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.

CE217640 - BLE Phone Alert Client/Server with PSoC 6 MCU with BLE Connectivity

Objective

This code example demonstrates the current Ringer mode of the Phone Alert Server (a phone or CY8CKIT-062 PSoC® 6 BLE Pioneer Kit) and the Ringer and Vibrate states on the user interface LEDs of the Phone Alert Client on the CY8CKIT-062 PSoC 6 BLE Pioneer Kit).

Overview

This code example demonstrates the Phone Alert operation of the PSoC Creator™ Bluetooth Low Energy (BLE) Component. It contains two projects:

- Phone Alert Client
- Phone Alert Server

The Phone Alert Client project uses the BLE Phone Alert Status profile to monitor and control the Alert state and Ringer setting of the Phone Alert Server project. The project demonstrates the core functionality of the BLE Component configured as a Phone Alert Client in the GAP Peripheral role and a Phone Alert Server in the GAP Central role.

This example supports all the GATT sub-procedures defined in the Phone Alert Status Service Specification. The device remains in Sleep mode between BLE connection intervals.

Requirements

Tool: PSoC Creator 4.2

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

Associated Parts: All PSoC 6 MCU with BLE Connectivity 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} greater than 2.7 V is described in the Operation section.

Software Setup

Terminal Tool

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

Operation

The Phone Alert Client device can be connected to any BLE (4.0 or later)-compatible device configured as a GAP Central role and GATT Server that exposes the Phone Alert Status Service. To connect to the Phone Alert Client 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 Central device connects successfully, both red and green LEDs are turned OFF, and the device waits for an authorization request from the server.

When the authorization is done, the Phone Alert Client discovers the server's GATT database (including the Phone Alert Server's characteristics and descriptors). After the discovery process, the device reads the Alert status and Ringer setting characteristic and indicates their states as well as configures their descriptors to notifications. If either the Ringer or Vibrate is active (incoming call), the green or blue LED blinks. If the Ringer is active (blinking green LED), the SW2 button performs a Mute Once function, so it mutes the Ringer (clears the Ringer bit of the Alert Status characteristic), but does not affect the Ringer Setting characteristic.

If there is no incoming call (Ringer and Vibrate are passive), the green and blue LEDs represent a Ringer Setting value: the LED blue light indicates "Normal," and the LED green light indicates "Silent." The SW2 button performs the Ringer setting toggling from "Normal" to "Silent" and vice versa. The BLE stack timer is used to control the LED's blinking.

The example project uses the UART Component for displaying debug information and entering commands through the terminal emulator app. Freeware, such as HyperTerminal and PuTTY, is available on the web and can be used with this example. The commands listed in Table 1 are the procedures the user can perform in Phone Alert Status Server Project

Table 1. List of Commands of Phone Alert Status Server Project

PASS-Specified Commands	
a – Alert Status Notification	
z - Ringer Setting Notification	
General Commands	
h – Help screen	p – Pair
s – Start scanning	d – Disconnection
c – Initiate connection	r – Unbounding all devices

The commands listed in Table 2 are the procedures the user can perform in Phone Alert Status Client Project

Table 2. Commands of Phone Alert Status Client Project

General Commands	
h – Help screen	d – Disconnection
p – Pair	r– Unbounding all devices

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 BLE Phone Alert Client 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. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
5. 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.

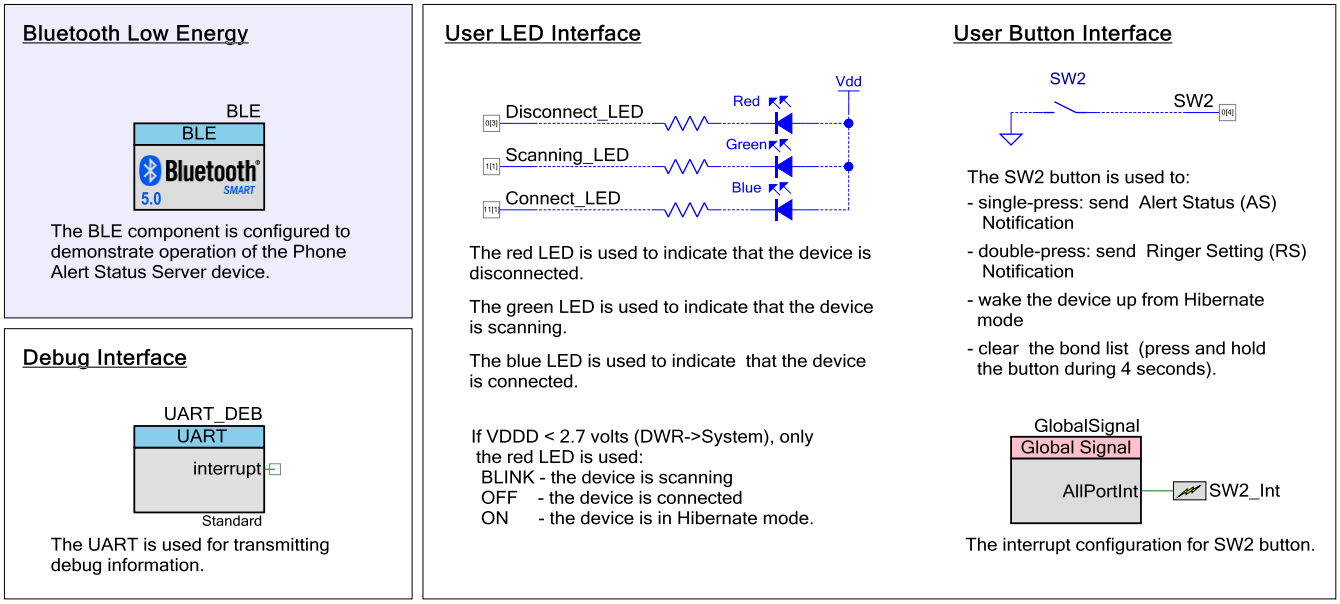
6. Build the BLE Phone Alert Client project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.
7. Observe information about the device address, advertising, scanning, connecting, discovering, and pairing in the Client Terminal and Server Terminal.
8. To complete the authentication procedure, enter the passkey in the Server Terminal that is shown in the Client Terminal.

   Observe information about storing bonding data, reading the Alert Status and Ringer Setting characteristics, enabling the notification for the Alert Status and Ringer Setting characteristics in the Client Terminal and Server Terminal.
9. Enter the [**h**] command in the Client Terminal and Server Terminal to display information about the available project commands.
10. Enter the [**a**] command in the Server Terminal (or press the **SW2** button on the Server board) to change and send the Alert Status characteristic notification by using the cycle No Alert > Ringer > Vibrate > Vibrate + Ringer.
11. Enter the [**z**] command in the Server Terminal (or double-press the **SW2** button on the Server board) to change and send the Ringer Setting characteristic by using the cycle Ringer Normal > Ringer Silent.
12. Enter the [**a**] command in the Server Terminal to send the Alert Status characteristic notification again. Observe the LED on the CY8CKIT-062 PSoC® 6 BLE Pioneer Kit board. The green LED blinks after the notification if the value of the Alert Status characteristic is Ringer. The blue LED blinks after the notification if the value of the Alert Status characteristic is Vibrate. If the value of the Alert Status characteristic is No Alert, the LED indicates the value of the Ringer Setting characteristic: the blue LED – Normal mode and the green LED – Silent mode.
13. Press the client **SW2** button on the CY8CKIT-062 PSoC® 6 BLE Pioneer Kit board to write the Control Point characteristic in the server project. If the Alert Status characteristic is Ringer, pressing **SW2** sends the **Mute Once** command to the server. If the Alert Status characteristic is Vibrate, pressing **SW2** sends the **Set Silent Mode** or **Cancel Silent Mode** command to the server.

Design and Implementation

The schematic of BLE Phone Alert Client Code Example is shown in Figure 1.

Figure 1. BLE Phone Alert Client Schematic



The project demonstrates the functionality of the BLE Component configured as a Phone Alert Client.

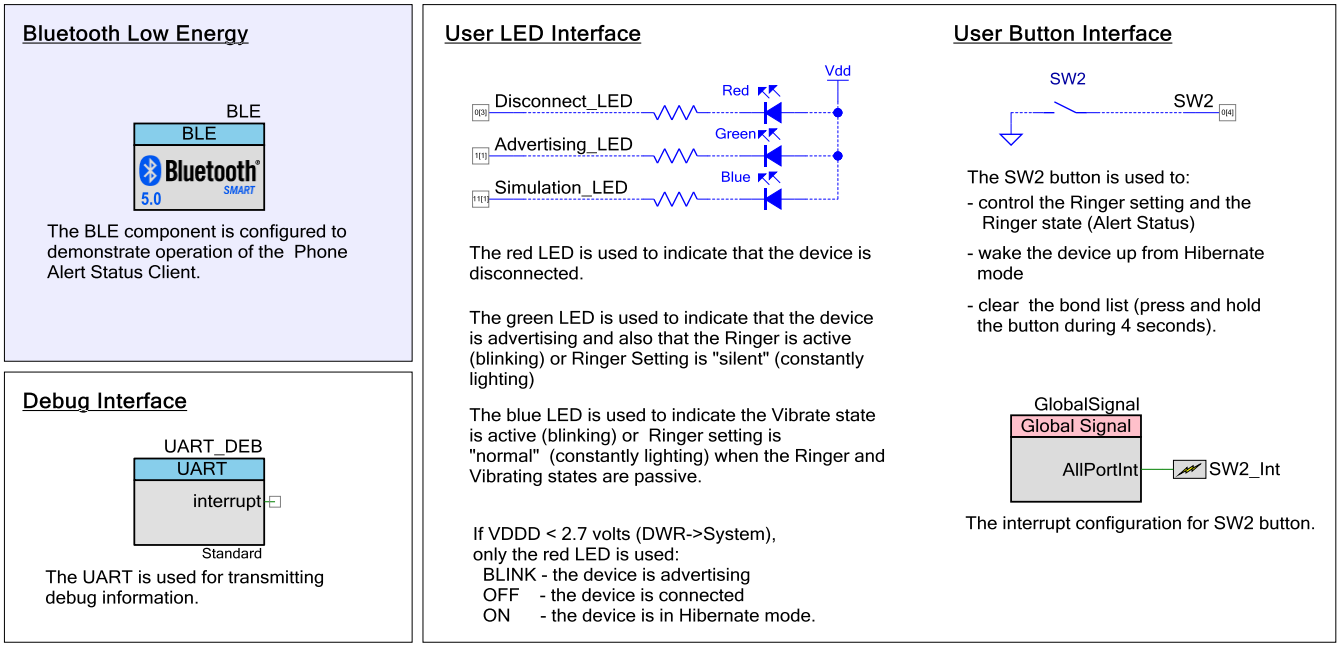
After startup, the device performs the BLE Component initialization. In this project, two callback functions are required for the BLE operation: `AppCallBack()` is required to receive generic events from the BLE Stack and the service-specific callback `PassCallBack()` is required for Phone Alert Status 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 (Figure 7). The BLE Component stops advertising when the 180-second advertising period expires.

The device is in Sleep mode when it is connected to the Server and between connection intervals.

This example project uses the UART Component for displaying debug information and entering commands through the terminal emulator app. Freeware, such as HyperTerminal, PuTTY, and so on, is available on the web and can be used with this example.

The schematic of BLE Phone Alert Server Code Example is shown in Figure 2.

Figure 2. BLE Phone Alert Server Schematic



After startup, the device initializes the BLE Component. The Component requires several callback functions to receive events from the BLE Stack. `AppCallBack()` is used to receive general BLE events. Another callback, (`IpsCallBack()`), is used to receive events specific to the service's attribute operations.

The `CYBLE_EVT_STACK_ON` event indicates the successful initialization of the BLE Stack. After this event is received, the Component starts fast advertising with the packet structure, as configured in the BLE Component Customizer (Figure 7).

The device is in Sleep mode when it is connected to the server and between connection intervals.

Pin Assignments

The pin assignments for the Phone Alert Status Client Project are in Table 3 and for the Phone Alert Status Server Project are in Table 4.

Table 3. Pin Assignment Phone Alert Status Client Project

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P5[0]	
\UART_DEB:tx\	P5[1]	
\UART_DEB:rts\	P5[2]	

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:cts\	P5[3]	
Scanning_LED	P1[1]	The green color of the RGB LED
Disconnect_LED	P0[3]	The red color of the RGB LED
Connected_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Table 4. Pin Assignment of Phone Alert Status Server 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]	
Advertising_LED	P1[1]	The green color of the RGB LED
Disconnect_LED	P0[3]	The red color of the RGB LED
Simulation_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Components and Settings

Table 5 lists the PSoC Creator Components used in Phone Alert Client example, 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 Phone Alert Client Example

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE component is configured to demonstrate operation of the Phone Alert Status Client.	See the Parameter Settings for Phone Alert Client Project section
Digital Input Pin	SW2	This pin is used to generate interrupts when the user button (**SW2**) is pressed.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Disconnect_LED Advertising_LED Simulation_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
SysInt	SW2_Int	This Component is configured to extract interrupts from GlobalSignal.	**[Basic tab]** DeepSleepCapable = true
GSRef	GlobalSignal	This Component is used to detect if any of the interrupt enabled pins triggered an interrupt. It is a separate resource from the dedicated port interrupts, and it has the ability to wake up the chip from deep-sleep mode	**[Basic tab]** Global signal name: HWCombined Port Interrupt (AllPortInt)
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default

Table 6 lists the PSoC Creator Components used in Phone Alert Server example, how they are used in the design, and the non-default settings required so they function as intended.

Table 6. PSoC Creator Components used in Phone Alert Server Example

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE component is configured to demonstrate operation of the Phone Alert Status Server device.	See the Parameter Settings for Phone Alert Server Project section
Digital Input Pin	SW2	This pin is used to generate interrupts when the user button (**SW2**) is pressed.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Disconnect_LED Advertising_LED Connected_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
SysInt	SW2_Int	This Component is configured to extract interrupts from GlobalSignal.	**[Basic tab]** DeepSleepCapable = true
GSRef	GlobalSignal	This Component is used to detect if any of the interrupt enabled pins triggered an interrupt. It is a separate resource from the dedicated port interrupts, and it has the ability to wake up the chip from deep-sleep mode	**[Basic tab]** Global signal name: HWCombined Port Interrupt (AllPortInt)
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 Phone Alert Client Project

BLE Component

The BLE Component is configured as the PASS Client in the GAP Peripheral role with the settings shown in Figure 3

Figure 3. General Settings

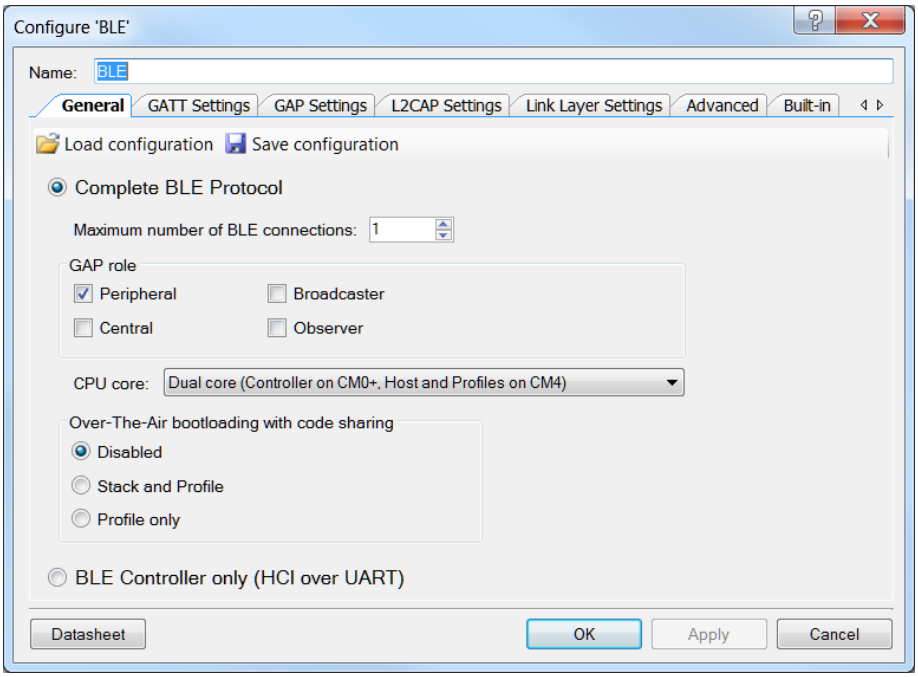


Figure 4. GATT Settings

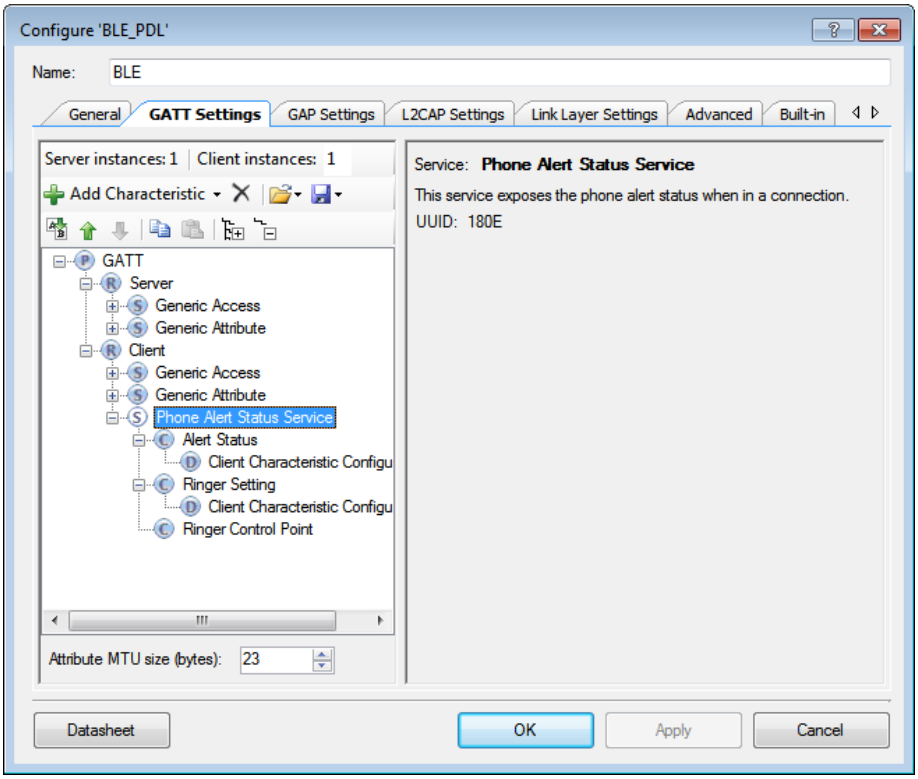


Figure 5. GAP Settings

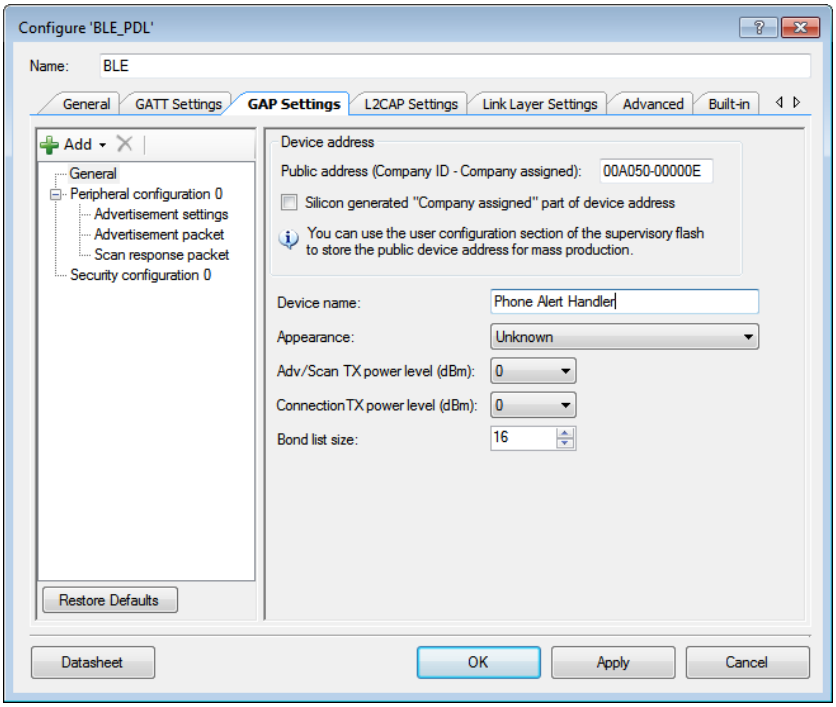


Figure 6. GAP Settings: Advertisement Settings

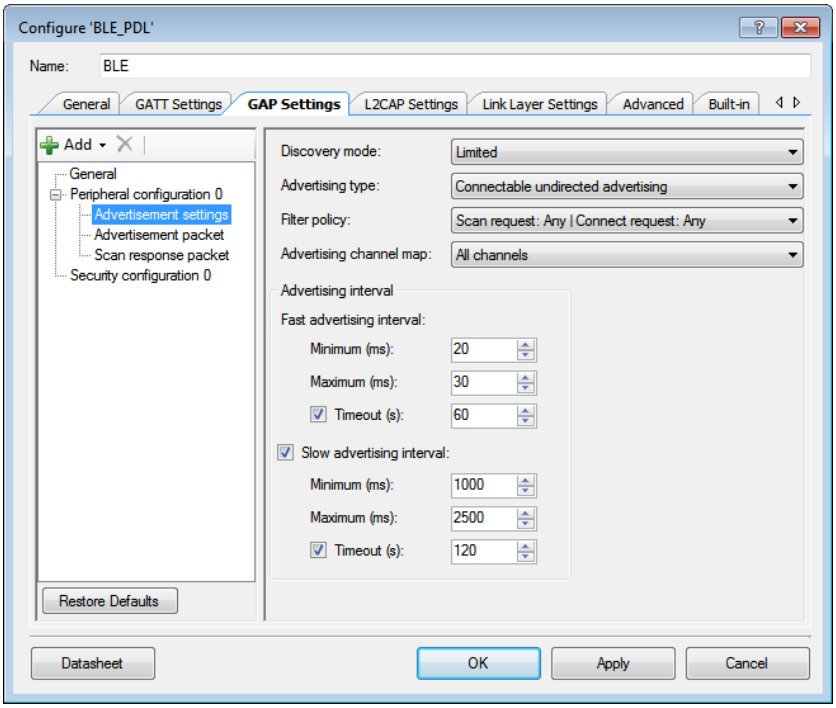


Figure 7. GAP Settings: Advertisement Packet

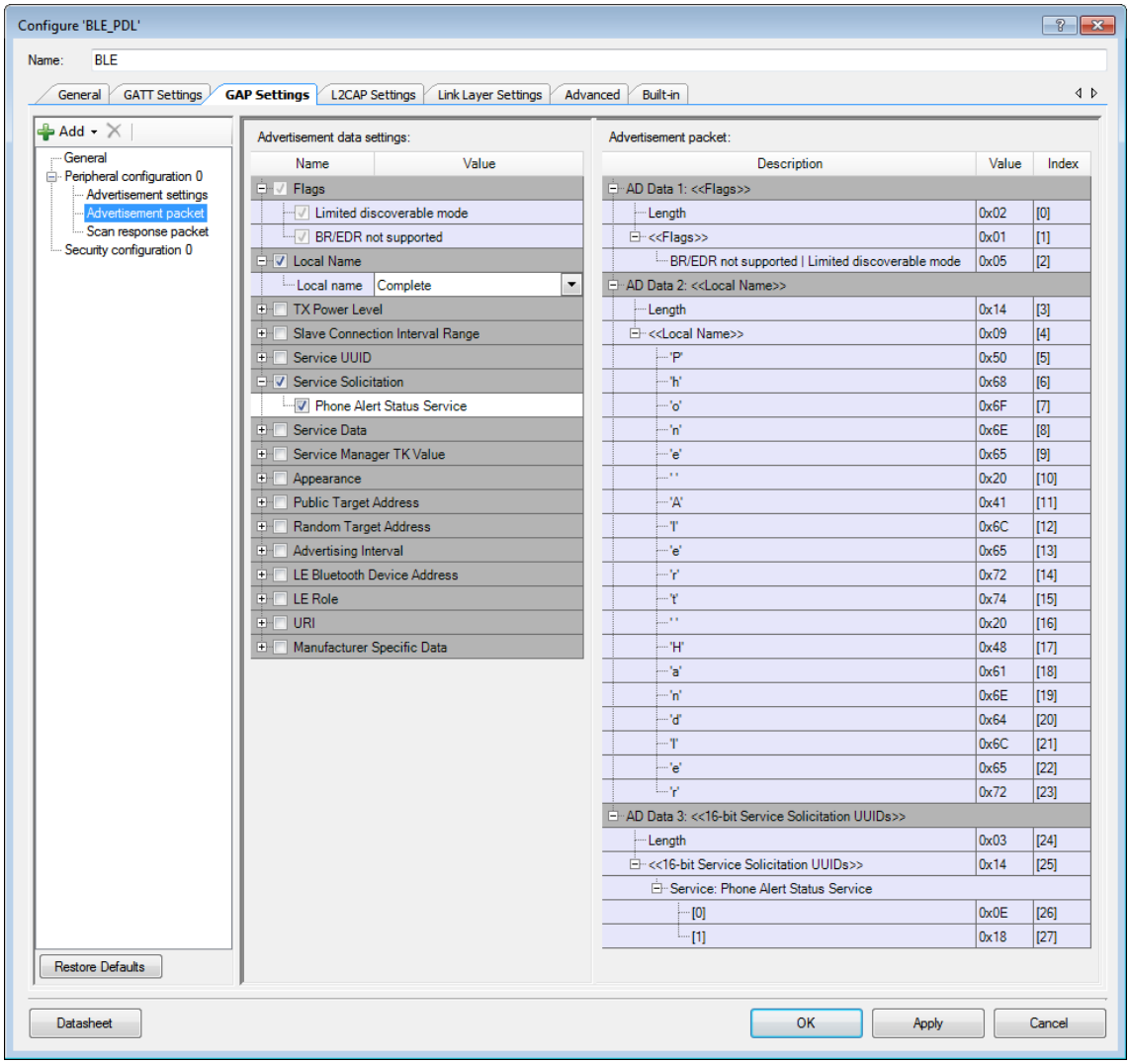
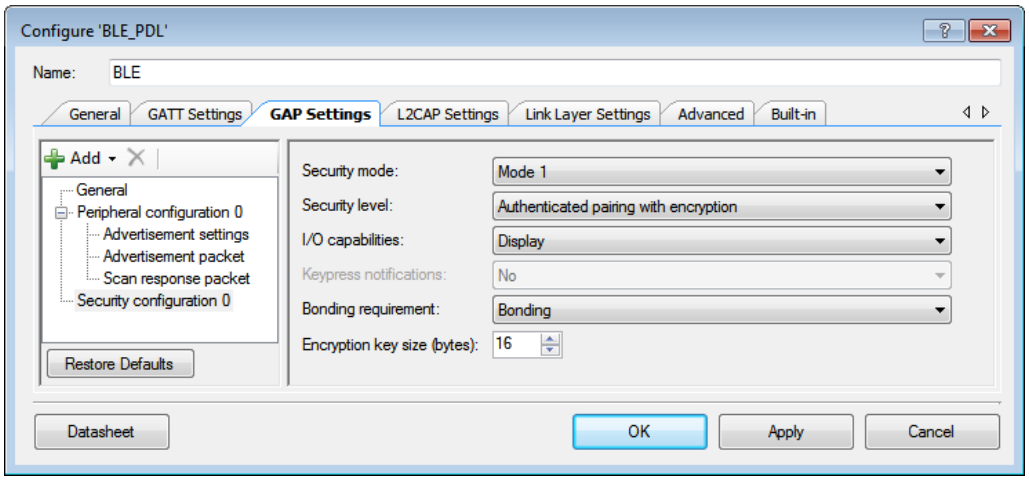


Figure 8. Security Settings



Parameter Settings for Phone Alert Server Project

BLE Component

The BLE Component is configured as the Phone Alert Status Server in the GAP Peripheral role with the settings shown Figure 9 to Figure 12

Figure 9. General Settings

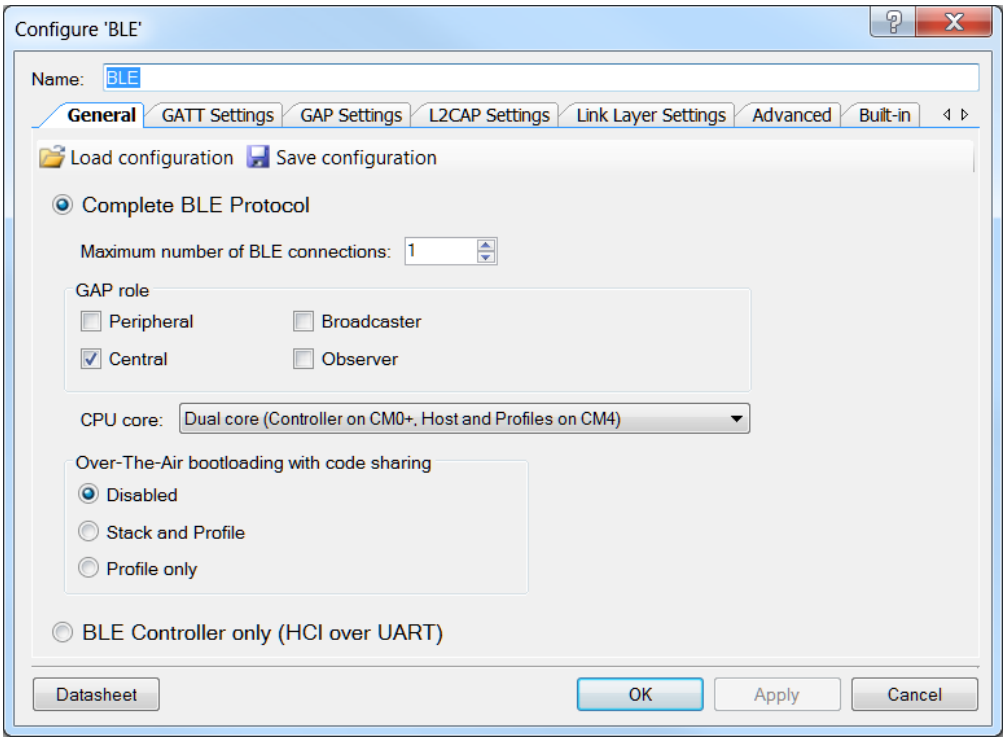


Figure 10. GATT Settings

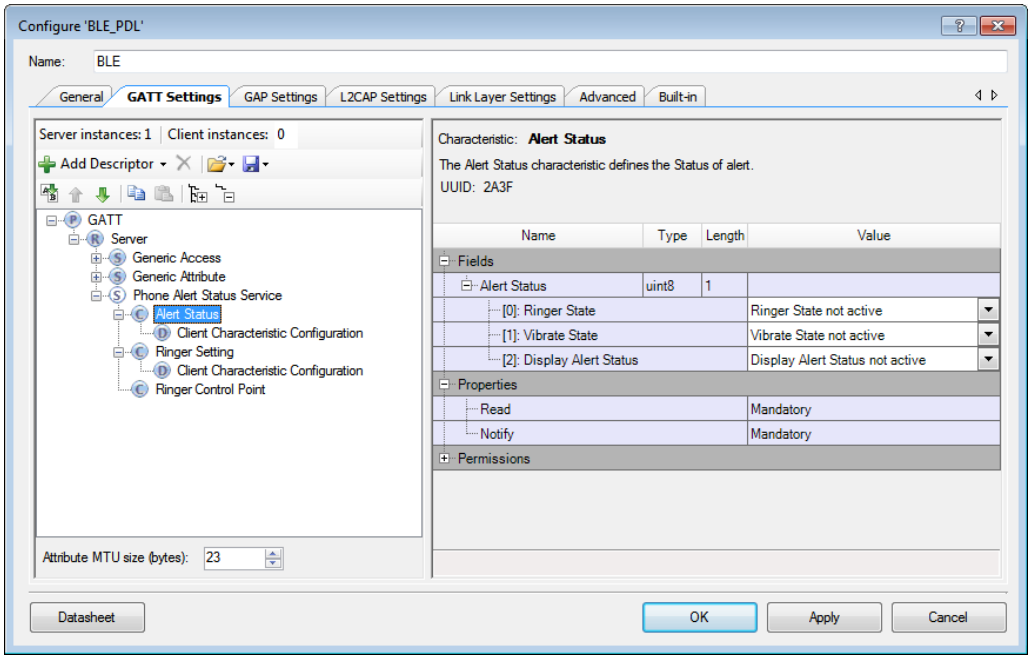


Figure 11. GAP Settings

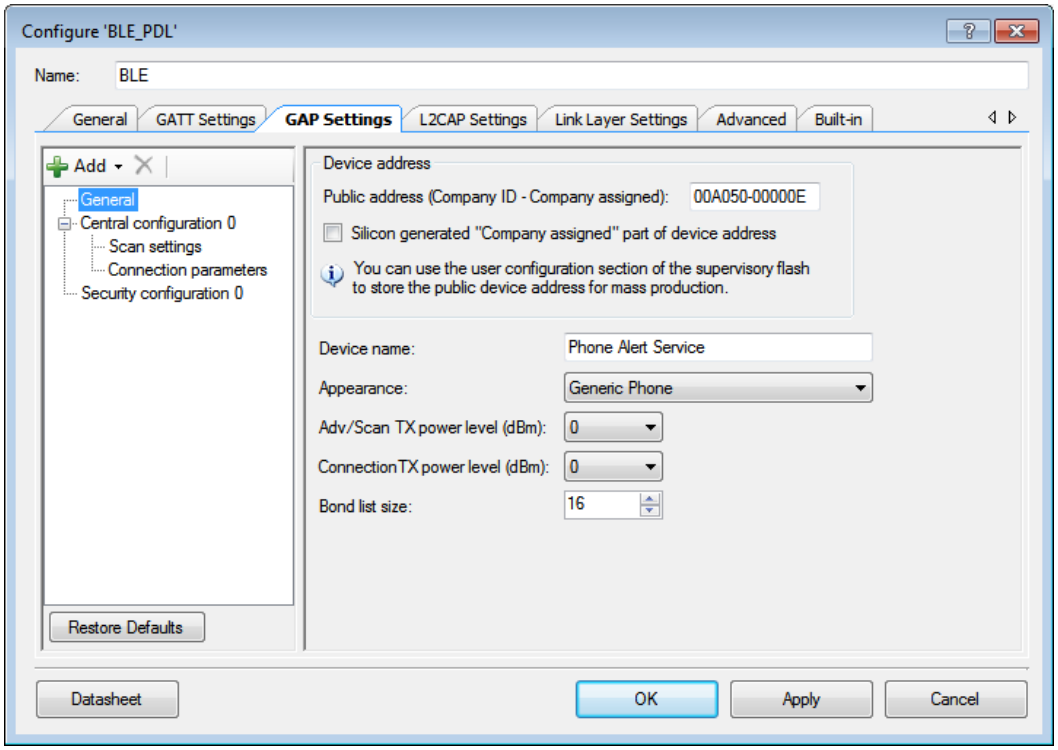
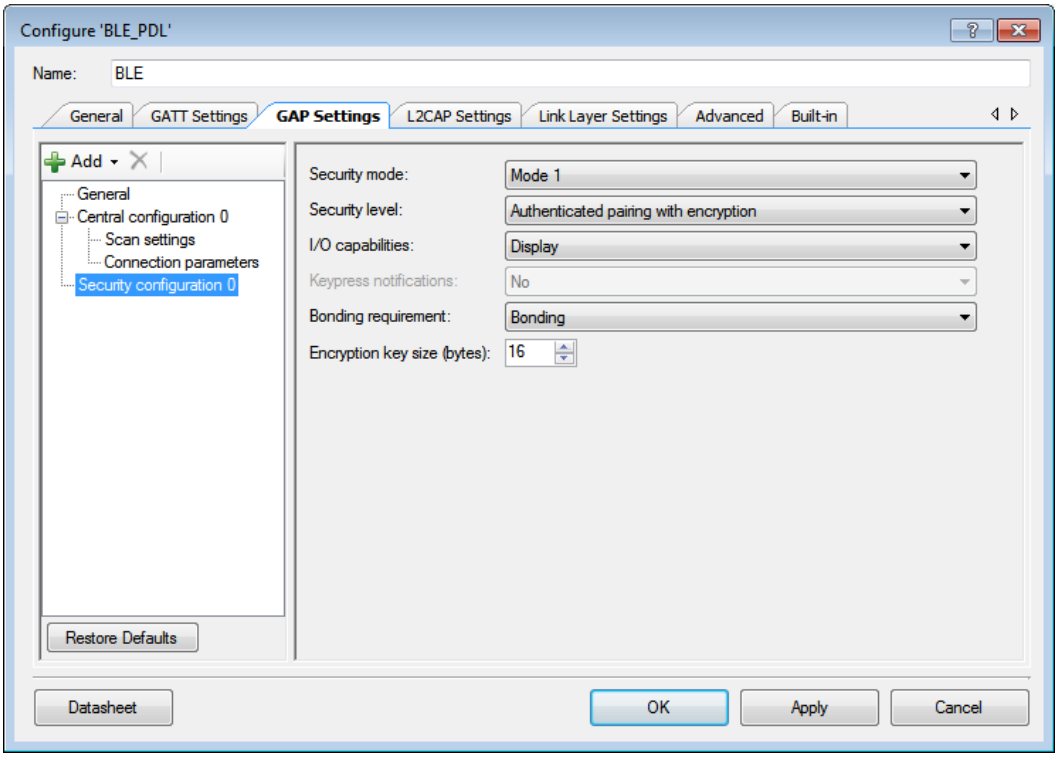


Figure 12. Security Settings



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.

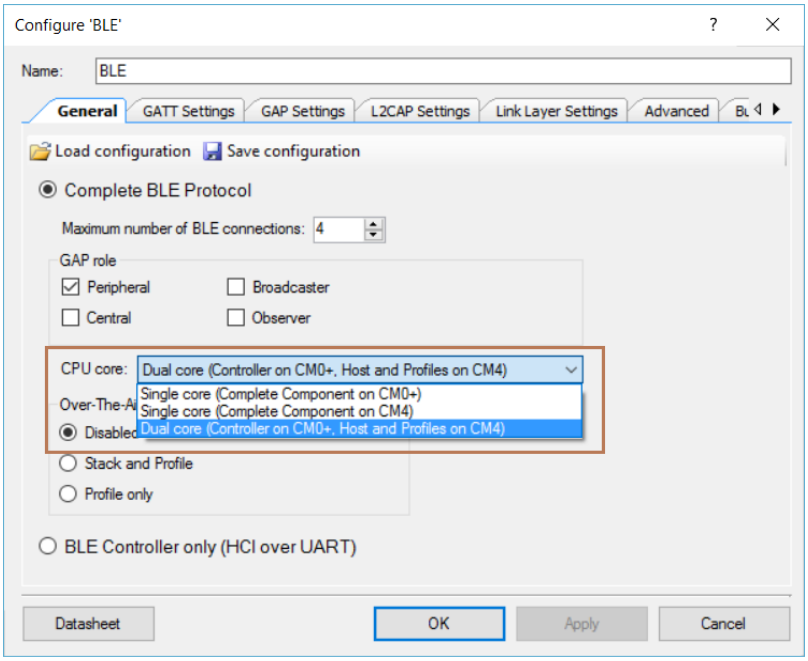
The BLE example structure allows easy switching between different CPU cores options. Here are some important points to remember:

- 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 and son.) used in the example must be assigned to the host core.

Do the following to switch CPU Cores usage:

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

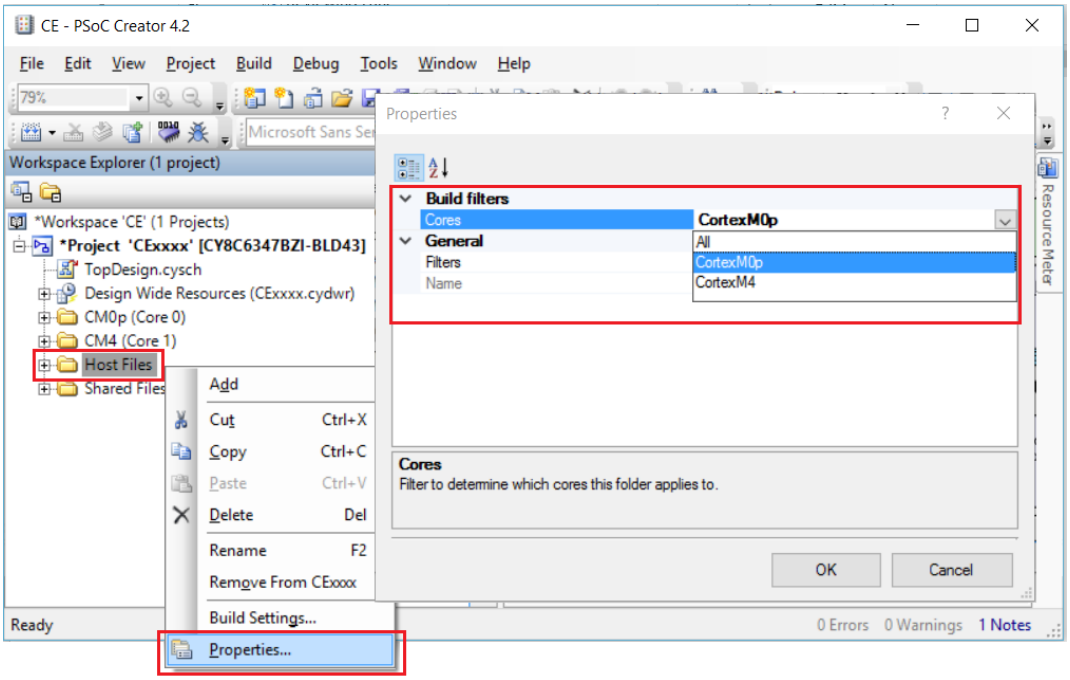
Figure 13. Select CPU Core



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 14.

   - 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

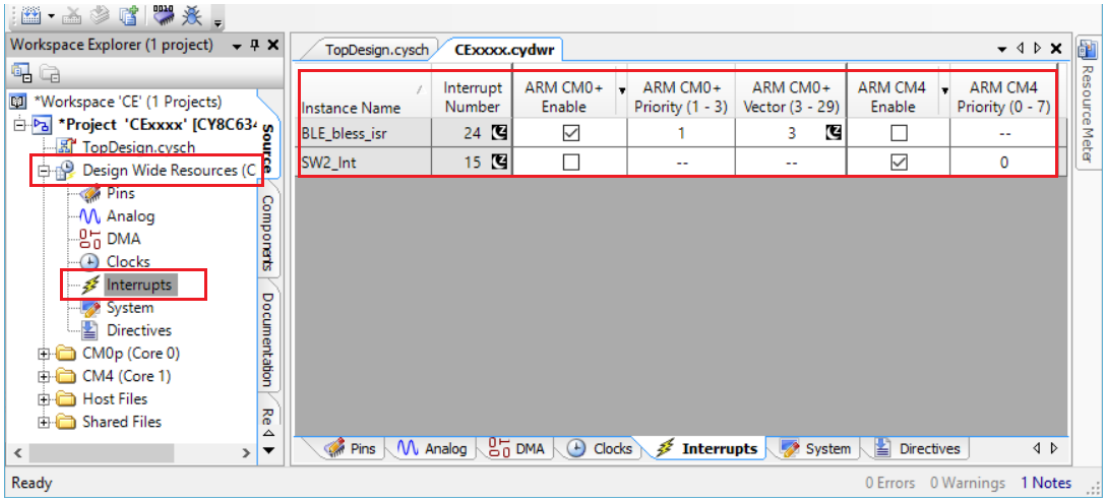
Figure 14. Change Core Properties



3. Assign 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 15. 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: CE217640 - BLE Phone Alert Client/Server with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17640

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6091451	NPAL	06/05/2018	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.