PSoC 4 BLE Lab #2: Setup a BLE Connection

Description

This lab introduces you to the Bluetooth Low Energy feature of PSoC 4 BLE. It helps you create your first BLE application by implementing a BLE Standard Find-Me Profile.

Pre-Reading

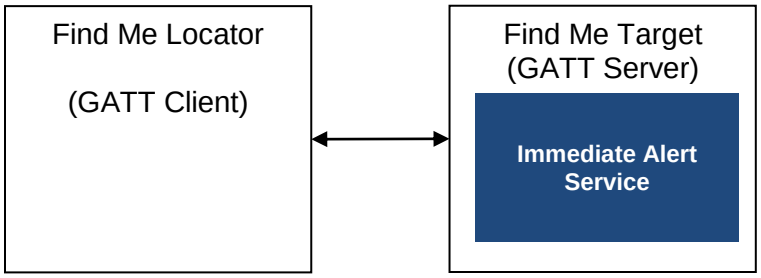
BLE Find Me Profile

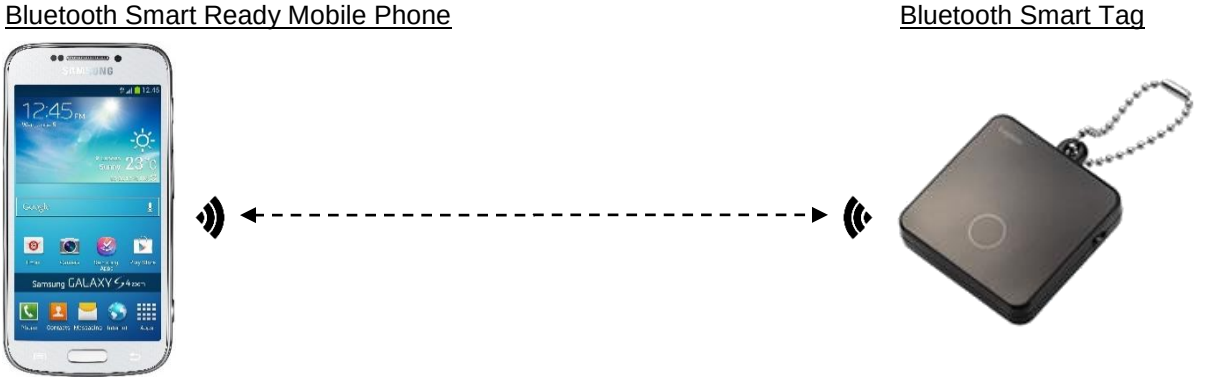
The BLE Find Me Profile defines how pressing a button on one BLE device causes an alerting signal on another BLE device. This can be used to find misplaced devices.

There are two BLE Profile roles that are defined by the Find Me Profile, as shown in Figure 1.

- The device that initiates the alerting signal (e.g. iPhone) is called the Find Me Locator. The Find Me Locator is a GATT Client.
- The device that receives the alerting message and triggers a user alert (e.g. blink an LED, drive a buzzer, drive a vibration motor, etc.) is called the Find Me Target (eg. the Tile device). The Find Me Target is a GATT Server running the Immediate Alert Service (IAS).

Figure 1: BLE Find Me Profile Roles





Profile	Find Me Locator			Profile	Find Me Target
GAP Role	Central	Scans for Services / Initiates Connection / Writes Alert Level	Advertises Services / Accepts Connection / Executes Alert Level	GAP Role	Peripheral
GATT Role	Client			GATT Role	Server

Immediate Alert Service (IAS)

This Service allows a GATT Client to cause the GATT Server to issue alerts. IAS defines a single mandatory Characteristic called Alert Level. The GATT Client can write one of three possible values to this Alert Level Characteristic. The Server application defines its behavior based on these Alert Levels.

- If the Client writes “No Alert” (0x00), no alerting will be done on this GATT Server.
- If the Client write “Mild Alert” (0x01), the GATT Server will alert.
- If the Client writes “High Alert” (0x02), the GATT Server will alert in the strongest possible way.

Connection Establishment

The IAS specification recommends that the GATT Server advertise using the parameters in Table 1. The interval values in the first row are intended for a fast connection during the first 30 seconds; if a connection is not established within that time, the interval values in the second row are intended to reduce power consumption for devices that continue to advertise.

Table 1: Recommended Advertising Interval Values

Advertising Duration	Parameter	Value
First 30 seconds (fast connection)	Advertising Interval	20 ms to 30 ms
After 30 seconds (reduced power)	Advertising Interval	1 s to 2.5 s

PSoC 4 BLE Lab #2: Setup a BLE Connection

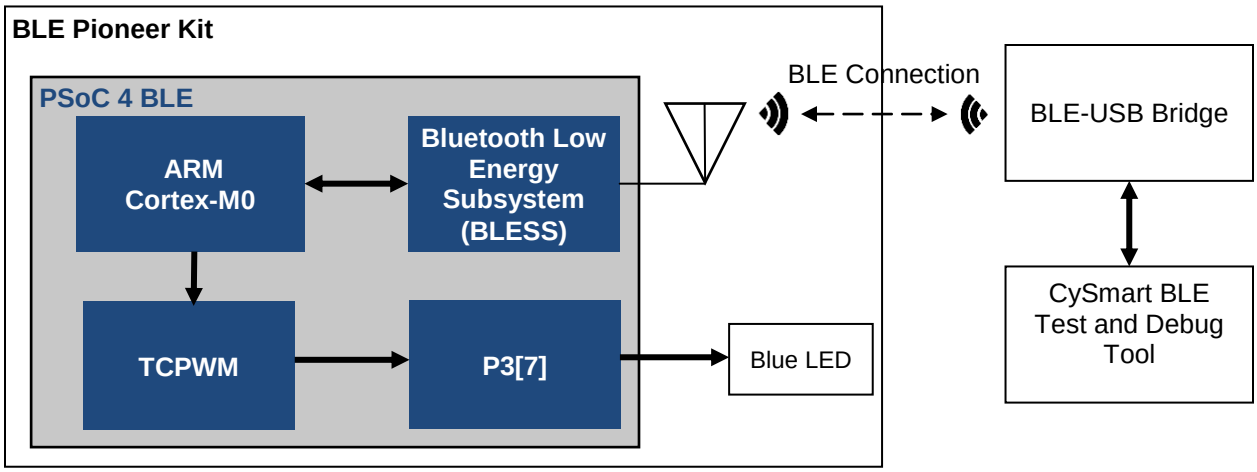
Objectives

1. Learn how to use the BLE Component
2. Implement a standard BLE Find Me Profile with the Immediate Alert Service (IAS)
3. Learn how to use the CySmart BLE Test and Debug Tool to debug BLE designs

Requirements	Details
Hardware	BLE Pioneer Kit (CY8CKIT-042-BLE)
Software	PSoC Creator 3.1 (or newer)
	CySmart 1.0

Block Diagram

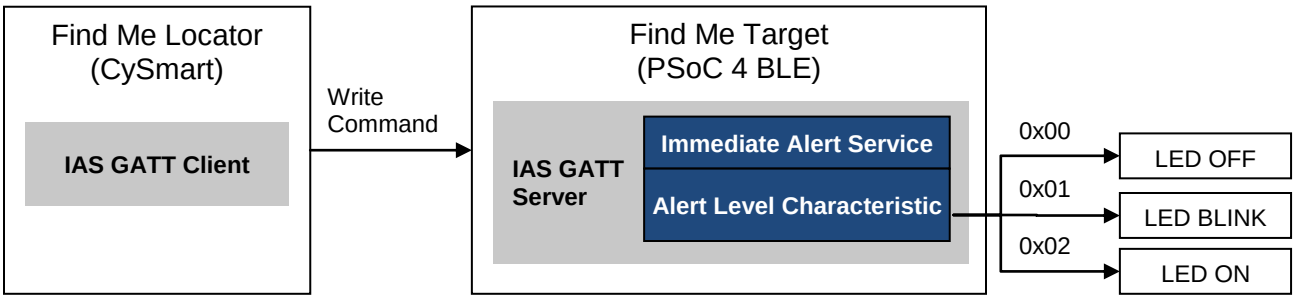
Figure #2: Lab 2 Block Diagram



Theory

The BLE Pioneer Kit acts as the GATT Server. It is detected by the CySmart BLE Test and Debug Tool (GATT Client). The GATT Server contains the Immediate Alert Service with Alert Level Characteristic.

Figure 3: BLE Find Me Lab Description



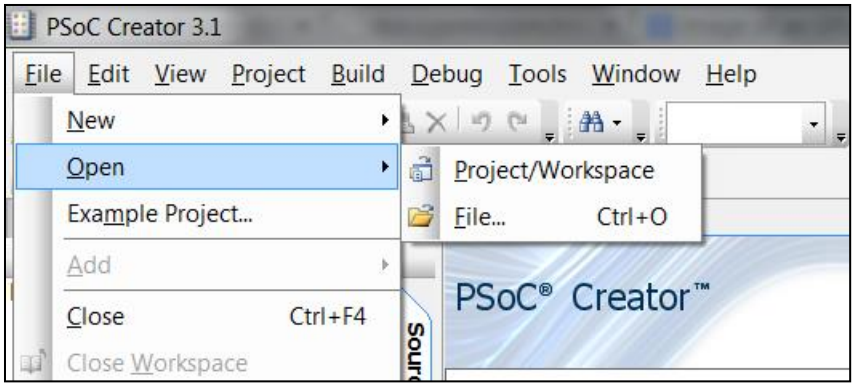
Procedure

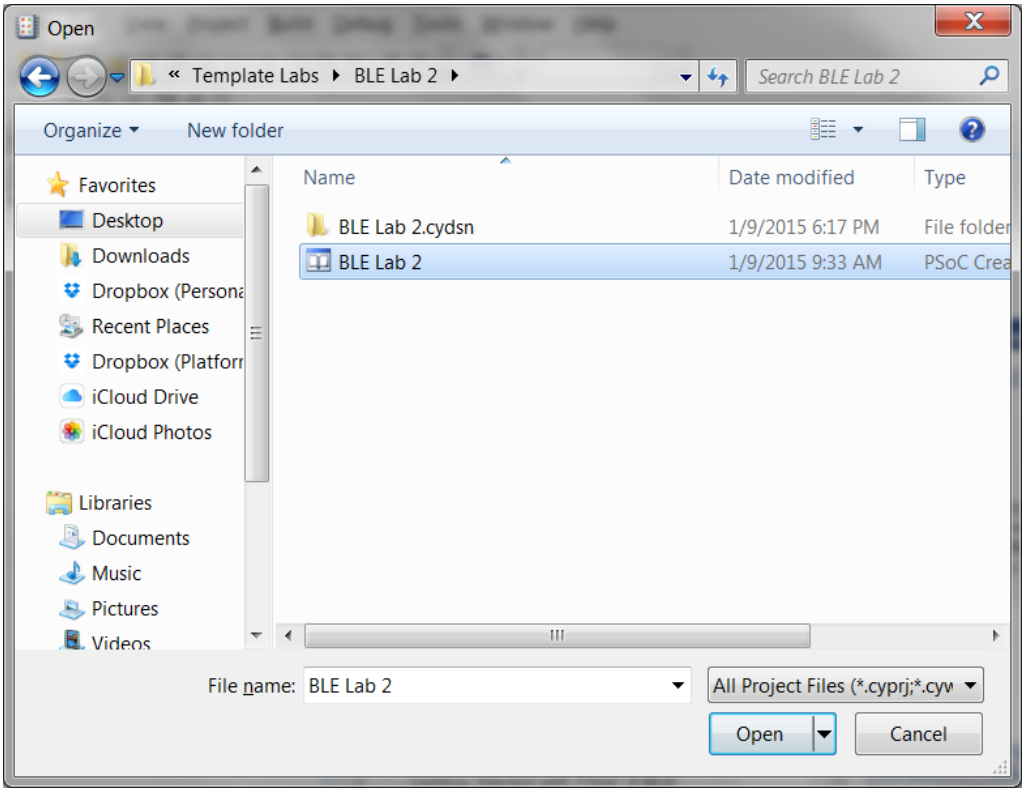
Start this lab from the template project that is provided. The template project already has the PWM Component placed and configured; you only need to configure the BLE Component as instructed.

Configure Schematic

1. Open the template project named **BLE Lab 2** by clicking the menu item **File > Open Project/Workspace** as shown in Figure 4.

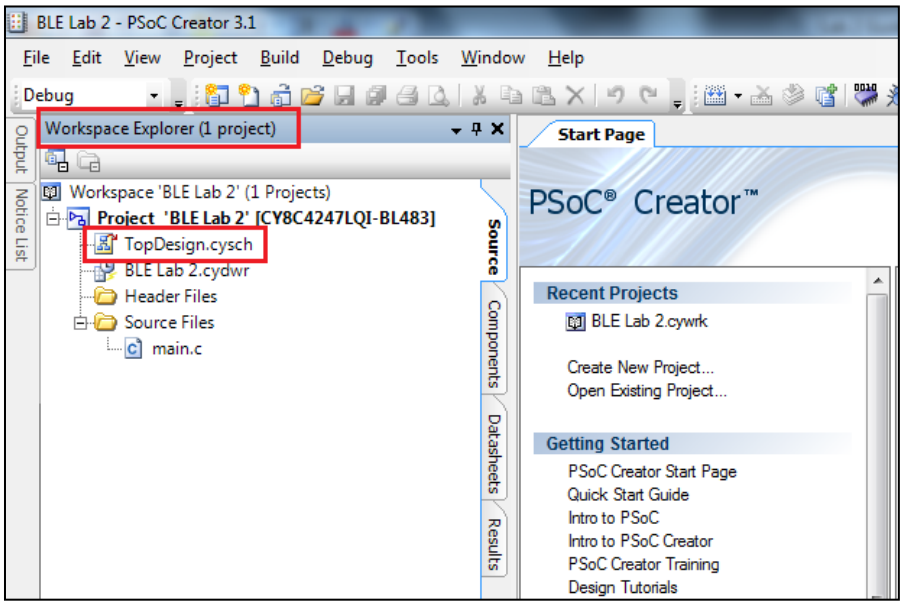
Figure 4: Open Project





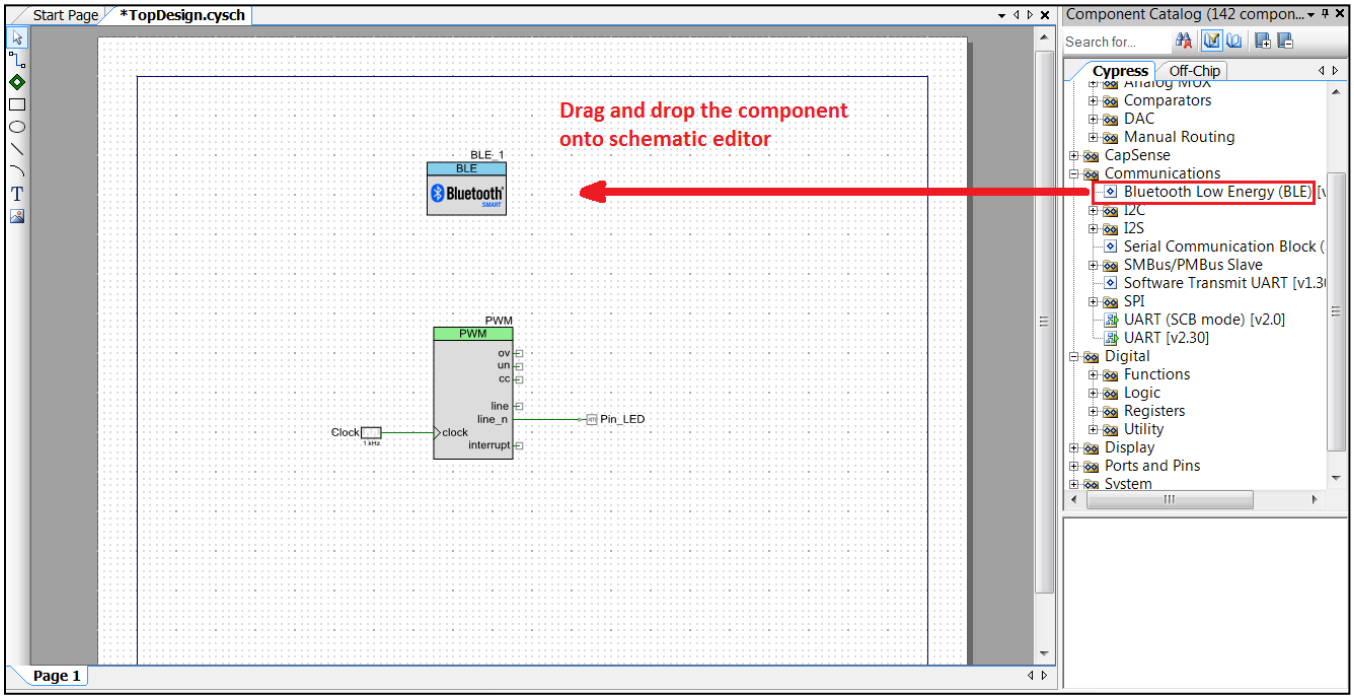
2. If not already selected, double-click **TopDesign.cysch** from the **Workspace Explorer** to open the schematic editor, as shown in Figure 5.

Figure 5: Opening Schematic Editor in the Project



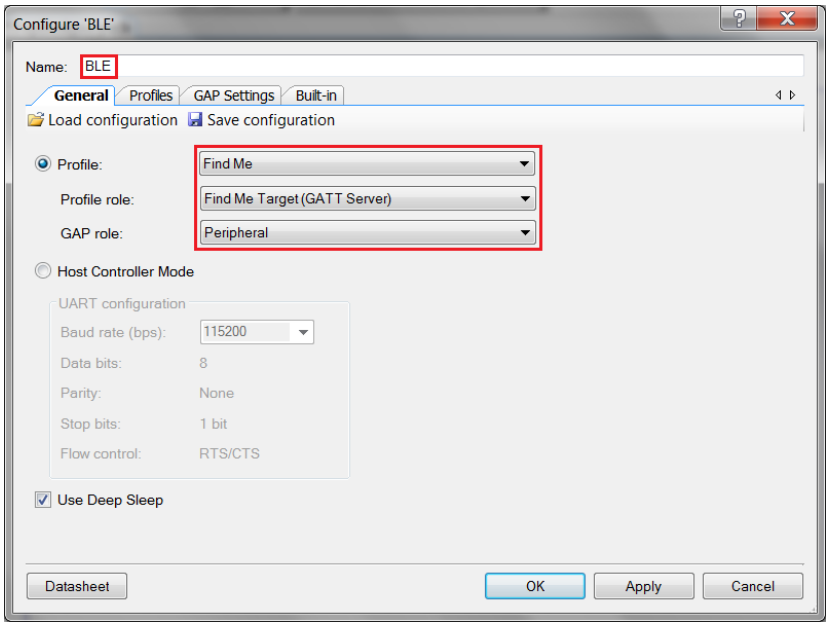
3. From the **Component Catalog** window on the right, locate the **Bluetooth Low Energy** Component under the **Communications** category. Drag and drop this Component to the schematic editor. See Figure 6.

Figure 6: Placing the BLE Component on the Schematic Editor



4. Double-click the Component to open its Component Configuration Tool. You can refer to the Component datasheet to learn more about the configuration parameters.
5. **General Tab** – Set the **Profile** to **Find Me**. The **Profile Role** is automatically set to **Find Me Target (GATT Server)** and the **GAP role** is set to **Peripheral**. See Figure 7.

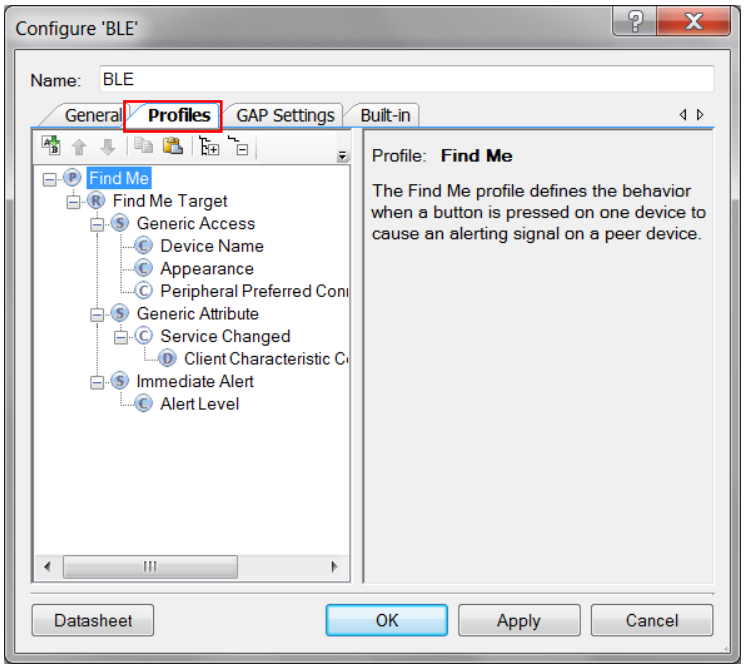
Figure 7: BLE Component Configuration – General Tab



Note: You can choose any name for the BLE Component. In this example, we've used **BLE**. See Figure 7.

6. **Profiles Tab** - The Profiles tab by default configures the GATT Server with an Immediate Alert Service that consists of an Alert Level Characteristic as required by the IAS specification. No changes are required here. See Figure 8.

a. Figure 8: BLE Component Configuration – Profiles Tab



7. **GAP Settings Tab** - This tab defines the GAP connection parameters for advertisement, discovery, scan response, device address and security settings. To learn more about these parameters, refer to the Bluetooth Component datasheet.

   7.1. **General**

   These settings are shown in Figure 9.

   a. Provide a unique BLE **Device address** for your device. This must be unique so that the GATT Client can differentiate between your device and another device. To automatically generate a unique address, check **Silicon generated "Company assigned" part of device address**.

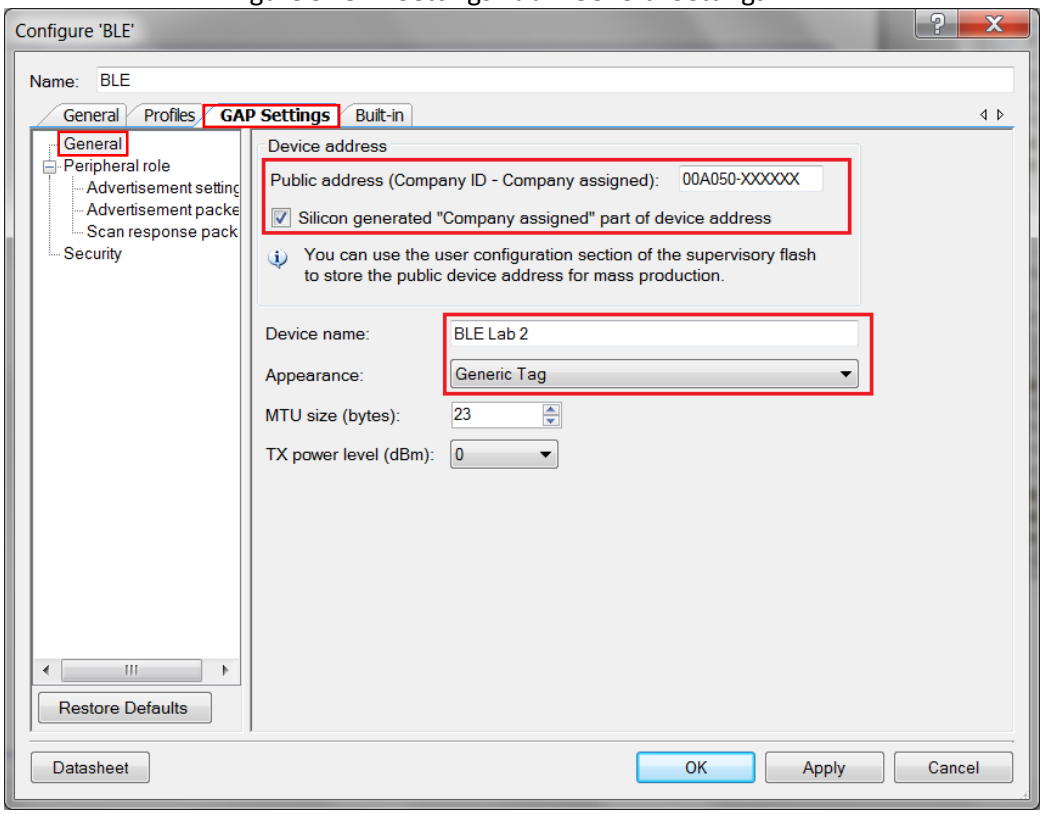
   b. Give your device a name. The **Device name** shows up on the GATT Client when it scans for your device.

   c. Set the device **Appearance** to an appropriate selection that represents your design. The appearance configuration will show up on a GATT Client when it scans for your device. This is just a string representing how your device looks, and does not affect the functionality of your device.

   d. Keep the **Maximum Transmission Unit (MTU)** size for your device at **23**. MTU determines the maximum size of a BLE packet. Its value can range from 23 to 512 per the BLE specification. Increasing the MTU size results in increased SRAM consumption as larger buffers are required to store the packet.

   e. Leave the **TX power level (dBm)** at the default value of **0**.

Figure 9: GAP Settings Tab – General Settings



7.2. Peripheral Role -> Advertisement Settings

This section configures the advertisement settings for the GAP Peripheral. Configure these parameters as described below. See Figure 10. To learn more about these parameters, refer to the Bluetooth Component datasheet.

a. **Discovery mode**: This parameter defines how you GATT Server can be discovered by other devices. For this lab session, a generally discoverable device will work. Select **General**.

b. **Advertisement type**: BLE devices have the ability to advertise their functionality and status information. The Advertisement type parameter defines whether your device transmits directed or undirected advertisement, and whether it is connectable, scannable, or non-connectable. We need **Connectable undirected advertising** for our GAP Peripheral.

c. **Filter policy**: This parameter defines whether scan requests and connection requests can come from any GATT Client or from a known "white list" only (a list of pre-defined BLE devices from which the GATT Server can accept requests). We are not defining a white list now, so select **Scan request: Any | Connect request: Any**.

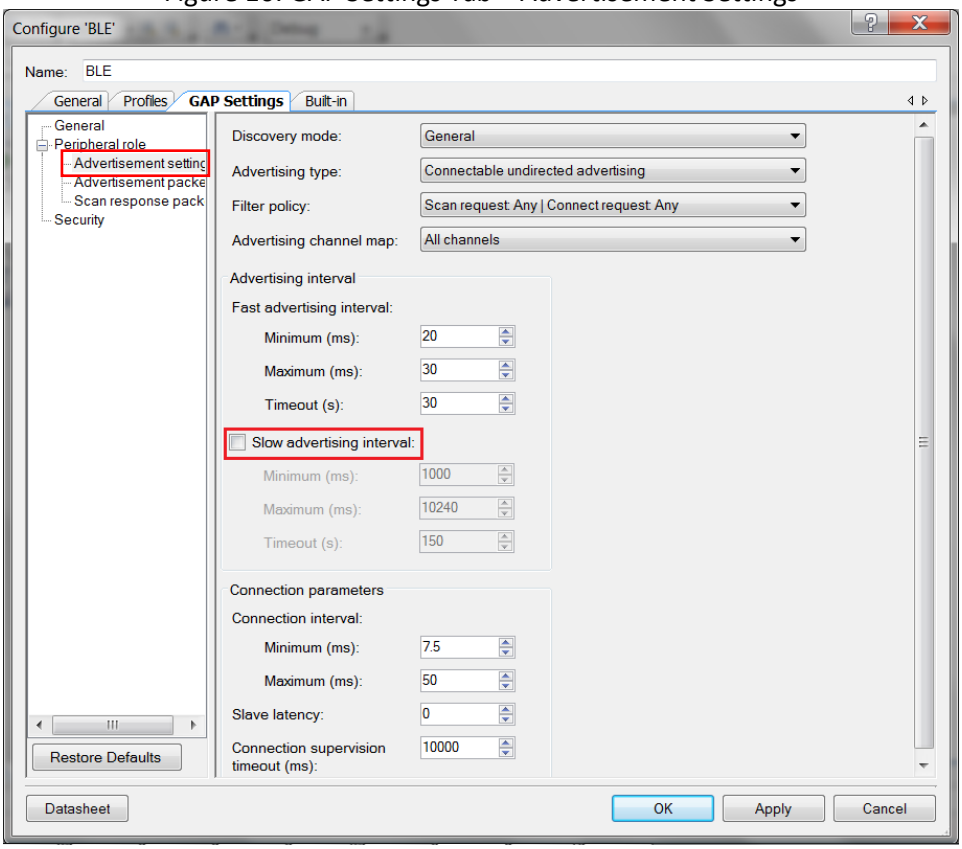
d. **Advertising channel map**: Defines which channels to advertise on. For this lab, we will advertise on **All channels**.

e. **Fast advertising interval**: Select **20** for **Minimum (ms)** and **30** for **Maximum (ms)**. The **Timeout (s)** should be **30**. Once this timeout has expired without a connection request, the device stops advertising.

f. **Slow advertising interval**: Uncheck this setting.

g. **Connection parameters**: Leave them at default.

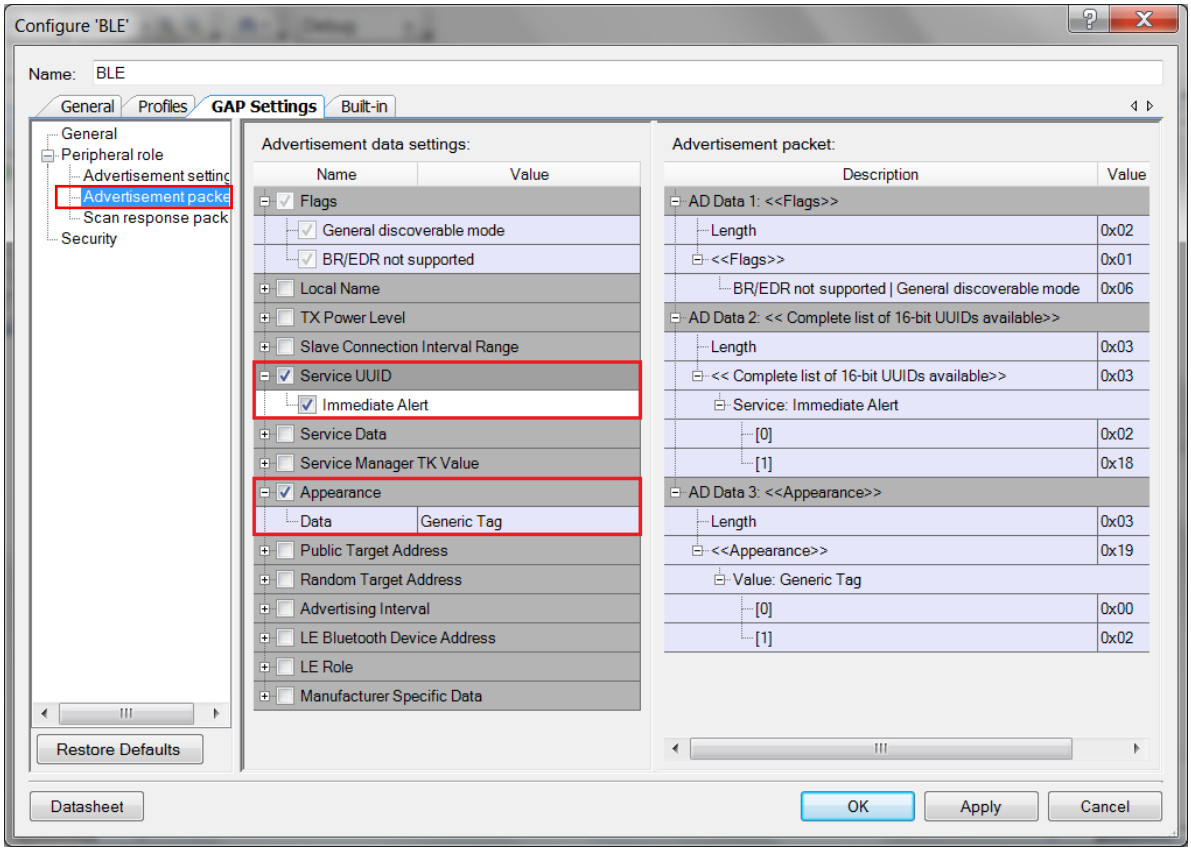
Figure 10: GAP Settings Tab – Advertisement Settings



7.3. Peripheral Role -> Advertisement Packet

The center panel shows the various details you can send as part of the advertisement packet. On the right is the actual packet sent by the device. For our lab session, we send the **Flags** (always present), the **Service UUID** (Universally Unique Identifier) for **Immediate Alert**, and the **Appearance**. See Figure 11.

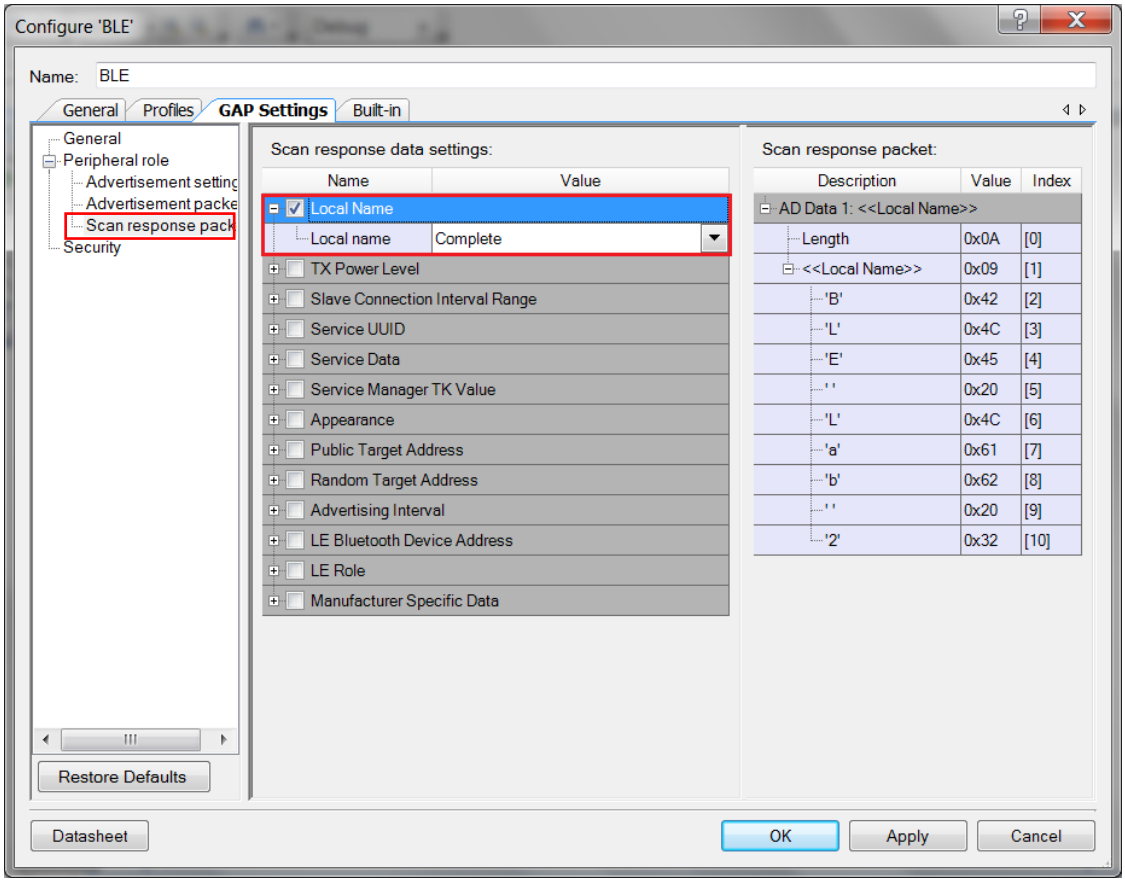
Figure 11: GAP Settings Tab - Advertisement Packet



7.4. **Peripheral Role -> Scan Response Packet**

This data is sent when the GATT Server responds to the GATT Client's scan requests. It provides additional details beyond what is sent in the advertisement packet. Send the **Local Name** as part of the Scan Response Packet. See Figure 12.

Figure 12: GAP Settings Tab - Scan Response Packet



7.5. **Security**

Finally, configure the BLE security settings. Set the parameters as shown in Figure 13. To learn more about these parameters, refer to the Bluetooth Component datasheet.

a. **Security mode**: Determines which security mode to implement. We use **Mode 1** security.

b. **Security level**: Based on the security mode, the security levels are defined. Select **No Security (No Authentication, No Encryption)**.

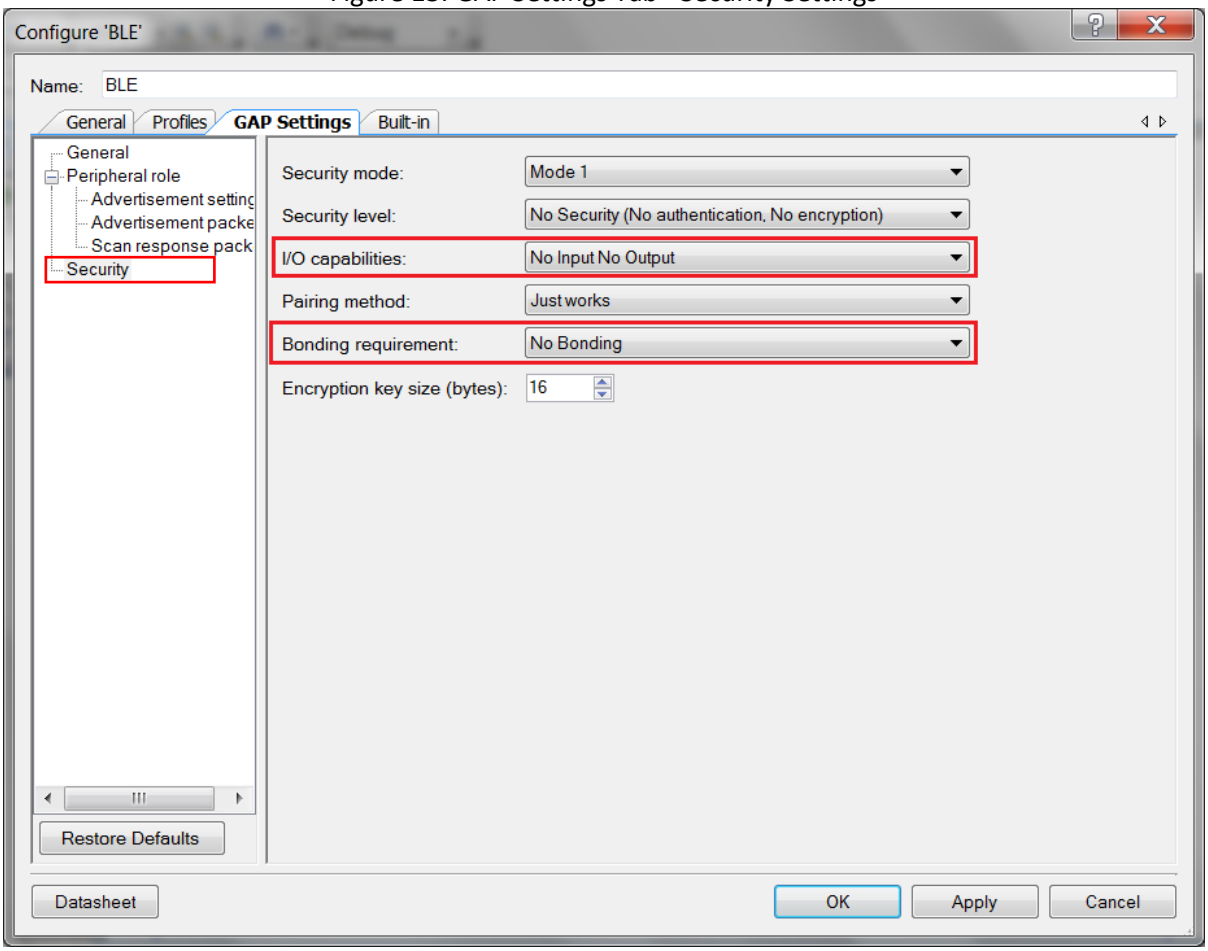
c. **I/O Capabilities**: Our device right now does not have any input or output capabilities and so we will set this to **No Input No Output**.

d. **Pairing method**: Pairing involves authenticating the identity of two BLE devices by means of exchanging pairing keys. The Pairing method parameter determines the method to generate the pairing keys. Since we have no I/O capabilities and there is no third device to mediate the connection, the pairing method is **Just works**.

e. **Bonding requirement**: Determines whether the keys generated during pairing are stored in the device, for speedier connections in the future. Set this to **No Bonding**.

f. **Encryption key size (bytes)**: Determines the size of encryption keys while pairing. Leave this parameter to the default value of **16**.

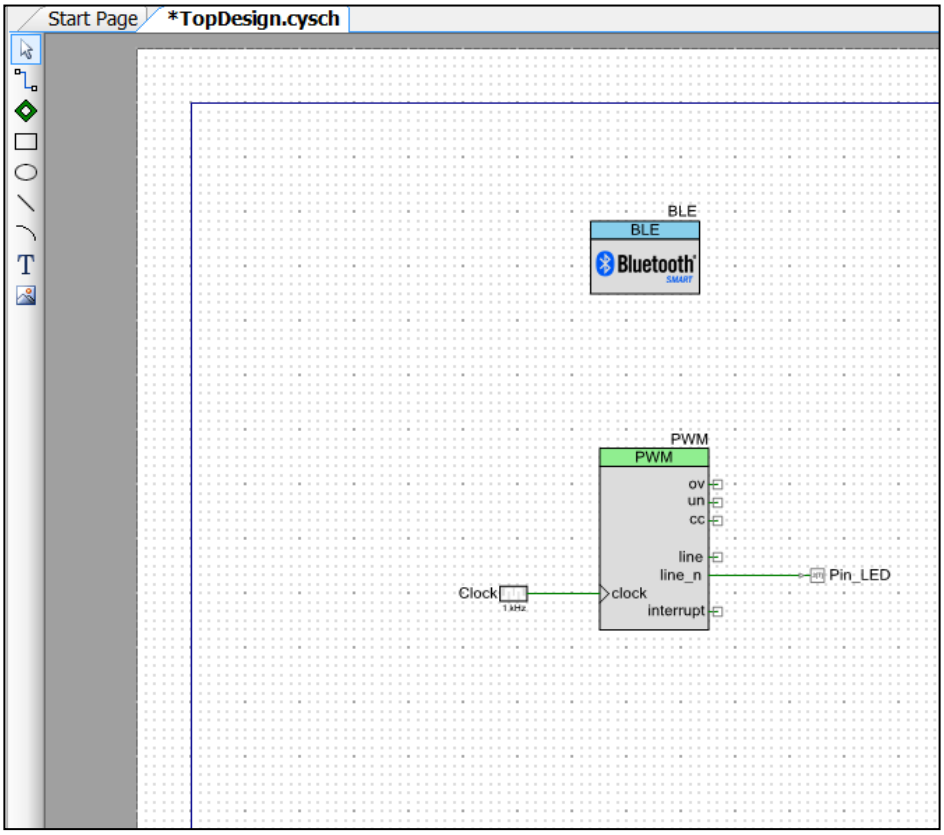
Figure 13: GAP Settings Tab - Security Settings



8. The Component configuration for a standard Find Me profile is now complete! Click **OK**.

9. The PWM Component has already been configured for you as a part of the project template.
10. Your schematic should look as shown in Figure 14.

Figure 14: Completed Schematic View



11. It is now time to build your project. Click the menu item **Build** -> **Build BLE Lab 2**. This starts the project build. All Component source code is automatically generated.

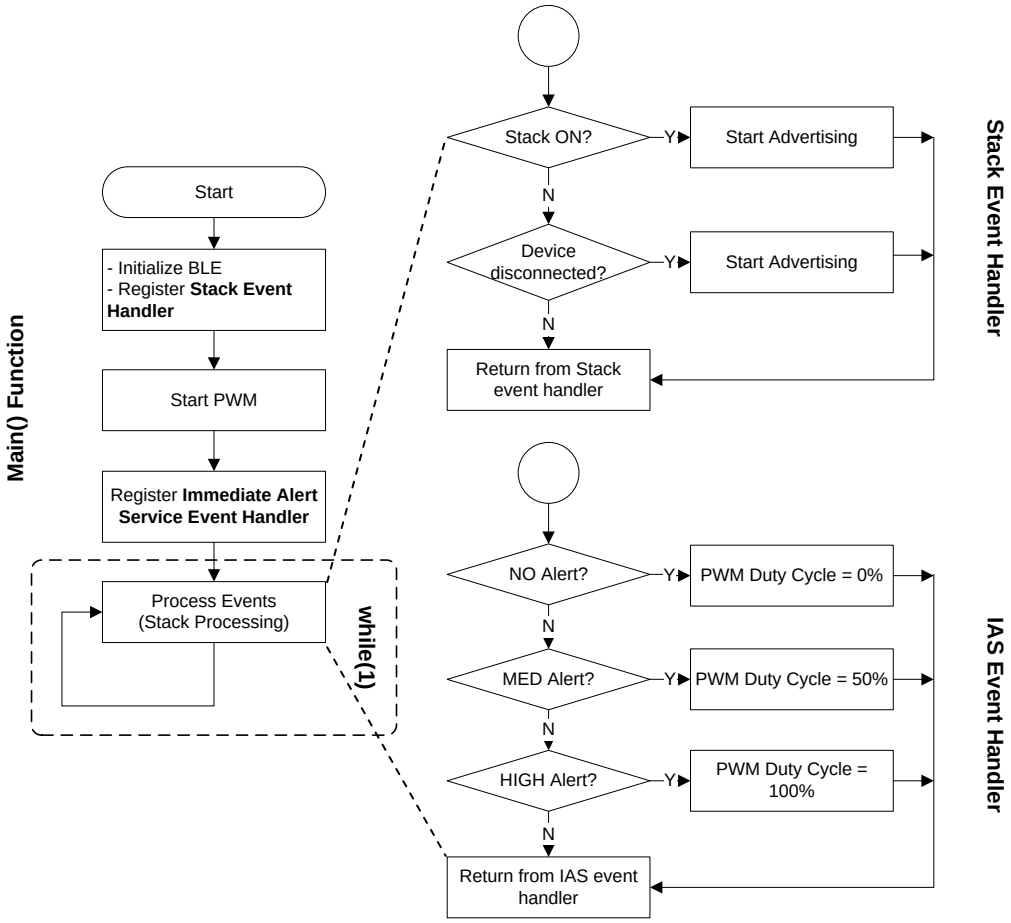
Review Firmware

The flow chart in Figure 15 provides the firmware flow.

An **Event Handler** is an asynchronous firmware routine that executes operations in response to specific events. In our main.c firmware for this lab, the *StackEventHandler()* receives BLE Stack events such as connection establishment, disconnection, etc.

A custom event handler, *IASEventHandler* as used in our main.c, is used to provide user-defined responses to events occurring on the Immediate Alert Service. To use this custom event handler, you must register its name using the *CyBle_IasRegisterAttrCallback()* API function provided by the Cypress BLE Component - this is commonly known as a **Callback**.

Figure 15: 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. **StackEventHandler() function**: This function handles the common events generated for the BLE Stack. For example, the event *CYBLE_EVT_STACK_ON* is received when the Stack is initialized and turned ON. The event *CYBLE_EVT_GAP_DEVICE_DISCONNECTED* is received when the BLE connection is disconnected.
3. **IasEventHandler() function**: This function handles the events for Immediate Alert Service. As a part of the event, it receives the alert levels which are used to drive the LED as per Table 2.

Table 2: Alert Level vs LED Blink Rate

Alert Level	PWM Duty Cycle	LED Status
NO_ALERT	100%	Always OFF
MILD_ALERT	50%	LED toggling every second
HIGH_ALERT	0%	Always ON

Note: LED Pin Component is connected to the inverted terminal (line_n) of the PWM Component, thus 100% duty cycle corresponds to LED always OFF.
The firmware has already been implemented as a part of the template project.

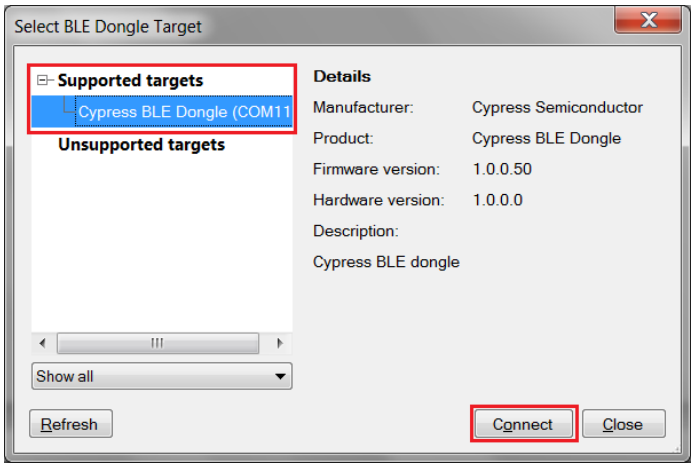
Build and Program

1. Build your final application by clicking the menu item **Build** -> **Build BLE Lab 2**.
2. Click the menu item **Debug -> Program** to program the generated hex file to the PSoC 4 BLE chip on the BLE Pioneer Kit.

Testing

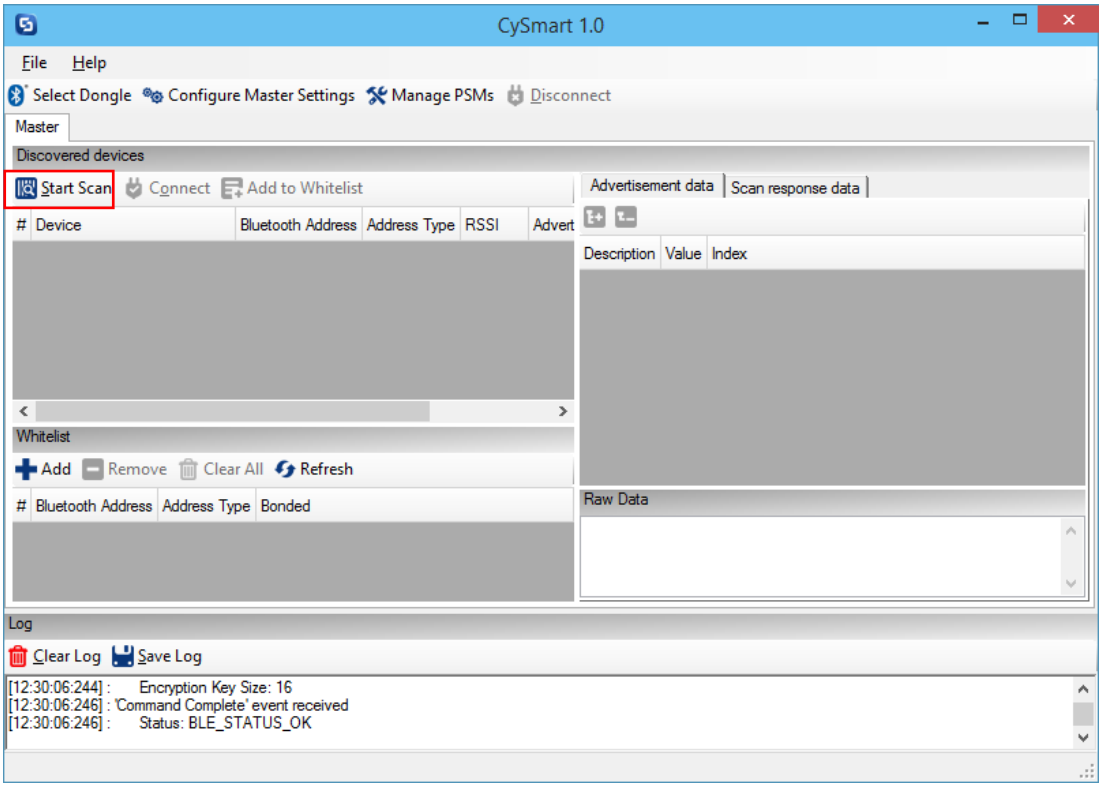
1. Plug the BLE-USB Bridge (included with the BLE Pioneer Kit) in your computer's USB port.
2. On your computer, launch **CySmart 1.0**. It is located in the **All Programs -> Cypress -> CySmart** folder in the Windows start menu. The tool opens up and asks you to **Select BLE Dongle Target**. Select the **Cypress BLE Dongle (COMxx)** and click **Connect**, as shown in Figure 16.

Figure 16: CySmart 1.0: Select BLE Dongle Target



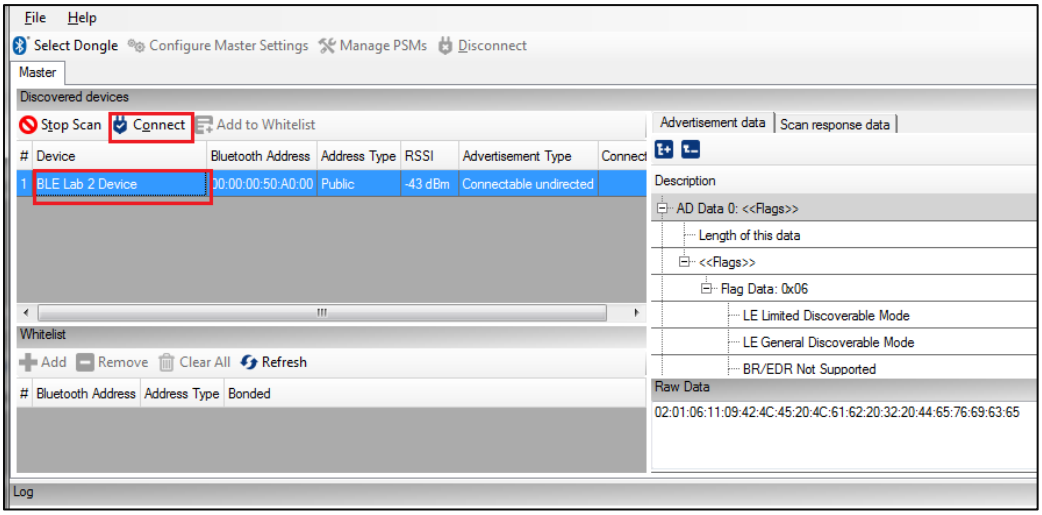
3. When the BLE-USB Bridge is connected, click **Start Scan** to find your BLE device. See Figure 17.

Figure 17: Finding a BLE Device



4. The scanning stops automatically once all advertising BLE devices are shown. The tool lists them all in the **Discovered devices** section.
5. Click your device name to see the **Advertisement data** and **Scan response data** packets on the right. See Figure 18.

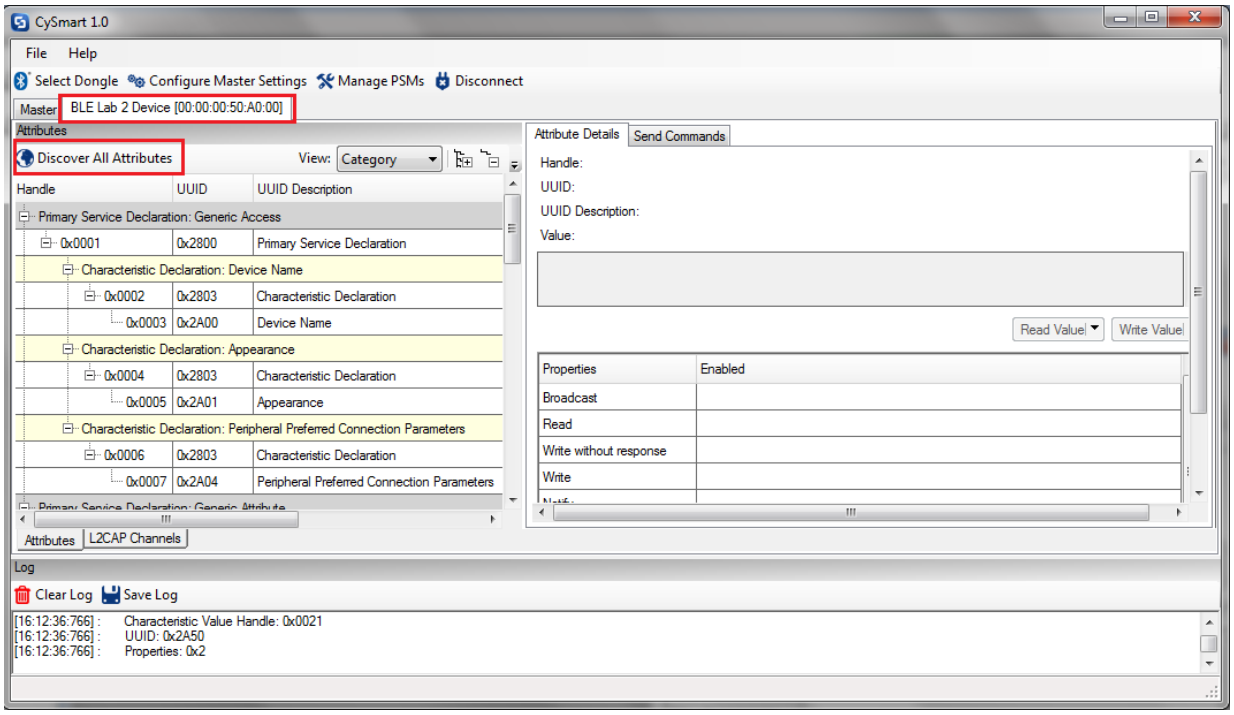
Figure 18: Checking Discovery Details of a Connected BLE Device



6. Click **Connect** as seen in Figure 18 to connect to the device.

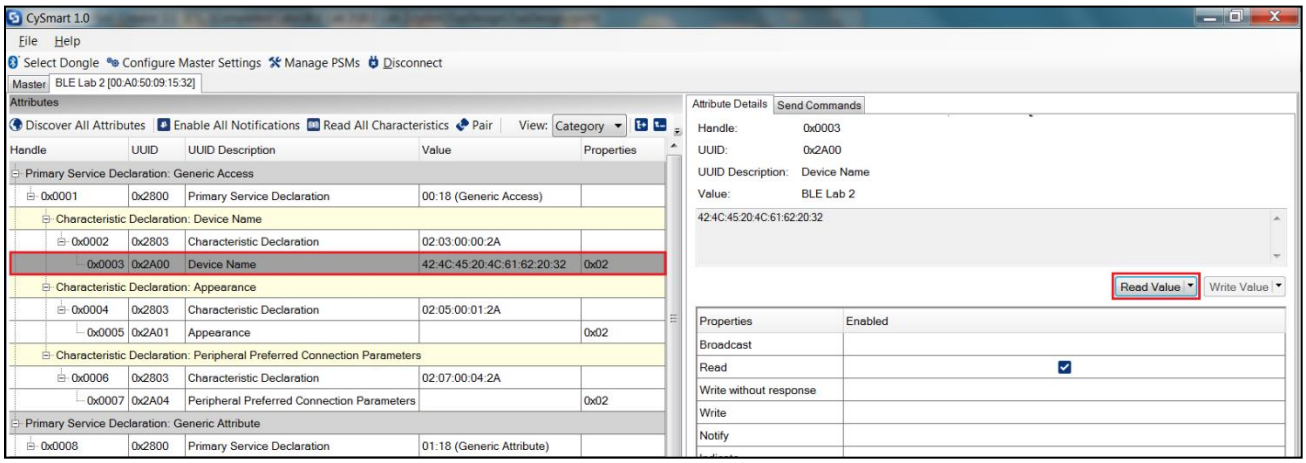
7. The tool now opens a new tab for the connected device. Click **Discover All Attributes** to list all the Attributes in the device, with their respective UUIDs (Universally Unique Identifier) and descriptions. See Figure 19.

Figure 19: Discovering Attributes of a Connected BLE Device



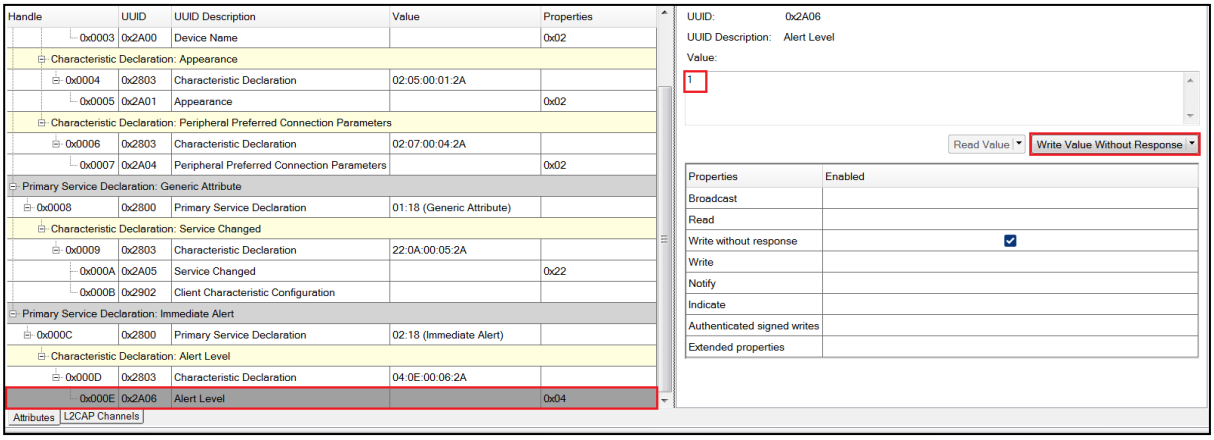
8. Click any row in the list of Attributes to see its details on the right. To read an Attribute's value, click **Read Value** on the right, as shown in Figure 20.

Figure 20: Reading Attribute Value



9. Locate the **Alert Level** Attribute for the **Immediate Alert Service**. On the right, write a value of **1** to start blinking the LED. See Figure 21.

Figure 21: Writing Attribute Value



10. Write a value of **2** to keep the LED always on.
11. Write a value of **0** to turn off the LED.

Congratulations, you have successfully completed your second PSoC 4 BLE design.

Additional Exercises

1. Configure the PWM Component's Period parameter value to change the LED blink rate to 1-Hz.

2. Add the Device Information Service (DIS) to the Find Me Profile.
   Hint: Additional Services can be added by right-clicking the **Find Me Target** in the BLE Component Configuration Tool, and selecting **Add Service**.

3. Repeat this lab with a PRoC BLE device.
   Hints:
   a. Create a New Project using the PRoC BLE device: CYBL10563-56LQXI.
   b. Disable the unused Components by right-clicking on the Component and selecting the Disable option.
   c. Copy over the main.c firwmware from the PSoC 4 BLE lab 2 template.

Document Revision History

Revision	By	Description
**	PMAD	Initial Release
*A	GUL	Edits for BLE terminology