© Cypress Semiconductor Corporation, 2013-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, “Security Breach”). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. “High-Risk Device” means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. “Critical Component” means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress’s published data sheet for the product explicit states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.1</td>
<td>Abstract</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>Introduction</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>CapSense Features</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>PSoC 4 and PSoC 6 MCU CapSense Plus Features</td>
<td>7</td>
</tr>
<tr>
<td>1.5</td>
<td>CapSense Design Flow</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>CapSense Technology</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>CapSense Fundamentals</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Self-Capacitance Sensing</td>
<td>12</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Mutual-Capacitance Sensing</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Capacitive Touch Sensing Method</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1</td>
<td>CapSense Sigma Delta (CSD)</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2</td>
<td>CapSense Crosspoint (CSX)</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Signal-to-Noise Ratio</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>CapSense Widgets</td>
<td>18</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Buttons (Zero-Dimensional)</td>
<td>18</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Sliders (One-Dimensional)</td>
<td>19</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Touchpads / Trackpads (Two-Dimensional)</td>
<td>20</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Proximity (Three-Dimensional)</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Liquid Tolerance</td>
<td>21</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Effect of Liquid Droplets and Liquid Stream on a CapSense Sensor</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Driven-Shield Signal and Shield Electrode</td>
<td>24</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Guard Sensor</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>PSoC 4 and PSoC 6 MCU CapSense</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>CapSense CSD Sensing Method</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1</td>
<td>GPIO Cell Capacitance to Current Converter</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2</td>
<td>IDAC Sourcing Mode</td>
<td>28</td>
</tr>
<tr>
<td>3.1.3</td>
<td>IDAC Sinking Mode</td>
<td>29</td>
</tr>
<tr>
<td>3.1.4</td>
<td>CapSense Clock Generator</td>
<td>30</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Sigma Delta Converter</td>
<td>31</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Analog Multiplexer</td>
<td>33</td>
</tr>
<tr>
<td>3.1.7</td>
<td>CapSense CSD Shielding</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>CapSense CSX Sensing Method</td>
<td>33</td>
</tr>
<tr>
<td>3.2.1</td>
<td>CapSense Architecture in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>CapSense in PSoC 4xxM/4xxL-Series</td>
<td>36</td>
</tr>
</tbody>
</table>
4 CapSense Design and Development Tools

4.1 PSoC Creator

4.1.1 CapSense Component

4.1.2 CapSense_ADC Component

4.1.3 Tuner Helper

4.1.4 Example Projects

4.2 ModusToolbox

4.3 Hardware Kits

5 CapSense Performance Tuning

5.1 Selecting between SmartSense and Manual Tuning

5.2 SmartSense

5.2.1 Component Configuration for SmartSense

5.3 Manual Tuning

5.3.1 Overview

5.3.2 CSD Sensing Method

5.3.3 CSX Sensing Method

5.3.4 Manual Tuning Trade-offs

5.3.5 Tuning Debug FAQs

6 Gesture in CapSense

6.1 Gesture Support in CapSense

6.2 Gesture Groups

7 Design Considerations

7.1 Firmware

7.1.1 Low-Power Design

7.2 Sensor Construction

7.3 Overlay Selection

7.3.1 Overlay Material

7.3.2 Overlay Thickness

7.3.3 Overlay Adhesives

7.4 PCB Layout Guidelines

7.4.1 Parasitic Capacitance, C_P

7.4.2 Board Layers

7.4.3 Slider Design

7.4.4 Sensor and Device Placement

7.4.5 Trace Length and Width

7.4.6 Trace Routing

7.4.7 Crosstalk Solutions

7.4.8 Vias

7.4.9 Ground Plane

7.4.10 Power Supply Layout Recommendations

7.4.11 Layout Guidelines for Liquid Tolerance

7.4.12 Schematic Rule Checklist

7.4.13 Layout Rule Checklist

7.5 ESD Protection
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
- Allows CapSense block re-configuration as an ADC, and supports ADC input on any GPIO pin
- Provides superior SNR with programmable voltage reference (VREF)
- 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 For PSoC 6 family devices follow recommendations stated in the Errata section of the corresponding device datasheet to achieve the best CapSense sensitivity and accuracy.
Opamps with comparator mode and SAR input buffering capability
- Segment LCD direct drive that supports up to four commons
- SPI/EUART/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
- Nine serial communication block (SCB) configurable as I2C, SPI, or UART interfaces
- 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 Design Guide to select an appropriate CapSense device based on your requirements.
1.5 CapSense Design Flow

Figure 1-1 shows the typical flow of a product design cycle with capacitive sensing; the information in this guide is highlighted in green. Table 1-1 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

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

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

CapSense also requires external capacitor \( C_{MOD} \) for self-capacitance sensing and \( C_{INTA} \) and \( C_{INTB} \) capacitors for mutual-capacitance sensing. These external capacitors are connected between a dedicated GPIO pin and ground. If shield electrode is implemented for liquid tolerance, or for large proximity sensing distance, an additional \( C_{TANK} \) capacitor may be required. The recommended values of the external capacitors are listed in Table 7-6.
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.
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](image)

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:

- \( \varepsilon_0 \) = Free space permittivity
- \( \varepsilon_r \) = Relative permittivity of overlay
- \( A \) = Area of finger and sensor pad overlap
- \( d \) = Thickness of the overlay

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

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

\[ C_S = C_P + C_F \]

In the absence of touch, \( C_S \) is equal to \( C_P \).

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

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

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

In a mutual-capacitance sensing system, a digital voltage signal switching between VDDIO\(^2\) or VDD\(^3\) (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. 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 \(C_{1M}\) 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.

---

\(^2\) VDDIO is the power supply for I/O pin.

\(^3\) VDD is the device power supply for digital section.
2.2 Capacitive Touch Sensing Method

PSoc 4 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 \( C_P + C_r \), and the raw count increases. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

2.2.2 CapSense Crosspoint (CSX)

Figure 2-8 shows the simplified block diagram of the CSX method.
With CSX, a voltage on the Tx pin (or Tx electrode) couples charge on to the RX pin. This charge is proportional to the mutual capacitance between the Tx and Rx electrodes. An analog multiplexer then selects one of the Rx channel and feeds it into the current to digital converter.

The output count of the current to digital converter, known as \( \text{Rawcount}_{\text{Counter}} \), 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

\[
\text{Rawcount}_{\text{Counter}} = G_{CM} C_M
\]

Where \( G_{CM} \) is the capacitance to digital conversion gain of Mutual Capacitance method, and \( C_M \) is the mutual-capacitance between two electrodes.

Figure 2-9 shows a plot of raw count over time. When a finger touches the sensor, \( C_M \) decreases from \( C_M \) to \( C_M' \) (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 rawcount is computed using Equation 3-11.
For an in-depth discussion of the PSoC 4 CapSense CSD and CSX blocks, see PSoC 4 CapSense.

2.3 Signal-to-Noise Ratio

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

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

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

![Button Sensor](image1)
![Slider Sensor](image2)
![Proximity Sensor](image3)

2.4.1 Buttons (Zero-Dimensional)

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

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

Figure 2-12. Simple CapSense Buttons

![Buttons](image4)

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

![Buttons](image5)
If the application requires a large number of buttons, such as in a calculator keypad or a QWERTY keyboard, you can arrange the CapSense buttons in a matrix, as Figure 2-14 shows. This allows a design to have multiple buttons per GPIO. For example, the 12-button design in Figure 2-14 requires only seven GPIOs.

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

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

### 2.4.2 Sliders (One-Dimensional)

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

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

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

In a linear slider, the segments are arranged inline, as Figure 2-15 shows. Each slider segment connects to a PSoC GPIO. A zigzag pattern (double chevron) is recommended for slider segments. This layout ensures that when a segment is touched, the adjacent segments are also partially touched, which aids estimation of the centroid position.
Radial sliders are similar to linear sliders except that radial sliders are continuous. Figure 2-16 shows a typical radial slider.

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

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

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

![Figure 2-18. Proximity Sensor](image)

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

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, and humidity changes. In a capacitive-sensing application design, false sensing of touch or proximity detection may happen due to the presence of a film of liquid or liquid droplets on the sensor surface, due to the conductive nature of some liquids. Cypress’s CSD sensing method can compensate for variation in raw count due to these causes and provide a robust, reliable, capacitive sensing application operation.

![Figure 2-19 Liquid-Tolerant CapSense-Based Touch User Interface in a Washing Machine](image)

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

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

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

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

$C_P$ – Sensor parasitic capacitance

$C_{LD}$ – Capacitance added by the liquid droplet
To nullify the effect of capacitance added by the liquid droplet to the CapSense circuitry, you should drive the hatch fill around the sensor with the driven-shield signal.

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

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

\[
\begin{align*}
C_S & \quad \text{Sensor parasitic capacitance} \\
C_{SH} & \quad \text{Capacitance between the sensor and the hatch fill} \\
C_{HG} & \quad \text{Capacitance between the hatch fill and ground} \\
C_{LD} & \quad \text{Capacitance added by the liquid droplet}
\end{align*}
\]

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

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

![Figure 2-26. Driven Shield Signal](image)

Shield electrode is used for the following purposes:

- To implement liquid-tolerant CapSense designs: Shield electrode helps in making CapSense designs liquid-tolerant as explained above.

- To improve the proximity sensing distance in the presence of floating or grounded conductive objects: A shield electrode, when placed between the proximity sensor and a floating or a grounded conductive object, reduces the effect of these objects on the proximity-sensing distance and helps in achieving large proximity-sensing distance. See the “Proximity Sensing” section in the Getting Started with CapSense Design Guide for more details.

- To reduce the parasitic capacitance of the sensor: When a CapSense sensor has a long trace, the \( C_p \) of the sensor will be very high because of the increased coupling of sensor electric field lines from the sensor trace to the surrounding ground. By implementing a shield electrode, the coupling of electric field lines to ground is reduced, which results in reducing the \( C_p \) of the sensor.

See the Layout Guidelines for Liquid Tolerance section for layout guidelines of shield electrode.
2.5.3 Guard Sensor

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

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

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

![CapSense Controller](image)

2.5.3.1 Effect of Liquid Properties on Liquid-Tolerance Performance

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

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

1. If your application requires tolerance to liquid droplets, implement a shield electrode. If your application requires tolerance to streaming liquids along with liquid droplets, implement a shield electrode and a guard sensor. Follow the schematic and layout guidelines explained in the Layout Guidelines for Liquid Tolerance section to construct the shield electrode and guard sensor respectively.

2. In the CapSense Component, enable the driven-shield signal and specify the **Inactive sensor connection** option as **Shield**.

3. If the SmartSense algorithm is used, set the finger capacitance of the guard sensor (if implemented) such that it will be triggered only when there is a liquid stream on the touch surface.

If manual tuning is used, set the resolution of the guard sensor such that it will be triggered only when there is a liquid stream on the touch surface.
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 4 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.

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 CapSense architecture in PSoC 4000, PSoC 4200, PSoC 4200 BLE, PSoC 4200M, and PSoC 4200L devices and section 3.2.1 explains the differences in CapSense architecture in the PSoC 4 S-Series, 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.
As explained in Capacitive Touch Sensing Method, this block works by first converting the sensor capacitance into an equivalent current. An analog multiplexer then selects one of the currents and feeds it into the current-to-digital converter. This current-to-digital converter consists of a sigma-delta converter, which controls the modulation IDAC such that for a specific period, the total current sourced or sunk by the IDACs is the same as the total current sunk or sourced by the sensor capacitance. The digital count output of the sigma-delta converter is an indicator of the sensor capacitance and is called a raw count. This block can be configured in either IDAC Sourcing mode or IDAC Sinking mode. In the IDAC Sourcing mode, the IDACs source current to AMUXBUS while the GPIO cells sink current from AMUXBUS. In the IDAC Sinking mode, the IDACs sink current from AMUXBUS while the GPIO cells source current to AMUXBUS.

### 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.

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.

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.
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 Sunked 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.
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 (ICS)

\[ I_{CS} = C_s F_{SW} (V_{DDD} - V_{REF}) \]

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 pseudo random sequence (PRS), and 12-bit PRS. Some PSoC 4 and PSoC 6 parts also support additional Spread Spectrum Clock (SSCx) modes. For more details on the supported modes for PSoC device, see the CapSense component datasheet.

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

Pseudo Random Sequence (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 the Sense Clock Related Parameters section 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**
\[
\text{Sensor scan time} = \text{Hardware scan time} + \text{Sensor Initialization time}
\]

**Equation 3-5. Hardware Scan Time**
\[
\text{Hardware scan time} = \frac{2^{\text{Resolution}} - 1}{\text{Modulator Clock Frequency}}
\]

Here, “Resolution” is the scan resolution.

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, a clock generator, and two current sourcing/sinking digital-to-analog converters (IDACs), as Figure 3-1 shows.

The sigma-delta modulator controls the current of the modulator IDAC in an on/off manner. The compensation IDAC is either always ON or always OFF.

The sigma-delta converter can operate in either single IDAC mode or dual IDAC mode:
- In the **single IDAC mode**, the modulation IDAC is controlled by the sigma-delta modulator; the compensation IDAC is always OFF.
- In the **dual IDAC mode**, the modulation IDAC is controlled by the sigma-delta modulator; the compensation IDAC is always ON.

The sigma-delta converter also requires an external integrating capacitor, called modulator capacitor \( C_{\text{MOD}} \), as Figure 3-1 shows. The recommended value of \( C_{\text{MOD}} \) is 2.2 nF.

The sigma delta modulator maintains the voltage across \( C_{\text{MOD}} \) at \( V_{\text{REF}} \). It works in one of the following modes:
- **IDAC Sourcing Mode**: In this mode, the switched-capacitor circuit sinks current from \( C_{\text{MOD}} \) through \( \text{AMUXBUS A} \), and the IDACs then source current to \( \text{AMUXBUS A} \) to balance its voltage.
- **IDAC Sinking Mode**: In this mode, the IDACs sink current from \( C_{\text{MOD}} \) through \( \text{AMUXBUS A} \), and the switched-capacitor circuit sources current to \( \text{AMUXBUS A} \) to balance its voltage.

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

The sigma-delta converter can operate from 8-bit to 16-bit resolutions. In the **single IDAC mode**, the raw count is proportional to the sensor capacitance. If \( N \) is the resolution of the sigma-delta converter and \( I_{\text{MOD}} \) 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**
\[
\text{raw count} = (2^N - 1) \frac{V_{\text{REF}} F_{\text{SW}}}{I_{\text{MOD}}} C_S
\]
Similarly, the approximate value of raw count in the IDAC Sinking mode is:

**Equation 3-7. Single IDAC Sinking Raw Count**

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

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

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

**Equation 3-8. 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**

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

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>
<thead>
<tr>
<th>Sl. No.</th>
<th>Parameter</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( N )</td>
<td>Scan Resolution</td>
<td>Scan resolution is configurable from 8-bit to 16-bit.</td>
</tr>
<tr>
<td>2</td>
<td>( V_{REF} )</td>
<td>N/A</td>
<td>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 the CapSense Component datasheet for more details.</td>
</tr>
<tr>
<td>3</td>
<td>( F_{SW} )</td>
<td>Sense Clock Frequency</td>
<td>Sense clock frequency and sense clock source decides the frequency at which sensor is switching. See section Sense Clock for more details.</td>
</tr>
<tr>
<td>4</td>
<td>( I_{MOD} )</td>
<td>Modulator IDAC</td>
<td>( I_{MOD} = ) Modulation IDAC current</td>
</tr>
<tr>
<td>5</td>
<td>( I_{COMP} )</td>
<td>Compensation IDAC</td>
<td>( I_{COMP} = ) Compensation IDAC current</td>
</tr>
<tr>
<td>6</td>
<td>( V_{DD} )</td>
<td>N/A</td>
<td>This parameter is the device supply voltage.</td>
</tr>
<tr>
<td>7</td>
<td>( C_S )</td>
<td>N/A</td>
<td>This parameter is the sensor parasitic capacitance.</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>Modulator Clock Frequency</td>
<td>Modulator clock divider does not impact raw count. See the Modulator Clock section for more details.</td>
</tr>
</tbody>
</table>
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 in Figure 3-6) form this analog multiplexer. AMUXBUS A connects to all GPIOs that support CapSense. See your corresponding device Datasheet for a list of port pins that support CapSense. AMUXBUS A also connects the integrating capacitor $C_{MOD}$ 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 the Driven-Shield Signal and Shield Electrode and Effect of Liquid Droplets and Liquid Stream on a CapSense Sensor sections for more 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 $V_{DD}$ 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. Shield drive can be configured through the Shield tank capacitor option in the CSD Settings tab of the CapSense Component.

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, Comparator (CMP) and counter and digital logic
- AMUX-A, Tx clock and Modulator clock, $V_{REF}$ and port pins for Tx and Rx electrodes and external caps.
- Two external capacitors (CINTA and CINTB)

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

![CapSense CSX Sensing Method Configuration](image-url)
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 (VDDD if VDDIO is not available) and ground. The Rx electrode is statically connected to AMUXBUS A. The CSX method requires two external integration capacitors, C\textsubscript{INTA} and C\textsubscript{INTB}. The value of these capacitors is listed in Table 7-6.

Figure 3-9 shows the waveforms on the Tx electrode and C\textsubscript{INTA} and C\textsubscript{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\textsubscript{REF} and performing a series of sub-conversions. The sum of all sub-conversions in a sample is the result, and is referred to as "raw count". A sub-conversion is a capacitance to count conversion performed within a Tx clock cycle.

During a sub-conversion, both SW1 and SW2 switches are operated in phase with the Tx clock. On the rising edge of the Tx clock, SW1 is closed (SW2 is open during this time) and charge flows from the Tx electrode to the Rx electrode. This charge is integrated onto the C\textsubscript{INTA} capacitor, which increases the voltage on C\textsubscript{INTA}. The IDAC is configured in sink mode to discharge the C\textsubscript{INTA} capacitor back to voltage V\textsubscript{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\textsubscript{INTB} to go below V\textsubscript{REF}. The IDAC is configured in source mode to bring the voltage on C\textsubscript{INTB} back to V\textsubscript{REF}. The charge transferred between Tx and Rx electrodes in both the cycles is proportional to mutual capacitance, C\textsubscript{M}, between the electrodes. The comparator output enables the counter while the IDAC is charging or discharging the external capacitors. The counter counts in terms of modulator clock cycles during a sub-conversion. Multiple sub-conversions are performed and the result is accumulated to the same counter to produce "raw count" for a sensor.

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

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

\[
\text{Rawcount}_{\text{Counter}} = \frac{2 \ast V_{\text{TX}} \ast F_{\text{TX}} \ast C_{M} \ast \text{MaxCount}}{\text{IDAC}}
\]

\[
\text{MaxCount} = \frac{F_{\text{Mod}} \ast N_{\text{Sub}}}{F_{\text{TX}}}
\]

Where,

IDAC – IDAC current

C\textsubscript{M} – Mutual capacitance between Tx and Rx electrodes

V\textsubscript{TX} – Amplitude of the Tx signal

F\textsubscript{TX} – Tx clock frequency
F_{Mod} – Modulator clock frequency
N_{Sub} – Number of Sub-Conversions

When you place a finger over a CSX button, the mutual cap between the Rx and Tx electrode decreases, which decreases the rawcount. The Raw Count obtained from the hardware is processed by the CapSense Component to increase the counts for a touch event (similar to that in CSD). The final inverted Raw Count that you see is given by Equation 3-11.

\[
\text{Rawcount}_{\text{Component}} = \text{MaxCount} - \text{Rawcount}_{\text{Counter}}
\]

See the CSX Sensing Method section for more detailed explanation of the above CSX Hardware parameters.

### 3.2.1 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 previous generation CapSense architecture. The main differences in the CapSense architecture between the PSoC 4 devices are listed in Table 3-2.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Third-Generation CapSense (PSoC 4, 4-M, 4-BLE, and 4-L)</th>
<th>Fourth-Generation CapSense (PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU)</th>
<th>Advantages of Fourth-Generation over Third Generation CapSense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing Modes</td>
<td>Self-Cap and Mutual-Cap modes</td>
<td>Self-Cap, Mutual-Cap, and ADC modes</td>
<td>Additional ADC functionality with the same hardware</td>
</tr>
<tr>
<td>Sensor Parasitic Capacitance (C_p) Range</td>
<td>5 pF – 60 pF</td>
<td>5 pF – 200 pF</td>
<td>Supports high-C_p design applications</td>
</tr>
<tr>
<td>V_{REF}</td>
<td>1.2 V</td>
<td>0.6 V to VDDA-0.6 V</td>
<td>Improved SNR</td>
</tr>
<tr>
<td>IDAC LSB Size</td>
<td>1.2 ( \mu )A, 2.4 ( \mu )A</td>
<td>37.5 nA, 300 nA, 2.4 ( \mu )A</td>
<td>Improved sensitivity</td>
</tr>
<tr>
<td>Split IDAC Capability</td>
<td>Requires two IDACs</td>
<td>Requires one IDAC</td>
<td>Requires less resource to achieve same performance and frees up one IDAC for general purpose use</td>
</tr>
<tr>
<td>EMI Reduction - Digital</td>
<td>-</td>
<td>Spread Spectrum - CSD controlled</td>
<td>Spread Spectrum clock is generated by hardware, and CPU is completely free</td>
</tr>
<tr>
<td>10-bit ADC</td>
<td>No</td>
<td>Yes</td>
<td>The same CSD hardware can be used as an ADC when CapSense Scanning is not in progress.</td>
</tr>
<tr>
<td>Hardware State Machine</td>
<td>No</td>
<td>Yes</td>
<td>CPU is no longer required for Initialization or Spread Spectrum SenseClk generation</td>
</tr>
</tbody>
</table>

The CapSense hardware in the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU supports self-capacitance (CSD) and mutual-capacitance (CSX)-based capacitive sensing. The hardware also supports input voltage measurement when CapSense scanning is not in progress.

This section explains how the CapSense hardware is used for performing self-capacitance and mutual-capacitance sensing. See the CapSense chapter in the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU device Technical Reference Manual for a detailed explanation of the CapSense hardware. A basic knowledge of CapSense architecture is required to understand the working of self-capacitance and mutual-capacitance.
3.3 CapSense in PSoC 4xxxM/4xxxL-Series

The PSoC 4xxxM/4xxxL series of devices support two CapSense blocks – CSD0 and CSD1. Each block has the same functionality and performance as explained in the 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](image)

*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 \( \text{CMOD} \) and \( \text{CSH}\_\text{TANK} \) capacitor. The summary of differences between CSD0 and CSD1 blocks is listed in Table 3-3.

<table>
<thead>
<tr>
<th></th>
<th>CSD0</th>
<th>CSD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{CMOD} )</td>
<td>( \text{CSH}_\text{TANK} )</td>
<td>( \text{CSH}_\text{TANK} )</td>
</tr>
<tr>
<td>( \text{INTAB} )</td>
<td>Any pin except PORT5 pins (for PSoC 4xxxM).</td>
<td>Any pin in PORT5 (for PSoC 4xxxM).</td>
</tr>
<tr>
<td>CapSense Pin</td>
<td>Any pin except PORT5 pins (for PSoC 4xxxM).</td>
<td>Any pin in PORT5 (for PSoC 4xxxM).</td>
</tr>
<tr>
<td>Shield Pin</td>
<td>Any pin except PORT5 pins (for PSoC 4xxxM).</td>
<td>Any pin in PORT5 (for PSoC 4xxxM).</td>
</tr>
<tr>
<td>Max Number of</td>
<td>47 (for PSoC 4xxxM)</td>
<td>4* (for PSoC 4xxxM)</td>
</tr>
<tr>
<td>CapSense Pins</td>
<td>68 (for PSoC 4xxxL)</td>
<td>22* (for PSoC 4xxxL)</td>
</tr>
</tbody>
</table>

* Maximum number of pins are specified, excluding two pins used for \( \text{CMOD} \) and \( \text{CSH}\_\text{TANK} \) 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 v3.0 Component does not support sensing using the CSD1 block for the PSoC 4-M series. If you need to use both CapSense blocks, you should use the CapSense_CSD v2.40 Component.
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:

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

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

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

        /* Start scanning all enabled sensors */
        CapSense_1_ScanEnabledWidgets();
        CapSense_2_ScanEnabledWidgets();
    }
}
```
Cypress provides a complete set of hardware and software tools to develop your CapSense application.

### 4.1 PSoC Creator

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

#### 4.1.1 CapSense Component

PSoC Creator provides a CapSense Component, which is used to create a capacitive touch system in PSoC 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 [Datasheet](#) to see if your device supports this Component).

![Figure 4-1. PSoC Creator Component Placement](#)

**Features**

- Offers best-in-class signal-to-noise ratio (SNR)
- Supports Self-Capacitance (CSD) and Mutual-Capacitance (CSX) sensing methods
- Features SmartSense™ auto-tuning technology for CSD sensing to avoid complex manual tuning process
- Supports various Widgets, such as Buttons, Matrix Buttons, Sliders, Touchpads, and Proximity Sensors
- Provides ultra-low power consumption and liquid tolerant capacitive sensing technology
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 Helper, to help with the tuning process.

### 4.1.2 CapSense_ADC Component

The CapSense_ADC Component is only applicable for the PSoC 4 S-Series, PSoC 4100S Plus, and PSoC 4100PS devices. This Component should be used when both CapSense and ADC operations are required. Using the CapSense block for ADC operation, touch functionality is performed in a time-multiplexed manner.

### 4.1.3 Tuner Helper

Tuner Helper is included with the CapSense Component and assists in tuning CapSense parameters and monitoring sensor data such as raw count, baseline, and difference count. See the respective CapSense Component datasheet for the detailed procedure on how to use Tuner Helper.

### 4.1.4 Example Projects

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

Figure 4-2. PSoC Creator Example Project
4.2 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. The CapSense® Configurator Guide explains steps for a simple CapSense Linear Slider example to get started. You can also refer to the CapSense® Tuner Guide for tuning your CapSense design.

Note that PSoC Creator does not support all the PSoC 6 devices. The CY8C62x8 and CY8C62xA device families are supported in ModusToolbox only. The CY8C6xx7 device family is supported in ModusToolbox and PSoC Creator 4.2.

4.3 Hardware Kits

Table 4-1 lists the development kits that support evaluation of PSoC 4 and PSoC 6 CapSense.

<table>
<thead>
<tr>
<th>Development Kit</th>
<th>Supported CapSense Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSoC 4000 Pioneer Kit (CY8CKIT-040)</td>
<td>A 5x6 CapSense touchpad and a wire proximity sensor</td>
</tr>
<tr>
<td>PSoC 4 S-Series Pioneer Kit (CY8CKIT-041)</td>
<td>Two self- or mutual-capacitive sensing buttons A 7x7 self- or mutual-capacitive sensing touchpad</td>
</tr>
<tr>
<td>PSoC 4 S-Series Prototyping Kit (CY8CKIT-145)</td>
<td>Three self- or mutual-capacitive sensing buttons A five-segment self- or mutual-capacitive sensing linear slider</td>
</tr>
<tr>
<td>PSoC 4100S Plus Prototyping Kit (CY8CKIT-149)</td>
<td>Three self- or mutual-capacitive sensing buttons A six-segment self- or mutual-capacitive sensing linear slider</td>
</tr>
<tr>
<td>PSoC 4 Pioneer Kit (CY8CKIT-042)</td>
<td>A five-segment linear slider</td>
</tr>
<tr>
<td>PSoC 4 BLE Bluetooth Low Energy Pioneer Kit (CY8CKIT-042-BLE)</td>
<td>A five-segment linear slider and a wire proximity sensor</td>
</tr>
<tr>
<td>PSoC 4200-M Pioneer Kit (CY8CKIT-044)</td>
<td>A five-element gesture detection and two proximity wire sensors</td>
</tr>
<tr>
<td>PSoC 4200-L Pioneer Kit (CY8CKIT-046)</td>
<td>A five-element gesture detection, two proximity wire sensors, and an eight-element radial slider</td>
</tr>
<tr>
<td>PSoC 4100PS Prototyping Kit (CY8CKIT-147)</td>
<td>No onboard CapSense sensors. The kit can be used to connect external sensors to any I/O pin.</td>
</tr>
<tr>
<td>CapSense Proximity Shield (CY8CKIT-024)</td>
<td>A four-element gesture detection and one proximity loop sensor</td>
</tr>
<tr>
<td>CapSense® Liquid Level Sensing Shield (CY8CKIT-022)</td>
<td>A two-element flexible PCB and 12-element flexible PCB</td>
</tr>
<tr>
<td>PSoC 4 Processor Module (CY8CKIT-038), with PSoC Development Kit (CY8CKIT-001)</td>
<td>A five-segment linear slider and two buttons</td>
</tr>
<tr>
<td>CapSense Expansion Board Kit (CY8CKIT-031), to be used with CY8CKIT-038 and CY8CKIT-001</td>
<td>A 10-segment slider, five buttons and a 4x4 matrix button with LED indication</td>
</tr>
<tr>
<td>MiniProg3 Program and Debug Kit (CY8CKIT-002)</td>
<td>CapSense performance tuning in CY8CKIT-038</td>
</tr>
<tr>
<td>PSoC 6 Wi-Fi BT Pioneer Kit (CY8CKIT-062-WiFi-BT Pioneer Kit) and PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE Pioneer Kit)</td>
<td>A 5-segment CapSense Slider, two CapSense buttons, one CapSense proximity sensing header, a proximity sensor</td>
</tr>
<tr>
<td>PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-063-4343W)</td>
<td>A 5-segment CapSense Slider and two mutual-cap CapSense buttons</td>
</tr>
</tbody>
</table>
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 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, to allow runtime tuning of CapSense parameters. SmartSense is recommended mainly for conventional CapSense applications involving simple button and slider widgets, and is currently supported only for Self-Capacitance Sensing and not for Mutual-Capacitance Sensing.

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

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

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

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

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

Note that manual tuning requires I²C or UART communication with a host PC.
5.2 SmartSense

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

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

- **Performance is independent of PCB variations**: The parasitic capacitance, CP, 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 CP 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 CP.

- **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 firmware tuning parameters automatically at runtime. This is the recommended method for most designs.

2. **SmartSense (Hardware parameters only)** – This method auto-tunes all hardware parameters of CapSense automatically, but allows to set user-defined threshold values. This method consumes less flash/RAM resources than SmartSense (Full Auto-Tune). Also, this method avoids the extra processing needed for automatic threshold calculation and hence allows lower power consumption for a given scan rate. Use this method for low-power or noisy designs or in cases with constrained memory requirements.

3. **SmartSense for initial tuning** – You may also use SmartSense for initial tuning, to quickly find the best settings for a CapSense board and then change to manual tuning. This method is useful for cases with strict requirements on response time or power consumption. This is a quick method to find the best settings, instead of starting manual tuning from scratch.
5.2.1 Component Configuration for SmartSense

This section explains the Component configuration for the SmartSense mode. For details on manual tuning, see Manual Tuning. It is recommended to use the latest CapSense Component for all newer designs. If your design uses older versions of the CapSense Component, see the respective component datasheets for design guidance.

5.2.1.1 SmartSense Configuration

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

5.2.1.1.1 Basic Settings

Set the Basic tab configuration according to Figure 5-2.

Figure 5-2. CapSense Component Basic Tab

- **Widget Type**: Specify the widget type by clicking the + symbol and then selecting the widget (such as buttons, liner slider, and radial slider) in the drop-down list. Repeat this process until all the widgets are added to the Component.

- **CSD tuning mode**: Select the SmartSense (Full Auto-Tune) tuning method if you want to automatically tune all the hardware and software parameters; otherwise, select the SmartSense (Hardware parameters only) option.
CapSense Performance Tuning

to control the threshold settings manually. Table 5-1 lists the parameters that are automatically set by SmartSense, depending on the mode selected.

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Calculation Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finger Threshold</td>
<td>Calculated once on CapSense startup based on the selected Finger capacitance and updated after each sensor scan.</td>
</tr>
<tr>
<td>Noise Threshold</td>
<td>Manual selection</td>
</tr>
<tr>
<td>Hysteresis</td>
<td></td>
</tr>
<tr>
<td>Negative Noise Threshold</td>
<td></td>
</tr>
<tr>
<td>Scan Resolution</td>
<td></td>
</tr>
<tr>
<td>Compensation IDAC</td>
<td></td>
</tr>
<tr>
<td>Modulator IDAC</td>
<td></td>
</tr>
<tr>
<td>Sense Clock Frequency</td>
<td>Calculated once on CapSense startup.</td>
</tr>
<tr>
<td>Modulator Clock Frequency</td>
<td></td>
</tr>
<tr>
<td>Low Baseline Reset</td>
<td></td>
</tr>
</tbody>
</table>

c. Sensing mode: Select the CapSense sensing mode - CSD (Self-cap) or CSX (Mutual-cap) for each widget based on your hardware design. SmartSense currently supports only CSD widgets and not CSX widgets. CSX widgets must be tuned manually.

d. Sensing element(s): This field allows you to specify the number of sensors in each CSD widget. This option is useful in applications where there are multiple sensors that are identical, that is, have the same $C_F$ and sensor area. The SmartSense method will tune all the sensors under a single widget for same sensitivity. Having multiple sensors under a single widget saves processing time and reduces memory footprint. You can configure the number of sensors depending on the hardware design.

e. Finger Capacitance: This parameter indicates the minimum value of $C_F$ that should be sensed as a valid touch by the CapSense Component. If the actual $C_F$ added when the finger touches the button sensor is less than the value specified in the Component configuration window, the sensor status will not change to ‘1’ on touching the button; however, if the actual $C_F$ added by the finger is higher than the value specified in the Component configuration window, the sensor status will change to ‘1’ on a touch.

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:

- **For button sensors:** Start by specifying the highest value (from the available options in the list) and check the SNR and button status when the button is touched. Decrease the Finger capacitance parameter value until the button status changes to ‘1’ on touch and SNR is greater than 5. Figure 5-3 shows the detailed steps to find the right value for the Finger capacitance parameter in your design. 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 the SNR is less than 5:1 even when the smallest allowed value of finger capacitance 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.
Figure 5-3. Using SmartSense Auto-Tuning Based CapSense Project in PSoC Creator

For sliders: Set Finger capacitance to 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. Otherwise, decrease the Finger capacitance value until the above condition holds true.

If above 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 5-4 explains the process of setting finger capacitance value for sliders.
Start

Set the Finger capacitance value to the maximum allowed capacitance value

Slide Finger over Slider and Monitor Difference Count i.e. Sensor Signal

At any finger position, do at-least two slider segments provide Difference Count i.e. Sensor Signal > 0

Decrease Finger capacitance value by one unit

Yes

Is Finger capacitance >= Minimum allowed Finger capacitance value

No

A Hardware Change may be Required.
Review Slider Design* or use Manual tuning**

Yes

At any finger position, does at-least one slider-segment provide an SNR > 5:1 and Sensor signal > 50?

No

Yes

End

* 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.
5.2.1.1.2 Advanced Settings

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

The Advanced tab has the following sub-tabs.

5.2.1.1.2.1 General Tab

**General** tab contains the parameters common for all widgets regardless of the sensing method used for the widgets. This tab allows you to configure the following features:

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

Figure 5-5. General Tab under Advanced Settings

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

c. **Baseline IIR filter settings**: This parameter controls the Baseline update rate. The baseline is an IIR-filtered version of raw count. The baseline coefficient value is the parameter 'N' in the IIR filter equation listed in Table 5-2. This parameter controls the rate at which the baseline is updated. You can start with a lower value of N (N=1) and keep increasing the value until the baseline tracks slow variations in raw counts.

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

e. **Enable multi-frequency scan**: Enabling multi-frequency scan performs a triple-sensor scan with different frequencies. 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.

<table>
<thead>
<tr>
<th>Table 5-2. Raw Data Noise Filters in CapSense Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>First Order IIR</td>
</tr>
</tbody>
</table>

5.2.1.2.2 **CSD Settings**
The **CSD Settings** tab allows you to configure settings related to CSD widgets as shown in Figure 5-6.
a. **Scan settings**: The scan settings parameter allows you to specify the following parameters:

- **Modulator clock frequency (kHz)**: This parameter should be set to maximum value as it results in shorter scan time and hence helps in reducing average power consumption. See the Modulator Clock Related Parameters section for details on the recommended values of this parameter.

- **Sense clock source**: This parameter is used to specify the source for the sense clock. The options available are Direct, 8-bit pseudo random sequence (PRS), 12-bit PRS, Auto, and SSCx (SSC – Spread Spectrum, applicable only for PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family). The “Auto” option automatically selects the length of the PRS sequence (for all PSoC 4 family of devices) or SSCx value (only for PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family). You should use PRS or SSCx if your design has strict electromagnetic compatibility requirements, as it reduces the electromagnetic emission. See the Sense Clock Related Parameters section for details on the clock source selection guidelines.

- **Inactive sensor connection**: CapSense scans one sensor at a time. This option determines the connection of sensors when they are not being scanned.

  The “Ground” option is recommended because it reduces noise on the scanned sensors. For liquid-tolerant designs, this option should be specified as “Shield”.  
  
  **Note** You should select the Enable shield electrode option to specify the “Inactive sensor connection” option to “Shield”.

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

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

- **Enable shield tank capacitor**: Enable this option if you are using a C_{SH,TANK} capacitor; see CapSense CSD Shielding for details.

- **Shield electrode delay**: For proper operation of the shield electrode, the shield signal should match the sensor signal in phase. You can use an oscilloscope to view both sensor and shield signals to verify this delay.
condition. If they are not aligned, use this option to add delay to the shield signal to align the two signals. The available delays vary depending on the device selected.

- **Shield SW resistance:** This parameter controls the shield signal rise and fall times to reduce EMI. This parameter is valid only for PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices. It is recommended to start with the “Low EMI” setting because it dynamically controls the slew rate of shield signal. If there is electromagnetic emission due to shield switching, you can set this parameter to either “low”, “medium (default)”, or “high” value.

- **Number of shield electrode:** 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.

5.2.1.1.2.3 CSX Settings

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

5.2.1.1.2.4 Widget Details

The **Widget Details** tab allows you to configure the thresholds (Figure 5-7) and specify the ganged sensor elements (Figure 5-8). If the CSD tuning mode is selected as SmartSense (Full Auto-Tune) in Figure 5-2, thresholds are auto-tuned by the CapSense component and hence appear as Set by SmartSense in the Widget Details tab.

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

Figure 5-7. Widget Details Tab Settings
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 of the parameters automatically. However, there may be situations where you need to have strict control over the parameters. For example, use manual tuning if you need to strictly control the time PSoC 4 takes to scan a group of sensors or strictly control the sense clock frequency of each sensor (this can be done to reduce EMI in systems).

- Supports higher parasitic capacitances: SmartSense supports parasitic capacitances as high as 45 pF for a 0.2-pF finger capacitance, and as high as 35 pF for a 0.1-pF finger capacitance. If the parasitic capacitance is higher than the value supported by SmartSense, you should use manual tuning.

The manual tuning process can be summarized in the following three steps and is shown in Figure 5-9.

1. Set initial values of CapSense Component Hardware Parameters.
2. Tune CapSense component hardware parameters to ensure Signal-to-Noise Ratio is greater than 5:1 with a signal of at least 50 counts while meeting the system timing requirements.
3. Set optimum values of CapSense Component Software Parameters (threshold parameters).
The following sections describe the fundamentals of manual tuning and the above three steps in detail. Knowledge of the PSoC 4 CapSense architecture is a prerequisite for these sections. See Capacitive Touch Sensing Method and CapSense CSD Sensing to become familiar with PSoC 4 CapSense architecture. Depending upon the sensing method selected, the manual tuning procedure will differ. See the CSD Sensing Method section for manual tuning using CSD for self-capacitance and see the CSX Sensing Method section for manual tuning using CSX for mutual-capacitance. You can skip these sections if you are not planning to use manual tuning in your design.

Figure 5-9. 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 a manual tuning procedure. The section ends with three examples on manual tuning of button, slider and proximity widgets.

5.3.2.1 Basics

5.3.2.1.1 Conversion Gain and CapSense Signal
Conversion gain will influence how much signal the system sees. If there is more gain, the signal is higher and a higher signal means a higher achievable Signal-to-Noise Ratio. 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.

5.3.2.1.1.1 Conversion Gain in Single IDAC Mode
In the single IDAC mode, the raw count is directly proportional to the sensor capacitance.

Equation 5-1. 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 recommended IDAC Sourcing mode, according to Equation 3-6 and Equation 5-1 is:

Equation 5-2. Capacitance to Digital Converter Gain
\[ G_C = \left(2^N - 1\right) \frac{V_{\text{REF}} F_{SW}}{I_{\text{MOD}}} \]

Where,

\( V_{\text{REF}} \) is the comparator reference voltage, the value of \( V_{\text{REF}} \) is fixed and is equal to 1.2 V for all PSoC 4 family of devices except the PSoC 4 S-Series. For PSoC 4100S Plus, PSoC 4100PS, family and for the PSoC 6\(^1\) family, \( V_{\text{REF}} \) can vary from 0.6 V to \( V_{\text{DDA}} - 0.6 \) V.

\( F_{SW} \) is the Sense clock frequency

\( I_{\text{MOD}} \) is the Modulator IDAC current

The tunable parameters of the conversion gain are \( V_{\text{REF}} \) (for PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family only), \( F_{SW}, I_{\text{MOD}}, \) and \( N \). Figure 5-10 shows a plot of raw count versus sensor capacitance.

**Note:** The CapSense Component automatically select the \( V_{\text{REF}} \) value depending on the \( V_{\text{DDA}} \) voltage.
The change in raw counts when a finger is placed on the sensor is called CapSense signal. Figure 5-11 shows how the value of the signal changes with respect to the conversion gain.

Figure 5-11 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 $C_F$ (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 5-12 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 5-12. Recommended Tuning

5.3.2.1.1.2 Conversion Gain in Dual IDAC Mode
The equation for raw count in the dual IDAC mode, according to Equation 5-2 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 5-13 shows, to ensure high conversion gain, to avoid Flat Spots, and to avoid raw count saturation due to environmental changes.
As Figure 5-13 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:

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

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 $G_{C3}$ 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 5-11. 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 the figure Voltage across Sensor Capacitance 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 5-12 and Figure 5-13.

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 5-10 and Figure 5-13 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 5-14 shows. This range is known as a dead-zone or a flat-spot.

Figure 5-14. Flat Spots in Raw Counts versus Sensor Capacitance when Direct Clock is Used

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

For almost all systems, we recommend using PRS as the Sense Clock source because it limits the impact of flat spots and also provides EMI/EMC benefits as indicated in 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 your particular system.

Table 5-3. CapSense Component Hardware Parameters

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>CapSense Component Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sense Clock Frequency</td>
</tr>
<tr>
<td>2</td>
<td>Sense Clock Source</td>
</tr>
<tr>
<td>3</td>
<td>Modulator Clock Frequency</td>
</tr>
<tr>
<td>4</td>
<td>Modulator IDAC</td>
</tr>
<tr>
<td>5</td>
<td>Compensation IDAC</td>
</tr>
<tr>
<td>6</td>
<td>Scan Resolution</td>
</tr>
</tbody>
</table>
5.3.2.2.1 Sense Clock Related Parameters

For a given resolution \( N \), you can vary both \( F_{SW} \) and IDACs to make the raw count corresponding to \( C_p \) equal to 85 percent of the maximum raw count value. However, selecting an improper sense clock frequency \( (F_{SW}) \) can affect the operation of CapSense CSD. From Equation 5-2, it is best to use the maximum clock frequency but ensure a full charge and discharge of the sensor capacitor as indicated in Figure 3-5.

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

The charge and discharge paths of the sensor capacitor include series resistances that slow down the charging and discharging process as indicated by the \( R_{Series} \) and GPIO cell switch in Figure 3-3 and Figure 3-5.

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

\[
T_{SW}(\text{minimum}) = 10R_{SeriesTotal}C_p
\]

\[
F_{SW}(\text{maximum}) = \frac{1}{10R_{SeriesTotal}C_p}
\]

The typical value of the GPIO resistance is 500 \( \Omega \) and the recommended external resistance is 560 \( \Omega \) (see Series Resistors on CapSense Pins for details). Therefore, take the value of \( R_s \) as 1.06 k\( \Omega \) when calculating the maximum switching frequency. The source clock that drives \( F_{SW} \) is called as Sense Clock Source.

By default, the system will use a direct clock as sense clock source that drives \( F_{SW} \). In most cases because of Flat Spots, and especially if your system is susceptible to EMI/EMC noise, it is recommended to enable the PRS clock option. In addition to Direct and PRS clock sources, the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS and PSoC 6 family of devices support spread spectrum clock generation. In this case, the frequency of the sense clock is spread over a predetermined range to reduce EMI. The frequency spread range can be specified by selecting the clock divider as SSC2 to SSC5. The typical example is SSC3, and it is recommended to be \( \pm 10\% \) from the selected centre frequency of sense clock. See the PSoC 4000S TRM for details on the spread spectrum architecture.

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

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

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

\[
T_{PRS} = \frac{2^{\text{PRS Resolution} - 1}}{\text{Sense Clock Frequency}}
\]

After enabling PRS, make sure that \( T_{SCAN} \geq T_{PRS} \). By choosing PRS-Auto, this process will be automatically done for you. After selecting the sense clock frequency for PRS in the CapSense configuration window, the maximum frequency and the average frequency are displayed in the configuration window. Average frequency is half of the maximum sense clock frequency. The sensor should charge and discharge completely with respect to the maximum sense clock frequency and should satisfy Equation 5-6.
For a ready reference on $C_P$ versus sense clock frequency for 560 ohm series resistance, you can also see the CapSense Component Datasheet.

5.3.2.2.2 Modulator Clock Related Parameters
The modulator clock will impact scan time (Equation 3-4) as well as Signal-to-Noise Ratio. In general, it is recommended to choose the highest modulator clock frequency. This will give the best possible scan times as well as reduce measurement error as much as possible.

5.3.2.2.3 Modulator and Compensation IDACs
CSD supports two IDACs, Modulator 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 and manual tuning is used.

5.3.2.2.3.1 Compensation IDAC
For most cases, it is recommended to enable compensation IDAC for increased conversion gain and CapSense signal. Disable compensation IDAC where absolute capacitance measurement is required, or to free the IDAC for other analog functions.

5.3.2.2.3.2 Auto-calibration
This feature enables the firmware to automatically calibrate the IDAC to achieve the required calibration target of 85%. Enable auto-calibration for most cases. If you want a different calibration level, for example to decrease calibration percentage due to environment changes, you may also change IDAC auto-calibration percentage using API CapSense_CSDCalibrateWidget().

Disable IDAC auto-calibration where a change in $C_P$ needs to be detected by measuring 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

5.3.2.2.3.3 Selecting DAC codes
It is recommended to enable auto-calibration for most cases. However, if you want to disable auto-calibration for any reason, you may first configure CapSense Component with auto-calibration enabled, and read back the calibrated IDAC values using CapSense Tuner. Then, re-configure the CapSense Component to disable auto-calibration and use the fixed DAC codes read back from the tuner.

5.3.2.2.4 Scan Resolution
Scan resolution needs to be selected to maintain a balance between the signal and scan time. As resolution increases the signal will increase (Equation 5-2) but so will the scan time (according to Equation 3-4). For each bit, both will double. For most sensor designs, it is recommended to begin with the lowest possible resolution and slowly increase this value to increase signal. If there is room in the timing budget, the resolution can be increased to achieve an SNR beyond the minimum 5:1 recommendation to increase system robustness.

5.3.2.3 Selecting CapSense Software Parameters
CapSense software parameters govern the sensor status based on the raw count of a sensor. Table 5-4 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 5-4. CapSense Component Software Parameters

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>CapSense Component Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Finger Threshold</td>
</tr>
<tr>
<td>2.</td>
<td>Noise Threshold</td>
</tr>
<tr>
<td>3.</td>
<td>Hysteresis</td>
</tr>
<tr>
<td>4.</td>
<td>ON Debounce</td>
</tr>
<tr>
<td>5.</td>
<td>Sensor Auto-Reset</td>
</tr>
<tr>
<td>6.</td>
<td>Low Baseline Reset</td>
</tr>
<tr>
<td>7.</td>
<td>Negative Noise Threshold</td>
</tr>
</tbody>
</table>

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 5-15 shows the concept of raw count, baseline, and signal.

![Figure 5-15. Raw Count, Baseline, and Signal](image)

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

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

Baseline is basically a low-pass-filtered version of raw counts. As Figure 5-16 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 5-16. Baseline Update Algorithm](image)

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 5-17 shows.

![Figure 5-17. Low Baseline Reset](image)

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.

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

Note that signal in the above equation refers to the difference: Raw Count – Baseline, when the sensor is touched, as Figure 5-15 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 5-18 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.

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 5-16 shows. In other words, the baseline remains constant as long as the raw count is above baseline + noise threshold. This prevents the baseline from following raw counts during a finger touch.

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

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

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

5.3.2.3.6 Negative Noise Threshold

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

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

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

5.3.2.3.7 Low Baseline Reset

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

If a finger is placed on the sensor during device startup, the baseline is initialized to the high raw count value at startup. When the finger is removed, raw counts fall to a lower value. In this case, the baseline should track the low raw counts. The Low Baseline Reset parameter helps to handle this event. It resets the baseline to the low raw count value when the number of low samples reaches the low baseline reset number. Note that in this case, when the finger is removed from the sensor, the sensor will not respond to finger touches for a low baseline reset time given by Equation 5-10.
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.

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

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

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

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

5.3.2.3.9 Sensor Auto Reset

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

When auto reset is disabled, the baseline only updates if the current raw count is within a range of \((\text{Baseline} - \text{Negative Noise Threshold})\) to \((\text{Baseline} + \text{Noise Threshold})\) as Figure 5-16 shows and the Baseline Update Algorithm describes. However, when Auto Reset is enabled, baseline is always updated if the current raw count is higher than \((\text{Baseline} - \text{Negative Noise Threshold})\) as Figure 5-19 shows.
Because the baseline is always updated when sensor auto reset is enabled, this setting limits the maximum time duration for which the sensor will be reported as pressed. However, enabling this parameter prevents the sensors from permanently turning on if the raw count suddenly rises without anything touching the sensor. This sudden rise can be caused by a large power supply voltage fluctuation, a high-energy RF noise source, or a very quick temperature change.

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

5.3.2.4 Button Widget Example

The following examples explain tuning of self-capacitance based button widgets in a PSoC Creator CapSense Component using the Tuner Helper. For details on the Component and all related parameters, 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 5-20 shows.
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 5-21 and explained in Table 5-5.
Table 5-5. CapSense Component CSD Configuration Window

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulator clock frequency</td>
<td>Maximum available option</td>
<td>Higher modulator clock frequency reduces sensor scan time, which results in lower power; hence, it is recommended to use the highest possible frequency.</td>
</tr>
<tr>
<td>Sense clock source</td>
<td>Auto</td>
<td>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 and Sense Clock Related Parameters sections for more information on choosing direct or SSC clocks.</td>
</tr>
<tr>
<td>Inactive sensor connection</td>
<td>Ground (default)</td>
<td>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.</td>
</tr>
<tr>
<td>Enable IDAC auto-calibration</td>
<td>Enabled</td>
<td>Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (85 percent in single-IDAC mode and 70 percent for dual-IDAC mode). For systems that may need a different calibration point because of environmental factors, see Modulator and Compensation IDACs.</td>
</tr>
<tr>
<td>Enable compensation IDAC</td>
<td>Enabled</td>
<td>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 Modulator and Compensation IDACs.</td>
</tr>
<tr>
<td>Enable shield electrode</td>
<td>Per PCB design</td>
<td>Enable shield if your design requires large proximity sensing distance, Liquid Tolerance, or if the shield is being used to reduce C of sensors.</td>
</tr>
</tbody>
</table>
5. In the Advanced tab > Widget Details window, specify the settings as shown in Figure 5-22 and Figure 5-23 and explained in Table 5-6 and Table 5-7.

Figure 5-22. CapSense Component Widget Details Window

Figure 5-23. CapSense Component Widget Details Window – Sensor Settings
Table 5-6. CapSense Component Widget Details Window

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sense clock frequency</td>
<td>( \frac{1}{10R_{\text{Series}}C_{P}} )</td>
<td>See Sense Clock Related Parameters to choose the appropriate frequency.</td>
</tr>
<tr>
<td>Scan resolution</td>
<td>8-bit</td>
<td>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.</td>
</tr>
<tr>
<td>Modulator IDAC</td>
<td>NA</td>
<td>With auto-calibration enabled, the device automatically chooses this value. To choose a different IDAC value, see Modulator and Compensation IDACs.</td>
</tr>
<tr>
<td>Finger threshold (FT)</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
<tr>
<td>Noise threshold</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
<tr>
<td>Negative noise threshold</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
<tr>
<td>Low baseline reset</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
<tr>
<td>Hysteresis</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
<tr>
<td>ON Debounce</td>
<td>Default</td>
<td>Widget Threshold Parameters will be adjusted in Step 11 of the tuning process.</td>
</tr>
</tbody>
</table>

Table 5-7. CapSense Component Widget Details Window – Sensor Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensation IDAC</td>
<td>NA</td>
<td>With auto-calibration enabled, the device automatically chooses this value. To choose a different IDAC value, see Modulator and Compensation IDACs.</td>
</tr>
<tr>
<td>Selected Pins</td>
<td>Default</td>
<td>This parameter allows you to gang multiple sensors and scan as a single sensor.</td>
</tr>
</tbody>
</table>

6. Next, use the tuner to observe raw counts in the Graphing tab in Tuner GUI and calculate the Signal-to-Noise Ratio of the sensor. See CapSense Component datasheet for the detailed procedure on how to add tuner to your project. To calculate SNR, measure the raw count value when a finger is present and divide it by the peak-to-peak noise of the raw counts with no touch present. Based on your end system design, test with a finger that matches the size of your normal use case. Typically finger size targets are ~8 – 9 mm.

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 – 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.

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

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

Note that resolution can be updated directly in the Widget/Sensor Parameter window, as Figure 5-24 shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as Figure 5-25 shows, and reprogram the device to update filter settings. For details on filters, see the CapSense Component Datasheet.
Figure 5-24. Update Scan Resolution in Tuner GUI

Figure 5-25. Update Filter Settings in Configure ‘CapSense_CSD_P4’ Dialog
9. Check the total scan time (see Figure 5-26) 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 5-26. Sensor Scan Time in Scan Order Tab

10. If you meet the timing requirement of the system, skip to step 11. Otherwise, adjust the tuning to speed up the scan time. If SNR is greater than 10 on any sensor, then you can lower your resolution or remove filters to decrease scan time, but keep your SNR greater than 5. If you are unable to meet your timing requirements and maintain SNR greater than 5, you should see step 12.

11. After you have confirmed that your design meets the timing parameters, and the SNR is greater than 5, set the widget threshold parameters for your design as follows. Ensure that you observe the difference count (signal output) in the Graph View tab in Tuner GUI, not the raw count output for setting these thresholds. Based on your end system design, test the signal with a finger that matches the size of your normal use case. Typically, finger size targets are ~8 - 9 mm. Consider testing with smaller sizes that should be “rejected” by the system to ensure they do not reach the finger threshold.

   Finger Threshold = 80 percent of signal
   Noise Threshold = 40 percent of signal
   Negative Noise Threshold = 40 percent of signal
   Hysteresis = 10 percent of signal
   Debounce = 3

   Again, these settings can be first set in the tuner GUI, as Figure 5-27 shows, or they can be input directly in the CapSense Component customizer, as Figure 5-28 shows.

   For more information on these settings, see Selecting CapSense Software Parameters.
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 5-27. Updating Threshold Parameters in Tuner GUI

Figure 5-28. Updating Threshold Parameters in Configure ‘CapSense_CSD_P4’ Dialog
CapSense Performance Tuning

Figure 5-29. CSD Button Widget Tuning Flow Chart

Start

Set Initial parameters (leave all others as default):
Modulator clock frequency,
Sense clock frequency and
Resolution

Step 1

Measure SNR

Step 2

Is SNR > 5?

Yes

Decrease Resolution/Adjust Filters

Step 6

No

Is SNR > 5?

Yes

Step 5

Do any buttons have SNR > 10?

No

Does the system meet timing requirements?

Step 7

Yes

Set the following system thresholds based on raw count value with a finger present:
Finger Threshold = 80%
Noise Threshold = 40%
Negative Noise Threshold = 40%
Hysteresis = 10%
Debounce = 3

End

* 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

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

To avoid nonlinearity in the centroid, ensure that the signal from all the slider segments is equal, as Figure 5-30 shows. Here, the signal for each slider segments is the shift in Raw Count minus the Noise Threshold, when a finger is placed at the center of the slider segment. If the signal of the slider segments is different, then the centroid will be nonlinear, as Figure 5-31 shows. Note that a centroid of 0xFF is reported 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 5-30. Response of Centroid Versus Finger Location when Signals of All Slider Elements Are Equal

Note

Signal = Difference count – Noise Threshold
Difference count = Raw Count - Baseline

Figure 5-31. Response of Centroid Versus Finger Location when the Signal of All Slider Elements Are Different

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

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

2. Follow the manual tuning procedure shown in Figure 5-29 to tune the slider segment, which has the maximum $C_p$ value among all the slider segments.
3. For the remaining segments in the slider, set the same value as that of the maximum Cₚ slider segment for the following parameters:
   - Scan resolution
   - Sense clock divider
   - Modulator clock divider
   - Modulation IDAC

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

   The above parameters are widget-level parameters – they need not be set individually for each segment.

4. Measure the Signal-to-Noise Ratio for each slider segment and ensure that it is greater than 5:1 for all the slider segments. If the SNR is not greater than 5:1 for any slider segment, increase the resolution by one bit for all slider segments, until the SNR is greater than 5:1 for all segments.

5. After confirming that the SNR for all slider segments is greater than 5:1, set the following thresholds to the value listed in Table 5-8.

<table>
<thead>
<tr>
<th>Threshold Parameter</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finger Threshold</td>
<td>80 percent of the signal</td>
</tr>
<tr>
<td>Noise Threshold</td>
<td>40 percent of the signal</td>
</tr>
<tr>
<td>Negative Noise Threshold</td>
<td>40 percent of the signal</td>
</tr>
<tr>
<td>Low-Baseline Reset</td>
<td>Set to 30</td>
</tr>
</tbody>
</table>

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

CapSense component v3.0 and later support Mutual-Capacitance sensing as well. In a mutual-capacitance sensing system, the raw count is directly proportional to the mutual capacitance between the Tx and Rx sensors, as Equation 5-13 shows.

Equation 5-13. Raw Count Relationship to Sensor Capacitance

\[ \text{raw count} = G_{\text{CSX}} C_M \]

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

Equation 5-14. Capacitance to Digital Converter Gain

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

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

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

Note The value of \( N \) is controlled by the Number of Sub-Conversions parameter, as explained in the CapSense CSX Sensing Method section.

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

5.3.3.2 Selecting CapSense Hardware Parameters

CapSense hardware parameters govern the conversion gain and CapSense signal. Table 5-9 lists the CapSense hardware parameters that apply to CSX sensing method. The following section gives guidance on how to adjust these parameters to achieve optimal performance for your particular system.

Table 5-9. CapSense Component Hardware Parameters

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>CapSense Component Parameter Name</th>
<th>CapSense v3.0 or later</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Modulator Clock Frequency</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tx Clock Source</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tx Clock Frequency</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Number of Sub-Conversions</td>
<td></td>
</tr>
</tbody>
</table>

5.3.3.2.1 Modulator Clock Frequency

It is best to choose the highest allowed clock frequency for the given device as higher modulator clock frequency leads to higher sensitivity, and hence increased accuracy for a given \( C_M \) to digital count conversion. Also, a higher value of \( F_{\text{mod}}/F_{\text{tx}} \) ensures lower width of Flat Spots in the \( C_M \) to rawcount conversion.
5.3.3.2.2 Tx Clock Source

The Tx clock source parameter is only applicable for the PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices. The following clock sources are available for Tx clock:

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

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

5.3.3.2.3 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 and it can be verified by checking the signal in the oscilloscope or it can be set using the below equation

\[ F_{Tx} < \frac{1}{10R_{SeriesTotal}C_p} \]

To minimize the scan-time, it is recommended to use the maximum clock frequency available in the component drop-down list that satisfies the above-mentioned criteria.

5.3.3.2.4 Number of Sub-Conversions

The number of sub-conversions parameters decides the sensitivity of the sensor and should be selected such that the following criteria is met.

\[ \left( \frac{\text{Modulator Clock}}{\text{Tx Clock}} \times \text{Number of Sub Conversions} \right) < 2^{16} \]

Where, \( 2^{16} \) is the maximum counter value

Increasing the number of sub-conversions increases the signal and scan time of the sensor i.e. for a fixed modulator clock and Tx clock, the above equation shows that the resolution of the CapSense raw counter is directly proportional to the number of sub-conversions parameter. Reducing the number of sub-conversions by half, reduces the counter resolution by 1-bit and reduces the scan time by half, which in turn reduces the sensor signal, as shown in Equation 3-10. See Button Widget Example section for details on how to configure optimum value for “number of sub-conversions” parameter.

5.3.3.3 Selecting CapSense Software Parameters

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

5.3.3.4 Button Widget Example

This example explains tuning of mutual-capacitance-based button widget in a PSoC Creator CapSense Component using Tuner Helper. For details on the Component and all related parameters, see the Component Datasheet.

Follow the detailed process listed in the following steps 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.
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.
Table 5-10. CapSense Component General Configuration Window

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulator clock frequency</td>
<td>Maximum available option</td>
<td>Higher modulator clock frequency reduces sensor scan time and therefore results in lower power, and lower noise in the raw counts; hence it is recommended to use the highest possible frequency.</td>
</tr>
<tr>
<td>Tx clock source</td>
<td>Auto</td>
<td>This option is available only for PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 MCU family of devices. For other devices, the Tx clock is always direct. Enabling Auto (chooses optimum value of spread spectrum clock spread) helps to deal with EMC/EMI. See the Tx Clock Source section for more information on choosing Direct or SSC clocks.</td>
</tr>
<tr>
<td>Enable IDAC auto-calibration</td>
<td>Enabled</td>
<td>Enabling auto-calibration allows the device to automatically choose the optimal IDAC calibration point (for CSX this is 40 percent of maximum value).</td>
</tr>
</tbody>
</table>

Figure 5-34. CapSense Component Widget Details Window

Table 5-11. CapSense Component Widget Details Window

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx clock frequency</td>
<td>$\frac{1}{10R_{\text{SeriesTotal}}C_p}$</td>
<td>It is recommended to set the highest clock frequency that satisfies the condition $Tx \text{ Clock Freq} &lt; \frac{1}{10R_{\text{SeriesTotal}}C_p}$ The Tx pin can be probed using an oscilloscope to check if the sensor is completely charged and discharged.</td>
</tr>
<tr>
<td>Number of sub-conversions</td>
<td>$\frac{1}{2} \left( \frac{2^{16} \times Tx \text{ Clock}}{\text{Modulator Clock}} \right)$</td>
<td>It is good to start with half the maximum achievable sensitivity. This value will be adjusted as needed in Step 8.</td>
</tr>
<tr>
<td>Finger threshold (FT)</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
<tr>
<td>Noise Threshold</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
<tr>
<td>Negative Noise Threshold</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
<tr>
<td>Low baseline reset</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
</tbody>
</table>
CapSense Performance Tuning

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hysteresis</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
<tr>
<td>ON Debounce</td>
<td>Default</td>
<td>Threshold settings will be adjusted in Step 10 of the tuning process.</td>
</tr>
</tbody>
</table>

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. To do this, measure the raw count value when a finger is present and divide it by the peak-to-peak noise of the raw counts with no touch present. Based on your end system design, test with a finger that matches the size of your normal use case. Typically finger size targets are ~8 – 9 mm.

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 number of sub-conversions and filters. Select an appropriate filter for your application based on Table 5-2.

- Number of sub-conversions – Can be increased to increase signal at a disproportionate rate to noise to improve overall SNR. Increasing number of sub-conversions increases the resolution of the counter and increases overall scan time.
- Filters – Filters help to reduce noise, without increasing the signal. Based on your noise type you can enable a filter to improve SNR. Each filter will add additional processing time as well as memory use.

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

   Note that number of sub-conversions can be updated directly in the Widget/Sensor Parameters tab, as Figure 5-35 shows, but to adjust the filter settings you will need to open up the CapSense configuration and select the appropriate filter as Figure 5-36 shows, and reprogram the device to update filter settings. For details on filters, see the CapSense Component Datasheet.

Figure 5-35. Update Number of Sub-Conversions in Tuner GUI
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 your design thresholds as follows. Ensure that you observe the difference count (that is signal output) in Graph View tab in Tuner GUI, not the raw count output for setting these thresholds. Based on your end system design, test the signal with a finger that matches the size of your normal use case. Typically finger size targets are ~8-9 mm. Consider testing with smaller sizes that should be "rejected" by the system to ensure they do not reach the finger threshold. When the signal is measured, set the thresholds according to the following recommendations.

Finger Threshold = 80 percent of signal
Noise Threshold = 40 percent of signal
Negative Noise Threshold = 40 percent of signal
Hysteresis = 10 percent of signal
Debounce = 3

These settings can be first set in the tuner GUI, as Figure 5-37 shows, and saved by clicking on **Apply to Project** in the **File** menu of Tuner; or they can be input directly in the CapSense Component customizer as Figure 5-38 shows. For more information on these settings, see Selecting CapSense Software Parameters.
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

Start

Set Initial parameters (leave all others as default):
Modulator clock frequency, 
Tx clock frequency and 
Number of conversions

Is SNR > 5?

Yes

Decrease number of 
conversions/Adjust 
Filters

No

Is Number of 
conversions maximum? 
and/or are Filters 
enabled?

No

Increase 
Number of conversions 
or Enable Filters

Yes

A hardware change may be 
required. Review your hardware 
design*

Set the following system 
thresholds based on raw count 
value with a finger present: 
Finger Threshold = 80% 
Noise Threshold = 40% 
Negative Noise Threshold = 40% 
Hysteresis = 10% 
Debounce = 3

Does the system meet 
timing requirements?

Yes

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6

Step 7

Step 8

Yes

No

Yes

No

Yes

No

Yes

No

Yes

No

Yes

No

Does any buttons have SNR > 10?

N

Y

* 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](image-url)
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 (see Table 5-2 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 Configure SCB_P4 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 Helper 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 Helper section of the document.

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

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

If your hardware has an option to enable Driven-Shield Signal and Shield Electrode, use this option in the advanced settings of the CapSense Component configuration window. A driven shield around the sensors helps reduce the parasitic capacitance. When you enable this option, you may want to enable driving the shield to unused sensors by also changing the “Inactive Sensor connection” setting to “shield” in the advanced settings. If after enabling the shield, your C_P remains greater than 45 pF, contact Cypress Technical Support to review your layout.

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

Solution 3: Review your Overlay Thickness with respect to your Overlay Material.

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

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

5.3.5.5 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 section 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:** 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:** 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.

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 (the BIST option is only available in CapSense v2.40) and update the layout according to the Slider Design section. See the CapSense Component datasheet 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 Rawcounts 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 the Trace Routing and Crosstalk Solutions section in this Design Guide and update your layout as required. If the layout cannot be modified at the current stage, you could evaluate use of firmware filters to reduce the 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. Refer Schematic Rule Checklist. If the current sink through GPIOs is firmware controlled, infrequent and the rawcount-level-shift caused by current sink is having a large difference from the touch signal, you could implement firmware techniques like resetting/re-initializing the CapSense baseline whenever the current sink is enabled through the GPIOs.

**Cause 3:** You have not followed the guidelines mentioned in the errata section of your corresponding PSoC6 device family datasheet.

Solution: Follow recommendations like drive mode strength selection, switching frequency restrictions, CapSense pin-selection guidelines etc. in the errata section of the corresponding PSoC6 device datasheet’s errata section, for example.
6 Gesture in CapSense

6.1 Gesture Support in CapSense

CapSense Component in PSoC 4 and PSoC 6 MCU supports the gesture detection feature that identifies the different gestures based on predefined touch patterns.

The **Gestures** tab in CapSense Component provides all gesture-related configuration parameters. Once gestures are enabled, all gesture parameters are systematically arranged by widgets/gesture groups as shown in Figure 6-1.

Figure 6-1. Configuring Gestures in CapSense Component

In the right pane of the window, click a gesture’s group name to expand the parameters related to a specific gesture group. Follow these steps to enable gestures and configure corresponding parameters:

- Select the widget for which gesture feature has to be enabled in the left pane (widget pane).
- Check the desired gesture groups in the middle pane (gesture group pane).
- Configure all parameters in the right pane (parameter pane).
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.

<table>
<thead>
<tr>
<th>Widget Type</th>
<th>Gesture Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click</td>
</tr>
<tr>
<td>Button</td>
<td></td>
</tr>
<tr>
<td>Linear Slider</td>
<td>✓</td>
</tr>
<tr>
<td>Radial Slider</td>
<td>✓</td>
</tr>
<tr>
<td>Matrix Buttons</td>
<td>✓</td>
</tr>
<tr>
<td>Touchpad</td>
<td>✓</td>
</tr>
<tr>
<td>Proximity</td>
<td></td>
</tr>
</tbody>
</table>

See the respective Component datasheets (which support gesture) for more details on each of the parameters used in configuring the gestures.
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 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.

```c
/* 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();
    }
```
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 4 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 4 low-power modes allow you to reduce overall power consumption while retaining essential functionality. See AN86233 - PSoC 4 Low-Power Modes and Power Reduction Techniques, for a basic knowledge of PSoC 4’s low-power modes and AN210998 - PSoC® 4 Low-Power CapSense® Design, for design a low-power CapSense application.

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

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

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

As the number of sensors in the design increases, the device has to spend more time in the Active mode to scan all sensors. This, in turn, increases the average power consumption. If a design has many sensors, you should include a separate proximity sensor loop that surrounds the sensors. When the device wakes up from the Deep-Sleep mode, only scan this proximity sensor. If the proximity sensor is active, the device must stay in the Active mode and scan other sensors. If the proximity sensor is inactive, the device can return to the Deep-Sleep mode. Figure 7-1 illustrates this process.

Figure 7-1. Low Power CapSense Design
To further reduce the power consumption, you can make the device enter the Sleep mode when the CPU activity is not required, but other high-speed peripherals such as system timers and I2C are required.

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

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

### 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 capacitive controller using a conductive trace or link. This whole arrangement is placed below a non-conductive overlay material and the user interacts on top of the overlay.

Figure 7-2 shows the most common CapSense sensor construction.

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

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

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

![Sensor Construction using Springs as Sensors](image)

### 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>
<thead>
<tr>
<th>Material</th>
<th>$\varepsilon_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>1.0</td>
</tr>
<tr>
<td>Formica</td>
<td>4.6 – 4.9</td>
</tr>
<tr>
<td>Glass (Standard)</td>
<td>7.6 – 8.0</td>
</tr>
<tr>
<td>Glass (Ceramic)</td>
<td>6.0</td>
</tr>
<tr>
<td>PET Film (Mylar®)</td>
<td>3.2</td>
</tr>
<tr>
<td>Polycarbonate (Lexan®)</td>
<td>2.9 – 3.0</td>
</tr>
<tr>
<td>Acrylic (Plexiglas®)</td>
<td>2.8</td>
</tr>
<tr>
<td>ABS</td>
<td>2.4 – 4.1</td>
</tr>
<tr>
<td>Wood Table and Desktop</td>
<td>1.2 – 2.5</td>
</tr>
<tr>
<td>Gypsum (Drywall)</td>
<td>2.5 – 6.0</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Widget</th>
<th>Maximum Thickness (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>5</td>
</tr>
<tr>
<td>Slider</td>
<td>5^4</td>
</tr>
<tr>
<td>Touchpad</td>
<td>0.5</td>
</tr>
</tbody>
</table>

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.

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

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 Parasitic Capacitance, $C_P$

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

- Sensor pad size increases
- Trace length and width increases
- Gap between the sensor pad and the ground hatch decreases

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

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

See Driven-Shield Signal and Shield Electrode for details.

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

---

^4 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.
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. You should use flex circuits for curved surfaces. All PCB guidelines in this document also apply to flex. You should use flex circuits with thickness 0.01 inches (0.25 mm) or higher for CapSense. The high breakdown voltage of the Kapton® material (290 kV/mm) used in flex circuits provides built in ESD protection for the CapSense sensors.

7.4.2.1 Self-Capacitance Button Design
You should use circular sensor pads for CapSense buttons. Rectangular shapes with rounded corners are also acceptable. However, you should avoid sharp corners (less than 90º) since they concentrate electric fields. Figure 7-5 shows recommended button shapes.

![Figure 7-5. Recommended Button Shapes](image)

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

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

Select the spacing between two adjacent buttons such that when touching a button, the finger is not near the gap between the other button and the ground hatch, to prevent false touch detection on the adjacent buttons, as Figure 7-6 shows.

![Figure 7-6. Spacing between Buttons](image)
7.4.2.2 Mutual-Capacitance Button

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

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

\[ A = \frac{0.25pF \times t}{\varepsilon_0 \times \varepsilon_r} \]

Where \( \varepsilon_0 \) is the permittivity of free space (8.85 x 10^{-15} F/mm) and \( \varepsilon_r \) is the dielectric constant (also known as relative permittivity) of the overlay.

The minimum electrode size for patterns shown in Figure 7-8 and Figure 7-9 should be twice the overlay thickness to ensure good signal.

For thicker overlay (> 1.5 mm), the button size will be large and in such cases, it is recommended to increase the number of Tx and Rx segments as shown in Figure 7-9. This will increase the coupling between the electrodes and increase the signal when a finger touches the sensor.
The hatch ground around the sensor reduces the noise in the sensor raw counts and hence improves SNR. The hatch fill percentage can be same as the one listed in Table 7-9 for self-capacitance buttons.

Prongs or fish-bone are also standard shapes for mutual-capacitance buttons. The Tx forms a box or ring around the outside of the key to help shielding Rx from noise. There are interlaced Tx and Rx prongs inside the border to form the electric field. Figure 7-10 shows an example of a three-prong fish-bone sensor structure. The air-gap between the Tx electrode and Rx electrode should be half of the overlay thickness. 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 Slider Design

Figure 7-11 shows the recommended slider pattern for a linear slider and Table 7-3 shows the recommended values for each of the linear slider dimensions. A detailed explanation on the recommended layout guidelines is provided in the following sections.

![Typical Linear Slider Pattern](image)

Table 7-3. Linear Slider Dimensions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Acrylic Overlay Thickness</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width of the segment (W)</td>
<td></td>
<td>1 mm</td>
<td>2 mm</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 mm</td>
<td>4 mm</td>
<td>8 mm^5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 mm</td>
<td>6 mm</td>
<td>-</td>
</tr>
<tr>
<td>Height of the segment (H)</td>
<td></td>
<td></td>
<td></td>
<td>7 mm^6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12 mm</td>
</tr>
<tr>
<td>Air gap between segments (A)</td>
<td></td>
<td></td>
<td>0.5 mm</td>
<td>2 mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5 mm</td>
</tr>
<tr>
<td>Air gap between the hatch and the slider (A_HS)</td>
<td></td>
<td>0.5 mm</td>
<td>2 mm</td>
<td>Equal to overlay thickness</td>
</tr>
</tbody>
</table>

7.4.3.1 Slider-Segment Shape, Width, and Air Gap

A linear response of the reported finger position (that is, the Centroid position) versus the actual finger position on a slider requires that the slider design is such that whenever a finger is placed anywhere between the middle of the segment SLD0 and middle of segment SLDn-1, other than the exact middle of slider segments, exactly two sensors report a valid signal^7. 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 7-11 shows. This shape helps in achieving a centroid response close to the ideal response, as Figure 7-12 and Figure 7-13 show. For the same reason, the slider-segment width and air gap (dimensions “W” and “A” respectively, as marked in Figure 7-11) should follow the relationship mentioned in Equation 7-1.

---

^5 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.

^6 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.

^7 Here, a valid signal means that the difference count of the given slider segment is greater than or equal to the noise threshold value.
Typically, an average human finger diameter is approximately 9 mm. Based on this average finger diameter and Equation 7-1, the recommended slider-segment-width and air-gap is 8 mm and 0.5 mm respectively.

If the \( W + 2A \) is lesser than \( \text{finger diameter} \), 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 7-14 shows. Thus, calculated centroid position in Equation 7-2 will be non-linear as Figure 7-15 shows.
Figure 7-14. 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

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

Where,

Resolution – API resolution set in the CapSense Component Customizer

n – Number of sensor elements in the CapSense Component Customizer

maximum – Index of element which gives maximum signal

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

Figure 7-15. 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 7-15 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 hence, none of the slider-segments will have a 5:1 SNR, resulting in a reported centroid value of 0xFF, as Figure 7-16 shows.

Figure 7-16. Incorrect Centroid Reported when Slider-Segment-Width Is Too Low

![Graph showing signal vs. finger position](image)

The minimum value of slider-segment width for certain overlay thickness values for an acrylic overlay are provided in Table 7-3. For thickness values of acrylic overlays, which are not specified in Table 7-3, Figure 7-17 may be used to estimate the minimum slider-segment width.

Figure 7-17. Minimum Slider-Segment Width w.r.t. Overlay Thickness for an Acrylic Overlay

![Graph showing minimum slider-segment width vs. acrylic overlay thickness](image)

---

8 The CapSense Component in PSoC Creator reports a centroid of 0xFF when there is no finger detected on the slider, or when none of the slider segments reports a difference count value greater than the Finger Threshold parameter.
If the $\text{slider-segment-width} + 2 \times \text{air-gap}$ is higher than the $\text{finger diameter}$ 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 will remain constant as Figure 7-18 shows. This is because, as Figure 7-19 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 7-18. Flat Spots (Nonresponsive Centroid) when Slider-Segment Width Is Higher than Recommended

![Flat Spots](image)

Figure 7-19. Signal on Slider Segments when Slider-Segment Width Is Higher than Recommended

![Signal on Slider Segments](image)

Note that if the value of $\text{slider-segment-width} + 2 \times \text{air-gap}$ is higher than the $\text{finger diameter}$, 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 7-12); 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.

7.4.3.2 Dummy Segments at the Ends of a Slider

In a CapSense design, when one segment is scanned, adjacent segments are connected to either ground or to the driven-shield signal based on the option specified in the “Inactive sensor connection” parameter in the CapSense CSD Component. For a linear centroid response, the slider requires all the segments to have the same sensitivity, that is, the increase in the raw count (signal) when a finger is placed on the slider segment should be the same for all segments. To maintain a uniform signal level from all slider segments, it is recommended that you physically connect the two segments at both ends of a slider to either ground or driven shield signal. The connection to ground or to the driven-shield signal depends on the value specified in the “Inactive sensor connection” parameter. Therefore, if your application requires an ‘n’ segment slider, it is recommended that you create $n + 2$ physical segments, as Figure 7-11 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 7-20 shows. Also, if the total available space is still constrained, the width of these segments may be kept lesser than the width of segments SLD0 through SLDn-1, or these dummy segments may even be removed.

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

Figure 7-20. Linear Slider Pattern when First and Last Segments are Connected to Top Hatch Fill

7.4.3.3 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 7-11 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 7-3.

c. The slider-segment width and the air gap between slider segments should be as recommended in Table 7-3. The recommended slider-segment width and air-gap for an average finger diameter of 9 mm is 8 mm and 0.5 mm respectively.

d. For a given slider length L, calculate the number of segments required by using the following formula:

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

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 7-18 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 slide length as Figure 7-11 shows. PCB length can be related to the slider length as follows:

\[
PCB\ length = Slider\ Length + 3 \times slider\ segment\ width + 2 \times air\ gap
\]

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:

\[ \text{PCB length} = \text{Slider Length} + \text{Slider Segment Width} \]

### 7.4.3.4 Routing Slider Segment Trace

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

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

Implement the following layout design rules to meet this condition:
- Design the shape of all segments to be as uniform as possible.
- Ensure that the length and the width of the traces connecting the segments to the device are same for all the segments.
- Maintain the same air gap between the sensors or traces to ground plane or hatch fill.

### 7.4.3.5 Slider Design with LEDs

In some applications, it may be required to display the finger position by driving LEDs. You can either place the LEDs just above the slider segments or drill a hole in the middle of a slider segment for LED backlighting, as Figure 7-21 shows. When a hole is drilled for placing an LED, the effective area of the slider segment reduces. To achieve an \( \text{SNR} > 5:1 \), you need to have a slider segment with a width larger than the LED hole size. See Table 7-3 for the minimum slider width required to achieve an \( \text{SNR} > 5:1 \) for a given overlay thickness. Follow the guidelines provided in the Crosstalk Solutions section to route the LED traces.

![Figure 7-21. Slider Design with LED Backlighting](image)

### 7.4.4 Sensor and Device Placement

Follow these guidelines while placing the sensor and the device in your PCB design:
- Minimize the trace length from the device pins to the sensor pad.
- Mount series resistors within 10 mm of the device pins to reduce RF interference and provide ESD protection. See Series Resistors on CapSense Pins for details.
- Mount the device and the other components on the bottom layer of the PCB.
- Isolate switching signals, such as PWM, I2C communication lines, and LEDs, from the sensor and sensor traces. You should place them at least 4 mm apart and fill a hatched ground between the CapSense traces and the switching signals to avoid crosstalk.
- DC loads such as LEDs and I2C pins should be physically separate from the CapSense pins by a full port wherever possible. For example, if there are LED pins in port P1, it is recommended to avoid having a CapSense pin on the same port. Also, it is recommended to limit the total source or sink current through GPIOs to less than 40 mA while the
CapSense block is scanning the sensor. Sinking a current greater than 40 mA during the CapSense sensor scanning may result in excessive noise in the sensor raw count.

- Avoid connectors between the sensor and the device pins because connectors increase CP and noise pickup.

7.4.5 Trace Length and Width

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

7.4.6 Trace Routing

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

Do not run capacitive sensing traces closer than 0.25 mm to switching signals or communication lines. Increasing the distance between the sensing traces and other signals increases the noise immunity. If it is necessary to cross communication lines with sensor pins, make sure that the intersection is at right angles, as Figure 7-22 shows.

Figure 7-22. Routing of Sensor and Communication Lines

If, due to spacing constraints, sensor traces run in parallel with high-speed traces such as I²C 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 7-23. 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 CP of the CapSense sensor.
7.4.7 Crosstalk Solutions

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

You can also reduce crosstalk by removing the rapid transitions in the LED drive voltage, by using a filter as Figure 7-25 shows. Design the filter based on the required LED response speed.
7.4.8 Vias
Use the minimum number of vias possible to route CapSense signals, to minimize parasitic capacitance. Place the vias on the edge of the sensor pad to reduce trace length, as Figure 7-26 shows.

![Figure 7-26. Via Placement on the Sensor Pad](image)

Via at the Center of the Sensor (Long Trace)
Via Near the Edge of the Sensor (Short Trace)

7.4.9 Ground Plane
When designing the ground plane, follow these guidelines:
- Ground surrounding the sensors should be in a hatch pattern. If you are using ground 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 7-27 explains the star connection. The central point for different families is mentioned in Table 7-4.
Table 7-4. Central Point for Star Connection

<table>
<thead>
<tr>
<th>Family</th>
<th>Central Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSoC 4000</td>
<td>VSS pin</td>
</tr>
<tr>
<td>PSoC 4100/4100M</td>
<td>VSS pin</td>
</tr>
<tr>
<td>PSoC 4200/4200M/4200L/PSoC 4-S/PSoC 4100PS</td>
<td>VSS pin</td>
</tr>
<tr>
<td>PSoC 4100-BL</td>
<td>E-pad</td>
</tr>
<tr>
<td>PSoC 4200-BL</td>
<td>E-pad</td>
</tr>
</tbody>
</table>

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 \( C_{\text{MOD}} \), \( C_{\text{INTx}} \), and \( C_{\text{SH_TANK}} \) capacitor pads close to the chip, and keep their ground planes thick enough.

7.4.9.1 Using Packages Without E-pad

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

High-level layout diagrams of the top and bottom layers of a board when using a chip without the E-pad are shown in Figure 7-28 and Figure 7-29.

![Figure 7-28. PCB Top Layer Layout Using a Chip Without E-pad](image-url)
7.4.9.2 Using Packages with E-pad
If you are using packages with E-pad, the following guidelines must be followed:
- The E-pad must be the central point and the sole return path to the supply ground.
- The E-pad must have vias underneath to connect to the next layers for additional grounding. Usually unfilled vias are used in a design for cost purposes, but silver-epoxy filled vias are recommended for the best performance as they result in the lowest inductance in the ground path.

7.4.9.3 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 7.4.9.2 apply.
- The VSSA pin should be connected to the E-pad below the chip itself.
- The vias underneath the E-pad are recommended to be 5 x 5 vias of 10-mil size.
High-level layout diagrams of the top and bottom layers of a board when using PSoC 4 BLE chips are shown in Figure 7-30 and Figure 7-31.
Figure 7-30. PCB Top Layer Layout with PSoC 4 BLE (with E-pad)

Figure 7-31. PCB Bottom Layer Layout with PSoC 4 BLE (with E-pad)
7.4.10 Power Supply Layout Recommendations
CapSense is a high-sensitivity analog system. Therefore, a poor PCB layout introduces noise in high-sensitivity sensor configurations such as proximity sensors and buttons with thick overlays (>1 mm). To achieve low noise in a high-sensitivity CapSense design, the PCB layout should have decoupling capacitors on the power lines, as listed in Table 7-5.

Table 7-5. Decoupling Capacitors on Power Lines

<table>
<thead>
<tr>
<th>Power Line</th>
<th>Decoupling Capacitors</th>
<th>Corresponding Ground Terminal</th>
<th>Applicable Device Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>0.1 µF and 1 µF</td>
<td>VSS</td>
<td>PSoC 4000</td>
</tr>
<tr>
<td>VDDIO</td>
<td>0.1 µF and 1 µF</td>
<td>VSS</td>
<td>PSoC 4000, PSoC 6 MCU</td>
</tr>
<tr>
<td>VDDD</td>
<td>0.1 µF and 1 µF</td>
<td>VSS</td>
<td>PSoC 4100, PSoC 4200, PSoC 6 MCU</td>
</tr>
<tr>
<td></td>
<td>0.1 µF and 1 µF</td>
<td>VSSD</td>
<td>PSoC 4100-BL, PSoC 4200-BL, PSoC 4200 L, PSoC 4 C S-Series, PSoC 4100S Plus</td>
</tr>
<tr>
<td>VDDA⁹</td>
<td>0.1 µF and 1 µF (Battery powered supply)</td>
<td>VSSA</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>0.1µF and 10 µF (Mains Powered supply)</td>
<td>VSSA</td>
<td>PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS</td>
</tr>
<tr>
<td>VDDR</td>
<td>0.1 µF and 1 µF</td>
<td>VSSD</td>
<td>PSoC 4100-BL, PSoC 4200-BL, PSoC 6 MCU with BLE Connectivity</td>
</tr>
<tr>
<td>VCCD</td>
<td>See device datasheet</td>
<td>VSS (PSoC 4000) or VSSD</td>
<td>All device families</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(all others)</td>
<td></td>
</tr>
</tbody>
</table>

The decoupling capacitors and \( C_{\text{GDO}} \) 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 Datasheet.

7.4.11 Layout Guidelines for Liquid Tolerance
As explained in the Liquid Tolerance section, by implementing a shield electrode and a guard sensor, a liquid-tolerant CapSense system can be implemented. This section shows how to implement a shield electrode and a guard sensor.

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

Also, having a large shield electrode may increase radiated emissions. If the board area is very large, the area outside the 1-cm shield electrode should be left empty, as Figure 7-32 shows. For improved liquid tolerance, there should not be any hatch fill or a trace connected to ground in the top and bottom layers of the PCB.

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

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

---

⁹ The VDDA 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.
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.

**7.4.11.1 Guard Sensor**

As explained in the **Guard Sensor** section, the guard sensor is a copper trace that surrounds all sensors, as Figure 7-33 shows.
The guard sensor should be triggered only when there is a liquid stream on the touch surface. Make sure that the shield electrode pattern surrounds the guard sensor to prevent it from turning on due to liquid droplets. The guard sensor should be placed such that it meets the following conditions:

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

Follow these guidelines to implement the guard sensor:

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

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

- When there is a liquid stream, more than one button sensor will be active at a time. If your design does not require multitouch 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.

### 7.4.11.2 Liquid Tolerance with Ground Ring

In some applications, it is required to have a ground ring (solid trace or a hatch fill) around the periphery of the board for improved ESD and EMI/EMC performance, as shown in Figure 7-34. 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).
7.4.12 Schematic Rule Checklist
You can use the checklist provided to verify your CapSense schematic.

Table 7-6. Schematic Rule Checklist

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Recommendations/Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C_MOD</td>
<td>2.2 nF. See Table 7-7 for pin selection.</td>
</tr>
<tr>
<td>2</td>
<td>C_SH_TANK</td>
<td>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 7-7 for pin selection.</td>
</tr>
<tr>
<td>3</td>
<td>C_INTX,C_INTB</td>
<td>470 pF. See Table 7-7 for pin selection.</td>
</tr>
<tr>
<td>3</td>
<td>Series resistance on input lines</td>
<td>560 Ω for Self-capacitance and 2 kΩ for Mutual-capacitance. See Series Resistors on CapSense Pins for details.</td>
</tr>
<tr>
<td>4</td>
<td>Sensor pin selection</td>
<td>If possible, avoid pins that are close to the GPIOs carrying switching/communication signals. Physically separate DC loads such as LEDs and I2C pins from the CapSense pins by a full port wherever possible. See the Tuning Debug FAQ Rawcounts show a level-shift or increased noise when GPIOs are toggled for more details.</td>
</tr>
<tr>
<td>Note: For PSoC 6 family devices, to achieve the best CapSense sensitivity and accuracy, follow recommendations stated in the Errata section of the corresponding device datasheet (PSoC 61, PSoC 62, PSoC 63).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.4.12.1 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 7-7 listed 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 7-7. Recommended Pins for External Capacitors (PSoC Creator 3.3 SP1 or Earlier)

<table>
<thead>
<tr>
<th>Device</th>
<th>C\text{MOD}</th>
<th>C_{\text{SH _TANK}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSoC 4000</td>
<td>P0[4]</td>
<td>P0[2]</td>
</tr>
<tr>
<td></td>
<td>CSD1: P5[0]</td>
<td>CSD1: P5[1]</td>
</tr>
<tr>
<td>PSoC 4 BLE</td>
<td>P4[0]</td>
<td>P4[1]</td>
</tr>
</tbody>
</table>

### Table 7-8. Recommended Pins for External Capacitors (PSoC Creator 3.3 SP2 or After)

<table>
<thead>
<tr>
<th>Device</th>
<th>C\text{MOD}</th>
<th>C_{\text{SH _TANK}}</th>
<th>C_{\text{INTA}}</th>
<th>C_{\text{INTB}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSoC 4000</td>
<td>Port0[0:7], Port1 [0:7], Port2[0]</td>
<td>Port0 [0:7], Port1 [0:7], Port2[0]</td>
<td>P0[4]</td>
<td>P0[2]</td>
</tr>
<tr>
<td>PSoC 4100</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[2]</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[3]</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>PSoC 4200</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[2]</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[3]</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7]</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7]</td>
</tr>
<tr>
<td>PSoC 4200M</td>
<td>CSD0: Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:5], Port7[0:1]</td>
<td>CSD0: Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:5], Port7[0:1]</td>
<td>CSD0: P4[2]</td>
<td>CSD0: P4[3]</td>
</tr>
<tr>
<td></td>
<td>CSD1: Not Supported</td>
<td>CSD1: Not Supported</td>
<td>CSD1: Not Supported</td>
<td>CSD1: Not Supported</td>
</tr>
<tr>
<td>PSoC 4200L</td>
<td>CSD0: Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:5], Port7[0:7]</td>
<td>CSD0: Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:5], Port7[0:7]</td>
<td>CSD0: P4[2]</td>
<td>CSD0: P4[3]</td>
</tr>
<tr>
<td></td>
<td>CSD1: Port5 [0:7], Port8 [0:7], Port9[0:7]</td>
<td>CSD1: Port5 [0:7], Port8 [0:7], Port9[0:7]</td>
<td>CSD1: P5[0]</td>
<td>CSD1: P5[1]</td>
</tr>
<tr>
<td>PSoC 4 BLE</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:1], Port5 [0:1]</td>
<td>Port0 [0:7], Port1 [0:7], Port2 [0:7], Port3 [0:7], Port4[6], Port6 [0:1]</td>
<td>P4[0]</td>
<td>P4[1]</td>
</tr>
</tbody>
</table>

### 7.4.13 Layout Rule Checklist

You can use the checklist provided in Table 7-9 to help verify your layout design.
Table 7-9. Layout Rule Checklist

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Recommendations / Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Button</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shape</td>
<td>N/A</td>
<td>N/A</td>
<td>Circle or rectangular with curved edges</td>
</tr>
<tr>
<td></td>
<td>Size</td>
<td>5 mm</td>
<td>15 mm</td>
<td>10 mm</td>
</tr>
<tr>
<td></td>
<td>Clearance to ground hatch</td>
<td>0.5 mm</td>
<td>2 mm</td>
<td>Should be equal to overlay thickness</td>
</tr>
<tr>
<td>2</td>
<td>Slider</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Width of segment</td>
<td>1.5 mm</td>
<td>8 mm</td>
<td>8 mm</td>
</tr>
<tr>
<td></td>
<td>Clearance between segments</td>
<td>0.5 mm</td>
<td>2 mm</td>
<td>0.5 mm</td>
</tr>
<tr>
<td></td>
<td>Height of segment</td>
<td>7 mm</td>
<td>15 mm</td>
<td>12 mm</td>
</tr>
<tr>
<td>3</td>
<td>Overlay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>N/A</td>
<td>N/A</td>
<td>Material with high relative permittivity (except conductors)</td>
</tr>
<tr>
<td></td>
<td>Remove any air gap between sensor board and overlay / front panel of the casing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thickness for buttons</td>
<td>N/A</td>
<td>5 mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thickness for sliders</td>
<td>N/A</td>
<td>5 mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thickness for touchpads</td>
<td>N/A</td>
<td>0.5 mm</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sensor Traces</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td>N/A</td>
<td>7 mil</td>
<td>Use the minimum width possible with the PCB technology that you use.</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>N/A</td>
<td>300 mm for a standard (FR4) PCB 50 mm for flex PCB</td>
<td>Keep as low as possible</td>
</tr>
<tr>
<td></td>
<td>Clearance to ground and other traces</td>
<td>0.25 mm</td>
<td>N/A</td>
<td>Use maximum clearance while keeping the trace length as low as possible</td>
</tr>
<tr>
<td></td>
<td>Routing</td>
<td>N/A</td>
<td>N/A</td>
<td>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.</td>
</tr>
<tr>
<td>5</td>
<td>Via</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of vias</td>
<td>1</td>
<td>2</td>
<td>At least one via is required to route the traces on the opposite side of the sensor layer</td>
</tr>
<tr>
<td></td>
<td>Hole size</td>
<td>N/A</td>
<td>N/A</td>
<td>10 mil</td>
</tr>
<tr>
<td>6</td>
<td>Ground</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hatch Fill Percentage</td>
<td>N/A</td>
<td>N/A</td>
<td>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)</td>
</tr>
<tr>
<td>7</td>
<td>Series resistor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Placement</td>
<td>N/A</td>
<td>N/A</td>
<td>Place the resistor within 10 mm of the PSoC pin. See Figure 7-35 for an example placement of series resistance on board.</td>
</tr>
<tr>
<td>8</td>
<td>Shield electrode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spread</td>
<td>N/A</td>
<td>1 cm</td>
<td>If you have PCB space, use 1-cm spread.</td>
</tr>
<tr>
<td>9</td>
<td>Shape</td>
<td>N/A</td>
<td>N/A</td>
<td>Rectangle with curved edges</td>
</tr>
</tbody>
</table>
### Design Considerations

#### Guard sensor (for water tolerance)
- **Category:** Thickness
- **Minimum Value:** N/A
- **Maximum Value:** N/A
- **Recommendations / Remarks:**
  - Recommended thickness of guard trace is 2 mm and distance of guard trace to shield electrode is 1 mm.

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Recommendations / Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>C_MOD</td>
<td>N/A</td>
<td>N/A</td>
<td>Place close to the PSoC pin. See Figure 7-35 for an example placement of C_MOD on PC board.</td>
</tr>
<tr>
<td>11</td>
<td>C_SH_TANK</td>
<td>N/A</td>
<td>N/A</td>
<td>Place close to the PSoC pin. See Figure 7-35 for an example placement of C_SH_TANK on board.</td>
</tr>
<tr>
<td>12</td>
<td>C_INTA</td>
<td>N/A</td>
<td>N/A</td>
<td>Place close to the PSoC pin. See Figure 7-35 for an example placement of C_INTA on the PCB.</td>
</tr>
<tr>
<td>13</td>
<td>C_INTB</td>
<td>N/A</td>
<td>N/A</td>
<td>Place close to the PSoC pin. See Figure 7-35 for an example placement of C_INTB on the PCB.</td>
</tr>
</tbody>
</table>

**Figure 7-35.** Example Placement for C_MOD, C_INTx, C_SH_TANK, and Series Resistance on Input Lines in PSoC 4200M Device

![PCB Diagram with Component Placements](image)
7.5  ESD Protection

The nonconductive overlay material used in CapSense provides inherent protection against ESD. Table 7-10 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>
<thead>
<tr>
<th>Material</th>
<th>Breakdown Voltage (V/mm)</th>
<th>Minimum Overlay Thickness for Protection Against 12 kV ESD (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>120</td>
<td>10</td>
</tr>
<tr>
<td>Wood – dry</td>
<td>3900</td>
<td>3</td>
</tr>
<tr>
<td>Glass – common</td>
<td>7900</td>
<td>1.5</td>
</tr>
<tr>
<td>Glass – Borosilicate (Pyrex®)</td>
<td>13,000</td>
<td>0.9</td>
</tr>
<tr>
<td>PMMA Plastic (Plexiglas®)</td>
<td>13,000</td>
<td>0.9</td>
</tr>
<tr>
<td>ABS</td>
<td>16,000</td>
<td>0.8</td>
</tr>
<tr>
<td>Polycarbonate (Lexan®)</td>
<td>16,000</td>
<td>0.8</td>
</tr>
<tr>
<td>Formica</td>
<td>18,000</td>
<td>0.7</td>
</tr>
<tr>
<td>FR-4</td>
<td>28,000</td>
<td>0.4</td>
</tr>
<tr>
<td>PET Film (Mylar®)</td>
<td>280,000</td>
<td>0.04</td>
</tr>
<tr>
<td>Polymide film (Kapton®)</td>
<td>290,000</td>
<td>0.04</td>
</tr>
</tbody>
</table>

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.

7.5.1 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 7-36, if L1 and L2 are greater than 10 mm, the system can withstand a 12-kV ESD.

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 7-10 for other material dielectric strengths.
7.5.2 Redirect

If your product is densely packed, preventing the discharge event may not be possible. In such cases, you can protect the PSoC from ESD by redirecting the ESD. A standard practice is to place a ground ring on the perimeter of the circuit board, as Figure 7-37 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 7-37. Ground Ring

7.5.3 ESD Protection Devices

You can use ESD protection devices on vulnerable traces. Select ESD protection devices with a low input capacitance to avoid reduction in CapSense sensitivity. Table 7-11 lists the recommended ESD protection devices.

<table>
<thead>
<tr>
<th>ESD Protection Device</th>
<th>Input Capacitance</th>
<th>Leakage Current</th>
<th>Contact Maximum ESD Limit</th>
<th>Air Discharge Maximum ESD Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>Part Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Littelfuse</td>
<td>SP723</td>
<td>5 pF</td>
<td>2 nA</td>
<td>8 kV</td>
</tr>
<tr>
<td>Vishay</td>
<td>VBUS05L1-DD1</td>
<td>0.3 pF</td>
<td>0.1 µA</td>
<td>±15 kV</td>
</tr>
<tr>
<td>NXP</td>
<td>NUP1301</td>
<td>0.75 pF</td>
<td>30 nA</td>
<td>8 kV</td>
</tr>
</tbody>
</table>
7.6 Electromagnetic Compatibility (EMC) Considerations

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

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

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

7.6.1 Radiated Interference and Emissions

While PSoC 4 and PSoC 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.

7.6.1.1 Hardware Considerations

7.6.1.1.1 Ground Plane

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

7.6.1.1.2 Series Resistors on CapSense Pins

Every CapSense controller pin has some parasitic capacitance, \( C_P \), associated with it. As Figure 7-38 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 7-38. RC Filter](image)

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.

### 7.6.1.1.3 Series Resistors on Digital Communication Lines

Communication lines, such as I²C 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 I²C 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 I²C operation as the \( V_{IL} \) level still remains within the I²C specification limit of 0.3 \( V_{DD} \) when PSoC outputs a LOW.

![Figure 7-39. Series Resistors on Communication Lines](image)

### 7.6.1.1.4 Trace Length

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

### 7.6.1.1.5 Current Loop Area

Another important layout consideration is to minimize the return path for currents. This is important as the current flows in loops. Unless there is a proper return path for high-speed signals, the return current will flow through a longer return path forming a larger loop, thus leading to increased emissions and interference.

If you isolate the CapSense ground hatch and the ground fill around the device, the sensor-switching current may take a longer return path, as Figure 7-40 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 7-41 shows.

![Figure 7-40. Improper Current Loop Layout](image)
7.6.1.1.6 RF Source Location

If your system has a circuit that generates RF noise, such as a switched-mode power supply (SMPS) or an inverter, you should place these circuits away from the CapSense interface. You should also shield such circuits to reduce the emitted RF. Figure 7-42 shows an example of separating the RF noise source from the CapSense interface.

7.6.1.2 Firmware Considerations

The following parameters affect Radiated Emissions (RE) in a CapSense system:

- Device operating voltage
- Device operation frequency
- Sensor switching frequency
- Shield signal
- Sensor scan time
- Sense Clock Source Inactive sensor termination

The following sections explain the effect of each parameter.
7.6.1.2.1 Device Operating Voltage

The emission is directly proportional to the voltage levels at which switching happens. Reducing the operating voltage helps to reduce the emissions as the amplitude of the switching signal at any output pin directly depends on the operating voltage of the device.

PSoC allows you to operate at lower operating voltages, thereby reducing the emissions. Figure 7-43 and Figure 7-44 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 7-43. Effect of VDD on Radiated Emissions (150 kHz – 30 MHz)

![Figure 7-43. Effect of VDD on Radiated Emissions (150 kHz – 30 MHz)](image)

Spike at 24 MHz

Figure 7-44. Effect of VDD on Radiated Emissions (30 MHz – 1 GHz)

![Figure 7-44. Effect of VDD on Radiated Emissions (30 MHz – 1 GHz)](image)

Note Frequency axis is in log scale.

7.6.1.2.2 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.
7.6.1.2.3 Sensor-Switching Frequency
Reducing the sensor-switching frequency (see Sense Clock) also helps to reduce radiated emissions. See Figure 7-45 and Figure 7-46. Because \( IMO = 24 \text{ MHz} \), there is a spike at 24 MHz and the other spikes are caused by different hardware and firmware operations of the device.

Figure 7-45. Effect of Sensor-Switching Frequency on Radiated Emissions (150 kHz – 30 MHz)

![Figure 7-45](image)

Figure 7-46. Effect of Sensor-Switching Frequency on Radiated Emissions (30 MHz – 1 GHz)

![Figure 7-46](image)

**Note** Frequency axis is in log scale.

7.6.1.2.3.1 Pseudo Random Sense Clock
The PSoC 4 device supports PRS-based sense clock generation. A PRS is used instead of a fixed clock source to attenuate emitted noise on the CapSense pins by reducing the amount of EMI created by a fixed-frequency source and to increase EMI immunity from other sources and their harmonics.
7.6.1.2.3.2  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 7-47. Sense Clock Sources in PSoC 4 S-Series, PSoC 4100S Plus, and PSoC 4100PS Family

7.6.1.2.4  Shield Signal

Enabling the shield signal (see Driven-Shield Signal and Shield Electrode) on the hatch pattern increases the radiated emissions. Enable the driven-shield signal only for liquid-tolerant, proximity-sensing, or high-parasitic-capacitance designs. Also, if the shield must be used, ensure that the shield electrode area is limited to a width of 1 cm from the sensors, as Figure 7-32 shows.

Figure 7-48 and Figure 7-49 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.
7.6.1.2.5  Sensor Scan Time

Reducing the sensor scan time reduces the average radiated emissions. The sensor-scan time depends on the scan resolution and modulator clock divider (See Equation 3-5). Increasing the scan resolution or modulator clock divider increases the scan time. Figure 7-50 and Figure 7-51 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>
<thead>
<tr>
<th>Parameter</th>
<th>Total Scan time for 5 Buttons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.426 ms</td>
</tr>
<tr>
<td>Modulation Clock Divider</td>
<td>2</td>
</tr>
<tr>
<td>Scan Resolution</td>
<td>10 bits</td>
</tr>
<tr>
<td>Individual Sensor Scan Time</td>
<td>0.085 ms</td>
</tr>
</tbody>
</table>
Figure 7-50. Effect of Scan Time on Radiated Emissions (150 kHz – 30 MHz)

Figure 7-51. Effect of Scan Time on Radiated Emissions (30 MHz – 1 GHz)

**Note** Frequency axis is in log scale.

7.6.1.2.6  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 7-52 and Figure 7-53. 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 7-52. Effect of Sense Clock Source on Radiated Emissions (150 kHz – 30 MHz)
Design Considerations

Figure 7-53. Effect of Sense Clock Source on Radiated Emissions (30 MHz – 1 GHz)

Note Frequency axis is in log scale.

7.6.1.2.7 Inactive Sensor Termination

Connecting inactive sensors to ground reduces the radiated emission by a greater degree than connecting them to the shield. Figure 7-54 and Figure 7-55 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 7-54. Effect of Inactive Sensor Termination on Radiated Emissions (150 kHz – 30 MHz)

Spike at 24 MHz
7.6.2 Conducted RF Noise

The noise current that enters the CapSense system through the power and communication lines is called conducted noise. You can use the following techniques to reduce the conducted RF noise.

- Use decoupling capacitors on the power supply pins to reduce the conducted noise from the power supply. See section 7.4.10 and the 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.

**Note** Frequency axis is in log scale.
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](image)

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

<table>
<thead>
<tr>
<th>Application</th>
<th>CapSense</th>
<th>Opamp</th>
<th>ADC</th>
<th>Comp</th>
<th>Comm (BLE, I2C, SPI, UART)</th>
<th>LCD drive</th>
<th>GPIOs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart rate monitor (wrist band)</td>
<td>User interface: buttons, linear sliders</td>
<td>TIA, Buffer</td>
<td>Heart Rate Measurement, Battery voltage measurement</td>
<td>LED Driving</td>
<td>BLE</td>
<td>Segment LCD</td>
<td>LED indication</td>
</tr>
<tr>
<td>LED bulb</td>
<td>User interface: buttons, radial sliders</td>
<td>Amplifier</td>
<td>LED current measurement</td>
<td>Short Circuit Protection, LED color control (PrISM*)</td>
<td>BLE</td>
<td>LED indication</td>
<td></td>
</tr>
<tr>
<td>Washing machine</td>
<td>User interface: buttons, radial sliders</td>
<td>Temperature sensor</td>
<td>Water level monitor</td>
<td>Buzzer, FOC** motor control</td>
<td>I2C LCD display, UART network interface</td>
<td>Segment LCD</td>
<td>LED indication</td>
</tr>
<tr>
<td>Water heater</td>
<td>User interface: buttons, linear sliders</td>
<td>Temperature sensor, water flux sensor</td>
<td>Water level monitor</td>
<td>Buzzer</td>
<td>I2C LCD display, UART Network Interface</td>
<td>Segment LCD</td>
<td>LED indication</td>
</tr>
<tr>
<td>IR remote controllers</td>
<td>User interface: buttons, linear and radial sliders, touchpads</td>
<td>Manchester encoder</td>
<td></td>
<td></td>
<td></td>
<td>LED indication</td>
<td></td>
</tr>
<tr>
<td>Induction cookers</td>
<td>User interface: buttons, linear sliders</td>
<td>Temperature sensor</td>
<td></td>
<td></td>
<td>Segment LCD</td>
<td>LED indication</td>
<td></td>
</tr>
<tr>
<td>Motor control systems</td>
<td>User interface: buttons, linear sliders</td>
<td></td>
<td>BLDC*** and FOC motor control</td>
<td></td>
<td>LED indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaming / simulation controllers</td>
<td>User interface: buttons, touchpads</td>
<td>Reading analog joysticks</td>
<td></td>
<td>I2C/SPI/UART communication interface</td>
<td>Segment LCD</td>
<td>LED indication</td>
<td></td>
</tr>
<tr>
<td>Thermal printers</td>
<td>User interface: buttons</td>
<td>Overheat protection, paper sensor</td>
<td>Stepper motor control</td>
<td>SPI communication interface</td>
<td>LED indication</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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.

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.
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 Datasheet

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


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.

9.4 Development Kits

Table 4-1 lists Cypress development kits that support PSoC 4 and PSoC 6 CapSense.

9.5 PSoC Creator

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

9.6 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
- PSocCreator to ModusToolbox
- ModusToolbox Command Line
9.7 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

9.8 Design Support

Cypress has a variety of design support channels to ensure the success of your CapSense solutions.

- **Knowledge Base Articles** – Browse technical articles by product family or perform a search on CapSense topics.
- **White Papers** – Learn about advanced capacitive-touch interface topics.
- **Cypress Developer Community** – Connect with the Cypress technical community and exchange information.
- **Video Library** – Quickly get up to speed with tutorial videos.
- **Quality & Reliability** – Cypress is committed to complete customer satisfaction. At our Quality website, you can find reliability and product qualification reports.
- **Technical Support** – Submit your design for review by creating a Cypress Support Case. You need to register and login at Cypress website to be able to contact technical support. Cypress recommends PDF prints for the schematic and Gerber files with layer information for the layout.
Glossary

AMUXBUS
Analog multiplexer bus available inside PSoC that helps to connect I/O pins with multiple internal analog signals.

SmartSense™ Auto-Tuning
A CapSense algorithm that automatically sets sensing parameters for optimal performance after the design phase and continuously compensates for system, manufacturing, and environmental changes.

Baseline
A value resulting from a firmware algorithm that estimates a trend in the Raw Count when there is no human finger present on the sensor. The Baseline is less sensitive to sudden changes in the Raw Count and provides a reference point for computing the Difference Count.

Button or Button Widget
A widget with an associated sensor that can report the active or inactive state (that is, only two states) of the sensor. For example, it can detect the touch or no-touch state of a finger on the sensor.

Difference Count
The difference between Raw Count and Baseline. If the difference is negative, or if it is below Noise Threshold, the Difference Count is always set to zero.

Capacitive Sensor
A conductor and substrate, such as a copper button on a printed circuit board (PCB), which reacts to a touch or an approaching object with a change in capacitance.

CapSense®
Cypress’s touch-sensing user interface solution. The industry’s No. 1 solution in sales by 4x over No. 2.

CapSense Mechanical Button Replacement (MBR)
Cypress’s configurable solution to upgrade mechanical buttons to capacitive buttons, requires minimal engineering effort to configure the sensor parameters and does not require firmware development. These devices include the CY8CMBR3XXX and CY8CMBR2XXX families.

Centroid or Centroid Position
A number indicating the finger position on a slider within the range given by the Slider Resolution. This number is calculated by the CapSense centroid calculation algorithm.

Compensation IDAC
A programmable constant current source, which is used by CSD to compensate for excess sensor $C_P$. This IDAC is not controlled by the Sigma-Delta Modulator in the CSD block unlike the Modulation IDAC.

CSD
CapSense Sigma Delta (CSD) is a Cypress-patented method of performing self-capacitance (also called self-cap) measurements for capacitive sensing applications.

In CSD mode, the sensing system measures the self-capacitance of an electrode, and a change in the self-capacitance is detected to identify the presence or absence of a finger.
Debounce
A parameter that defines the number of consecutive scan samples for which the touch should be present for it to become valid. This parameter helps to reject spurious touch signals.

A finger touch is reported only if the Difference Count is greater than Finger Threshold + Hysteresis for a consecutive Debounce number of scan samples.

Driven-Shield
A technique used by CSD for enabling liquid tolerance in which the Shield Electrode is driven by a signal that is equal to the sensor switching signal in phase and amplitude.

Electrode
A conductive material such as a pad or a layer on PCB, ITO, or FPCB. The electrode is connected to a port pin on a CapSense device and is used as a CapSense sensor or to drive specific signals associated with CapSense functionality.

Finger Threshold
A parameter used with Hysteresis to determine the state of the sensor. Sensor state is reported ON if the Difference Count is higher than Finger Threshold + Hysteresis, and it is reported OFF if the Difference Count is below Finger Threshold – Hysteresis.

Ganged Sensors
The method of connecting multiple sensors together and scanning them as a single sensor. Used for increasing the sensor area for proximity sensing and to reduce power consumption.

To reduce power when the system is in low-power mode, all the sensors can be ganged together and scanned as a single sensor taking less time instead of scanning all the sensors individually. When you touch any of the sensors, the system can transition into active mode where it scans all the sensors individually to detect which sensor is activated.

PSoC supports sensor-ganging in firmware, that is, multiple sensors can be connected simultaneously to AMUXBUS for scanning.

Gesture
Gesture is an action, such as swiping and pinch-zoom, performed by the user. CapSense has a gesture detection feature that identifies the different gestures based on predefined touch patterns. In the CapSense Component, the Gesture feature is supported only by the Touchpad Widget.

Guard Sensor
Copper trace that surrounds all the sensors on the PCB, similar to a button sensor and is used to detect a liquid stream. When the Guard Sensor is triggered, firmware can disable scanning of all other sensors to prevent false touches.

Hatch Fill or Hatch Ground or Hatched Ground
While designing a PCB for capacitive sensing, a grounded copper plane should be placed surrounding the sensors for good noise immunity. But a solid ground increases the parasitic capacitance of the sensor which is not desired. Therefore, the ground should be filled in a special hatch pattern. A hatch pattern has closely-placed, crisscrossed lines looking like a mesh and the line width and the spacing between two lines determine the fill percentage. In case of liquid tolerance, this hatch fill referred as a shield electrode is driven with a shield signal instead of ground.

Hysteresis
A parameter used to prevent the sensor status output from random toggling due to system noise, used in conjunction with the Finger Threshold to determine the sensor state. See Finger Threshold.

IDAC (Current-Output Digital-to-Analog Converter)
Programmable constant current source available inside PSoC, used for CapSense and ADC operations.

Liquid Tolerance
The ability of a capacitive sensing system to work reliably in the presence of liquid droplets, streaming liquids or mist.
Linear Slider
A widget consisting of more than one sensor arranged in a specific linear fashion to detect the physical position (in single axis) of a finger.

Low Baseline Reset
A parameter that represents the maximum number of scan samples where the Raw Count is abnormally below the Negative Noise Threshold. If the Low Baseline Reset value is exceeded, the Baseline is reset to the current Raw Count.

Manual-Tuning
The manual process of setting (or tuning) the CapSense parameters.

Matrix Buttons
A widget consisting of more than two sensors arranged in a matrix fashion, used to detect the presence or absence of a human finger (a touch) on the intersections of vertically and horizontally arranged sensors.

If M is the number of sensors on the horizontal axis and N is the number of sensors on the vertical axis, the Matrix Buttons Widget can monitor a total of M x N intersections using ONLY M + N port pins.

When using the CSD sensing method (self-capacitance), this Widget can detect a valid touch on only one intersection position at a time.

Modulation Capacitor (CMOD)
An external capacitor required for the operation of a CSD block in Self-Capacitance sensing mode.

Modulator Clock
A clock source that is used to sample the modulator output from a CSD block during a sensor scan. This clock is also fed to the Raw Count counter. The scan time (excluding pre and post processing times) is given by \((2^N - 1)/\text{Modulator Clock Frequency}\), where \(N\) is the Scan Resolution.

Modulation IDAC
Modulation IDAC is a programmable constant current source, whose output is controlled (ON/OFF) by the sigma-delta modulator output in a CSD block to maintain the AMUXBUS voltage at \(V_{\text{REF}}\). 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

**Document Title:** AN85951 – PSoC® 4 and PSoC 6 MCU CapSense® Design Guide  
**Document Number:** 001-85951

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN#</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>3973432</td>
<td>04/19/2013</td>
<td>NIDH</td>
<td>New Design Guide</td>
</tr>
<tr>
<td>*A</td>
<td>4059171</td>
<td>07/29/2013</td>
<td>NIDH</td>
<td>Added dual IDAC support. Updated some schematics in chapter 6. Other minor changes to chapters 3, 5, and 6.</td>
</tr>
<tr>
<td>*B</td>
<td>4189700</td>
<td>11/13/2013</td>
<td>NIDH</td>
<td>Added support of CY8C4000 devices. Minor fixes throughout the document.</td>
</tr>
<tr>
<td>*C</td>
<td>4289925</td>
<td>02/24/2014</td>
<td>NIDH</td>
<td>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.</td>
</tr>
<tr>
<td>*D</td>
<td>4293476</td>
<td>02/27/2014</td>
<td>NIDH</td>
<td>Updated Table 1-1 per PSoC 4000 datasheet.</td>
</tr>
<tr>
<td>*E</td>
<td>4314223</td>
<td>03/20/2014</td>
<td>NIDH</td>
<td>Added firmware design considerations to Chapter 6. Added power supply layout and schematic considerations to Chapter 6. Updated the IMO range for PSoC 4000</td>
</tr>
<tr>
<td>*F</td>
<td>4339713</td>
<td>04/15/2014</td>
<td>NIDH</td>
<td>Updated to support PSoC 4000 and PSoC Creator 3.0 SP1.</td>
</tr>
</tbody>
</table>
| *G      | 4494249 | 08/29/2014 | DCHE             | Added Reference to Getting Started with CapSense in Section 0 Proximity (Three-Dimensional)  
Renamed Section 2.5 to Liquid Tolerance and re-wrote this section.  
Updated the recommendations for Shield drive i.e. C_sh_tank precharge and C_mod 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 7.3.2 Overlay Thickness  
Re-wrote Section Slider Design  
Added recommendations on DC loads in Section 6.3.5  
Renamed and rewrote section 7.4.11 to Layout Guidelines for Liquid Tolerance  
Added Section 7.4.12.1 External Capacitors Pin Selection  
Updated slider related recommendations in Table 7-9. Layout Rule Checklist  
Updated Section 6.5 Electromagnetic Compatibility (EMC) Considerations, added extensive data on hardware and firmware considerations. |
| *H      | 4602375 | 12/19/2014 | UDYG             | Added information for the PSoC 4 BLE family of devices.  
Added information for the PReC BLE family of devices.  
Updated ground and power layout guidelines in Section 7.4.9 and Section 7.4.10. |
| *I      | 4624027 | 01/21/2015 | NIDH             | Added information for PSoC 4200-M family of devices  
Added footnote in section Slider Design |
<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN#</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added GPIO source/sink current limit in Table 7-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Changed document title to “AN85951 – PSoC® 4 CapSense® Design Guide” – AN85951</td>
</tr>
<tr>
<td>*J</td>
<td>4771699</td>
<td>06/02/2015</td>
<td>DCHE</td>
<td>Updated Design Considerations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated ESD Protection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Preventing ESD Discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Figure 7-36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Redirect:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Replaced “Guard Ring” with “Ground Ring”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Figure 7-37</td>
</tr>
<tr>
<td>*K</td>
<td>4891423</td>
<td>08/20/2015</td>
<td>DCHE</td>
<td>Added Table 3-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removed section 3.2.1 CMOD Precharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added section CapSense in PSoC 4xxxM/4xxxL-Series</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated section Trace Routing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added reference of AN2397</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added recommendation for modulator clock divider in section Manual Tuning Trade-offs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added Figure 7-35</td>
</tr>
<tr>
<td>*L</td>
<td>4905591</td>
<td>09/16/2015</td>
<td>DIMA</td>
<td>Updated Section 3.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Figure 3-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Table 3-3, Table 4-1, Table 7-4, Table 7-5, Table 7-7</td>
</tr>
<tr>
<td>*M</td>
<td>5076590</td>
<td>01/19/2016</td>
<td>PRIA</td>
<td>Updated Introduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Moved Signal-to-Noise Ratio to Chapter 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Chapters PSoC 4 and PSoC 6 MCU CapSense and CapSense Performance Tuning for details</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added section to Chapter 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added Glossary</td>
</tr>
<tr>
<td>*N</td>
<td>5131335</td>
<td>02/23/2016</td>
<td>DCHE</td>
<td>Added information on mutual-capacitance sensing in PSoC 4 device series</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added information on CapSense 3.0 changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Mutual-Capacitance Sensing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- CapSense Architecture in PSoC 4 S-Series</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Introduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- CapSense Widgets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- CapSense Design and Development Tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- CapSense Performance Tuning</td>
</tr>
<tr>
<td>*O</td>
<td>5162301</td>
<td>03/04/2016</td>
<td>DCHE</td>
<td>Added PSoC Analog Coprocessor references</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated External Capacitors Pin Selection section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Development Kits section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated document title</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Copyright notice</td>
</tr>
<tr>
<td>*P</td>
<td>5307639</td>
<td>06/14/2016</td>
<td>DCHE</td>
<td>Updated IDAC sinking mode recommendation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated template</td>
</tr>
<tr>
<td>*Q</td>
<td>5526001</td>
<td>11/18/2016</td>
<td>DCHE</td>
<td>Updated Table 7-7</td>
</tr>
<tr>
<td>*R</td>
<td>5687926</td>
<td>04/19/2017</td>
<td>BENV</td>
<td>Updated logo and copyright</td>
</tr>
<tr>
<td>*S</td>
<td>5896262</td>
<td>09/22/2017</td>
<td>TAVA</td>
<td>Added references to PSoC 4100S Plus throughout the document</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Section 1.3 CapSense Features with PSoC 4100S Plus features</td>
</tr>
<tr>
<td>Revision</td>
<td>ECN#</td>
<td>Issue Date</td>
<td>Origin of Change</td>
<td>Description of Change</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Table 4-1, PSoC 4 and PSoC 6 CapSense Development Kits with CY8CKIT-149 PSoC 4100S Plus Prototyping Kit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Section 9.7 Application Notes with specific list of CapSense Application Notes</td>
</tr>
<tr>
<td>*T</td>
<td>6036561</td>
<td>01/18/2018</td>
<td>VKVK</td>
<td>Changed document title</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added references to PSoC 6 MCU features throughout the document</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Section 3.1 CapSense CSD Sensing Method with generalized architecture block diagram for CSD sensing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Added Section 6 Gesture in CapSense</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Table 4-1, Table 7-5, Table 7-8</td>
</tr>
<tr>
<td>*U</td>
<td>6084086</td>
<td>02/28/2018</td>
<td>DIMA</td>
<td>Added references to PSoC 4100PS throughout the document</td>
</tr>
<tr>
<td>*V</td>
<td>6375492</td>
<td>11/08/2018</td>
<td>TAVA</td>
<td>Updated the entire document with references to CY8C62x8 and CY8C62xA devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated the entire document with references to ModusToolbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Table 4-1 with the information of PSoC 6 kits.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated section Mutual-Capacitance Button with the information of additional mutual cap key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removed all references to PRoC BLE devices.</td>
</tr>
<tr>
<td>*W</td>
<td>6540965</td>
<td>04/11/2019</td>
<td>BLPD</td>
<td>Updated SmartSense and Manual Tuning with respect to the latest component</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removed details on different shield drive mode from CapSense CSD Shielding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated CapSense CSX Sensing Method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated figures in PSoC Creator, SmartSense, and Gesture in CapSense with respect to the latest component</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Removed a table in External Capacitors Pin Selection section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Table 3-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Slider Design</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Updated Mutual-Capacitance Button</td>
</tr>
</tbody>
</table>