BLE Peripheral - Custom Service

Welcome back to Cypress Academy, PSoC 6 101. In the last lesson I showed you how to build your first PSoC 6 BLE project. In that project we used a Bluetooth sig specified service … specifically the Immediate Alert Service. While I was looking through the Bluetooth SIG specifications for custom services I couldn’t find a terminator robot service definition anywhere. That is a problem because we know that we want to control that thing with Bluetooth. So what do we do? Well the answer to that question is simple. We need a custom service. Is that going to be hard? Nope… not at all.

So what is this project going to do? Simple.

1. When the device turns on it is going to start advertising
2. When there is no connection the red led is going to blink
3. When there is a connection the red led is going to turn off
4. And when there is a connection the other side… also called the GAP Central … will be able to change the brightness of an LED.

Simple enough eh. Allright lets build a project. Get started by creating a new PSoC 6 projected called 3-2-SimpleBLEPeripheral. Next, lets change the build settings to include FreeRTOS and the Standard IO redirection.

Now Ill Go to the schematic and add the BLE Component, A UART, two PWMs, Two digital output pins and two clocks.

Lets wire the two output pins to the two pwms. And the two clocks to the two PWMs.

Now for some configuration. First lets setup the blinking PWM. Double click it… then change its name to PWM\_BLINK. Set the compare value to 500 and the period to 1000. Now change the input clock to 1KZ. And finally rename the LED to be RED. So Im sure that you guys remember from before that this will result in a 1HZ blinking LED.

Lets follow almost the same process for the other PWM circuit. First change the name of the pin to Green. Then change the name of the PWM to be PWM\_DIM. Change the period to 100 and the compare value to 0. This circuit will give us a dimmer… and by chaning the value of the compare from 0 to 100 it will change the brightness from off to all of the way on. Cool.

Now lets configure the BLE. On the general tab I am going to run this BLE in dual processor mode… just like I did in the last example. The device is going to be a peripheral and only allow 1 connection.

Click on the GATT settings. Then right click on the server and select add service… notice that there is no killer robot service in the list of predefined services… maybe Ill send a note to Misha to get him to add it… just joking… In this case we are going to create a custom service… so select “Custom service”.

Right click on the custom service and select rename .. then change the name to LED. Now lets change the name of the custom characterstic to be green. On the other side… remember the gap central or phone side… we want them to be able to write a uint8 so leave this set as uint8 and click “write” so that the perimission will be set as writable. Next lets add some information so that the other side knows what the “green means”. Right click and add descriptor … charceteristic user description… then lets type a description like “Green Brightness 0-100”. Finally delete the custom descriptor as it is not needed.

Now lets configure the GAP settings. First give this bad boy a name… how about P6LED. Then chage the advertising settings so that it never times out (which wastes power… but makes it easier to find)

The last step in the ble configuration is to setup the advertising pack to have the device name and the fact that it has a custom service.

Now we will go to the Design Wide Resources and configure the pins. Set the uart to P5[0] and p5[1] then green to p1[1] and red to p0[3]

OK… run generate application to get PSoC creator to do its magic.

Lets modify the FreeRTOS.h to get rid of the warning, include semaphores, have more heap and set the max syscall interrupt priority.

Then ill modify the stdio\_user.h to know about our project and which uart we are using.

In order for the system to run in dual core mode I need to add the processing commands to the main\_cmop.c … first launch the controller part of the stck… then infinitely loop and process events.

And now we are ready for the main event… main\_cm4.c

At the top I need to have all of the required ihcludes… project.h … freertos.h task.h and semphr.h…

Then I need to declare the two variable that will be used to signal from the BLE interrupt service routine… a task handle for the ble task… and a semaphore called bleSemaphore.

Remember from the last video I told you that you need to build an event handler… well lets do it. First create the function custom event handler. As I told you before, the BLE stack will call this function with an event code and an event parameter to tell you what is happening in BLE land.

I am going to make the code a little bit cleaner by making a variable called writereqparameter.. ill tell you about it in a second.

For this project there are four events that I am interested in.

Stack on … which happens when the stack turns on.

Gap disconnected … which happens when the remote device disconnects

Connection indication … which happens when there is a connection

And write request …w hich happens with the other side send you a write request.

As before … the event handler function is just a big switch.

So… when the stack turns on… OR when there has been a disconnection we are going to do exactly the same thing. Specifically I start the blinking LED pwm. Then Ill start advertising… and finally ill reset the green led pwm and then disable it.

The next case handles a new connection. In this case ill turn off the blinking LED PWM and startup the dimmer PWM.

The last case occurs when the other side writes to my device. Up until now we have only looked at the event codes. Now we need to use the other parameter to the function which we declared as a void pointer… as you guys remember a void pointer is a generic pointer… it can point to anything…. Well in the write request case that pointer is going to point to a structure called the write request parameter. So I will cast the void pointer to a pointer of type cy\_stc\_ble\_gatts\_write\_cmd\_req\_paramter\_t … which is quite a mouth full.

The write request parameter has a bunch of useful information in it… including which gatt characterstic was written by the central. So the next thing that I need to do is make sure that the GAP Central was acutally writing into the Green Brightness characterstic.

If it is… then I extract the value that it wrote…. Make sure that it 100 or less (remember the value from the PWM…. Then it will update the compare value in the PWM which will change the brightness.

Now that I have built the BLE handler… all we are left to do is make the BLE task and the startup code.

Ill start by copying and pasting that code from the last project as it is almost exactly the same. This time we don’t have an IAS callback… so Ill delete that. Then in main Ill start the two PWMs.

Now… the moment of truth. Hit little chip button and build program this dog.

Allright… the light is blinking red… that’s good.

Last time I showed you CySmart on the iPhone. This time Im going to run another application called light blue.

When I run light blue I see the “P6LED”… good

Then when I click it.. the red LED turns off and I can see my GATT database… there is the gree brightness characteristic.

When I click it.. I can write a new value into the green… Ill start by writing 0x64 which is also known as 100 %... and look the green is full bright.

Now Ill try 0x32… yup dimmer…

And when I try 0 … the green goes off…

Now disconnect… and look the green goes off and the red starts blinking again. Cool.

Now we know how to build a custom characteristic… in the next video ill add BLE to our main controller… specifically characteristics for the two motors so that we can change the position of our robot arm with a 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.