# **Description**

This lab introduces you to the PSoC Creator IDE, used for developing and programming applications for PSoC 3 PSoC 4 and PSoC 5 family of devices, and the Bluetooth Low Energy feature of PSoC 4 BLE. This lab is divided into two parts:

- A. Learn how to use PSoC Creator to implement and debug PSoC designs by implementing a simple blinking LED design
- B. Design a 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

#### Bluetooth Smart Ready Mobile Phone



		_				
Profile	Find Me Locator	Scans for Services	<b>←&gt;</b>	Advertises Services	Profile	Find Me Target
GAP Role	Central	Initiates Connection	<b>←&gt;</b>	Accepts Connection	GAP Role	Peripheral
GATTRole	Client	Writes Alert Level	<b>←&gt;</b>	Executes Alert Level	GATTRole	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.

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

#### Table 1: Recommended Advertising Interval Values



## **Initial Kit Setup**

The BLE Pioneer Kit connects to the PC over a USB interface. The kit enumerates as a composite device and three separate devices appear under the Device Manager window in the Windows operating system. Follow these steps to get started:

- 1. If you have not already installed the BLE Pioneer Kit Software, do that first.
- 2. Before power-on, verify that the PSoC 4 BLE module (red color) is plugged into the baseboard on your kit.
- 3. Plug in your BLE Pioneer Kit to your PC using the provided USB cable. You will see the Windows driverenumeration process begin.
- 4. Wait for the driver installation to complete as shown in Figure 2 and Figure 3. Click on **Skip obtaining driver software from Windows Update** to speed up the process, especially if you do not have an Internet connection. The required drivers are already installed on your computer with the kit software and therefore do not need to be downloaded via the Windows Update.

Figure 2: BLE Pioneer Kit Driver Installation in Progress

ſ	Driver Software Installation		×
	Your device is ready to use		
	USB Composite Device KitBridge KitProg Programmer KitProg USBUART	Ready to use Searching Windows Update Searching Windows Update Searching Windows Update	
	Obtaining device driver software from Windows Skip obtaining driver software from Windows Up	Update might take a while. odate	
			Close

Figure 3: BLE Pioneer Kit Driver Installation Complete

Driver Software Installation	Agennal, 12-bit Dalty 1	×
Your device is ready to use		
USB Composite Device USB Input Device KitProg (1.2.3.3) KitProg USB-UART (COM4)	Ready to use Ready to use Ready to use Ready to use	
		Close



## **Objectives**

- 1. Learn how to use PSoC Creator to implement and debug PSoC designs
- 2. Implement a simple blinking LED design

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

# **Block Diagram**





# Theory

The goal of this lab is to learn the basics of the PSoC Creator IDE by implementing a simple blinking LED controlled by the hardware TCPWM (Timer, Counter, PWM) block. A new PSoC Creator project is created, Components are placed and configured, the LED pin is assigned, and firmware is written to start the TCPWM Component to blink the LED. The project is then programmed onto the kit and the results are observed.

The red LED on the kit is connected to pin P2[6] of the PSoC 4 BLE chip. The LED turns ON when the PWM drives the pin low, and OFF when the pin drives the pin high.



## **Procedure** Create a New Project

Use PSoC Creator 3.1 to create the project for the BLE Pioneer Kit. Follow these steps to create your first project:

- 1. Open PSoC Creator 3.1. It is located in the All Programs -> Cypress -> PSoC Creator 3.1 folder in the Windows start menu.
- 2. Create a new project by using the **New -> Project...** option in the **File** Menu, as shown in Figure 5.

Figure 5: PSoC Creator New Project Creation Menu



3. The New Project dialog appears. In the Default Templates menu, select PSoC 4100 BLE / PSoC 4200 BLE Design. Give a name to your project and the location where you want to store it. In the example shown in Figure 6, the project is named BLE Lab 1 and is saved on the user's desktop. Ensure the selected Device is set to the default CY8C4247LQI-BLE483 and Project template is set to Empty schematic.

Figure 6: PSoC Creator New Project Configuration



4. Click **OK** to create the blank project with your selected device.



# **Configure Schematic**

After the project is created, the schematic editor opens. Here, you can place and configure Components. Follow these steps:

 The Component Catalog window is on the right side. Locate the PWM (TCPWM mode) Component under the Digital → Functions category. Drag and drop this Component to the schematic editor (TopDesign.cysch) in the middle, as shown in Figure 7. Note that you can also find this Component by typing PWM in the Search for... box.



#### Figure 7: Placing the PWM Component on the Schematic

- 2. Drag and drop a **Clock** Component and a **Digital Output Pin** Component to the schematic editor. **Clock** Component is available under **System** category and **Digital Output Pin** Component is available under the **Ports and Pins** category.
- Wire the Clock\_1 Component output to the input of the PWM\_1 Component. For wiring, use the wire tool available on the top-left corner of the schematic editor as shown in Figure 8. You can also use the hot key W, to select the wire tool.



Figure 8: Selecting the wire tool



- 4. Wire the **Digital Output Pin** component to the **line\_n** terminal of the **PWM** Component.
- 5. The final schematic should look as shown in Figure 9.



#### Figure 9: Schematic for Lab 1 Part A

6. Double-click the PWM\_1 Component to open its Component Configuration Tool. Note, you can also right-click the Component and select Configure.... Rename the Component to PWM. This will be the name used as a prefix for the generated API functions. In the PWM tab of the configuration tool, change the Period value to 500 and Compare value to 250 as shown in Figure 10. Click OK to apply the changes and close the Component Configuration Tool.



Figure 10: TCPWM Component Configuration Tool

Configure 'TCPWM_P4'				3	×
Name: PWM	Duilt in			4	
					-
Prescaler:	1x V	Input Prese	nt Mode		
PWM align:	Left align 👻	reload	Rising edge	-	
PWM mode:	PWM 🔻	stop	Rising edge		
Dead time cycle:	0	switch	Rising edge	•	
Stop signal event:	Don't stop on kill	count 📃	Level	-	
Kill signal events				_	
Kill signal event:	Asynchronous	Reg	gister Swap Registe	erBuf	
Output line signal:	Direct output 🔹	Period 500	65535		
Output line_n signal:	Direct output 🔹	Compare 250	65535		
<ul> <li>On terminal count</li> <li>On compare/capture</li> </ul>	On terminal count On compare/capture count				
	PWM, left aligned				
coun	counter 250, 250, 250, 250, 250, 250, 250, 250,				
(interrupt only)	(interrupt only) TC + + + + + CC + + + + + +				
Datasheet			OK Apply	Cancel	

7. Now, double click on the **Clock\_1** Component and set the **Frequency** as **1 kHz** as shown in Figure 11.

Figure 11. Clock Configuration Tool

Configure 'cy_clock'	×
Name: Clock_1	
Basic Built-in 4	Þ
Clock type: <ul> <li>New</li> <li>Existing</li> </ul>	
Source: <auto></auto>	•
Initially align to: HFCLK (48.000 MHz)	•
Specify: Frequency: 1 kHz	
✓ Tolerance: - 5% + 5%	
Use fractional divider	
Summary API Generated: Yes Uses Clock Tree Resource: Yes	_
By default, all clocks are marked as 'start on reset'. The setting can be changed in the Design Wid Resources editor.	Э
Datasheet OK Apply Cancel	5

8. Now, double click on the Pin\_1 Component and set the name as **Red\_LED** as shown in Figure 12.



Figure 12: Pin Configuration Tool

Configure 'cy_pins'	Buit-In	utput ) Drive mode Strong drive	Initial drive state:       High (1)       Min. supply voltage:       ☐ Hot swap
Datasheet		ОК	Apply Cancel
Datasheet		ОК	Apply Cancel

# **Configure Design Wide Resources**

Once the schematic configuration is complete, it is time to configure the Design Wide Resources (DWR). These settings are used to configure overall chip settings, such as pin assignments, clock configurations, system debugging options etc. Do the following:

1. On the left-hand side of the PSoC Creator IDE is the **Workspace Explorer** which shows the files contained within this project. Double-click on the **BLE Lab 1.cydwr** file to open the Design Wide Resources window, as shown in Figure 13.





\_ 🗆 🗙



2. The **Pins** tab of the Design Wide Resources opens by default. For **Red\_LED**, click the drop-down list under the **Port** column, and assign it to **P2[6]**. You can also drag and drop the **Red\_LED** to **P2[6]** on the chip view, or just type **P2[6]** into box in the **Port** column. This step is shown in Figure 14.

Alias	Name 🗠		Port	Pin	Lock
	Red_LED		<b>-</b>	•	
		P0[0] P0[2] P0[2] P0[4] P0[5] P1[0] P1[1] P1[3] P1[4] P1[5] P1[6] P2[0] P2[1] P2[2] P2[4] P2[4] P2[4]	LPCOMP:in_p[0], TCPWM0:lin LPCOMP:in_n[0], TCPWM0:lin TCPWM1:line_out, SCB1:uart TCPWM1:line_out_compl, SCB LPCOMP:in_p[1], TCPWM1:lin LPCOMP:in_n[1], TCPWM1:lin OA2:vplus, TCPWM0:line_out OA2:vout_10x, TCPWM1:line_ OA3:vout_10x, TCPWM1:line_ OA3:vout_10x, TCPWM2:line_out OA2:vplus, TCPWM2:line_out OA3:vplus, TCPWM2:line_out OA3:vplus, TCPWM2:line_out OA3:vplus, TCPWM2:line_out OA3:vplus_alt, TCPWM3:line OA3:vplus_alt, TCPWM3:line OA0:vplus, SCB0:spi_select OA0:vout_10x, SRSS:wakeup, OA1:vout_10x, SRSS:ext_clk OA1:vminus		
		P2[6]	OA0:vplus_alt		
		P3[0]	SARMUX:pads[0], TCPWM0:lin		
		P3[1]	SARMUX:pads[1], TCPWM0:lin		
		P3[3]	SARMUX:pads[2], TCPWMI:IIn SARMUX:pads[3], TCPWMI:lin		
		P3[4]	SARMUX:pads[4], TCPWM2:lin		
		P3[5]	SARMUX:pads[5], TCPWM2:lin		
		P3[6]	SARMUX:pads[6], TCPWM3:lin T		

Figure 14: Pin Assignment

This completes DWR configuration. Before you proceed, build your project once to generate the component source files. Once these files are available, PSoC Creator auto-completes code for you when you write firmware. On the menu bar, click **Build -> Generate Application** to generate the code files for your project.



## **Implement Firmware**

The firmware consists of just one line of C-code added to the provided template.

1. From the **Workspace Explorer** double-click **main.c** to open the source file in the code editor, as shown in Figure 15.

#### Figure 15: PSoC Creator Code Editor Showing main.c template



2. Start the PWM Component as per the code shown in Figure 16. This API starts the PWM block with the period and compare value configured in the configuration tool.

Figure 16: Snapshot of Lab 1 Code

12 <b>#in</b>	12 #include <project.h></project.h>				
13	13				
14 int	14 int main()				
15 🖂 {					
16	<pre>PWM_Start(); // Starts the PWM block</pre>				
17					
18	for(;;)				
19 🖕	{				
20貞	<pre>/* Place your application code here. */</pre>				
21 -	}				
22 <sup>L</sup> }					



## **Build and Program**

You are now ready to build your project and program it to the kit. Follow these steps:

1. Click the menu item **Build -> Build BLE Lab 1** to build your project, as shown in Figure 17.

File Edit View Project Build Debug Tools <u>Window</u> <u>H</u>elp 🔁 🔁 🧃 💕 🖬 🖉 🎒 🕵 🛗 Build BLE\_Lab 1 Shift+F6 ug Clean BLE\_Lab 1 🛗 • 🚵 🦃 🍟 🌺 📮 🗐 🖵 Clean and Build BLE\_Lab 1 Workspace Explorer (1 proje 🕮 esign.cysch 🛛 BLE **.** . Ctrl+Break Cancel Build 💷 Workspace 'BLE Lab 1' (1 🖡 🖻 🔁 Project 'BLE\_Lab 1' [C 🋸 Ctrl+F6 Compile File right YOUR CO TopDesign.cysch Rights Reserv 6 Generate Application BLE\_Lab 1.cydwr BLISHED, LICE 🗀 Header Files Generate Project Datasheet 🗄 🧰 Source Files CONFIDENTIAL AND omponents main.c 8 \* WHICH IS THE PROP 🖻 🧰 Generated\_Source 9 🗄 🗀 PSoC4 10 \* \_\_\_\_\_ 🗄 🧀 Clock 1 \*/ 11 Clock\_1.c Data 12 #include <project.h> Clock\_1.h

Figure 17: Building a Project

Click the menu item Debug -> Program to program your kit, as shown in Figure 18. After the programming
is complete, the red LED on the kit blinks at a rate of 2Hz with 50% duty-cycle.



Figure 18: Programming a Project

You may also see a pop-up window asking you to confirm which device to program (**Select Debug Target**, Figure 19). Simply choose the **KitProg** (the PSoC 5-based programmer and debugger on the baseboard) that is connected to **PSoC 4200-BL CY8C4247LQI-BL483** as shown below, and click **Connect** and **OK**.



Figure 19: PSoC Creator "Select Debug Target" Window

Select Debug Target	? <mark>×</mark>
⊡-5 KitProg/1911022A011A3400	PSoC 4200-BL CY8C4247LQI-BL483
PSoC 4200-BL CY8C4247LQI-BL483	PSoC 4200-BL (ARM CM0) Silicon ID: 0x0BB11477 Cypress ID: 0x0E34119E Revision: PRODUCTION
	Target unacquired
Show all targets	Connect
	ОК

3. To debug and step through the firmware, click the menu item Debug -> Debug. This programs the PSoC 4 BLE device and starts the debug. Once debug is started, click the Step Over button in the Debug menu or in the control panel shown in Figure 20. This will single step through the top level code. Many other options are available in the debugger such as breakpoints at specific locations of the code and conditional breakpoints. Use the menu option Help -> PSoC Creator Help Topics -> Using the Debugger fir additional information.





Congratulations! Your blinking LED completes part A of lab 1



# Part B: 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 21: Lab 1 Part B 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.







## **Procedure**

If you have skipped part A of this lab, then 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.

If you have implemented part A of this lab, then you can skip step 1 and 2.

### Configure Schematic

 Open the template project named BLE Lab 1 by clicking the menu item File > Open Project/Workspace as shown in Figure 23.

<u>File Edit View Proj</u>	ect <u>B</u> uild	<u>D</u> ebug <u>T</u> ools <u>W</u> indow <u>H</u> elp	
New	•	LXIOCI A-	•
<u>O</u> pen	•	Droject/Workspace	
Example Project		🗳 <u>F</u> ile Ctrl+O	
Add	Þ		
<u>C</u> lose	Ctrl+F4	PSoC <sup>®</sup> Creator	
Close Workspace			

#### Figure 23: Open Project

r								
🛄 Open			×					
🚱 💭 🛡 📕 🕨 BLE Labs 🕨	Template Labs + BLE_Lab 1 +	✓ Search BLE_Lab 1	٩					
Organize 🔻 New folder			0					
★ Favorites	Name	Date modified	Туре					
💻 Desktop	🐌 BLE_Lab 1.cydsn	3/9/2015 5:58 PM	File folder					
, Downloads	🛄 BLE_Lab 1	3/9/2015 4:43 PM	PSoC Crea					
😻 Dropbox (Persona 🗏								
🐉 Recent Places								
Dropbox (Platforr								
iCloud Drive								
🜸 iCloud Photos								
闫 Libraries								
Documents								
🕹 Music 🔻 🗸			•					
File name	BLE Lab 1	All Project Files (* cypr	::* cuu ▼					
	<u>O</u> pen ▼ Cancel							



2. If not already selected, double-click **TopDesign.cysch** from the **Workspace Explorer** to open the schematic editor, as shown in Figure 24.

#### Figure 24: 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 25.



Figure 25: Placing the BLE Component on the Schematic Editor



PSoC 4 BLE Lab #1: Setup a BLE Connection

- 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 26.

Figure 26: BLE Component Configuration – General Tab

Name: BLE	
General Profiles GAP Settings L2CAP Settings Built-in	4 Þ
📴 Load configuration 🗟 Save configuration	
Profile Collection	
This made allows to change the target profile from a list of supported profiles	
This mode allows to choose the target prome nonn a list of supported promes.	
O Broadcaster/Observer	
This mode does not support connections.	
Host Controller Mode	
This mode allows communication with a host stack using a component embedded UART.	
Profile: Find Me	
Profile role: Find Me Target (GATT Server)	
GAP role: Peripheral	
Vise BLE low power mode	
	Canad
Uatasneet OK Apply	Cancel

Note: You can choose any name for the BLE Component. Unlike other Components, BLE uses a pre-compiled library so its functions always start with **CyBle** no matter what the component is called. In this example, we've used **BLE**. See Figure 26.

6. **Profiles Tab** - Because we chose the standard **Find Me** profile, 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 27.



Figure 27: 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 28.

- 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 unique 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 28: GAP Settings Tab – General Settings

Name: BLE								
General Profiles GA	P Settings L2CAP Settings Built	-in	4 ۵					
General	Device address							
Advertisement setting Advertisement packe Scan response pack	Public address (Company ID - Company assigned): 00A050-XXXXX							
Security	You can use the user config to store the public device ad	You can use the user configuration section of the supervisory flash to store the public device address for mass production.						
	Device name:	BLE Lab 1						
	Appearance:	Generic Tag 🗸						
	Attribute MTU size (bytes):	23						
	Adv/Scan TX power level (dBm):	0 -						
	ConnectionTX power level (dBm):	0 -						
Restore Defaults								
Datasheet		OK Apply Ca	ncel					



#### 7.2. Peripheral Role -> Advertisement Settings

This section configures the advertisement settings for the GAP Peripheral. Configure these parameters as described below. See Figure 29. To learn more about these parameters, refer to the Bluetooth Component datasheet.

- a. **Discovery mode**: This parameter defines how your 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.
- 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). Uncheck the Timeout (s) checkbox. Disabling the timeout ensures device continuously advertises even if no connection request is received.
- f. **Slow advertising interval**: Uncheck the timeout will automatically disable the slow advertising interval.

	0		5	
Name: BLE				
General Profiles GAF	Settings L2CAP Settings	Built-in	4 ۵	
General	Discovery mode:	General	^^	
<ul> <li>Advertisement setting</li> <li>Advertisement packet</li> </ul>	Advertising type:	Connectable undirected advertising		
- Scan response pack Peripheral preferred	Filter policy:	Scan request Any   Connect request Any		
Security	Advertising channel map:	All channels		
	Advertising interval			
	Fast advertising interval:		=	
	Minimum (ms):	20	-	
	Maximum (ms):	30 🖨		
	Timeout (s):	30 🔺		
	Slow advertising interval			
	Minimum (ms):	1000		
۰ III ۲	Maximum (ms):	10240		
Restore Defaults	✓ Timeout (s):	150 A		
Datasheet		OK Apply	Cancel	

Figure 29: 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** (<u>U</u>niversally <u>U</u>nique <u>Id</u>entifier) for **Immediate Alert**, and the **Appearance**. See Figure 30.



Figure 30: GAP Settings Tab - Advertisement Packet

Name: BLE General Profiles GAP	• Settings L2CAP Setti	ngs Built-in		4 Þ			
General	Advertisement data se	Advertisement packet:					
- Advertisement settinc	Name	Value	•				
Advertisement packe	🖶 🔽 Flags			- AD Data 1: < <flag< td=""></flag<>			
- Scan response pack	General discove	erable mode		Length			
Security	BR/EDR not su	pported		⊡< <flags>&gt;</flags>			
	🕀 📃 Local Name			BR/EDR no			
	TX Power Level			AD Data 2: << Cor			
	+ Slave Connection I	🕂 🔄 Slave Connection Interval Range					
	🖙 🔽 Service UUID		⊡<< Complete li:				
	Immediate Aler	t		-Service: Im			
	🗉 📃 Service Data	🕂 🗌 Service Data					
	🙂 📃 Service Manager T	Service Manager TK Value		[1]			
	P ✓ Appearance			⊢AD Data 3: < <app< td=""></app<>			
	Data (	Generic Tag	_	Length			
	Public Target Addr	Public Target Address					
	Random Target Address						
Restore Defaults	Advertising Interva		*	<ul> <li>▲ Ⅲ ▶</li> </ul>			
Datasheet	r	ОК	A	Apply Cancel			

#### 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 31.

Name: BLE					
General Profiles GAI	Settings L2CAP Settings	Built-in	4 Þ		
General	Scan response data setting	Scan response packel			
- Advertisement setting	Name	Description			
- Advertisement packe	🗉 🗹 Local Name		AD Data 1: < <local n<="" td=""></local>		
Peripheral preferred	Local name Comp	lete	<ul> <li>Length</li> </ul>		
Security	+ TX Power Level		⊡-< <local name="">&gt;</local>		
	+ Slave Connection Interv	al Range	'B'		
	Service UUID		'Ľ'		
	+ Service Data		'E'		
	🖶 📃 Service Manager TK Va				
	Appearance	'נ'			
	Public Target Address	'a'			
	🕀 📃 Random Target Address	'b'			
	Advertising Interval				
	E E Bluetooth Device Ad	LE Bluetooth Device Address			
	E LE Role				
< <u> </u>	Manufacturer Specific E				
Restore Defaults			∢		
	P				
Datasheet		ОК	Apply Cancel		

Figure 31: GAP Settings Tab - Scan Response Packet

#### 7.5. Peripheral preferred connection parameters

These parameters are used when the Peripheral wants to update the connection parameters. In this case, these parameters are sent to the Central to update the connection parameters. For this application keep these settings to default.

#### 7.6. Security

Finally, configure the BLE security settings. Set the parameters as shown in Figure 32. 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. **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**.
- e. **Encryption key size (bytes)**: Determines the size of encryption keys while pairing. Leave this parameter to the default value of **16**.

Name: BLE			
General Profiles GAP	Settings L2CAP Settings	Built-in	4 Þ
General Peripheral role Advertisement settinc	Security mode:	Mode 1	
- Advertisement packe	Security level:	No Security (No authentication, No encryption)	
- Scan response pack Peripheral preferred	I/O capabilities:	No Input No Output	
- Security	Bonding requirement:	No Bonding -	
	Encryption key size (bytes):	16	
Restore Defaults			
Datasheet		OK Apply Canc	el

Figure 32: 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 this project.



PSoC 4 BLE Lab #1: Setup a BLE Connection

10. Your schematic should look as shown in Figure 33.

Figure 33: Completed Schematic View



11. It is now time to generate application for your project. Click the menu item **Build** -> **Generate Application**. All Component source code automatically gets generated.



### **Firmware**

If you have completed Part A of this lab and are using the same project for Part B then replace the code of main.c of this project with the code available in Lab1\_main.txt file available in the **BLE Workshop -> Labs -> Supporting Files** folder. If you started with the template project, main.c already has the code for this project.

#### **Review Firmware**

The flow chart in Figure 34 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. This function is registered when the BLE component is started using the *CyBle\_Start()* API function provided by the Cypress BLE Component – this is commonly known as a **Callback**.

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\_lasRegisterAttrCallback()* API function provided by the Cypress BLE Component.





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.

Next, the handler for IAS events is registered by calling CyBle\_IasRegisterAttrCallback(IasEventHandler). Again, the function can have any name – in this project we use IasEventHandler.

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. In this project, any time we receive an event for which the device is not connected we start advertising.
- 3. **IasEventHandler() function**: This function handles the events for the 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.

Alert Level	PWM Duty Cycle	LED Status
NO_ALERT	0%	Always OFF
MILD_ALERT	50%	LED toggling at 2Hz
HIGH_ALERT	100%	Always ON

Note: LED Pin Component is connected to the inverted terminal (line\_n) of the PWM Component and the LED is active low, thus 0% 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 1**.
- 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.
- 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 35.

elect ble bongle rurget		
□- Supported targets	Details	
Cypress BLE Dongle (COM11	Manufacturer:	Cypress Semiconductor
Unsupported targets	Product:	Cypress BLE Dongle
	Firmware version:	1.0.0.50
	Hardware version:	1.0.0.0
	Description:	
	Cypress BLE dongle	
↓ III ▶		
Show all 🔹		
Refresh		Connect Close

Figure 35: 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 36.

Figure 36: Finding a BLE Device

<u>F</u> ile	<u>H</u> elp										
👸 Seleo	t Dongle 🍳	Configure	Master Settings	🛠 Manage P	SMs 👹	Discon	nnect				
Master											
Discove	ered devices										
Kan Star	t Scan 👹 🕻	onnect 🖪	Add to Whitelist				Advertisem	nent dat	ta Scan response dat	a	
# Devi	ce		Bluetooth Address	Address Type	RSSI	Advert	E+ E-				
							Description	Value	Index		
<						>					
Whitelis	t										
📥 Ado	d 📃 Remov	e 🎁 Clear	All 😏 Refresh								
# Bluet	ooth Address	Address Typ	e Bonded				Raw Data				
											~
Log											
💼 <u>C</u> lea	rLog <u> </u> av	ve Log									
[12:30:00	6:244] : En	cryption Key	Size: 16								^
[12:30:00	5:246] : Comm 5:246] : Sta	and Complete atus: BLE_ST	ATUS_OK								
											~

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 37.

<u>F</u> ile	<u>H</u> elp									
🚯 Se	🚯 Select Dongle 🤏 Configure Master Settings 🛠 Manage PSMs 🌼 Disconnect									
Mast	Master									
Disc	Discovered devices									
Stop Scan 🕏 Connect 📰 Add to Whitelist						Advertisement data Scan response data				
# D	evice	Bluetooth Address	Address Type	RSSI	Advertisement Type	Coni	E E			
1 K	ensington Eureka 3127	20:CD:39:8D:31:27	Public	-38 dBm	Connectable undirected		Description			
2 BI	LE Lab 1	00:A0:50:0B:12:10	Public	-57 dBm	Connectable undirected		AD Data 0: < <flags>&gt;</flags>	Ξ		
3 P	eer Device	D6:BB:1E:1F:2A:91	Random	-75 dBm	Connectable undirected		- Length of this data			
		1			1		⊟< <flags>&gt;</flags>			
							⊟ Flag Data: 0x06			
							- LE Limited Discoverable Mode			
							- LE General Discoverable Mode			
Whit	elist		,			,	BR/EDR Not Supported			
	dd 🗖 Remove 🏛 Cle	ear All 🗿 Refresh					Simultaneous LE and BR/EDR to Same Device	Ŧ		
4 0		an Turne Bended					∢			
# DI	uelooin Address Addre	ess Type Bonded	_	Raw Data						
							02:01:06:0A:09:42:4C:45:20:4C:61:62:20:31:03:03:02:18	^		
								-		
1.00										
LUg										
Scan	Scanning for BLE devices									

- 6. Click **Connect** as seen in Figure 37 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 (<u>Universally Unique Identifier</u>) and descriptions. See Figure 38.

Figure 38: Discovering Attributes of a Connected BLE Device

<u>F</u> ile	<u>H</u> elp								
🚯 Sele	🚯 Select Dongle 👒 Configure Master Settings 🛠 Manage PSMs 👹 Disconnect								
Master	Master BLE Lab 1 [00:A0:50:0B:12:10]								
Attribut	Attributes Attribute Details Send Commands								
🔇 Disc	over All Attribut	es Enable All Notifications	View: Simple	- 🗉 🖬 💂	Handle:	<b>^</b>			
Handle	UUID	UUID Description Value	Properties		UUID:				
					UUID Description:				
					Value:				
						=			
					Read Value	Write Value			
					Properties	Enabled			
					Broadcast				
					Read				
					Write without response				
					Write				
					III				
Attribut	tes L2CAP Chann	nels							
Log									
209									

 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 39.



Figure 39: Reading Attribute Value

<u>F</u> ile <u>H</u> elp	D											
8 Select Do	ngle 👒 C	onfigure N	Master Settings 🛠 Manage PSMs   🖞 <u>D</u> iscor	nnect								
Master BLE	Lab 1 [00:A	0:50:0B:12:	10]			3						
Attributes							Attribute Details Ser	nd Comma	nds		_	_
Oiscover I	All Attribu	tes 🛛 🛃 Er	nable All Notifications 🔟 Read All Characte	eristics 🖑 Pair View: Catego	ory 🔻 🗄 🎦	Ð	Handle:	0x0003			1	•
Handle		UUID	UUID Description	Value	Properties		UUID:	0x2A00				
Primary Se	ervice Dec	laration: G	eneric Access				UUID Description:	Device N	lame			1
- 0x0001		0x2800	800 Primary Service Declaration 00:18 (Generic Access)				Value: BLE Lab 1					
🖯 Cha	racteristic	Declaration	n: Device Name				42:4C:45:20:4C:61:62	5:20:4C:61:62:20:31		*		
			02:03:00:00:2A		Ξ			- C				
	0x0003	0x2A00	Device Name	42:4C:45:20:4C:61:62:20:31	0x02	1				~	1	=
⊜ Cha	e- Characteristic Declaration: Appearance					Read Value Value			Write Value	Ð		
⊡ 0x0004 0x28		0x2803	Characteristic Declaration	02:05:00:01:2A			Dreparties		Enchlad		1	1
	0x0005	0x2A01	Appearance		0x02		Froperties		Enabled		11	1
⊡- Cha	racteristic	Declaration	n: Peripheral Preferred Connection Parameter	s			Broadcast			_		1
⊟- 0	x0006	0x2803	Characteristic Declaration	02:07:00:04:2A			Read			<b>~</b>		
	0x0007	0x2A04	Peripheral Preferred Connection Parameters		0x02		Write without respo	onse				
P Primary Service Declaration: Generic Attribute												
		0x2800	Primary Service Declaration	01-18 (Generic Attribute)		-	Notify					÷
Attributes L	2CAP Char	inels		,		1 8						
Log												

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 40.

#### Figure 40: Writing Attribute Value

_						. 3			
Handl	e	UUID	UUID Description	Value	Properties		UUID:	0x2A06	
	0x0003	0x2A00	Device Name		0x02		UUID Description:	Alert Le	vel
	E Characteristic Declaration: Appearance					Value:			
	⊡-0x0004	0x2803	Characteristic Declaration	02:05:00:01:2A			1		*
	0x0005	0x2A01	Appearance		0x02		—		
	Characteristic Declaration: Peripheral Preferred Connection Parameters						Ŧ		
		0x2803	Characteristic Declaration	02:07:00:04:2A					Read Value Vite Value Without Response -
	0x0007	0x2A04	Peripheral Preferred Connection Parameters		0x02				
🗄 Pri	Primary Service Declaration: Generic Attribute			1	Properties		Enabled		
÷	0x0008	0x2800	Primary Service Declaration	01:18 (Generic Attribute)		1	Broadcast		
	- Characteristic Declaration: Service Changed					Read			
	⊡-0x0009	0x2803	Characteristic Declaration	22:0A:00:05:2A		Ξ	Write without respo	inse	
	0x000A	0x2A05	Service Changed		0x22		Write		
	0x000B	0x2902	Client Characteristic Configuration				Notify		
- Pri	mary Service Dec	laration: Ir	nmediate Alert				Indicate		
÷	0x000C	0x2800	Primary Service Declaration	02:18 (Immediate Alert)		1	Authenticated sign	ed writes	
	E- Characteristic Declaration: Alert Level					Extended propertie	s		
		0x2803	Characteristic Declaration	04:0E:00:06:2A					
	0x000E	0x2A06	Alert Level		0x04	I.			
Attrib	utes L2CAP Char	nels							
_						_			

- 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 first PSoC 4 BLE design.



## **Additional Exercises**

Change the Tx Power of the BLE Component and check the difference in range.
 Additional Information: Reducing the Tx power reduces the range over which it can communicate but will result in lower noise emissions. You can check the difference in range by observing the change in RSSI level, shown in CySmart when device is discovered.

Hint: Tx Power can be changed from the General Settings of the BLE Component

 Add the Device Information Service (DIS) to the Find Me Profile.
 Additional Information: The DIS contains nine Characteristics which provides additional information about the device like Manufactuer Name, Model Name, Serial Number etc.

You can add a Service to an existing profile, by right clicking on the Profile role (**Find Me Target**) under the **Profiles** tab of the BLE Component Configuration tool and selecting **Add Service -> Device Information**, as shown in Figure 41. You can configure the Characteristics of this service by adding data of your choice.

Name: BLE		
General Profiles GAP Settings 12CA	Alert Notification	4.6
	Battery	
	Blood Pressure	
	Bond Management	
🛓 💠 Add Service 🕨	Bootloader	
× Delete	Continuous Glucose Monitoring	
<ul> <li>Reset branch to default</li> </ul>	Current Time	
Сору	Cycling Power	
Paste	Cycling Speed and Cadence	
E E Savo	Device Information	
Save	Environmental Sensing	
Rename	Glucose	
1 Move up	Health Thermometer	
Move down	Heart Rate	
	Human Interface Device	
	Immediate Alert	
	Internet Protocol Support Service	
	Link Loss	
	Location and Navigation	
	Next DST Change	
	Phone Alert Status	
Datasheet	Reference Time Update	ppiy Cancei
em-rever components have been i	Running Speed and Cadence	ne <u>pesign-wiae</u> ke
	Scan Parameters	
	Tx Power	
	Custom Service	
	😂 From File	

### Figure 41: Adding Device Information Service

When Component files are generated, the BLE GATT database is re-generated and a DIS gets added to the database. Upon connection, the GATT Client can read all the Characteristics of the DIS. Please note that the GATT Client only reads the Characteristics of the DIS, thus you do not need to configure a service-specific event handler for DIS. The BLE stack will automatically provide the information whenever a read request is initiated by the GATT Client.



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 in the PRoC BLE project by right-clicking on the Component and selecting the **Disable** option.
- c. Enable a TCPWM block, configure it in PWM mode and rename the component name as PWM. Also, set the **Period** and **Compare** value appropriately.
- d. Change the clock source of the PWM block to 1kHz.
- e. Copy over the firmware from the Lab\_1\_main.txt.



# PSoC 4 BLE Lab #1: Setup a BLE Connection

# **Document Revision History (001-96274)**

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

# **Document Revision History (001-98279)**

Revision By D		Description
		Labs moved to new spec number
**	PMAD	Updated to PSoC Creator 3.2
		Integrated Lab 1 and Lab 2
**	PMAD	Updated to PSoC Creator 3.2 Integrated Lab 1 and Lab 2