PSoC BLE 101: 4. Add a Custom CapSense Service

My name is Alan Hawse. I'm Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor.
In the previous lessons, I showed you how to build applications using the built in services. Specifically the "Immediate Alert Service" and the "Battery Level Service". Although the Bluetooth SIG has defined quite a few common Services, which I am sure you saw when you looked at the component. What happens when you want to implement something that isn't in the list? It's really simple; all you need to do is add a "Custom Service".

Let's say you would like to use a cellphone to display the position of your finger on the CapSense slider. We are going to write this application from scratch.
Start by creating a new project.

Then add the BLE component.

Now give it a personality. In the general tab, select Custom profile, GATT Server, GAP Peripheral. I'll configure GAP first. In the GAP settings tab give your device a memorable name. Also, remember to check the silicon generated ID box to create unique UUID for your device.
In the advertisement packet, make sure you tell the client your name and service UUID. That's all that's required to make your device connectable.
In the Profiles tab, you can see we already have a custom service. PSoC Creator built it for you automatically.

Rename that Service to CapSense Ð this name will set the automatically created #define prefix for this Service. Then give it a 16-bit UUID with the value CAB5. That hex value corresponds to the CapSense profile built into the CySmart tool.

By using this UUID we are telling the client that we support this Cypress-specific profile.
You can give your custom service a different UUID if you wish. But if you do that then you will either need to view the raw data in the GATT database or write your own client application to handle the data.

Next we rename the custom characteristic to "slider".
This will set the next part of the #define for this service. Then give it a 16-bit UUID of CAA2. This UUID indicates that the device is a slider, as opposed to a button or track pad.

In the descriptor properties, check "read", so the client can access the data, and notify to enable the Gatt server notification support Ð Notifications are GATT server initiated updates that are sent to the Gatt Client. We do not need the custom descriptor in this example and so you can just delete it.
Finally, add a client configuration descriptor and give it a memorable name. This characteristic is often referred to as the CCCD. The client configuration descriptor is written by the Gatt client (the phone) to enable the Gatt server (your peripheral) to send notifications of data being changed.
That completes the BLE setup.

Next, add a CapSense block and set it up for a slider.
Rename it to just “CapSense” to make the generated APIs simpler. In the widgets tab, add the slider. It defaults to 5 sensors, which is exactly what is on the PSoC 4 Pioneer board.
In the pin editor file, assign the sensors to port 2, pins 1 through 5. Your board has the CapSense modulation capacitor attached to port 4 pin 0 Ð so assign it that way.
Now, generate the application to check your design and to create the APIs.

In main.c, I’ll create three global variables. Two are Booleans for the notification status and the connection state and one is the ID handle of the connection. The use of these global variables will become obvious in a moment.

As you have seen before, we need a handler to react to the GAP events and the GATT database access events. This handler is just a big switch statement.
You need to start advertising when the stack gets turned on or when there has been a disconnection.
When a connection, is made you assign the connection state global variable.
Then you need to handle a write event. This happens after a connection, when CySmart tells the device that it wants to be notified about changes to the CapSense characteristic.

You will store what is written from the Gatt client into the Gatt server database. In addition, you will set the global CapSense notify variable.
Now, let’s go to our main function and start the components.
First, turn on global interrupts.
Next, start the CapSense and begin scanning the slider.
Now start BLE and register the event handler we just wrote.
In the main loop, we process the events, as usual. Just as we’ve done in previous lessons.
Then, we will check to see if a CapSense scan has been completed and we are connected to a client.
There is no point sending information if we’re not connected to anything.
If not, we just loop again and continue processing events.
If there is data ready, we read the slider position and start the next scan.

If there is a finger present, then the return value will be something other than hex FF and so we simply write that to the Gatt database and notify the client of the change of position.
If we program the device and then connect to it from CySmart you can see that moving your finger along the slider updates the screen. That’s pretty cool.
As always you are welcome to email me at alan_hawse@cypress.com with your comments, suggestions, criticisms and questions.
Thank You.
Watch all PSoC Creator 101 lessons at www.cypress.com/creator101