Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE210709 – CapSense Linear Slider and Buttons

Objective

This code example demonstrates the implementation of CapSense® linear slider and buttons and LED control. It sends information on CapSense linear slider and buttons over I²C to the CapSense Tuner or EZ-BLE™ PRoC™ Module on the CY8CKIT-145-40XX PSoC® 4000S Prototyping Kit. It also demonstrates a simple breathing LED using the SmartIO and TCPWM Components.

Overview

This code example demonstrates the operation of a CapSense linear slider with five segments and three CapSense buttons. Data from the CapSense linear slider and buttons is sent to the CapSense Tuner or EZ-BLE PRoC Module using I²C communication. The CapSense linear slider touch position and button status are used to turn ON/OFF corresponding LEDs on the kit. It also demonstrates the connectivity between the EZ-BLE PRoC Module (acting as a Peripheral and GATT server device) and mobile device running the CySmart™ mobile application (acting as a Central and GATT client device). This code example also demonstrates a simple breathing effect of LED with breath-in and breath-out rate of 1 Hz using the SmartIO and TCPWM (configured as PWM) Components.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 4000S parts

Related Hardware: CY8CKIT-145-40XX PSoC 4000S Prototyping Kit

Hardware Setup

This example uses the CY8CKIT-145-40XX PSoC 4000S Prototyping Kit's default configuration. See the kit guide to make sure the kit is configured correctly.

Software Setup

None.

Operation

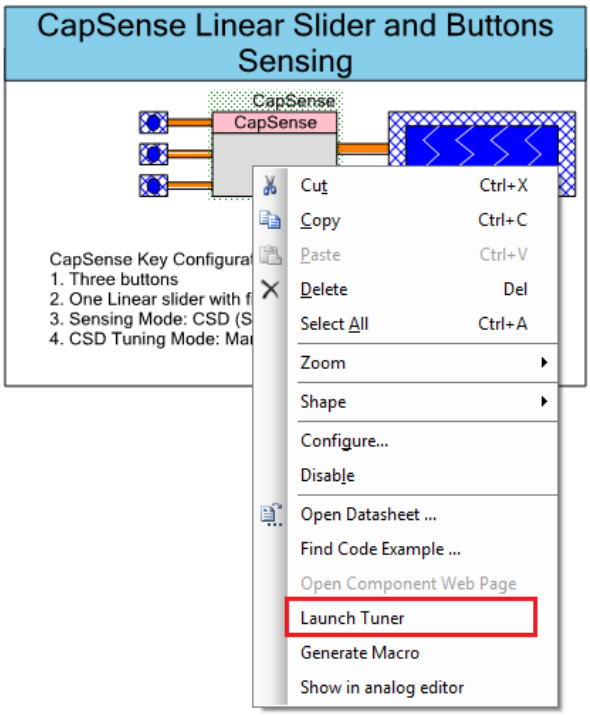
Follow these steps to verify the code example:

1. Select the *CE210709_CapSense_Linear_Slider_and_Buttons.cywrk* file in the PSoC Creator Start page under **Examples and Kits** > **Kits** > **CY8CKIT-145-40XX**. Select a location to save the code example.
2. Build the project (select the PSoC Creator menu item **Build** > **Build CE210709_CapSense_Linear_Slider_and_Buttons**).
3. Connect the PSoC 4000S Prototyping Kit to your computer's USB port.
4. Push the SWD Select switch (SW4) to the PSoC 4000S position to program the PSoC 4000S device.
5. Program the PSoC 4000S device (select **Debug** > **Program**). Blue LED (LED1) glows with breathing effect.
6. Touch the CapSense buttons and observe that the corresponding LEDs turn ON.
7. Slide your finger over the CapSense linear slider and observe that the LEDs turn ON depending on the touch position.

Verifying Code Example Using the CapSense Tuner

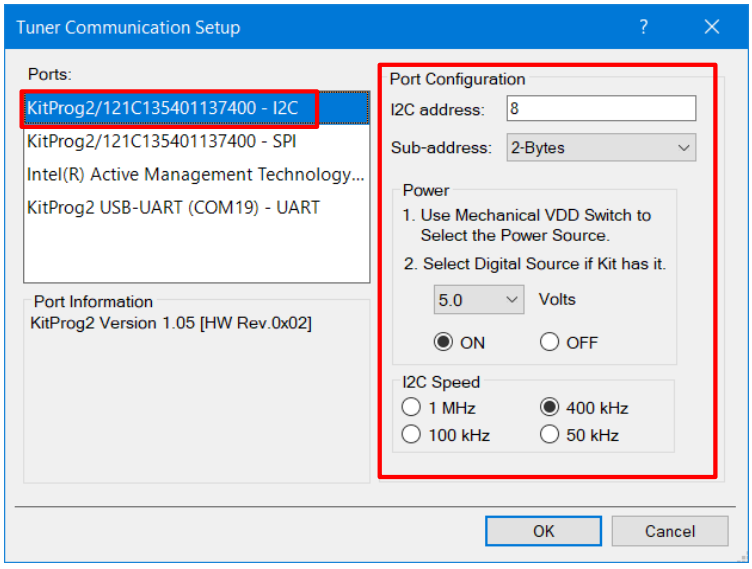
1. Launch the CapSense Tuner. Right-click the CapSense Component and select **Launch Tuner** from the menu as shown in Figure 1.

Figure 1. Launch Tuner GUI



2. Navigate to **Tools** > **Tuner Communication Setup** in the CapSense Tuner GUI menu to set up the I²C communication (Figure 2).
   - Select **KitProg2/<serial_number> - I2C**
   - Choose the I²C address, sub-address size, and speed as used in EZI2C configuration (Figure 2) and click **OK**.

Figure 2. Setting up I²C Communication



3. Click **Connect** and then **Start** as shown in Figure 3 and Figure 4.

Figure 3. CapSense Tuner Connect Button

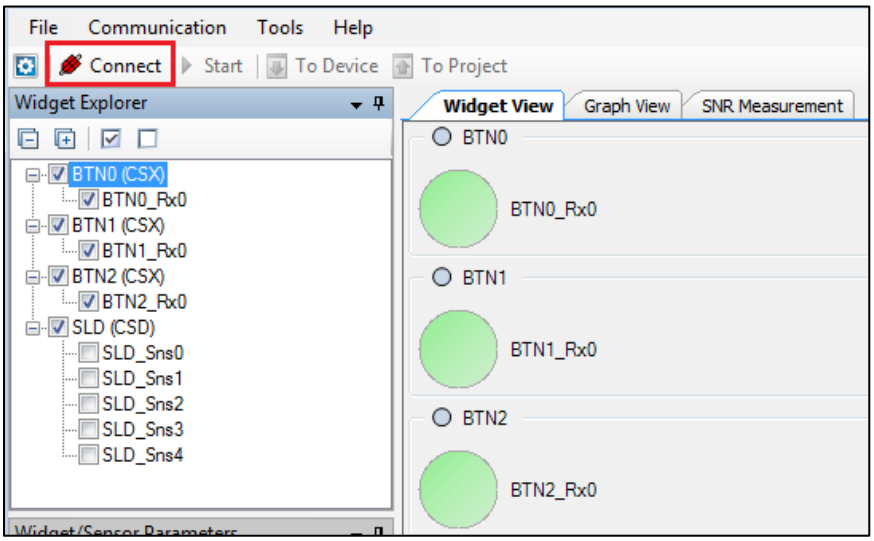
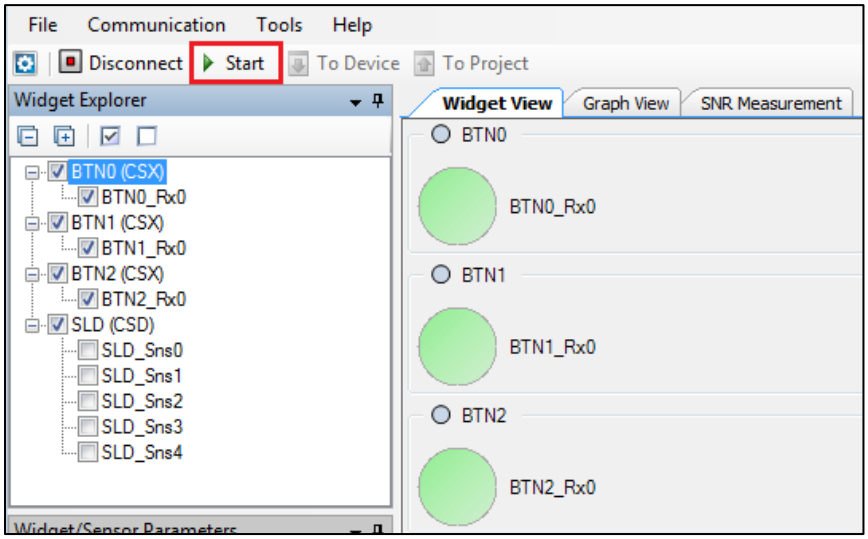


Figure 4. CapSense Tuner Start Button



After establishing the I^2C communication with the CapSense Tuner, you can observe the touch position on the CapSense linear slider (centroid) and CapSense button state (ON/OFF).

Figure 5 and Figure 6 show the scanning results for all the sensing elements.

Figure 5. CapSense Tuner: Widget View of CapSense Buttons and Linear Sliders

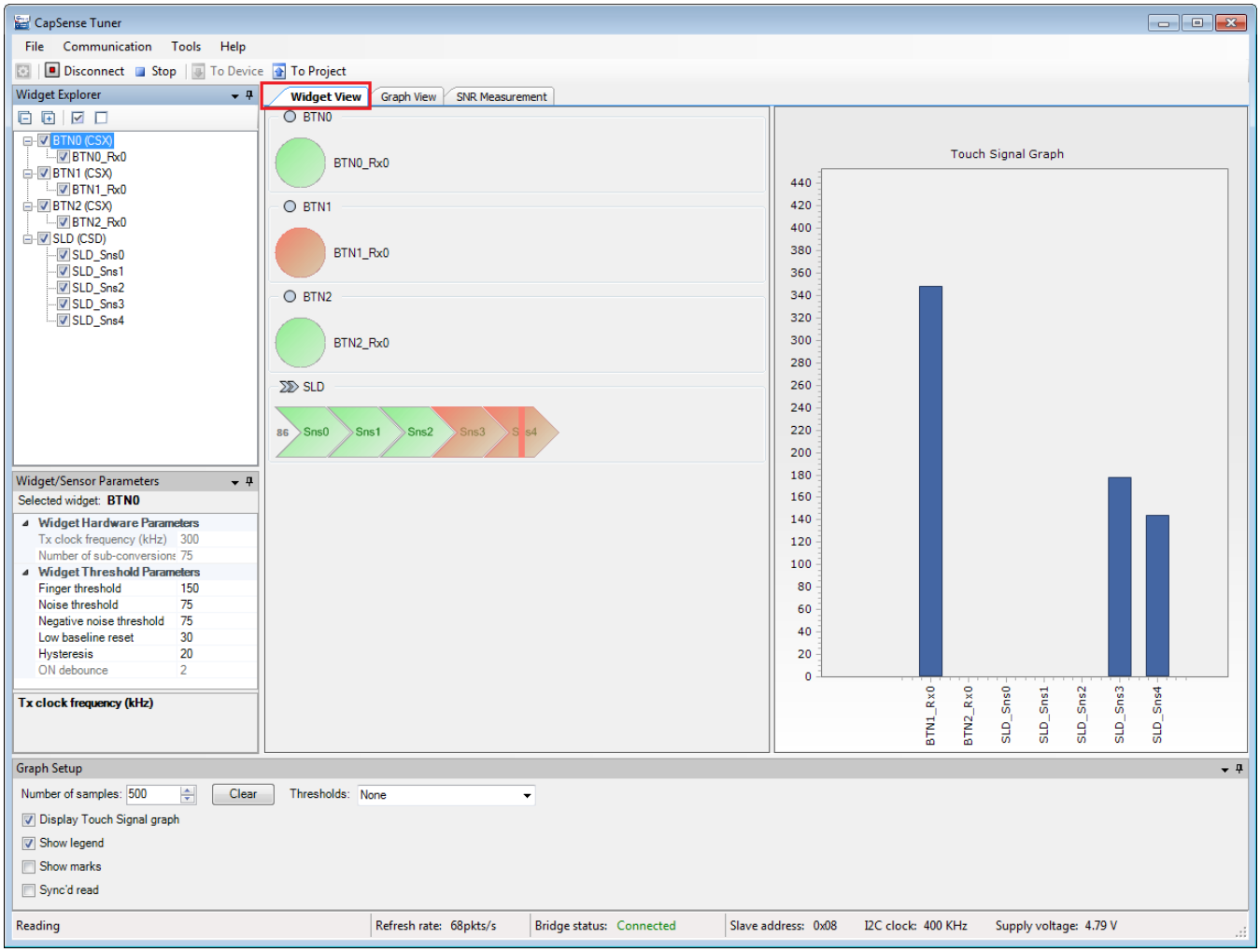
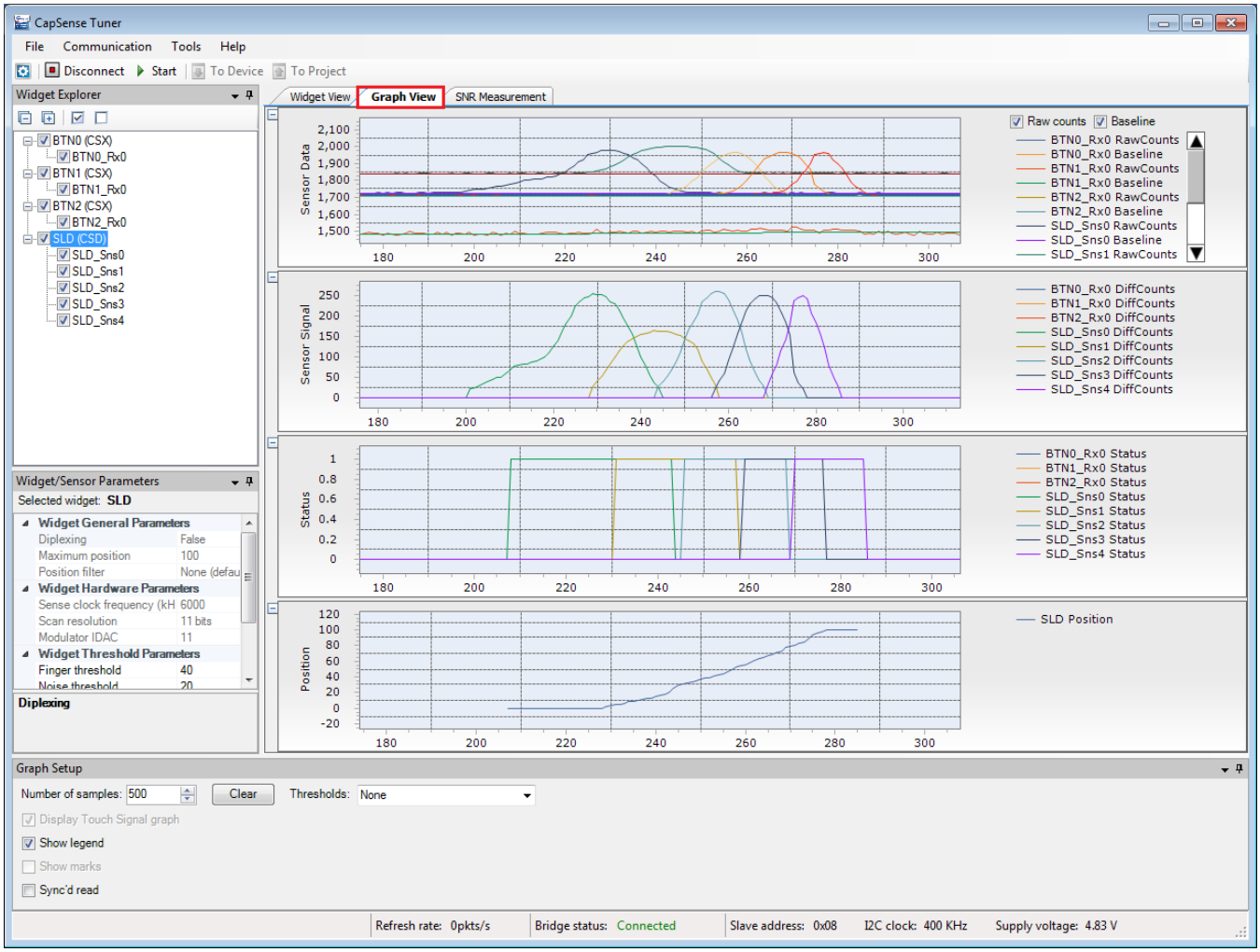


Figure 6. CapSense Tuner: Graph View of CapSense Linear Slider



Verifying Code Example Using the CySmart Mobile Application

To verify the CapSense linear slider operation using the CySmart mobile application, make sure that your mobile phone has the CySmart application installed. See the CySmart Mobile Application webpage and then follow the steps listed below:

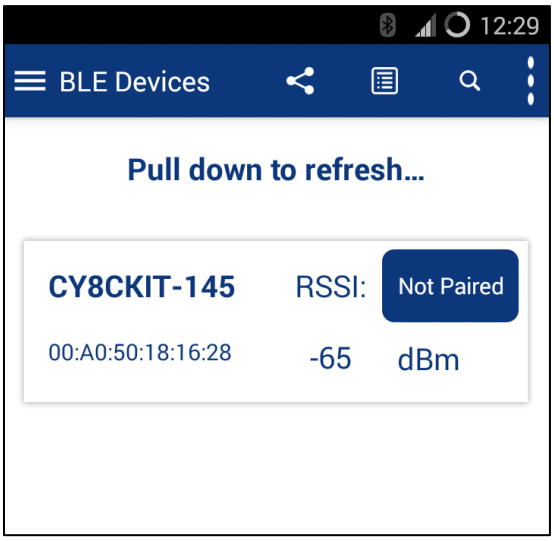
1. Open the *main.c* file corresponding to CE210709_CapSense_Linear_Slider_and_Buttons project and change the `ENABLE_TUNER` macro value to '0u'.

   Note: When the `ENABLE_TUNER` macro value is '0u', the PSoC 4000S device sends the data only to the Bluetooth Low Energy (BLE) CapSense Service. Change the `ENABLE_TUNER` macro value to '1u' to use the CapSense Tuner.

2. Build the project (select the PSoC Creator menu item **Build** > **Build CE210709_CapSense_Linear_Slider_and_Buttons**).
3. Program the PSoC 4000S device; select **Debug** > **Program**.
4. In the Workspace Explorer, right-click the CE210709_EZ-BLE_Peripheral project and select **Set As Active Project**.
5. Build the project (select the PSoC Creator menu item **Build** > **CE210709_EZ-BLE_Peripheral**).
6. Push the SWD Select switch (SW4) to the 'EZ-BLE' position to program the EZ-BLE PRoC Module.
7. Program the EZ-BLE PRoC Module (select **Debug** > **Program**).
8. Open the CySmart application on the mobile device. If Bluetooth is not enabled on the device, the application will prompt to enable it.

9. After Bluetooth is enabled, the CySmart mobile application will automatically search for available peripherals and list them. Select the **CY8CKIT-145** peripheral as shown in Figure 7.

Figure 7. CY8CKIT-145 Peripheral



10. When connected, the CySmart mobile application will list the profiles supported by the peripherals. Scroll and select the CapSense icon, as shown in Figure 8.

Figure 8. CapSense Service Page



11. Swipe your finger on the CapSense linear slider on the kit and see a similar response on the CapSense page in the CySmart application, as shown in Figure 9.

Figure 9. CapSense Slider



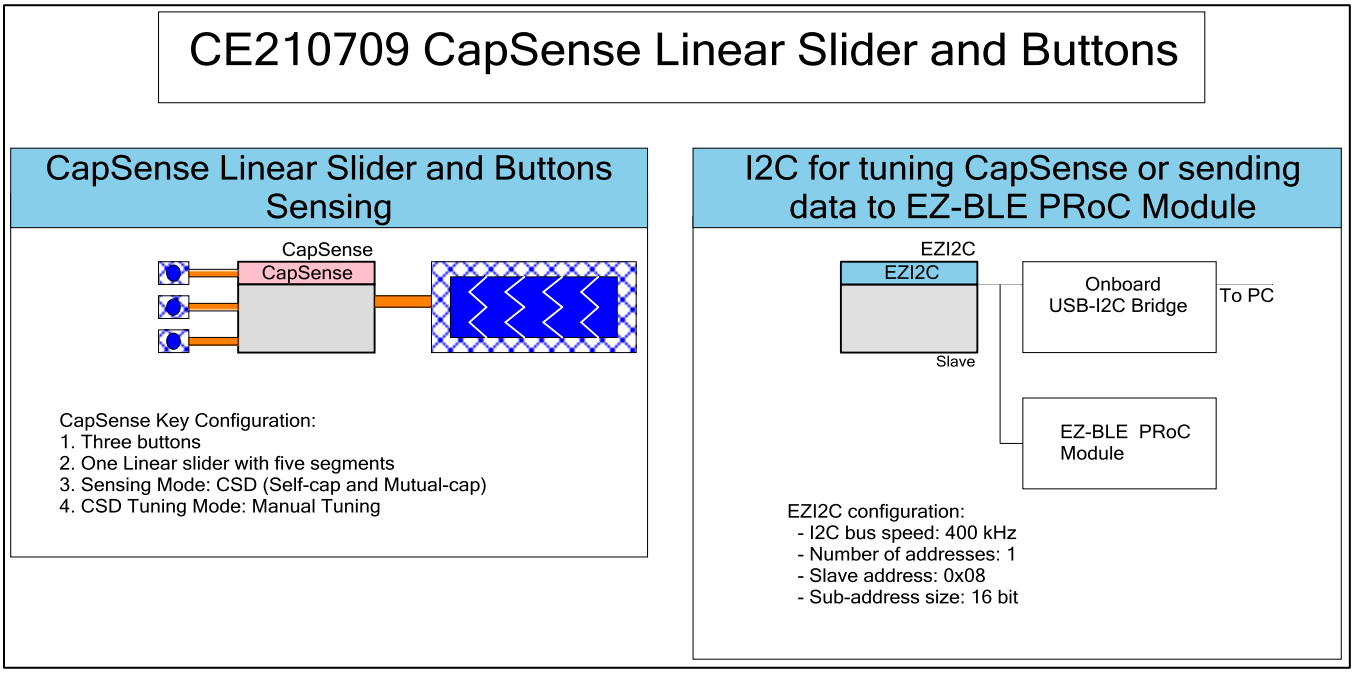
Design and Implementation

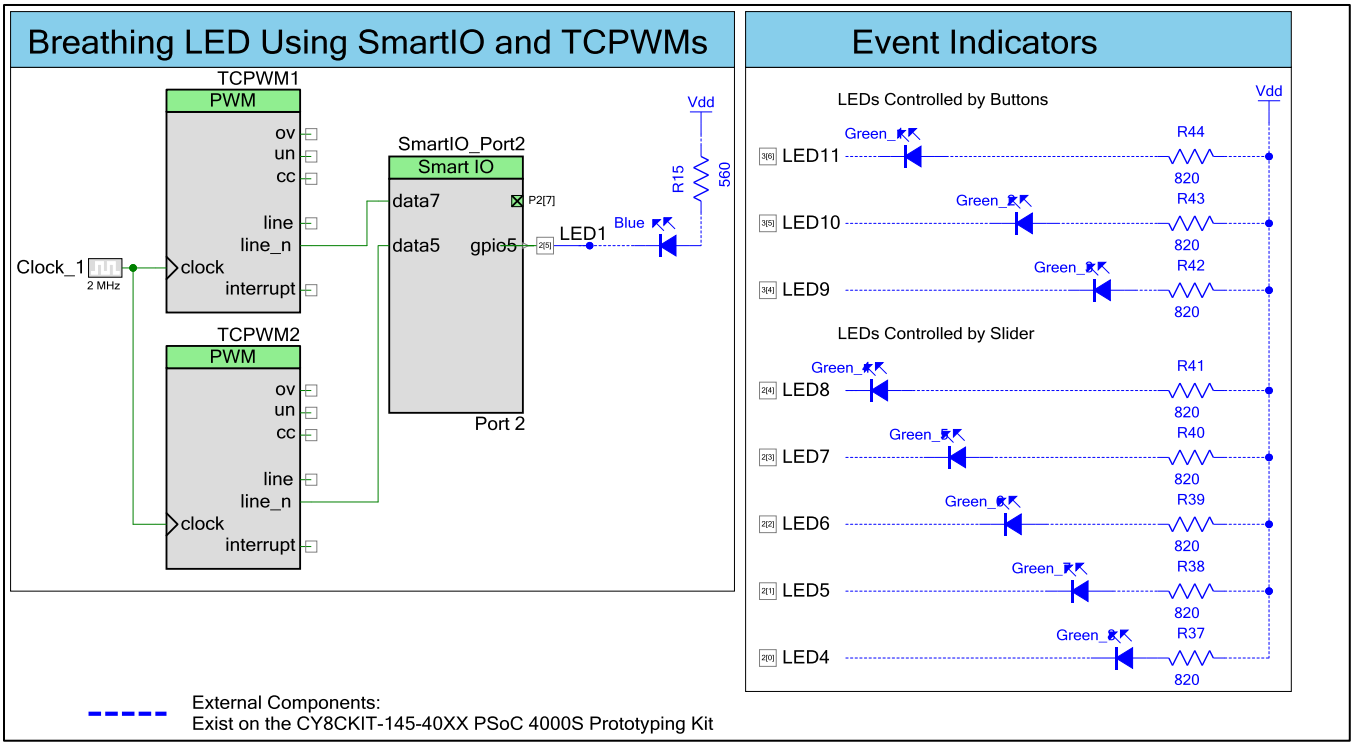
This code example has two PSoC Creator projects in a single workspace - CE210709_CapSense_Linear_Slider_and_Buttons and CE210709_EZ-BLE_Peripheral.

CE210709_CapSense_Linear_Slider_and_Buttons

Figure 10 shows the PSoC Creator schematics of the CE210709_CapSense_Linear_Slider_and_Buttons project. This project uses CapSense, EZI2C Slave, PWM, Pin, Clock, and SmartIO Components.

Figure 10. PSoC Creator Schematics for CE210709_CapSense_Linear_Slider_and_Buttons





The CapSense Component is configured with one linear slider widget with five segments and three button widgets. This project uses the CapSense Sigma-Delta (CSD) manual tuning mode with IDAC auto-calibration enabled. When the CapSense linear slider is touched, depending on the touch position, the corresponding LEDs are turned ON as shown in Table 1. In the case of CapSense buttons, if a touch is detected, corresponding LEDs are turned ON.

Table 1. LEDs State Depending on the Finger Position on CapSense Linear Slider

Finger Position (Centroid) on CapSense Linear Slider (C)[1]	LED8	LED7	LED6	LED5	LED4
No touch	OFF	OFF	OFF	OFF	OFF
0% <= C <= 20% (1st Segment of CapSense Linear Slider)	OFF	OFF	OFF	OFF	ON
20% < C <= 40% (2nd Segment of CapSense Linear Slider)	OFF	OFF	OFF	ON	ON
40% < C <= 60% (3rd Segment of CapSense Linear Slider)	OFF	OFF	ON	ON	ON
60% < C <= 80% (4th Segment of CapSense Linear Slider)	OFF	ON	ON	ON	ON
80% < C <= 100% (5th Segment of CapSense Linear Slider)	ON	ON	ON	ON	ON

[1] Touch position from segment 0 (SLD0) as a percentage of CapSense linear slider length.

The EZI2C Slave Component is used to monitor the sensor data on the PC using the CapSense Tuner available in the PSoC Creator integrated design environment (IDE). The CapSense linear slider touch position can also be viewed in the CySmart mobile application when data is sent to the EZ-BLE PRoC Module. When the CapSense Tuner is disabled in firmware, the CapSense linear slider touch position and button ON/OFF status is sent to the EZ-BLE PRoC Module on the PSoC 4000S Prototyping Kit via I^2C using EZI2C Slave Component.

In this project, an LED (LED1) is used to demonstrate the breathing effect. As shown in Figure 11, the breathing effect of LED is generated by XORing two PWM signals that have slightly different frequencies with a duty cycle of 50%. The XOR gate is implemented using the SmartIO Component. The SmartIO Component is configured in combinatorial mode with the Look-Up-Table (LUT) configured for an XOR gate implementation. The two PWMs are of 100 Hz and 101.01 Hz frequency. Routing these PWM signals through an exclusive-OR (XOR) gate yields an output signal with a gradually changing duty cycle. Driving an LED with this signal results in a “breathing” effect, where the LED gradually gets brighter and dimmer. The rate of change is proportional to the difference between the PWM output frequencies.

Figure 11. LED Breathing Effect by XORing Two PWM Signals

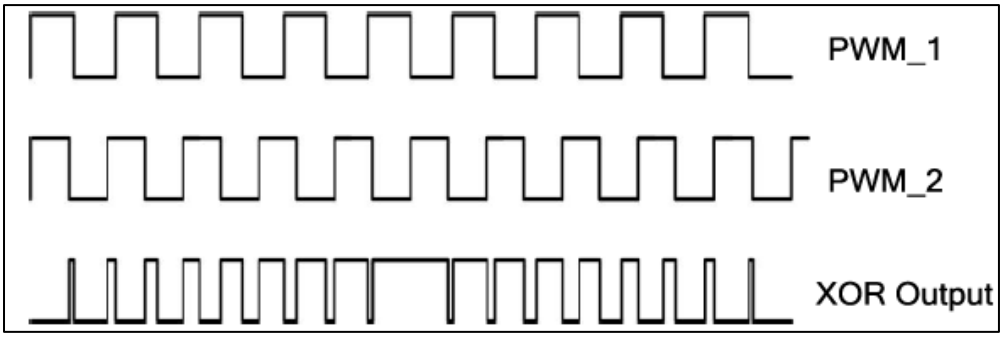
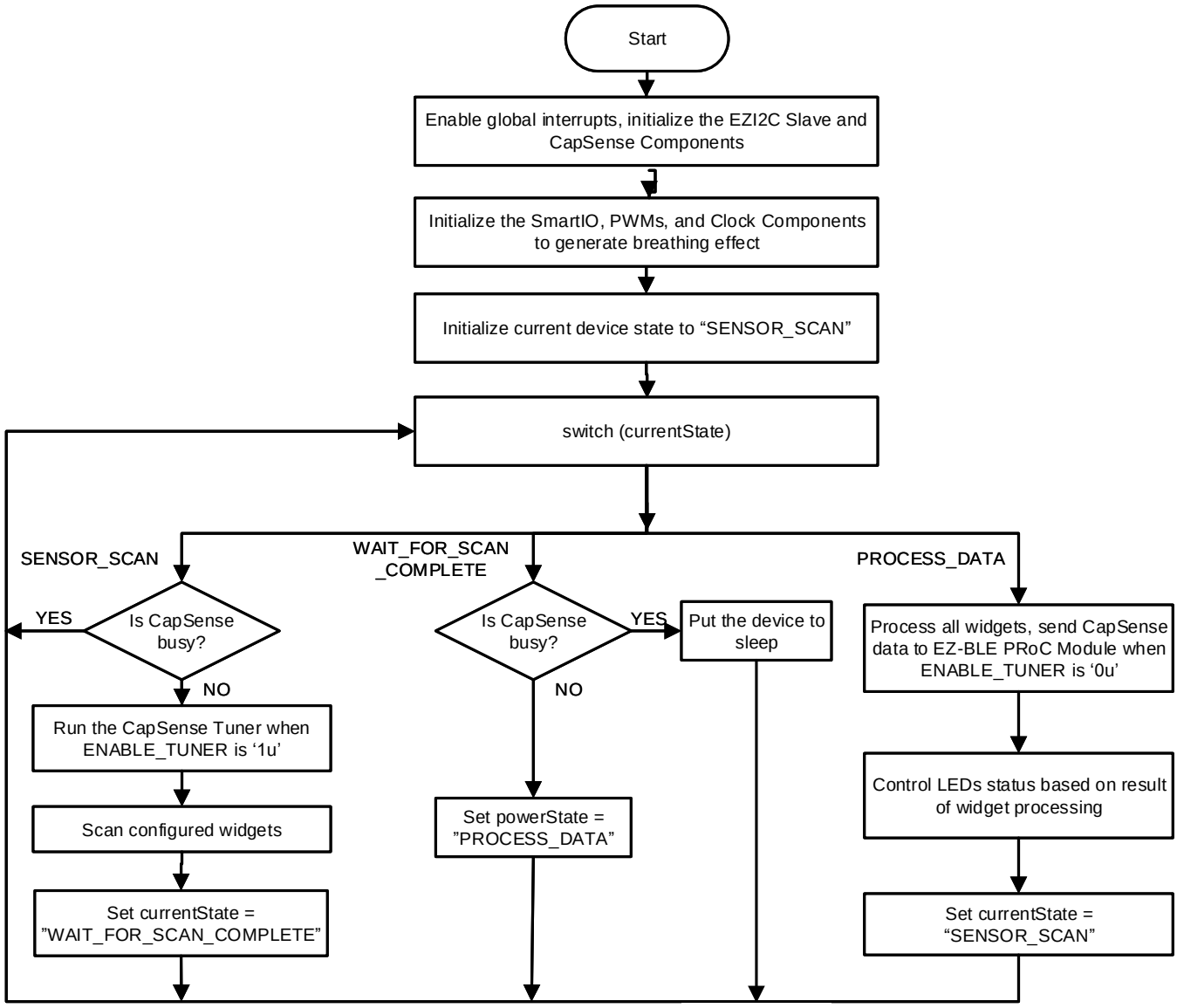


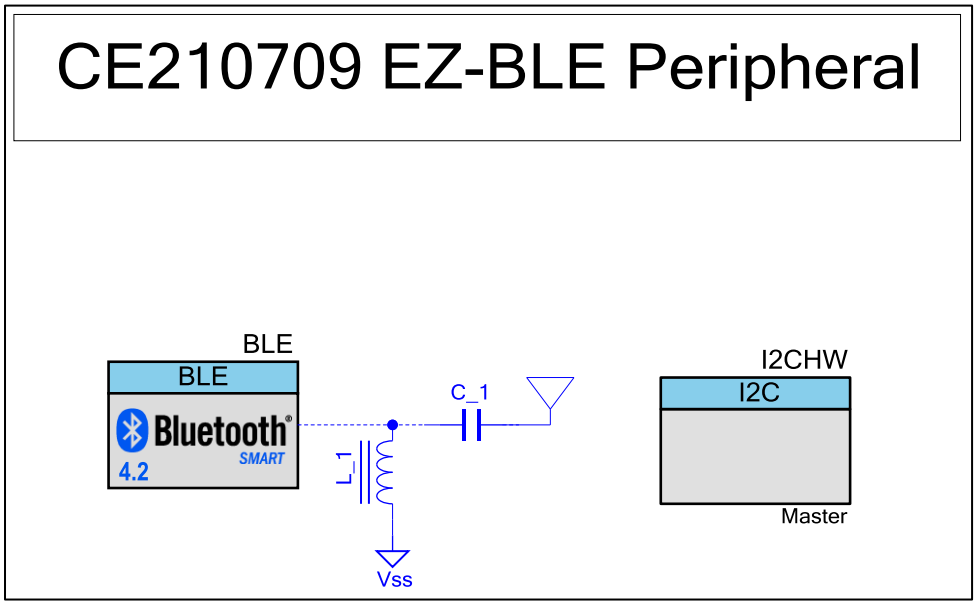
Figure 12. Firmware Flowchart



CE210709_EZ-BLE_Peripheral

Figure 13 shows the PSoC Creator schematics of CE210709_EZ-BLE_Peripheral project. This project uses BLE and I^2C Components.

Figure 13. PSoC Creator Schematics for CE210709_EZ-BLE_Peripheral



The I²C Component configured in Master mode is used to receive the CapSense buttons ON/OFF status and linear slider touch position from the PSoC 4000S device on the PSoC 4000S Prototyping Kit.

The **Custom** profile in this project consists of a single BLE Custom Service - **CapSense**. The **CapSense** Service consists of Custom Characteristics - **CapSense Slider** and **CapSense Button**. The **CapSense Slider** Characteristic is used to send one-byte data, ranging from 0 to 100, as a notification to the GATT client device. This data is the finger location read by the CapSense Component on the 5-segment linear slider present on the kit. The **CapSense Button** Characteristic is used to send the number of buttons and their ON/OFF status, as a notification to the GATT client device. For more details on the CapSense Service implementation, see the *Cypress CapSense Service* guide.

Components and Settings

Table 2 and Table 3 list 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 2. List of PSoC Creator Components in the CE210709_CapSense_Linear_Slider_and_Buttons Project

Component	Instance Name	Purpose	Non-default Settings
CapSense	CapSense	Scan 5-segment linear slider and three CapSense buttons	See Figure 14 through Figure 18
EZI2C Slave (SCB mode)	EZI2C	I²C Slave operation	See Figure 19
PWM (TCPWM mode)	TCPWM1	Generate square wave and bring out the signal to GPIO	See Figure 20 and Figure 21
	TCPWM2		
Clock	Clock_1	Drive the PWM at 2 MHz	See Figure 22
Digital Output Pin	LED1	Drive the SmartIO output to LED	External terminal: Checked
	LED4	Firmware controlled digital output pins that control status LEDs	See Figure 23
	LED5		
	LED6		
	LED7		
	LED8		
	LED9		

Component	Instance Name	Purpose	Non-default Settings
	LED10		
	LED11		
SmartIO	SmartIO_Port2	Implement XOR gate for two PWM signals	See Figure 24 and Figure 25

Table 3. List of PSoC Creator Components in CE210709_EZ-BLE_Peripheral Project

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	Establish BLE connectivity. Custom services are used for notifying the BLE Central device of CapSense slider and button data.	See Figure 26 for Profile tab settings. For details on other non-default settings, open the Component Configuration window.
I^2C (SCB mode)	I2CHW	I^2C Master operation	See Figure 27

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

Figure 14 through Figure 18 show the settings for the CapSense Component.

Figure 14. CapSense Component's Basic Tab

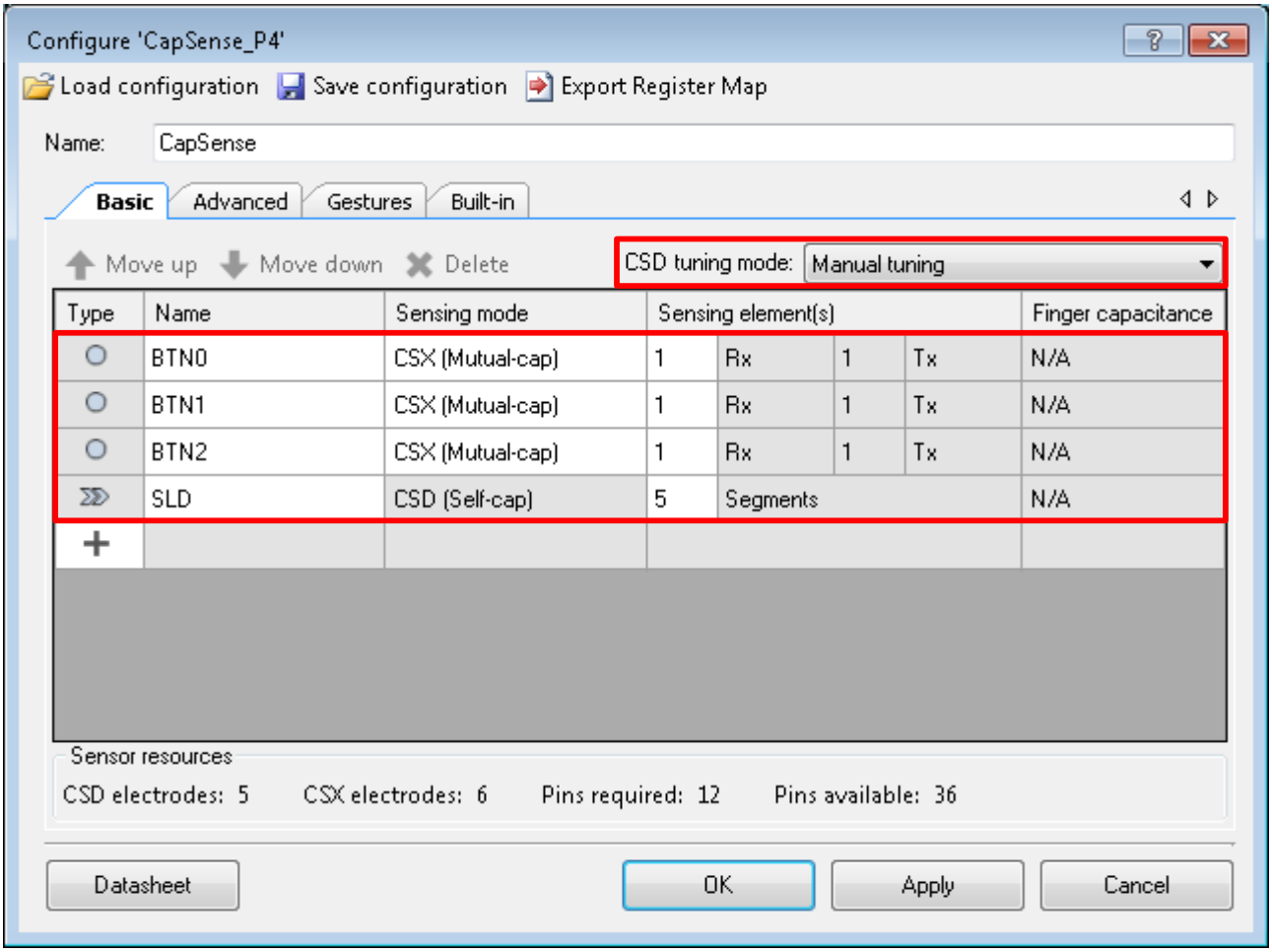


Figure 15. CapSense Component's Button (BTN0) Widget Details in Advanced Tab

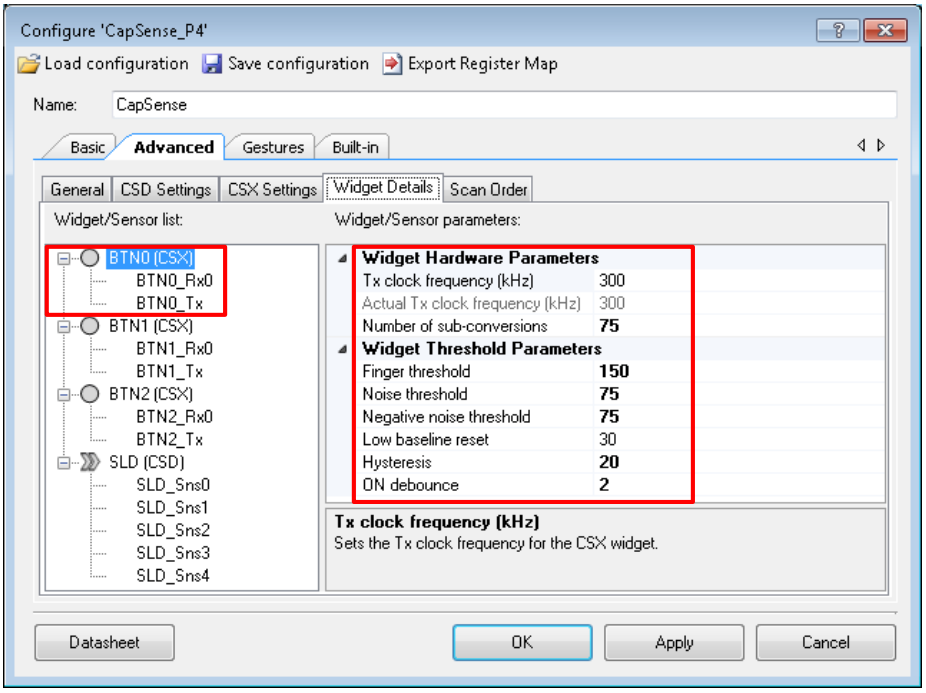


Figure 16. CapSense Component's Button (BTN1) Widget Details in Advanced Tab

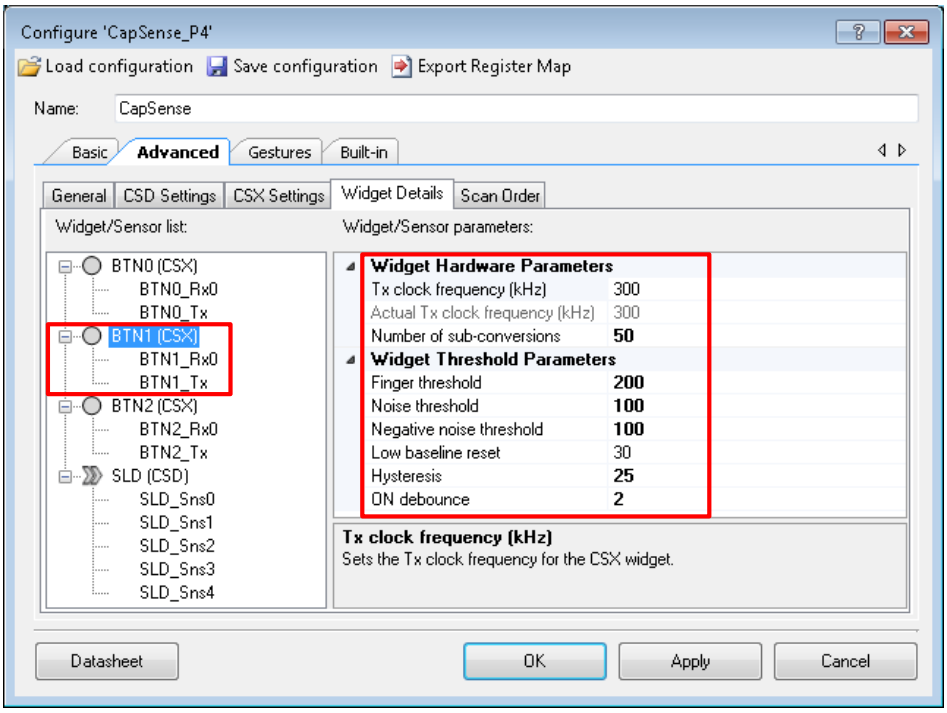


Figure 17. CapSense Component's Button (BTN2) Widget Details in Advanced Tab

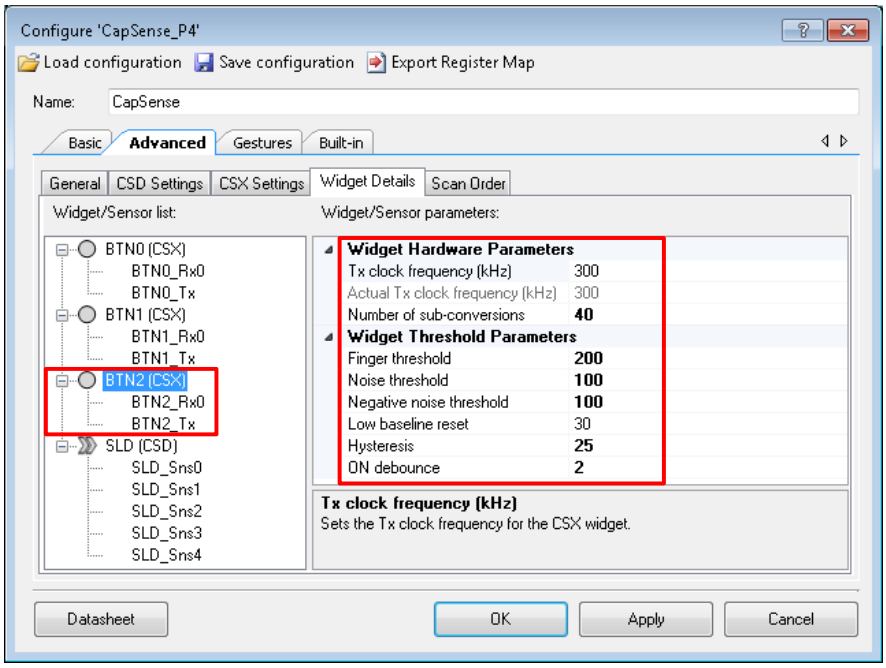


Figure 18. CapSense Component's Linear Slider Widget Details in Advanced Tab

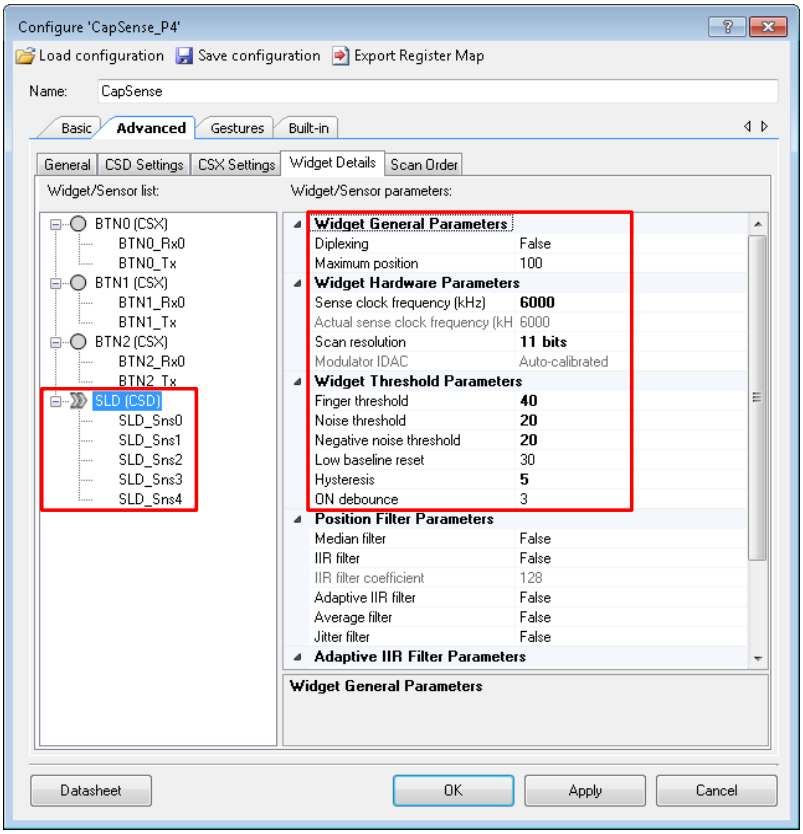


Figure 19 shows the settings for the EZI2C Component.

Figure 19. EZI2C Slave Component's Basic Tab

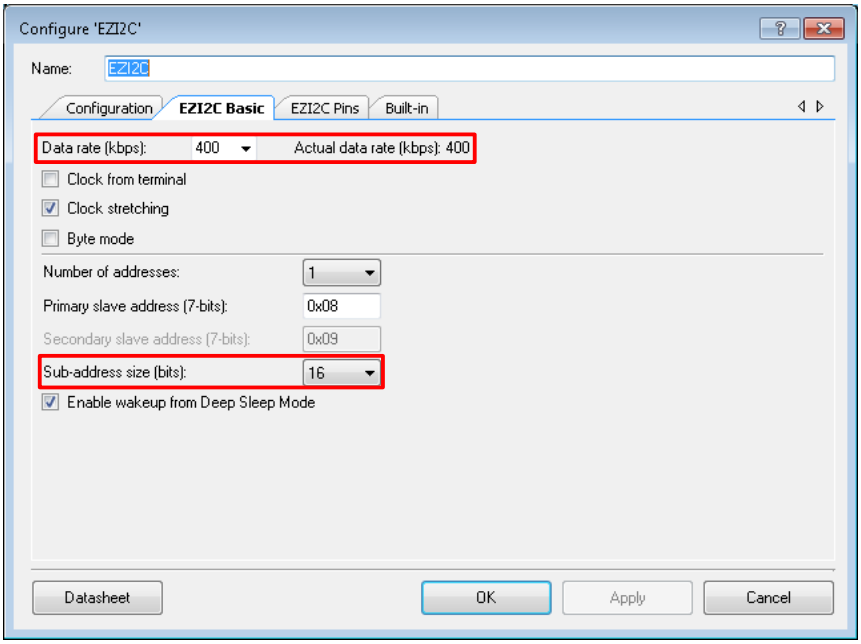


Figure 20 and Figure 21 show the settings for two TCPWM Components.

Figure 20. TCPWM1 Component's PWM Tab

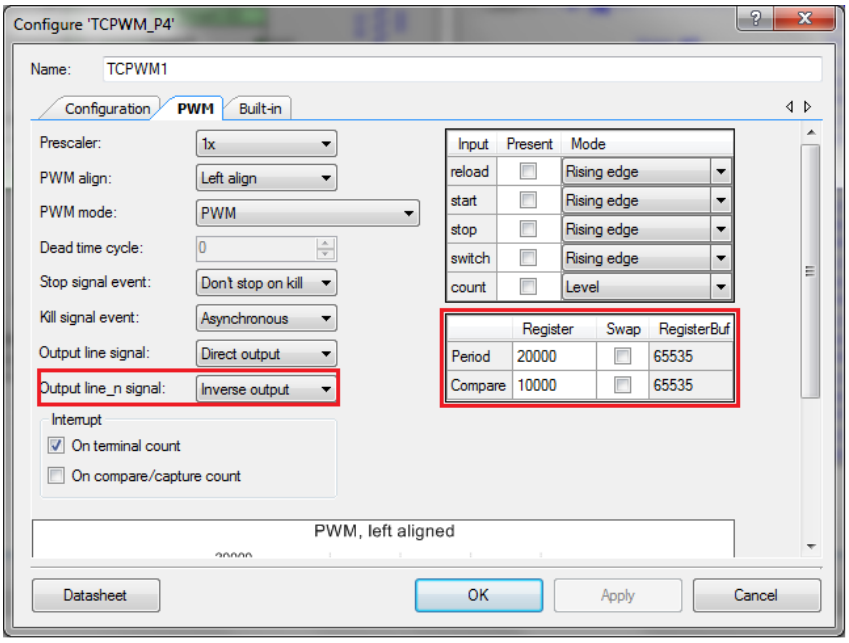


Figure 21. TCPWM2 Component's PWM Tab

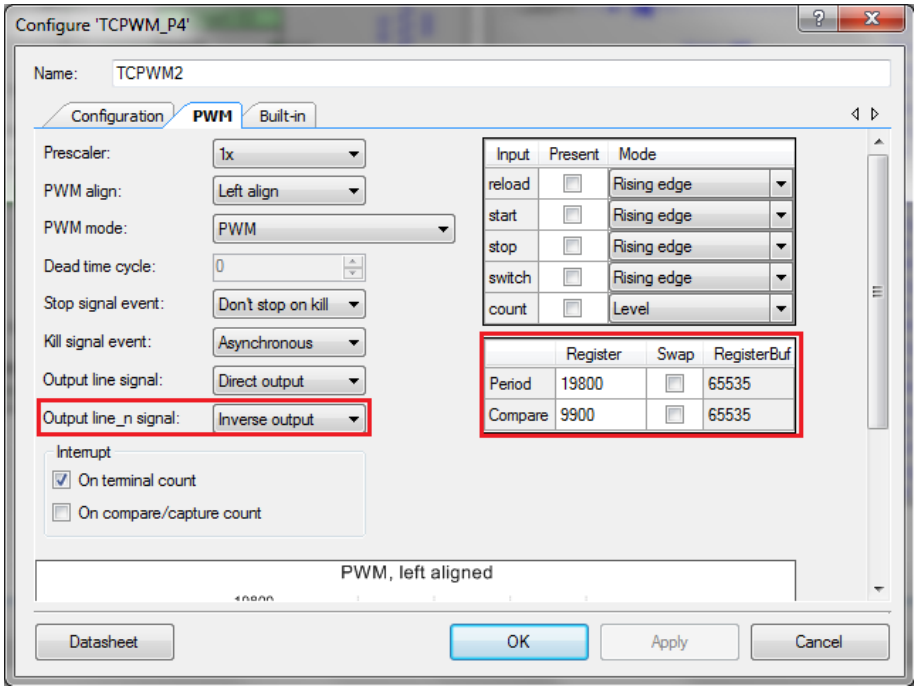


Figure 22 shows the settings for the Clock Component.

Figure 22. Clock Component's Basic Tab

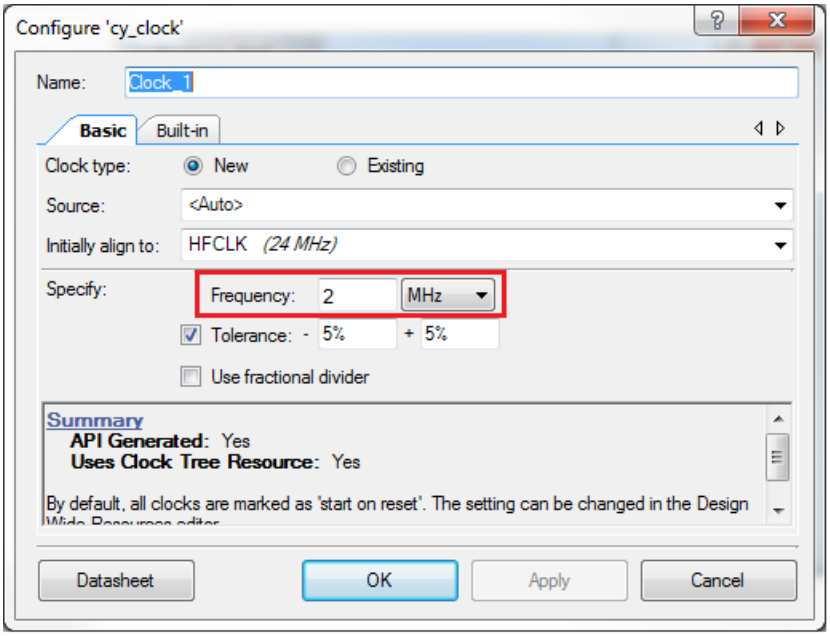


Figure 23 shows the settings for the Pin Component.

Figure 23. Pin Component's Basic Tab

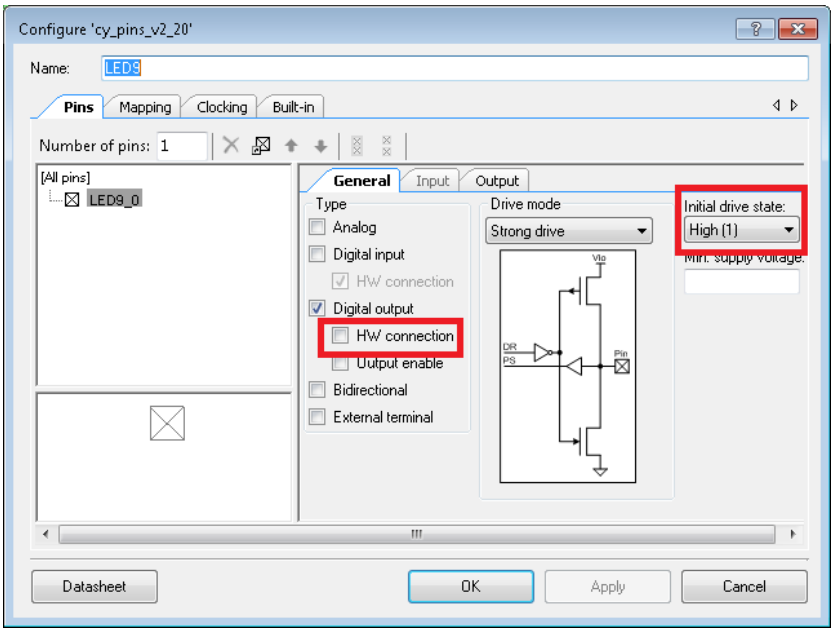


Figure 24 and Figure 25 show the settings for the SmartIO Component.

Figure 24. SmartIO Component's General Tab

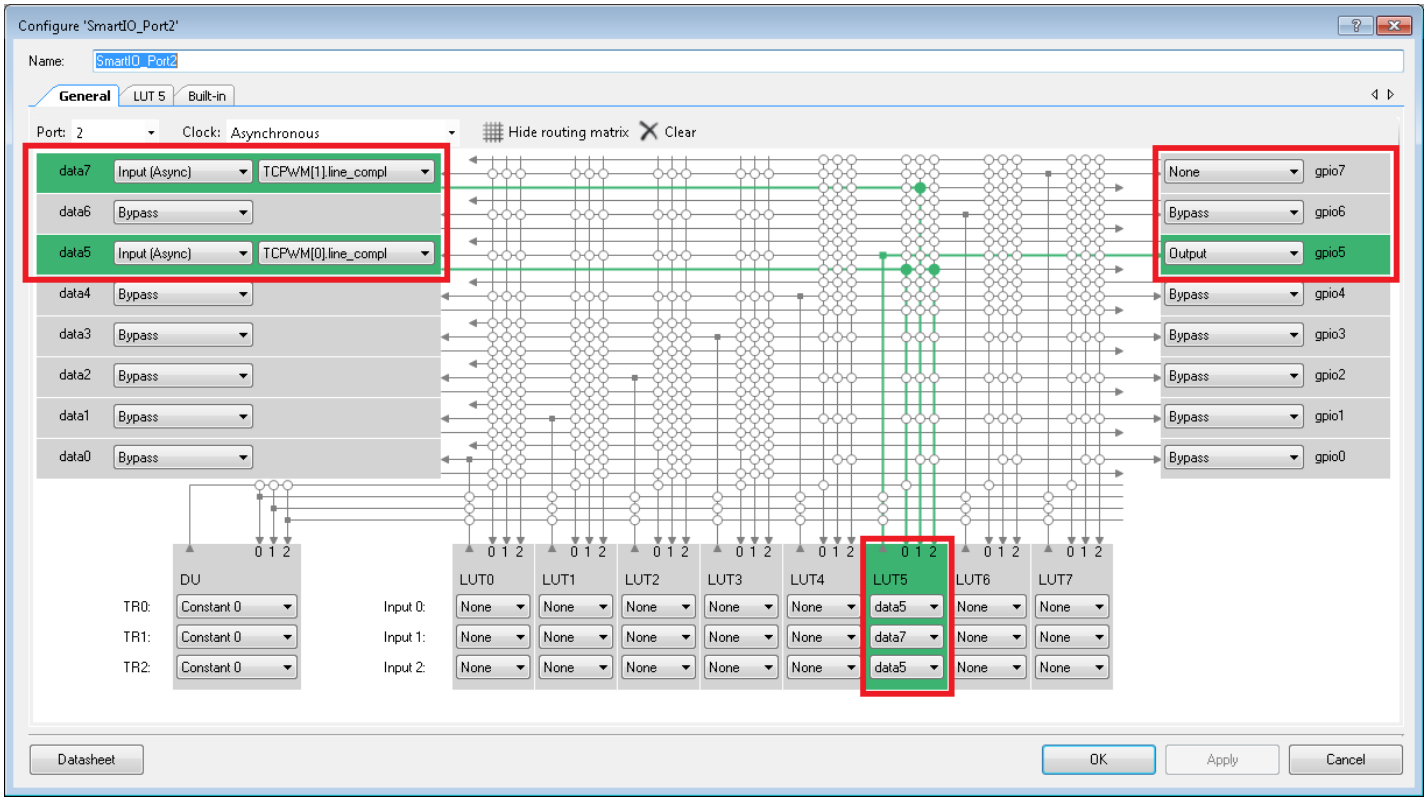


Figure 25. SmartIO Component's LUT5 Tab

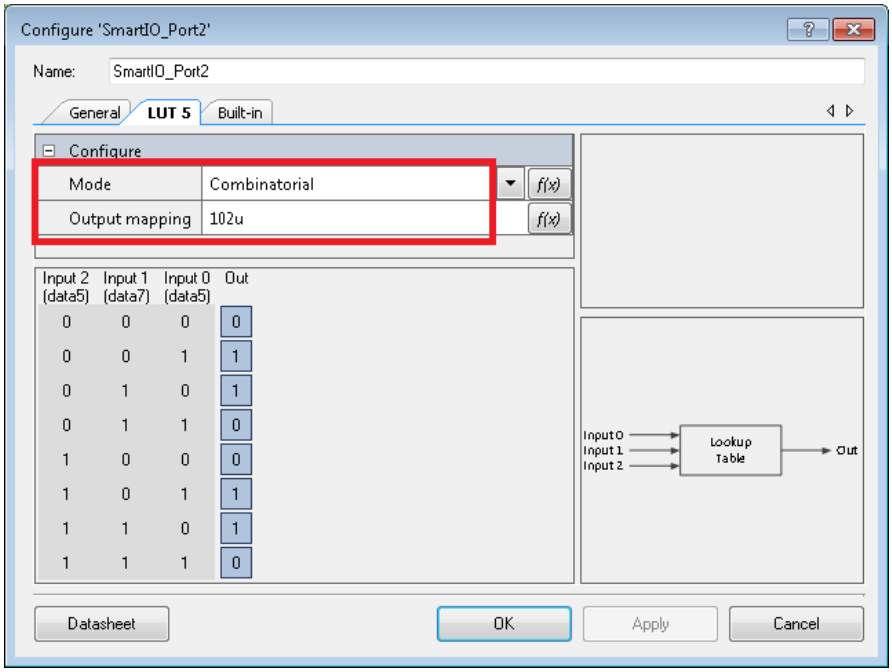


Figure 26 shows the settings for the BLE Component.

Figure 26. BLE Component's Profiles Tab

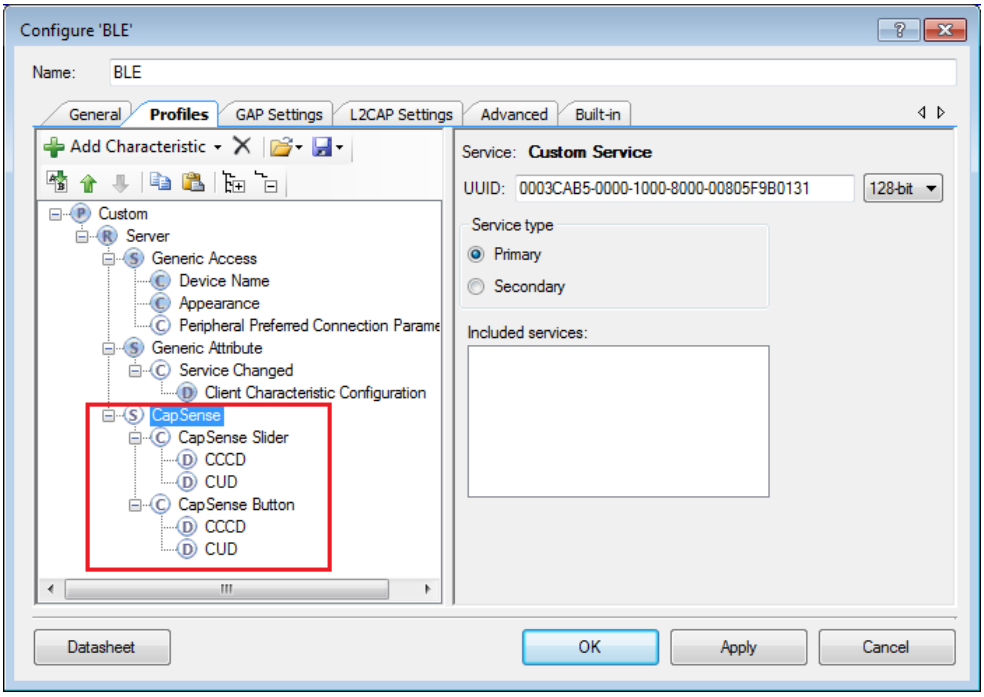


Figure 27 shows the settings for the I²C Component.

Figure 27. I²C Component's Basic Tab

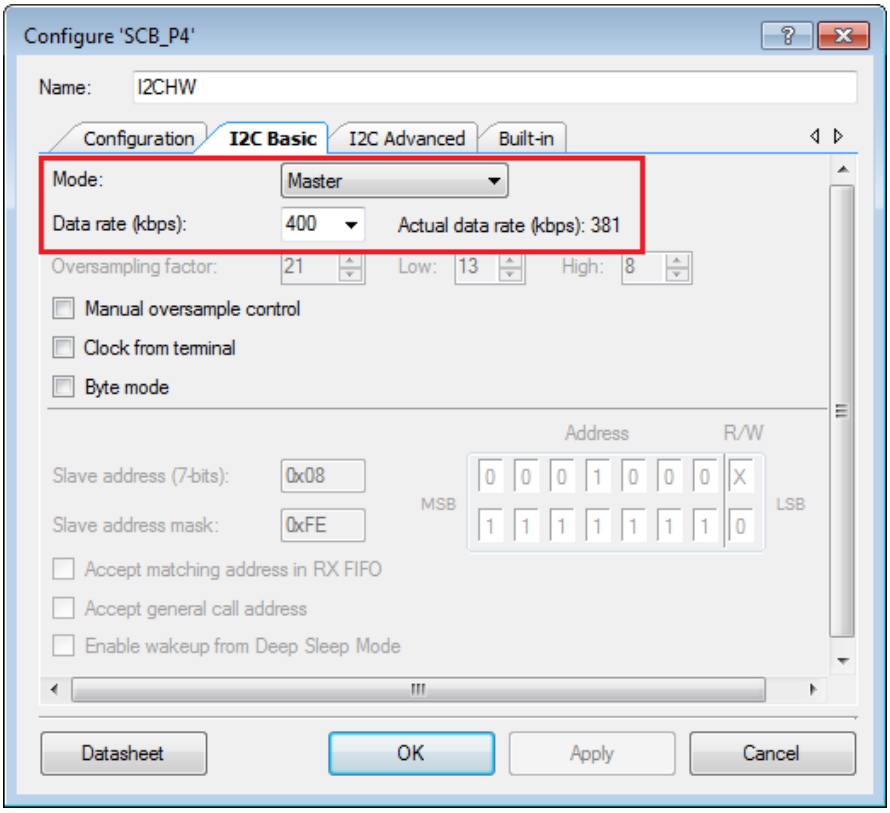


Table 4 and Table 5 show the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window.

Table 4. Pin Assignment for the CE210709_CapSense_Linear_Slider_and_Buttons Project

Pin name	CY8CKIT-145-40XX (CY8C4045AZI-S413)
\CapSense:CintA\	P4[2]
\CapSense:CintB\	P4[3]
\CapSense:Cmod\ (Cmod)	P4[1]
\CapSense:Rx[0]\ (BTN0_Rx0)	P1[4]
\CapSense:Rx[1]\ (BTN1_Rx0)	P1[5]
\CapSense:Rx[2]\ (BTN2_Rx0)	P1[6]
\CapSense:Sns[0]\ (SLD_Sns0)	P0[0]
\CapSense:Sns[1]\ (SLD _Sns1)	P0[1]
\CapSense:Sns[2]\ (SLD _Sns2)	P0[2]
\CapSense:Sns[3]\ (SLD _Sns3)	P0[3]
\CapSense:Sns[4]\ (SLD _Sns4)	P0[6]
\CapSense:Tx[0]\ (BTN0_Tx)	P1[3]
\EZI2C:scl\	P1[0]
\EZI2C:sda\	P1[1]
\SmartIO_Port2:internalPin_o7\	P2[7]
LED1	P2[5]
LED4	P2[0]
LED5	P2[1]
LED6	P2[2]
LED7	P2[3]
LED8	P2[4]
LED9	P3[4]
LED10	P3[5]
LED11	P3[6]

Table 5. Pin Assignment for the CE210709_EZ-BLE_Peripheral Project

Pin name	CYBLE-022001-00
\I2CHW:scl\	P5[1]
\I2CHW:sda\	P5[0]

Reusing This Example

This example is designed for the CY8CKIT-145-40XX PSoC 4000S Prototyping Kit. To port the design to a different PSoC 4 device, kit, or both, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

Application Notes	
AN85951 PSoC 4 and PSoC Analog Coprocessor CapSense Design Guide	Design Guide shows how to design capacitive touch sensing applications with PSoC 4 and PSoC Analog Coprocessor
AN79953 Getting Started with PSoC 4	Describes PSoC 4 and explains how to build a first PSoC Creator project
PSoC Creator Component Datasheets	
CapSense	Supports various interfaces such as Button, Matrix Buttons, Slider, Touchpad, and Proximity Sensor
EZI2C Slave	Supports one or two address decoding with independent memory buffers
PWM	Supports Terminal Count Output for 8-, 16-, 24-, and 32-bit sequence lengths
Pins	Supports connection of hardware resources to physical pins.
SmartIO	Supports glue logic functionality, combinatorial and clocked operations on IO ports
I2C	Supports I^2C Slave, Master, Multi-Master and Multi-Master-Slave configurations
BLE	Supports BLE 4.2 connectivity
Device Documentation	
PSoC 4000S Family Datasheet	PSoC 4000S Technical Reference Manuals
Development Kit Documentation	
CY8CKIT-145-40XX PSoC 4000S Prototyping Kit	

Document History

Document Title: CE210709 – CapSense Linear Slider and Buttons

Document Number: 002-10709

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5999211	SRDS	12/27/2017	New code example.
*A	6078151	SAGA	03/12/2018	Updated template

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 Forums | 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, 2018. 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.