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.

CE217633 - BLE Blood Pressure Sensor with PSoC 6 MCU with BLE Connectivity

Objective

This example project demonstrates the Bluetooth Low Energy (BLE) Blood Pressure Sensor application workflow.

Overview

This example project demonstrates the BLE Blood Pressure Sensor application workflow. The Blood Pressure Sensor application uses the BLE Blood Pressure profile to report blood pressure measurement records to a client. Also, the Blood Pressure Sensor application uses the Battery Service to notify the Battery Level and the Device Information services to assert the Device Name and so on.

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. See 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 lesser 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 greater than 2.7 V is described in the 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.

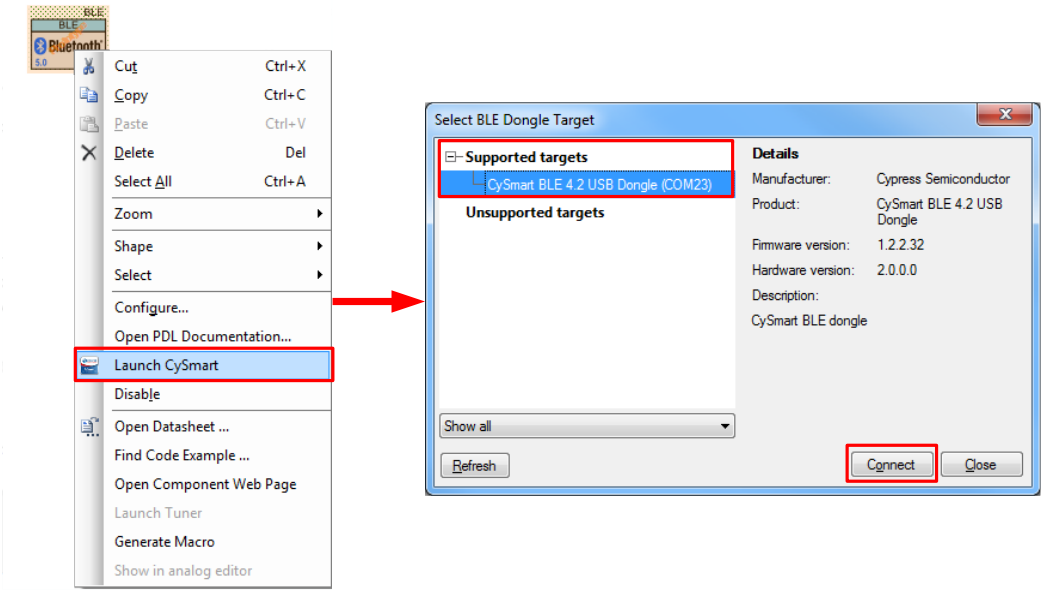
Operation

The Blood Pressure Sensor device can be connected to any BLE (4.0 or later) compatible device configured in the GAP Central role and the GATT Client, which supports Blood Pressure Profile. Also, the Battery and Device Information services may be optionally used. To connect to the Blood Pressure Sensor device, send a connection request to the device while the device is advertising. 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. If the client is connected to the Blood Pressure Sensor and the Blood Pressure Measurement (BPM) characteristic indication, the Intermediate Cuff pressure (ICF) (if it is supported by the Client) characteristic, or both notifications are enabled. The device simulates blood pressure measurement process continuously and periodically (once a second) sends the ICF notification, or BPM indication, or both.

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 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 green 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:
   a. Connect the BLE Dongle to your Windows PC. Wait for the driver installation to complete, if necessary.
   b. Right-click the BLE Component and select Launch CySmart to launch the CySmart Host Emulation Tool. Alternatively, you can navigate to **Start** > **Programs** > **Cypress** and click **CySmart** to launch the tool.
   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.

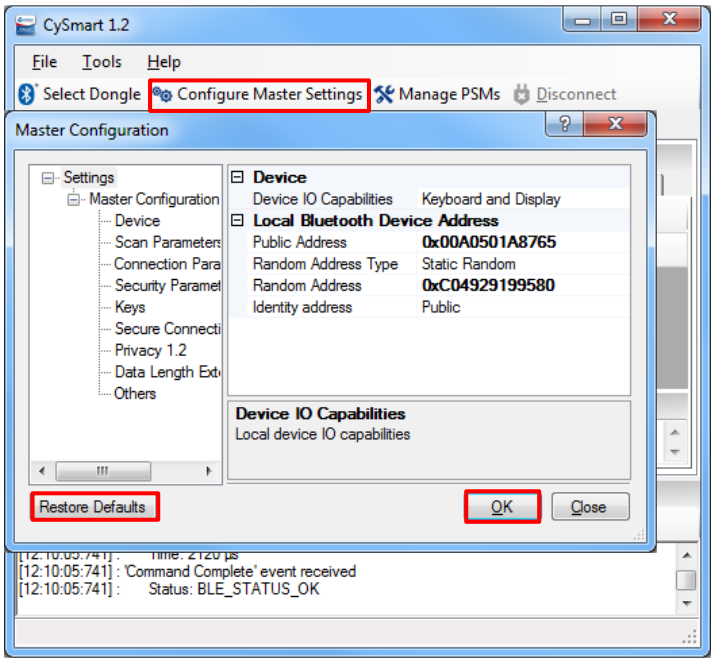
Figure 1. CySmart BLE Dongle Selection



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**.

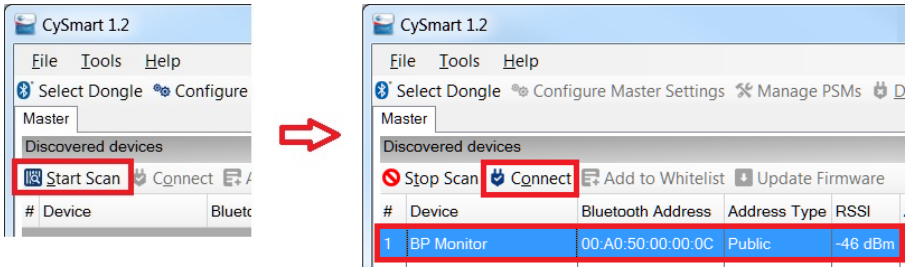
Figure 2. CySmart Master Settings Configuration



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

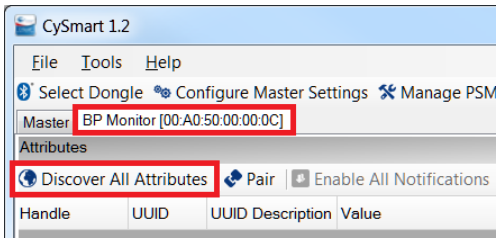
f. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name (configured as **Blood Pressure Monitor**) should appear in the Discovered devices list, as Figure 3 shows. Select the device and click **Connect** to establish a BLE connection between the CySmart Host Emulation Tool and your device.

Figure 3. CySmart Device Discovery and Connection



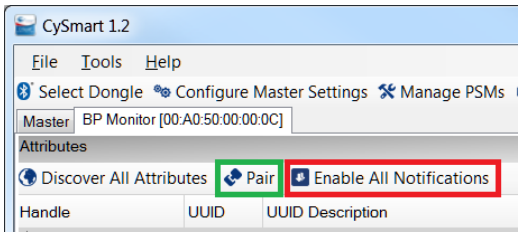
g. Once connected, switch to the **BP Monitor <device>** tab and click **Discover All Attributes** on your design from the CySmart Host Emulation Tool, as shown in Figure 4.

Figure 4. CySmart Attribute Discovery



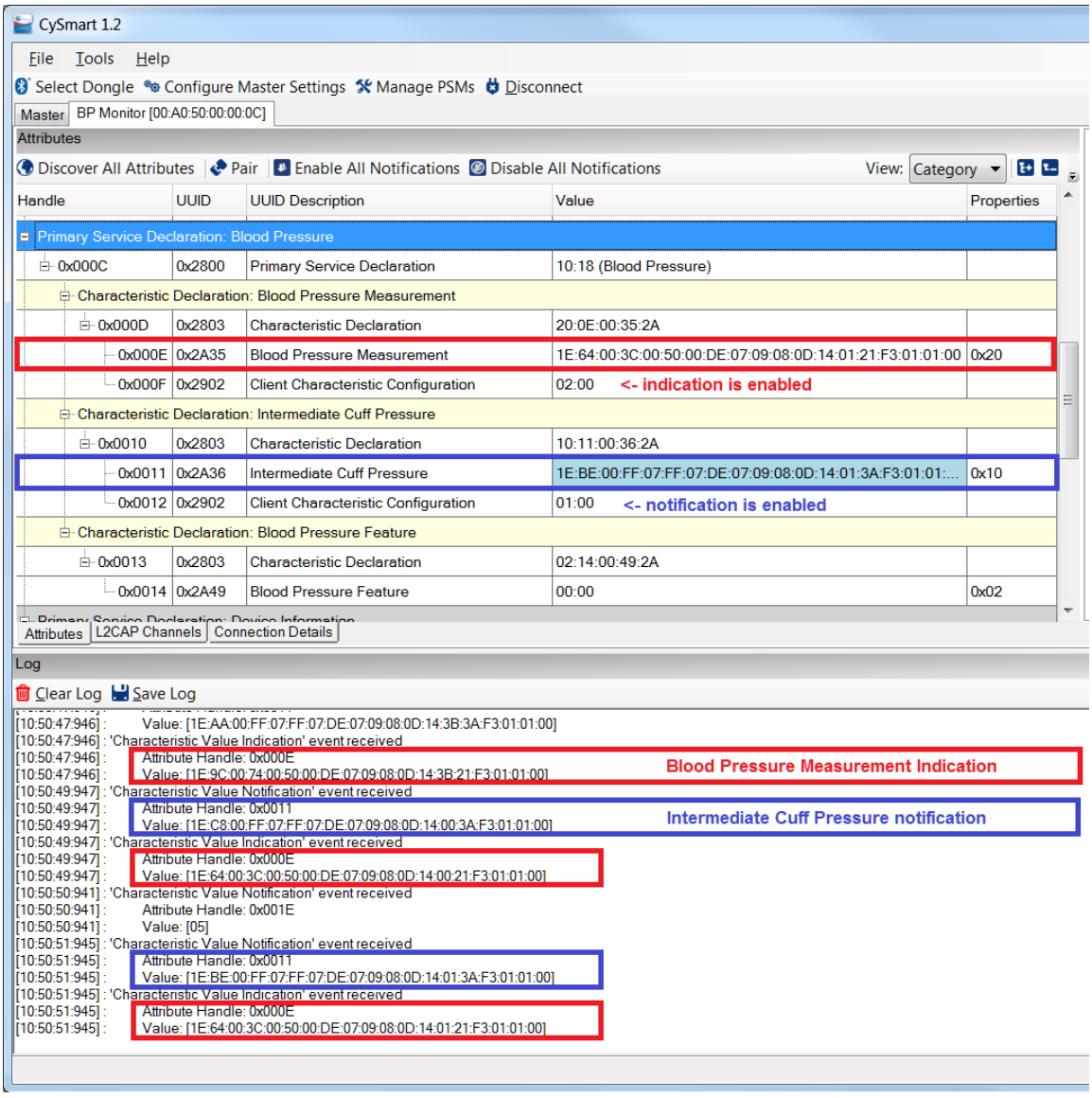
h. Click **Pair** and then click **Enable All Notifications** in the CySmart app as shown in Figure 5.

Figure 5. CySmart Pair and Enable All Notification



i. Observe the Blood Pressure Measurement and Intermediate Cuff Pressure characteristics indications or notifications with the simulated data.

Figure 6. Results in CySmart App



6. Do the following to test example using the CySmart mobile app:
   a. Launch CySmart mobile app and swipe down the screen to refresh the list of BLE devices available nearby.

b. Make sure that the development kit is advertising (green LED is blinking): you may need to press the **SW1** button to wake up the device from the Hibernate mode.

c. Once the **BT Monitor** device appears on the BLE devices list, connect to it, pair, and choose **Blood Pressure Service** in the service selector.

Figure 7. CySmart iOS App Pairs with Blood Pressure Sensor Service

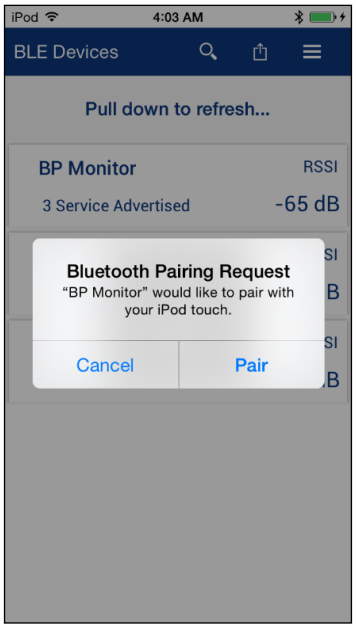
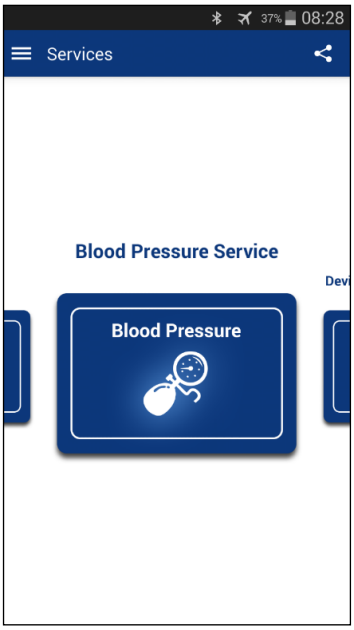


Figure 8. CySmart Android-App Recognized Blood-Pressure Sensor- Profile



d. Observe simulated values of the Blood Pressure with the CySmart mobile app.

Figure 9. CySmart App on iOS Displays Simulated Blood-Pressure Values



Figure 10. CySmart App on Android Displays Blood-Pressure Values



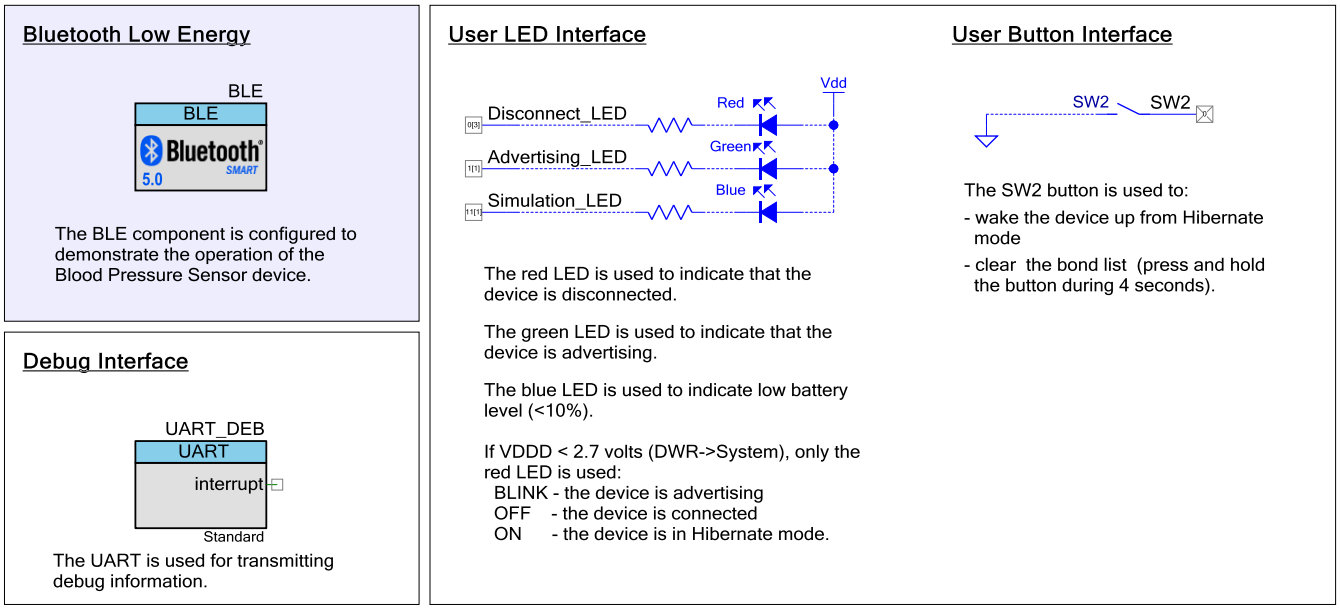
7. For more details on the Blood Pressure Service characteristic data structures, see the BLS Specification.
8. Use the UART debug port to view verbose messages:
   a. The code example ships with the UART debug port enabled. To disable it, set the macro `DEBUG_UART_ENABLED` in *common.h* to `DISABLED` and rebuild the code.
   b. The output of the debug serial port looks like the sample below:

```
BLE Apple Notification Center Example
CY_BLE_EVT_STACK_ON, StartAdvertisement
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a05000001e
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
CY_BLE_EVT_GATT_CONNECT_IND: 3, 7
CY_BLE_EVT_GAP_DEVICE_CONNECTED: connIntv = 48 ms
CY_BLE_EVT_CONNECTION_UPDATE_COMPLETE: connIntv = 7 ms
CY_BLE_EVT_CONNECTION_UPDATE_COMPLETE: connIntv = 48 ms
CY_BLE_EVT_GAP_AUTH_REQ: bdHandle=7, security=3, bonding=1, ekeySize=10, err=0
CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: bdHandle=7, security=2, bonding=1, ekeySize=10,
err=0
CY_BLE_EVT_GAP_PASSKEY_DISPLAY_REQUEST: 890203
CY_BLE_EVT_GAP_ENCRYPT_CHANGE: 0
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: bdHandle=7, security=2, bonding=1, ekeySize=10, err=0
CY_BLE_EVT_PENDING_FLASH_WRITE
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_XCNHG_MTU_REQ 3, 7, final mtu= 23
SimulBatteryLevelUpdate: 3
Blood Pressure Measurement Indication is Enabled
Store bonding data, status: 0, pending: 0
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Measurement Indication is Disabled
Store bonding data, status: 0, pending: 0
Blood Pressure Measurement Indication is Enabled
Store bonding data, status: 0, pending: 0
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Measurement Indication is Disabled
Store bonding data, status: 0, pending: 0
Blood Pressure Measurement Indication is Enabled
Store bonding data, status: 0, pending: 0
Blood Pressure Ind  sys:100 mmHg, dia:60 mmHg
Blood Pressure Measurement Indication is Confirmed
Blood Pressure Measurement Indication is Disabled
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATT_DISCONNECT_IND: 3, 7
CY_BLE_EVT_GAP_DEVICE_DISCONNECTED: bdHandle=7, reason=13, status=0
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
```

Design and Implementation

Figure 11 shows the top design schematic.

Figure 11. BLE Blood Pressure Sensor Code-Example Schematic



The project demonstrates the core functionality of the BLE Component configured as the Blood Pressure Server.

After a startup, the device initializes the BLE Component. In this project, three callback functions are required for the BLE operation. The callback function (`AppCallBack()`) is required to receive generic events from BLE Stack and the service specific callbacks `BasCallBack()` and `BlsCallBack()` are required for Battery and Blood Pressure service-specific events accordingly. 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.

The watchdog timer (WDT) is used to time blood pressure measurement simulations, battery level measurement, and LED blinking.

While connected to the client and between the connection intervals, the device is put into Low-Power mode.

Pin Assignments

The pin assignments and connections required on the development board for supported kits are in Table 1.

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

Components and Settings

Table 2 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 2. 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 Blood Pressure Sensor device.	See the Parameter Settings section
Digital Input Pin	SW2	This pin is used to connect the user button (**SW2**).	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Disconnect_LED Advertising_LED Simelation_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

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

Parameter Settings

The BLE Component is configured as the Blood Pressure Server in the GAP Peripheral role. Also, the BAS and DIS services are included.

Figure 12. General Settings

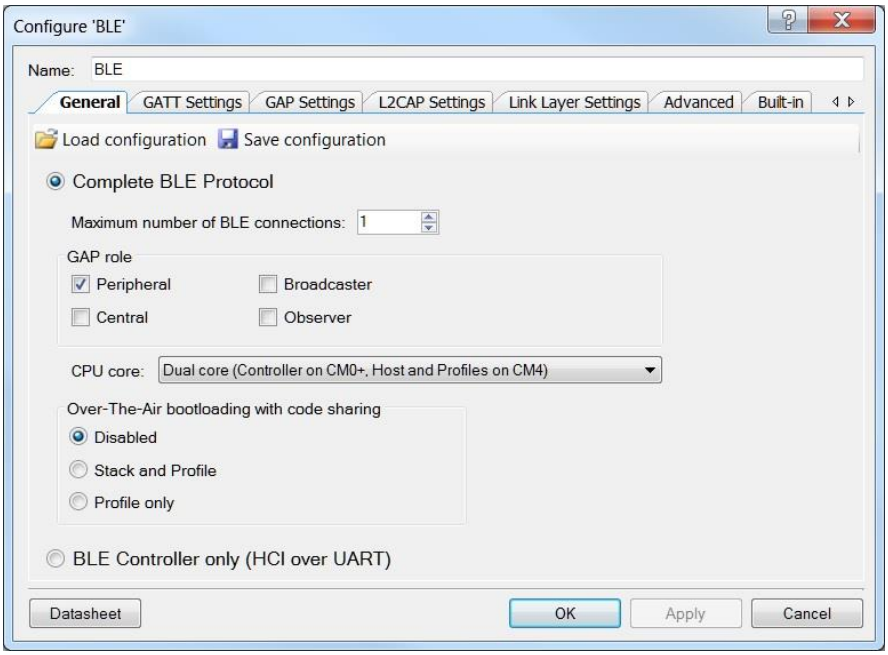


Figure 13. GATT Settings

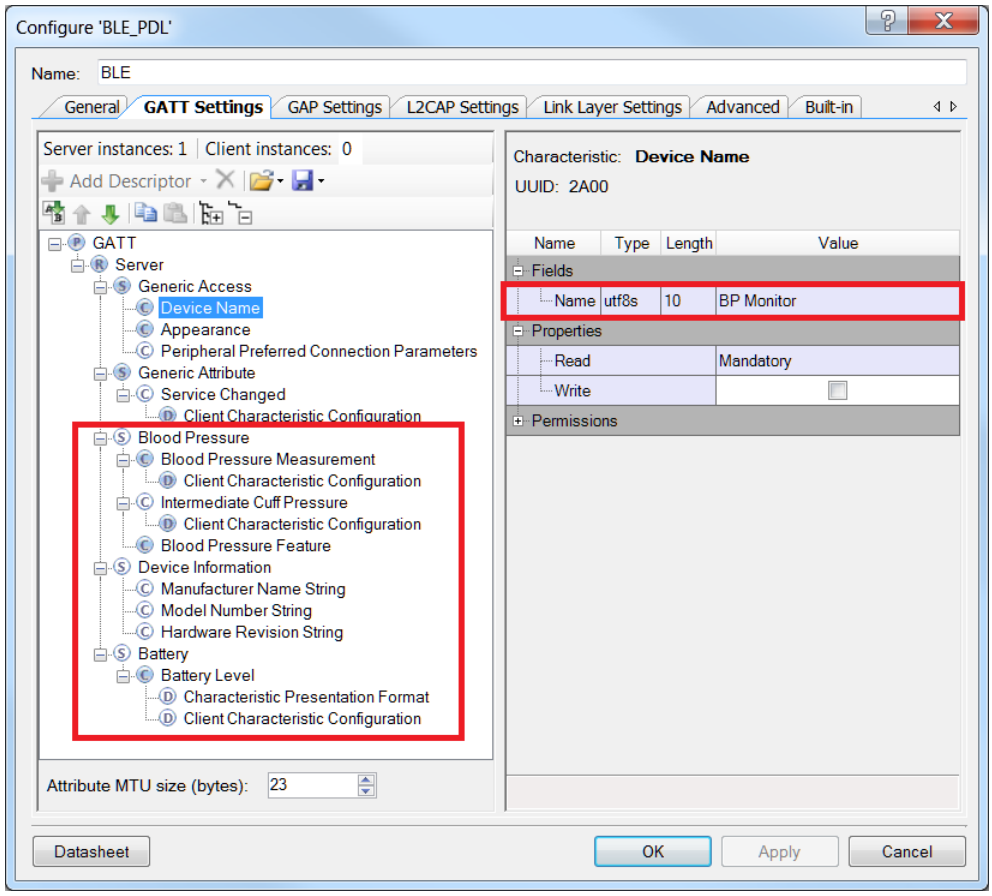


Figure 14. GAP Settings

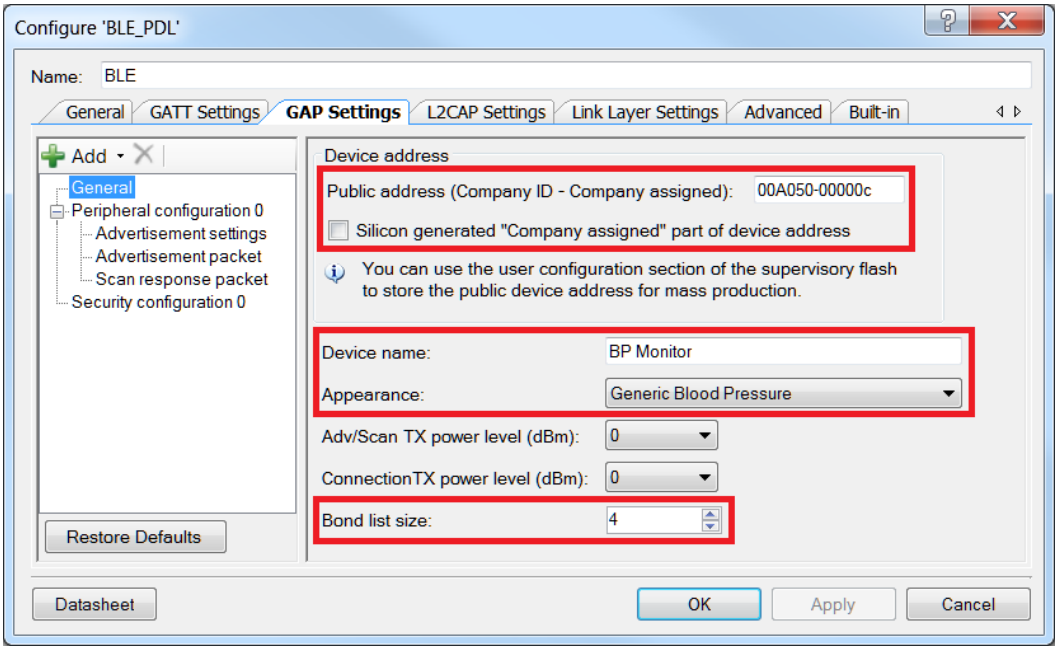


Figure 15. GAP Settings → Advertisement Setting

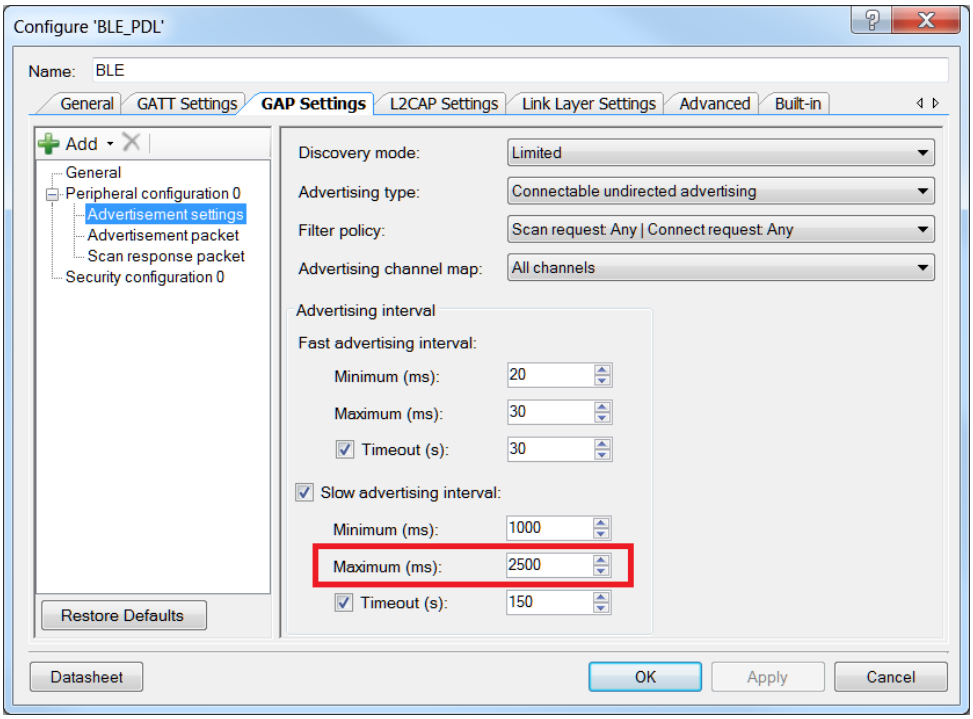


Figure 16. GAP Settings → Advertisement Packet

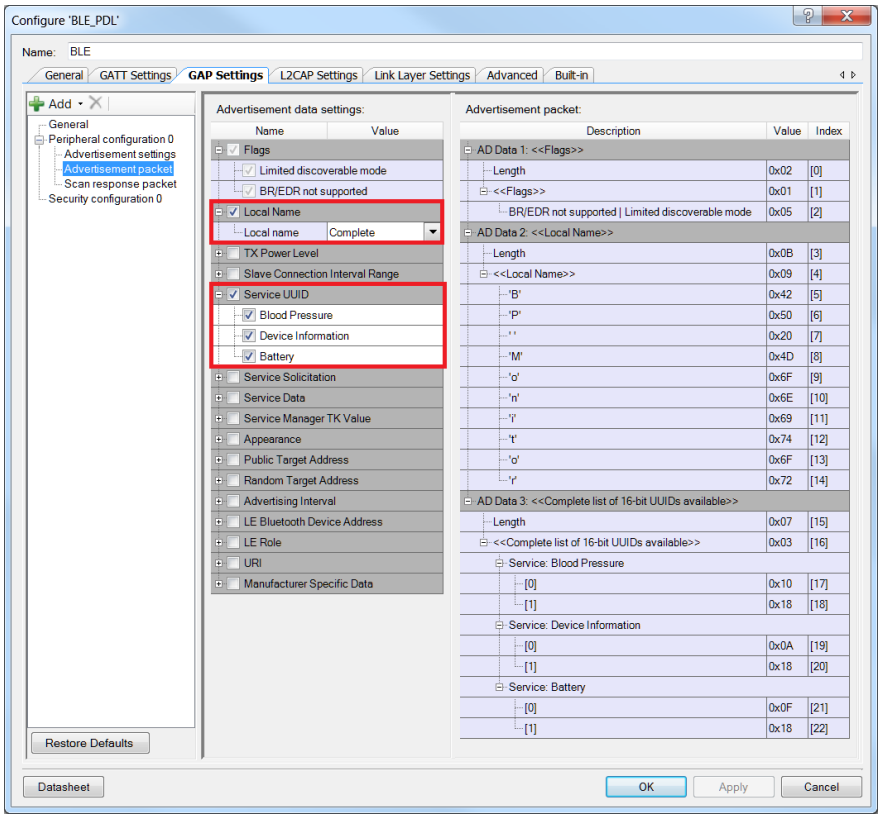
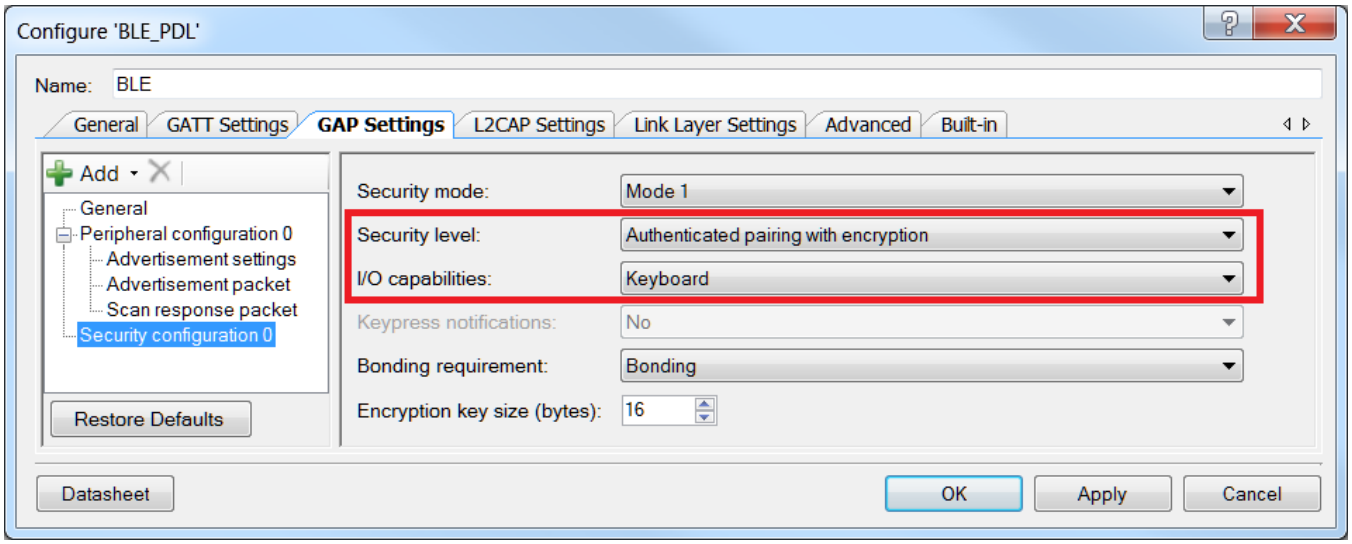


Figure 17. GAP Settings → Security Configuration



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+ 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.

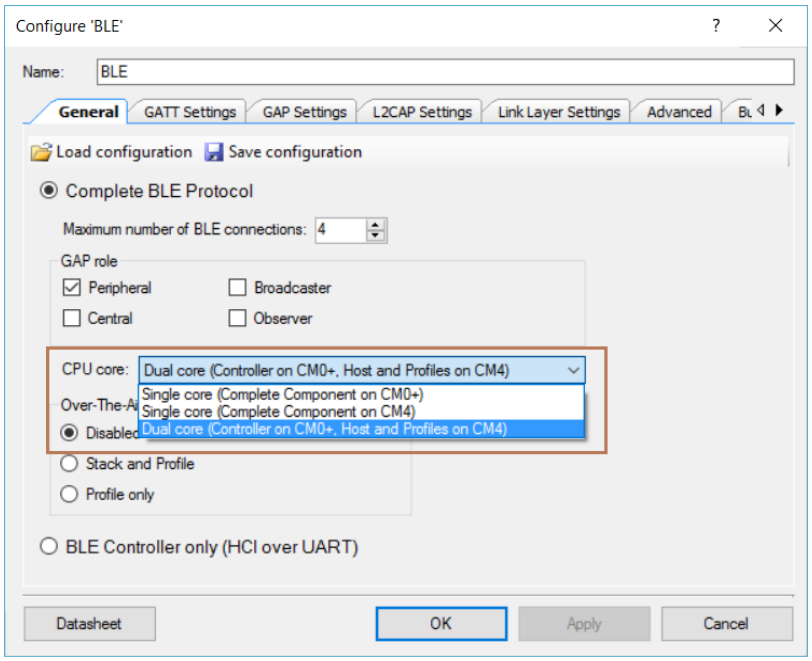
Important to remember:

- 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.

Do the following to switch the CPU cores usage:

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

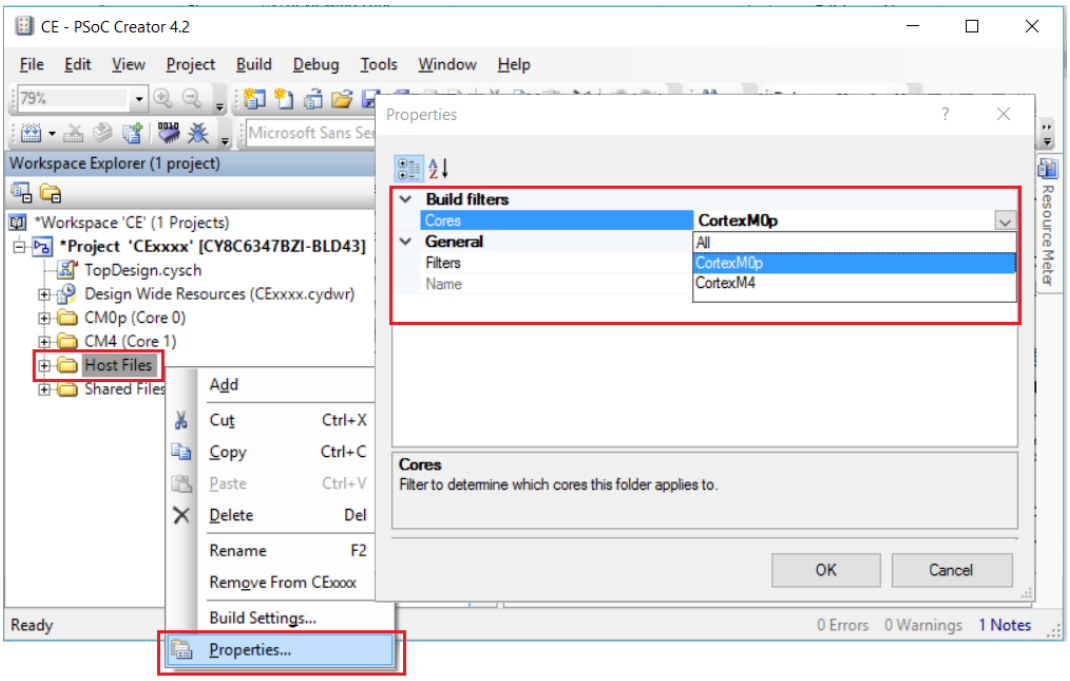
Figure 18. Select CPU Core



2. Identify the core 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 19.

   - 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

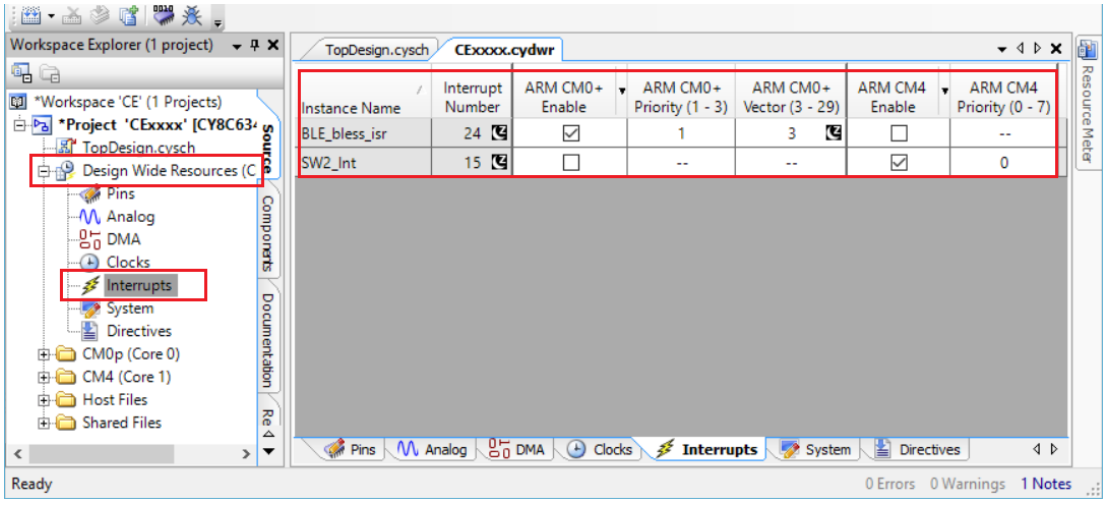
Figure 19. Change Core Properties



3. Assign the BLE_bless_isr and other peripheral (button – SW2, timer(s), and so on) 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 20. 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, kit, or both, 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: CE217633 - BLE Blood Pressure Sensor with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17633

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6086769	NPAL	06/12/2018	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 | 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, 2018. 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. 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.