AN64846

Getting Started with CapSense

Document Number. 001-64846 Rev. *Y

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
http://www.cypress.com

Copyrights

Contents

1. Introduction

1.1 How to Use this Guide

This guide is an ideal starting point for those who are new to capacitive touch sensing (CapSense®). You can use this guide to:

- Become familiar with the technology underlying CapSense solutions
- Understand important design considerations, such as schematic, layout, and EMI (Electro Magnetic Interference)
- Select the right device for your application
- Find a CapSense resource to help with your design

When you are ready to design your application, consult the Design Guide specific to the CapSense device family you have selected. See the Glossary for the definitions of CapSense terms.

1.2 Code Examples

To access an ever-growing list of hundreds of PSoC® code examples, visit our code examples webpage. You can also explore the PSoC 4 video library here.

1.3 Cypress CapSense Differentiation

Capacitive touch sensing has changed the face of industrial design in products such as cellphones, PCs, consumer electronics, automotive, and white goods. Cypress CapSense solutions bring elegant, reliable, and easy-to-use capacitive touch sensing functionality to your design. Our capacitive touch sensing solutions have replaced more than four billion mechanical buttons. A CapSense-based user interface design has the following advantages over a mechanical-buttons based design:

- Mechanical buttons are less reliable and wear out over time due to the physical movement. CapSense designs do not involve moving parts.
- Mechanical buttons pose problems when moisture seeps through the gaps in the assembly. CapSense-based front panels can be completely sealed under the overlay.
- Mechanical buttons require a small force to operate compared to the touch buttons and this force can increase over time due to the accumulation of dirt in the gaps.
- Mechanical buttons require multiple parts and increase the BOM cost whereas many CapSense designs consist of only a PCB and an overlay with adhesive.
- Mechanical buttons include the cost of tools required to make cutouts in the front panel. CapSense designs do not require such cutouts.
- Mechanical buttons yield poor aesthetics compared to the sleek and elegant touch buttons. CapSense designs also offer more flexibility in designing the user interface in terms of button shape and graphical representation.

Cypress' robust CapSense solutions leverage our flexible Programmable System-on-Chip (PSoC) architecture, which accelerates time-to-market, integrates critical system functions, and reduces BOM cost. Cypress offers a wide range of configurable and programmable CapSense controllers. Configurable CapSense controllers are hardware or I^2C configurable. Programmable devices provide complete flexibility to meet your exact design requirements, including reducing BOM cost by integrating further system functionality. Following are some of the unique features offered by CapSense products.

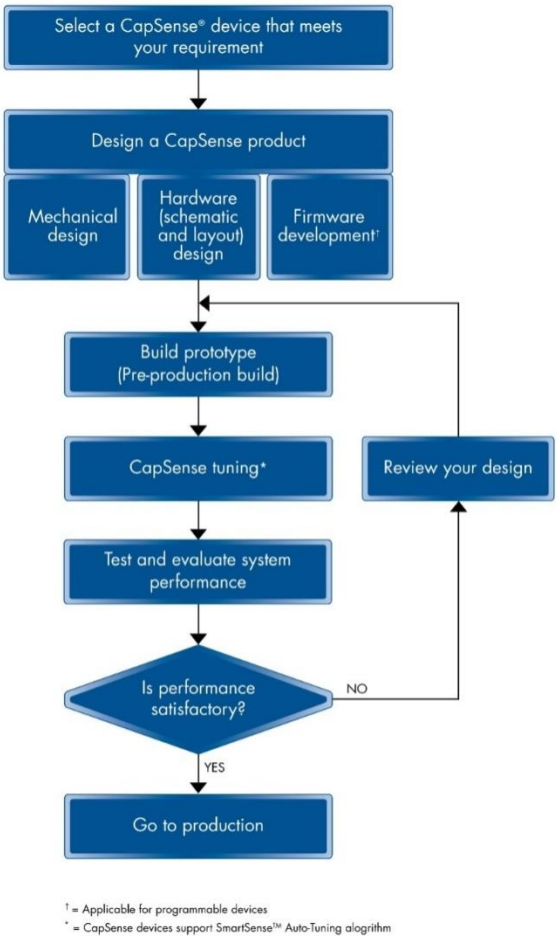
- Robust sensing technology
- High noise immunity
- High-performance sensing across a variety of overlay materials and thicknesses
- SmartSense™ Auto-Tuning technology
- Proximity sensing
- Liquid-tolerant operation
- Complete user interface solution including audio, visual, and haptic feedback
- Low power consumption
- Wide operating voltage range (1.71 V - 5.5 V)
- Small form-factor packaging
- Reduced BOM cost with integrated features like ADC, DAC, timer, counter, and PWM

1.4 CapSense Design Flow

Figure 1-1 depicts the typical flow of a CapSense product design. This flow is similar to any other electronic system design flow except that CapSense designs involve an additional step called Tuning. This is the process of finding the optimum values for various hardware and software parameters required for CapSense operation. These parameters vary depending on the board layout, sensor dimensions, overlay properties, and application requirements such as power consumption and response time. Therefore, this step is usually performed when the pre-production builds are available. Many of the CapSense devices support Cypress' Auto-tuning algorithm called SmartSense that automatically sets parameters for optimal performance after the design phase and continuously compensates for system, manufacturing, and environmental changes.

The enclosure or casing design is an integral part of a CapSense product design as the aesthetic feel and the performance of the end product depend on the casing material and its design. Since the casing acts as an overlay for the sensors, the touch-sensing performance depends on the overlay properties such as thickness and material type. Therefore, it is important to test and evaluate the performance along with the overlay material, which is similar to the one used in the end-product right from the prototype stage.

Figure 1-1. Typical CapSense Product Design Flow



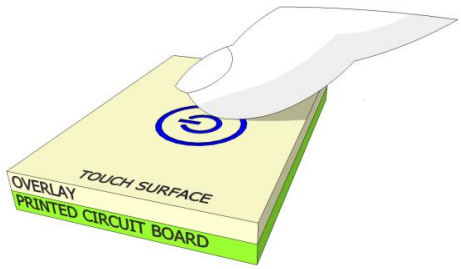
1.5 Document Conventions

Convention	Usage
`Courier New`	Displays file locations, user entered text, and source code: `C:\ ...cd\icc\`
Italics	Displays file names and reference documentation: Read about the *sourcefile.hex* file in the *PSoC Designer User Guide*.
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the **File** icon and then click **Open**.
Times New Roman	Displays an equation: 2 + 2 = 4
Text in gray boxes	Describes Cautions or unique functionality of the product.

2. CapSense Technology

Cypress' CapSense controllers use changes in capacitance to detect the presence of a finger on or near a touch surface, as shown in Figure 2-1. This touch button example illustrates a capacitive sensor replacing a mechanical button. The sensing function is achieved using a combination of hardware and firmware. See the Glossary for the definitions of CapSense terms.

Figure 2-1. Illustration of a Capacitive Sensor Application



2.1 CapSense System Overview

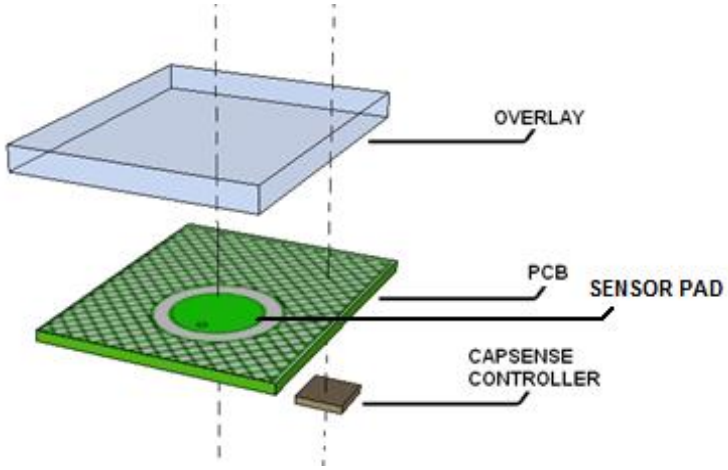
CapSense touch sensing solutions include the entire system environment in which they operate. This includes:

- Hardware components such as PCB and guard sensor
- Firmware component to process the sensor data

2.1.1 Hardware Component

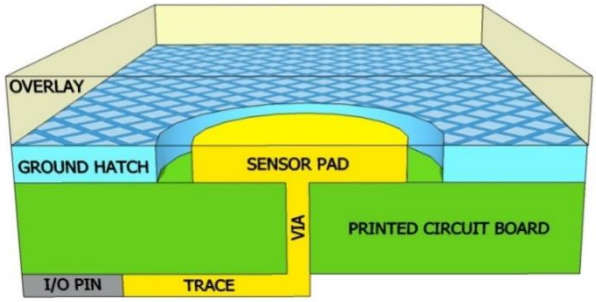
The CapSense controller resides within a larger system composed of a printed circuit board (PCB), and a touch-surface called the overlay that protects the PCB.

Figure 2-2. Exploded View of the CapSense Hardware



The capacitive sensor pads of a sensor board are formed by the PCB traces. The most common PCB format is a two-layer board with sensor pads and a hatched ground plane (see Ground Plane) on the top, and the electrical components on the bottom. The ground plane is also provided surrounding the electrical components. The electrical components include the CapSense controller and associated parts that convert the sensor capacitance into digital raw counts. Figure 2-3 shows a cross-sectional view of a two-layer board stack-up. The four-layer design is an option when the board area must be minimized. PCB layout plays a very important role in CapSense system performance. Best practices are discussed in the device-specific Design Guides.

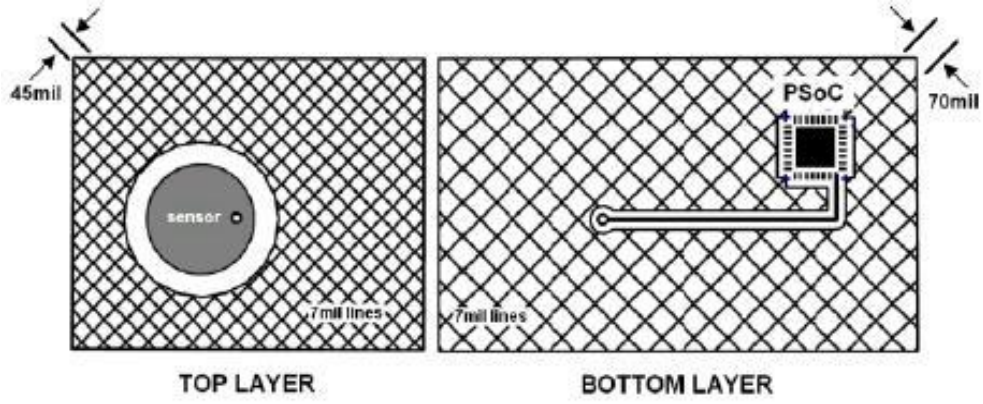
Figure 2-3. Two-Layer Stack-Up of a CapSense Board



2.1.1.1 Ground Plane

In general, a proper ground plane on the PCB reduces both RF emissions and interference. However, solid grounds near CapSense sensors, or traces connecting these sensors to the PSoC pins, increase the parasitic capacitance of the sensors. The increase in parasitic capacitance is unwanted as it reduces the sensitivity. It is thus recommended that you use hatched ground planes surrounding the sensor and on the bottom layer of the PCB, below the sensors, as Figure 2-4 shows. Typical hatching for the ground fill is 7-mil line, 45 mil spacing on the top layer, and 7-mil line, and 70-mil spacing on the bottom layer. The same hatch-fill on the top layer is driven with shield signal when liquid tolerance is required. See Liquid Tolerance to learn more.

Figure 2-4. Ground Fill on a PCB

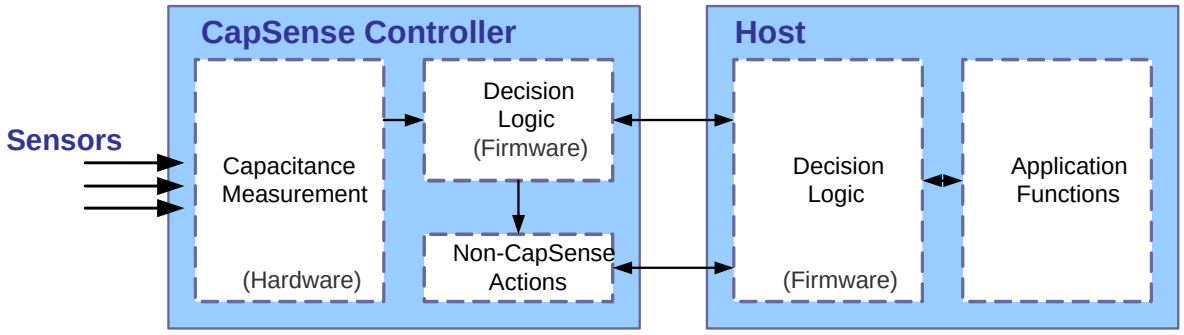


2.1.2 Firmware Component

Firmware is a vital component of the CapSense system. It processes the raw count data and makes logical decisions. The amount of firmware development required for your application depends on which CapSense controller family you select.

Devices from the CapSense Express family are fully configurable either through hardware or through I^2C and do not require any firmware development on the CapSense controller. The finger touch data is sent to a host for higher level processing; see Figure 2-5. These devices are appropriate for systems where simplicity of design and short time-to-market are the key requirements.

Figure 2-5. Example CapSense Express System Implementation



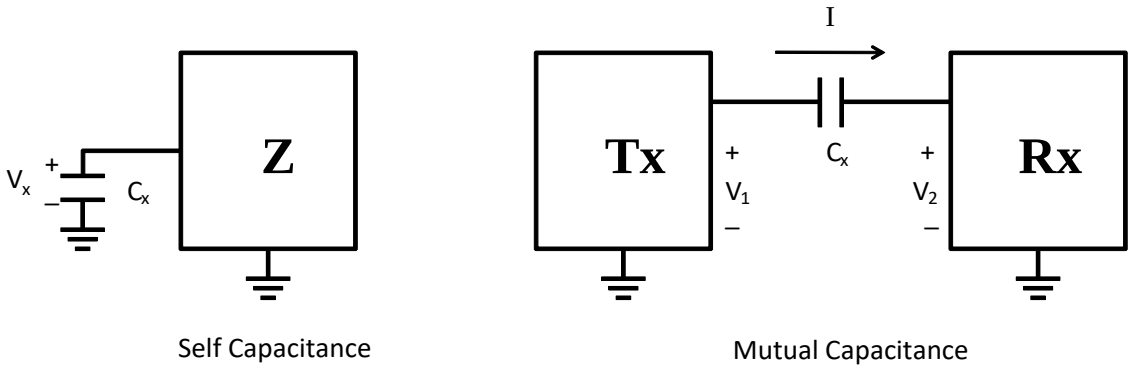
The programmable devices allow complex system-level integration. These controllers can process the raw count data as well as perform other system functions.

See CapSense Selector Guide for additional details. Cypress' PSoC Creator™, ModusToolbox, and PSoC Designer accommodate firmware development in C and assembly languages. See Software Tools for more information on this and other tools.

2.2 CapSense Fundamentals

Capacitance can be measured between two points using either self-capacitance or mutual capacitance. The left side of Figure 2-6 shows the self-capacitance method and the right side shows the mutual-capacitance method.

Figure 2-6. Self-Capacitance and Mutual-Capacitance Methods



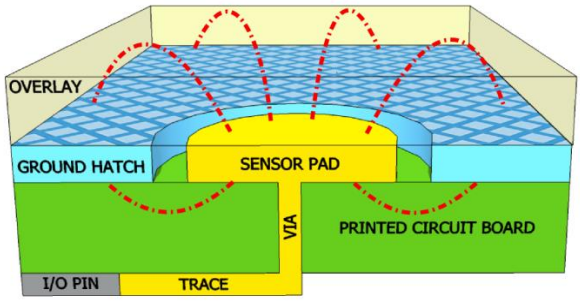
2.2.1 Self Capacitance

Self-capacitance uses a single pin and measures the capacitance between that pin and ground. A self-capacitance sensing system operates by driving current on a pin connected to a sensor and measuring the voltage. When a finger is placed on the sensor, it increases the measured capacitance. Self-capacitance sensing is best suited for single-touch sensors, such as buttons and sliders.

Cypress' CapSense solutions use self-capacitance sensing because it enables efficient use of pins for single-touch sensors and sliders.

In a CapSense self-capacitance system, the sensor capacitance measured by the controller is called C_S. When a finger is not on the sensor, C_S equals the parasitic capacitance (C_P) of the system. This parasitic capacitance is a simplification of the distributed capacitance that includes the effects of the sensor pad, the overlay, the trace between the CapSense controller pin and the sensor pad, the vias through the circuit board, and the pin capacitance of the CapSense controller. C_P is related to the electric field around the sensor pad. Although the following diagram shows field lines only around the sensor pad, the actual electric field is more complicated.

Figure 2-7. C_P and Electric Field



When a finger touches the sensor surface, it forms a simple parallel plate capacitor with the sensor pad through the overlay. The result is called finger capacitance, C_F, and is defined by Equation 1. C_F is a simplification of a distributed capacitance that includes the effects of the human body and the return path to the circuit board ground.

$$C_F = \frac{\varepsilon_0 \varepsilon_r A}{D}$$ Equation 1

Where:

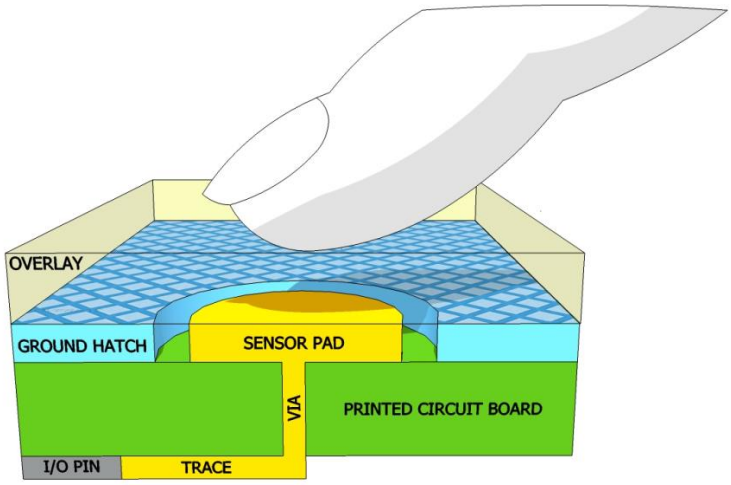
ε_0 = Free space permittivity

ε_r = Dielectric constant of overlay

A = Area of finger and sensor pad overlap

D = Overlay thickness

Figure 2-8. CapSense System Equivalent Model



With a finger on the sensor surface, C_S equals the sum of C_P and C_F.

$$C_S = C_P + C_F$$ Equation 2

2.2.2 Mutual Capacitance

Figure 2-9 shows the button sensor layout for mutual-capacitance sensing. Mutual-capacitance sensing measures the capacitance between two electrodes. One of the electrodes is called the transmit (TX) electrode and the other electrode is called the receive (RX) electrode.

In a mutual-capacitance measurement system, a digital voltage (signal switching between V_{DDD} 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^1_M as shown in Figure 2-10. Because of the reduction in 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 the touch/no touch condition.

Figure 2-9. Mutual Capacitance Sensing Working

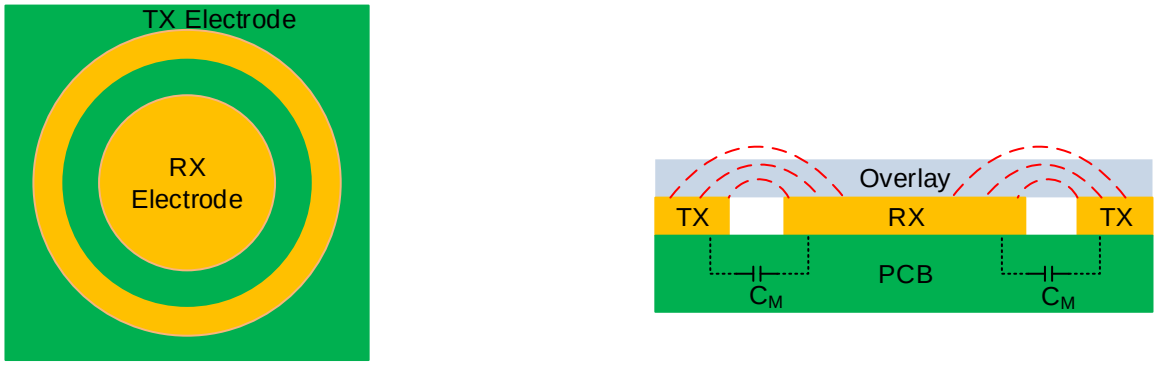
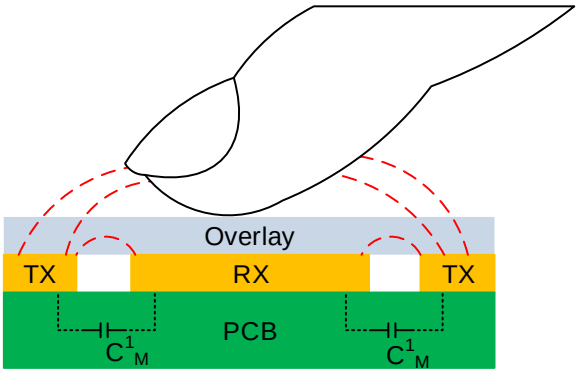


Figure 2-10. Mutual Capacitance with Finger Touch



The mutual-capacitance effect is best suited to multi-touch systems such as touchscreens and trackpads. Cypress offers mutual-capacitance-based trackpad solutions for consumer applications and TrueTouch® touchscreen solutions for automotive and home appliance applications. Contact your local Cypress sales office directly for more information. To find your local sales office, click here.

2.3 Capacitive Touch Sensing Method

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

2.3.1 CapSense Sigma Delta Modulator (CSD) Sensing Method

Figure 2-11 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 2-11. Simplified Diagram of CapSense Sigma Delta Method

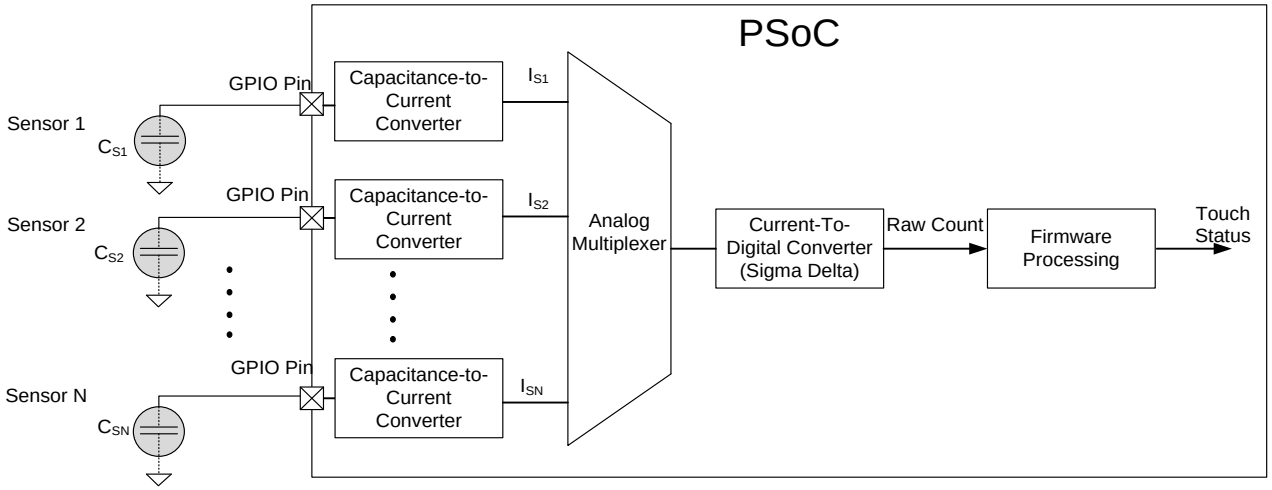
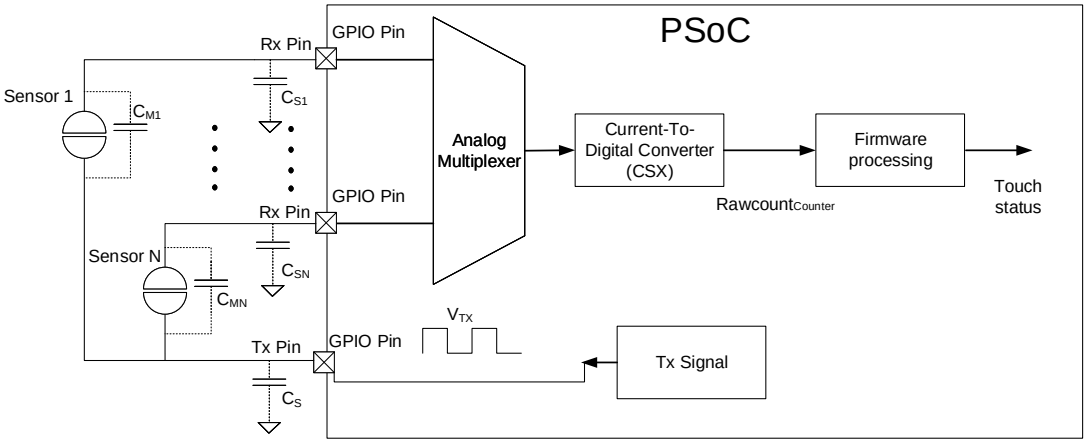


Figure 2-13 shows a plot of raw count over time. When a finger touches the sensor, the C_S increases from C_P to $C_P + C_F$, and the raw count increases. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

2.3.2 CapSense Crosspoint (CSX) Sensing Method

Figure 2-12 shows the simplified block diagram of the CSX method.

Figure 2-12. Simplified Diagram of CSX Method



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

The output count of the current to digital converter, known as $\mathbf{Rawcount_{Counter}}$, is a digital value that is proportional to the mutual-capacitance between the Rx and Tx electrodes as shown by Equation 2-4.

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

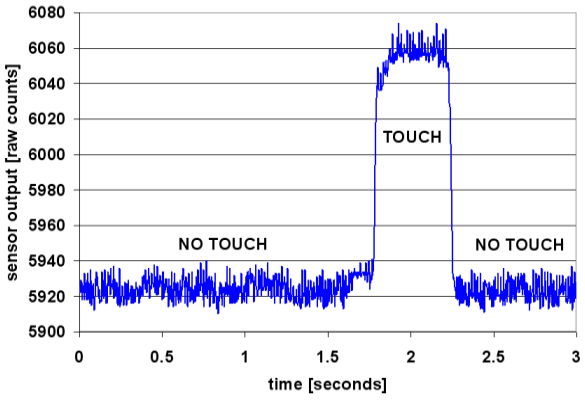
$$\mathrm{Rawcount_{Counter}} = G_{CM} C_M$$

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 2-13 shows a plot of raw count over time. When a finger touches the sensor, C_M decreases from C_M to C^1_M (see Figure 2-10) hence the counter output decreases. The firmware normalizes the raw count such that the raw counts go high when C_M decreases. This is to maintain the same visual representation of raw count between CSD and CSX methods. By comparing the change in raw count to a predetermined threshold, logic in firmware decides whether the sensor is active (finger is present).

Figure 2-13. Raw Count versus Time



For a detailed derivation of raw count as a function of Sensor Capacitance and Finger Capacitance, see the PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide.

2.4 Comparison of CapSense Architecture in Different Devices

The fourth-generation CapSense functionality in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 devices is an improved version of the previous generation. The main differences between the two generations of CapSense architecture are listed in Table 2-1.

Table 2-1. Comparison of CapSense Architecture

Feature	Third-Generation CapSense	Fourth-Generation CapSense	Advantages of Fourth-Generation over Third Generation CapSense	
			For CSD	For CSX
Sensor Parasitic Capacitance (C_P) Range	5 pF – 60 pF	5 pF – 200 pF	Supports high-C_P design applications	
Sensing modes	Self-Cap and Mutual-Cap modes[1]	Self-Cap, Mutual-Cap, and ADC modes	The CapSense hardware block can be used as a 10-bit ADC when CapSense sensor scanning is not in progress. Refer CSDADC component/middleware datasheet for detailed ADC specifications.	
V_{REF}	1.2 V	0.6 V to VDDA-0.6 V[2]	Higher Vref allows improved SNR	NA
IDAC LSB Size	1.2 µA, 2.4 µA	37.5 nA, 300 nA, 2.4 µA	Improved sensitivity, small IDAC for improved tuning	
Split IDAC Capability	Requires two IDACs	Requires one IDAC[3]	Requires fewer resources to achieve the same performance and frees up one IDAC for general-purpose use.	NA
EMI Reduction - Digital	Supports only PRS method	Supports additional spread Spectrum clock (SSC) method	More options to control the sense/Tx clock frequency spread for EMI reduction	
Modulator Clock Frequency Range	Lower	Higher	Higher modulator-clock frequency implies faster scans.	Higher modulator-clock frequency implies increased sensitivity and accuracy
Hardware State Machine[4]	No	Yes	Initiation of sensor scanning is less dependent on CPU; there are fewer critical sections during scan initialization.	
Tx Clock Frequency	Supports up-to 300 kHz	Supports much higher clock frequencies (up-to 3 MHz)	NA	Higher Tx clock results in shorter scan time.

The CapSense hardware in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 support both self-capacitance- and mutual capacitance-based capacitive sensing. The hardware also supports input voltage measurement when CapSense scanning is not in progress. See the **CapSense** chapter in the respective device datasheet for a detailed explanation of the CapSense hardware in PSoC 4 S-Series, PSoC 4100S Plus, PSoC 4100PS, and PSoC 6 devices. See the device-specific Design Guide for the basic knowledge of fourth-generation CapSense architecture.

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

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

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

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

Table 2-2. PSoC Device Family and CapSense Architecture

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

2.5 CapSense Tuning

Optimal CapSense system performance depends on the board layout, button dimensions, overlay material, and application requirements. In addition to these factors, switching frequency and threshold levels must be carefully selected for robust and reliable performance. Tuning is the process of determining the optimum values for these parameters. Tuning is required to maintain high sensitivity to touch and to compensate for process variations in the sensor board, overlay material, and environmental conditions.

Many of the CapSense devices support SmartSense, Cypress' Auto-tuning algorithm, which automatically sets parameters for optimal performance and continuously compensates for system, manufacturing and environmental changes. See SmartSense Auto-Tuning for more information.

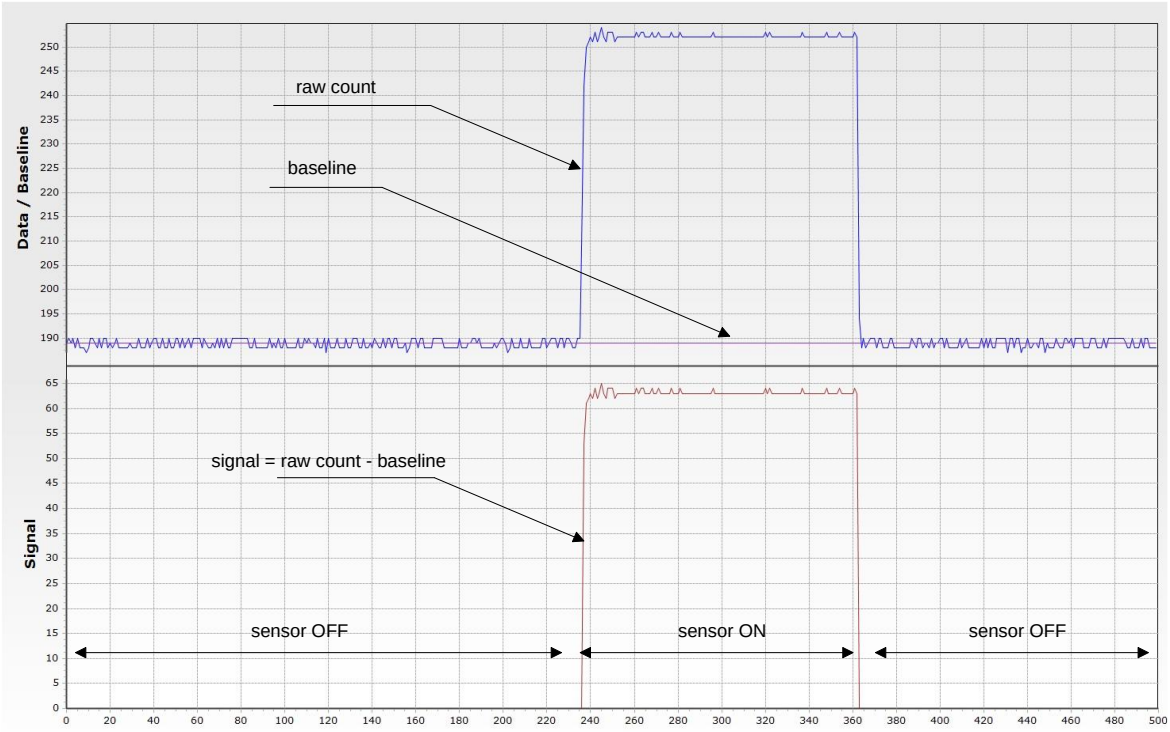
This section gives an introduction about the tuning process. For details on all the parameters involved in CapSense operation and the step-by-step tuning procedure, see the device-specific Design Guide. Cypress provides many tools to make tuning and data monitoring easy. See Data Monitoring Tools to learn more about these tools.

The following section defines some terms that will help you understand the tuning process.

2.5.1 Definitions

- **Raw Count:** As seen in Figure 2-18, sensor capacitance is converted into a count value by the CapSense algorithm. The unprocessed count value is referred to as raw count. Processing of the raw count results in ON/OFF states for the sensor.
- **Baseline:** The raw count value of a sensor may vary gradually due to changes in the environment such as temperature and humidity. Therefore, the raw count is low-pass filtered to create a new count value known as baseline that 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 the reference level for computing the signals (explained below). Figure 2-14 shows the concept of raw count, baseline and signal.

Figure 2-14. Raw Count and Baseline

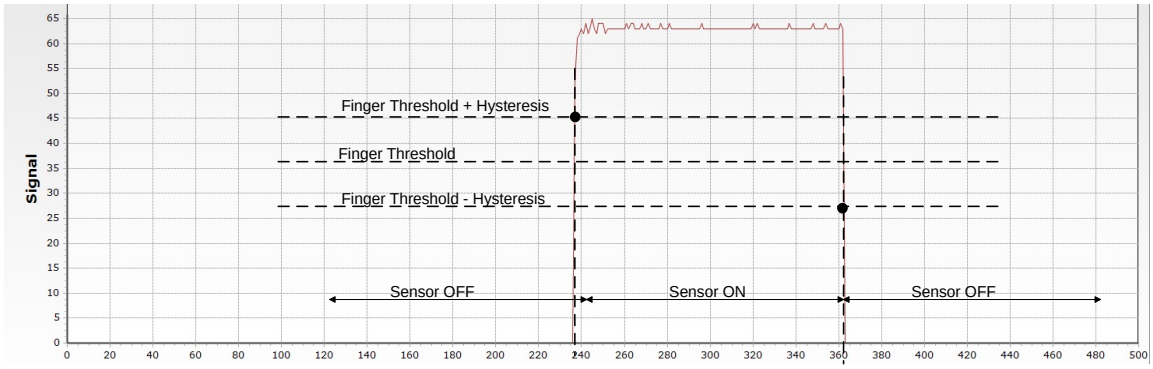


- **Difference Count or Signal:** Subtracting the baseline level from the raw count produces the difference count that is used in the ON/OFF decision process. The thresholds are offset by a constant amount from the baseline level. The thresholds have the following functions:
- **Noise Threshold:** A parameter used to differentiate signal from noise. If the raw count is above noise threshold, then the baseline is not updated and the difference count indicates the difference between raw count and baseline. If the raw count is below the noise threshold, then the baseline is updated and the difference count is zero. See Figure 2-15 for details.
- **Finger Threshold**: A parameter used with Hysteresis to determine the state of the sensor, as Equation 6 and Figure 2-15 show.

$$\text{Sensor State} = \begin{cases} \text{ON} & \text{if (Signal} \geq \text{Finger Threshold} + \text{Hysteresis)} \\ \text{OFF} & \text{if (Signal} \leq \text{Finger Threshold} - \text{Hysteresis)} \end{cases} \quad \text{Equation 5}$$

- **Hysteresis:** A parameter used with Finger Threshold to determine the state of the sensor, as Equation 6 and Figure 2-15 shows. Hysteresis provides immunity against noisy transitions of sensor state.

Figure 2-15. Hysteresis



2.5.2 SmartSense Auto-Tuning

2.5.2.1 What Is SmartSense?

Tuning the touch sensing user interface is a critical step in ensuring proper system operation and a pleasant user experience. The typical design flow involves tuning the sensor interface in the initial design phase, during system integration, and finally production fine-tuning before the production ramp. Because tuning is an iterative process, it can be time-consuming. SmartSense Auto-Tuning helps to simplify the user interface development cycle. In addition, the method is easy to use and reduces the design cycle time by eliminating the tuning process throughout the product development cycle, from prototype to mass production.

2.5.2.2 What Does SmartSense Do?

SmartSense tunes each CapSense sensor automatically at power-up and then monitors and maintains optimum sensor performance during runtime. The number of parameters to be tuned is reduced from 17 in CSD to 4 with SmartSense.

- **Power-up tuning:** SmartSense tunes the parameters of each sensor based on the individual sensor parasitic capacitance to get the desired sensitivity for the sensor.
- **Runtime tuning:** Noise in the system is measured dynamically. The thresholds are adjusted accordingly for each sensor to overcome false triggering due to dynamic variations in noise in the CapSense system.

2.5.2.3 How and Where is SmartSense Helpful?

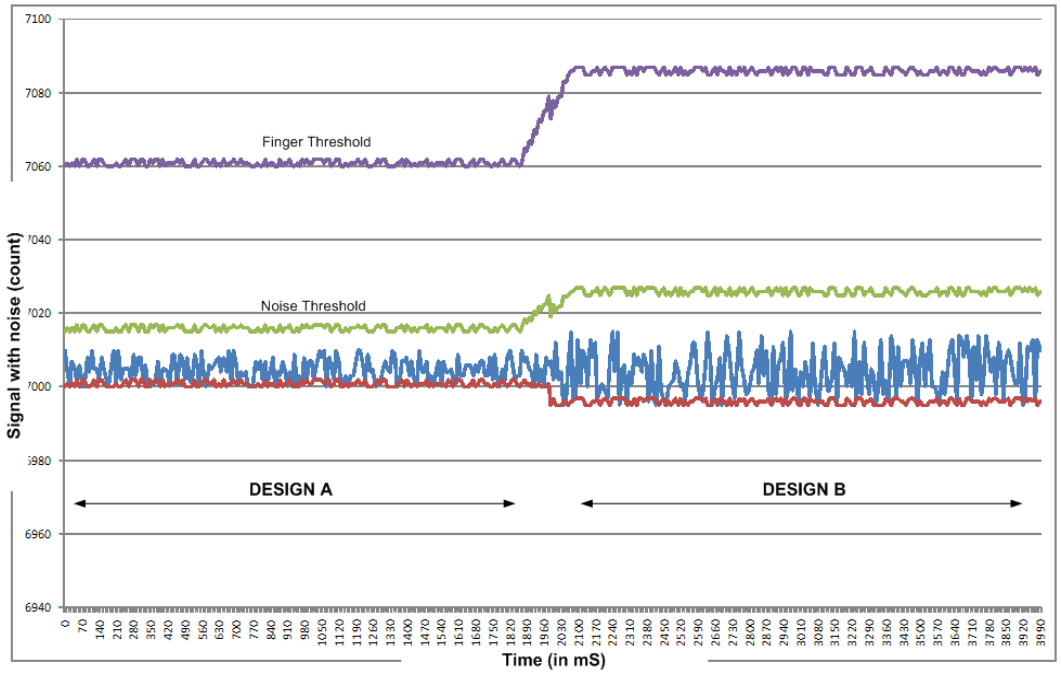
SmartSense technology adapts for manufacturing variations in PCBs, overlays, and noise generators, such as LCD inverters, AC line noise, and switch-mode power supplies and automatically tunes them out. SmartSense handles changes in system environment, such as temperature, humidity, and noise sources such as RF, SMPS, LCD Inverter, and AC line noise.

The following sections describe scenarios in which SmartSense is instrumental in adapting to the external noise. By maintaining a robust signal-to-noise ratio, the false triggering of buttons is prevented.

2.5.2.3.1 Different Noise Levels in Different Designs

SmartSense technology dynamically tunes itself (adjusts noise and finger thresholds) for different noise environments. In Figure 2-16, Design A and Design B have different noise levels. To maintain a minimum SNR of 5:1, you must adjust the dynamic threshold. SmartSense does this automatically, allowing seamless transition from one model to another with minimal or no tuning required.

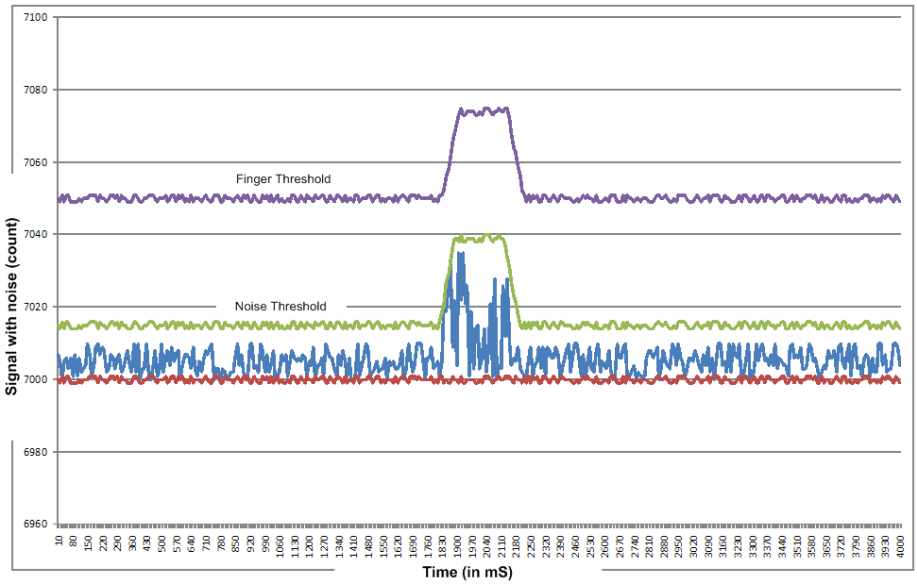
Figure 2-16. Different Noise Levels in Design A and B Being Compensated Automatically



2.5.2.3.2 Noise Spikes During Production

SmartSense technology also automatically tunes out the noise spikes (in production) that may not be seen during the design stage, as indicated in Figure 2-17. This is a powerful SmartSense feature that prevents false button presses in the end system, which prevents a failure analysis for a mass production design.

Figure 2-17. Finger Threshold Dynamically Adjusted to Prevent False Button Touches



2.5.2.4 When Is Manual-Tuning Advantageous?

SmartSense 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 the case where SmartSense will not work or there are specific SNR or power requirements, the CapSense CSD parameters can be manually adjusted to meet system requirements. This is called manual tuning. Some advantages of manual tuning, as opposed to SmartSense Auto-tuning are:

- Strict control over parameter settings: SmartSense sets all of the parameters automatically. However, there may be situations where you need strict control over the parameters. For example, use manual tuning if you need to strictly control the time CSD takes to scan a group of sensors. 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, use manual tuning.

See the device-specific Design Guide for the step-by-step procedure on manual tuning.

2.6 Signal-to-Noise Ratio (SNR)

Signal is a generic engineering term that can have many meanings. For CapSense applications, signal is defined as the change in the raw count between the OFF and ON states. Signal is also called Difference Count.

Noise is another term that has many meanings. The following discussion presents a definition of CapSense noise that uses a simple mathematical model of the sensor output over time.

When the sensor is in the OFF state, the counts, X(t), can be modeled by an average count and a noise component.

$$X(t) = X0 + N0(t)$$ Equation 6

- X0 is the average of X(t)
- N0(t) is the noise component for t during the OFF state

The same model applies when the sensor is in the ON state.

$$X(t) = X1 + N1(t)$$ Equation 7

- X1 is the average of X(t)
- N1(t) is the noise component for t during the ON state

X0 is called the baseline level of the raw counts. The difference between X1 and X0 is called the signal, S.

$$S = X1 - X0$$ Equation 8

The noise components N0(t) and N1(t) are similar but not identical. For example, N1(t) usually contains a higher level of AC line noise in finger sensing applications compared to N0(t). This occurs because the human body acts as an antenna to 50-Hz and 60-Hz line noise, and the finger contact with the sensor overlay couples the noise into the CapSense system.

We define the noise level N as the worst case measured peak noise in the OFF state.

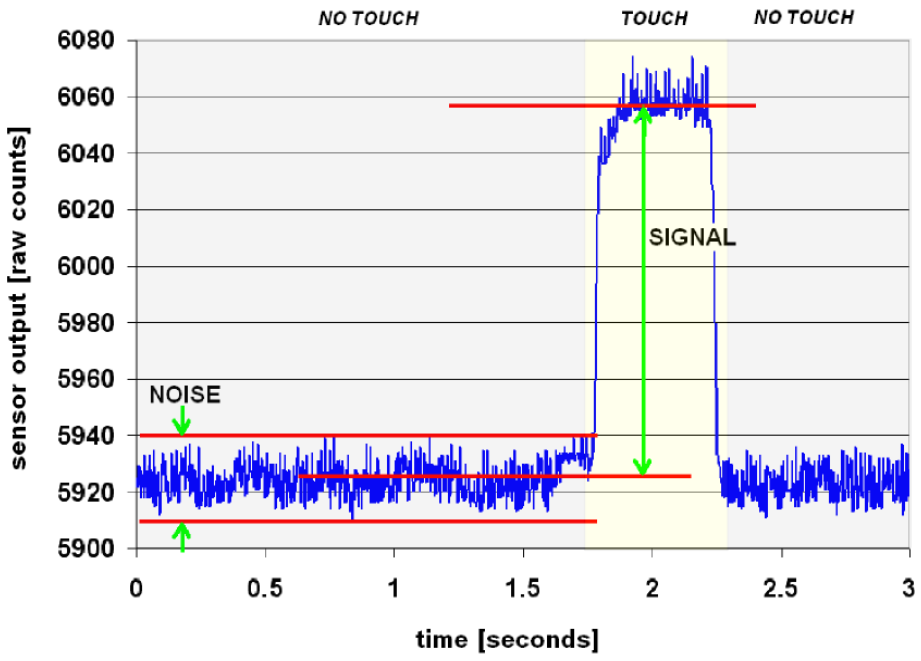
$$\mathrm{N} = \max(\mathrm{N0(t)}) = \max(\mathrm{X(t)}) - \min(\mathrm{X(t)})$$ Equation 9

Thus, CapSense SNR, is defined as the ratio of signal (S) to noise (N).

$$SNR = S{:}N$$ Equation 10

Based on the experiments and knowledge from many CapSense applications, Cypress recommends a minimum SNR of 5:1 to ensure sufficient margin between noise and signal for robust ON/OFF operation.

Figure 2-18. Signal and Noise



2.6.1 Measuring SNR

SNR should be measured in the noise environment where CapSense is intended to be used. In other words, measure the system SNR under worst-case noise conditions.

The first step in measuring SNR is to monitor the raw count for each sensor. This can be done using data logging to a text file and plotting in a spreadsheet, or using the Cypress Bridge Control Panel and Miniprog3 or MiniProg4[5] or by using the Tuner tool, which directly displays the SNR, available with the CapSense component in PSoC Creator (see Data Monitoring Tools for more details). See AN2397 – CapSense Data Viewing Tools that teaches how to monitor raw count using these tools. Whatever the method, the raw count should be observed for SNR measurement. The difference count should not be used in the measurement of SNR since it is a function of the baseline update process, which involves filtering (filling the "bucket") and nonlinear threshold events.

Another factor to consider is how the signal is produced. The worst-case ON and OFF scenario should be used when measuring SNR. If the system is designed to sense the presence of a finger, then measure SNR with a light touch of the sensor area, and position the contact point slightly off-center. For automated testing, a worst-case finger touch (0.1 pF) can often be simulated by an equivalent metal disc that is the size and shape of a small coin.

As an example of measuring SNR, consider the raw count waveform in Figure 2-18.

X0 = 5925 counts

X1 = 6055 counts

S = 130 counts

N = 5940 - 5910 = 30 counts

SNR = 130:30 = 4.3:1

2.7 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 2-19 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 2-19. Several Types of Widgets

Button Sensor | Slider Sensor | Proximity Sensor

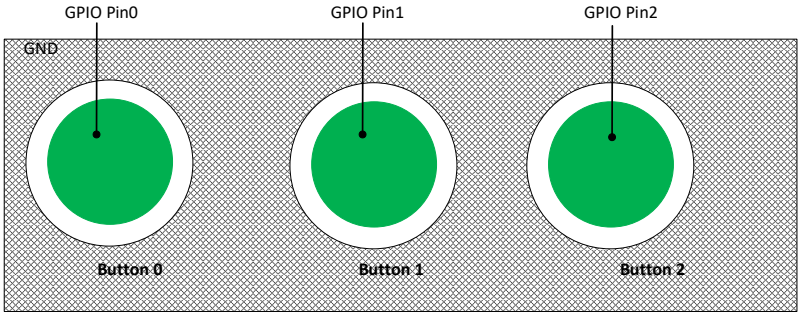
[5] To know more about the Programming and Debugging kits, see Kits for Programming and Debugging.

2.7.1 Buttons (Zero-Dimensional)

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

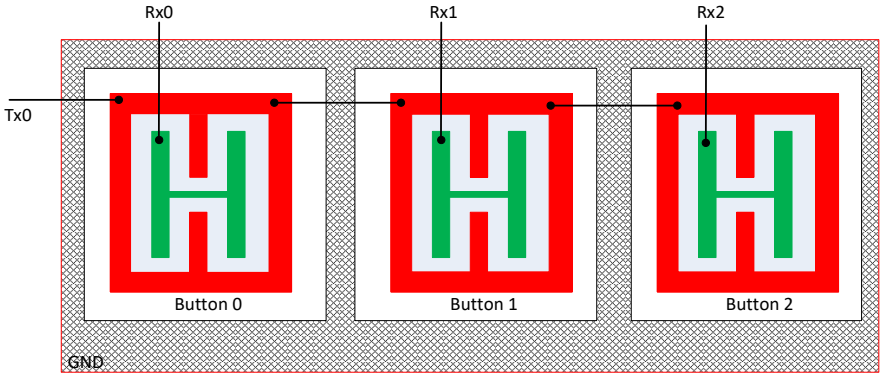
For the self-capacitance based i.e. CSD sensing method, a simple CapSense button consists of a circular copper pad connected to a PSoC GPIO with a PCB trace. The button is surrounded by grounded copper hatch to isolate it from other buttons and traces. A circular gap separates the button pad and the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

Figure 2-20. Simple CapSense Buttons for Self-Capacitance Sensing Method



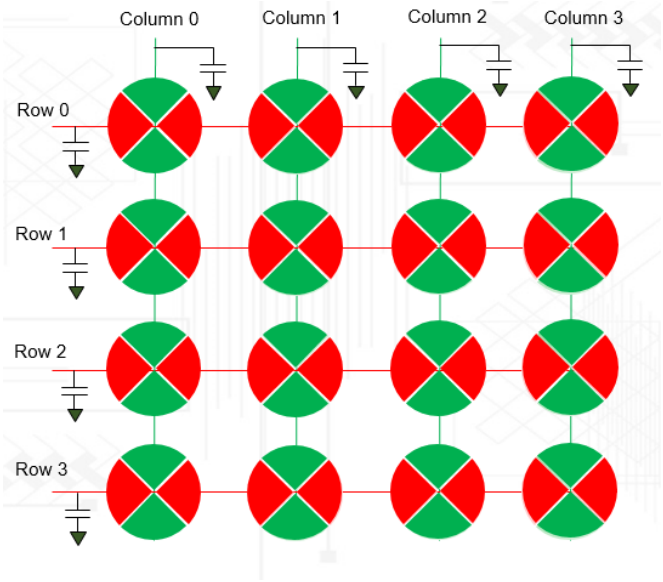
For the mutual-capacitance based i.e. CSX sensing method, each button requires one GPIO pin configured as Tx electrode and one GPIO pin configured as Rx electrode. The Tx pin can be shared across multiple buttons, as shown in Figure 2-21.

Figure 2-21. Simple CapSense Buttons for Mutual-Capacitance Sensing Method



If the application requires many buttons, such as in a calculator keypad or a QWERTY keyboard, you can arrange the CapSense buttons in a matrix, as Figure 2-22 shows. This allows a design to have multiple buttons per GPIO. For example, the 12-button design in Figure 2-22 requires only eight GPIOs.

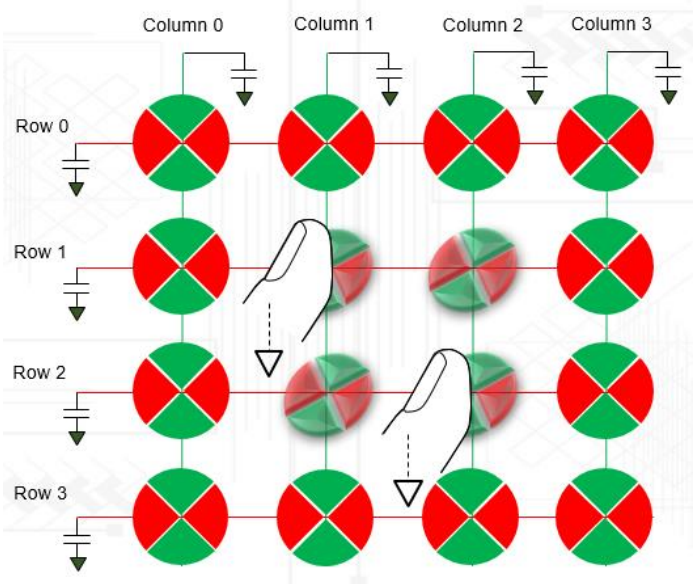
Figure 2-22. Matrix Buttons Based on CSD



A matrix button design has two groups of capacitive sensors: row sensors and column sensors. The matrix button architecture can be used for both self-capacitance (CSD) and mutual-capacitance (CSX) methods.

In self-capacitance mode, each button consists of a row sensor and a column sensor, as Figure 2-22 shows. When a button is touched, both row and column sensors of that button become active. The CSD-based matrix button should be used only if the user is expected to touch one button at a time. If the user touches more than one diagonally opposite buttons, the finger location cannot be resolved as Figure 2-23 shows. This effect is called as ghost effect, which is considered an invalid condition.

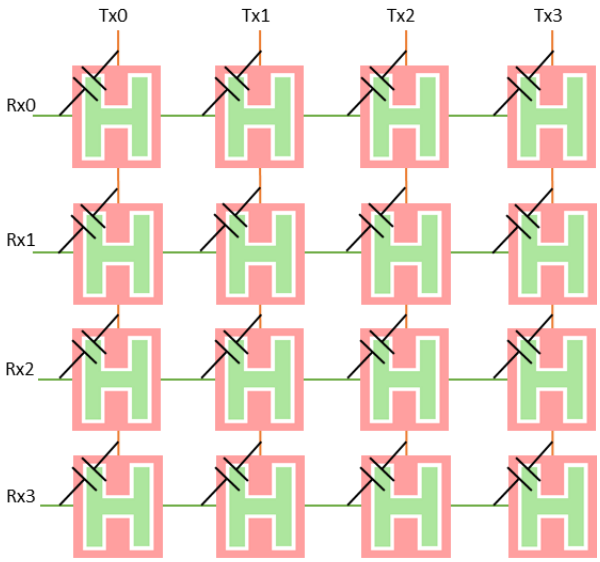
Figure 2-23. Ghost Effect in Matrix Button Based on CSD



Mutual capacitance is the recommended sensing method for matrix buttons because it doesn't suffer from ghost touch and provides better SNR for high Cp sensors. This is because it senses mutual capacitance formed at each intersection

rather than sensing rows and columns as shown in Figure 2-24. Applications that require simultaneous sensing of multiple buttons, such as a keyboard with Shift, Ctrl, and Alt keys can use mutual-capacitance sensing method or you should design the Shift, Ctrl, and Alt keys as individual CSD buttons.

Figure 2-24. Matrix Button Based on CSX



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

2.7.2 Sliders (One-Dimensional)

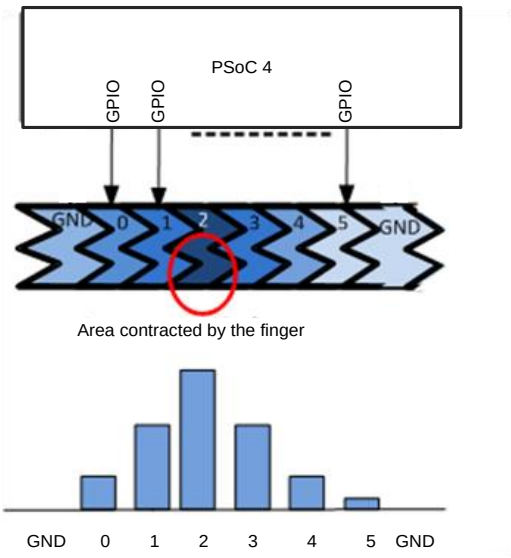
Sliders are used when the required input is in the form of a gradual increment or decrement. Examples include lighting control (dimmer), volume control, graphic equalizer, and speed control. Currently, the CapSense Component in PSoC Creator and ModusToolbox supports only self-capacitance-based sliders.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

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

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

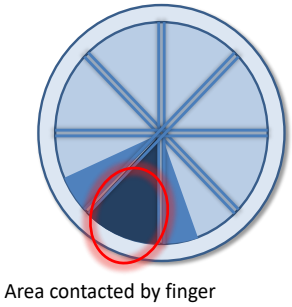
In a linear slider, the segments are arranged inline, as Figure 2-25 shows. Each slider segment connects to a PSoC GPIO. A zigzag pattern (double chevron) is recommended for slider segments. This layout ensures that when a segment is touched, the adjacent segments are also partially touched, which aids estimation of the centroid position.

Figure 2-25. Linear Slider



Radial sliders are similar to linear sliders except that radial sliders are continuous. Figure 2-26 shows a typical radial slider.

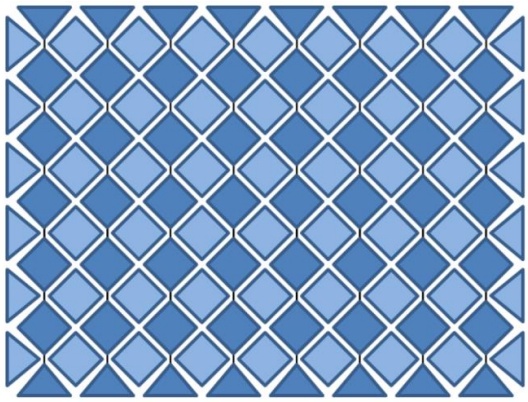
Figure 2-26. Radial Slider



2.7.3 Touchscreens and Trackpads (Two-Dimensional Sensors)

A trackpad (also known as touch pad) 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 2-27 shows a typical arrangement of a track pad sensor.

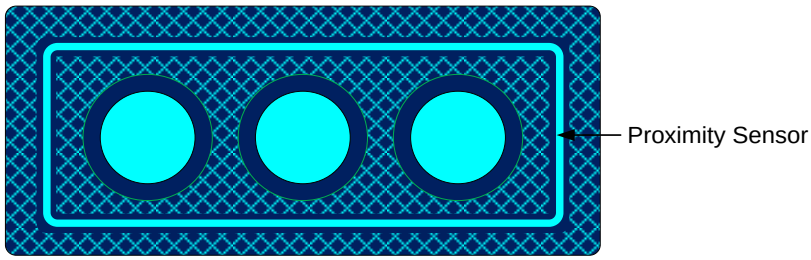
Figure 2-27. Trackpad Sensor Arrangement



2.7.4 Proximity (Three-Dimensional Sensors)

Proximity sensors detect the presence of a hand or other conductive object before it makes contact with the touch surface. Imagine a hand stretched out to operate a car audio system in the dark. The proximity sensor causes the buttons of the audio system to glow through backlight LEDs when the user's hand is near. One implementation of a proximity sensor consists of a long trace on the perimeter of the user interface, as shown in Figure 2-28. Another way to implement a proximity sensor is by ganging sensors together. See Proximity Sensing to learn more.

Figure 2-28. Proximity Sensor

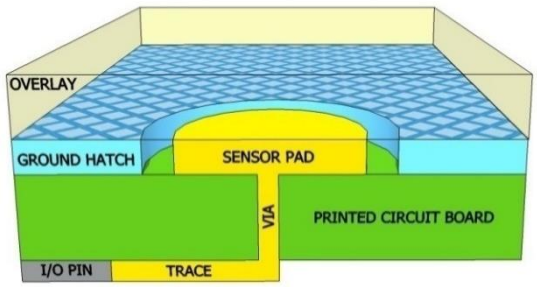


2.8 Sensor Construction

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 the user touches 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. A very common method of sensor construction is to etch copper pads and traces on a FR4 PCB. However, in touchscreen applications, Indium Tin Oxide (ITO) is used to construct transparent sensors. This section presents various methods of constructing a sensor and the features of each method so that you can choose one that fits your requirements.

2.8.1 Field-Coupled Via Copper Trace (PCB)

Figure 2-29. Field Coupled Using PCB

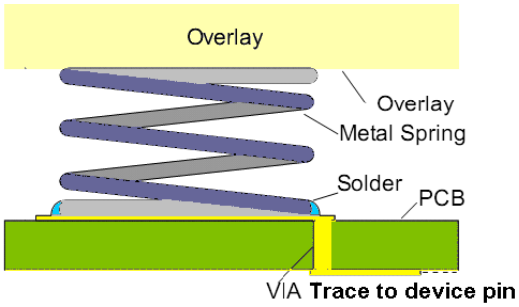


Features of a PCB-based design:

- Most common implementation
- Copper pads etched on the surface of the PCB act as sensor pads
- Electric field emanates from the copper sensor pad to ground plane
- No mechanical moving parts
- A nonconductive overlay serves as the touch surface for the button
- Ideal topology for simple flat panel designs
- Low BOM cost

2.8.2 Field Coupled Via Spring/Gasket/Foam

Figure 2-30. Field Coupled Via Spring



Features of a design based on springs/gaskets/foam:

- Electrical field coupled from PCB to overlay using a compressed spring, or conductive gasket or foam (Closed-cell conductive foam should be used. Materials that absorb moisture should be avoided.)
- Conductive material itself acts as capacitive sensor pad
- No mechanical moving parts. Springs and foam do not move
- Any non-conductive overlay serves as the button touch surface
- Ideal topology for curved, sloping, or otherwise irregular front panels
- Ideal for designs where touch sensor surface is physically separated from silicon or mother board
- Ideal for designs where CapSense and mechanical button combination is desired

2.8.3 Field Coupled Via Printed Ink

Features of a design based on printed ink:

- Electric field coupled with printed patterns on a flexible substrate using conductive ink
- High series resistance due to higher sheet resistance (ohms-per-square) of printed ink compared to copper. However, the series resistance of the materials such as silver loaded inks, ITO, and PEDOT depend on the thickness variation as their sheet resistance is relatively low.
- High parasitic capacitance due to thin PCB substrate
- No mechanical moving parts, but substrate is flexible
- Coupled to the touch sensor surface with a nonconductive overlay
- Ideal topology for flexible front panels
- Flexible PCB can be one-layer or two-layer film

2.8.4 Field Coupled via ITO Film on Glass

Features of a design based on ITO film:

- Electric field coupled with printed or deposited patterns on glass
- Higher series resistance of ITO films compared to copper
- No mechanical moving parts
- Ideal topology for graphical front panels

2.9 Liquid Tolerance

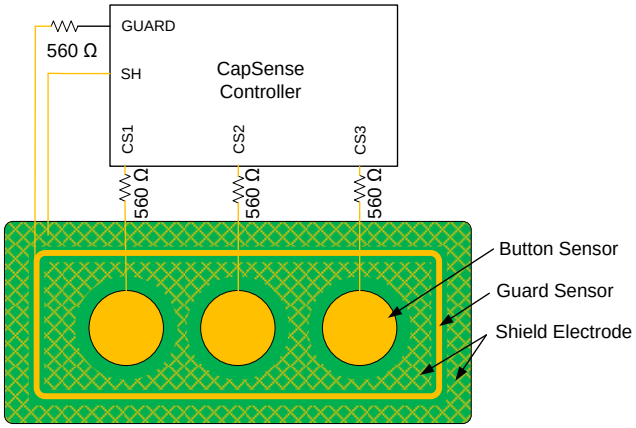
CapSense is used in a variety of applications such as home appliances, automotive, and industrial applications. These applications require robust CapSense operation even in the presence of mist, moisture, water, ice, and humidity changes. In a CapSense design, false sensing of touch may happen due to the presence of a film of liquid or liquid droplets on the touch surface. Cypress' CapSense sensing method can compensate for variation in raw count due to mist, moisture, water, ice, and humidity changes and provide a robust, reliable, CapSense operation.

Figure 2-31. CapSense-based Touch User Interface in a Washing Machine with Liquid Tolerance

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 sensor false triggers. To compensate for changes in raw count due to a liquid droplet or liquid flow, you should implement a Shield Electrode and a Guard Sensor to provide robust touch sensing, as Figure 2-32 shows. All sensor pins can optionally have a 560-Ω series resistor for improved noise immunity.

When liquid droplets are present on the touch surface and if the shield electrode is implemented, the CapSense system can reliably work even in the presence of liquid droplets and report sensor ON/OFF status. When there is a liquid flow or a liquid pool on the touch surface, the CapSense system detects the liquid by using a guard sensor and disables the scanning for all other sensors in the system to prevent false triggers. Therefore, when there is a liquid flow or liquid pool on the touch surface, the CapSense system will not detect a finger touch as long as the liquid is present on the touch surface.

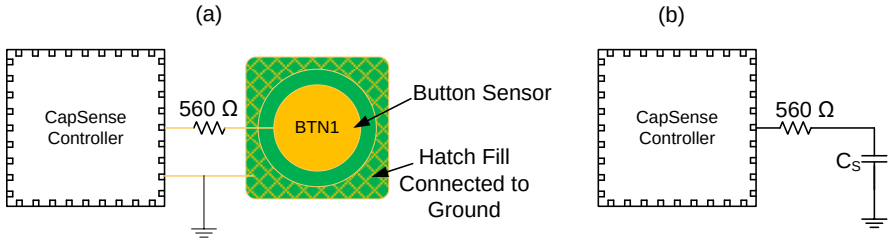
Figure 2-32. Shield Electrode (SH) and Guard Sensor (GUARD) Connected to CapSense Controller



2.9.1 Effect of Liquid Droplets and Liquid Stream on CapSense

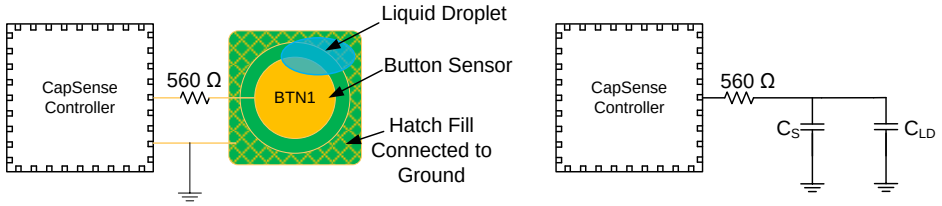
To understand the effect of a liquid droplet or a liquid stream on a CapSense sensor, consider a CapSense system in which the hatch fill around the sensor is connected to ground, as Figure 2-33 (a) shows. Surrounding the sensor with a hatch fill connected to ground improves the noise immunity of the sensor. The parasitic capacitance of sensor is denoted as C_S in Figure 2-33 (b).

Figure 2-33. Typical CapSense System Layout



As shown in Figure 2-34, when a liquid droplet falls on the touch surface, due to its conductive nature, it provides a strong coupling path for the electric field lines to return to ground and therefore adds a capacitance C_{LD} in parallel to C_P. When the sensor is charged and discharged, the capacitance C_{LD} draws some amount of charge from the AMUX bus because of the nonzero voltage difference across the capacitance C_{LD}. This increases the overall capacitance seen by the CapSense circuitry and results in an increase in the sensor raw count. In some cases, where the liquid is highly conductive (salty water or water with high mineral content), the increase in raw count when a liquid droplet falls on the touch surface might be equal to the increase in raw count due to a finger touch and thus causes false triggers, as Figure 2-35 shows.

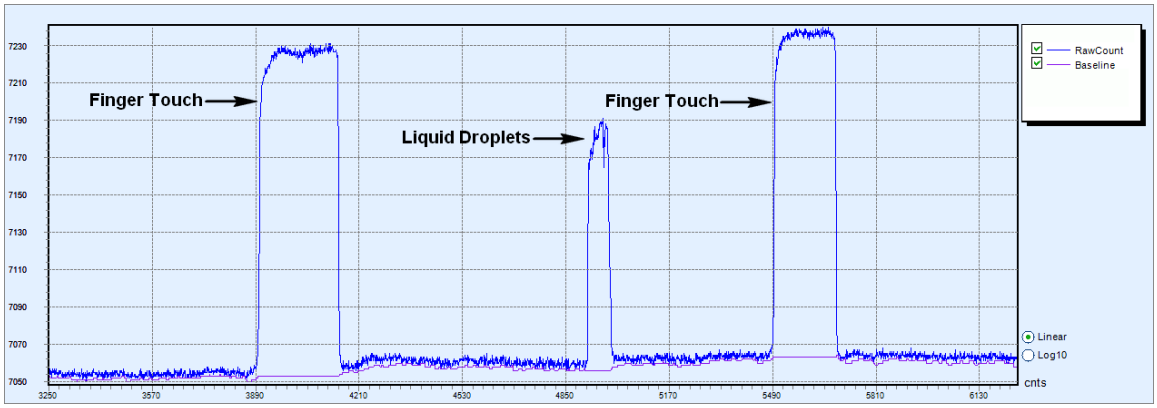
Figure 2-34. Capacitance Added by Liquid Droplet when the Hatch Fill Is Connected to Ground



C_S – Sensor Parasitic Capacitance

C_{LD} – Capacitance added by Liquid Droplet

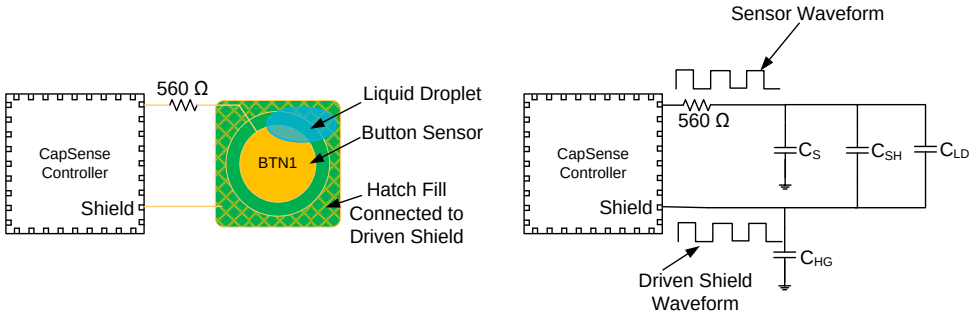
Figure 2-35. Effect of Liquid Droplet when the Hatch Fill Around the Sensor Is Connected to Ground



To compensate the capacitance added by the liquid droplet to the CapSense circuitry, you should drive the hatch fill surrounding the sensor with the driven-shield signal.

As shown in Figure 2-36, when the hatch fill surrounding the sensor is connected to the driven-shield signal and when a liquid droplet falls on the touch surface, because the voltage on both the sides of liquid droplet is kept at the same potential, the capacitance C_{LD} added by the liquid droplet is nullified. Because the voltage difference across the capacitance C_{LD} is zero, it will not draw any charge from the AMUX bus and, therefore, the increase in the raw count when a liquid droplet falls on the sensor will be very small, as Figure 2-37 shows.

Figure 2-36. Capacitance Added by Liquid Droplet when the Hatch Fill around the Sensor Is Connected to Shield



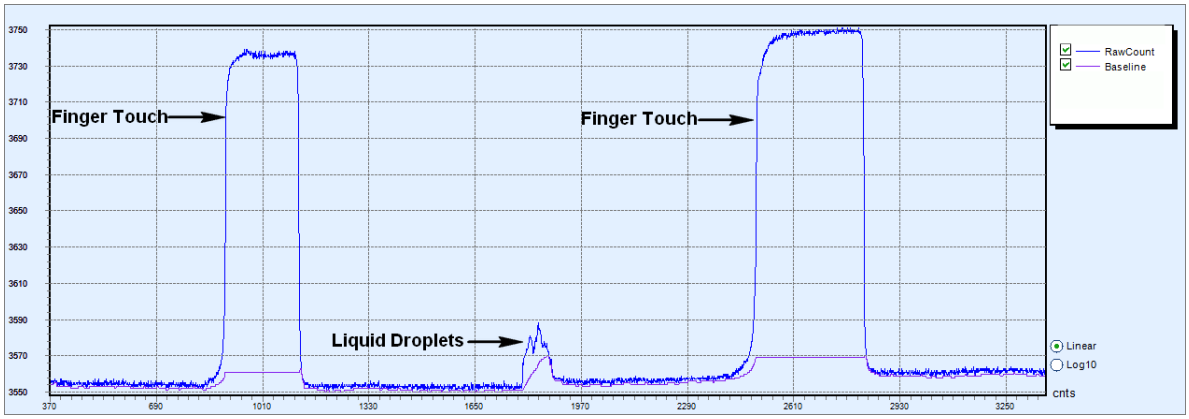
C_S – Sensor Parasitic Capacitance

C_{SH} – Capacitance between Sensor and Hatch Fill

C_{HG} – Capacitance between Hatch Fill and Ground

C_{LD} – Capacitance Added by Liquid Droplet

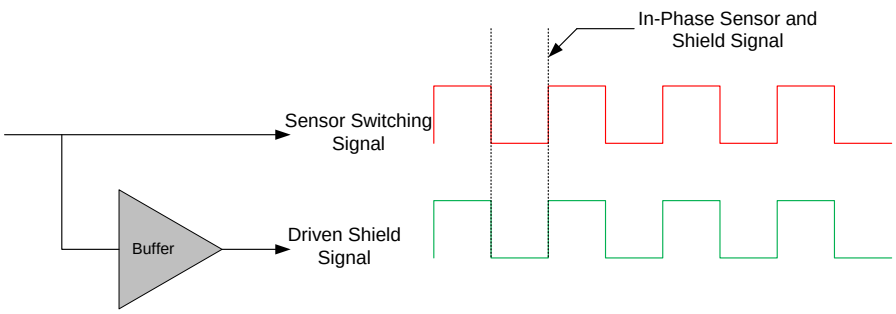
Figure 2-37. Effect of Liquid Droplet when the Hatch Fill Around the Sensor Is Connected to the Driven Shield



2.9.2 Driven-Shield Signal and Shield Electrode

The driven-shield signal is a buffered version of the sensor-switching signal, as Figure 2-38 shows. The driven-shield signal has the sample amplitude, frequency, and phase as that of the sensor-switching signal. The buffer provides sufficient current for the driven-shield signal to drive the high parasitic capacitance of the hatch fill on the PCB. When the hatch fill surrounding the sensor is connected to the driven-shield signal, it is referred as Shield Electrode. As explained in Effect of Liquid Droplets and Liquid Stream on CapSense, because the shield electrode is driven with a voltage which is the same as the sensor-switching signal, the capacitance added by a liquid droplet when it falls on the touch surface will be nullified. To achieve the best liquid-tolerance performance, it is required that the driven shield signal has the same voltage and phase as that of the sensor-switching signal.

Figure 2-38. Driven-Shield Signal

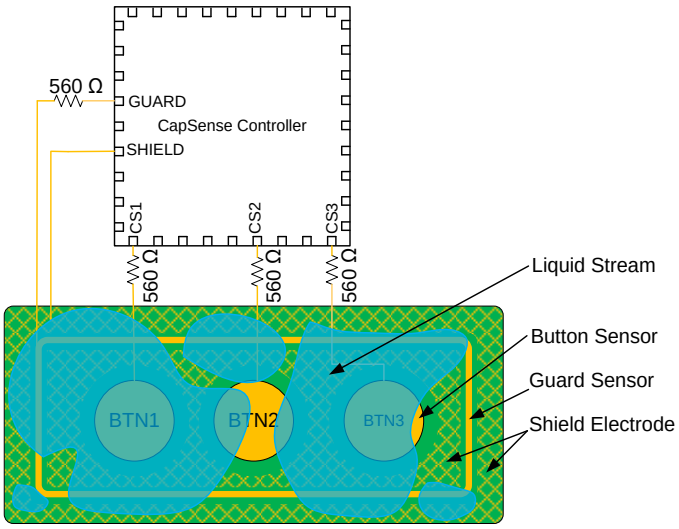


2.9.3 Guard Sensor

When a continuous liquid stream is applied to the touch interface, the liquid stream adds a large capacitance (C_{ST}) to the CapSense circuitry. This capacitance might be several times larger than C_{LD}. Because of this, the effect of the shield electrode is completely masked, resulting in the increase in sensor raw count potentially higher than that due to a finger touch. In such situations, the guard sensor is useful to prevent a false touch-sensing.

A guard sensor is a copper trace that surrounds all the sensors on the PCB, as Figure 2-39 shows. A guard sensor is similar to a button sensor and is used to detect the presence of liquid stream. When a guard sensor is triggered, the firmware can disable the scanning of all other sensors in the system to prevent a false touch-sensing. Because the sensors are not scanned when the guard sensor is triggered, the touch cannot be detected when there is a liquid stream.

Figure 2-39. Capacitance Measurement with a Liquid Stream

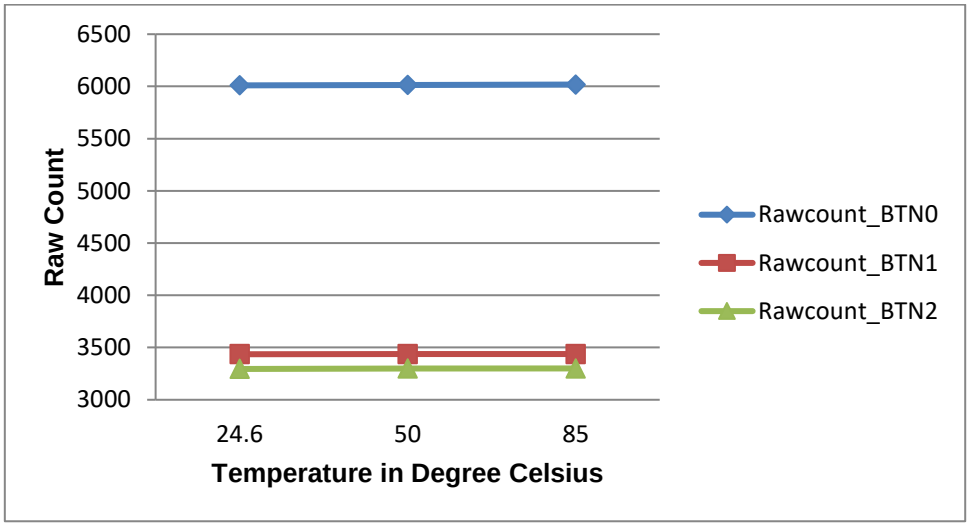


2.9.4 Effect of Liquid Properties on the Liquid-Tolerance Performance

In certain applications, the CapSense system has to work reliably in the presence of a variety of liquids such as soap water, sea water, and water with high mineral content. In such applications, it is recommended that you tune the CapSense parameters for the sensors by considering the worst-case shift in the raw count due to liquids on the touch surface. To simulate the worst-case condition, you can prepare a salty water solution by dissolving 40 gm of cooking salt (NaCl) in 1 liter of water and measure the shift in raw counts when water droplets fall on the sensor.

In applications such as an induction cook-top, there might be chances of hot water spilling on the CapSense touch surface. To determine the impact of temperature of a liquid droplet on the liquid-tolerance performance, tests were done with liquid droplets at different temperatures. Experiment results show that the effect of hot liquid droplets is the same as that of the liquid droplets at room temperature. This is because a hot liquid droplet cools down immediately to room temperature when it falls on the touch surface.

Figure 2-40. Raw Count Variation versus Water Temperature



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

1. Choose a CapSense controller that supports the liquid tolerance feature. See the CapSense Selector Guide to select the CapSense controller that supports the liquid tolerance feature.
2. Follow the schematic and layout guidelines explained in the device-specific Design Guide to construct the shield electrode and guard sensor.
3. Tune the guard sensor (if implemented) such that it is triggered only when there is a liquid stream. In the firmware, ensure that the sensors are not scanned when the guard sensor is triggered.

See the individual CapSense design guides for detailed procedures of how to tune the CapSense parameters to achieve liquid tolerance. The application note, AN92239 – Proximity Sensing with CapSense, shows how to implement a proximity-sensing system with liquid tolerance for PSoC 4 devices.

2.10 Proximity Sensing

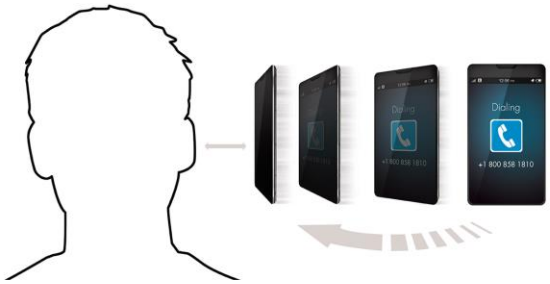
Proximity sensing is the process of detecting a nearby object without any physical contact. Proximity sensors use an electromagnetic field, beam of electromagnetic radiation, or changes in ambient conditions to detect the proximity of a nearby object. There are various types of proximity sensors such as capacitive, inductive, and magnetic, Hall effect, optical, and ultrasonic sensors; each has its own advantages and disadvantages. Capacitive proximity sensing has gained huge popularity because of its low cost, high reliability, low power, sleek aesthetics, and seamless integration with existing user interfaces. This section presents an introduction to CapSense-based proximity sensing. See AN92239 – Proximity Sensing with CapSense for the complete proximity design guidelines.

2.10.1 Proximity-Sensing Applications Based on CapSense

Proximity sensing based on CapSense is used in a variety of applications as explained below.

Face detection in mobile phones and tablets: Face detection is a feature in mobile phones that disables the phone's touchscreen and dims the brightness of the display when a user answers a phone call, as Figure 2-41 shows. Face detection prevents false touches when the phone is placed on the ear and optimizes the device's power consumption. Using proximity sensing based on CapSense in this application offers advantages over IR proximity sensing because it does not require cutouts in the overlay material, which reduces the tooling cost and improves the aesthetics of the end product.

Figure 2-41. Face Detection in Mobile Phones



SAR regulation in tablets and mobile phones: SAR is a measure of the rate at which energy is absorbed by the human body when exposed to an RF electromagnetic field. Regulatory bodies, such as the Federal Communications Commission (FCC), require devices to limit the absorption of RF energy by reducing the device's RF transmit power when the device is in close proximity to the human body, as Figure 2-42 shows. Proximity sensors based on CapSense can be used to detect the proximity of the human body and reduce RF power.

Figure 2-42. SAR Regulation in Tablets

Wake-on-approach: This feature activates the system from the sleep or standby mode when an object approaches the system, as Figure 2-43 shows. Wake-on-approach is also used to control the backlight LEDs when the user approaches the system. This feature reduces system wakeup time, improves device responsiveness, reduces device power consumption, and improves aesthetics. It is useful in battery-operated applications such as mice and keyboards.

Figure 2-43. Wake-on-Approach Implemented in a Mouse

Gesture detection: Gesture detection is the technique of interpreting human body movements and providing gesture-type information to the device. Gesture-based user interfaces provide an intuitive way for the user to interact with the system, improving the user experience. Gesture detection is used in applications such as laptops, tablets, and mobile phones for controlling the user interface.

Proximity sensors based on CapSense can be used in these applications to detect gestures without any physical contact between the user and the device. Figure 2-44 shows an example of an implementation of gesture detection in a laptop where proximity sensors placed near the trackpad are used for the pan movement of the onscreen map.

Figure 2-44. Gesture Detection Implementation in a Laptop

IR replacement: Proximity sensors based on CapSense can replace IR proximity sensors in applications such as faucets and soap dispensers, as Figure 2-45 shows. Proximity sensing based on CapSense offers the following advantages over IR proximity sensing:

- It is a low-cost solution compared to IR proximity sensing. Proximity sensors based on CapSense can be constructed using a copper trace on the PCB, whereas IR proximity sensing requires extra IR sensors.
- Proximity sensors based on CapSense do not require any cutout in the overlay material to detect proximity sensing, unlike IR proximity sensors. Therefore, proximity sensing based on CapSense reduces the tooling cost and improves the aesthetics of the end product.
- They consume less power than IR proximity sensors.
- They are immune to ambient light, whereas IR-based proximity sensors can have performance issues with varying ambient light.

Figure 2-45. Proximity Sensing Based on CapSense in a Soap Dispenser

2.10.2 Proximity Sensing with CapSense

The proximity-sensing technique based on CapSense involves measuring the change in capacitance of a proximity sensor when a target object approaches the sensor. The target object can be a human finger, hand, or any conductive object. Proximity sensors can be constructed using a conductive (usually copper or indium tin oxide) pad or trace laid on a nonconductive material like PCB or glass. In essence, a proximity sensor is like any other sensor but designed with very minimum ground near the sensor and tuned for maximum sensitivity.

For detecting the target object, the Signal-to-Noise Ratio (SNR) should be greater than or equal to 5:1. Therefore, you can detect the proximity of the target object without error up to a certain distance from the sensor. This distance is

called the proximity-sensing distance. See Proximity Sensing Design to learn the various ways of constructing a proximity sensor and to learn the various parameters that affect the proximity distance.

2.11 User Interface Feedback

Effective user interface designs include feedback to the user when they are using the capacitive touch sense buttons. There are various forms of feedback, including visual, audio, and haptic (tactile). Depending on the user interface design, multiple types of feedback can be used in combination.

2.11.1 Visual Feedback

LEDs and LCDs provide visual feedback.

2.11.1.1 LED-Based Visual Feedback

Visual feedback is widely used in user interfaces. LEDs are used to indicate the status of buttons, sliders, and proximity sensors. LEDs can implement different effects when the sensor status changes as listed below.

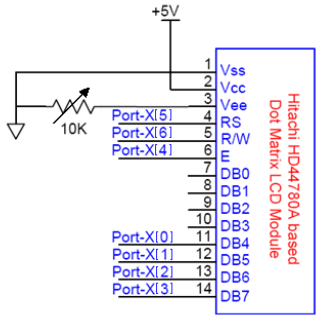
- LED ON/OFF – GPOs are used to drive LEDs in either a sourcing (GPO supplies current to the LED) or sinking (GPO sinks current from the LED) configuration.
- LED Brightness Control - For user interfaces that require sophisticated visual effects, a single hardware PWM or timer can be used to drive the LEDs. By varying the duty cycle of the PWM output, you can adjust the LED brightness. This enables adjusting your user interface brightness in response to ambient lighting conditions.
- LED Fading - By gradually changing the duty cycle between LED states, you can achieve a fading effect. For example, the LED appears to “fade in” (from OFF to ON) when the duty cycle is increased in a series of small steps.
- LED Breathing - Gradually increasing and decreasing the duty cycle between two levels on a continuous basis makes the LED appear to “breathe”. LED breathing is useful when a system is in idle or stand-by mode. For example, a power button can appear to breathe to alert the user that it is active and can be operated.

2.11.1.2 LCD-Based Visual Feedback

LCDs provide visual feedback for CapSense buttons and sliders. The main advantage of using an LCD is that it can provide more information along with the feedback for each button press event. Depending on the device family, programmable devices support different types of LCD technologies with pre-built components and user modules which provide APIs for displaying data with ease.

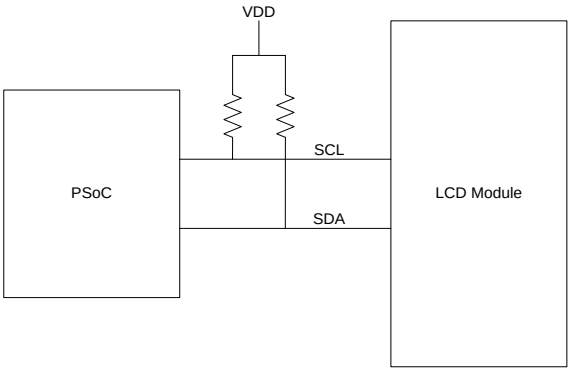
- Character LCD with parallel interface - PSoC devices support interface with the Hitachi HD44780A LCD module. Figure 2-46 shows the typical connection for using the Hitachi HD44780A LCD module. See the character LCD component datasheet to learn more.

Figure 2-46. Hitachi Dot Matrix LCD Pin Connections



- Character LCD with I^2C Interface - PSoC can also control LCDs through I^2C with support for the NXP PCF2119x command format. Figure 2-47 shows the typical circuit diagram for driving an LCD with I^2C interface. See the component datasheet to learn more.

Figure 2-47. Interfacing an LCD with I2C Interface

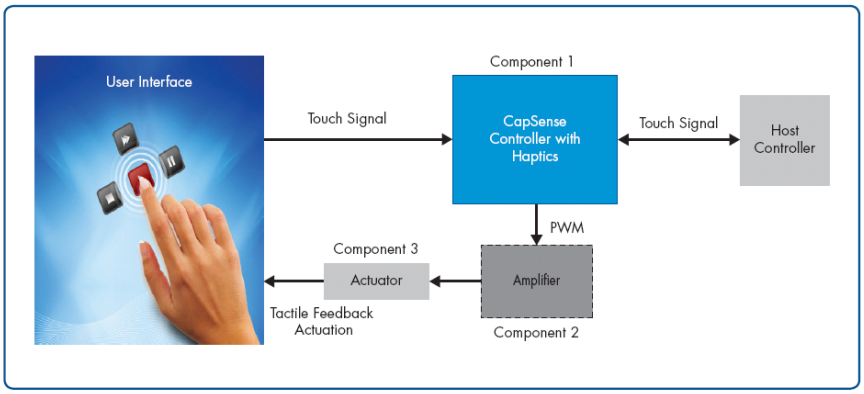


- Segment LCD Glass - PSoC devices integrate the LCD driver to directly drive a segment LCD glass. See the CapSense Selector Guide to know the LCD drive capability of different PSoC devices.

2.11.2 Haptic Feedback

Haptic or tactile feedback uses vibration to let you know that the system has detected a finger touch.

Figure 2-48. Cypress Haptics Ecosystem



Different haptic effects are created by varying the duration, frequency, and shape of the vibration profile. Vibrations are created by an actuator. The key requirements for an actuator are - Response time, Power consumption, Size, Form factor, Durability, and Vibration strength. Multiple actuator options are available with varying drive requirement. The four types of actuators are:

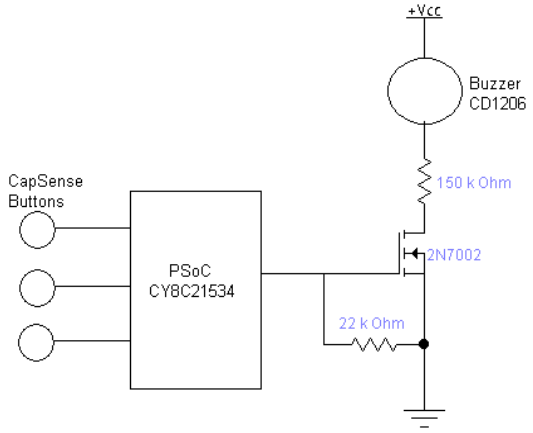
- Eccentric Rotation Mass (ERM) actuators – These actuators are most cost-effective, have a current requirement of 130 to 160 mA, and require an external power amplifier for drive. These are not suitable for applications requiring fast response such as typing.
- Linear Resonant Actuators (LRAs) - LRAs are used in many smartphones, require AC input, draw current at around 65 to 70 mA, are relatively faster, but cost more than ERM actuators. LRAs require an external power amplifier for drive.
- Piezo Modules – Piezo modules respond fast, making them suitable for typing applications, but they cost more than ERM and LRA actuators. Though their instantaneous current draw is more at around 300 mA at 3-V supply, their average current consumption is not more than the ERM and LRA actuators.
- Electro-Active Polymer Actuators (EAPs) – EAPs offer a performance similar to that of the piezo modules but integrating them into tight spaces is a challenge. They require 800 V for a drive signal.

The CY8C20XX6H CapSense controller integrates Immersion's TS2000 Haptic effects library for Eccentric Rotating Motor (ERM) drive control and can generate 14 predefined haptic effects. The Haptics User Module datasheet has more information about these effects and also provides a code example. To know more about CY8C20XX6H, see the device datasheet.

2.11.3 Audible Feedback

Audible feedback for CapSense buttons is implemented using a buzzer. The pulse-width modulator (PWM) is used to output the PWM signal required for driving the buzzer as specified in the buzzer datasheet. When a button press event occurs, the feedback is given by driving the buzzer at a particular intensity level. The circuit diagram for implementing the buzzer feedback follows.

Figure 2-49. Implementing Audible Feedback for CapSense



3. Design Considerations

When designing capacitive touch sense technology into your application, it is important to remember that the CapSense device exists within a larger framework. Careful attention to every level of detail from PCB layout to user interface to end-use operating environment will enable robust and reliable system performance.

3.1 Overlay Selection

In a CapSense design, overlay material is placed over the sensor pad to protect it from the environment and prevent direct finger contact.

3.1.1 Overlay Material

In Self Capacitance, Equation 1 shows the finger capacitance.

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

Where:

ε_0 = Free space permittivity

ε_r = Dielectric constant of overlay

A = Area of finger and sensor pad overlap

D = Overlay thickness

The geometry of a CapSense system is more complex than a parallel plate capacitor. The conductors in the sensor include the finger and PCB copper. However, like a parallel plate capacitor, C_F is directly proportional to ε_r. High dielectric constants lead to high sensitivity. Because air has the lowest dielectric constant, any air gaps between the sensor pad and overlay must be eliminated.

Dielectric constants of some common overlay materials are listed in Table 3-1. Materials with dielectrics between 2.0 and 8.0 are well suited to capacitive sensing applications.

Table 3-1. Dielectric Constants of Common Materials

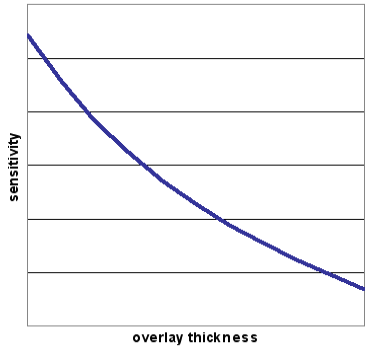
Material	ε_r
Air	1.0
Formica®	4.6–4.9
Glass (Standard)	7.6–8.0
Glass (Ceramic)	6.0
PET Film (Mylar®)	3.2
Polycarbonate (Lexan®)	2.9–3.0
Acrylic (Plexiglass®)	2.8
ABS	2.4–4.1
Wood Table and Desktop	1.2–2.5
Gypsum (Drywall)	2.5–6.0

Conductive material cannot be used as an overlay because it interferes with the electric field pattern. For this reason, do not use paints that contain metal particles in the overlay.

3.1.2 Overlay Thickness

Sensitivity is inversely proportional to overlay thickness, as illustrated in Figure 3-1.

Figure 3-1. Sensitivity Versus Overlay Thickness



Both signal and noise are affected by the overlay properties. Table 3-2 lists the recommended maximum overlay thicknesses for PSoC CapSense applications with an acrylic overlay material.

Table 3-2. Maximum Overlay Thickness with an Acrylic Overlay Material

Design Element	Max. Overlay Thickness (mm)
Button	5
Slider	5
Touchpad	0.5

3.1.3 Overlay Adhesives

Overlay materials must have good mechanical contact with the sensing PCB. This is achieved using a nonconductive adhesive film. This film increases the sensitivity of the system by eliminating any air gaps between overlay and the sensor pads. 3M™ makes a high-performance acrylic adhesive called 200MP that is widely used in CapSense applications in the form of adhesive transfer tape (product numbers 467MP and 468MP).

3.2 ESD Protection

Robust ESD tolerance is a natural byproduct of a thoughtful system design. By considering how contact discharge will occur in your end-product, particularly in your user interface, it is possible to withstand an 18-kV discharge event without any damage to the CapSense controller.

Table 3-3. Overlay Material Dielectric Strength

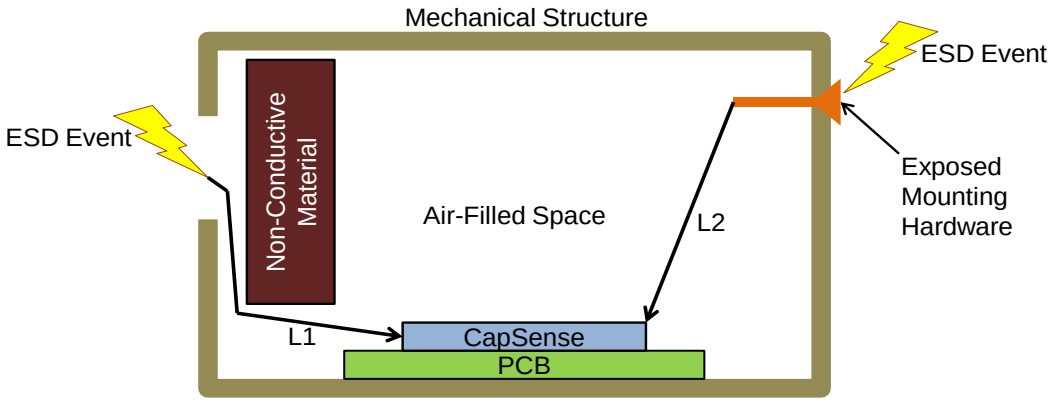
Material	Breakdown Voltage (V/mm)	Min. Overlay Thickness at 12 kV (mm)
Air	1200–2800	10
Wood – dry	3900	3
Glass – common	7900	1.5
Glass – Borosilicate (Pyrex®)	13,000	0.9
PMMA Plastic (Plexiglass)	13,000	0.9
ABS	16,000	0.8
Polycarbonate (Lexan)	16,000	0.8
Formica	18,000	0.7
FR-4	28,000	0.4
PET Film (Mylar)	280,000	0.04
Polymide film (Kapton®)	290,000	0.04

CapSense controller pins can withstand a direct 2-kV event. In most cases, the overlay material provides sufficient ESD protection for the controller pins. Table 3-3 lists the thickness of various overlay materials required to protect the CapSense sensors from a 12-kV discharge as specified in IEC 61000-4-2. If the overlay material does not provide sufficient protection, ESD countermeasures should be applied in the following order: Prevent, Redirect, Clamp.

3.2.1 Preventing ESD Discharge

Preventing the ESD discharge from reaching the CapSense controller is the best countermeasure you can take. Make certain that all paths on the touch surface have a breakdown voltage greater than any voltage to which the surface may be exposed. Also, design your system to maintain an appropriate distance between the CapSense controller and possible sources of ESD. In the example illustrated in Figure 3-2, if L1 and L2 are greater than 10 mm, the system will withstand 12 kV.

Figure 3-2. ESD Path

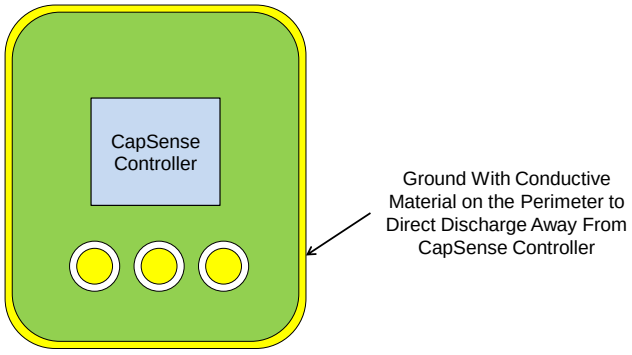


If it is not possible to maintain adequate distance, place a protective layer of a high-breakdown voltage material between the ESD source and CapSense controller. One layer of 5 mil-thick Kapton tape will withstand 18 kV. See Table 3-3 for other material dielectric strengths.

3.2.2 Redirect

If your product is densely packed, it may not be possible to prevent the discharge event. In this case, you can protect the CapSense controller by controlling where the discharge occurs. This can be achieved through a combination of PCB layout, mechanical layout of the system, and conductive tape or other shielding material. A standard practice is to place a guard ring on the perimeter of the circuit board. The guard ring should connect to chassis ground.

Figure 3-3. Guard Ring

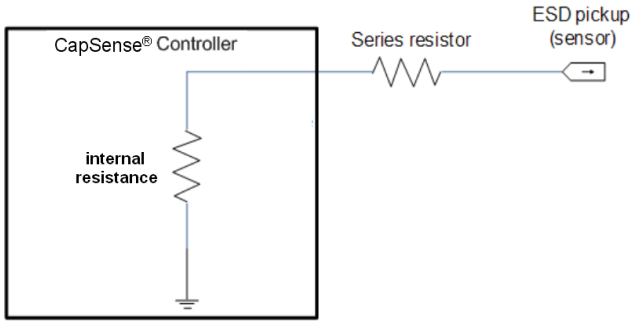


As recommended in PCB Layout Guidelines, providing a hatched ground plane around the button or slider sensor can also redirect the ESD event away from the sensor and CapSense controller.

3.2.3 Clamp

Because CapSense sensors are placed in close proximity to the touch surface, it may not be practical to redirect the discharge path. Including series resistors or special purpose ESD protection devices may be appropriate. Adding a series resistor on the vulnerable traces is a cost-effective protection method. This technique works by splitting the dissipation between the resistor and the controller. The recommended series resistance added to the CapSense inputs is 560 ohms. More details are available in Series Resistor.

Figure 3-4. ESD Protection Using a Series Resistor



A more effective method is to provide special-purpose ESD protection devices on the vulnerable traces. ESD protection devices for CapSense need to be low capacitance. Table 3-4 lists devices recommended for use with CapSense controllers.

Table 3-4. ESD Protection Devices

ESD Protection Device		Input Capacitance	Leakage Current	Contact Discharge Maximum Limit	Air Discharge Maximum Limit
Manufacturer	Part Number				
Littelfuse	SP723	5 pF	2 nA	8 kV	15 kV
Vishay	VBUS05L1-DD1	0.3 pF	0.1 µA <	+/-15 kV	+/-16 kV
NXP	NUP1301	0.75 pF	30 nA	8 kV	15 kV

3.3 Electromagnetic Compatibility (EMC) Considerations

EMC is related to the generation, transmission, and reception of electromagnetic energy that can upset the working of an electronic system. The source (emitter) produces the emission and a transfer or coupling path transfers the emission energy to a receptor, where it is processed, resulting in either desired or undesired behavior. 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.

3.3.1 Radiated Interference and Emissions

Radiated electrical energy can influence system measurements and potentially influence the operation of the CapSense processor core. The interference enters the CapSense chip at the PCB level through the sensor traces and other digital and analog inputs. While CapSense offers an intuitive and robust interface that increases product reliability by eliminating mechanical parts, it can also contribute to electromagnetic compatibility (EMC) issues in the form of radiated emissions (RE).

Use the following techniques to minimize radiated interference and emissions.

3.3.1.1 General EMI/EMC Guidelines

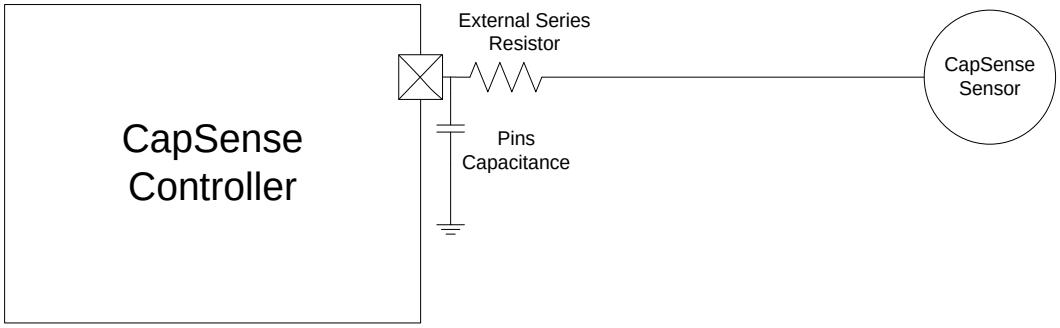
3.3.1.1.1 Ground Plane

In general, a proper ground plane on the PCB reduces both RF emissions and interference. However, solid grounds near CapSense sensors or traces connecting these sensors to the PSoC pins increase the parasitic capacitance of the sensors. It is thus recommended that you use hatched ground planes surrounding the sensor and on the bottom layer of the PCB, below the sensors, as explained in the Ground Plane section. A solid ground may be used below the device and other circuitry on the PCB, which is far from CapSense sensors and traces. A solid ground flood is not recommended within 10 mm of CapSense sensors or traces. Multiple-layer boards should be the preferred choice. If you are using a board with four layers or more, you can provide a complete layer for ground that will further help to reduce emissions as it reduces the ground bounce significantly.

3.3.1.1.2 Series Resistor

Every CapSense controller pin has some parasitic capacitance (C_P) associated with it. Adding an external resistor forms a low-pass RC filter that can dampen the RF noise amplitude coupled with the pin. This resistance also forms a low-pass filter with the parasitic capacitance of the trace connected to the pin (for example, sensor trace and sensor pad as shown in Figure 3-5) that can significantly reduce RF emissions. Thus, series resistors help in eliminating higher-order harmonics and attenuating the RF interference and emission.

Figure 3-5. RC Filter



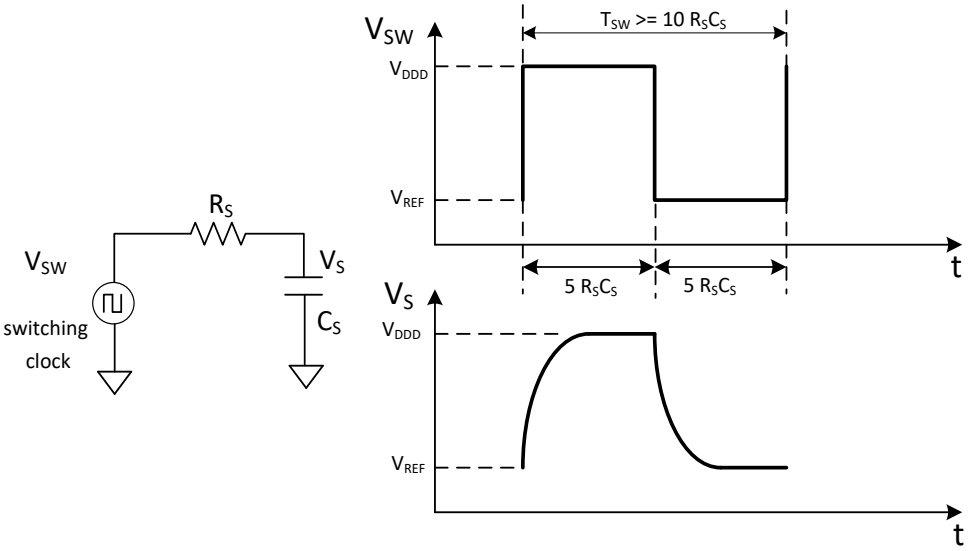
Series resistors should be placed close to the PSoC pins so that the radiated noise picked by the traces gets filtered at the input of the PSoC device. Thus, it is recommended that you place series resistors within 10 mm of the PSoC pins.

3.3.1.1.2.1 CapSense Input Lines

The sensor must be fully charged and discharged during each switching cycle for proper operation of CapSense. The charge and discharge paths of the sensor capacitor include series resistances that slow down the charging / discharging process. Figure 3-6 shows an equivalent circuit and resulting waveforms. Adding a resistance changes the time constant of the switched-capacitor circuit that converts C_P into an equivalent resistor. If the series resistance value is set high, the slower time constant of the switching circuit suppresses the emission but limits the amount of charge that it can transfer. Thus the sensors may not get charged and discharged completely. This lowers the signal level, which in turn lowers the SNR. Smaller values are better, but are less effective at blocking RF emissions and interference.

The recommended series resistance for CapSense input lines on general copper PCBs is 560 Ω. An ITO panel already provides a high resistance; one may not have too much flexibility in the value selection (range 100 Ω–1 kΩ). Series resistors are generally used in the range of 560 Ω–4.7 kΩ for EMC purpose. The actual maximum value of the series resistor that can be used varies from device to device. This depends on multiple factors such as the resistance of the GPIO used as sensor, the switching frequency used to scan sensors, and the SNR required.

Figure 3-6: Equivalent Circuit and Waveforms



R_S is the sum of the GPIO resistance and the external series resistance. C_S is the maximum capacitance of the sensor. For a given switching frequency, you must select the series-resistor value such that the sensor capacitor is charged and discharged fully. On the other hand, for a given series resistor, you must choose the switching frequency value such that the sensor capacitor is charged and discharged fully. Lowering the switching frequency lowers the SNR if you are unable to modify other CapSense parameters. Therefore, it is a tradeoff between the series resistor value and the switching frequency to achieve the desired performance.

The rule of thumb is to allow a period of $5R_SC_S$ for charging and discharging cycles. Equations for the minimum time period and maximum frequency are:

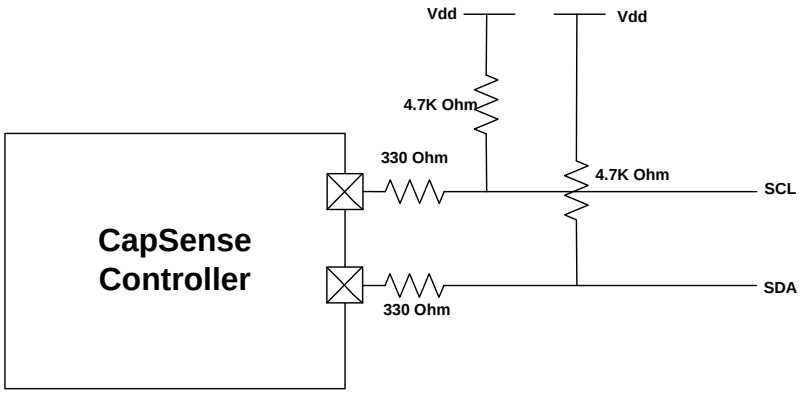
$$T_{SW}(\text{minimum}) = 10R_SC_S$$

$$F_{SW}(\text{maximum}) = \frac{1}{10R_SC_S}$$

3.3.1.1.2.2 Digital Communication Lines

Communication lines, such as I^2C and SPI, also benefit from series resistance and 330 Ω is recommended for communication lines. Communication lines have long traces that act as antennae, similar to CapSense traces. The recommended pull-up resistor value for communication lines is 4.7 kΩ. So, if more than 330 Ω is placed in series on these lines, the voltage levels (V_{IL} and V_{IH}) fall out of the specifications with the worst-case combination of supply voltages between systems and the input impedance of the receiver. 330 Ω will not affect the I^2C operation because the V_{IL} level still remains within the I^2C specification limit of 0.3 V_{DD} when the PSoC device outputs a LOW.

Figure 3-7. Series Resistors on Communication Lines



3.3.1.1.3 Trace Length

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

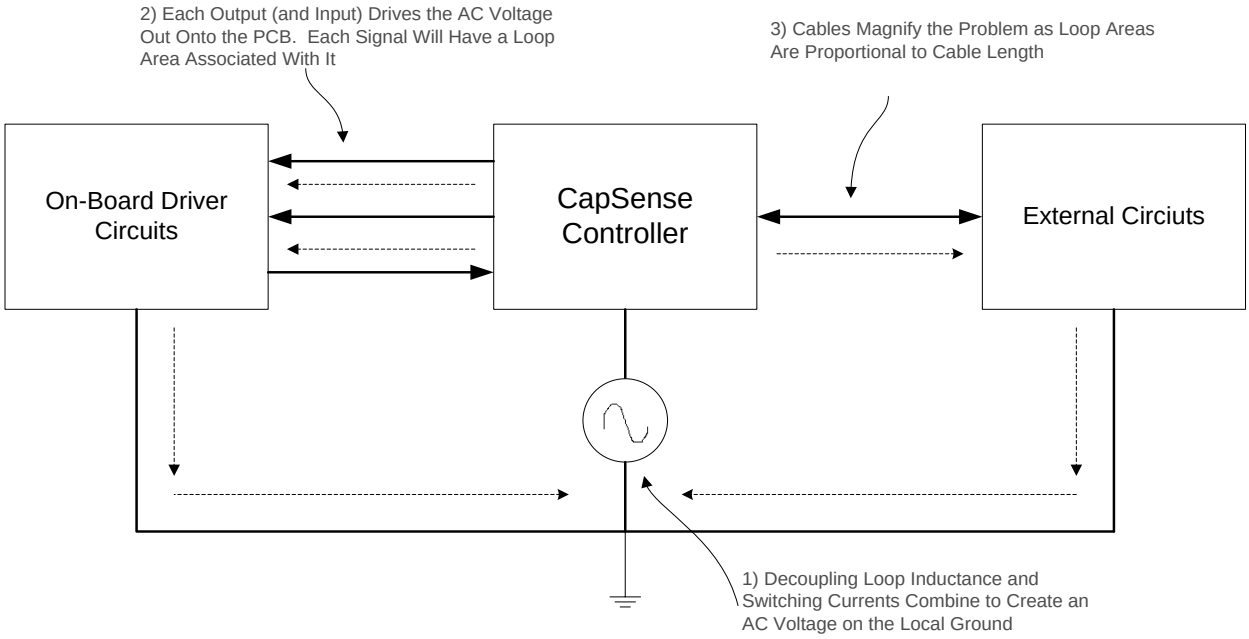
3.3.1.1.4 Current Loop Area

Another important layout consideration is to minimize the return path for current. A ground plane can lower the overall ground impedance, thus reducing the high-frequency ground bounce. Ensure good GND return paths for each sensor line. This is important as the current flows in loops. Unless there is a proper return path for high-frequency signals, the return current will flow through a longer return path forming a larger loop; this can cause signal issues due to mutual inductance. Thus, it leads to increased emissions and interference.

When a device package contains high-frequency current loops, energy can also be coupled out of the device through a magnetic field. It is possible for the magnetic flux to form a current loop in the device to link to the circuit loops outside the device. This mutual inductance can produce an unwanted voltage in the external loop. Likewise, an external magnetic flux can induce an unwanted voltage across an interior circuit loop. Magnetic field coupling can be minimized by keeping the power and signal loop areas as small as possible. Stitch all the grounds with as many vias as possible. This will reduce the overall ground impedance. High-frequency traces, such as those used for clock and oscillator circuits, should be contained by two ground lines. This will ensure that there is no coupling, which results in crosstalk. Use separate ground plane and power planes wherever possible.

Figure 3-8 shows an example of an improper grounding scheme. The layout greatly improves by reducing the loop area.

Figure 3-8. Improper Ground Scheme and Ground Loop



In Figure 3-9, two sensors are surrounded by a ground plane that is connected to a CapSense controller ground, while a third sensor is surrounded by ground. The third sensor is connected to the other ground plane through the long traces of other circuitry, which creates a large current loop. With this layout, the third sensor may be more susceptible to radiated noise and have increased emissions. These two sections of ground are in the same location on the schematic, so they can potentially be one connected area with a better layout.

Figure 3-9. Improper Current Loop Layout

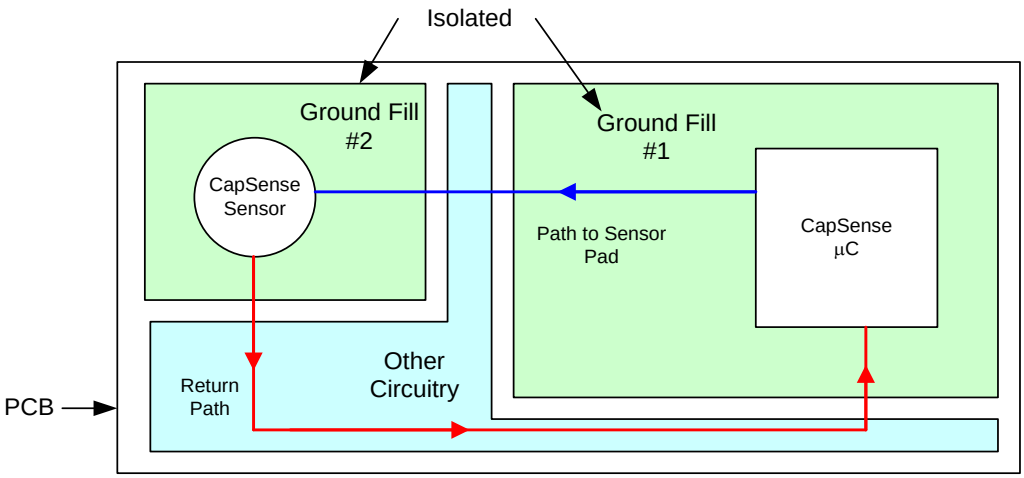
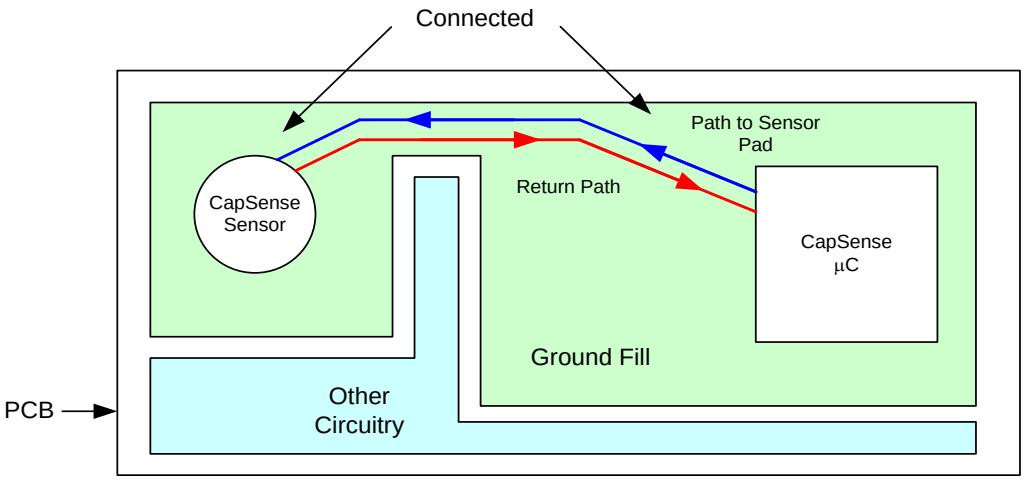


Figure 3-10 illustrates the proper layout for the previous example. The loop area is reduced by connecting the two grounded areas.

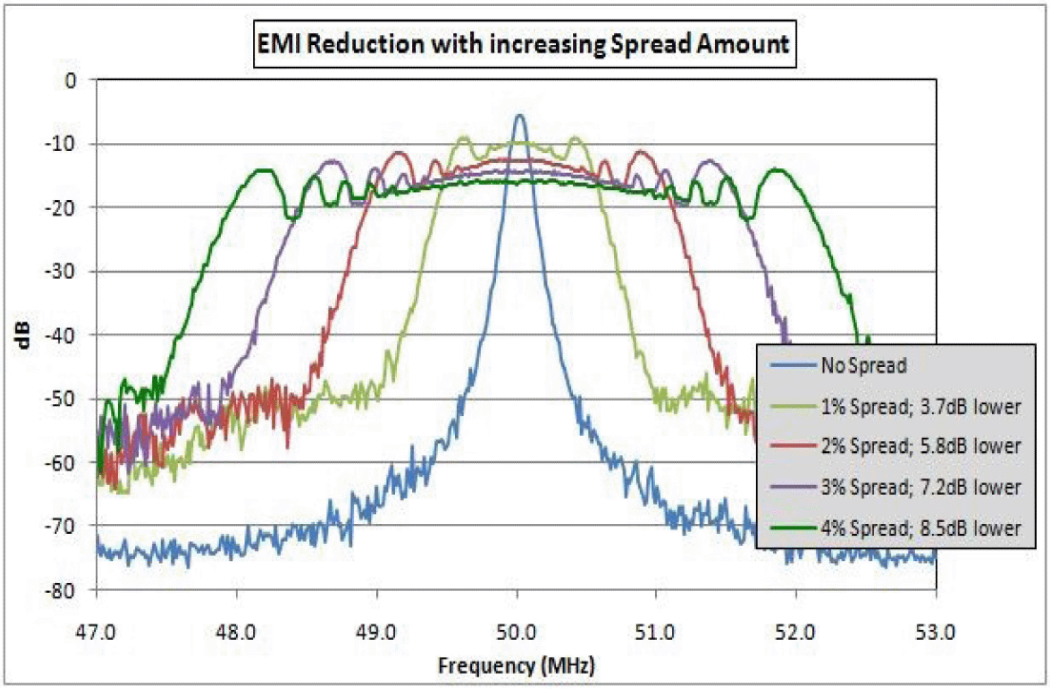
Figure 3-10. Proper Current Loop Layout



3.3.1.1.5 Frequency Hopping

Frequency hopping is a method of spreading the input or operating frequency over a narrow band of frequencies. This method helps to reduce the peaks and spread out the emissions as well as the interference over a range of frequencies as shown in Figure 3-11. The following are the methods of frequency hopping in PSoC:

Figure 3-11. Frequency Hopping



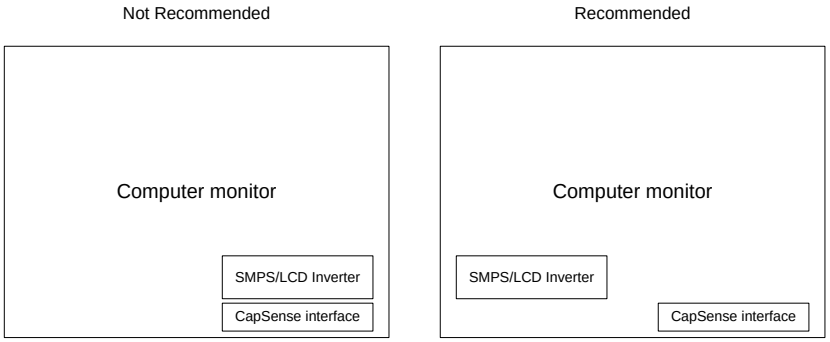
- **IMO dithering across sensor scans:** IMO dithering or trimming can be done across different sensors. For example, the IMO frequency is swept over a range from 24 MHz to 22 MHz when the base IMO frequency is 24 MHz. A sensor is scanned at one frequency always. Different sensors are scanned at different frequencies. This reduces the peaks by spreading the emissions.
- **IMO dithering within each scan:** IMO dithering can be done within each scan as well. When a sensor is being scanned, the IMO frequency is swept over a range from 24 MHz to 22 MHz. Thus, this method avoids a sensor being scanned at one frequency. This reduces the peaks by spreading the emissions. This also improves the immunity to RF interference.
- **Spread Spectrum Clock (SSC):** PSoC can also work using an external clock. Using a spread spectrum clock will help in spreading out the emissions over a wider range of frequencies, similar to IMO dithering. PSoC1 allows only port P1[4] to be used to supply the external clock. In this case, pin P1[4] drive mode must be set to HI-Z digital. This increases the immunity to RF interference and spreads the emissions.
- **Pseudo random sequencer (PRS):** A PRS is used instead of a fixed clock-source to attenuate the emitted noise on CapSense pins by reducing the amount of EMI created by a fixed frequency-source and to increase the EMI immunity from other sources and their harmonics. This increases the immunity to RF interference and spreads the emissions.

3.3.1.2 Radiated Immunity

3.3.1.2.1 RF Source Location

When systems, such as computer monitors or digital photo frames, are designed with CapSense devices, make sure you prevent noise from LCD inverters and switched-mode power supplies (SMPS) from upsetting the CapSense system. A simple technique to minimize this kind of interaction is to partition the system with noise sources from CapSense inputs, as demonstrated in Figure 3-12. Due to the practical limitations of product size, the noise source and the CapSense circuitry may only be separated by a few inches. This small separation can provide the extra margin required for good sensor performance, compared to the case with close proximity between noise source and CapSense.

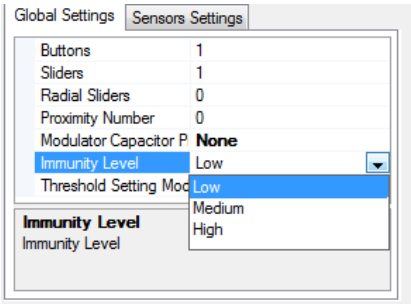
Figure 3-12. Separating Noise Sources



3.3.1.2.2 EMC Feature

CapSense User Modules and components with the EMC feature implement IMO dithering to scan each sensor. Each sensor is scanned at two or three different frequencies depending on the immunity level chosen for each sample for raw count. Use this option to improve the immunity against RF interference.

Figure 3-13. Immunity Level Selection

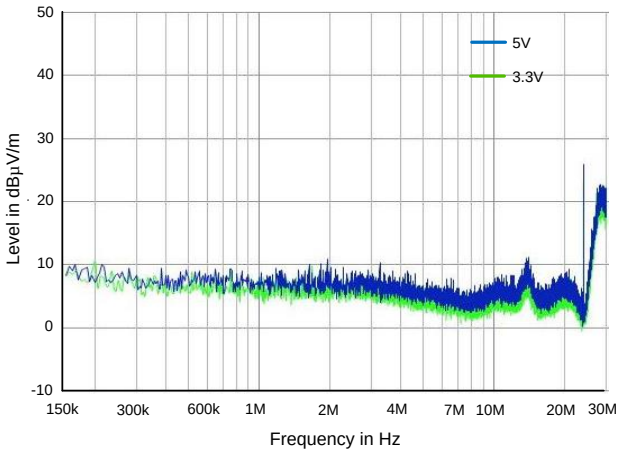


3.3.1.3 Radiated Emissions

3.3.1.3.1 Operating Voltage

For devices in which sensors switch between an operating voltage and a reference voltage, such as CY8C21x34, reducing the operating voltage helps to reduce emissions to a great extent. This is because the amplitude of the switching signal at any pin is dependent on the device's operating voltage and the emissions are directly proportional to voltage levels at which the switching happens. Figure 3-14 shows the impact of operating voltage on radiated emissions.

Figure 3-14. Impact of Operating Voltage on Emissions



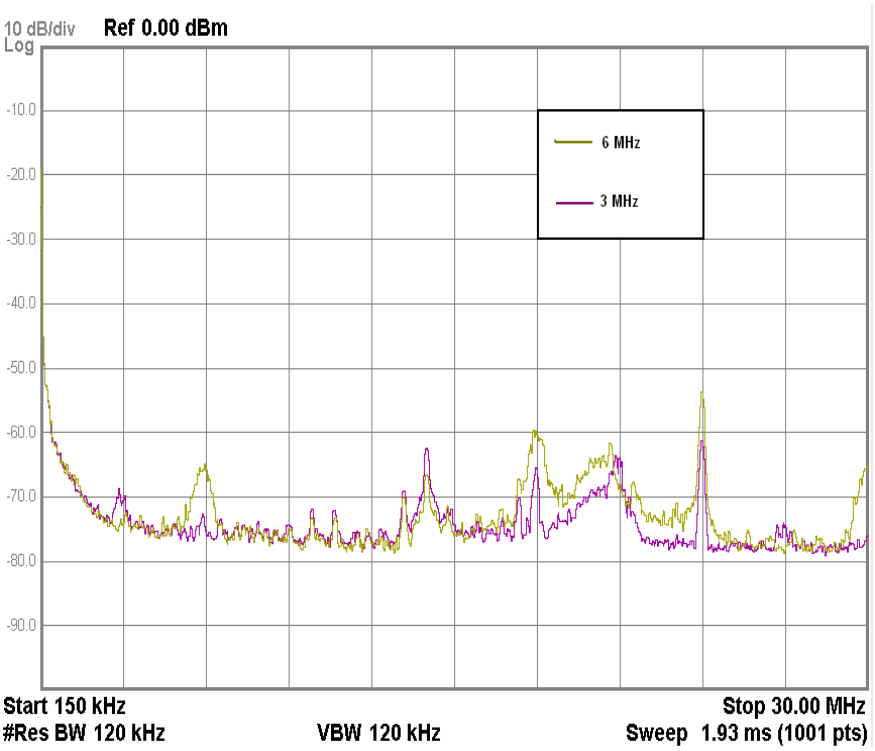
3.3.1.3.2 System Oscillator Frequency (IMO)

Lowering the system clock will significantly lower radiated emissions. However, lowering the system clock impacts the performance of your system because a low IMO can take more time to scan the sensors and perform the processing. Therefore, lower the system frequency depending on your application.

3.3.1.3.3 Sensor Switching Frequency

The CapSense sensing methods use a switched-capacitor front end to interact with sensors. Selecting a low frequency for the switched-capacitor clock helps you to reduce the radiated emissions from the CapSense sensors. Figure 3-15 shows the impact of the sensor-switching frequency.

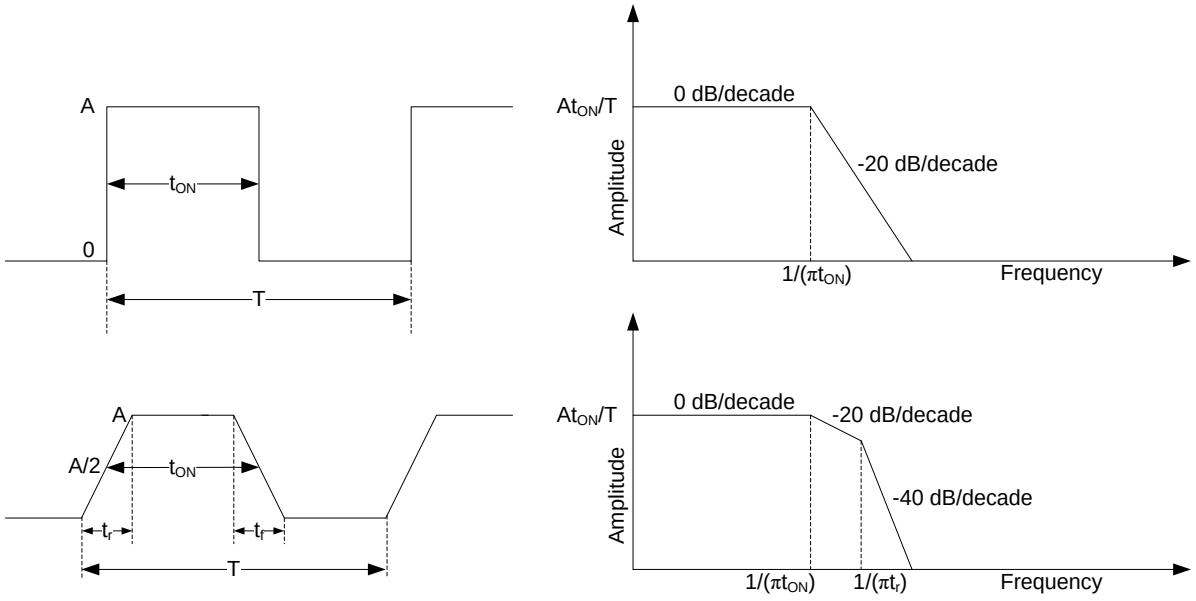
Figure 3-15. Impact of Switching Frequency



3.3.1.3.4 Slew Rate Control

Figure 3-16 shows the impact of rise/fall time of a square wave on radiated emissions. Note that slowing the transitions introduces the cutoff point and damps the radiated-energy level. Internal clock signals of the CapSense controller are slew-controlled to reduce the radiated emission.

Figure 3-16. Impact of Slew Rate on Emissions



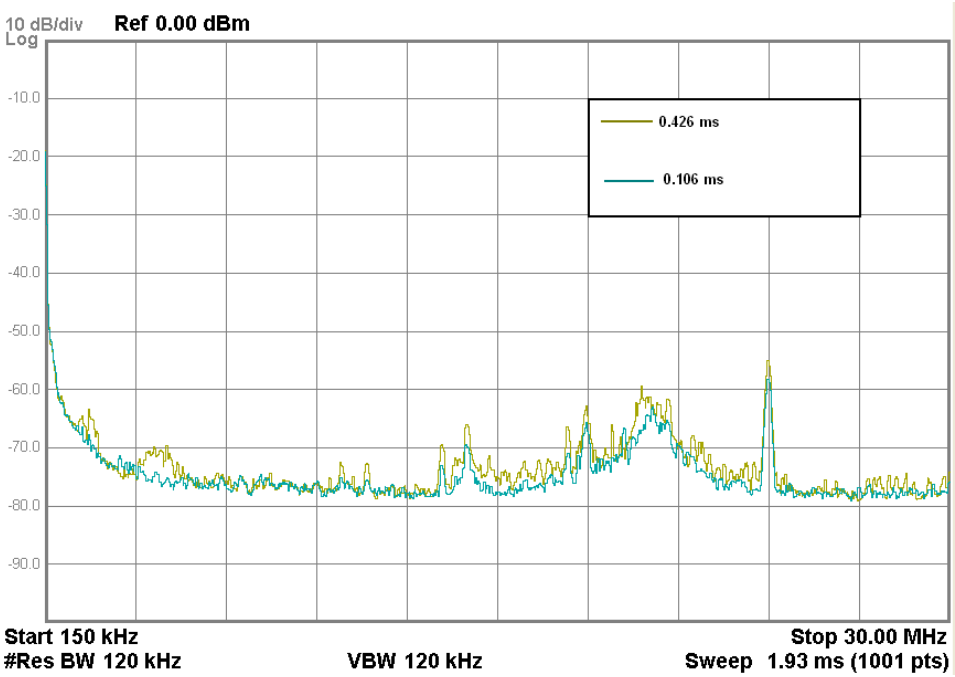
3.3.1.3.5 Sensor Scan time

The scan time of sensors impacts radiated emissions. Figure 3-17 shows the impact of the sensor-scan time on radiated emissions. An increase in the sensor-scan time results in increased emissions. Table 3-5 shows the parameter settings and associated sensor scan times.

Table 3-5. Sensor Scan time

Parameter	Value	
Scan resolution	8 bits	10 bits
Individual sensor scan time	0.021 ms	0.085 ms
Total scan time for five buttons	0.105 ms	0.425 ms

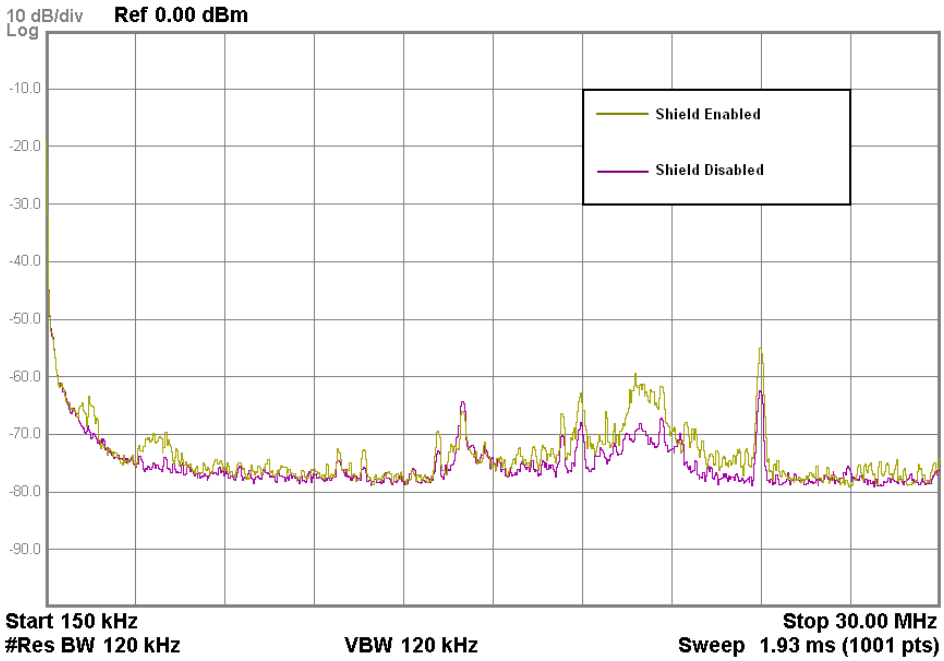
Figure 3-17. Effect of Scan Time



3.3.1.3.6 Shield Signal

The shield signal is driven on the hatch fill to reduce the parasitic capacitance of sensors for liquid tolerance and for proximity sensing. See the Shield Electrode and Guard Sensor section for more details. The shield signal is a replica of the sensor signal. The shield signal further increases radiated emissions as it is a high-frequency switching signal and is driven on a spread-out hatch fill. Figure 3-18 shows the emissions with and without the driven-shield signal:

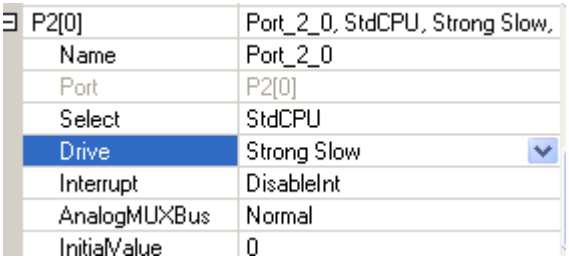
Figure 3-18. Impact of Shield on Emissions



You can reduce emissions by the following techniques:

1. Reduce the size of the shield hatch fill to have a maximum of 10 mm from the sensors. See the Shield Electrode and Guard Sensor section for more details.
2. Drive the shield signal only when required. The shield must be driven only when sensors are scanned, and only when the sensors that need shield protection are scanned.
3. Limit the placement of the shield to selected sensors only. Do not spread the shield around sensors that do not need shield protection.
4. Slow the edges of the shield waveform by one of the following means:
   a. Add a capacitor filter between the shield electrode port pin and ground
   b. In most of the PSoC 1 CapSense devices, the slew rate of the shield signal can also be controlled by changing the drive mode of the shield pin from Strong to Strong Slow as shown in Figure 3-19.

Figure 3-19. Drive Mode Selection for Shield



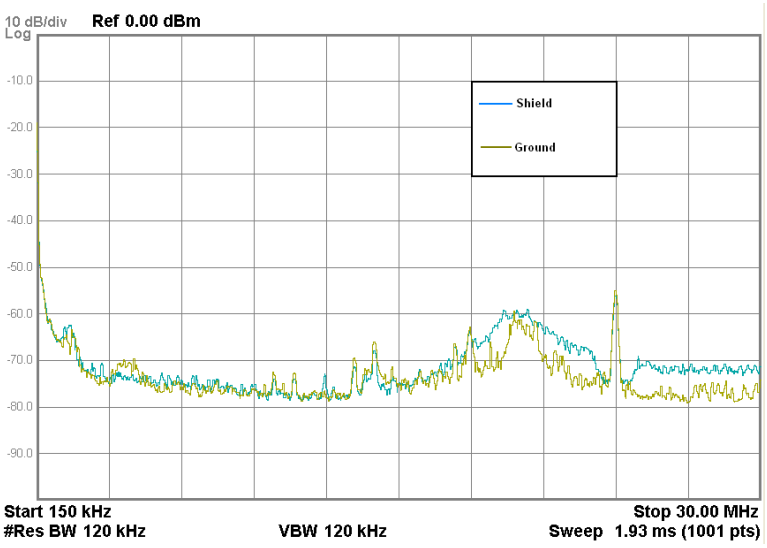
Add a passive low-pass filter (LPF) to the shield signal by putting a series resistor to the shield pin. An inherent LPF is formed by the resistance of the electrode material and the parasitic capacitance of the electrode. Therefore, adding a series resistor increases the RC of the filter and helps to improve emissions to a great extent because the RC filter formed will eliminate higher-order harmonics of these frequencies.

Note: Filtering the shield too much can cause a phase difference between the driven-shield signal and the sensor-switching signal. Ensure that the shield electrode gets charged and discharged completely when you add filters. See the CapSense Input Lines section for charge and discharge waveforms, and to see how to choose the right value for series resistors.

3.3.1.3.7 Inactive Sensor Termination

Connect inactive sensors to ground to reduce emissions instead of the shield unless it is a stringent requirement to connect them to the shield. Figure 3-20 shows the impact of different inactive-sensor terminations on radiated emissions.

Figure 3-20. Effect of Inactive Sensor Termination



For CapSense applications, it is very important to have a clean power supply for CapSense devices to reduce problems related to radiated interference and emissions. Guidelines to filter the noise at the power supply of CapSense devices are given in the following section. It is recommended that you incorporate these guidelines to handle any EMC/EMI issues.

3.3.2 Conducted Immunity and Emissions

The noise current generated by high-frequency switching circuits entering the system through the power and communication lines is called conducted noise.

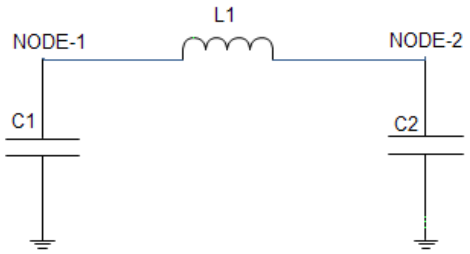
3.3.2.1 Board-Level Solutions

Proper use of decoupling capacitors as recommended by the datasheet can limit the problem with conducted emissions. For detailed information on general decoupling capacitors, see the section 3.8.13 . For unregulated power supplies, use a large electrolytic capacitor (typically 10 µF –100 µF), no more than one inch away from the chip. This capacitor acts as a reservoir of charge to supply the instantaneous charge requirements of the circuits locally so that the charge need not come through the inductance of the power trace.

For further protection, a passive filter can be used. Passive filter effectively limits not just the conducted noise emitted but also the noise entering the system. Thus, it improves the conducted noise immunity of the system.

A pi-filter is a simple bidirectional low-pass filter. The two main types of pi-filters are the series inductor and the series resistor. The series inductor pi-filter has two shunt capacitors and one series inductor configured similar to the Greek letter π, as shown in Figure 3-21. The noise is filtered by all three elements (L1, C1, and C2) in both directions. The bidirectional nature of the filter is important. Not only does it prevent the supply noise from affecting sensitive parts, it can also prevent the switching noise of the part from coupling back to the power planes.

Figure 3-21. Series Inductor Pi-Filter



The values of the components are selected based on the frequency that needs to be attenuated.

3.3.2.2 Power Supply Solutions

The following guidelines help you to prevent conducted noise from entering your CapSense design:

- Provide GND and V_{DD} planes that reduce current loops.
- If the CapSense controller PCB is connected to the power supply by a cable, minimize the cable length and consider using a shielded cable.
- To reduce high-frequency noise, place a ferrite bead around the power supply or communication lines.
  - ☐ Localizes the noise in the system.
  - ☐ Keeps external high frequency noise away from the IC.
  - ☐ Keeps internally generated noise from propagating to the rest of the system

For more information on design considerations for EMC, see the following documents:

- Top 10 EMC Design Considerations
- AN2155 - PSoC EMI Design Considerations
- AN80994 - Design Considerations for EFT Immunity

3.4 Software Filtering

Software filters are one of the techniques of dealing with high levels of system noise. Table 3-6 lists the types of filters that are useful for CapSense.

Table 3-6. CapSense Filter Types

Type	Description	Application
Average	Finite impulse response filter (no feedback) with equally weighted coefficients	Periodic noise from power supplies
IIR	Infinite impulse response filter (feedback) with a step response similar to an RC filter	High frequency white noise (1/f noise)
Median	Nonlinear filter that computes median input value from a buffer of size N	Noise spikes from motors and switching power supplies
Jitter	Nonlinear filter that limits current input based on previous input	Noise from thick overlay (SNR < 5:1), especially useful for slider centroid data
Event-Based	Nonlinear filter that causes a predefined response to a pattern observed in the sensor data	Commonly used to block generation or posting of nonexistent events
Rule-Based	Nonlinear filter that causes a predefined response to a pattern observed in the sensor data	Commonly used during normal operation of the touch surface to respond to special scenarios such as accidental multi-button selection

3.4.1 Average Filter

An average filter is a finite impulse response (FIR) filter with equal-weighted coefficients. The average filters work well with periodic noise that is attenuated by spacing the samples out over one noise cycle. Sample spacing is not critical. For example, power line noise can be anywhere from 50 Hz to 60 Hz. Without adjusting the sampling rate, the average filter works well for 50-Hz and 60-Hz noise. Figure 3-22 shows a sample rate that is synchronized with a simple periodic waveform. There is no feedback path in this filter.

Figure 3-22. Synchronized Sample Rate

The general equation for an average filter is:

$$y[i] = \frac{1}{N}(x[i] + x[i-1] + \cdots + x[i-N+1]) \quad \text{Equation 11}$$

Figure 3-23 and Figure 3-24 illustrate the results of using an average filter on real CapSense data using the 16-sample filter equation:

$$y[i] = \frac{1}{16}(x[i] + x[i-1] + \cdots + x[i-15]) \quad \text{Equation 12}$$

Figure 3-23. Average Filter Noise (16 Samples)

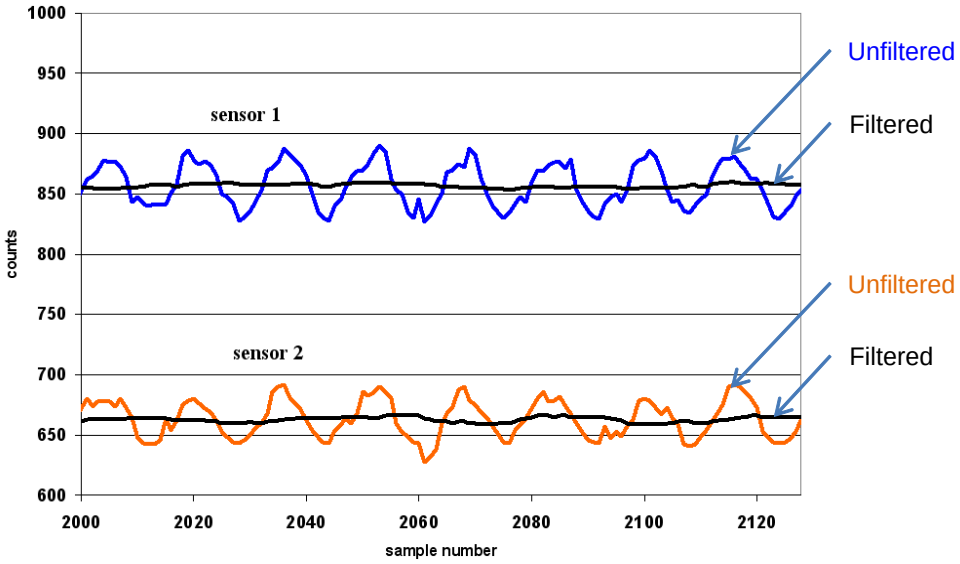
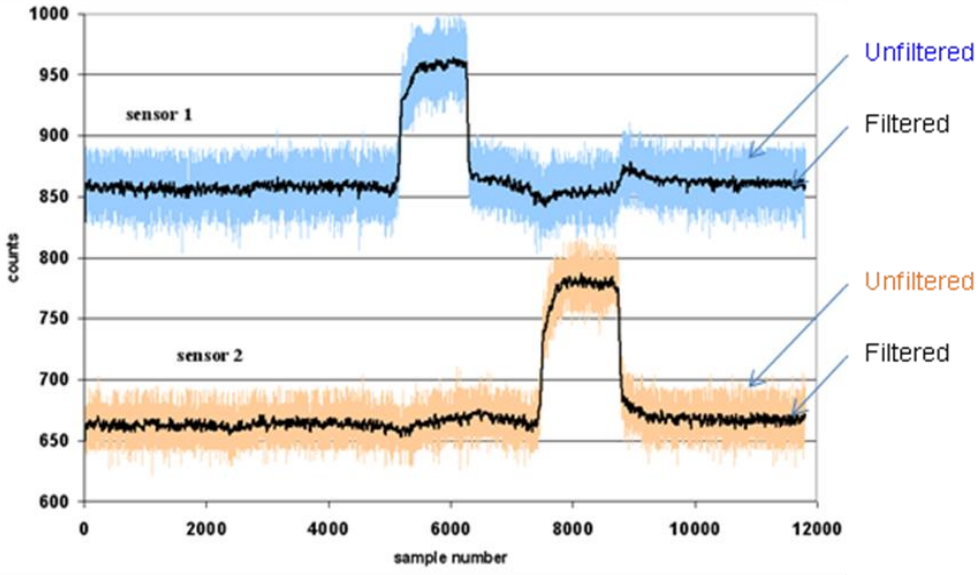


Figure 3-24. Average Filter Finger Touch (16 Samples)

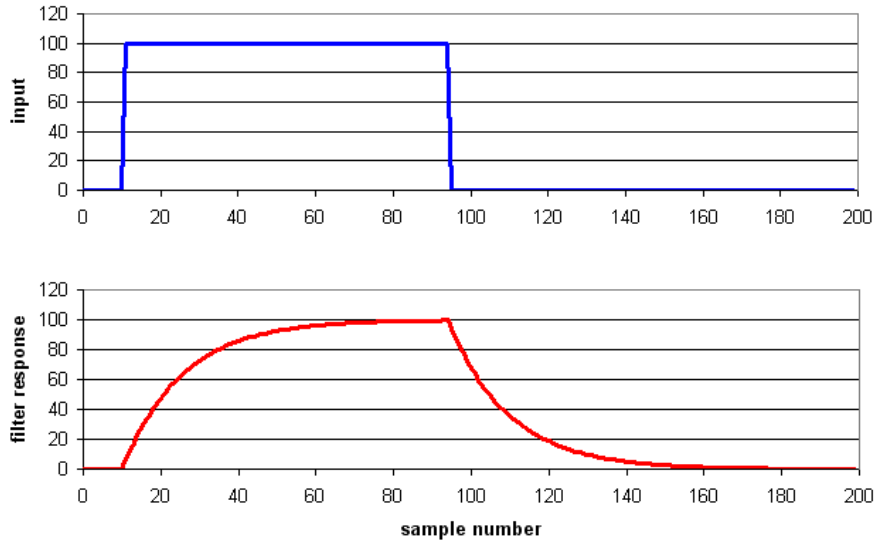


The previous examples are representative of power supply noise. The filter works well in this example because the period of the noise is close to the length of the filter (N = 16). For more information about how to implement an average filter, see the code example CSA Software Filters with EzI2Cs Slave on CY8C20XX6.

3.4.2 IIR Filter

Infinite impulse response filters (IIR) produce a step response similar to RC filters. IIR filters attenuate high-frequency noise components and pass lower frequency signals, such as finger touch-response waveforms.

Figure 3-25. IIR Filter Step Response



The general equation for a first-order IIR filter is:

$$y[i] = \frac{1}{k}\left(x[i] + \left((k-1) \times y[i-1]\right)\right)$$ Equation 13

Figure 3-26 and Figure 3-27 illustrate the results of a first-order IIR filter on real CapSense data using the filter equation with k = 16:

$$y[i] = \frac{1}{16}\left(x[i] + (15 \times y[i-1])\right)$$ Equation 14

Figure 3-26. IIR Filter Noise

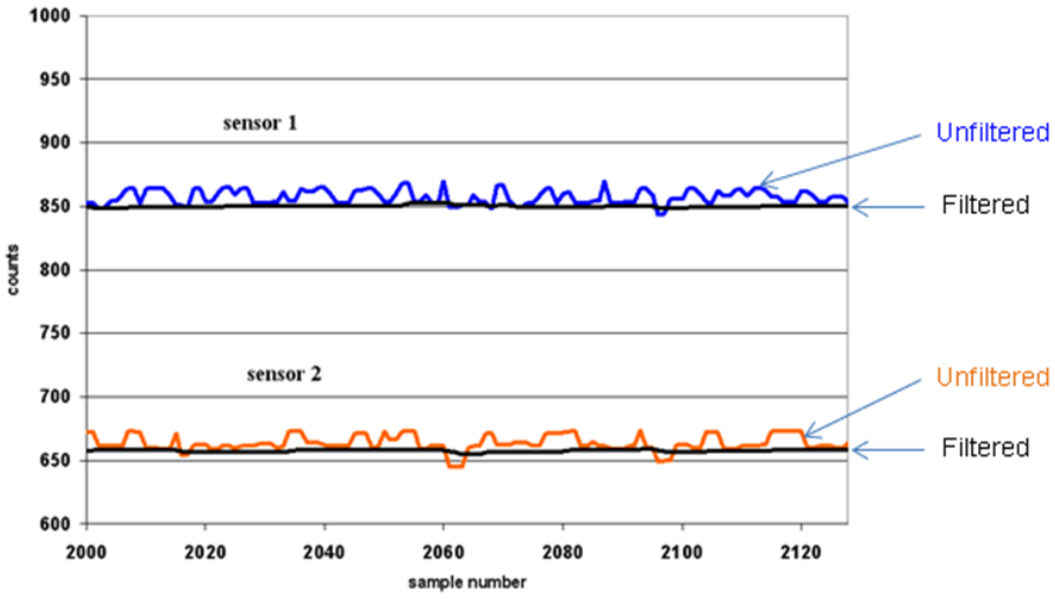
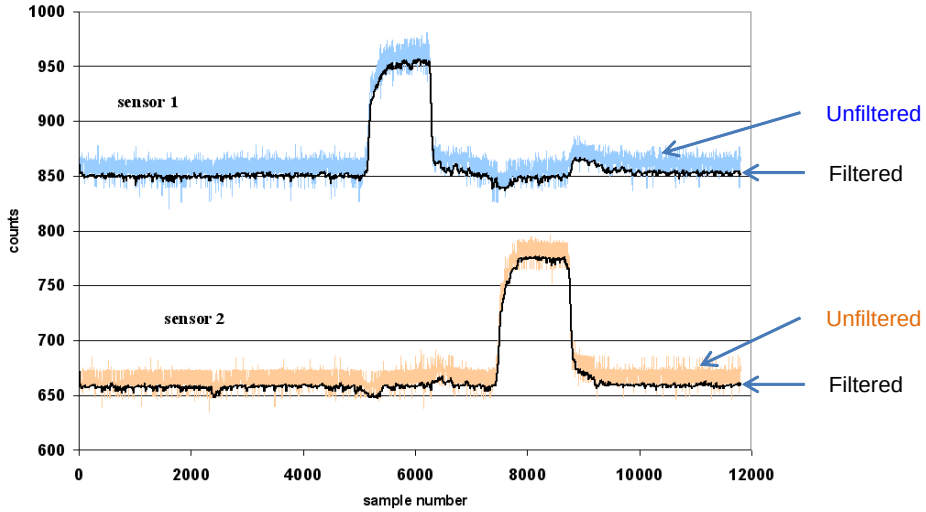


Figure 3-27. IIR Filter Finger Touch



For more information about how to implement an IIR filter, see the code example CSA Software Filters with EzI2Cs Slave on CY8C20XX6.

3.4.3 Median Filter

Median filters eliminate noise spikes most commonly associated with motors and switching power supplies. In a median filter, a buffer of size N stores the N most recent samples of the input. The median is then computed using a two-step process. First, the buffer values are sorted from smallest to largest; then, the middle value is selected from the ordered list. The buffer is scanned for the median with each update of the buffer. This is a nonlinear filter. The general equation for a median filter is:

$$y[i] = median(x[i],\ x[i-1], \ldots, x[i-N+1]) \qquad \text{Equation 15}$$

Figure 3-28 and Figure 3-29 show the results of a median filter on real CapSense data using the general filter equation with N = 16.

$$y[i] = median(x[i],\ x[i-1], \ldots, x[i-15]) \qquad \text{Equation 16}$$

Figure 3-28. Median Filter Noise Spike

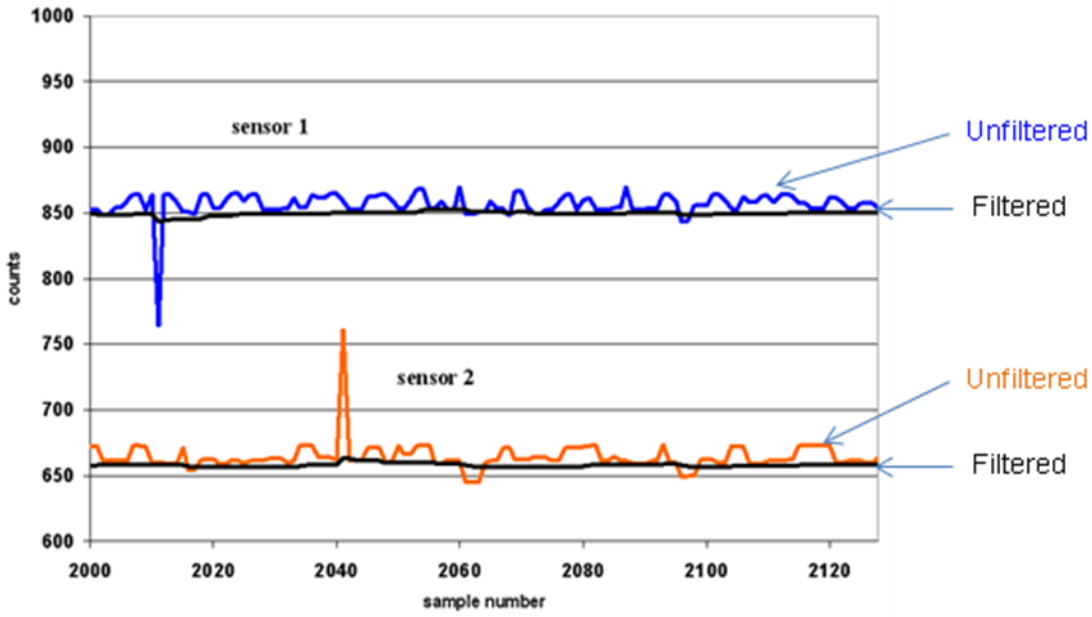
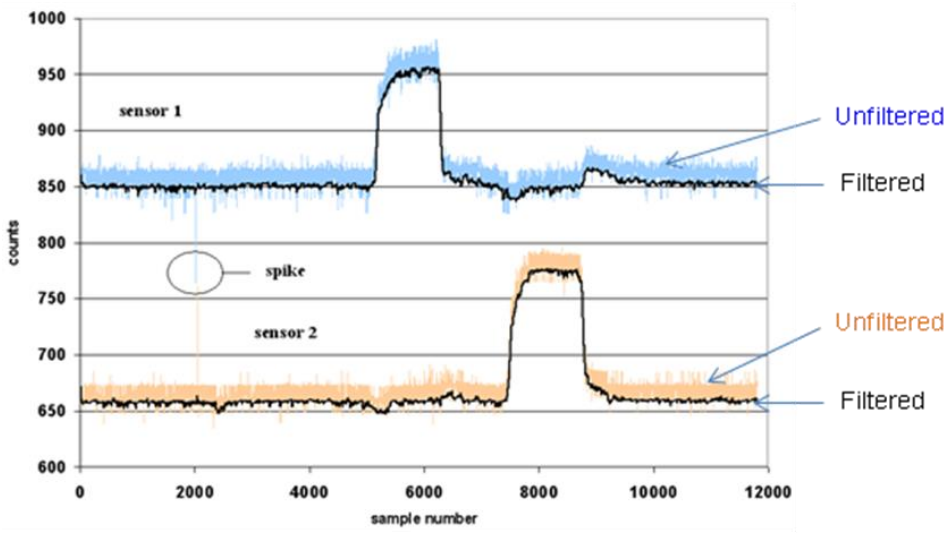


Figure 3-29. Median Filter (16-Sample) Finger Touch



For more information about how to implement a median filter, see the code example CSA Software Filters with EzI2Cs Slave on CY8C20XX6.

3.4.4 Jitter Filter

3.4.4.1 Jitter Filter for Noisy Slider Data

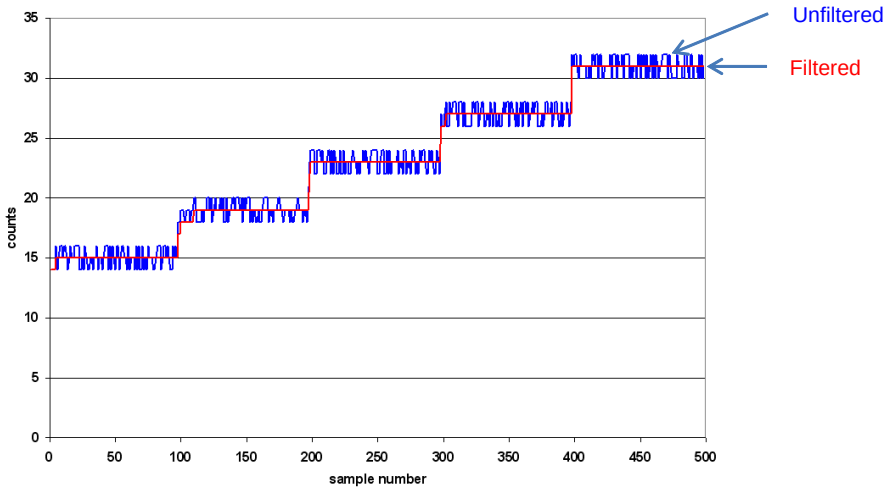
The centroid function is used to estimate finger position on a slider. When the signal level is low, usually because of thick overlay on the slider, the estimate of finger position will appear to shake and jitter even when the finger is held at a fixed position. This jitter noise can be removed using a jitter filter. To do this, the previous input is stored in a buffer. The current input is compared to the previous output. If the difference is greater than ±1, the output is changed by ±1 (matching sign), as shown in Equation 21. This is a nonlinear filter.

$$y[i] = x[i] - 1, \quad if\ x[i] > y[i-1] + 1$$
$$y[i] = x[i] + 1, \quad if\ x[i] < y[i-1] - 1$$
$$y[i] = y[i-1], \quad otherwise$$

Equation 17

Figure 3-30 shows the results of applying a jitter filter applied to noisy centroid data.

Figure 3-30. Jitter Filter Applied to Noisy Centroid Data



3.4.4.2 Jitter Filter for Raw Counts

Although the jitter filter is intended for use with noisy slider data, it is also used with noisy buttons. If the change in the current input exceeds a set threshold level, the output is reverted to the previous input plus or minus the threshold amount. The output does not change if the current input changes by less than the threshold amount. The general equation for a jitter filter applied to buttons is:

$$y[i] = x[i] - threshold, \quad if\ x[i] > y[i-1] + threshold$$
$$y[i] = x[i] + threshold, \quad if\ x[i] < y[i-1] - threshold$$
$$y[i] = y[i-1], \quad otherwise$$

Equation 18

Figure 3-31 and Figure 3-32 show the result of using a jitter filter on real button data with a large component of periodic noise.

Figure 3-31. Jitter Filter for Button Noise

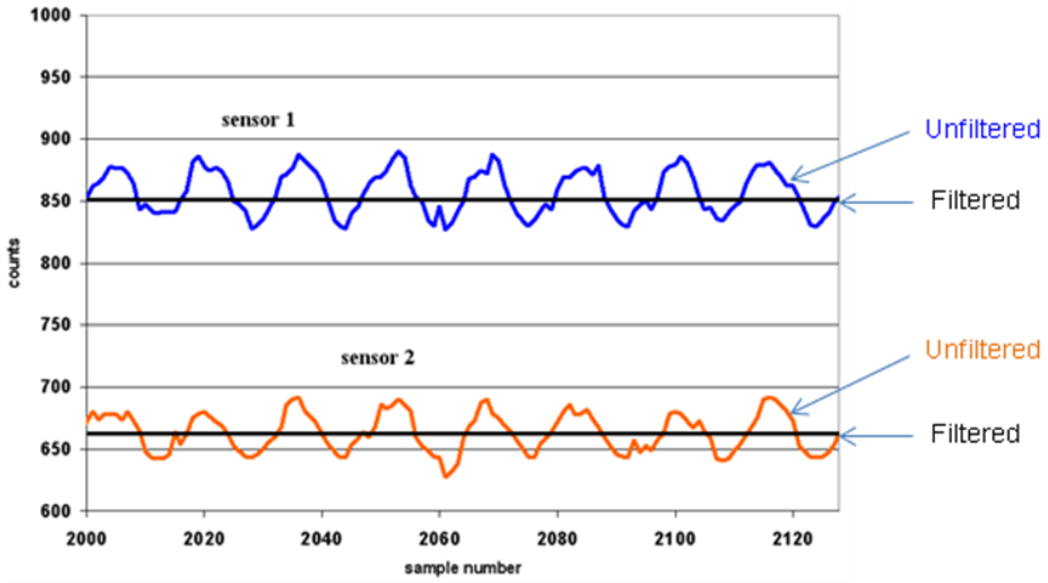
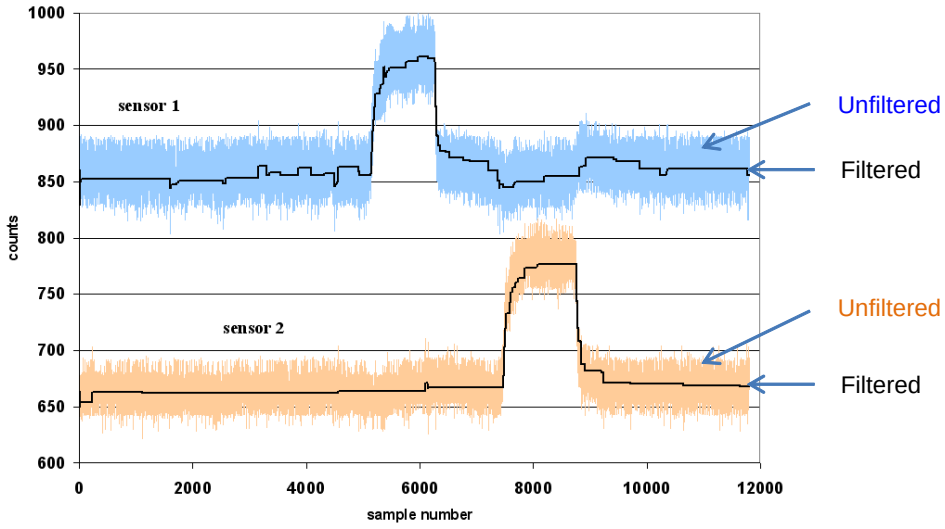


Figure 3-32. Jitter Filter for Button Finger Touch



For more information about how to implement a jitter filter, see the code example CSA Software Filters with EzI2Cs Slave on CY8C20XX6.

3.4.5 Event-Based Filters

Event-based filters involve a special filtering method, where a pattern observed in the sensor data causes a predefined response in the CapSense system. The pattern in the data is triggered by an event, such as a handheld product placed into a pocket, or the power supply voltage (V_{DD}) dropping suddenly in a camera phone when the camera flash circuit is being charged. Following are the common responses used with an event-based filter:

- To block the CapSense data transmission until the pattern returns to normal
- To reset the level of the baseline reference defined in SNR
- To drop or ignore the sample of data when the event occurred

An example for an event-based filter is dropping the sample or resampling of the data when the interrupt occurred. I^2C is one of the common communication protocols used in CapSense applications. Because I^2C interrupts are asynchronous in nature, they may occur when the sensors are being scanned. Interrupts that occur during the scan may increase the noise and, therefore, decrease the SNR. In such cases, you may implement an event-based filter, such as ignoring the raw count sample corresponding to that scan when interrupts occurred and rescanning.

3.4.6 Rule-Based Filters

Rule-based filters are another special filtering method, where a pattern observed in the sensor data causes a rule-based response in the CapSense system. Unlike the event-based filter, the rule-based filter acts on patterns in the sensor data that are encountered during normal operation of the touch surface. The rule-based filter takes into account special scenarios on how sensors are used.

For example, with a set of radio channel selection buttons, two buttons can be touched accidentally, but only one should be selected. The rule-based filter sorts out these kind of situations in a predefined way. Another example is having a virtual sensor in CapSense applications. The virtual sensor is not expected to be triggered during normal operation of the sensors, but it may occur in unexpected scenarios such as presence of water (for example, the guard sensor) or when the system is subjected to RF noise. Therefore, when the virtual sensor is triggered, all the actual sensors are turned off so that there is no unintended triggering of the actual sensors.

3.5 Power Consumption

Minimizing power consumption is an important design goal. For many CapSense systems, extending battery life is critical to the success of the product. In systems that do not use batteries, power consumption still plays a role in optimizing power supply designs to reduce costs and PCB area.

3.5.1 Active and Sleep Current

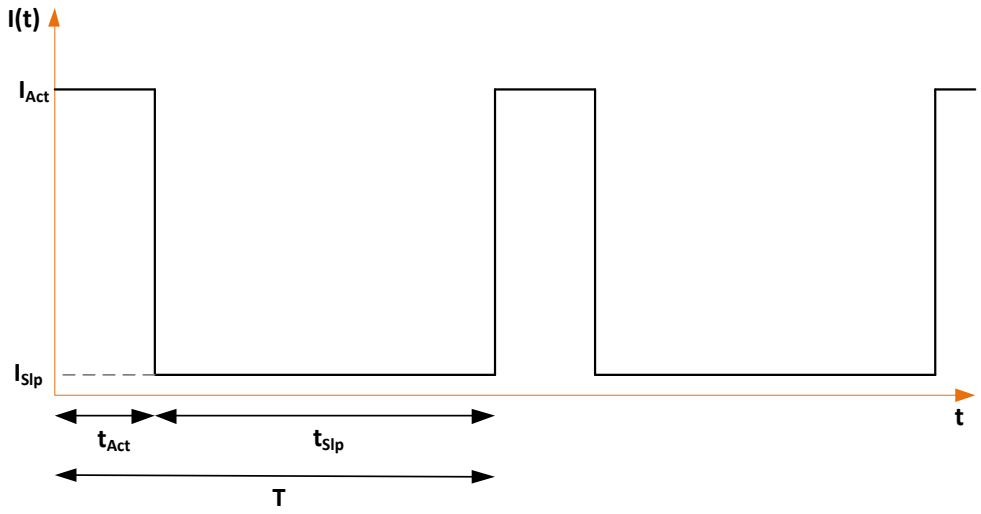
Active current is the current consumed by the device when all the selected analog and digital blocks are enabled and the CPU is running. In typical applications, the CapSense controller does not need to be in the Active state all the time.

The device can be put into the Sleep state to stop the CPU and the major blocks of the device. Current consumed by the device in Sleep state is called sleep current. Sleep current is much lower than the active current.

3.5.2 Average Current

In typical applications, sleep state can be invoked periodically to reduce power consumption. This means that during a preset time period, the CapSense controller wakes up from sleep state, performs all necessary operations in the active state (scan all sensors, update all baselines, check if any sensor is in the TOUCH state, and so on), and then returns to sleep state. The resulting instantaneous current graph is shown in Figure 3-33.

Figure 3-33. Instantaneous Current



Where:

I(t) = Instantaneous current

I_{Act} = Active current

I_{Slp} = Sleep current

t_{Act} = Active time

t_{Slp} = Sleep time

T = Time period of a cycle

The average current consumed by the device over a long period can be calculated by using the following equation.

$$I_{AVE} = \frac{(I_{Act} \times t_{Act}) + (I_{Slp} \times t_{Slp})}{T}$$ Equation 19

The average power consumed by the device can be calculated as follows:

$$P_{AVE} = V_{DD} \times I_{AVE}$$ Equation 20

3.5.3 Response Time Versus Power Consumption

As illustrated in Equation 24, the average power consumption can be reduced by decreasing I_{AVE} or V_{DD}. I_{AVE} may be decreased by increasing sleep time. Increasing sleep time to a very high value leads to poor response time of the CapSense button. Because of this tradeoff between response time and power consumption, the application developer must carefully select the sleep time based on system requirements.

In any application, if both power consumption and response time are important parameters to be considered, then, an optimized method can be used that incorporates both continuous-scan and sleep-scan modes. In this method, the device spends most of its time in sleep-scan mode where it scans the sensors and goes to sleep periodically as explained in the previous section and thereby consuming less power. When you touch a sensor to operate the system, the device jumps to continuous-scan mode where the sensors are scanned continuously without invoking sleep and thereby giving very good response time. The device remains in continuous-scan mode for a specified time-out period. If you do not operate any sensor within this time-out period, the device returns to the sleep-scan mode.

3.6 Proximity Sensing Design

This section presents the steps involved in implementing a proximity sensor and the various factors that affect the proximity distance. For the detailed design guidelines, see AN92239 – Proximity Sensing with CapSense.

3.6.1 Implementing Proximity Sensing with CapSense

Figure 3-34 shows the steps for designing a proximity-sensing system based on CapSense.

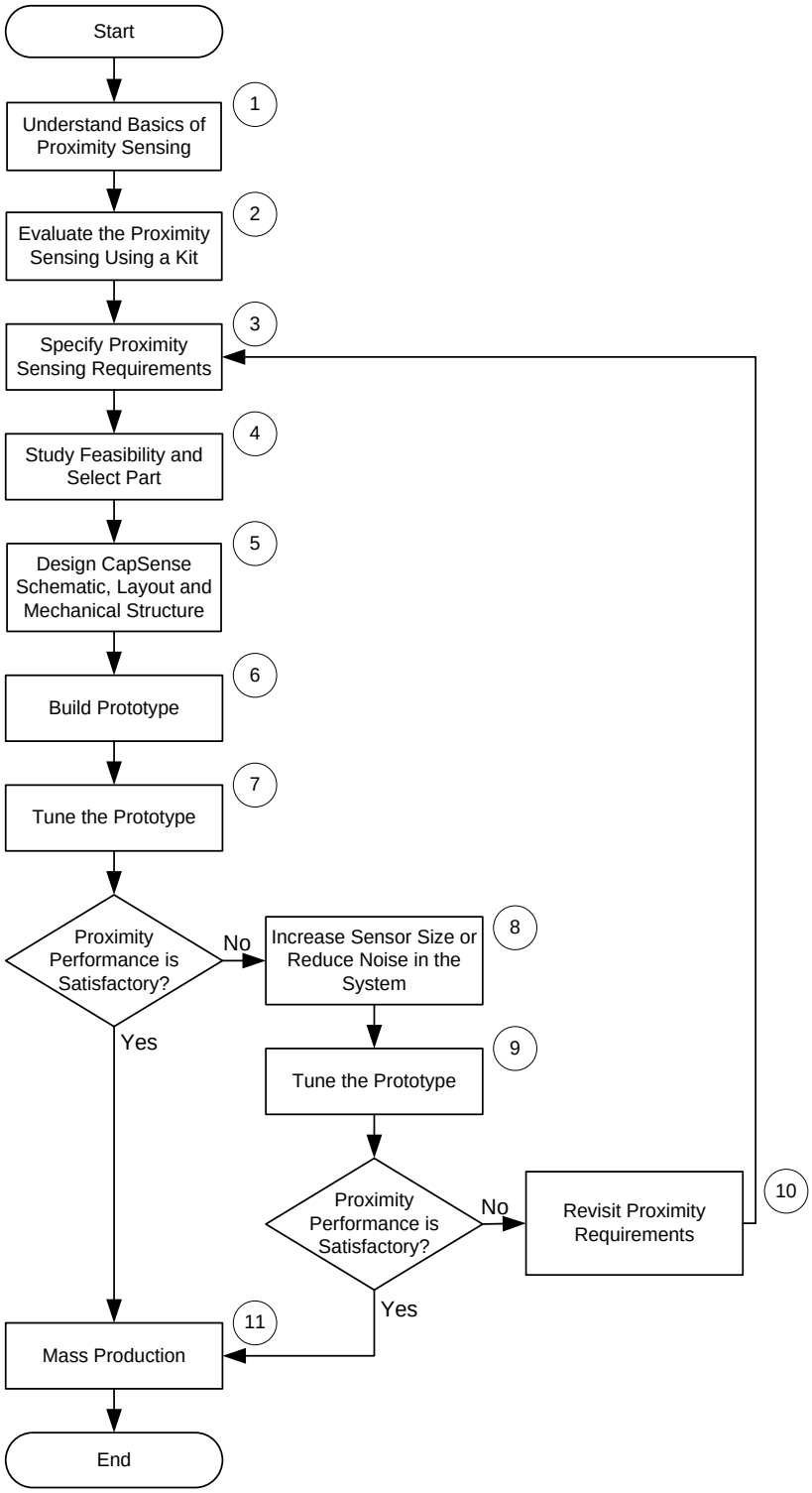
1. **Understand proximity sensing:** The Proximity Sensing section in this design guide explains how proximity sensing, based on CapSense, works and describes the parameters that affect proximity-sensing distance.
2. **Evaluate how proximity sensing works:** Use Cypress's CY8CKIT-024 – CapSense Proximity Shield.
3. **Specify the proximity-sensing requirements:** After evaluating the proximity sensor performance, specify the proximity-sensing requirements such as the required proximity-sensing distance, area available on the PCB for sensor construction, system power consumption requirements, and EMI/ESD performance. These requirements help you to select the right CapSense device and design the sensor layout.
4. **Select the right CapSense device:** After the requirements are finalized, see the CapSense Selector Guide chapter to select the right CapSense device based on the required proximity-sensing distance.
5. **Design the schematic and layout:** After selecting the CapSense device, design the schematic and layout. Follow the guidelines mentioned in the Pin Assignments section to design the schematic. You should also see the device datasheet and device-specific CapSense Design Guides for details on the schematic design. For proximity-sensor layout guidelines such as proximity-sensor type and size, see the Proximity Sensor Design and Factors Affecting Proximity Distance sections.
6. **Build the prototype:** After the schematic and layout design is completed, build the prototype of the design to check if the design meets the performance requirements.
7. **Tune:** Tune the prototype board to achieve the required performance. See the AN92239 – Proximity Sensing with CapSense and device-specific CapSense Design Guides.

   After tuning the sensor, check if the proximity sensor performance meets the requirements. If the requirements are met, proceed to step 11; otherwise continue with Step 8.
8. **Redesign if necessary:** If the proximity sensor does not provide the required performance after you have set the optimum parameters, increase the sensor size or reduce the noise in the system by shielding the sensor from noise sources and continue with Step 9.
9. **Retune:** After redesigning the proximity sensor, retune the sensor and check if the sensor performance meets the requirements. If the requirements are met, proceed to Step 11; otherwise continue with Step 10.
10. **Revisit the design or requirements:** If the proximity sensor does not meet the required performance even after you have changed the sensor dimensions to the maximum possible value and tuned it with the optimum parameters, revisit the requirements.

    If you are not able to achieve the required proximity-sensing distance, select a device that has a better proximity performance than the current device.

    If you are not able to achieve the required proximity-sensing distance even with the best device, you need to change the proximity sensor requirements, such as the area available for the sensor or the required proximity-sensing distance, and repeat the procedure from Step 1.
11. **Proceed to mass production:** If the proximity sensor meets the required performance, you can proceed to mass production.

Figure 3-34. CapSense-based Proximity Sensing Design

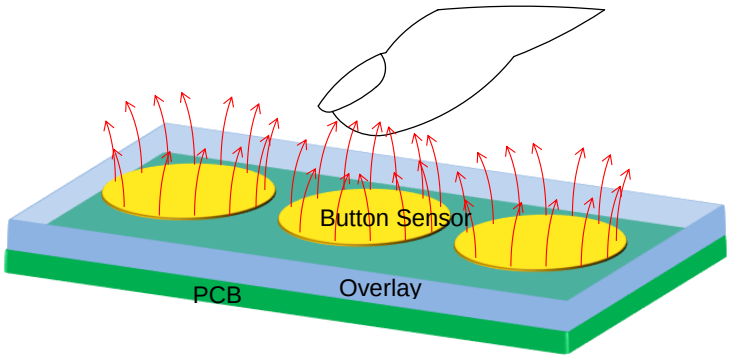


3.6.2 Proximity Sensor Design

A capacitive proximity sensor can be constructed using one of the following methods:

Button: A button sensor, when tuned for high sensitivity, can be used as a proximity sensor, as shown in Figure 3-35 The proximity-sensing distance is directly proportional to the sensor area. Because the diameter of a button sensor typically ranges from 5 mm to 15 mm, the proximity-sensing distance achieved with a button sensor is very less when compared to other sensor implementation methods.

Figure 3-35. CapSense-based Proximity Sensing with Button Sensor

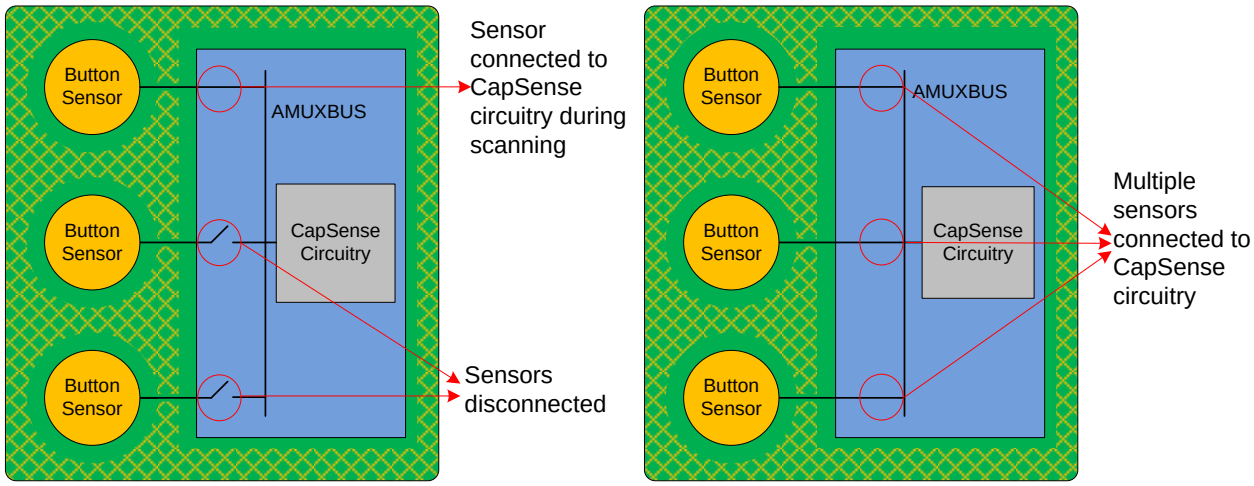


Sensor Ganging: Sensor ganging refers to connecting multiple sensors (Buttons, Proximity trace, Proximity loop) to the CapSense circuitry and scanning them as a single sensor as shown in Figure 3-36 (b). Ganging multiple sensors increases the effective sensor area and results in higher proximity-sensing distance, but ensure that the C_P of the ganged sensor does not cross the maximum C_P limit of 45 pF. See AN92239 for details on how to implement proximity sensing using sensor ganging.

Figure 3-36. CapSense-based Proximity Sensing with Sensor Ganging

(a) Only one sensor is connected to AMUXBUS during scanning

(b) Multiple sensors connected to AMUXBUS at the same time for scanning

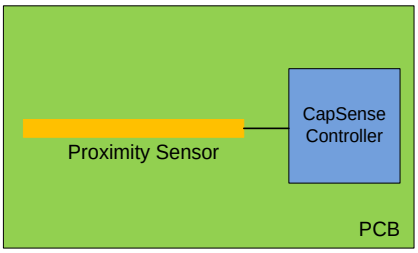


PCB Trace: A long PCB trace on a FR4 or a Flexible Printed Circuit (FPC) board can form a proximity sensor. The trace can be a straight line (Figure 3-37 (a)), or it can surround the perimeter of a system's user interface, as shown in Figure 3-37 (b). Implementing a proximity sensor with a PCB trace has the following advantages when compared to other sensor implementation methods:

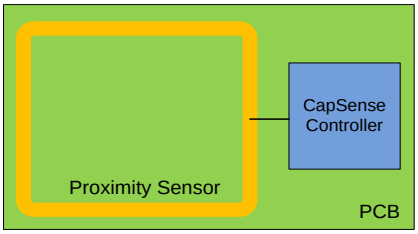
- Proximity-sensor C_P is less
- Proximity-sensing distance is higher because more electric field lines couples to the hand
- More appropriate for mass production

Figure 3-37. CapSense-Based Proximity Sensing with PCB Trace

(a) Bar Proximity Sensor



1. (b) Loop Proximity Sensor



Wire: A single length of wire works well as a proximity sensor. The proximity distance achieved with a wire loop sensor is higher compared to a **PCB** trace. But using a wire sensor is not an optimal solution for mass production because of manufacturing cost and complexity.

3.6.3 Factors Affecting Proximity Distance

Proximity-sensing distance depends on the following hardware, software, and system parameters:

- Hardware parameters
  - ☐ Type of the sensor
  - ☐ Size of the sensor
  - ☐ Parasitic Capacitance (C_P) of the sensor
  - ☐ Overlay material and thickness
  - ☐ Nearby floating or grounded conductive objects
- Software parameters
  - ☐ Resolution of the sensor
  - ☐ Firmware filters
- System parameters
  - ☐ Power consumption
  - ☐ Response time
  - ☐ EMI/EMC/ESD performance

3.6.3.1 Hardware Parameters

- **Type of the sensor** – Proximity-sensing distance is directly proportional to the area of the sensor. A proximity sensor implemented with the button sensor (typical diameter of 5-15 mm) has a small proximity sensing distance compared to a proximity sensor implemented using a PCB trace or wire with a diameter of 1-30 cm. Therefore, the required proximity-sensing distance decides the selection of the proximity-sensor type. Table 3-7 shows when to use a specific proximity-sensor implementation method.

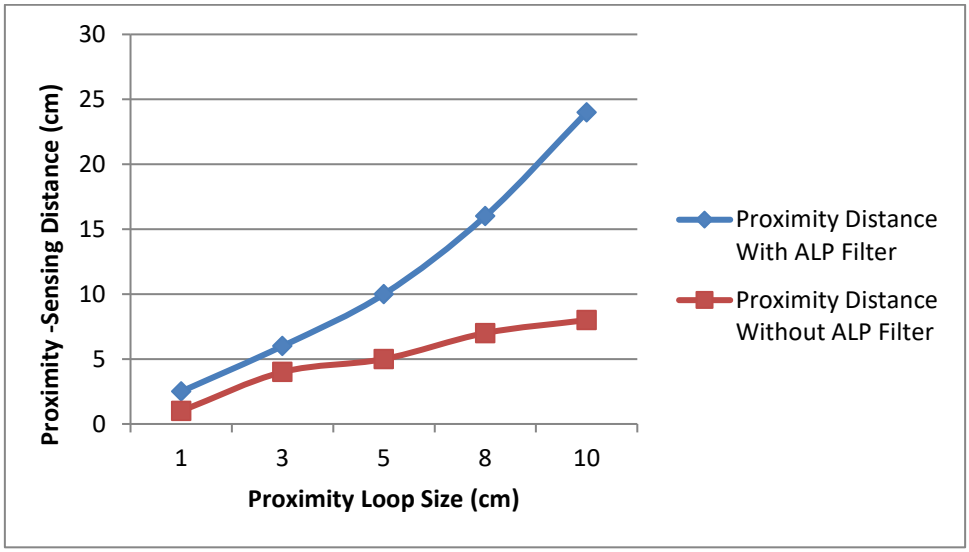
Table 3-7. Selecting Proximity Sensor Implementation Method

Proximity Sensor Type	When to Use
Button sensor	Use this method when the required proximity-sensing distance or the area available for the sensor is very small.
Ganged sensor	Use this method when there is no sensor pin or area available on the PCB for implementing a proximity sensor. Ganging sensors can achieve a larger proximity-sensing distance compared to using button sensors.
PCB trace	Use this method when the required proximity-sensing distance is very large. This method is preferred in most cases.
Wire loop	Use this method when the required proximity-sensing distance is very large. This method has the disadvantage of higher manufacturing cost compared to the implementation with PCB trace.

- **Size of the sensor:** The proximity sensor size depends on various factors, such as the required proximity-sensing distance, presence of noise sources, and floating or grounded conductive objects. Noise sources and floating or grounded conductive objects reduce the SNR and the proximity-sensing distance. Therefore, large proximity sensors are needed to achieve the proximity-sensing distance required in your design.

  Figure 3-38 shows the relationship between the proximity-sensor loop size and the proximity-sensing distance for a given system. A larger sensor area results in more electric field lines coupling with the target object, resulting in increase in the sensor signal. However, a large sensor area results in a high sensor C_P and high noise, and thus reduces the proximity-sensing distance. Using a loop sensor (Figure 3-37 (b)) instead of a solid-fill sensor results in a low sensor C_P, low noise, and thus a large proximity-sensing distance. Also, loop sensors require less sensor area, leaving more space to place components on the PCB.

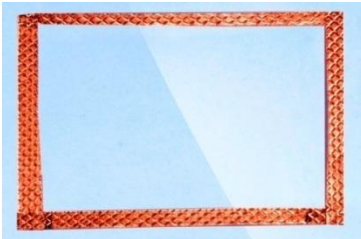
Figure 3-38. Proximity Loop Size versus Proximity Distance



Note In the above graph, proximity-sensing distance for different loop sizes was measured under lab conditions. The actual proximity-sensing distance varies depending on the end-system environment.

It is very difficult to derive a relationship between the sensor size and the proximity-sensing distance. Depending on the end-system environment, the proximity-sensing distance may vary for a specific sensor size. You can find the sensor size needed to achieve a required proximity-sensing distance by making sensor prototypes. You can use a copper foil, as Figure 3-39 shows, to make a quick sensor prototype to determine the sensor size needed to achieve the required proximity-sensing distance.

Figure 3-39. Proximity Sensor Prototype Using Copper Tape



As a rule of thumb, it is recommended that you start with a minimum loop diameter (in the case of a circular loop) or diagonal (in the case of a square loop) equal to the required proximity-sensing distance. If you cannot achieve the required proximity-sensing distance with a loop diameter or diagonal equal to the required proximity-sensing distance, you can increase the sensor loop diameter or diagonal until the required proximity-sensing distance is achieved.

Table 3-8 summarizes the proximity-sensor layout guidelines. If the area available for the proximity sensor is less than the area required to achieve the required proximity-sensing distance, you can implement firmware filters such as the Advanced Low Pass (ALP) Filter. The ALP filter attenuates the noise in the sensor raw count and increases the SNR. An increase in SNR results in a large proximity-sensing distance. See AN92239 – Proximity Sensing with CapSense for details on ALP filter.

Table 3-8. Proximity Sensor Layout Recommendations

Details	Minimum	Recommendation
Proximity sensor loop diameter or diagonal	The sensor loop diameter or diagonal should be equal to or greater than the required proximity-sensing distance if the ALP filter is disabled. If the ALP filter is enabled, the sensor loop diameter or diagonal should be equal to or greater than half of the required proximity-sensing distance.	Start with a sensor loop diameter or diagonal equal to the required proximity-sensing distance and increase the diameter or diagonal until the required proximity-sensing distance is achieved.
Proximity sensor trace width	1.5 mm	1.5 mm

- **Parasitic capacitance of the sensor:** The proximity-sensing distance depends on the ratio of the C_F to the C_P. The proximity-sensing distance increases with an increase in the C_F/C_P ratio. For a given sensor size, the value of C_F depends on the distance between the sensor and the target object. To maximize this ratio, you need to increase C_F and decrease C_P. The C_P of the sensor can be minimized by selecting an optimum sensor area, reducing the sensor trace length, and minimizing the coupling of sensor electric field lines to the ground.

  To reduce the coupling of sensor electric field lines to the ground, drive the hatch fill in the top and bottom layer of the PCB (if there is any) with the driven-shield signal. A hatch fill that is connected to the driven-shield signal is called a “shield electrode.” The driven shield signal is a replica of the sensor signal.

  For shield electrode layout guidelines, see Shield Electrode and Guard Sensor.

  To minimize the sensor trace length and, thereby, the sensor C_P, place the CapSense device as close as possible to the sensor.

- **Nearby floating or grounded conductive objects:** The proximity-sensing distance reduces drastically if there is any floating or grounded conductive object nearby. The following factors cause the proximity-sensing distance to reduce drastically when conductive objects are placed close to the proximity sensor:
  - ☐ The C_P of the sensor increases. Larger sensor C_P often requires reducing the sensor switching frequency, causing the proximity-sensing distance to decrease.
  - ☐ A grounded conductive object catches a part of the sensor electric field and reduces the capacitance added by the target as shown in Figure 3-41.

  You should either remove the nearby conductive object or use a shield electrode to isolate the proximity sensor from the conductive object. The influence of a nearby metal surface on the proximity sensor is reduced by placing a shield electrode between the proximity sensor and the metal object, as shown in Figure 3-42. For layout recommendations on shield electrode, see the Shield Electrode and Guard Sensor section.

Figure 3-40. Electrical Field Propagation for a Single Sensor Configuration without a Metal Object

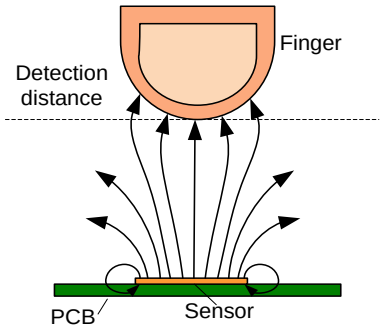


Figure 3-41. Electrical Field Propagation for a Single Sensor Configuration with a Solid Metal Object

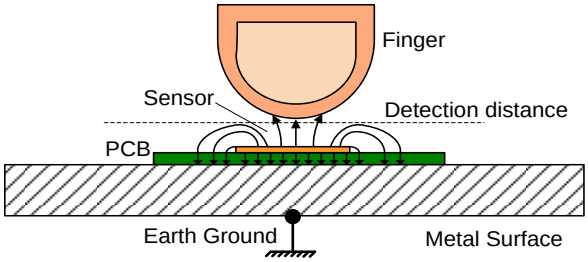
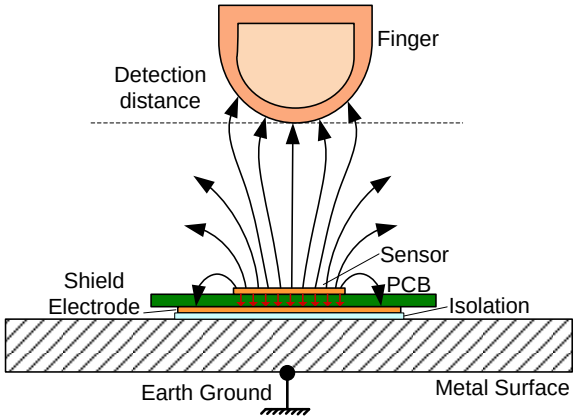


Figure 3-42. Using a Shield Electrode to Decrease the Metal Object's Influence



3.6.3.2 Software Parameters

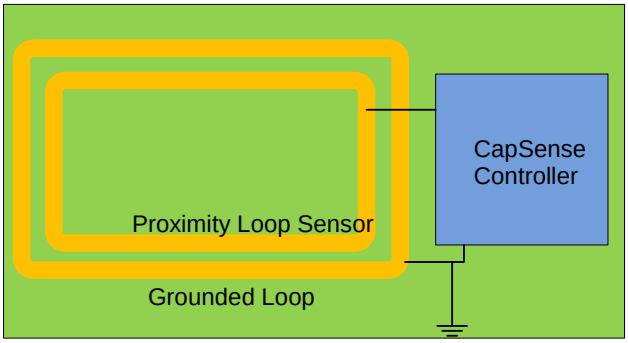
- **Resolution of CSD:** The proximity-sensing distance is directly proportional to the resolution parameter of the CapSense sensing method. With a high-resolution value, you can detect small changes in C_F with an SNR > 5:1. Detecting small changes in C_F essentially means a large proximity distance.
- **Firmware Filters:** Proximity sensors are more susceptible to noise because of their large sensor area and high-sensitivity settings. High noise decreases SNR and hence reduces the proximity-sensing distance. Firmware filters help in reducing the noise, thereby increasing the SNR and the proximity-sensing distance. You can use IIR, median, average or ALP filters to reduce the noise. See the Software Filtering section for details on IIR, median, and average filters. See the application note AN92239 for details on the ALP filter.

3.6.3.3 System Parameters

- **Power consumption:** Proximity sensors require scanning the sensor at a high resolution (15 or 16 bits) to achieve a large proximity-sensing distance. A higher resolution results in a longer scan time and increases the device active time, which leads to a higher power consumption. Therefore, larger proximity distance requires higher power consumption.
- **EMI/EMC/ESD performance:** To achieve a large proximity-sensing distance, the proximity sensor must be tuned for high sensitivity. A high-sensitivity setting results in reduced EMI/EMC performance. Therefore, there is a tradeoff between the proximity-sensing distance and the EMI/EMC performance.

  To improve the ESD performance of the proximity sensor, you can surround the sensor with a ground loop as shown in Figure 3-43.

Figure 3-43. Ground Loop Surrounding the Sensor for Improved ESD Performance



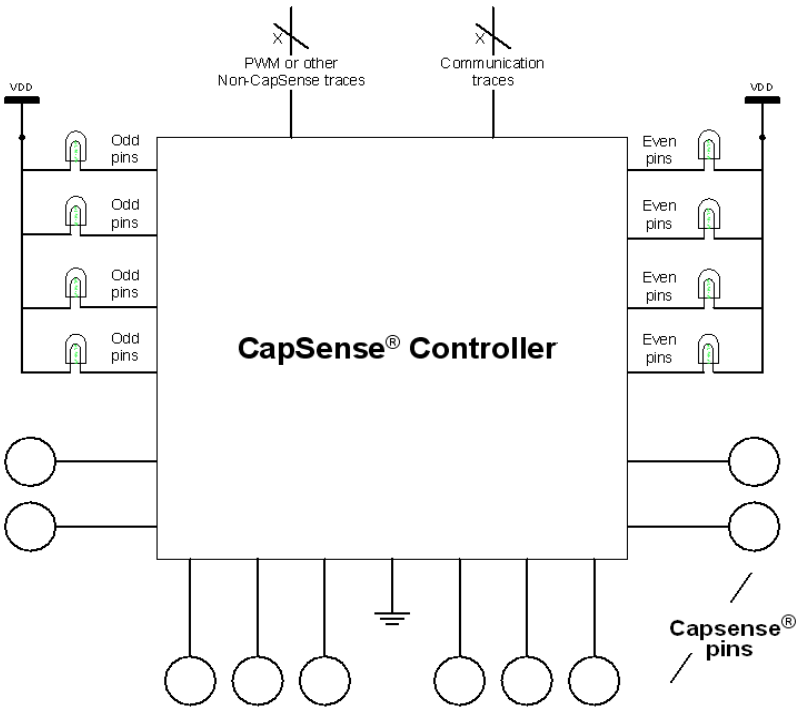
Having a ground loop around the sensor reduces noise in the proximity-sensor and provides a discharge path to the ground during ESD events but reduces the proximity-sensing distance. Therefore, there is a tradeoff between noise immunity and the proximity-sensing distance. The minimum recommended width for ground loop is 1.5 mm and the minimum air gap between the proximity loop and ground loop is 1 mm.

Note Having a ground loop with a small trace width (1.5 mm) around the sensor will not reduce the proximity distance by a drastic amount unlike a large grounded/floating conductive object.

3.7 Pin Assignments

An effective method to reduce interaction between CapSense sensor traces and communication and non-CapSense traces is to isolate each by port assignment. Figure 3-44 shows a basic version of this isolation for a 32-pin QFN package. Because each function is isolated, the CapSense controller is oriented such that there is no crossing of communication, LED, and sensing traces.

Figure 3-44. Recommended: Port Isolation for Communication, CapSense, and LEDs



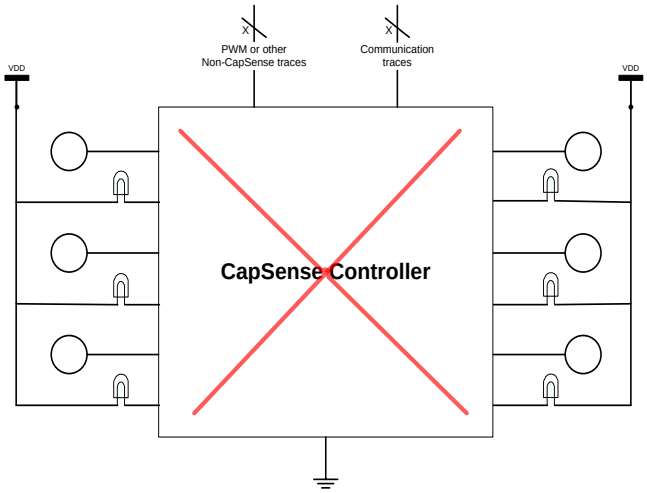
The CapSense controller architecture imposes a restriction on current budget for even and odd port pin numbers. For a CapSense controller, if the current budget of an odd port pin is 100 mA, the total current drawn though all odd port

pins should not exceed 100 mA. In addition to the total current budget limitation, there is also a maximum current limitation for each port pin. See the datasheet of the CapSense controller used in the application to know the specification of that particular CapSense controller.

All CapSense controllers provide high current sink and source capable port pins. When using high current sink or source from port pins, select the ports that are closest to the device ground pin to minimize the noise.

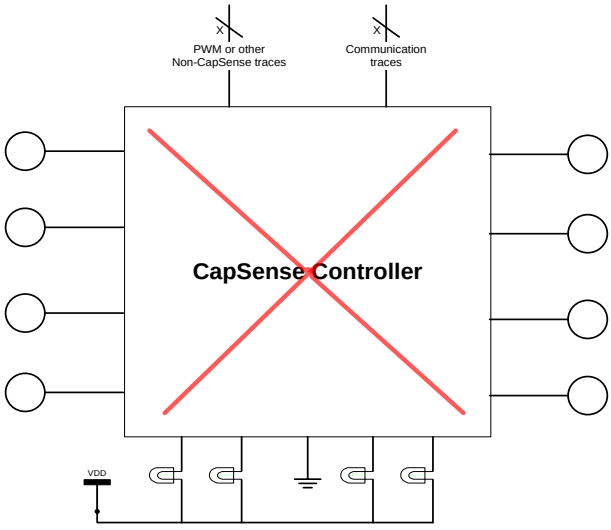
The following three examples demonstrate common pin assignment mistakes. In Figure 3-45, CapSense and non-CapSense traces are not isolated, and CapSense pins are far from ground. This is an example of a bad pin assignment.

Figure 3-45. Not Recommended - CapSense and Non-CapSense Pins in Proximity



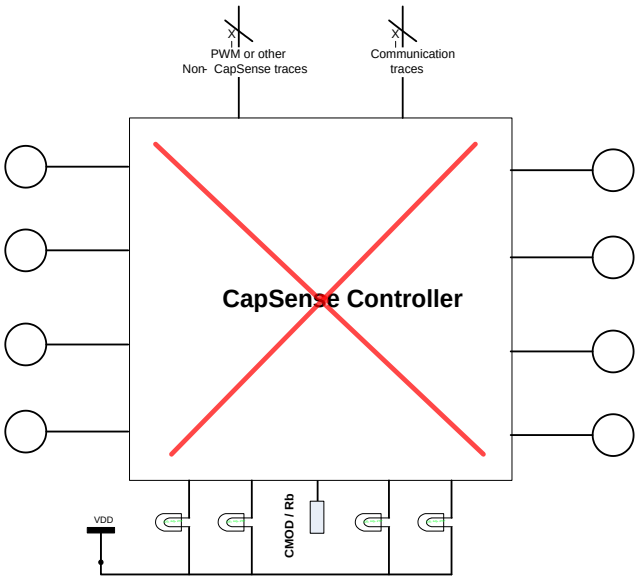
The example in Figure 3-46 achieves good isolation, but it has a bad pin assignment because the LEDs are placed next to the ground pin. The CapSense sensors are assigned to the side of the chip that does not include ground. If the CapSense pins are away from the ground pin, the impedance of the ground path increases, which in turn causes the drive circuit's reference voltage to shift. This shift may lead to false triggering of sensors. For this reason, it is recommended to have CapSense pins near the ground pin.

Figure 3-46. Not Recommended - LEDs and Ground Pins in Proximity



Further, LEDs should not be placed close to C_{MOD}/R_B pin to avoid crosstalk as illustrated in Figure 3-47.

Figure 3-47. Not Recommended: C_{MOD}/R_B and LED Pins in Proximity



Note that in PSoC1, using the P1.0 and P1.1 pins for LEDs or for communication purposes is not recommended. This is because, P1.0 and P1.1 pins are programming lines and upon power up, there will be a low pulse on the P1.0 and P1.1 pins. For further clarity, you can also see the individual device design guides webpage having the sample schematics of all the CapSense devices:

- CY8C21X34 Design Guide
- CY8C20X34 Design Guide
- CY8C20XX6A Design Guide
- CY8C20XX7/S Design Guide

For similar guidelines on PSoC3, PSoC4 and PSoC5LP, see the respective datasheets and design guides.

3.8 PCB Layout Guidelines

In the typical CapSense application, the capacitive sensors are formed by the traces of a printed circuit board (PCB) or flex circuit. Following CapSense layout best practices will help your design achieve higher noise immunity, lower C_P, and higher signal-to-noise ratio (SNR). The CapSense signal drops off at high C_P levels due to drive limits of the internal current sources that are part of the CapSense circuitry. The long time constants associated with high C_P are another reason to avoid high C_P.

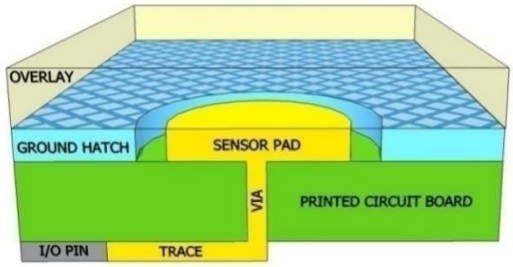
3.8.1 Parasitic Capacitance, C_P

The main components of C_P are trace capacitance and sensor capacitance. C_P is a nonlinear function of sensor diameter, trace length, trace width, and the annular gap. There is no simple relation between C_P and PCB layout features, but here are the general trends. An increase in sensor size, an increase in trace length and width, and a decrease in the annular gap all cause an increase in C_P. One way to reduce C_P is to increase the air gap between the sensor and ground. Unfortunately, widening the gap between sensor and ground will decrease noise immunity.

3.8.2 Board Layers

Most applications use a two-layer board with sensor pads and a hatched ground plane on the top side and all other components on the bottom side. The two-layer stack-up is shown in Figure 3-48. In applications where board space is limited or the CapSense circuit is part of a PCB design containing complex circuitry, four-layer PCBs are used.

Figure 3-48. Two-Layer Stack-Up for CapSense Boards



3.8.3 Board Thickness

FR4-based PCB designs perform well with board thicknesses ranging from 0.020 inches (0.5 mm) to 0.063 inches (1.6 mm).

Flex circuits work well with CapSense, and are recommended for curved surfaces. All guidelines presented for PCBs also apply to flex. Ideally, flex circuits should be no thinner than 0.01 inches (0.25 mm). The high breakdown voltage of the Kapton® material (290 kV/mm) used for flex circuits provides built in ESD protection for the CapSense sensors.

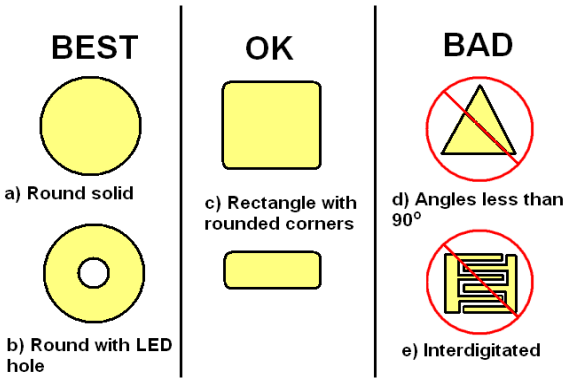
3.8.4 Button Design

This section explains the structure for self cap and mutual cap buttons.

3.8.4.1 Self Cap Button Structure

The best shape for self cap buttons is round. Rectangular shapes with rounded corners are also acceptable. Because sharp points concentrate fields, avoid sharp corners (less than 90º) when designing your sensor pad.

Figure 3-49. Recommended Button Shapes



Button diameter can range from 5 mm to 15 mm, with 10 mm being suitable for the majority of applications. A larger diameter helps with thicker overlays.

Annular gap size should be equal to the overlay thickness, but no smaller than 0.5 mm, and no larger than 2 mm. For example, a PCB layout for a system with a 1-mm overlay should have a 1-mm annular gap, while a 3-mm overlay design should have a 2-mm annular gap. The spacing between the two adjacent buttons should be large enough that if one button is pressed, a finger should not reach the annular gap of the other button.

3.8.4.2 Mutual Cap Buttons of Fishbone Structure

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.

Prongs or fishbone are 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 between the two electrodes. Figure 3-50 shows an example of a three-prong fishbone sensor structure. The gap between the outer wall of the Tx electrode and the coplanar hatch ground should be greater than the air-gap of Tx and

Rx electrodes. The reference plane (PCB bottom layer) of the fishbone structure should have void region as shown in Figure 3-50.

Figure 3-50. Fishbone Pattern of Mutual-Capacitance Button Design

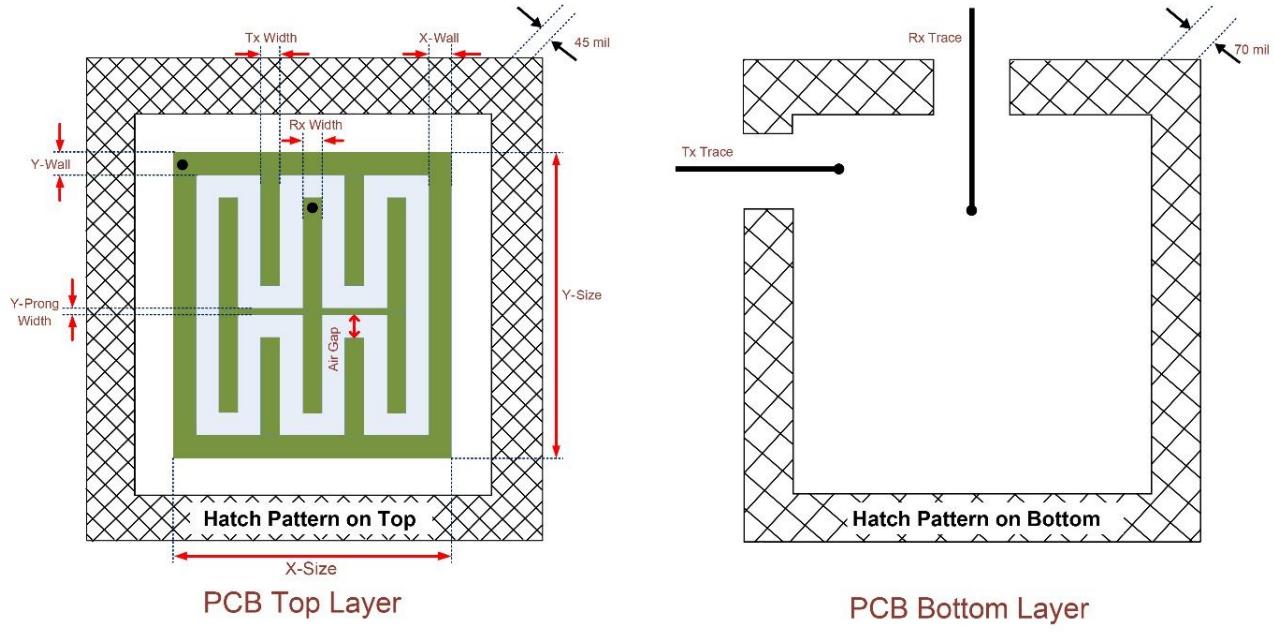


Table 3-9 lists some of the common fishbone structure dimensions.

Table 3-9. Dimension of Fishbone Buttons (all units in mm)

Key Size (X-Size, Y-Size)	PCB Thickness	Number of Rx-prongs	Gap between Tx and Rx	Tx Width	Rx Width	X-Wall Width	Y-Wall Width
13, 10	1.5	2	1.4	1.8	1.8	1	1
13, 10	0.1	3	1.2	0.7	1.2	0.4	0.3
20, 13	0.1	2	1.2	2.7	2.7	3	3.55
10, 13	1.5	2	1.2	0.6	0.6	1.7	1.5
10, 10	1.5	2	1.4	1	1	0.7	0.7

The key dimensions listed in Table 3-9 are examples of some proven button structures that show good SNR performance for different types of PCBs (FR4, flex PCB, ITO based touch film), 4 mm thick overlay, and of overlay permittivity value 3.5. You can contact Cypress technical support team if you need to create custom key pattern in your application.

3.8.5 Slider Design

Figure 3-51 shows the recommended slider pattern for a linear slider and Table 3-10 shows the recommended values for each of the linear slider dimensions. Detailed explanation on the recommended layout guidelines are provided in the following sections.

Figure 3-51. Typical Linear Slider Pattern

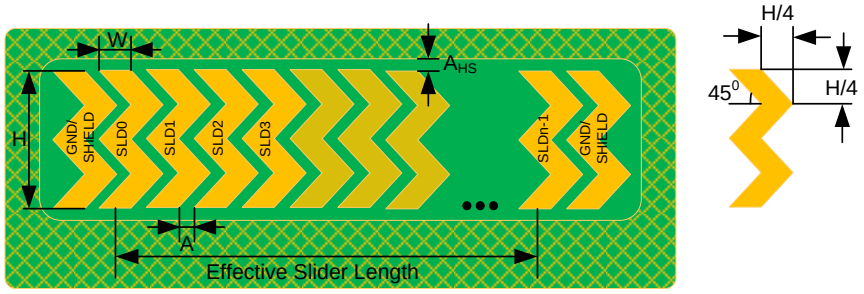


Table 3-10. Linear Slider Dimensions

Parameter	Acrylic Overlay Thickness	Minimum	Maximum	Recommended
Width of the Segment (W)	1 mm	2 mm	-	8 mm[6]
	3 mm	4 mm	-	
	4 mm	6 mm	-	
Height of the Segment (H)	-	7 mm[b]	15 mm	12 mm
Air-gap between Segments (A)	-	0.5 mm	2 mm	0.5 mm
Air-gap between hatch and slider (A_{HS})	-	0.5 mm	2 mm	Equal to overlay thickness

3.8.5.1 Slider-Segment Shape, Width, and Air Gap

A linear response of reported finger position (i.e. Centroid) vs. actual finger position on a slider requires that a slider design be such that whenever a finger is placed anywhere between middle of 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 finger is placed at the exact middle of any slider segment, the adjacent sensors should report difference count = noise threshold. Therefore, it is recommended to use a double chevron shape, as Figure 3-51 shows. This shape helps in achieving a centroid response close to the ideal response, as Figure 3-52 and Figure 3-53 show.

For the same reason, the slider-segment width and air-gap (i.e. dimensions “W” and “A” respectively, as marked in Figure 3-51) should follow the relation mentioned in Equation 3-21.

[6] The recommended slider-segment-width is based on an average human finger diameter of 9 mm. See Section 3.8.5.1, “Slider-Segment Shape, Width, and Air Gap” for more details.

[b] 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, provided the overlay thickness and CapSense tuning is such that an SNR ≥ 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

Figure 3-52. Ideal Slider Segment Signals and Centroid Response

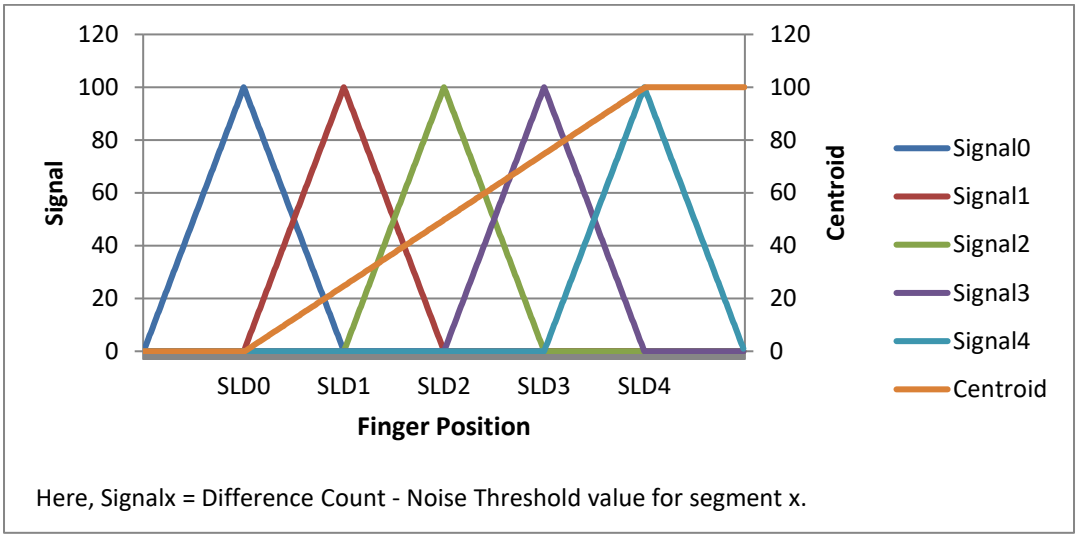
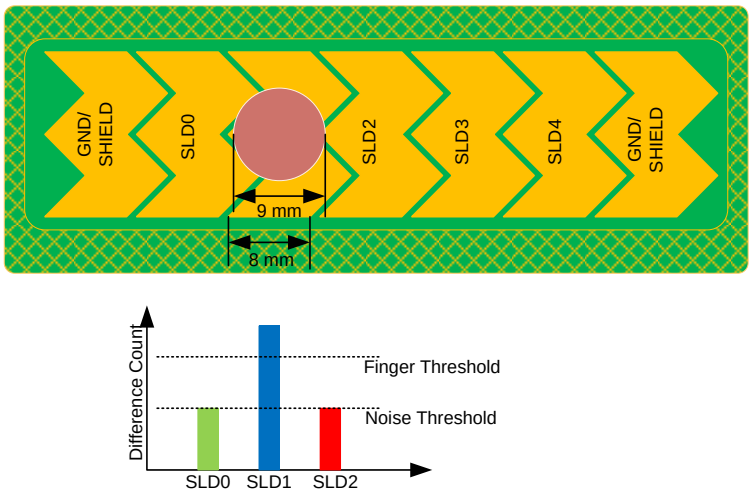


Figure 3-53. Ideal Slider Signals



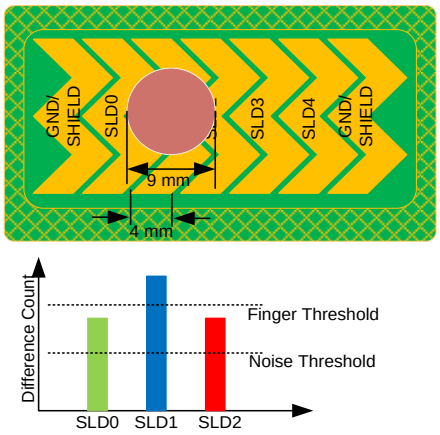
Equation 3-21. Segment width and air-gap relation with finger diameter

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

Typically, an average human finger diameter is approximately 9 mm. Based on this average finger diameter and Equation 3-21, the recommended slider-segment-width and air-gap is 8 mm and 0.5 mm respectively.

If the *slider-segment-width + 2 * air-gap* is lesser than *finger diameter*, as required per Equation 3-21, 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 3-54 shows. Thus, calculated centroid position per Equation 3-21 will be non-linear, as Figure 3-54 shows.

Figure 3-54. Finger Causes Valid Signal on More Than Two Segments when Slider Segment Width Is Lower Than Recommended



Equation 3-22. Centroid algorithm used by CapSense

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

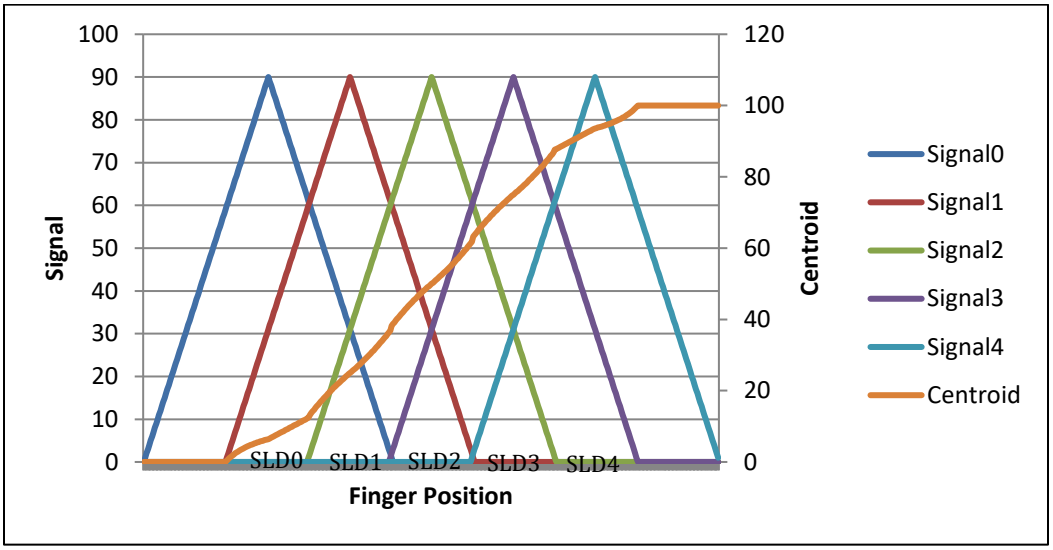
Resolution – API Resolution set in the Customizer,

n – Number of sensor elements in the Customizer.

maximum: Index of element which gives maximum signal.

Si – different counts (with subtracted Noise Threshold value) near by the maximum position

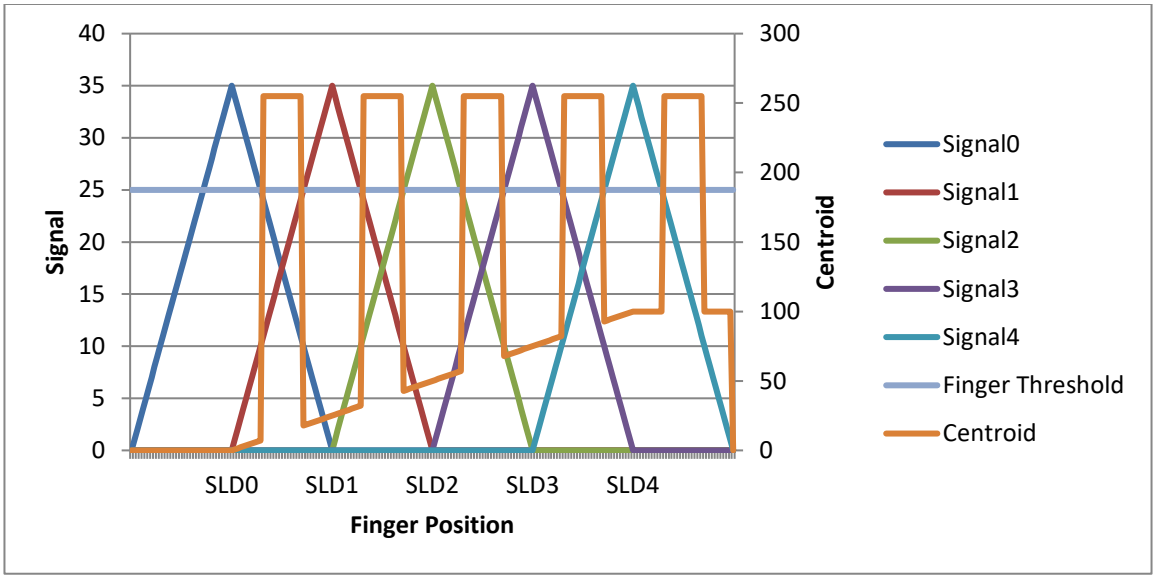
Figure 3-55. Nonlinear Centroid Response when Slider Segment Width Is Lower Than Recommended



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 3-54 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 an SNR of >=5:1 (i.e. signal >= Finger Threshold parameter) at that position. If the slider-

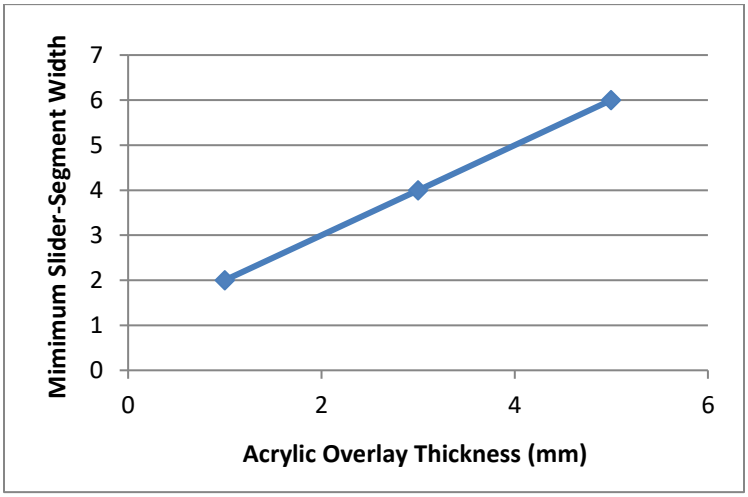
segment-width is too low, a finger may not be able to couple enough capacitance, and hence, none of the slider-segments will have a 5:1 SNR, resulting in a reported centroid value of 0xFF[8], as Figure 3-56 shows.

Figure 3-56. Incorrect Centroid Reported when Slider Segment Width Is Too Low



The minimum value of slider-segment-width for certain specific overlay thickness values, for an acrylic overlay, are provided in Table 3-10. For acrylic overlays of thickness values, which are not specified in Table 3-10 and Figure 3-57 may be used to estimate the minimum slider-segment-width.

Figure 3-57. Minimum Slider-Segment-Width w.r.t. Overlay Thickness for an Acrylic Overlay



If the *slider-segment-width + 2 * air-gap* is higher than *finger diameter*, as required per Equation 3-21, the centroid response will have flat spots i.e., if the finger is moved a little near the middle of any segment, the reported centroid position will remain constant as Figure 3-58 shows. This is because, as Figure 3-59 shows, when the finger is placed in the middle of a slider-segment, it will add valid signal only to that segment even if the finger is moved a little towards the adjacent segments.

[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 Finger Threshold parameter.

Figure 3-58. Flat Spots (Nonresponsive Centroid) when Slider Segment Width Is Higher Than Recommended

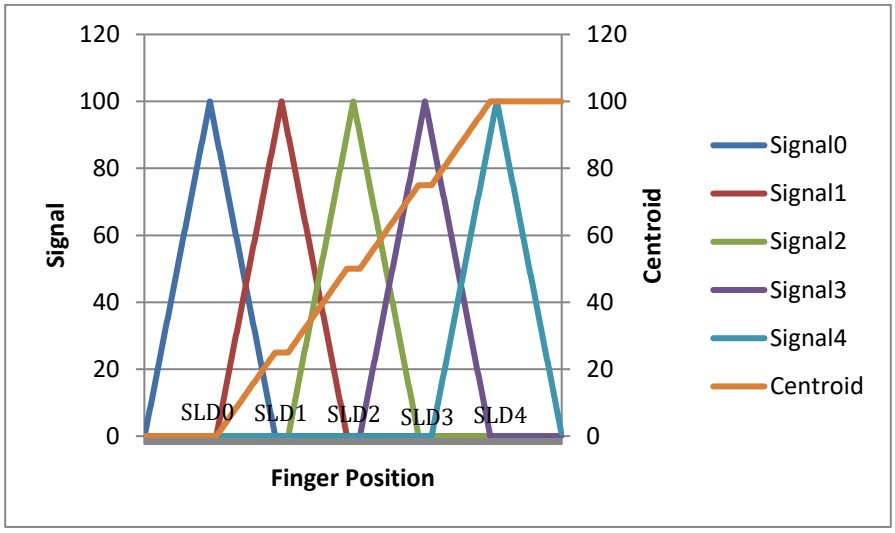
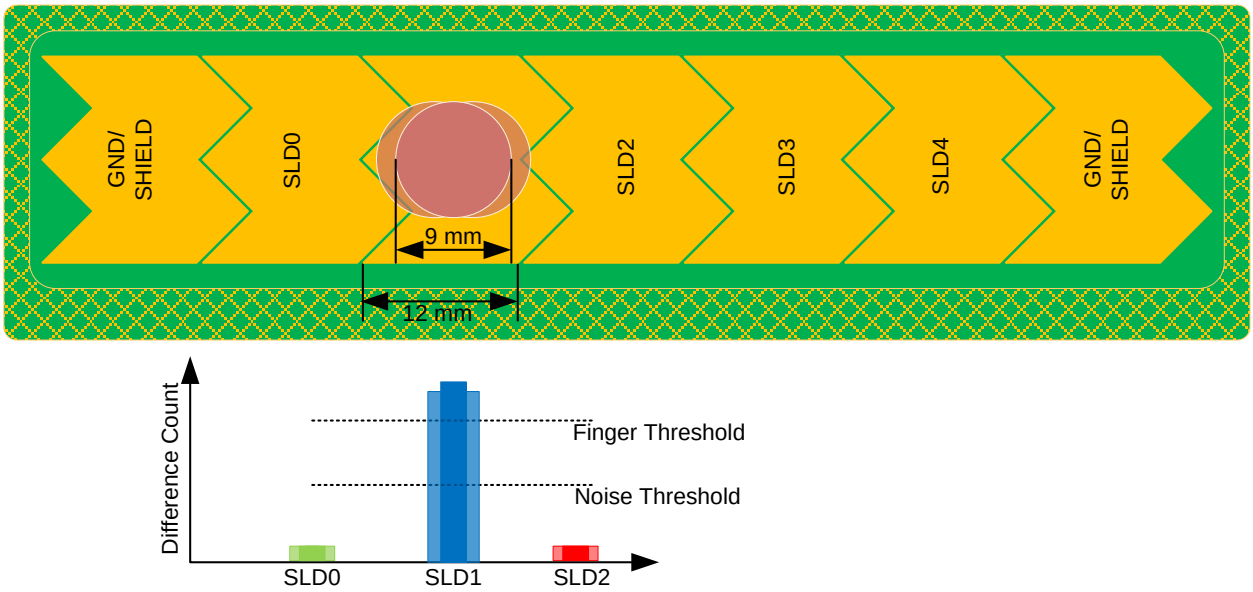


Figure 3-59. Signal on Slider Segments when Slider Segment Width Is Higher Than Recommended



Note that if the *slider-segment-width + 2 * air-gap* is higher than *finger diameter,* it may be possible to increase and adjust the sensitivity of all the slider segments such that even if the finger is placed in middle of a slider-segment, the adjacent sensors report a difference count value equal to noise threshold value (as required per

Figure 3-52); however, this will result in hover effect i.e. the slider may report a centroid position even if the finger is hovering above the slider and not touching the slider.

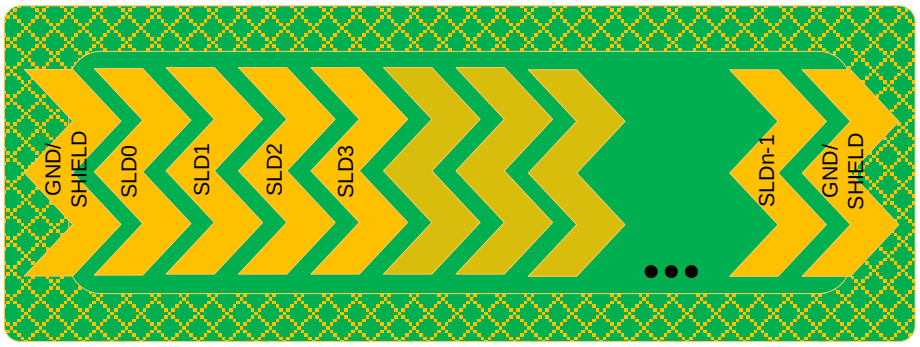
3.8.5.2 Dummy Segments at the Ends of Slider

In a CapSense design, when one segment is scanned; the adjacent segments are connected to either ground or to the driven shield signal based on the option that will be specified in the “Inactive sensor connection” parameter in the CapSense CSD component. For linear centroid response, the slider requires all the segments to have same sensitivity i.e. the increase in the raw count (signal) when a finger is placed on the slider segment should be same for all the segments. To maintain a uniform signal level from all the slider segments, it is recommended to physically connect the two segments at the 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 that will be specified in the “Inactive sensor connection” parameter. Therefore, if your application requires an ‘n’ segment slider, it is recommended to create n + 2 physical segments, as Figure 3-51 shows.

If it is not possible to have two segments at the both ends of a slider due to space constraints, you can implement these segments in the top hatch fill, as Figure 3-60 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 that will be 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 driven shield signal.

Figure 3-60. Linear Slider Pattern when First and Last Segments Are Connected to Top Hatch Fill



3.8.5.3 Deciding Slider Dimensions

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

a) Decide the required length of slider (L), based on application requirements. This is same as the “effective slider length” as Figure 3-51 shows.
b) Decide the height of segment based on available space on the board. Use maximum allowed segment height (15 mm) if board space permits, else, use a lesser height but ensure that the height is greater than the minimum specified in Table 3-10.
c) The slider-segment-width and the air-gap between slider segments should be as recommended in Table 3-10. The recommended slider-segment-width and air-gap for an average finger diameter of 9 mm is 8 mm and 0.5mm respectively.
d) For a given slider length, L, calculate the number of segments required 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 lesser than the calculated number of segments for a certain application, you may increase the segment width to achieve the required slider length with given number of pins. For example, a 10.2 cm slider requires 13 segments. However, if only 10 pins are available, segment width may be increased to 10.6. This will either result in a non-linear response as Figure 3-58 shows, or a hover effect; however, this layout may be used if the end application does not need a high linearity.

Note that the PCB length is higher than the required slider length as Figure 3-51 shows. PCB length can be related to slider length as follows:

Equation 3-23. Relationship between minimum PCB length and slider length

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

If the available PCB area is lesser than that required per above equation, you can remove the dummy segments.

In this case, the minimum PCB length required will be as follows:

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

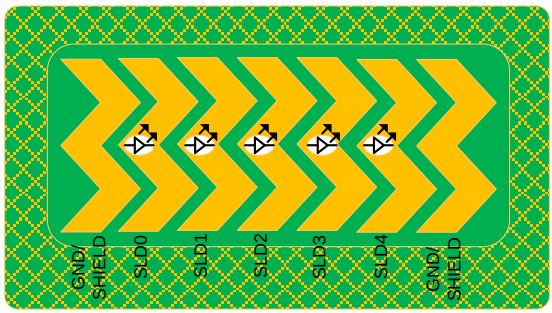
Keep in mind the following layout guidelines while designing a slider:

- 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 PSoC device are same for all the segments.
- Maintain the same air gap between the sensors or traces to ground plane or hatch fill.

3.8.5.4 Slider Design with LEDs

In some applications, it might be required to display 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 3-61 shows. When a hole is drilled for placing an LED, the effective area of the slider segment reduces. To achieve an SNR > 5:1, you need to have a slider segment with a width larger than the LED hole size. See Table 3-10 for minimum slider width required to achieve an SNR > 5:1 for a given overlay thickness. Follow the guidelines provided in Crosstalk Solutions section for routing the LED traces.

Figure 3-61. Slider Design with LED Backlighting



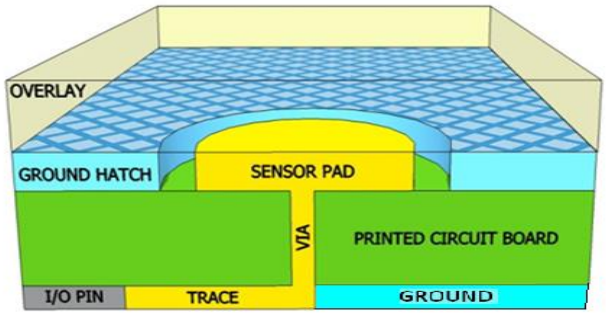
3.8.6 Sensor and Device Placement

For a CapSense design on 2-Layer and 4-Layer PCBs, follow the below guidelines for sensor and component placement. If your design requires liquid tolerance, follow the guidelines explained in the Hardware Component section.

3.8.6.1 2-Layer PCB:

- Place the sensors on the top layer of the PCB, as Figure 3-62 shows.
- Place the components and route the sensor traces on the bottom layer of the PCB.

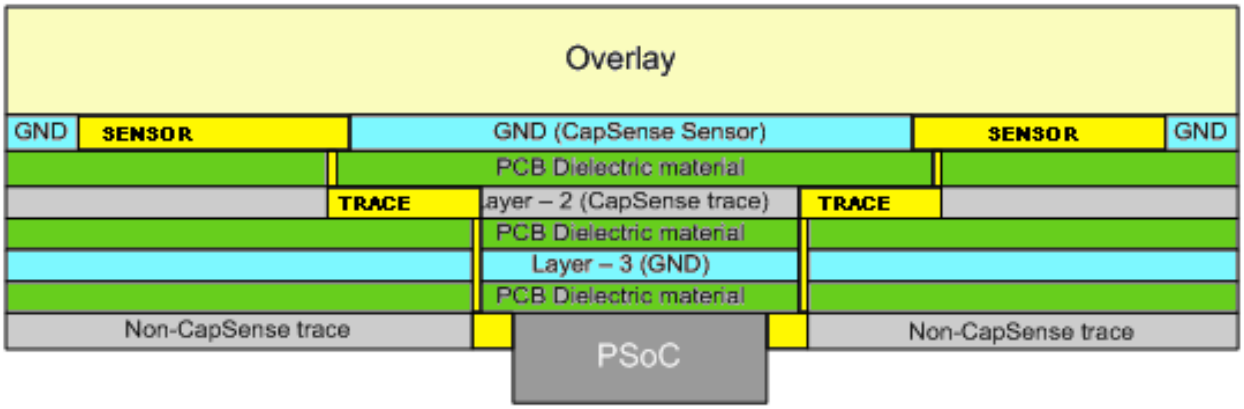
Figure 3-62. CapSense Design on 2-Layer PCB



3.8.6.2 4-Layer PCB:

- Place the sensors on the top layer of the PCB.
- Route the sensor traces in the layer-2.
- Place a hatch fill of 7-mil trace and 70-mil spacing and connect it to ground in layer-3.
- Place components in the bottom layer, as Figure 3-63 shows. The unoccupied areas can be filled with a hatch copper fill of 7-mil trace and 70-mil spacing and should be connected to ground.

Figure 3-63. CapSense Design on 4-Layer PCB



In addition to these guidelines, follow the best practices to ensure a robust and reliable CapSense design.

- Minimize the trace length from the CapSense controller pins to the sensor pad to optimize signal strength.
- Mount series resistors within 10 mm of the controller pins to reduce RF interference and provide ESD protection.
- Mount the controller and all other components on the bottom layer of the PCB.
- Isolate switching signals, such as PWM, I^2C communication lines, and LEDs, from the sensor and the sensor PCB traces. Do this by placing them at least 4 mm apart and fill a hatched ground between CapSense traces and non-CapSense traces to avoid crosstalk.
- Avoid connectors between the sensor and the controller pins because connectors increase C_P and decrease noise immunity.

3.8.7 Trace Length and Width

Minimize the parasitic capacitance of the traces and sensor pad. Trace capacitance is minimized when they are short and narrow.

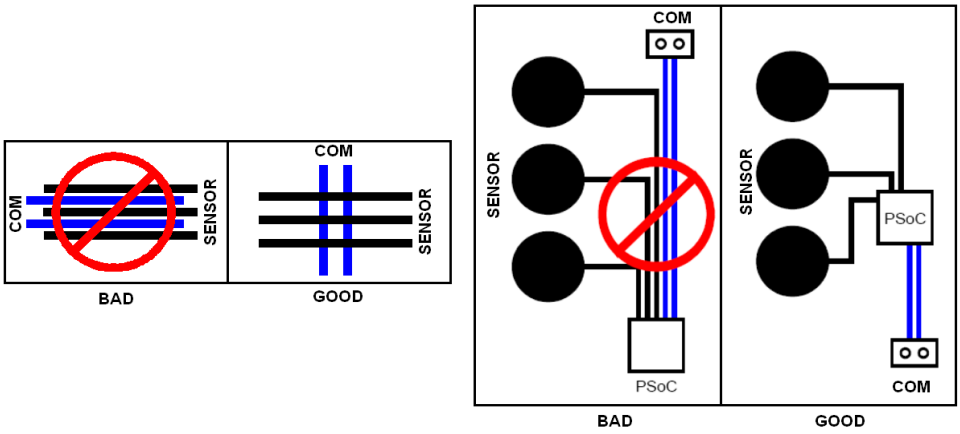
- The maximum recommended **trace length** is 12 inches (300 mm) for a standard PCB and 2 inches (50 mm) for flex circuits.
- **Trace width** should not be greater than 7 mil (0.18 mm). CapSense traces should be surrounded by hatched ground with trace-to-ground air gap of 10 mil to 20 mil (0.25 mm to 0.51 mm).

3.8.8 Trace Routing

Route sensor traces on the bottom layer of the PCB, so that the only user interaction with the CapSense sensors is with the active sensing area. Do not route traces directly under any sensor pad unless the trace is connected to that sensor.

Do not run capacitive sensing traces in close proximity to communication lines, such as I^2C or SPI masters. If it is necessary to cross communication lines with sensor pins, make sure the intersection is at right angles, as illustrated in Figure 3-64.

Figure 3-64. Routing of Sensing and Communication Lines



3.8.9 Crosstalk Solutions

A common backlighting technique for panels is to mount an LED under the sensor pad so that it shines through a hole in the middle of the sensor. When the LED is switched on or off, the voltage transitions on the trace that drives the LED can couple into the capacitive sensor input, creating noisy sensor data. This coupling is referred to as crosstalk. To prevent crosstalk, isolate CapSense and non-CapSense traces from one another. In the case of CY8C21X34/B, the crosstalk can also occur due to the coupling of the LED voltage transitions with the R_B resistor. To avoid this, isolate the R_B trace from the non-CapSense traces. A minimum separation of 4 mm is recommended. A hatched ground plane also can be placed between those traces to isolate them. LED drive traces and CapSense traces (including R_B trace) should not be routed together.

Figure 3-65. Not Recommended - LED and CapSense in Close Proximity

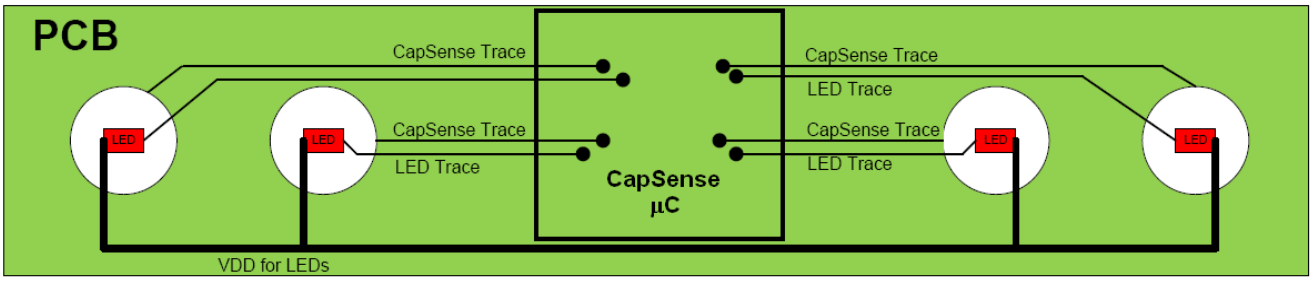
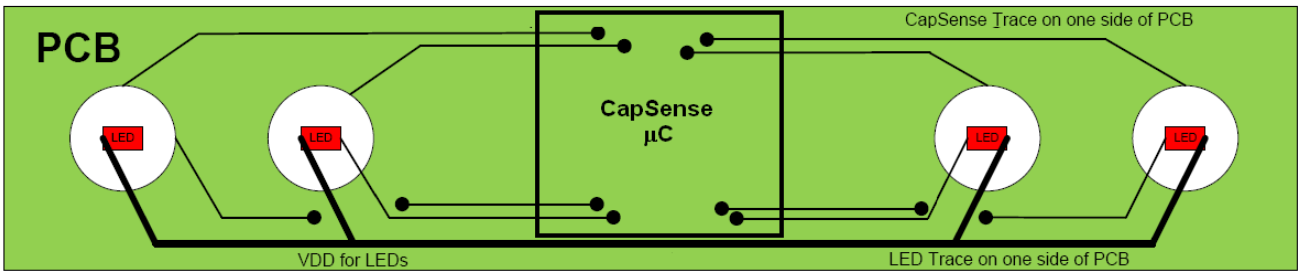
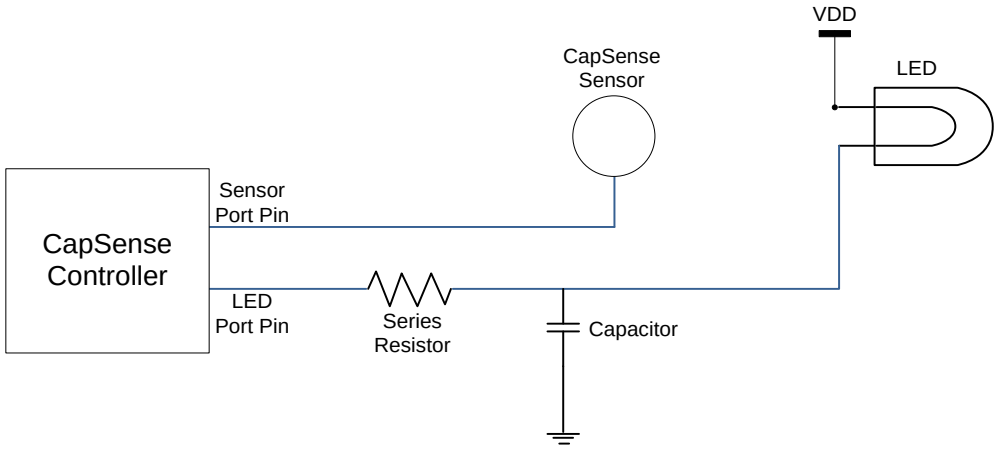


Figure 3-66. Recommended - LED and CapSense with Wide Separation



Another approach to reducing crosstalk is to slow down the rising and falling edges of the LED drive voltage using a filter capacitor. Figure 3-67 shows an example circuit of this solution. The value of the added capacitor depends on the drive current requirements of the LED; however, a value of 0.1 µF is typical.

Figure 3-67. Filter Capacitor Solution for Crosstalk



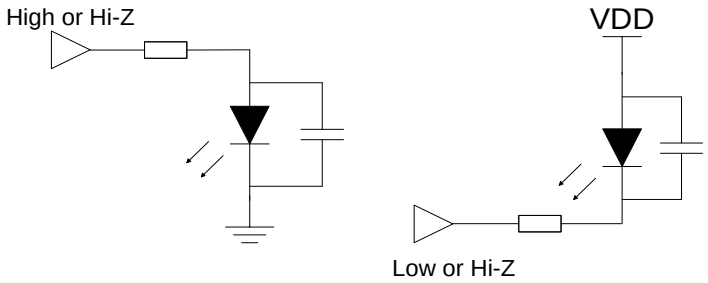
3.8.10 LEDs Close to CapSense Sensors

If LEDs are placed close to the CapSense sensors (within 4-mm distance), and if either end of the LED changes to a non-low impedance state at any point in time, the capacitance of the sensors changes between the On and Off states of the LEDs. The change in output impedance of the LED driver circuit can cause the sensors to false-trigger or sensors to go off unintentionally when the LEDs change state.

To avoid the effect of LEDs that are placed close to the sensors, the LEDs must be bypassed with a capacitor with a typical value of 1 nF. This is important in scenarios where LEDs are pulled down or pulled up to switch on and are left floating when switched off.

The value of the bypass capacitor must be such that it provides a constant low-impedance path as less as 1 kΩ at 100 KHz, as seen by the sensor on both the ends of the LEDs.

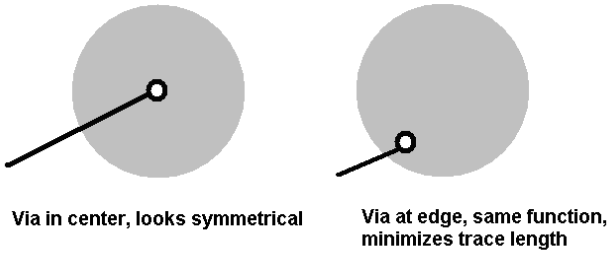
Figure 3-68. LED Circuits



3.8.11 Vias

Use the minimum number of vias to route CapSense inputs to minimize parasitic capacitance. The vias should be placed to minimize the trace length, which is usually on the edge of the sensor pad, as shown in Figure 3-69.

Figure 3-69. Vias Placement on Sensor Pad



3.8.12 Ground Plane

Ground fill is added to both the top and bottom of the sensing board. When ground fill is added near a CapSense sensor pad, there is a tradeoff between maintaining a high level of CapSense signal and increasing the noise immunity of the system. Typical hatching for the ground fill is 25 percent on the top layer (7 mil line, 45 mil spacing) and 17 percent on the bottom layer (7-mil line, 70-mil spacing).

Figure 3-70. Recommended Button and Slider Layout Top Layer

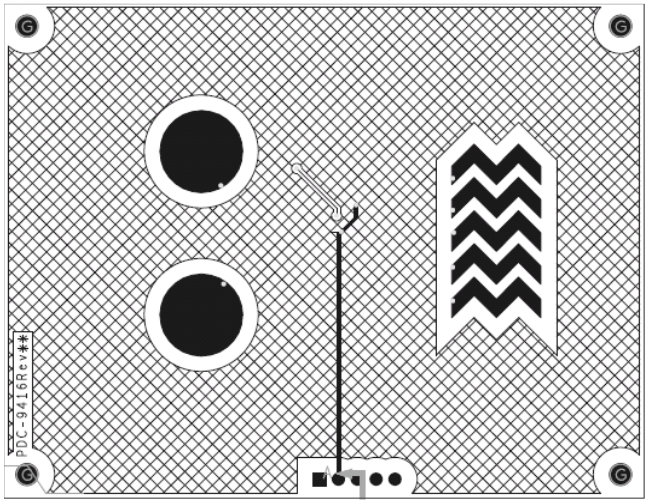
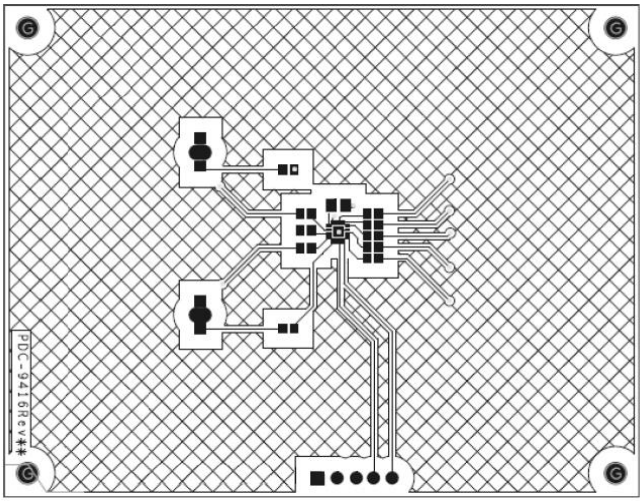


Figure 3-71. Recommended Button and Slider Layout Bottom Layer



3.8.13 Power Supply Layout Recommendations

A poor PCB layout would introduce noise in high-sensitivity sensors (e.g., proximity sensors and button sensors that use overlays thicker than 1 mm). To achieve low noise on high-sensitivity CapSense sensors in designs, it is important that the PCB layout follows the best practices on power supply trace and placement.

1. Two decoupling capacitors must be connected between the VDD and VSS pins. (Capacitor specification: 0.1 µF and 1 µF, 16 V, Ceramic, X7R).
2. One decoupling capacitor must be connected between the VCC and VSS pins for CY8CMBR3XXX devices. (Capacitor specification: 0.1 µF, 16 V, Ceramic, X7R).
3. When using packages E-pad (paddle), it must be connected to the board GND.
4. The decoupling capacitors and CMOD capacitor must be connected as close as possible to the chip to keep impedance of the ground and supply traces as low as possible.

Figure 3-72 illustrates the schematic diagram with these recommendations. C1, C2, and C4 are decoupling capacitors and C3 is the CMOD capacitor.

Figure 3-72. Example Schematics for Improved SNR

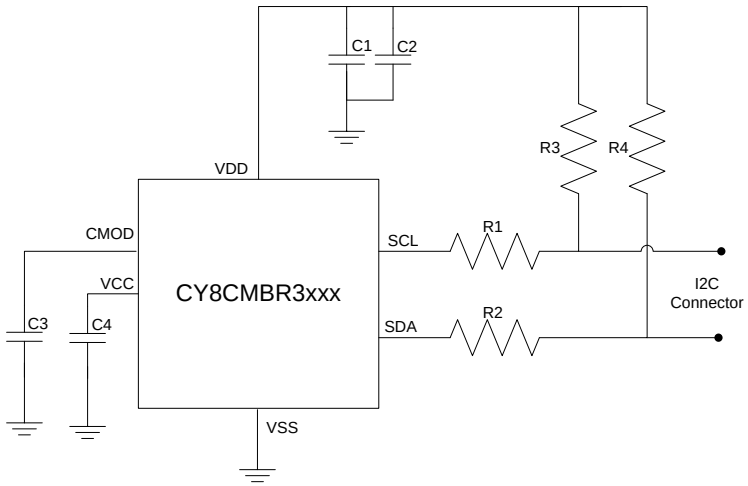
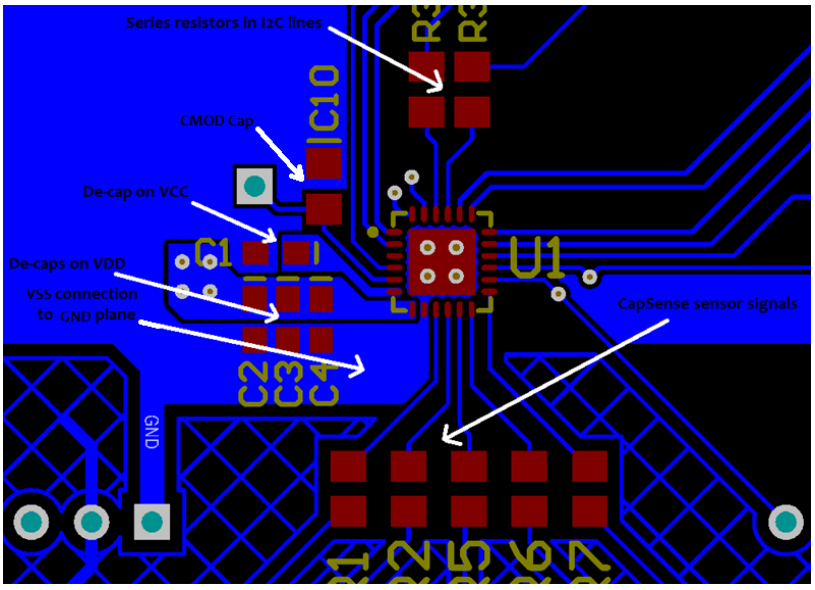


Figure 3-73 shows an example board layout diagram with placements of decoupling and CMOD capacitors and routing of ground and supply. (Note that the I^2C pull-ups present in Figure 3-72 are not shown in the layout in this figure).

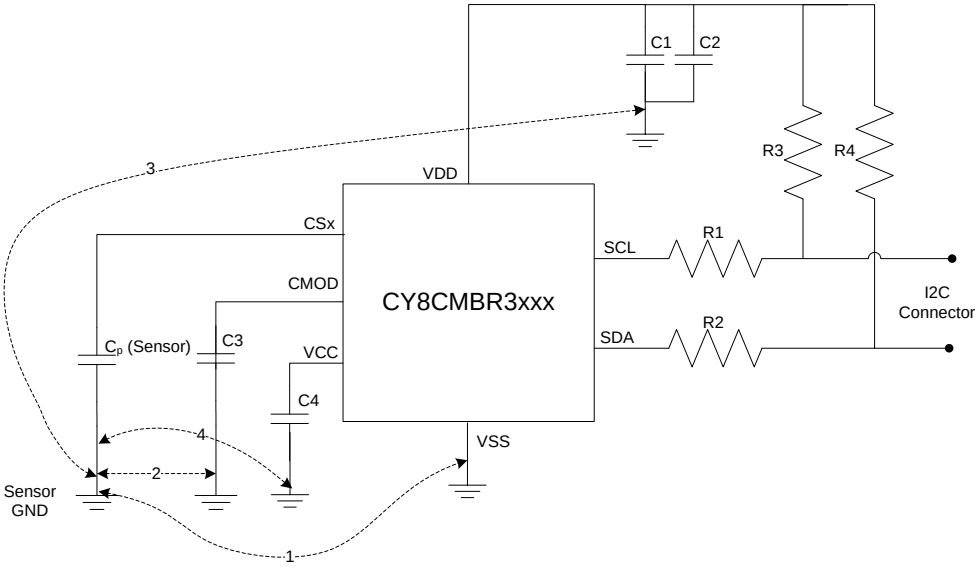
Figure 3-73. Example Board Layout for Improved SNR



A good CapSense schematics diagram must have all passive components shown in the schematics diagram.

The PCB layout shown above is for guidelines only. For a good layout, the inductance between multiple ground nodes must be below 0.2 nH. The ground nodes are indicated in the schematics diagram in Figure 3-74.

Figure 3-74. Important GND Nodes in CapSense Design



3.8.14 Shield Electrode and Guard Sensor

A shield electrode is a hatched fill that is driven with a signal which is the replica of the sensor signal. A shield electrode is used for the following purposes:

- **Reduce sensor parasitic capacitance (C_P):** In most of the CapSense applications it is recommended to have a hatch fill in the top and bottom layer of the PCB surrounding the sensor and its traces. This hatch fill is connected to ground for improved noise immunity. When a sensor has a large trace length it results in high sensor C_P. High sensor C_P results is low sensor sensitivity and increases power consumption. To reduce the sensor C_P you should drive the hatch fill in the top and bottom layer with a driven shield signal.

- **Reduce the effect of nearby floating/grounded conductive objects:** As explained in the Factors Affecting Proximity Distance section, shield electrode can be used to reduce the effect of floating/grounded conductive objects on proximity distance. In this case, the shield electrode should be placed in between the conductive object and the proximity sensor, as shown in Figure 3-42.
- **Provide directionality to proximity sensing:** The electric field of a proximity sensor is omnidirectional and can detect proximity in all directions. In most of the applications, it is required to detect proximity from only one direction. In such cases, you can use a shield electrode to make the proximity sensor sense the target object in a single direction.
- **Provide liquid tolerance:** As explained in Liquid Tolerance, shield electrodes prevent false triggers due to the presence of liquid droplets on the CapSense sensor.

3.8.14.1 Shield Electrode for Proximity Sensing

If you want to use shield electrode for reducing sensor C_P or reduce the effect of nearby floating/grounded conductive objects or provide directionality to proximity sensing, follow the below guidelines:

- To reduce the sensor C_P, use a hatch fill of 0.17 mm (7 mil) trace and 1.143 mm (45 mil grid) in the top layer and a hatch fill of 0.17 mm (7 mil) trace and 1.778 mm (70 mil grid) in the bottom layer and drive it with driven shield signal.
- To reduce the effect of floating/grounded conductive object on the proximity-sensing distance, place a hatch fill of 0.17 mm (7 mil) trace and 1.143 mm (45 mil grid) in between the sensor and the conductive object and drive the hatch fill with the driven shield signal.
- To make the proximity sensing unidirectional, place a hatch fill of 0.17 mm (7 mil) trace and 1.143 mm (45 mil grid) in between the sensor and the direction in which proximity detection should be avoided and drive the hatch fill with the driven shield signal.

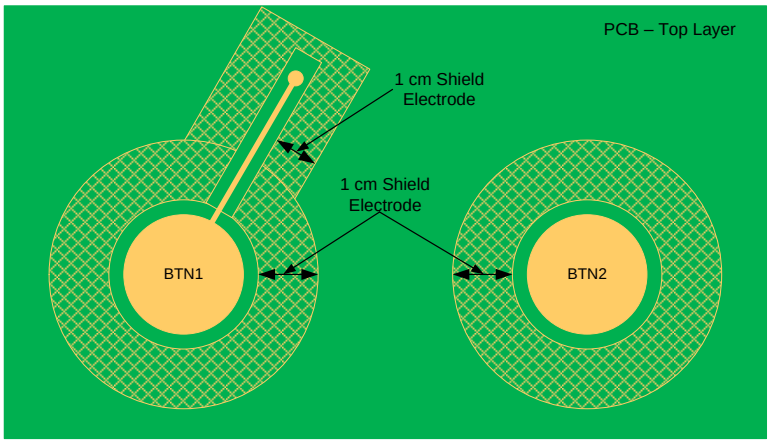
3.8.14.2 Shield Electrode Construction for Liquid Tolerance

As explained in Liquid Tolerance, 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 shield electrode area 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 might increase the radiated emissions. If the board area is very large, the area outside the 1-cm shield electrode should be left empty, as Figure 3-75 shows. For improved liquid tolerance performance there should not be any hatch fill or a trace connected to ground in the top and bottom layer of PCB. When there is a grounded hatch fill or a trace then, when a liquid droplet falls on the touch interface, it might cause sensor false triggers. Even if there is a shield electrode in between the sensor and ground, the effect of shield electrode will be totally masked out and sensors might false trigger.

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

Figure 3-75. Shield Electrode Placement when Sensor Trace is Routed in Top and Bottom Layer



Follow the below guidelines implementing the shield electrode in a 2-Layer and 4-layer PCB:

2-Layer PCB:

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

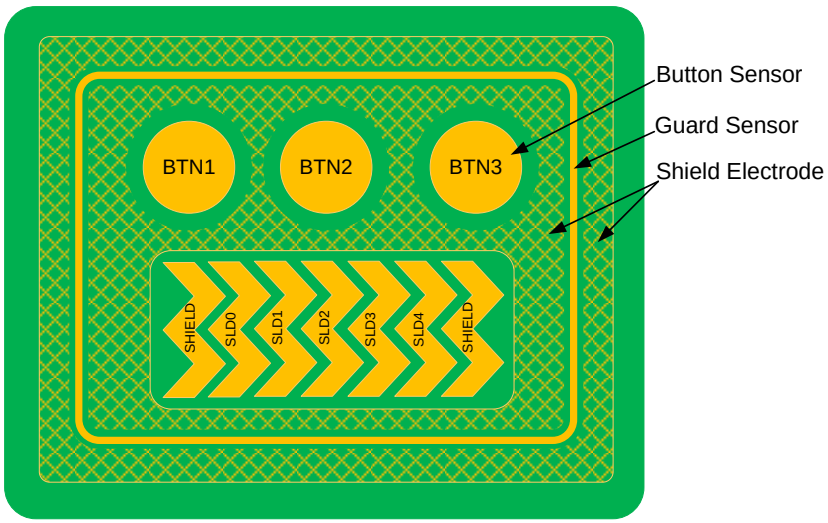
4-Layer PCB:

- Top layer: Hatch fill with 7-mil trace and 45-mil grid (25 percent fill). Hatch fill should be connected to driven shield signal.
- Layer-2: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to driven shield signal.
- Layer-3: V_{DD} Plane
- Bottom layer: Hatch fill with 7-mil trace and 70-mil grid (17 percent fill). Hatch fill should be connected to ground.
- The recommended air-gap between the sensor and the shield electrode is 1 mm.

3.8.14.3 Guard Sensor

As explained in Liquid Tolerance, the guard sensor is a copper trace that surrounds all of the sensors, as Figure 3-76 shows.

Figure 3-76. PCB Layout with Shield Electrode and Guard Sensor



The guard sensor layout should be placed such that:

- 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 surrounds all the sensors in a CapSense system as Figure 3-76 shows.
- It should not be triggered while pressing a button or slider sensor. Otherwise, the button sensors and slider sensor scanning will be disabled and the CapSense system become non-operational 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 the below guidelines for implementing 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 air gap between the guard sensor and the shield electrode is 1 mm.

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

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

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

3.8.15 CapSense System Design with Single Layer PCB

Electronic product manufacturers face constant pressure to lower system costs. Several markets, including consumer and home appliances, are switching to single layer PCBs to support their required product margins. Cypress's CapSense controllers provide robust touch sensing on single layer PCBs, and their driven shield capability enables longer trace length, proximity sensing, and liquid tolerance. CapSense delivers IEC (IEC 61000-6-1, IEC 61000-6-2) noise compliant performance for accurate touch responses even in noisy environments using sophisticated firmware algorithms. For more details on implementing CapSense touch sensing on single layer boards, contact capsense@cypress.com.

See Schematic and Layout Checklist before you start your design to ensure that the best practices for a CapSense design are followed.

3.8.16 CapSense System Design with ITO

For applications where the CapSense sensors need to be transparent (sensors are positioned over the display), CapSense sensors can be implemented with ITO. However, if transparent sensors are not required, it is recommended to use copper pads as it provides a higher yield, lower cost, and better performance when compared to ITO sensors.

The ITO sensors can be implemented on a glass substrate or a plastic film (polyethylene terephthalate) substrate. The sensor shape recommended in section Button Design for button sensors and section Slider Design for slider sensors applies to the sensor design on ITO. Make sure that the sensor length or width (where length is the always the longest dimension) aspect ratio does not exceed 5/3.

Trace length for the sensors should be kept at minimum to reduce the overall sensor resistance. The formula for trace resistance is provided in the following equation:

$$Resistance = Trace\ sheet\ resistance \times Trace\ length \div Trace\ width$$

Trace resistance is evaluated relative to the sensor resistance. High trace resistance, compared to the sensor resistance, degrades the touch performance. Therefore, it is recommended to keep the sensor trace length as short as possible. The layout guidelines for ITO sensors are summarized in Table 3-11.

Table 3-11. Layout Guidelines for ITO Sensors

Category	Parameter	Min	Typ	Max	Units	Remarks
ITO	Sheet Resistance (Glass Substrate)	-	-	120	Ω/sq	-
	Sheet Resistance (Film Substrate)	-	-	270	Ω/sq	-
	Sensor Max Resistance	-	1	30	kΩ	End-to-end
	Spacing Between Traces	30	50	100	µm	-
	Routing Channel Trace Width	10	30	50	µm	-

3.9 Example Schematic and Layout

See the design files of the Development Kits corresponding to your chosen device for reference schematics. See any development kit for a reference layout and sensor design. Layout files of the kits are the examples of different PCB stack-up, placement of devices, routing procedure, and implementation of different CapSense buttons and sliders. For example:

- CY8CKIT-149 kit has the implementation of recommended fishbone structure of mutual capacitance buttons of different dimensions
- PSoC 62S3 Wi-Fi BT Prototyping Kit (CY8CPROTO-062S3-4343W) kit has fishbone button structure optimized for both mutual capacitance and self capacitance based sensing.

Visit the corresponding kit webpages for accessing the design files.

You can also see the individual device design guides webpage for sample layouts of all the CapSense devices:

- CY8C21X34 Design Guide
- CY8C20X34 Design Guide
- CY8C20XX6A Design Guide
- CY8C20XX7/S Design Guide
- CY8CMBR3XXX CapSense Design Guide
- PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide

3.10 PCB Assembly and Soldering

It is important to follow PCB assembly and soldering standards and guidelines for any CapSense design. Although CapSense PCB assembly and soldering do not have any specific guidelines, the following specifications and the application notes give the standard and guidelines respectively.

- IPC-A-610, Acceptability of Electronic Assemblies
- AN72845 - Design Guidelines for Cypress Quad Flat No Extended Lead (QFN) Packaged Devices
- AN69061- Design, Manufacturing, and Handling Guidelines for Cypress Wafer Level Chip Scale Packages

4. CapSense Selector Guide

Cypress is the world leader in capacitive sensing technologies. Our broad range of solutions provide robust noise immunity, enable quick time to market and system scalability, and have replaced more than 5 billion mechanical buttons over the past several years. The CapSense portfolio ranges from simple buttons and sliders to more sophisticated solutions integrating other system components to reduce total BOM cost and form factor. Cypress CapSense controllers feature best-in-class liquid tolerance and capacitive proximity sensing, which is easy to implement with SmartSense Auto-Tuning, an algorithm that constantly monitors and compensates for environmental conditions.

Salient features include:

- Advanced sensing techniques for easy finger detection through 15 mm of glass or 5 mm of plastic
- Revolutionary and unique SmartSense Auto-Tuning algorithm
- Industry's best solutions for advanced features such as proximity and water tolerance
- Ultra-low power with industry's widest voltage range
- Industry's leading small form factor packaging such as WLCSP (2 mm x 2 mm)

4.1 Defining CapSense Requirements

You must consider several key system requirements when selecting the best CapSense device for your application.

- Configurable/Programmable

  Choose configurable controllers if you are looking for an easy and quick-to-market solution without firmware development. These devices can work with or without a host controller. Choose programmable controllers if you need more integration in one chip and more flexibility in designing your application. Cypress offers both hardware-configurable and register-configurable controllers.

- Configuration Interface

  Configurable controllers have two configuration interfaces – hardware-based and register-based. Hardware-configurable controllers require external resistors for configuring different features and eliminate the configuration step required during production or during system operation from a host controller. However, register-configurable devices support register-based configuration and comprehensive status reporting through I^2C. The same I^2C interface can serve as both configuration and host communication interfaces. These controllers support retaining the configuration in EEPROM, enabling standalone operation without a host chip.

- Programming Interface

  The flexible design of PSoC allows the programming pins to be reused as GPIOs thus reducing the I/O count. However, you should ensure in the design that the external components or long trace-length, required for the GPIO function, do not interfere with programming and vice versa.

  Updating the device firmware in-system from a host controller is an important requirement, particularly for mobile applications. PSoC controllers offer two solutions.

- Host-Sourced Serial Programming (HSSP)

  This method uses the dedicated PSoC programming interface and does not require a change in the PSoC firmware. However, it requires the reset pin of the PSoC to be controlled by the host. In this method, the host requires a special firmware that programs the PSoC through bit-banging and may require the host-side I/Os to be dedicated for programming.

- Programming through Bootloader

  This method uses the standard communication interfaces such as I²C, UART, and USB for programming and requires bootloader to be implemented as part of the PSoC firmware.

  Cypress provides many HSSP and bootloader application notes. See Table 5-1 to pick one and learn more.

- Number and type of capacitive sensors

  PSoC's unique design allows any I/O pin[9] to be used as a CapSense sensor of any type, which includes a button, a slider segment, or a proximity sensor. This offers full flexibility to implement a wide variety of capacitive touch sensing applications. In addition to the I/O requirement for the sensors, most of the latest CapSense devices require only one external capacitor (CMOD). The following table takes a typical CapSense application and calculates the number of IOs required.

Table 4-1. I/O Requirement Calculation Example

Requirement	I/O Count
10 Button Sensors	10
1 Five segment Slider	5 (1 pin for each segment)
1 Proximity Sensor	1
10 LEDs	10
Liquid Tolerance	1 pin for shield electrode[10]
I2C/Programming through Bootloader	2
CMOD[11] Capacitor	1
Total I/Os	30

- Number of CapSense Blocks

  Some CapSense controllers provide two CapSense blocks, which can be used to scan two sensors simultaneously. You should choose these controllers if your application includes many sensors and has a very strict response time requirement.

- Communication Interface

  Communication interface is an important requirement if your CapSense design involves a host controller. This interface will allow the host controller to configure the device and get the user interface data in the system. I²C is a popular interface in capacitive touch sensing applications. Cypress controllers support I²C, SPI, and UART depending on the device.

- CPU, Flash, and RAM requirements

  Arm® is the popular CPU choice for embedded applications as it allows portability across different vendors and scalability for complex application requirements. Cypress offers the Arm Cortex®-M0 based PSoC 4 family, the Arm Cortex-M3 based PSoC 5LP and dual core (Arm Cortex-M4 and Cortex-M0+) based PSoC 6 families with different Flash and RAM combinations to suit your needs.

- Liquid tolerance

  Liquid tolerance is a critical requirement for home appliance and other high-reliability applications for preventing false touches in the presence of liquid. CapSense controllers support liquid tolerance through the driven-shield[10] technique.

- Operating voltage range – CapSense controllers support a wide operating voltage range of 1.71 V to 5.5 V
- Feedback – CapSense solutions support standard I/O as well as PWM outputs for buzzers, LEDs, and Haptics
- Package size and pin count – CapSense controllers support a wide variety of packages suitable for mobile, home appliances, and industrial markets. The packages available, depending on the device, include QFN, SSOP, SOIC,

[9] Some devices do not support sensing on all the I/O pins. See the pinouts section in the corresponding device datasheet to understand the capability of I/O pins.

[10] See Liquid Tolerance to learn how CapSense controllers enable liquid tolerance using shield electrode.

[11] CSD requires one external capacitor called CMOD for operation. See CapSense Sigma Delta Modulator (CSD) for details.

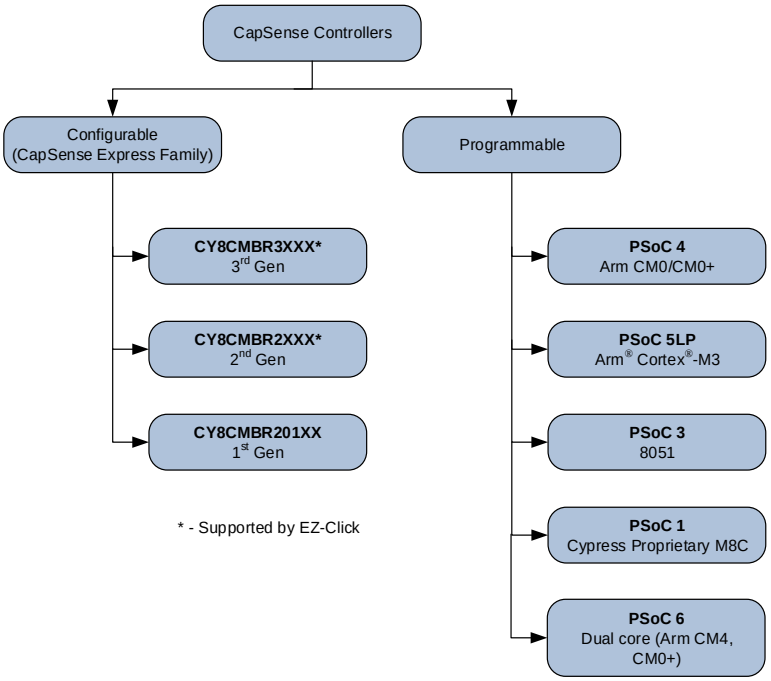
TQFP, and LQFP. CapSense controllers also include chip-scale packages (CSP) than can be as small as 2 mm x 2 mm. After you select a device, See the datasheet for more information on the packages.

- Additional features such as ADC, PWM, Timers, LCD driver, and analog peripherals such as opamps, and comparators.

4.2 CapSense Portfolio

Cypress offers a wide range of configurable and programmable CapSense controllers. Figure 4-1 presents the overview of the CapSense portfolio. The following sections explain the different CapSense families.

Figure 4-1. CapSense Portfolio Overview



4.2.1 Configurable CapSense Controllers (CapSense Express Family)

Configurable CapSense controllers, also called CapSense Express controllers, eliminate the need for firmware development and make capacitive touch designs very easy and quick. These controllers are configurable either through I^2C or through hardware straps (resistors) and feature SmartSense Auto-Tuning (except CY8C201xx family), which eliminates the tuning process and compensates for environmental changes during runtime. The parasitic capacitance (C_P) supported by these devices is limited to 45 pF.

- **CY8CMBR3XXX** controllers are the third and latest generation of configurable controllers that can be configured over I^2C and are supported by the GUI-based configuration tool EZ-Click. These devices support up to 16 buttons, up to eight LEDs, up to two proximity sensors, up to two 5-segment sliders, and an operating voltage range of 1.71 V to 5.5 V. See product overview to understand the difference between the devices available in this family. These controllers provide improved noise immunity over the previous generation devices and Cypress recommends these controllers for new designs.
- **CY8CMBR2XXX** controllers are second generation configurable devices, which include both hardware-configurable (CY8CMBR20XX) and I^2C-configurable (CY8CMBR21XX) devices and the I^2C-configurable devices are supported by the EZ-Click tool. Choose these devices if you need a hardware-based configuration or advanced LED effects such as dimming and fading in your application. These devices support up to 16 buttons and operating voltage range of 1.71 V to 5.5 V. See the product overview to understand the difference between the parts available in this family.
- **CY8C201XX** controllers are first generation configurable devices supporting I^2C interface for configuration. These devices support up to 10 I/Os for buttons, LEDs, sliders, and operating voltage range of 2.4 V to 5.5 V. Cypress does not recommend these controllers for new designs.

Table 4-2 shows the comparison between the different parts in the configurable category.

Table 4-2. CapSense Express Family Features Comparison

Features/ Device Family	CY8CMBR3XXX						CY8CMBR2XXX				CY8C201XX
	CY8CMBR3116	CY8CMBR 3106S	CY8CMBR 3110	CY8CMBR 3108	CY8CMBR 3102	CY8CMBR 3002	CY8CMBR 2016	CY8CMBR 2110	CY8CMBR 2010	CY8CMBR 2044	
Datasheet	CY8CMBR3XXX						CY8CMBR2016	CY8CMBR2110	CY8CMBR2010	CY8CMBR2044	CY8C201XX
Maximum Number of Sensors	16	11	10	8	2	2	16	10	10	4	10
Buttons	Up to 16	Up to 11	Up to 10	Up to 8	Up to 2	2	Up to 16	Up to 10	Up to 10	Up to 4	Up to 10
Sliders	No	Up to 2	No	No	No	No	No	No	No	No	1
Proximity	Up to 2	Up to 2	Up to 2	Up to 2	Up to 2	No	No	No	No	No	No
Liquid Tolerance	Yes					No					
LED/GPO	Up to 8	0	Up to 5	Up to 4	Up to 1	2	Up to 8	Up to 10	Up to 10	Up to 4	Up to 10
Buzzer	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No
Configuration Interface	I2C	I2C	I2C	I2C	I2C	No (Fixed Function)	Hardware	I2C	Hardware	Hardware	I2C
Communication Interface	I2C/GPO	I2C	I2C/GPO	I2C/GPO	I2C/GPO	GPO	GPO	I2C/GPO	GPO	GPO	I2C/GPO
Power-On Selft-Test	Yes										No
Operating Voltage	1.71-5.5V										2.4 V-5.25 V
Automotive Qualified (AEC-Q100)[12]	No										
Package	24-pin QFN	24-pin QFN	16-pin SOIC	16-pin QFN	8-pin SOIC	8-pin SOIC	48-pin QFN	32-pin QFN	32-pin QFN	16-pin QFN	8/16-pin SOIC, 16-pin QFN

[12] Contact Cypress to know the latest status of the products.

4.2.2 Programmable CapSense Controllers

Programmable CapSense controllers are based on the PSoC platform and offer a rich set of analog and digital peripherals along with CapSense. Cypress provides a great deal of pre-built, production-ready, and GUI-configurable firmware components to speed-up your PSoC system design.

- **PSoC 6** family features 32-bit dual core CPU subsystem (Arm Cortex-M4 and CM0+) clocked up to 150 MHz and integrates many advanced peripherals 12-bit SAR ADC, up to two opamps, up to two comparators, PWM, USB-FS host and device peripherals, I2S and PDM channels for audio subsystem. Table 4-3 shows the comparison between the different sub-families of PSoC 6.
- **PSoC 4** family features 32-bit Arm CM0 CPU clocked up to 48 MHz and integrates many advanced peripherals – 12-bit SAR ADC, opamps, comparators, PWM, Bluetooth Low Energy (BLE), CAN, and a segment LCD driver. PSoC 4 supports up to 54 sensors. PSoC 4 is the best low-power mixed-signal architecture and the most cost-effective device family. Table 4-4 shows the comparison between the different sub-families of PSoC 4.
- **PSoC 5LP** family features 32-bit Arm Cortex-M3 CPU clocked up to 80 MHz and integrates many advanced peripherals, such as two CapSense blocks, 20-bit delta-sigma ADC, 12-bit SAR ADC, 8-bit DAC, opamps, comparators, PWM, USB 2.0 Full-Speed peripheral, CAN, and a segment LCD driver. PSoC 5LP supports up to 62 sensors and is suitable for large and complex systems that need higher levels of integration in a single chip. Table 4-5 shows the comparison between the different PSoC 5LP sub-families.
- **PSoC 3** family features a 8-bit, single-cycle 8051 CPU clocked up to 67 MHz and integrates many advanced peripherals, such as a 20-bit delta-sigma ADC, 8-bit DAC, opamps, comparators, PWM, USB 2.0 Full-Speed peripheral, CAN, and a segment LCD driver. PSoC 3 supports up to 62 sensors. Choose PSoC 3 if you require the same large-scale integration offered by PSoC 5LP but do not require a high-performance CM3 CPU. Table 4-6 shows the comparison between the different PSoC 3 sub-families.
- **PSoC 1** family is the Cypress' first programmable system-on-chip solution featuring the proprietary 8-bit M8C CPU clocked up to 24 MHz. This family supports many CapSense features such as buttons, sliders, proximity sensing, and liquid tolerance at an economical price. PSoC 1 supports up to 44 sensors and includes many automotive qualified (AEC-Q100) parts compared to other device families. Table 4-7 shows the comparison between the different PSoC 1 sub-families.

Table 4-3. PSoC 6 Family Features Comparison

Features/Device Family	PSoC 61	PSoC 62	PSoC 63	CY8C62xA, CY8C62x8[13]	CY8C62x5
Datasheet	PSoC 61 Datasheet	PSoC 62: CY8C62x6, CY8C62x7 Datasheet	PSoC 63 with BLE Datasheet	PSoC 62: CY8C62x8, CY8C62xA Datasheet	CY8C62x5 Datasheet
CPU	150-MHz Arm Cortex-M4	150-MHz Arm Cortex-M4 and 100-MHz Cortex M0+	150-MHz Arm Cortex-M4 and 100-MHz Cortex M0+	150-MHz Arm Cortex-M4, 100-MHz Cortex-M0+	150-MHz Arm Cortex-M4, 100-MHz Cortex-M0+
Flash	Up to 1 MB Application Flash with 32 KB EEPROM area and 32 KB Supervisory Flash	1 MB Application Flash with 32-KB EEPROM area and 32-KB Supervisory Flash	Up to 1-MB Application Flash, 32-KB emulated EEPROM area, and 32-KB Supervisory Flash	2 MB Application Flash, 32-KB emulated EEPROM area, and 32-KB Supervisory Flash	512 KB Application Flash, 32-KB emulated EEPROM area, and 32-KB Supervisory Flash
SRAM	Up to 288 KB	288 KB	Up to 288 KB	1024-KB	256 KB
CapSense Architecture	Fourth Generation (supports CSD and CSX)	Fourth Generation (supports CSD and CSX)	Fourth Generation (supports CSD and CSX)	Fourth Generation (supports CSD and CSX)	Fourth Generation (supports CSD and CSX)
CapSense Average Current Consumption per Sensor	Programmable from 37.5 nA to 2400 µA	Programmable from 37.5 nA to 2400 µA	Programmable from 37.5 nA to 2400 µA	Programmable from 37.5 nA to 609 µA	Programmable from 37.5 nA to 609 µA
SSC/PRS Feature	Yes	Yes	Yes	Yes	Yes
ADC	12-bit 1-Msps SAR ADC	12-bit 1-Msps SAR ADC	12-bit 1-Msps SAR ADC	12-bit 1-Msps SAR ADC	12-bit, 1-Msps SAR ADC
Total I/Os	Up to 100[14]	Up to 100[14]	Up to 100[14]	Up to 100[14]	Up to 64[14]
Number of CapSense blocks	1	1	1	1	1
Sensor Parasitic Capacitance (CP) Range	5 pF – 200 pF	5 pF – 200 pF	5 pF – 200 pF	5 pF – 200 pF	5 pF – 200 pF
DAC	1x 12-bit	1x 12-bit	1x 12-bit	0	0
Comparators	2	2	2	2	2
Op-Amps	2	2	2	0	0
Serial Communication Block (SCB[15])	9	9	9	13	7
Universal Digital Block (UDB[16])	12	12	12	0	0
Secure Digital Host Controller (SDHC) Block	0	0	0	2	1

[13] These devices support 2 MB Flash memory and have MPNs that belong to both PSoC 61 (single CPU) and PSoC 62 (dual CPU) series.
[14] In PSoC 6 family devices, CapSense use is restricted to few ports with switching restrictions on other ports. Follow the recommendations stated in PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide.to achieve the best CapSense sensitivity and accuracy.

[15] Each Serial Communication Block (SCB) can function as an I^2C, an SPI, or a UART communication block. In addition, PSoC Creator provides a Software Transmit UART (TX8) Component for all devices.

[16] Universal Digital Block (UDB) allows implementing custom digital logic. PSoC Creator provides a wide array of UDB-based Components such as I^2C, I^2S, UART, SPI, PWM, and Counter/Timer.

Table 4-4. PSoC 4 Family Features Comparison

Features/Device Family	PSoC 4000-Series	PSoC 4100-Series	PSoC 4200-Series	PSoC 4100 M-Series	PSoC 4200 M-Series	PSoC 4100 BLE-Series	PSoC 4200 BLE-Series	PSoC 4200 L-Series	PSoC 4000 S- Series	PSoC 4100 S-Series/PSoC 4100PS	PSoC 4100S Plus
Datasheet	CY8C401X	CY8C412X	CY8C424X	CY8C412X-M	CY8C424X-M	PSoC 4xx7_BLE - 128 KB Flash, BLE 4.1 PSoC 4xx8_BLE - 256 KB Flash, BLE 4.1 PSoC 4xx8_BLE 4.2 - 256 KB Flash, BLE 4.2		CY8C424X-L	CY8C40XX-S	CY8C41XX-S CY8C41XX-PS	CY8C41XX-S
CPU	16-MHz ARM Cortex-M0	24 MHz ARM Cortex-M0	48 MHz ARM Cortex-M0	24 MHz ARM Cortex-M0	48 MHz ARM Cortex-M0	24 MHz ARM Cortex-M0	48 MHz ARM Cortex-M0	48 MHz ARM Cortex-M0	48-MHz ARM Cortex-M0+	48-MHz ARM Cortex-M0+	48-MHz ARM Cortex-M0+
Flash/SRAM	Up to 16 KB/ Up to 2 KB	Up to 32 KB/ Up to 4 KB	Up to 32 KB/ Up to 4 KB	Up to 128 KB/ Up to 16 KB	Up to 128 KB/Up to 16 KB	Up to 256 KB/Up to 32 KB		Up to 256 KB/ Up to 32 KB	Up to 32 KB/ Up to 4 KB	Up to 64 KB/ Up to 8 KB (For PSoC 4100PS) Up to 32 KB/Up to 4 KB	Up to 128 KB/ Up to 16 KB
Total I/Os	Up to 20	Up to 36	Up to 36	Up to 55	Up to 55	Up to 36	Up to 36	Up to 98	Up to 36	Up to 36 Up to 38 (For PSoC 4100PS)	Up to 54
CapSense I/Os (Supports Button, Slider, Proximity, Shield)	Up to 16	Up to 35	Up to 35	Up to 54	Up to 54	Up to 35	Up to 35	Up to 97	Up to 35	Up to 35 Up to 33 (For PSoC 4100PS)	Up to 53
CapSense Architecture	Third Generation	Third Generation	Third Generation	Third Generation	Third Generation	Third Generation	Third Generation	Third Generation	Fourth Generation	Fourth Generation	Fourth Generation
Sensor Parasitic Capacitance (CP) Range	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 60pF	5pF – 200pF	5pF – 200pF	5pF – 200pF
Number of CapSense Blocks	1	1	1	2	2	1	1	2	1	1	1
Self-Capacitance and Mutual-Capacitance support	Both	Self	Both	Both	Both	Both	Both	Both	Both	Both	Both
CapSense average current consumption per sensor	6 µA	6 µA	6 µA	6 µA	6 µA	6 µA	6 µA	6 µA	3 µA	3 µA	3 µA
SSC / PRS feature	PRS	PRS	PRS	PRS	PRS	PRS	PRS	PRS	SSC and PRS	SSC and PRS	SSC and PRS

Features/Device Family	PSoC 4000-Series	PSoC 4100-Series	PSoC 4200-Series	PSoC 4100 M-Series	PSoC 4200 M-Series	PSoC 4100 BLE-Series	PSoC 4200 BLE-Series	PSoC 4200 L-Series	PSoC 4000 S- Series	PSoC 4100 S-Series/PSoC 4100PS	PSoC 4100S Plus
Liquid Tolerance		Yes									
SmartSense Auto-Tuning	Yes								Yes		
ADC	CSD ADC[17]	12-bit SAR at 806 ksps	12-bit SAR at 1 Msps	12-bit SAR at 806 ksps	12-bit SAR at 1 Msps	12-bit SAR at 806 ksps	12-bit SAR at 1 Msps	12-bit SAR at 1 Msps	10-bit Slope ADC at 46 ksps	10-bit Slope ADC at 46 ksps and 12-bit SAR ADC at 1 Msps	10-bit Slope ADC at 46 ksps and 12-bit SAR ADC at 1 Msps
DAC[18]	1x 8-bit 1x 7-bit	1x 8-bit 1x 7-bit	1x 8-bit 1x 7-bit	2x 8-bit 2x 7-bit	2x 8-bit 2x 7-bit	1x 8-bit 1x 7-bit	1x 8-bit 1x 7-bit	2x 8-bit 2x 7-bit	2x 7-bit	2x 7-bit	2x 7-bit
Comparators	1 with fixed 1.2-V threshold	Up to 4	Up to 4	Up to 6	Up to 6	Up to 2	Up to 4	Up to 6	3	Up to 5 Up to 7 (For PSoC 4100PS)	2
Op-Amps	0	Up to 2	Up to 2	Up to 4	Up to 4	2	4	Up to 4	0	Up to 2 Up to 4 Op-Amps/PGAs (For PSoC 4100PS)	Up to 2
Programmable Voltage Reference (PVref)	No	No	No	No	No	No	No	No	No	No Up to 4 Channels (For PSoC 4100PS only)	No
Voltage DAC (VDAC)	No	No	No	No	No	No	No	No	No	No 13-bit VDAC (For PSoC 4100PS only)	No
Serial Communication Block (SCB[19])	1 (I2C only[20])	2	2	4	4	2	2	4	2	3	5

[17] CapSense block is repurposed as an ADC to measure the input voltage of 0-5 V with the result in mV. When this ADC is used, CapSense is not available for capacitive sensing.

[18] DAC with current output, except PSoC 4000S PSoC 4100S, and PSoC 4100PS. Each CSD block uses 1x 8-bit DAC in single IDAC mode and 1x 8-bit and 1x 7-bit IDACs in dual-IDAC mode. See PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide to learn more about these modes. In PSoC 4000S, PSoC 4100S, and PSoC 4100PS, the CSD block uses 1x 7-bit DAC in single IDAC mode and two 1x 7-bit IDACs in dual-IDAC mode.

[19] Each Serial Communication Block (SCB) can function as an I^2C, an SPI, or a UART communication block. In addition, PSoC Creator provides a Software Transmit UART (TX8) Component for all devices.

[20] The 16-pin QFN device in this family supports the I^2C bus voltage to be different from the device operating voltage.

Features/Device Family	PSoC 4000-Series	PSoC 4100-Series	PSoC 4200-Series	PSoC 4100 M-Series	PSoC 4200 M-Series	PSoC 4100 BLE-Series	PSoC 4200 BLE-Series	PSoC 4200 L-Series	PSoC 4000 S- Series	PSoC 4100 S-Series/PSoC 4100PS	PSoC 4100S Plus
Universal Digital Block (UDB[21])	0	0	Up to 4	0	4	0	4	8	0	0	0
TCPWM[22]	1	4	4	8	8	4	4	8	5	5 8 (For PSoC 4100PS)	8
Segment LCD Drive	No	Up to 4 commons, 32 segments		Up to 4 commons, 51 segments		Up to 4 commons, 32 segments		Up to 8 commons, 64 segments	Up to 8 commons, 36 segments		Up to 4 commons, 49 segments
Bluetooth Low Energy (BLE)	No					Yes		No	No	No	No
CAN 2.0[23]	No	No	No	No	Yes	No	No	Yes	No	No	Yes
USB	No	No	No	No	No	No	No	Yes	No	No	No
Operating Voltage	1.71-5.5 V										
Automotive Qualified (AEC-Q100)[24]	Yes			No							
Package	8/16-pin SOIC 16/24-pin QFN 28-pin SSOP 16-pin WLCSP[25]	28-pin SSOP 40-pin QFN 44-pin TQFP 48-pin LQFP 35-pin WLCSP[25]		48-pin TQFP 64-pin TQFP 68-pin QFN		56-pin QFN 68-pin WLCSP[25] 76-pin WLCSP[25]		124-BGA 68-QFN 64-pin TQFP 48-pin TQFP 48-pin TQFP-USB	48-pin TQFP 24-pin QFN 32-pin QFN 25-pin WLCSP	For PSoC 4100S, 48-pin TQFP 40-pin QFN 32-pin QFN 35-pin WLCSP For PSoC 4100PS, 28-pin SSOP 45-pin WLCSP 48-pin TQFP 48-pin QFN	44-pin TQFP 64-pin TQFP

[21] Universal Digital Block (UDB) allows implementing custom digital logic. PSoC Creator provides a wide array of UDB-based Components such as I^2C, I^2S, UART, SPI, PWM, and Counter/Timer.

[22] Timer, Counter, PWM block

[23] Controller Area Network

[24] Contact Cypress to know the latest status of these products

[25] Wafer-Level Chip-Scale Package

Table 4-5. PSoC 5LP Family Features Comparison

Features/Device Family	PSoC 5200	PSoC 5400	PSoC 5600	PSoC 5800
Datasheet	CY8C52XX	CY8C54XX	CY8C56XX	CY8C58XX
CPU and Speed	ARM Cortex-M3 clocked up to 80 MHz			
Flash, SRAM, EEPROM	Up to 256 KB, Up to 64 KB, 2 KB			
Total I/Os	Up to 72			
CapSense I/Os (Supports Button, Slider, Proximity, Shield)	Up to 62			
Number of CapSense Blocks	2			
Liquid Tolerance	Yes			
SmartSense Auto-Tuning	Yes			
ADC	1x 12-bit SAR	1x 12-bit SAR or 1x 12-bit Del-Sig	2x 12-bit SAR or 1x 12-bit Del-Sig and 1x 12-bit SAR	2x 12-bit SAR and 1x 20-bit Del-Sig
DAC	1x 8-bit	2x 8-bit	4x 8-bit	4x 8-bit
Comparators	2	4	4	4
Op-Amps	0	2	4	4
SC/ST Analog Block[26]	0	2	4	4
Universal Digital Block (UDB[27])	Up to 24			
16-bit Timer/PWM	4			
Digital Filter Block (DFB[28])	No	No	Yes	Yes
Segment LCD Drive	Up to 46x16 segments			
USB 2.0 Full-Speed Peripheral	Yes			
CAN 2.0[29]	No	No	Yes	Yes
Package	68-pin QFN 100-pin TQFP 99-pin WLCSP[30]			
Automotive Qualified (AEC-Q100)	No			
Operating Voltage	1.71-5.5V			

[26] Switched Capacitor/Continuous Time blocks, which are programmable to work as Opamp, Unity Gain Buffer, Programmable Gain Amplifier, Transimpedance Amplifier (TIA), etc.

[27] Universal Digital Block (UDB) allows implementing custom digital logic. PSoC Creator provides wide variety of UDB based components such as I2C, I2S, UART, SPI, LIN Slave, PWM, Counter/Timer etc.

[28] Digital Filter Block (DFB) is programmable to perform IIR and FIR digital filters and several custom functions. It is capable of implementing filters with up to 64 taps and of 48-bit single-cycle multiply-accumulate (MAC) operation.

[29] Controller Area Network

[30] Wafer-Level Chip-scale Package

Table 4-6. PSoC 3 Family Features Comparison

Features/Device Family	PSoC 3200	PSoC 3400	PSoC 3600	PSoC 3800
Datasheet	CY8C324X	CY8C346X	CY8C366X	CY8C386X
CPU and Speed	Single-Cycle 8051 clocked up to 50 MHz		Single-Cycle 8051 clocked up to 67 MHz	
Flash, SRAM, EEPROM	Up to 64 KB, Up to 8 KB, Up to 2 KB			
Total I/Os	Up to 72			
CapSense I/Os (Supports Button, Slider, Proximity, Shield)	Up to 62			
Number of CapSense Blocks	2			
Liquid Tolerance	Yes			
SmartSense Auto-Tuning	Yes			
ADC	1x 12-bit Del-Sig	1x 12-bit Del-Sig	1x 12-bit Del-Sig	1x 20-bit Del-Sig
DAC	1x 8-bit	2x 8-bit	Up to 4x 8-bit	Up to 4x 8-bit
Comparators	2	4	Up to 4	Up to 4
Op-Amps	0	2	Up to 4	Up to 4
SC/ST Analog Block[31]	0	2	Up to 4	Up to 4
Universal Digital Block (UDB[32])	Up to 24			
16-bit Timer/PWM	4	4	Up to 4	4
Digital Filter Block (DFB[33])	No	No	Yes	Yes
Segment LCD Drive	Up to 46x16 segments			
USB 2.0 Full-Speed Peripheral	Yes			
CAN 2.0[34]	No	Yes	Yes	Yes
Operating Voltage	1.71-5.5V			
Automotive Qualified (AEC-Q100)	No			
Package	48-pin SSOP 48-pin QFN 68-pin QFN 100-pin TQFP 72-pin WLCSP[35]	48-pin SSOP 48-pin QFN 68-pin QFN 100-pin TQFP	48-pin SSOP 48-pin QFN 68-pin QFN 100-pin TQFP 72-pin WLCSP[35]	48-pin SSOP 48-pin QFN 68-pin QFN 100-pin TQFP 72-pin WLCSP[35]

[31] Switched Capacitor/Continuous Time blocks, which are programmable to work as Op-Amp, Unity Gain Buffer, Programmable Gain Amplifier, and Transimpedance Amplifier (TIA), etc.

[32] Universal Digital Block (UDB) allows implementing custom digital logic. PSoC Creator provides a wide array of UDB based components such as I2C, I2S, UART, SPI, LIN Slave, PWM, Counter/Timer etc.

[33] Digital Filter Block (DFB) is programmable to perform IIR and FIR digital filters and several custom functions. It is capable of implementing filters with up to 64 taps and of 48-bit single-cycle multiply-accumulate (MAC) operation.

[34] Controller Area Network

[35] Wafer-Level Chip-scale Package

Table 4-7. PSoC 1 Family Features Comparison

Features/Device Family	CY8C20XX7/S	CY8C21X34/B	CY8C20XX6A/S	CY8C20XX6H	CY8C24X94	CY8C22X45	CY8C28XXX	CY8C20X34[36]
Datasheet	CY8C20XX7/S	CY8C21X34/B	CY8C20XX6A/S	CY8C20XX6H	CY8C24X94	CY8C22X45	CY8C28XXX	CY8C20X34
CPU and Speed	24 MHz M8C							12 MHz M8C
Flash/SRAM	Up to 32 KB/ Up to 2 KB	8 KB/512 Bytes	Up to 32 KB/ Up to 2 KB	Up to 16 KB/ Up to 2 KB	16 KB/1 KB	16 KB/1 KB	16 KB/1 KB	8 KB/ 512 Bytes
CapSense I/Os (Supports Buttons, Sliders, Proximity)	31	24	33	33	44	37	41	25
Number of CapSense Blocks	1	1	1	1	1	2	Up to 2	1
Liquid Tolerance	Yes	Yes	No	No	Yes	Yes	Yes	No
SmartSense Auto-Tuning	Yes (only on -S parts)	Yes (only on –B parts)	Yes	Yes	No	Yes	Yes	No
Comparators	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes
ADC	-	8- and 10-bit	8- and 10-bit	8- and 10-bit	7- to 13-bit	10-bit SAR	Up to 14-bit	-
DAC	-	-	-	-	6,8 and 9-bit	8-bit	Up to 9-bit	-
Amplifiers	-	-	-	-	Yes	-	Yes	-
I2C	Master and Slave Interface							
SPI								
UART	Transmitter - Software	UART	Transmitter - Software	Transmitter - Software	UART	UART	UART	Transmitter - Software
USB	-	--	-	Full Speed USB	Full Speed USB	-	-	-
Timer	16-bit timer	8- to 32-bit timer/counter	16-bit timer	16-bit timer	8- to 32-bit timer/counter	8- to 32-bit timer/counter	8- to 32-bit timer/counter	13-bit timer
PWM	-	8- to 32-bit deadband option	-	-	8- to 32-bit	8- to 32-bit	8- and 16-bit	-
LCD	16x2 character LCD interface and Up to 4 commons segment LCD drive							
EEPROM	Emulation							
Haptics	-	-	-	Yes	-	-	-	-
Operating Voltage	1.71 V–5.5 V	2.4 V–5.25 V	1.71 V–5.5 V	1.71 V–5.5 V	3.0 V-5.25 V	3.0 V-5.25 V	3.0 V-5.25 V	2.4 V-5.25 V
Automotive Qualified (AEC-Q100)	No	Yes	Yes	No	Yes	Yes	No	Yes
Package	16/24/32/48-pin QFN 16-pin SOIC 30-pin WLCSP[37]	16-pin SOIC 20/28/56-pin SSOP 32-pin QFN	16/24/32/48-pin QFN 48-pin SSOP 30-pin WLCSP[37]	24-pin QFN 32-pin QFN	56-pin QFN 68-pin QFN	28-pin SOIC 44-pin TQFP	20/28-pin SSOP 44-pin TQFP 48-pin QFN	8/16-pin SOIC 28-pin SSOP 16/24/32-pin QFN 30-pin WLCSP[37]

[36] Not recommended for new designs

[37] Wafer-Level Chip-scale Package

5. CapSense Resources

Cypress provides many resources to simplify the CapSense design process. To make it easier for you to find what you need from this rich selection, this section organizes the available documentation based on a typical workflow for a CapSense design. Table 5-1 is a quick reference for finding the correct documentation.

Table 5-1. CapSense Resources Navigator

I Want To	Where To Go?
Evaluate CapSense	1. In this document, you can learn about - Capacitive Touch Sensing Method - CapSense Tuning - CapSense Widgets - Liquid Tolerance - Proximity Sensing 2. Buy a Development Kit. Use the CY3280-MBR3 kit if you want to quickly develop a CapSense solution without firmware development. Use CY8CKIT-040 to evaluate the complete programmability of the CapSense family of devices. 3. Go through the code examples provided with any of our Development Kits. The user guide has step-by-step procedures for running the code examples.
Select a CapSense Part	1. See CapSense Selector Guide in this guide. 2. Once you have selected a part, see the device-specific Design Guide to help with your design.
Design CapSense Hardware	1. Watch the CapSense layout best practices videos – Part 1 and Part 2. 2. See the device-specific Design Guide for schematic, layout, and overlay design guidelines, sample schematics and layouts, and other design considerations. 3. Find all schematic symbols and footprints for Cypress controllers at this page. 4. Visit the webpages of different development kits listed in Development Kit for the example schematic and layout files. 5. You can use Altium Designer sensor patterns such as a button, or slider, to make your layout design easier. See this page to learn more. 6. Submit your design for review through Cypress Tech Support. You need to register at Cypress website to be able to contact tech support. Cypress recommends PDF prints for the schematic and Gerber files with layer information for the layout.
Configure a CapSense MBR[1] Device	1. Download the EZ-Click software tool. 2. See the device-specific Design Guide for configuration instructions. Design guides also present other methods of configuration.

I Want To	Where To Go?
Develop Firmware	This step is applicable only to the programmable[2] CapSense devices. 1. Download a development environment. Use PSoC Creator for PSoC 3, PSoC 4, PSoC 5LP systems. Use PSoC Designer for PSoC 1 systems. For PSoC 6 systems: • Use PSoC Creator or ModusToolbox for CY8C62x6, CY8C62x7 devices • Use PSoC Creator or ModusToolbox for CY8C63xx devices • Use ModusToolbox for CY8C62x8, CY8C62xA devices • Use ModusToolbox for CY8C62x5 devices See Software Tools to learn more. 2. Study the example code. Use the code examples provided with the Kits and the IDEs. See the CapSense Controller Code Examples document. Implement dynamic reconfiguration with based CapSense devices, AN49079 - CapSense Plus: Dynamically Configuring CapSense 3. See the CapSense Component Datasheet API guide 4. Optimize the code. See these application notes to learn how: AN89610 - PSoC 4 and PSoC 5LP ARM Cortex Code Optimization AN60630 - PSoC 3 8051 Code and Memory Optimization AN60486 - PSoC 1 M8C ImageCraft C Code Optimization
Tune a CapSense Design	1. Enable SmartSense[3] to reduce or eliminate manual tuning. 2. See the CapSense Manual Tuning section in the device-specific Design Guide. 3. See AN2397 - CapSense Data Viewing Tools to learn how to monitor and log real-time sensor data over I2C or UART.
Design CapSense Proximity Sensing	1. In this document, you can learn about Proximity Sensing 2. Evaluate proximity sensing using CY8CKIT-024 CapSense Proximity Shield along with a PSoC 4 pioneer kit. 3. See Design CapSense Hardware in this table above to help with your schematic and layout design. 4. Explore additional information on proximity design guidelines, AN92239 - Proximity Sensing with CapSense. This application note also provides the code examples.
Design CapSense Liquid Level Sensing	1. Evaluate liquid level sensing using CY8CKIT-022 CapSense Liquid Level Sensing Shield along with a PSoC 4 pioneer kit. 2. Explore information on theory, sensor layout, and other system design guidelines, AN202478 - PSoC 4 - Capacitive Liquid-Level Sensing. 3. Go through the code example CE202479.
Design Wireless Touch Solutions with PRoC-CapSense Devices	AN88890 - Adding Touch-Sensing User Interfaces to Wireless HID Products Using PRoC-CS AN86272 - PRoC-CS Hardware Design Guidelines

I Want To	Where To Go?
Design for Low Power	AN210998 - PSoC 4 Low Power CapSense Design CE95288 - CapSense Low Power with PSoC 4 AN90114 - PSoC 4000 Family Low-Power System Design Techniques AN86233 - PSoC 4 Low-Power Modes and Power Reduction Techniques AN77900 - PSoC 3 and PSoC 5LP Low-power Modes and Power Reduction Techniques AN47310 - PSoC 1 Power Savings Using Sleep Mode CapSense Controller Code Examples for PSoC 1 Devices See the device-specific Design Guide for more low power techniques.
Solve ESD, EMC, and EFT Issues	AN96475 - Design Considerations for Electrical Fast Transient Immunity of a CapSense System AN80994 - Design Considerations for Electrical Fast Transient (EFT) Immunity AN72362 - Reducing Radiated Emissions in Automotive CapSense Applications See the device-specific Design Guide for ESD and EMC design considerations.
Implement Class-B Safety Functions	AN89056 - PSoC 4 – IEC 60730 Class B and IEC 61508 SIL Safety Software Library AN78175 - PSoC 3 and PSoC 5LP - IEC 60730 Class B Safety Software Library AN79973 - PSoC 3 and PSoC 5LP CapSense CSD - IEC 60730 Class B Safety Software Library AN81828 - PSoC 1 – IEC 60730 Class B Safety Software Library
Program the CapSense Controller In-System From an External Host	For programming through a bootloader: AN73854 - PSoC 3, PSoC 4, PSoC 5LP Introduction to Bootloaders AN97060 - PSoC 4 BLE and PRoC BLE - Over-The-Air (OTA) Device Firmware Upgrade (DFU) Guide AN2100 - Bootloader: PSoC 1 For programming through a dedicated programming interface: AN84858 - PSoC 4 Programming Using an External Microcontroller AN73054 - PSoC 3 and PSoC 5LP Programming Using an External Microcontroller AN44168 - PSoC 1 Device Programming using External Microcontroller AN59389 - Host Sourced Serial Programming for CY8C20xx6A, CY8C20xx6AS, CY8C20xx6L and CY8C20xx7/S
Learn CapSense Design Tips and Tricks	See the extensive Knowledge Base Articles on the Cypress website. Use the **Product Category** filters to narrow the results.

[1] MBR stands for Mechanical Button Replacement. CapSense MBR devices are configurable but require no firmware development. See the CapSense Selector Guide for details.

[2] Programmable devices support various programmable digital and analog peripherals in addition to CapSense and require firmware development. See the CapSense Selector Guide for details.

[3] Cypress' SmartSense Auto-Tuning algorithm automatically sets sensing parameters for optimal performance and continuously compensates for system, manufacturing and environmental changes.

5.1 CapSense Design Guides and Application Notes

Our technical documentation includes special CapSense Design Guides. Design guides contain all the information such as schematic and layout guidelines, tuning steps, ESD and EMC design considerations, and other information required to complete a CapSense design with a particular part family. Also listed below are the Getting Started Applications Notes for the various device families.

Design Guides for PSoC 3, PSoC 4, and PSoC 5LP Devices

- PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide
- 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 Notes for PSoC 1, PSoC 3, PSoC 4, and PSoC 5LP

- AN75320 - Getting Started with PSoC® 1
- AN54181 - Getting Started with PSoC® 3
- AN79953 - Getting Started with PSoC® 4
- AN77759 - Getting Started with PSoC® 5LP

5.2 Additional CapSense Resources

5.2.1 Cypress Document Manager

The Cypress Document Manager (CDM) is an intuitive, Windows-based tool allowing users to navigate, filter, search, view, and manage Cypress documentation. CDM can directly open PDF and html pages. It allows you to select documents to download locally and to easily check for updates to those documents.

5.2.2 Website

At the Cypress CapSense Controllers website, you can access all the reference materials such as datasheets, technical reference manuals (TRM), application notes, and code examples. Click on the resource types listed on the right side of the

web page under Design Support as shown in Figure 5-1. Use the search box at the top right of Cypress website to search for a resource. Select the type of collateral you want, and type in your key words, as shown in Figure 5-2.

Figure 5-1. Design Support in Cypress Website

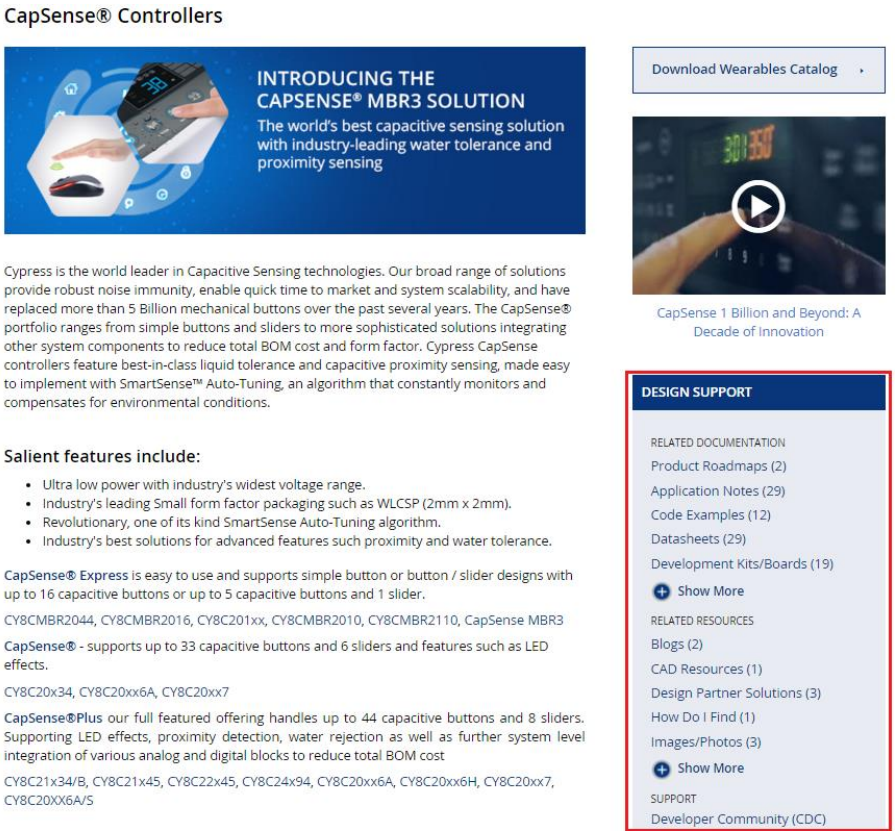
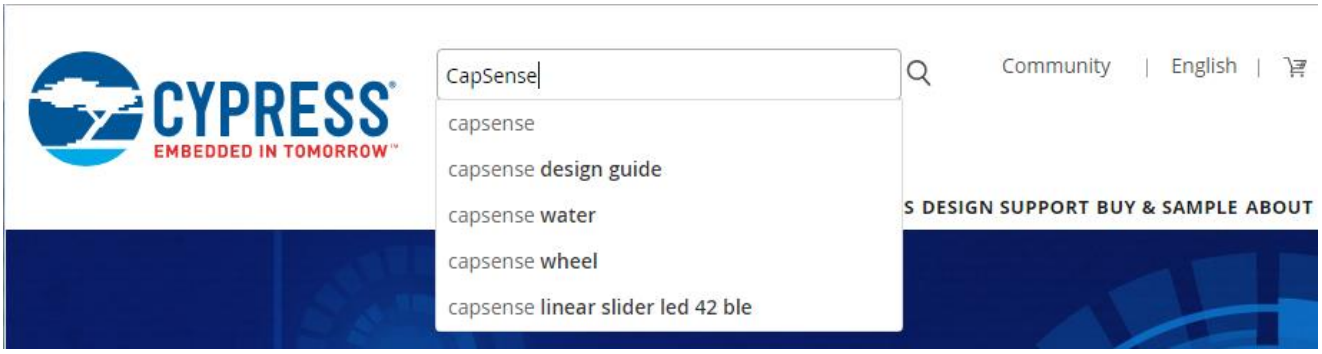


Figure 5-2. Cypress Web Search



5.3 Software Tools

5.3.1 Integrated Development Environments

5.3.1.1 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 simple CapSense Buttons and a Linear Slider example to get started. You can also see the CapSense® Tuner Guide for tuning your CapSense design.

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

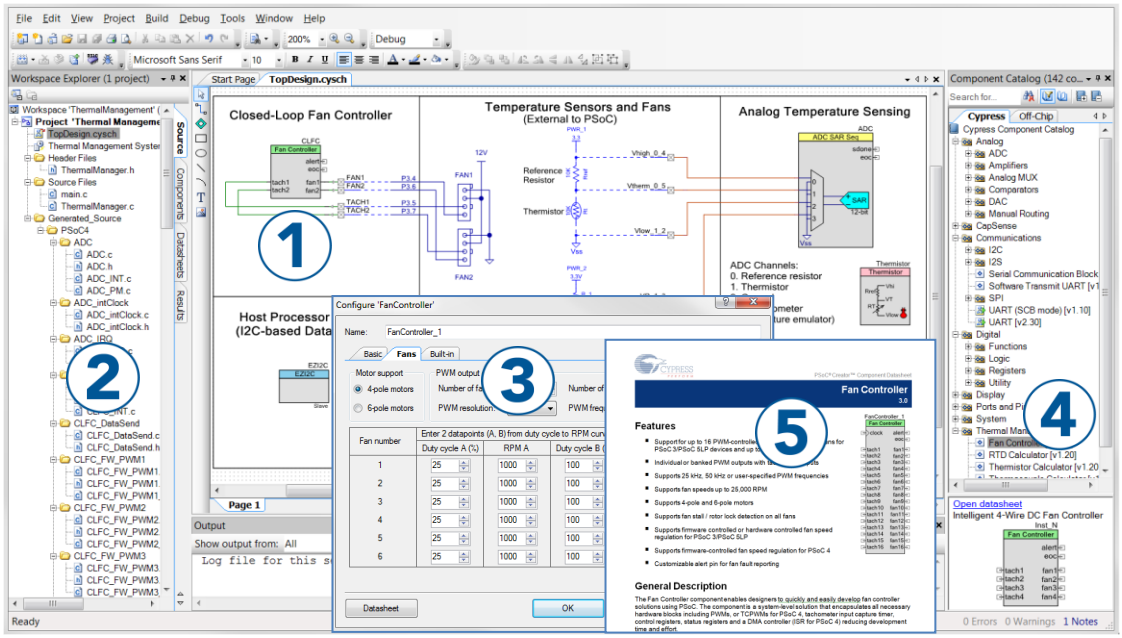
Note that PSoC Creator does not support all the PSoC 6 devices. The CY8C62x8 and CY8C62xA device families are supported in ModusToolbox only. The CY8C6xx7 device family is supported in ModusToolbox and PSoC Creator 4.2.

5.3.1.2 PSoC Creator

PSoC Creator is a free Windows-based Integrated Development Environment (IDE). It enables concurrent hardware and firmware design of PSoC 3, PSoC 4, and PSoC 5LP systems. See Figure 5-3. With PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace
2. Co-design your application firmware with the PSoC hardware
3. Configure Components using configuration tools
4. Explore the library of 100+ Components
5. Review Component datasheet

Figure 5-3. PSoC Creator Features



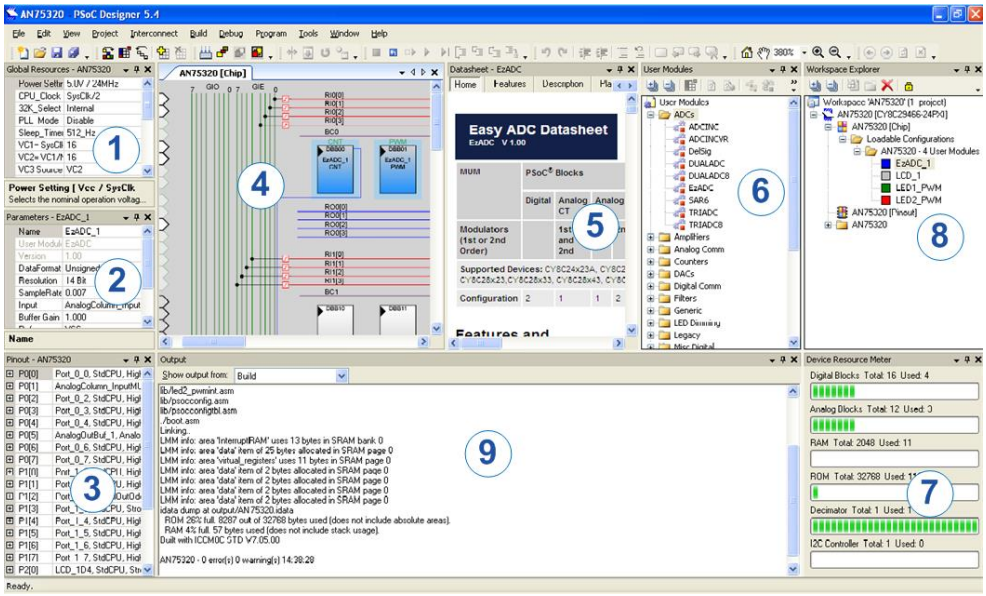
5.3.1.3 PSoC Designer

PSoC Designer is a free Windows-based Integrated Design Environment (IDE) offered by Cypress for developing PSoC 1 systems. Develop your applications using a library of pre-characterized analog and digital peripherals in a drag-and-drop design environment. Then, customize your design leveraging the dynamically generated API libraries of code. Figure 5-4 shows the PSoC Designer windows. **Note:** This is not the default view.

1. **Global Resources –** all device hardware settings.
2. **Parameters –** the parameters of the currently selected User Modules
3. **Pinout –** information related to device pins
4. **Chip-Level Editor –** a diagram of the resources available on the selected chip
5. **Datasheet –** the datasheet for the currently selected UM
6. **User Modules –** all available User Modules for the selected device
7. **Device Resource Meter –** device resource usage for the current project configuration
8. **Workspace –** a tree-level diagram of files associated with the project
9. **Output –** output from project build and debug operations

Note: For detailed information on PSoC Designer, go to **PSoC® Designer** > **Help** > **Documentation** > **Designer Specific Documents** > **IDE User Guide**.

Figure 5-4. PSoC Designer Layout



5.3.1.4 Programmer

PSoC Programmer is a Windows-based, stand-alone application used to program PSoC devices using the hardware tools MiniProg3, MiniProg4, or KitProg, the on-board programmer integrated with the development kits. PSoC Programmer also provides APIs to develop your own application utilizing Cypress' programmers and bridge devices in C, C#, Perl, and Python languages. See the PSoC Programmer Component Object Model (COM) Interface Guide available under *Program Files (x86)\Cypress\Programmer\Documents* folder for more details. PSoC Creator and PSoC Designer provide in-window programming interface that in turn invokes the PSoC Programmer APIs.

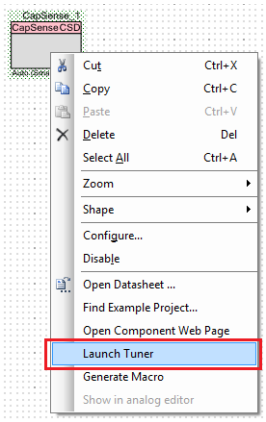
5.3.2 Data Monitoring Tools

These tools help you to debug, monitor, and tune CapSense designs. For more details on the tools, see the application note AN2397 – CapSense Data Viewing Tools.

5.3.3 CapSense Tuner

CapSense Tuner is a graphic user interface (GUI) tool available in PSoC Creator to debug and tune CapSense applications over I2C. To launch the Tuner GUI, right-click the CapSense symbol placed in the schematic editor and select **Launch Tuner**, as shown in Figure 5-5.

Figure 5-5. Launching Tuner



5.3.4 EZ-Click™

EZ-Click is a simple yet powerful software tool that enables development of CapSense MBR[a] solutions. The tool allows you to configure, debug, monitor real time sensor output, and run production-line system diagnostics of all register-configurable CapSense MBR families.

5.3.5 Bridge Control Panel

Bridge Control Panel is a Windows-based software tool installed along with PSoC Programmer. This tool is used along with MiniProg3 or MiniProg4 to monitor and log serial communication data from I^2C slave devices. This tool can also receive data from UART devices.

5.4 Development Kits

5.4.1 PSoC 4 Development Kits

5.4.1.1 Pioneer Kits

The pioneer kits are an easy-to-use and inexpensive development platform that support Arduino™ compatible shields.

- CY8CKIT-040 Pioneer Kit for PSoC 4000 devices
- CY8CKIT-042 Pioneer Kit for PSoC 4100/4200 devices
- CY8CKIT-044 Pioneer Kit for PSoC 4 M-Series devices
- CY8CKIT-046 Pioneer Kit for PSoC 4 L-Series devices
- CY8CKIT-041 Pioneer Kit for PSoC 4 S-Series devices

5.4.1.2 Shield Kits

Shield kits are designed to be Arduino compatible and to work with the PSoC Pioneer kits.

- CY8CKIT-022 CapSense Liquid Level Sensing Shield
- CY8CKIT-024 CapSense Proximity Shield

5.4.1.3 Prototyping Kits

Prototyping Kits are low-cost platforms for prototyping products using PSoC 4 devices.

- PSoC® 4 CY8CKIT-049 4xxx Prototyping Kits
- CY8CKIT-043 PSoC® 4 M-Series Prototyping Kit
- CY8CKIT-145-40XX PSoC® 4000S CapSense Prototyping Kit
- CY8CKIT-147 PSoC 4100PS Prototyping Kit
- CY8CKIT-149 PSoC 4100S Plus Prototyping Kit

5.4.2 PSoC 3 and PSoC 5LP Development Kits

- CY8CKIT-030 PSoC® 3 Development Kit
- CY8CKIT-050B PSoC® 5LP Development Kit
- CY8CKIT-059 PSoC® 5LP Prototyping Kit
- CY8CKIT-001 PSoC® Development Kit

The CY8CKIT-031 PSoC CapSense Expansion Board Kit connects any of the PSoC 3 and PSoC 5LP Development Kits to any of the Universal CapSense Module Boards (listed in PSoC 1 Development Kits).

[a] MBR stands for Mechanical Button Replacement. CapSense MBR devices include CY8CMBR3xxx and CY8CMBR2xxx families. See the CapSense Selector Guide for details.

5.4.3 CapSense Express Development Kits

- CY3280-MBR3 Evaluation Kit for CY8CMBR3xxx family
- CY3280-MBR for CY8CMBR2044 device
- CY3218-CAPEXP1 CapSense Express Kit for CY8C20110[a] device
- CY3218-CAPEXP2 CapSense Express Kit for CY8C201A0mm device

5.4.4 PSoC 1 Development Kits

5.4.4.1 Universal CapSense Controller Kits

- CY3280-BK1 Universal CapSense Controller - Basic Kit 1 for CY8C20x34, CY8C21x34 devices
- CY3280-20x66 Universal CapSense Controller Kit for CY8C20xx6A devices

5.4.4.2 Universal CapSense Modules

These modules implement different types of CapSense sensors and are designed to use with any universal CapSense Controller Kit and also with PSoC 3/5LP kits with the help of CY8CKIT-031.

- CY3280-BSM Simple Button Module Kit supporting 10 buttons and 10 LEDs
- CY3280-BMM Matrix Button Module Kit supporting 8 LEDs and 16 buttons organized in 4x4 matrix fashion utilizing only 8 IOs.
- CY3280-SLM Linear Slider Module Kit supporting 5 buttons, 1 linear slider with 10 sensors, and 5 LEDs.
- CY3280-SRM Radial Slider Module Kit supporting 4 buttons, 1 radial slider with 10 sensors, and 4 LEDs.
- CY3280-BBM Universal CapSense Prototyping Module Kit supporting breadboard access to every signal routed to the 44-pin connector that can plug into any of the universal CapSense controller kits.

5.4.5 PSoC 6 Development Kits

- PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-062-4343W)
- PSoC® 6 BLE Pioneer Kit (CY8CKIT-062-BLE)
- PSoC® 6 WiFi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT)
- PSoC 62S3 Wi-Fi BT Prototyping Kit (CY8CPROTO-062S3-4343W)

5.4.6 Kits for Programming and Debugging

5.4.6.1 Miniprog3

The CY8CKIT-002 PSoC MiniProg3 Program and Debug Kit is an all-in-one programmer for PSoC 1, PSoC 3, PSoC 4, and PSoC 5LP architectures, a debug tool for PSoC 3, PSoC 4, and PSoC 5LP architectures, and a USB-I2C Bridge for communicating with PSoC devices. Other than for programming, this device is mainly used as a USB-to-I2C Bridge in data monitoring and tuning of CapSense solutions.

5.4.6.2 CY3215-DK Kit

CY3215-DK Kit is the debug tool for PSoC 1 devices and contains an In-Circuit Emulator (ICE) that manages all the emulation communication between the debugger software running in PSoC Designer and the PSoC 1 on-chip debugger (OCD) enabled chip. This kit is used along with the pod kits that contain a PSoC 1 OCD chip. The available pod kits are listed in Appendix D. See the application note AN73212 – Debugging with PSoC 1 for more details.

5.4.6.3 Miniprog4

The CY8CKIT-005 MiniProg4 Program and Debug Kit is an all-in-one development programmer and debugger for PSoC 4, PSoC 5LP, and PSoC 6 MCU devices. MiniProg4 is used as a programmer or debug probe for supported Cypress devices. and development kits,

[a] Not recommended for new designs. See Configurable CapSense Controllers (CapSense Express Family) for more details.

5.5 Design Support

Cypress has several support channels to ensure the success of your CapSense design:

- Cypress Developer Community – Connect with the Cypress technical community and exchange information.
- Video Library – Get up to speed with tutorial videos.
- Cypress Design Partner Program – An expansion of our engineering capabilities providing customers with access to design services and solutions from trusted and capable partners.
- Technical Support – Excellent technical support is available online.
- Quality and Reliability – Cypress is committed to complete customer satisfaction. At our Quality website, you can find reliability and product qualification reports.

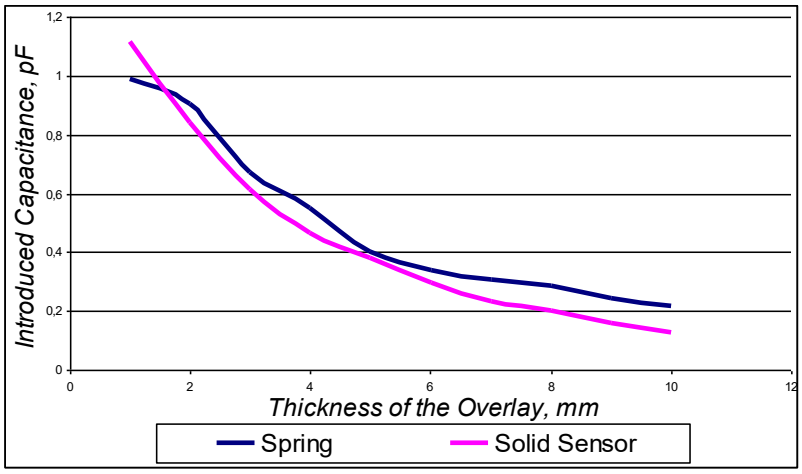
A. Springs

A.1 Finger-Introduced Capacitance

This section gives the influence of various physical parameters on finger-introduced capacitance in a CapSense design with springs.

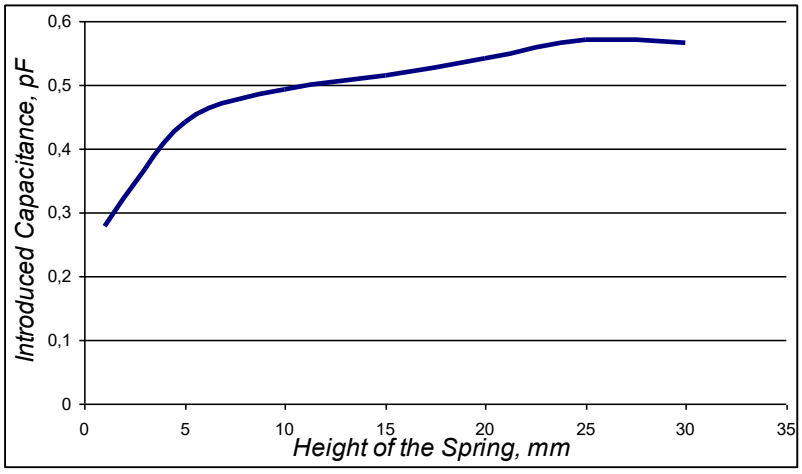
- Influence of overlay thickness on Finger Touch added Capacitance (FTC) with springs is similar to that with solid sensors

Figure 5-6. FTC Versus Overlay Thickness



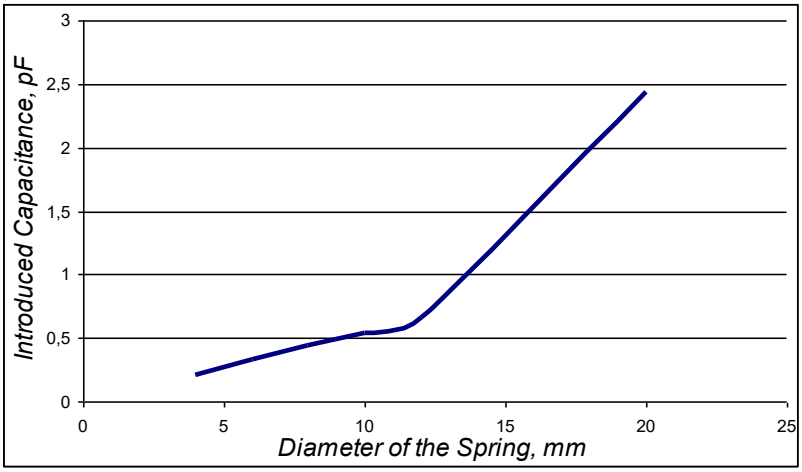
- Influence of height on FTC

Figure 5-7. FTC Versus Spring Height



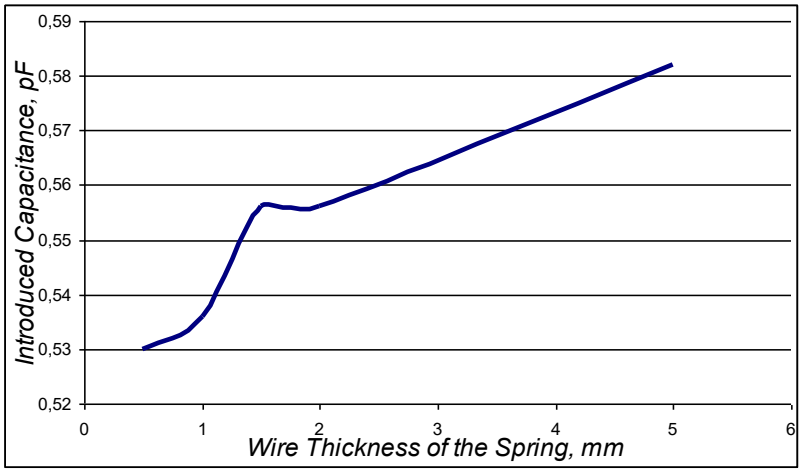
- Influence of diameter on FTC

Figure 5-8. FTC Versus Spring Diameter



- Influence of wire thickness of the spring on FTC

Figure 5-9. FTC Versus Wire Thickness of Spring



A.1.1 Mounting Springs to the PCB

Figure 5-10 shows an example of spring mounting. This section discusses how to design spring sensors. Because springs have higher side sensitivity, the neighboring spring sensors must be placed as far as possible from each other to prevent false detections. Add a comparison level if the sensor pitch is small.

The requirements for the sensitive area of a spring are the same as the requirements for solid buttons. When using thick overlays, the spring diameter must be larger than the overlay thickness by at least 2 or 3 times. The distance between the PCB and the overlay must be 5 mm or more.

Figure 5-10. Spring-Mounting Example

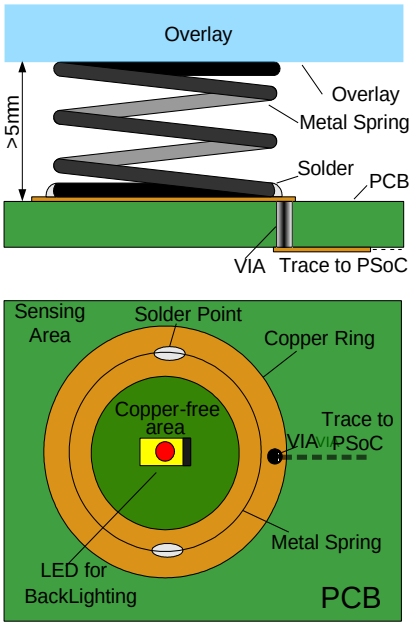
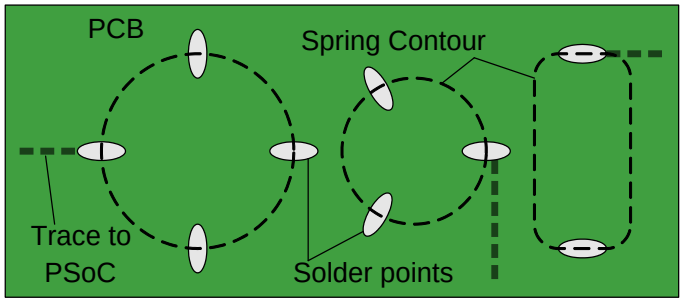


Figure 5-11 shows examples of footprints for springs. Unlike a button sensor, which is a copper pad on the PCB surrounded by the ground hatch, it is not possible to surround a spring with ground as its surface is above the PCB. However, you can still provide the ground hatch near the footprint of the spring similar to buttons with an air gap of 1 mm between the sensor ring and the ground. See Ground Plane for details on the ground hatch fill.

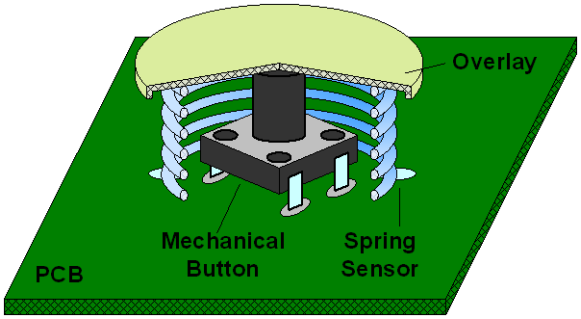
Figure 5-11. Proposed Spring Footprints



A.2 CapSense and Mechanical Button Combination

The hollow space inside a spring can also be used as a mechanical button, as shown in Figure 5-12.

Figure 5-12. CapSense and Mechanical Button Combination



Touching such a button only triggers the sensor, while pressing the button activates both the sensor and mechanical button. In this case, preparatory actions such as backlighting, prompt showing, and others are possible only if the sensor works. The final action is performed when both buttons work. For example, in a GPS navigation system, touching a button shows only a hint and pressing the button takes an action.

A.3 Design Examples

Figure 5-13 and Figure 5-14 show project demonstrator examples for white goods applications.

Figure 5-13. Demo Cooktop

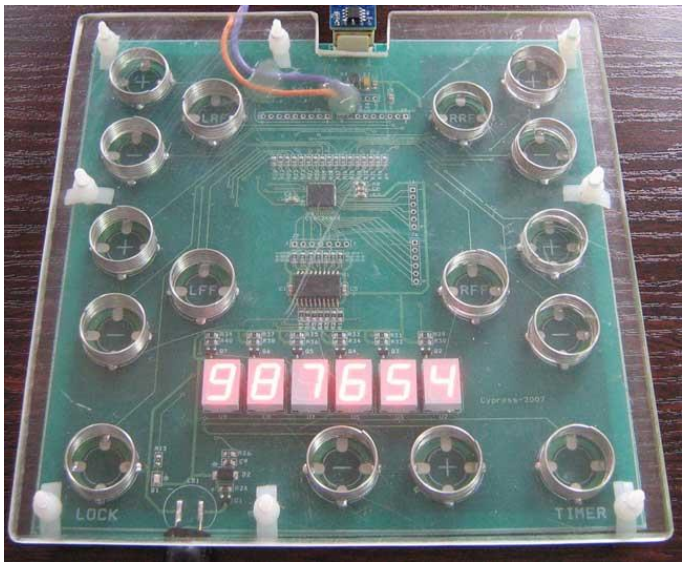


Figure 5-14. Cooktop Front Panel

B. Schematic and Layout Checklist

B.1 Schematic Checklist

See the PSoC 3 datasheet for PSoC 3 devices, PSoC 4 datasheet for PSoC 4 devices, PSoC 5LP datasheet for PSoC 5LP devices, and PSoC 6 datasheet for PSoC 6 devices.

No.	Category	Recommendations/Remarks
1	Decoupling capacitor	0.1 µF
2	Bulk capacitor	1 µF
3	Pin Assignment	Sensors should be placed near to ground. No switching signal should be placed near to sensors
4	CMOD	2.2 nF Ensure that CMOD is not adjacent to any switching or communication pins
5	C_{INTA} and C_{INTB}	470 pF Ensure that C_{INTA} and C_{INTB} are not adjacent to any switching or communication pins
6	Sensor pin selection	If possible, avoid pins that are close to the GPIOs carrying switching/communication signals. Physically separate DC loads such as LEDs and 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. **Note**: For PSoC 6 family devices, to achieve the best CapSense sensitivity and accuracy, follow the recommendations stated in PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide.
7	RB	Ensure that RB is not adjacent to any switching or communication pins
8	Series resistor on CapSense Lines	560 Ω
9	Series resistor on Communication lines	330 Ω
10	Pull-up resistor on Communication lines	4.7 kΩ
11	Avoid using programming pins as I2C pins if possible	

B.1.1 Decoupling Capacitor

See Power Supply Layout Recommendations for complete details.

B.1.2 Bulk Capacitor

See Power Supply Layout Recommendations for complete details.

B.1.3 Pin Assignment

- Distribute the LEDs evenly among even and odd pins such as P2[0], P2[2] and P2[1], P2[3]
- Try not to keep LEDs next to CapSense pins.

- Reserve pins for R_B (if required), C_{MOD} and Shield tank capacitors (if required).
- It is recommended that you keep CapSense pins near the ground pin. Otherwise, the increase in the impedance of the ground path will cause the drive circuit's reference voltage to shift.
- LEDs or any switching signal should not be placed close to C_{MOD}/R_B pin to avoid crosstalk.

See Pin Assignments for more details.

B.1.4 C_{MOD}

Ensure that the C_{MOD} is not adjacent to any switching/communication pin. Since there is an analog signal on C_{MOD}, it should be surrounded by the CapSense (analog) signal rather than being surrounded by the switching/communication (digital) signal. Ground of C_{MOD} should have a least possible path to device ground.

Family	C_{MOD} Value Recommended
CY8C20xx6/A/AS	2.2 nF for CSD 1.2 nF – 5.6 nF for CSA
CY8C21x34	5.6 nF-10 nF for PRS8 and PRS16 configuration. 22nF- 47nF for Prescaler Configuration
CY8C21x34 SmartSense,	10 nF
CY8C24x94, CY8C22x45	5.6 nF-10 nF
CY8C20x34	1.2 nF – 5.6 nF
CY8CMBR3xxx, CY8CMBR2xxx, CY8C20xx7A/S, PSoC3/4/5LP	2.2 nF

See the User Module and Component datasheets for more details. The User Module and Component datasheets get downloaded when you install PSoC Designer and PSoC Creator respectively.

B.1.5 R_B

Ensure that the R_B is not adjacent to any switching/communication pin.

Family	R_B Value Recommended
CY8C21x34	Minimum of 2 kΩ
CY8C21x34 SmartSense	15 kΩ

B.1.6 Series Resistor on CapSense Lines

A 560-ohm resistor on CapSense signal lines. If the series resistance value is set larger than 560 ohms, the slower time constant of the switching circuit limits the amount of charge that can transfer. This lowers the signal level, which in turn lowers SNR. Smaller values are better, but are less effective at blocking RF. For complete details, see Series Resistor.

B.1.7 Series Resistor on Communication Lines

A 330-Ω resistor is recommended on communication lines. If more than 330 Ω is placed in series on these lines, the voltage levels fall out of specifications with the worst-case combination of the supply voltages between systems and the input impedance of the receiver. 330 Ω will not affect the I^2C operation as the V_{IL} level still remains within the I^2C specification limit of 0.3 VDD when PSoC outputs a LOW. For complete details, see Series Resistor.

B.2 Layout Checklist

No.	Category		Min	Max	Recommendations
1	Buttons	Shape	NA	NA	Solid round or rectangle with curved edges
		Diameter/Diagonal	5 mm	15 mm	10 mm
		Air gap between button and hatch	0.5 mm	2 mm	Should be equal to overlay thickness. Hatch can be connected to ground or shield.
		Placement near any switching element	NA	NA	Isolate switching signals from sensor and the sensor PCB traces.
2	Slider	Width of segment for 1 mm acrylic overlay thickness	2 mm		8 mm
		Width of segment for 3 mm acrylic overlay thickness	4 mm		
		Width of segment for 4 mm acrylic overlay thickness	6 mm		
		Air gap between segments	0.5 mm	2 mm	0.5 mm
		Air gap between hatch and slider	0.5 mm	2 mm	Equal to overlay thickness. Hatch can be connected to ground or shield.
		Height of segment	7 mm	15 mm	12 mm
3	Overlay	Type			Use material with a high dielectric constant except conductors. There should be no air gap between sensor board and overlay/Front panel of the casing.
		Thickness for acrylic overlay		5 mm	
		Thickness for glass overlay		15 mm	Overlay thickness should be low. This is applicable to both buttons and sliders. For proximity sensors, thicker overlays increase the capacitance coupled with human hand/finger and the sensor and hence increases the signal. However, the parasitic capacitance might increase by a small amount.
4	Sensor traces	Width		7 mil	
		Length		300 mm for a standard (FR4) PCB 50 mm for flex PCB.	
		Air gap between ground and sensor traces	10 mil	20 mil	
		Turns			No sharp turns
		Routing			Should be routed on non-sensor side. If any non CapSense trace crosses CapSense trace; ensure that intersection is orthogonal.
5	Via on sensors	Number of vias	1	2	1(including sensor trace and sensor pad)
		Via diameter			10 mil
6	Ground				Use hatch ground which reduces parasitic capacitance. Typical hatching: 25% on the top layer (7 mil line, 45 mil spacing), 17% on the bottom layer (7 mil line, 70 mil spacing)
7	Series resistor				Place resistor within 10 mm of CapSense controller pin

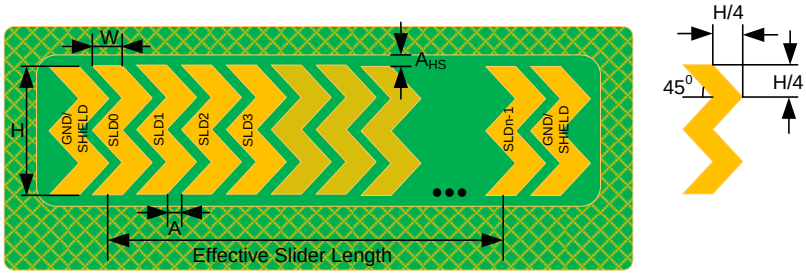
No.	Category		Min	Max	Recommendations
8	Shield electrode	Size of shield fill		Width of 1 cm around sensors	See Figure 3-75. Drive the shield only when required such as in applications requiring liquid tolerance and proximity sensing.
		Shield pattern			Use hatch fill instead of solid fill to reduce the parasitic capacitance of the shield electrode and to reduce the emissions. The hatching specifications are same as that of ground hatching. For detailed guidelines for reducing emissions, see Shield electrode.
9	Guard Sensor	Shape			Rectangular trace with curved edges
		Thickness			Recommended thickness of copper trace is 2 mm and distance of copper trace to hatch(ground/shield) is 1 mm.

B.2.1 Buttons

- The best shape for CSD buttons is round. Rectangular shapes with rounded corners are also acceptable. Because sharp points concentrate fields, avoid sharp corners (less than 90º) when designing your sensor pad. Refer Figure 3-49 for details.
- The best geometry to implement the CSX button is the fishbone structure. See Mutual Cap Buttons of Fishbone Structure to get idea of some common fishbone patterns, or contact Cypress.
- Size can range from 5 mm to 15 mm. Go for larger diameter for thicker overlays.
- Air gap to ground and other sensors should be equal to the overlay thickness, but no smaller than 0.5 mm, and no larger than 2 mm. The spacing between the two adjacent buttons should be large enough that if one button is touched, a finger should not reach the air gap of the other button.
- Placement near any switching elements
  - ☐ Reduce the trace length from controller pins.
  - ☐ Mount series resistors within 10 mm of the controller pins.
  - ☐ Avoid connectors between sensors and other controller pins.

B.2.2 Slider

Figure B-1. Typical Liner Slider Pattern



- Keep the width of the segment (W) at 8 mm for an average finger width of 9 mm.
- Keep the air gap between segments (A) at 0.5 mm.
- Keep the height of the segment (H) at 12 mm.
- Keep the air gap between the hatch and the slider (A_{HS}) equal to the overlay thickness.
- For a slider with 'n' segments, employ n+2 segments. The first and last segments of the slider should be grounded or shielded depending on the application.

See Slider Design for more details.

B.2.3 Overlay

- Type (material): Do not use conductive materials as overlay since it interferes with the electric field pattern. Select a material with a higher dielectric constant.
- Overlay thickness should be kept low to have high signal. Sensor signal is directly proportional to the finger capacitance. Finger capacitance is inversely proportional to the thickness of the overlay which is the distance between the two electrodes (sensor pad and finger) which together form a parallel plate capacitor. Hence reducing the overlay thickness increases the signal.
- For sliders, with SmartSense, the maximum acrylic overlay thickness is 4 mm and that of glass is 12 mm.

B.2.4 Sensor Traces

- Keep the width less than or equal to 7 mil (0.18 mm)
- Keep the maximum trace length as 12 inches (300 mm) for a standard PCB and 2 inches (50 mm) for flex circuits.
- Keep the air gap between a CapSense trace and ground in the range of 10 mil to 20 mil (0.25 mm to 0.51 mm).
- Do not have sharp (90 degrees) turns as this will pick up noise; in addition, charges concentrate at sharp corners.
- Routing of traces (See Trace Routing for details)
  - ☐ Route sensor traces on the bottom layer of the PCB.
  - ☐ Do not run traces underneath a sensor unless the sensor and the trace are connected.
  - ☐ Do not route sensor traces in parallel to noisy clock and LED lines. Use ground or shield around sensor traces for shielding.
  - ☐ Do not run sensor traces in close proximity to communication lines, such as I2C or SPI masters. If it is necessary to cross communication lines with sensor trace, make sure that the intersection is at right angles.

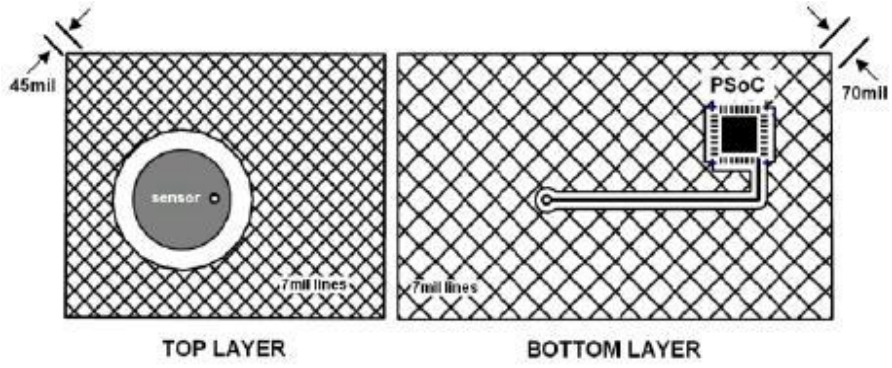
B.2.5 Vias on Sensors

- Placement should be at the edges of the CapSense button to minimize increase in C_p due to vias.
- Minimize the number of vias to reduce the parasitic capacitance, at up to two on a sensor (trace and pad).
- Keep the via hole diameter for sensor traces at 10 mil (See Figure 3-69 for details).

B.2.6 Ground Plane/Mesh

Placing solid ground around sensors and decreasing the air gap between the sensor and the surrounding ground reduce noise, but increase the parasitic capacitance. Thus, there is a tradeoff between the CapSense signal and noise immunity. Hatching the ground decreases the ground area and therefore the parasitic capacitance.

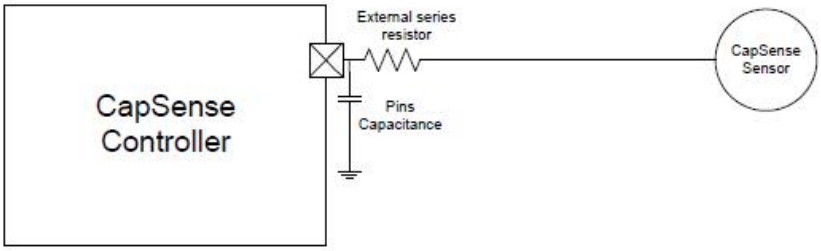
Figure B-2. Typical Ground Fill



B.2.7 Series Resistor

Place series resistors within 10 mm of the CapSense controller pins. Adding an external resistor forms a low-pass RC filter that can dampen RF noise amplitude.

Figure B-3. Series Resistor Placement



B.2.8 Shield Electrode

a. Reduce the size of the shield fill (maximum 1 cm from the sensor).
b. Limit the placement of the shield to only the selected sensors.
c. Slow the edges of the shield waveform to reduce emissions:
   - Adding a capacitor filter between the shield electrode port pin and ground will reduce slew rate.
   - Put a small value of series resistor

For more details on shield design considerations for emission reduction, see Shield .

B.2.9 Guard Sensor

a. The shield electrode should surround the guard sensor pad and exposed traces, and spread no further than 10 mm from these features.
b. The recommended shape for a guard sensor is rectangular trace with curved edges.
c. The recommended thickness of a copper trace is 2 mm and distance of the copper trace to the shield hatch is 1 mm.

C. Clearance Between Sensor and Ground

The ground plane is placed on the same layer of the board as the buttons as shown in Figure C-1. The clearance between the button and ground plane plays an important role in the performance of the button. Electric field lines fringing between a button and the ground plane are illustrated in Figure C-2. The parasitic capacitance of the sensor, C_p, is related to this electric field.

Figure C-1. CapSense Board Top and Bottom Layer

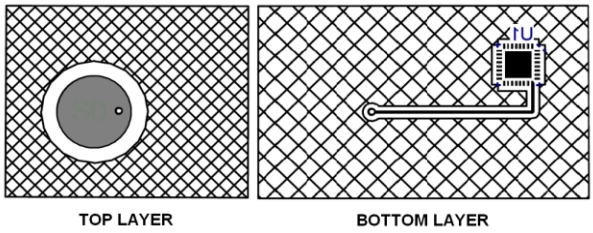
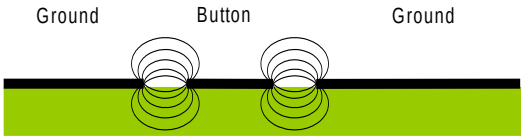


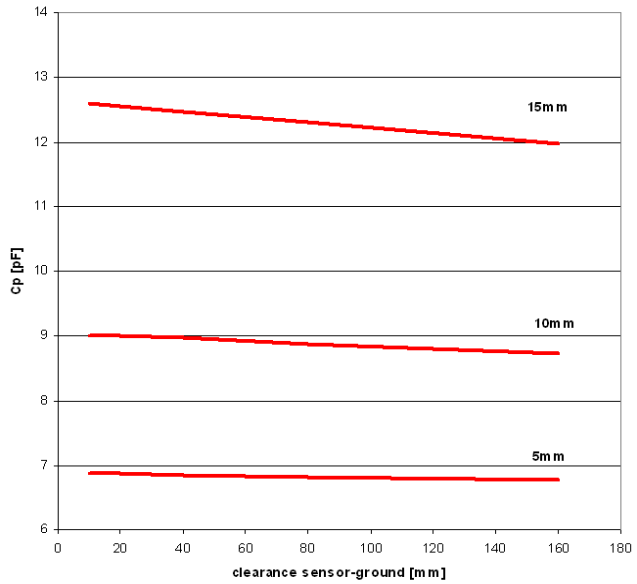
Figure C-2: Button-Ground Plane Fringing Fields



The capacitance, C_P, decreases as the clearance surrounding the button increases. An example of this dependence of C_P on the gap is shown in Figure C-3 through Figure C-6. In these plots, the board material is FR4 with a thickness of 62 mils (1.57 mm), and the acrylic overlay has a thickness of 2 mm. Each plot contains data for three button sizes (5 mm, 10 mm, and 15 mm diameter).

The C_P in Figure C-3 does not include the effect of the traces or vias. It is only the parasitic capacitance of the sensor pad.

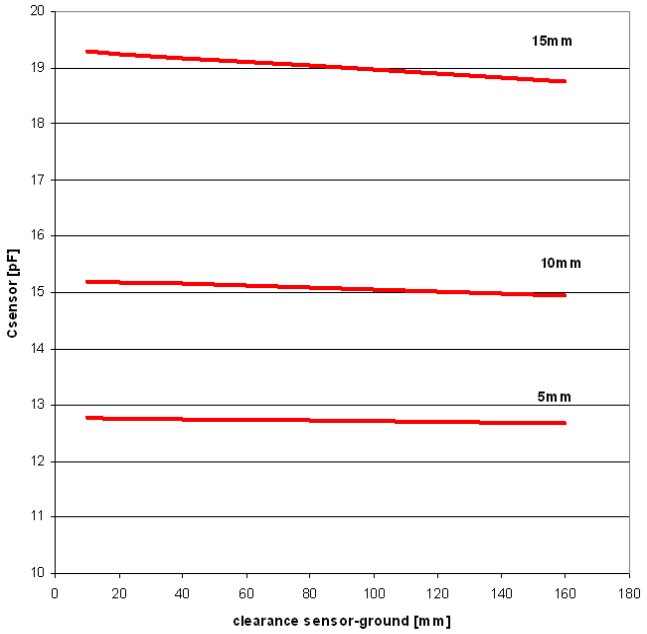
Figure C-3. Parasitic Capacitance, C_p, as a Function of Button-Ground Clearance and Button Diameter



Note: The finger is not on the sensor. Capacitance increases with sensor size, but decreases with the gap.

The capacitance, C_{sensor}, is the total sensor capacitance when the finger is not on the sensor. It includes the effect of the sensor pad, the traces, and vias. Figure C-4 shows the sensor capacitance for a board routed with 50-mm trace length, 8-mil (0.3 mm) trace width, and 20-mil (0.8 mm) spacing from trace to the coplanar ground.

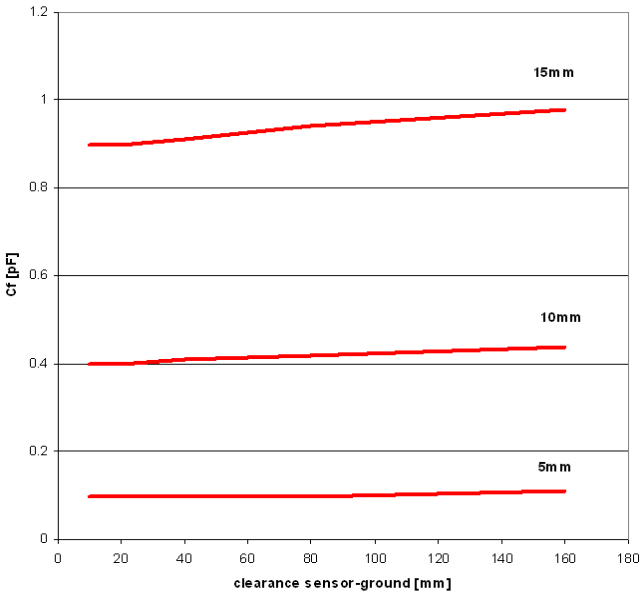
Figure C-4. Sensor Capacitance, C_{sensor}, as a Function of Button-Ground Clearance and Button Diameter



Note: The finger is not on the sensor. Sensor capacitance decreases with the size of the gap.

The capacitance, C_F, in Figure C-5 is the capacitance added by the touch of the finger. Total capacitance of the sensor pad and the finger is Cp + Cf.

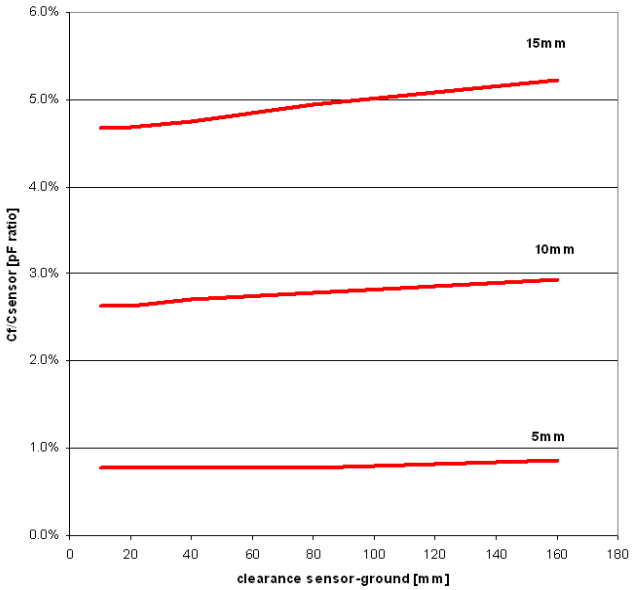
Figure C-5. Finger Capacitance, C_f, as a Function of Button-Ground Clearance and Button Diameter



Note: The finger is on the sensor. Capacitance increases with both sensor size and gap.

Figure C-6 plots finger capacitance as a percentage of the sensor capacitance. This is the sensitivity of the sensor. The sensitivity of the system changes with the routing of the CapSense traces. For example, increasing the trace length between the PSoC and the sensor pad decreases the button's sensitivity.

Figure C-6. Sensitivity, C_f/C_{sensor} as a Function of Button-Ground Clearance and Button Diameter



Note: The sensitivity increases with both button size and gap.

D. PSoC 1 In-Circuit Emulation (ICE) Pods

D.1 Evaluation Pods

PSoC EvalPods are pods that connect to the CY3215-DK In-Circuit Emulator (ICE) kit to allow debugging capability. They can also function as a standalone device without debugging capability. The EvalPod has a 28-pin DIP footprint on the bottom for easy connection to development kits or other hardware. The top of the EvalPod has prototyping headers for easy connection to the devices pins. The following evaluation pods are available.

- CY3210-CY8C20X34 PSoC Evaluation Pod (EvalPod)
- CY3210-CY8C21X34 PSoC Evaluation Pod (EvalPod)
- CY3210-CY8C20X36/46/66 PSoC Evaluation Pod (EvalPod)
- CY3210-CY8C24X94 PSoC Evaluation Pod (EvalPod)

D.2 In-Circuit Emulation (ICE) Pod Kits

The ICE pod provides the interconnection between the CY3215-DK ICE kit via a flex cable and the target PSoC device in a prototype system or PCB via package-specific pod feet. The user guide and the quick start guide for the ICE kit are available here. Following are the pods available. Note that some of the following pod kits are not in stock but a web page is kept for accessing the documents. Contact Cypress technical support if you need help with a particular kit.

- CY3250-21X34QFN ICE Pod Kit to debug QFN CY8C21X34 PSoC devices
- CY3250-24X94QFN ICE Pod Kit to debug QFN CY8C24X94 PSoC devices
- CY3250-20246QFN ICE Pod Kit to debug CY8C20236/46A/46AS PSoC devices
- CY3250-20346QFN ICE Pod Kit to debug CY8C20336/346A/346AS CapSense PSoC devices
- CY3250-20666QFN ICE Pod Kit to debug CY8C20636/646/666A/646AS/666AS CapSense PSoC devices
- CY3250-20566 ICE Pod Kit to debug CY8C20536/546/566A CapSense PSoC devices
- CY3250-20466QFN ICE Pod Kit to debug CY8C20436/46/66/46AS/66AS CapSense PSoC devices
- CY3250-20334QFN ICE Pods (2) to debug QFN CY8C20334 PSoC devices

The replacement ICE pods are available here.

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, which is the industry's leading solution in sales.

CapSense Mechanical Button Replacement (MBR)

Cypress's configurable solution to upgrade mechanical buttons to capacitive buttons. It 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.

CINTA and CINTB Capacitors

Two external capacitors required for the operation of a CSX block in Mutual-Capacitance sensing mode.

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 the 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.

CSX

CapSense Crosspoint (CSX) is a Cypress-patented method of performing mutual-capacitance (also called mutual-cap) measurements for capacitive sensing applications. In the CSX mode, the sensing system measures the capacitance between two electrodes, and a change in the 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 the user touches 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 Linear Slider, Radial Slider and Touchpad Widgets.

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, crisscross lines looking like a mesh and the line width and the spacing between two lines determine the fill percentage. In case of liquid tolerance, this hatch fill (referred as a shield electrode) is driven with a shield signal instead of ground.

Hysteresis

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

IDAC (Current-Output Digital-to-Analog Converter)

Programmable constant current source available inside PSoC, used for CapSense and ADC operations.

Liquid Tolerance

The ability of a capacitive sensing system to work reliably in the presence of liquid droplets, streaming liquids, or mist.

Linear Slider

A widget consisting of more than one sensor arranged in a specific linear fashion to detect the physical position (in single axis) of a finger.

Low Baseline Reset

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

Manual-Tuning

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

Matrix Buttons

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

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

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

Modulation Capacitor (CMOD)

An external capacitor required for the operation of a CSD block in Self-Capacitance sensing mode.

Modulator Clock

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

Modulation IDAC

Modulation IDAC is a programmable constant current source, whose output is controlled (ON/OFF) by the sigma-delta modulator output in a CSD block to maintain the AMUXBUS voltage at V_{REF}. The average current supplied by this IDAC is equal to the average current drawn out by the sensor capacitor.

Mutual-Capacitance

Capacitance associated with an electrode (say TX) with respect to another electrode (say RX) is known as mutual capacitance.

Negative Noise Threshold

A threshold used to differentiate usual noise from the spurious signals appearing in negative direction. This parameter is used in conjunction with the Low Baseline Reset parameter.

Baseline is updated to track the change in the Raw Count as long as the Raw Count stays within Negative Noise Threshold, that is, the difference between Baseline and Raw count (Baseline – Raw count) is less than Negative Noise Threshold.

Scenarios that may trigger such spurious signals in a negative direction include: a finger on the sensor on power-up, removal of a metal object placed near the sensor, removing a liquid-tolerant CapSense-enabled product from the water; and other sudden environmental changes.

Noise (CapSense Noise)

The variation in the Raw Count when a sensor is in the OFF state (no touch), measured as peak-to-peak counts.

Noise Threshold

A parameter used to differentiate signal from noise for a sensor. If Raw Count – Baseline is greater than Noise Threshold, it indicates a likely valid signal. If the difference is less than Noise Threshold, Raw Count contains nothing but noise.

Overlay

A non-conductive material, such as plastic and glass, which covers the capacitive sensors and acts as a touch-surface. The PCB with the sensors is directly placed under the overlay or is connected through springs. The casing for a product often becomes the overlay.

Parasitic Capacitance (C_P)

Parasitic capacitance is the intrinsic capacitance of the sensor electrode contributed by PCB trace, sensor pad, vias, and air gap. It is unwanted because it reduces the sensitivity of CSD.

Proximity Sensor

A sensor that can detect the presence of nearby objects without any physical contact.

Radial Slider

A widget consisting of more than one sensor arranged in a specific circular fashion to detect the physical position of a finger.

Raw Count

The unprocessed digital count output of the CapSense hardware block that represents the physical capacitance of the sensor.

Refresh Interval

The time between two consecutive scans of a sensor.

Scan Resolution

Resolution (in bits) of the Raw Count produced by the CSD block.

Scan Time

Time taken for completing the scan of a sensor.

Self-Capacitance

The capacitance associated with an electrode with respect to circuit ground.

Sensitivity

The change in Raw Count corresponding to the change in sensor capacitance, expressed in counts/pF. Sensitivity of a sensor is dependent on the board layout, overlay properties, sensing method, and tuning parameters.

Sense Clock

A clock source used to implement a switched-capacitor front-end for the CSD sensing method.

Sensor

See Capacitive Sensor.

Sensor Auto Reset

A setting to prevent a sensor from reporting false touch status indefinitely due to system failure, or when a metal object is continuously present near the sensor.

When Sensor Auto Reset is enabled, the Baseline is always updated even if the Difference Count is greater than the Noise Threshold. This prevents the sensor from reporting the ON status for an indefinite period of time. When Sensor Auto Reset is disabled, the Baseline is updated only when the Difference Count is less than the Noise Threshold.

Sensor Ganging

See Ganged Sensors.

Shield Electrode

Copper fill around sensors to prevent false touches due to the presence of water or other liquids. Shield Electrode is driven by the shield signal output from the CSD block. See Driven-Shield.

Shield Tank Capacitor (C_{SH})

An optional external capacitor (C_{SH} Tank Capacitor) used to enhance the drive capability of the CSD shield, when there is a large shield layer with high parasitic capacitance.

Signal (CapSense Signal)

Difference Count is also called Signal. See Difference Count.

Signal-to-Noise Ratio (SNR)

The ratio of the sensor signal, when touched, to the noise signal of an untouched sensor.

Slider Resolution

A parameter indicating the total number of finger positions to be resolved on a slider.

Touchpad

A Widget consisting of multiple sensors arranged in a specific horizontal and vertical fashion to detect the X and Y position of a touch.

Trackpad

See Touchpad.

Tuning

The process of finding the optimum values for various hardware and software or threshold parameters required for CapSense operation.

V_{REF}

Programmable reference voltage block available inside PSoC used for CapSense and ADC operation.

Widget

A user-interface element in the CapSense component that consists of one sensor or a group of similar sensors. Button, proximity sensor, linear slider, radial slider, matrix buttons, and touchpad are the supported widgets.

Revision History

Document Revision History

Document Title: AN64846 - Getting Started with CapSense

Document Number: 001-64846

Revision	Issue Date	Description of Change
**	12/17/2010	New guide
*A	03/04/2011	Multiple chapter enhancements for content and reader clarity
*B	08/16/2011	Multiple section and table updates
*C	12/07/2011	Multiple chapter enhancements for content clarity
*D	04/27/2012	Updated slider section. Updated PCB layout guidelines. Updated table 5-2. Corrected phone number in the title page.
*E	07/19/2012	Updated sample schematics and layouts. Included information on layout/trace routing guidelines for C_{MOD}, R_B pin.
*F	08/31/2012	Updated references to external documents
*G	10/16/2012	Updated Section 3.1 (Overlay Selection) and moved Table 3-1 to Section 3.2.
*H	01/07/2013	Updated Section 3.1. Added Section 3.7.13.
*I	03/07/2013	Updated Section 2.7.2. Haptic Feedback.
*J	06/07/2013	Added the CY8C20XX7/S part family. Added proximity information.
*K	09/04/2013	Added the CY8C22X45 part family information.
*L	10/04/2013	Corrected document revision on the footer of some pages.
*M	02/19/2014	Added information specific to CY8CMBR3XXX
*N	03/05/2014	Updated the Operating Voltage Specifications in Table 4-7. PSoC 1 Family Features Comparison
*O	09/17/2014	Updated Figure 1-1 to have a mention of PSoC Creator Updated CSD Block Diagram to remove unnecessary bends. Updated Equation 6 to reflect the correct noise definition Updated Liquid Tolerance and Proximity (Three-Dimensional Sensors) Section Updated references to PSoC 3/4/5LP in multiple places Removed reference of CY3235 – Proximity Detection Demonstration Kit In Chapter 4. , added PSoC 3/4/5LP references in multiple places
*P	01/22/2015	Added guidelines on LEDs close to the sensor Added a note on slider guidelines Updated template Changed document title
*Q	06/18/2015	Fixed broken reference

Revision	Issue Date	Description of Change
*R	01/19/2016	Updated Chapter 1 Introduction – added the benefits of CapSense over mechanical buttons.Added CapSense Design Flow. Updated Chapter 2 CapSense Technology. Updated the sections Capacitive Touch Sensing Method, CapSense Tuning, CapSense Widgets, and User Interface Feedback. Added Ground Plane and Proximity Sensing, re-aligned the sections CapSense System Overview and Liquid Tolerance. Added chapters CapSense Selector Guide and CapSense Resources. Added a Glossary of CapSense terms.
*S	02/11/2016	Updated Table 5-1. CapSense Resources Navigator – Added reference to AN202478, AN72362, AN88890, AN86272, and AN49079. Added Mutual cap content in Mutual Capacitance section. Updated CapSense Sigma Delta Modulator (CSD) section with content on fourth generation CapSense. Updated Figure 4-1 to add cortex M0+ for PSoC 4000S and PSoC 4100S family of devices. Updated Table 4-4. PSoC 4 Family Features Comparison with PSoC 4200 L, PSoC 4000S and PSoC 4100S family of devices details. Updated PSoC 4 Development Kits with PSoC 4 Pioneer Kits and Shield Kits
*T	02/24/2016	Updated Table 5-1. CapSense Resources Navigator – Added reference to AN210998, AN96475.
*U	10/26/2016	Fixed a typo Updated template Fixed formatting throughout
*V	04/19/2017	Updated logo and copyright
*W	09/27/2017	Added references to PSoC 4100S Plus throughout the document Updated Table 4-4. PSoC 4 Family Features Comparison with PSoC 4100S Plus features Updated Section 5.4.1 PSoC 4 Development Kits with CY8CKIT-149 PSoC 4100S Plus Prototyping Kit Updated Section 5.1 CapSense Design Guides and Application Notes with links to Getting Started Application Notes
*X	02/28/2018	Added support for PSoC 4100PS throughout the document Updated Introduction chapter with the code example and video link Updated Table 4-4. PSoC 4 Family Features Comparison with additional CapSense features
*Y	02/13/2020	Added information related to PSoC 6 devices Added information on ModusToolbox Added Section 2.3.1 CapSense Sigma Delta Modulator (CSD) Sensing Method Added Section 2.3.2 CapSense Crosspoint (CSX) Sensing Method Added Section 3.8.4.1 Self Cap Button Structure Added Section 3.8.4.2 Mutual Cap Buttons of Fishbone Structure Added a note on CapSense pin selection with GPIO toggling in section B.1 Schematic Checklist Added CSX information in Glossary Added Section 5.4.6.3 Miniprog4 Updated Design CapSense Hardware and Develop Firmware sections of Table 5-1 to include references to different development kits Added Section 3.9 Example Schematic and Layout