3-3B: BLE UART Remote

Welcome back to Cypress Academy, PSoC 6 101. In the video 3-3a I showed you how to use PSoC 6 BLE to build a BLE Central that could control the project from 3-2a Simple BLE Peripheral LED Dimmer.

In this video we are going to start building a complete BLE remote control for the robot arm. When the remote control is done you will be able to use the UART, CapSense and the Bosch motion sensor to control the PSoC BLE Robot.

I think that I will start this project by making a copy of the project from video 3-3a and carving it up…. Let's start by editing the schematic. Open up the BLE component customizer, go to GATT Settings and delete the LED Service… then add in the motor service by right clicking on Client, Add service, From file… and picking the file for the motor service.

Now run generate application so that we get all of the BLE updates into the middleware and generated source. You are going to get a bunch of errors… but don't worry about them for now.

Now let's carve up the firmware into a more manageable structure. I'll follow the template that I have in the MainCotroller... meaning I'll have the BLE stuff isolated into bleTask.h and .c … and I'll have a UART based controller in uartTask.h and uartTaskc.

Start by making a file bleTask.h.

I know that we are going to want to send changes in motor positions… so let me make an enum with the motors. I'll also make an enum so that I can specify whether I want an absolute or a relative motor position change. Now let me create the protype for the function that will make the motors move… when I call that function I'll send it a motor number, change type, and a percent.

The last thing that is needed in the bleTask header file is a definition of the task.

Next, I'll make bleTask.c by right clicking the source files folder and picking add new item… C file … then call it bleTask.c

Let's add in the includes that we need.

Now I am going to go to main_cm4.c and move from the top of the writeLED function through the end of the BLE task into bleTask.c

I am going to modify the writeLed function… so I'll copy the function prototype from bleTask.h and replace the writeLed function declaration.

Instead of brightness let's printout the information about the requested motor movement.

If you recall from the previous video, in order to write, we need to know which handle to write to. If you remember from the BLE Motor Control Service, there are four possible handles that we are interested in: M1, M2, M1 Relative and M2 Relative.

When we do the service discovery, our BLE stack will discover those characteristics and build an array of handles for those characteristics called cy_ble_customCServ [which service] dot customServChar [which index] dot customServCharHandle[0]

So, in order to figure out the handle we need to do 4 if statements that lookup the correct handle based on M1 or M2 and Relative or Absolute.

After that you assign the percent to the right variable… then write it using the Cy_BLE_GATTC_WriteChractersticValue function.

Now I need to change the scanner to look for the motor service instead of the LED service. I'll change the comment… then the index… and finally the printf message.

When I wrote this code originally, I was lazy and didn't put in the BLE Semaphore. Let's fix that … first I'll add the semaphore to the top of the file… then I'll copy the interrupt service routine and bleTask from the previous BLE project, then I'll paste it into my bleTask. And finally, I'll chop out the stuff about event groups. Now we have a nice generic bleTask handler.

Now I need to add the semaphore into FreeRTOS.h, and update the MAX_SYSCALL_INTERRUPT_PRIORITY.

Almost done.

Make the file uartTask.h. This file will only have the pragma once and the definition of the uartTask.

Now one more cheat. I am going to just copy the uartTask from the BLE MainController into my project since it is almost exactly what we want.

First, add "bleTask.h" to the top. Then I'll add key commands for o … p …j … l… that will just call the writeMotorPosition function…. Which I will also add to the help printout.

OK program your development kit.

When I look on the UART, it starts searching for device… and right quick you can see that it finds the robot… and both connection lights turn on.

Now when I press the o and p buttons you can see the arm move back and forth. Sweet!

Next time we will add CapSense to our remote control.

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.