BLE CapSense Remote Control

Welcome back to Cypress Academy, PSoC 6 101. So far, we've created a BLE-controlled robotic arm, and added a second PSoC 6 BLE Pioneer Kit to the system to act as a remote control. The first version of the remote control wasn't that exciting as it used UART keyboard commands…. So, let's make it a bit more exciting. We are now going to add CapSense capacitive-sensing to our BLE Remote Control.

So, open up the remote-control project from last time and open the schematic. Now add the CapSense component. I think that we will use the slider to move the robot arm… and we will use the buttons to select which motor to move.

Change the name to CapSense. The add a linear slider and two buttons. We will use the CapSense buttons in mutual cap mode. So, change that setting. This board has only one transmit pin for the CapSense buttons... meaning they share the transmit pin… so go to the advanced tab -> widget details… and change the Button1_tx to be Button0_tx.

OK... we added a bunch of pins to our design... they are buried in the component… but we still need to assign them… so open up the DWR pins configuration window.

Now Assign the pins for the linear slider to pin P8[3] -> P8[7], The Button0 RX to P8[1] and Button 1 to P8[2] … then the Button Tx to P1[0] … then you assign the capacitors to their default locations… see they are labeled in green when you do the pulldown menu.

All right … now because I am into code reuse, I'll copy the capsenseTask.h and .c from the MainController project… so expand the main controller… then ctl-c to copy the .h file… then ctrl v it into the header files of my remote control project… all right… now do the same thing to the capsenseTask.c … except that goes in the source files.

Now that I have bleTask.h/.c , capsenseTask.h/.c and uartTask.h/c let's go one by one and make sure that they are what we want.

First the UART. I still want to be able to send commands based on the keyboard…. So, I think that I'll just leave them alone.

Next… bleTask.h/.c … we tested this earlier and it seemed to work just fine… in our capsenseTask I am just going to call the writeMotorPosition function… OK so I don't need to change that either.

Now the CapSense task files… hmm… they are going to be very similar to the main controller… First capseneTask.h… let's see here… that will be exactly the same. The only thing that it needs is a function prototype of the capsenseTask so that the main function can start it up.

Now… the capsenseTask.c … start with the includes… this task needs to know about the project, FreeRTOS, tasks, and the bleTask… so let me do those includes.

I'm going to use the writeMotorPosition function that I wrote in the last video… so I can delete all of this stuff about PWM messages… now look at how awesome this is… all I need to do is when I there is a touch on the CapSense slider… I just take that number and call the writeMototPosition… that's it.

The last thing that has to happen is that I need to start the CapSense task in main… so edit main_cm4.c … add the capsenseTask.h include… then call the task startup function…. Sweet.

Hit program…

As soon as the programming is done look at how fast it connects… see the red lights… OK. now I run my finger on the slider… back and forth and back and forth… now press button two and see if we can run the other motor… sure enough… this works as well.

Cool.

Now we have a BLE remote control with CapSense and a BLE-controlled robotic arm. Next step, let's add in some sensors. For the next few videos, we'll be implementing the motion sensor and the E-ink Display for the BLE remote controller!

You can post your comments and questions in our PSoC 6 community or as always you are welcome to email me at alan_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions.