PSoC 4 CapSense Hybrid Sensing

About this document

Scope and purpose

This code example demonstrates how to use the PSoC® Creator™ CapSense® Component to implement CSD and CSX sensing on the same device, and how to use CSD and CSX sensing on the same sensor.

Requirements

Tool: **PSoC Creator** 4.4

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: PSoC 4 devices that can support a PSoC Creator CapSense component with Self-Capacitance (CSD) and Mutual-Capacitance (CSX). The following devices support both CSD and CSX: PSoC **4000**, **4000S**, **4100 BLE**, **4100PS**, **4100S**, **4100S Plus**, **4200**, **4200DS**, **4200L**, **4200M**, **4700S**

Related Hardware:

- CapSense Hybrid Buttons project: **CY8CKIT-149 PSoC 4100S Plus Prototyping Kit**
- Low Power CapSense Hybrid Sensing project: **CY8CKIT-041-41XX PSoC 4100S Pioneer Kit**

Table of contents

1 Overview 2
2 CapSense Sigma Delta (CSD) 3
3 CapSense Crosspoint (CSX) 4
4 Hardware Setup 5
5 Operation 6
5.1 CapSense Hybrid Buttons 6
5.2 Low Power CapSense Hybrid Sensing 6
6 Design and Implementation 8
6.1 Components and Settings 11
7 Reusing This Example 16
References 17
Revision history 18

1 Overview

This example contains two projects. The first project uses the two buttons that are implemented through the PSoC Creator CapSense Component. One button is set up using CSD, while the second button uses CSX. An LED lights up when each button is pushed.

The second project uses the CapSense Component to set up three different interfaces (widgets): a touchpad, a linear slider, and a button. Both the linear slider and the button are ganged to the sensors that make up the columns of the touchpad widget. The touchpad uses CSX sensing while the linear slider and button use CSD sensing. The device goes into three different modes; each one scans only one widget and each mode uses a different amount of power. The device switches between modes depending on the activity of the scanned widget. For more information, see **References**.

2 CapSense Sigma Delta (CSD)

This block works by first converting the sensor capacitance into an equivalent current. An analog multiplexer then selects one of the currents and feeds it into the current-to-digital converter. This current-to-digital converter consists of a sigma-delta converter, which controls the modulation IDAC for a specific period; the total current sourced or sinked by the IDACs is the same as the total current sinked or sourced by the sensor capacitance. The digital count output of the sigma-delta converter is an indicator of the sensor capacitance and is called the raw count. This block can be configured in either IDAC Sourcing mode or IDAC Sinking mode. In IDAC Sourcing mode, the IDACs source current to the AMUXBUS, while the GPIO cells sink current from the AMUXBUS. In IDAC Sinking mode, the IDACs sink current from the AMUXBUS while the GPIO cells source current to the AMUXBUS. For more information, see the **CapSense Design Guide**.

3 CapSense Crosspoint (CSX)

With CSX, a voltage on the Tx pin (or Tx electrode) couples the 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 channels and feeds it into the current-to-digital converter.

The output count of the current-to-digital converter, known as **RawcountCounter**, is a digital value that is proportional to the mutual-capacitance between the Rx and Tx electrodes as follows:

Equation 1. Raw Count and Sensor Capacitance Relationship in CSX

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

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

When a finger touches the sensor, C_M decreases which then decreases the counter output. 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, the logic in firmware determines whether the sensor is active (finger is present). For more information, see the **CapSense Design Guide**.

4 Hardware Setup

The ‘CapSense Hybrid Buttons’ project is set up for CY8CKIT-149 and the ‘Low Power CapSense Hybrid Sensing’ project is set up for CY8CKIT-041-41XX. If you are using a different kit, see **Reusing this Example.**

5 Operation

5.1 CapSense Hybrid Buttons

1. Connect CY8CKIT-149 to your computer using a USB cable.
2. Build the project and program it into the PSoC 4 device. Choose **Debug > Program**. For more information on device programming, see *PSoC* Creator Help.
3. Touch BTN0, which implements CSX sensing, and confirm that the green LED to the left of the button turns ON.
4. Touch BTN1, which implements CSD sensing and confirm that the green LED to the left of the button turns ON.
5. Ensure both LEDs turn OFF when you remove your finger from the buttons.
6. Right-click the CapSense Component and select **Launch Tuner**. Click **Connect**, select **I2C**, and then click **Start**. Ensure that the data rate is set to 400 kHz. Touch BTN0 and confirm that the signal increases for Button0 on the **Touch Signal Graph**. Touch BTN1 and confirm that the signal increases for Button1 on the **Touch Signal Graph**. For more information, check the CapSense Component datasheet under

5.2 Low Power CapSense Hybrid Sensing

1. Connect CY8CKIT-041-41XX to your computer using a USB cable.
2. Build the project and program it into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming, see *PSoC* Creator Help.
3. When the device is started:
   a) Touch the touchpad and confirm that as your finger moves upward, the green LED gets brighter. As your finger moves downward, confirm that the green LED gets dimmer.
   b) Move your finger to the right and confirm that the blue LED gets brighter. As your finger moves to the left and confirm that the blue LED gets dimmer.
4. Remove the finger from the touchpad and note that the blue and green LEDs have turned OFF. This signifies that the device is in the lowest power mode and all sensors that make up the touchpad's columns are configured as one a button.
5. Tap the touchpad and confirm that the red LED turns ON. This signifies that the device is in a low-power mode and the sensors are configured as a slider.
6. Slide your finger horizontally from one side of the touchpad to the other. Confirm that the LED changes from red to blue and green. This signifies that the device is out of low-power mode and the sensors are configured as a touchpad.
   a) If no touch is detected for 10 seconds, the device will return to the lowest-power mode and the red LED is turned OFF.
   b) When the device is out of the low-power mode, if the device is not touched for five seconds, the device will turn OFF the blue and green LEDs. This signifies that the device is in the lowest power mode.
7. Right-click the CapSense Component and select **Launch Tuner**. Click **Connect**, select **I2C**, and then click **Start**. Ensure that the data rate is set to **400 kHz**. Touch the touchpad while the blue and green LEDs are ON and notice that only the touchpad widget is active on the **Touch Signal Graph**. Remove your finger and wait until the both the blue and green LEDs turn OFF. Tap the touchpad and confirm that only the button widget is active, and the red LED turns ON. When the red LED is ON, confirm that only the linear slider widget is active when the touchpad is touched until a finger is slid horizontally

across the touchpad which will activate the touchpad widget. For more information, check the CapSense Component datasheet under **References**.

8. A widget may look active in the CapSense Tuner even though that widget is not scanned. This is because when enough activity is sensed on a widget, the device will change scan modes immediately, which means that the device will continue to send the last scanned data (that shows the sensor is active) from the widget to the tuner until that widget is scanned again.

6 Design and Implementation

CapSense Hybrid Buttons: The PSoC Creator CapSense Component scans both button widgets. One widget is scanned using CSD and the other widget is scanned using CSX to show that different types of sensing can occur on different sensors at the same time. Two LEDs are set up, one for each button to turn ON if a button is touched and turned OFF if the button is not touched.

In the CapSense_Hybrid_Buttons example, the following functions are performed:

1. Initializes and starts all hardware.
2. Links the communication buffer for EZI2C to the CapSense data structure.
3. Initial scan of all CapSense widgets.
4. If CapSense is not busy, processes all data.
5. Turns the LEDs ON or OFF depending on button activity.
6. Sends all data to the tuner.
7. Scans all CapSense widgets and returns to Step 4.

Figure 1 shows the top-level of the PSoC Creator project:

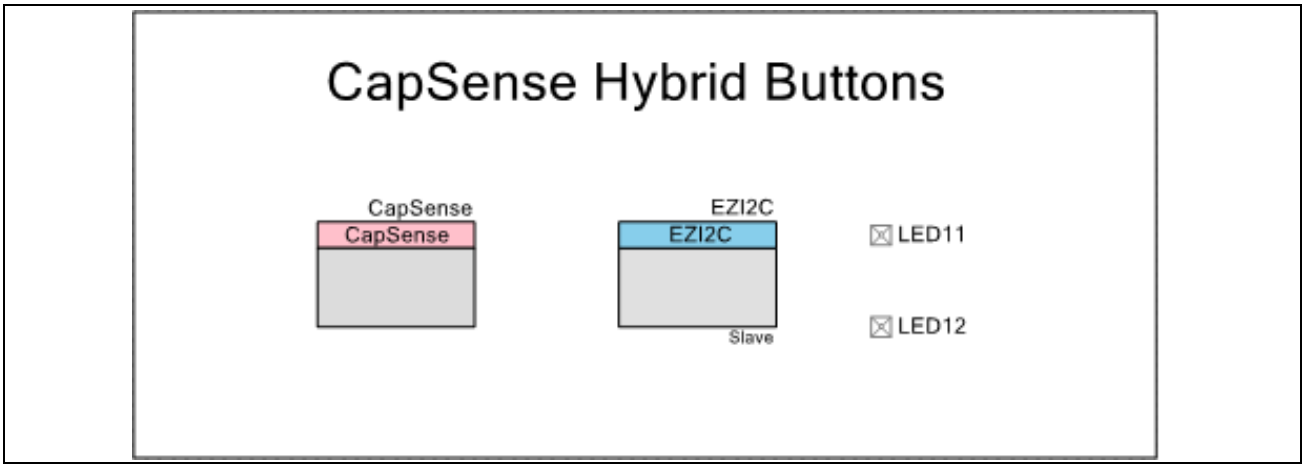


Figure 1 CapSense_Hybrid_Buttons

Low Power CapSense Hybrid Sensing: The PSoC Creator CapSense Component sets up three different widgets: a touchpad that uses CSX sensing, a linear slider that uses CSD sensing, and a button that uses CSD sensing. The device scans only one widget at a time and goes into three different power levels, one for each widget scanned. When scanning the touchpad, the device stays in the active power mode, which allows the widget to be scanned fast. When scanning the linear slider, the device goes into Deep Sleep and wakes up every 50 ms to scan the widget. This limits the linear slider to being scanned 20 times a second. When scanning the button, the device goes into Deep Sleep and wakes up every 100 ms to scan the widget. This limits the button to being scanned 10 times a second.

The firmware changes the widget that is being scanned based on the activity of the widgets. The device starts with fast scanning of the touchpad widget. If there is no activity on the widget, the firmware changes the scanned widget to the button. When the device scans the button, it also goes into Deep Sleep and wakes up at 100 ms intervals. If there is activity on the button, the firmware will change the scanned widget to the linear slider, the device will also go into Deep Sleep and wake up every 50 ms. If there is no activity on the sensor, the firmware will switch back to scanning the button. If there is a 30-position count

Design and Implementation

change on the slider, the device changes to the touchpad mode. **Figure 2** illustrates the flow of the firmware.

In the Low_Power_CapSense_Hybrid_Sensing example, the following functions are preformed:

1. Initializes and starts all hardware
2. Sets current state to SENSOR_SCAN and device scan mode to TOUCHPAD_SCAN_MODE
3. Sets up the watch dog timer to keep track of time for both slow and fast scanning modes and wakes the device up when in Deep Sleep.
4. Sets up touchpad widget to be scanned first
5. Enters a state machine with state SENSOR_SCAN.
   a) I2C transfers all CapSense scan data to the CapSense tuner
   b) Checks the device scan mode, and scans the current widget based of the current device scan mode.
   c) Changes the current state to WAIT_FOR_SCAN_COMPLETE
6. Enters a state of WAIT_FOR_SCAN_COMPLETE
   a) Puts the CPU to Sleep and waits until the scan is complete
   b) Changes the current state to PROCESS_DATA
7. Enters a state of PROCESS_DATA
   a) Sets the current state to SLEEP.
   b) If device scan mode is set to TOUCHPAD_SCAN_MODE:
      - Sets the current state to SENSOR_SCAN as the device never goes into Deep Sleep in this mode.
      - Processes all data for the touchpad and checks if the touchpad widget is active.
      - If the touchpad is active, gets the current position of the finger touch and changes the PWM's compare value based off finger position to control the blue and green LEDs. Sets the soft counter to 0.
      - If the touchpad is not active, adds one to the soft counter and checks if the soft counter has exceeded the MAX_SOFT_COUNT. If it has exceeded the maximum count, turns OFF the LEDs, sets up the button widget to be scanned, sets the device scan mode to BUTTON_SCAN_MODE, and sets the watchdog match value to slow scan.
   c) If device scan mode is set to SLIDER_SCAN_MODE:
      - Processes all data for the slider and check if the slider is active.
      - If the slider is active, gets the position of the slider the first time the widget goes active and the position of the slider every time the sensor is active. Resets the soft counter and checks if the position of the first touch is 30 positions different from the current touch. If so, sets up the touchpad widget and sets the device scan mode to TOUCHPAD_SCAN_MODE.
      - If the slider is not active, adds one to the soft counter and checks if the soft counter has exceeded the MAX_SOFT_COUNT. If it has, turns the red LED OFF, sets up the button widget to be scanned next, sets the device scan mode to BUTTON_SCAN_MODE, and sets the watchdog match value to slow scan.
   d) If device scan mode is set to BUTTON_SCAN_MODE:
      - Processes all data for the button and checks if the button is active.
      - If the button is active, turns the red LED ON, sets the soft counter to 1, sets up the linear slider widget to be scanned next, sets the device scan mode to SLIDER_SCAN_MODE, and sets the scan mode to fast scan.
8. Enters a state of Sleep.

Design and Implementation

a) Puts the device into Deep Sleep.

b) Checks if the watchdog timer interrupt occurred; if it did occur, change the current state to SENSOR_SCAN and changes the watchdog timer interrupt flag to false.

9. Repeats the process from SENSOR_SCAN at Step 5.

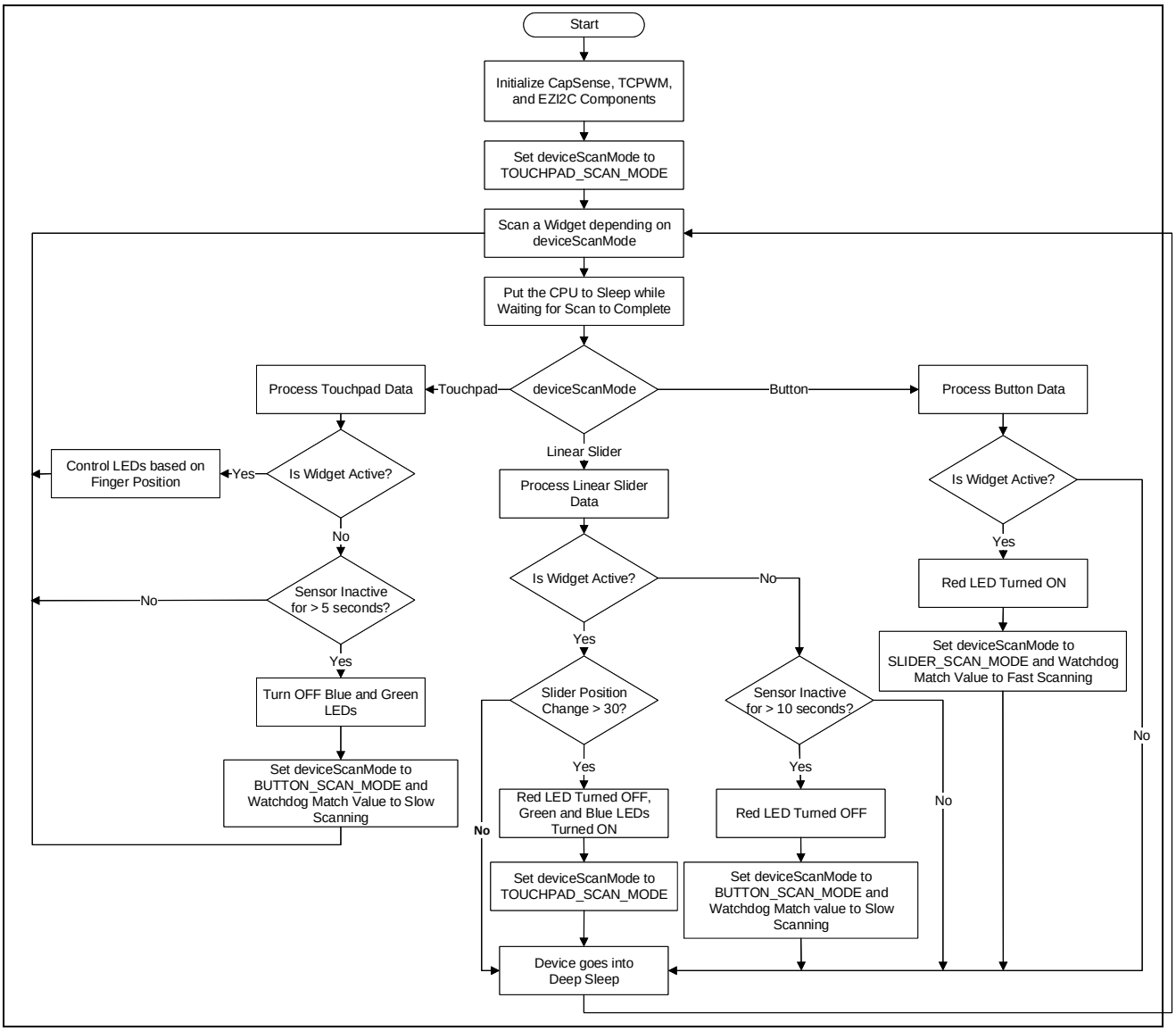


Figure 2 Firmware Flowchart

6.1 Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1 PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
CapSense	CapSense	Gather and process all data sent from CapSense sensors	CapSense_Hybrid_Buttons
			For Basic settings, see **Figure 3**.
			Low_Power_CapSense_Hybrid_Sensing
			For Basic settings, see **Figure 4**. For General settings, see **Figure 5**. Enable common Tx clock under CSX Settings. For Touchpad Widget Details, see **Figure 6**. For LinearSlider Widget Details, see **Figure 7**. For Button Widget Details, see **Figure 8**.
EZI2C	EZI2C	Transmits data from the device to the tuner	CapSense_Hybrid_Buttons
			Change the Data rate to 400 (kbps) and the Sub-address size to 16.
			Low_Power_CapSense_Hybrid_Sensing
			Change the Data rate to 400 (kbps), the Sub-address size to 16, and turn ON Enable wakeup from Deep Sleep Mode.
Low_Power_CapSense_Hybrid_Sensing only			
Global Signal Reference	WDT	Wake up device from Deep Sleep	No change.
PWM	PWM_Blue, PWM_Green	Control the duty cycle of the green and blue LEDs	For both PWMs setup, see **Figure 9**.

Note: CapSense threshold parameters are derived by following the tuning process in the **CapSense Design Guide**.

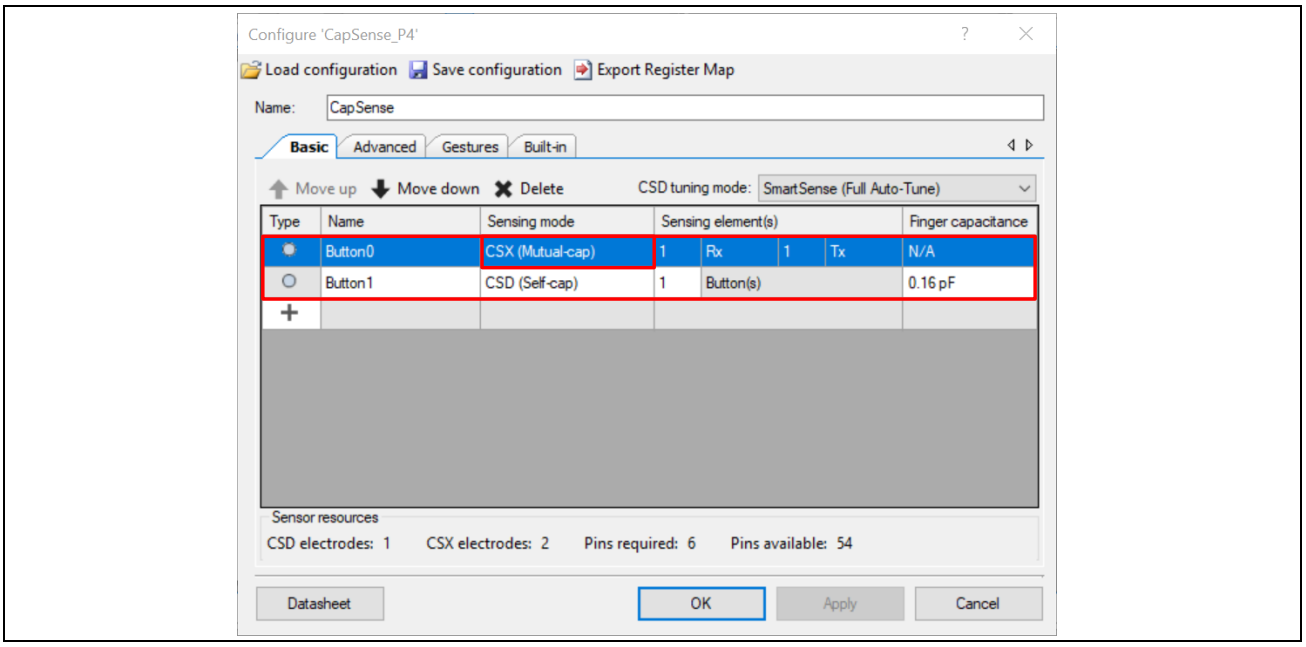


Figure 3 CapSense_Hybrid_Buttons Basic Settings

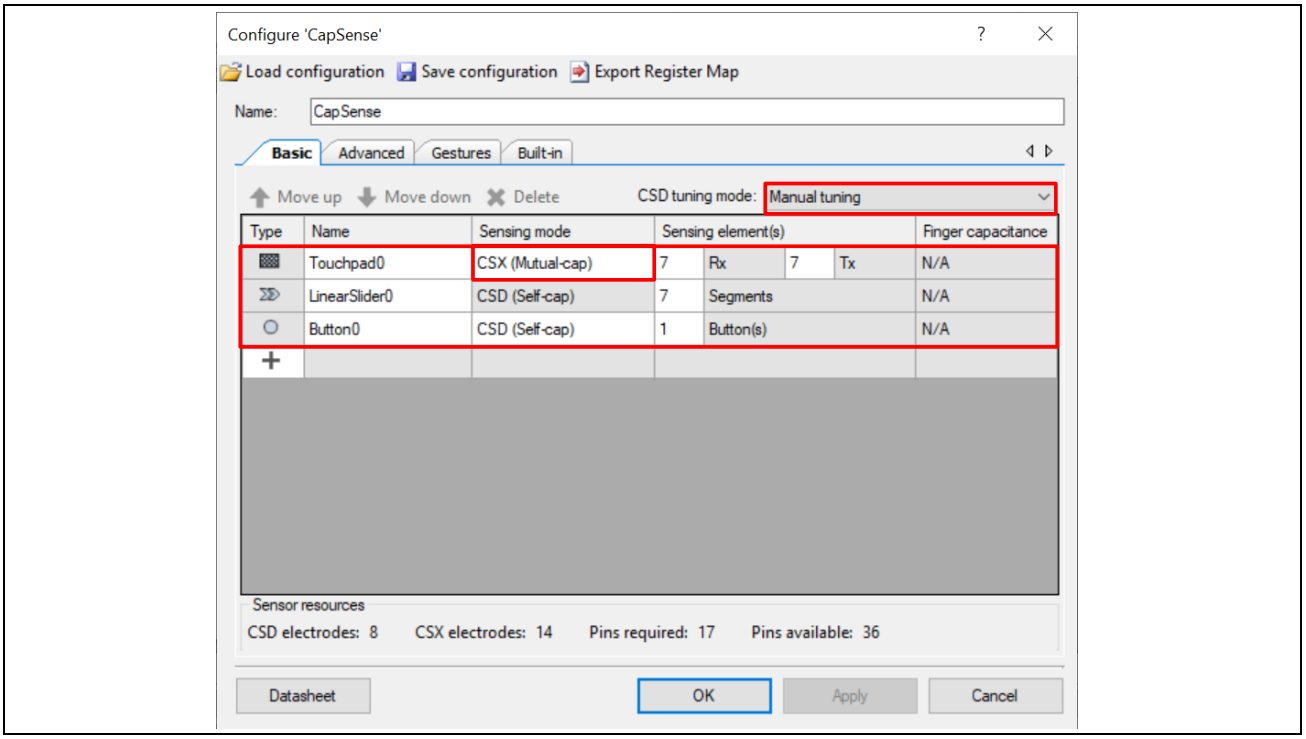


Figure 4 Low_Power_CapSense_Hybrid_Sensing Basic Settings

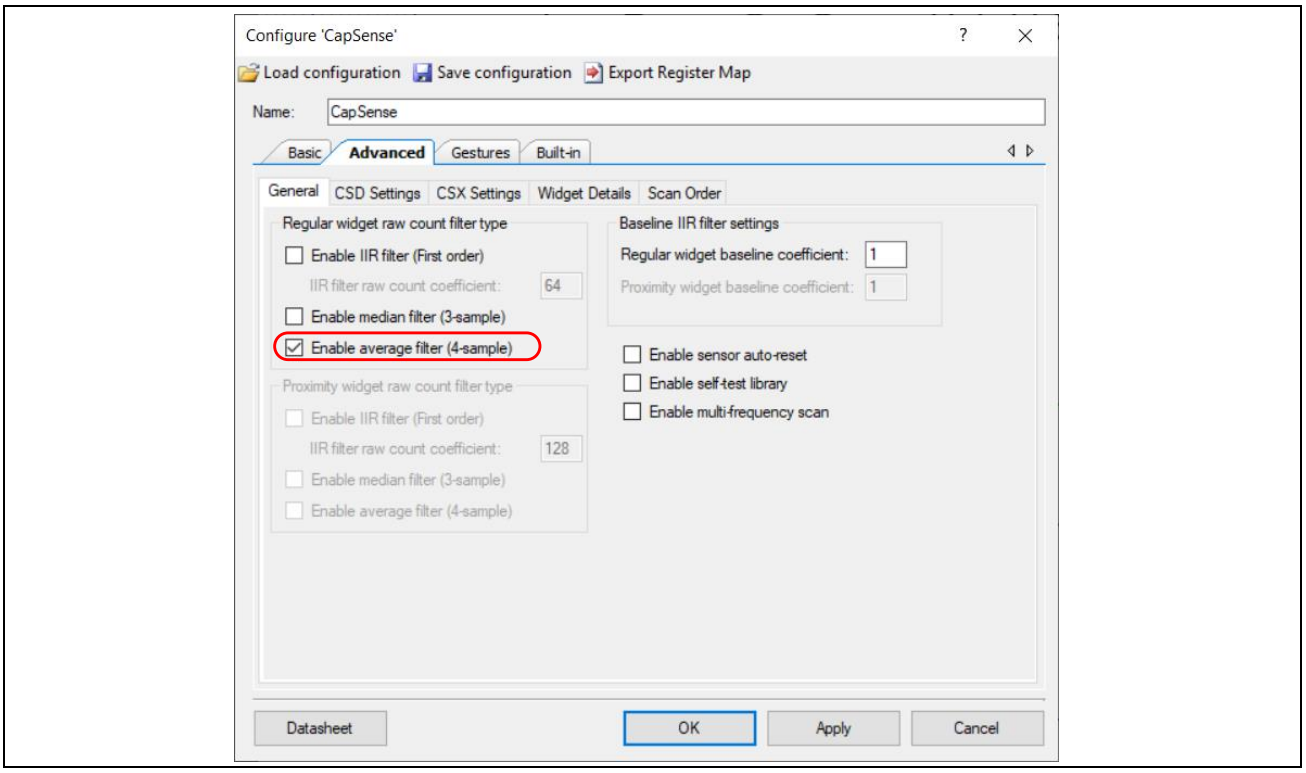


Figure 5 Low_Power_CapSense_Hybrid_Sensing General Settings

Note: The average filter is enabled to reduce noise.

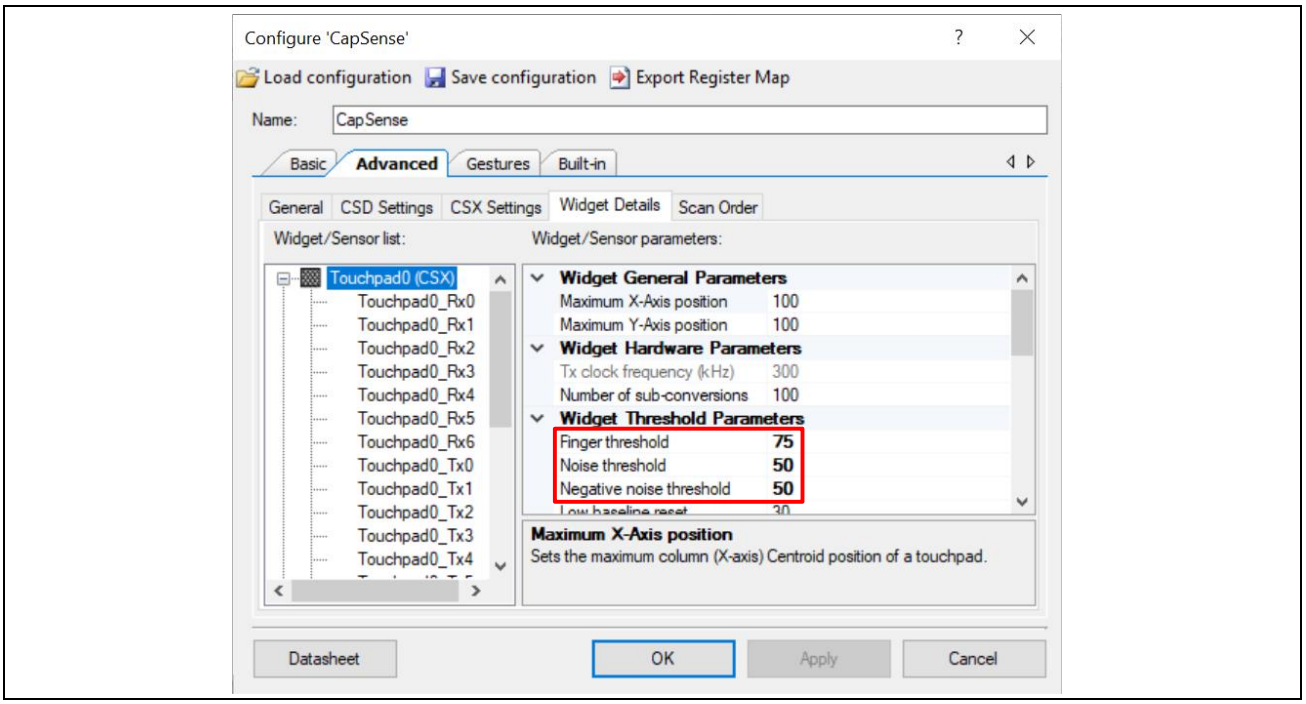


Figure 6 Low_Power_CapSense_Hybrid_Sensing Touchpad Widget Details

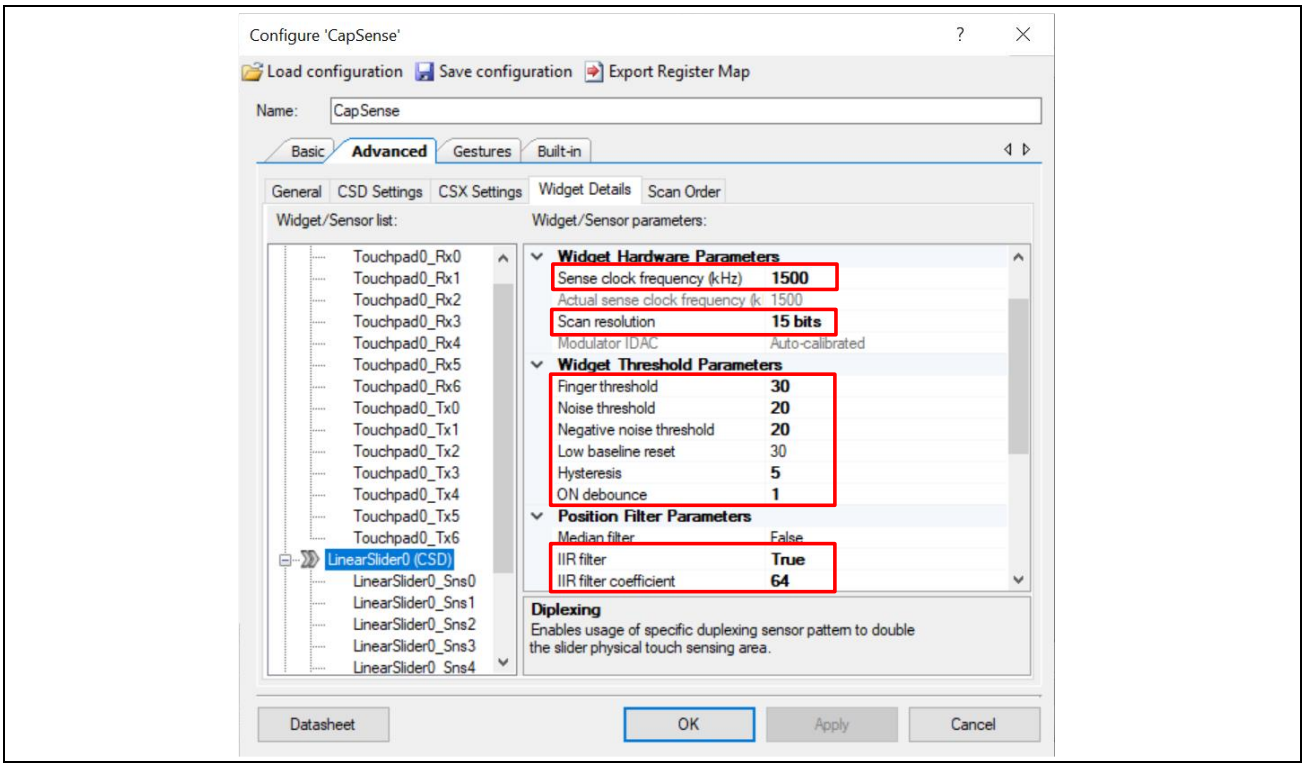


Figure 7 Low_Power_CapSense_Hybrid_Sensing LinearSlider Widget Details

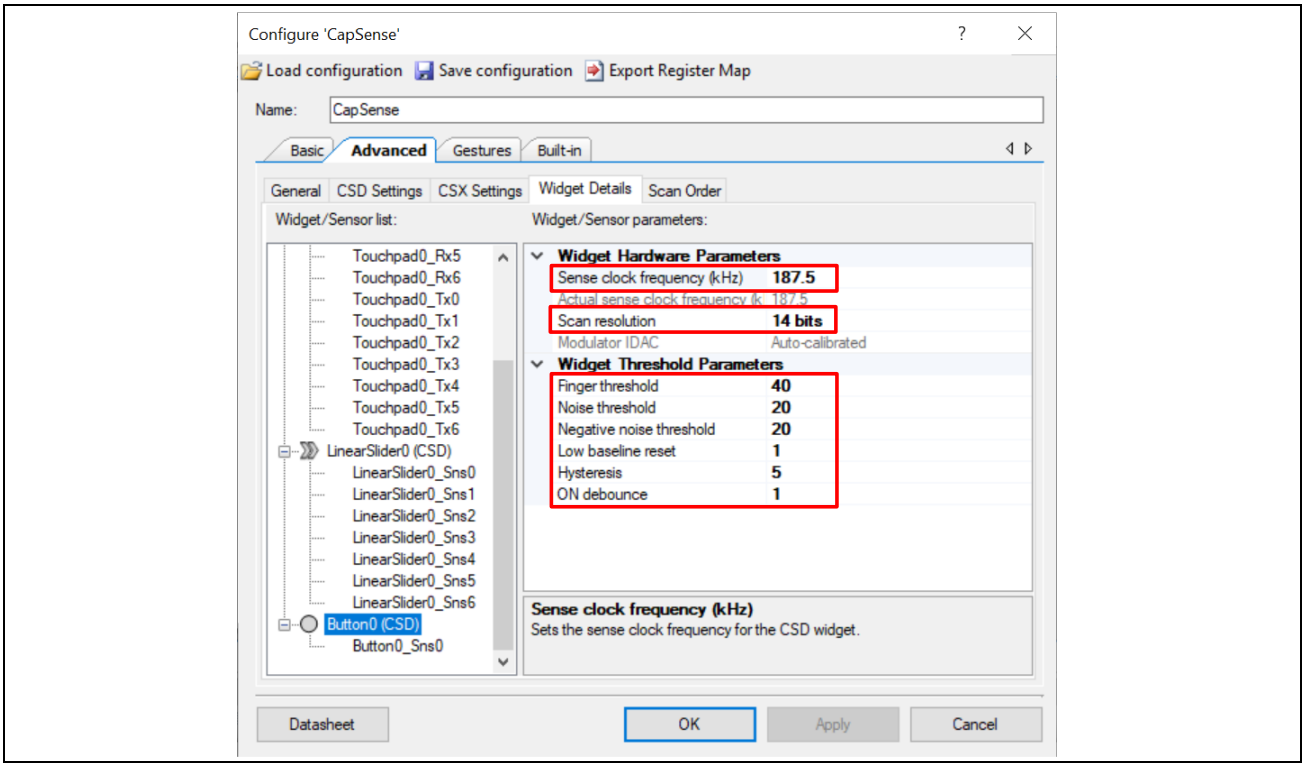


Figure 8 Low_Power_CapSense_Hybrid_Sensing Button Widget Details

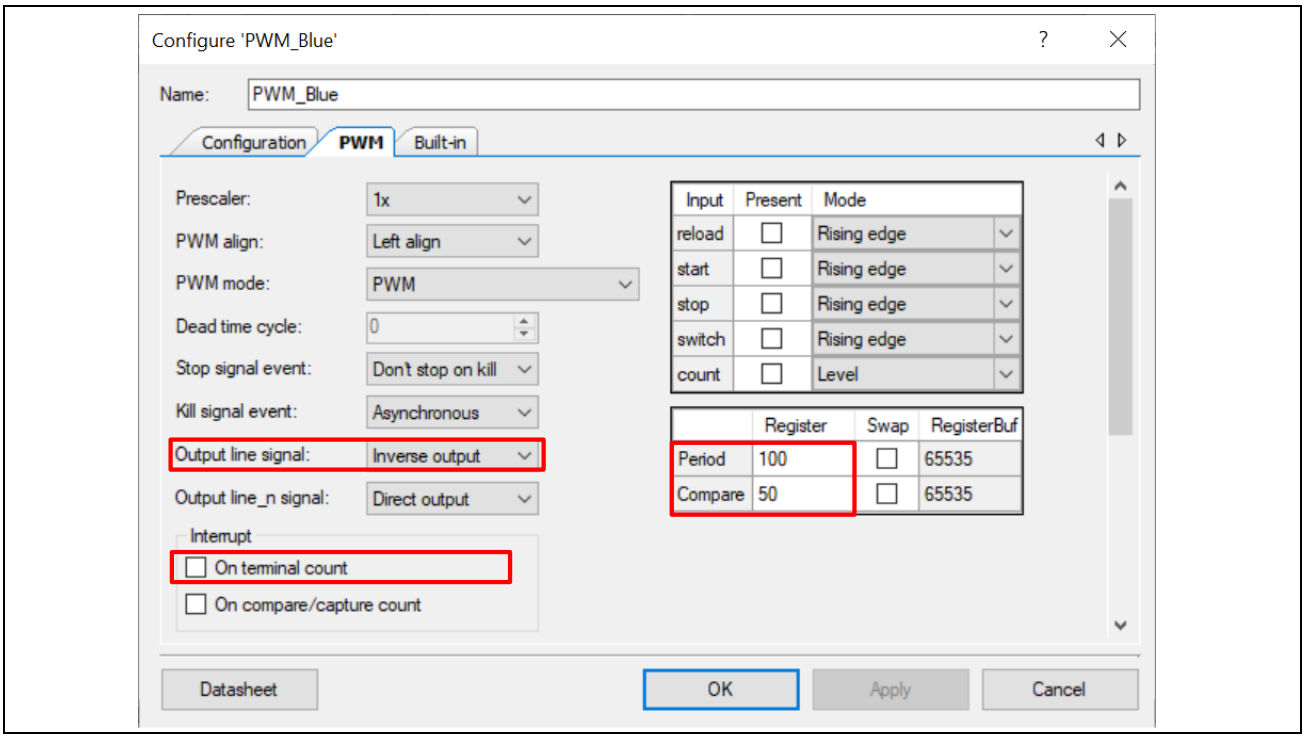


Figure 9 Low_Power_CapSense_Hybrid_Sensing PWM Settings

For information on the hardware resources used by a Component, see the Component datasheet.

7 Reusing This Example

The CapSense_Hybrid_Buttons project can be used with two kits, CY8CKIT-149 PSoC 4100 Plus Prototyping Kit and CY8CKIT-145-40xx PSoC 4000S Prototyping Kit. If these kits are used, the tuning parameters do not need to be adjusted.

The Low_Power_CapSense_Hybrid_Sensing project can be used with two kits: CY8CKIT-041-41XX PSoC 4100S Pioneer Kit and CY8CKIT-041-40XX PSoC 4000S Pioneer Kit. If these kits are used, the tuning parameters do not need to be adjusted.

Ensure the following before porting one of these code examples:

1. One of the kits listed is used.
2. All pins are unlocked in the Design Wide Resources.

To port the code example to a device that is not listed, in PSoC Creator select **Project** > **Device Selector** and change to the target device. Before porting this example to another device, note the following:

1. Not all PSoC 4 devices can support a CapSense Component with both CSD and CSX sensing. See Associated Parts under the Requirements section for devices that support both CSD and CSX.
2. Pinouts change from device to device. Some pins may need to be moved. See the **Pin Layout** tab in PSoC Creator.

Note: *If porting this code example to a device that is not listed, the code and tuning parameters will need to be adjusted for the specific device.*

In some cases, a resource used by a code example (for example, a Universal Digital Block) is not supported on another device. In that case, the example does not work. If you build the code targeted at such a device, errors occur. See the device datasheet for information on what a particular device supports.

References

For a comprehensive list of PSoC 3, PSoC 4, and PSoC 5LP resources, see **KBA86521** in the Cypress community.

Application Notes

[1] **AN79953** – Getting Started with PSoC 4: Describes PSoC 4 devices and how to build your first PSoC Creator project.

[2] **AN85951** – PSoC 4 and PSoC 6 MCU CapSense Design Guide: Describes how to tune and use the CapSense Component.

Code Examples

[3] **PSoC 4 CapSense Code examples:** Examples that use the CapSense component with PSoC 4

PSoC Creator Component Datasheets

[4] **CapSense:** CapSense Component datasheet for more information

[5] **EZI2C:** SCB Component datasheet for more information on EZI2C

[6] **Watchdog Timer:** Global Signal Reference datasheet for more information on the Watchdog Timer

[7] **TCPWM:** TCPWM Component datasheet for more information

Device Documentation

[8] **PSoC 4 MCU Datasheets**

[9] **PSoC 4 Technical Reference Manuals**

Development Kit Documentation

[10] **CY8CKIT-149 PSoC 4100 Plus Prototyping Kit**

[11] **CY8CKIT-145-40xx PSoC 4000S Prototyping Kit**

[12] **CY8CKIT-041-41XX PSoC 4100S Pioneer Kit**

[13] **CY8CKIT-041-40XX PSoC 4000S Pioneer Kit**

[14] **PSoC 4 Kits**

Tool Documentation

[15] **PSoC Creator:** Look in the downloads tab for Quick Start and User Guides

Revision history

Document version	Date of release	Description of changes
**	2019-07-26	New code example
*A	2020-12-02	Updated code example template.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-12-02

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2020 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Go to www.cypress.com/support

Document reference
002-24899 Rev. *A

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.