AN66269 - CY8C20x34

CapSense® Design Guide

Doc. No. 001-66269 Rev. *J

Cypress Semiconductor

198 Champion Court

San Jose, CA 95134-1709

www.cypress.com

© Cypress Semiconductor Corporation, 2010-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

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

Contents

1. Introduction

1.1 Abstract

This document gives design guidance for implementing capacitive touch sensing (CapSense®) functionality with the CY8C20x34 family of CapSense Controllers. The following topics are covered in this guide:

- Features of the CY8C20x34 family of CapSense devices
- CapSense principles of operation
- Introduction to CapSense design tools
- In depth guide to tuning the CapSense touch sensing system for optimal performance
- Step-by-step CapSense tuning example
- Electrical and mechanical system design considerations for CapSense
- Additional resources and support for designing CapSense into your system

1.2 Cypress's CapSense Documentation Ecosystem

Figure 1-1 and Table 1-1 summarize the Cypress CapSense documentation ecosystem. These resources allow you to quickly access the information you need to successfully complete a CapSense product design. Figure 1-1 shows the typical flow of a product design cycle with capacitive sensing. The information in this guide is most pertinent to the topics highlighted in green. Table 1-1 provides links to the supporting documents for each of the numbered tasks in Figure 1-1.

Figure 1-1. Design Cycle Flow

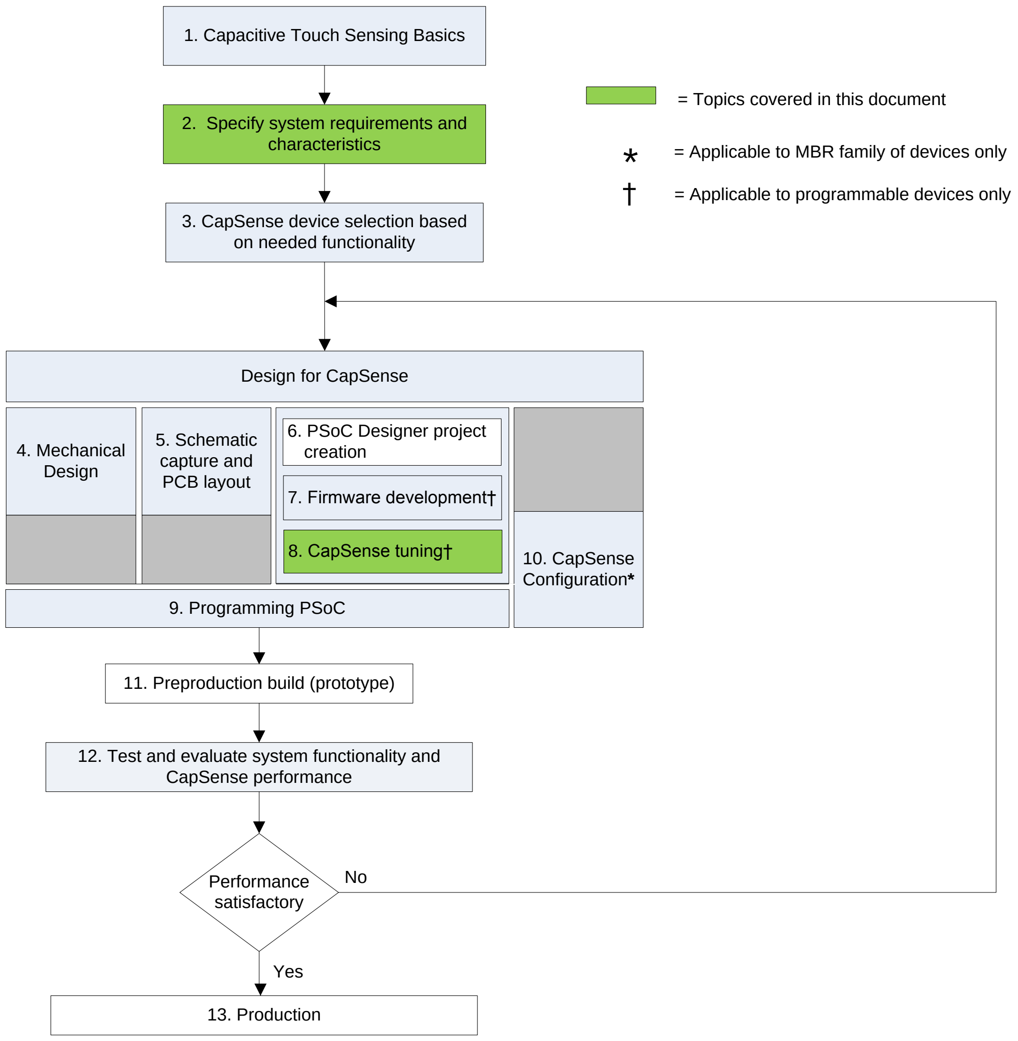

1. Capacitive Touch Sensing Basics
2. Specify system requirements and characteristics
3. CapSense device selection based on needed functionality
= Topics covered in this document
* = Applicable to MBR family of devices only
† = Applicable to programmable devices only
Design for CapSense
4. Mechanical Design
5. Schematic capture and PCB layout
6. PSoC Designer project creation
7. Firmware development†
8. CapSense tuning†
10. CapSense Configuration*
9. Programming PSoC
11. Preproduction build (prototype)
12. Test and evaluate system functionality and CapSense performance
Performance satisfactory
No
Yes
13. Production

Table 1-1. Cypress Documents Supporting Numbered Design Tasks of Figure 1-1

Numbered Design Task of Figure 1-1	Supporting Cypress CapSense® Documentation
1	■ Getting Started with CapSense®
2	■ Getting Started with CapSense® ■ CY8C20x34 CapSense® Device Datasheets ■ PSoC Family Specific CapSense® Design Guide (this document)
3	■ Getting Started with CapSense®
4	■ Getting Started with CapSense®
5	■ Getting Started with CapSense®
6	■ PSoC Designer User Guides

Numbered Design Task of Figure 1-1	Supporting Cypress CapSense®Documentation
7	■ Assembly Language User Guide ■ C Language Compiler User Guide ■ CapSense Code Examples ■ PSoC Family Specific Technical Reference Manual (for CY8C20x34)
8	■ PSoC Family Specific CapSense®Design Guide (this document) ■ PSoC Family Specific CapSense® User Module Datasheets (CSA_EMC) ■ CapSense Data Viewing Tools -AN2397 ■ CapSense Controller Code Examples Design Guide
9	■ Programmer User Guide ■ MiniProg3 User Guide
11	■ Code Examples

1.3 CY8C20x34 CapSense Family Features

Cypress's CY8C20x34 is a low-power, high-performance, programmable CapSense controller family that features:

1.3.1 Advanced Touch Sensing Features

- Programmable capacitive sensing elements
  - Supports a combination of CapSense buttons, sliders, and proximity sensors with CSA_EMC capacitive sensing technology
  - Contains an integrated API to implement buttons and sliders
  - Supports up to 25 capacitive sensors and 6 sliders
  - CY8C20x24 supports 25 capacitive sensors and 1 slider
  - Supports proximity sensing up to 2 cm (with on-board PCB trace)
- Enhanced Noise Immunity
  - CSA_EMC offers superior noise immunity for applications with challenging conducted and radiated noise conditions
- Low Power consumption
  - Two different power modes for optimized power consumption
  - 1.5 mA active mode current; 2.6 uA sleep mode current

1.3.2 Device Features

- High-performance, low-power M8C Harvard-architecture processor
  - Up to 2 MIPS with 12-MHz Internal clock, external clock signal
- Flexible On-chip memory
  - Up to 8 KB of flash and 512 B of SRAM
  - Emulated EEPROM supported
- Precision, programmable clocking
  - Internal main oscillator (IMO): 6/12 MHz ± 5%
  - Internal low speed oscillator at 32 kHz for watchdog and sleep

- Enhanced general-purpose input/output (GPIO) features
  - Up to 28 general-purpose I/Os (GPIOs) with programmable pin configuration
  - 20-mA sink current on all GPIOs
  - Internal restive pull-up, HI-Z, open-drain, and strong drive modes on all GPIOs
- Peripheral features
  - 13-bit programmable timer
  - Watchdog and sleep timers
  - I^2C Master – clocked at 100 kHz; I^2C Slave– up to 400 kHz
  - SPI Master and Slave – Configurable Range of 46.9 kHz to 3 MHz
- Operating Conditions
  - Operating Voltage: 2.4 V to 5.25 V
  - Temperature range: –40 °C to +85 °C

1.4 Document Conventions

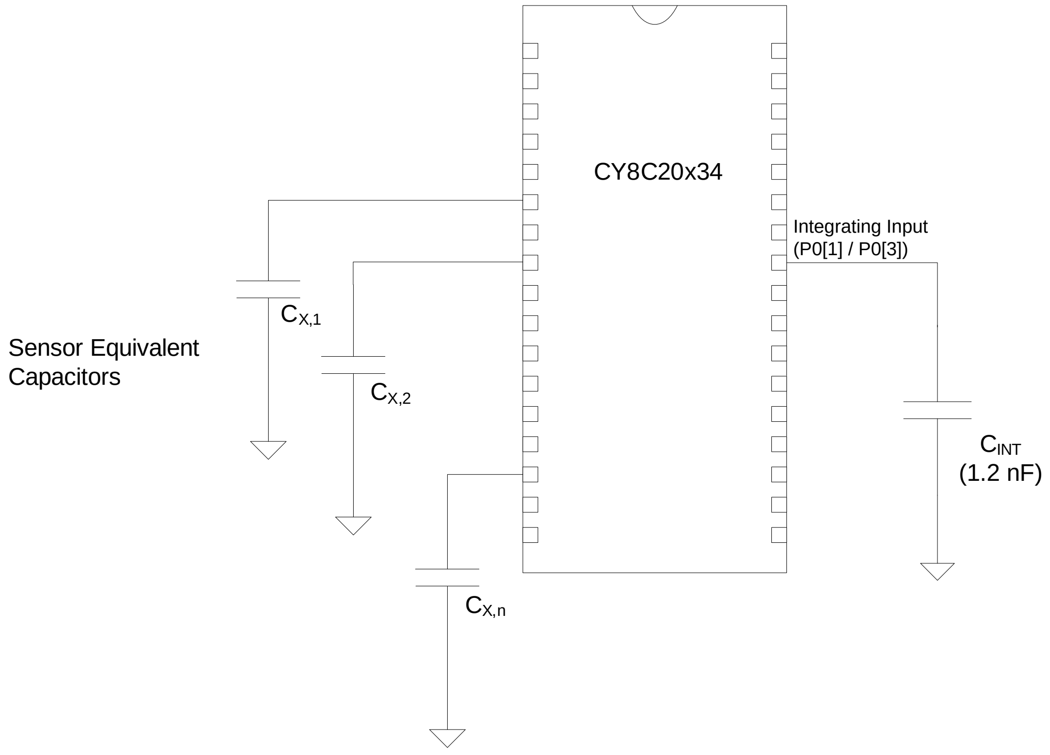
Convention	Usage
`Courier New`	Displays file locations, user entered text, and source code: C:\ ...cd\icc\
Italics	Displays file names and reference documentation: Read about the *sourcefile.hex* file in the *PSoC Designer User Guide*.
[Bracketed, Bold]	Displays keyboard commands in procedures: [**Enter**] or [**Ctrl]** [**C**]
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

2.1 CapSense Fundamentals

CapSense is a touch sensing technology that works by measuring the capacitance of each I/O pin on the CapSense controller that has been designated as a sensor. As shown in Figure 2-1, the total capacitance on each of the sensor pins can be modeled as equivalent lumped capacitors with values of $C_{X,1}$ through $C_{X,n}$ for a design with n sensors. Circuitry internal to the CY8C20x34 device converts the magnitude of each C_X into a digital code that is stored for post processing. The other component, namely Integration Capacitor C_{INT}, is used by the CapSense controller's internal circuitry.

Figure 2-1. CapSense-Specific Device Implementation in a CY8C20x34 PSoC Device



As shown in Figure 2-1, each sensor I/O pin is connected to a sensor pad by traces, vias, or both as necessary. The overlay is a nonconductive cover over the sensor pad that constitutes the product's touch interface. When a finger comes into contact with the overlay, the conductivity and mass of the body effectively introduces a grounded conductive plane parallel to the sensor pad. This is represented in Figure 2-2. This arrangement constitutes a parallel plate capacitor, whose capacitance is given by:

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

Where:

C_F = The capacitance affected by a finger in contact with the overlay over a sensor

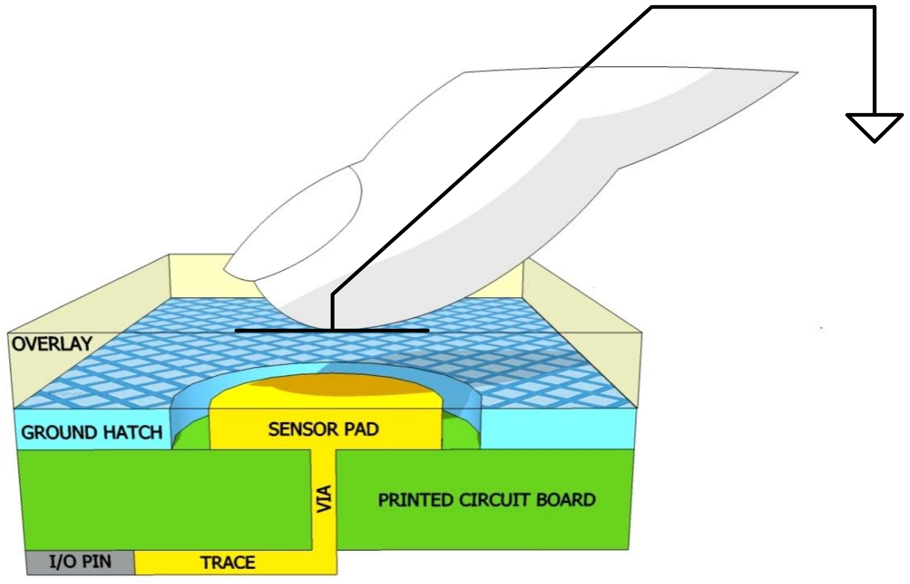
ε_0 = Free space permittivity

ε_r = Dielectric constant (relative permittivity) of overlay

A = Area of finger and sensor pad overlap

D = Overlay thickness

Figure 2-2. Section of Typical CapSense PCB with the Sensor Being Activated by a Finger



In addition to the parallel plate capacitance, a finger in contact with the overlay causes electric field fringing between itself and other conductors in the immediate vicinity. The effect of these fringing fields is typically minor compared to that of the parallel plate capacitor and can usually be ignored.

Even without a finger touching the overlay, the sensor I/O pin has some parasitic capacitance (C_P). C_P results from the combination of the CapSense controller internal parasitics and electric field coupling between the sensor pad, traces, and vias, and other conductors in the system such as ground plane, other traces, any metal in the product's chassis or enclosure, and so on. The CapSense controller measures the total capacitance (C_X) connected to a sensor pin.

When a finger is not touching a sensor:

$$C_X = C_P \qquad \text{Equation 2}$$

With a finger on the sensor pad, C_X equals the sum of C_P and C_F:

$$C_X = C_P + C_F \qquad \text{Equation 3}$$

In general, C_P is an order of magnitude greater than C_F. C_P usually ranges from 6 pF to 15 pF, but in extreme cases can be as high as 50 pF. C_F usually ranges from 0.1 pF to 0.4 pF. The magnitude of C_P is of critical importance when tuning a CapSense system and is discussed in CapSense Performance Tuning with User Modules.

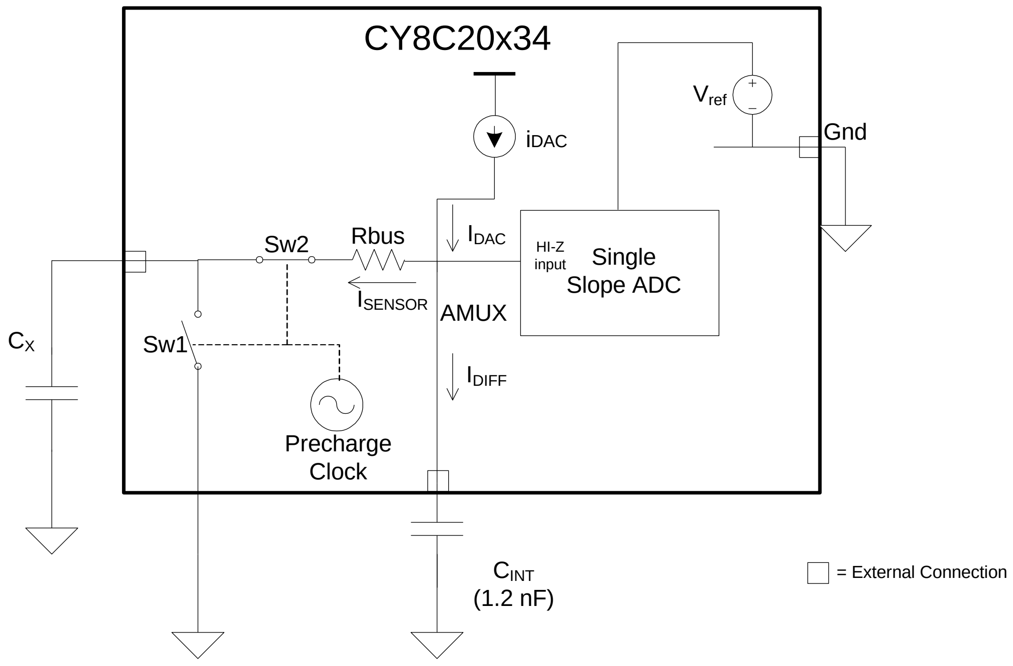
2.2 CapSense Method in CY8C20x34

The CapSense method, CapSense Successive Approximation Electromagnetic Compatibility (CSA_EMC), is supported by the CY8C20x34 device to convert sensor capacitance (C_X) into a digital code. The CSA_EMC method is implemented in the form of a PSoC Designer User Module and is described in the following section.

2.2.1 CapSense Successive Approximation Electromagnetic Compatibility (CSA_EMC)

The CSA_EMC method used in CY8C20x34 devices incorporates C_X into a switched capacitor circuit as shown in Figure 2-3.

Figure 2-3. CSA_EMC Block Diagram



The Constant current source (iDAC) provides I_{DAC} amount of current into the Analog MUX. The sensor (C_X), which is alternatively connected between Analog MUX (AMUX) bus and GND by the switches Sw1 and Sw2, respectively, drains away I_{SENSOR} amount of current from the AMUX bus. The magnitude of I_{SENSOR} is directly proportional to the magnitude of $C_{X.}$ The switches Sw1 and Sw2 are clocked by a nonoverlapping clock called the precharge clock.

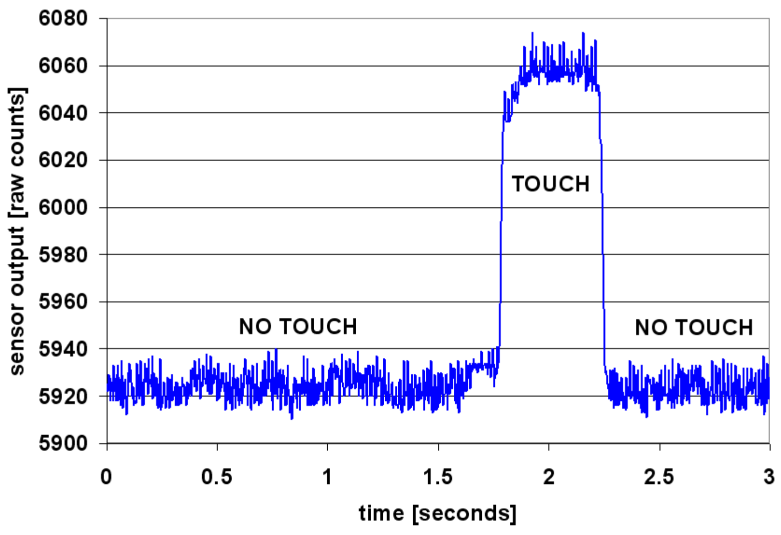
The integration capacitor C_{INT} integrates the difference current I_{DIFF} *(*difference of I_{DAC} and I_{SENSOR})and increases its potential. This charge integration continues until potential developed across C_{INT} reaches an equilibrium level at which I_{SENSOR} becomes equal to IDAC. This integration time is referred to as settling time.

A single-slope ADC is used to convert the equilibrium potential on C_{INT} to digital output counts, known as raw count, which is proportional to C_X. This raw count is interpreted by high-level algorithms to resolve the sensor state.

The I_{DAC} current is set using the successive approximation method to make sure that the equilibrium voltage on C_{INT} is in the linear conversion region of the ADC.

Figure 2-4 plots the CSA_EMC raw counts from a number of consecutive scans during which the sensor is touched and then released by a finger. As explained in CapSense Fundamentals, the finger touch causes C_X to increase by C_F, which in turn causes raw counts to increase proportionally. By comparing the shift in steady-state raw count level to a predetermined threshold, the high-level algorithms can determine whether the sensor is in an On (Touch) or Off (No Touch) state.

Figure 2-4. CSA_EMC Raw Counts during a Finger Touch



The CSA_EMC CapSense algorithm was designed to perform well in the presence of RF interference. CSA_EMC is used in applications where CapSense is exposed to conducted interference, AC noise, and other noise sources such as inverters, transformers, and power supplies. Low-Level Parameters discusses this topic in detail.

3. CapSense Design Tools

3.1 Overview

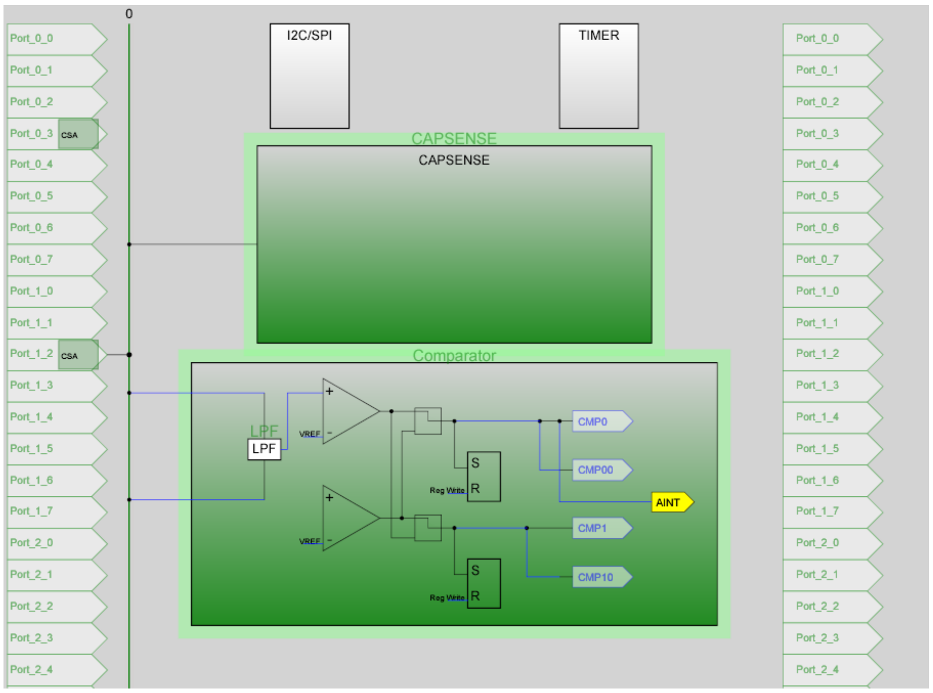
Cypress offers a full line of hardware and software tools for developing your CapSense capacitive touch sense application. A basic development system for the CY8C20x34 family includes the components discussed in this chapter. See Resources for ordering information.

3.1.1 PSoC Designer and User Modules

Cypress's exclusive integrated design environment, PSoC Designer, allows you to configure analog and digital blocks, develop firmware, and tune and debug your design. Applications are developed in a drag-and-drop design environment using a library of user modules. User modules are configured either through the Device Editor GUI or by writing into specific registers with firmware. PSoC Designer comes with a built-in C compiler and an embedded programmer. A pro compiler is available for complex designs.

The CSA_EMC User Module implements capacitive touch sensors using switched-capacitor circuitry, an analog multiplexer, a comparator, digital counting functions, and high-level software routines (APIs). User modules for other analog and digital peripherals are available to implement additional functionality such as I^{2}C, SPI, TX8, and Timers.

Figure 3-1. PSoC Designer Device Editor



3.1.1.1 Getting Started with CapSense User Modules

To create a new CY8C20x34 project in PSoC Designer:

1. Create a new PSoC Designer project with CY8C20x34 as the target device.
2. Select and place any user modules requiring dedicated pins (such as I^{2}C or LCD). Assign ports and pins.
3. Select and place the CSA_EMC User Module.
4. Right-click the CSA_EMC User Module to access the CSA_EMC wizard.
5. Set the number of sensors, sliders or radial sliders that you want.
6. Set pins and global user module parameters.
7. Generate the application and switch to Application Editor.
8. Adapt sample code from the user module datasheet to implement buttons, sliders, or touch pad.

For code examples on CapSense user modules, see Code Examples.

3.1.2 Universal CapSense Controller Kit

The Universal CY3280-20x34 CapSense Controller Kit features predefined control circuitry and plug-in hardware to make prototyping and debugging easy. The MiniProg 1 hardware is included with the kit for programming, and I^{2}C-to-USB Bridge hardware (CY3240 - I2USB Bridge) is included for tuning and data acquisition.

3.1.3 Universal CapSense Controller Module Board

Cypress's module boards feature a variety of sensors, LEDs, and interfaces to meet your application's needs.

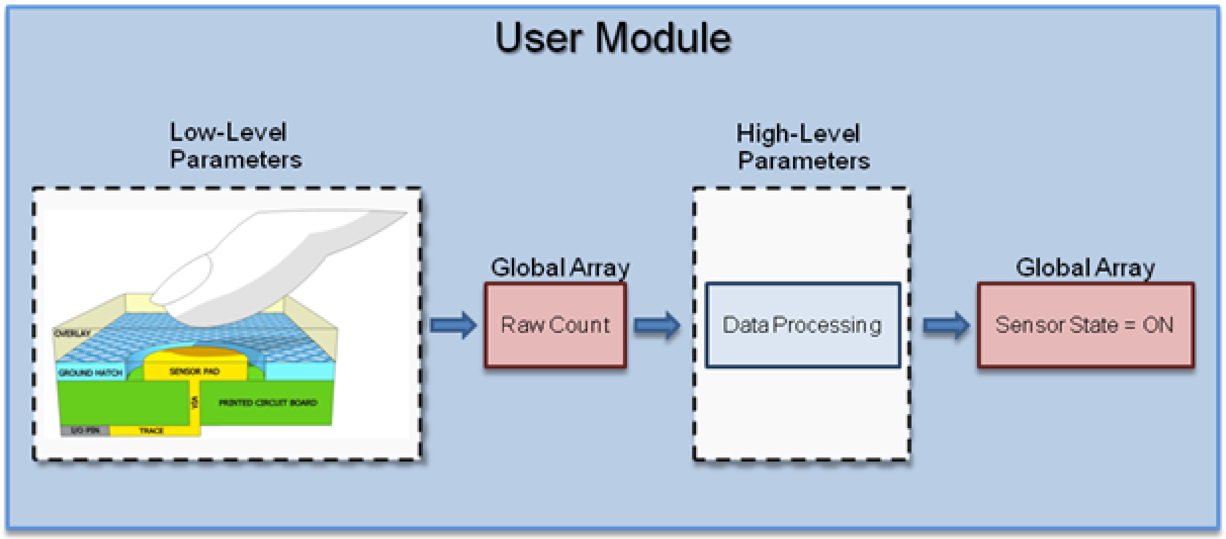
- CY3280-BSM Simple Button Module
- CY3280-BMMMatrix Button Module
- CY3280-SLM Linear Slider Module
- CY3280-SRM Radial Slider Module
- CY3280-BBM Universal CapSense Prototyping Module

3.1.4 CapSense Data Viewing Tools

Many times during CapSense design, you will want to monitor CapSense data (raw counts, baseline, difference counts, and so on) for tuning and debugging purposes. There are two CapSense Data Viewing Tools, MultiChart and Bridge Control Panel. These tools are explained in more detail in the application note AN2397 – CapSense Data Viewing Tools.

3.2 User Module Overview

Figure 3-2. User Module Block Diagram



User modules contain an entire CapSense system from physical sensing to data processing. The behavior of the user module is defined using a variety of parameters. These parameters affect different parts of the sensing system and can be separated into low-level and high-level parameters, which communicate with one another using global arrays.

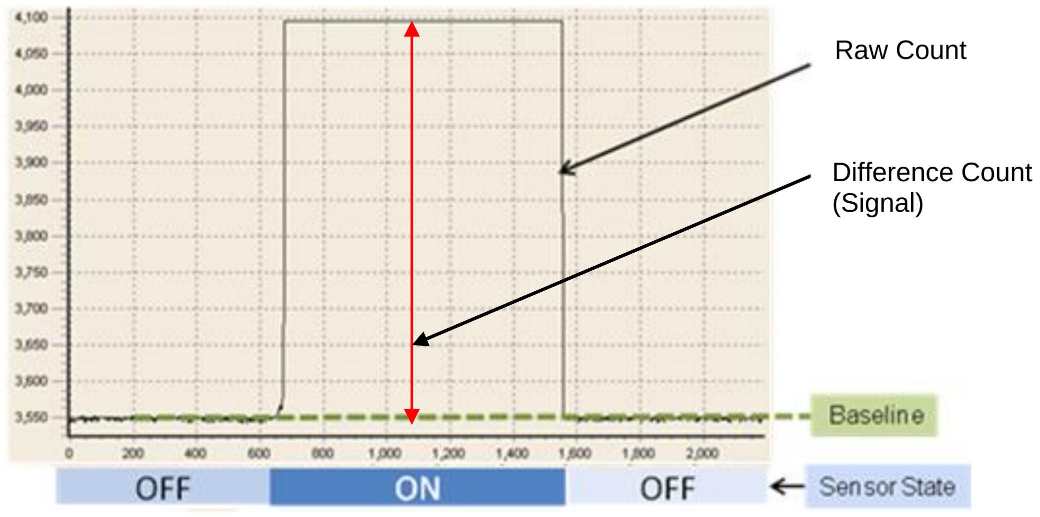
Low-level parameters, such as speed and resolutions for scanning sensors, define the behavior of the sensing method at the physical layer and relate to the conversion from capacitance to raw count. These parameters are unique to each type of sensing method and are described in Low-Level Parameters.

High-level parameters, such as debounce counts and noise thresholds, define how the raw counts are processed to produce information such as the sensor ON/OFF state and the estimated finger position on a slider. These parameters are the same for all sensing methods and are described in High-Level Parameters.

3.3 CapSense User Module Global Arrays

Before learning CapSense User Module parameters, you must be familiar with certain global arrays used by the CapSense system. These arrays should not be altered manually, but may be inspected for debugging purposes.

Figure 3-3. Raw Count, Baseline, Difference Count, and Sensor State



3.3.1 Raw Count

The hardware circuit in CapSense controller measures the sensor capacitance and stores the result in a digital form called raw count upon calling the user module API CSA_EMC_ScanSensor().

The raw count of a sensor is proportional to its sensor capacitance. Raw count increases when the sensor capacitance value increases.

The CSA_EMC user module stores the raw count values of sensors in the CSA_EMC_waSnsResult[] integer array. This array is defined in the header file CSA_EMC.h

3.3.2 Baseline

Gradual environmental changes such as temperature and humidity affect the sensor raw count, which results in variations in the counts.

A complex baselining algorithm is used by the user module to compensate for these variations. The algorithm uses baseline variables to accomplish this. The baseline variables keep track of any gradual variations in raw count values. Essentially, the baseline variables hold the output of a digital low pass filter to which input raw count values are fed.

The baselining algorithm is executed by the user module API CSA_EMC_UpdateSensorBaseline.

Baseline values of sensors are stored in the CSA_EMC_waSnsBaseline[] integer array. This array is defined in the header file CSA_EMC.h.

3.3.3 Difference Count (Signal)

Difference Count, also known as the signal of a sensor, is defined as the difference in counts between the sensor’s raw count and baseline values. The Difference Count is zero when the sensor is inactive. Activating sensors (by touching) results in a positive difference count value.

Difference Count values of sensors are stored in the CSA_waSnsDiff[] integer array. This array is defined in the header file CSA_EMC.h.

Difference count variables are updated by the user module API CSA_EMC_UpdateSensorBaseline().

3.3.4 Sensor State

Sensor State represents the active/inactive status of physical sensors. The state of sensor changes from 0 to 1 upon finger touch and returns to 0 upon finger release.

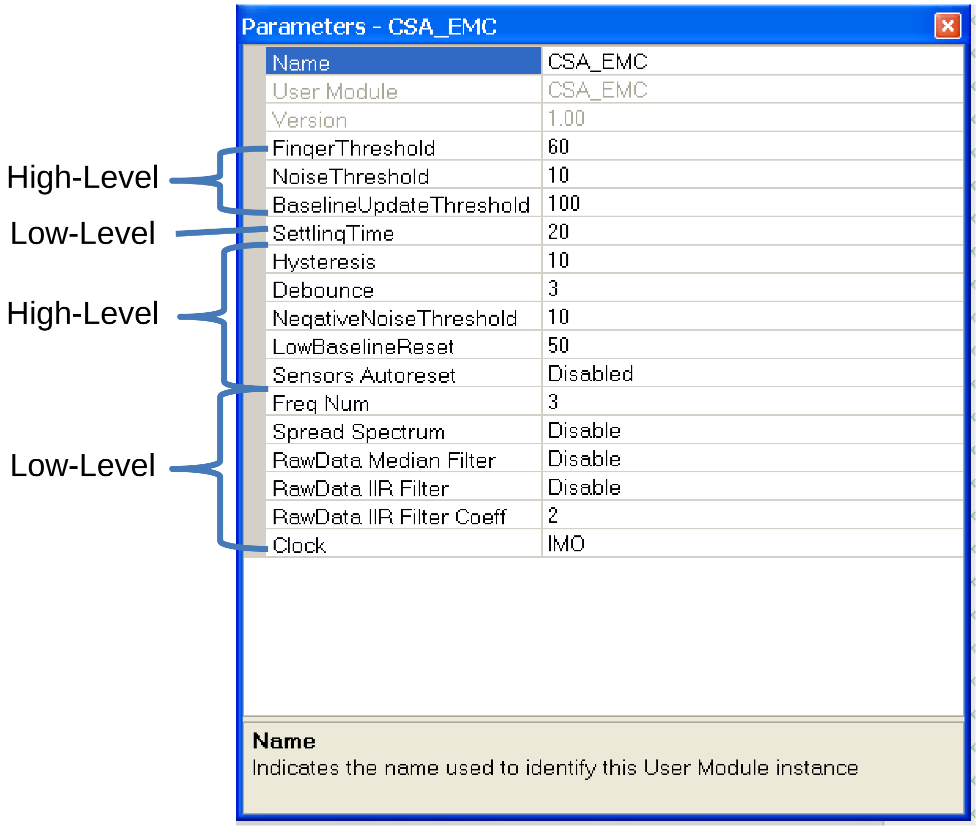
Sensor States are stored in a byte array named CSA_EMC_baSnsOnMask[].Each array element stores the Sensor State of eight consecutive sensors. This array is defined in the header file CSA_EMC.h.

Sensor States are updated by the user module API CSA_bIsAnySensorActive().

3.4 CSA_EMC User Module Parameters

CSA_EMC user module parameters are classified as either high-level or low-level parameters. See Figure 3-4 for a list of CSA_EMC user module parameters and how they are classified.

Figure 3-4. PSoC Designer - CSA_EMC Parameter Window



Name	CSA_EMC
User Module	CSA_EMC
Version	1.00
FingerThreshold	60
NoiseThreshold	10
BaselineUpdateThreshold	100
SettlingTime	20
Hysteresis	10
Debounce	3
NegativeNoiseThreshold	10
LowBaselineReset	50
Sensors Autoreset	Disabled
Freq Num	3
Spread Spectrum	Disable
RawData Median Filter	Disable
RawData IIR Filter	Disable
RawData IIR Filter Coeff	2
Clock	IMO

Name
Indicates the name used to identify this User Module instance

3.4.1 High-Level Parameters

3.4.1.1 Finger Threshold

This parameter is used by the user module to judge the active/inactive state of a sensor. If the difference count value of a sensor is above finger threshold, the sensor is judged as active. This definition assumes that Hysteresis is set to zero and Debounce is set to 1.

Possible values are 3 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.2 Hysteresis

The Hysteresis setting prevents the sensor ON state from chattering due to system noise. The function of Hysteresis is given in Equation 4. The equation assumes that Debounce is set to 1.

Figure 3-5. Sensor State Versus Difference Count with Hysteresis Set to Zero

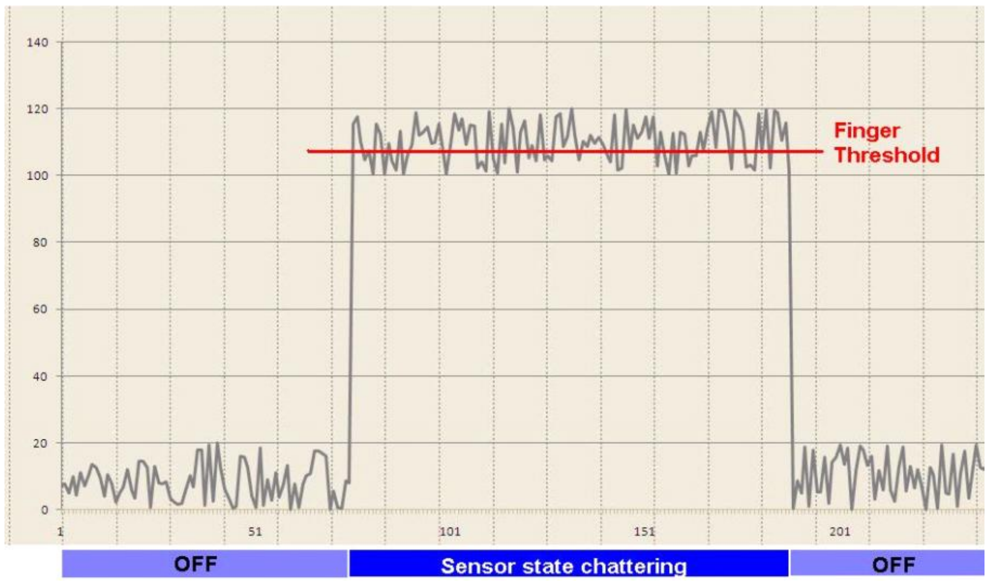
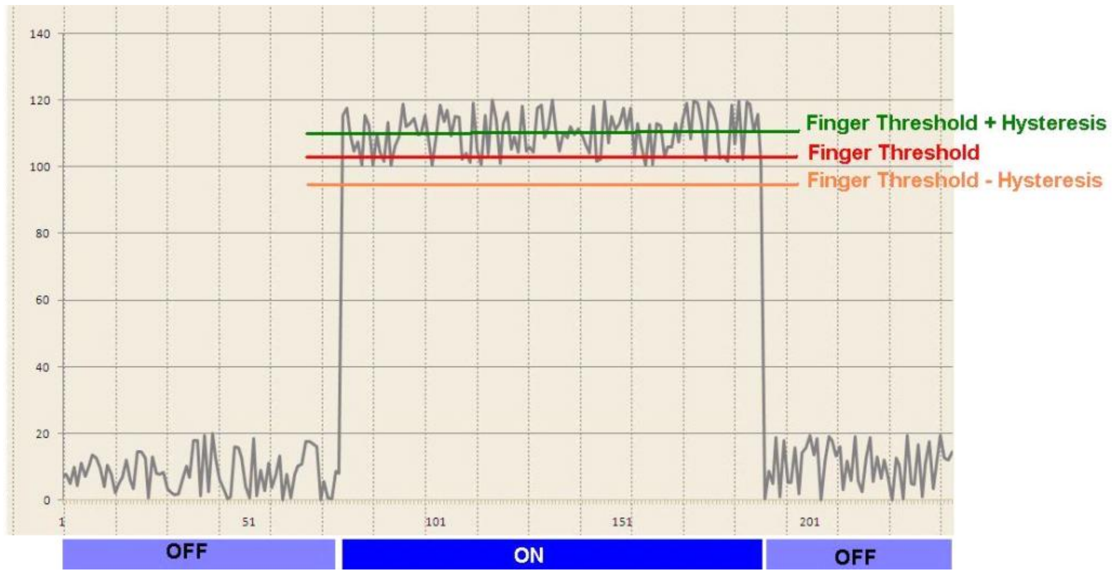


Figure 3-6.Sensor State Versus Difference Count with Hysteresis



If DifferenceCount ≥ FingerThreshold + Hysteresis, Sensor State = ON

If DifferenceCount ≥ FingerThreshold - Hysteresis, Sensor State = OFF Equation 4

Possible values are 0 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.3 *Debounce*

Debounce prevents spikes in raw counts from changing the sensor state from OFF to ON. For the sensor state to transition from OFF to ON, the difference count value must stay above the finger threshold plus hysteresis level for the number of samples specified.

Possible values are 1 to 255. A setting of 1 provides no debouncing.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.4 *Baseline Update Threshold*

As previously explained, the baseline variables keep track of any gradual variations in raw count values. In other words, baseline variables hold the output of a digital low pass filter to which the input raw count values are fed. The Baseline Update Threshold parameter is used to adjust the time constant of this low-pass filter.

The baseline update threshold is directly proportional to the time constant of this filter. The higher the baseline update threshold value, the higher the time constant.

Possible values are 1 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.5 *Noise Threshold*

The noise threshold helps the user module to understand the upper limit of noise counts in the raw count. For individual sensors, the baselining update algorithm is paused when the raw count is above the baseline and the difference between them is greater than this threshold.

For slider sensors, the centroid calculation is paused when the difference count is greater than the Noise Threshold value.

Possible values are 3 to 255. For proper user module operation, the Noise Threshold value should never be set higher than Finger Threshold minus Hysteresis.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.6 *Negative Noise Threshold*

The Negative Noise threshold helps the user module to understand the lower limit of noise counts in the raw count. The baselining update algorithm is paused when the raw count is below the baseline and the difference between them is greater than this threshold.

Possible values are 0 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.7 *Low Baseline Reset*

The Low Baseline Reset parameter works in conjunction with the Negative Noise Threshold parameter. If the sample count values are less than the baseline minus the negative noise threshold for the specified number of samples, the baseline is set to the new raw count value. It essentially counts the number of abnormally low samples required to reset the baseline. It is used to correct the finger-on-at-startup condition.

Possible values are 0 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.1.8 *Sensors Autoreset*

The Sensors Autoreset parameter determines whether the baseline is updated at all times, or only when the difference counts are less than the noise threshold.

When Sensors Autoreset is enabled, the baseline is updated constantly. This limits the maximum time duration of the sensor but prevents the sensors from permanently turning on when the raw count accidentally rises without anything touching the sensor. This sudden rise can be caused by a large power supply voltage fluctuation, a high energy RF noise source, or a very quick temperature change.

When Sensors Autoreset is disabled, the baseline is updated only when the difference counts are less than the Noise Threshold parameter.

Possible values are Enabled and Disabled.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.2 Low-Level Parameters

The CSA_EMC User Module has several low-level parameters in addition to the high-level parameters. These parameters are specific to the CSA_EMC sensing method and determine how raw count data is acquired from the sensor.

3.4.2.1 Settling Time

The Settling Time parameter controls the software delay that allows the voltage on the C_{INT} capacitor to reach the integration equilibrium potential.

Possible values are 2 to 255.

For the recommended value, see CSA_EMC User Module – Tuning Guide.

3.4.2.2 Immunity Level

The Immunity Level parameter defines the noise immunity level. The available options are Low and High.

Selecting the High option improves the performance in a high-noise environment. However, this increases the memory usage, and, as a result, decreases the maximum number of supported sensors.

Also, the Scan time with the High setting takes three times longer than it does with the Low setting.

3.4.2.3 Spread Spectrum

Enabling Spread Spectrum improves the performance of the CSA_EMC algorithm in the presence of external RF interference. It also improves the emission characteristic of the algorithm.

Possible values are Enabled and Disabled.

3.4.2.4 Raw Data Median Filter

The median filter looks at the three most recent samples from a sensor and reports the median value. This filter is used to remove short noise spikes. It generates a delay of one sample and is disabled by default.

Possible values are Enabled and Disabled.

3.4.2.5 Raw Data IIR Filter

This infinite impulse response (IIR) filter reduces noise in the conversion result (raw count). It is disabled by default.

Possible values are Enabled and Disabled.

3.4.2.6 Raw Data IIR Filter Coefficient

This setting is the coefficient for the Raw Count IIR filter.

Possible values are 2 (½ previous sample + ½ current sample) and 4 (¾ previous sample + ¼ current sample).

3.4.2.7 Clock

The Clock parameter sets the frequency of the nonoverlapping precharge clock that controls the switches Sw1 and Sw2. Sw1 and Sw2 are used to alternately connect the sensor capacitor between then AMUX bus and GND. To achieve the maximum signal, the precharge clock frequency should be set such that the sensor capacitor charges and discharges fully during each cycle.

Possible values are IMO, IMO/2, IMO/4, and IMO/8.

4. CapSense Performance Tuning with User Modules

Optimal User Module parameter settings depend on board layout, button dimensions, overlay material, and application requirements. These factors are discussed in Design Considerations. Tuning is the process of identifying the optimal parameter settings for robust and reliable sensor operation.

4.1 General Considerations

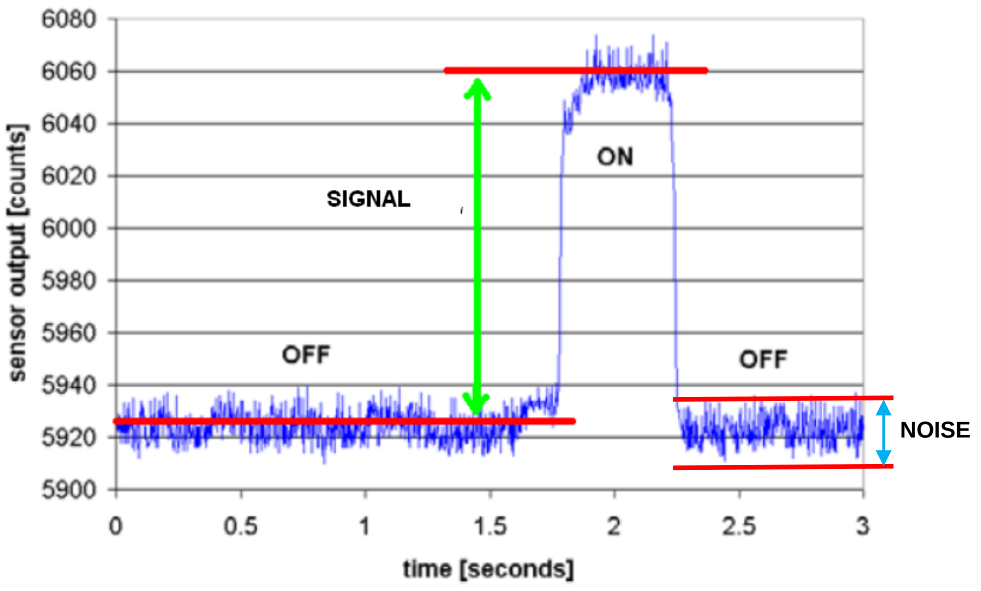
4.1.1 Signal, Noise, and SNR

A well-tuned CapSense system reliably discriminates between ON and OFF sensor states. To achieve this level of performance, the CapSense signal must be significantly larger than the CapSense noise. The CapSense signal is compared to the CapSense noise by using a quantity called signal to noise ratio (SNR). Before discussing the meaning of SNR for CapSense, it is first necessary to define what signal and noise are in the context of touch sensing.

4.1.1.1 CapSense Signal

The CapSense signal is the change in the sensor response when a finger is placed on the sensor, as shown in Figure 4-1. The output of the sensor is a digital counter with a value that tracks the sensor capacitance. In this example, the average level without a finger on the sensor is 5925 counts. When a finger is placed on the sensor, the average output increases to 6060 counts. Because the CapSense signal tracks the change in counts due to the finger, Signal = 6060 – 5925 = 135 counts.

Figure 4-1. CapSense Signal and Noise



4.1.1.2 CapSense Noise

CapSense noise is the peak-to-peak variation in sensor response when a finger is not present, as shown in Figure 4-1. In this example, the output waveform without a finger is bounded by a minimum of 5912 counts and a maximum of 5938 counts. Because the noise is the difference between the minimum and the maximum values of this waveform, Noise = 5938 – 5912 = 26 counts.

4.1.1.3 CapSense SNR

CapSense SNR is the simple ratio of signal and noise. Continuing with the example, if the signal is 135 counts and noise is 26 counts, SNR is 135:26, which reduces to an SNR of 5.2:1. The minimum recommended SNR for CapSense is 5:1, which means the signal is five times larger than the noise. Filters are commonly implemented in firmware to reduce noise. For more information, see Software Filtering.

4.1.2 Importance of Baseline Update Threshold Verification

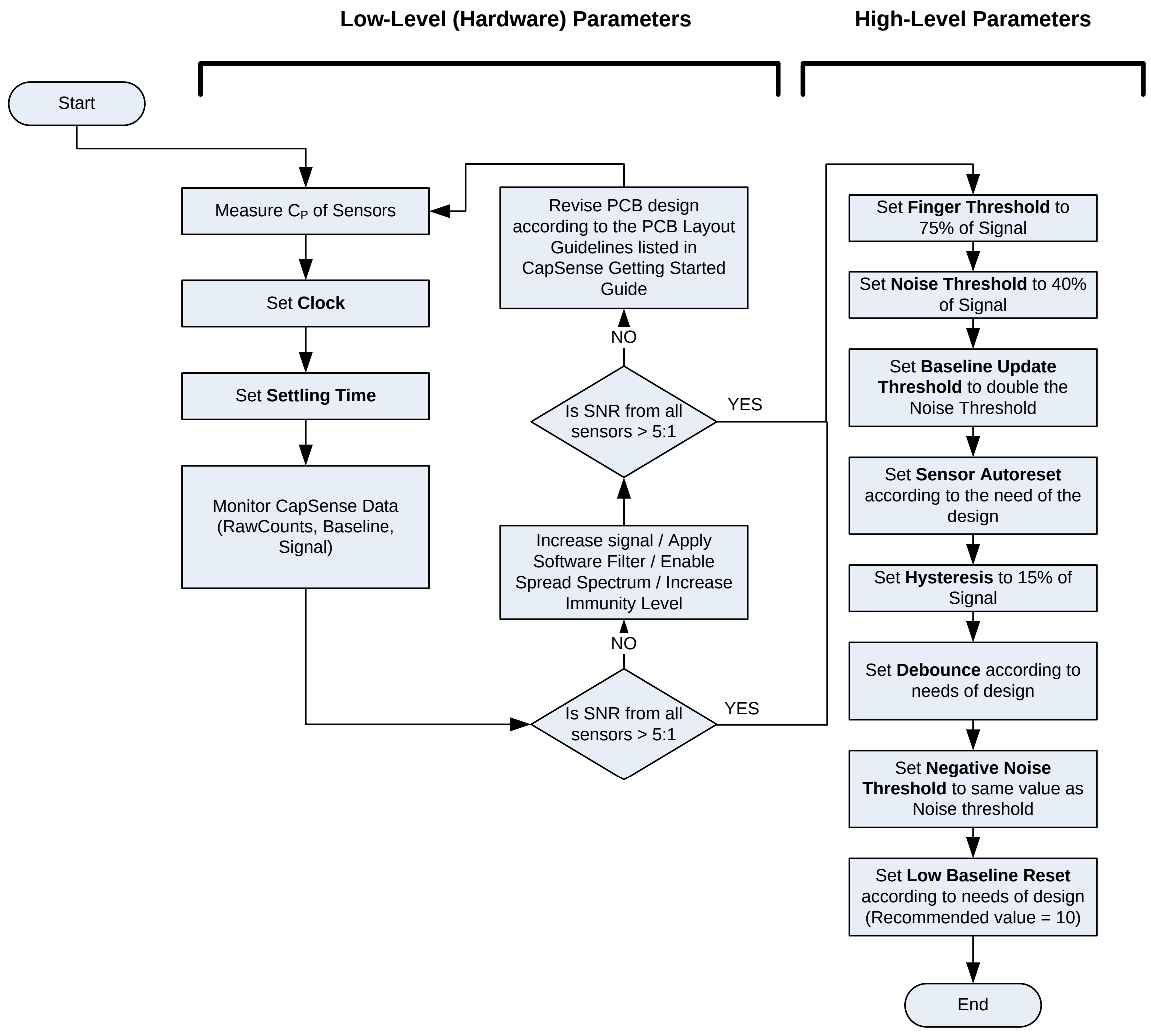
Temperature and humidity both cause the average number of counts to drift over time. The baseline is a reference count level for CapSense measurements that plays an important role in compensating for environmental effects. High-level decisions, such as Finger Present and Finger Absent states, are based on the reference level established by the baseline. Because each sensor has unique parasitic capacitance associated with it, each capacitive sensor has its own baseline.

Baseline tracks the change in counts at a rate set by the Baseline Update Threshold parameter. Make sure to match the update rate to the intended application. If the update rate is too fast, then the baseline will compensate for any changes introduced by a finger, and the moving finger will not be detected. If the update rate is too slow, then relatively slow environmental changes may be mistaken for fingers. During development, it is recommended that the Baseline Update Threshold parameter setting is verified.

4.2 CSA_EMC User Module – Tuning Guide

Figure 4-2 is a flowchart showing the tuning process of CSA_EMC parameters. CSA_EMC User Module parameters can be separated into two broad categories: low-level (hardware) parameters and high-level parameters. The parameters in these categories affect the behavior of the capacitive sensing system in different ways. However, there is a complementary relationship between the sensitivity of each sensor as determined by the hardware parameter settings and many of the high-level parameter settings. You must consider this fact when you change any hardware parameter to make sure that the corresponding high-level parameters are adjusted accordingly. Tuning CSA_EMC User Module parameters should always begin with the hardware parameters.

Figure 4-2. CSA_EMC User Module Parameters Tuning Flowchart


Low-Level (Hardware) Parameters
High-Level Parameters
Start
Measure CP of Sensors
Set Clock
Set Settling Time
Monitor CapSense Data (RawCounts, Baseline, Signal)
Revise PCB design according to the PCB Layout Guidelines listed in CapSense Getting Started Guide
NO
Is SNR from all sensors > 5:1
YES
Increase signal / Apply Software Filter / Enable Spread Spectrum / Increase Immunity Level
NO
Is SNR from all sensors > 5:1
YES
Set Finger Threshold to 75% of Signal
Set Noise Threshold to 40% of Signal
Set Baseline Update Threshold to double the Noise Threshold
Set Sensor Autoreset according to the need of the design
Set Hysteresis to 15% of Signal
Set Debounce according to needs of design
Set Negative Noise Threshold to same value as Noise threshold
Set Low Baseline Reset according to needs of design (Recommended value = 10)
End

4.1 Recommended C_{INT} Value

Start the tuning process with the recommended Integration Capacitor C_{INT} value of 1.2nF. In the process of tuning, if you find that the sensor signal is not adequate to get 5:1 SNR, you can increase C_{INT}. The recommended maximum value of C_{INT} is 5.6 nF. X7R or NPO type capacitors are recommended for C_{INT} stability over temperature.

4.2 Measuring Sensor C_P

The first step in the tuning procedure is to measure the sensor parasitic capacitance (C_P). A step-by-step procedure to do this follows:

1. Set **CPU_CLK** equal to SysClk/2.
2. Set **Clock** equal to IMO/8.
3. Set **Settling Time** equal to 255.
4. Read back the IDAC code set by the algorithm for the particular sensor. The value is stored in the array `CSA_EMC_baDACCodeBaseline[]`
5. Measure the IDAC current corresponding to the IDAC code.

Create a PSoC Designer project with the following code. The code routes the IDAC to port pin P1[4].

```
//configure P1[4] to HI-Z
PRT1DM0 &= ~ (1<<4);
PRT1DM1 |= (1<<4);
//connect P1[4] to analog mux bus
MUX_CR1 = (1<<4);
// set IDAC to read back IDAC Code
IDAC_D = <IDAC CODE>
// turn ON IDAC
CS_CR2 = 0xD0;
```

Place a current meter between pin P1[4] and ground, and measure current. Let $I_{MEASURED}$ be its value.

6. Calculate C_P using Equation 5.

$$C_P = \frac{I_{MEASURED}}{(\frac{IMO}{8} \times 1.3)} \qquad \text{Equation 5}$$

Alternatively, sensor C_P can also be measured using an LCR meter. Connect one terminal of the LCR meter to the sensor pin and the other to GND to measure C_P.

4.3 Estimating the CSA_EMC Clock

The precharge clock is set by the CSA_EMC Clock user module parameter. This is the most critical hardware UM parameter in properly tuning a CSA_EMC design. Recommended values for the CSA_EMC clock are given in Table 4-1.

Table 4-1. CSA_EMC Clock Setting Based on C_P and IMO

	CSA_EMC Clock	
C_P(pF)	**IMO = 12MHz**	**IMO = 6MHz**
< 5	IMO	IMO
5 to 10	IMO/2	IMO
10 to 15	IMO/4	IMO/2
15 to 20	IMO/4	IMO/2
20 to 25	IMO/8	IMO/4
25 to 30	IMO/8	IMO/4
30 to 35	IMO/8	IMO/4
35 to 40	IMO/8	IMO/4
40 to 45	IMO/8	IMO/4
45 to 50	IMO/8	IMO/8

4.4 Setting Settling Time

The minimum value for the Settling Time parameter is estimated using Equation 6.

$$Settling\ Time = \frac{5 \times C_{INT}}{(Clock \times C_P \times 25 \times \frac{1}{FCPU})} \qquad \text{Equation 6}$$

Where,

C_{INT} = Value of integration capacitor

Clock = Precharge clock frequency (CSA_EMC Clock)

C_P = Sensor parasitic capacitance value

F_{CPU} = CPU clock frequency

4.5 Monitoring CapSense Data

Refer to CapSense Data Viewing Tools.

4.6 Methods to Increase SNR

To increase SNR, either reduce the noise counts or increase the signal.

4.6.1 Reduce Noise

One way to increase SNR is to reduce the noise counts. You can use one of these options:

- Use software filters—For details, see Software Filtering.
- Enable Spread Spectrum—For details, see Spread Spectrum.
- Increase Immunity Level—For details, see Immunity Level

4.6.2 Increase Signal

SNR can also be improved by increasing the signal. Use one of these two methods:

- Increase the value defined by the macro `CSA_EMC_BASELINE`. This macro is located in the file *CSA_EMC.inc.* By default, the macro is assigned a value of 0x0800*.*
- Increase the value of the C_{INT} capacitor.

4.7 Tuning Example

This section gives an example of how to tune a single CapSense sensor by following the tuning guidelines. This example assumes that you are familiar with PSoC Designer and CapSense Data Viewing Tools. For more information, see the PSoC Designer User Guides and AN2397 - CapSense Data Viewing Tools.

4.7.1 Hardware Requirements

Tuning can be carried out in your own board or by using the standard Cypress CapSense UCC boards (links given below).

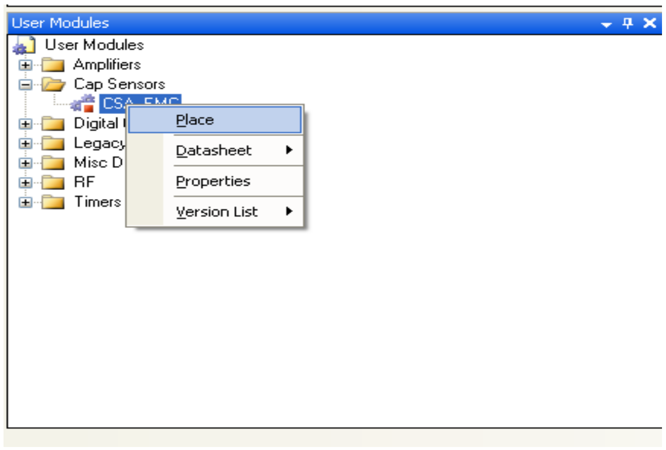
- CY3280-20x34 Universal CapSense Controller
- CY3280-BSM simple button module kit

The step-by-step tuning procedure given below uses the CapSense sensor at port pin P1[4] and Integration capacitor at port pin P0[3].

4.7.2 Step-By-Step Tuning Procedure

1. Create a new PSoC Designer project with CY8C20x34 as the target device.
2. Go to the **User Module Catalog** window. Select and place the CSA_EMC user module from the Cap Sensors category.

Figure 4-3. Selecting and Placing the CSA_EMC User Module



3. Right-click the CSA_EMC User Module in the Workspace Explorer to access the CapSense Wizard. Assign pins for sensors and C_{INT}.

Figure 4-4. Right-Click the CSA_EMC User Module to Start the CapSense Wizard

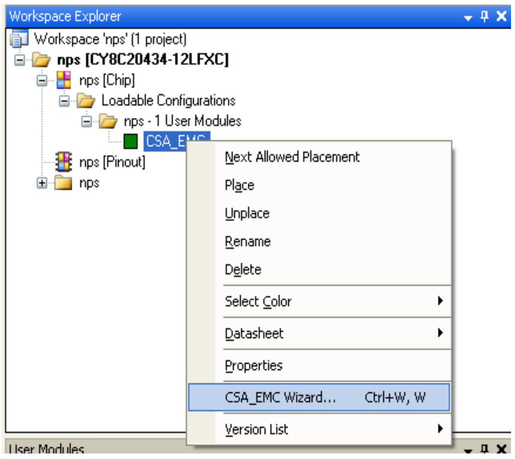
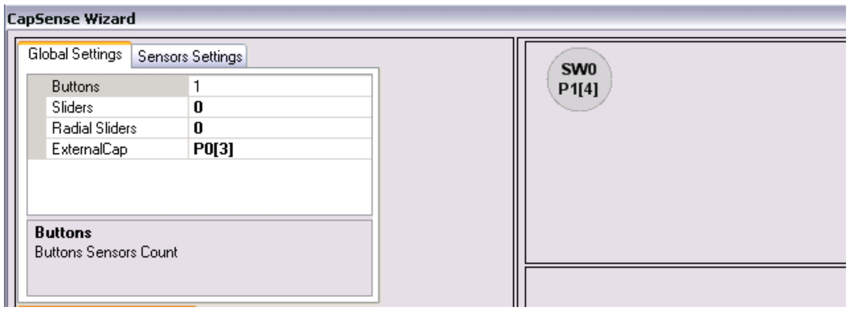


Figure 4-5. CapSense Wizard (Sensor Pin = P1[4], C_{INT} Pin = P0[3])

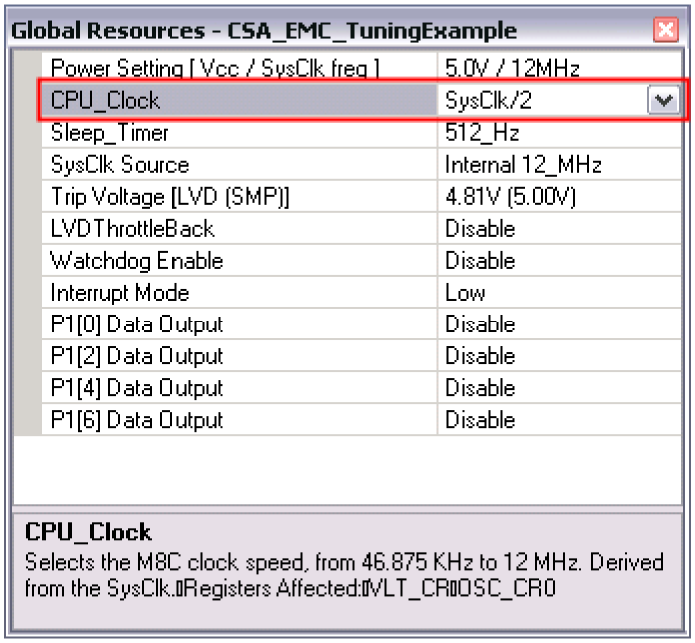


Buttons	1
Sliders	0
Radial Sliders	0
ExternalCap	P0[3]

Buttons
Buttons Sensors Count

4. Set **CPU_CLK** equal to SysClk/2 in the **Global Resources** window. Configure the remaining parameters in the **Global Resources** window based on your design needs

Figure 4-6. Set CPU_CLK = SysClk/2



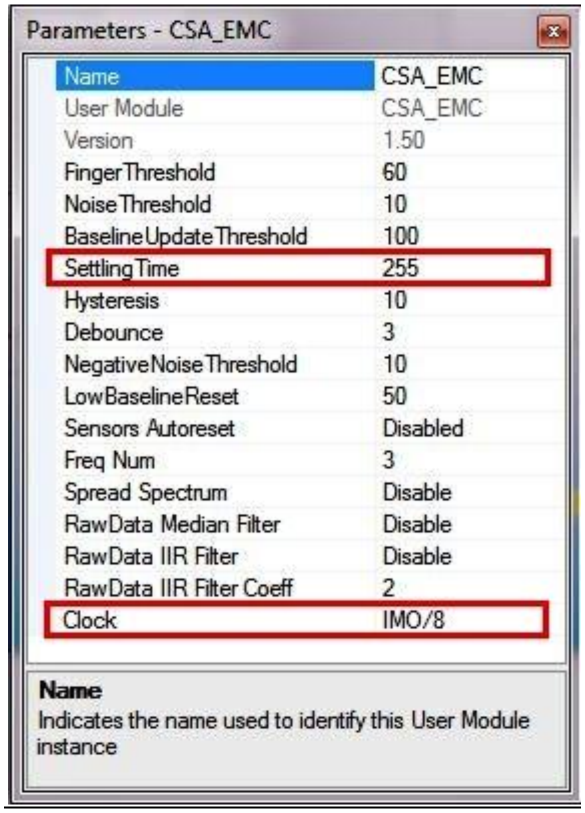
Global Resources - CSA_EMC_TuningExample

Power Setting [Vcc / SysClk freq]	5.0V / 12MHz
CPU_Clock	SysClk/2
Sleep_Timer	512_Hz
SysClk Source	Internal 12_MHz
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable
Interrupt Mode	Low
P1[0] Data Output	Disable
P1[2] Data Output	Disable
P1[4] Data Output	Disable
P1[6] Data Output	Disable

CPU_Clock
Selects the M8C clock speed, from 46.875 KHz to 12 MHz. Derived from the SysClk.Registers Affected:VLT_CR OSC_CR0

5. Set **Clock** = IMO/8 and **Settling Time** = 255 in the **Parameters - CSA_EMC** window. Keep the default values of all other parameters.

Figure 4-7. Set Clock = IMO/8, Settling Time = 255



Parameters - CSA_EMC

Name	CSA_EMC
User Module	CSA_EMC
Version	1.50
FingerThreshold	60
NoiseThreshold	10
BaselineUpdateThreshold	100
SettlingTime	255
Hysteresis	10
Debounce	3
NegativeNoiseThreshold	10
LowBaselineReset	50
Sensors Autoreset	Disabled
Freq Num	3
Spread Spectrum	Disable
RawData Median Filter	Disable
RawData IIR Filter	Disable
RawData IIR Filter Coeff	2
Clock	IMO/8

Name
Indicates the name used to identify this User Module instance

6. With the help of a digital communication interface (I^2C or SPI), read back the IDAC code in `CSA_EMC_baDACCodeBaseline[]` using any of the CapSense viewing tools (MultiChart or Bridge Control Panel). For more information, refer to AN2397 - CapSense Data Viewing Tool.

   Sample code:

```
void main(void)
{
    M8C_EnableGInt;
    CSA_EMC_Start();
    …
        …
    while(1)
    {
    // OUTPUT CSA_EMC_baDACCodeBaseline[0] <- IDAC code for sensor 0;
    }
}
```

Let the read-back code found in this step be 0x08.

7. Measure the I_{DAC} current that corresponds to the I_{DAC} code of 0x08.

   Sample code:

```
void main(void)
{
        //configure P1[4] to HI-Z
        PRT1DM0 &= ~ (1<<4);
        PRT1DM1 |= (1<<4);
        //connect P1[4] to analog mux bus
        MUX_CR1 = (1<<4);
        // set IDAC to read back IDAC Code
        IDAC_D = 0x08;
        // turn ON IDAC
        CS_CR2 = 0xD0;
}
```

For details about how to measure the current, see Step 5 in Measuring Sensor CP.

Let the measured current be 23 µA.

8. Calculate C_P.

   From Equation 5, $C_P = I_{MEASURED} / ((IMO/8) * 1.3)$

   If you substitute the following:

   $I_{MEASURED} = 23µA$

   IMO = 12 MHz (in this example IMO is set to 12 MHz)

   The result is: C_P = 12 pF

9. Calculate CSA_EMC Clock.

   From Table 4-1, for C_P = 12 pF and IMO = 12 MHz, the recommended Clock is IMO/4.

10. Calculate Settling Time.

    From Equation 5:

$$Settling\ Time = \frac{5 \times C_{INT}}{(Clock \times C_P \times 25 \times \frac{1}{FCPU})}$$

If you substitute the following:

C_{INT}= 1.2 nF

Clock = IMO/4 = 3 MHz

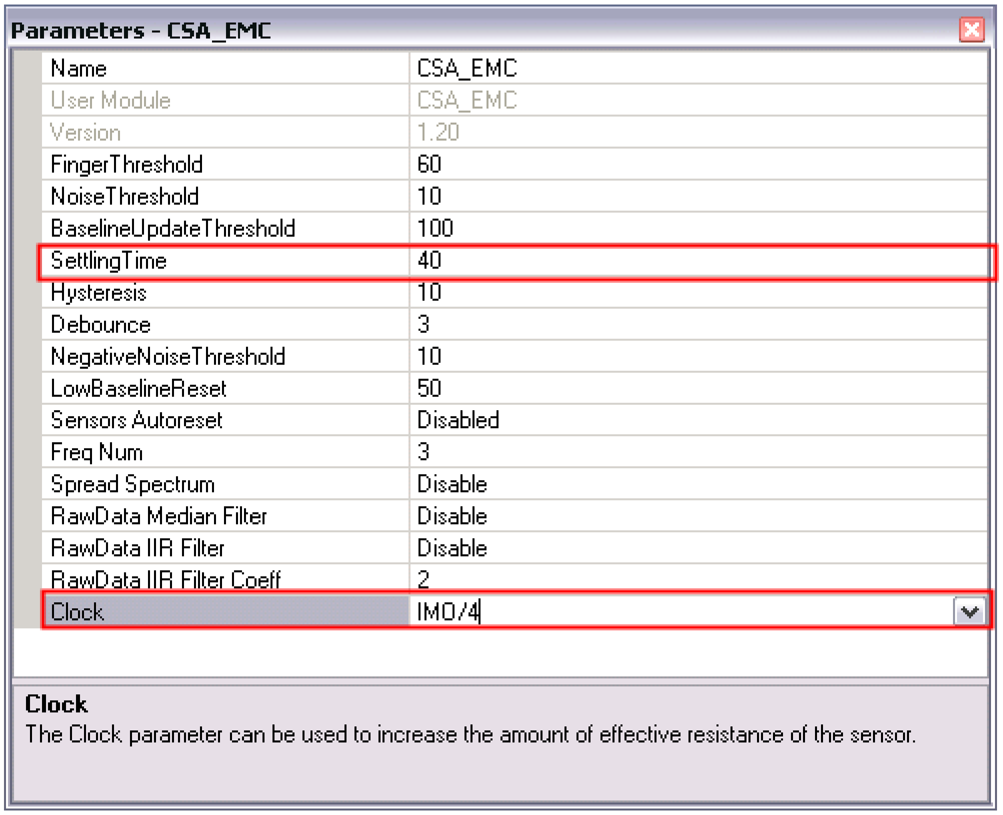
C_P = 12 pF

F_{CPU} = SysClk/2 = 12 MHz/2 = 6 MHz (SysClk is set to 12 MHz in this example)

The result is: Settling Time = 40

11. Update the Clock and Settling Time value in the **Parameters - CSA_EMC** window.

Figure 4-8. Set Clock = IMO/4, Settling Time = 40



Parameters - CSA_EMC

Name	CSA_EMC
User Module	CSA_EMC
Version	1.20
FingerThreshold	60
NoiseThreshold	10
BaselineUpdateThreshold	100
SettlingTime	40
Hysteresis	10
Debounce	3
NegativeNoiseThreshold	10
LowBaselineReset	50
Sensors Autoreset	Disabled
Freq Num	3
Spread Spectrum	Disable
RawData Median Filter	Disable
RawData IIR Filter	Disable
RawData IIR Filter Coeff	2
Clock	IMO/4

Clock
The Clock parameter can be used to increase the amount of effective resistance of the sensor.

12. With the new setting in place, measure the SNR.

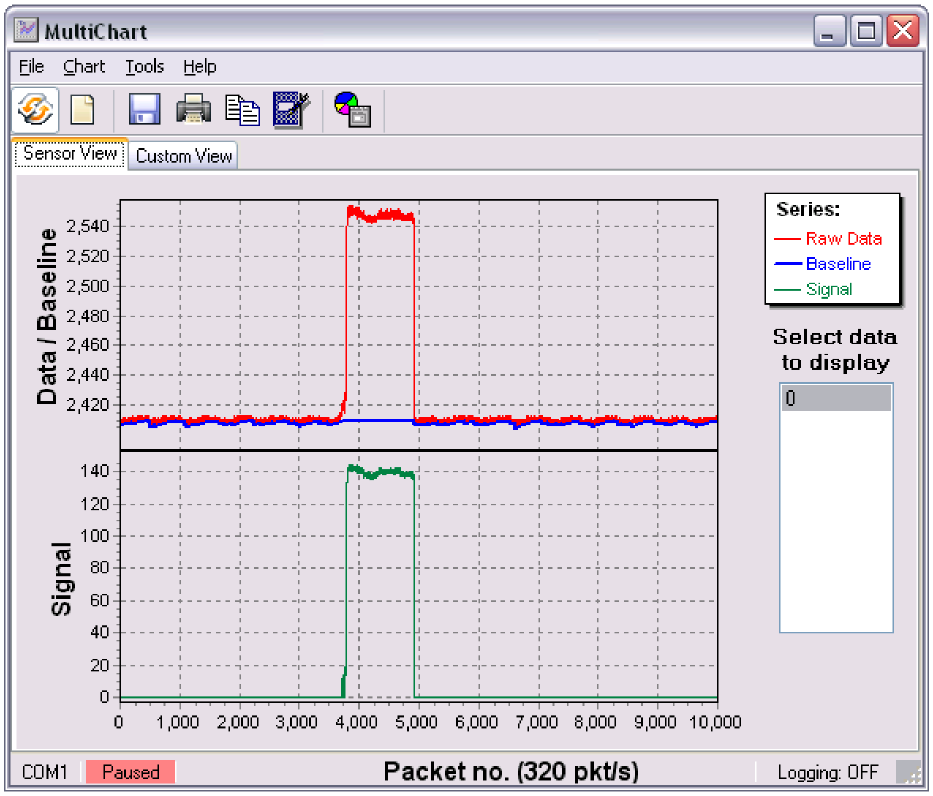
Figure 4-9 shows the observed raw count, baseline, and signal levels.

Signal = 140 counts (approximately)

Noise = 10 counts (approximately)

Therefore, SNR is 14. This is greater than 5.

Figure 4-9. Signal = 140 Counts, Noise = 10 Counts



13. Set the high-level parameters:

    Finger Threshold = 75% of Signal = 0.75 × 140 = 105

    Noise Threshold = 40% of Signal = 0.40 × 140 = 56

    Baseline Update Threshold = 2 × Noise Threshold = 112

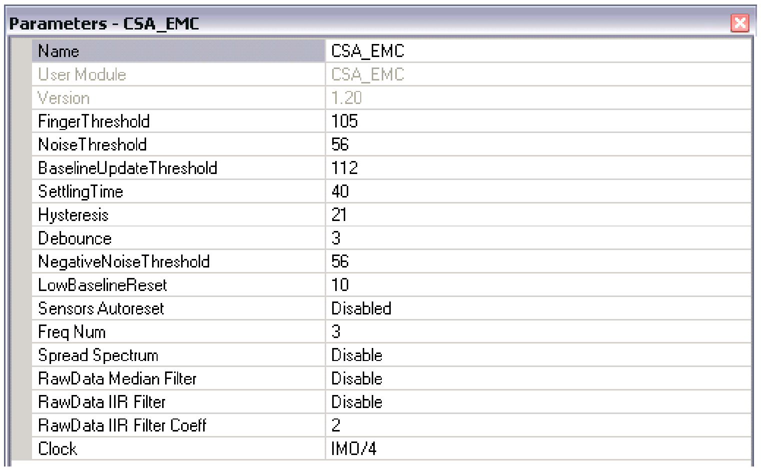
    Hysteresis = 15% of Signal = 0.15 × 140 = 21

    Negative Noise Threshold = Noise Threshold = 56

    Low Baseline Reset = 10

14. Update the CSA_EMC parameter values based on the previous calculations.

Figure 4-10. Updated CSA_EMC Parameters Window



Parameters - CSA_EMC

Name	CSA_EMC
User Module	CSA_EMC
Version	1.20
FingerThreshold	105
NoiseThreshold	56
BaselineUpdateThreshold	112
SettlingTime	40
Hysteresis	21
Debounce	3
NegativeNoiseThreshold	56
LowBaselineReset	10
Sensors Autoreset	Disabled
Freq Num	3
Spread Spectrum	Disable
RawData Median Filter	Disable
RawData IIR Filter	Disable
RawData IIR Filter Coeff	2
Clock	IMO/4

15. With the new settings in place, regenerate the project with necessary design specific code in the *main.c* file.

5. Design Considerations

When designing capacitive touch sense technology into your application, it is crucial 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 lead to robust and reliable system performance. For more in-depth information, refer to Getting Started with CapSense.

5.1 Overlay Selection

In CapSense Fundamentals Model, the CapSense equivalent model was presented along with the equation for finger capacitance:

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

Where:

ε_0 = Free space permittivity

ε_r = Dielectric constant of overlay

A = Area of finger and sensor pad overlap (mm^2)

D = Overlay thickness (mm)

To increase the CapSense signal strength, choose an overlay material with a higher dielectric constant, decrease the overlay thickness, and increase the button diameter.

Table 5-1. 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 (Plexiglas®)	13,000	0.9
ABS	16,000	0.8
Polycarbonate (Lexan®)	16,000	0.8
Formica	18,000	0.7
FR-4	28,000	0.4
PET Film (Mylar®)	280,000	0.04
Polymide film (Kapton®)	290,000	0.04

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

An adhesive is typically used to bond the overlay to the CapSense PCB. A transparent acrylic adhesive film from 3M™ called 200MP is qualified for use in CapSense applications. This special adhesive is dispensed from paper-backed tape rolls (3M™ product numbers 467MP and 468MP).

5.2 ESD Protection

Robust ESD tolerance is a natural byproduct of careful 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 incurring any damage to the CapSense controller.

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 5-1 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.

5.2.1 Prevent

Make sure that all paths on the touch surface have a breakdown voltage greater than potential high-voltage contacts. Also, design your system to maintain an appropriate distance between the CapSense controller and possible sources of ESD. 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.

5.2.2 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. A standard practice is to place a guard ring on the perimeter of the circuit board that is connected to the chassis ground. As recommended in PCB Layout Guidelines, providing a hatched ground plane around the button or slider sensor can redirect the ESD event away from the sensor and CapSense controller.

5.2.3 Clamp

Because CapSense sensors are purposely placed in close proximity to the touch surface, it may not be practical to redirect the discharge path. In this case, including series resistors or special purpose ESD protection devices may be appropriate.

The recommended series resistance value is 560 Ω.

A more effective method is to provide special-purpose ESD protection devices on the vulnerable traces. ESD protection devices for CapSense must be low capacitance. Table 5-2 lists devices recommended for use with CapSense controllers.

Table 5-2. Low-Capacitance ESD Protection Devices Recommended for CapSense

ESD Protection device		**Input Capacitance**	**Leakage Current**	**Contact Discharge Maximum Limit**	**Air Discharge Maximum Limit**
Manufacturer	**Part Number**				
Littlefuse	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

5.3 Electromagnetic Compatibility (EMC) Considerations

5.3.1 Radiated Interference

Radiated electrical energy can influence system measurements and potentially influence the operation of the processor core. The interference enters the PSoC chip at the PCB level, through CapSense sensor traces, and through any other digital or analog inputs. Layout guidelines for minimizing the effects of RF interference include:

- **Ground Plane**: Provides a ground plane on the PCB.
- **Series Resistor**: Place series resistors within 10 mm of the CapSense controller pins
  - The recommended series resistance for CapSense input lines is 560 Ω.
  - The recommended series resistance for communication lines, such as I^2C, and SPI is 330 Ω.
- **Trace Length**: Minimize trace length whenever possible.
- **Current Loop Area**: Minimize the return path for the current. Provide hatched ground instead of solid fill within 1 cm of the sensors and traces to reduce the impact of parasitic capacitance.
- **RF Source Location**: Partition systems with noise sources such as LCD inverters and switched-mode power supplies (SMPS) to keep them separated from CapSense inputs. Shielding the power supply is another common technique for preventing interference.

5.3.2 Radiated Emissions

To reduce radiated emissions from the CapSense sensor, select a low frequency for the switched capacitor clock. This clock is controlled in firmware using the Prescaler option. Increasing the Prescaler value decreases the frequency of the switching clock.

5.3.3 Conducted Immunity and Emissions

Noise entering a system through interconnections with other systems is referred to as conducted noise. These interconnections include power and communication lines. Because CapSense controllers are low-power devices, conducted emissions must be avoided. The following guidelines will help reduce conducted emission and immunity:

- Use decoupling capacitors as recommended by the datasheet.
- Add a bidirectional filter on the input to the system power supply. This is effective for both conducted emissions and immunity. A pi-filter can prevent power-supply noise from affecting sensitive parts, while also preventing the switching noise of the part itself from coupling back onto the power planes.
- If the CapSense controller PCB is connected to the power supply by a cable, minimize the cable length and consider using a shielded cable.
- You can place a ferrite bead around the power supply or communication lines to filter out high-frequency noise.

See Getting Started with CapSense for more information about EMC Considerations.

5.4 Software Filtering

Software filters are one of the techniques for dealing with high levels of system noise. Table 5-3 lists the types of filters that have been found useful for CapSense.

Table 5-3. Table of CapSense Filters

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 during non-touch events to block CapSense data transmission
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 multibutton selection

Table 5-4 details the RAM and Flash requirements for different software filters. The amount of flash required for each filter type depends on the performance of the compiler. The requirements listed here are for both the ImageCraft compiler and the ImageCraft Pro compiler.

Table 5-4. RAM and Flash Requirements

Filter Type	Filter Order	RAM (Bytes Per Sensor)	Flash (Bytes) ImageCraft Compiler	Flash (Bytes) ImageCraft Pro Compiler
Average	2-8	6	675	665
IIR	1	2	429	412
	2	6	767	622
Median	3	6	516	450
	5	10	516	450
Jitter filter on Raw Counts	N/A	2	277	250
Jitter filter on slider centroid	N/A	2	131	109

See Getting Started with CapSense for more information about software filters.

5.5 Power Consumption

5.5.1 System Design Recommendations

For many designs, minimizing power consumption is an important goal. There are several ways to reduce the power consumption of your CapSense capacitive touch sensing system.

- Set GPIO drive mode for low power
- Turnoff the high power blocks
- Optimize CPU speed for low power
- Operate at a lower V_{DD}

In addition to these methods, applying the sleep-scan method can be very effective.

5.5.2 Sleep-Scan Method

5.5.2.1 Average Power

In typical applications, the CapSense controller does not always need to be in the active state. 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 much lower than the active current.

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

$$I_{AVE} = \frac{(I_{Act} \times t_{Act}) + (I_{Slp} \times t_{Slp})}{T} \qquad \text{Equation 7}$$

Where,

I_{AVE} = Device average current

I_{Act} = Device active current

t_{Act} = Device active time

I_{Slp} = Device sleep current

t_{Slp} = Device sleep time

$T = t_{Act} + t_{Slp}$

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

$$P_{AVE} = V_{DD} \times I_{AVE} \qquad \text{Equation 8}$$

Where,

P_{AVE} = Device average power

V_{DD} = Device supply voltage

I_{AVE} = Device average current

5.5.2.2 Response Time versus Power Consumption

As illustrated in Equation 8, 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 will lead to poor CapSense button response time. As a result, the sleep time must be based on system requirements.

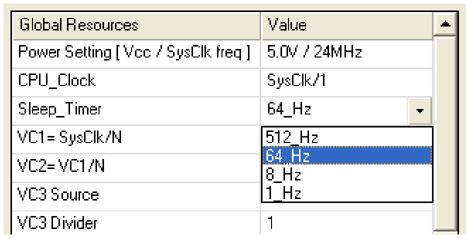
In any application, if both power consumption and response time are important parameters to be considered, then you can use an optimized method that incorporates both continuous-scan and sleep-scan modes. In this method, the device spends most of its time in sleep-scan mode where it scans the sensors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

5.5.2.3 Measuring Average Power Consumption

The following instructions describe how to determine average power consumption when using the sleep-scan method:

1. Build a project that scans all of the sensors without going to sleep (continuous-scan mode). Include a pin-toggle feature in the code before scanning the sensors. Toggling the state of the output pin serves as a time marker that can be tracked with a scope.
2. Download the project to the CapSense device and measure the current consumption. Assign the measured current to IAct.
3. Get the sleep current information from the datasheet and assign it to ISlp.
4. Monitor the toggling output pin with an oscilloscope and measure the time period between two toggles. This gives the active time. Assign this value to tAct.
5. Apply sleep-scan to the project. The time period of the sleep-scan cycle, T, is set by selecting the sleep-timer frequency in the global resources window, as shown in Figure 5-1.
6. Subtract active time from the sleep-scan cycle time period to get the sleep time. tSlp = T – tAct.
7. Calculate the average current using Equation 7.
8. Calculate the average power consumption using Equation 8.

Figure 5-1. Global Resources Window



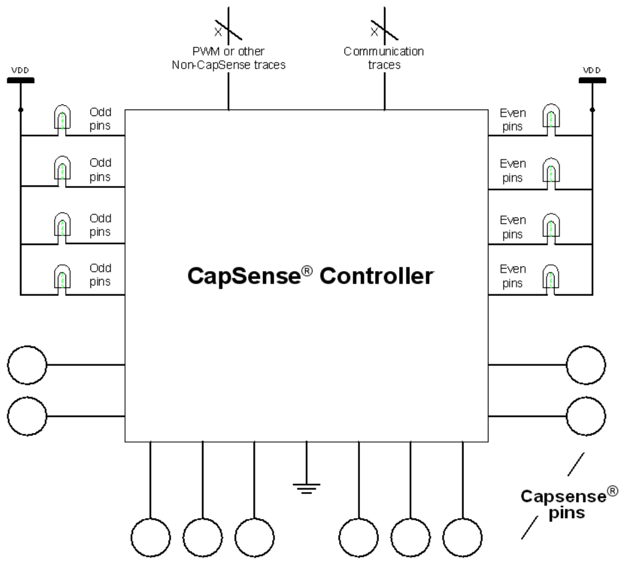
Global Resources	Value
Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/1
Sleep_Timer	64_Hz
VC1= SysClk/N	
VC2= VC1/N	
VC3 Source	
VC3 Divider	1

5.6 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 5-2 shows a basic version of this isolation for a 32-pin QFN package. Because each function is isolated, the CapSense controller is oriented so that there is no crossing of communication, LED, and sensing traces.

In devices with multiple VSS pins, all VSS pins should be brought out to one common GND plane.

Figure 5-2. Recommended: Port Isolation for Communication, CapSense, and LEDs



The architecture of the CapSense controller imposes a restriction on the current budget for even and odd port pin numbers. For a CapSense controller, if the current budget of odd port pins is 100 mA, total current drawn though all odd port pins should not exceed 100 mA. In addition to the total current budget limitation, there is a maximum current limitation for each port pin, which is defined in the CapSense controller datasheet.

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

Sensing lines in capacitive sensing applications must be connected to sensors only. Sensing lines that are attached to other board elements, such as ISSP programming headers, are more sensitive to external noise and have a higher parasitic capacitance caused by increased surface area of the conductive path. Avoid placing sensors on the programming pins, P1[0] and P1[1].

Note also that programming pins P1[0] and P1[1] respond differently to a POR or XRES event than other I/O pins do. After an XRES event, both pins are pulled down to ground by going into the resistive zero drive mode, before reaching the HI-Z drive mode. After a POR event, P1[0] drives out a one, then goes to the resistive zero state for some time, and finally reaches the HI-Z drive mode state. After POR, P1[1] goes into a resistive zero state for a while, before going to the HI-Z Drive mode. Keep this point in mind when using these pins in your application.

5.7 PCB Layout Guidelines

Detailed PCB layout guidelines are available in the Getting Started with CapSense Design Guide.

6. Resources

6.1 Website

The Cypress CapSense Controllers website gives access to all of the reference material discussed in this section.

The CY8C20x34 web page has a variety of technical resources for the CapSense CY8C20x34 family of devices.

6.2 Datasheet

The datasheet for the CapSense CY8C20x34 family of devices is available at www.cypress.com.

- CY8C20234, CY8C20334, CY8C20434, CY8C20534, CY8C20634
- CY8C20224, CY8C20324, CY8C20424, CY8C20524

6.3 Technical Reference Manual

Cypress has created the following technical reference manual to give quick and easy access to information on CapSense controller functionality, including top-level architectural diagrams, registers, and timing diagrams.

- PSoC® CY8C20x24, CY8C20x34 Technical Reference Manual (TRM)

6.4 Development Kits

6.4.1 Universal CapSense Controller - Basic Kit 1

The CY3280-BK1 Universal CapSense Controller Kit is designed for easy prototyping and debugging of CapSense designs with predefined control circuitry and plug-in hardware.

6.4.2 Universal CapSense Module Boards

6.4.2.1 Simple Button Module Board

The CY3280-BSM Simple Button Module consists of 10 CapSense buttons and 10 LEDs. This module connects to any CY3280 Universal CapSense Controller Board.

6.4.2.2 Matrix Button Module Board

The CY3280-BMM Matrix Button Module consists of eight LEDs and eight CapSense sensors organized in a 4×4 matrix format to form 16 physical buttons. This module connects to any CY3280 Universal CapSense Controller Board.

6.4.2.3 *Linear Slider Module Board*

The CY3280-SLM Linear Slider Module consists of five CapSense buttons, one linear slider (with ten sensors), and five LEDs. This module connects to any CY3280 Universal CapSense Controller Board.

6.4.2.4 *Radial Slider Module Board*

The CY3280-SRM Radial Slider Module consists of four CapSense buttons, one radial slider (with ten sensors), and four LEDs. This module connects to any CY3280 Universal CapSense Controller Board

6.4.2.5 *Universal CapSense Prototyping Module*

The CY3280-BBM Universal CapSense Prototyping Module provides access to every signal routed to the 44-pin connector on the attached controller boards. The Prototyping Module board is used in conjunction with a Universal CapSense Controller board to implement additional functionality that is not part of the other single-purpose Universal CapSense Module boards

6.4.3 In-Circuit Emulation (ICE) Kits

The ICE pod provides the interconnection between the CY3215-DK In-Circuit Emulator and the target PSoC device in a prototype system or PCB by way of package-specific pod feet through a flex cable. The following pods are available.

- CY3250-20334QFN In-Circuit Emulation (ICE) Pods (2) for Debugging QFN CY8C20334 PSoC Devices
- CY3250-20434QFN In-Circuit Emulation (ICE) Pods (2) for Debugging QFN CY8C20434 PSoC Devices

6.5 Sample Board Files

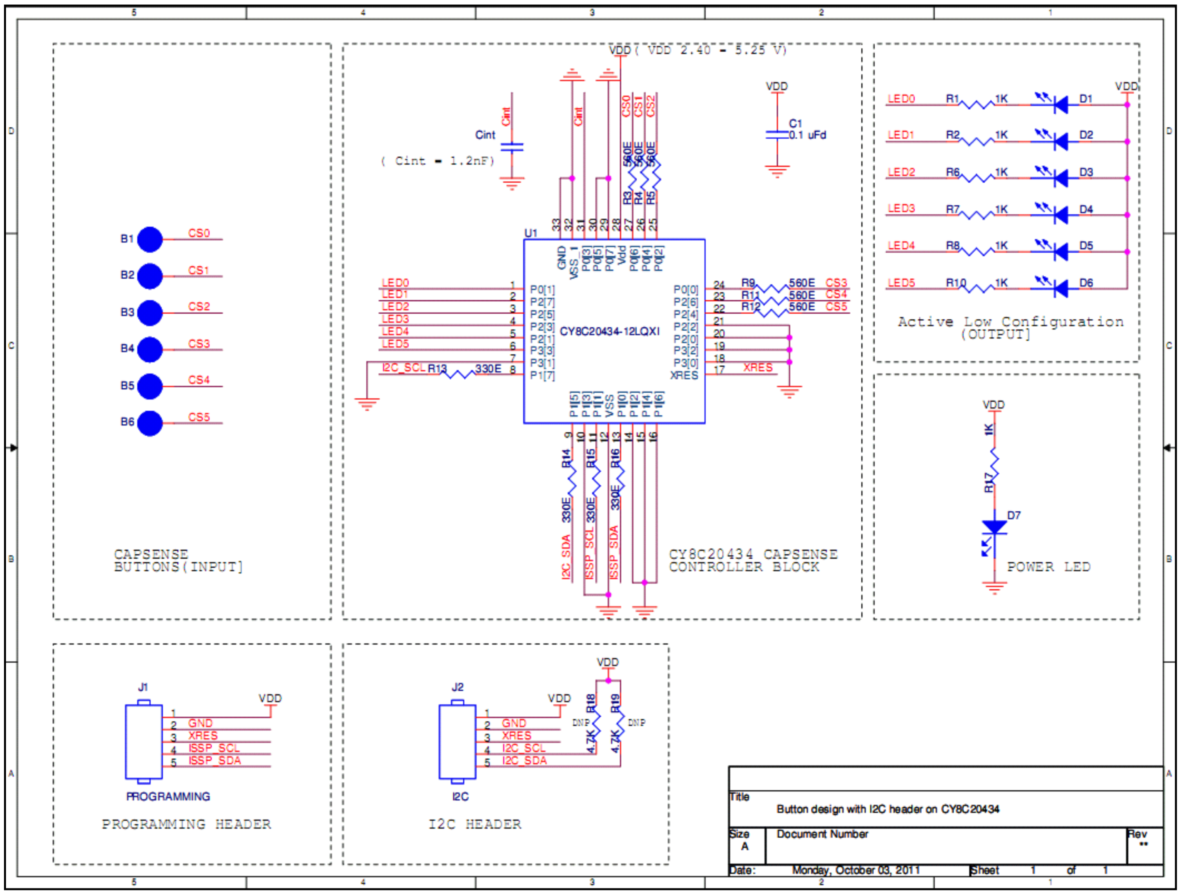
Cypress offers sample schematic and board files that you can use as a reference to quickly complete your PCB design process.

6.5.1 Button design with I^{2}C header on CY8C20434

Figure 6-1 shows the schematic of a button design with I^{2}C header on CY8C20434. This design supports:

- Six CapSense sensors. The sensors are assigned to pins P0[0], P0[2], P0[4], P0[6],P2[4], and P2[6] of the CY8C20434-12LQXI device.
- Six GPOs connected to pins P0[1],P2[1],P2[3],P2[5],P2[7], and P3[3] of CY8C20434-12LQXI to drive LEDs D1, D2,D3,D4,D5, and D6.
- Programming of CY8C20434-12LQXI by way of the programming header J1.
- I^{2}C communication with CY8C20434-12LQXI by way of the I^{2}C header J2.

Figure 6-1. Button Design with I^{2}C Header on CY8C20434 - Board Schematic

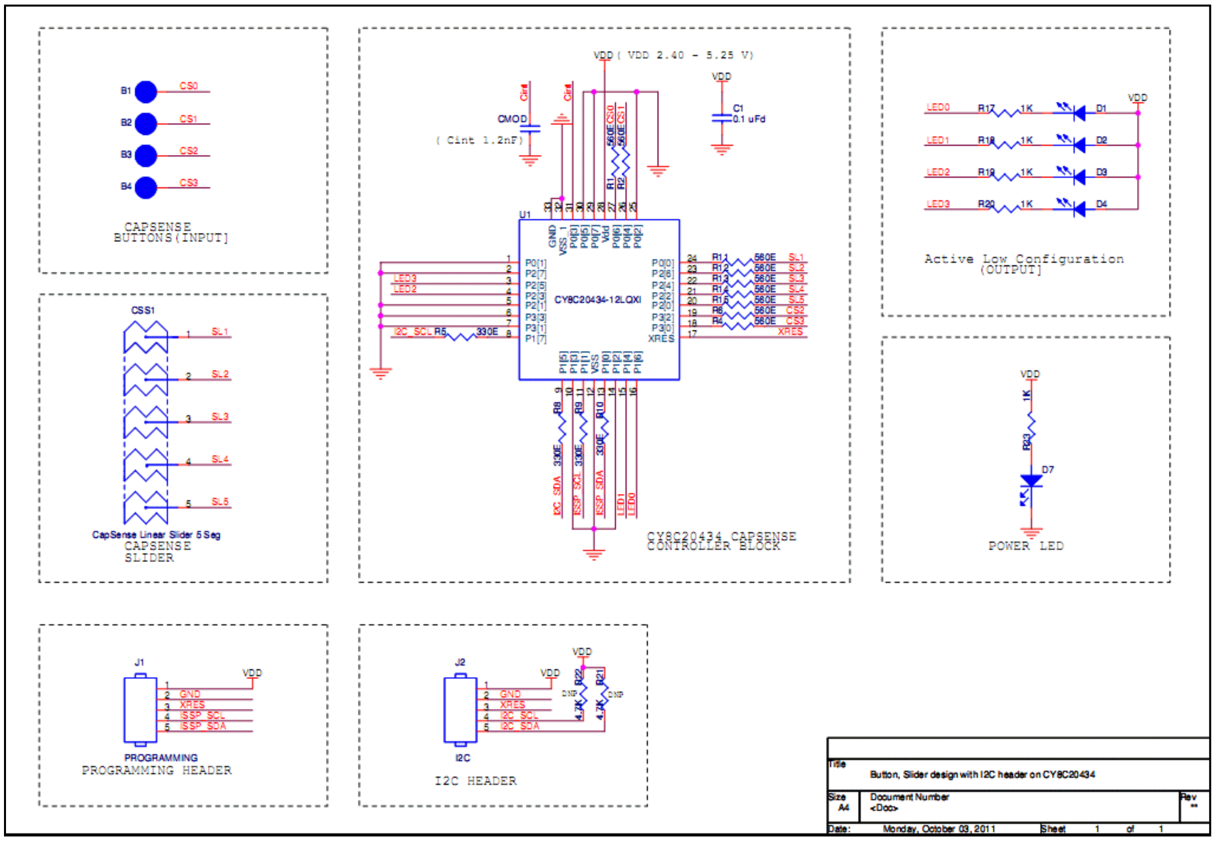


6.5.2 Button and Slider Design with I^{2}C Header on CY8C20434

Figure 6-2 shows the schematic of a button and slider design with I^{2}C header on CY8C20434. This design supports:

- Four CapSense sensors. The sensors are assigned to pins P0[4],P0[6],P3[0], and P3[2] of the CY8C20434-12LQXI device.
- Linear slider with five segments. The segments are assigned to pins P0[0],P2[0],P2[2],P2[4], and P2[6] of the CY8C20434-12LQXI device.
- Four GPOs connected to pins P1[4],P1[6],P2[5], and P2[7] of CY8C20434-12LQXI to drive LEDs D1, D2,D3, and D4.
- Programming of CY8C20434-12LQXI by way of the programming header J1.
- I^{2}C communication with CY8C20434-12LQXI by way of the I^{2}C header J2.

Figure 6-2. Button and Slider Design with I^{2}C Header on CY8C20434



6.6 PSoC Designer

Cypress offers an exclusive Integrated Design Environment, PSoC Designer. With PSoC Designer you can configure analog and digital blocks, develop firmware, and tune your design. Applications are developed in a drag-and-drop design environment using a library of pre-characterized analog and digital functions, including CapSense. PSoC Designer comes with a built-in C compiler and an embedded programmer. A pro compiler is available for complex designs.

6.7 PSoC Programmer

PSoC Programmer is a flexible, integrated programming application for programming PSoC devices. PSoC Programmer can be used with PSoC Designer and PSoC Creator to program any design onto a PSoC device.

PSoC Programmer provides a hardware layer with APIs to design specific applications that use the programmers and bridge devices. The PSoC Programmer hardware layer is fully detailed in the COM guide documentation as well as the example code across the following languages: C#, C, Perl, and Python.

6.8 CapSense Data Viewing Tools

Many times during a CapSense design, you will want to monitor relevant CapSense data (raw counts, baseline, difference counts, and so on) for tuning and debugging purposes.

Application note AN2397 – CapSense Data Viewing Tools provides necessary information that helps you to identify and use the right tools for CapSense data viewing and logging.

6.9 Code Examples

Cypress offers several code examples to get your design up and running fast. Refer to the CapSense Controller Code Examples Design Guide to evaluate different CapSense features on CapSense development kits.

6.10 Design Support

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

- Knowledge Base Articles – Refer to technical articles by product family or perform a search on various CapSense topics.
- CapSense Application Notes – Refer to a wide variety of application notes built on information presented in this document.
- White Papers – Learn about advanced capacitive touch interface topics.
- Cypress Developer Community – Connect with the Cypress technical community and exchange information.
- CapSense Product Selector Guide – See the complete product offering of the Cypress CapSense product line.
- Video Library – Quickly get up to speed with tutorial videos.
- Quality & Reliability – Find reliability and product qualification reports at our Quality website. Cypress is committed to complete customer satisfaction.
- Technical Support – World-class technical support is available online.

Glossary

AMUXBUS

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

SmartSense™ Auto-Tuning

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

Baseline

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

Button or Button Widget

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

Difference Count

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

Capacitive Sensor

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

CapSense®

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

CapSense Mechanical Button Replacement (MBR)

Cypress's configurable solution to upgrade mechanical buttons to capacitive buttons, requires minimal engineering effort to configure the sensor parameters and does not require firmware development. These devices include the CY8CMBR3XXX and CY8CMBR2XXX families.

Centroid or Centroid Position

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

Compensation IDAC

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

CSD

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

In CSD mode, the sensing system measures the self-capacitance of an electrode, and a change in the self-capacitance is detected to identify the presence or absence of a finger.

Debounce

A parameter that defines the number of consecutive scan samples for which the touch should be present for it to become valid. This parameter helps to reject spurious touch signals.

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

Driven-Shield

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

Electrode

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

Finger Threshold

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

Ganged Sensors

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

To reduce power when the system is in low-power mode, all the sensors can be ganged together and scanned as a single sensor taking less time instead of scanning all the sensors individually. When 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 Touchpad Widget.

Guard Sensor

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

Hatch Fill or Hatch Ground or Hatched Ground

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

Hysteresis

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

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

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

Liquid Tolerance

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

Linear Slider

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

Low Baseline Reset

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

Manual-Tuning

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

Matrix Buttons

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

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

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

Modulation Capacitor (CMOD)

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

Modulator Clock

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

Modulation IDAC

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

Mutual-Capacitance

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

Negative Noise Threshold

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

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

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

Noise (CapSense Noise)

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

Noise Threshold

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

Overlay

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

Parasitic Capacitance (C_P)

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

Proximity Sensor

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

Radial Slider

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

Raw Count

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

Refresh Interval

The time between two consecutive scans of a sensor.

Scan Resolution

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

Scan Time

Time taken for completing the scan of a sensor.

Self-Capacitance

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

Sensitivity

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

Sense Clock

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

Sensor

See Capacitive Sensor.

Sensor Auto Reset

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

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

Sensor Ganging

See Ganged Sensors.

Shield Electrode

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

Shield Tank Capacitor (C_{SH})

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

Signal (CapSense Signal)

Difference Count is also called Signal. See Difference Count.

Signal-to-Noise Ratio (SNR)

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

Slider Resolution

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

Touchpad

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

Trackpad

See Touchpad.

Tuning

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

V_{REF}

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

Widget

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

Document Revision History

Document Revision History

Document Title: AN66269 - CY8C20x34 CapSense® Design Guide **Document Number: 001-66269**			
Revision	**Issue Date**	**Origin of Change**	**Description of Change**
**	12/29/2010	KPOL	New Design Guide
*A	03/03/2011	KPOL	Multiple chapter enhancements for content and reader clarity
*B	05/12/2011	KPOL	Additional information to support CY8C20x24 device. 1) Under chapter "Introduction" section "CY8C20x34 CapSense Family Features" 2) Under chapter "Resources" section "Technical Reference Manual"
*C	06/14/2011	KPOL	Incorrect revision number mentioned as *A on the first page. Updated to *C version.
*D	12/01/2011	KPOL	Enhancements to Introduction, CapSense Technology, CSA_EMC User Module Parameters, CSA_EMC User Module – Tuning Guide. Removed mention of Multi-Chart. Minor changes throughout.
*E	08/30/2012	ZINE	Updated links to external documents
*F	09/03/2014	PRKU/DCHE	Added a note to Section 5.6: Pin Assignments. Updated all the equations
*G	01/19/2015	DIMA	No technical updates. Completing Sunset Review.
*H	07/28/2015	DCHE	Fixed formatting issue with Figure 2-1 Updated hyperlinks in the Design Support section.
*I	01/21/2016	VAIR	Added Glossary.
*J	07/21/2017	AESATMP8	Updated logo and Copyright.