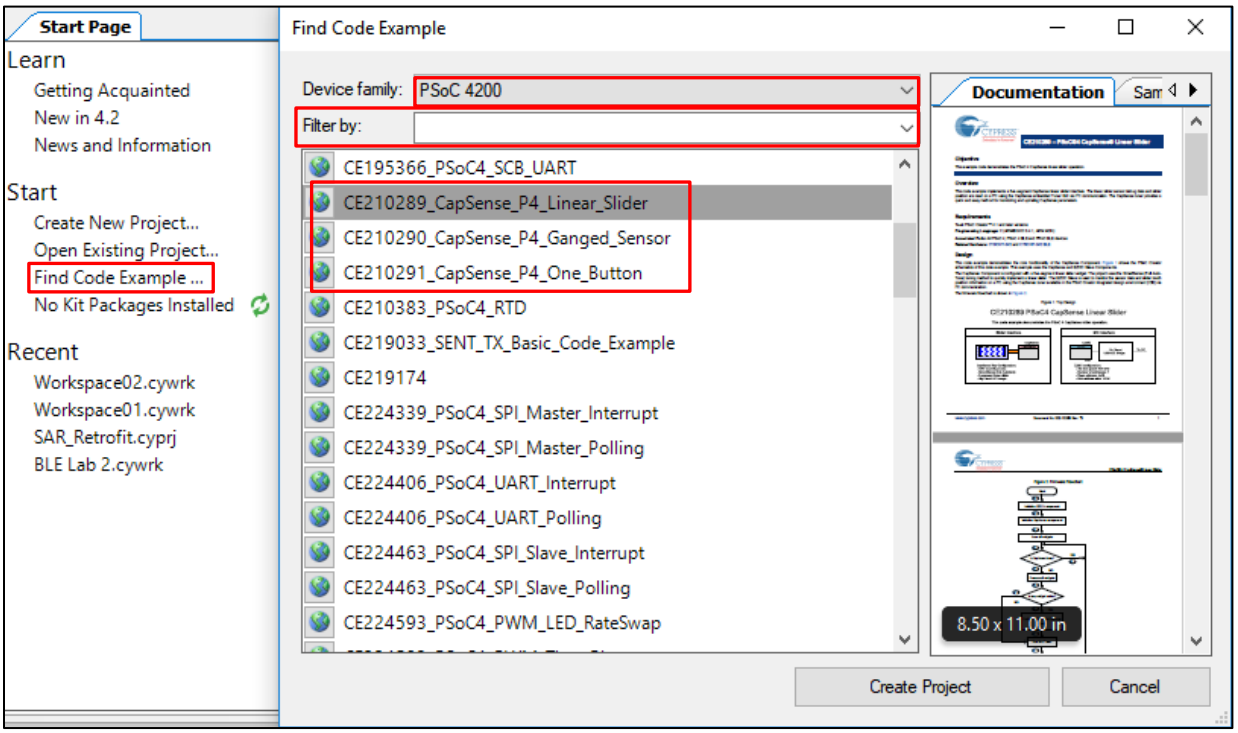
4.1.3 Tuner GUI

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 See the respective Component Datasheet / Middleware Document for the detailed procedure on how to use Tuner GUI.

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 ModusToolbox

Cypress introduces the ModusToolbox software suite for the development of PSoC 6 based CapSense applications. You can download ModusToolbox from here. Before you start working with this software, Cypress recommends that you go through the Quick Start Guide and User Guide. If you have ModusToolbox IDE installed in your system, you can create a CapSense application for the devices supported in ModusToolbox.

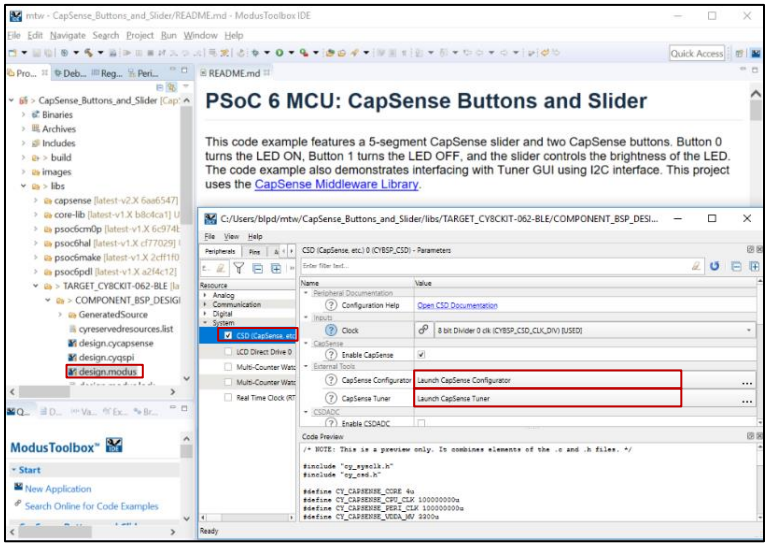
4.2.1 CapSense Middleware

ModusToolbox provides a CapSense middleware, which can be used to create a capacitive touch system in PSoC by simply configuring parameters in the CapSense configuration tool. The middleware also provides an application programming interface (APIs) to simplify firmware development. See the CapSense Middleware library for more details.

4.2.2 CapSense Configurator

The CapSense configurator tool in ModusToolbox is similar to that in PSoC Creator which is used to configure the CapSense hardware and software parameters. For more details on configuring CapSense in ModusToolbox, see the ModusToolbox CapSense Configurator Guide and CapSense Middleware library. Figure 4-3 shows how to open the CapSense configuration tool in ModusToolbox. Alternatively, it can also be opened from the Quick panel in the ModusToolbox. For simplicity of documentation, this design guide shows selecting the CapSense parameter in PSoC Creator CapSense component.

Figure 4-3. CapSense Configurator Tool in ModusToolbox



4.2.3 CSDADC Middleware

This middleware should be used when both the CapSense and ADC operations are required. This middleware allows using the CapSense hardware block for ADC operation and touch functionality in a time-multiplexed manner. It could be used for all three sensing modes i.e., CSD, ADC, and CSX. See the CSDADC Middleware library documentation for more details.

4.2.4 CSDIDAC Middleware

The CSDIDAC middleware allows you to use the CapSense IDAC in a standalone mode. You can use this middleware if you are not using CapSense middleware or if you are using only one IDAC for CapSense. See the CSDIDAC Middleware library documentation.

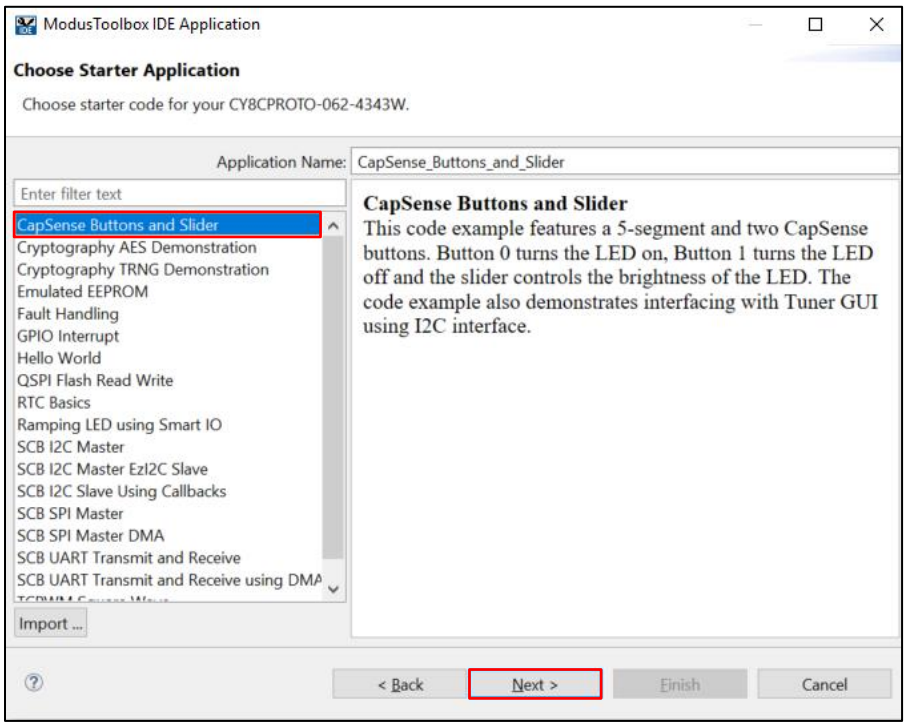
4.2.5 CapSense Tuner

ModusToolbox also supports a GUI tool that can be used for tuning CapSense parameters. This tool can be opened from the Device configurator by selecting *Launch CapSense Tuner* as shown in Figure 4-3. See the CapSense® Tuner Guide documentation.

4.2.6 Example Projects

To quickly start the CapSense system design, start with the example projects provided in ModusToolbox. You can find a CapSense example project by navigating to **File** > **New** > **ModusToolbox IDE**. Choose the appropriate Board Support Package with a device. The Figure 4-4 shows creating a CapSense slider example starter code in ModusToolbox from the list of available code examples.

Figure 4-4. Creating CapSense Slider Example Project in ModusToolbox



4.3 Hardware Kits

Table 4-2 lists the development kits that support evaluation of PSoC 4 and PSoC 6 CapSense.

Table 4-2. PSoC 4 and PSoC 6 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 7×7 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 4100S Plus Prototyping Kit (CY8CKIT-149)	Three self- or mutual-capacitive sensing buttons A six-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
PSoC 4100PS Prototyping Kit (CY8CKIT-147)	No onboard CapSense sensors. The kit can be used to connect external sensors to any I/O pin.
CapSense Proximity Shield (CY8CKIT-024)	A four-element gesture detection and one proximity loop sensor

Development Kit	Supported CapSense Features
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
PSoC 6 Wi-Fi BT Pioneer Kit (CY8CKIT-062-WiFi-BT Pioneer Kit) and PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE Pioneer Kit)	A 5-segment CapSense Slider, two CapSense buttons, one CapSense proximity sensing header, a proximity sensor
PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-063-4343W)	A 5-segment CapSense Slider and two mutual-cap CapSense buttons

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 PSoC 6 CapSense. SmartSense is a firmware algorithm that automatically sets all 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 / Middleware Document or ModusToolbox CapSense Configurator Guide, 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.

SmartSense is the recommended tuning method for all the conventional CapSense applications. You should use SmartSense auto-tuning if your design meets the following requirements:

- The design is for conventional user-interface application like buttons, sliders, and touchpad.
- The parasitic capacitance (C_P) of the sensors is within SmartSense-supported range as mentioned in the “SmartSense operating conditions” section in the Component Datasheet / Middleware Document or ModusToolbox CapSense Configurator Guide.
- The sensor scan time chosen by SmartSense meets the response time/power requirements of the end system.
- SmartSense auto-tuning meets the RAM/flash requirements of the design.

For all other applications, use Manual Tuning. In such cases, you can also use SmartSense as an initial step to find the optimum hardware parameters such as Sense Clock frequency, and then change the tuning mode to manual tuning for further tuning of the CapSense parameters. See Using SmartSense to Determine Hardware Parameters.

Note that manual tuning requires I^2C or UART communication with a host PC.

5.2 SmartSense

5.2.1 Overview

The CapSense algorithm is a combination of hardware and firmware blocks inside PSoC. Therefore, it has several hardware and firmware parameters required for proper operation. These parameters need to be tuned to optimum values for reliable touch detection and fast response.

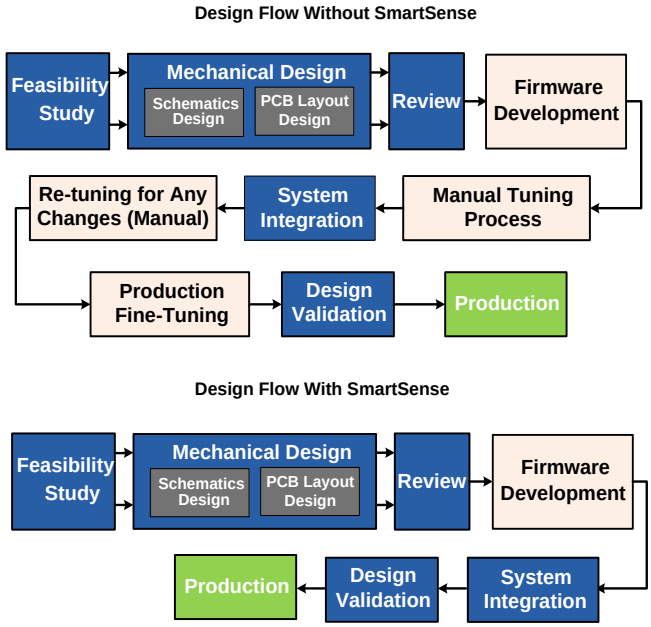
SmartSense is a CapSense tuning method that automatically sets sensing parameters for optimal performance, based on user-specified finger capacitance values, and continuously compensates for system, manufacturing, and environmental changes.

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

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.

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 CapSense hardware and software 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, but allows to set user-defined threshold values (Refer Table 5-6). 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. Refer to the section Using SmartSense to Determine Hardware Parameters for more details.

Table 5-1. CapSense Parameters Auto-tuned in SmartSense

Parameter	Full Auto-Tune Mode	Hardware Parameters Only Mode
Scan Resolution	Calculated once on CapSense initialization.	
Compensation IDAC		
Modulator IDAC		
Sense Clock Frequency		
Modulator Clock Frequency		
Finger Threshold	Calculated once on CapSense initialization based on the selected finger capacitance and updated after each sensor scan.	Manual selection. See Table 5-6.
Noise Threshold		
Hysteresis		
Negative Noise Threshold		
Low Baseline Reset		

5.2.2 SmartSense Full Auto-Tune

In SmartSense Full Auto-tune mode, the only parameter that needs to be tuned by the user is the Finger Capacitance parameter. The Finger Capacitance parameter (C_F) indicates the minimum value of finger capacitance that should be detected as a valid touch by the CapSense Component. Whenever the actual C_F that is added when the finger touches the button sensor is greater than the value specified for the Finger Capacitance parameter in the Component configuration window, the sensor status will change to '1'; however, if the actual C_F added by the finger touch is less than the value specified in the Component configuration window, the sensor status will remain '0'. The way of tuning the finger capacitance is different for button and slider widgets.

Note that even for SmartSense auto-tuning, the CapSense Component allows manual configuration of some general parameters like enable/disable of compensation IDAC, filters, shield such as liquid-tolerance-related parameters and modulator clock. These can be left at their default values for most cases or configured based on the respective sections in this guide.

5.2.2.1 Tuning Button Widgets

This section explains how to choose the Finger capacitance value for the Button widget. You may perform only a coarse tuning of the Finger capacitance parameter for a working design, or you may choose to fine-tune the Finger capacitance value. Coarse-tuning will satisfy the requirements of most designs, but fine-tuning will allow you to choose the most efficient CapSense parameters (i.e., minimum sensor scan time) using SmartSense.

If you do not know the value of C_F (C_F can be estimated based on Equation 2-1), set the Finger capacitance as follows:

1. Start by specifying the highest value for Finger capacitance (from the available options in the list) and check the SNR and button status when the button is touched. Use the Tuner GUI to find the SNR.
2. Decrease the Finger capacitance parameter value until the button status changes to '1' on touch and SNR is greater than 5. Figure 5-2 shows the detailed steps to find the right value for the Finger capacitance parameter in your design.

Enable filters if the SNR of one or more sensors is less than 5:1 when the set finger capacitance is already at the least finger capacitance supported in the Component. You can also enable filters if externally induced noise is causing a decrease in SNR. See Table 5-2 to choose the right filter in this case. There are various types of filters available in the CapSense Component such as Median Filter, IIR filter, and Average Filter; you can enable more than one filter to reduce the noise in the raw count according to the requirement.

If you choose to use an IIR filter, begin by selecting a filter with a higher value of the filter coefficient and keep decreasing it until you achieve an SNR greater than or equal to 5:1. Using filters will affect the response time. You must properly select the filter coefficient such that the response time and SNR requirement are satisfied.

If the SNR is still less than 5:1 even when the smallest allowed value of finger capacitance and proper filter is chosen, see PCB Layout Guidelines, Manual Tuning, or Tuning Debug FAQs like 5.3.5.4, 5.3.5.7, or 5.3.5.10 for more details on debugging the issue.

Figure 5-2. Using SmartSense Auto-Tuning Based CapSense Project in PSoC Creator

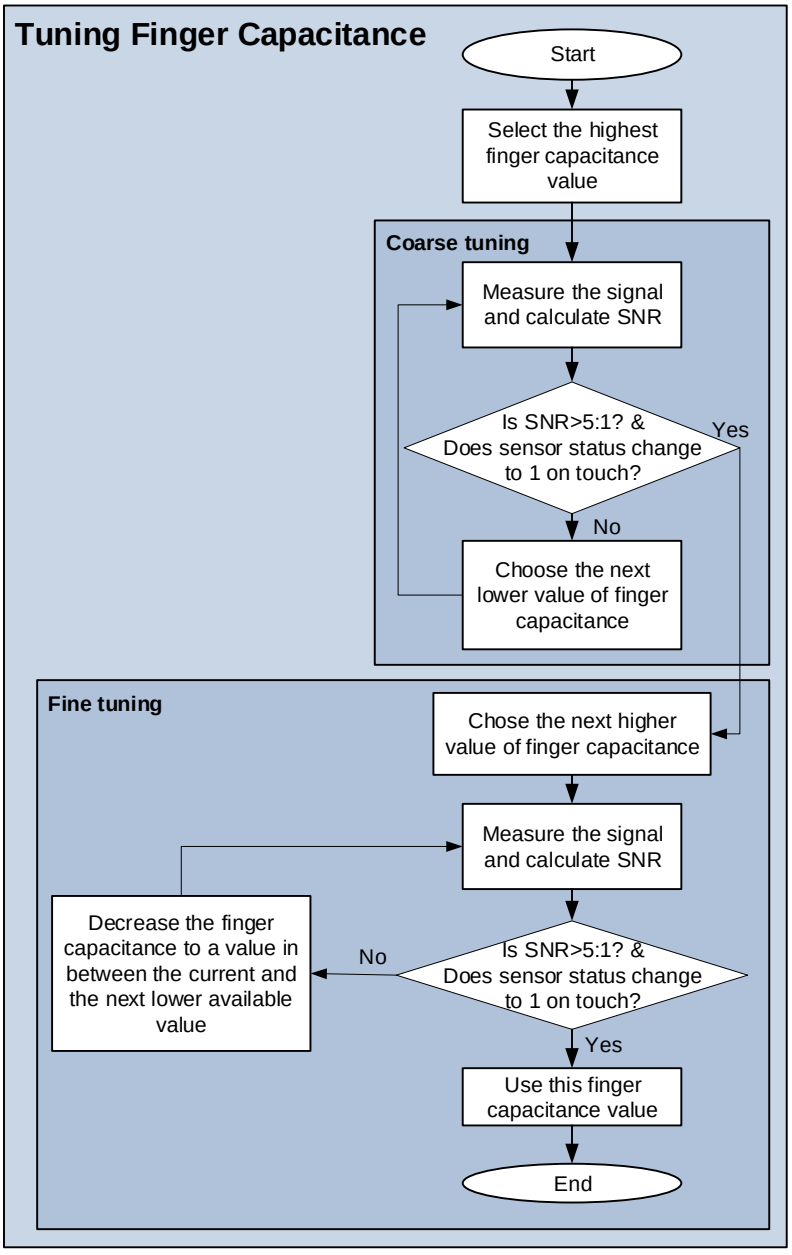


Table 5-2. Raw Data Noise Filters in CapSense Component

Filter	Description	Mathematical Description	Application
Median	Nonlinear filter that takes the three most recent samples and computes the median value.	$y[i] = median\ (x[i],\ \ x[i-1],\ \ x[i-2])$	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 **IIR filter raw count coefficient**. 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.

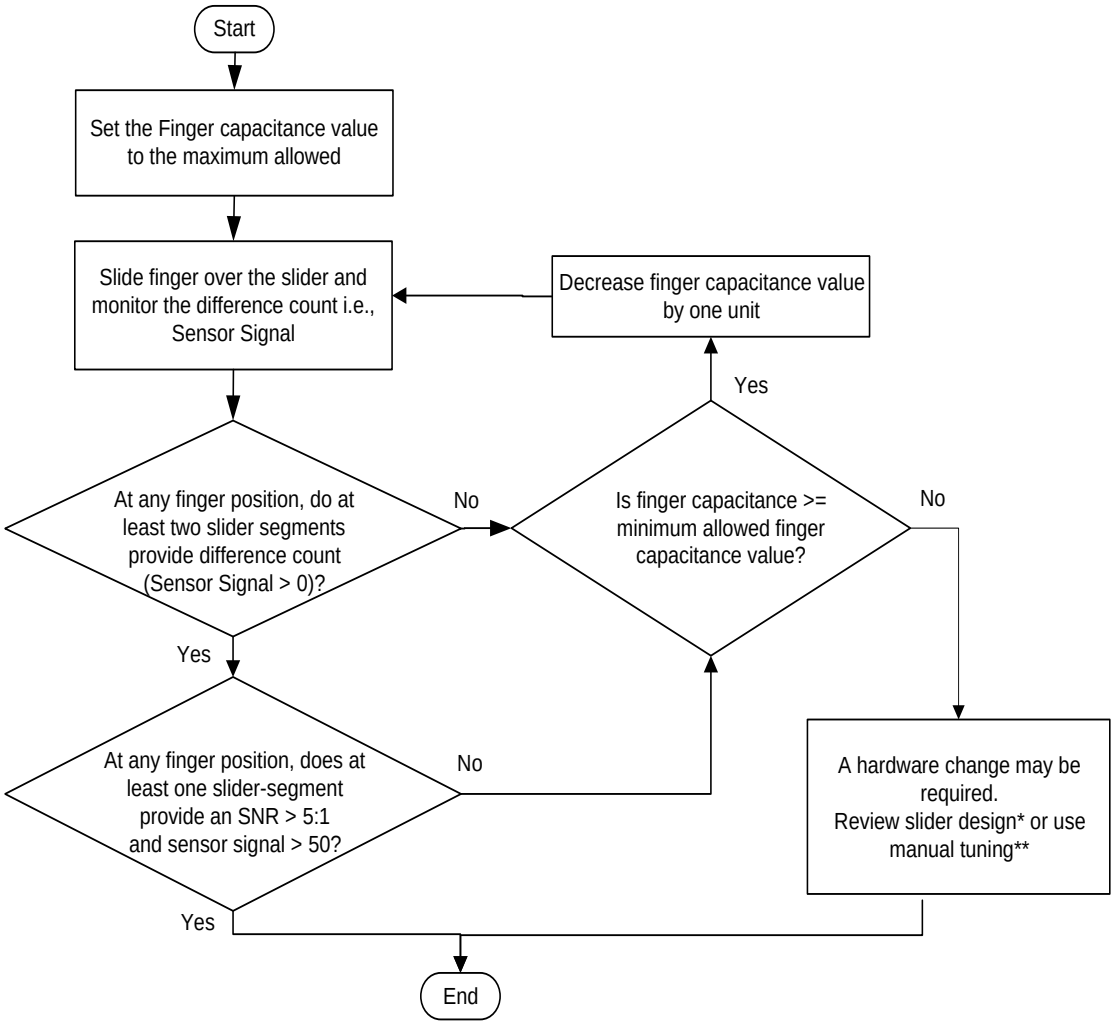
5.2.2.2 Tuning Slider Widgets

For sliders, set finger capacitance to the highest value initially. Slide your finger on the slider. If at any position on the slider, at least one slider segment status is ON and has an SNR >5:1, and at least two slider segments report a “difference count” i.e., a “sensor signal” value greater than 0, use this finger capacitance value**.** Otherwise, decrease the finger capacitance value until the above condition holds true. Figure 5-3 shows how to tune the finger capacitance for slider widget.

If these conditions are not met even after setting minimum allowed Finger Capacitance, use Manual Tuning or revise the hardware according to Slider Design considerations or see Tuning Debug FAQs. Figure 5-3 explains the process of setting finger capacitance value for sliders.

Note: It is recommended to use the compensation IDAC because it allows a higher variation in the parasitic capacitance of the slider segment with respect to the slider segment that has the maximum C_P.

Figure 5-3. Setting Finger Capacitance Value for Sliders



* To review slider design, see the Slider Design section in the Design Considerations chapter.

** To do manual tuning, see the Manual Tuning section in the CapSense Performance Tuning chapter.

5.2.2.3 Tuning Proximity Widgets

See AN92239 Proximity Sensing with CapSense and the “Proximity Sensing” section in Getting Started with CapSense Design Guide.

5.2.3 SmartSense Hardware Parameters-Only Mode

See Table 5-6 for the recommended values for thresholds when the CSD tuning method is SmartSense (Hardware parameters only**)**.

5.2.4 SmartSense for Initial Tuning

See Using SmartSense to Determine Hardware Parameters for more 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 can be achieved by manual tuning.

Some advantages of manual tuning, as opposed to SmartSense auto-tuning are:

- Strict control over parameter settings: SmartSense sets all 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 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: If the parasitic capacitance is higher than the value supported by SmartSense, you should use manual tuning. See the Component Datasheet / Middleware Document for more details on the supported range of parasitic capacitance by SmartSense.

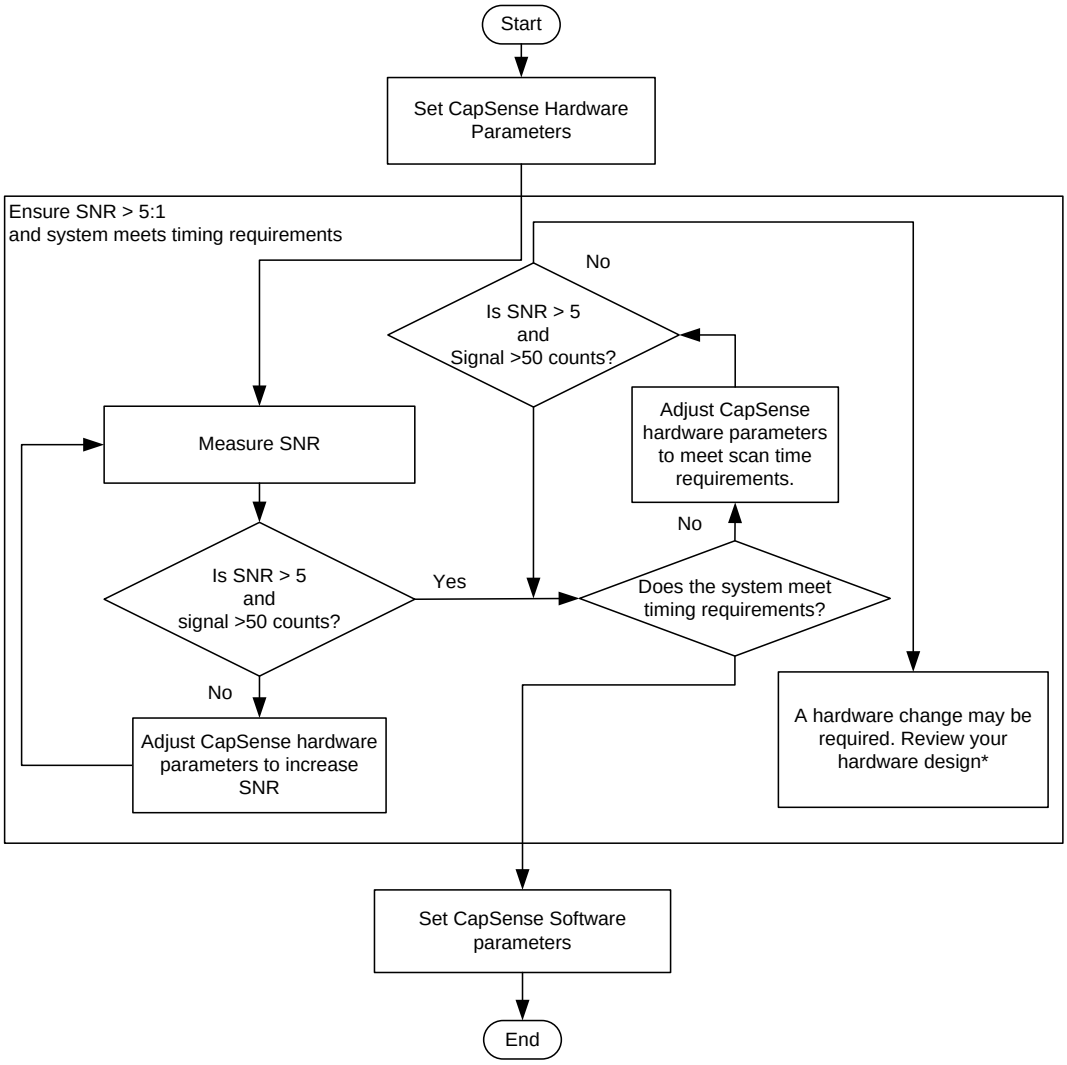
The manual tuning process can be summarized in the following three steps and is shown in Figure 5-4.

1. Set initial values of CapSense Component Hardware Parameters using SmartSense (see Using SmartSense to Determine Hardware Parameters) or determine the values manually.
2. Tune CapSense Component hardware parameters to ensure that Signal-to-Noise Ratio is greater than 5:1 with a signal of at least 50 counts while meeting the system timing requirements.
3. Set optimum values of CapSense Component Software Parameters.

The following sections describe the fundamentals of manual tuning and the above three steps in detail. Knowledge of the CapSense architecture in PSoC is a prerequisite for these sections. See Capacitive Touch Sensing Method and CapSense CSD Sensing or CapSense CSX Sensing Method to become familiar with CapSense architecture in PSoC.

Depending upon the sensing method selected, the manual tuning procedure will differ. See CSD Sensing Method, CSX Sensing Method section for their respective manual tuning procedures. You can skip these sections if you are not planning to use manual tuning in your design. Figure 5-4 shows a general manual tuning procedure.

Figure 5-4. Manual Tuning Process Overview



* To review the hardware design, see the Sensor Construction and PCB Layout Guidelines sections in the Design Considerations chapter.

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 CSD sensing method and procedure of manual tuning for 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 for a finger touch on the sensor. If there is more gain, the signal is higher, and a higher signal means a higher achievable Signal-to-Noise Ratio. Note that an increased gain may result in an increase in both signal and noise. However, if required, you can use firmware filters to decrease noise. For details on available firmware filters, see Table 5-2.

Conversion Gain in Single IDAC Mode

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

Equation 5-1. Raw Count Relationship to Sensor Capacitance

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

Here, C_S is the sensor capacitance, $C_S = C_P$ if there is no finger present on sensor, and $C_S = (C_P + C_F)$ when there is a finger present on the sensor and G_C is the capacitance to digital conversion gain of CapSense CSD. The approximate value of this conversion gain using the IDAC Sourcing mode, according to Equation 3-6 and Equation 5-1 is:

Equation 5-2. Capacitance to Digital Converter Gain

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

Where,

V_{REF} is the comparator reference voltage.

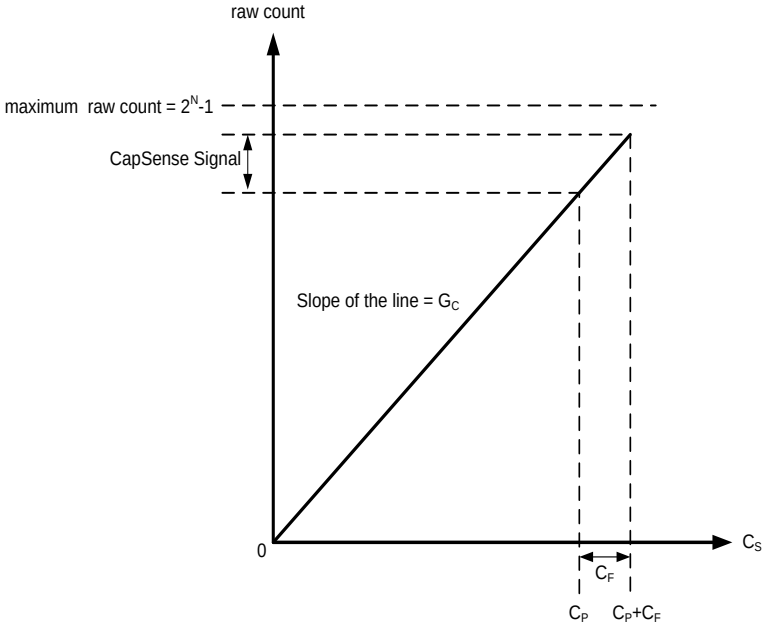
F_{SW} is the Sense clock frequency

I_{MOD} is the Modulator IDAC current

N is the resolution of the sigma to delta converter.

The tunable parameters of the conversion gain are V_{REF}, F_{SW}, I_{MOD}, and N. Figure 5-5 shows a plot of raw count versus sensor capacitance.

Figure 5-5. Raw Count versus Sensor Capacitance



The change in raw counts when a finger is placed on the sensor is called CapSense signal. Figure 5-6 shows how the value of the signal changes with respect to the conversion gain.

Figure 5-6. Signal Values for Different Conversion Gains

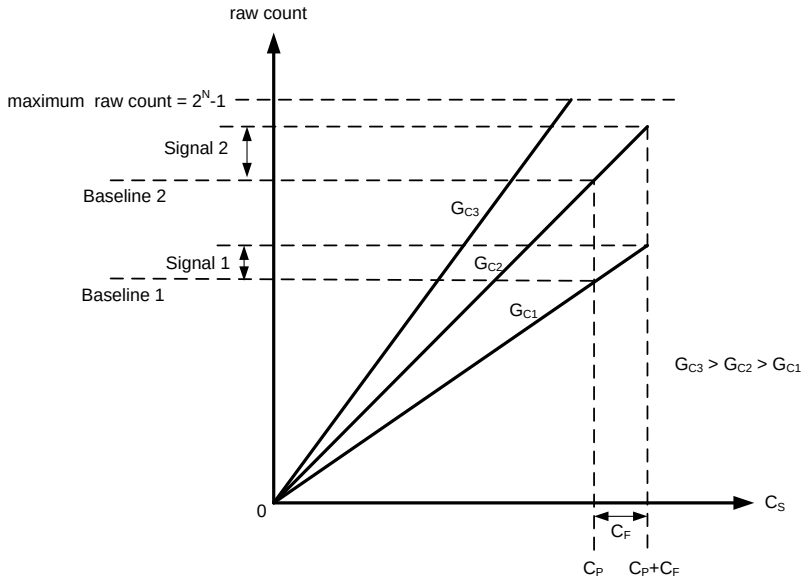
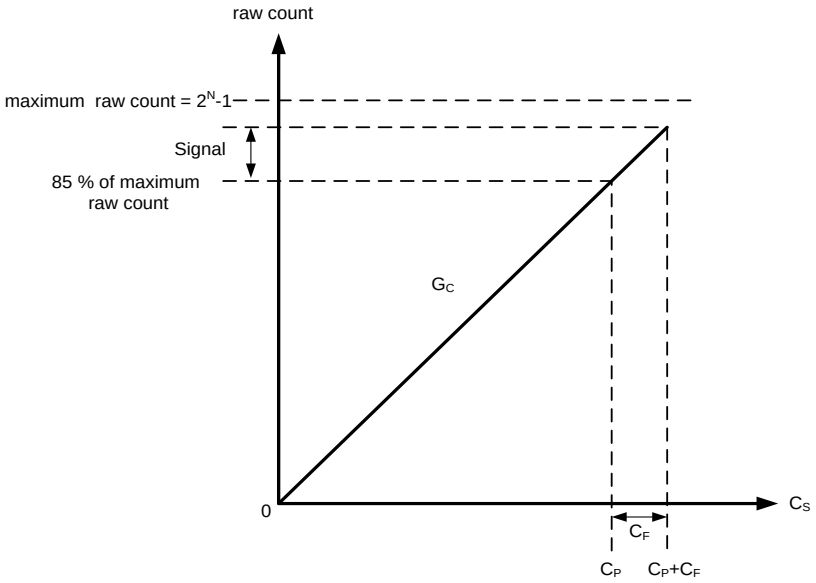


Figure 5-6 shows three plots corresponding to three conversion gain values G_{C3}, G_{C2}, and G_{C1}. An increase in the conversion gain results in higher signal value. However, this increase in the conversion gain also moves the raw count corresponding to C_P (i.e., Baseline) towards the maximum value 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 CSD parameters such that when there is no finger on the sensor, i.e. when $C_S = C_P$, the raw count = 85% of (2^N-1) as Figure 5-7 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 5-7. Recommended Tuning



Conversion Gain in Dual IDAC Mode

The equation for raw count in the dual IDAC mode, according to Equation 5-2 and Equation 3-8 is:

Equation 5-3. Dual IDAC Mode Raw Counts

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

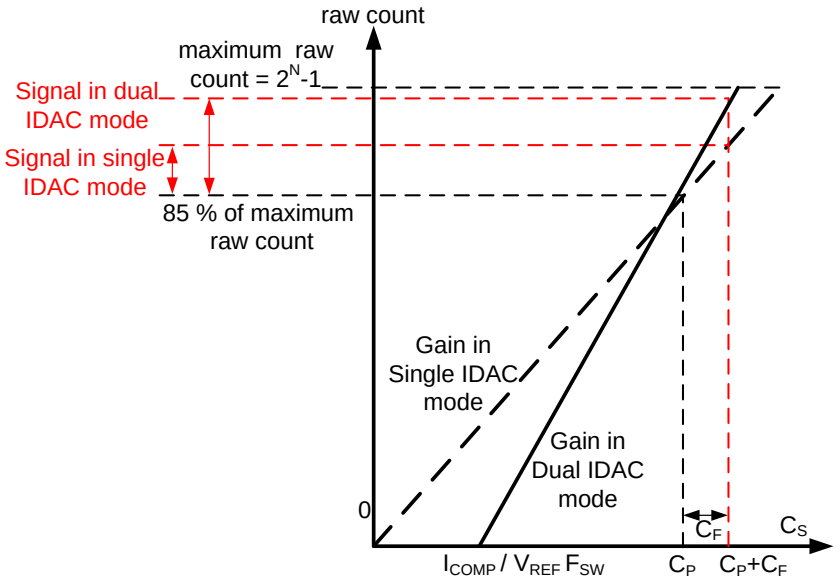
Where,

I_{COMP} is the compensation IDAC current

G_C is given by Equation 5-2

In both single IDAC and dual IDAC mode, tune the CSD parameters, so that when there is no finger on the sensor, i.e. when $C_S = C_P$, the raw count = 85% of (2^N-1), as Figure 5-8 shows, to ensure high conversion gain, to avoid Flat Spots, and to avoid raw count saturation due to environmental changes.

Figure 5-8. Recommended Tuning in Dual IDAC Mode



As Figure 5-8 shows, the 85% requirement restricts to a fixed gain in single-IDAC mode, while in dual-IDAC mode, gain can be increased by moving the C_S axis intercept to the right (by increasing I_{COMP}) and correspondingly decreasing the modulator IDAC (I_{MOD}) to still achieve raw count = 85% of (2^N-1) for $C_S = C_P$. Using dual IDAC mode this way brings the following changes to the Raw Count versus C_P graph:

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

Equation 5-4. Cs Axis Intercept with Regards to I_{COMP}

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

b. The value of I_{MOD} in the dual IDAC mode is half compared to the value of I_{MOD} in the single IDAC mode (all other parameters remaining the same), so the gain G_C in the dual IDAC mode is double the gain in the single IDAC mode according to Equation 5-2. 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 5-2 and Equation 5-3 as well as the following points in mind:

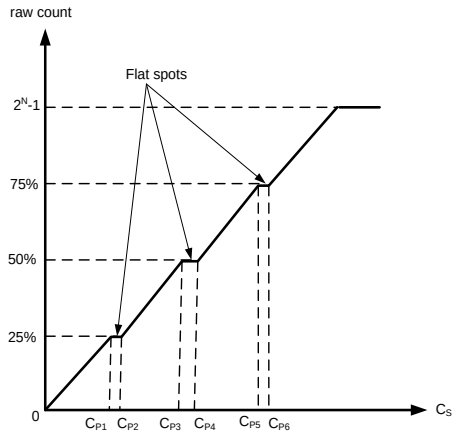
1. Higher gain leads to increased sensitivity and better overall system performance. However, do not set the gain such that raw counts saturate, as the plot of gain G_{C3} shows in Figure 5-6. It is recommended to set the gain in such a way that the raw count corresponding to C_P is 85 percent of the maximum raw count for both the single IDAC and dual IDAC mode.
   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 Equation 5-7 indicates.
2. 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.
3. 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 5-7 and Figure 5-8.

4. 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 3-4. 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 5-5 and Figure 5-8 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 5-9 shows. This range is known as a dead-zone or a flat-spot.

Figure 5-9. 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 Using SmartSense to Determine Hardware 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 Sense Clock 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 5-3 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 CapSense CSD system.

For simplicity of documentation, this design guide shows selecting the CapSense parameters in PSoC Creator. You can use the same procedure to set the parameters in ModusToolbox. However, in ModusToolbox, you set the Sense clock and Modulator clock using divider values while in the PSoC Creator you specify the frequency value directly in the configurator. For more details on configuring CapSense, see the Component Datasheet / Middleware Document.

Table 5-3. CapSense Component Hardware Parameters

Sl. No.	CapSense Parameter in PSoC Creator	CapSense Parameter in ModusToolbox
1	Sense Clock Frequency	Sense Clock Divider
2	Sense Clock Source	Sense Clock Source
3	Modulator Clock Frequency	Modulator Clock Divider
4	Modulator IDAC	Modulator IDAC
5	Compensation IDAC	Compensation IDAC
6	Scan Resolution	Scan Resolution

5.3.2.2.1 Using SmartSense to Determine Hardware Parameters

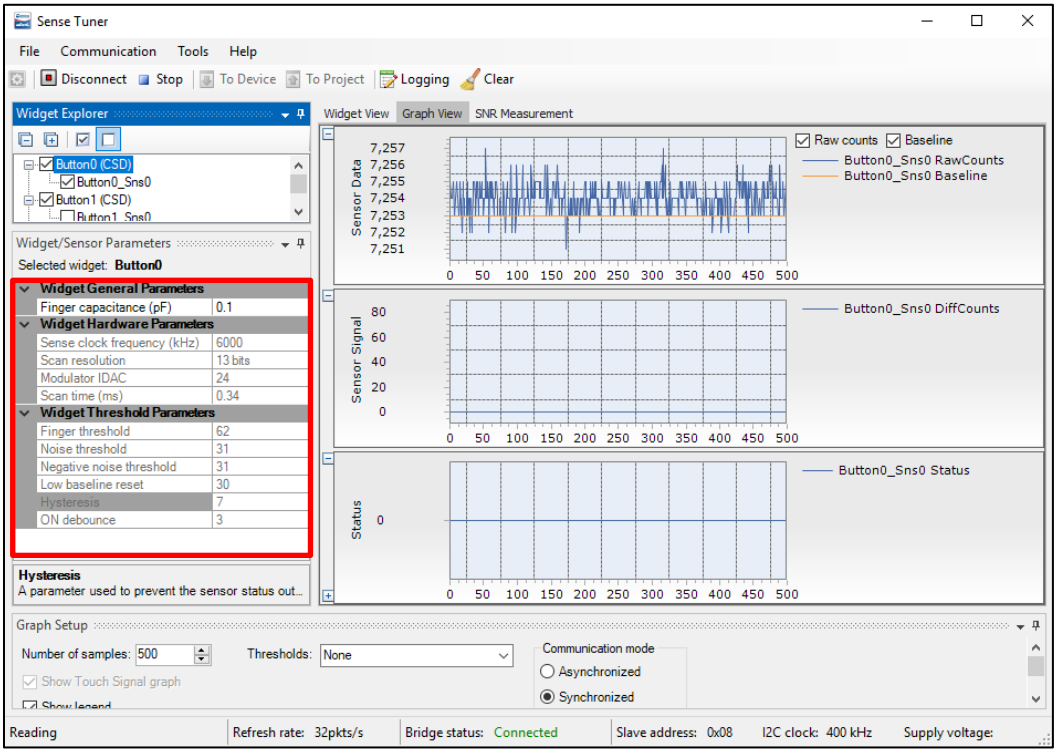
Parameters listed in Table 5-3 are CapSense hardware parameters. Tuning these parameters manually for optimal value is a time-consuming task. You can use SmartSense to determine these hardware parameters and take it as an

initial value for manual tuning. You can fine-tune these values to further optimize the scan time, SNR, power consumption, or improving EMI/EMC capability of the CapSense system.

Set the tuning mode to SmartSense and configure default values for parameters other than finger capacitance. see to the SmartSense section for the tuning procedure and use the Tuner GUI to read back all the hardware parameters set by SmartSense. See the Component Datasheet / Middleware Document for more details on how to use the Tuner GUI.

Figure 5-10 shows the best hardware parameter values in the Tuner GUI that are tuned by SmartSense for a specific hardware to sense a minimum finger capacitance of 0.1 pF.

Figure 5-10. Read-back Hardware Parameter Values in Tuner GUI



5.3.2.2.2 Manually Tuning Hardware Parameters

5.3.2.2.2.1 Sense Clock Parameters

There are two parameters that are related to Sense clock: Sense clock source and Sense clock frequency.

5.3.2.2.2.1.1 Sense Clock Source

Select “Auto” to let the Component automatically choose the best Sense clock source from Direct, PRSx, and SSCx for each widget. If not selecting Auto, select the clock source based on the following:

- Use PRSx (Pseudo Random Sequence) modes to remove flat spots.
- Use SSCx (Spread Spectrum Clock) modes for reducing EMI/EMC noise at a particular frequency. This feature is available in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 family of devices. In this case, the frequency of the sense clock is spread over a predetermined range.
- Use Direct clock for absolute capacitance measurement.

When selecting PRSx as the sense clock source, ensure that the sequence completes within one conversion cycle; not letting the sequence complete may cause high noise in raw count. i.e., $T_{PRS} << T_{SCAN}$

For PRS clock, use the following equations to calculate one PRS sequence completion cycle and scan time.

Equation 5-5. Sensor Scan Time

$$T_{SCAN} = \frac{2^N - 1}{F_{MOD}}, \textit{ here } N \textit{ is the Scan resolution}$$

Equation 5-6. PRS Sequence Period

$$T_{PRS} = \frac{2^{N\_PRS}-1}{F_{SW}}, \; here \; N\_PRS \; is \; either \; 8 \; or \; 12$$

See the Component Datasheet / Middleware Document for more details on the rules and recommendations for SSCx selection.

5.3.2.2.2.1.2 Sense Clock Frequency

The sense clock frequency should be selected so that the sensor will charge and discharge completely in each sense clock period as Figure 3-5 shows.

This requires that the maximum sense clock frequency be chosen per Equation 5-7:

Equation 5-7. Sense Clock Maximum Frequency

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

Equation 5-8 Total Series Resistance

$$R_{SeriesTotal} = R_{EXT} + R_{GPIO}$$

Here, C_P is the sensor parasitic capacitance, and $R_{SeriesTotal}$ is the total series-resistance, including the 500-ohm resistance of the internal switches, the recommended external series resistance of 560 ohm (connected on PCB trace connecting sensor pad to the device pin), and trace resistance if using highly resistive materials (example ITO or conductive ink); i.e., a total of 1.06 kΩ plus the trace resistance.

The value for C_P can be estimated using the CSD Built-in-Self-test API; `GetSensorCapacitance()`. See the Component Datasheet / Middleware Document for details.

Equation 5-2 shows that it is best to use the maximum clock frequency to have a good gain; however, you should ensure that the sensor capacitor fully charges and discharges as shown in Figure 3-5.

Generally, the C_P of the shield electrode will be higher compared to sensor C_P. For good liquid tolerance, the shield signal should satisfy the condition mentioned in section 5.3.2.2.3.1. If it is not satisfied, reduce the sense clock frequency further to satisfy the condition.

5.3.2.2.2.2 Modulator Clock Frequency

The modulator clock governs the conversion time for capacitance-to-digital conversion, also called the “sensor scan time” (Equation 3-4).

A lower modulator clock frequency implies the following:

- Longer conversion time (see Equation 5-7, Equation 5-5)
- Lower peak-to-peak noise on raw count because of longer integration time of the sigma-delta converter
- Wider Flat Spots

Select the highest frequency for the shortest conversion time and narrower flat spots for most cases. Use slower modulator clock to reduce peak-to-peak noise in raw counts if required.

5.3.2.2.2.3 Modulation and Compensation IDACs

CSD supports two IDACs: Modulation IDAC and Compensation IDAC that charge C_{MOD} as Figure 3-1 shows. These govern the Conversion Gain and CapSense Signal for capacitance-to-digital conversion. The CapSense Component allows the following configurations of the IDACs:

- Enabling or disabling of Compensation IDAC
- Enabling or disabling of Auto-calibration for the IDACs
- DAC code selection for Modulator and Compensation IDACs if auto-calibration is disabled

5.3.2.2.2.4 Compensation IDAC

Enabling the compensation IDAC is called “dual IDAC” mode, and results in increased signal as explained in

Conversion Gain in Dual IDAC **Mode**. Enable the compensation IDAC for most cases. Disable the compensation IDAC only if you want to free the IDAC for other general-purpose analog functions.

5.3.2.2.2.5 Auto-Calibration

This feature enables the firmware to automatically calibrate the IDAC to achieve the required calibration target of 85%. It is recommended to enable auto-calibration for most cases. Enabling this feature will result in the following:

- Fixed raw count calibration to 85% of max raw count even with part-to-part C_P variation
- Avoids Flat Spots
- Automatically selects the optimum gain

If your design environment includes large temperature variation, you may find that the 85% IDAC calibration level is too high, and that the raw counts saturate easily over large changes in temperature, leading to lower SNR. If this is the case, you can adjust the calibration level lower by using `CapSense_CSDCalibrateWidget()` in your firmware.

For proper functioning of CapSense under diverse environmental conditions, it is recommended to avoid very low or high IDAC codes. For a 7-bit IDAC, it is recommended to use IDAC codes between 18-110 from the possible 0 to 127 range. You can use CapSense tuner to confirm that the auto-calibrated IDAC values fall in this recommended range. If the IDAC values are out of the recommended range, based on Equation 5-1, Equation 5-2, and Equation 5-3, you may change the V_{ref} or F_{sw} to get the IDAC code in proper range.

Disable IDAC auto-calibration if a change in C_P needs to be detected by measuring the raw count level at reset, for example:

- Detecting large variations in sensor C_P across boards or layout problems
- Detecting finger touch at reset
- Advanced CapSense methods like liquid-level sensing, for example, to have different raw count level for different liquid levels at reset

5.3.2.2.2.6 Selecting DAC Codes

This is not the recommended approach. However, this could be used only If you want to disable auto-calibration for any reason. To get the IDAC code, you may first configure CapSense Component with auto-calibration enabled and all other hardware parameters the same as required for final tuning and read back the calibrated IDAC values using Tuner GUI. Then, re-configure the CapSense Component to disable auto-calibration and use the obtained IDAC codes as fixed DAC codes read-back from the Tuner GUI.

5.3.2.2.2.7 Scan Resolution

It governs the sensor scan time per Equation 5-5 and the conversion gain per Equation 5-1, Equation 5-2, and Equation 5-3. Scan resolution needs to be selected to maintain a balance between the signal and scan time.

Higher Scan resolution implies the following:

- Longer scan time per Equation 5-5
- Higher SNR on raw counts (increase in resolution increases the signal at a disproportionate rate to noise)

In general, it is recommended to tune the resolution to achieve as high SNR as possible; however if the system is constrained on power consumption and/or response time, set the lowest resolution to achieve at-least 5:1 SNR in the end system. Note that you should tune the scan resolution for less than 10:1 SNR only if you have scan time or power number constraints.

5.3.2.2.3 Tuning Shield Electrode

The shield related parameters need to be additionally configured or tuned differently when you enable the Shield electrode in the CSD sensing method for liquid tolerance or reducing the Cp of the sensor.

5.3.2.2.3.1 Shield Electrode Tuning Theory

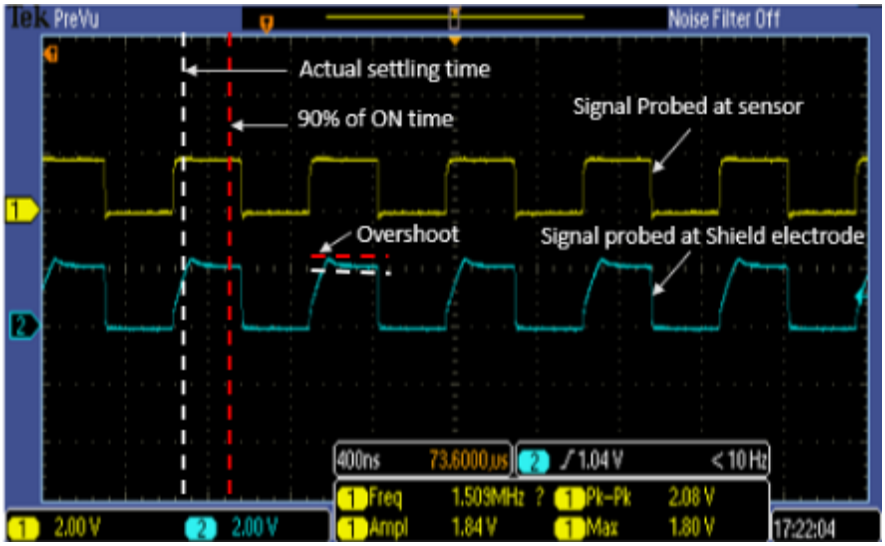
Ideally, the shield waveform should be exactly the same as that of the sensor as explained in Driven-Shield Signal and Shield Electrode. However, in practical applications, the shield waveform may have a higher settling time and an overshoot error. Observe the sensor and shield waveform in the oscilloscope; an example waveform is shown in Figure 5-11. The shield waveform should settle to the sensor voltage within 90% of ON time of the sense clock waveform and the overshoot error of the shield signal with respect to VREF should be less than 10%.

If these conditions are not satisfied, you will observe a change in raw count of the sensors when touching the shield hatch; in addition, if inactive sensors are connected to shield as mentioned in Inactive Sensor Connection, touching

one sensor can cause change in raw count on other sensors, which indicates that there is cross talk if the shield electrode is not tuned properly.

In SmartSense, the sense clock frequency is automatically set. Check if these conditions are satisfied. If not satisfied, switch to Manual Tuning and set the Sense clock frequency manually so that these conditions are satisfied. You can also tune the Shield SW Resistance parameter to reduce the overshoot error.

Figure 5-11. Properly Tuned Shield Waveform



5.3.2.2.3.2 Tuning Shield-Related Parameters

5.3.2.2.3.2.1 Enable Shield Tank Capacitor

Enabling a shield tank capacitor increases the drive strength of the shield thus allowing the shield signal to settle to the sensor voltage faster as required. It is recommended to use the shield tank capacitor for PSoC 4A-S and PSoC 6 MCU family of devices. For PSoC 4A, PSoC 4A-L, and PSoC 4A-M family of devices, the shield tank capacitor doesn't prove very advantageous because it doubles the shield series resistance. It is recommended to keep this option disabled for these device families.

5.3.2.2.3.2.2 Shield Electrode Delay

For proper operation of the shield electrode, the shield signal should match the sensor signal in phase. Due to the difference in trace lengths of the sensor and shield electrodes, the shield waveform may arrive earlier to the sensor waveform. 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. Available delays vary depending on the device selected.

5.3.2.2.3.2.3 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, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices. The default value of shield switch resistance is Medium. Table 5-4 shows the effect of the Shield SW resistance value. You should select this value based on the application requirement; in addition, ensure that it satisfies the conditions in Shield Electrode Tuning Theory.

Table 5-4. Shield SW Resistance Selection Guidelines

Lower Switch Resistance	Higher Switch Resistance
Large overshoot error	Smaller overshoot error
Higher electromagnetic emission	Lower electromagnetic emission
Faster settling time i.e., higher max sense clock frequency	Slower settling time i.e., lower max sense clock frequency

5.3.2.2.3.2.4 Number of Shield Electrodes

This parameter specifies the number of shield electrodes required in the design. Most designs work 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. See Layout Guidelines for Shield Electrode.

5.3.2.2.3.2.5 Inactive Sensor Connection

When the shield electrode is enabled for liquid-tolerant designs, or if you want to use shield to reduce the sensor parasitic capacitance, this option should be specified as “Shield”; otherwise, select “Ground”.

However, there is a risk of higher radiated emission due to inactive sensors getting connected to Shield. In such situations, use the CapSense API to manually control inactive sensor connections. Instead of connecting all unused sensors to the shield, connect only the opposing inactive sensors or inactive sensors closer to the sensor being scanned to shield for reducing the radiated emission.

5.3.2.3 Selecting CapSense Software Parameters

CapSense software parameters govern the sensor status based on the raw count of a sensor. Table 5-5 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 shows the recommended values for the software threshold parameter and they are applicable for most of the designs. However, if there are any external noise present in the end system, you must modify these thresholds accordingly to avoid any sensor false trigger.

Table 5-5. CapSense Component Widget Threshold Parameters

Sl. No.	CapSense Component Parameter Name in PSoC Creator / ModusToolbox
1.	Finger Threshold
2.	Noise Threshold
3.	Hysteresis
4.	ON Debounce
5.	Sensor Auto-Reset
6.	Low Baseline Reset
7.	Negative Noise Threshold

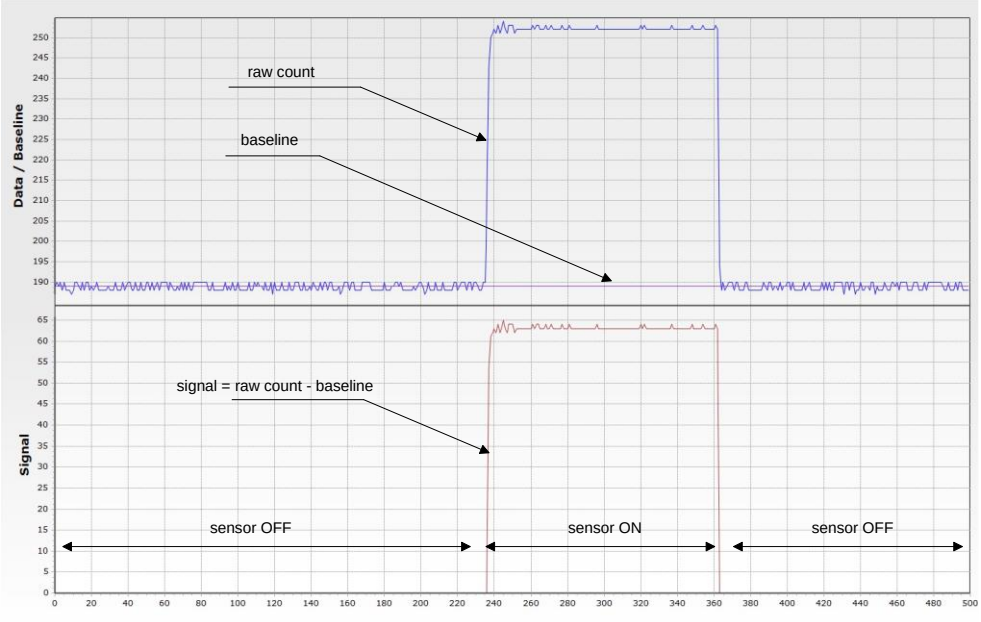
Table 5-6. Recommended values for the Threshold parameters

Sl. No.	CapSense Threshold Parameter	Recommended Value
1.	Finger Threshold	80 percent of signal
2.	Noise Threshold	40 percent of signal
3.	Hysteresis	10 percent of signal
4.	ON Debounce	3
5.	Low Baseline Reset	30
6.	Negative Noise Threshold	40 percent of signal

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 5-12 shows the concept of raw count, baseline, and signal.

Figure 5-12. Raw Count, Baseline, and Signal

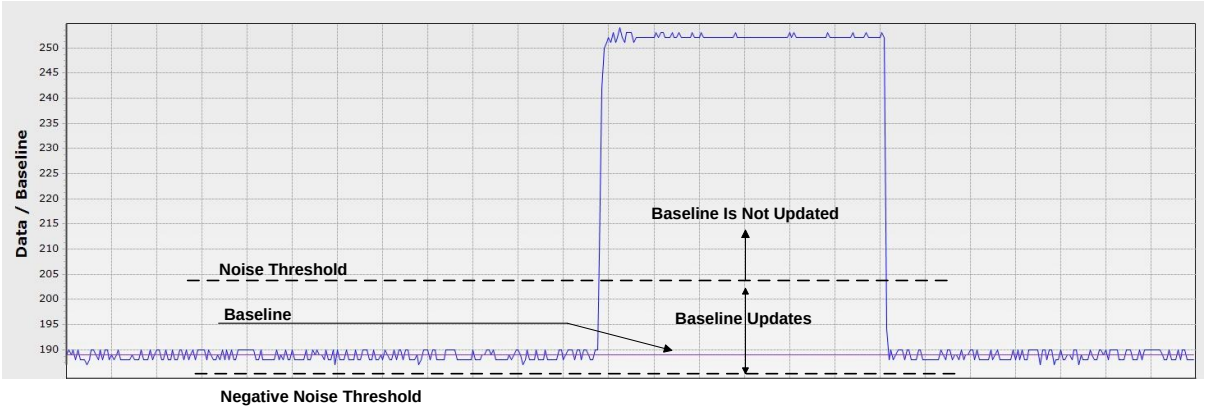


5.3.2.3.2 Baseline Update Algorithm

To properly tune the CapSense software, that is, the threshold parameters, it is important to understand how baseline is calculated and how the threshold parameters affect the baseline update.

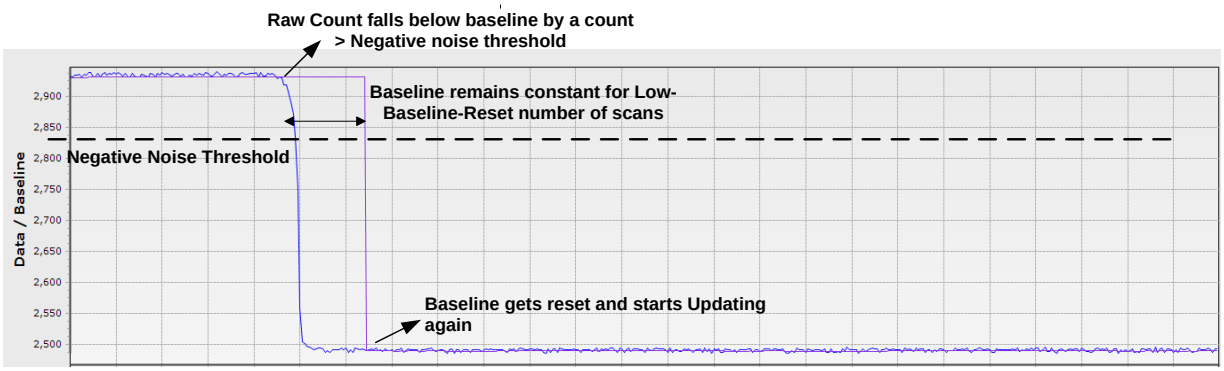
Baseline is a low-pass-filtered version of raw counts. As Figure 5-13 shows, baseline is updated by low-pass-filtering raw counts if the current raw count is within a range of (Baseline – Negative Noise Threshold) to (Baseline + Noise Threshold). If the current raw count is higher than baseline by a value greater than noise threshold, baseline remains at a constant value equal to prior baseline value.

Figure 5-13. 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 5-14 shows.

Figure 5-14. 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 5-9 shows.

Equation 5-9. Sensor State

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

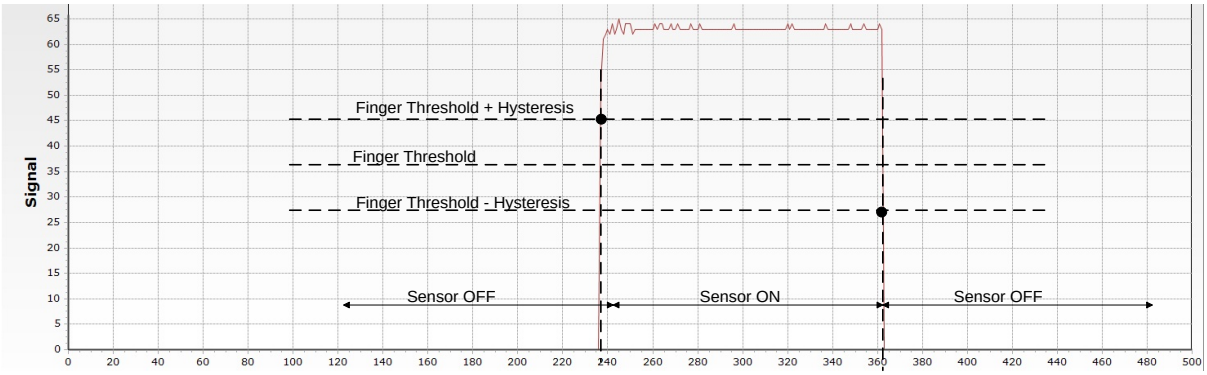
Note that signal in the above equation refers to the difference: Raw Count – Baseline, when the sensor is touched, as Figure 5-12 shows.

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 5-9 and Figure 5-15 show. Hysteresis provides immunity against noisy transitions of sensor state. The hysteresis parameter setting must be lower than the finger threshold parameter setting. It is recommended to set hysteresis to 10 percent of the signal.

Figure 5-15. Hysteresis



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 5-13 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 2-29 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 2-29 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 5-10.

Equation 5-10. 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:

- 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 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 5-11. Sensor State with Debounce

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

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 5-12. Relationship between Debounce and Sensor Response Time

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

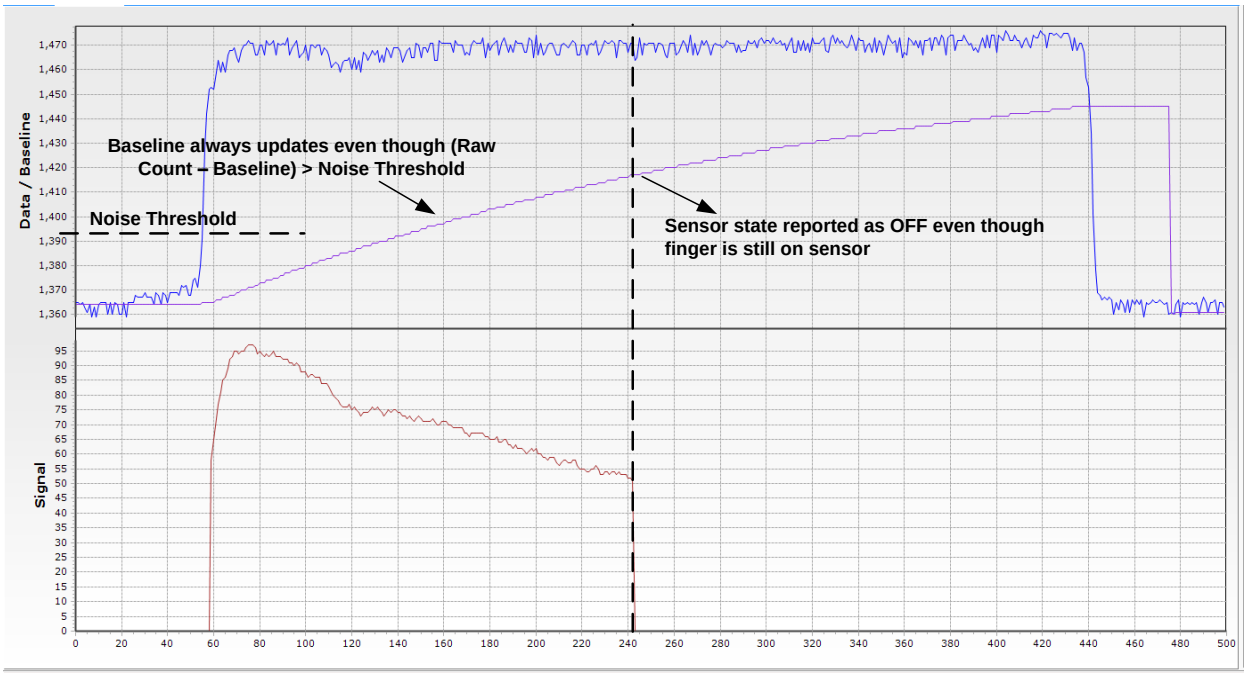
The Debounce parameter is generally set to a value of '3' for reliable sensor status detection. It can be raised or lowered based on the noise aspects of the end user system

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 (*Baseline – Negative Noise Threshold*) to (*Baseline + Noise Threshold*) as Figure 5-13 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 (*Baseline – Negative Noise Threshold*) as Figure 5-16 shows.

Figure 5-16. 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.

5.3.2.3.10 Multi-Frequency Scan

Enabling multi-frequency scan, the CapSense component performs a sensor scan with three different sense clock frequencies and obtains corresponding difference count. The median of the sensor difference-count is selected for further processing. Use this feature for robust operation in the presence of external noise at a certain sensor scan frequency. This option is not available in SmartSense FullAutotune mode. See the code example CE227719 CapSense with Multi-Frequency Scan.

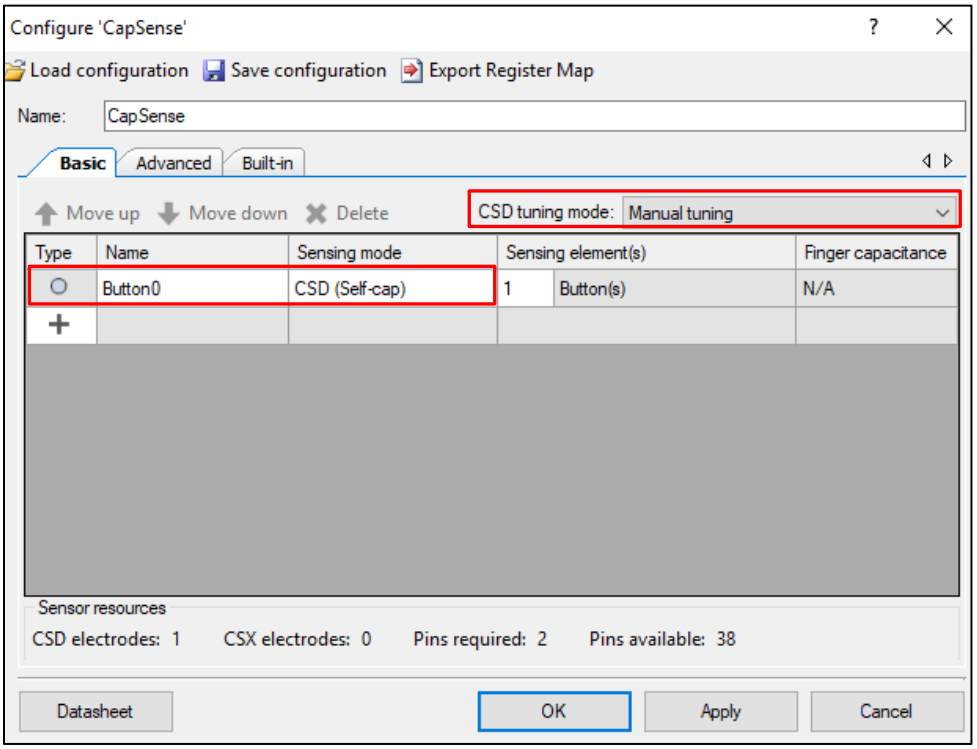
5.3.2.4 Button Widget Example

The following examples explain tuning of self-capacitance based button widgets in a PSoC Creator CapSense Component using the Tuner . For details on the Component and all related parameters, see the Component Datasheet.

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

1. Double-click the **Component** or right-click the **Component** and select **Configure** to open the CapSense Component configuration window.
2. In the **Basic** tab, click the **+** symbol to add the button widget. Select **CSD (Self-cap)** as the **Sensing mode** and **Manual tuning** as the **CSD tuning mode**, as Figure 5-17 shows.

Figure 5-17. 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 5-18 and explained in Table 5-7.

Figure 5-18. CSD Widget Settings

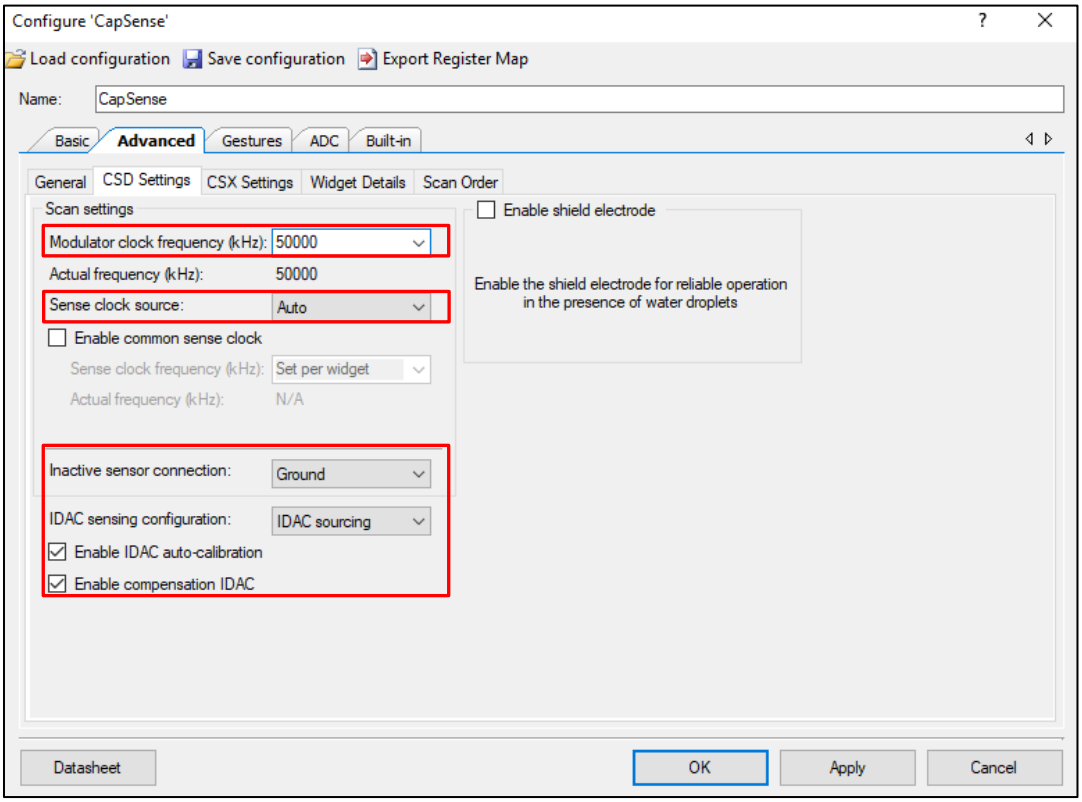


Table 5-7. CapSense Component CSD Configuration Window

Parameter	Value	Rationale
Modulator Clock Frequency	Maximum available option	Higher modulator clock frequency reduces sensor scan time, which results in lower power; hence, it is recommended to use the highest frequency.
Sense Clock Source	Auto	Selecting **Auto** automatically chooses the right Spread Spectrum (SSC) or PRS clock as the sense clock source to deal with EMC/EMI or Flat Spots issues. See Sense Clock **Source** sections for more information on choosing sense clock source.
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 Liquid Tolerance section and AN92239 Proximity Sensing with CapSense.
IDAC Sensing Configuration	IDAC sourcing (default)	IDAC sourcing mode is less susceptible to VDD noise compared to IDAC sinking mode. See GPIO Cell Capacitance to Current Converter. However, if you have clean/noise-free VDD, you may choose IDAC sinking mode for a higher SNR per Equation 3-3.
Enable IDAC Auto-calibration	Enabled	Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration level (85 percent). For systems that may need a different calibration level because of environmental factors, see Modulation and Compensation IDACs.
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 Modulation and Compensation IDACs.
Enable Shield Electrode	Per PCB design	Enable shield if your design requires large proximity sensing distance, Liquid Tolerance, 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 5-19 and Figure 5-20 and explained in Table 5-8 and Table 5-9.

Figure 5-19. CapSense Component Widget Details Window

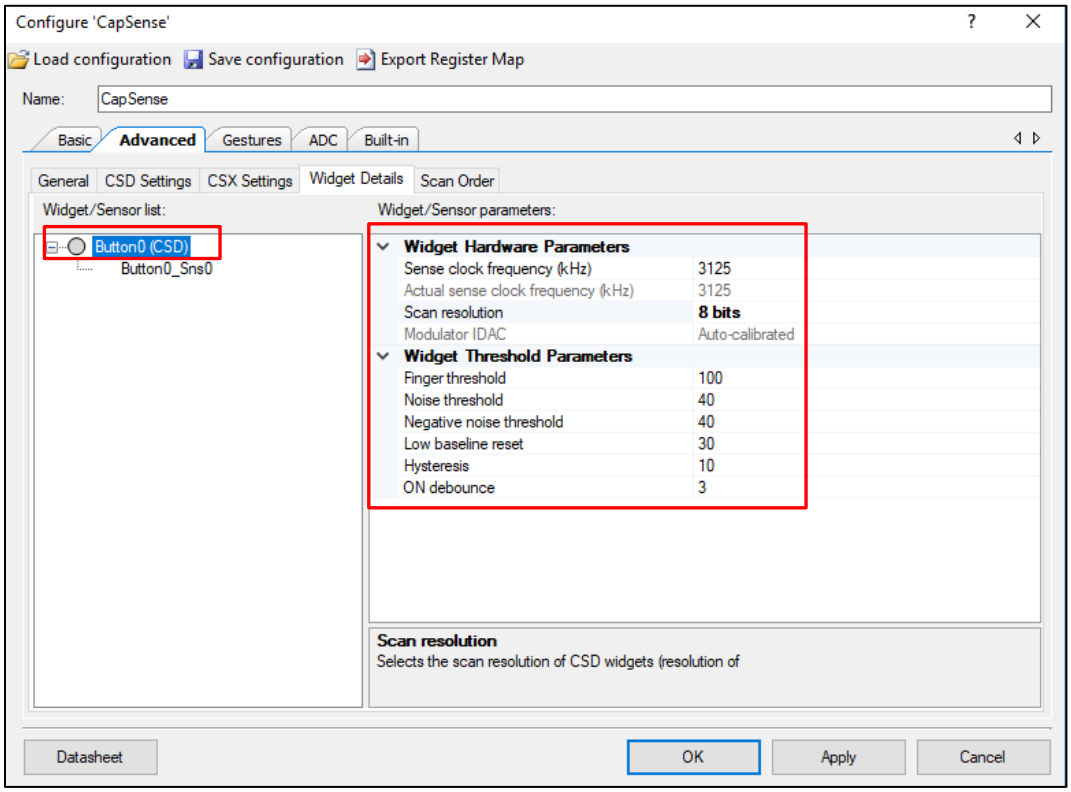


Figure 5-20. CapSense Component Widget Details Window – Sensor Settings

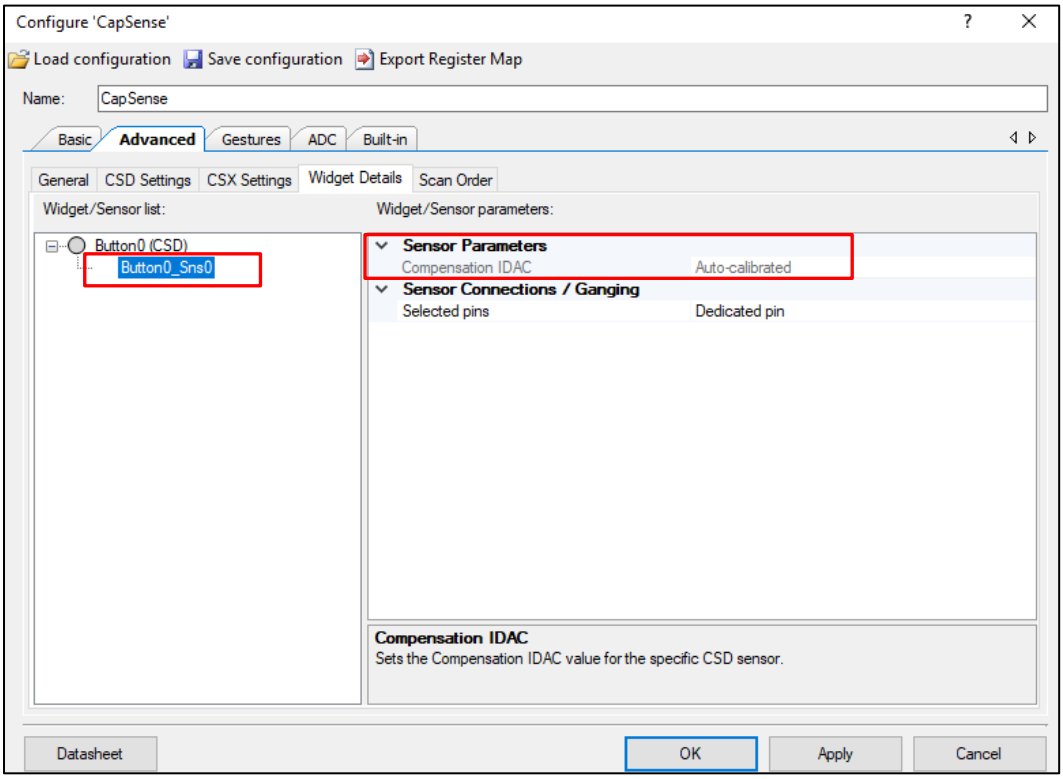


Table 5-8. CapSense Component Widget Details Window

Parameter	Value	Rationale
Sense clock frequency	$\frac{1}{10R_{SeriesTotal}C_P}$	See Sense Clock Parameters to choose the appropriate frequency.
Scan resolution	8-bit	8 bits is a good starting point to ensure fast scan time, and sufficient 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 Modulation and Compensation IDACs.
Finger threshold (FT)	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Noise threshold	Default	Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.
Negative noise threshold	Default	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 Modulation and Compensation IDACs.
Selected Pins	Default	This parameter allows you to gang multiple sensors and scan as a single sensor.

6. 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. See Component Datasheet / Middleware Document the detailed procedure on how to add tuner to your project. Use the SNR measurement tab in the Tuner GUI to calculate the SNR. Based on your end system design, check the signal with a finger that matches the size of your normal use case. Typically finger size targets are ~8 – 9 mm diameter.
7. If the initial SNR is greater than 5, you can move to step 9. Otherwise, move to step 8.
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.

   ***Scan Resolution*:** It 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 3-4. Refer to the section Scan Resolution for more details.

   Filters: It 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 should focus on resolution, as this will increase the sensitivity and signal of your system.

   Note that scan resolution can be updated directly in the Widget/Sensor Parameter in the Tuner GUI , as Figure 5-21 shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as Figure 5-22 shows, and reprogram the device to update filter settings. For details on filters, see the Component Datasheet / Middleware Document.

Figure 5-21. Update Scan Resolution in Tuner GUI

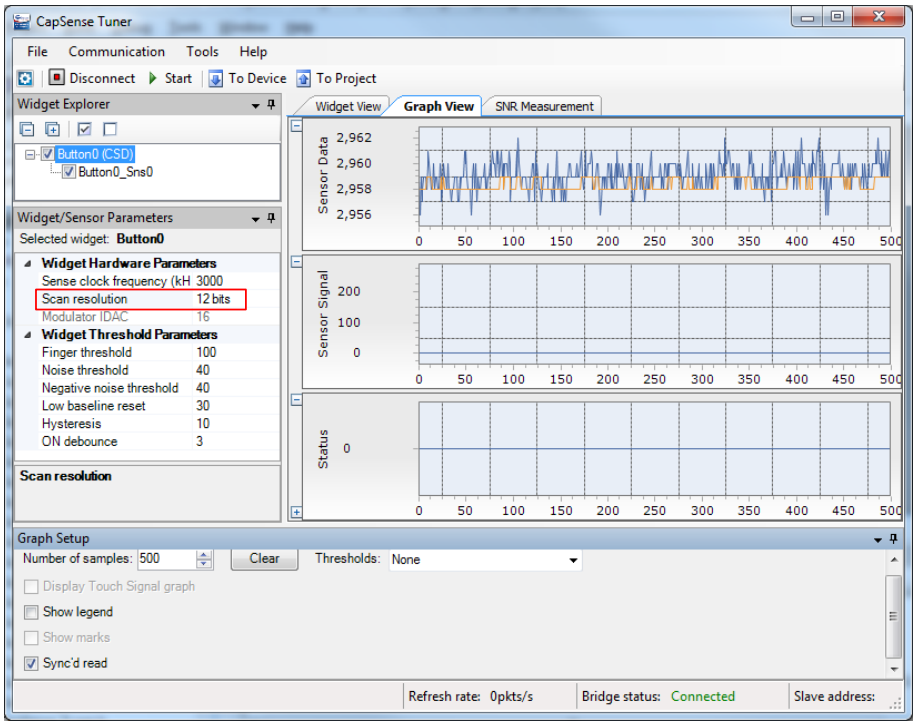
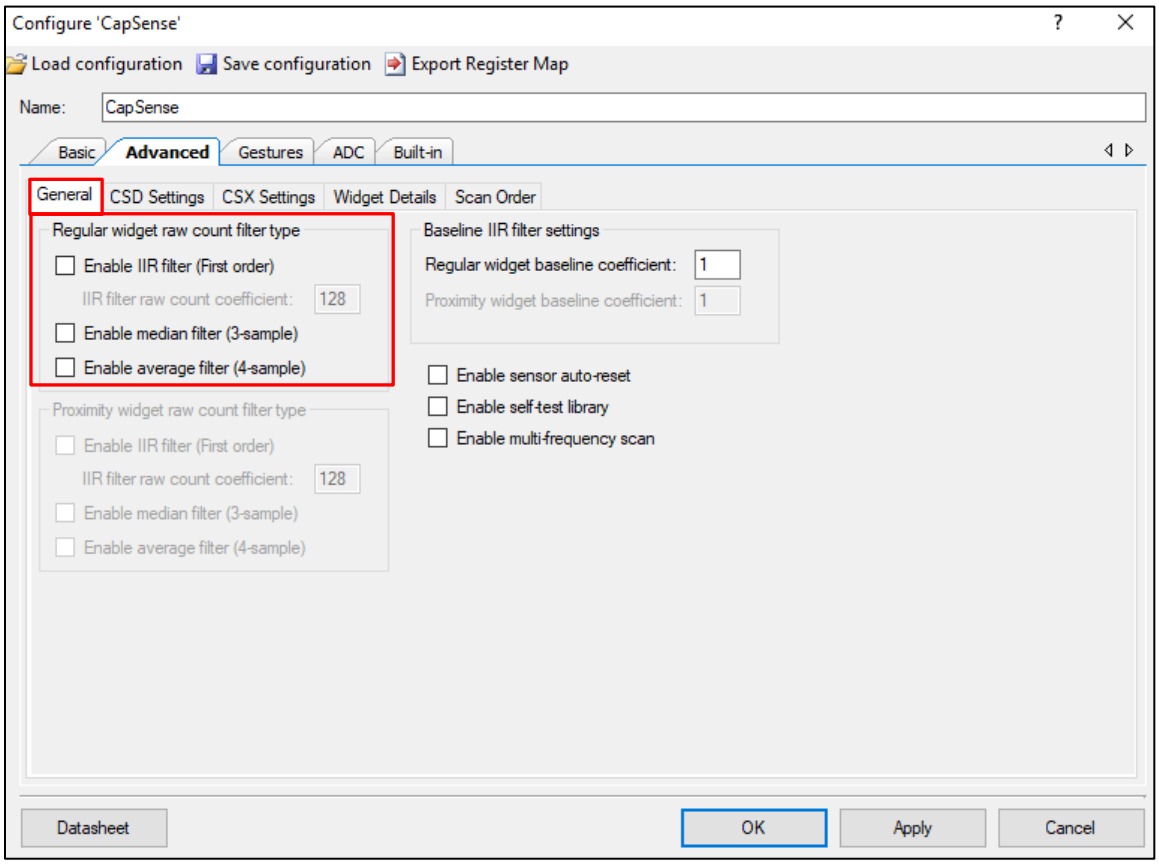
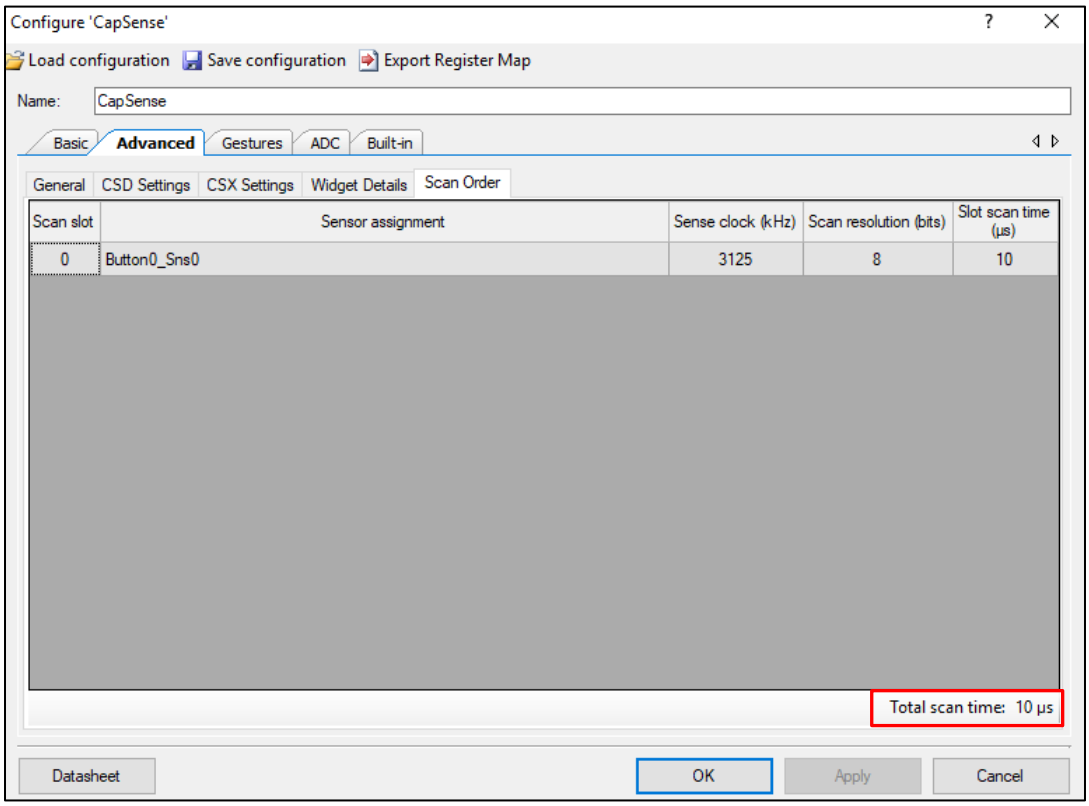


Figure 5-22. Update Filter Settings in CapSense Configurator



9. Check the total scan time (see Figure 5-23) to determine if it meets the system requirements. This timing will impact the response time and is a crucial factor in the overall power consumption of the device in CapSense applications, as indicated in Power Consumption and Response Time.

Figure 5-23. 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 shown in Table 5-6 . 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

    Again, these settings can be first set in the Tuner GUI, as Figure 5-24 shows, or they can be input directly in the CapSense Component customizer, as Figure 5-25 shows.

    For more information on these settings, see Selecting CapSense Software Parameters.

12. If you are not able to achieve an SNR greater than 5 or cannot meet your timing requirements, see Manual Tuning Basics for more information on how to tune your system, see PCB Layout Guidelines or Tuning Debug FAQs like 5.3.5.4, 5.3.5.7, or 5.3.5.10. 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 5-24. Updating Threshold Parameters in Tuner GUI

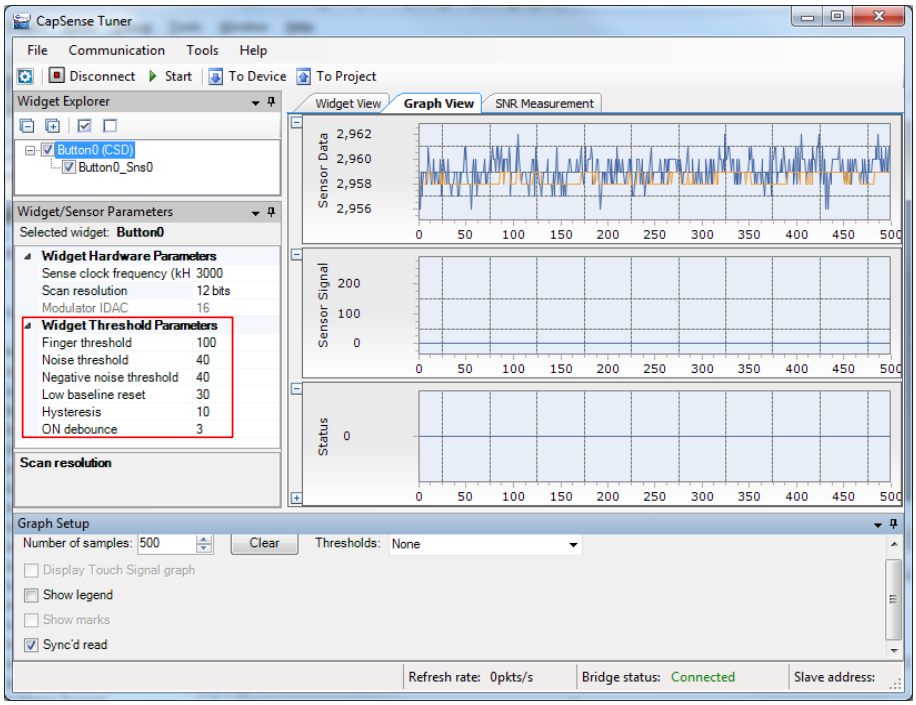


Figure 5-25. Updating Threshold Parameters in CapSense Configurator

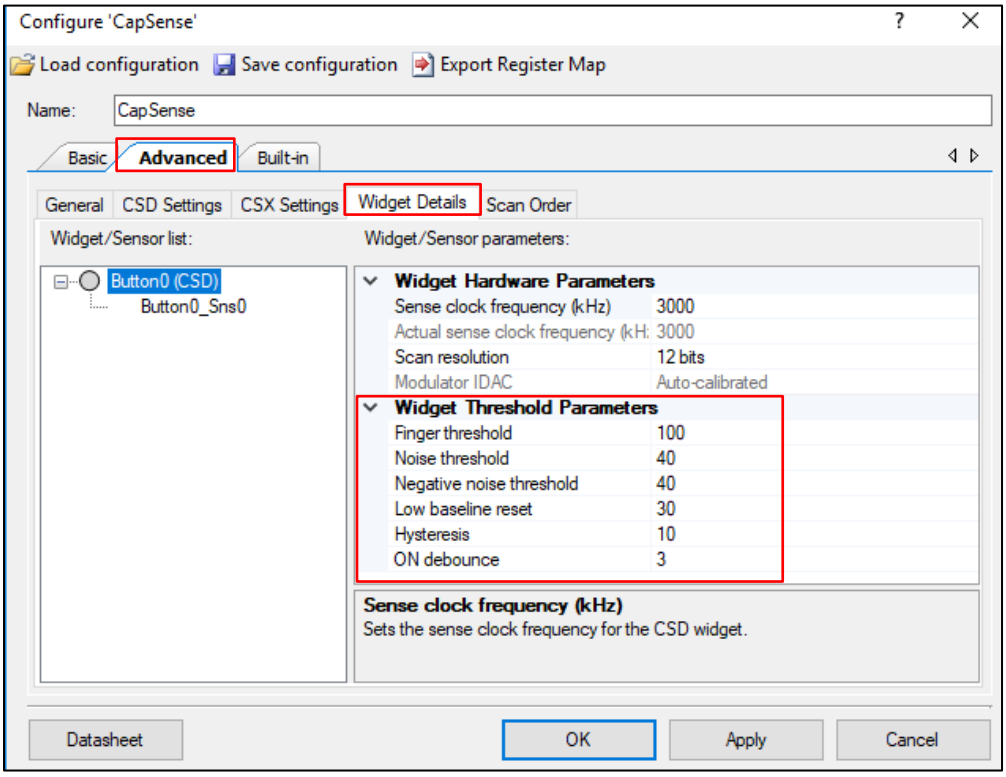
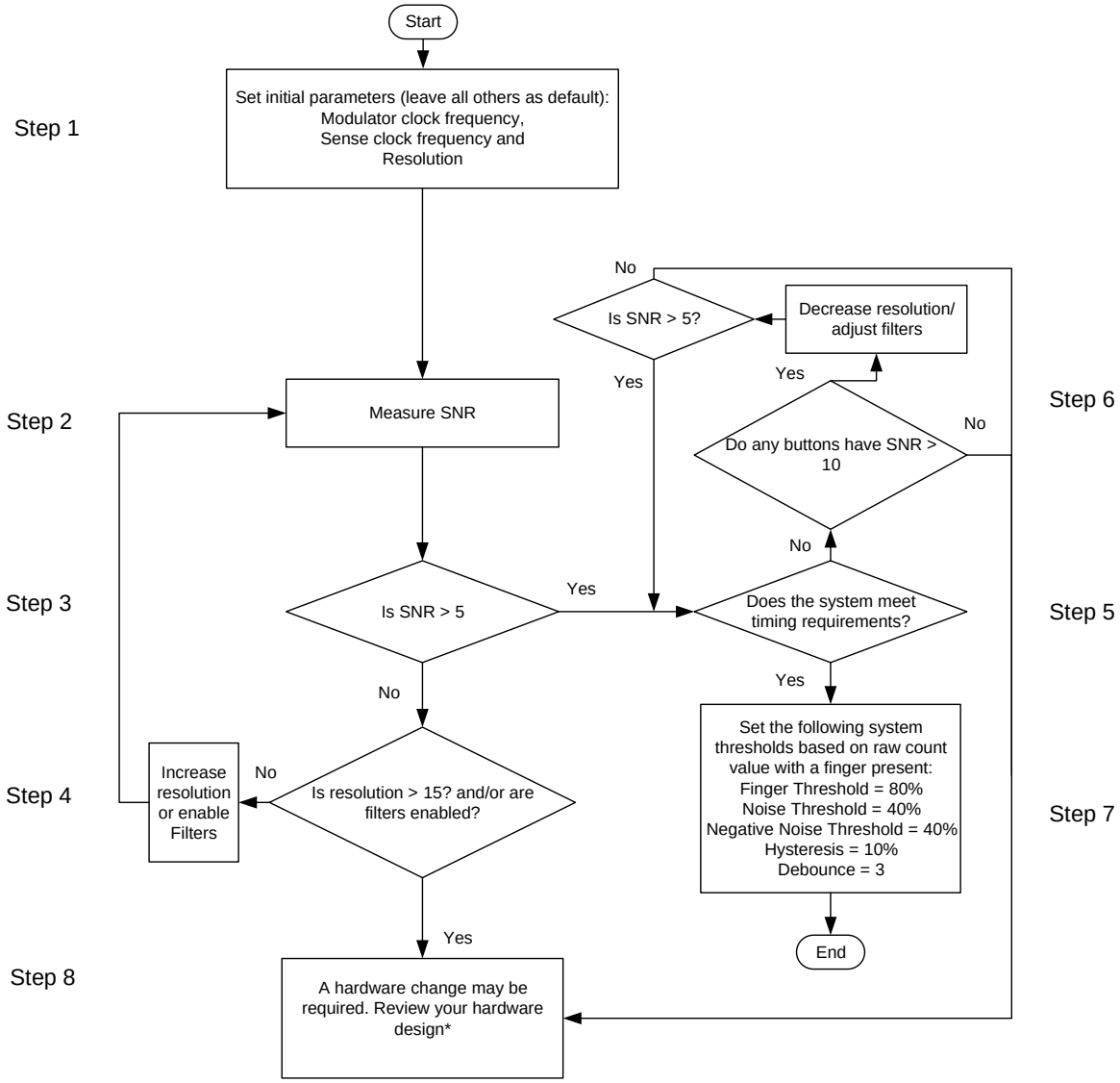


Figure 5-26. CSD Button Widget Tuning Flowchart



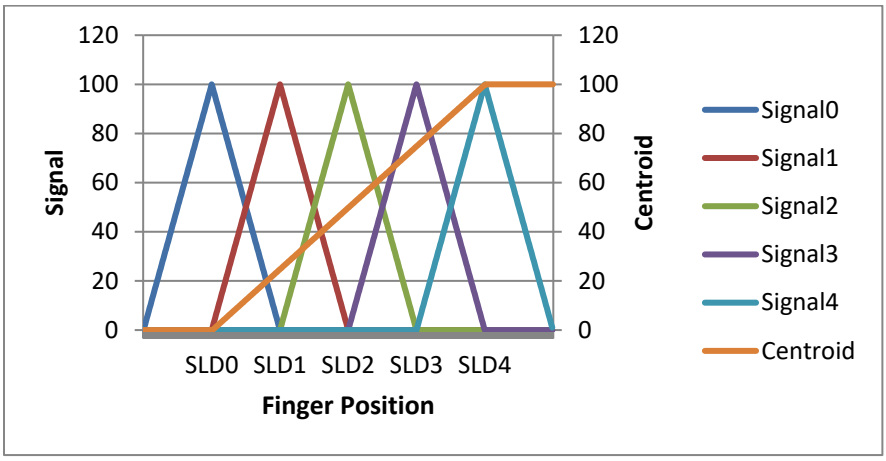
* To review the hardware design, see the Sensor Construction and PCB Layout Guidelines sections in the Design Considerations chapter. Also, see the Tuning Debug FAQs section for guidelines on advanced debug.

5.3.2.5 Slider Widget Example

5.3.2.5.1 Slider Tuning Basics

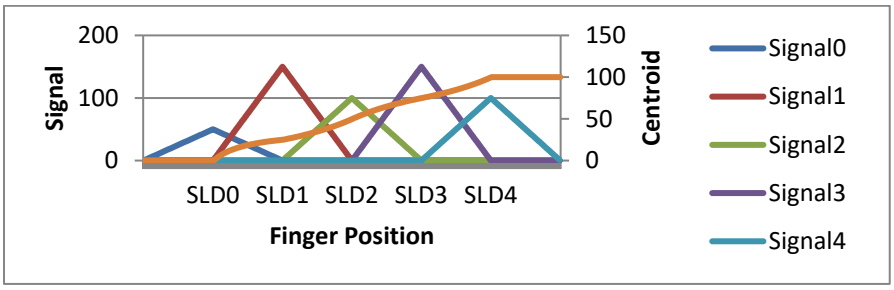
A slider has many segments, each of which is connected to the CapSense input pins of the PSoC device. Unlike the simple on/off operation of a button widget sensor, slider widget sensors work together to track the location of a finger or other conductive object. Because of this, the slider layout design 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. See Slider Design for details on slider layout design guidelines to avoid nonlinearity in the centroid, ensure that the signal from all the slider segments is equal, as Figure 5-27 shows, 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 5-28 shows. Note that in PSoC Creator and in ModusToolbox, a centroid of 0xFFFF and 0x0000 is reported respectively when a finger is not detected on the slider, or when none of the slider segments report a difference count value greater than the Finger Threshold parameter.

Figure 5-27. Response of Centroid Versus Finger Location when Signals of All Slider Elements Are Equal



Note Signal = Raw Count - Baseline

Figure 5-28. Response of Centroid Versus Finger Location when the Signal of All Slider Elements Are Different



5.3.2.5.2 Slider Tuning Guidelines

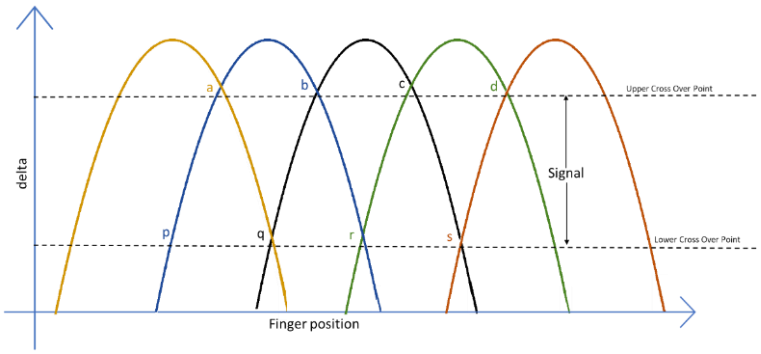
Use the following steps to manually tune the slider segments to achieve an equal signal for all slider segments:

1. Use the Built in Self Test (BIST) APIs to get the C_P values of the slider segments and then identify the segment with maximum C_P. You can also use an LCR meter to measure the C_P of the slider segments. Refer to the Component Datasheet / Middleware Document for more details.
2. Follow the manual tuning procedure mentioned in the Manually Tuning Hardware Parameters section to tune the following parameters for the slider segment which has the maximum C_P.
   a. Sense clock source
   b. Sense clock frequency
   c. Modulator clock frequency
   d. Modulation IDAC (If auto-calibration is disabled)
   e. Scan Resolution

   While tuning, ensure to have Compensation IDAC enabled and IDAC auto-calibration enabled. This will help in achieving required IDAC calibration level for other segments in the widget while maintaining their sensitivity to be same as the highest C_P segment. See section 5.3.2.1.1 for more details.
3. The parameters tuned in Step 1 are widget-level parameters that they need not be set individually for each segment. Set the obtained parameters in the Component for the slider widget.
4. Do the following to capture the difference count of all the segments for setting the software parameters for the slider widget.
   a. Disable filters.
   b. Capture the peak-to-peak noise of all the segments of the sliders and note the peak-to-peak noise for each segment.

a) Get the graphs of sensor signal vs finger position by swiping the finger on the slider from beginning to the end of the slider with slow, constant speed. Do the following to get the graph in Tuner GUI. These steps are only to get the graph and the software parameters and scan resolution will be determined using this data.
   a. Set the Noise threshold (NT) and Finger Threshold (FT) to lowest value allowed by the CapSense configurator. It is recommended to set to NT = 5, FT = 5. This reduces the influence of baseline on the sensor signal which helps to get the true difference count.
   b. Use the metal finger (grounded) typically 8 mm or 9 mm and swipe it slowly at a constant speed from start to end of the slider. Now you will get a graph similar to Figure 5-29 in the Sensor signal graph in the Graph view tab of Tuner GUI.

Figure 5-29. Difference count (Delta) Vs Finger position



All the slider segments are expected to have the same sensitivity if the slider layout is designed per the guidelines in Slider Design. That means all the segments will give the same change in raw count for a finger touch. This is also observed in the graph as shown in Figure 5-29. From Figure 5-29, it is also obvious that

- Sensor signal value at points a, b, c, d are expected to be at approximately the same level
- Sensor signal value at points p, q, r, s are expected to be at approximately the same level

5. Get the Upper and lower crossover points as shown in Figure 5-29. Ensure that all the upper crossover points meet at least 5:1 SNR and all the lower cross over points are greater than twice the peak-to-peak noise for all the slider segments. You can increase the scan resolution to achieve this requirement.

Equation 5-13. SNR Calculation for Slider Segments

$$SNR_N = \frac{Sensor\ signal_N}{Pk - Pk\ noise_N}$$

6. If the condition in Step 5 is not achieved even with the highest resolution, enable filters. See Table 5-2 to understand how to use filters.
7. Set the following recommended thresholds values for the slider widget as listed in Table 5-10. See Selecting CapSense Software Parameters for more details.

Table 5-10. Recommended Values for Threshold Parameters

Sl. No.	CapSense Threshold Parameter	Recommended Value
1.	Finger Threshold	80 percent of signal
2.	Noise Threshold	Signal at lower cross over point
3.	Hysteresis	10 percent of signal
4.	ON Debounce	3
6.	Low Baseline Reset	30
7.	Negative Noise Threshold	Same as noise threshold

5.3.2.6 Proximity Widget Example

For tuning a proximity sensor, see AN92239 - Proximity Sensing with CapSense.

5.3.3 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.

5.3.3.1 Basics

5.3.3.1.1 Conversion Gain and CapSense Signal

In a mutual-capacitance sensing system, the $Rawcount_{counter}$ is directly proportional to the mutual capacitance between the Tx and Rx electrodes, as Equation 5-14 shows.

Equation 5-14. Raw Count Relationship to Sensor Capacitance

$$Rawcount_{Counter} = G_{CSX} C_M$$

Where G_{CSX} is the capacitance to digital conversion gain of CapSense CSX and C_M is the mutual capacitance between the Tx and Rx electrodes. Figure 5-31 shows the relationship between raw count and mutual capacitance of the CSX sensor. The tunable parameters of the conversion gain in Equation 5-15 are F_{TX}, N_{Sub}, F_{MOD} and IDAC.

The approximate value of this conversion gain is:

Equation 5-15. Capacitance to Digital Converter Gain

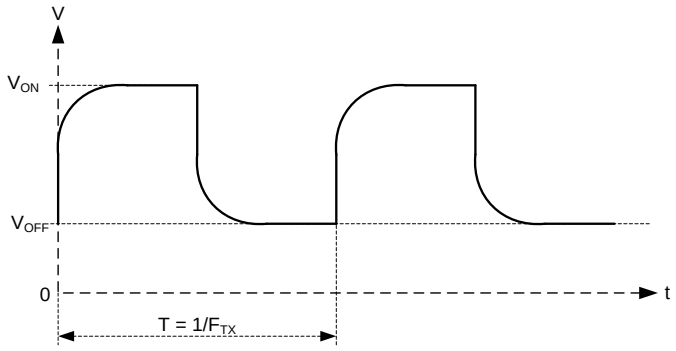
$$G_{CSX} = \frac{2\, V_{TX}\, F_{TX}\, MaxCount}{IDAC}$$

Equation 5-16. MaxCount Equation

$$MaxCount = \frac{F_{Mod}\, N_{Sub}}{F_{TX}}$$

Where, V_{TX} is the voltage at the Tx node of the sensor as shown in Figure 5-30, $V_{TX} = V_{ON} - V_{OFF}$. The value of V_{TX} is always V_{DDIO} or V_{DDD} (if V_{DDIO} is not available) if the Tx clock frequency can completely charge and discharge the Tx electrode. F_{TX} is the Tx clock frequency, I_{DAC} is the current drawn for charging and discharging the C_{INT} capacitors, and N_{Sub} is the number of sub-conversions.

Figure 5-30. Voltage at Tx Node of the CSX Sensor

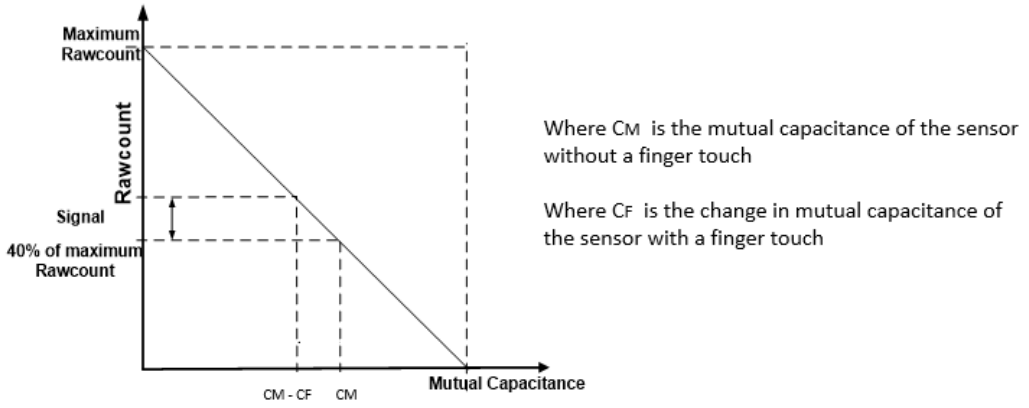


Note that the raw count observed from the Component is given by Equation 5-17. See CapSense CSX Sensing Method for more details on $Rawcount_{component}$.

Equation 5-17. $Rawcount_{Component}$

$$Rawcount_{Component} = MaxCount - Rawcount_{Counter}$$

Figure 5-31. Raw Count vs Sensor Mutual Capacitance



5.3.3.2 Selecting CapSense Hardware Parameters

CapSense hardware parameters govern the conversion gain and CapSense signal. Table 5-11 lists the CapSense hardware parameters that apply to the CSX sensing method. Table 5-11 also shows the mapping of each parameter in the PSoC Creator CapSense Component to the one in the ModusToolbox middleware. For simplicity of documentation, this design guide shows selecting the CapSense parameter using the CapSense Configurator in PSoC Creator. The same procedure could be followed in configuring CapSense in ModusToolbox. However, in ModusToolbox, you set the Tx clock and Modulator clock using divider values. On the other hand, in PSoC Creator, you specify the frequency value directly in the configurator. See Component Datasheet / Middleware Document.

Table 5-11. CapSense Component Hardware Parameters

S. No	CapSense Parameter in PSoC Creator	CapSense Parameter in ModusToolbox
1	Modulator Clock Frequency	Modulator Clock Divider
2	Tx Clock Source	Tx Clock Source
3	Tx Clock Frequency	Tx Clock Divider
4	IDAC	IDAC
5	Number of Sub-Conversions	Number of Sub-Conversions

5.3.3.2.1 Tx Clock Parameters

There are two parameters that are related to the Tx clock: Sense clock source and Sense clock frequency.

5.3.3.2.1.1 Tx Clock Source

Select “Auto” to let the Component automatically choose the best Tx clock source between Direct and Spread spectrum clock (SSCx) for each widget. If not selecting Auto, select the clock source based on the following:

- Direct – Clock signal with a fixed clock frequency.

  Use this option for most cases.

- Spread spectrum clock (SSCx) – If you chose this option, the Tx clock signal frequency is dynamically spread over a predetermined range. Use this option for reduced EMI interference and avoiding Flat Spots.

  However, when selecting SSCx clock, you need to select the Tx clock frequency, Modulator clock frequency, and number of sub conversion such that the conditions mentioned in Component Datasheet / ModusToolbox CapSense Configurator Guide for SSCx clock source selection are satisfied.

5.3.3.2.1.2 Tx Clock Frequency

The Tx clock frequency determines the duration of each sub-conversion as explained in the CapSense CSX Sensing Method section. The Tx clock signal must completely charge and discharge the sensor parasitic capacitance; it can be verified by checking the signal in an oscilloscope, or it can be set using the Equation 5-18. In addition, you should ensure that the auto-calibrated IDAC code lies in the mid-range (for example, 30-90) for the selected F_{TX}. If the auto-

calibrated IDAC code lies out of the recommended range, tune F_{TX} such that it IDAC falls in the recommended range and satisfies Equation 5-18.

Equation 5-18. Condition for Selecting Tx Clock Frequency

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

To minimize the scan time, as Equation 5-19 shows, it is recommended to use the maximum Tx clock frequency available in the component drop-down list that satisfies this criteria.

Equation 5-19. Scan Time of CSX Sensor

$$T_{CSX} = \frac{NoC}{F_{TX}}$$

Where NoC is Number of Sub-Conversions.

Additionally, if you are using the SSCx clock source, ensure that you select the Tx clock frequency that meets the conditions mentioned in Component Datasheet / Middleware Document / ModusToolbox CapSense Configurator Guide in addition to these conditions.

The maximum value of F_{TX} depends on the selected device. For the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices, the maximum F_{TX} is 3000 kHz and for other devices it is 300 kHz.

5.3.3.2.2 Modulator Clock Frequency

It is best to choose the highest allowed clock frequency for the given device because a higher modulator clock frequency leads to a higher sensitivity/signal, increased accuracy, and lower noise for a given C_M to digital count conversion as Equation 5-15, Equation 5-16 indicates. Also, a higher value of F_{mod}/F_{tx} ensures lower width of Flat Spots in C_M to raw count conversion.

5.3.3.2.3 IDAC

It is recommended to enable IDAC auto-calibration. It is best to avoid very high and very low IDAC codes. The recommended IDAC code range is between 30-90. If the IDAC values are away from the recommended range, tune the Tx clock frequency to adjust the IDAC level. If the IDAC is failing to calibrate properly, it may be due to low Cm in the design. Refer to the section I am observing a low Cm for my CSX Button for mitigating impact of low Cm in the design.

5.3.3.2.4 Number of Sub-Conversions

The number of sub-conversions decides the sensitivity of the sensor and sensor scan time. From Equation 3-10 for a fixed modulator clock and Tx clock, increasing the number of sub-conversions (N_{Sub}) increases the signal and SNR . However, increasing the number of sub-conversions also increases the scan time of the sensor per Equation 5-20.

Equation 5-20 CSX Scan time

$$Scan\ time = \frac{N_{Sub}}{F_{TX}}$$

Initially, set the value to a low number (for example, 20), and use the Tuner GUI to find the SNR of the sensor. If the SNR is not > 5:1 with the selected N_{Sub}, try to increase the N_{Sub} in steps such that the SNR requirement is met.

5.3.3.3 Selecting CapSense Software Parameters

CapSense software parameters for mutual capacitance are the same as that for self-capacitance; therefore, these parameters could be selected as mentioned in the section Selecting CapSense Software Parameters.

5.3.3.4 Button Widget Example

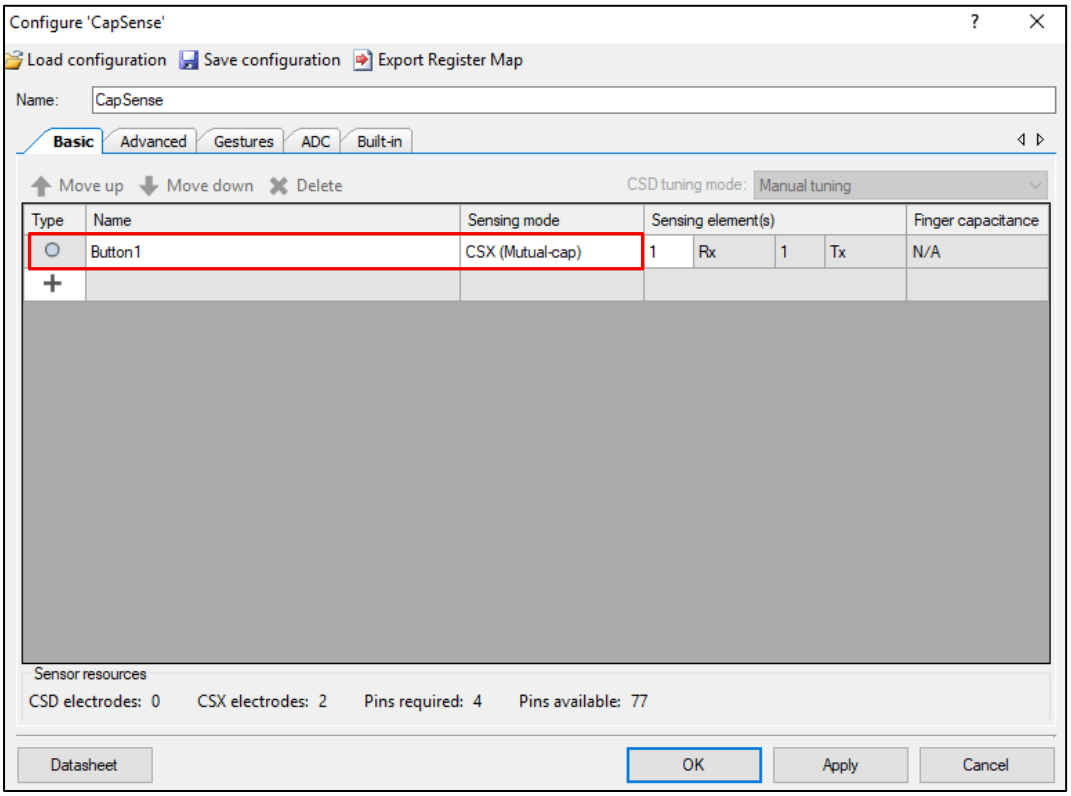
This example explains the tuning of mutual-capacitance-based button widget in a PSoC Creator CapSense Component using Tuner GUI. The same procedure can be used in the ModusToolbox for CapSense configuration. See the respective Component Datasheet / Middleware Document.

Do the following to manually set all the tuning parameters. See Figure 5-39 for a quick reference flow chart.

1. Double-click the Component or right-click the **Component** and select **Configure** to open the CapSense Component configuration window.

2. In the **Basic** tab, click the **+** symbol to add the button widget. Select **CSX (Mutual-cap)** as the **Sensing mode**, as Figure 5-32 shows.

Figure 5-32. 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 5-33.
5. In the **Advanced** tab **- Widget Details** window, specify the parameter settings as shown in Figure 5-34.

Figure 5-33. CSX Widget Settings

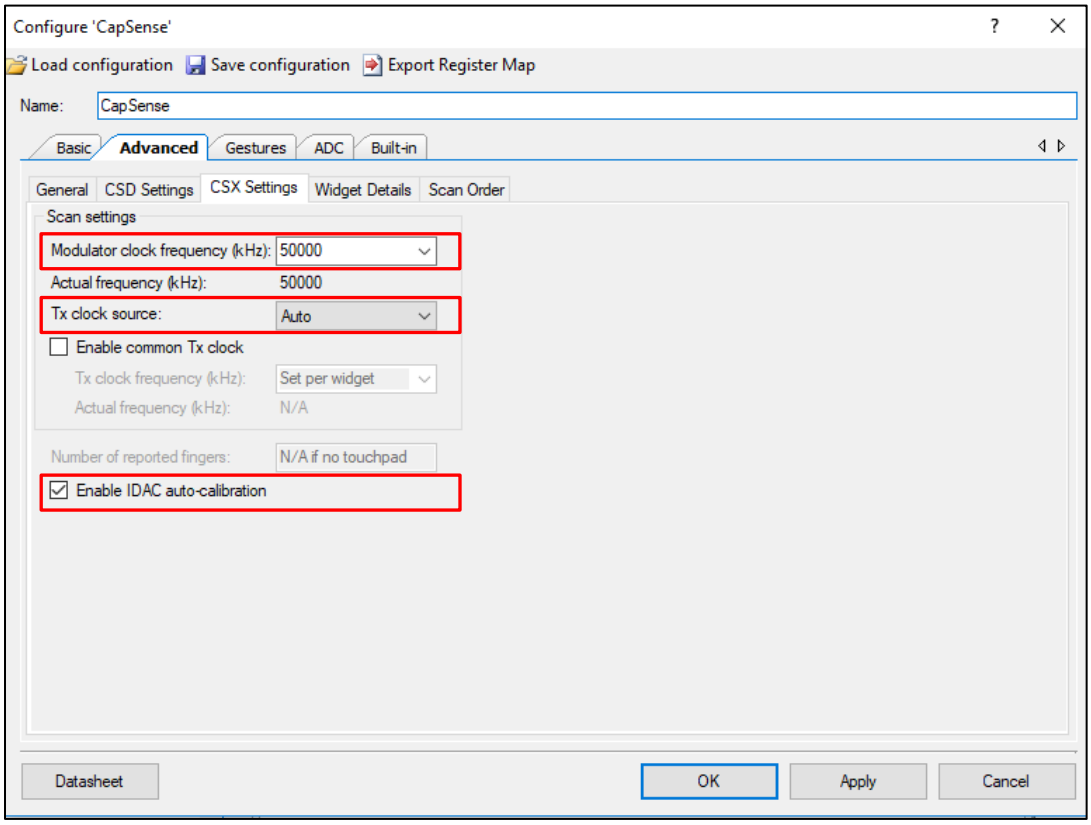


Table 5-12. CapSense Component General Configuration Window

Parameter	Value	Rationale
Modulator clock frequency	Maximum available option	A higher modulator clock frequency reduces flat spots and increases sensitivity. Thus, it is recommended to select the highest possible available modulator clock frequency.
Tx clock source	Auto	Enabling Auto lets the Component automatically chose between the available clock sources. See the Tx Clock Source section for more information.
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 max count).

Figure 5-34. CapSense Component Widget Details Window

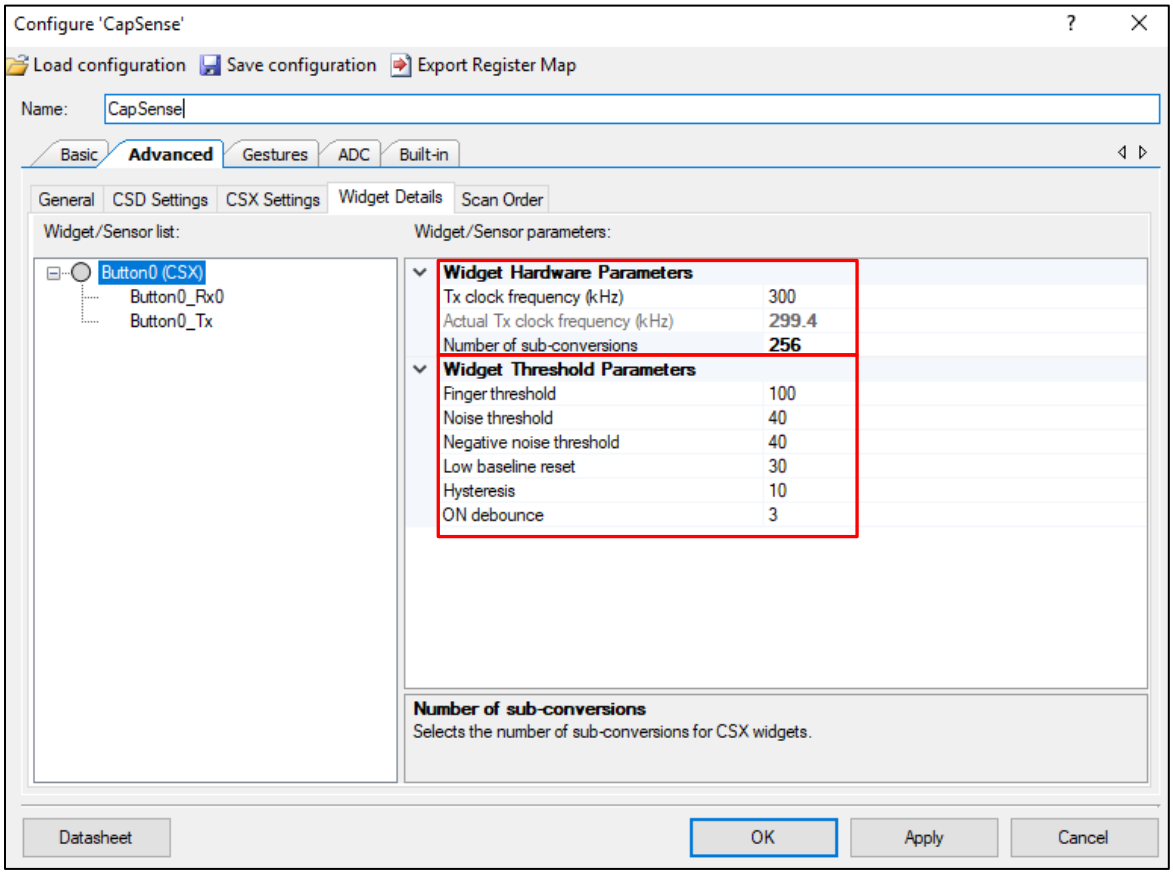


Table 5-13. CapSense Component Widget Details Window

Parameter	Value	Rationale
Tx clock frequency	$\frac{1}{10R_{SeriesTx}C_P}$	It is recommended to set the highest clock frequency that satisfies this criterion. See Tx Clock Frequency.
Number of sub-conversions	A low value like 20	It is good to start with a low number. This value will be adjusted as needed in Step 8. See Number of Sub-Conversions.
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.
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.

6. Next, observe raw counts in the Graph View tab in Tuner GUI and use the SNR measurement tab to calculate the SNR. 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.
7. If the initial SNR is greater than 5, you can move to step 9. Otherwise, move to step 8.
8. When the SNR is less than 5, use the number of sub-conversions or filters to increase the SNR to >5:1.

   ***Number of sub-conversions*:** It can be increased to increase signal at a disproportionate rate to noise to improve overall SNR. Note that increasing number of sub-conversions also increases the scan time per Equation 5-19.

***Filters*:** It 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. Select an appropriate filter for your application based on Table 5-2.

It is best to find a balance between the number of sub-conversions and filters. Refer section Manual Tuning Trade-offs to choose one over the other. In general, 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), focus on number of sub-conversions, as this will increase the sensitivity and signal of your system.

Note that number of sub-conversions can be updated directly in the Widget/Sensor Parameters tab of the Tuner GUI, as Figure 5-35 shows, but to adjust the filter settings you will need to open up the CapSense configurator and select the appropriate filter as Figure 5-36 shows, and reprogram the device to update filter settings.

Figure 5-35. Update Number of Sub-Conversions in Tuner GUI

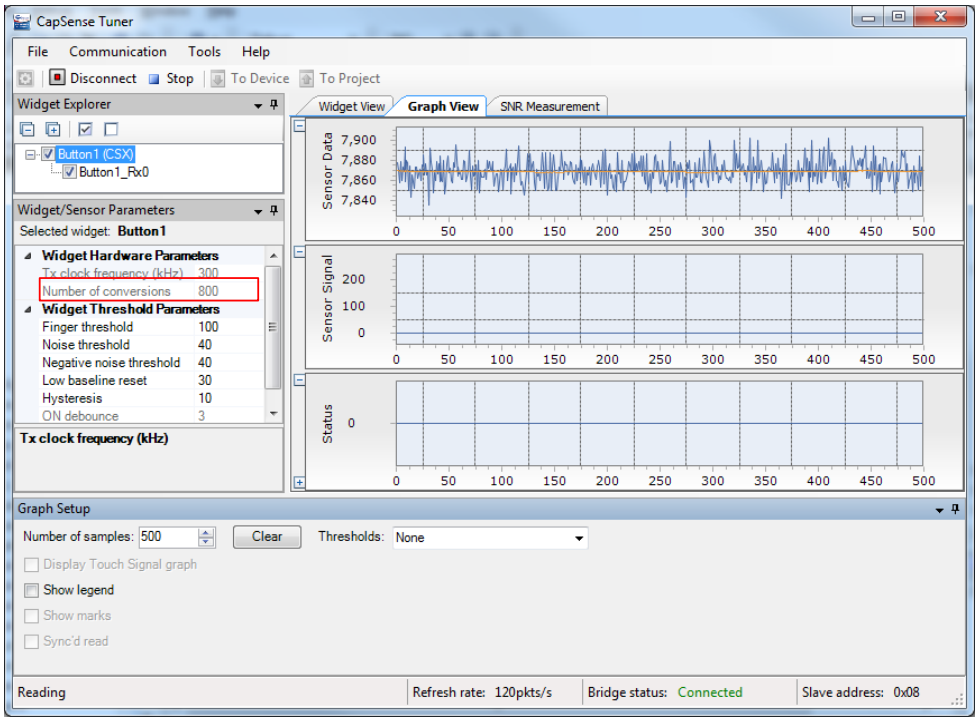
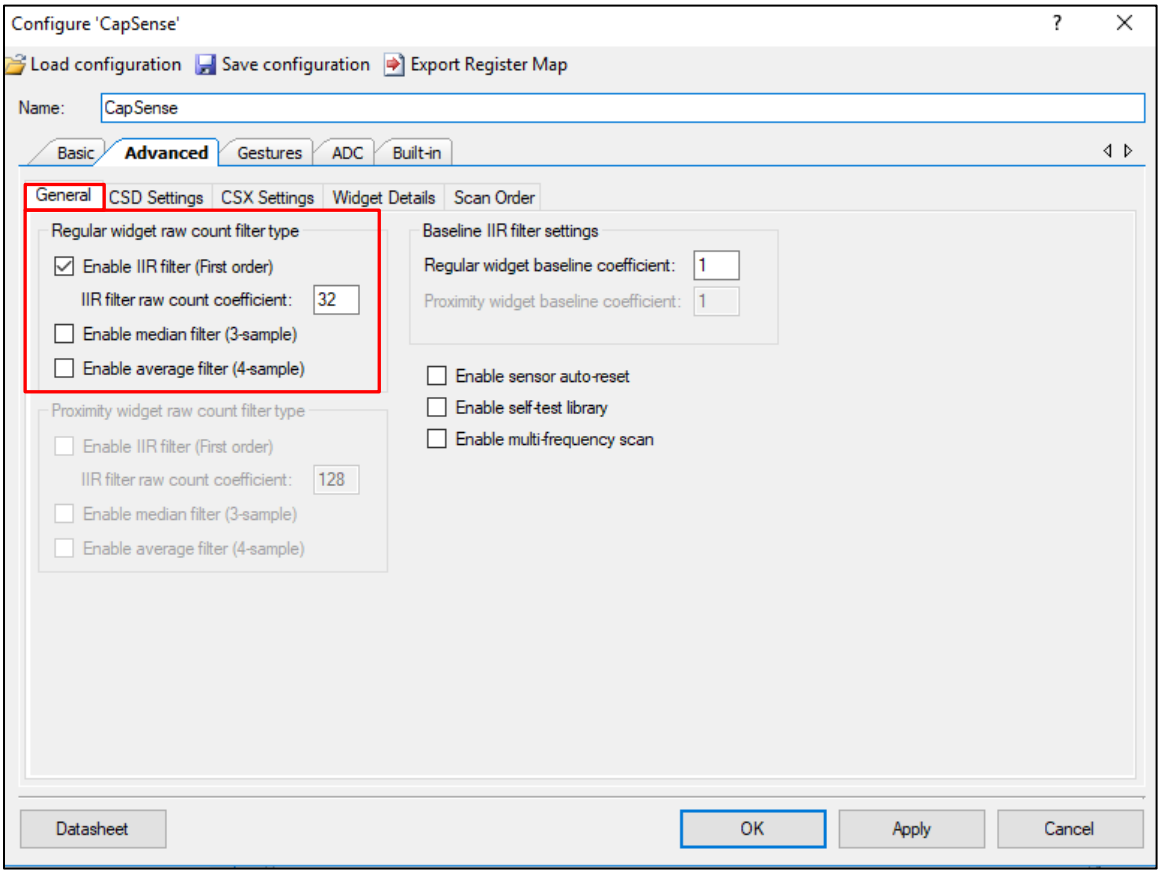


Figure 5-36. Update Filter Settings



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 sub-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 the threshold parameters per Table 5-6. See Selecting CapSense Software Parameters. 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.

    These settings can be first set in the Tuner GUI, as Figure 5-37 shows, and saved by clicking on **Apply to Project** in the **File** menu of Tuner GUI; or they can be input directly in the CapSense Component configurator as Figure 5-38 shows.

Figure 5-37. Updating Threshold Parameters in Tuner GUI

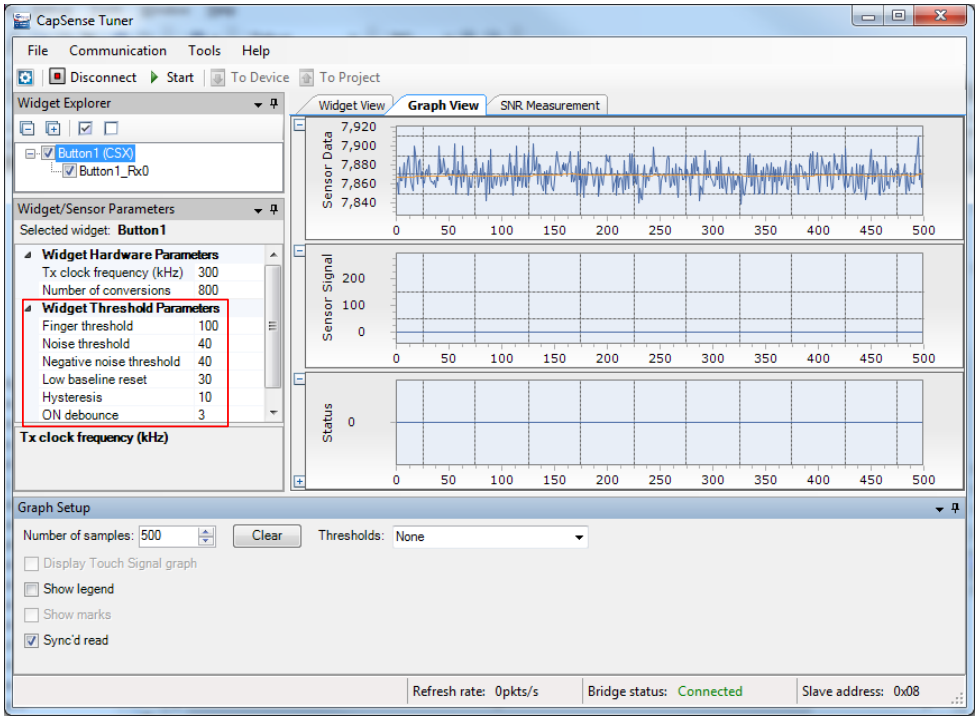


Figure 5-38 Updating Threshold Parameters in Configure ‘CapSense_CSD_P4’ Dialog

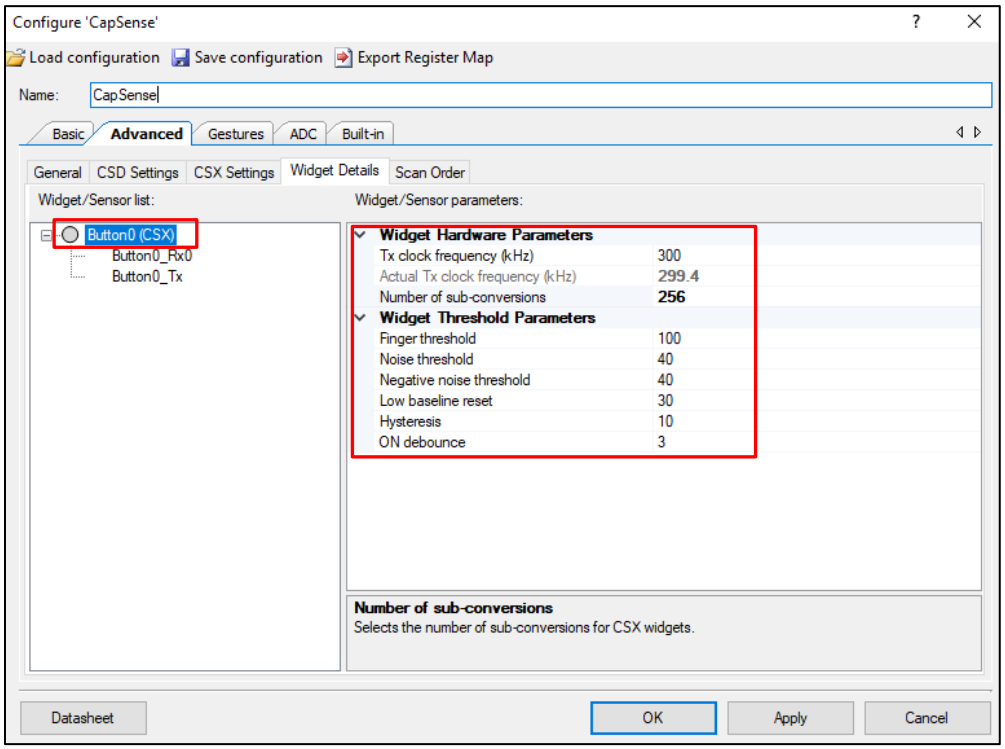
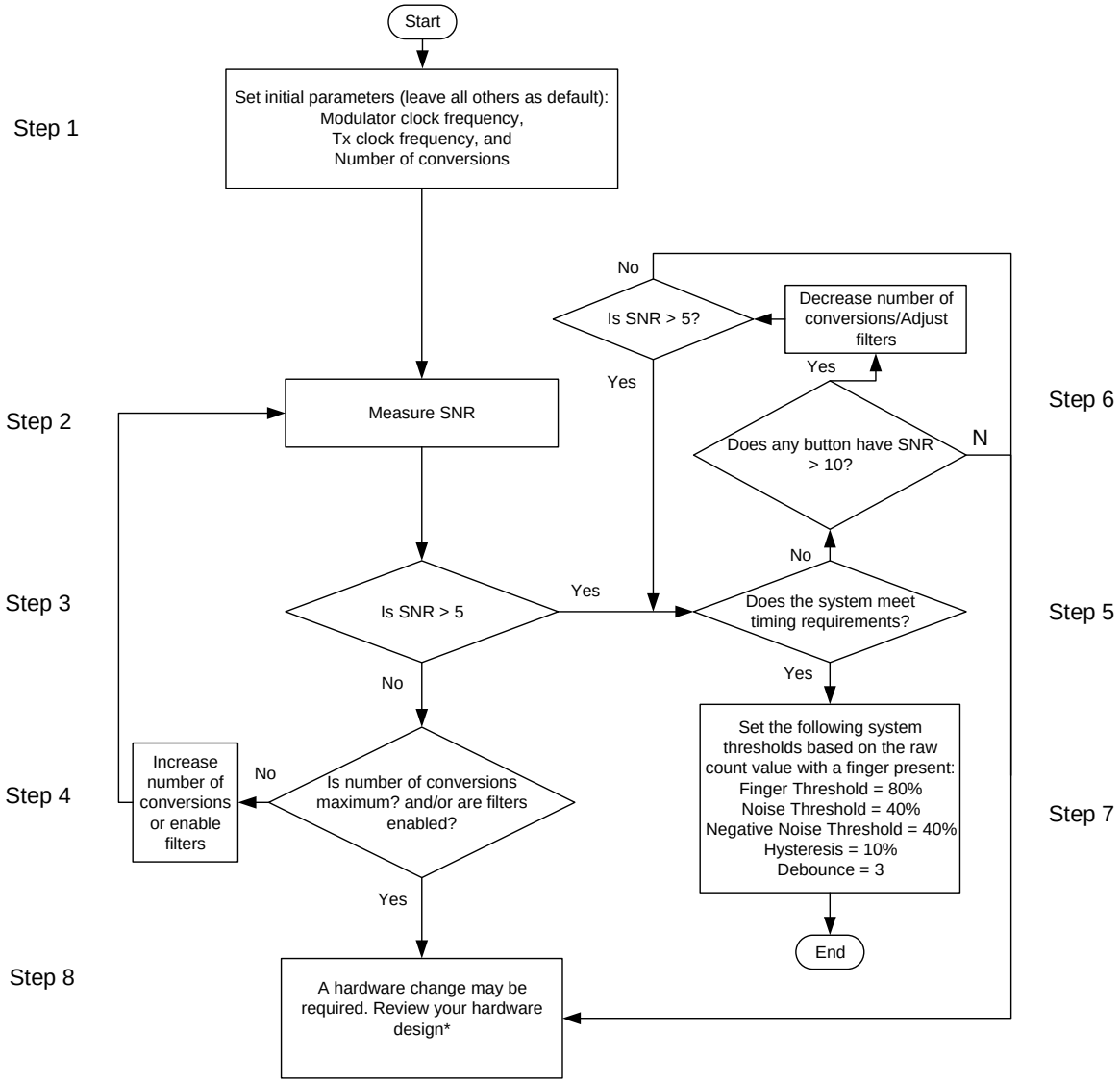


Figure 5-39. CSX Button Widget Tuning Example



* To review the hardware design, see the Sensor Construction and PCB Layout Guidelines sections in the Design Considerations chapter. Also, see the Tuning Debug FAQs section for guidelines on advanced debug.

5.3.4 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

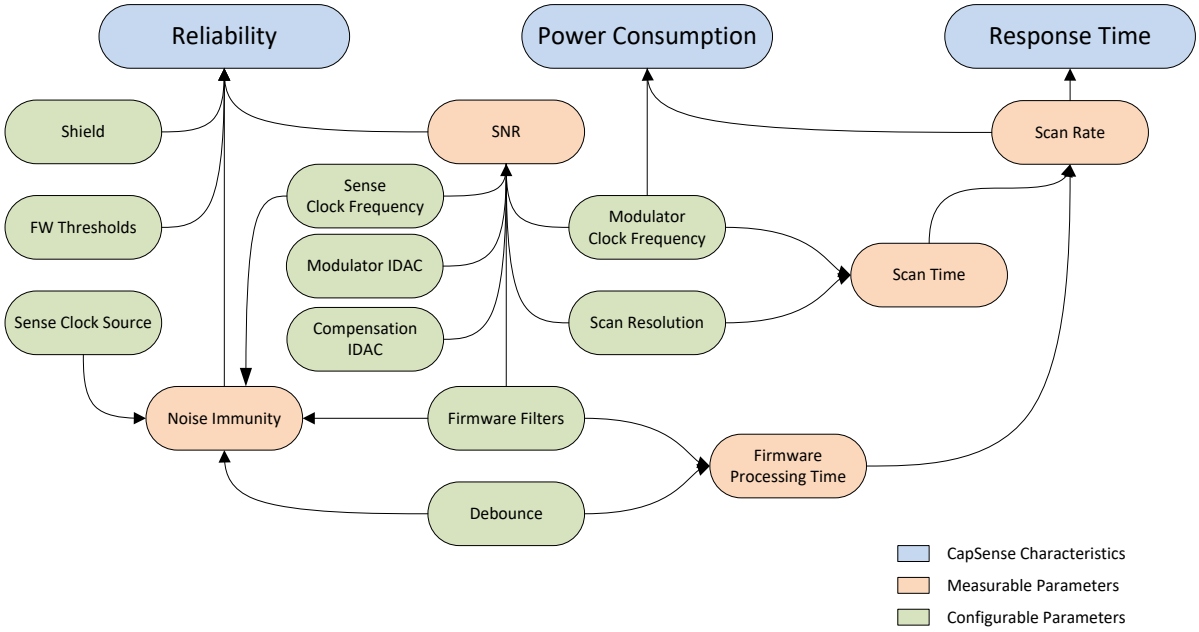
sensors, it is important to properly define and follow a timing budget. A good target for total response time is below 100 ms.

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 5-40. CapSense Parameter Relationships



5.3.4.1 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.

5.3.4.2 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 depending on the CapSense component version you are using). Both affect scan time by adding to the processing time of a sensor and 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 the Overview, Selecting CapSense Hardware Parameters, and Selecting CapSense Software Parameters sections to achieve optimal manual tuning for your design.

5.3.5 Tuning Debug FAQs

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

5.3.5.1 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. See 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 another required tuner code.

Solution 4: Add the tuner code listed in CapSense Component datasheet to your *main.c* and reprogram the device.

5.3.5.2 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 UART/EZI2C configurator dialog and Tuner Communication Setup dialog match. Make sure that the sub-address size is equal.

5.3.5.3 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 GUI section and add the tuner code to your *main.c* and reprogram the device.

5.3.5.4 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 GUI section of this 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 the supported range of parasitic capacitance by the PSoC device, review your board layout to reduce C_P further, by following the PCB Layout Guidelines, and/or contact Cypress Technical Support to review your layout. See Component Datasheet / Middleware Document for more details on the supported range of C_P.

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 max raw count for a given sensor according to the Modulation and Compensation IDACs section.

5.3.5.5 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 Electromagnetic Compatibility (EMC) Considerations. If it is already enabled, see 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.

5.3.5.6 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.

5.3.5.7 I am unable to calibrate my system to 85 percent

Cause 1: Your sensor may have a short to ground.

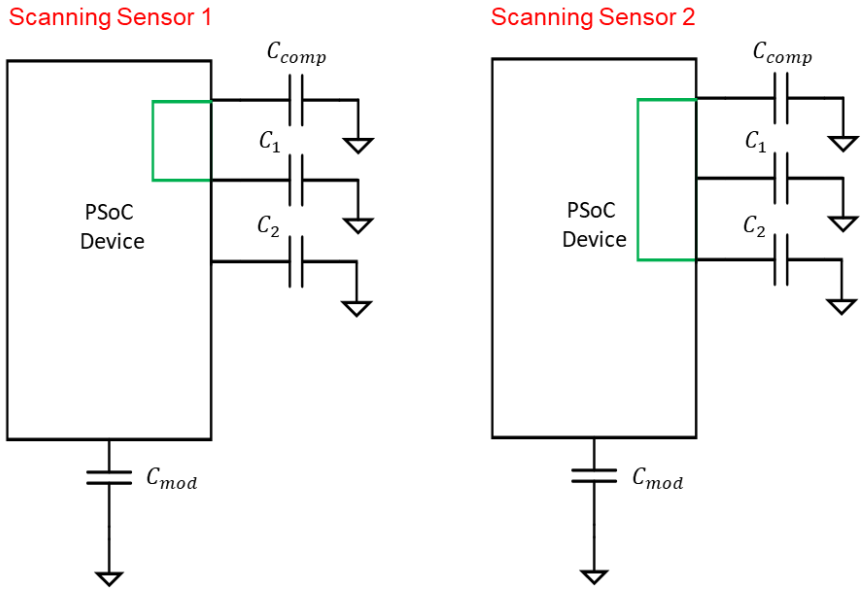
Solution: First, use a multimeter to check if there is a short between your sensor and ground. If it is present, review your schematic and layout for errors.

Cause 2: Your sensor C_P may be too high or too low.

Solution: 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. If you do not have a hardware option to use shield or if after enabling the shield, your C_P remains greater than the device supported C_P, contact Cypress Technical Support to review your layout or for further application-specific guidance.

If you suspect the capacitance to be low compared to the minimum supported parasitic capacitance by the device, add a footprint of the capacitor to a pin. In the final design, if the C_P is identified to be lower than the supported range, place an additional compensation capacitor to increase the sensor C_P to the supported range by dynamically connecting it to the sensor while scanning. See the Component Datasheet / Middleware Document to understand how to gang the sensors to an external compensation capacitor connected to a pin to increase the C_P whenever required.

Figure 5-41. Gang the Sensors to the External Compensation Capacitor



5.3.5.8 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 and update the layout according to the Slider Design section. See the Component Datasheet / Middleware Document for details on BIST API.

5.3.5.9 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.

5.3.5.10 Raw counts show a level-shift or increased noise when GPIOs are toggled

Cause 1: The sensor traces are routed parallel to the toggling GPIOs on your PCB.

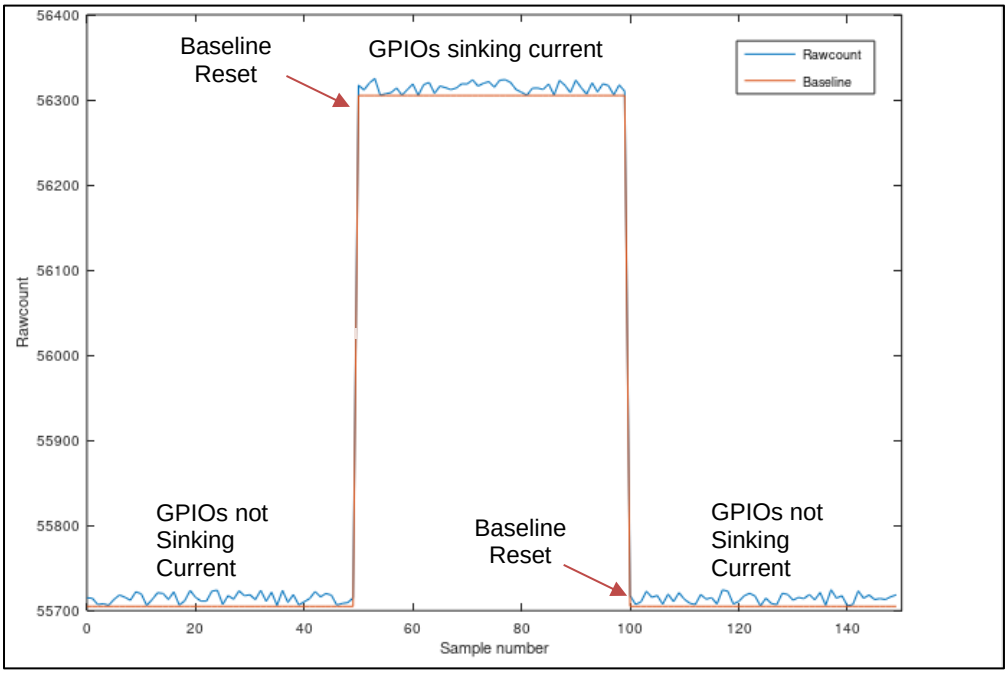
Solution: Your layout needs to be updated. Review Trace Routing and update your layout as required. If the layout cannot be modified at the current stage, you could evaluate the use of firmware filters to reduce the peak-to-peak noise and hence improve SNR.

Cause 2: A large amount of current is being sinked through the GPIOs.

Solution: Limit the amount of DC current sink through the GPIOs when CapSense sensors are being scanned. See Schematic Rule Checklist. If the current sink through GPIOs is firmware-controlled, and the raw count-level-shift caused by current sink has a large difference compared to the touch signal, you could implement firmware techniques like resetting or re-initializing the CapSense baseline whenever the current sink is enabled through the GPIOs. The baseline of the CapSense sensor could be reset by using the `CapSense_InitializeWidgetBaseline()` API function as shown below:

```
CapSense_InitializeWidgetBaseline(CapSense_CSD_BUTTON_WDGT_ID);
```

Figure 5-42.Resetting Baseline using Firmware Technique



Cause 3: You did not follow the guidelines mentioned in Sensor Pin Selection section.

Solution: Follow the recommendations in Sensor Pin Selection section. In addition, for PSoC 6 family of devices, follow these guidelines on drive mode strength, switching frequency and slew rate selection, and so on:

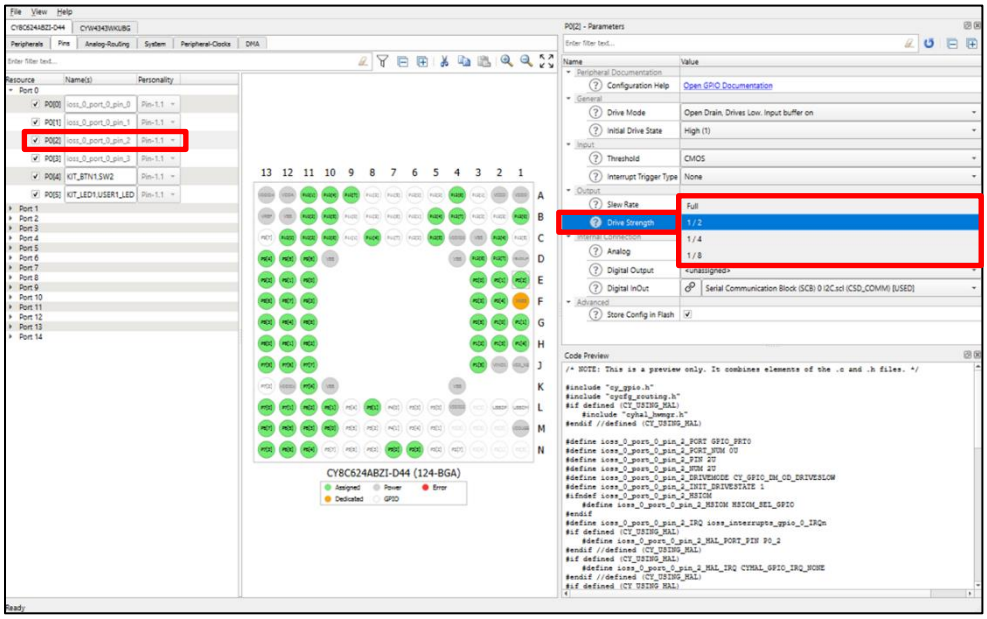
- Reduce the drive strength of the switching GPIOs. Table 5-14 lists the available drive strength options for the GPIOs. Figure 5-43 shows an example on how to select the drive strength of the GPIOs using the Device Configurator in the ModusToolbox project.

Table 5-14. Drive Strength for GPIOs

Drive Strength	Drive Current in mA
Full	8
$\frac{1}{2}$	4
$\frac{1}{4}$	2
$\frac{1}{8}$	1

- Decrease the switching frequency of the GPIO being toggled.
- Use GPIO slew rate as SLOW mode (note that this limits the toggling frequency to 1.5 MHz). See Table 7-13 or more details.
- Use PRS as the Sense Clock source.
- If possible, reduce VDDA to lower than 2.7 V.
- Try to restrict GPIO switching to intervals between CapSense scans.

Figure 5-43. Selecting Drive Strength for GPIOs



5.3.5.11 I am getting a low SNR

Cause 1: Sensor is not tuned properly.

Solution: Follow the tuning guidelines in CapSense Performance Tuning.

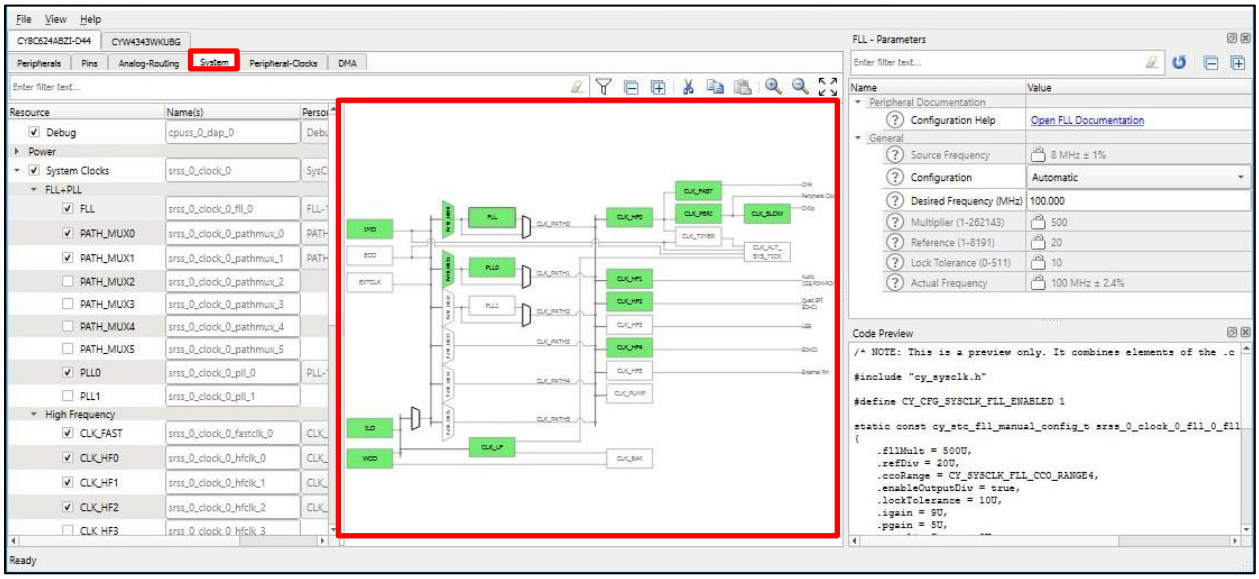
Cause 2: CapSense and other peripherals are not properly assigned to the recommended pin.

Solution: See Sensor Pin Selection and Raw counts show a level-shift or increased noise when GPIOs are toggled for more details.

Cause 3: HFCLK source may be causing higher noise for a PSoC 6 device.

Solution: For the best performance of CapSense in PSoC 6 family of devices, use HFCLK derived from the IMO/ECO+PLL clock source. This clock source provides the best SNR performance. Figure 5-44 shows how to change the clock settings using the **System** tab in the Device Configurator for a ModusToolbox project. Also see AN221774 - Getting Started with PSoC 6 MCU for more details on changing the clock settings of the device.

Figure 5-44. Changing Clock Settings in Device Configurator



5.3.5.12 I am observing a low Cm for my CSX Button

Cause: The mutual capacitance between the Tx and Rx electrode should be higher than approximately 750fF for proper IDAC calibration.

Solution: It is recommended to have two free pins in your device with footprint to add extra Capacitance if Cm of the button turn out to be low. We could then increase the sensor Cm to the supported range by dynamically connecting external capacitor to the CSX sensor while scanning as shown in the below figure, where Pin1 is ganged to the Tx pin and Pin2 is ganged to the Rx pin of the sensor respectively. This will help in mitigating low Cm risk if it is found during testing phase. See Component Datasheet / Middleware Document to understand how to gang the sensor. Figure 5-46 shows the addition of the external capacitor as a button widget in the CapSense component and assigning dedicated pins to the Tx and Rx electrode. Figure 5-47 shows the ganging of the sensor to the external capacitor by assigning Selected pins to both sensor pin and external capacitor pin, this must be done for both Rx and Tx electrode. There is no need to scan the external capacitor while scanning of the widgets, thus we can selectively scan widgets using the APIs CapSense_SetupWidget() and CapSense_Scan() provided by the CapSense component.

Figure 5-45. Ganging External Capacitor to Increase the Cm of the Sensor

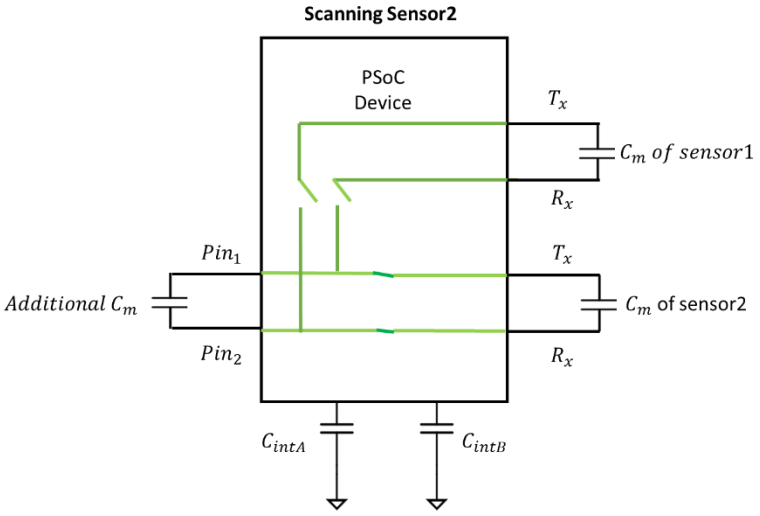


Figure 5-46. Assigning Dedicated Pins to the External Capacitors

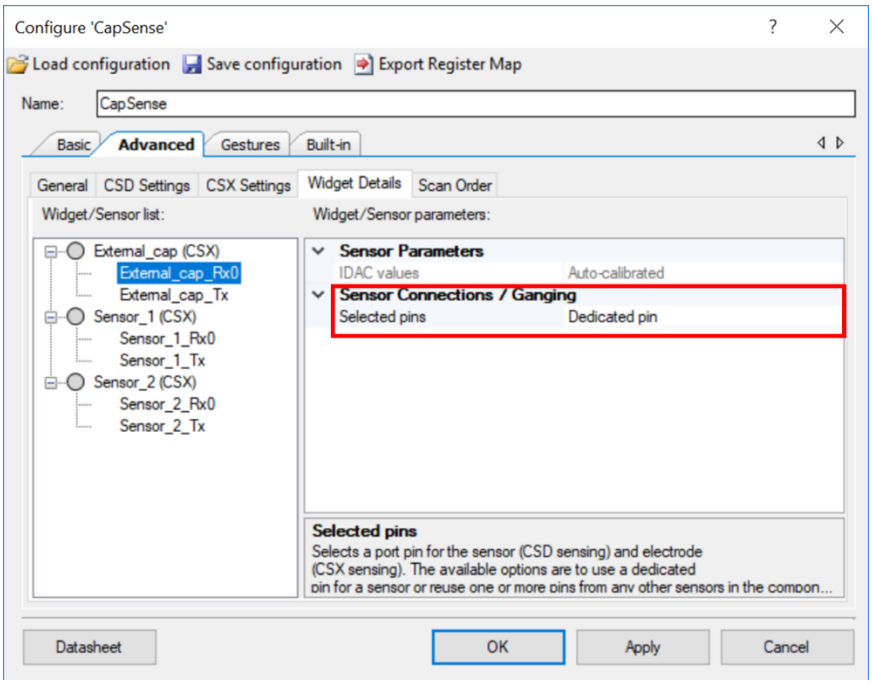
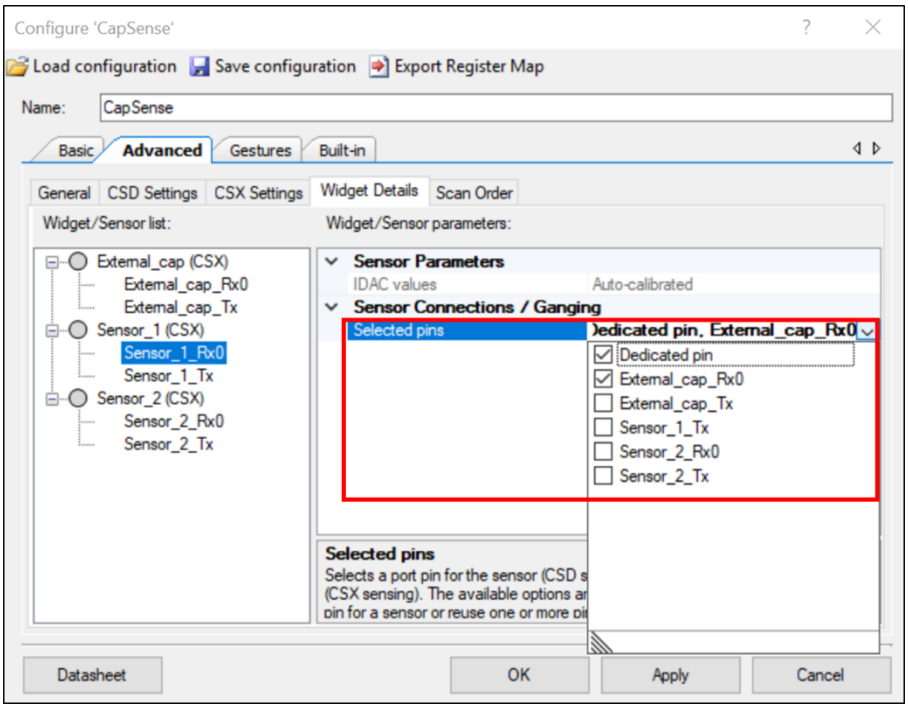


Figure 5-47. Gaining the External Capacitor and Sensor Pin



6 Gesture in CapSense

6.1 Touch Gesture Support

The CapSense Component in PSoC 4 and PSoC 6 MCU supports the gesture detection feature for sliders and touchpad widgets. It allows to identify different predefined gestures based on touch patterns on sliders and touchpad widget.

Note that the gesture detection feature is available for selected device part numbers. If you intend to use the gesture feature of the component, ensure that you select the device that supports this feature.

6.2 Gesture Groups

Gestures are divided into several groups: Click, One-finger Scroll, Two-finger Scroll, Two-finger Zoom, One-finger Edge Swipe, One-finger Flick, and One-finger Rotate.

Table 6-1 shows gestures supported by various widgets. See Component Datasheet / Middleware Document for more details on how these gestures are defined and what are the parameter that must be configured in the CapSense configurator to detect these gestures.

Table 6-1. Gesture Supported by Different CapSense Widgets

Widget Type	Gesture Groups						
	Click	One-finger Scroll	Two-finger Scroll	One-finger Flick	One-finger Edge Swipe	Two-finger Zoom	One-finger Rotate
Button							
Linear Slider	✓	✓		✓			
Radial Slider	✓						
Matrix Buttons							
Touchpad	✓	✓	✓	✓	✓	✓	✓
Proximity							

6.3 One-Finger Gesture Implementation

Implementing gesture detection involves following steps:

1. Tuning the Widget
2. Selecting Predefined Gesture
3. Firmware Implementation with a Timestamp
4. Tuning Gesture Parameters

6.3.1 Tuning the Widget

Tune the CapSense hardware and software parameters for the widget. Usually, in a gesture application, because of the speed and orientation of the finger movement changes, the finger may make a very little contact with the widget. This could be confirmed by viewing the centroid data in the Tuner GUI when the gesture is being performed. If the sensitivity is good enough, you will get the data without any break. If you observe any break in the centroid data, increase the sensitivity until the data for the gesture is complete and appear without any break.

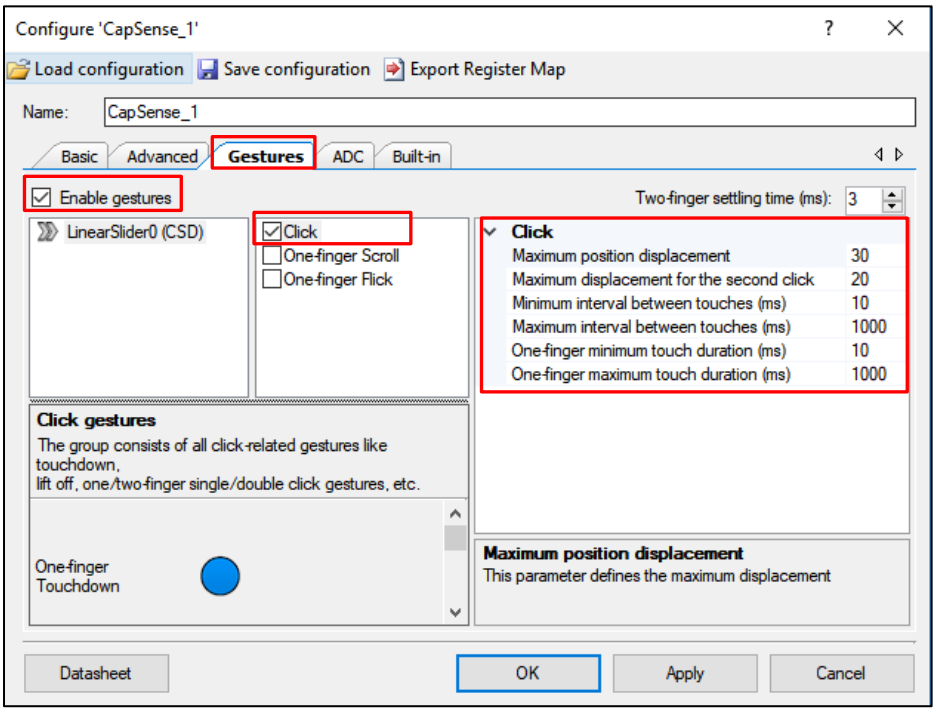
Ensure that you get a SNR above 5:1 for the slight finger contact that you may want to detect. Also, ensure that you have a linear centroid response w.r.t the finger position on the slider or touchpad. Tune the sensors using guidelines in section Slider Tuning Guidelines for achieving the same

6.3.2 Selecting Predefined Gesture

First, you need to enable **Gestures** in the Gesture tab in CapSense Component. All gesture-related configuration parameters appear after enabling gestures; these parameters are systematically arranged by widgets / gesture groups as Table 6-1 shows. According to the application requirement, you can enable and disable gestures by selecting the checkbox. Do the following to enable gestures and configure the corresponding parameters.

- Select the widget for which gesture feature must be enabled (in the Widget pane). If you have multiple widgets in the project, the PSoC Creator allows gesture recognition only one widget. However, in ModusToolbox, gesture recognition can be enabled on more than one widget.
- Check the desired gesture groups (in the Gesture Group pane). In PSoC Creator, you cannot enable scroll gesture and flick gesture at the same time. This is applicable for both sliders and touchpad. However, in ModusToolbox, you can enable more than one gesture according to the application requirement.
- Configure all parameters (in the Parameter pane). When a gesture is selected, the right pane of the window shows parameters related to that gesture group. See the Component Datasheet / Middleware Document.

Figure 6-1. Configuring Gestures in CapSense Component



6.3.3 Firmware Implementation with a Timestamp

See the code example PSoC 4 CapSense Touchpad Gestures to understand how to implement timestamp for gesture recognition. Because each gesture has a pattern of touch that changes with time, a reference timestamp is needed for properly getting the touch data with respect to time. This time stamp represents the sampling rate for the gesture recognition algorithm. Both the centroid positions and their respective timestamp are used by the gesture decoding API to determine different predefined gesture patterns that are applicable for the widget.

First, tune the widget by the procedure described in Section 6.3.1 and determine the time interval between two successive CapSense scans in the firmware. Update the timestamp exactly with this duration. The way to accurately determine it is to toggle a GPIO in the firmware after the CapSense scan is complete and find the time duration using an oscilloscope.

6.3.4 Tuning Gesture Parameters

This section describes how to set gesture parameters for sliders; the same procedure could be extended to the gesture groups supported by touchpads. CapSense sliders supports Click, One finger Scroll, and One finger flick gesture features. See the Component Datasheet / Middleware Document.

6.3.4.1 Using Tuner GUI for Tuning Gesture Parameters

You can use the Cypress Gesture View in Tuner GUI for tuning the gesture parameters and visualize and analyze the performance of the gesture detected in the end system.

Keep in mind the following while using Tuner GUI for gestures:

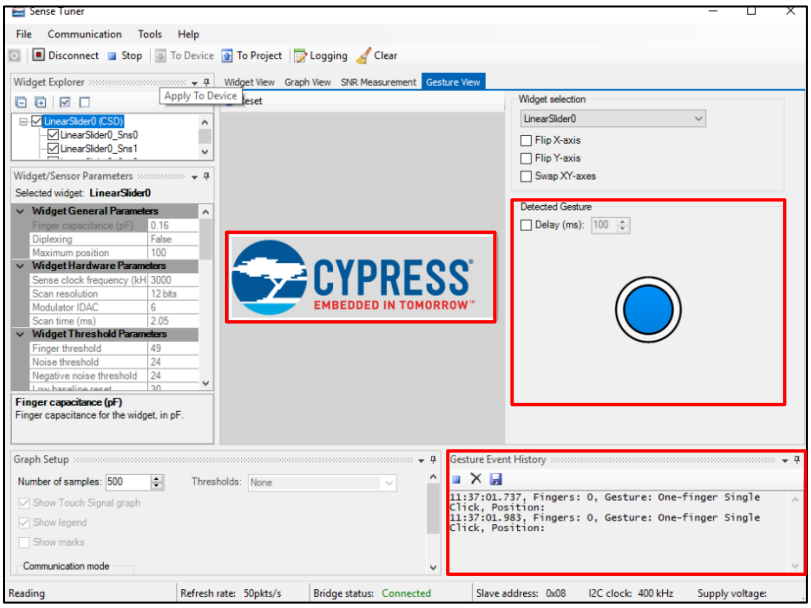
1. For tuning gesture parameters in runtime, Tuner GUI must be used with EZI2C. Use Synchronized communication mode for visualizing the detected gestures in runtime. For more details on using the Tuner GUI, see the Component Datasheet / Middleware Document and the PSoC 4 CapSense Touchpad Gestures Code Example. All the parameters for the gestures that are available in the CapSense configurator are available in Tuner GUI, where you can directly edit these values for tuning.
2. As Figure 6-2 shows, the Gesture View tab is organized into different panes as follows:

   Gesture Event History pane shows detected gestures and their positions on the widget.

   Detected gesture pane indicates the detected gesture. If the delay checkbox is enabled, a gesture picture is displayed for the specified time-interval; if delay is disabled, the last reported gesture picture is displayed until a new gesture is reported.

   Cypress Icon in the Tuner GUI moves according to the scroll gesture. It indicates how well the parameter of the scroll gesture is tuned. This dynamic feature gives performance feedback for further fine-tuning gesture parameters.

Figure 6-2. Tuner GUI for Gestures



3. Determining the event duration using Tuner GUI. A general equation to determine the event duration is given by Equation 6-1.

Equation 6-1. Gesture Duration

$$Event\ duration = No.of\ Samples\ x\ T_{sample}$$

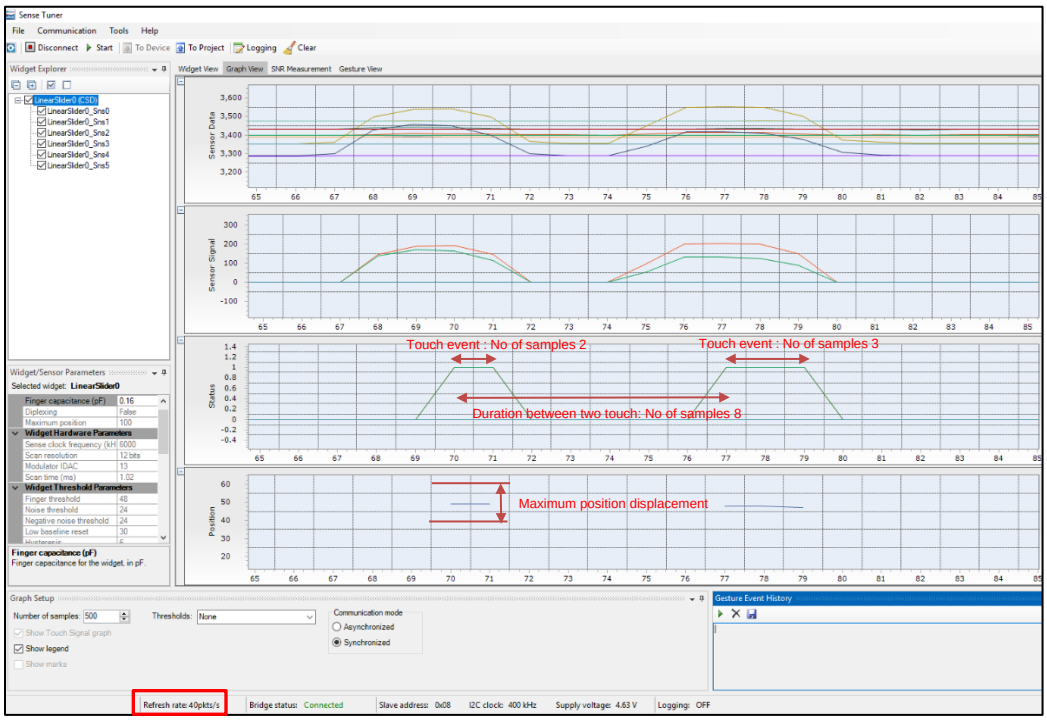
Where,

$No.of\ Samples$ represents the number of samples the gesture event occurred. This data could be obtained from the Graph view in the Tuner GUI.

T_{sample} represents the time interval between two samples.

$$T_{sample} = \frac{1}{Refresh\ rate}$$

Figure 6-3. Determining the Gesture Parameters Using Tuner GUI



6.3.4.2 Click

There are two type of click gestures: single click and double click. The Table 6-2 shows the list of parameters that need to be configured for the click gesture in both PSoC creator and in ModusToolbox. See Component Datasheet / Middleware Document. Table 6-3 shows the recommended values of the gesture parameter for the click gesture.

Table 6-2. Click Gesture Parameters

Gesture	PSoC Creator	ModusToolbox
Single click	One finger minimum touch duration	Minimum click timeout
	One finger maximum touch duration	Maximum click timeout
	Maximum position displacement	Maximum click distance
Double click	Minimum interval between touches	Minimum second click interval
	Maximum interval between touches	Maximum second click interval
	Maximum displacement for the second click	Maximum second click distance

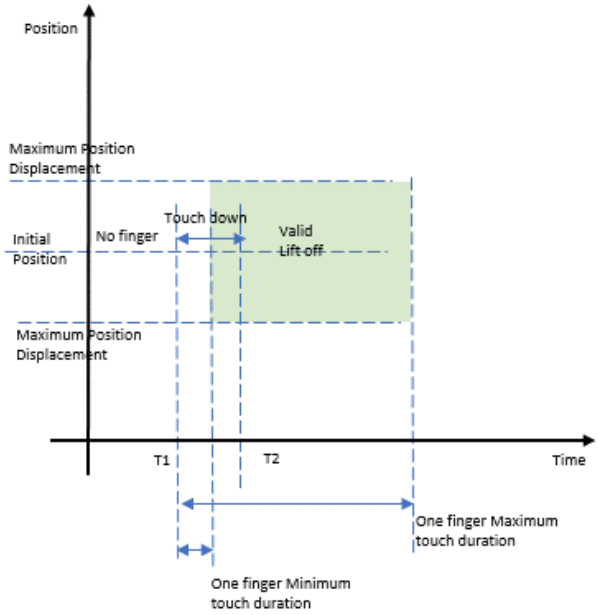
Table 6-3 Recommended Values for Click Gestures

Parameters	Typical Values
Maximum position displacement	20% of Max position of the slider
Maximum position displacement for the second click	20% of Max position of the slider
Minimum interval between touches (ms)	60
Maximum interval between touches (ms)	400
One finger minimum touch duration (ms)	20
One finger maximum touch duration (ms)	400

6.3.4.2.1 Single Click

A single click is defined as a touch-down event followed by a lift-off. Figure 6-4 shows the spatial and timing condition that must be satisfied for a valid single-click event.

Figure 6-4. Single-Click Gesture



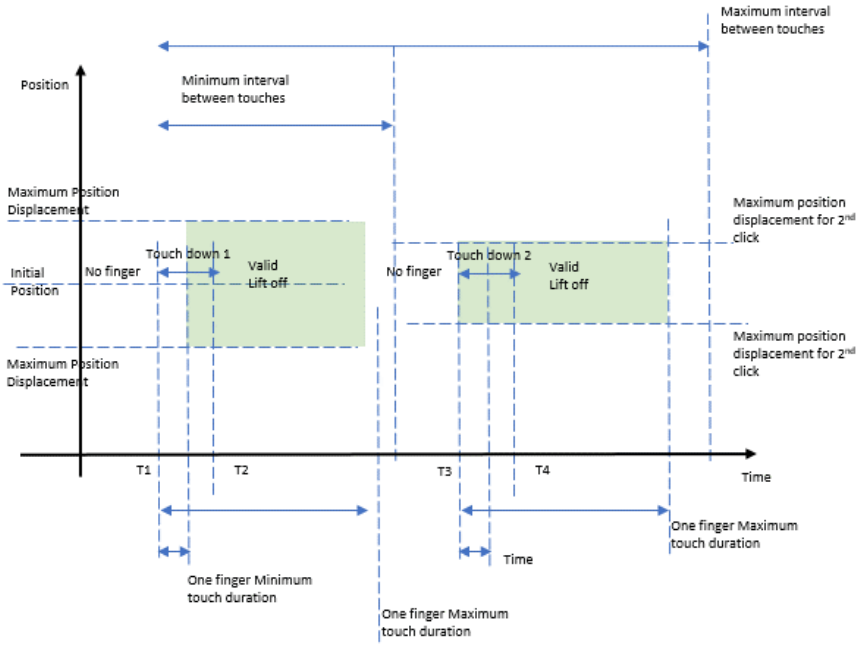
From Figure 6-4, at time T1, the finger touched down on the slider; at time T2, the finger is lifted off from the slider. For a valid single click, the touch-down duration should be between the “One finger minimum touch duration” and “one-finger maximum touch duration” and the relative position of the liftoff from the initial position of touch should be less than the “Maximum position displacement” parameter.

The duration of each single-click event can be determined by using Equation 6-1 by finding the number of samples for the single click in the Graph view of Tuner GUI and the refresh rate as shown in Figure 6-3. From the single-click event duration, fix the parameters “One-finger minimum touch duration” and “one-finger maximum touch duration”. The Maximum position displacement parameter can be determined by observing the maximum variation in the centroid position using the Tuner GUI as shown in Figure 6-3. Its recommended value is 20 percent of the Maximum centroid position of the slider as mentioned in Table 6-3.

6.3.4.2.2 Double Click

A double click is two single-clicks event occurring one after another with the second click occurring between the minimum and maximum time interval between the two touches. In addition, the relative position of the second click from the initial position of touchdown event should be less than the Maximum position displacement for the second click. Figure 6-5 shows the spatial and timing conditions that must be satisfied for a valid double-click event.

Figure 6-5. Double-Click Gesture



From Figure 6-5, at time T1, the finger touched down on the slider for the first click; at time T2, the finger is lifted off from the slider. At T3, the finger touched down on the slider for the second click; at T4, the finger is lifted off from the slider. For a valid double click, each click should satisfy the condition of single click, and the second click should occur between Minimum and Maximum interval between touch parameters.

Using the Graph View in the Tuner GUI, observe the double-click touch data and by using Equation 6-1. Determine the parameter of single click as mentioned in the Single Click section. After these, determine the duration between the two touches using the Graph View in the Tuner GUI and set the value of the minimum and maximum intervals between touch parameters. A typical captured data for the double-click event is shown in Figure 6-3.

6.3.4.3 Scroll

There are two different scroll gestures that can be detected on sliders: One Finger Scroll and One finger inertial scroll. See Component Datasheet / Middleware Document. Table 6-4 shows the parameters that must be configured for the scroll gesture. Note that One finger inertial scroll gesture is not supported in ModusToolbox.

Table 6-4 One Finger Scroll Parameters

Gesture	PSoC Creator	ModusToolbox
One Finger Scroll	Position threshold N	Minimum scroll distance
	Scroll step	-
	Debounce	Scroll debounce
One Finger Inertial Scroll	Position inertial threshold	NA
	Count level	

6.3.4.3.1 One Finger Scroll

A one-finger Scroll gesture is a combination of a touchdown followed by a displacement in a specific direction. The change in position between two consecutive scans must exceed the Position Threshold value given in the configurator after tuning. See Component Datasheet / Middleware Document.

Do the following to set scroll gesture parameter values as shown in Table 6-4.

1. Determine the number of samples of the scroll gesture from the Graph View (Centroid position) in Tuner GUI.
2. By using Equation 6-1, determine the duration of the complete scroll.

3. Determine the change in centroid position for the complete scroll using the Tuner GUI.
4. Determine Position Threshold from Equation 6-2. Each gesture is scanned at a sample rate that is set in the timestamp in the application code. The position threshold is given by the change in the centroid position for the duration that is set in the timestamp.

Equation 6-2. Equation to Determine Position Threshold

$$Position\ Threshold = \frac{Change\ in\ Centroid\ position}{Total\ duration\ of\ scroll}\ x\ Duration\ of\ timestamp$$

5. In PSoC Creator, set four different position thresholds and their scroll count values in the configurator, which are determined by varying the speed of the scroll gesture. Now, change the speed of scroll and repeat the steps 1 – 4 and set these position threshold values. In ModusToolbox has only one parameter: Minimum Scroll distance; determine its value in the same way you determined the position threshold.
6. Read the scroll step from the CapSense data structure and use it to control the speed and smoothness of the scroll gesture. The scroll step depends on the position threshold. This scroll step is used in the application code to control the actual variable value to be changed with respect to scroll. Note that the scroll step parameter is not available in ModusToolbox.
7. Set the maximum slider position as ten times the dimension of the slider as a general rule. If you set scrollDistanceMin=10, everything below a 1-mm movement will not detect the scroll gesture. Everything above this number might detect a gesture.

Observe the Cypress icon in the Tuner GUI (Figure 6-2) to get a feedback on how well the tuning has been done for the scroll gesture in the given hardware. You can also print the variable that must be controlled by scroll through UART to visualize how the value is changing with respect to scroll. This could be used as a visual feedback. The position threshold parameters and the corresponding step counts should be tuned until the variation in the variable value with respect to scroll meet the requirement of the end user application.

6.3.4.3.2 One-Finger Inertial Scroll

The one-finger inertial scroll gesture is defined as a touchdown event followed by a minimum displacement in a specific direction, and then a liftoff. The movement of scroll will automatically stop when it reaches the end value of the variable. See Component Datasheet / Middleware Document.

The gesture parameter is given in Table 6-4. The position inertial threshold parameter is given by the minimum change in centroid position that is required before a liftoff; its value can also be determined by steps in the One Finger Scroll section. The count value parameter defines the momentum of scroll; it can take two possible values: low or high. Choose the count value according to the end application requirement.

6.3.4.4 One-Finger Flick

A flick gesture is a touchdown event followed by a high-speed displacement and a liftoff event (see Component Datasheet / Middleware Document). The flick gesture is similar to the one-finger inertial scroll; the only difference is that it requires a high-speed displacement followed by a liftoff event within the maximum sample interval defined in the configurator. You can determine the position threshold and the maximum sample interval by following the same procedure as in the One Finger Scroll section and by using Equation 6-1.

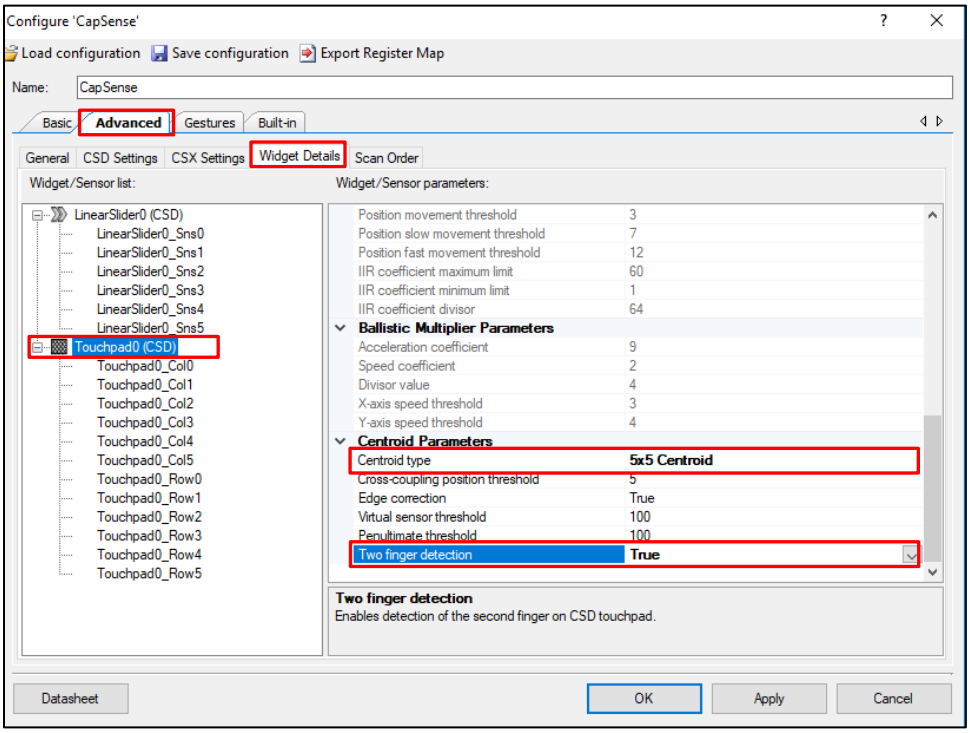
Table 6-5. One-Finger Flick Gesture Parameters

Gesture	PSoC Creator	ModusToolbox
One Finger Flick Gesture	Position threshold	Minimum flick distance
	Maximum sample interval	Maximum flick timeout

6.4 Two-Finger Gesture Implementation

Two-finger gestures such as two-finger scroll and two-finger zoom are supported in the touchpad widget. You must enable this feature in the Widget Details tab of the Touchpad Widget. The procedure for tuning the parameters is the same as mentioned in the One-Finger Gesture Implementation section (see Component Datasheet / Middleware Document). Figure 6-6 shows how to enable two-finger touch gestures in the configurator, select the centroid type as 5x5 Centroid, and set the two-finger detection as True.

Figure 6-6. Enabling Two-Finger Touch Gestures in the CapSense Component



6.5 Advanced Filters for Gestures

Cypress provides more advanced filtering features for gestures such as Ballistic multiplier, and Adaptive IIR filter, and the edge correction feature to improve gesture recognition and the user experience. See Component Datasheet / Middleware Document.

7 Design Considerations

This chapter explains firmware and hardware design considerations for CapSense.

7.1 Firmware

The PSoC 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 or ModusToolbox 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 mode of PSoC 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.

If you want to develop firmware using the ModusToolbox software, see the references in the section ModusToolbox of this document.

7.1.1 Low-Power Design

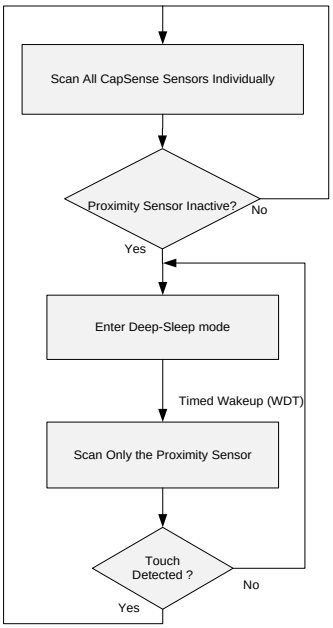
PSoC low-power modes allow you to reduce overall power consumption while retaining essential functionality. See AN86233 - PSoC 4 Low-Power Modes and Power Reduction Techniques, for a basic knowledge of PSoC 4's low-power modes, and see AN219528 - PSoC® 6 MCU Low-Power Modes and Power Reduction Techniques, for PSoC 6'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.

There are different firmware and hardware techniques to reduce the power consumption of the CapSense system.

1. If you use APIs that scan multiple widgets together, the device returns to Active mode after finishing the scan of a single widget. Therefore, you should scan each widget individually for reducing the power consumption in the design. See the CapSense Component datasheet.
2. You can use the Deep-Sleep mode of PSoC 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 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.
3. 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. For saving power in a design with multiple sensors, you should include a separate proximity loop that surrounds all the sensor. 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 7-1 illustrates this process.

Figure 7-1. Low-Power CapSense Design



4. The CapSense Component can reduce power consumption by reducing the execution time of scanning by ganging sensors together and managing scanning at the application level. In this case, all the sensors in the design are “ganged” i.e., simultaneously connected to the AMUX bus to form a virtual sensor. See the code example PSoC 4 Low Power Ganged Sensor and AN92239 - Proximity Sensing with CapSense for details on ganged sensor implementation. A ganged sensor has different tuning parameters because its properties are different compared to considering the sensors individually. Therefore, it should be considered as a single CSD button and tuned separately; see Manual Tuning. The ganged sensor is periodically scanned by using a watchdog timer (WDT); if the ganged sensor reports a touch event, enable the scanning of the actual widgets that need to be scanned. This is helpful in CapSense designs that requires Wake on Touch modes. The procedure is similar to what is explained in Figure 7-1. You can achieve very low system current while maintaining a good touch response, by properly tuning CapSense and the wakeup interval. This technique could also be used with the CSX touchpad widget.
5. If high-speed peripherals such as system timers and I^2C are required, you can put the CPU to sleep mode instead of going to deep sleep mode.
6. You can also add a shield hatch in the design, as explained in Driven-Shield Signal and Shield Electrode to reduce the parasitic capacitance and therefore, the scan time. The scan time and power consumption is directly related; thus, the power consumption is reduced by lowering the scan time.

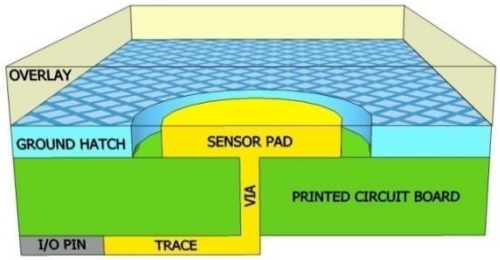
Note: In PSoC 4000 devices, it is not recommended to enter Sleep mode if a CapSense scan is in progress.

7.2 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 PSoC 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 7-2 shows the most common CapSense sensor construction.

Figure 7-2. CapSense Sensor Construction

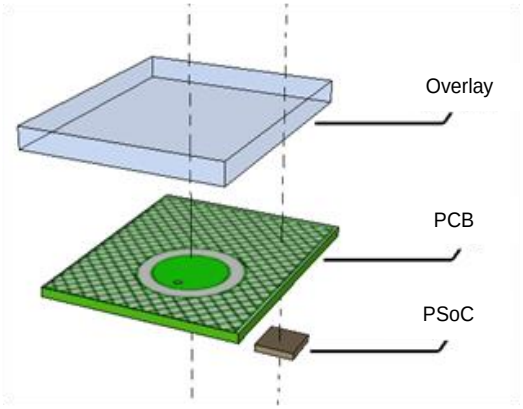


The copper pads etched on the surface of the PCB act as CapSense sensors. A nonconductive overlay serves as the touch surface. The overlay also protects the sensor from the environment and prevents direct finger contact. A ground hatch surrounding the sensor pad isolates the sensor from other sensors 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 it to ground. See Liquid Tolerance section for details.

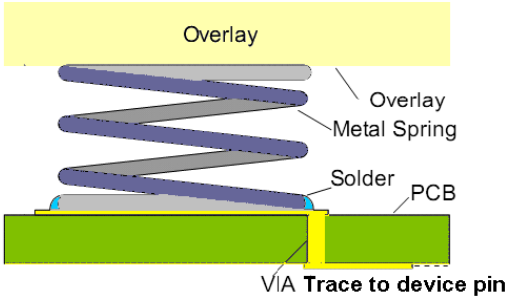
The simplest CapSense PCB design is a two-layer board with sensor pads and hatched ground plane on the top, and the electrical components on the bottom. Figure 7-3 shows an exploded view of the CapSense hardware.

Figure 7-3. 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 7-4 shows, to create elevated sensors that allow overlay to be placed at an elevated distance from the PCB. See 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 7-4. Sensor Construction Using Springs as Sensors



7.3 Overlay Selection

7.3.1 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 7-1 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 7-1. Relative Permittivity of Overlay Materials

Material	ε_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.

7.3.2 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 7-2 lists the recommended maximum thickness of acrylic overlay for different CapSense widgets.

Table 7-2. Maximum Thickness of Acrylic Overlay

Widget	Maximum Thickness (mm)
Button	5
Slider	5[8]
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 $\varepsilon_r = 8$, while acrylic has approximately $\varepsilon_r = 2.5$. The ratio of $\varepsilon_r/2.5$ 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.

In addition, avoid using very thin or no overlay. It is important to have a minimum overlay thickness in a CapSense design for the following reasons:

[8] 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.

a. An overlay provides protection from the environmental condition, prevents direct finger contact, and gives ESD protection. The overlay thickness should be small enough to give a good signal, and decided based on the button size and the strength to withstand ESD. See AN64846 Getting Started with the CapSense.

b. For the CSD button, if there is no overlay the buttons will be over sensitive.

c. For sliders, if there is no overlay, the raw count may saturate for the slider segments and may cause non-linear centroid response for slider. See Slider Design.

d. For the CSX sensor, it is recommended to have a minimum overlay thickness of 0.5 mm. If it is violated, sudden decrease in raw count is observed when a finger is placed on a sensor or a water drop falls on the Tx and Rx electrodes. See Effect of Grounding.

7.3.3 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.

7.4 PCB Layout Guidelines

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

7.4.1 Sensor C_P

In a CapSense system design, the C_P of the sensor must be within the supported range of the device. You can find the supported C_P range in the Component Datasheet / Middleware Document. The main components of C_P are trace capacitance, sensor pad capacitance, and pin capacitance of the device. The pin capacitance is device-dependent (see the respective Device Datasheet), so you can only design your sensor and trace capacitance to be able to meet the C_P criteria in the datasheet. The relationship between C_P and the PCB layout features is not simple. C_P increases with an increase in the sensor pad size and trace length and width, and with a decrease in the gap between the sensor pad and the ground hatch.

There are many ways to decrease the C_P:

- Decrease the trace length and width as much as possible. Reducing the trace length increases noise immunity.
- Drive the hatch with a shield signal. See Driven-Shield Signal and Shield Electrode.

Reducing the sensor pad size is not recommended because it also reduces the finger capacitance. In some special cases, such as small sensor pad and very small trace length due to placement of the sensor pad close to the device, there is a possibility of the sensor C_P to be lower than the supported minimum C_P by the device. In that case, add a footprint of the capacitor across the sensor or any unused pin. If the C_P is identified to be lower than the supported range, place a 4.7-pF capacitor across the sensor or on the unused pin and gang the capacitor during the CapSense scan, refer to the FAQ 5.3.5.7 for more details. This will increase the C_P of the sensor to the supported range.

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

7.4.2 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 too. You can 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.

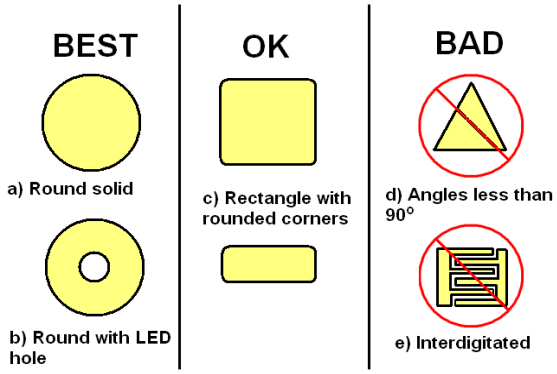
7.4.3 Button Design

7.4.3.1 Self-Capacitance Button Design

The self-capacitance button has a single electrode and can have different shapes and size as recommended below.

Shape: 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 90º) since they concentrate electric fields. Figure 7-5 shows recommended button shapes.

Figure 7-5. Recommended Button Shapes

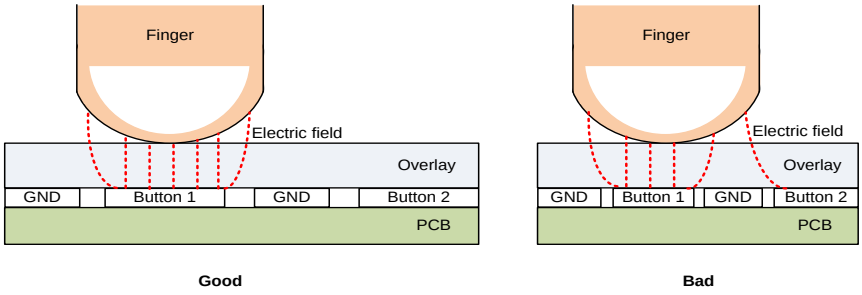


Size: Button diameter should be 5 mm to 15 mm, with 10 mm suitable for most applications. A larger diameter is appropriate for thicker overlays.

Spacing: 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 7-6 shows.

Figure 7-6. Spacing between Buttons



7.4.3.2 Mutual- Capacitance Button Design

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 Tx and Rx electrodes to a maximum extent. There are standard button patterns that could be used for the mutual capacitance button design and their parameters could be modified based on the application requirement. Fishbone pattern is one of the mutual capacitance patterns which give better performance in terms of SNR.

7.4.3.2.1 Fishbone pattern

Prongs or fishbone are standard shapes for mutual-capacitance buttons. The Tx forms a box or ring around the button for shielding Rx from noise. There are interlaced Tx and Rx prongs inside the border to form the electric field. Figure 7-7 shows an example of a two-prong fishbone sensor structure with top and bottom view with hatched ground. The gap between the outer wall of the Tx electrode and the coplanar hatch ground should be greater than the air-gap of Tx and Rx electrodes. The reference plane (PCB bottom layer) of the Fishbone structure should have void region as shown in Figure 7-7.

Figure 7-7. CSX Fish bone button pattern with 2 Rx prongs

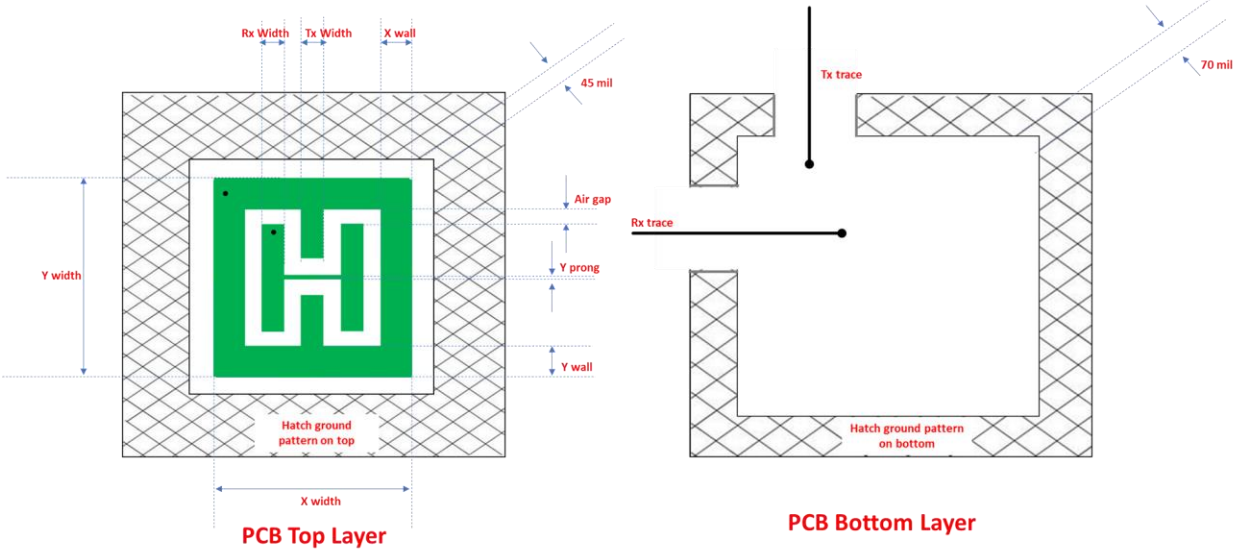


Table 7-3 lists the suggested fishbone button design parameters for some commonly used sensor sizes and overlay like glass and acrylic. As explained in section Sensor Size, the recommended button size is to keep the button X and Y dimension close to the sum of expected user finger size and overlay thickness. However, the table lists multiple button sizes that you can chose from if you have constraints on available space on board or if you would like to have a bigger button for your application for ease of user interaction etc.

Also, note that for a given button size, the achievable SNR decreases with increased overlay thickness. Thus, if you plan to use thick overlays (approx. > 1mm acrylic or 2mm glass) ensure to avoid compromising on button size due to board space because that will further limit the button SNR performance. Ensure to use bigger buttons (>= biggest expected finger size) for such thick overlays. And also for small buttons better to have thin overlays for getting good SNR.

Table 7-3. Dimension of Fishbone Buttons (all units in mm)

Button Size (X-Size, Y-Size) (mm)	Number of Rx-prongs	Air Gap between Tx and Rx in mm	Tx Width in mm	Rx Width in mm	X-Wall Width in mm	Y-Wall Width in mm	Y Prong in mm
5, 5	3	0.35	0.48	0.48	0.24	0.24	0.2
10,7	3	0.75	0.92	0.92	0.46	0.46	0.2
10,5	3	0.5	1.17	1.17	0.58	0.58	0.2
10,10	2	0.9	1.60	1.60	0.80	0.80	0.2
12, 12	2	1.3	1.70	1.70	0.85	0.85	0.2
13, 10	2	1.1	2.15	2.15	1.08	1.08	0.2
13, 13	2	1.5	1.75	1.75	0.88	0.88	0.2
15, 15	2	1.7	2.05	2.05	1.03	1.03	0.2
17, 17	2	2.3	1.95	1.95	0.98	0.98	0.2
20, 13	2	1.8	3.20	3.20	1.60	1.60	0.2

Button Size (X-Size, Y-Size) (mm)	Number of Rx-prongs	Air Gap between Tx and Rx in mm	Tx Width in mm	Rx Width in mm	X-Wall Width in mm	Y-Wall Width in mm	Y Prong in mm
25, 13	2	2	4.25	4.25	2.13	2.13	0.2

The above button design parameters in Table 7-3 ensure a good SNR performance if you follow the schematics and layout guidelines in this chapter.

Note: In case if you expect a higher external noise in the design and for other complex cases you can contact Cypress Technical Support for any assistance in the button design. Refer to the section Noise in CapSense System for more details about the external noise. And in the design if you expect a low Cm then follow the guidelines mentioned in the section I am observing a low Cm for my CSX Button for mitigating it.

7.4.3.2.2 Button design for arbitrary shapes and dimensions

Figure 7-8 shows the different orientation of Rx prongs in the Fish bone pattern, in Button A the Rx prong is perpendicular to the side of the button with larger dimension and in Button B the Rx prongs is parallel to the side of the button with larger dimension. Orientation of Rx prongs like in Button A results in optimized button pattern compared to Button B. Thus, it is always recommended to have Rx prongs perpendicular to the side of the button with larger dimension. Thus if you need a 10x13mm button, then simply use the 13x10mm button from Table 7-3 and rotate it 90 degree to get 13x10 mm button pattern as shown in Figure 7-9.

Figure 7-8. Orientation of Rx prongs

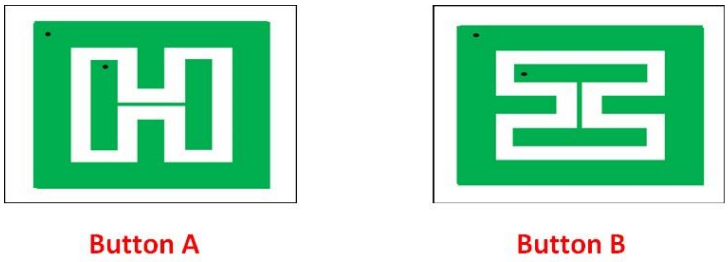
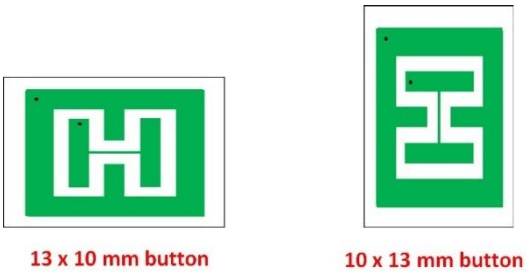


Figure 7-9 Rotating the button 90 deg to get the desired button dimension

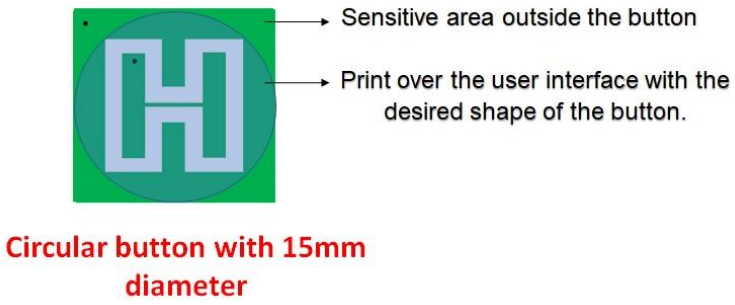


There may be some design where you need a different button shape than the recommended rectangular, like an oval or circular shape etc. The below steps explain how to construct the button with nonstandard shape from the standard Fish bone pattern.

- First select the Fishbone pattern (Rectangular shape for oval shape and Square button for circular shape) from Table 7-3 to cover the desired button shape.
- Then in the user interface or above the overlay print button shape with required dimension over the fishbone pattern as shown in Figure 7-10.

Mutual capacitance buttons designed using this method have some oversensitive area or less sensitive outside the button shape as shown in the below figures, this could be mitigated by properly tuning the software thresholds of the mutual capacitance button. The below figure shows an example of a circular button made using a square fishbone pattern.

Figure 7-10. Arbitrary shape button design based on arbitrary pattern



If you want a pattern that is not present in Table 7-3, you can obtain the button parameters by following few steps. For example, if you want a 19x19 pattern, choose the pattern that is close to the required pattern from Table 7-3 like 17x17, and scale the Air Gap between Tx and Rx with respect to the button areas. For example

$$New_{gap} = TxRx_{gap} * \left(\frac{ButttonArea_{desired}}{ButtonArea_{reference}}\right)$$

Where $\mathrm{ButttonArea} = \mathrm{X\ x\ Y}$ dimension of the sensor.

Compute the Tx, Rx width and Tx wall based on the assumption below and by considering the New_{gap} as obtained above. The obtained values of the button design parameters are shown in Table 7-4. Refer to the Figure 7-7 to understand the description of the button design parameters.

$$Tx_{width} = Rx_{width}$$

$$Tx_{wall} = \frac{Tx_{width}}{2}$$

Table 7-4 Button parameter for 19x19 Button form 17x17 button

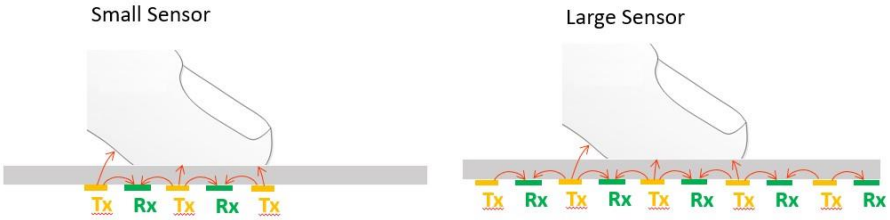
Button Size (X-Size, Y-Size) (mm)	Number of Rx-prongs	Air Gap between Tx and Rx in mm	Tx Width in mm	Rx Width in mm	X-Wall Width in mm	Y-Wall Width in mm	Y Prong in mm
17, 17	2	2.3	1.95	1.95	0.98	0.98	0.2
19,19	2	2.9	1.85	1.85	0.93	0.93	0.2

7.4.3.2.3 General recommendations on Fishbone pattern parameters

7.4.3.2.3.1 Sensor Size

The sensor size is the XY dimension of the button, it is selected based on the board space availability, expected user finger size and overlay material and thickness. Sensor size selection also depends upon the number of required buttons on the PCB considering required button-to-button gap and space availability in the PCB. But if the space is not the constrain then choose higher button size which will result in getting a good SNR. Note that increasing the sensor size beyond a point will cause the SNR to saturate, this is because some of the electric field lines from Tx/Rx electrode do not interact with the finger as shown in below figure.

Figure 7-11 Interaction of electric filed with the finger



The SNR of the button is decreased with usage of thick overlays. Thus, the recommended minimum sensor size is finger size plus overlay thickness to achieve a good SNR even with thick overlays. For example, the minimum sensor size recommended could be 13x10 mm, considering the finger size around 10mm in diameter and 3 mm overlay thickness. As mentioned in the section 7.4.3.2.2 Rx prongs should be perpendicular to the side with large dimension.

7.4.3.2.3.2 Button Spacing

The Button spacing is the gap between the Tx wall of two buttons. It helps to prevent user error by isolating the buttons from each other and reduces the cross talk. It is recommended to keep a minimum of 8mm spacing between the buttons this will ensure a good single touch and multi touch performance.

7.4.3.2.3.3 Overlay

The overlay thickness and overlay permittivity influences the SNR of the button and immunity towards the external noise such as ESD. Refer to the Overlay Selection section for more details. It is recommended to keep the overlay thickness as minimum as possible which will help in getting a higher SNR for the button and it should be high enough to provide immunity towards ESD noise. In some cases, if the overlay thickness is more and you cannot avoid it due to mechanical design consideration. In such a case, for getting a better SNR use a mutual capacitance button with bigger size than the recommended size. Refer to the section Sensor Size for selecting the minimum button dimension with respect to the overlay thickness. Using an overlay material with higher dielectric constant will also leads to higher SNR. So always use material with high dielectric constant when we use thick overlay. And also, for smaller buttons better to have thin overlays for getting good SNR.

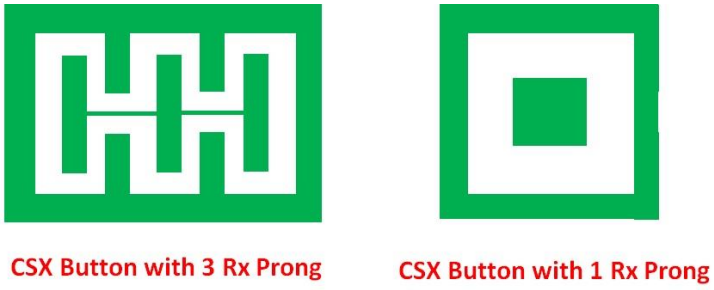
7.4.3.2.3.4 Air gap between Tx and Rx electrode

The gap between the Tx and Rx electrode influences the mutual capacitance between the Tx and Rx electrode. Increasing the gap reduces the mutual capacitance. It is the most critical parameter in the Fishbone pattern design and the gap between the Tx and Rx electrode such that the mutual capacitance is above 750fF.

7.4.3.2.3.5 Number of Rx-prongs

The number of Rx prongs influence the mutual capacitance between the Tx and Rx electrode, because increasing the number of Rx prongs decreases the gap between the Tx and Rx electrode for a given button size. Higher mutual capacitance implies higher electric field lines between Tx and Rx electrode. Thus, we get a higher signal when we touch the button, because the finger touch the will disturb the electric field to a maximum extent. But higher Cm also increases the impact of external noise such as VDDA ripple noise. Thus there is a tradeoff in selecting the number of Rx prongs to get a higher signal verses getting good noise immunity. The optimal number of Rx prongs is 2 for the Fishbone pattern (i.e. Fishbone pattern with a single Tx prong and two Rx prongs). The below figure shows the mutual capacitance button with three and one number of Rx prongs.

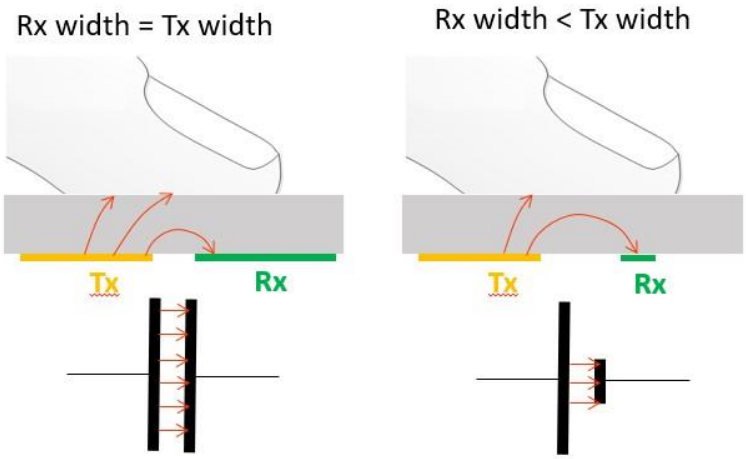
Figure 7-12 Mutual capacitance button with different number of prongs



7.4.3.2.3.6 Tx electrode and Rx electrode width

The Tx electrode and Rx electrode width influences the mutual capacitance between Tx and Rx electrode. Best signal response is achieved when Rx width/area is equal to Tx width/area in the case of less external noise in the system. The below figure shows the electric filed lines from Tx to Rx electrode with equal and unequal widths. Thus, from the below figure it is clear that having equal Tx and Rx width will eventually leads to higher change in Cm for a finger touch.

Figure 7-13 Effect of Tx and Rx width



In some cases, it is required to provide liquid tolerance to the mutual capacitance button as mentioned in section Liquid Tolerance for Mutual-Capacitance Sensing. To achieve that we have to use a hybrid sensing technique with both CSX and CSD sensing method. In such a case the Tx or Rx electrode whichever is scanned in CSD technique should have a significant width so that it ensures a good signal for a finger touch.

7.4.3.2.3.7 Co planar ground

Presence of coplanar ground decreases the impact of noise in the system and it also provides good ground resulting in decreased signal disparity effect. It is recommended to have as much area surrounding the sensor with hatched pattern and connected it to device ground. Also follow the recommendations as mentioned in the layout and schematics guidelines in this chapter. Ground plane reduces the coupling of electric field lines to the approaching finger, which decreases the change in mutual capacitance caused by a finger touch. It is suggested to avoid having ground plane underneath the sensor unless you expect strong coupling to a noise source present right below the sensor. Figure 7-7 shows the coplanar ground on the top and bottom layer of the PCB. The gap between the outer wall of the Tx electrode and the coplanar hatch ground should be greater than the air-gap of Tx and Rx electrodes.

7.4.3.2.3.8 Tx Wall (X-wall and Y-wall width)

Tx wall act as a shield to the Rx electrode from noise. Wide Tx wall also reduces the effect of cross talk and the impact of Co-planar ground. It is recommended to keep the Tx wall width approximately equal to half of Tx electrode width. The below figure shows the effect of wider Tx wall, it increases the number of electric field lines reaching the finger from the Tx electrode by reducing the impact of Coplanar ground. The width of Tx wall can also be slightly increased in case Tx electrode is scanned as a CSD sensor as mentioned in section Liquid Tolerance for Mutual-Capacitance Sensing. An example 10x10 pattern with increased Tx wall is given in Table 7-5.

Figure 7-14 Effect of width of Tx wall

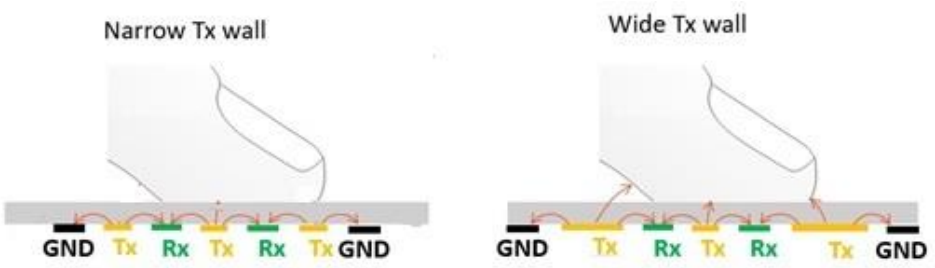


Table 7-5 Dimension of 10x10 Button with Increased Tx Wall (All Units in mm)

Button Size (X-Size, Y-Size) (mm)	Number of Rx-Prongs	Air Gap between Tx and Rx in mm	Tx Width in mm	Rx Width in mm	X-Wall Width in mm	Y-Wall Width in mm	Y Prong in mm
10, 10	2	0.8	1.2	1.2	1.5	1.6	0.2

7.4.4 Slider Design

Figure 7-15 shows the recommended slider pattern for a linear slider and Table 7-6 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 7-15. Typical Linear Slider Pattern

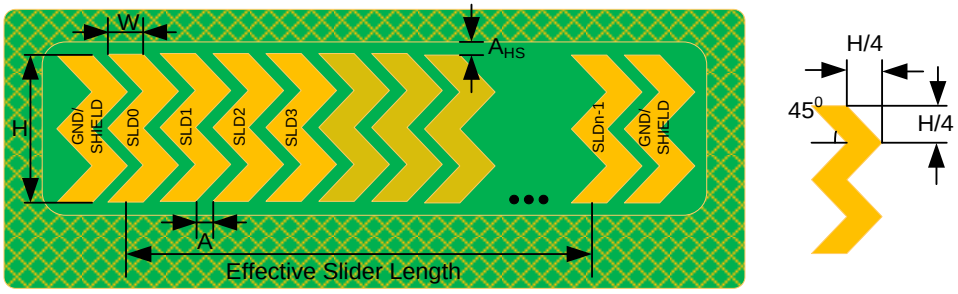


Table 7-6. Linear Slider Dimensions

Parameter	Acrylic Overlay Thickness	Minimum	Maximum	Recommended
Width of the segment (W)	1 mm	2 mm	-	8 mm[9]
	3 mm	4 mm	-	
	4 mm	6 mm	-	
Height of the segment (H)	-	7 mm[10]	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_{HS})	-	0.5 mm	2 mm	Equal to overlay thickness

7.4.4.1 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[11]. 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 7-15 shows. This shape helps in achieving a centroid response close to the ideal response, as Figure 7-16 and Figure 7-17 show. For the same reason, the slider-segment width and air gap (dimensions “W” and “A” respectively, as marked in Figure 7-15) should follow the relationship mentioned in Equation 7-1.

[9] The recommended slider-segment width is based on an average human finger diameter of 9 mm. See section Slider-Segment Shape, Width, and Air Gap section for more details.

[10] 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 ≥ 5:1 is achieved when the finger is placed in the middle of any segment.

[11] Here, a valid signal means that the difference count of the given slider segment is greater than or equal to the noise threshold value.

Figure 7-16. Ideal Slider Segment Signals and Centroid Response

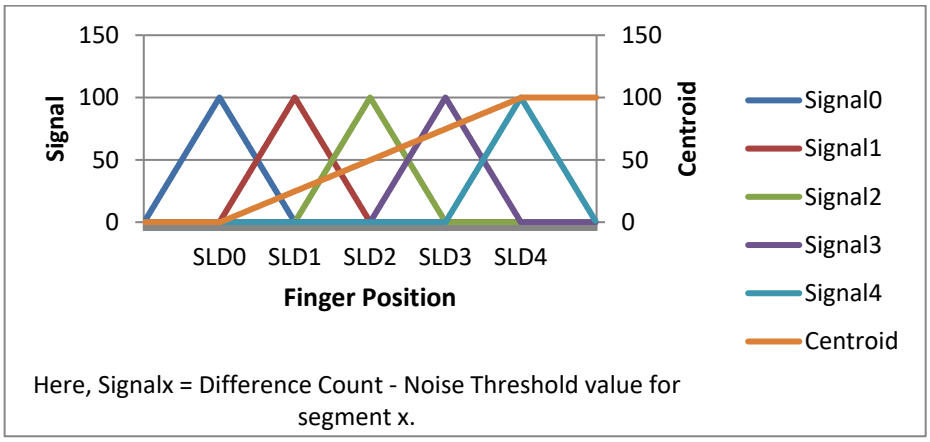
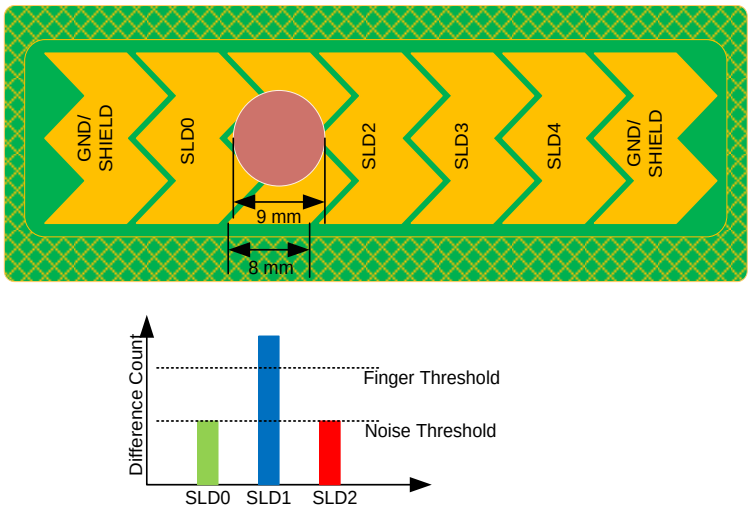


Figure 7-17. Ideal Slider Signals



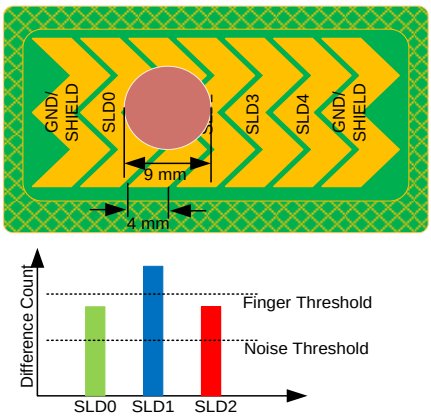
Equation 7-1. Segment width and air gap relation with the finger diameter

$$W + 2A = finger\ diameter$$

Typically, an average human finger diameter is approximately 9 mm. Based on this average finger diameter and Equation 7-1, the recommended slider-segment-width and air-gap is 8 mm and 0.5 mm respectively.

If the sum of *slider-segment-width* and *2 * air-gap* is lesser than *finger diameter*, as required according to Equation 7-1, 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 7-18 shows. Thus, calculated centroid position in Equation 7-2 will be non-linear as Figure 7-19 shows.

Figure 7-18. Finger Causes Valid Signal on More Than Two Segments When Slider Segment Width Is Lower Than Recommended



Equation 7-2. Centroid Algorithm used by CapSense Component in PSoC Creator

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

Where,

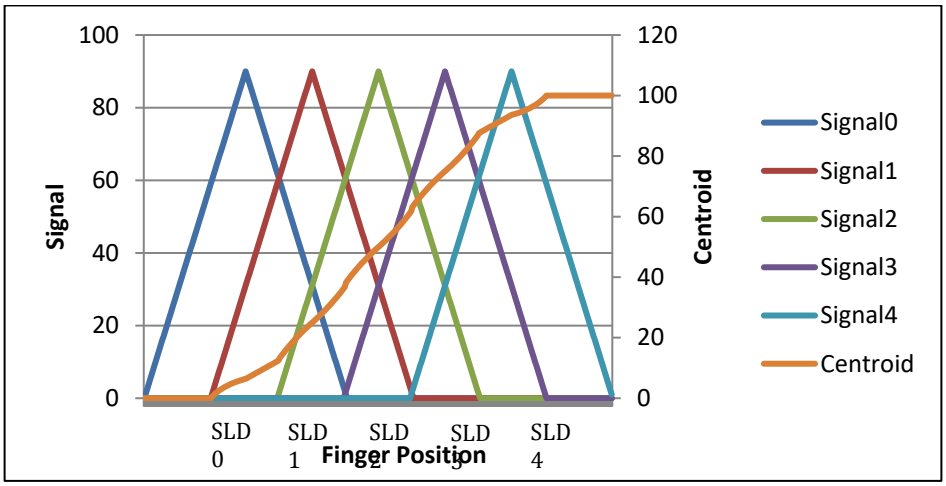
Resolution – API resolution set in the CapSense Component Customizer

n – Number of sensor elements in the CapSense Component Customizer

x – Index of element which gives maximum signal

S_i – different counts (with subtracted noise threshold value) of the slider segment

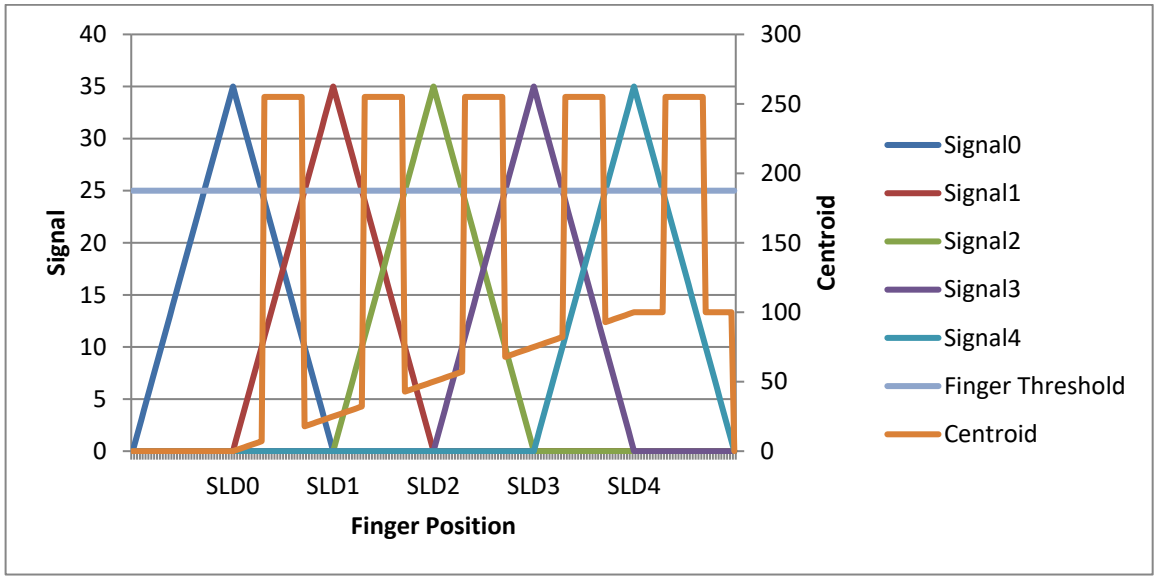
Figure 7-19. Nonlinear Centroid Response when Slider Segment Width Is Lower Than Recommended



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 7-19 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 ≥ 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 therefore, none of the slider-

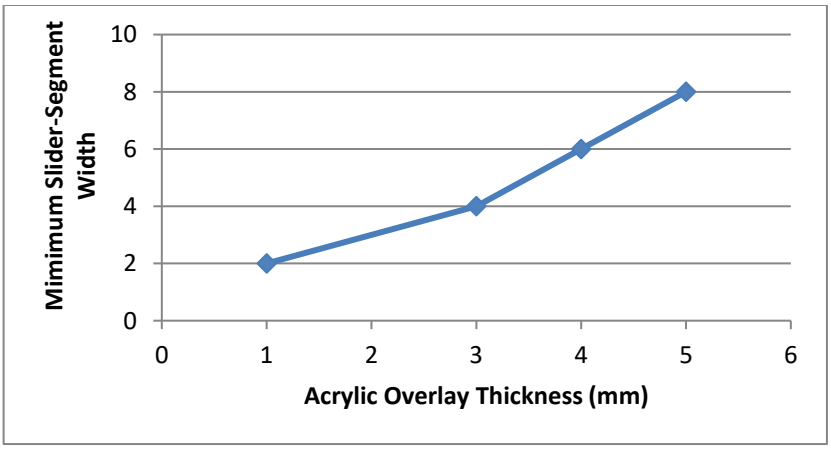
segments will have a 5:1 SNR, resulting in a reported centroid value of 0xFFFF[12] in PSoC Creator as Figure 7-20 shows, and 0x0000[13] in ModusToolbox.

Figure 7-20. 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 7-6. For thickness values of acrylic overlays, which are not specified in Table 7-6, Figure 7-21 may be used to estimate the minimum slider-segment width. Very thin overlay or no overlay may cause a nonlinear centroid response due to saturation of raw count or due to high finger capacitance; centroid position may be detected before touching the slider. In these conditions, the CapSense centroid algorithm will not be able to correctly estimate the finger position on the slider using Equation 7-2. It is recommended to have the overlay thickness for the CSD sensor as mentioned in Table 7-2.

Figure 7-21. 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 7-1, the centroid response will have flat spots; that is, if the finger is moved towards the middle of any segment, the reported centroid position

[12] The CapSense Component in PSoC creator reports a centroid of 0xFFFF 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.

[13] The CapSense middleware in ModusToolbox reports a centroid of 0x0000 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.

will remain constant as Figure 7-22 shows. This is because, as Figure 7-23 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 7-22. Flat Spots (Nonresponsive Centroid) when Slider-Segment Width Is Higher than Recommended

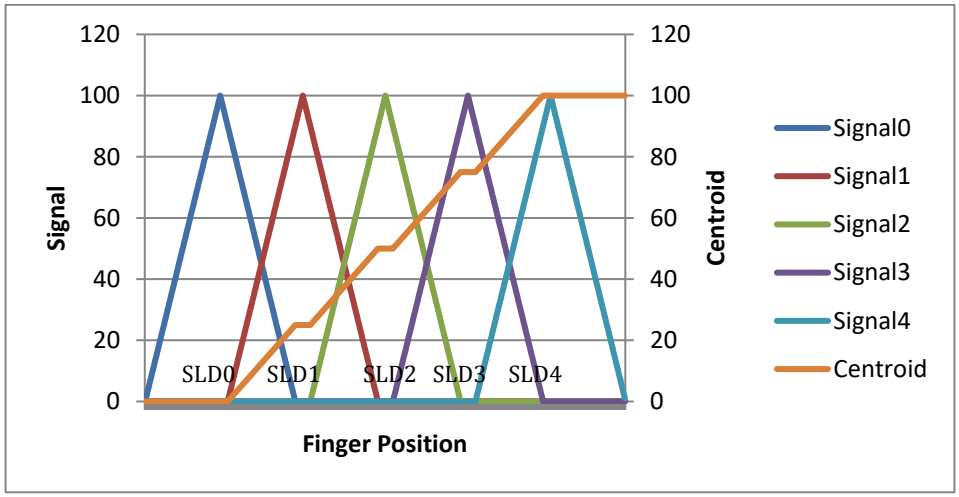
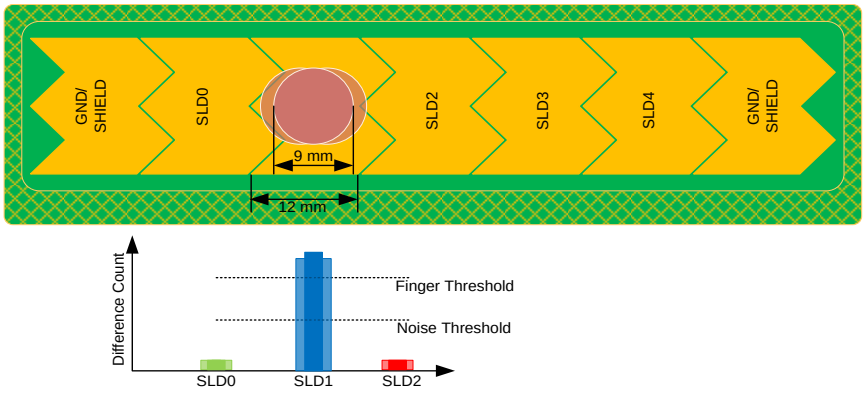


Figure 7-23. Signal on Slider Segments when Slider-Segment Width Is Higher than Recommended



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 7-16); 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.

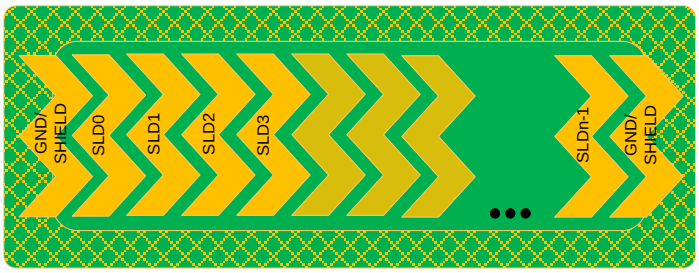
7.4.4.2 Dummy Segments at the Ends of a Slider

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 7-15 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 7-24 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 Self-Capacitance Sensor section for more details.

Figure 7-24. Linear Slider Pattern when First and Last Segments are Connected to Top Hatch Fill



7.4.4.3 Deciding Slider Dimensions

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

a. Decide the required length of the slider (L) based on application requirements. This is same as the “effective slider length” as Figure 7-15 shows.
b. 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 7-6.
c. The slider-segment width and the air gap between slider segments should be as recommended in Table 7-6. The recommended slider-segment-width and air-gap for an average finger diameter of 9 mm is 8 mm and 0.5 mm respectively.
d. For a given slider length *L*, calculate the number of segments required by using the following formula:

Equation 7-3: Number of Segments Required for a Slider

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

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 7-22 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 7-15 shows. PCB length can be related to the slider length as follows:

Equation 7-4. 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:

$$PCB\ length = Slider\ Length + Slider\ Segment\ Width$$

7.4.4.4 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 C_P as mentioned in the Component Datasheet.
2. C_P of the slider segment should be as close as possible. However, in the practical scenario C_P of each slider segment might vary because of differences in trace routing for each segment. The maximum allowed variation in the segment parasitic capacitance is 44% max C_P of the slider segment for an 85% IDAC calibration level. If the variation in C_P is beyond this limit then it may cause a change in the sensitivity among the slider segments leading to a non-linear slider response.

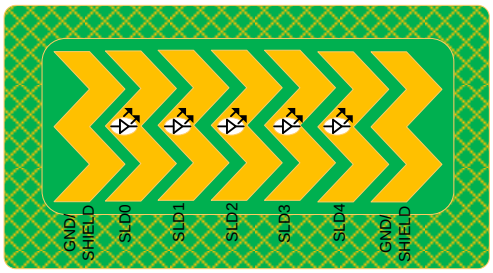
Implement the following layout design rules to meet a good slider design with linear response.

- 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 if possible.
- Maintain the same air gap between the sensors or traces to ground plane or hatch fill.

7.4.4.5 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 7-25 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. See Table 7-6 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 7-25. Slider Design with LED Backlighting



7.4.5 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.
- Avoid connectors between the sensor and the device pins because connectors increase C_P and noise pickup.
- Button to Button distance (edge to edge) must be greater than 8mm. If keys have less than 8mm between them, there will be cross talk between the keys. Also, from a usability standpoint, it increases the risk of the user touching two keys at the same time. Key to key distance must be greater than 8mm
- Spacing from a touch line to any metal should be greater than 5mm. This includes the metal chassis, decorative chrome trim, screws, etc.
- Isolate or provide physical separation between CapSense components and their signals from noisy subsystems such as transformers. A CapSense system in general is sensitive to external noise.

7.4.6 Trace Length and Width

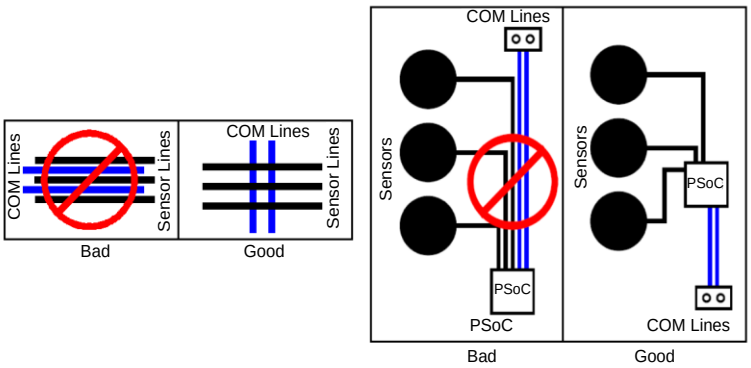
Use short and narrow PCB traces to minimize the parasitic capacitance of the sensor. The maximum recommended trace length is 12 inches (300 mm) for a standard PCB and 2 inches (50 mm) for flex circuits. The maximum recommended trace width is 7 mil (0.18 mm). You should surround the CapSense traces with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

7.4.7 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.

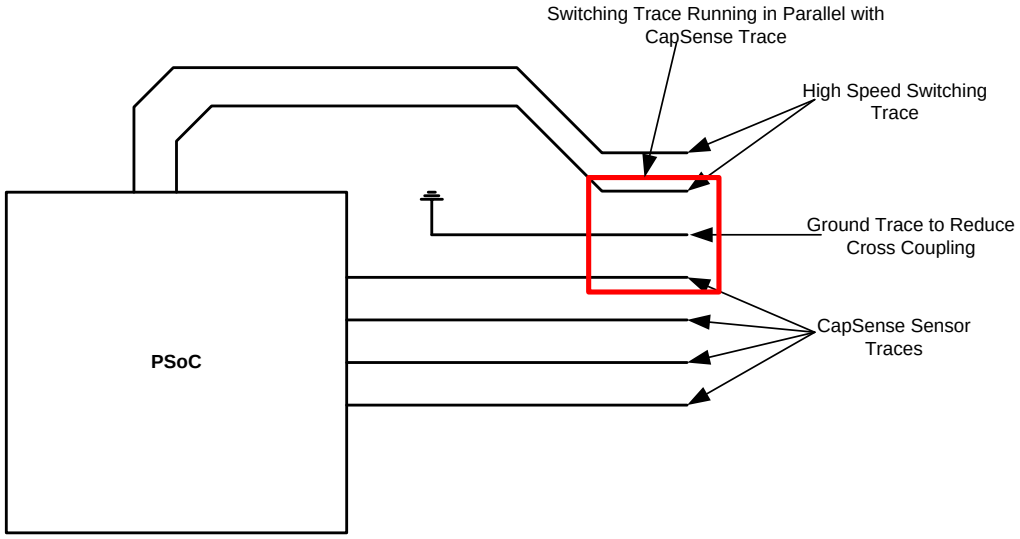
Do not run capacitive sensing traces closer than 0.25 mm to switching signals or communication lines. Increasing the distance between the sensing traces and other signals increases the noise immunity. If it is necessary to cross communication lines with sensor pins, make sure that the intersection is at right angles, as Figure 7-26 shows.

Figure 7-26. Routing of Sensor and Communication Lines



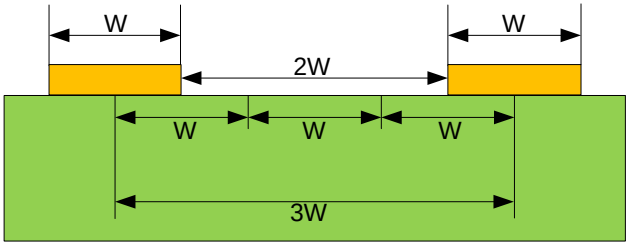
If, due to spacing constraints, sensor traces run in parallel with high-speed traces such as I^2C communication lines or BLE antenna traces, it is recommended to place a ground trace between the sensor trace and the high-speed trace as shown in Figure 7-27. This guideline also applies to the cross talk caused by CapSense sensor trace with precision analog trace such as 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_P of the CapSense sensor.

Figure 7-27. 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 7-28.

Figure 7-28. 3W Trace Spacing to Minimize Cross Talk



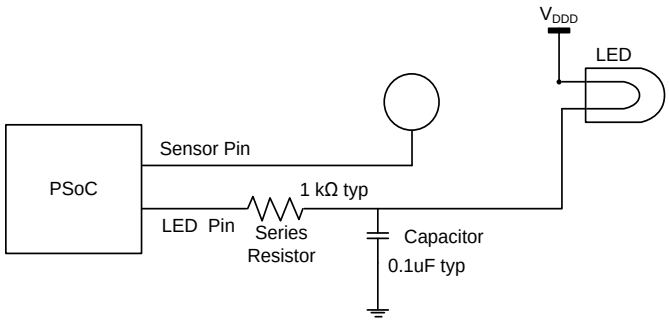
- Do not run Tx and Rx lines parallel to each other. The trace routing should be separated as much as possible.
- If the layout constraints require Tx and Rx run parallel for short distances, the space between Tx and Rx should be greater than the distance between Tx and Rx inside the key (2 times the Tx-Rx key spacing is preferred) or add ground between them.
- Keep as much clearance around Rx as possible to prevent noise on the touch keys. It is critical to follow this guideline for spacing to power traces and LED lines (high speed switching, power). Ground should also follow this rule, but it is less critical. Ground will provide noise protection but will reduce key sensitivity.
- For a given set of sensors, the number of Rx lines must be less than or equal to Tx lines. Rx lines are susceptible to noise, whereas Tx lines are relatively less susceptible

7.4.8 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 7-29 shows. Design the filter based on the required LED response speed.

Figure 7-29. Reducing Crosstalk



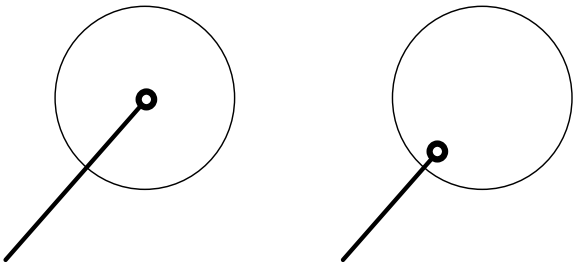
A guard trace is a ground trace running close to or above/below a TX/Rx line of a mutual-capacitance button. Guard traces can be used to protect sensor traces from noise if the layout does not allow for a ground hatch. Similar to ground hatch, guard traces add parasitic capacitance and reduce button sensitivity. Guard traces are usually needed on a case-by-case basis. Typical situations where guard traces have been used in the past include:

- Reduce cross talk
- Protect from noise of high speed lines (I2C, SPI, UART) and toggling LED traces.
- Border around the HMI or around a LCD

7.4.9 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 7-30 shows.

Figure 7-30. Via Placement on the Sensor Pad



Via at the Center of the Sensor (Long Trace)

Via Near the Edge of the Sensor (Short Trace)

7.4.10 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 or driven-shield 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).
- 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}, C_{INTX} must terminate at the central point
  - The ground plane for C_{SH_TANK} must terminate at the central point

Figure 7-31 explains the star connection. The central point for different families is mentioned in Table 7-7.

Figure 7-31. Star Connection for Ground

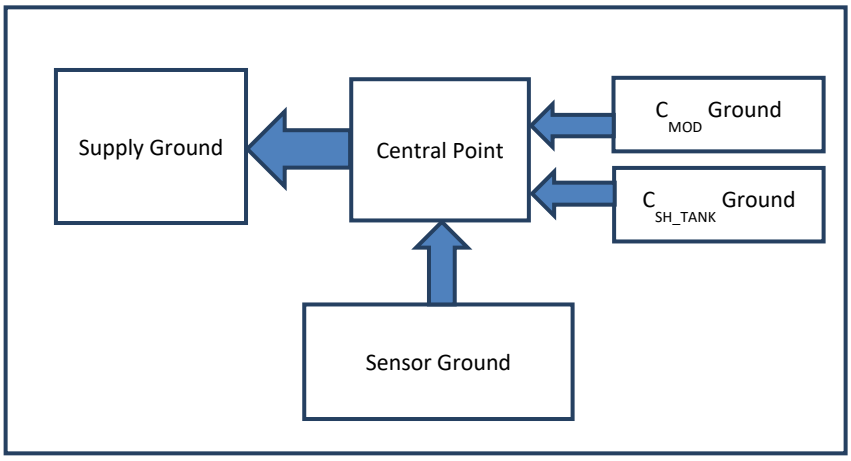


Table 7-7. Central Point for Star Connection

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

7.4.10.1 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 7-32 and Figure 7-33.

Figure 7-32. PCB Top Layer Layout Using a Chip Without E-pad

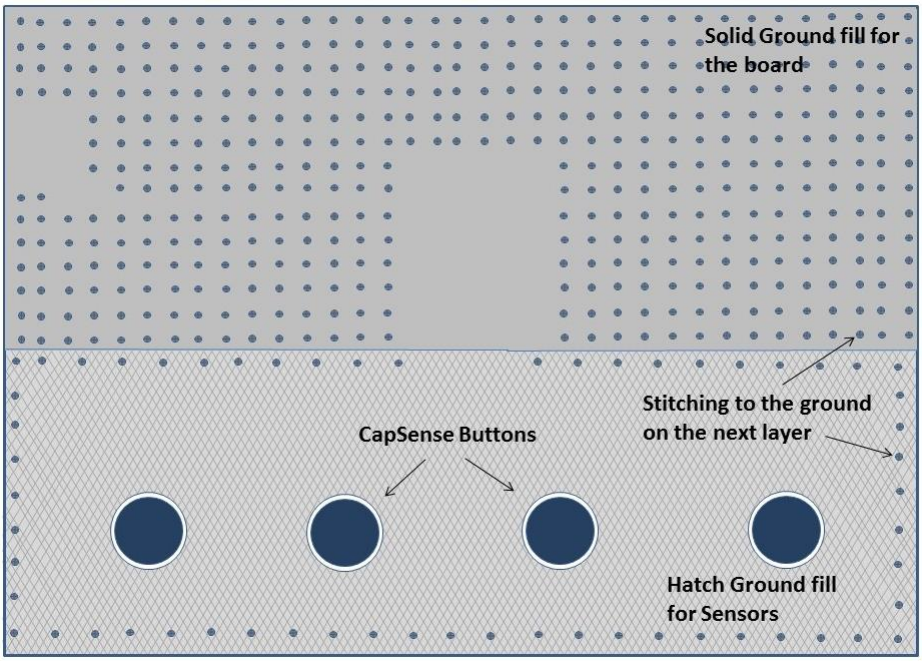
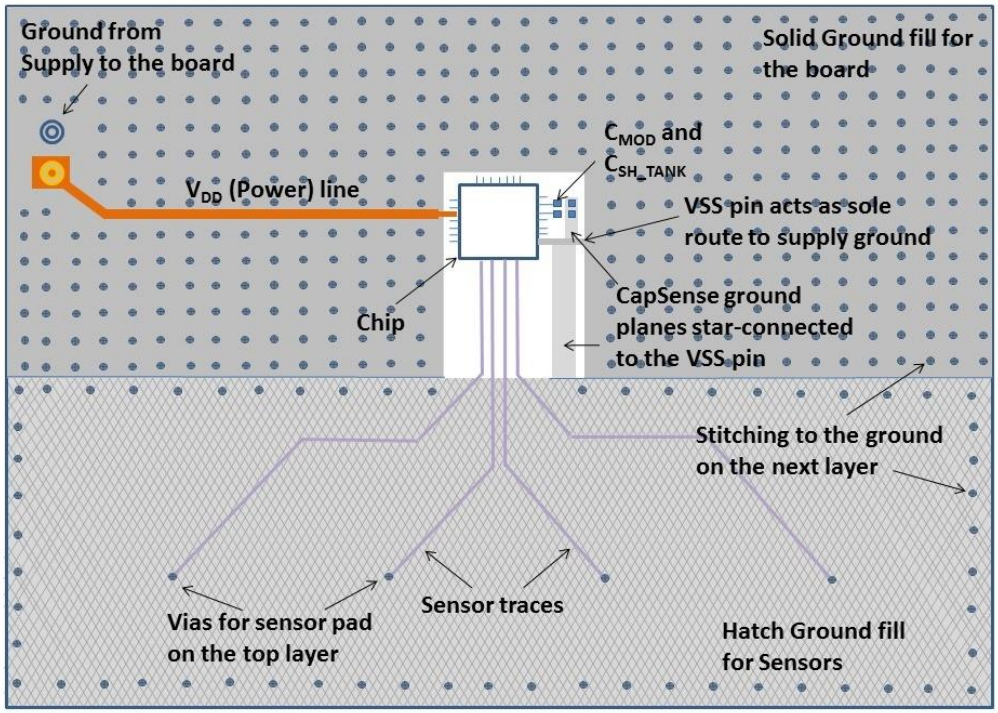


Figure 7-33. PCB Bottom Layer Layout Using a Chip Without E-pad



7.4.10.2 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.

7.4.10.3 Using PSoC 4 BLE Chips

In the case of PSoC 4 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 7.4.10.2 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 chips are shown in Figure 7-34 and Figure 7-35.

Figure 7-34. PCB Top Layer Layout with PSoC 4 BLE (with E-pad)

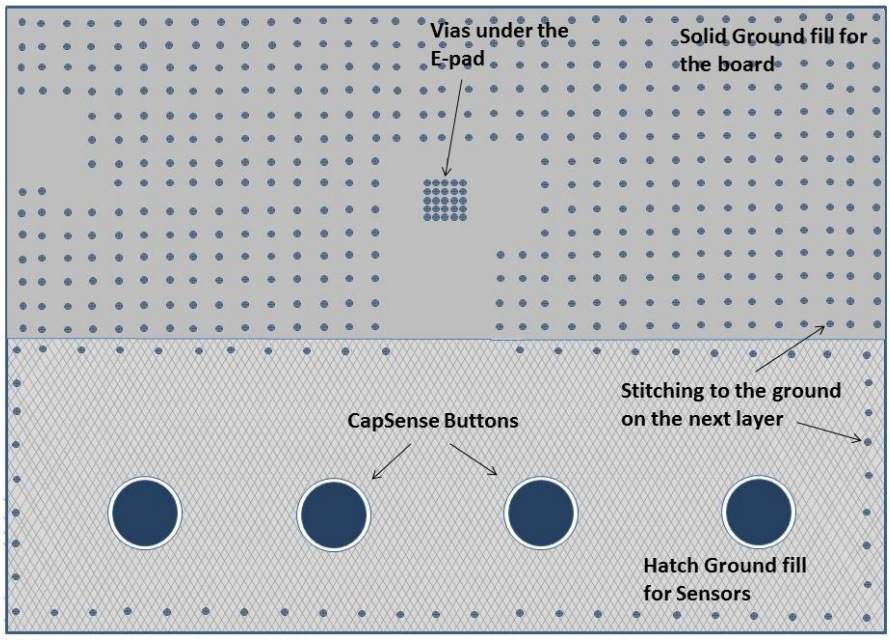
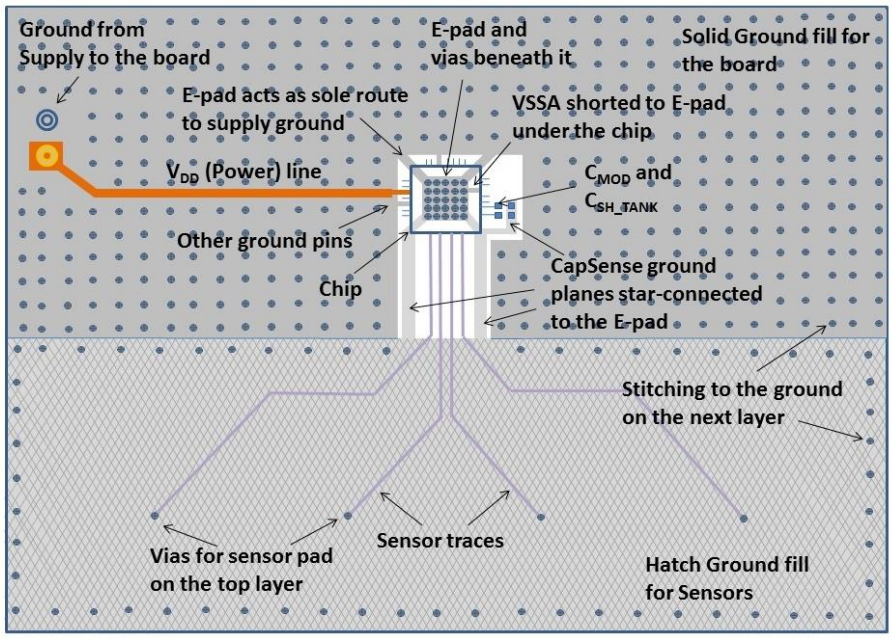


Figure 7-35. PCB Bottom Layer Layout with PSoC 4 BLE (with E-pad)



7.4.11 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 7-8.

Table 7-8. Decoupling Capacitors on Power Lines

Power Line	Decoupling Capacitors	Corresponding Ground Terminal	Applicable Device Family
VDD	0.1 µF and 1 µF	VSS	PSoC 4000
VDDIO	0.1 µF and 1 µF	VSS	PSoC 4000, PSoC 6 MCU
VDDD	0.1 µF and 1 µF	VSS	PSoC 4100, PSoC 4200, PSoC 6 MCU
	0.1 µF and 1 µF	VSSD	PSoC 4100-BL, PSoC 4200-BL, PSoC 4200 L, PSoC 4 S-Series, PSoC 4100S Plus
VDDA[14]	0.1 µF and 1 µF (Battery powered supply)	VSSA	PSoC 4100, PSoC 4200, PSoC 4100-BL, PSoC 4200-BL, PSoC 4200 L, PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, PSoC 6 MCU
	0.1µF and 10 µF (Mains Powered supply)	VSSA	PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS
VDDR	0.1 µF and 1 µF	VSSD	PSoC 4100-BL, PSoC 4200-BL, PSoC 6 MCU with BLE Connectivity
VCCD	See device datasheet	VSS (PSoC 4000) or VSSD (all others)	All device families

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, see the Power section in the device Device Datasheet.

7.4.12 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. If there are multiple CSD blocks in the device, each CSD block should have a dedicated shield electrode. This section shows how to implement a shield electrode and a guard sensor.

7.4.12.1 Layout Guidelines for Shield Electrode

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 7-36 shows. The board design should focus on reducing the coupling capacitance between the liquid droplet and ground. Thus, 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 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.

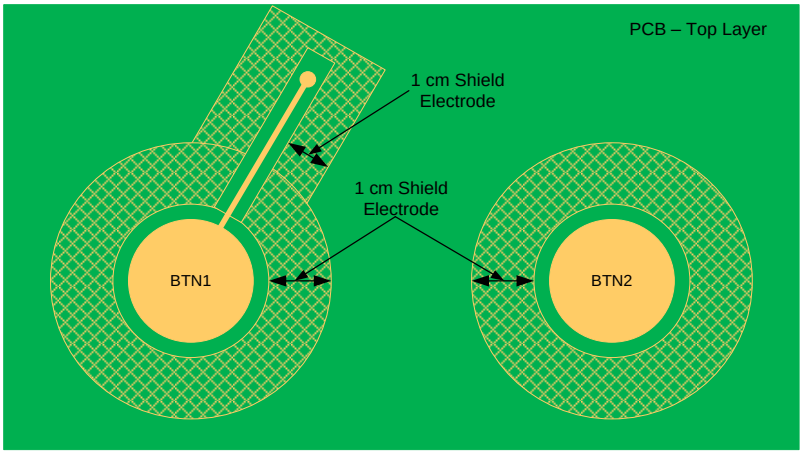
In some applications, the capacitance of the shield electrode will be very high; you can reduce it with the following techniques:

- *Using multiple shield electrode instead of single shield electrode:* If there is a single hatch pattern with a higher C_P, split the hatch pattern into multiple hatch patterns and drive it with the shield signal to decrease the shield C_P. This will also allow the use of a higher range of sense clock frequencies for the sensors which will improve the sensitivity of the CapSense system. In a complex layout design, this approach will make trace routing simple.

[14] The V_{DDA} pin on PSoC 4 S-Series, PSoC 4100S Plus, and PSoC 4100PS family 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-µF and 1-µF capacitors in parallel and if the device is mains powered, it is recommended to use 0.1 µF and 10 µF in parallel. This is to improve the power supply rejection ratio of reference generator (REFGEN) used in the CapSense block.

- *Connecting multiple shield pins to the same electrode:* If splitting the shield electrode in the layout is not feasible, connect multiple shield pins to the same electrode. This will make all the series resistance of the sensor pins in parallel and reduce the effective time constant of the shield electrode, which will allow using a higher range of sense clock frequencies for sensors, which will improve the sensitivity of the CapSense system.

Figure 7-36. 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:

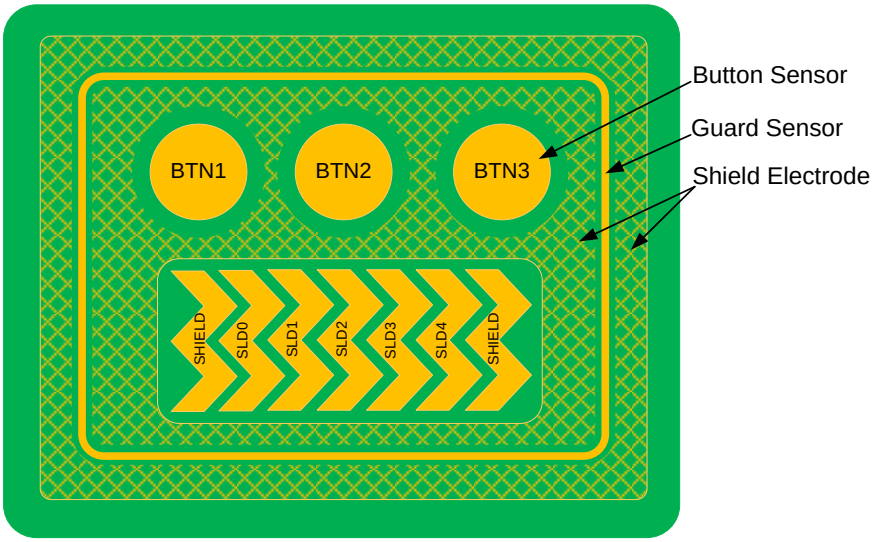
- Top layer: Hatch fill with 7-mil trace and 45-mil grid (25 percent fill). Hatch fill should be connected to the driven-shield signal.
- Layer-2: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to the driven-shield signal.
- Layer-3: VDD Plane
- Bottom layer: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

7.4.12.2 Layout guidelines for Guard Sensor

As explained in the Guard Sensor section, the guard sensor is a copper trace that surrounds all sensors, as Figure 7-37 shows.

Figure 7-37. 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 depending on the use case data.

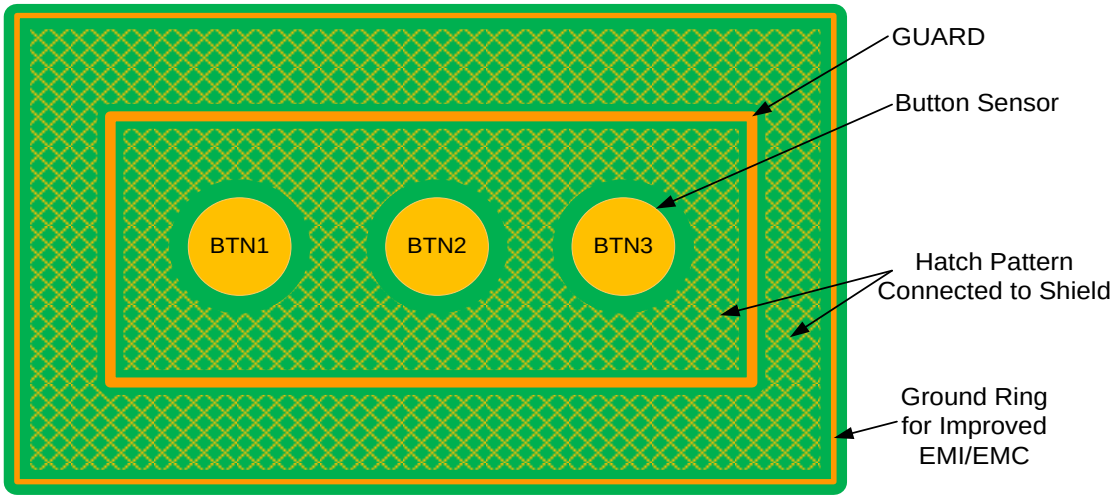
The following conditions can be used to detect a liquid stream on the touch surface:

- 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.
- A firmware algorithm to detect the false touch due to water drop from the use case data can be made to improve the false touch rejection capability sensors.

7.4.12.3 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 7-38. 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 7-38. CapSense Design with a Ground Ring for Improved ESD and EMI/EMC Performance



7.4.13 Schematic Rule Checklist

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

Table 7-9. Schematic Rule Checklist

No.	Category	Recommendations/Remarks
1	C_{MOD}	2.2 nF. See Table 7-10 for pin selection.
2	C_{SH_TANK}	10 nF if shield electrode is being used, NA otherwise. See Driven-Shield Signal and Shield Electrode and CapSense CSD Shielding for details on shield electrode and use of C_{SH_TANK} respectively. See Table 7-10 for pin selection.
3	C_{INTA}/C_{INTB}	470 pF. See Table 7-10 for pin selection.
3	Series resistance on input lines	560 Ω for Self-capacitance and 2 kΩ for Mutual-capacitance. See Series Resistors on CapSense Pinsfor 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^2C pins from the CapSense pins by a full port wherever possible. See Sensor Pin Selection section for more 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.

7.4.13.1 External Capacitors Pin Selection

As explained in the CapSense Fundamentals section, CapSense require external capacitors - C_{MOD} (CSD sensing method), C_{TANK} (Only when Shield is implemented), and C_{INTX} (CSX sensing method) for reliable operation. Starting from PSoC Creator 3.3 SP2, the number of pins that can support C_{MOD} and C_{SH_TANK} is increased to improve design flexibility. Table 7-10 lists the recommended pins for C_{MOD}, C_{INTX} and C_{SH_TANK} capacitors for PSoC Creator 3.3 SP2 or later versions.

Note For PSoC 4100/PSoC 4200, if a pin other than P4[2] is selected for C_{MOD}, P4[2] 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].

Table 7-10. Recommended Pins for External Capacitors

Device	C_{MOD}	C_{SH_TANK}
PSoC 4000	P0[4]	P0[2]
PSoC 4100/ PSoC 4200	P4[2]	P4[3]

Device	C_{MOD}	C_{SH_TANK}
PSoC 4200M/ PSoC 4200L	CSD0: P4[2]	CSD0: P4[3]
	CSD1: P5[0]	CSD1: P5[1]
PSoC 4 BLE	P4[0]	P4[1]
PSoC 6 MCU	P7[1]	P7[2]
PSoC 4 S-Series, PSoC 4100S Plus	P4[2]	P4[3]
PSoC 4100PS	P5[2]	P5[3]

Table 7-11. Supported Pins for External Capacitors

Device	C_{MOD}	C_{SH_TANK}	C_{INTA}	C_{INTB}
PSoC 4000	Port0[0:7], Port1 [0:7] P2[0]	Port0 [0:7], Port1 [0:7] P2[0]	P0[4]	P0[2]
PSoC 4100	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] P4[2]	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] P4[3]	Not Supported	Not Supported
PSoC 4200	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] P4[2]	Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7] P4[3]	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7]	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7]
PSoC 4200M	CSD0: Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] Port4 [0:6], Port6 [0:5] Port7 [0:1]	CSD0: Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7], Port4 [0:6], Port6 [0:5] Port7 [0:1]	CSD0: P4[2]	CSD0: P4[3]
	CSD1: Not Supported	CSD1: Not Supported	CSD1: Not Supported	CSD1: Not Supported
PSoC 4200L	CSD0: Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] Port4 [0:6], Port6 [0:5] Port7 [0:7], Port10 [0:7] Port11 [0:7]	CSD0: Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] Port4 [0:6], Port6 [0:5] Port7 [0:7], Port10 [0:7] Port11 [0:7]	CSD0: P4[2]	CSD0: P4[3]
	CSD1: Port5 [0:7], Port8 [0:7] Port9 [0:7]	CSD1: Port5 [0:7], Port8 [0:7] Port9 [0:7]	CSD1: P5[0]	CSD1: P5[1]
PSoC 4 BLE	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] Port4 [0:1], Port5 [0:1] Port6 [0:1]	Port0 [0:7], Port1 [0:7] Port2 [0:7], Port3 [0:7] Port4 [0:1], Port5 [0:1] Port6 [0:1]	P4[0]	P4[1]
PSoC 6 MCU	P7[1] or P7[2] or P7[7]	P7[1] or P7[2] or P7[7]	P7[1]	P7[2]
PSoC 4 S-Series, PSoC 4100S Plus	P4[2], P4[3], P4[1]	P4[2], P4[3], P4[1]	P4[2]	P4[3]
PSoC 4100PS	P5[0], P5[2], P5[3]	P5[0], P5[2], P5[3]	P5[2]	P5[3]

7.4.13.2 Sensor Pin Selection

As explained in CapSense Fundamentals, PSoC supports CSD and CSX capacitive sensing methods. Each CSD sensor requires a single sensor pin and CSX sensor will require two sensor pins for Tx and Rx electrode in addition to the required external capacitors for each sensing technique.

The selection of the sensor pins should be in a way such that the CapSense sensor traces and communication or other toggling GPIO traces are isolated by proper port/pin assignment. The following are some recommended guidelines:

- Isolate switching signals, such as PWM, I2C communication lines, and LEDs from the sensor and sensor traces. Place them at least 4 mm apart and fill a hatched ground between the CapSense traces and the switching signals to avoid crosstalk.
- Distribute the placement of DC loads on different ports to reduce the noise in CapSense. It is recommended to have digital I/Os spread on different ports rather than concentrating in a single port.
- While the CapSense block is scanning the sensor, 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 while the CapSense sensor is scanning may result in excessive noise in the sensor raw count.
- For a PSoC 4 device it is recommended to place all the digital DC loads like LEDs, I2C/UART communication pins on the port powered by only VSSD; see the Device Datasheet for determining the ports that are powered by VSSD. Placing DC loads on ports powered by VSSA will shift the VSSA up. Since CapSense is powered by VSSA, it will affect its performance.
- For PSoC 6 family of devices:
  - Table 7-12 lists the ports that support CapSense. Selecting ports 5, 6, 7, and 8 for CapSense ensures lesser noise.
  - It is recommended to place all digital switching pins such as LEDs, I2C, UART, SPI, SMIF communication pins on the ports that are powered by a different power supply domain which is not shared with the CapSense ports. Table 7-13 lists the ports, their supply domains, and recommendations for using these ports with CapSense. For more details, see the Errata section of the Device Datasheet. A deviation from these guidelines might cause a noise due to level shift in raw count. For more details, see Raw counts show a level-shift or increased noise when GPIOs are toggled. To isolate the supply domains further, it is better to externally isolate them using ferrite beads as shown in Figure 7-40.

Table 7-12. CapSense Capable Ports in PSoC 6 Devices

Device	CapSense Capable Ports
CY8C62x6, CY8C62x7	P0, P1, P2, P4, P5, P6, P7, P8, P9, P10, P11
CY8C63x6, CY8C63x7	P0, P1, P2, P4, P5, P6, P7, P8, P9, P10, P11
CY8C62x5	P7.0 to P7.7, P8.0 to P8.3, P9.0 to P9.3

Table 7-13. Recommendations of Port Usage with CapSense for PSoC 6 device

Ports	Supply Domain	Recommended for CapSense	Recommendations for GPIOs if used for Communication, LEDs, and Other High Frequency Functionality with CapSense
P0	V_{BACKUP}	No*	Switching frequency < 8MHz
P1	V_{DDD}	No*	Switching frequency < 1MHz, SLOW Slew Rate
P2, P3, P4	V_{DDIO2}	No*	Switching frequency < 25MHz
P5, P6, P7, P8	V_{DDIO1}	Yes	Not recommended
P9, P10	V_{DDIOA}	No*	Switching frequency < 1MHz, SLOW Slew Rate
P11, P12, P13	V_{DDIO0}	No*	Switching frequency < 80MHz
P14	V_{DDUSB}	No*	NA

* If you need additional CapSense pins and if you must use GPIOs in ports P1, P9, and P10 as Tx electrode for CSX sensor, restrict the Tx clock frequency within 1 MHz and use SLOW slew rate. Figure 7-39 shows an example on how to select the **Slew Rate** of the GPIO using the Device Configurator in the ModusToolbox

project. Note that using the ports other than the recommended ports for CapSense might cause higher noise in raw count.

Figure 7-39. Selecting Slew Rate for GPIOs

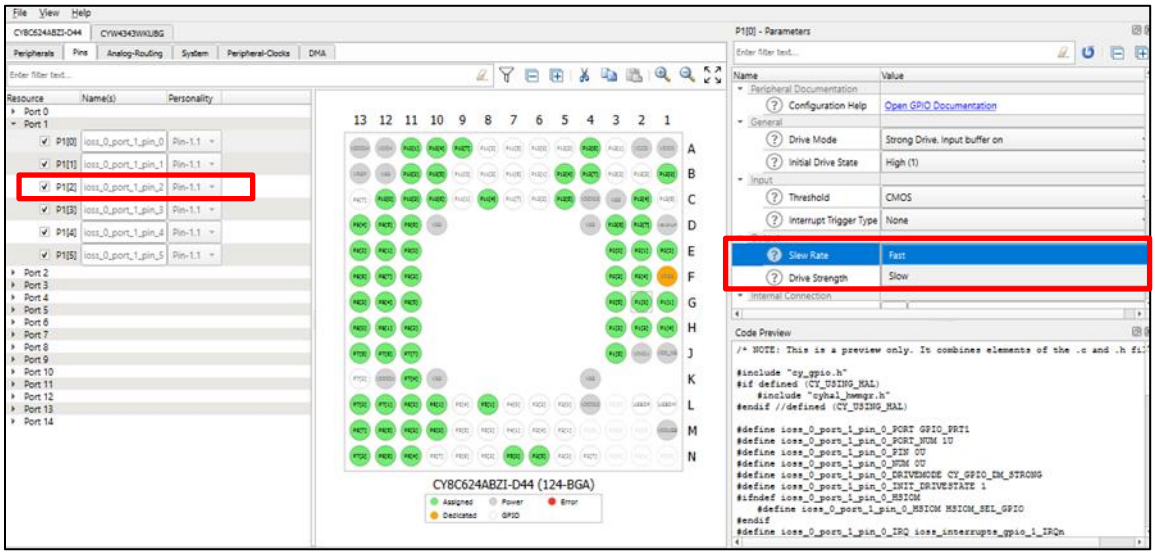
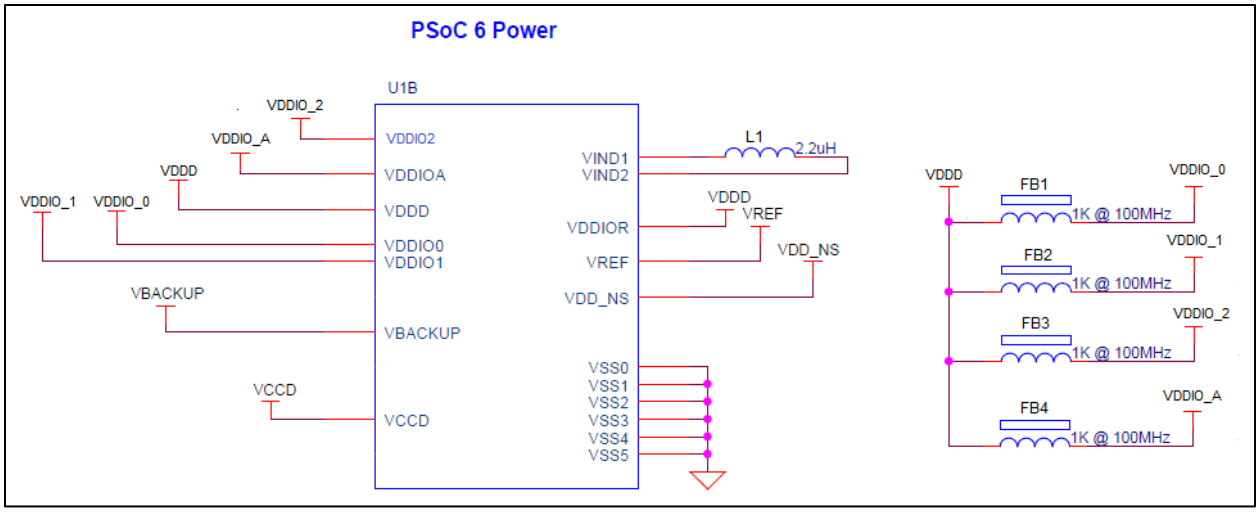


Figure 7-40. Externally Isolated Supply Domains



7.4.14 Layout Rule Checklist

You can use the checklist provided in Table 7-14 to help verify your layout design.

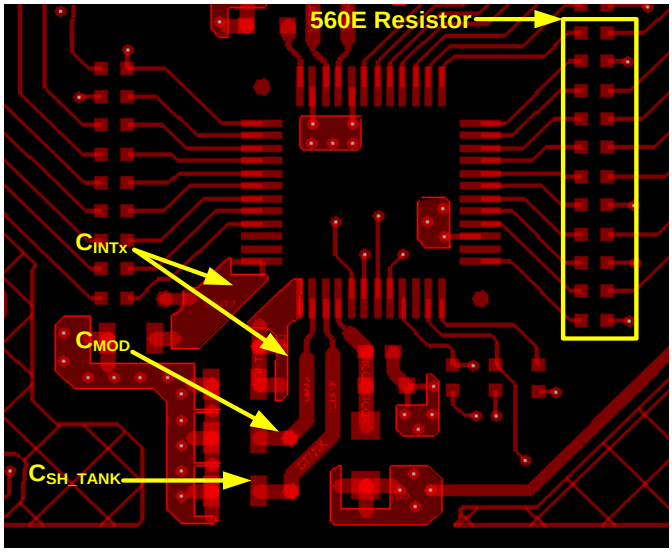
Table 7-14. Layout Rule Checklist

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

No.	Category		Minimum Value	Maximum Value	Recommendations / Remarks
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.
		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. See Figure 7-41 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	CMOD	Placement	N/A	N/A	Place close to the PSoC pin. See Figure 7-41 for an example placement of CMOD on printed circuit board.
11	CSH_TANK	Placement	N/A	N/A	Place close to the PSoC pin. See Figure 7-41 for an example placement of CSH_TANK on board.
12	CINTA	Placement	N/A	N/A	Place close to the PSoC pin. See Figure 7-41 for an example placement of CINTA on the PCB.

No.	Category		Minimum Value	Maximum Value	Recommendations / Remarks
13	CINTB	Placement	N/A	N/A	Place close to the PSoC pin. See Figure 7-41 for an example placement of CINTA on the PCB.

Figure 7-41. Example Placement for C_{MOD}, C_{INTx}, C_{SH_TANK}, and Series Resistance on Input Lines in PSoC 4200M Device



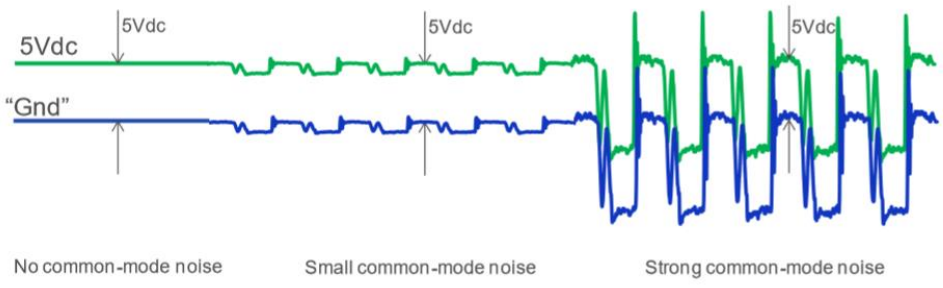
7.5 Noise in CapSense System

7.5.1 Finger Injected Noise

If the power supply design of the system is poor, the power and ground supplies of a device fluctuates in voltage relative to the finger ground (earth ground) in a common mode fashion. This type of noise is called common mode noise. Figure 7-42 illustrates the common mode noise, where both the 5V and the 0V output leads of the power supply remain 5V from each other, but they move up and down together, in a “common mode” manner.

This is not a problem, until a finger touch occurs on the button. A finger touch on the button introduces a (capacitive) path to the same earth ground and it will create a path for charge flow, which is equivalent to a noise signal injected exactly at the finger touch location. This injected noise caused by the common mode noise in power supply is called finger injected noise. It is observed only during the finger touch on the button in AC powered application and it doesn’t occur in battery powered application.

Figure 7-42. Common Mode Noise in the Power Supply



Note that when the complete system powered by AC supply is held in hand of the user, the entire system will be grounded to earth sufficiently and no significant “common-mode” noise would flow through the touching finger to earth. However, if the system is connected to the power supply and placed on a desk, a touch on the button, can introduce a problematic discharge path to ground.

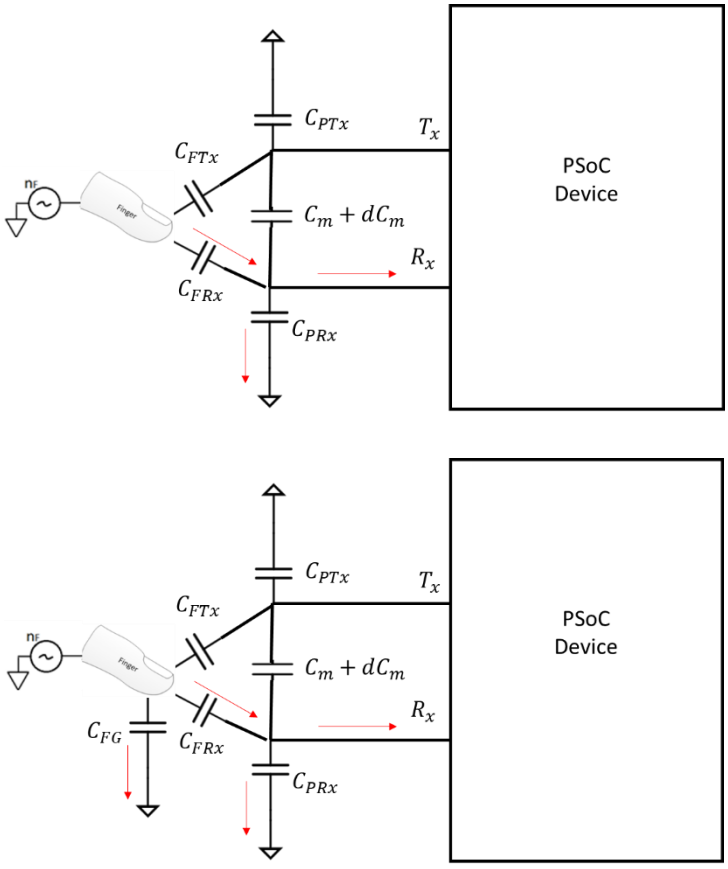
7.5.1.1 Recommendations to Reduce the Finger Injected Noise

The finger injected noise could be reduced by properly following the layout and schematics guidelines described in this section. The general recommendations to reduce the finger injected noise is explained below.

a. Fill the PCB board around the button with hatched pattern and connected it to device ground. Follow the recommendations as mentioned in the section Ground Plane.

   The Figure 7-43 shows the impact of ground on the finger injected noise for mutual capacitance button and it is also true for CSD sensing technique. In the left figure, the system doesn’t have the hatched ground around the button and most of the injected noise through the finger pass to the Rx pin of the device through the Capacitance formed between the finger and Rx electrode. In the right figure, the system has the hatched ground around the button and thus the finger injected noise is having an alternate path to flow which results in the reduction of the noise reaching to the device Rx pin.

Figure 7-43. Effect of Ground on Finger Injected Noise



b. Better power supply design of the system could easily eliminate the common mode noise, which will in turn reduce the finger injected noise.

c. Use software technique that are available in the CapSense component to combat the finger injected noise such as selecting optimal sensing clock frequency and Multi frequency scanning etc.

d. Increase the overlay thickness will reduce the finger injected noise as it will decrease the capacitance formed between the finger and Rx electrode.

7.5.2 VDDA Noise

The noise in the system due to unwanted voltage ripples in the VDD supply is called VDDA noise.

7.5.2.1 Recommendations to Reduce the VDDA Noise

The VDDA noise could be reduced by properly following the layout and schematics guidelines in this chapter. The general recommendations to reduce the VDDA noise as follows:

a. Use clean power supply and have VDD ripples below the limits mentioned in the device datasheet.
b. Use filters or LDO regulator in the VDD power lines.
c. Use decoupling capacitors on the power supply pins to reduce the conducted noise from the power supply.
d. To reduce high-frequency noise, place a ferrite bead around power supply or communication lines.
e. Selecting the proper supply configuration as mentioned in the Power section in the Device Datasheet and using the internal regulator to the device might help in reducing the VDDA noise.

7.5.3 External Noise

Any noise that is injected into to the system through the routing trace lines like ESD, EMI, conducted noise are coming into the category of external noise. The recommended guidelines for reducing the impact of the external noise are discussed in this section.

7.5.3.1 ESD Protection

The nonconductive overlay material used in CapSense provides inherent protection against ESD. Table 7-15 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 7-15. Overlay Thickness for ESD Protection

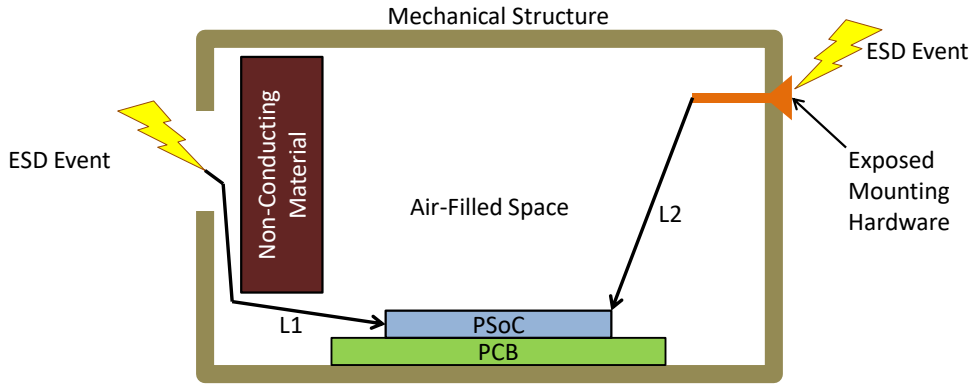
Material	Breakdown Voltage (V/mm)	Minimum Overlay Thickness for Protection Against 12 kV ESD (mm)
Air	1200 – 2800	10
Wood – dry	3900	3
Glass – common	7900	1.5
Glass – Borosilicate (Pyrex®)	13,000	0.9
PMMA Plastic (Plexiglas®)	13,000	0.9
ABS	16,000	0.8
Polycarbonate (Lexan®)	16,000	0.8
Formica	18,000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

7.5.3.1.1 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 7-44, if L1 and L2 are greater than 10 mm, the system can withstand a 12-kV ESD.

Figure 7-44. ESD Paths

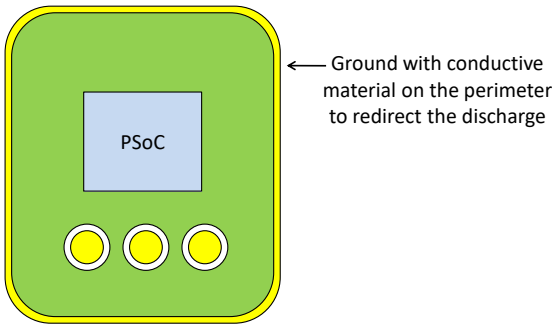


If it is not possible to maintain adequate distance, place a protective layer of nonconductive material with a high breakdown voltage between the possible ESD source and PSoC. One layer of 5-mil thick Kapton® tape can withstand 18 kV. See Table 7-15 for other material dielectric strengths.

7.5.3.1.2 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 7-45 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 7-45. Ground Ring



7.5.3.1.3 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 7-16 lists the recommended ESD protection devices.

Table 7-16. 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 µA	±15 kV	±16 kV
NXP	NUP1301	0.75 pF	30 nA	8 kV	15 kV

7.5.3.2 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.

7.5.3.2.1 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.

7.5.3.2.1.1 Hardware Considerations

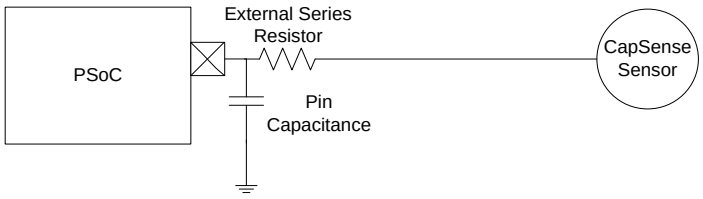
7.5.3.2.1.1.1 Ground Plane

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.

7.5.3.2.1.1.2 Series Resistors on CapSense Pins

Every CapSense controller pin has some parasitic capacitance, C_P, associated with it. As Figure 7-46 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 7-46. RC Filter



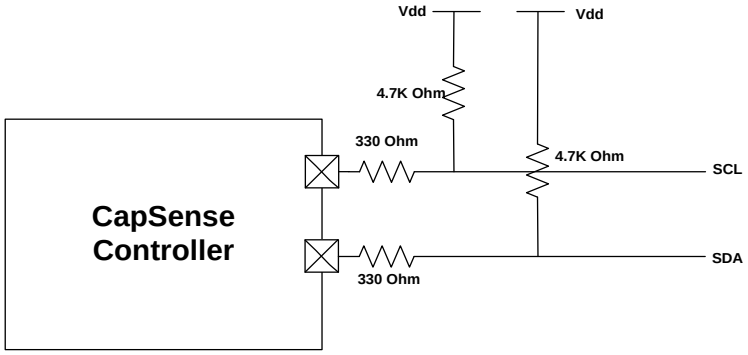
Series resistors should be placed close to the device pins so that the radiated noise picked by the traces gets filtered at the input of the device. Thus, it is recommended to place series resistors within 10 mm of the pins.

For CapSense designs using copper on PCBs, the recommended series resistance for CapSense input lines is 560 Ω. 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 Ω, 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.

7.5.3.2.1.1.3 Series Resistors on Digital Communication Lines

Communication lines, such as I^2C and SPI, also benefit from series resistance; 330 Ω 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^2C communication lines is 4.7 kΩ. So, if more than 330 Ω is placed in series on these lines, the V_{IL} and V_{IH} voltage levels may fall out of specifications. 330 Ω will not affect I^2C operation as the V_{IL} level still remains within the I^2C specification limit of 0.3 V_{DD} when PSoC outputs a LOW.

Figure 7-47. Series Resistors on Communication Lines



7.5.3.2.1.1.4 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.

7.5.3.2.1.1.5 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 7-48 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 7-49 shows.

Figure 7-48. Improper Current Loop Layout

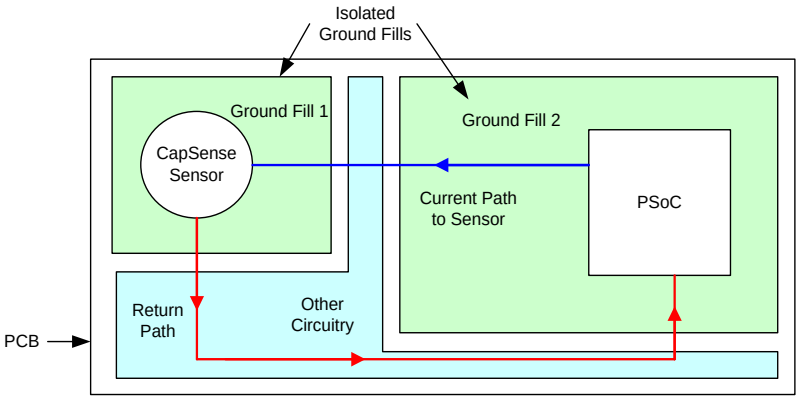
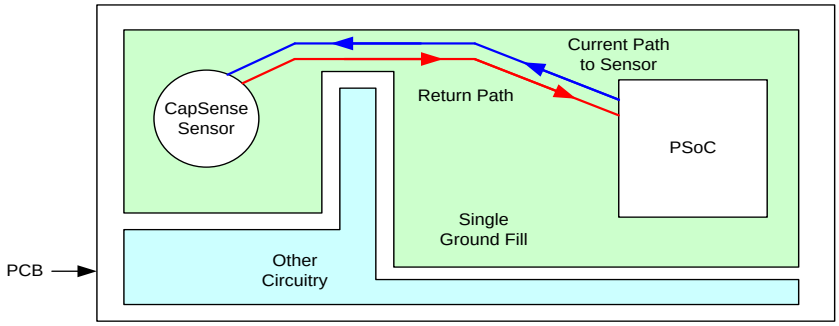


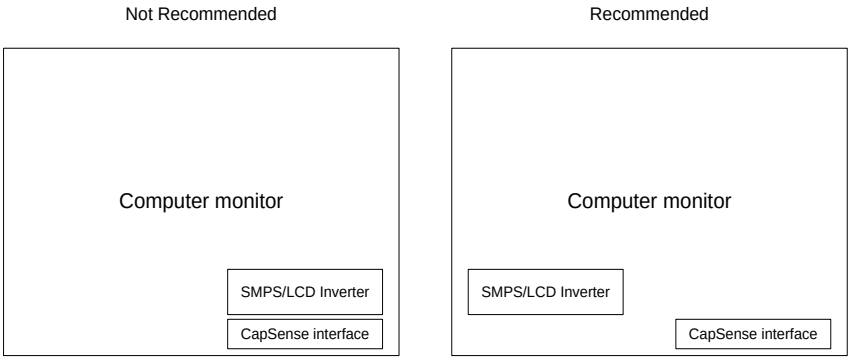
Figure 7-49. Proper Current Loop Layout



7.5.3.2.1.1.6 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 7-50 shows an example of separating the RF noise source from the CapSense interface.

Figure 7-50. Separating Noise Sources



7.5.3.2.1.2 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
- Sense Clock Source Inactive sensor termination

The following sections explain the effect of each parameter.

7.5.3.2.1.2.1 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 7-51 and Figure 7-52 show the impact of operating voltage on radiated emissions. 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 7-51. Effect of V_{DD} on Radiated Emissions (150 kHz – 30 MHz)

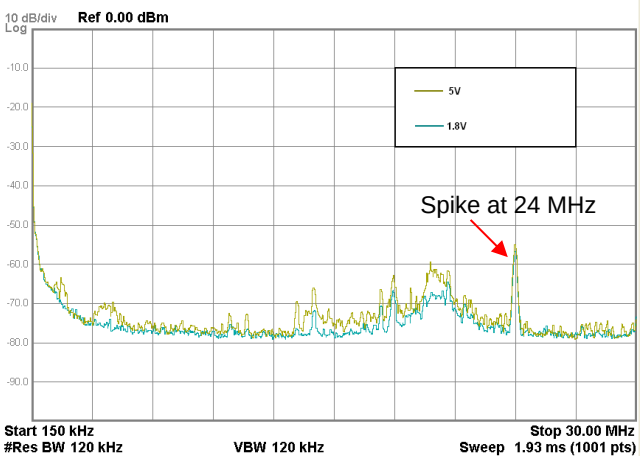
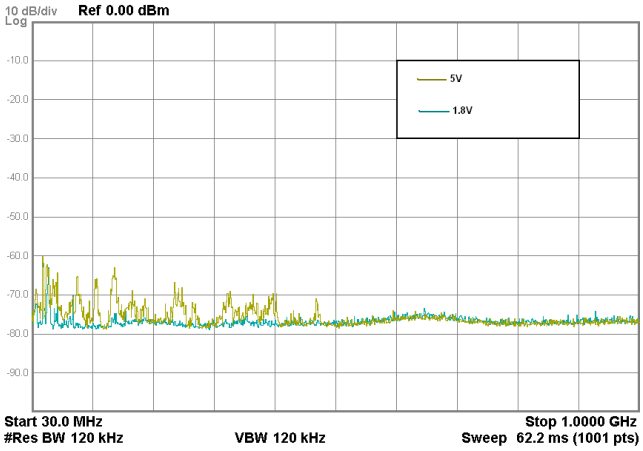


Figure 7-52. Effect of V_{DD} on Radiated Emissions (30 MHz – 1 GHz)



Note: Frequency axis is in log scale.

7.5.3.2.1.2.2 Device Operating Frequency

Reducing the system clock frequency (IMO frequency) reduces radiated emissions. However, reducing the IMO frequency may not 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.

7.5.3.2.1.2.3 Sensor-Switching Frequency

Reducing the sensor-switching frequency (see Sense Clock) also helps to reduce radiated emissions. See Figure 7-53 and Figure 7-54. 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 7-53. Effect of Sensor-Switching Frequency on Radiated Emissions (150 kHz – 30 MHz)

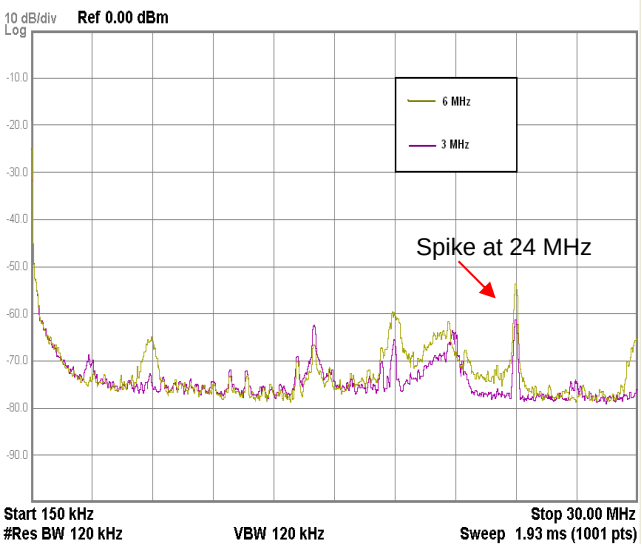
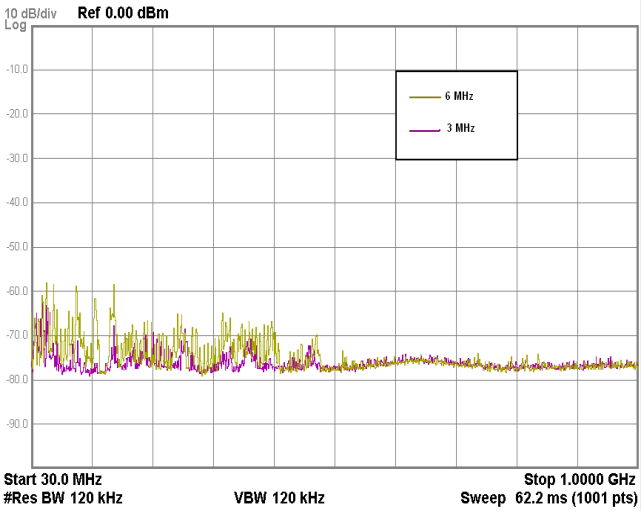


Figure 7-54. Effect of Sensor-Switching Frequency on Radiated Emissions (30 MHz – 1 GHz)



Note Frequency axis is in log scale.

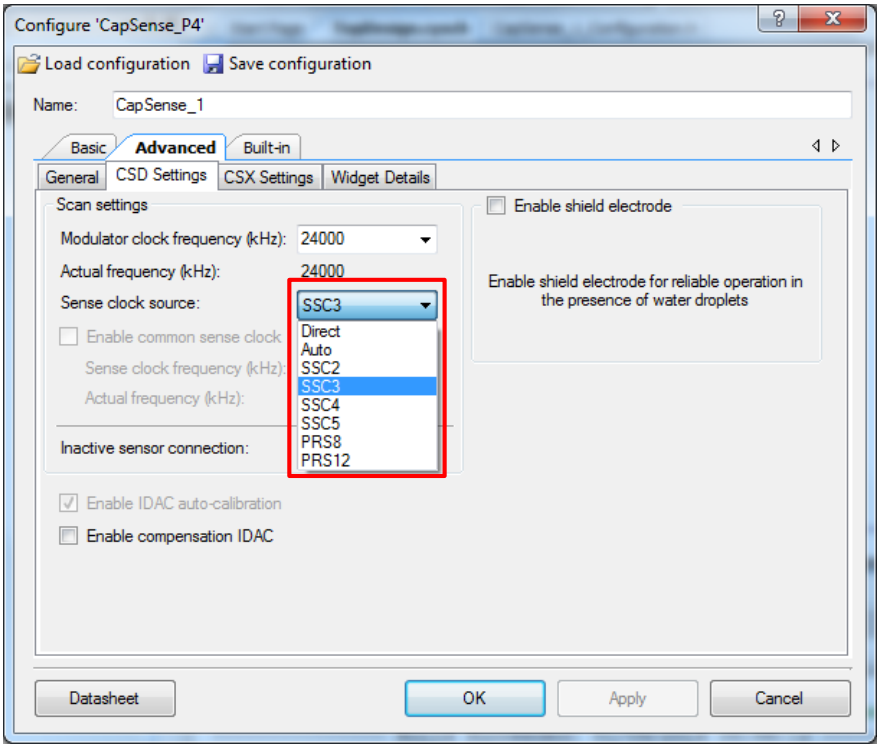
7.5.3.2.1.2.4 Pseudo Random Sense Clock

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.

7.5.3.2.1.2.5 Spread Spectrum Sense Clock

In addition to the PRS-based clock generation, the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU 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, PSoC 4100S Plus, and PSoC 4100PS family, see the Spread Spectrum Clock section in the CapSense chapter of the respective device Technical Reference Manual.

Figure 7-55. Sense Clock Sources in PSoC 4 S-Series, PSoC 4100S Plus, and PSoC 4100PS Family



7.5.3.2.1.2.6 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 7-36 shows.

Figure 7-56 and Figure 7-57 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 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 7-56. Effect of Shield Electrode on Radiated Emissions (150 kHz – 30 MHz)

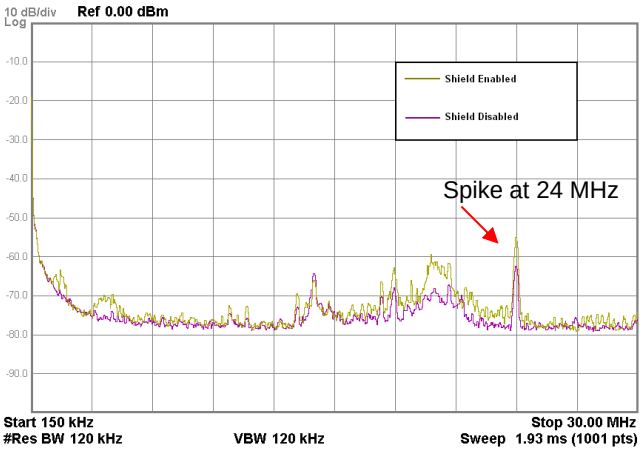
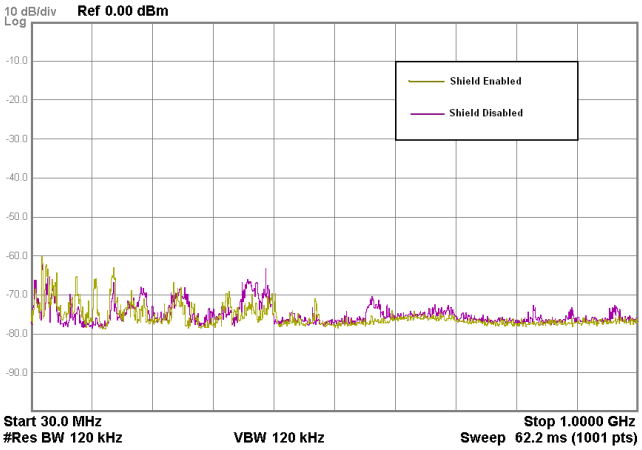


Figure 7-57. Effect of Shield Electrode on Radiated Emissions (30 MHz – 1 GHz)



Note: Frequency axis is in log scale.

7.5.3.2.1.2.7 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 3-5). Increasing the scan resolution or modulator clock divider increases the scan time. Figure 7-58 and Figure 7-59 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 7-17. 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 7-58. Effect of Scan Time on Radiated Emissions (150 kHz – 30 MHz)

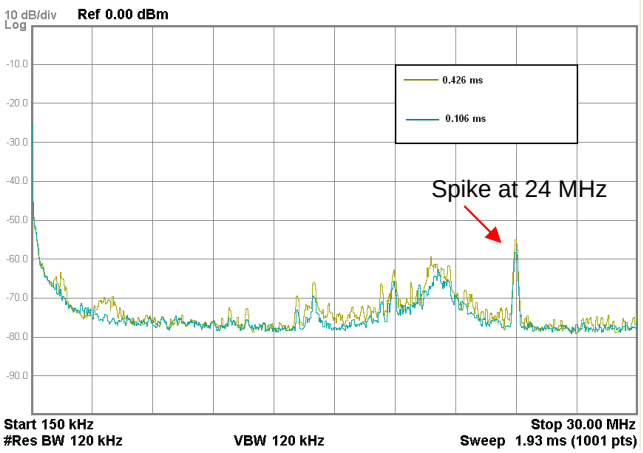
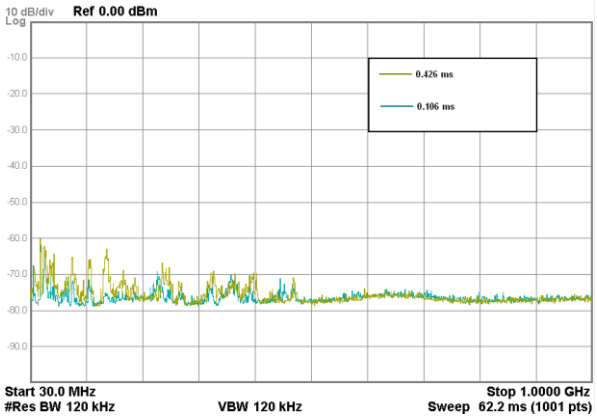


Figure 7-59. Effect of Scan Time on Radiated Emissions (30 MHz – 1 GHz)



Note Frequency axis is in log scale.

7.5.3.2.1.2.8 Sense Clock Source

Using PRS instead of direct clock drive as sense clock source spreads the radiated spectrum and hence reduces the average radiated emissions. See Figure 7-60 and Figure 7-61.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 7-60. Effect of Sense Clock Source on Radiated Emissions (150 kHz – 30 MHz)

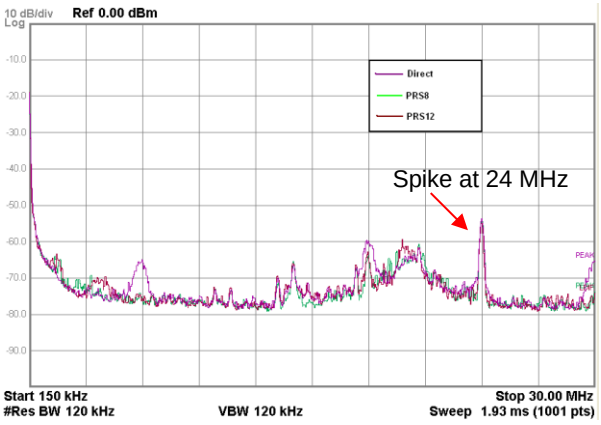
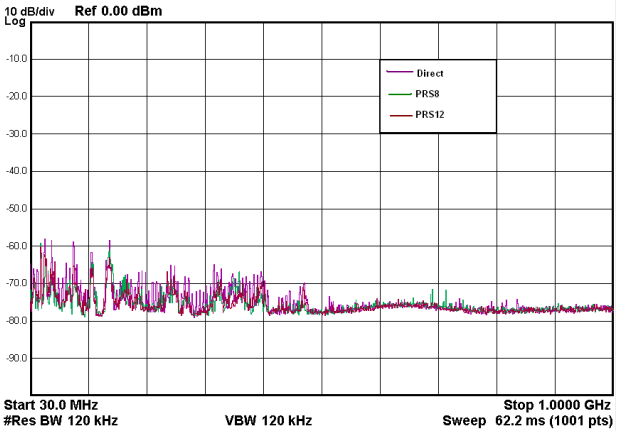


Figure 7-61. Effect of Sense Clock Source on Radiated Emissions (30 MHz – 1 GHz)



Note Frequency axis is in log scale.

7.5.3.2.1.2.9 Inactive Sensor Termination

Connecting inactive sensors to ground reduces the radiated emission by a greater degree than connecting them to the shield. Figure 7-62 and Figure 7-63 show the impact of different inactive sensor terminations on radiated emission. 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 7-62. Effect of Inactive Sensor Termination on Radiated Emissions (150 kHz – 30 MHz)

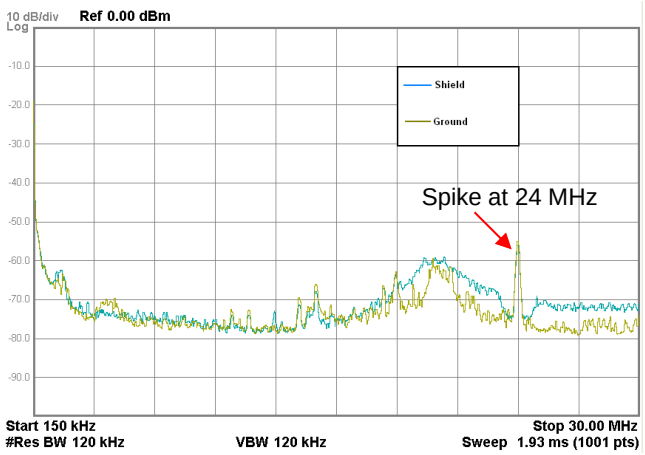
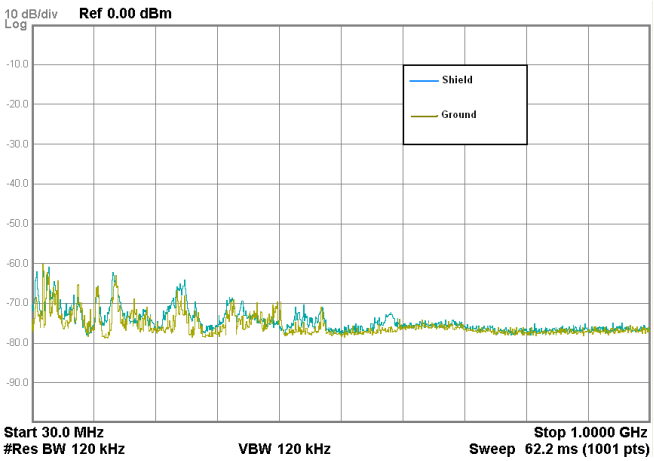


Figure 7-63. Effect of Inactive Sensor Termination on Radiated Emissions (30 MHz – 1 GHz)



Note Frequency axis is in log scale.

7.5.3.2.2 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 7.4.11 and the device Device Datasheet for details.
- Provide GND and VDD 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.

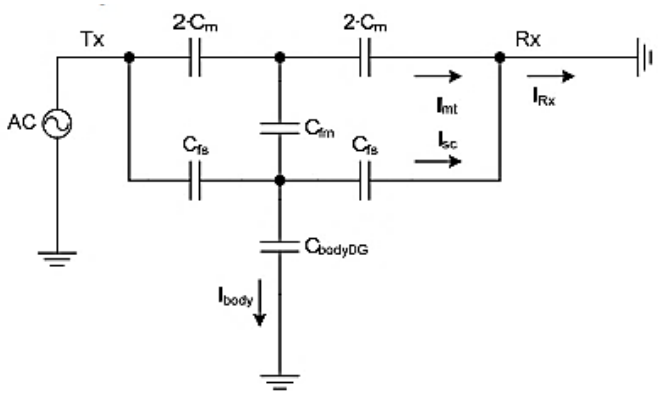
To reduce high-frequency noise, place a ferrite bead around power supply or communication lines.

7.6 Effect of Grounding

7.6.1 CSX Method

The equivalent capacitances formed in the CSX method when a finger touches the CSX sensor is shown in Figure 7-64. From Figure 7-64, current drawn from the IDAC (I_{RX}) has two components: I_{mt} and I_{sc}. These two components depend on the ratio of C_{bodyDG}/C_{fs}. Because the raw count depends on the amount of current drawn from IDAC, the increase and decrease of C_{bodyDG}/C_{fs} will affect the raw count of the sensor and cause a sudden change in the behavior on some conditions. To understand it better, consider two extreme conditions which cause $C_{bodyDG}>>C_{fs}$ and $C_{bodyDG}<<C_{fs}$.

Figure 7-64. Equivalent Circuit of the CSX Sensor when Finger Is Placed on the Button



Where,

C_m is the mutual capacitance between the Rx and Tx electrode

C_{fs} is the capacitance formed between the surface of the finger and electrode

C_{fm} is a virtual capacitance which reduces the mutual capacitance C_m due to placing a finger

C_{bodyDG} is the body capacitance relative to the device ground

Equation 7-5. Equation of Current Drawn from IDAC in CSX Method

$$I_{Rx} = I_{mt} + I_{sc}$$

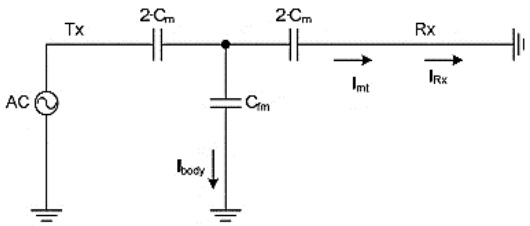
I_{mt} is due to the effective mutual capacitance between the Tx and Rx electrode.

I_{sc} is the parasitic current that flows due to the capacitance formed between the sensor and finger

7.6.1.1 CbodyDG>>Cfs

Because $C_{bodyDG}>>C_{fs}$, you can replace C_{bodyDG} with a ground conductor; the resulting equivalent circuit appears as shown in Figure 7-65. Whenever there is a finger touch, the current drawn from the IDAC is directly dependent upon the effective mutual capacitance between the Tx and Rx. This is condition is observed in a good board design.

Figure 7-65. Equivalent Circuit of the CSX Sensor when Cbody>>Cfs

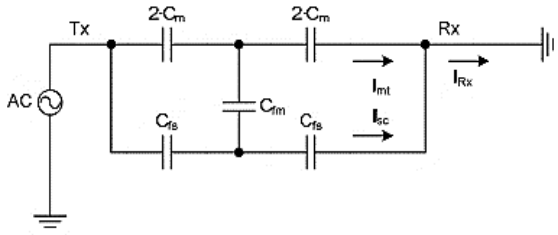


7.6.1.2 CbodyDG<<Cfs

This condition (C_{bodyDG}<<C_{fs}) is observed when a finger touches a CSX button with a very thin overlay or no overlay, or a finger touching the Rx and Tx electrodes directly, or a water drop being present on the Rx and Tx electrode only. Because C_{bodyDG}<<C_{fs}, you can remove C_{bodyDG}; the equivalent circuit for this case is as shown in Figure 7-66. In this condition, the capacitance introduced by the finger to the electrode C_{fs} is very high compared to the capacitance of the finger relative to the device ground C_{bodyDG}.

From Figure 7-66, it forms a balanced bridge circuit. Due to this, no current flows through C_{fm}, and also due to increase in C_{fs}, I_{sc} increases and thus additional current is drawn from the IDAC. This causes an unexpected behavior of decrease in the raw count.

Figure 7-66. Equivalent Circuit of the CSX Sensor when Cbody<<Cfs



Thus, for CSX sensors, design should focus on increasing the ratio of C_{bodyDG}/C_{fs}. This could be achieved in many ways; here are some examples:

1. C_{bodyDG}/C_{fs} ratio depends on the thickness of the overlay, size of the sensor, and many other factors. By experimental data, you are recommended not to use overlay thickness below 0.5 mm for CSX sensor. See Overlay Thickness.

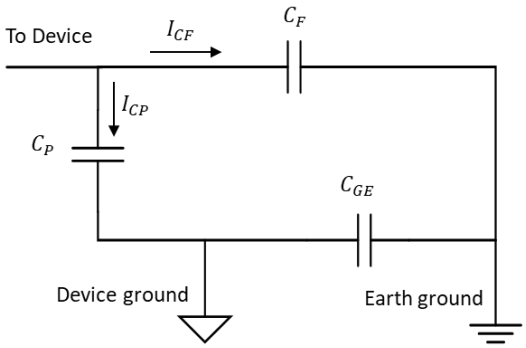
2. If the sensor is surrounded by hatch fill connected to ground, there is a lower chance that C_{bodyDG}<<C_{fs}. Therefore, ensure good ground in the design. Follow the best practices for the PCB Layout Guidelines described in this chapter.

3. In the design, it is recommended to isolate the trace lines of Rx and Tx electrode, external capacitors, and resistors of the CSX touch sensing system from any conducting surface or a finger touch to avoid direct interaction. Not following this recommendation may causes C_{bodyDG}<<C_{fs}.

7.6.2 CSD Method

The equivalent capacitances formed in the CSD method when a finger touches the CSD sensor is shown in Figure 7-67. It shows that the current drawn from the IDAC directly depends on the capacitance introduced by the finger touch. I_{CP} is a fixed component and I_{CF} depends on C_F, C_{BG}, C_{GE}. From Equation 3-6, the raw count depends on the amount of current drawn from IDAC. To understand it better, consider two scenarios of an AC/ DC mains-powered application and a battery-powered application.

Figure 7-67. Equivalent Circuit of the CSD Sensor



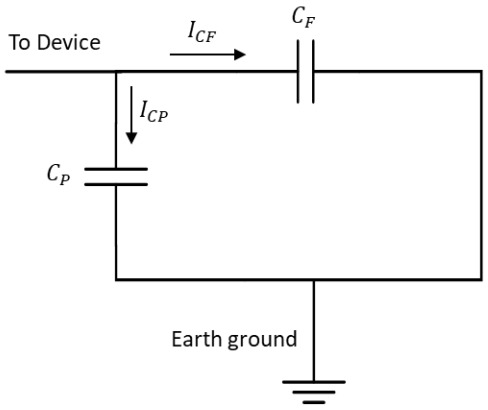
Equation 7-6. Equation of Current Drawn from IDAC in CSD Method

$$I = I_{CP} + I_{CF}$$

7.6.2.1 AC / DC-Powered Application

In an AC / DC-powered application using the mains supply, device ground is strongly coupled to earth ground. Thus, you can replace C_{GE} with a conductor and C_{BG} is usually 100 pF to 200 pF. Since C_{BG} is large when compared to C_F, you can neglect its effect. Finally, the resulting equivalent circuit is shown in Figure 7-68. The increase in total capacitance draws a higher current from the IDAC achieving a higher change in raw count for a finger touch. Thus, in this condition, you get a higher sensitivity, which means that you will get a higher signal for a finger touch.

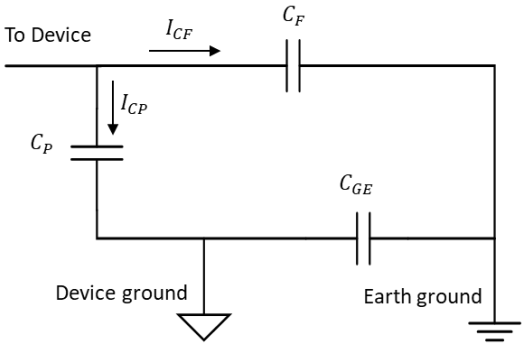
Figure 7-68. Equivalent Circuit of the CSD Sensor for Mains-Powered Application



7.6.2.2 Battery-Powered Application

In battery-powered portable applications, device ground and earth ground are lightly coupled, thus C_{GE} is small. The resulting equivalent circuit is shown in Figure 7-69. Thus, in this condition, you get a lower sensitivity; that means you will get a lower signal for a finger touch, which is due to a decrease in capacitance seen at the device.

Figure 7-69. Equivalent Circuit of the CSD for Battery-Powered Application



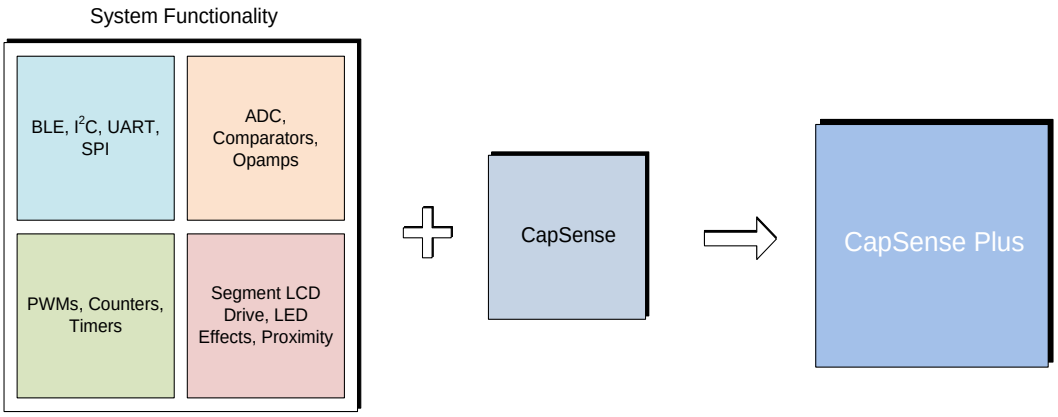
Thus, the following are the recommendations for a CSD system design in a portable application powered by a battery:

1. Add a large ground plane to the system. The ground plane should be away from the sensing element such that it doesn't increase the parasitic capacitance of the sensor. Follow the best practices for the PCB Layout Guidelines described in this chapter.
2. Use a driven shield to improve the sensitivity of portable devices. Refer to the Layout Guidelines for Shield Electrode for more details.
3. Reduce the thickness of the overlay material or use an overlay with better dielectric value to improve sensitivity.
4. Tune the CapSense system with powering it by a battery source.

8 CapSense Plus

PSoC 4 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 8-1 shows. Such applications are known as CapSense Plus applications.

Figure 8-1. CapSense Plus



The additional features available in a PSoC 4 device include:

- Communication: BLE, I2C, 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, see AN79953 - Getting Started with PSoC 4, or AN91267 - Getting Started with PSoC 4 BLE.

The flexibility of the PSoC 4 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 8-1 shows a list of example applications, where using CapSense Plus can result in significant cost savings.

Table 8-1. Examples of CapSense Plus

Application	CapSense	Opamp	ADC	Comp	PWM, Counter, Timer, UDBs	Comm (BLE, I²C, 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	I²C 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	I²C 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			I²C/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

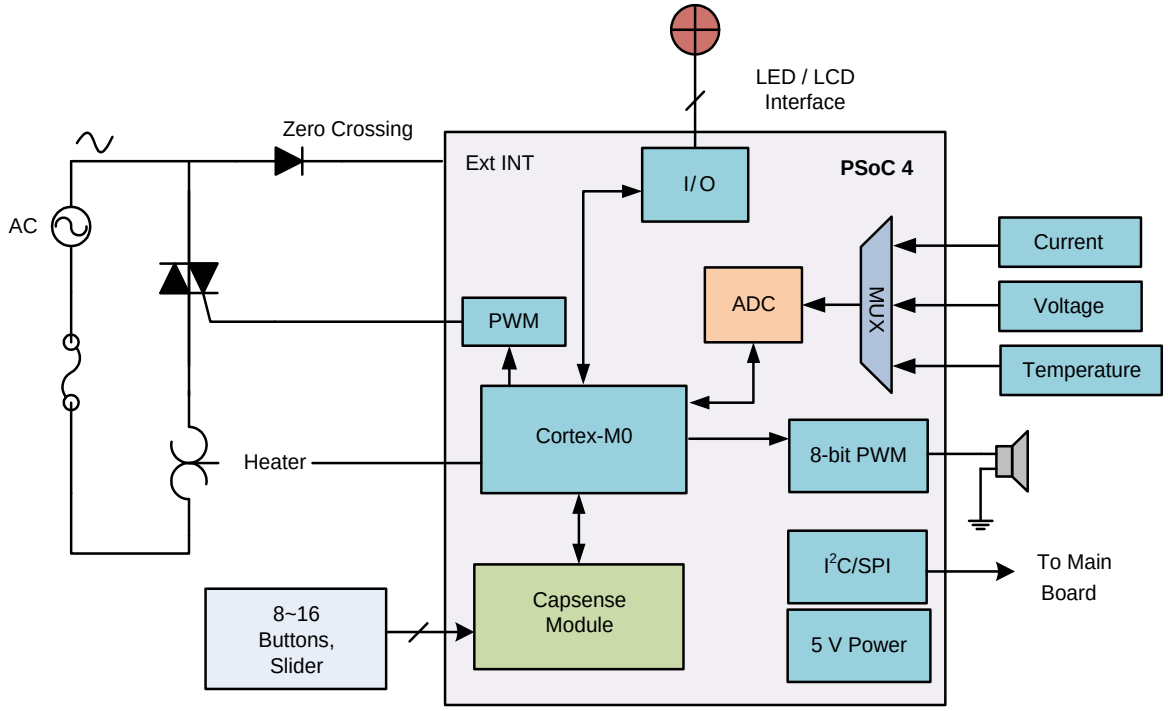
* PrISM stands for Precision Illumination Signal Modulation

** FOC stands for Field Oriented Control

*** BLDC stands for Brushless DC Motor

Figure 8-2 shows a general block diagram of a CapSense Plus application, such as an induction cooker or a microwave oven.

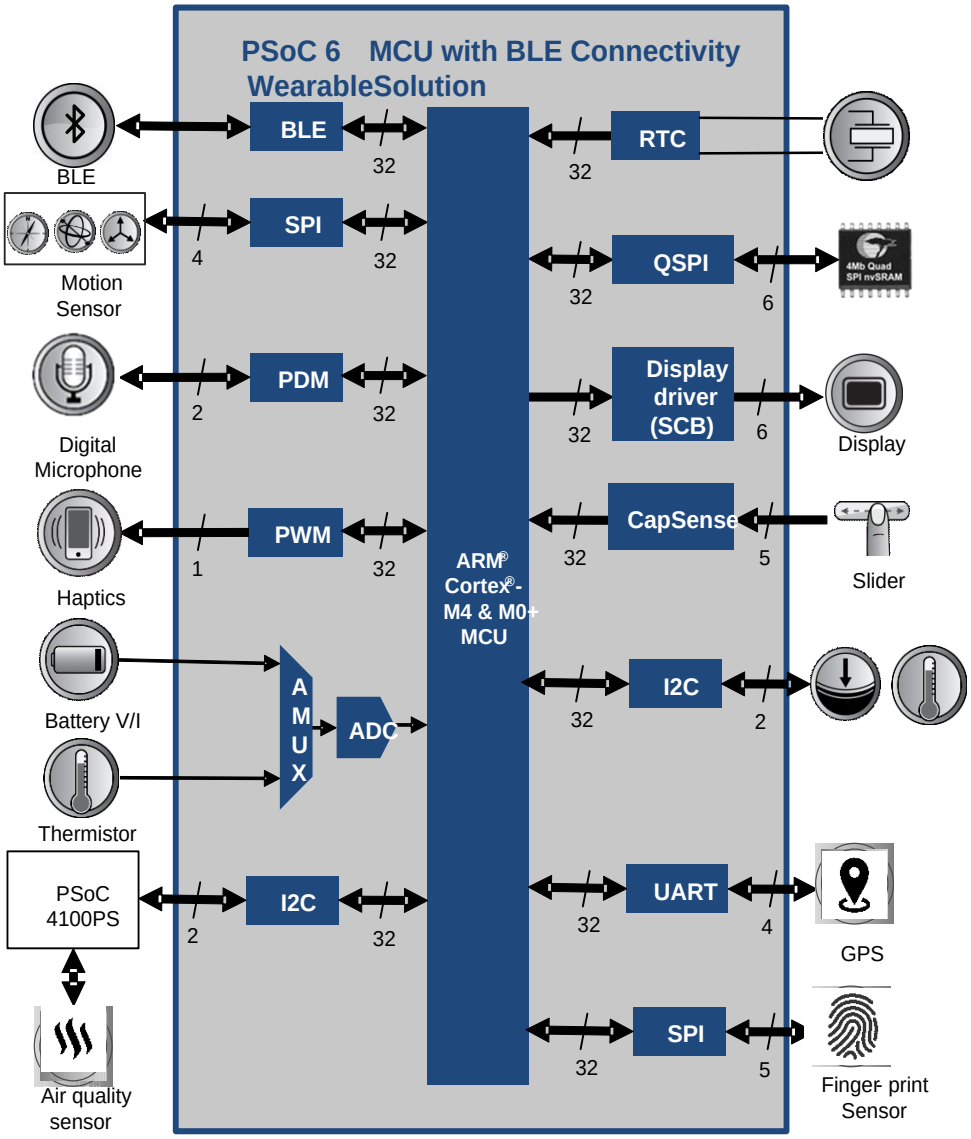
Figure 8-2. CapSense Plus System with PSoC 4



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.

Figure 8-3 shows the application-level block diagram of a fitness tracker based on PSoC 6 MCU with BLE Connectivity. The device provides a one-chip solution and includes features like activity monitoring, environment monitoring, CapSense for user interface, Bluetooth Low Energy (BLE) connectivity etc. For more information on PSoC 6 MCU, see AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

Figure 8-3. Fitness Tracker application with PSoC 6 MCU with BLE Connectivity Block Diagram



CapSense Plus systems, such as the above two examples, allow you to reduce your board size, BOM cost, and power consumption.

9 Resources

9.1 Website

Visit the Getting Started with PSoC 4, Getting Started with PSoC 4 BLE, Getting Started with PSoC 6 MCU, and Getting Started with PSoC 6 MCU with Bluetooth Low energy (BLE) Connectivity website to understand the PSoC 4, PSoC 6 MCU with BLE Connectivity.

9.2 Device Datasheet

- PSoC 4 Datasheet
- PSoC 4 BLE Datasheet
- PSoC 6 MCU Devices

9.3 Component Datasheet / Middleware Document

- PSoC 4 Capacitive sensing
- PSoC 6 Capacitive sensing
- PSoC 6 CapSense Middleware Library
- PSoC 6 CapSense Configurator in ModusToolbox

9.4 Technical Reference Manual

The PSoC 4 Technical Reference Manual (TRM) and PSoC 6 Technical Reference Manual (TRM) provide quick and easy access to information on PSoC 4 and PSoC 6 architecture including top-level architectural diagrams, register summaries, and timing diagrams.

9.5 Development Kits

Table 4-2 lists Cypress development kits that support PSoC 4 and PSoC 6 CapSense.

9.6 PSoC Creator

PSoC Creator is a state-of-the-art, easy-to-use integrated development environment. See the PSoC Creator home page.

9.7 ModusToolbox®

Cypress introduces the ModusToolbox software suite for the development of PSoC 6 based CapSense applications. You can download the ModusToolbox software here. The related documents are as follows:

- ModusToolbox Release Notes
- ModusToolbox Install Guide
- ModusToolbox User Guide

- ModusToolbox Quick Start Guide
- ModusToolbox CapSense Config
- ModusToolbox CapSense Tuner
- ModusToolbox Device Config
- ModusToolbox SmartIO Config
- PSoC Creator to ModusToolbox
- ModusToolbox Command Line

9.8 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, CapSense Application Notes and Design Guides. Here is a list of CapSense specific applications notes:

Design Guides for PSoC 3 and PSoC 5LP Devices

- PSoC® 3 and PSoC® 5LP CapSense® Design Guide

Design Guides for the CapSense Express Family

- CY8CMBR3XXX CapSense® Design Guide
- CY8CMBR2110 CapSense® Design Guide
- CY8CMBR2016 CapSense® Design Guide
- CY8CMBR2010 CapSense® Design Guide
- CY8CMBR2044 CapSense® Design Guide
- CapSense® Express™: CY8C201XX Application Notes

Design Guides for PSoC 1 Devices

- CY8C20XX7/S Design Guide
- CY8C20XX6A/H CapSense® Design Guide
- CY8C21X34/B CapSense® Design Guide
- CY8C20X34 CapSense® Design Guide

Getting Started Application Note

- AN79953 - Getting Started with PSoC® 4
- AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity
- AN221774 – Getting Started with PSoC 6 MCU

9.9 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 $(2^N - 1)$/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 (Baseline – 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 Raw Count – 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_{REF}

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

Revision	ECN#	Issue Date	Description of Change
**	3973432	04/19/2013	New Design Guide
*A	4059171	07/29/2013	Added dual IDAC support. Updated some schematics in chapter 6. Other minor changes to chapters 3, 5, and 6.
*B	4189700	11/13/2013	Added support of CY8C4000 devices. Minor fixes throughout the document.
*C	4289925	02/24/2014	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	Updated Table 1-1 per PSoC 4000 datasheet.
*E	4314223	03/20/2014	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	Updated to support PSoC 4000 and PSoC Creator 3.0 SP1.
*G	4494249	08/29/2014	Added Reference to Getting Started with CapSense in Proximity (Three-Dimensional) Renamed Section 2.5 to Liquid Tolerance and re-wrote this section. Updated the recommendations for Shield drive i.e. Csh_tank precharge and Cmod precharge in Section 3.1.7 CapSense CSD Shielding Added recommendation for setting “API resolution” in Section Added guidelines on how to select value of “Sensitivity” parameter in Section Updated recommended values of threshold and hysteresis parameters in Section Manual Tuning Trade-offs. Added Section Manual Tuning Slider Example Updated maximum overlay thickness value for sliders in Table 7-2 Added guideline on maximum thickness for overlays of materials other than acrylic in Section 7.3.2Overlay Thickness Re-wrote Section Slider Design Added recommendations on DC loads in Section 6.3.5 Renamed and rewrote section 7.4.12to Layout Guidelines for Liquid Tolerance Added Section 7.4.13.1 External Capacitors Pin Selection Updated slider related recommendations in Table 7-14. Layout Rule Checklist Updated Section 6.5 Electromagnetic Compatibility (EMC) Considerations, added extensive data on hardware and firmware considerations.
*H	4602375	12/19/2014	Added information for the PSoC 4 BLE family of devices. Added information for the PRoC BLE family of devices. Updated ground and power layout guidelines in Section 7.4.10 and Section 7.4.11.
*I	4624027	01/21/2015	Added information for PSoC 4200-M family of devices Added footnote in section Slider Design Added GPIO source/sink current limit in Table 7-9 Changed document title to PSoC® 4 CapSense® Design Guide – AN85951
*J	4771699	06/02/2015	Changed Document Title to “AN85951 – PSoC® 4 CapSense® Design Guide” Updated Design Considerations Updated Preventing ESD Discharge

Revision	ECN#	Issue Date	Description of Change
			Updated Figure 7-44 Updated Redirect Replaced "Guard Ring" with "Ground Ring" Updated Figure 7-45
*K	4891423	08/20/2015	Added Table 3-1 Removed section 3.2.1 CMOD Precharge Added section CapSense in PSoC 4xxxM/4xxxL-Series Updated section Trace Routing Added reference of AN2397 Added recommendation for modulator clock divider in section Manual Tuning Trade-offs Added Figure 7-41
*L	4905591	09/16/2015	Updated Section 3.4 Updated Figure 3-10 Updated Table 3-4, Table 4-2, Table 7-7, Table 7-8, Table 7-10
*M	5076590	01/19/2016	Updated Introduction Moved Signal-to-Noise Ratio to Chapter 2 Updated Chapters PSoC 4 and PSoC 6 MCU CapSense and CapSense Performance Tuning for details Added section to Chapter 4 Added Glossary
*N	5131335	02/23/2016	Added information on mutual-capacitance sensing in PSoC 4 device series Added information on CapSense 3.0 changes Added following sections: - Mutual-Capacitance Sensing - CapSense Architecture in PSoC 4 S-Series Updated following sections: - Introduction - CapSense Widgets - CapSense Design and Development Tools - CapSense Performance Tuning
*O	5162301	03/04/2016	Added PSoC Analog Coprocessor references Updated External Capacitors Pin Selection section Updated Development Kits section Updated document title Updated Copyright notice
*P	5307639	06/14/2016	Updated IDAC sinking mode recommendation Updated template
*Q	5526001	11/18/2016	Updated Table 7-10
*R	5687926	04/19/2017	Updated logo and copyright
*S	5896262	09/22/2017	Added references to PSoC 4100S Plus throughout the document Updated Section 1.3 CapSense Features with PSoC 4100S Plus features Updated Table 4-2. PSoC 4 and PSoC 6 CapSense Development Kits with CY8CKIT-149 PSoC 4100S Plus Prototyping Kit Updated Section 9.8Application Notes with specific list of CapSense Application Notes
*T	6036561	01/18/2018	Changed document title Added references to PSoC 6 MCU features throughout the document Updated Section 3.1CapSense CSD Sensing Method with generalized architecture block diagram for CSD sensing Added Section 6Gesture in CapSense Updated Table 4-2, Table 7-8, Table 7-11
*U	6084086	02/28/2018	Added references to PSoC 4100PS throughout the document
*V	6375492	11/08/2018	Updated the entire document with references to CY8C62x8 and CY8C62xA devices.

Revision	ECN#	Issue Date	Description of Change
			Updated the entire document with references to ModusToolbox. Updated Table 4-2 with the information of PSoC 6 kits. Updated section Mutual-Capacitance Button with the information of additional mutual cap key. Removed all references to PRoC BLE devices.
*W	6540965	04/11/2019	Updated SmartSense and Manual Tuning with respect to the latest component. Removed details on different shield drive mode from CapSense CSD Shielding Updated CapSense CSX Sensing Method Updated figures in PSoC Creator, SmartSense, and Gesture in CapSense with respect to the latest component Removed a table in External Capacitors Pin Selection section Updated Table 3-1
*X	6759285	01/07/2020	Added Liquid tolerance for Mutual Capacitance Sensing section Removed Mutual Capacitance Button Design section Updated Table 3-2 and Table 3-3 Updated CapSense CSX Sensing Method Added ModusToolBox section in Chapter 4 Updated SmartSense and Manual Tuning section with respect to the latest component. Updated Slider Tuning Guidelines section Added Tuning Shield Electrode section Updated Gesture chapter with gesture tuning guidelines Updated the Low Power design section Updated Sensor and Device placement section Updated Slider Design section Added Effect of Grounding in CSX method and Effect of Grounding in CSD method section
*Y	6833908	03/18/2020	Updated Sensor Pin Selection and Tuning Debug FAQs sections.
*Z	6975032	10/08/2020	Added 4.2.2 CapSense Configurator. Moved Tuning Shield Electrode section under 5.3.2 CSD Sensing Method. Added 5.3.5.12 I am observing a low Cm for my CSX Button. Added 7.4.3.2 Mutual- Capacitance Button Design. Added additional layout guidelines in 7.4.5 Sensor and Device Placement. Added additional trace routing guidelines in 7.4.7 Trace Routing. Added additional guard trace guidelines in 7.4.8 Crosstalk Solutions. Added new section 7.5 Noise in CapSense System. Added a single line description on self cap buttons in 7.4.3.1 Self-Capacitance Button Design. Moved ESD Protection and Electromagnetic Compatibility (EMC) Considerations under 7.5.3 External Noise. Moved Effect of grounding on CSX method and Effect of grounding on CSD method under 7.6 Effect of Grounding.