CE217646 - BLE Wireless Power Transfer with PSoC 6 MCU with BLE Connectivity

Objective

This example demonstrates the Wireless Power Transfer service in the Power Transmitter Unit and Power Receiver Unit role.

Overview

The design demonstrates the Wireless Power Transfer Profile operation of the BLE Component. The example consists of two projects: the Wireless Power Transmitter (GATT Client) and Wireless Power Receiver (GATT Server).

This example demonstrates the communication between Power Receiver Unit (PRU) and Power Transmitter Unit (PTU) in the Wireless Power Transfer systems. The PTU Central device supports connection with up to four PRU Peripheral devices. The PRU simulates the power receiver data and reports the simulated data to a PTU using the Wireless Power Transfer Service (WPTS).

Requirements

Tool: PSoC Creator™ 4.2

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

Associated Parts: All PSoC 6 BLE parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

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

1. Connect the BLE Pioneer Kit to the computer's USB port.
2. Connect the BLE Dongle to one of the USB ports on the computer.

LED Behavior

If the V_{DDD} voltage is set to less than 2.7 V in the DWR settings **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.

LED behavior for V_{DDD} Voltage > 2.7 volts is described in **Operation** section.

Software Setup

BLE Host Emulation Tool

This example requires the CySmart application. Download and install either the CySmart Host Emulation Tool PC application or the CySmart app for iOS or Android. You can test behavior with any of the two options, but the CySmart app is simpler. Scan one of the following QR codes from your mobile phone to download the CySmart app.

iOS

Android

Terminal Tool

This example uses a terminal window. You must have terminal software, such as Tera Term, or PuTTY.

Operation

The Wireless Power Receiver device (PRU) can be connected to any BLE (4.0 or later) - compatible device configured as GAP Central role and GATT Client which supports the Wireless Power Profile. The green LED blinks while the device is advertising. The red LED is turned ON after disconnection to indicate that no Client is connected to the device. When the Client connects successfully, the red and green LEDs are turned OFF. When a Client enables charging, the blue LED turns ON.

The project simulates voltage changes at the charge/battery port (Vrect) and writes the result to the PRU Dynamic Parameter characteristic. It increases when the charger is enabled by the PTU and decreases when charging is complete. Press the **SW2** button to increase the simulation step. The simulation interval value is set to 1 second.

The Wireless Power Transmitter project (PTU) automatically connects to the PRU devices. The PTU uses the WPT Service handle from the advertising packet for quick PRU discovery instead of the classic device discovery procedure.

When a connection is established, the PTU automatically initiates a basic information exchange procedure, i.e., sends a read request to the PRU Static Parameter characteristic, a write request to PTU Static Parameter characteristic, enables notification of the Alert characteristic, and enables charging. A read request to PRU Dynamic Parameter characteristic is sent every second.

The red LED blinks while the device is scanning. The red LED is turned ON after disconnection to indicate that no Peripheral is connected to the device. When the device connects successfully with any Peripheral, the red LED is turned OFF.

Advertising packets received during the scanning procedure from Peripheral devices are parsed and filtered. Only packets with WPT service-specific data are handled and showed in the debug terminal.

The example project uses the UART Component for displaying debug information and for sending commands through a terminal emulator app. The commands are the procedures, which the user can perform.

Table 1. List of Commands

Command	Description
's'	Start discovery procedure.
'1'	Enable notifications for the Alert characteristic.
'2'	Enable indications for the Alert characteristic.
'3'	Disable notifications and indications for the Alert characteristic.
'4'	Send Read request for the PRU Static Parameter characteristic.
'5'	Send Read request for the PRU Dynamic Parameter characteristic.
'6'	Send “Enable Charging” command to the PRU control characteristic.
'7'	Send “Disable Charging” command to the PRU Control characteristic.
'8'	Enable sequential read of the PRU Dynamic Parameter characteristic.
'9'	Disable sequential read of the PRU Dynamic Parameter characteristic.

The above list is prompted to the terminal emulator when ‘h’ is entered in the app.

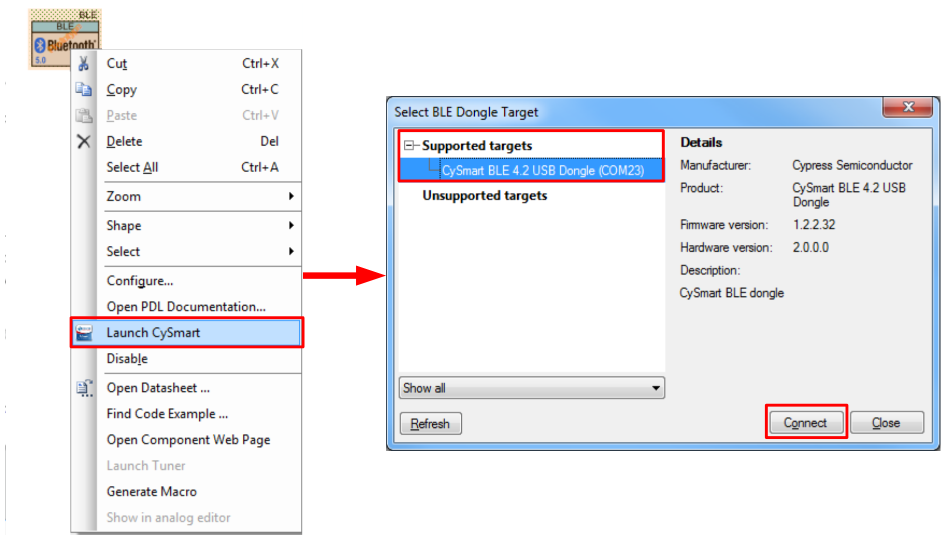
Use the **‘6’** and **‘7’** commands to send the **Enable** and **Disable Charging** commands to the PRU Control characteristic. The PRU device will indicate charging with the blue LED.

Operation Steps

1. Plug the CY8CKIT-062-BLE kit board into your computer’s USB port.
2. Open a terminal window and perform following configuration: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. Build the Wireless Power Receiver project and program it into the PSoC 6 MCU 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 blue LED blinks while the device is advertising, and the output in the terminal window.
5. Do the following to test example, using the CySmart Host Emulation Tool application:
   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 on 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 1.

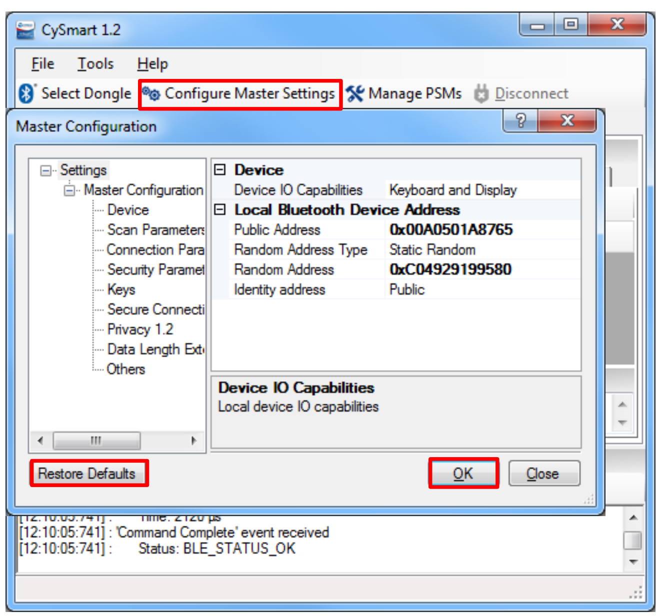
Figure 1. CySmart BLE Dongle Selection



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.

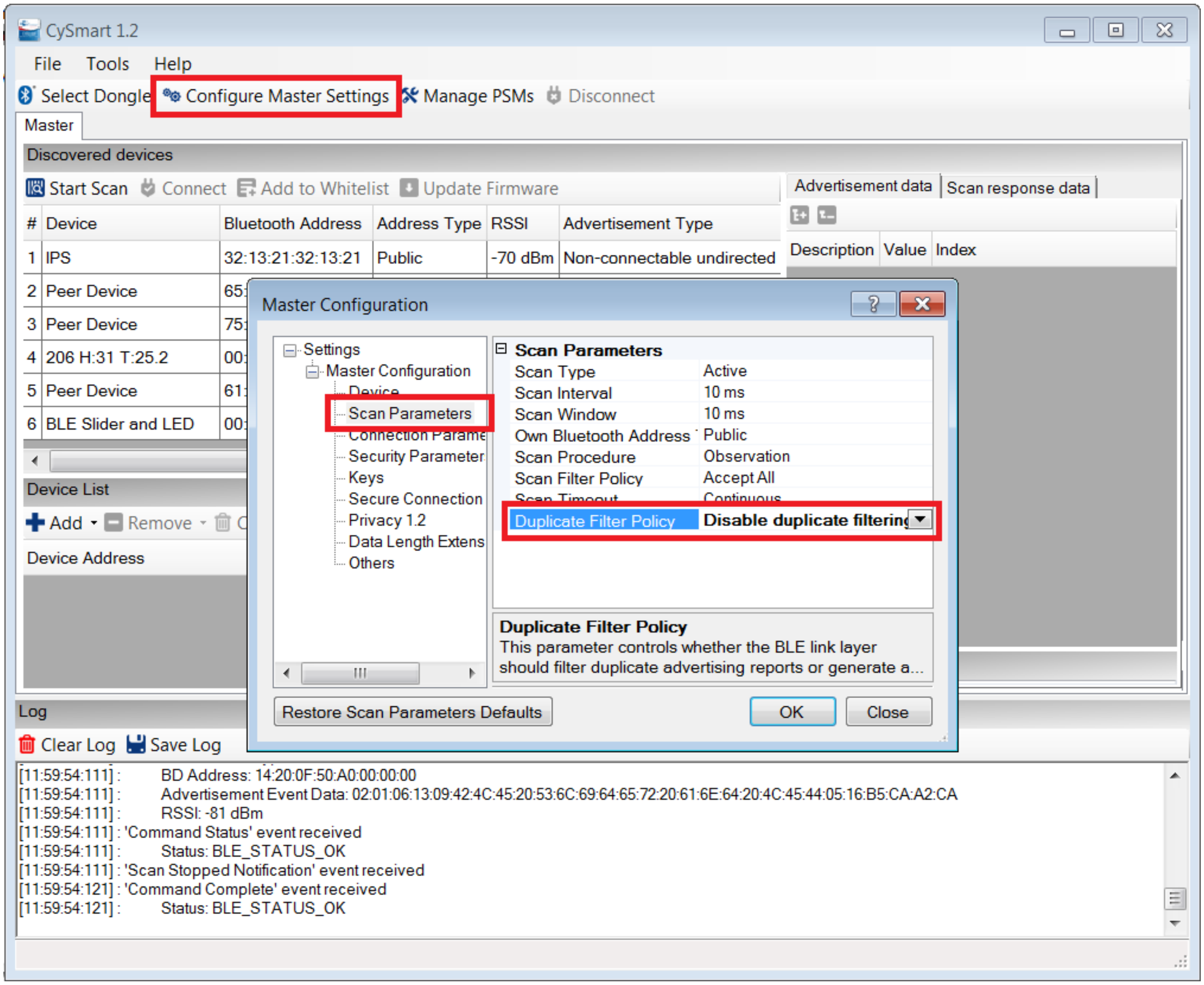
   d. Select **Configure Master Settings** and then click **Restore Defaults**, as Figure 2 shows. Then click **OK**.

Figure 2. CySmart Master Settings Configuration



   e. Set the **Duplicate Filter Policy = Disable duplicate filtering** in the **Master Configuration > Scan parameters** window. See Figure 3.

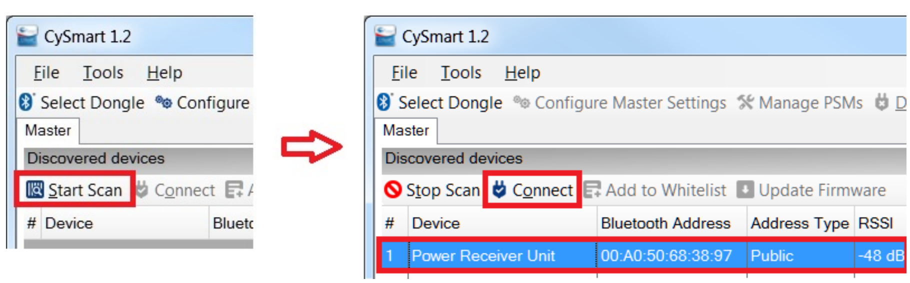
Figure 3. Master Configuration → Scan Parameters



f. Press the reset switch on the Pioneer Kit to start BLE advertisement if no device is connected or device is in Hibernate mode (red LED is on). Otherwise, skip this step.

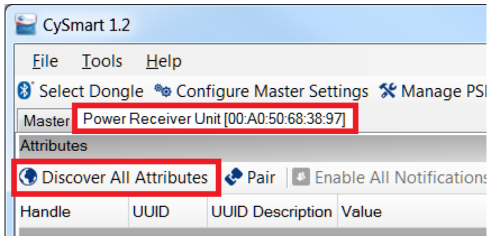
g. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name (configured as **Power Receiver Unit**) should appear in the Discovered devices list, as Figure 4 shows. Select the device and click **Connect** to establish a BLE connection between the CySmart Host Emulation Tool and your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

Figure 4. CySmart Device Discovery and Connection



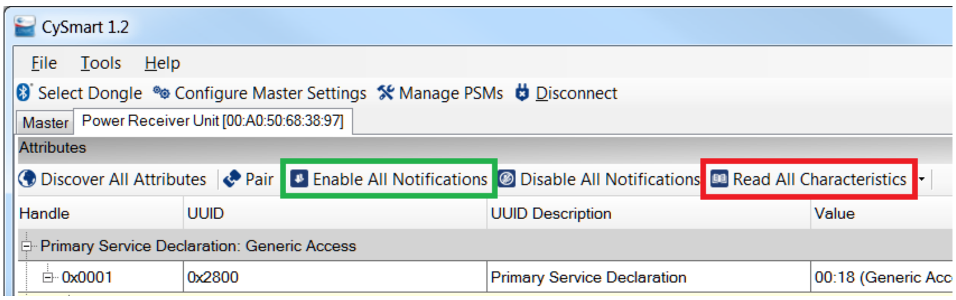
h. Once connected, switch to the '**Power Receiver Unit**' device tab and '**Discover all Attributes**' on your design from the CySmart Host Emulation Tool, as shown in Figure 5.

Figure 5. CySmart Attribute Discovery



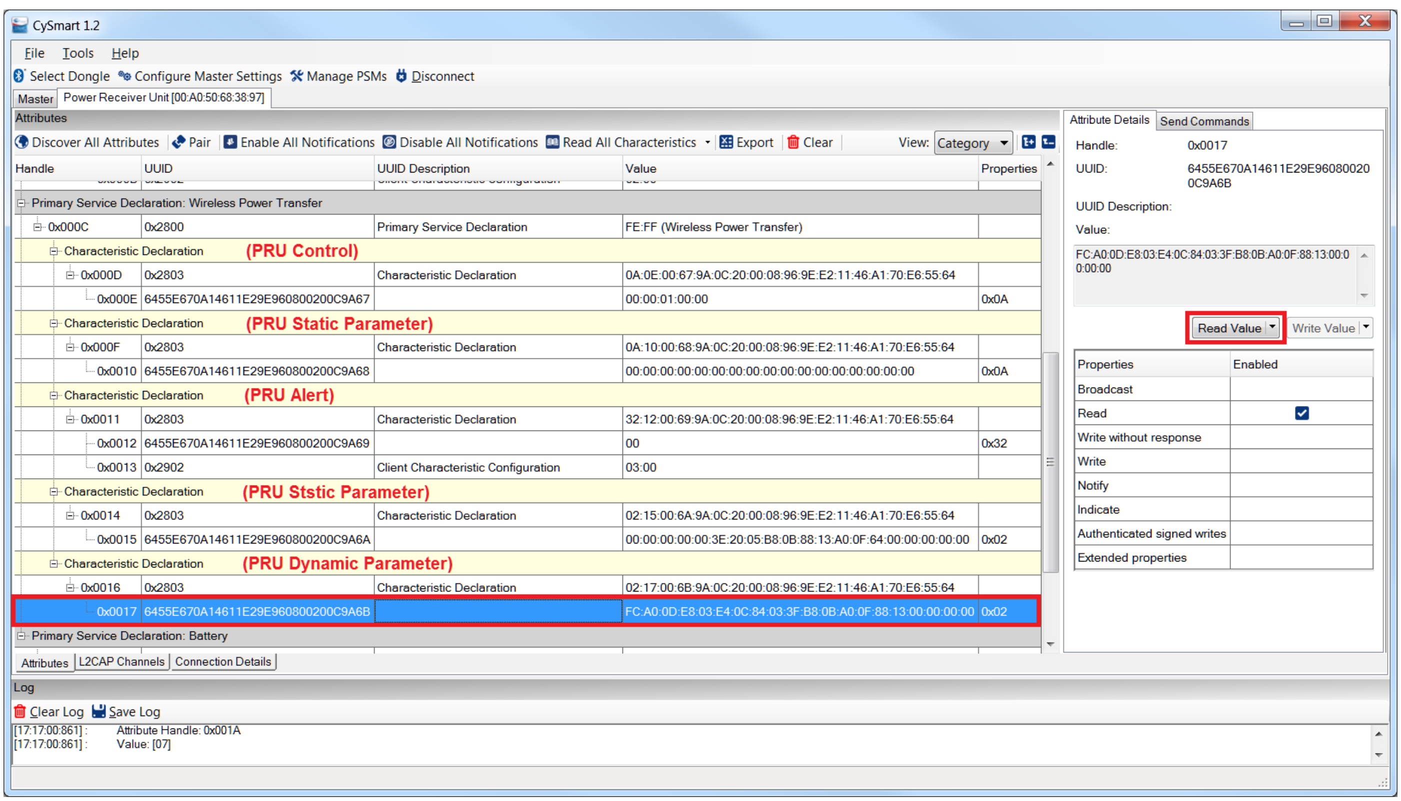
i. Click **Pair** after discovery finishes, then **Enable All Notifications** and **Read All Characteristics**. Observe the received characteristic values as shown in Figure 6.

Figure 6. CySmart Enable All Notification and Read All Characteristics



j. Select the **PRU Dynamic Parameter** characteristic and press Read Value to observe the simulated values. Refer to the A4WP Wireless Power Transfer System Baseline System Specification for details on characteristic structure.

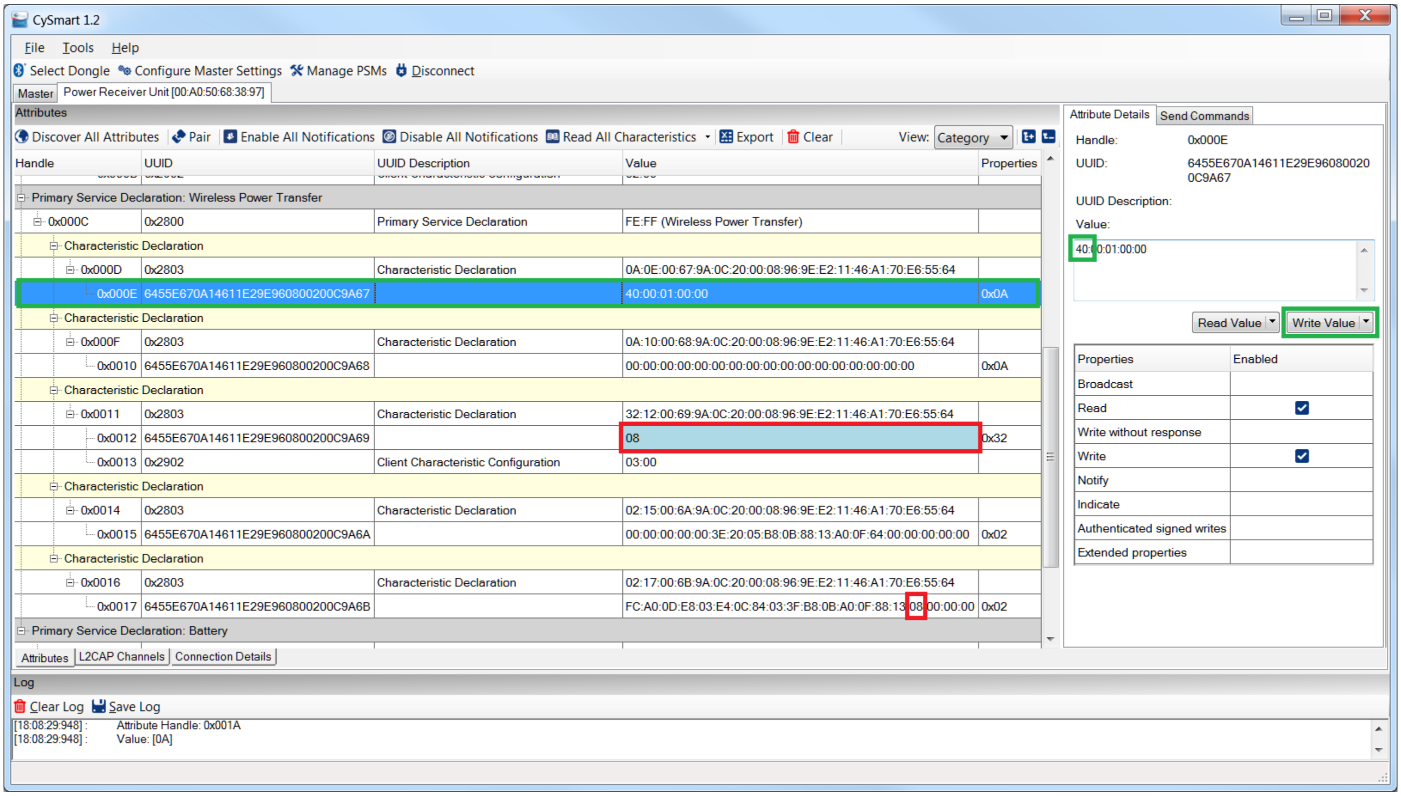
Figure 7. CySmart Windows App



k. Select the **PRU Control** characteristic, write the 0x40 value (charge enable) to the first byte and press **Write Byte**. Observe that the blue LED indicates that the PRU device received the command.

l. After some time, observe the Notification of the Alert characteristic received with value 0x08 (Charge Complete). Read the PRU Dynamic Parameter characteristic and observe the same value in the PRU Alert position (17th byte):

Figure 8. CySmart Windows App



m. If you have some problems with the usage of the CySmart app, refer to the CySmart User Guide.

6. The CySmart mobile app (Android/iOS) does not have Wireless Power Service implementation, but still can be used in GATT Data Base mode for test this example. You can repeat test flow for CySmart mobile app in step 5. Refer to Android and iOS CySmart User Guide.

7. The code example ships with the UART debug port enabled. To disable it, set the macro DEBUG_UART_ ENABLED in *common.h* to DISABLED and rebuild the code.

8. The BLE Wireless Power Transmitter project is intended to work in a pair with BLE Wireless Power Receiver. Do the following to test Wireless Power Receiver project, using Wireless Power Transmitter project:

   a. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.

   b. Open a terminal window and perform following configuration: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.

   c. Build the BLE Wireless Power Transmitter project and program it into the PSoC 6 MCU 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.

   d. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.

   e. Open a terminal window and perform following configuration: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.

   f. Build the BLE Wireless Power Receiver project and program it into the PSoC 6 MCU 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.

g. Two projects send log messages through UART. After starting, the PTU project logs Advertising and Scan response reports from the PRU, for example:

```
Advertisement report: peerBdAddr -00 a0 50 e1 61 a8 , rssi - -36 dBm, data - 02 01 05
07 16 fe ff 0c 00 a0 00
```

h. The PTU automatically connects with the advertised PRU. Observe the initialization procedure. The blue LED on PRU indicates charging. After Vrect reaches VrectHighDyn, the PRU sends the simulated Alert Notification with the Complete Charge status and switches off the blue LED.

i. The output of the debug serial port looks like the sample below:

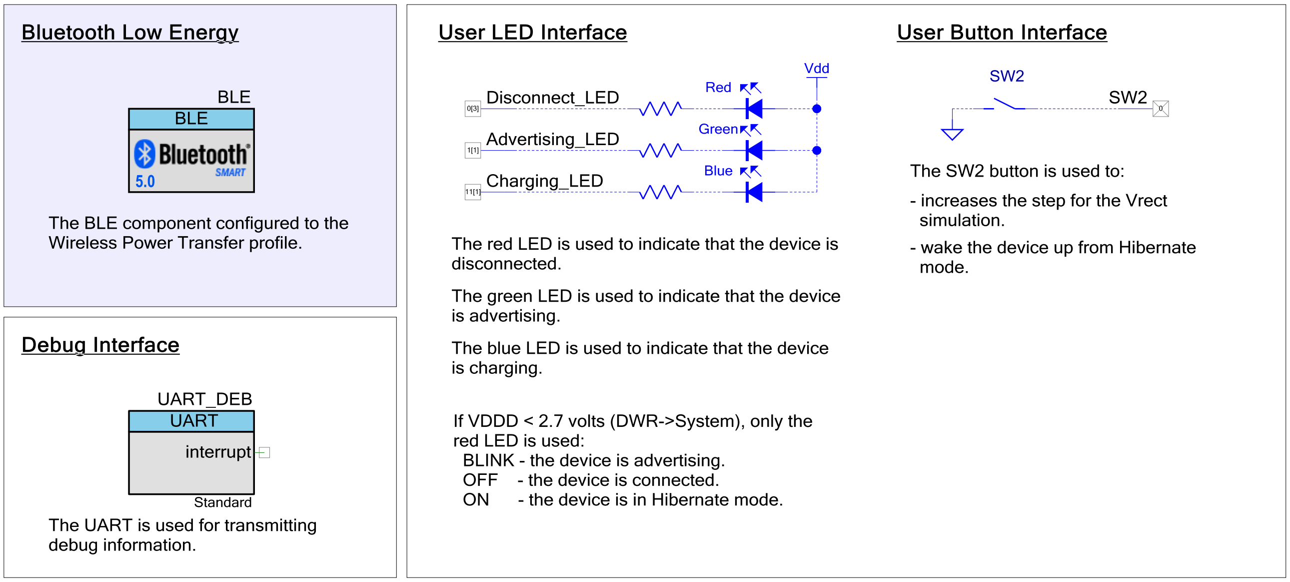
The PTU example log:
BLE Wireless Power Transmitter Code Example
CY_BLE_EVT_STACK_ON, StartScanning
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GAPC_SCAN_START_STOP, state: 2
Advertisement report: peerBdAddr -00 a0 50 06 00 00 , rssi - -47 dBm, data - 02 01 05 07 16 fe ff 0c 00 a0 00
Connecting to the device (address - 00a050060000)
CY_BLE_EVT_GATT_CONNECT_IND: 3, 3
Get PRU Static Parameter char value, attId: 3, apiResult: 0
CY_BLE_EVT_GAP_DEVICE_CONNECTED: bdHandle=3, connIntv = 53 ms
Device 3<--PRU_STATIC_PARAMETER: flags: 0, protocol rev: 0, category: Category 0 , information: 0 , hardware rev: 62 , firmware rev: 32 Prect_max: 500 mW, Vrect_min_static: 3000 mV, Vrect_high_static: 5000 mV, Vrect_set: 4000 mV,
Device 3-->Set PTU Static Parameter char value, apiResult: 0
Device 3<--CY_BLE_EVT_WPTSC_WRITE_CHAR_RESPONSE: charIndex =1
Device 3-->Enable Alert Notification, apiResult: 0
Device 3<--CY_BLE_EVT_WPTSC_WRITE_DESCR_RESPONSE charIndex =2
Device 3-->Send Connection Parameter Update Request to Peripheral, apiResult: 0
Device 3-->Set PRU Control char (enable charging), apiResult: 0
Device 3<--CY_BLE_EVT_WPTSC_WRITE_CHAR_RESPONSE: charIndex =0
CY_BLE_EVT_GAP_CONNECTION_UPDATE_COMPLETE: bdHandle=3, connIntv = 250 ms
N of Devices: 1; #3 Charge=1, Vrect: 3996 mV;
N of Devices: 1; #3 Charge=1, Vrect: 3998 mV;
N of Devices: 1; #3 Charge=1, Vrect: 4000 mV;
N of Devices: 1; #3 Charge=1, Vrect: 4003 mV;
N of Devices: 1; #3 Charge=1, Vrect: 4006 mV;

The PRU example log:
BLE Wireless Power Receiver Code Example
CY_BLE_EVT_STACK_ON, StartAdvertisement
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a050060000
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
CY_BLE_EVT_GATT_CONNECT_IND: 3, 3
CY_BLE_EVT_GAP_DEVICE_CONNECTED: connIntv = 53 ms
Simul Vrect: 3999 mV
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 15
PTU_STATIC_PARAMETER: flags: c0, power: watts, maxLoadResistance: 50 ohms, supported devices number: 1 , class: Class4 , hardware rev: 62, firmware rev: 32, protocol rev: 0
CY_BLE_EVT_WPTS_NOTIFICATION_ENABLED: char: 2
CY_BLE_EVT_WPTS_INDICATION_DISABLED: char: 2
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 17
Simul Vrect: 3998 mV
PRU_CONTROL: enables: 40, Output - DISABLE,Charge indicator -ENABLE ,Power:Max permission: 0, Permitted,timeSet 0 ms
CY_BLE_EVT_CONNECTION_UPDATE_COMPLETE: connIntv = 250 ms
SimulBatteryLevelUpdate: 3
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 17
Simul Vrect: 3999 mV
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 17
Simul Vrect: 4000 mV
Simul Vrect: 4001 mV
SimulBatteryLevelUpdate: 4
Simul Vrect: 4002 mV

Design and Implementation

Figure 9 shows Wireless Power Receiver design schematic.

Figure 9. BLE Wireless Power Receiver Code Example Schematic



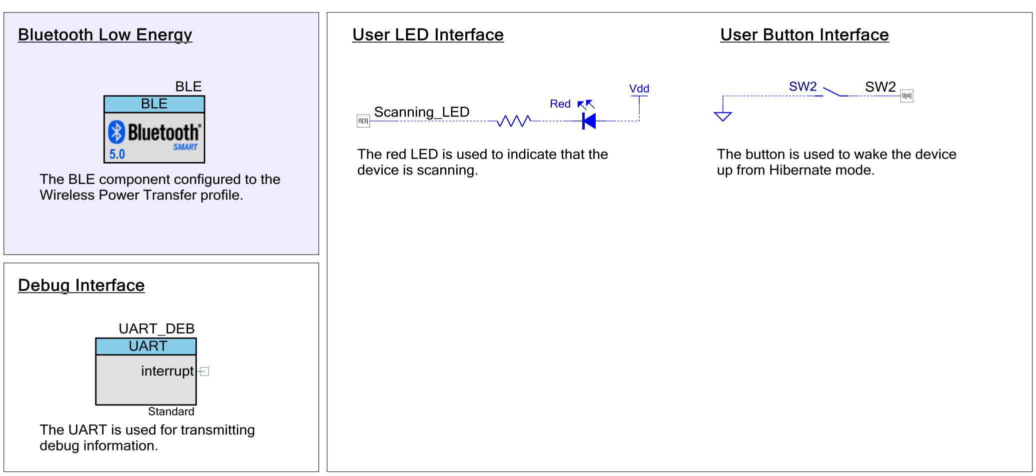
The project demonstrates the core functionality of the BLE Component configured as the PRU with an additional Battery Service (BAS). The purpose of the BAS service is to demonstrate the battery level measurement possibilities along with measurement of other parameters for a wireless power-transfer system.

After a startup, the device performs BLE Component initialization. In this project, two callback functions are required for the BLE operation. The `AppCallBack()` callback function is required to receive generic events from the BLE stack and the service-specific-callback `WptsCallBack()` is required for WPTS service-specific events. The `CY_BLE_EVT_STACK_ON` event indicates 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 after a 180-second advertising period expires.

On an advertisement timeout, the system remains in Hibernate mode. Press the mechanical button **SW2** to wake the system and start re-advertising. While connected to the Client and between the connection intervals, the device is put into Deep Sleep mode.

Figure 10 shows the top Wireless Power Transmitter design schematic.

Figure 10. BLE Wireless Power Transmitter Code-Example Schematic



The project demonstrates the core functionality of the BLE Component configured as a PTU.

After a startup, the device performs the BLE Component initialization. In this project, three callback functions are required for BLE operation. The `AppCallBack()` callback function is required to receive generic events from the BLE stack and the service-specific callback `WptsCallBack()` is required for WPTS service-specific events. The `CY_BLE_EVT_STACK_ON` event indicates a successful initialization of the BLE stack. After this event is received, the Component starts scanning. The BLE Component stops scanning after a 180-second scanning period expires.

On the scanning timeout, the system remains in Hibernate mode. Press the mechanical button **SW2** to wake the system and start re-scanning.

Pin assignments

Table 2 lists the pin assignments and connections required on the development board for supported kits for Wireless Power Receiver project.

Table 2. Pin Assignment for Wireless Power Receiver project

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]	
Disconnect_LED	P0[3]	The red color of the RGB LED
Advertising_LED	P1[1]	The green color of the RGB LED
Charging_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Table 3 lists the pin assignments and connections required on the development board for supported kits for Wireless Power Transmitter project.

Table 3. Pin Assignment for Wireless Power Transmitter project

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]	
Scanning_LED	P0[3]	The red color of the RGB LED
SW2	P0[4]	

Components and Settings

Table 4 lists the PSoC Creator Components used in Wireless Power Receiver project, how they are used in the design, and the non-default settings required so they function as intended.

Table 4. PSoC Creator Components used in Wireless Power Receiver project

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE component configured to the Wireless Power Transfer profile.	Refer to Parameter Settings for Wireless Power Receiver project
Digital Input Pin	SW2	This pin is used to connection the user button (**SW2**).	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Disconnect_LED Advertising_LED Charging_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default

Table 5 lists the PSoC Creator Components used in Wireless Power Transmitter project, how they are used in the design, and the non-default settings required so they function as intended.

Table 5. PSoC Creator Components used in Wireless Power Transmitter project

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE component configured to the Wireless Power Transfer profile.	Refer to Parameter Settings for Wireless Power Transmitter project
Digital Input Pin	SW2	This pin is used to connection the user button (**SW2**).	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Scanning_LED	This GPIO is configured as firmware-controlled digital output pin that control LED.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default

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

Parameter Settings for Wireless Power Receiver project

The BLE Component is configured as the WPT Power Receiver Unit (GATT Server) in the GAP Peripheral role. The PRU Advertisement packet is configured to the connectable undirected advertising type with WPTS specific-service data. The scan response packet includes a complete local name.

Figure 11. General Settings

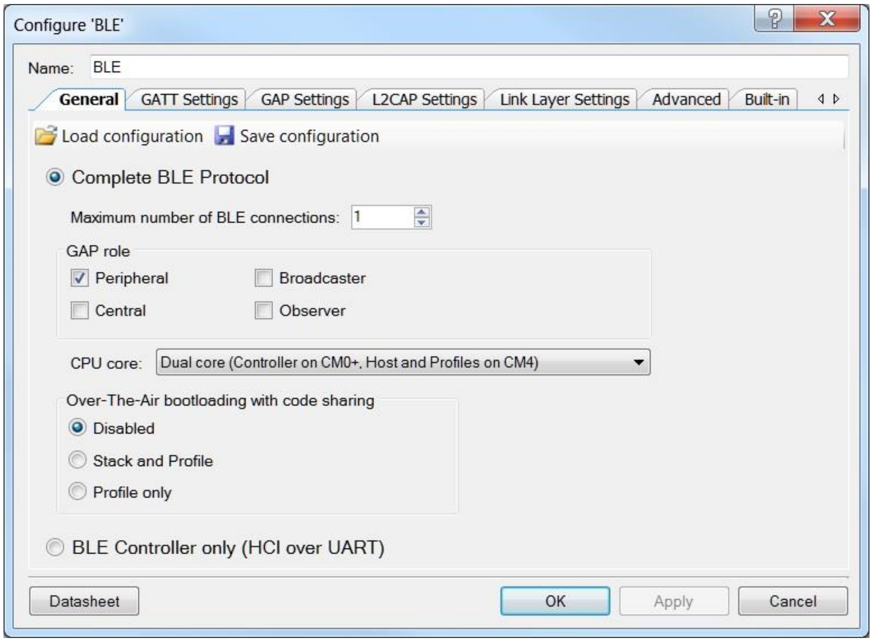


Figure 12. GATT Settings

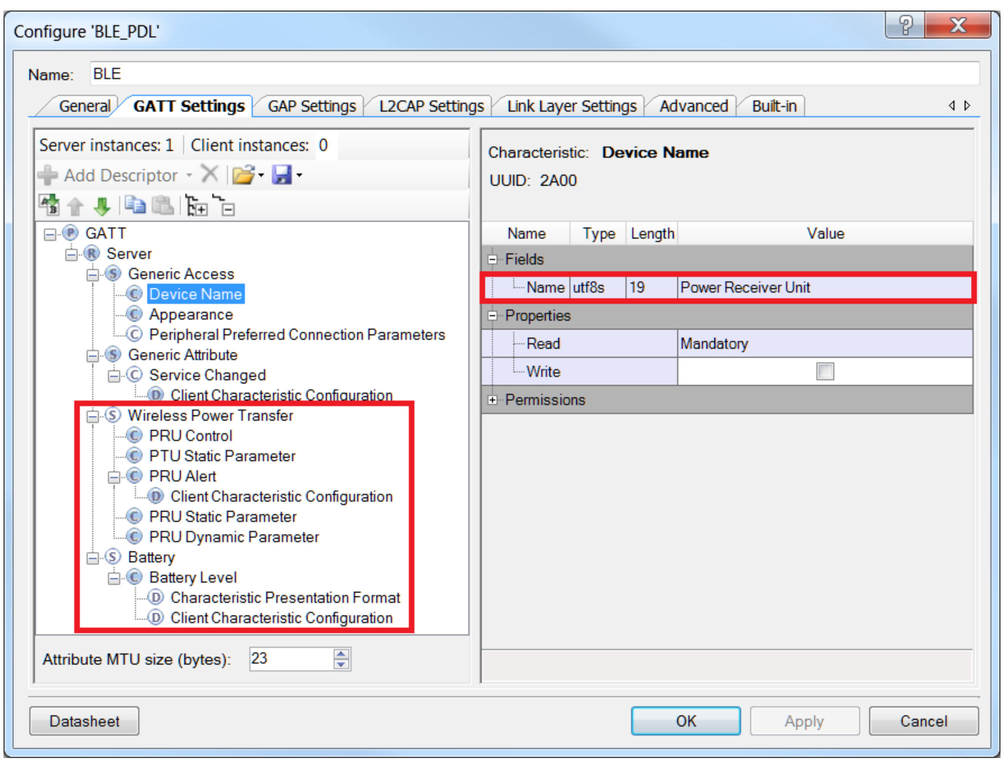


Figure 13. GAP Settings

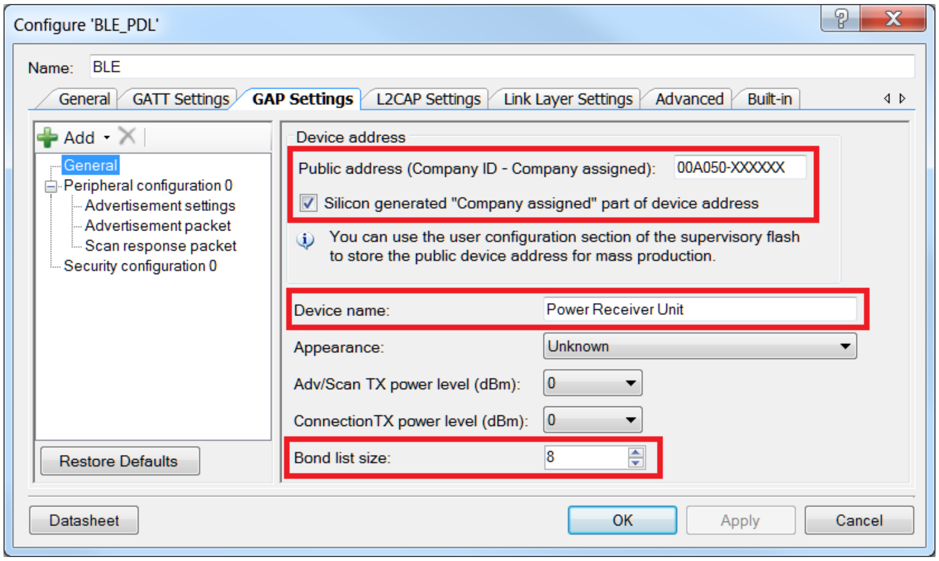


Figure 14. GAP Settings: Advertisement Settings

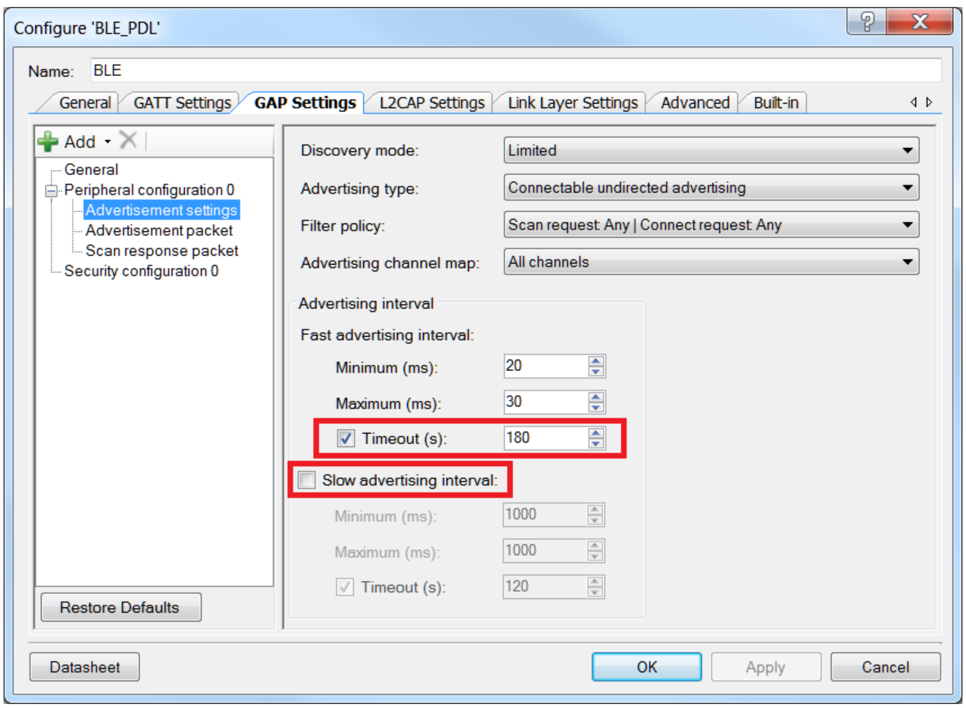


Figure 15. GAP Settings: Advertisement Packet

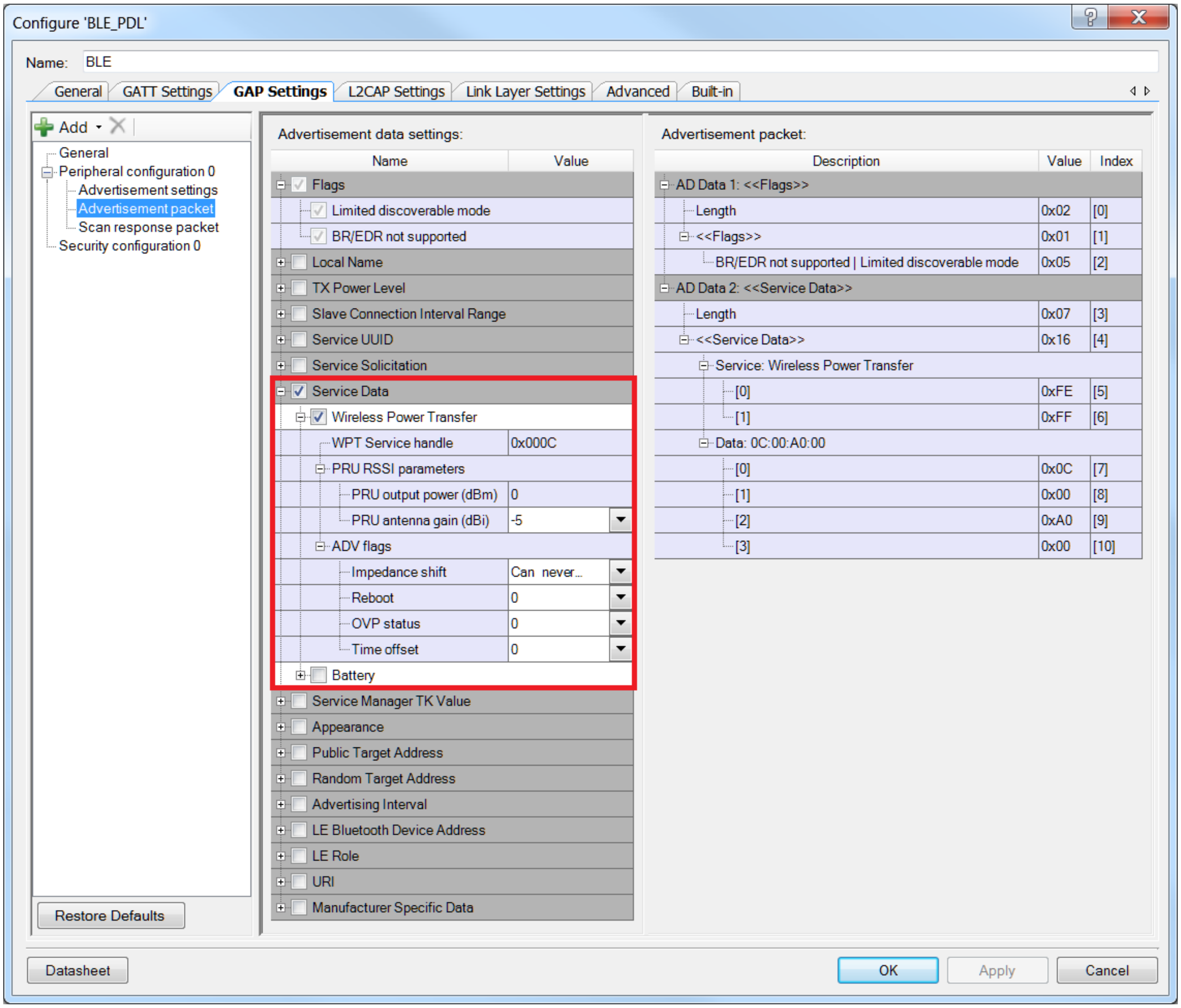
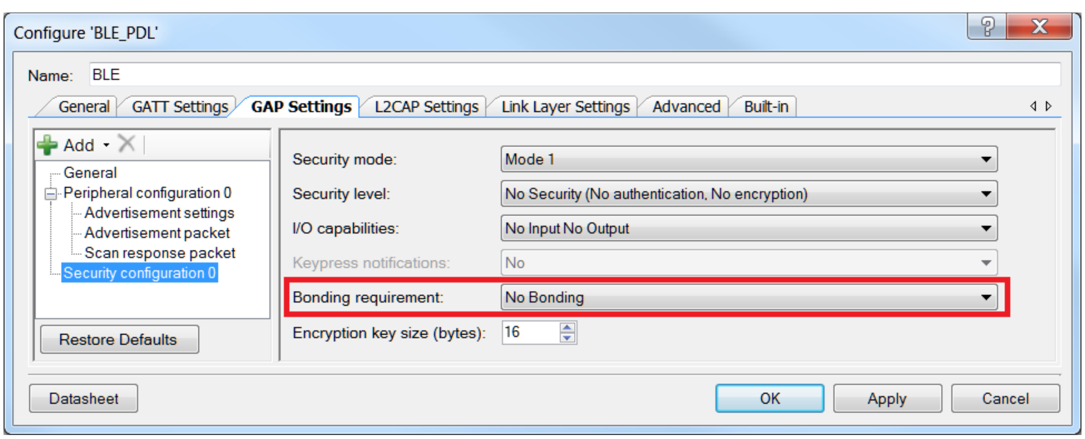


Figure 16. Security Settings



Parameter Settings for Wireless Power Transmitter project

The BLE Component is configured as the WPT Power Transmitter Unit (GATT Client) in the GAP Central role.

Figure 17. General Settings

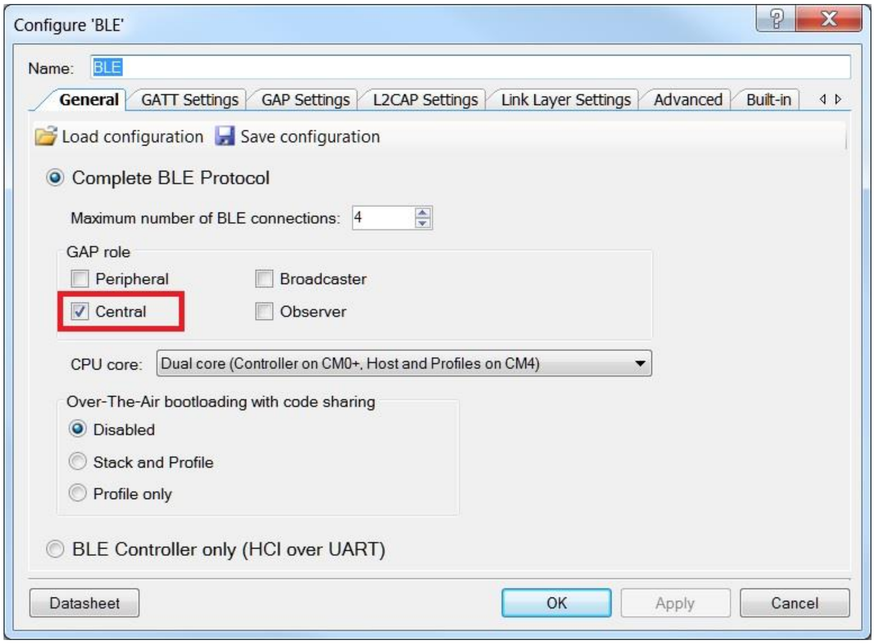


Figure 18. GATT Settings

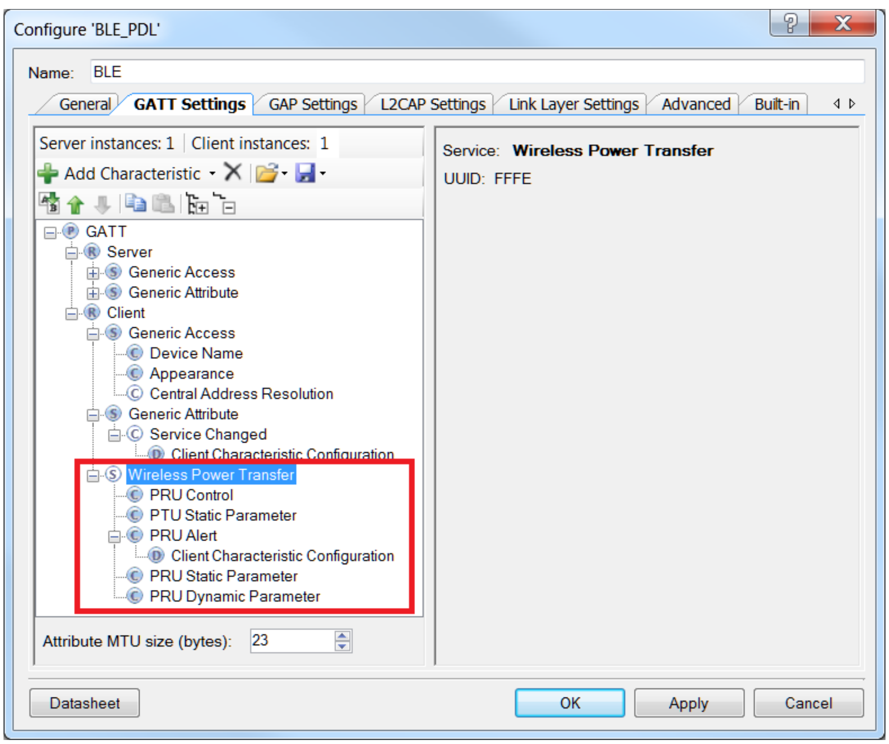


Figure 19. GAP Settings

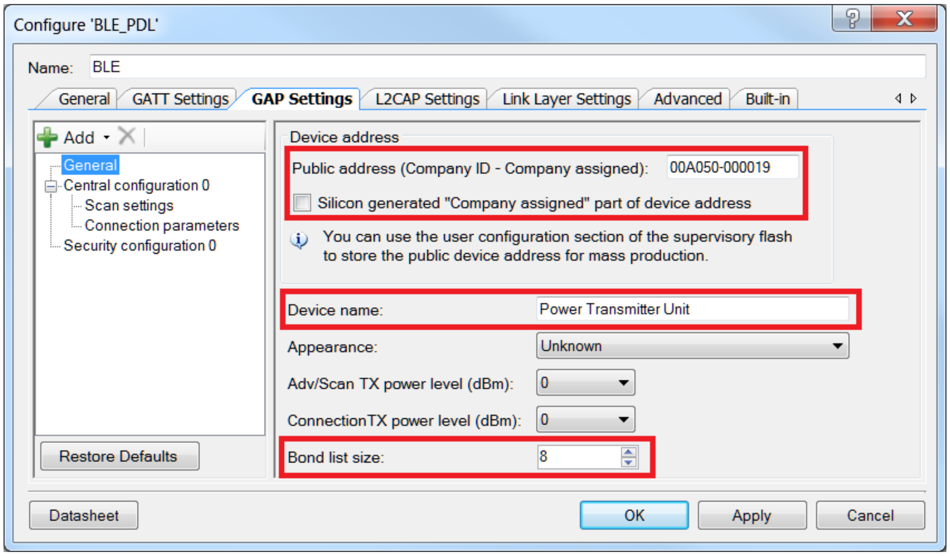


Figure 20. GAP Settings: Scan Settings

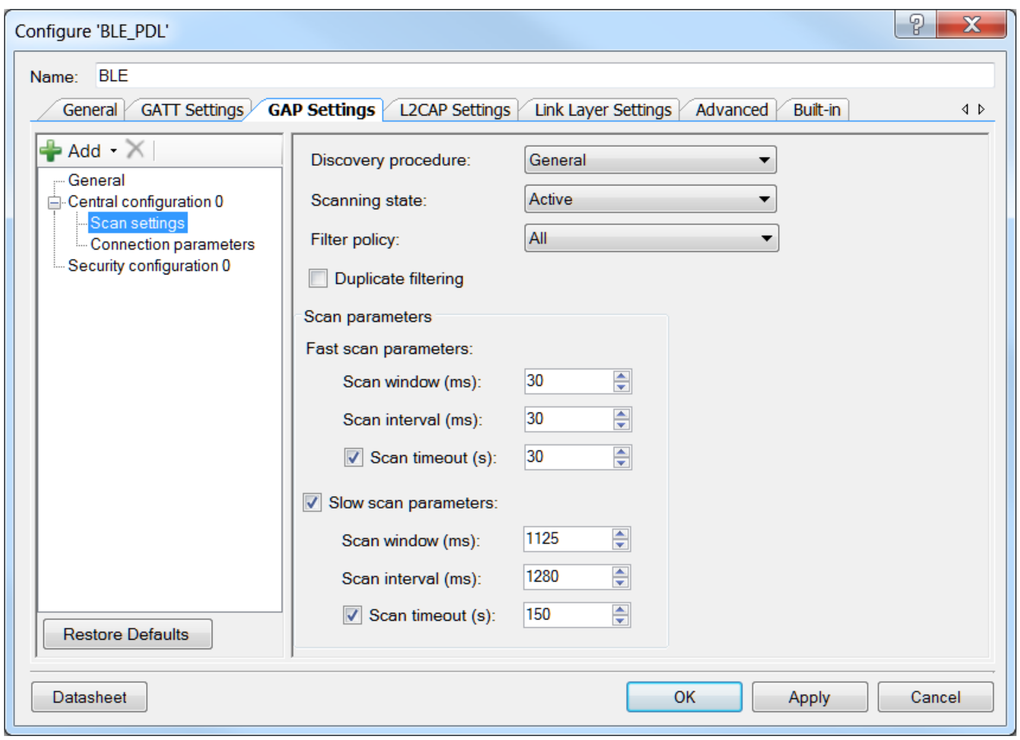
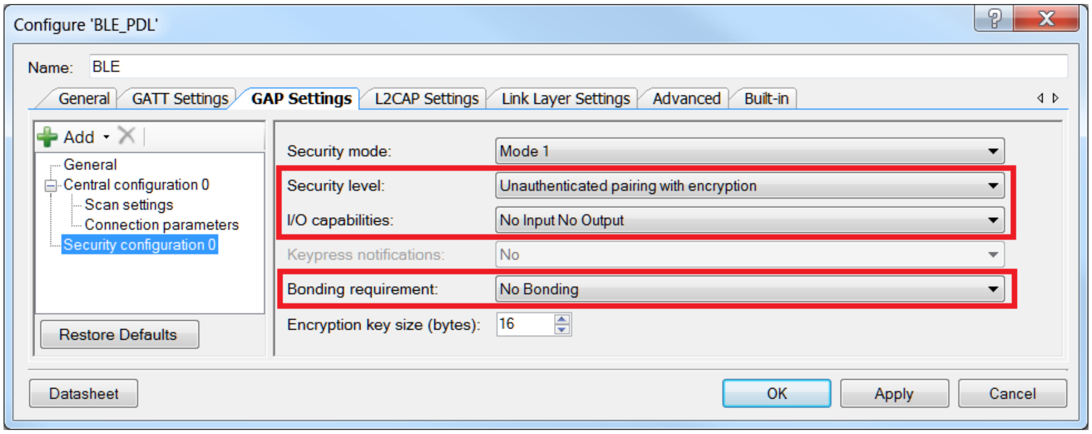


Figure 21. Security Settings



Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core / Dual core) in the BLE 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+ 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.

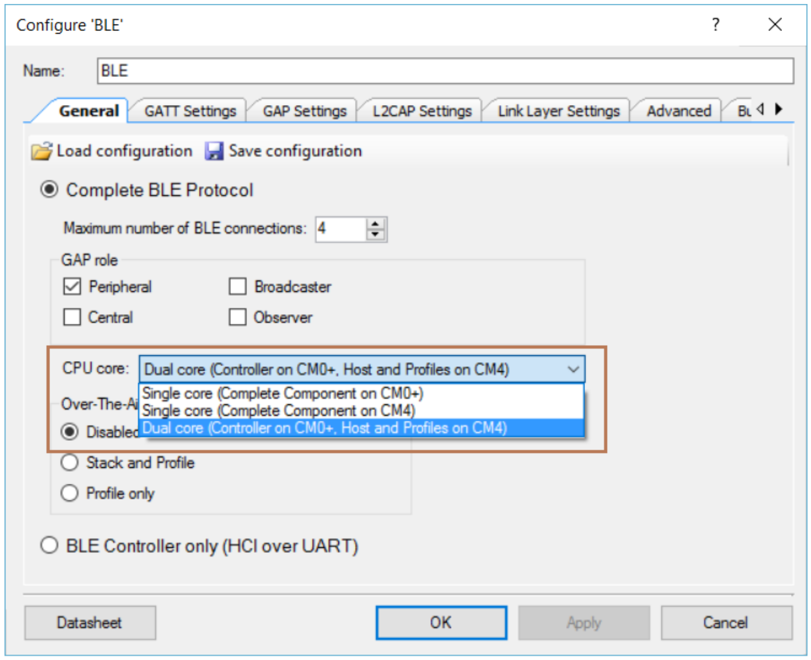
The BLE example structure allows easy switching between different CPU cores options. Important to remember:

- 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, etc.) used in the example must be assigned to the host core.

Steps for switching the CPU Cores usage:

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

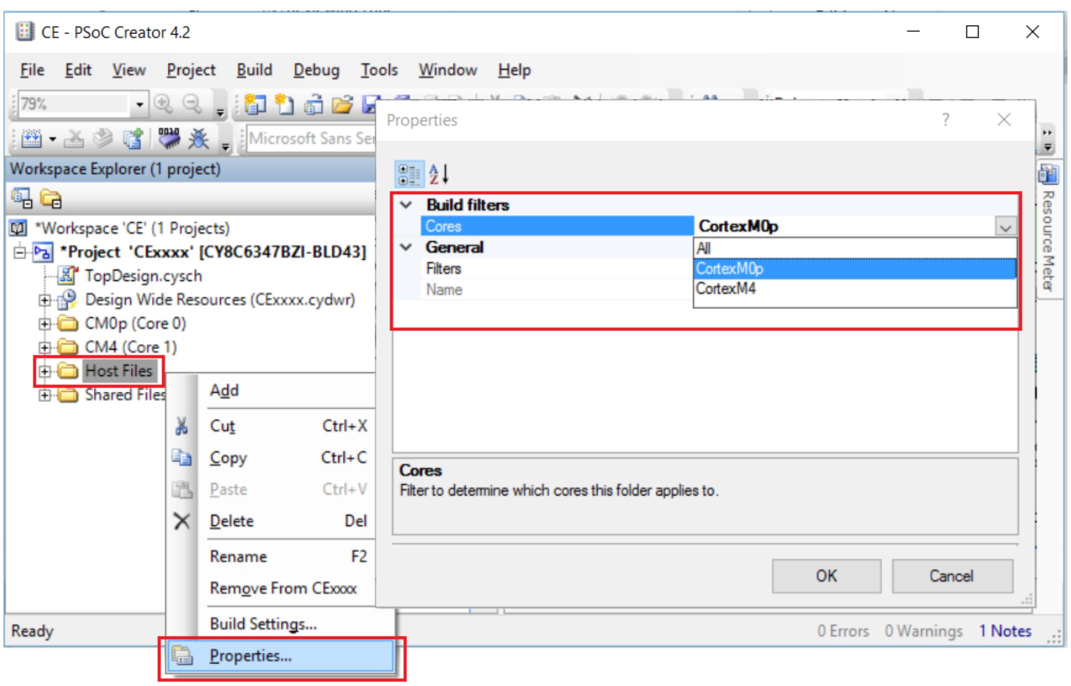
Figure 22. Select CPU Core



2. Identify the CPU on which host files will run. In the workspace explorer panel, right-click **Host Files**, choose **Properties**. Set the **Cores** property corresponding to the CPU core chosen in Step 1, as shown in Figure 23.

- 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

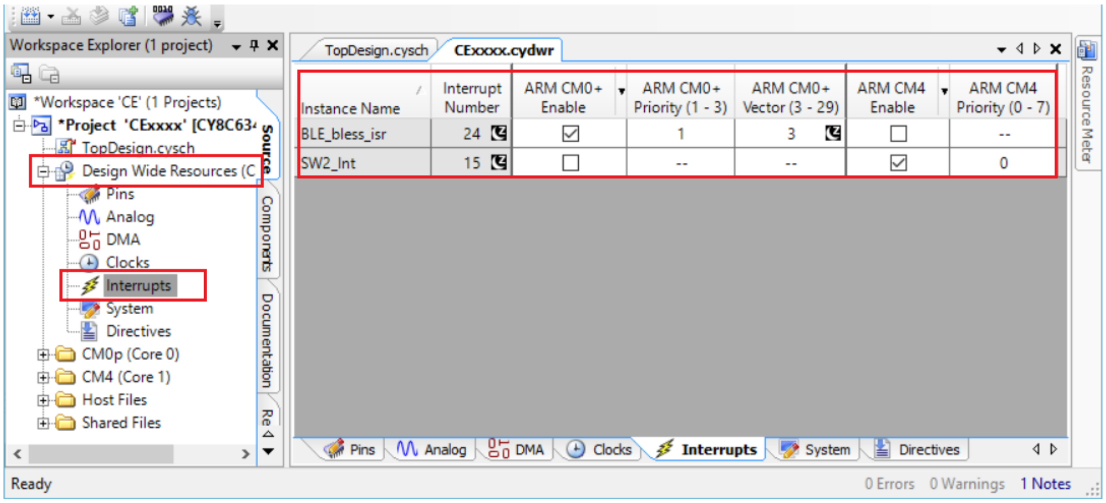
Figure 23. Change Core Properties



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 24. Assign Interrupts



Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU 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		

Document History

Document Title: CE217646 - BLE Wireless Power Transfer with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17646

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6092397	NPAL	06/07/2018	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 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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