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 lets make it a bit more exicting. We are now going to add CapSense capacitive-sensing to our BLE Remote Control for our robotic arm.

So, open up your remote control project and then open the schematic. Now add the the capsense component. I think that we will use the slider to move the robot arms… and we will use the buttons to select which axis to move.

Now change the name to capsense. The add linear slider and two buttons. We will use the capsense buttons in mutual capo 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 they are there… so open up the DWR pins configuration window.

Now Assign the pins 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 location… see they are labled in green when you do the pulldown menu

All right … now I am going to cheat a little bit to make things simpler. Because I am into code reuse Ill 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 lets put in in the source files.

Now that I have bleTask.h/.c , capsenseTask.h.c and uartTask.h/c lets 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 Ill just leave them along.

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.

Now the capsense task files… hmm… they are going to be very similar to the main controller… First capseneTask.h… lets see here that will be exactlhy the same. The only thing that it needs is a function prototype of the capsensetask so that the main function can startup the capsense task.

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.

Im 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 is it.

The last thing that has to happen is that I need to startup the capsensene task in the 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 axis… sure enough… this works as well.

Cool.

Now we have a BLE remote control with CapSense and a BLE-controlled robotic arm. Next step, lets 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.