CE217637 - BLE Find Me Profile with PSoC 6 MCU with BLE Connectivity

Objective

This code example demonstrates the Find Me Profile application workflow with PSoC® 6 MCU with BLE Connectivity (PSoC 6 BLE).

Overview

This code example demonstrates the Find Me Profile operation of the BLE Component. The Find Me Target uses the Find Me Profile with one instance of the Immediate Alert Service to display the alerts if the Client has configured the device for alerting. The Find Me Target operates with other devices that implement the Find Me Profile. The device switches to Deep Sleep mode between BLE connection intervals.

Requirements

Tool: PSoC® Creator™ 4.2, PDL 3.0.4, PSoC Programmer™ 3.27.3, CySmart™ 1.3

Programming Language: C (Arm® GCC 5.4-2016-q2-update)

Associated Parts: All PSoC 6 MCU with BLE Connectivity (PSoC 6 BLE) parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit and CY8CPROTO-063-BLE PSoC 6 BLE Prototyping Kit

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that the kit is configured correctly.

1. Connect the BLE Pioneer Kit or the CY8CPROTO-063 board to the computer's USB port.
2. Connect the BLE Dongle to one of the USB ports on the computer. (Optional, not included with CY8CPROTO-063 kit.)

LED Behavior

The Advertising LED (green) blinks to indicate that the device is advertising. The Alert LED (red) may be OFF, steady ON, or blinking depending on the alert status. The Disconnect LED (blue) is ON when the device is not connected to a peer device and not advertising. When the device is in Hibernate mode, the Disconnect LED stays ON. The CY8CPROTO-063-BLE kit does not have a blue LED, so the disconnected status is not demonstrated on this kit.

If the V_{DDD} voltage is set to less than 2.7 V in the DWR settings **System** tab, only the Alert LED (red) is used. The other two LEDs will remain OFF.

Software Setup

BLE Host Emulation Tool

This example requires a BLE tool that can connect to the development board. Cypress provides two solutions: the CySmart mobile app for iOS or Android mobile devices and the CySmart Host Emulation Tool PC application for Windows. You can evaluate the behavior with any of the two options, but the CySmart mobile app is simpler. Scan one of the following QR codes from your mobile phone to download the CySmart app. You can also use any other BLE app that supports the Find Me profile or that can read/write GATT attributes.

iOS	Android

Operation

You can connect to the Find Me Target device with a BLE 4.0 or BLE 4.1-compatible device configured in the GAP Central role and is capable of discovering the Immediate Alert service and Alert Level Characteristic.

To connect to the Find Me Target device, send a connection request to the device when the device is advertising. The Advertising LED (green) blinks while the device is advertising. If the Client is connected to the Find Me Target, the Alert Level Characteristic can be written to the trigger alerts on a remote device. If the Alert Level is set to CY_BLE_MILD_ALERT, the Alert LED (red) starts blinking. If the Alert Level is set to CY_BLE_HIGH_ALERT, the Alert LED is turned ON.

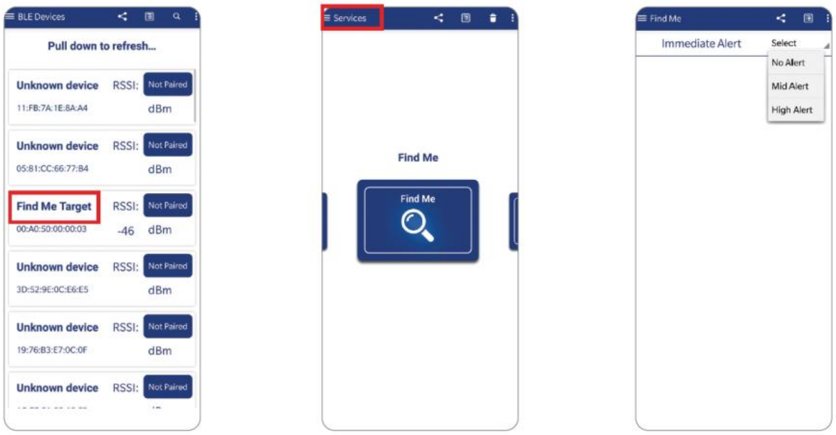
To clear the alerts, send a request from the Client to set the Alert Level Characteristic to CY_BLE_NO_ALERT. The alerts are also cleared when the connection with the Client is canceled or lost.

Operation Steps

1. Connect the CY8CKIT-062-BLE kit board or CY8CPROTO-063-BLE PSoC 6 Prototyping Kit Board into your computer's USB port.
2. If you are using the CY8CKIT-062-BLE kit, you can skip this step. If using the CY8CPROTO-063-BLE PSoC 6 Prototyping Kit Board, you will need to change the device selection as documented in Table 1. Also, pin assignments for the LEDs will need to be changed for proper LED operation. See Table 2 for LED pin assignments.
3. Build the project and program it into the PSoC 6 BLE device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.
4. Observe the green LED blinks while the device is advertising.

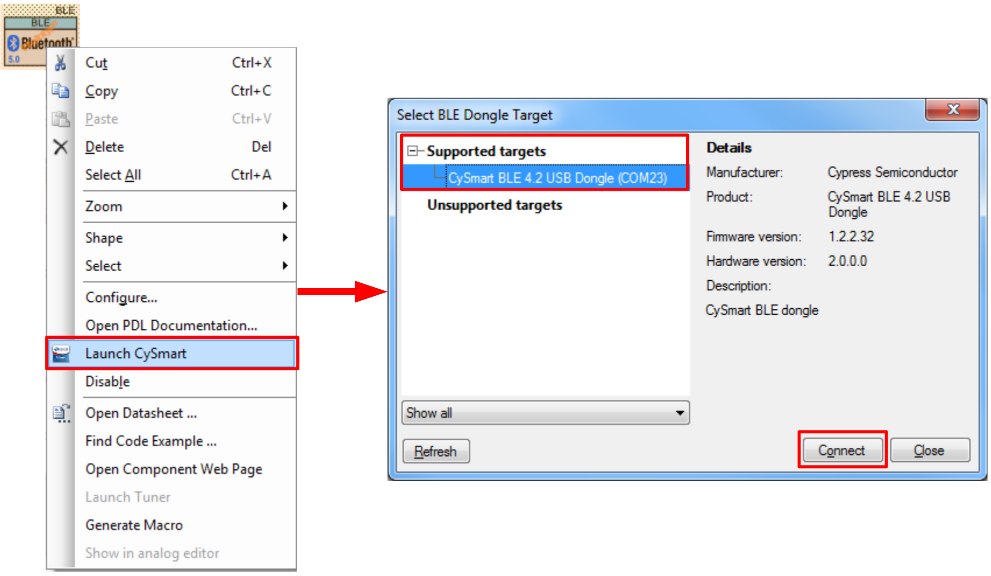
   Note: Step 5 explains how to use the CySmart mobile app. Step 6 will walk you through using the more comprehensive Windows-based CySmart Host Emulation Tool. (See Table 2 for switch and LED pin assignments.)
5. Exercise the code example with a mobile device.
   a. Install the CySmart app on your iOS or Android device from the App Store or Google Play store respectively. Launch the app after you install it.
   b. Power the development kit by connecting it to your PC using a USB cable.
   c. The BLE device will advertise for 180 seconds. The green LED blinks during this period to indicate the BLE advertising state. If more than 180 seconds has passed, pressing the user button will restart the advertising state.
   d. Connect to the "Find Me Target" device in the CySmart app on your mobile device and navigate to the "Find Me" service in the application. The red LED state can be put into three states: OFF, blink, or ON depending on the alert state. See Figure 1 for an example of CySmart mobile app interface.
      i. No Alert: LED OFF
      ii. Med Alert: LED Blinking
      iii. High Alert: LED ON

Figure 1. CySmart Mobile App Interface


BLE Devices
Pull down to refresh...
Unknown device RSSI: Not Paired
11:FB:7A:1E:8A:A4 dBm
Unknown device RSSI: Not Paired
05:81:CC:66:77:B4 dBm
Find Me Target RSSI: Not Paired
00:A0:50:00:00:03 -46 dBm
Unknown device RSSI: Not Paired
3D:52:9E:0C:E6:E5 dBm
Unknown device RSSI: Not Paired
19:76:B3:E7:0C:0F dBm
Unknown device RSSI: Not Paired
Services
Find Me
Find Me
Find Me
Immediate Alert
Select
No Alert
Mid Alert
High Alert

6. Do the following to test example using the CySmart Host Emulation PC Tool as the Find Me Target Client. This step will require the CY5670 CySmart USB Dongle, which comes with the CY8CKIT-062-BLE kit, or can be purchased separately.
   a. Connect the BLE Dongle to your Windows PC. Wait for the driver installation to complete, if necessary.
   b. Launch the CySmart Host Emulation Tool by right-clicking the BLE Component and selecting **Launch CySmart**. Alternatively, you can launch the tool by navigating to **Start** > **Programs** > **Cypress** and clicking on **CySmart**.
   c. CySmart automatically detects the BLE dongle connected to the PC. Click **Refresh** if the BLE dongle does not appear in the **Select BLE Dongle Target** pop-up window. Click **Connect**, as shown in Figure 2.

Figure 2. CySmart BLE Dongle Selection



Details

Manufacturer: Cypress Semiconductor

Product: CySmart BLE 4.2 USB Dongle

Firmware version: 1.2.2.32

Hardware version: 2.0.0.0

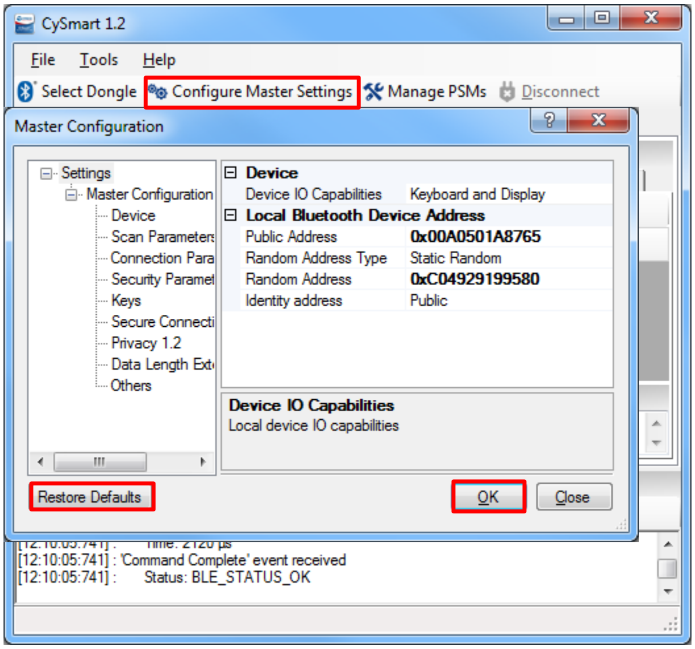
Description:

CySmart BLE dongle

Note: If the dongle firmware is outdated, you will be alerted with an appropriate message. You must upgrade the firmware before you can complete this step. Follow the instructions in the window to update the dongle firmware.

   d. Select **Configure Master Settings** and then click **Restore Defaults**, as Figure 3 shows, and then click **OK**.

Figure 3. CySmart Master Settings Configuration



Device	
Device IO Capabilities	Keyboard and Display
Local Bluetooth Device Address	
Public Address	**0x00A0501A8765**
Random Address Type	Static Random
Random Address	**0xC04929199580**
Identity address	Public

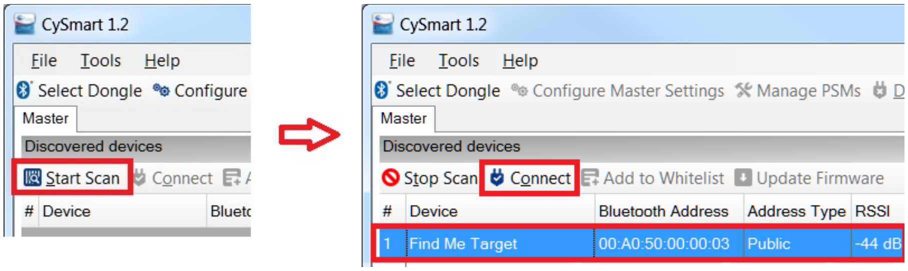
Device IO Capabilities
Local device IO capabilities

```
[12:10:05:741] :       Time: 2120 µs
[12:10:05:741] : 'Command Complete' event received
[12:10:05:741] :       Status: BLE_STATUS_OK
```

e. Press the user switch on the development kit to start BLE advertisement if the device is in Hibernate mode (Disconnect LED (blue) is ON). Otherwise, skip this step.

f. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name (configured as **Find Me Target**) should appear in the Discovered devices list, as Figure 4 shows. Select the device and click **Connect** to establish a BLE connection between the CySmart Host Emulation Tool and your device.

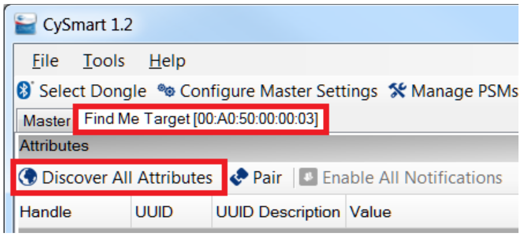
Figure 4. CySmart Device Discovery and Connection



#	Device	Bluetooth Address	Address Type	RSSI
1	Find Me Target	00:A0:50:00:00:03	Public	-44 dB

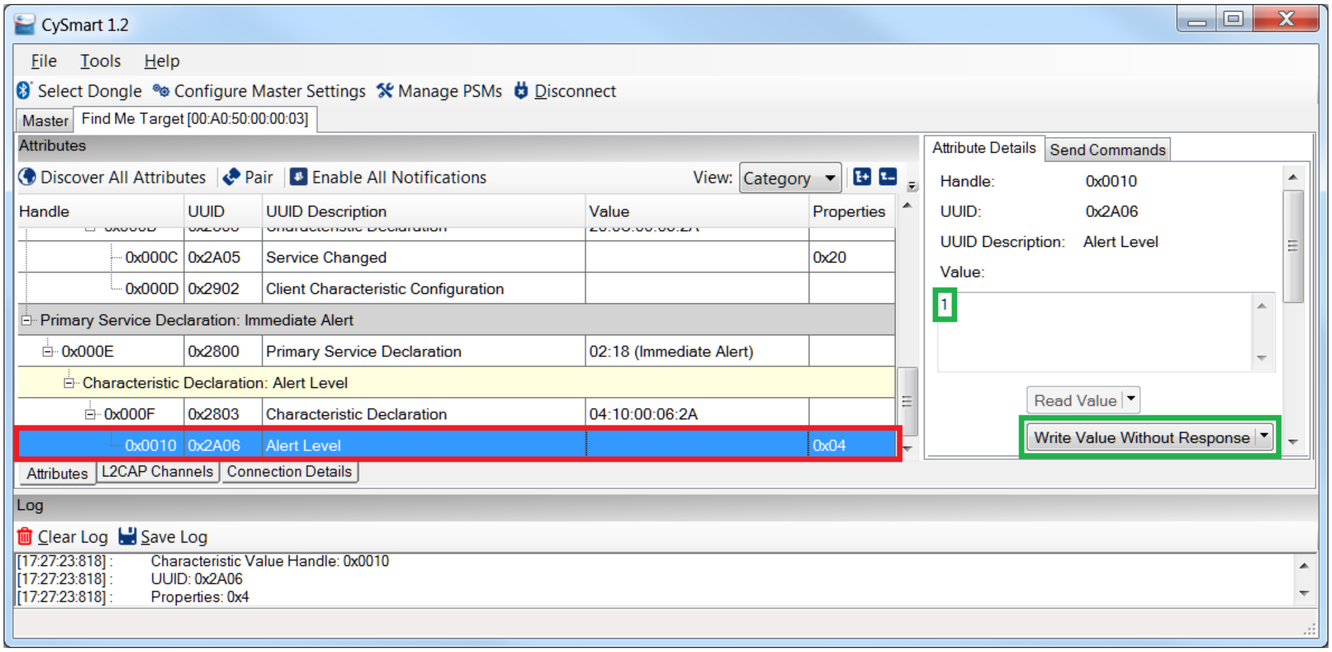
g. Once connected, switch to the **Find Me Target** device tab and click **Discover all Attributes** from the CySmart Host Emulation Tool, as shown in Figure 5.

Figure 5. CySmart Attribute Discovery



h. Select the **Alert Level** Characteristic value as shown in Figure 6 and write ‘1’ in it (mid alert) and observe that the Alert LED (red) is blinking.

Figure 6. CySmart Write Alert Level Characteristic



Handle	UUID	UUID Description	Value	Properties
0x000C	0x2A05	Service Changed		0x20
0x000D	0x2902	Client Characteristic Configuration		
Primary Service Declaration: Immediate Alert				
0x000E	0x2800	Primary Service Declaration	02:18 (Immediate Alert)	
Characteristic Declaration: Alert Level				
0x000F	0x2803	Characteristic Declaration	04:10:00:06:2A	
0x0010	0x2A06	Alert Level		0x04

Handle: 0x0010
UUID: 0x2A06
UUID Description: Alert Level
Value:
1

```
[17:27:23:818] :    Characteristic Value Handle: 0x0010
[17:27:23:818] :    UUID: 0x2A06
[17:27:23:818] :    Properties: 0x4
```

Design and Implementation

The software flow is rather simple. After the device boots, it advertises for 180 seconds for a connection to a Central such as a smartphone of PC. If a connection is made in that time, it begins a simple loop where it processes BLE events, updates the alert status, goes to Deep Sleep (to save power), and then comes back to processing BLE events. If no connection is made in the 180 seconds, the device goes into Hibernate (very low power mode) until SW2 is pressed. It then restarts the 180 second advertisement period.

Figure 7. Source File Description

Source File	Description
main_cm0p.c	CM0+ main that enables the CM4 CPU and starts the BLE stack.
main_cm4.c	Contains CM4 main(), which handles I/O unfreeze coming out of Hibernate and calls the `HostMain()` function, which is the main BLE flow.
host_main.c	Contains the main program loop and the BLE `AppCallBack()` function.
user_interface.c	This file contains the functions to update the LEDs (user interface).
ias.c	This file contains a few very simple functions to support IAS (Immediate Alert Service).

Figure 8. Firmware Flow

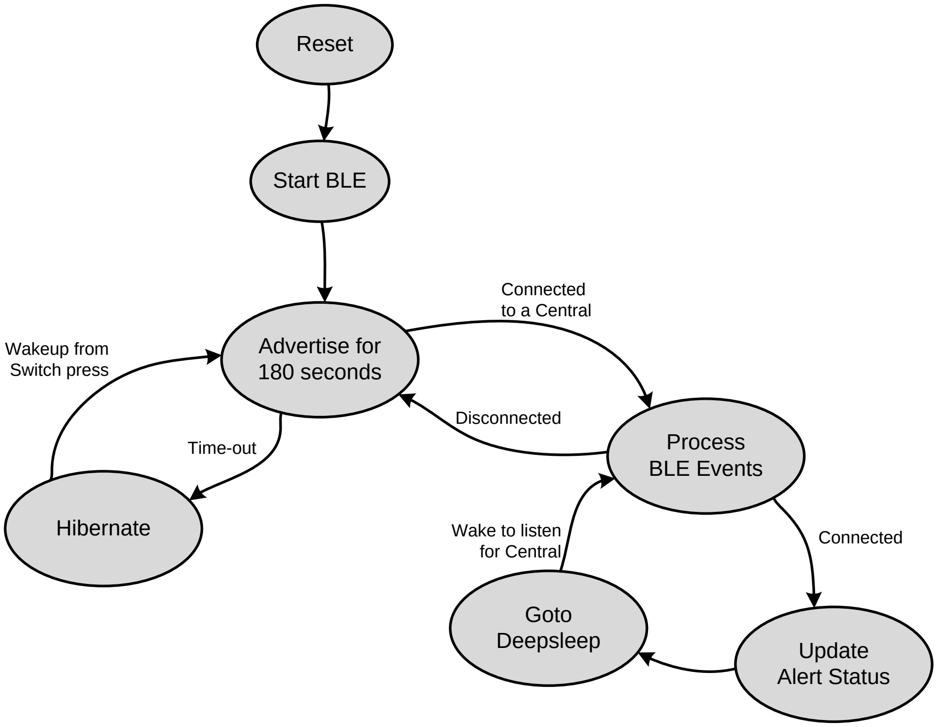
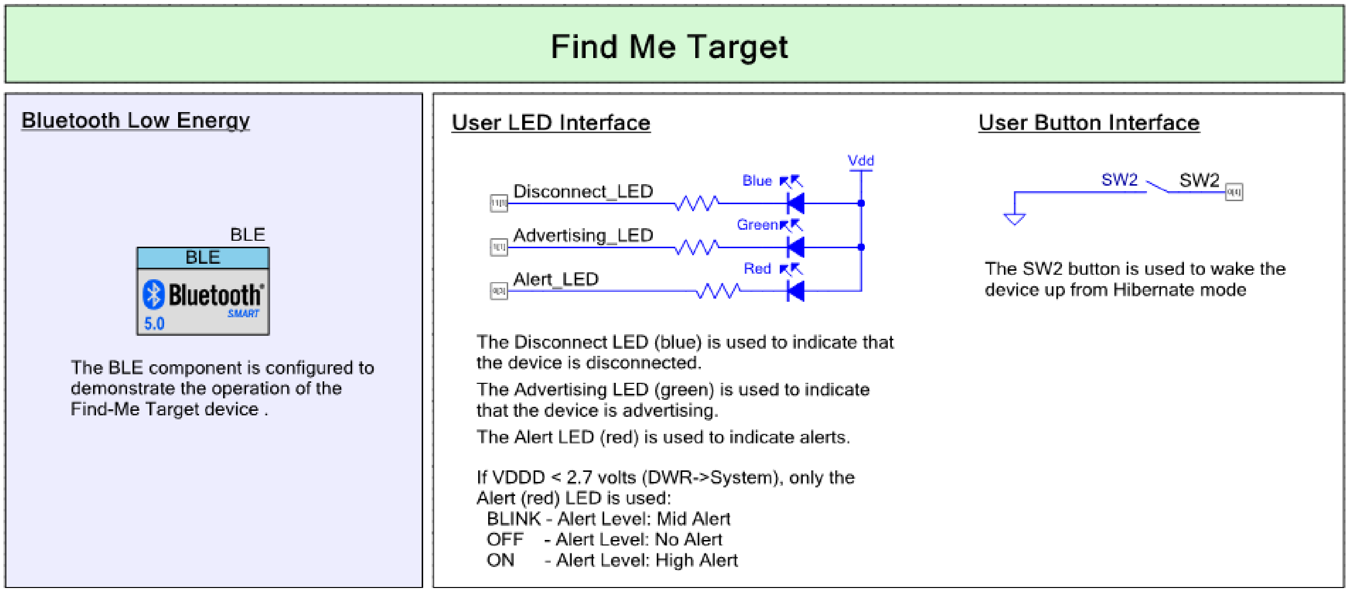


Figure 9 shows the top design schematic.

Figure 9. BLE Find Me Profile Code Example Schematic



The BLE component is configured to demonstrate the operation of the Find-Me Target device .

The Disconnect LED (blue) is used to indicate that the device is disconnected.

The Advertising LED (green) is used to indicate that the device is advertising.

The Alert LED (red) is used to indicate alerts.

If VDDD < 2.7 volts (DWR->System), only the Alert (red) LED is used:
BLINK - Alert Level: Mid Alert
OFF - Alert Level: No Alert
ON - Alert Level: High Alert

The SW2 button is used to wake the device up from Hibernate mode

The project demonstrates the core functionality of the BLE Component configured as a Find Me Target. After a startup, the device performs initialization of the BLE Component. In this project, two callback functions are used for the BLE operation. One callback function (`StackEventHandler()`) is required for receiving generic events from the BLE stack, and the other (`IasEventHandler()`) is required for receiving events from the Immediate Alert Service. The `CY_BLE_EVT_STACK_ON` event indicates the successful initialization of the BLE stack. After this event is received, the Component starts advertising with the packet structure as configured in the BLE Component Customizer. The BLE Component stops advertising when the 180-second advertising period expires. On an advertisement event timeout, the device goes to a low-power mode (Stop mode) and waits for a device reset event to wake up the device again.

While connected to a Client and between the connection intervals, the device is put into Deep Sleep mode.

Device Selection

The PSoC 6 BLE device comes in several packages and modules. Table 1 provides a list of kits and the device used for each. To change a device selection, right-click the project name in Workspace Explorer and select **Device Selector**. Choose the PSoC 6 MCU device that is used in the development kit or custom hardware.

Table 1. Device Selection

Development Kit	Device	Comment
CY8CKIT-062	CY8C6347BZI-BLD43	This is the default kit and device for this kit.
CY8CPROTO-063-BLE	CYBLE-416045-02	Change the default device selection if using the CY8CPROTO-063-BLE kit.

Pin Assignments

Pin assignments and connections required on the development board are configured for CY8CKIT-062 by default. If using a different kit, you may need to reassign the pins to be compatible with that board. Table 2 shows the pin configurations for some of the Cypress kits. Note that CY8CKIT-062 uses a single RGB (Red Green Blue) LED instead of individual LEDs.

Table 2. Pin Assignment

Pin Name	Development Kit		Comment
	CY8CKIT-062 (default)	CY8CPROTO-063-BLE	
Advertising_LED (green)	P1[1] (LED5)	P7[1] (LED4)	Part of RGB LED on CY8CKIT-062.
Disconnect_LED (blue)	P0[3] (LED5)	Any spare GPIO pin (optional)	Part of RGB LED on CY8CKIT-062. You may connect an additional LED to any available pin on the CY8CPROTO-063-BLE kit.
Alert_LED (red)	P11[1] (LED5)	P6[3] (LED3)	Part of RGB LED on CY8CKIT-062.
User Switch	P0[4] (SW2)	P0[4] (SW2)	Same pin used for both kits.

Components and Settings

Table 3 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 3. PSoC Creator Components

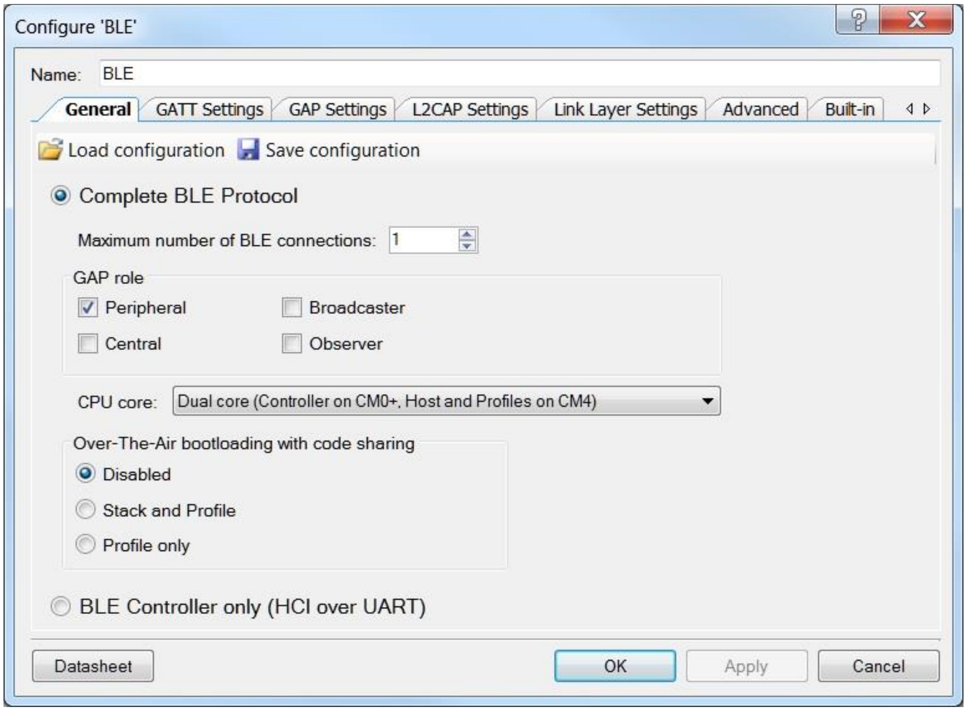
Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE Component is configured to demonstrate the operation of the Find Me Target device.	See Parameter Settings.
Digital Input Pin	SW2	The **SW2** button is used to wake the device up from Hibernate mode.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output Pin	Disconnect_LED Advertising_LED Alert_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive

For information on the hardware resources used by a Component, see the Component datasheet.

Parameter Settings

The BLE Component is configured as the Find Me Target in the GAP Peripheral role with the settings shown in the figures below.

Figure 10. General Settings



Configure 'BLE'

Name: BLE

● Complete BLE Protocol

Maximum number of BLE connections: 1

GAP role

☑ Peripheral ☐ Broadcaster

☐ Central ☐ Observer

CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)

Over-The-Air bootloading with code sharing

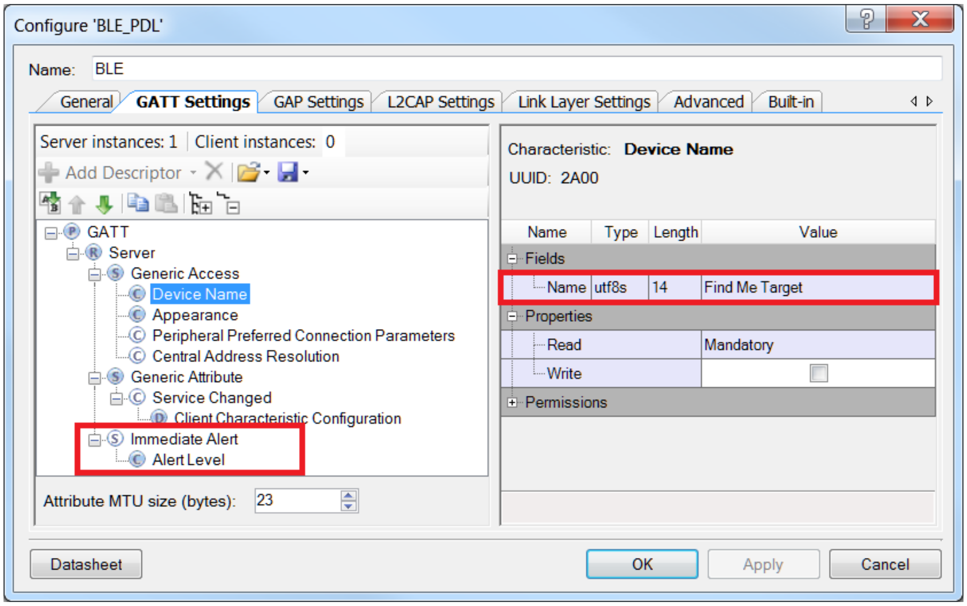
● Disabled

○ Stack and Profile

○ Profile only

○ BLE Controller only (HCI over UART)

Figure 11. GATT Settings



Configure 'BLE_PDL'

Name: BLE

Server instances: 1 | Client instances: 0

- GATT
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
      - Central Address Resolution
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
    - Immediate Alert
      - Alert Level

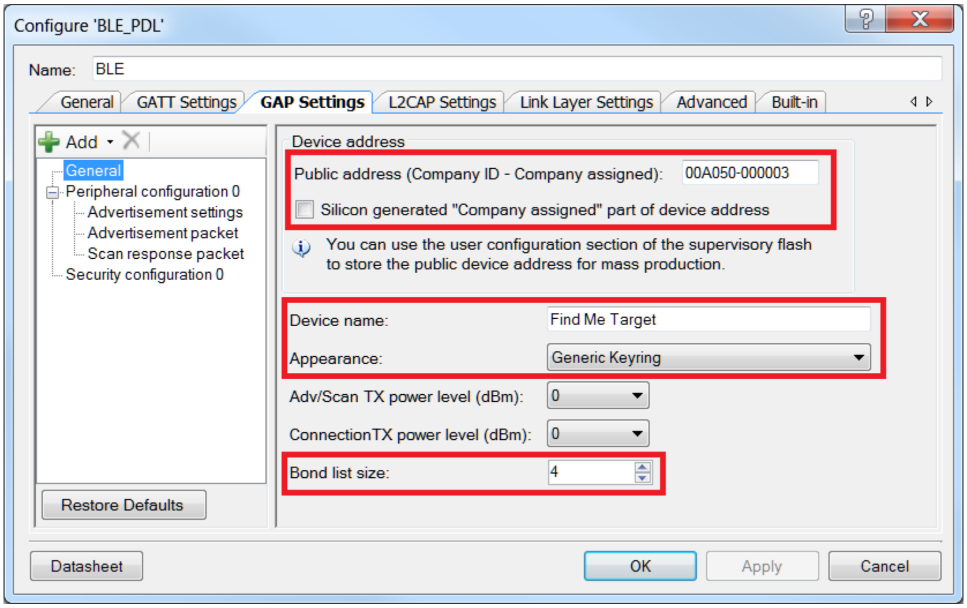
Attribute MTU size (bytes): 23

Characteristic: **Device Name**

UUID: 2A00

Name	Type	Length	Value
Fields			
Name	utf8s	14	Find Me Target
Properties			
Read			Mandatory
Write			☐
Permissions			

Figure 12. GAP Settings



Name: BLE

Device address

Public address (Company ID - Company assigned): 00A050-000003

□ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: Find Me Target

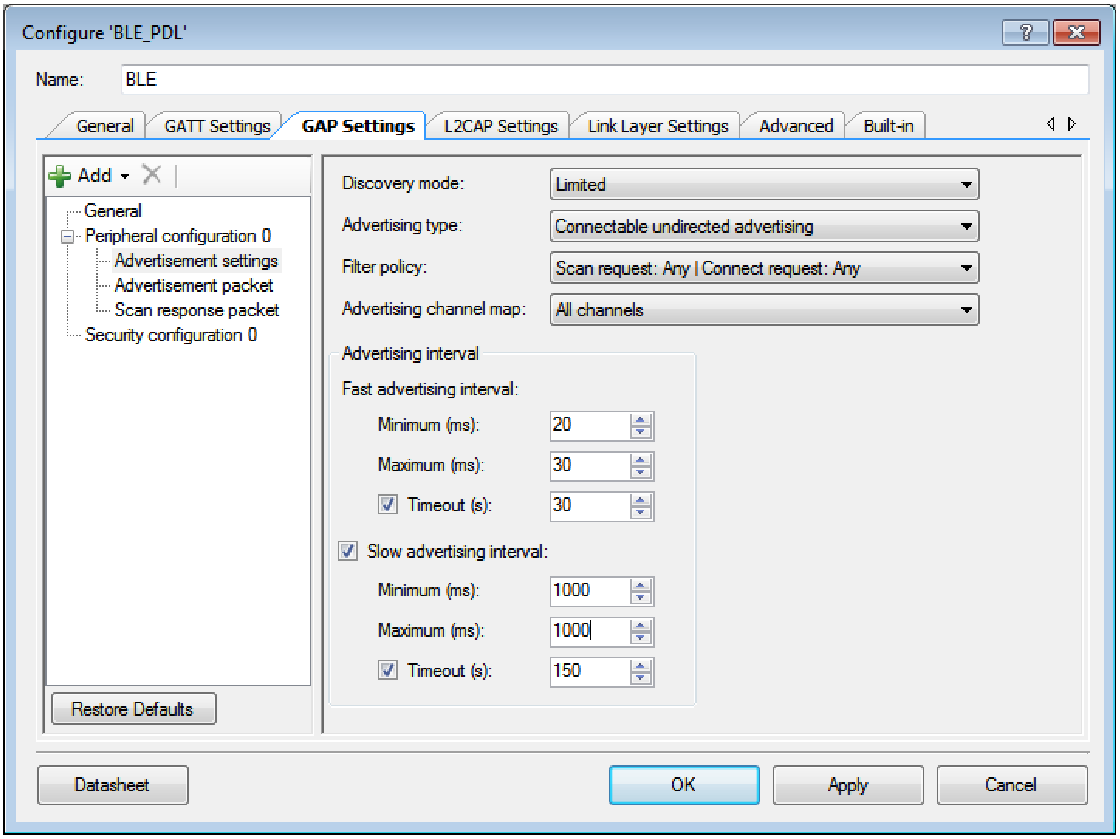
Appearance: Generic Keyring

Adv/Scan TX power level (dBm): 0

ConnectionTX power level (dBm): 0

Bond list size: 4

Figure 13. GAP Settings: Advertisement Settings



Name: BLE

Discovery mode: Limited

Advertising type: Connectable undirected advertising

Filter policy: Scan request: Any | Connect request: Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 20

Maximum (ms): 30

☑ Timeout (s): 30

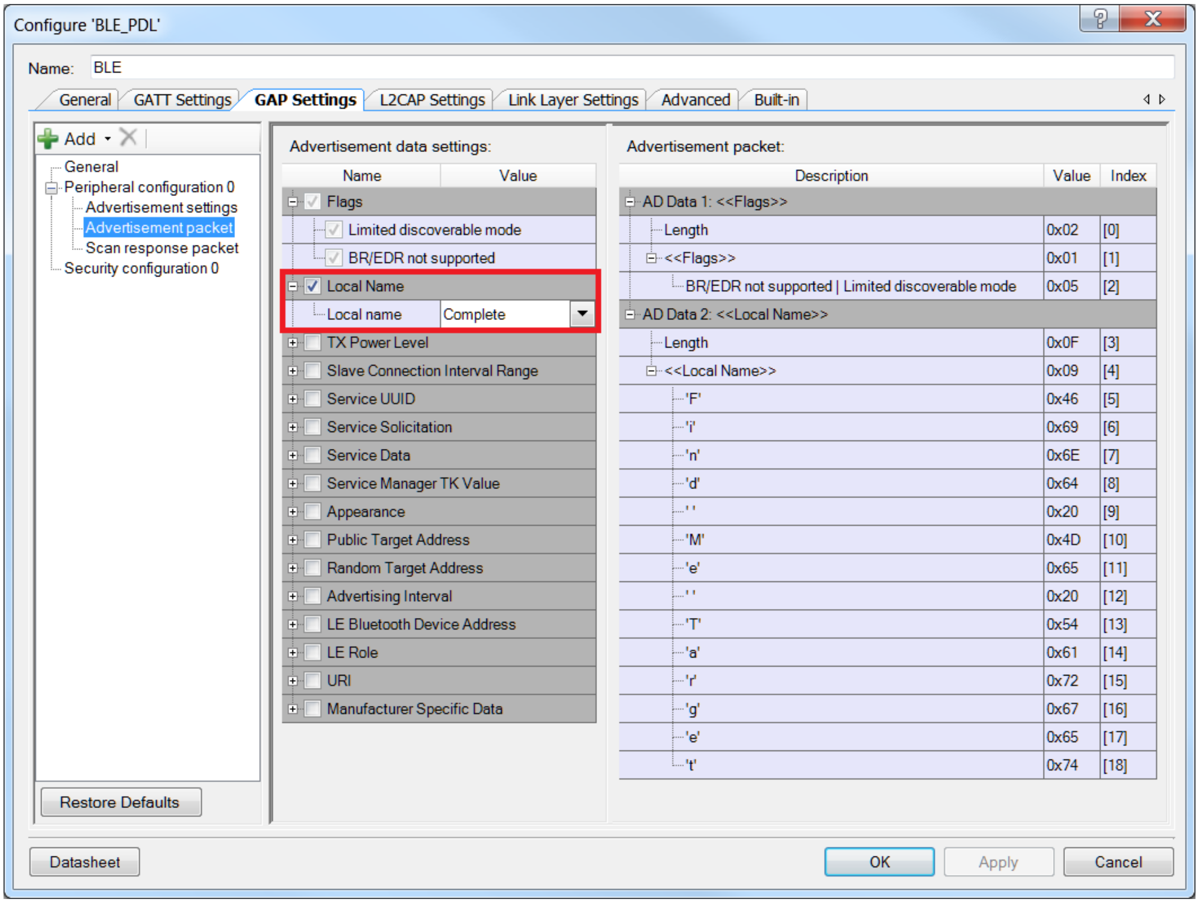
☑ Slow advertising interval:

Minimum (ms): 1000

Maximum (ms): 1000

☑ Timeout (s): 150

Figure 14. GAP Settings: Advertisement Packet



Name: BLE

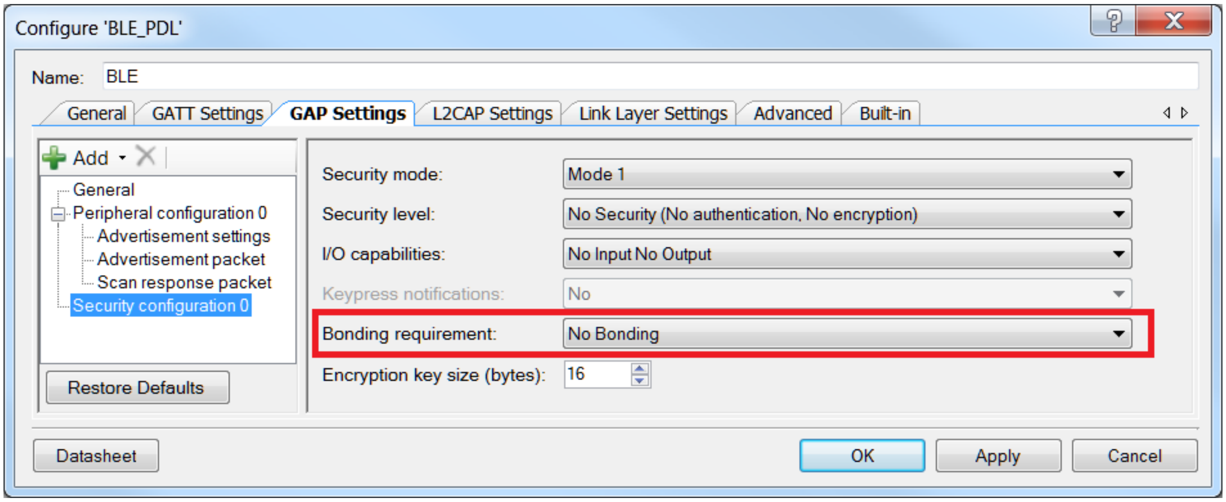
Advertisement data settings:

Name	Value
☑ Flags	
☑ Limited discoverable mode	
☑ BR/EDR not supported	
☑ Local Name	
Local name	Complete
□ TX Power Level	
□ Slave Connection Interval Range	
□ Service UUID	
□ Service Solicitation	
□ Service Data	
□ Service Manager TK Value	
□ Appearance	
□ Public Target Address	
□ Random Target Address	
□ Advertising Interval	
□ LE Bluetooth Device Address	
□ LE Role	
□ URI	
□ Manufacturer Specific Data	

Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported | Limited discoverable mode	0x05	[2]
AD Data 2: <<Local Name>>		
Length	0x0F	[3]
<<Local Name>>	0x09	[4]
'F'	0x46	[5]
'i'	0x69	[6]
'n'	0x6E	[7]
'd'	0x64	[8]
' '	0x20	[9]
'M'	0x4D	[10]
'e'	0x65	[11]
' '	0x20	[12]
'T'	0x54	[13]
'a'	0x61	[14]
'r'	0x72	[15]
'g'	0x67	[16]
'e'	0x65	[17]
't'	0x74	[18]

Figure 15. Security Settings



Name: BLE

Security mode: Mode 1

Security level: No Security (No authentication, No encryption)

I/O capabilities: No Input No Output

Keypress notifications: No

Bonding requirement: No Bonding

Encryption key size (bytes): 16

Switching the CPU Core Usage

This section describes how to switch between different CPU usage (Single CPU/ Dual CPU) in BLE PDL examples.

The BLE Component has a CPU Core parameter that defines the usage of the two cores. It can take the following values:

- Single core (Complete Component on CM0+) – only CM0+ will be used.
- Single core (Complete Component on CM4) – only CM4 will be used.
- Dual core (Controller on CM0+, Host and Profiles on CM4) – CM0+ and CM4 will be used: CM0+ for the Controller and CM4 for the Host and Profiles.

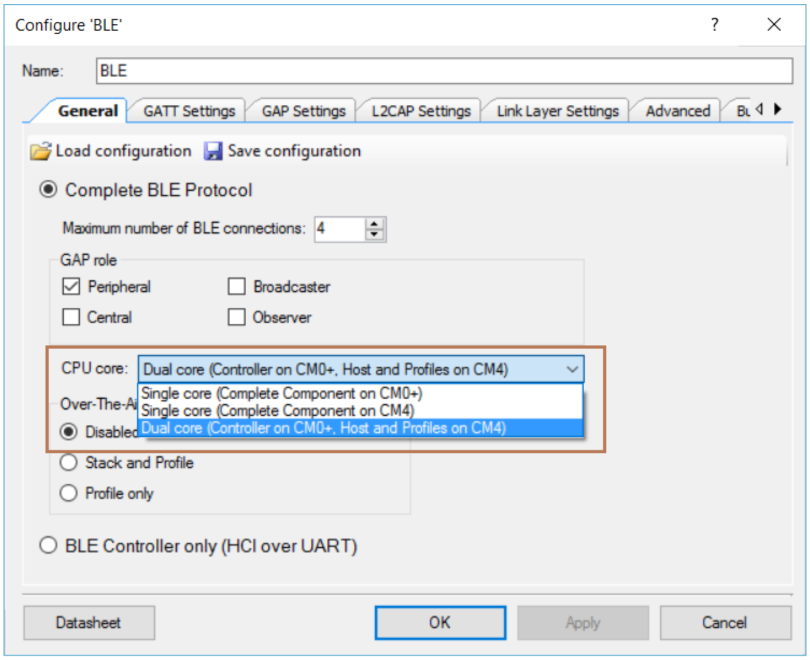
The BLE example structure allows easy switching between different CPU core options. Important to remember:

- All application host files must be run on the host core.
- The BLESS interrupt must be assigned to the core where the controller runs.

Do the following to switch CPU core usage:

1. In the BLE Component Customizer **General** tab, select the appropriate CPU core option.

Figure 16. Select CPU Core



Configure 'BLE'

Name: BLE

● Complete BLE Protocol

Maximum number of BLE connections: 4

GAP role

☑ Peripheral ☐ Broadcaster

☐ Central ☐ Observer

CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)

Single core (Complete Component on CM0+)
Single core (Complete Component on CM4)
Dual core (Controller on CM0+, Host and Profiles on CM4)

Over-The-Ai

● Disabled

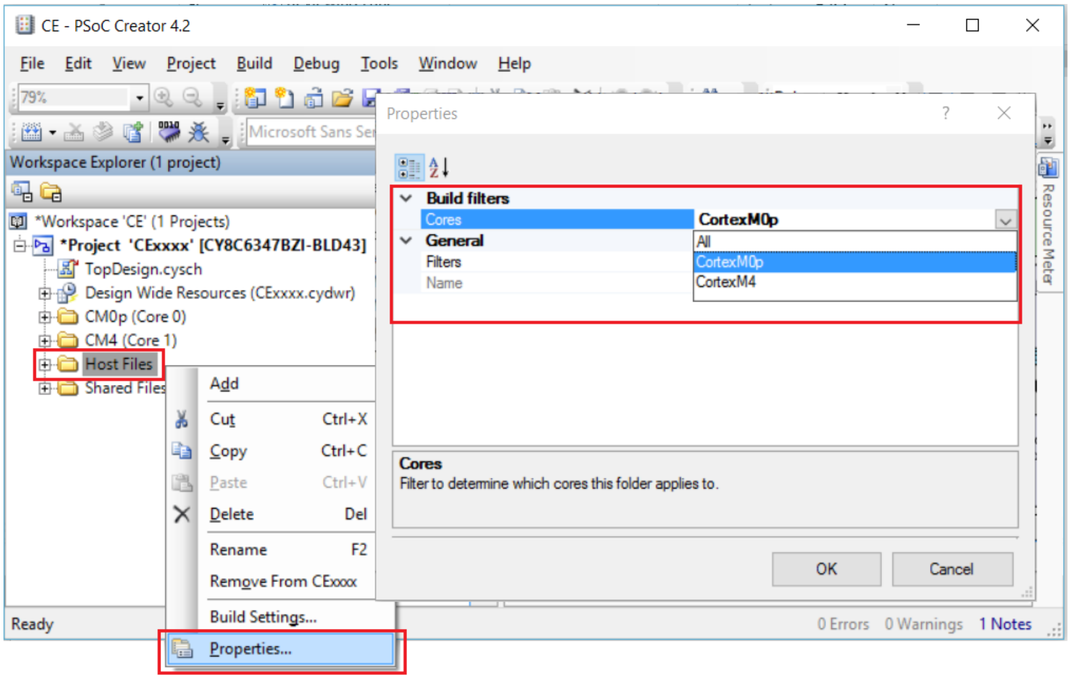
○ Stack and Profile

○ Profile only

○ BLE Controller only (HCI over UART)

2. Identify the core on which host files will run. In the Workspace Explorer panel, right-click **Host Files** and then choose **Properties**. Set the **Cores** property corresponding to the CPU core chosen in Step 1, as shown in Figure 17.
    - for Single core (Complete Component on CM0+) option – CM0+
    - for Single core (Complete Component on CM4) option – CM4
    - for Dual core (Controller on CM0+, Host and Profiles on CM4) option – CM4

Figure 17. Change Core Properties



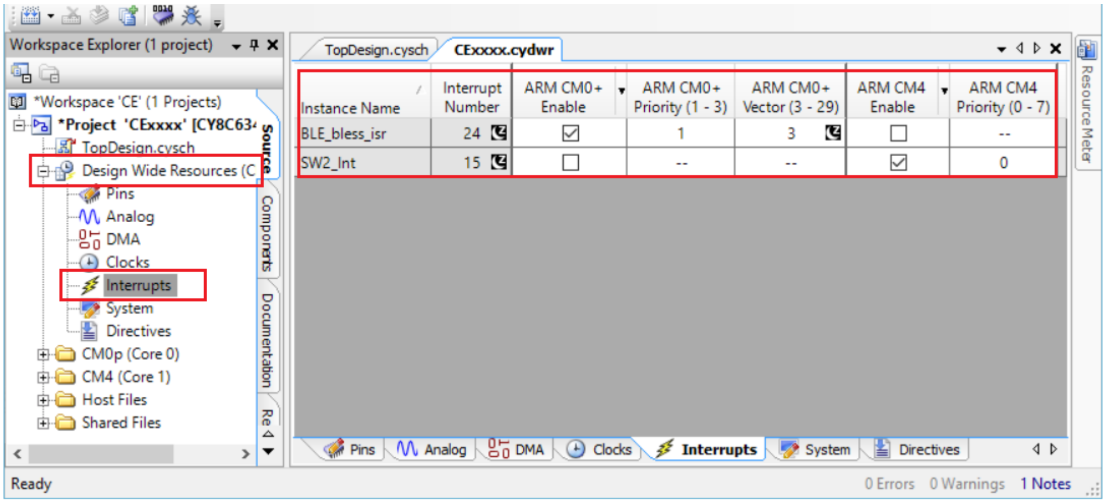
Build filters	
Cores	**CortexM0p**
General	All CortexM0p CortexM4
Filters	
Name	

Cores
Filter to determine which cores this folder applies to.

3. Assign BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to the appropriate core in **DWR** > **Interrupts** tab:

- for **Single core (Complete Component on CM0+)** option: BLE_bless_isr and peripheral interrupts on **CM0+**
- for **Single core (Complete Component on CM4)** option: BLE_bless_isr and peripheral interrupts on **CM4**
- for **Dual core (Controller on CM0+, Host and Profiles on CM4)** option: BLE_bless_isr interrupt on **CM0+,** other peripheral interrupts on **CM4**

Figure 18. Assign Interrupts



Instance Name	Interrupt Number	ARM CM0+ Enable	ARM CM0+ Priority (1 - 3)	ARM CM0+ Vector (3 - 29)	ARM CM4 Enable	ARM CM4 Priority (0 - 7)
BLE_bless_isr	24	☑	1	3	□	--
SW2_Int	15	□	--	--	☑	0

Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 BLE device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 BLE, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – Bluetooth® LE Test and Debug Tool		CySmart is a Bluetooth LE host emulation tool for Windows PCs. The tool provides an easy-to-use Graphical User Interface (GUI) to enable the user to test and debug their Bluetooth LE peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE_PDL) Component		The Bluetooth Low Energy (BLE_PDL) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
Device Documentation		
PSoC® 6 MCU: PSoC 63 with BLE. Datasheet.		PSoC® 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		
CY8CPROTO-063-BLE PSoC 6 BLE Prototyping Kit		

Document History

Document Title: CE217637 - BLE Find Me Profile with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17637

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6090384	NPAL	06/05/2018	New spec
*A	6301235	MEH	10/04/2018	Changed LED usage to make project compatible with CY8CPROTO-063-BLE Kit.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.