PSoC BLE 101: 2. Complete the Find Me Profile Firmware

Hello, my name is Alan Hawse. I'm Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor. In the last lesson we did the first half of a Find Me application. First we created a new PSoC4 BLE design and then we configured the BLE component.
Recall that I talked about GAP- the mechanism by which devices connect. And I talked about GATT Ð which is the mechanism by which devices exchange data. I also talked about Profiles, Services and Characteristics. Recall that our Find Me peripheral will have an Immediate Alert Service that contains and Alert level characteristic.

In this lesson we will take the Find Me profile that you configured in the first lesson as well as add a little bit of firmware that is required to make it work on the PSoC 4.
You'll remember that we chose the Find Me profile, then gave the device a name and asked for a unique ID, and finally we asked for the name and the service UUID to be included in the advertising packet.
One thing that makes BLE a little different from other PSoC Creator components is the number of APIs that are needed to support all of the configurations and services. To help you navigate these APIs we added a browsable Help window showing all of them.

It's a good way to learn about the component firmware and find the right API for your needs. I recommend you open it every time and refer back to it as you write your C code.
At this point we have a BLE component that is configured and ready to go.
First, let's add a pin to the project that we will use to drive an LED based on the Alert level I get from the phone. I would like to have the LED be off when there is no Alert, blinking with mid Alert and solid with high Alert.

I will use one of the TCPWMs to perform this function.
Add the TCPWM to your design and configure its period to be 1000 and its compare to be 0. Then add a 1khz clock to drive it. Lastly wire the LED to the line_n output of the TCPWM. A little bit later we will write the firmware to implement this blinking pattern.
Now you need to assign the LED pin to a real pin on the chip. On the pioneer kit, the red LED is assigned to Port 2 Pin 6.

The next step is to "generate application". This will build all of the APIs required to run your device.
The BLE firmware is implemented with event handlers that are scheduled by the stack.
An event handler is simply a C-function that is called automatically by the stack at the right time.
For our application, we will require two event handers.

The first handler is the generic stack event handler. It will be called when the stack starts, or when it establish or tears down a connection, as well as many other situations. You can read about all of the events that can be triggered by the stack in the BLE component datasheet or in the online API browser.
For our application we will simply start the GAP types of advertising process after the stack is started, as well as restart the advertising after a disconnection.

I will implement this functionality using a switch statement to process the type of event. This may seem to be a bit of an overkill, but as we add more events in the future the switch will be a much more robust implementation.

Then I'll set up the event handler for the Alert service.
The immediate Alert service handler is called by the BLE stack when the phone writes into the Alert level characteristic. Remember it can write a 0 for No Alert, a 1 for a Mid Alert or a 2 for a High Alert. To implement this functionality, I will change the compare value of the TCPWM. When I want the LED off I will set it to 0, when I want it to blink I will set it to 500 and when I want it on solid I will set it to 1000. This will correspond to the No Alert, Mid Alert and High Alert states from the Find Me service.
The last step is to build the main function.
First start the BLE stack and register the callbacks.
Then start the PWM.
Lastly, in the main loop we just keep telling the stack to process events.
To understand the data structure, go back to the API documentation.

All right, let's program the kit and see that it works!
I've installed the CySmart App on my phone. You can get this from the Google Play Store for Android or from the Apple App Store for IOS.

First, I'll reset the kit and then refresh the screen. CySmart will give you a list of all BLE devices that it can see advertising. Here you can see my device, which I recognize by the name that I gave it in the GAP settings. Our device says it has an "Immediate-Alert" service. This is because we asked the advertising packet to contain the UUID information. Recall this was set-able from the GAP screen in the BLE Component customizer. This helps me pick out our device when there are a lot of devices around me.
When I select my device it shows me the Find Me profile. If I swipe the screen I can also see the GATT database. But selecting the Find Me gives me this pull-down that I can use to send an Alert.
When I select High or Medium Alert, the red LED blinks appropriately.
When I go back to No Alert, the LED turns off.
So, there you have it, your first BLE application.

In the next lessons I will add another service to this application, I'll talk more about GATT and GAP, and talk about power and lastly I'll show you an example of a custom service.
As always you are welcome to email me at alan_hawse@cypress.com with your comments, suggestions, criticisms and questions.

Watch all PSoC Creator 101 lessons at www.cypress.com/creator101