CapSense

Welcome back to Cypress Academy, PSoC 6 101. In this video, I will show you how to add a capacitive-sensing interface to the project we've been working on to provide a local touch interface on the PSoC 6 BLE Pioneer Kit to control the robotic arm.

CapSense is Cypress’ capacitive-sensing technology. It’s the same technology that nearly everyone uses every day - it’s in your smart phone, your wearable device, your stove, your refrigerator, your smart thermostat, it’s everywhere – it really is. It enables designers to remove clunky mechanical buttons and switches, and it really makes a nice sleek, sexy interface. What’s great about Cypress’ CapSense solution? Well, it’s awesome because it's easy to use. Now everyone asks me: why it’s so easy and why does it work so well? And I say, you know, we really didn't have to do that much to make it work…it only took us a couple dozen guys…years and years of research and development…millions of dollars in chips and software…and a whole boatload of patents. And that's all it took to make it easy, and what I say to you is: if you want your device to work you should use Cypress' CapSense.

The bottom line is, CapSense is built on over 16-years of research and development, it's experts optimizing and perfecting the algorithms, and the sensing technologies and the IP. It was all done to make a solution that just works out of the box. Look, don’t just take my word for it, you’re going to see for yourself in this lesson.

Let’s start with a new project that I’ll call “BasicCapSense”.

In this project I want to use the capacitive slider and the two capacitive buttons on the PSoC 6 BLE board to act as a dimmer and an on/off switch for the Red LED. I’ll also use the Blue LED to indicate the status of the dimming function. If the blue LED is on, then the dimming is turned off.

So, we’re controlling an LED, so we know we’re going to need our trusty PWM and LED setup that we’ve been using for the last several lessons -so let’s just copy and paste that from the BasicI2C lesson’s schematic.

We also need an additional digital output for the blue LED. So, drag and drop that or copy and paste the Red LED digital output pin. Double click and change its name to Blue and deselect the hardware control option.

Now we need a CapSense component. Drag and drop that into the design. Let’s configure that component. Start by renaming it to CapSense, dropping the underscore 1. Now let’s add the linear slider and buttons. Click the plus sign, pick linear slider. Click the plus sign again and add button0, then again for button1.

We’ll use the CSX Mutual-Cap scheme for buttons 0 and 1.

Let’s pause here for a moment…CSD and CSX are two different types of patented capacitive sensing schemes that Cypress has developed and perfected over the years. CSD is our self-capacitance mode and CSX is mutual-cap. We have lots more documentation and getting started guides which you can find right here <point>.

Next, click on the advanced tab, CSX settings sub-tab, and set the modulator clock frequency to 12,500. Then click on the widget details sub-tab. The two buttons share the same TX pin. So, click on Button1\_TX and change the selected pin's setting to Button0\_TX.

Okay.

Now, let’s assign the pins for the two LEDs and the CapSense widgets. The red LED is P0[3], the blue is P11[1]. Because we have both a CSD and a CSX set of CapSensing widgets, the scheme requires three capacitors that are connected to P7[1], P7[2], and P7[7]. The two button RX pins are on P8[1] and P8[2]. The linear slider is a 5-element slider, so the 5 pins connect to P8[3] through P8[7]. And finally, the button TX pin is connected P1[0].

Let’s generate the application.

Alright, on to the firmware. In the CM4 main application, let’s start the CapSense component, start the scanning, and turn on the PWM.

Now, if our CapSense hardware block is not busy, then we’re allowed to ask it what is the state it's in. So, check if it’s not busy; if it’s not busy, process the widgets, find the middle of the finger on the slider - the centroid position as it's known. If it’s being touched, as indicated by a value less than 0xFFFF, then we’ll set the compare value of the PWM. Since the possible values of the slider are 0 to 100, it matches up nicely with the possible compare values for the PWM.

We’ll now check to see if someone is touching button 0. If it's being touched, then we’ll turn the PWM off and turn on the blue LED.

If someone is touching button 1, then we’ll turn the PWM on and turn off the blue LED.

Finally, we need to update the baselines, which represent the environment that the board is sitting in. This is where some of the magic happens. You want your board to be robust regardless of the temperature, humidity, location, and don't forget there's even the other manufacturing variances – how thick the printed circuit board is, how thick the overlay is. All of these things are key in making your wearable and your IoT device work well with CapSense, and we do it automatically.

Now, we need to start the scanning again.

Alright, that’s it. Let's build, program and test.

Hey, look, when I move my finger on the slider you can see how the intensity varies. And when I push button 0 the blue LED turns on and the slider is disabled. And when I push on button 1 the red LED is back on again and I can use the slider to vary the LED intensity.

Sweet!

Now we have the basic CapSense implementation working. In the next video we’ll add this functionality to the BLE-controlled robotic arm project.

As always, you can post your comments and your questions in our PSoC 6 developer community or you're welcome to email me at alan\_hawse@cypress.com or tweet me @askioexpert with your comments, and your suggestions, and your criticisms, or just to chat, or about any other questions you have about Cypress. Thank you.