PSoC BLE 101: 5. Extending Battery Life with PSoC Low Energy Modes

Hello, my name is Alan Hawse. I'm Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor.
In the last lessons, we created a Find Me peripheral with a Battery Level service. With most battery-powered devices you want to minimize power consumption, so it seems kind of ridiculous that I haven't put power savings into the design yet. So let's fix that.

With traditional radio applications, receive and transmit power dominates the overall power equation. However, this is not true in BLE. The way to reduce power in BLE is to reduce the active duty cycle of your application. Turn on, get your work done as quickly as possible and then turn off preferably as off as possible.
In an earlier lesson, I talked about the power modes of the PSoC 4 Active, Sleep, Deep-sleep, Hibernate and Stop. All of these modes are available to you in the PSoC 4 BLE.

While PSoC does give you control over the current to the radio circuit, as I mentioned before, this is actually not the crucial factor with BLE. Bluetooth Low Energy is a connectionless protocol. Actually, it is a short connection protocol, meaning that pairs of devices do not remain in communication for long periods. Instead, they advertise, establish a connection, exchange data, and then disconnect, leaving your device free to turn off the radio and go to sleep
That's why you see those advertising rate options under the GAP Settings tab in the Component dialog. Careful selection of these parameters allows you to create a device that is actually off for such a large percentage of time that the radio power is almost irrelevant.
The key to meeting your power budget is to go to sleep whenever you can and as deeply as you can.
Now, upgrade your project to support lower power.
To show off the savings, let's add another pin to the design that will indicate the power state. You can connect this pin to a header on the kit and view it with an oscilloscope. Disconnect J15 on your PSoC 4 Pioneer board; there you can attach inline a multi-meter to measure the current that is going into the chip. Or, if you don't have one, just pick another color on the RGB LED because we will slow things down enough to make that LED visible.

Attach the pin to Port 3 pin 7, if you want to use the blue LED as your power indicator.
Now, let's change the firmware a little and save some power. First, we can slow down the advertising rate by choosing the slow rate on the Gatts setting tab.

Now the great thing about our BLE sub-system, or BLESS, is that it has built-in timer control when the device is to be active. These timers can wake the part from its Deep-sleep mode. The stack offers an API to request BLESS to enter sleep mode, so it is really easy to figure out when to put the whole device to sleep.

In your main loop you just call the CyBle_EnterLPM() API with either the sleep or deep sleep argument. It will return the power status of the BLESS. If it has gone to sleep you can put the rest of the part to sleep.

Remember to toggle the pin as you go into and come out of sleep.
After you build and program the target you can see the blue LED flicker at a rate between 1 and 10 seconds.

With just a few lines of code we have greatly reduced the power consumption of the application.
Try changing the power setting and the advertising rates in the GAP settings dialog Ð you can make them really long, but then you'll start creating timeouts, and that's no good. Just remember, the less your application has to do, the better its power profile will be.
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