PSoC 4 BLE – Low Power Modes

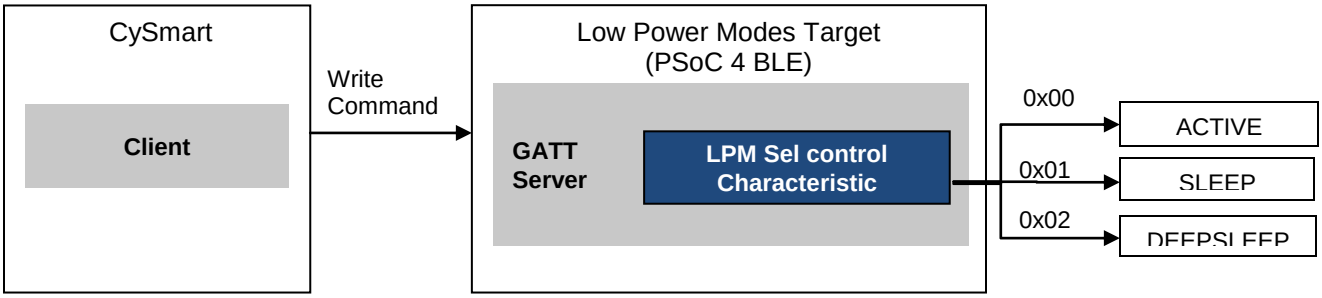
Objective

This example demonstrates the use of the BLE Component to design an application with Low Power Modes.

Overview

This example uses the BLE Pioneer Kit to design a simple application using the standard services defined by the Bluetooth SIG, which can be allowed to put the chip into Low power modes using BLE. In this example, BLE commands are used to put the chip into various low power modes, while still maintaining connection. The project also shows how the BLESS can be put into deep sleep and how the system itself can go to low power modes as well.

Figure 1: PSoC 4 BLE Low Power Modes



Requirements

Design Tool: PSoC Creator 3.1 CP1, CySmart 1.0

Programming Language: C (GCC 4.8.4 – included with PSoC Creator)

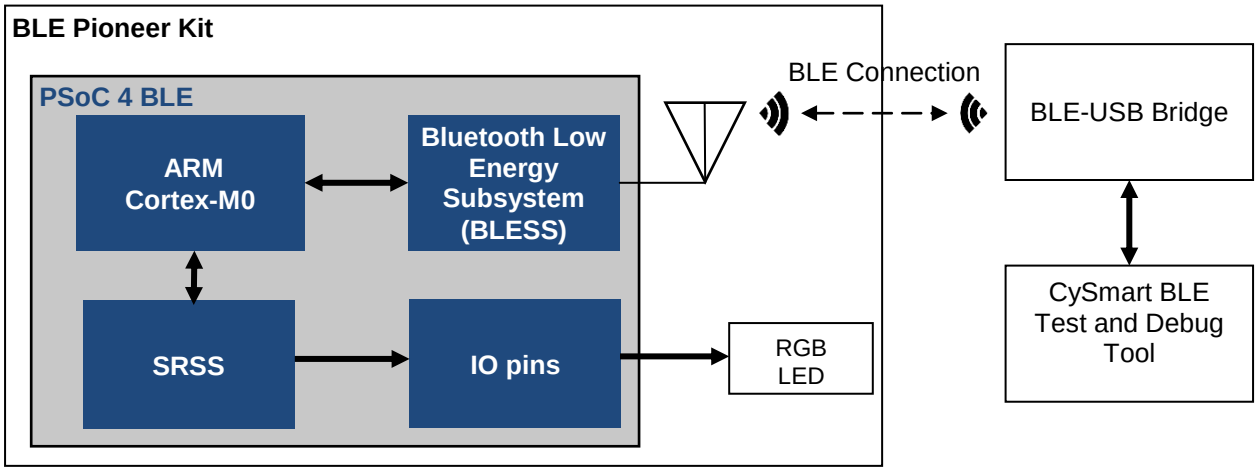
Associated Devices: All PSoC 4 BLE devices

Required Hardware: CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit

Hardware Setup

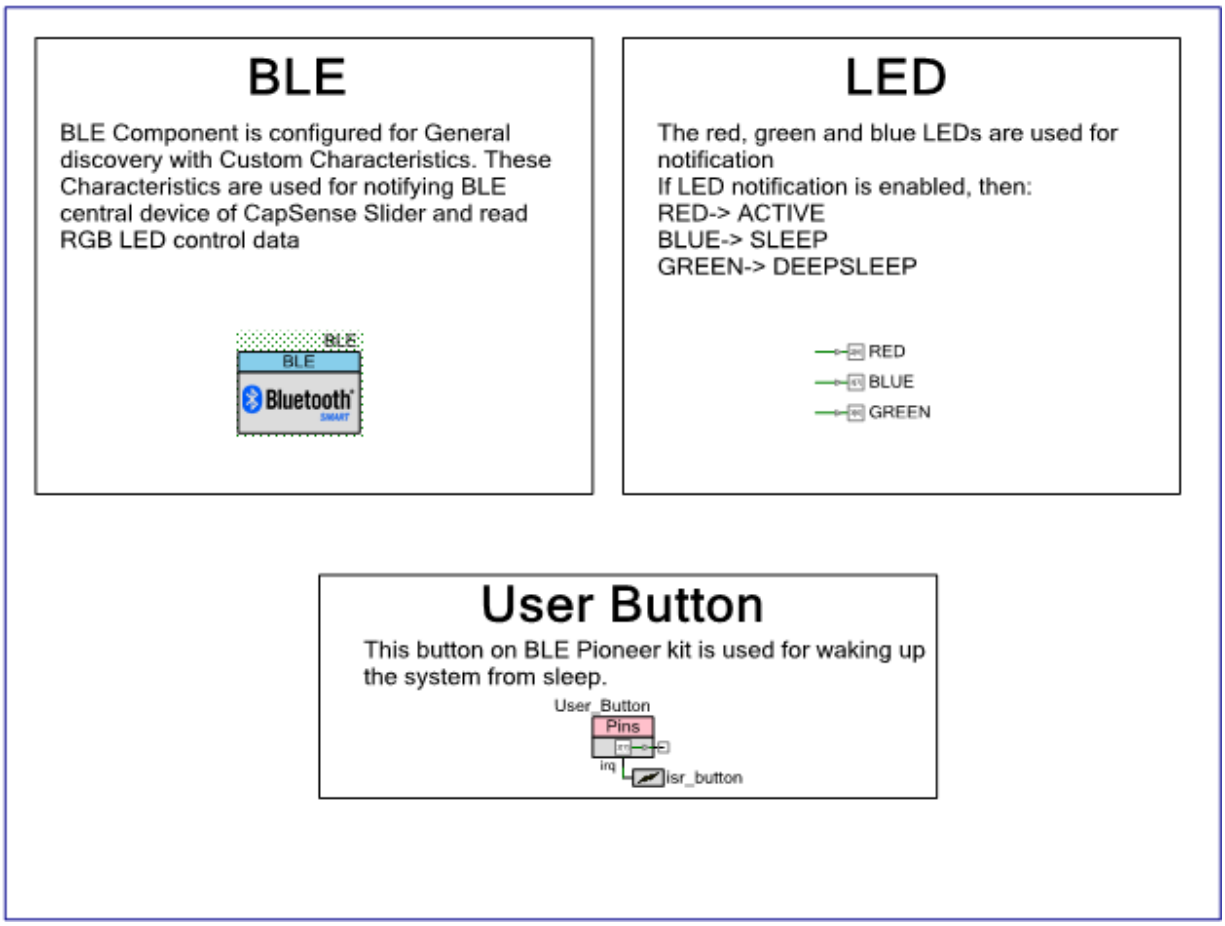
The BLE Pioneer Kit has all of the necessary hardware required for this lab. In this setup, the Red, Green and Blue LEDs are connected to pin P2.6, P3.6 and P3.7 of the PSoC 4 BLE device. A user switch is also connected to P2.7 of the device.

Figure 2: Block Diagram



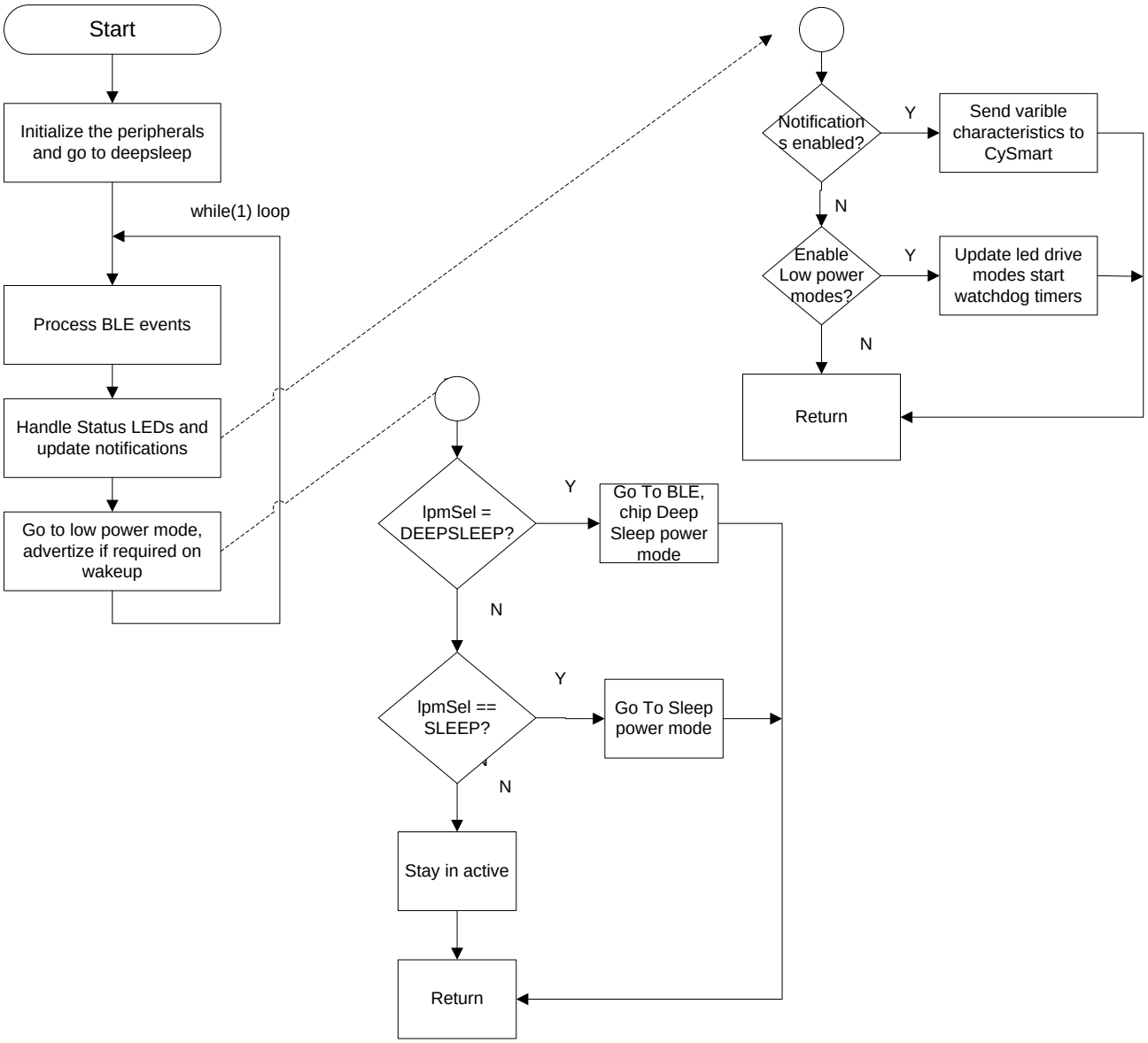
PSoC Creator Schematic

Figure 3. PSoC Creator Schematic



Firmware Flow

Figure 4. Firmware Flow



1. **main() function**: This is the central function which performs the initialization of the BLE Stack and PWM for the LED control. It then executes the necessary routines to process the BLE events and maintain the connection.
   In the initial section of the *main()* function, the API function *CyBle_Start(StackEventHandler)* is called to start the BLE Component and register a callback to the Stack event handler. Note that the callback function can have any name – in this project, we used StackEventHandler. Once the system is initialized, *main()* continuously operates in a *while(1)* loop executing *CyBle_ProcessEvents()*. This function processes the events received by the BLE Stack and enables the application layer to use them and take the appropriate action

2. **HandleLowPowerMode () function**: This function handles the low power input from use over BLE and selects the low power modes accordingly when BLE is idle. If LED notification is enabled, then the tricolor LED is setup before chip goes to the desired power mode after setting up the BLE hardware power modes, the LED notification colors are shown in Table 1.

Table 1: LED state vs power mode

RGB LED colour	Power mode
RED	Active
BLUE	Sleep mode
GREEN	Deep Sleep mode

Note: LED pin are toggled only on LED notifications enabled. Do not use enable LED notifications while measuring current in low power modes.

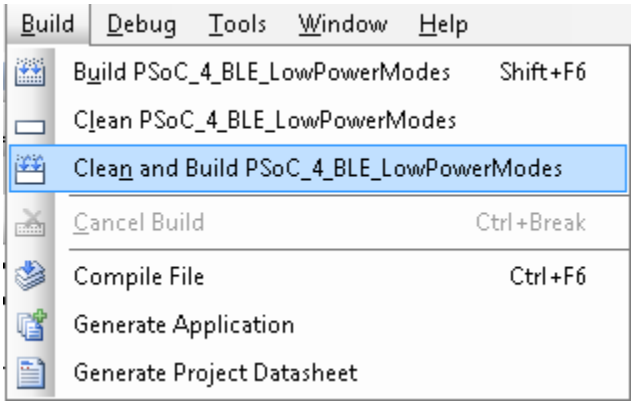
Build and Program

This section shows how to build the project and program the PSoC 4 BLE device. If you are using a development kit with a built-in programmer (BLE Pioneer Kit, for example), connect the BLE Pioneer Baseboard to your computer using the USB Standard-A to Mini-B cable. For other kits, refer to the kit user guide.

If you are developing on your own hardware, you need a hardware debugger, for example, a Cypress CY8CKIT-002 MiniProg3.

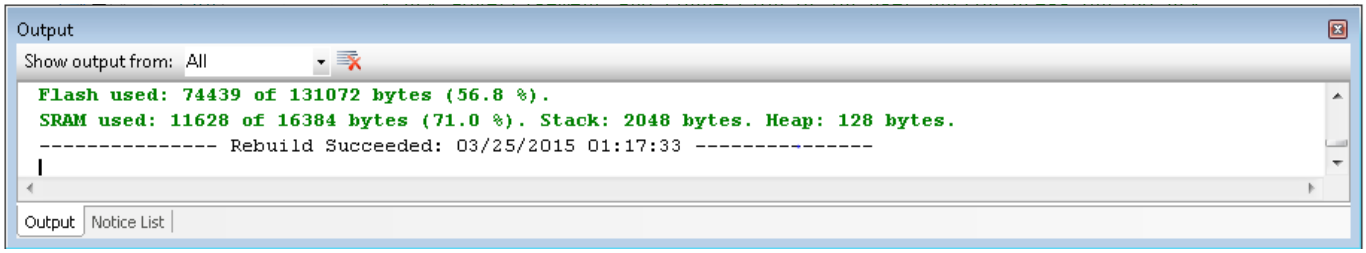
1. On PSoC Creator, select **Build** > **Clean and Build PSoC 4BLE_LowPowerModes**, as shown in Figure 5.

Figure 5. Build Project



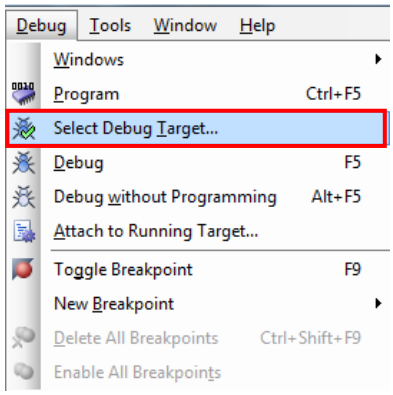
2. On a successful build, the total flash and SRAM usage is reported as shown in Figure 6. Please note that the numbers are indicative and may not match this project.

Figure 6. Build Succeeded



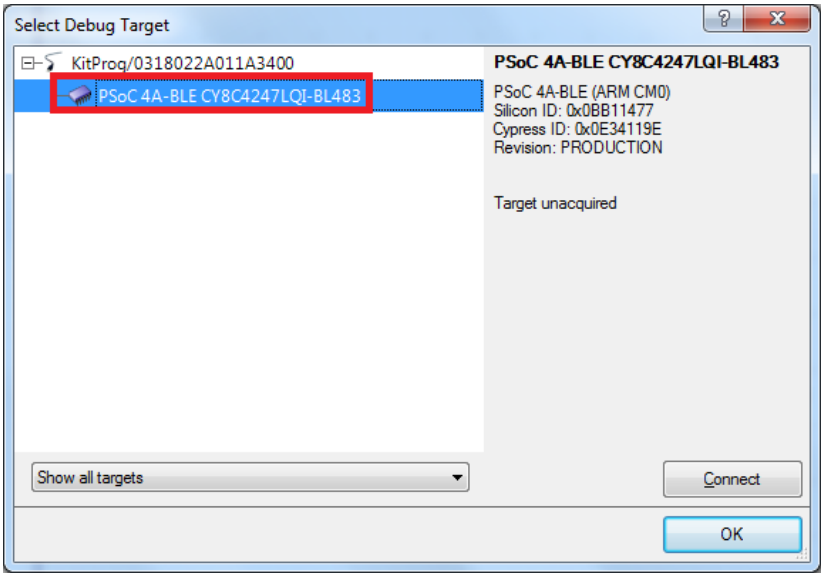
3. Select **Debug** > **Select Debug Target**, as shown in Figure 7.

Figure 7. Selecting Debug Target



4. In the **Select Debug Target** dialog box, click **Port Acquire**, and then click **Connect** as shown in Figure 8. Click **OK** to close the dialog box.

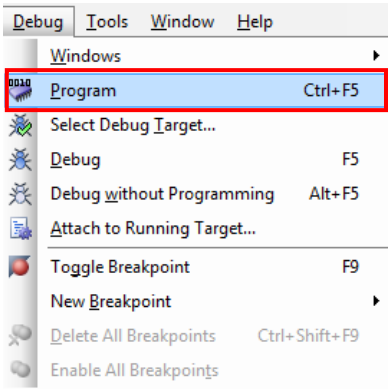
Figure 8. Connecting to a Device



If you are using your own hardware, make sure the Port Setting configuration under Select Debug Target window for your programming hardware is configured as per your setup.

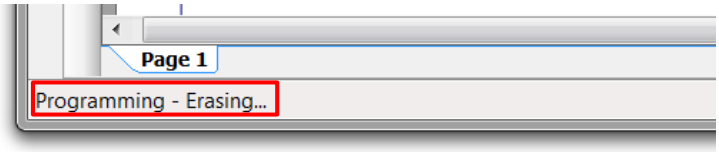
5. Select **Debug** > **Program** to program the device with the project, as shown in Figure 9.

Figure 9. Programming the Device



You can view the programming status on the PSoC Creator status bar (lower-left corner of the window), as shown in Figure 10.

Figure 10. Programming Status

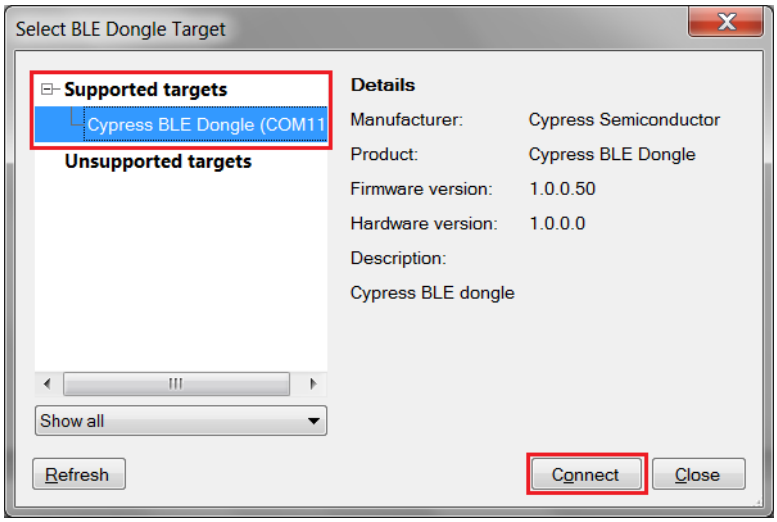


Testing

Testing with the CySmart BLE Test and Debug Utility for Windows PC:

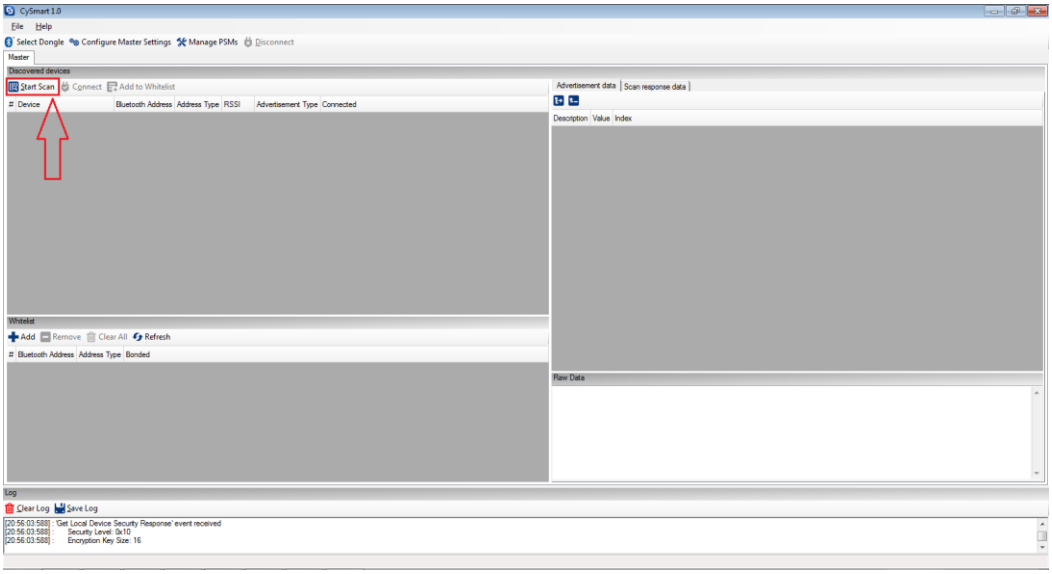
1. Plug the BLE-USB Bridge (included with the BLE Pioneer Kit) in your computer's USB port.
2. On your computer, launch **CySmart 1.0**. It is located in the **All Programs -> Cypress -> CySmart** folder in the Windows start menu. The tool opens up and asks you to **Select BLE Dongle Target**. Select the **Cypress BLE Dongle (COMxx)** and click **Connect**, as shown in Figure 11.

Figure 11: CySmart: Select BLE Dongle Target



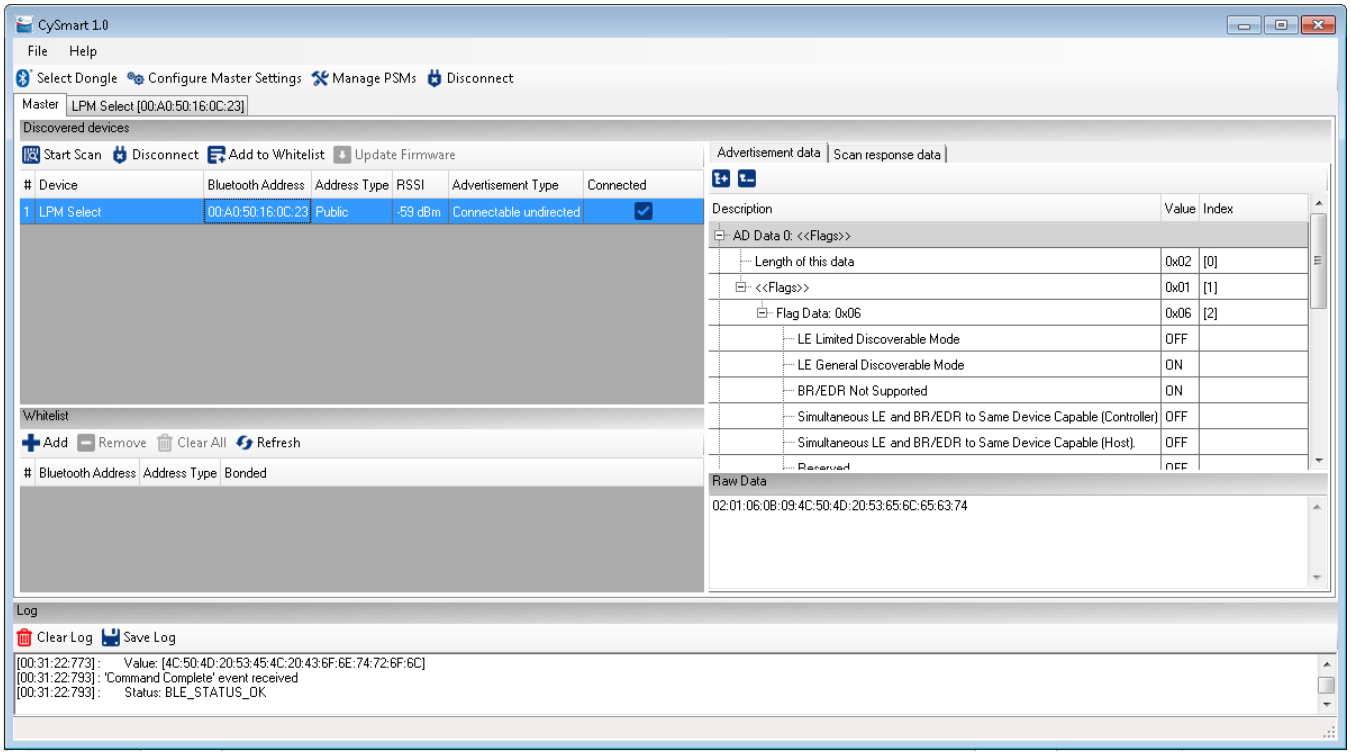
3. When the BLE-USB Bridge is connected, click on **Start Scan** to find your BLE device. Press the switch SW2 on the BLE baseboard to see the device. See Figure 12.

Figure 12: Finding a BLE Device



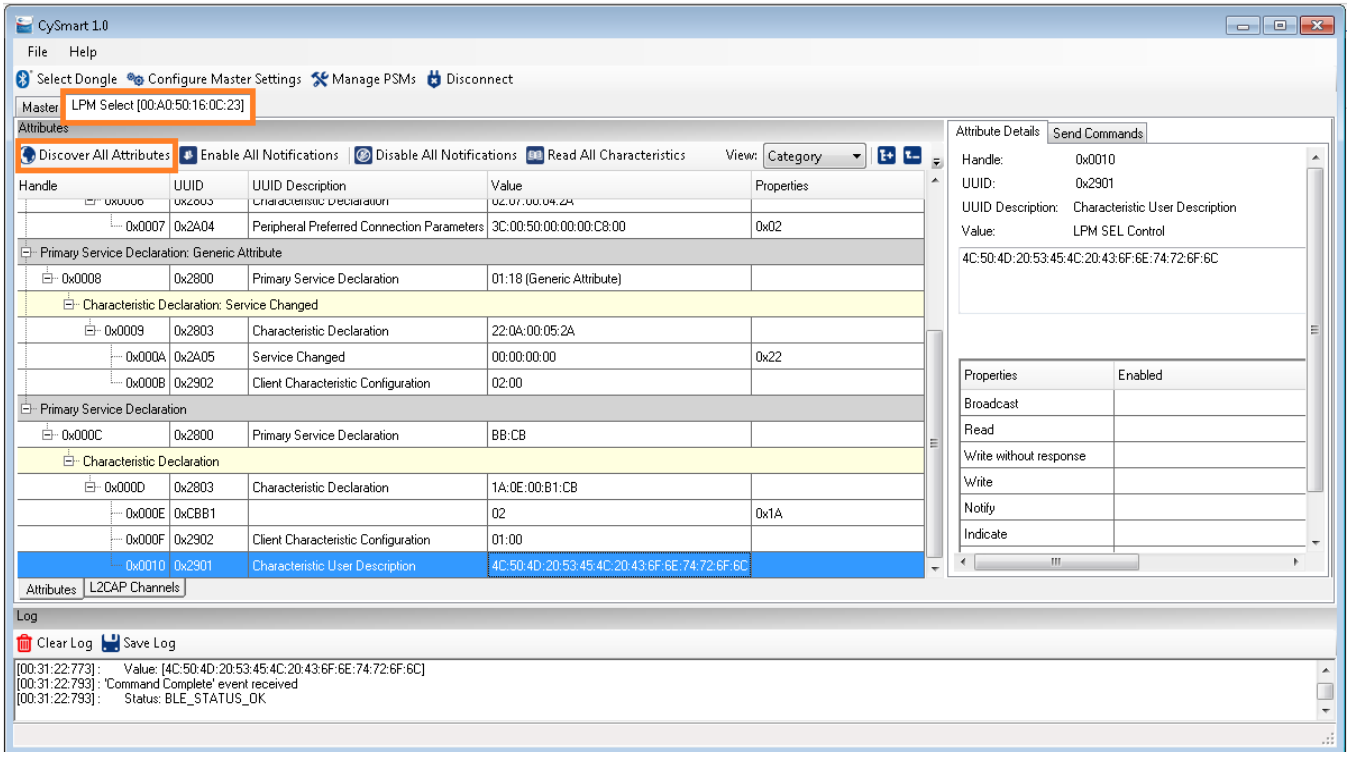
4. The scanning stops automatically once all the nearby devices are known. The tool lists all the nearby devices in the Discovered devices section.
5. Click on your device name to see the Advertisement data and Scan response data packets on the right. See Figure 13.

Figure 13: Checking Discovery Details of a Connected BLE Device



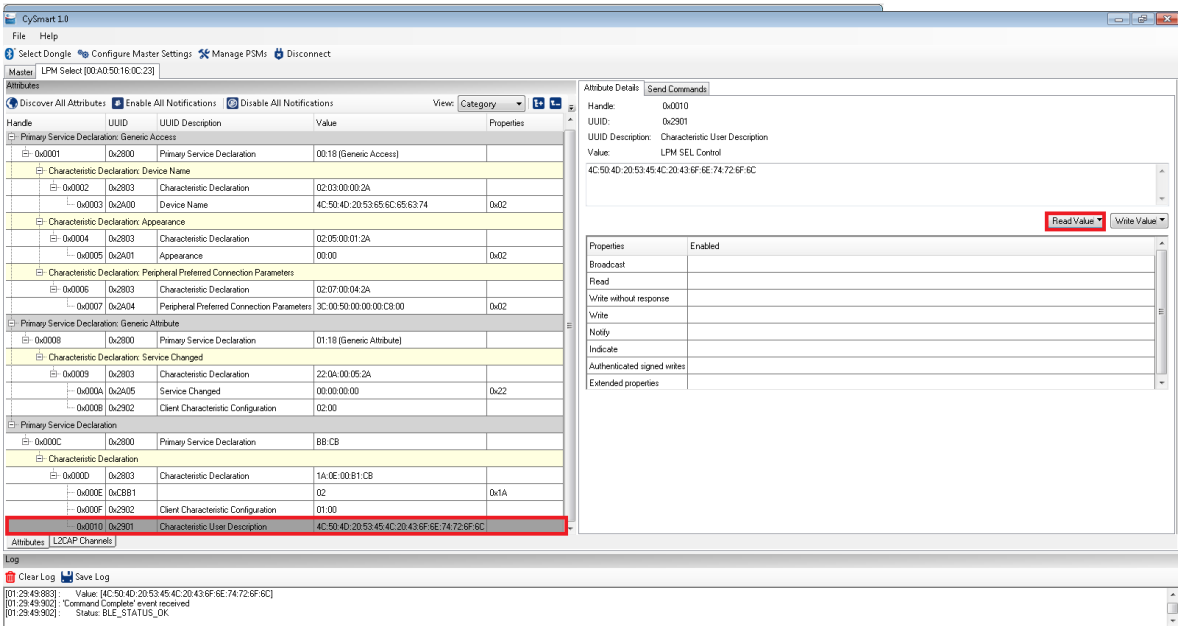
6. Click **Connect** as seen in Figure 13 to connect to the device.
7. The tool will now open a separate tab for the device. Click **Discover All Attributes** to list all the Attributes in the device, with their respective UUIDs and descriptions. See Figure 14.

Figure 14: Discovering Attributes of a Connected BLE Device



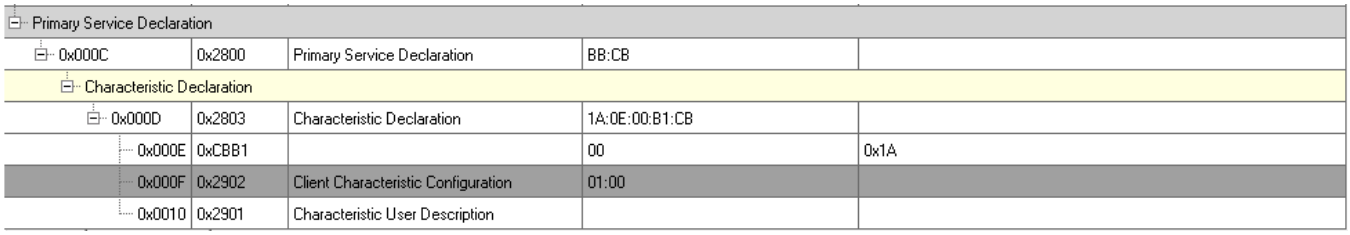
8. Click on any row in the list of Attributes to see its details on the right. To read an Attribute's value, click **Read Value** on the right as shown in Figure 15.

Figure 15: Reading Attribute Value



9. Locate the **Client Characteristic Configuration** Attribute for the **LPM SEL control service**. On the right, write a value of **1** to see the notifications. See Figure 16.

Figure 16: Writing Attribute Value



10. Write a value of **0, 1, and 2** to the line with **UUID CBB1 as shown in Fig19.** Observe that the tricolor LED changes color to **red, blue and green respectively.**
11. Build the project with the following definition commented and program again to remove LED notification while conserving power in BLEApplications.h in Figure 20:

Figure 17: Writing Attribute Value

```
54
55  /* 'ENABLE_LED_NOTIFICATION' pre-processor directive enables the LED
56   * handling in the firmware, ensuring that different LED indications
57   * are provided in different power modes during project usage.
58   * To disable, comment the following #define.
59   * If disabled, prevent usage of the project with coin cell */
60  #define ENABLE_LED_NOTIFICATION
```

Refer to the kit guide for more on how to measure current on the BLE Pioneer kit

12. Some Current numbers measured with this project using a Keithley 2000 multimeter(medium/slow rate) are provided below:
    With LED notification: Start Current (green light):

Table 2: Current vs power mode selection

Power Mode(value)	Current measured(mA)
Active(0)	6.5
Sleep(1)	4.0
Deep Sleep(2)	.7

Without LED notification: Start current: 1.4 uA

Table 3: Current vs power mode selection

Power Mode(value)	Current measured(mA)
Active(0)	5.8
Sleep(1)	3.4
Deep Sleep(2)	.14

13. Changing the connection update parameters: We can change the connection update parameters to reduce data rate and thus power consumption. A sample is provided below, with slower data rate. Update the BLEApplications.h with the data below and repeat the testing steps above after pioneer kit is programmed with the new hexfile.

Figure 18: Default update connection values in project

```
90
91  /* Minimum connection interval = CONN_PARAM_UPDATE_MIN_CONN_INTERVAL * 1.25 ms*/
92  #define CONN_PARAM_UPDATE_MIN_CONN_INTERVAL 100
93  /* Maximum connection interval = CONN_PARAM_UPDATE_MAX_CONN_INTERVAL * 1.25 ms */
94  #define CONN_PARAM_UPDATE_MAX_CONN_INTERVAL 110
95  /* Slave latency = Number of connection events */
96  #define CONN_PARAM_UPDATE_SLAVE_LATENCY     0
97  /* Supervision timeout = CONN_PARAM_UPDATE_SUPRV_TIMEOUT * 10*/
98  #define CONN_PARAM_UPDATE_SUPRV_TIMEOUT     200
```

Figure 19: Updated update connection values in project

```
85  /* Connection Update Parameter values to modify connection interval. These values
86   * are sent as part of CyBle_L2capLeConnectionParamUpdateRequest() which requests
87   * Client to update the existing Connection Interval to new value. Increasing
88   * connection interval will reduce data rate but will also reduce power consumption.
89   * These numbers will influence power consumption */
90
91  /* Minimum connection interval = CONN_PARAM_UPDATE_MIN_CONN_INTERVAL * 1.25 ms*/
92  #define CONN_PARAM_UPDATE_MIN_CONN_INTERVAL 1000
93  /* Maximum connection interval = CONN_PARAM_UPDATE_MAX_CONN_INTERVAL * 1.25 ms */
94  #define CONN_PARAM_UPDATE_MAX_CONN_INTERVAL 1100
95  /* Slave latency = Number of connection events */
96  #define CONN_PARAM_UPDATE_SLAVE_LATENCY     0
97  /* Supervision timeout = CONN_PARAM_UPDATE_SUPRV_TIMEOUT * 10*/
98  #define CONN_PARAM_UPDATE_SUPRV_TIMEOUT     2000
```

A set of current readings with this configuration using the same setup are:
Without LED notification: Start current: 1.4 uA

Power Mode(value)	Current measured(mA)
Active	5.8
Sleep	3.35
Deep Sleep	.03

We can see that the Deep Sleep current has come down, but at the cost of slower data rate.

Related Documents

Table 4 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component / user module datasheets.

Table 4. Related Documents

Document	Title	Comment
AN91267	Getting Started with PSoC 4 BLE	Provides an introduction to PSoC 4 BLE device that integrates a Bluetooth Low Energy radio system along with programmable analog and digital resources.
AN91445	Antenna Design Guide	Provides guidelines on how to design an antenna for BLE applications.