3-2b BLE - MainController

Welcome back to Cypress Academy, PSoC 6 101. In the last lesson I showed you how to build a BLE Peripheral with a custom service for an LED Dimmer. In this lesson I think that Ill add BLE to the maincontroller project… and modify it from LED dimming to motor control.

First, lets add the BLE component to the schematic. Then configure it for peripheral, 1 connection, dual core. Once that is done I want to create a motor service… well actually I already did that for you. So. Right click on the server and selected add service from file. Pick out the motor.service file and hit go… PSoC Creator will load in the configuration I created. When you look at it, you will see a custom service called Motor. The motor service will have four characteristics… 2 which we will use to set the position of the motor… and 2 that you will use to make relative changes to the position. Notice each characteristic has a characteristic user description as well as a client characteristic configuration also known as a CCCD. The M1 and M2 characteristics are uint8 that is read, write and notify… the notify is something new which ill tell you more about later in this video.

Now lets setup the GAP settings. Give this beast a name… how about P6ROBOT. On the advertising settings set it to “General discovery” and no timeout on the fast advertising.

Then in the advertising packet lets advertise the local name and the motor service UUID… this will let our remote control find us… more on that in a future video.

I told you in the previous videos I like to use LED9 to show that there is an active connection… so add a digital pin called LED9, no hardware connection and initialized high… then onto the DWR where I need to assign it to P13[7]

Next… generate application to pull in all of the BLE Middleware and let PSOC creator get everything connected

Remember from the previous video we need to run the controller in the CM0+ … so edit main\_cm0p.c … start the BLE and then process events in the main loop.

Now I will create bleTask.h … it will have pragma once … and a definition of the bleTask

Onto the main\_cm4.c … Ill add the include for the bleTask.h… define a task handle for the bleTask… and the startup the BLE Task.

Then I need to add an extern in global.h so that I can know the task handle of the ble task in order to send it semaphores.

And last but not least… I need to actually do the BLE work by creating bleTask.c. This task is going to be a lot like the LED Peripheral BLE Task. So … I think that Ill start by copying from that project. I will copy from the top all the way to just before the main.

At the top ill include global.h… and then Ill delete the task handle.

Now we need to make some changes to the ble event handler. First instead of a blinking led…we are using just led9…. So when there is no connection it will be off… and when there is a connection it will be on … so lets see here Cy\_GPIO\_Write(LED9\_PORT,LED9\_NUM,1); … now delete the TCPWM stuff..

And… when there is a connection turn on the LED9

Now the write request stuff is going to have to be redone… but for now I am going to put and #if 0 and an #endif… and see if it will start advertising… and we can connect… obviously we won’t be able to write to the characteristics… but that is OK… it is a good place to start. So hit the program and let it rip.

When I startup CySmart, I see that a peripheral called P6ROBOT is advertising.. that’s good. And when I connect… the LED9 turns on… and I can see the characteristics.

All right back to the bleTask.c

First lets add an include for the pwmTask so that I can send it messages.

Next I want to make a function called updateMotorGatt that will be responsible for actually writing values into the GATT database… meaning changing the characteristics for the motors.

There are two possibilities for changing of the position of the motors. It could happen locally because the user touched the capsense … or typed uart commands … or it could be getting a command from the BLE Central that is attached to it. Either way the GATT database needs to be kept up to date.

The reason that the GATT database needs to be up to date is because the remote control end for sure wants to be able to read the current position of the motors. So lets build this function so that it can make updates to the GATT database that are initiated locally … for instance by the PWMTask … or remotely.. from the BLE CEntral

The function takes motor … remember M1 or M2 from before… a value and a mysterious flag… actually the flags will tell the system if it is a local write … meaning the motors changed positions locally… or a write from the BLE central side.

When you call the Cy\_BLE\_GATTS\_WriteAttribute function you need to give a pointer to a “cy\_stc\_ble\_gatt\_handle\_value\_pair\_t” … so Ill declare one of those beasts.

Then Ill error check the percent to make sure it’s in range. Then figure out the handle of the characteristic we are talking about… either M1 or M2.

Once that is done Ill figure out if it is a local write or a remote write. If it is from the peer… or BLE Central side.. Ill first write it into the GATT database… then I want to send a message to the PWM that the value of the motor has changed… so Ill build that build up the message… and send it.

If it is a local write… then I just need to write into the database.

Now… remember earlier I told you about the CCCD … well here is where it comes into play. It is possible for the central side to ask to be notified if the value of a charactertic changes…. And remember that we setup M1 and M2 with this notify capability… well after we write the attribute to the database we need to call the CyBLEGATTSNotification function which will figure out if they have asked for notify… and then send the notification.

Remember that I #if out a section of the ble event handler… ok lets go put it back in.

When you get a CY\_BLE\_EVT\_GATTS\_WRITE\_REQ there are 6 possibilities. The central wrote M1 or M2 … the central write M1 relative or M2 relative … or lastly the central asked to be notified of changes to the M1 or M2 characterstics.

So lets deal with these six posibilties.

If it is a write to M1 then we will use the handy dandy helper function updateMotorsGatt.

Same for M2

If the Central asked for a relative change in M1 then lets make a PWM message that requests a relative change… and then send it.

And what do you know… the same for M2 relative.

And finally if the ble central changed the CCCD of M1 or M2… then lets make one of these crazy cy\_stc\_ble\_gatts\_db\_attr\_value\_info\_t … and fill it out with the connection and value.

And then write it into the database using the Cy\_BLE\_GATTS\_WriteAttributeValueCCCD.

That’s it… oh hang on… for all of these cases we need to send a write response… so call cy ble gatts write rsp.

Now that we have fixed up the ble event handler… you need to make one little change the event loop. Specifically after you have been woken up… you should check to see if the PWM value has changed locally… which you can tell by using the event bits… and if it has… then you should update the values of M1 and M2.

All right hit program and lets try this thing out.

test

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.