CE210290 - PSoC 4 CapSense Low-Power Ganged Sensor

Objective

This code example demonstrates how to implement a low-power CapSense® linear slider with an average current consumption of 6 µA.

Requirements

Tool: PSoC Creator™ 4.2 and later versions

Programming Language: C (Arm® GCC 5.4.1, Arm MDK)

Associated Parts: All PSoC 4, PSoC 4 BLE, and PRoC BLE devices

Related Hardware: CY8CKIT-042 and CY8CKIT-042 BLE

Overview

This code example implements a five-segment CapSense linear slider interface. Using the low-power modes available in the PSoC® 4 device, an average current of 6 µA is achieved when no touch is present. The example supports the CapSense tuner to read CapSense debug data such as raw count, baseline, and difference count.

Hardware Setup

There is no hardware setup required for this code example.

Software Setup

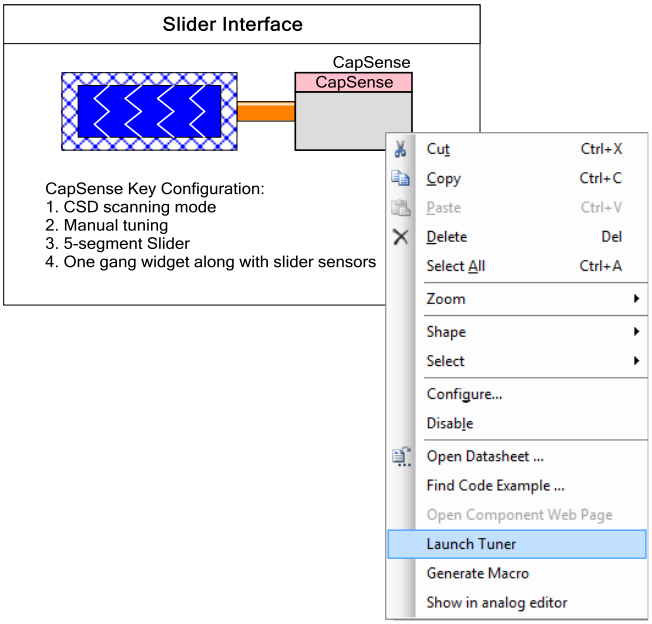
There is no software setup required for this code example.

Operation

1. Plug the kit into your computer's USB port.
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. Remove jumper J13 on CY8CKIT-042 (or jumper J15 on CY8CKIT-042 BLE) and connect an ammeter to measure the device current. Refer to the kit user guide for complete details on power measurement steps.
4. Measure the current consumption without touching the slider and note the value. Confirm that when the slider is not touched for more than three seconds, the current consumption is less than 6 µA.
5. Touch the slider and confirm that the device switches from Slow Scan mode to Fast Scan mode by observing the increase in average current on the ammeter. Confirm that when the slider is touched, the current consumption is less than 120 µA.
6. Release the finger on the slider and wait for three seconds. Confirm that the device switches back to Slow Scan mode by observing the reduction in average current on the ammeter.
7. By default, the Tuner is disabled in the firmware. Enabling the tuner means additional code will execute, which increases power consumption. To enable the tuner, set the `TUNER_UPDATE_ENABLE` macro in *main.c* to '1'. Build the project and program the device.

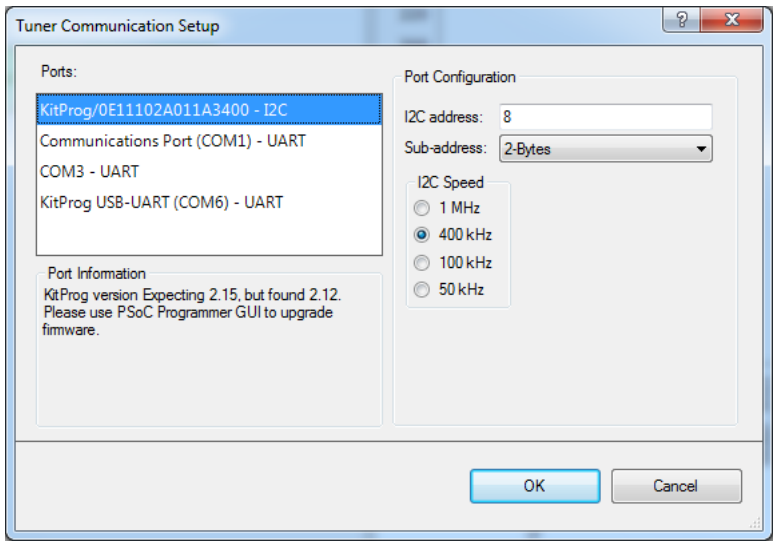
8. To launch the Tuner GUI, right-click the CapSense Component and select **Launch Tuner** from the menu, as Figure 1 shows.

Figure 1. Launch CapSense Tuner



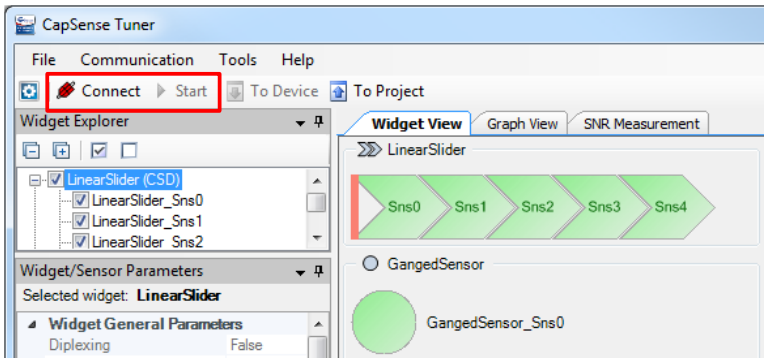
9. Navigate to **Tools** > **Tuner Communication Setup** in the Tuner GUI menu to set up I²C communication, as Figure 2 shows.
   - Select the kit with the I2C port.
   - Set the I2C address, sub-address size, and speed

Figure 2. Set Up I²C Communication



10. In the Tuner GUI, click **Connect** followed by **Start**, as Figure 3 shows.

Figure 3. Start Communication



11. After establishing I²C communication between the device and Tuner GUI, observe the sensor data and detect the slider position when touching the slider. Figure 4, and Figure 5, show example sensor debug data such as difference count, raw count, and baseline.

Figure 4. Tuner Widget View

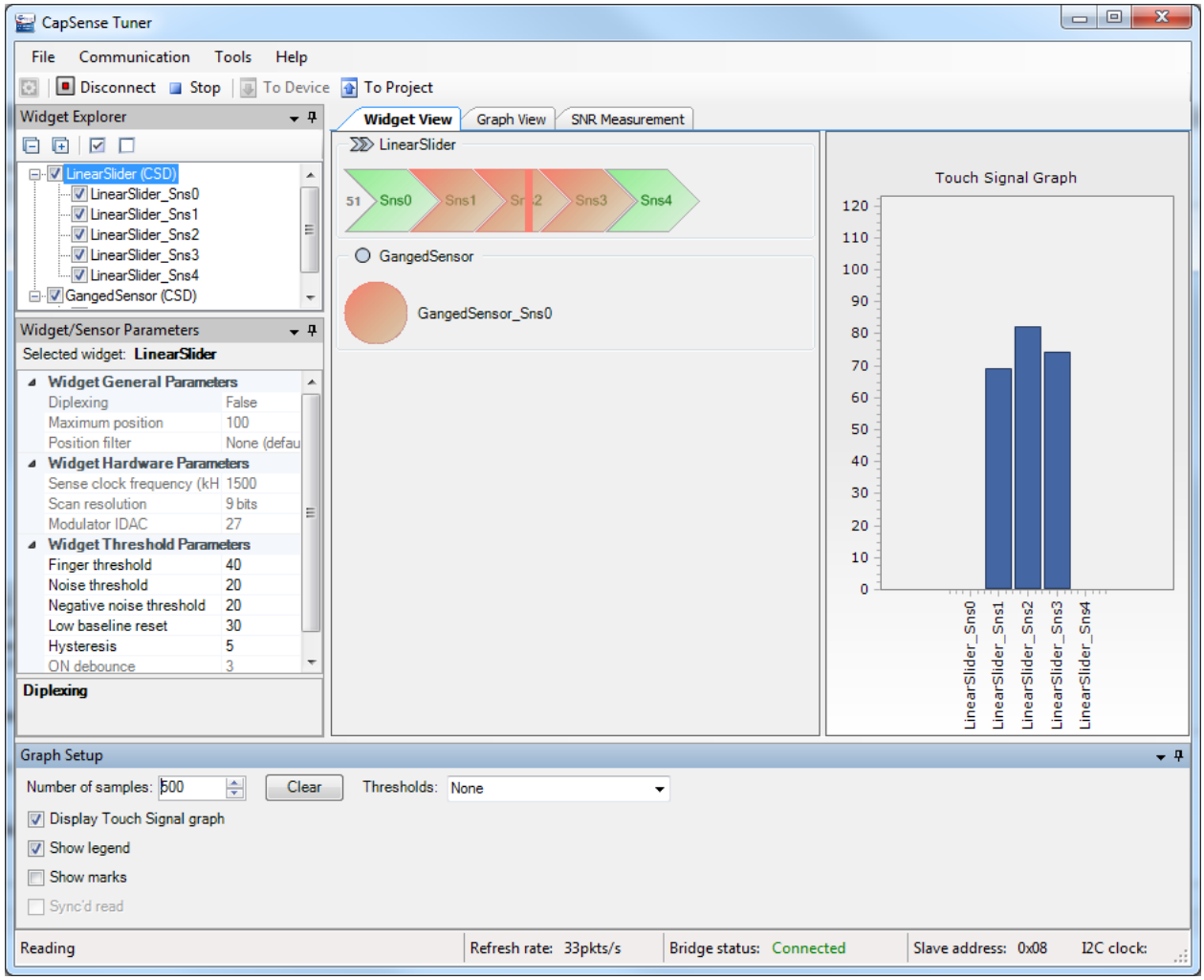
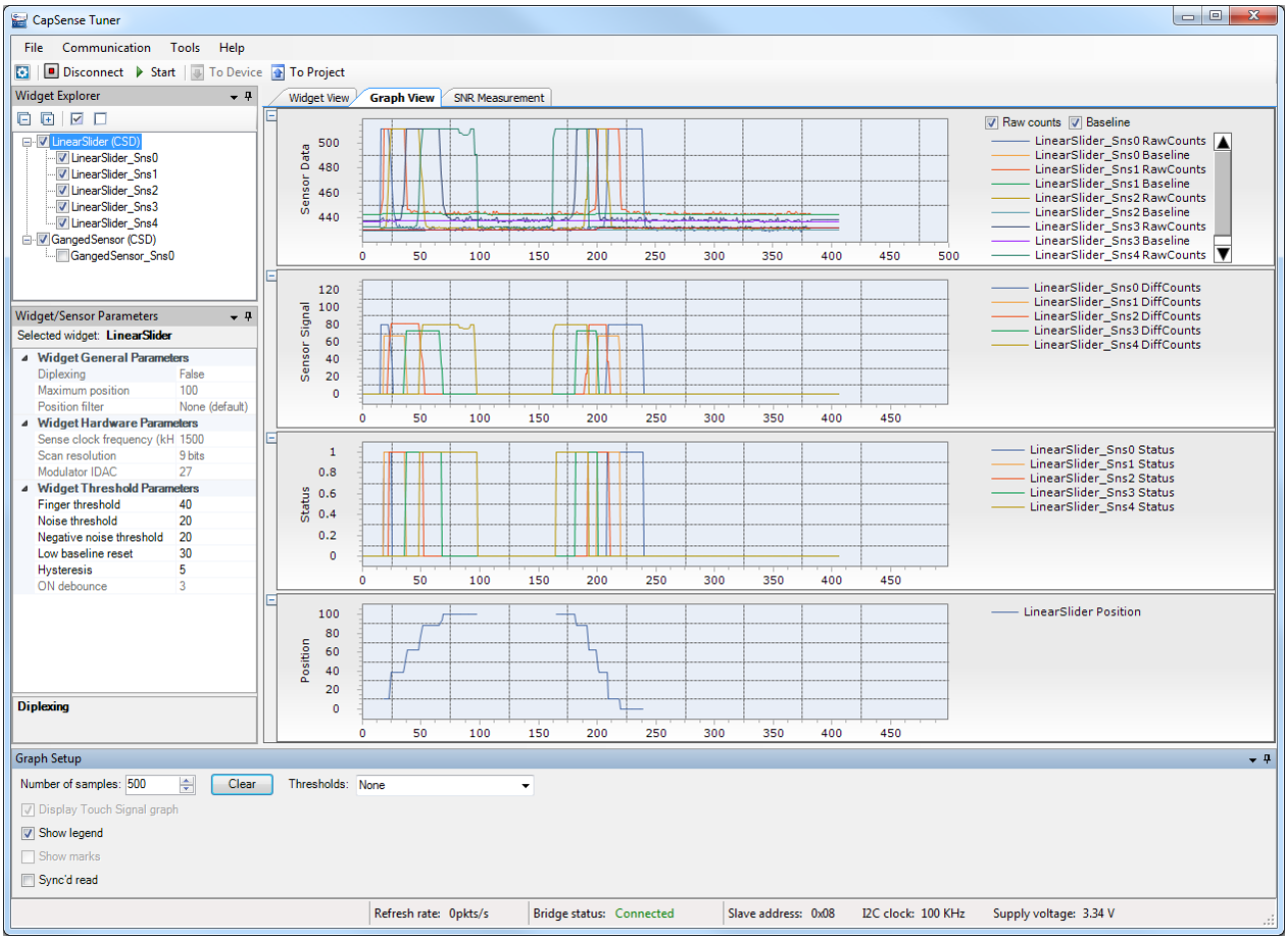


Figure 5. Tuner Graph View



12. If you change any of the parameters, and the `TUNER_UPDATE_ENABLE` macro in *main.c* is set to '1', you can click **To Project** to save the changes in your project code. After doing so, repeat steps 1 through 6.

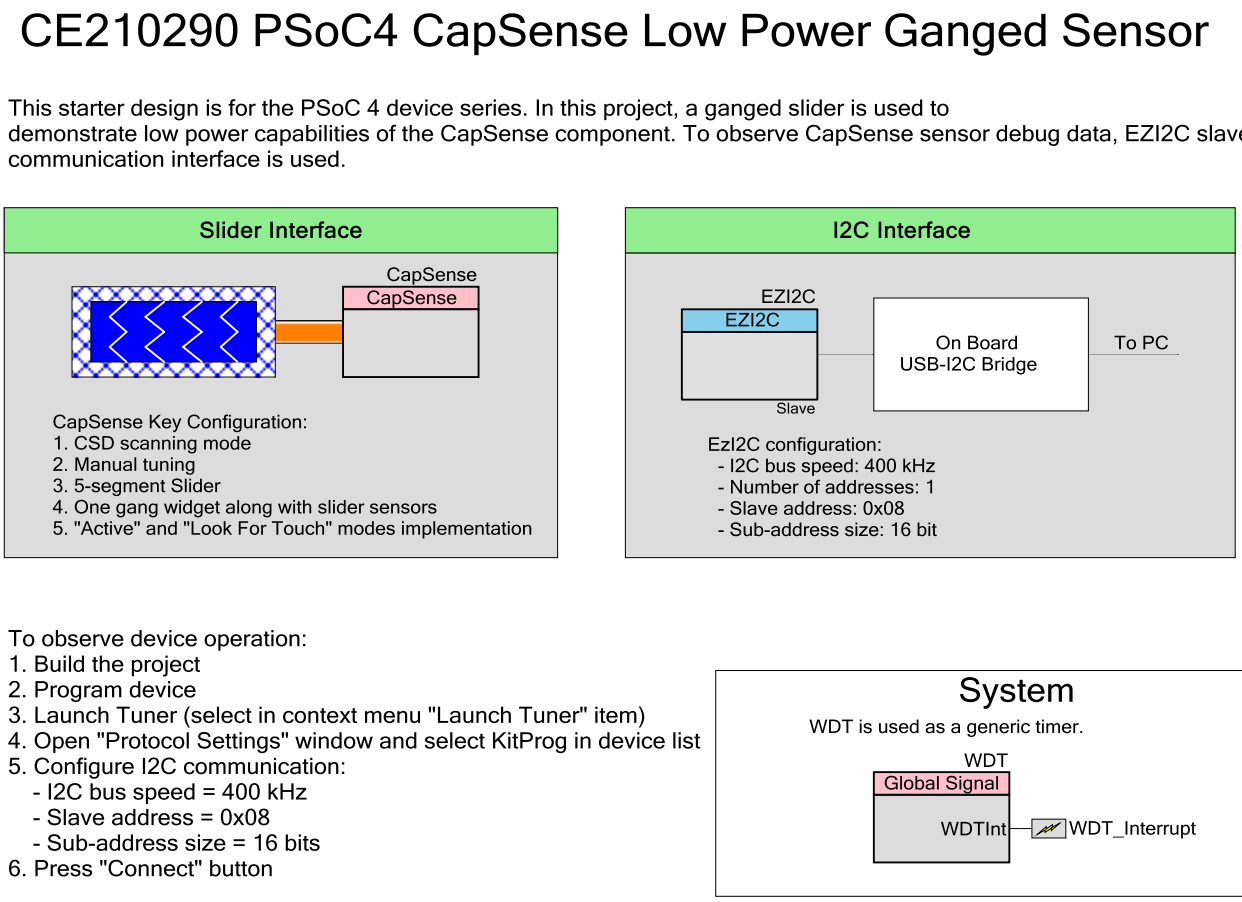
Design and Implementation

Figure 6 shows the PSoC Creator schematics of this code example. This code example uses the CapSense, EZI2C Slave, and Global Signal Reference Components.

The CapSense Component is configured with a five-segment linear slider widget and one ganged sensor. The ganged sensor is a combination of five slider segments. The project uses the CapSense Sigma Delta (CSD) manual tuning mode with current output digital to analog converter (IDAC) auto-calibration enabled.

The EZI2C Slave is used to monitor the sensor data and slider touch position information on a PC using the CapSense tuner available in the PSoC Creator integrated design environment (IDE).

Figure 6. PSoC Creator Project Schematic

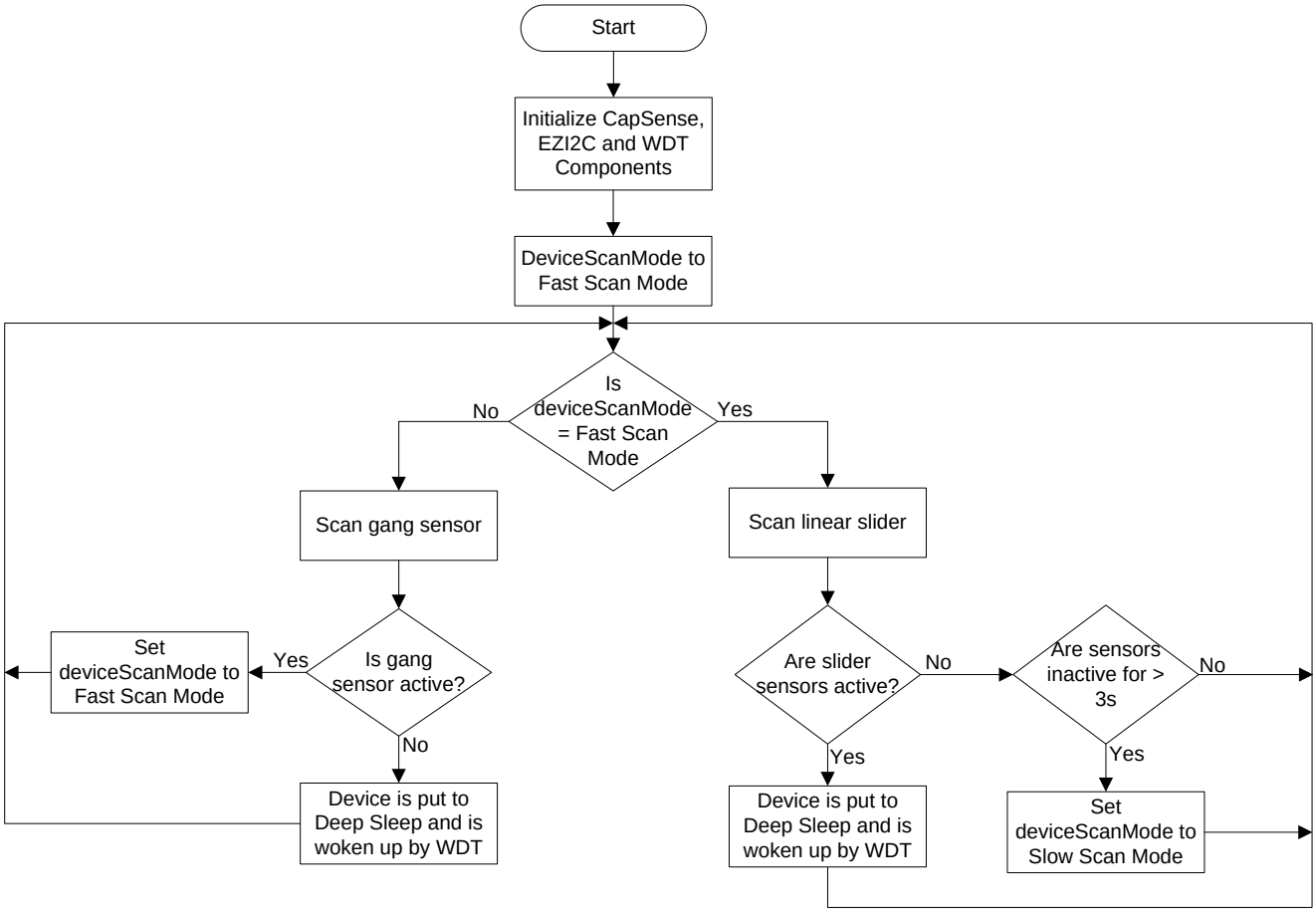


The firmware flowchart for the code example is shown in Figure 7 on page 6. To reduce the power consumed by the PSoC device and provide an optimum touch response, this code example implements two modes: Fast Scan mode and Slow Scan mode. When you are interacting with the slider, the PSoC 4 device is in the Fast Scan mode. When you are not interacting with the slider for a specific duration, the Slow Scan mode is used.

In the Fast Scan mode, all slider segments are scanned individually at a refresh rate of 33 Hz (or scan interval of 30 ms) and the touch position is calculated. The PSoC 4 device is put into Deep Sleep after the CapSense data is processed. The watchdog timer is used to periodically wake up the device from the Deep Sleep mode. This mode provides an optimum touch response, but consumes a higher average current (<120 µA) when compared to the Slow Scan mode.

In the Slow Scan mode, all slider segments are ganged and scanned as a single sensor at a refresh rate of 5 Hz (or scan interval of 200 ms). The PSoC 4 device is put into the Deep Sleep mode after the CapSense data is processed. The Slow Scan mode consumes a lower average current of 6 µA per button, but with a slower touch response. After touch is detected in the Slow Scan mode, the device switches to the Fast Scan mode to provide the optimum touch response at the expense of a higher power consumption.

Figure 7. Project Firmware Flowchart



Components and Settings

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each Component.

Table 1. List of PSoC Creator Components

Component	Instance Name	Component Version	Hardware Resources
CapSense	CapSense	V6.0	CSD and two GPIO pins
EZI2C Save (SCB mode)	EZI2C	v4.0	SCB and two GPIO pins
Interrupt	WDT_Interrupt	v1.70	Interrupt
Global Signal Reference	WDT	v2.10	Watchdog Timer

Parameter Settings

Figure 8 and Figure 9 show the changed settings for the CapSense Component.

Figure 8. CapSense Component Basic Tab Settings

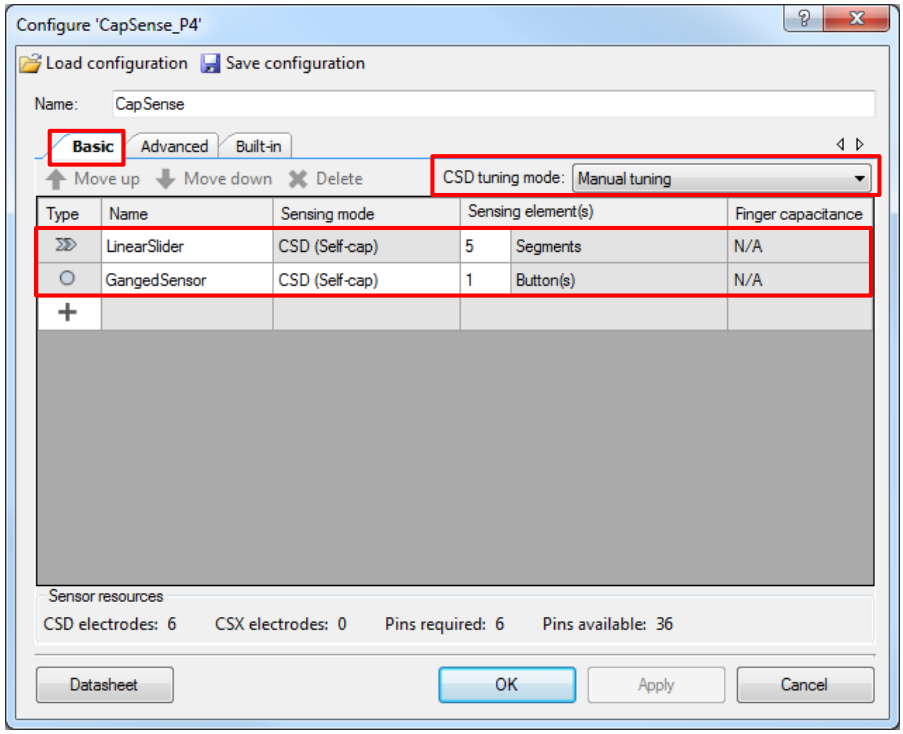


Figure 9. CapSense Component Advanced Tab – Widget Details Settings

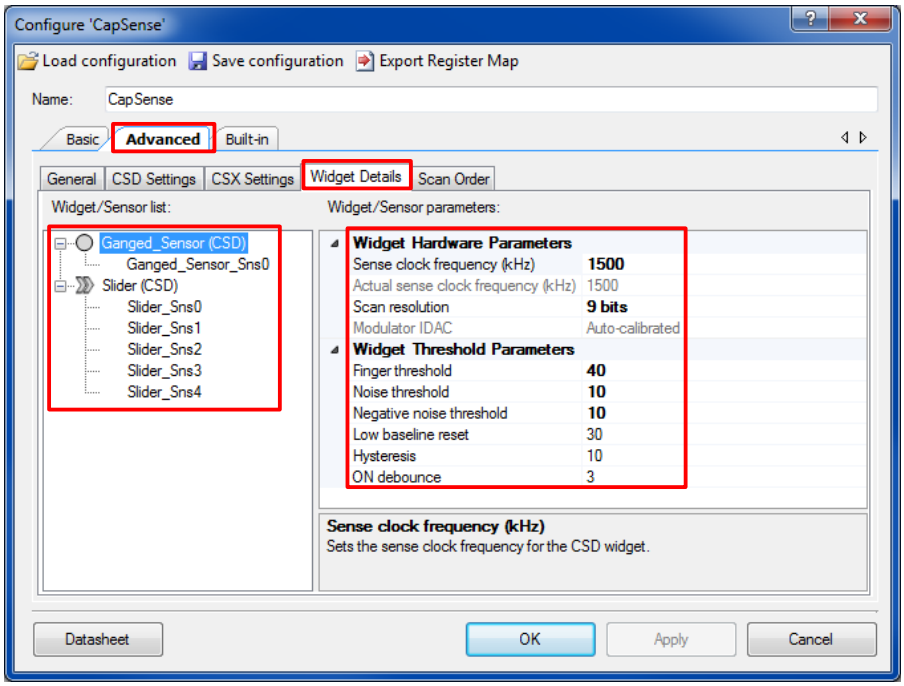
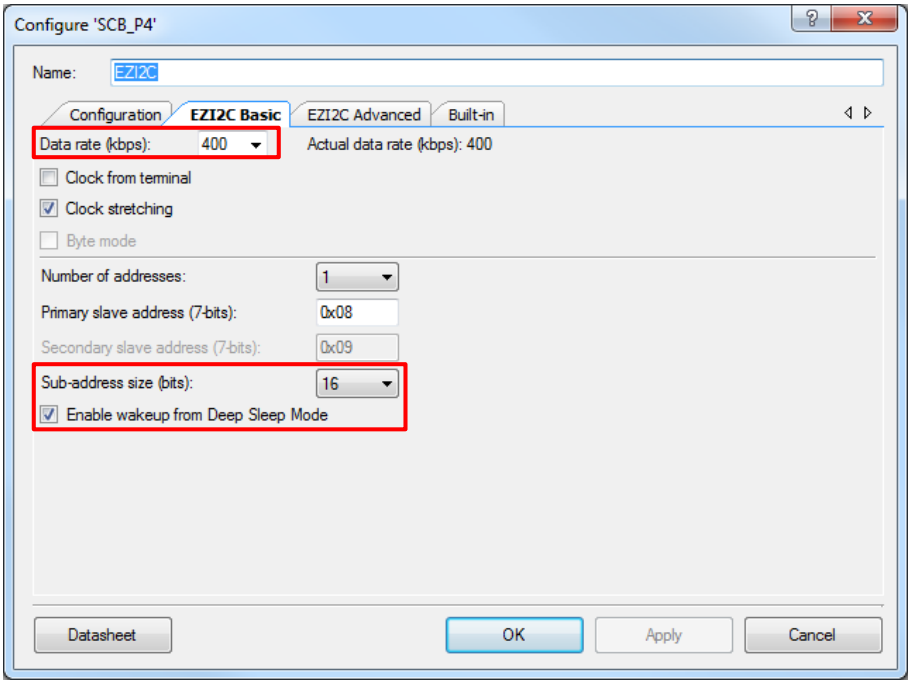


Figure 10 shows the changed settings for the EZI2C Component.

Figure 10. EZI2C Component Configuration



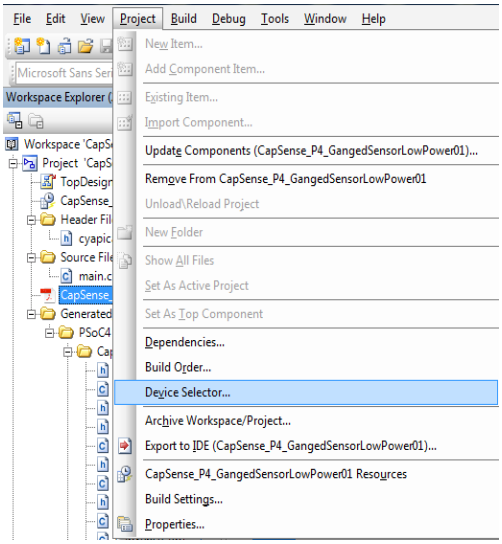
Reusing this Example

The code example is designed for the PSoC 4200 family and the associated CY8CKIT-042 kit. The design is easily portable to other PSoC 4 devices and kits, typically by just changing the device and pin assignments.

To switch from the CY8CKIT-042 to other PSoC 4 pioneer kits, follow the steps below:

1. Select the appropriate device with the Device Selector called from the project's context menu (Figure 11).

Figure 11. Select Device Selector from Project's Context Menu



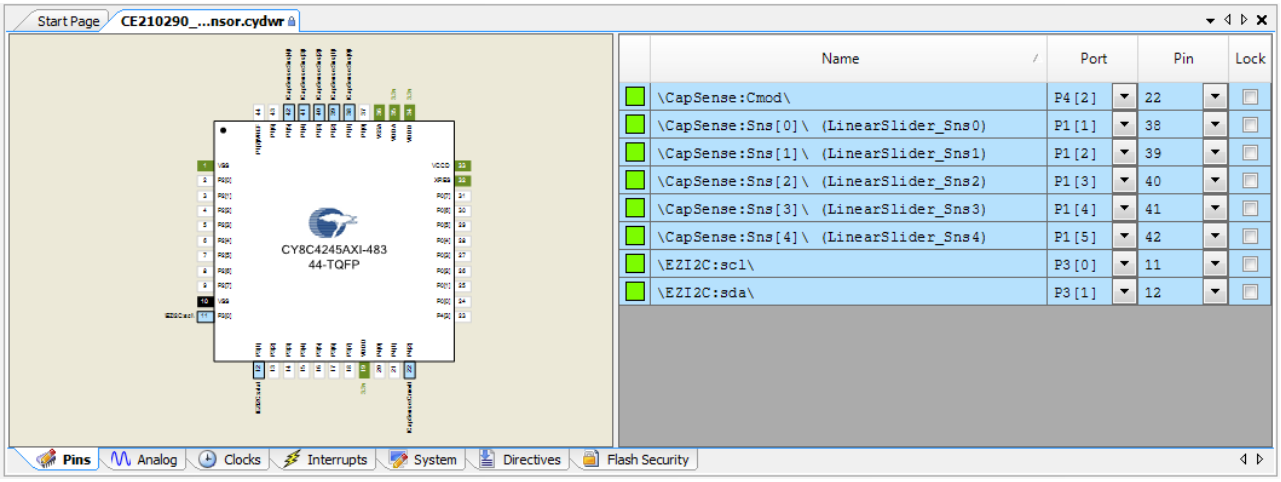
2. Table 2 lists the pin assignment for both the CY8CKIT-042 and CY8CKIT-042 BLE kits. When you select the device, the control file assigns the device pins automatically in the design wide resource file according to the device selected.

Table 2. Pin Assignment for the PSoC4 CapSense Low Power Ganged Sensor Project

Pin Name	Development Kit	
	CY8CKIT-042 (CY8C4245AXI-483)/ (CY8C4245AXQ-483)	CY8CKIT-042-BLE (CY8C4248LQI-BL583)/ (CY8C4248LQQ-BL583)
\Capsense: Cmod\	P4[2]	P4[0]
\CapSense: Sns[0]\ (LinearSlider_Sns0)	P1[1]	P2[1]
\CapSense: Sns[1]\ (LinearSlider_Sns1)	P1[2]	P2[2]
\CapSense: Sns[2]\ (LinearSlider_Sns2)	P1[3]	P2[3]
\CapSense: Sns[3]\ (LinearSlider_Sns3)	P1[4]	P2[4]
\CapSense: Sns[4]\ (LinearSlider_Sns4)	P1[5]	P2[5]
\EZI2C: scl\	P3[0]	P3[5]
\EZI2C: sda\	P3[1]	P3[4]

If the assigned Component's pins are not as shown in Table 2 or you want to overwrite the existing pin assignment, double-click the project's design wide resources (DWR) file in the Workspace Explorer window and assign the pins, as Figure 12 shows. See the device datasheet to use other pins.

Figure 12. Change Pin Assignments in Project's Design Wide Resource File



3. Select 3.3 V as the supply voltage by connecting the appropriate jumper. For CY8CKIT-042, connect the jumper on J9. For other kits, see the kit user guide.

4. Build the project and ensure that there are no errors or warnings.

Related Documents

Table 3 lists relevant application notes, code examples, device datasheets, and PSoC Creator Component datasheets.

Table 3. Related Documents

Application Notes		
AN79953	Getting Started with PSoC 4	Describes the PSoC 4, and how to build a first PSoC Creator project.
AN85951	PSoC® 4 and PSoC® 6 CapSense® Design Guide	Describes how to design capacitive touch sensing applications with the PSoC 4 and PSoC BLE families of devices.
AN86233	PSoC® 4 Low-Power Modes and Power Reduction Techniques	Describes how to use the PSoC 4 low-power modes and features to operate at very low power levels while retaining essential functionality.
AN92239	Proximity Sensing with CapSense®	This application note shows how to implement a robust, low-power, low-cost, CapSense-based proximity sensing system.
AN90114	PSoC® 4000 Family Low-Power System Design Techniques	This application note introduces the low-power modes offered by the PSoC® 4000 family and teaches the methods to design low-power systems.
Code Examples		
CE210291	PSoC® 4 CapSense® One Button	
PSoC Creator Component Datasheets		
CapSense	Supports capacitive touch sensing	
EZI2C Slave	Supports I2C slave operation	
Global Signal Reference	Supports Watchdog timer	
Device Documentation		
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals	
Development Kit (DVK) Documentation		
CY8CKIT-042 PSoC® 4 Pioneer Kit		
CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit		

Document History

Document Title: CE210290 – PSoC 4 CapSense Low-Power Ganged Sensor

Document Number: 002-10290

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5142645	AKSM	02/19/2016	New code example.
*A	5442715	VMED	09/20/2016	Updated components to the latest versions available in PSoC Creator 4.0.
*B	5638792	VMED	03/02/2017	Updated components to the latest versions available in PSoC Creator 4.1. Minor edits throughout the document. Updated template.
*C	5740212	AESATP12	05/26/2017	Updated logo and copyright.
*D	6305251	SYAO	9/21/2018	Updated Component to the latest versions available in PSoC Creator 4.2. Edits and revisions throughout the document. Reformatted document to match latest Code Example Template. Updated pin assignment section to accurate device names.
*E	6482031	JOBI	2/11/2019	Updated Table 1. Updated ‘Related Documents’ hyperlinks.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2016-2019. 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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. 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.