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.

CE229487 – PSoC 4 CapSense Structure Access

Objective

This code example shows two different ways for PSoC® 4 CapSense® structure access: predefined macros that point to variables in the CapSense data structure, and CapSense `GetParam()` and `SetParam()` functions. This example assumes that you are familiar with CapSense design.

Requirements

Tool: PSoC Creator™ 4.3

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

Associated Parts: All PSoC 4 MCU parts with CapSense

Related Hardware: CY8CKIT-042-BLE-A Bluetooth Low Energy 4.2 Compliant Pioneer Kit

Overview

This example contains two PSoC Creator projects that show different ways for PSoC 4 CapSense Structure access. Each example shows how to change a CapSense finger threshold variable. The variable is displayed using a UART.

- **Macro Access:** Uses predefined macros that point to variables in the CapSense data structure.
- **CapSense API Get/Set:** Uses CapSense `GetParam()` and `SetParam()` functions to access variables in the CapSense data structure.

See Data Structure Configuration for details about the architecture of the CapSense data structure.

Hardware Setup

This example uses the kit's default configuration. See the kit guide to ensure that the kit is configured correctly.

Software Setup

This design requires a terminal emulator such as PuTTY or Tera Term running on your computer.

There are two PSoC Creator projects: in the PSoC Creator workspace, right-click the project and click **Set as active project**.

Operation

The same instructions apply to each example.

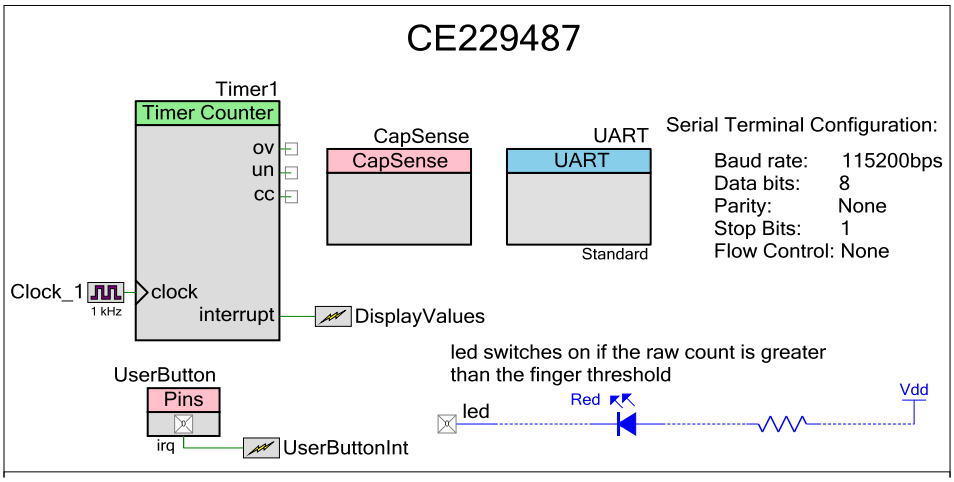
1. Plug the CY8CKIT-042-BLE-A kit board into your computer's USB port.
2. Build the project and program it into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help in the PSoC Creator toolbar.
3. Open a terminal emulator on your computer and configure the program to the appropriate COM port. Configure the baud rate to 115200, data bits to 8, no parity bits, stop bit as 1, and no control flow.
4. Press the CapSense button (P2.1 on the CapSense slider) and observe the current finger threshold on the terminal. The LED turns ON when the signal value is greater than the finger threshold value.
5. Press the user button (SW2) and observe the new finger threshold that is set. Each press of the user button increases the finger threshold by 50.
6. Increase the finger threshold by pressing the user button. When the finger threshold is greater than the signal value, the LED stays OFF when the CapSense button is pressed.

7. Press the RESET button to reset to the default settings of the finger threshold to 10.

Design and Implementation

This example has two projects, Macro Access and Get/Set API Access. Figure 1 shows the top design that is the same for both projects.

Figure 1. Top Design



Both examples perform the following functions in main:

1. Configures the Components.
2. Turns ON the LED when the CapSense button is touched and the CapSense signal (raw count-baseline) is greater than the finger threshold.
3. Increments the finger threshold when the user button (SW2) is pressed.
   a. When the user button is pressed, the `UserButtonInt` interrupt is triggered and the `newDisplayFlag` is set.
   b. If `newDisplayFlag` is true, increments the finger threshold by 50. The new finger threshold and the current signal value are displayed on the UART.
4. Periodically displays the current sensor and finger threshold value on the UART.
   a. An interrupt is triggered at a constant interval to set the `displayFlag`.
   b. If `displayFlag` is true, the current threshold is displayed on the UART. The current signal value will also be calculated and displayed on the UART.

Macro Access

The macro access design shows directly accessing the CapSense data structure using the macro for the variable to be accessed. This macro is defined in *Generated_Source\PsoC4\CapSense\CapSense_Register.h*. This method allows direct access to the variable in the structure. This is the fastest method; however, note that writing into read-only variables is not blocked and the CRC (Cyclic Redundancy Check) is not updated. If the CRC is not updated, there could be CRC BIST (Built in Self-Test) errors when BIST is enabled. It is best to use this when reading a variable and not writing to one. The Top Design of the project is shown in Figure 1.

The macro uses standard C syntax to read and write to the variable.

```
/* Using macro to set parameter value, defined in CapSense_RegisterMap header file. */
CapSense_BUTTON_FINGER_TH_VALUE = value;

/* Using macro to get parameter value, defined in CapSense_RegisterMap header file. */
Param = CapSense_BUTTON_FINGER_TH_VALUE;
```

API Get and Set Access

The API Get and Set access design shows using getter and setter function calls to access the CapSense data structure. These functions are defined in the *Generated_Source\PsoC4\CapSense\CapSense_Structure.c* file. This is the recommended method for reading and writing to the data structure. This is because these functions check whether the parameter is writable and validate the parameter. The `setParam()` function also checks whether self-test is enabled, and updates the widget or global CRC. The downside is this is the slowest at accessing the data structure due to multiple checks. The Top Design of the project is shown in Figure 1.

The `GetParam()` function takes two parameters: the `paramID` and a `*value`. The `paramId` is the predefined macro that is in the *Generated_Source\PsoC4\CapSense\CapSense_Register.h* file. This points to the specific parameter that will be accessed. The `*value` takes a `uint32` pointer that the function will set with the retrieved parameter.

```
cystatus CapSense_GetParam(uint32 paramId, uint32 *value)
```

The `SetParam()` function takes two parameters: the `paramID` and a `*value`. The paramId is the predefined macro that is in the *Generated_Source\PsoC4\CapSense\CapSense_Register.h* file. This points to the specific parameter that will be accessed. The value is the value that the parameter will be set to.

```
cystatus CapSense_SetParam(uint32 paramId, uint32 value)
```

Data Structure Configuration

The CapSense data structure is the only data container in the Component. It serves as storage for the configuration and the output data. All other Component firmware part as well as an application layer and Tuner software use the data structure for communication and data exchange. See the CapSense PSoC Creator Component datasheet.

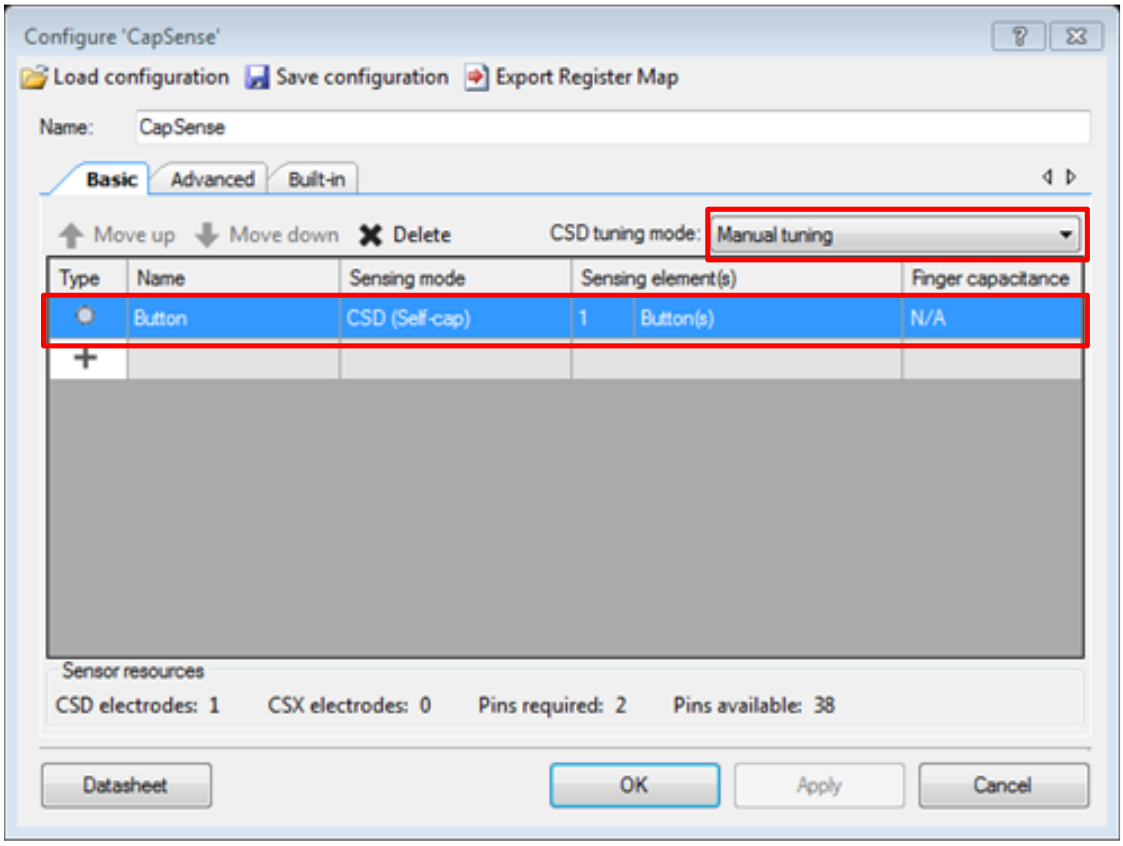
Components and Settings

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	Capacitive sensing	See Figure 2 for all projects
UART	UART	Print text to the PC terminal	None
Digital Output Pin	led	Connect to Pioneer board LED	HW Connection- not checked
Digital Input Pin	Button	Connected to SW2 on Pioneer board	Drive mode: resistive pull up HW connection: not checked Interrupt: Falling Edge- dedicated interrupt

Figure 2. CapSense Configuration



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

Reusing This Example

This example is designed for the supported kit(s). To port the design to a different PSoC 4 device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

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 will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a particular device supports.

The following three kits have the pinouts assigned and will automatically change when the target device is changed for the project. Note that the pins cannot be locked in the pin selector for the auto assignment to work.

Kit	Pins		Notes
	Name	Port	
CY8CKIT-145-400XXPSoC 4000S Prototyping Kit	\CapSense:Cmod\ (Cmod)	(4,1)	To switch to the correct device, in Device Selector, right-click the project and select the correct device. Note that the SwitchTH interrupt InterruptType setting must be set to **Derived**. This device cannot be configured with a Rising Edge interrupt.
	\CapSense:Sns\ (Button_Sns0)	(1,4)	
	\UART:rx\	(3,0)	
	\UART:tx\	(3,1)	
	Button	(0,7)	
	Led	(2,5)	

Kit	Pins		Notes
	Name	Port	
CY8CKIT-042 PSoC 4 Pioneer Kit	\CapSense:Cmod\ (Cmod)	(4,2)	To switch to the correct device, in Device Selector, right-click the project and select the correct device. The USB-UART Bridge must be connected from the KitProg to PSoC 4. Port 0.5 must be connected to 12.7 of KitProg and 0.5 connected to 12.6. For more information on the USB-UART bridge see the Kit Guide.
	\CapSense:Sns\ (Button_Sns0)	(1,5)	
	\UART:rx\	(0,4)	
	\UART:tx\	(0,5)	
	Button	(0,7)	
	Led	(1,6)	
CY8CKIT-044 PSoC 4 M-Series Pioneer Kit	\CapSense:Cmod\ (Cmod)	(4,2)	To switch to the correct device, in Device Selector, right-click the project and select the correct device.
	\CapSense:Sns\ (Button_Sns0)	(4,4)	
	\UART:rx\	(7,0)	
	\UART:tx\	(7,1)	
	Button	(0,7)	
	Led	(0,6)	

Related Documents

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

Application Notes	
AN79953 - Getting Started with PSoC 4	Describes getting started with PsoC4 MCU.
AN64846- Getting Started with CapSense	Describes getting started with capacitive touch sensing (CapSense).
Code Examples	
CE195286 – PSoC 4 CapSense Tuner	Demonstrates how to implement Tuner GUI interface for CapSense design using UART and I2C interfaces in PSoC 4 devices.
CE210488 - LP CapSense Buttons	This code example demonstrates how to implement low-power CapSense buttons with an average current consumption of 5 uA per button.
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
CapSense	Supports capacitive sensing
Interrupt	Supports generating interrupts from hardware signals
Serial Communication Block (SCB)	Supports I2C, EZI2C, SPI and UART communication
Device Documentation	
PSoC 4 Technical Reference Manuals	
Development Kit Documentation	
CY8CKIT-042-BLE-A Bluetooth Low Energy 4.2 Compliant Pioneer Kit	
Tool Documentation	
PSoC Creator	Look in the downloads tab for Quick Start and User Guides

Document History

Document Title: CE229487 – PSoC 4 CapSense Structure Access

Document Number: 002-29487

Revision	ECN	Date	Description of Change
**	6825378	03/05/2020	New code example

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

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