

Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Objective

This example demonstrates the operation of Bluetooth Low Energy (BLE) Battery Service (BAS) using the BLE_PDL Component.

Overview

The design implements a Custom Profile in a GATT server and generic attribute profile (GAP) peripheral roles with the Battery and Device Information services. The Battery Service is used for software simulation of the battery level. The simulated battery level value is continuously changed from 2 to 20 percent.

Requirements

Tool: PSoC Creator™ 4.2 or later

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

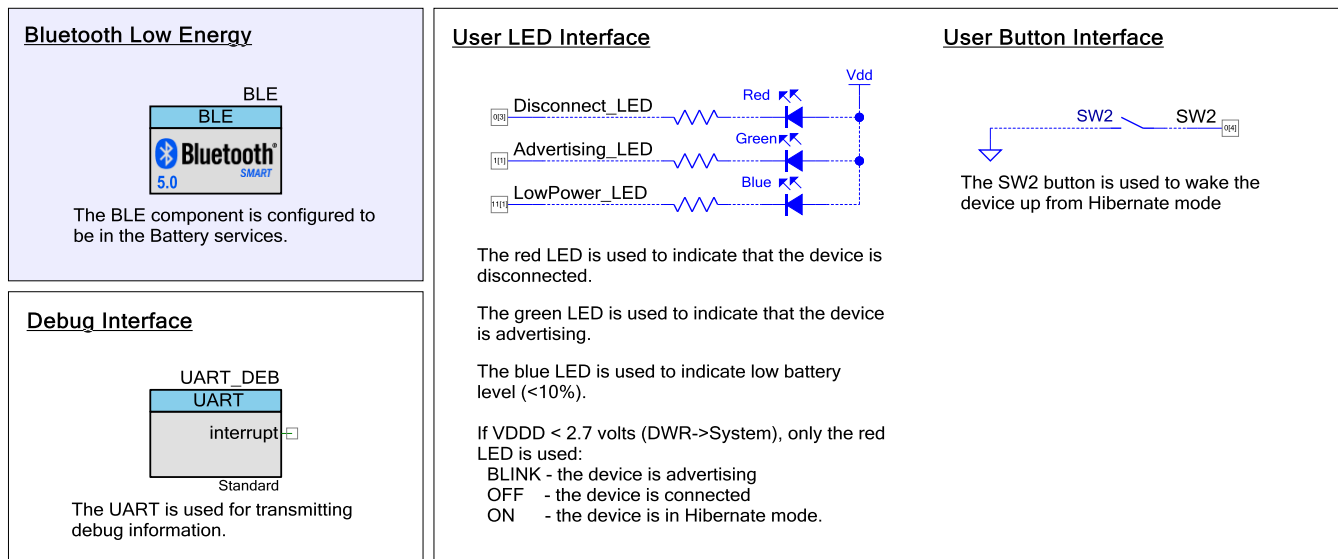
Associated Parts: All PSoC® 6 MCU with BLE Connectivity (PSoC 6 BLE) parts

Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

Design

Figure 1 shows the top design schematic.

Figure 1. BLE Battery Level Code-Example Schematic



After a start, the device performs the BLE Component initialization. Two callback functions are required in this project for the BLE operation:

- AppCallback() is required to receive generic events from the BLE stack.
- BasCallback() is required to receive events from the BAS service.

The CY_BLE_GAPP_StartAdvertisement() function is called after the CY_BLE_EVT_STACK_ON event to start advertising with the packet shown in Figure 7.

The green LED blinks to indicate that the device is advertising. The red LED will be turned ON after disconnection to indicate that no client is connected to the device. When a Client is connected successfully, red and green LEDs are turned OFF. When the measured battery voltage drops below the 10 percent limit, the blue LED is turned ON.

On an advertisement event timeout, if the device is not connected to a client, the device goes to Low-Power mode (Sleep mode) and waits for a **SW2** button press to wake up the device again and start advertising. While the device is in Hibernate mode, the red LED is turned ON.

Design Considerations

This code example is designed for the PSoC 6 MCU with BLE family and associated with CY8CKIT-062 PSoC 6 BLE. The design is easily portable to other PSoC MCU with BLE devices and kits, typically by just changing the device and components' pin assignments.

Hardware Setup

The code example was created for the [CY8CKIT-062 PSoC 6 BLE Pioneer Kit](#).

[Table 1](#) lists the pin assignment and connections required on the development board for the supported kits.

Table 1. Pin Assignment

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P5[0]	
\UART_DEB:tx\	P5[1]	
\UART_DEB:rts\	P5[2]	
\UART_DEB:cts\	P5[3]	
Advertising_LED	P1[1]	The green color of the RGB LED
Disconnect_LED	P0[3]	The red color of the RGB LED
LowPower_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

LED Behavior for V_{DD} voltage < 2.7 V

If the V_{DD} voltage is set to less than 2.7 V in the DWR settings of the **System** tab, only the red LED is used. The red LED blinks to indicate that the device is advertising. The red LED is OFF when a device is connected to a peer device. When the device is in Hibernate mode, the red LED stays ON.

Software Setup

Using UART for Debugging

A HyperTerminal program is required in a PC to receive debug information. If you don't have a HyperTerminal program installed, download and install any serial port communication program. Freeware such as HyperTerminal, Bray's Terminal, or PuTTY is available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the device manager program in your PC, find the COM port in which the kit is connected, and note the port number.
3. Open the HyperTerminal program and select the COM port to which the kit is connected.
4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the HyperTerminal configuration window. The default settings: Baud rate – 115200, Parity – None, Stop bits – 1 and Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART component in the project.
5. Start communicating with the device as explained in the project description.

The UART debugging can be disabled by setting the `DEBUG_UART_ENABLED` to `DISABLED` in `common.h`.

Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core and Dual core) in the BLE Peripheral Driver Library (PDL) examples.

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

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

The BLE examples' structure allows easy switching between different CPU cores options.

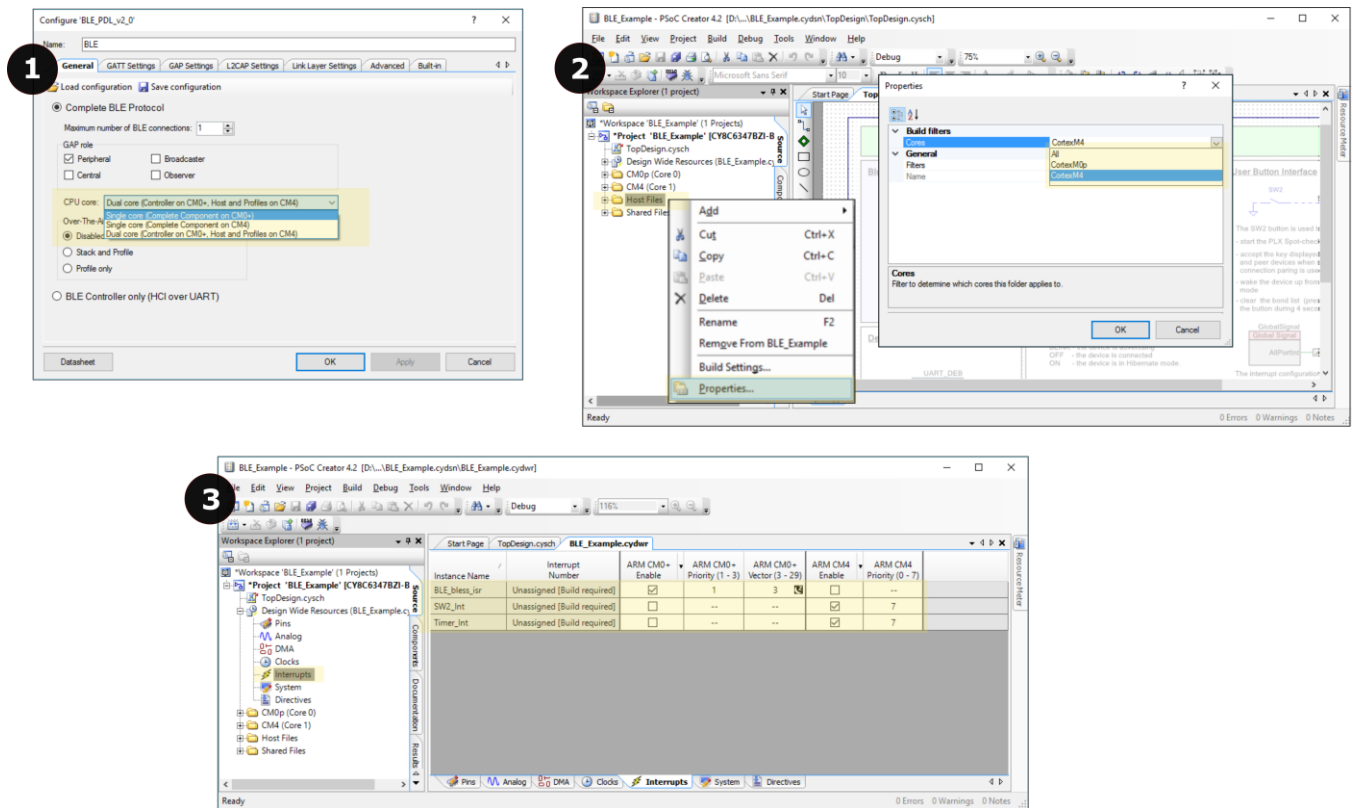
Important to remember:

- All application host-files must be run on the host core.
- The BLE Subsystem (BLESS) interrupt must be assigned to the core where the controller runs.
- All additional interrupts (SW2, MCWDT, etc.) used in the example must be assigned to the host core.

Do the following to switch the CPU cores usage:

1. In the BLE Component Customizer **General** tab, select appropriate CPU core option.
2. Change the core properties to CortexM4 or CortexC0p for the project folder Host Files based on the CPU core option selected in step 1. It should be:
 - 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
3. Assign the BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to 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 2. Steps for Switching the CPU Cores Usage



Components

Table 2 lists the PSoC Creator Components used in this example and the hardware resources used by each of the components.

Table 2. PSoC Creator Components List

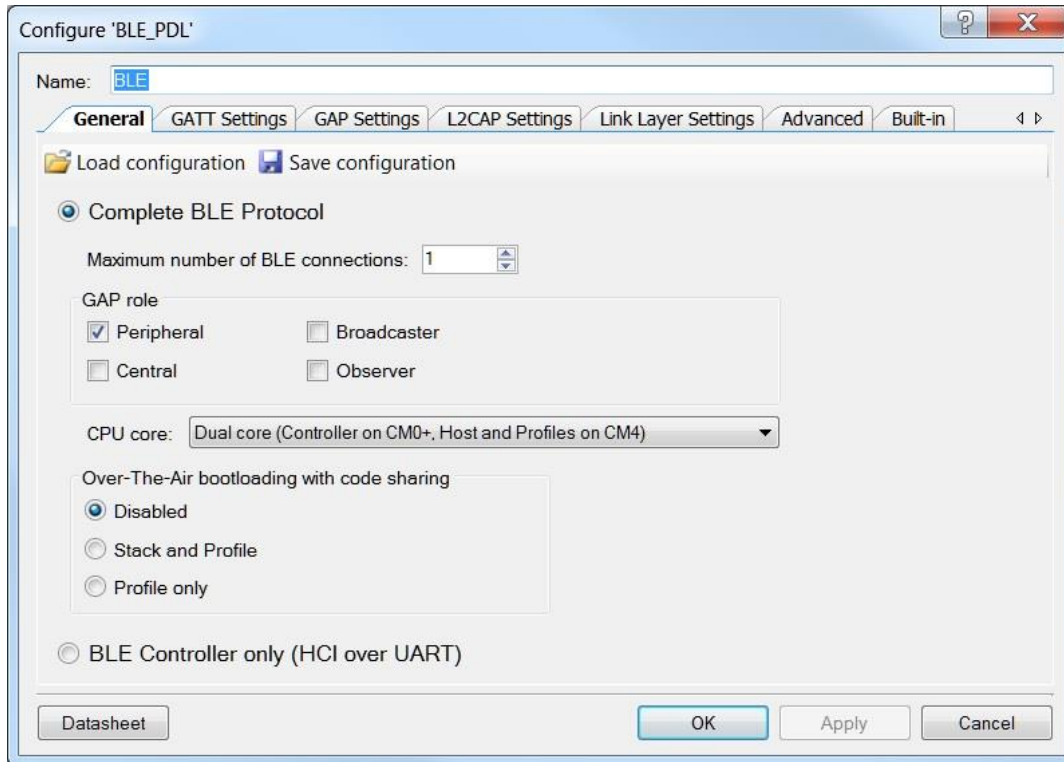
Component	Hardware Resources
UART_DEB	1 SCB
BLE	1 BLE, 1 Interrupt
SW2	1 pin
Disconnect_LED, Advertising_LED, LowPower_LED	3 pins

Parameter Settings

BLE_PDL Component

The BLE Component implements a custom profile in the peripheral GAP role with the Battery and Device Information services. The Device Information Service is added with the manufacture name string characteristic.

Figure 3. General Settings



The screenshot shows the 'Configure BLE_PDL' dialog box with the 'General' tab selected. The 'Name' field is set to 'BLE'. The 'Load configuration' and 'Save configuration' buttons are visible. The 'Complete BLE Protocol' radio button is selected. The 'Maximum number of BLE connections' is set to 1. The 'GAP role' section has 'Peripheral' selected. The 'CPU core' is set to 'Dual core (Controller on CM0+, Host and Profiles on CM4)'. The 'Over-The-Air bootloading with code sharing' section has 'Disabled' selected. The 'BLE Controller only (HCI over UART)' radio button is also visible. At the bottom, there are 'Datasheet', 'OK', 'Apply', and 'Cancel' buttons.

Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | GAP Settings | L2CAP Settings | Link Layer Settings | Advanced | Built-in

Load configuration | Save configuration

☒ 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)

Datasheet OK Apply Cancel

Figure 4. GATT Settings

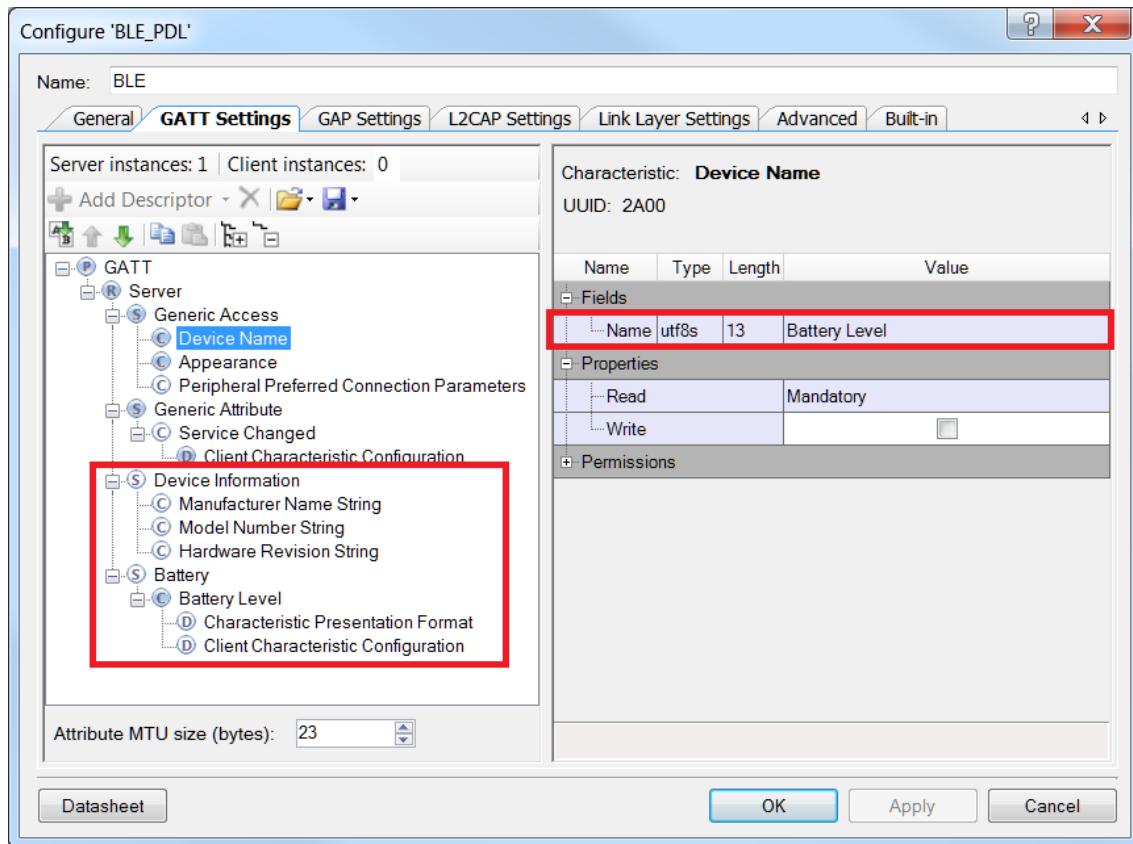
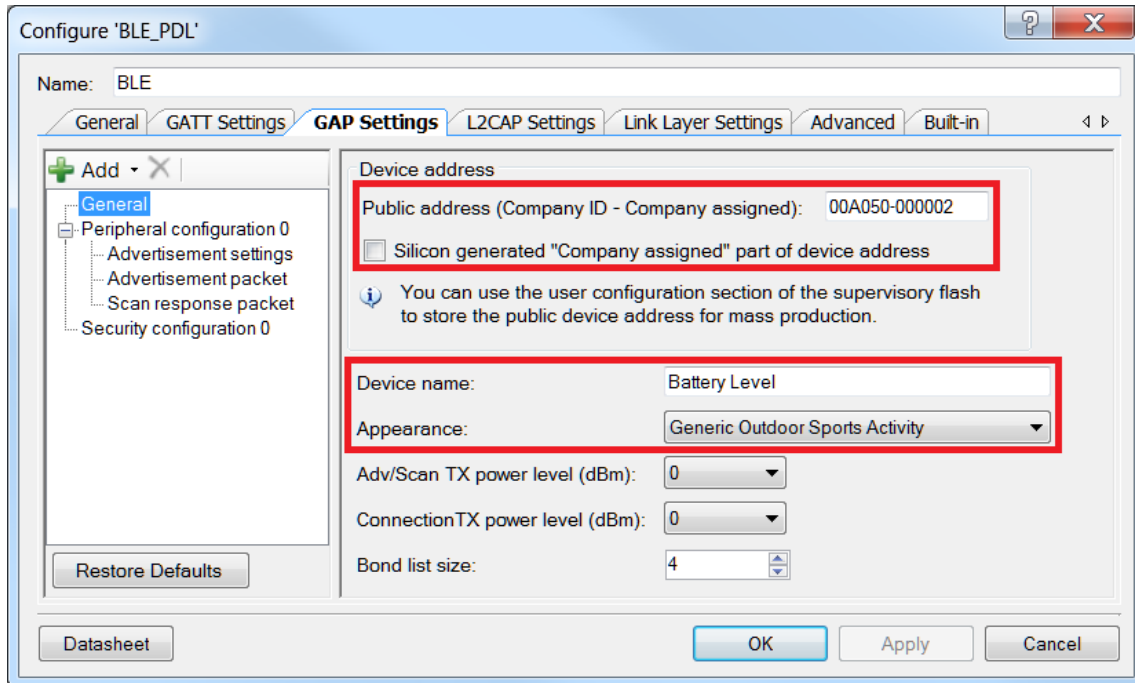


Figure 5. GAP Settings



Configure 'BLE_PDL'

Name: BLE

General | **GATT Settings** | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

+ Add - X

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
 - Security configuration 0

Restore Defaults

Datasheet

Device address

Public address (Company ID - Company assigned): 00A050-000002

☐ 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: Battery Level

Appearance: Generic Outdoor Sports Activity

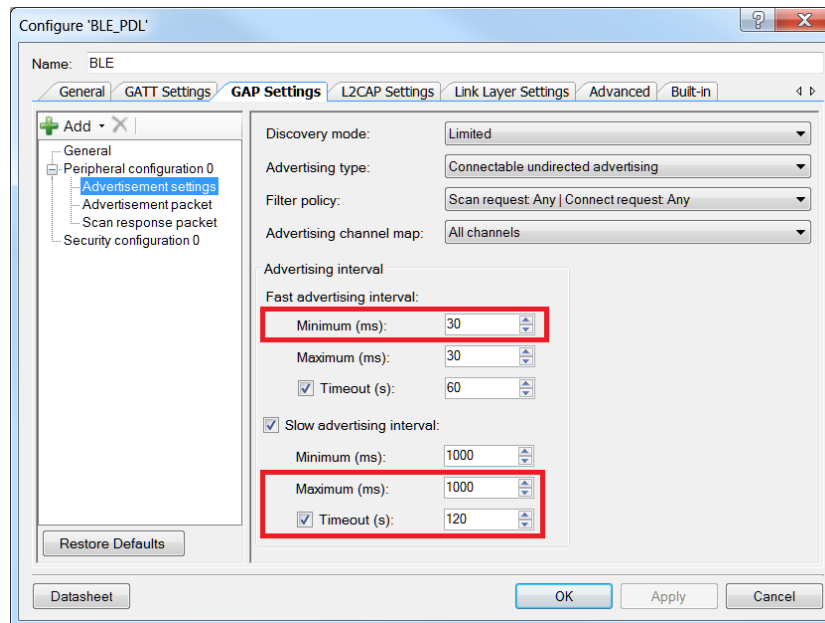
Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

Bond list size: 4

OK Apply Cancel

Figure 6. GAP Settings: Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

+ Add - X

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
 - Security configuration 0

Restore Defaults

Datasheet

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): 30

Maximum (ms): 30

☒ Timeout (s): 60

☒ Slow advertising interval:

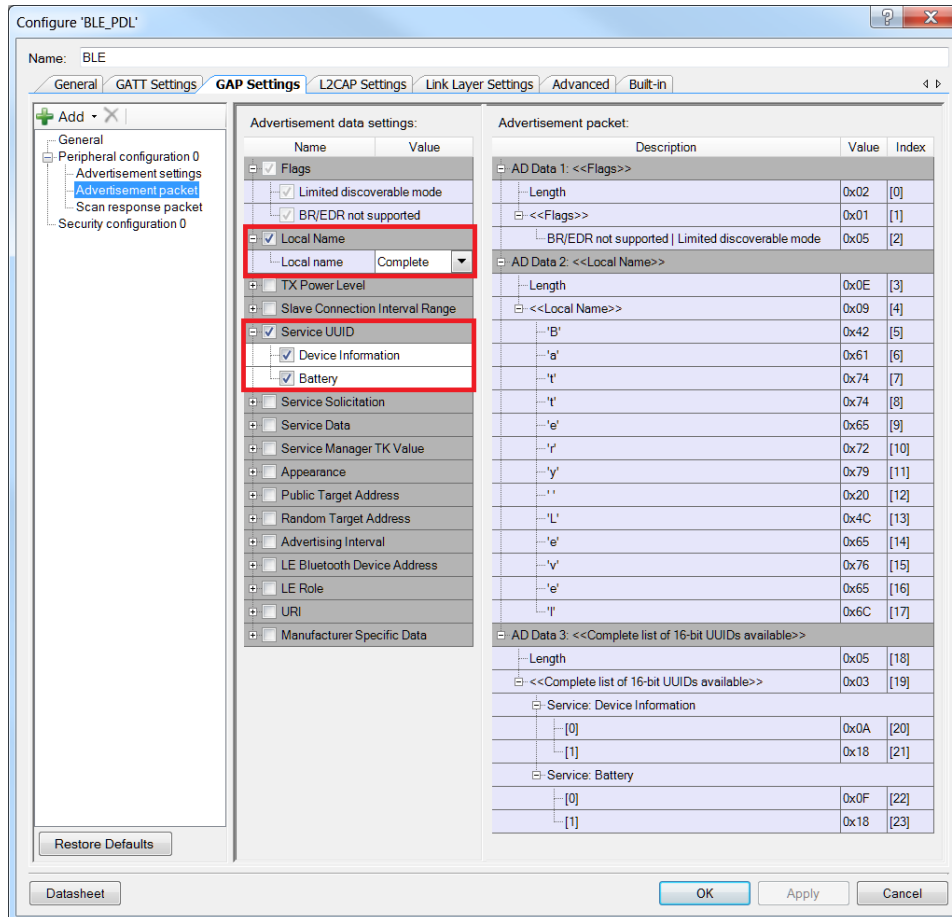
Minimum (ms): 1000

Maximum (ms): 1000

☒ Timeout (s): 120

OK Apply Cancel

Figure 7. GAP Settings: Advertisement Packet



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet**
 - Scan response packet
 - Security configuration 0

Restore Defaults

Datasheet

OK Apply Cancel

Advertisement data settings:

Name	Value
<input checked="" type="checkbox"/> Flags	
<input checked="" type="checkbox"/> Limited discoverable mode	
<input checked="" type="checkbox"/> BR/EDR not supported	
<input checked="" type="checkbox"/> Local Name	Complete
<input type="checkbox"/> TX Power Level	
<input type="checkbox"/> Slave Connection Interval Range	
<input checked="" type="checkbox"/> Service UUID	
<input checked="" type="checkbox"/> Device Information	
<input checked="" type="checkbox"/> Battery	
<input type="checkbox"/> Service Solicitation	
<input type="checkbox"/> Service Data	
<input type="checkbox"/> Service Manager TK Value	
<input type="checkbox"/> Appearance	
<input type="checkbox"/> Public Target Address	
<input type="checkbox"/> Random Target Address	
<input type="checkbox"/> Advertising Interval	
<input type="checkbox"/> LE Bluetooth Device Address	
<input type="checkbox"/> LE Role	
<input type="checkbox"/> URI	
<input type="checkbox"/> 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	0x0E	[3]
<<Local Name>>	0x09	[4]
'B'	0x42	[5]
'a'	0x61	[6]
't'	0x74	[7]
't'	0x74	[8]
'e'	0x65	[9]
't'	0x72	[10]
'y'	0x79	[11]
' '	0x20	[12]
'L'	0x4C	[13]
'e'	0x65	[14]
'v'	0x76	[15]
'e'	0x65	[16]
't'	0x6C	[17]
AD Data 3: <<Complete list of 16-bit UUIDs available>>		
Length	0x05	[18]
<<Complete list of 16-bit UUIDs available>>	0x03	[19]
Service: Device Information		
[0]	0x0A	[20]
[1]	0x18	[21]
Service: Battery		
[0]	0x0F	[22]
[1]	0x18	[23]

Figure 8. Security Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
 - Security configuration 0**

Restore Defaults

Datasheet

OK Apply Cancel

Security mode: Mode 1

Security level: Unauthenticated pairing with encryption

I/O capabilities: No Input No Output

Keypress notifications: No

Bonding requirement: No Bonding

Encryption key size (bytes): 16

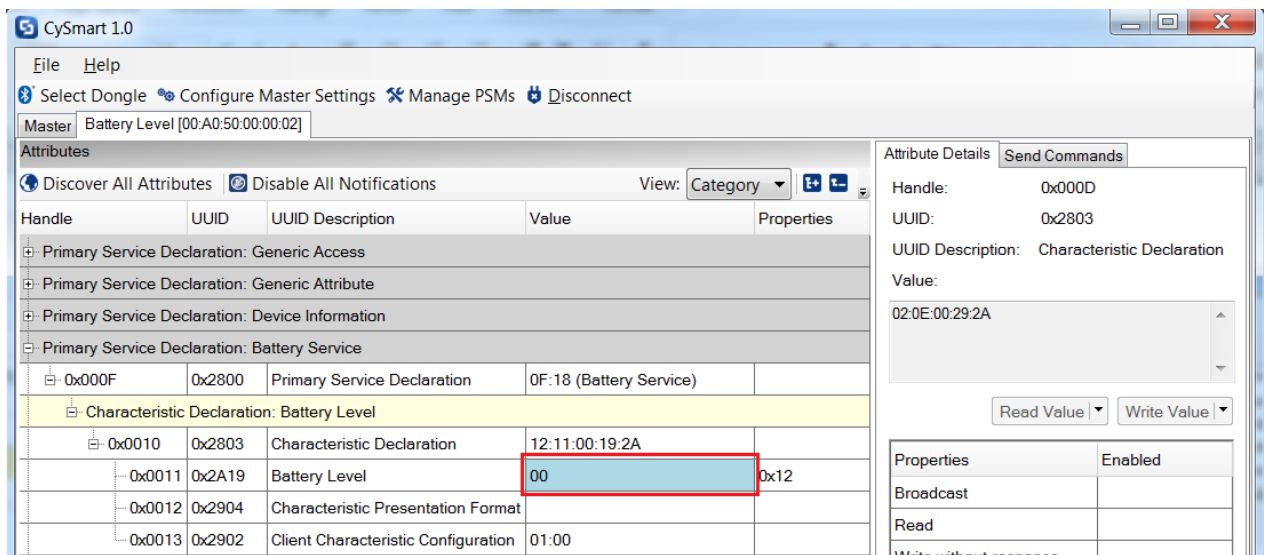
Operation

You can use the CySmart Central Emulation Tool on a [Windows PC](#), [Android](#), or iOS BLE 4.0-compatible device as a client to connect to this project.

To use the CySmart Windows application as a client:

1. Build and program the BLE Battery Level project into the [CY8CKIT-062 PSoC 6 BLE Pioneer Kit](#).
2. Connect the CySmart BLE dongle to a USB port on the PC.
3. Launch the CySmart app and select the connected dongle in the dialog window.
4. Press **SW1** to reset the development kit and to start advertising.
5. Click **Start Scan** to discover available devices.
6. Select **Battery Level** from the list of available devices and connect to it.
7. Click **Pair**, then **Discover All Attributes**, then **Read All Characteristics**, and finally **Enable All Notifications** in the CySmart app.
8. Notice the values of the battery level descriptors.

Figure 9. CySmart Windows App



For more information about the CySmart Central Emulation tool, see [CySmart User Guide](#).

To use the CySmart mobile app as a Battery Service client:

1. Launch the CySmart mobile app on the Android or iOS BLE-compatible device.
2. Swipe down to refresh the list of BLE devices.
3. Connect to the **Battery Level** device and select **Battery Service** from the service list.
4. Notice that the value of the battery level is continuously changing from 2 to 20 percent. Also, you can notice that the blue color LED will be ON when a measured battery level is less than 10 percent.

Figure 10. CySmart iOS App
Recognized Battery Service

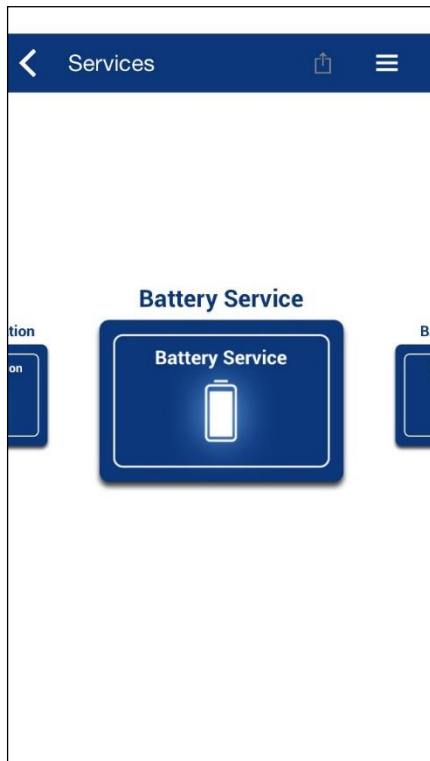
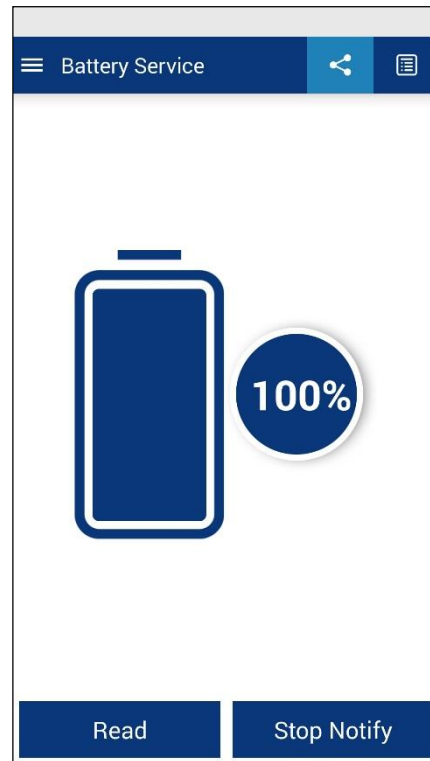


Figure 11. CySmart Android App
Shows Battery Level



Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with BLE Connectivity, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-Core CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – BLE Test and Debug Tool		CySmart is a BLE host emulation tool for Windows PCs. The tool provides an easy-to-use GUI to enable the user to test and debug their BLE 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 Programmable System-on-Chip		PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual (TRM)
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		

Document History

Document Title: CE215119 - BLE Battery Level with PSoC 6 MCU with BLE Connectivity

Document Number: 002-15119

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5968175	NPAL	11/21/17	New spec

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](#)

Cypress Developer Community

[Forums](#) | [WICED IOT Forums](#) | [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

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

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. 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.