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.

CE217631 – BLE Alert Notification Profile with PSoC 6 MCU with BLE Connectivity

Objective

This example project demonstrates the Alert Notification Profile application workflow.

Overview

This example project demonstrates the Alert Notification Client operation of the BLE PSoC Creator Component. The Alert Notification Client uses the BLE Alert Notification Profile with one instance of the Alert Notification Service to receive information about Email, missed call, and SMS/MMS alerts from the Alert Notification Server. The device remains in Sleep mode between the 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 project demonstrates the core functionality of the BLE Component configured as an Alert Notification Client in the GAP Peripheral role. For operation, the example project requires the Alert Notification Server configured in the GAP Central role. The example project requires the CY8CKIT-062 PSoC® 6 BLE Pioneer Kit.

After the initialization, the BLE Component begins operating and the RGB LED starts blinking green. This indicates that the device has started advertising and it is available for the connection with a Central device. After 30 seconds, if no Central device has connected to the Alert Notification Client, the Client stops advertising and the red LED is turned ON indicating the disconnection state. To connect to the Alert Notification Client device, send a connection request to the device when the device is advertising.

LEDs behavior:

- The green LED blinks when the device is advertising and turns on if the email alert is received.
- The red LED turns ON when the device is in the disconnected state and if a missed call alert is received.
- The blue LED blinks or turns ON when the device is alerting and if an SMS/MMS alert is received.

This example project uses the UART Component for displaying debug information and entering commands through the terminal emulator app. Freeware such as HyperTerminal, Bray's Terminal, and PuTTY are available on the web and can be used with this example. Commands are the procedures the user can perform. Table 1 has the list of terminal commands

Table 1. Terminal Commands List

Command	Description
General Commands	
'q'	Start advertising.
'a'	Start scanning.
'd'	Send a disconnect request to a peer device.
'f'	Unbond all the devices.
'i'	Out list of available commands.
Client Commands	
'n'	Turn off the LED of the New Alert Category that was notified previously.
'e'	Send a request with the immediate notification command for the New Alert Characteristic with the Category ID set to Email.
'm'	Send a request with the immediate notification command for the New Alert Characteristic with the Category ID set to Missed call.
's'	Send a request with the immediate notification command for the New Alert Characteristic with the Category ID set to SMS/MMS.
'r'	Read the Supported New Alert Category Characteristic. This command is required for the Client to configure local supported categories setting prior sending a notification request to the Server.
't'	Send the Enable New Alert Notification command to the Alert Notification Control Point Characteristic. The category ID is set to category All Categories.
'o'	Send the Disable New Alert Notification command to the Alert Notification Control Point Characteristic. The category ID is set to All Categories.
'0'	Enable notifications for the New Alert Characteristic.
'1'	Disable notifications for the New Alert Characteristic.
Server Commands	
'c'	Send a connect request to a peer device.
'7'	Send the Missed call notification.
'8'	Send the Email notification.
'9'	Send the SMS notification.

This list is prompted to the terminal when 'i' is entered in the app.

Note that for operation the example project requires a bonding procedure described in Bonding section.

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 Alert Notification (in the GAP Central) 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. In the terminal press 'q' to start advertising. This device will operate as Client in GAP Peripheral role after this operation.
5. Plug the second CY8CKIT-062-BLE kit board into your computer's USB port.
6. 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.
7. Build the same BLE Alert Notification project and program it into the second 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.
8. In the terminal press 'a' to start scanning. This device will operate as Server in GAP Central role after this operation.
9. In the Server terminal press "c" to connect to peer device. A similar to the example message will appear on the Server terminal emulator:

```
BLE Alert Notification Example


CY_BLE_EVT_STACK_ON
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a05000010d
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE


a
CY_BLE_EVT_GAPC_SCAN_START_STOP
GAPC_START_SCANNING


--------------------------------------------------------------------------------
uuid: ALERT NOTIFICATION SERVICE - YES, added to the connect list
ADV type: 0x0 address: 00a05000010d, rssi - -48 dBm, data - 02 01 06 0e 09 41 6c 65 72 74
20 48 61 6e 64 6c 65 72 03 03 11 18 03 14 11 18
--------------------------------------------------------------------------------


c
Detected device:
Device 1: 00a05000010d
Select the device for connection: (1..1):
```

```
1
Connecting to the device Device 2: 00a05000010d
CY_BLE_EVT_GATT_CONNECT_IND: 3, 7
Connected as Central (master role)


CY_BLE_EVT_GAPC_SCAN_START_STOP
Scan complete!


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: 133991
```

10. In the Client terminal enter a 6-digit passkey (displayed from Server terminal).
11. In the Client terminal press '0' to send a command to enable notifications for the New Alert Characteristic.
12. In the Client terminal press 't' to send a command to enable notifications for all categories in the Alert Notification Control Point Characteristic.
13. In the Server terminal press '7', '8' or '9' commands to send the immediate notification from the Server's side for the SMS/MMS, Email or Missed call categories.
14. In the Client terminal press the 's', 'm' or 'e' commands to request the Server to send immediate notification for the SMS/MMS, Email or Missed call categories.
15. Press 'd' on any of terminals to disconnect the devices.
16. A message will be displayed on the Client Terminal emulator:

```
BLE Alert Notification Example


CY_BLE_EVT_STACK_ON
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a05000010d
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE


q
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
The advertisement is enabled


CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE
CY_BLE_EVT_GATT_CONNECT_IND: 3, 7
Connected as Peripheral (slave role)
```

```
CY_BLE_EVT_GAP_AUTH_REQ: bdHandle=7, security=2, 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_ENTRY_REQUEST
Please, enter the passkey displayed on the peer device:
Enter a 6-digit passkey:
133991

CY_BLE_EVT_GAP_ENCRYPT_CHANGE: 0
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: security:2, bonding:1, ekeySize:10, authErr 0

StartDiscovery
CY_BLE_EVT_PENDING_FLASH_WRITE
The discovery is complete.
Service with UUID 0x1800 has range from 0x1 to 0x7
Service with UUID 0x1801 has range from 0x8 to 0xb
Service with UUID 0x1811 has range from 0xc to 0x18
0
Cy_BLE_ANSC_SetCharacteristicDescriptor routine Success
The ANS Characteristic's Descriptor was written successfully.
t
Cy_BLE_ANSC_SetCharacteristicValue(CY_BLE_ANS_ALERT_NTF_CONTROL_POINT) routine Success
The Alert Notification Control Point Characteristic was written successfully.

The New Alert Characteristic notification is received.
The notified value is:
The Category ID - Email.
The number of alerts: 5.
Text: "Hello".

The New Alert Characteristic notification is received.
The notified value is:
The Category ID - SMS/MMS.
The number of alerts: 2.
Text: ":)".

The New Alert Characteristic notification is received.
```

```
The notified value is:
The Category ID - Missed call.
The number of alerts: 1.

e
The New Alert Characteristic notification is received.
The notified value is:
The Category ID - Email.
The number of alerts: 5.
The Alert Notification Control Point Characteristic was written successfully.

s
Cy_BLE_ANSC_SetCharacteristicValue(CY_BLE_ANS_ALERT_NTF_CONTROL_POINT) routine Success

The New Alert Characteristic notification is received.
The notified value is:
The Category ID - SMS/MMS.
The number of alerts: 2.
The Alert Notification Control Point Characteristic was written successfully.

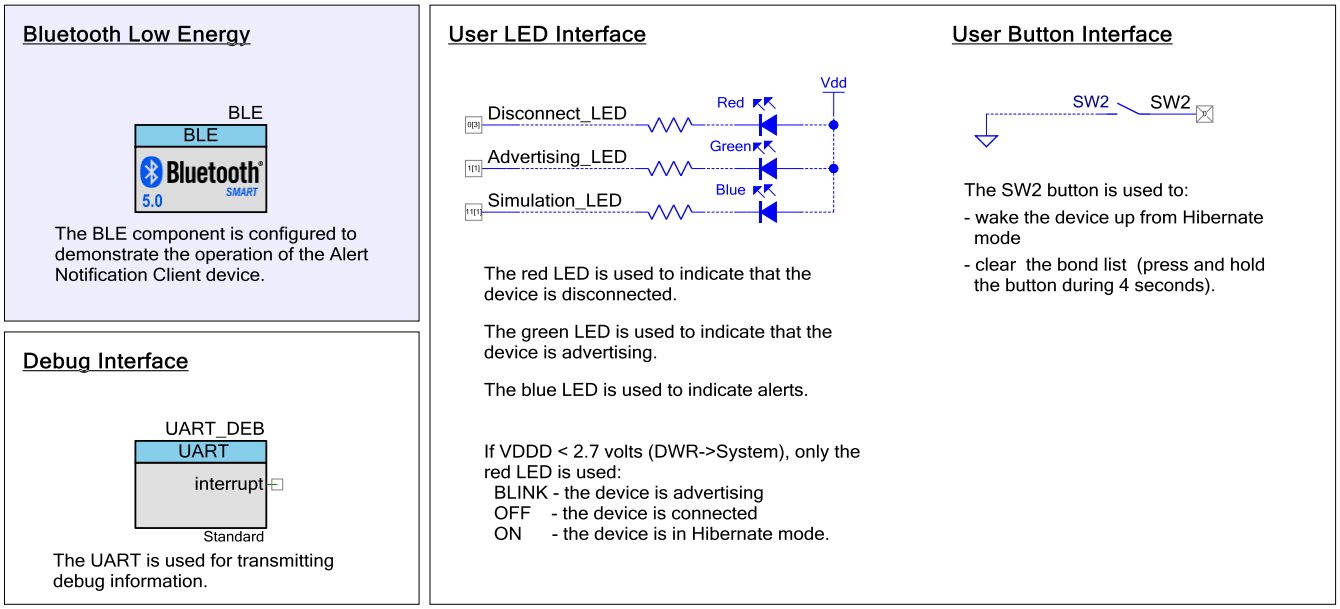
m
Cy_BLE_ANSC_SetCharacteristicValue(CY_BLE_ANS_ALERT_NTF_CONTROL_POINT) routine Success
The New Alert Characteristic notification is received.
The notified value is:
The Category ID - Missed call.
The number of alerts: 1.
The Alert Notification Control Point Characteristic was written successfully.

d
CY_BLE_EVT_GATT_DISCONNECT_IND: 3, 7
CY_BLE_EVT_GAP_DEVICE_DISCONNECTED: bdHandle=7, reason=16, status=0
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
The Advertisement is enabled.
```

Design and Implementation

Figure 1 shows the top design schematic.

Figure 1. BLE Alert Notification Client Code-Example Schematic



The example project uses two callback functions – `AppCallBack()` and `AnsServiceAppEventHandler()`. One callback function (`AppCallBack()`) is required for receiving generic events from the BLE Stack and the second (`AnsServiceAppEventHandler()`) – for receiving events from the Alert Notification Service. After a startup, the BLE, UART, and ISR Components are initialized. After the initialization, the BLE Component begins operating and the RGB LED starts blinking green. This indicates that the device has started advertising and it is available for the connection with a Central device.

Bonding

The example project uses authentication and encryption, so it requires a bonding procedure to establish a protected connection. The device will initiate the bonding procedure automatically after it is connected to the Server. The Client side terminal emulator will display the passkey to be entered in the Server.

```
CY_BLE_EVT_GAP_PASSKEY_DISPLAY_REQUEST: 392742
```

Enter the passkey on your Server device.

Note that the passkey prompted by the terminal emulator will be different. The passkey above is shown only as an example.

After entering a 6-digit passkey on the Server device, the following message should be displayed on the terminal emulator:

```
CY_BLE_EVT_GAP_ENCRYPT_CHANGE: 0
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: security:2, bonding:1, ekeySize:10, authErr 0
```

The bonding is completed.

This message indicates that a secure connection between two devices was established.

Alert Notifications

After successful bonding, the Server is successfully connected to the Alert Notification Client, and the Client is ready for the alert notification. Make sure that the peer device Server (Operating System or Application) supports ANS – the “The peer device supports ANS” message on the Terminal emulator will confirm it. Use the ‘0’ command to enable notifications for the New Alert Characteristic. After that, send the ‘t’ command to enable notifications for all categories in the Alert Notification Control Point Characteristic. Next, use the ‘s’, ‘m’ or ‘e’ commands to request the Server to send immediate notification for the SMS/MMS, Email or missed call categories. A message should be displayed on the terminal emulator:

```
The New Alert Characteristic notification is received.
The notified value is:
The Category ID - Email.
The number of alerts: 5.
The Alert Notification Control Point Characteristic was written successfully.

The New Alert Characteristic notification is received.
The notified value is:
The Category ID - SMS/MMS.
The number of alerts: 2.
The Alert Notification Control Point Characteristic was written successfully.

The New Alert Characteristic notification is received.
The notified value is:
The Category ID - Missed call.
The number of alerts: 1.
The Alert Notification Control Point Characteristic was written successfully.
```

Pin Assignments

Pin assignments and connections required on the development board for the supported kits are in Table 2.

Table 2. 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
Simulation_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Components and Settings

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

Table 3. 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 the operation of the Alert Notification Client and Server 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 Simulation_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 to have the following:

- Public Device Address: 00A050-00010d
- Device name: Alert Handler
- Appearances: Generic Watch
- Security Level: Authenticated pairing with encryption
- I/O capabilities: Keyboard and Display
- Bonding requirements: Bonding

Figure 2. General Settings

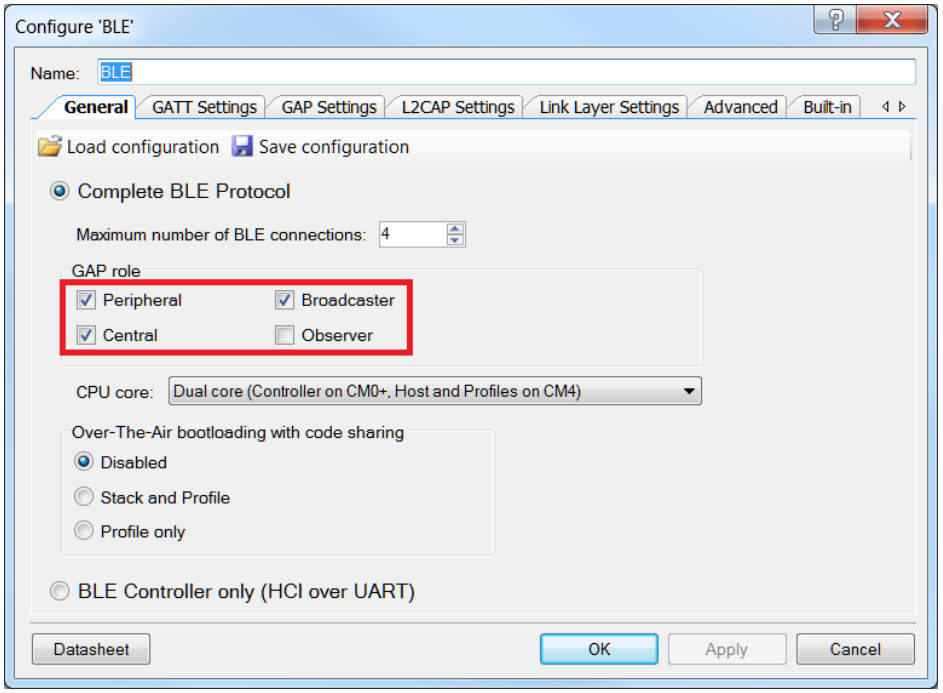


Figure 3. GATT Settings

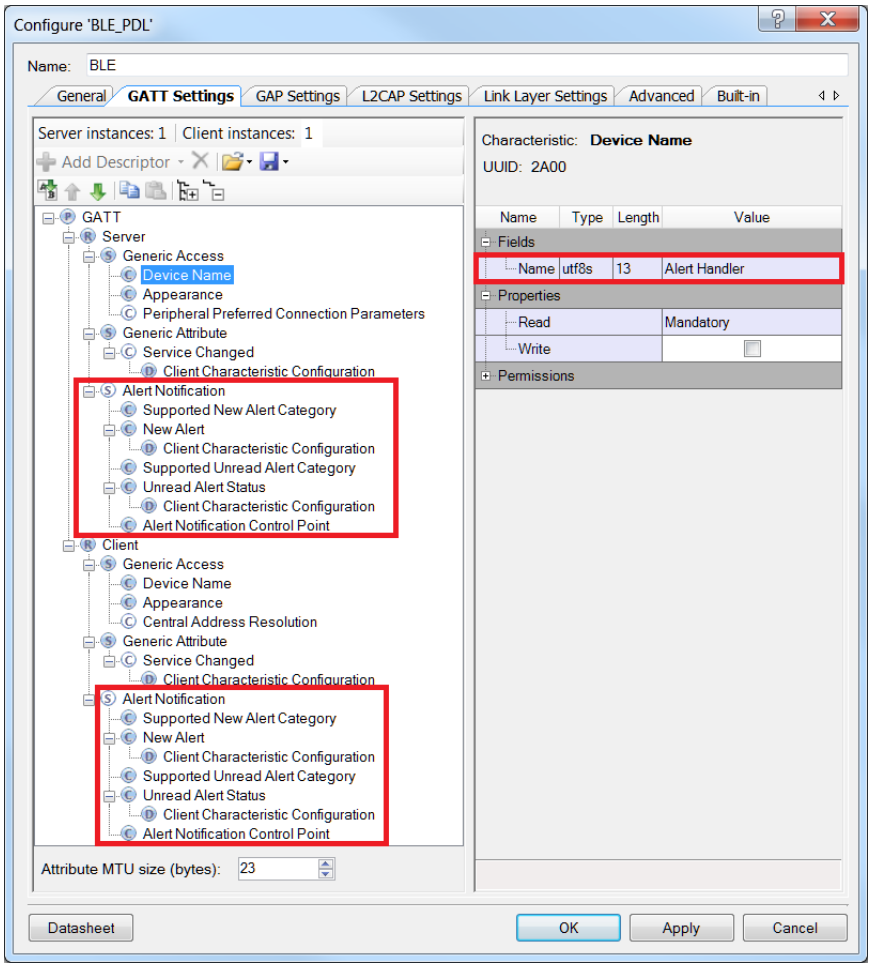


Figure 4. GAP Settings

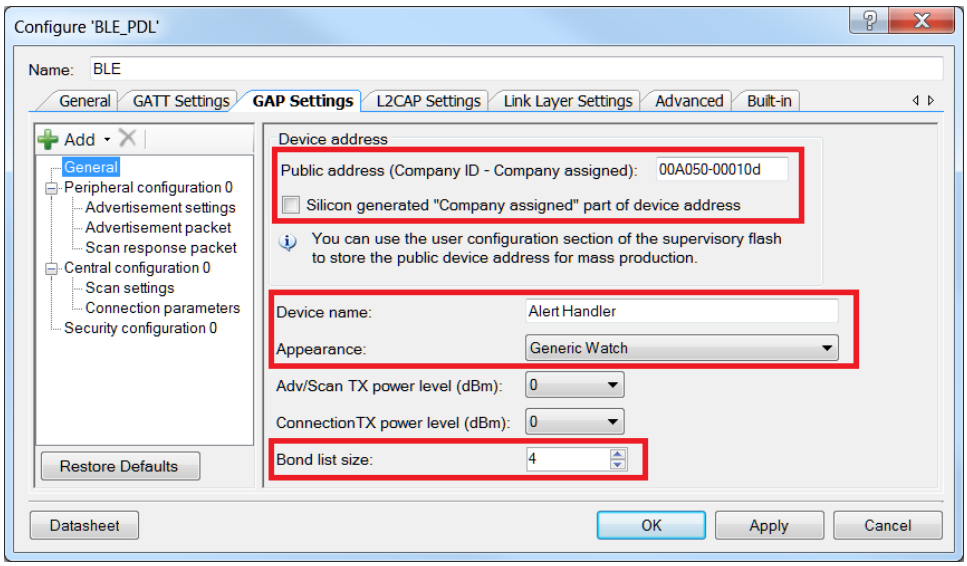


Figure 5. GAP Settings > Advertisement Settings

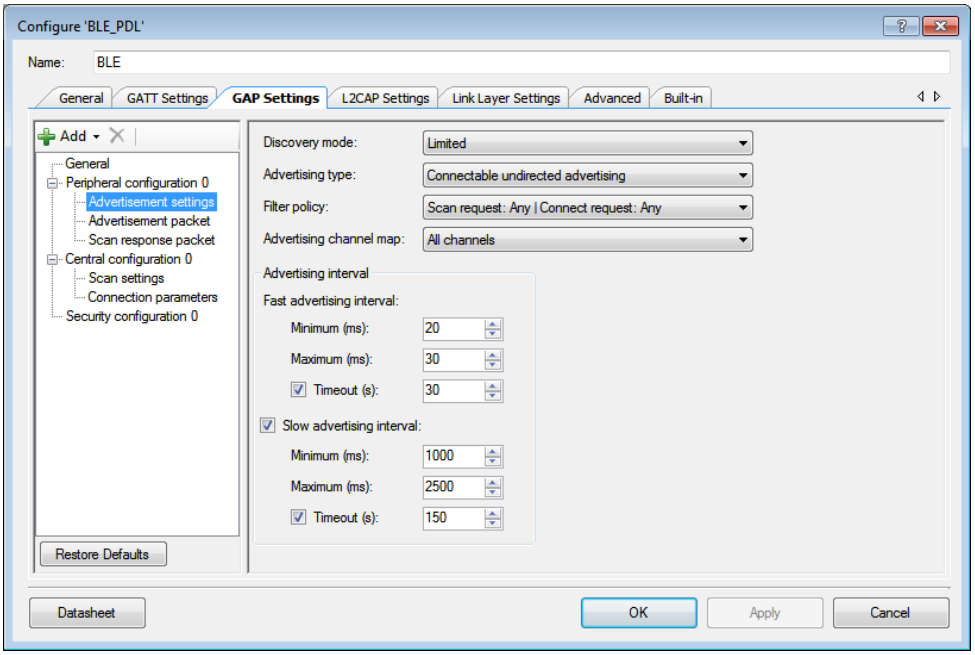


Figure 6. GAP Settings > Advertisement Packet

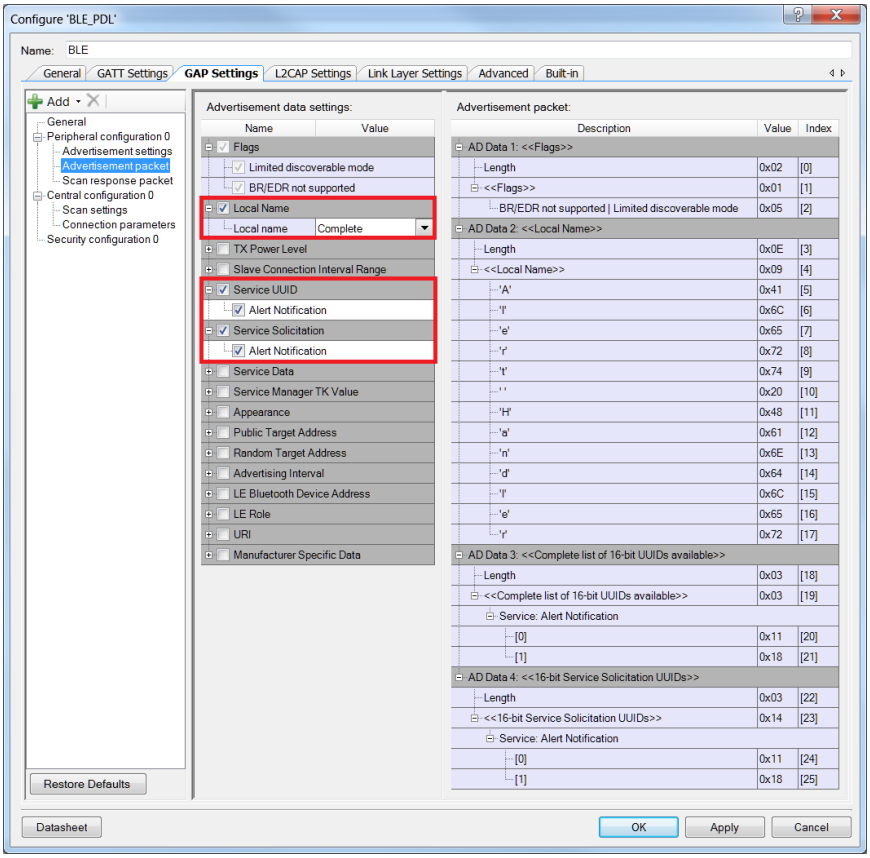


Figure 7. GAP Settings > Scan Settings

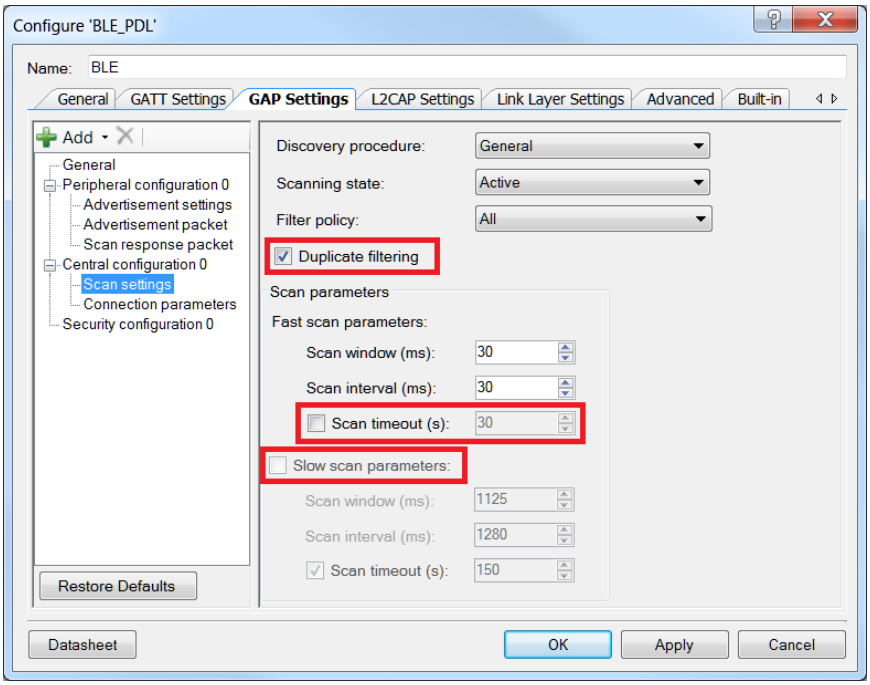
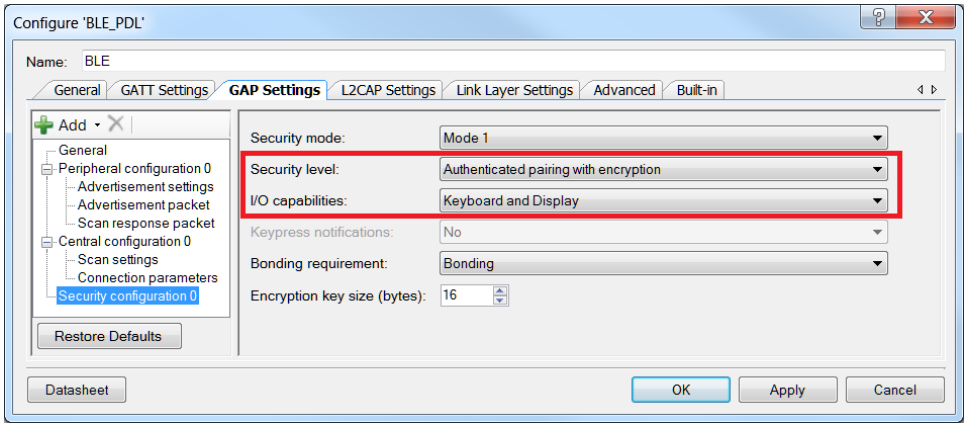


Figure 8. GAP Settings > Security Configuration



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.

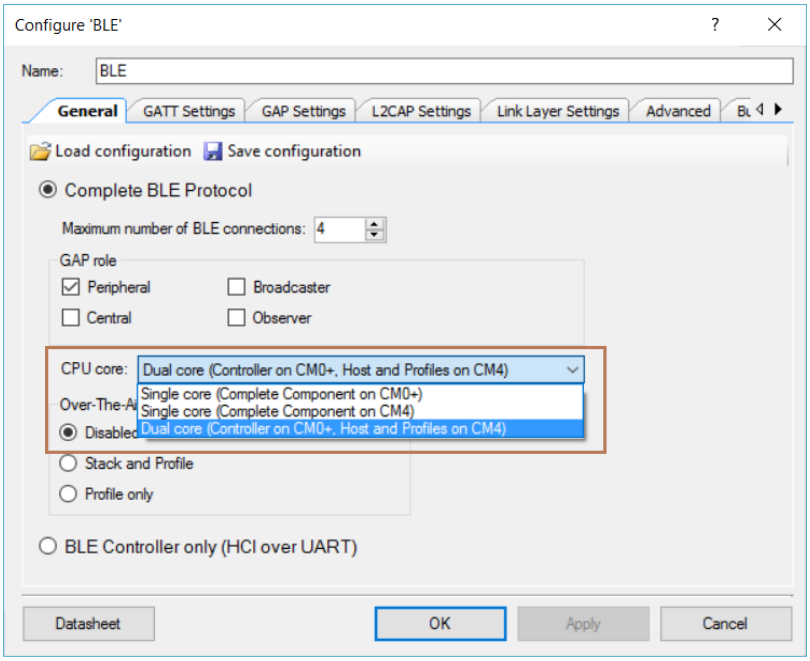
The BLE example structure allows easy switching between different CPU cores options. Here are some important points to remember:

- 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.

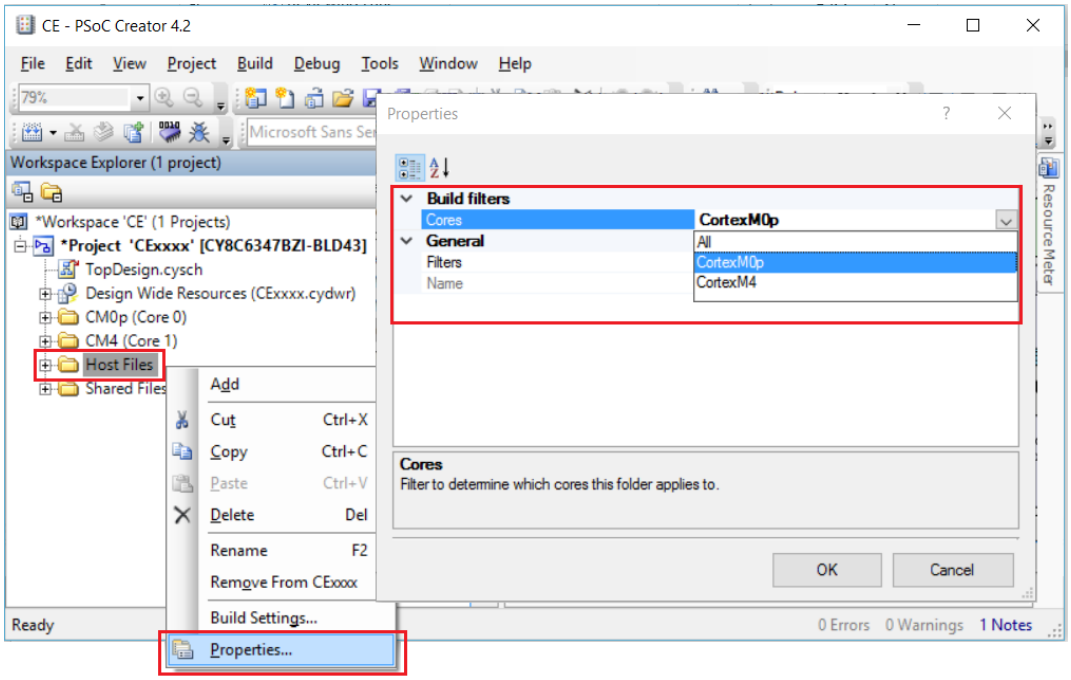
Figure 9. Select CPU Core



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 10.

   - 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

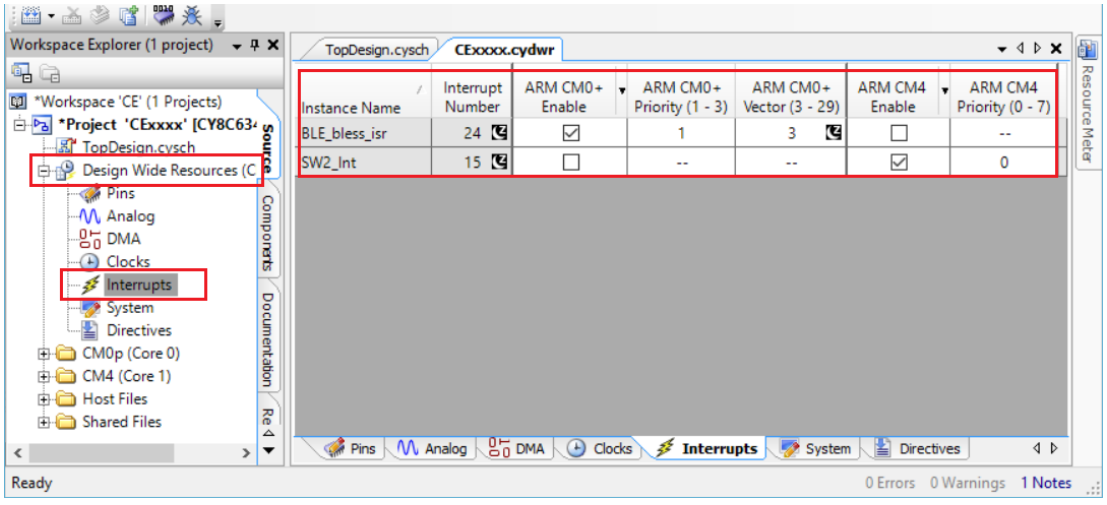
Figure 10. Change Core Properties



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 11. 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: CE217631 - BLE Alert Notification Profile with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17631

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6086712	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 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, 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.