CE217632 - BLE Apple Notification Client with PSoC 6 MCU with BLE Connectivity

Objective

This example project demonstrates the Bluetooth Low Energy (BLE) Apple Notification Client application workflow.

Overview

The design demonstrates the core functionality of the BLE Component configured as a BLE Apple Notification Service (ANCS) device in the GATT Client role. The application uses the BLE Apple Notification Center Service in the GATT Client mode to communicate with a BLE Apple Notification Center Server (iPhone, iPod, 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 Apple Notification Client device can be connected to any Apple gadget that supports the BLE Apple Notification Center Service configured as the GAP Central role and GATT Server. To connect to the Apple Notification Client device, go to **Settings → Bluetooth** and find ANCS while the device is advertising (green LED is blinking).

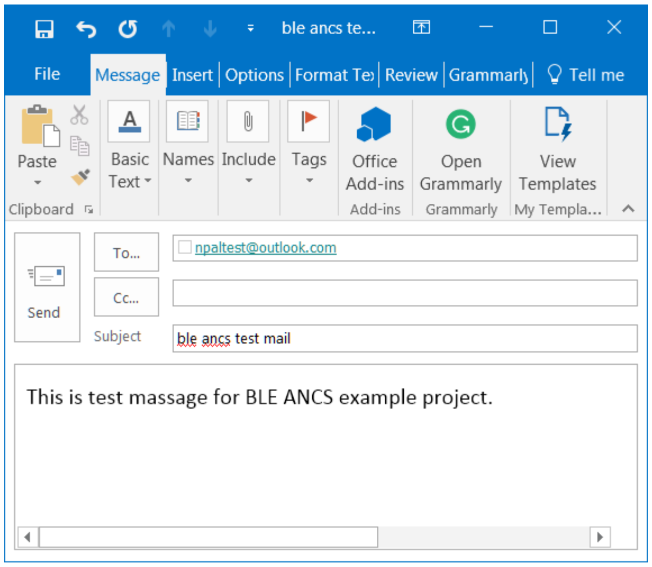
The red LED will turn ON after a fast and slow advertisement period elapses to indicate that no client is connected to the device and the device falls asleep in the Hibernate mode. To wake up a device, press **SW2**. When the Central device connects successfully, the Apple Notification Client discovers the server's GATT database (including Apple Notification Center Server's characteristics and descriptors) and enables the notifications.

The Apple Notification Client displays unread emails, incoming calls (also text messages, pending missed calls, and so on) from the Viber application (and declines them), and regular incoming calls on iPhone (and accepts or declines them). Pressing **SW2** once every second will decline the incoming calls. Pressing **SW2** twice very second will accept the incoming calls. The BLE Stack timer is used to make the LEDs blink.

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 blinking while the device is advertising, and the output in the terminal window.
5. Create an outlook account on your iPod (or iPhone) to operate with a regular Mail application.
6. Configure notifications (**Settings → Notifications → Mail → Allow Notifications**) on the iPod. (The project currently supports up to 10 notifications. You can easily change this number by modifying `CYBLE_ANCS_NS_CNT`).
7. Run the project (connect any terminal software to an appropriate COM port to observe the workflow).
8. Connect to the device: go to the **Settings → Bluetooth**, find ANCS and tap on it, and then tap on **Pair** in the dialog window.
9. Now the device should discover the server (iPod) and wait for notifications.
10. Send an email to the new outlook account, as shown in Figure 1.

Figure 1. Sending Email to the New Outlook Account.



npaltest@outlook.com

ble ancs test mail

This is test massage for BLE ANCS example project.

6. In the terminal window, observe the device receiving emails, for example:

```
EventID: Notification Added
EventFlags: Negative Action,
CategoryCount: 1
NotificationUID: 0x00000001
CategoryID: Email

Email from:
         npaltest@outlook.com
Subject:
         ble ancs test mail
Message:
         This is a test message for the BLE ANCS example project.

EventID: Notification Removed
EventFlags: Negative Action,
CategoryCount: 1
NotificationUID: 0x0000001e
CategoryID: Email
```

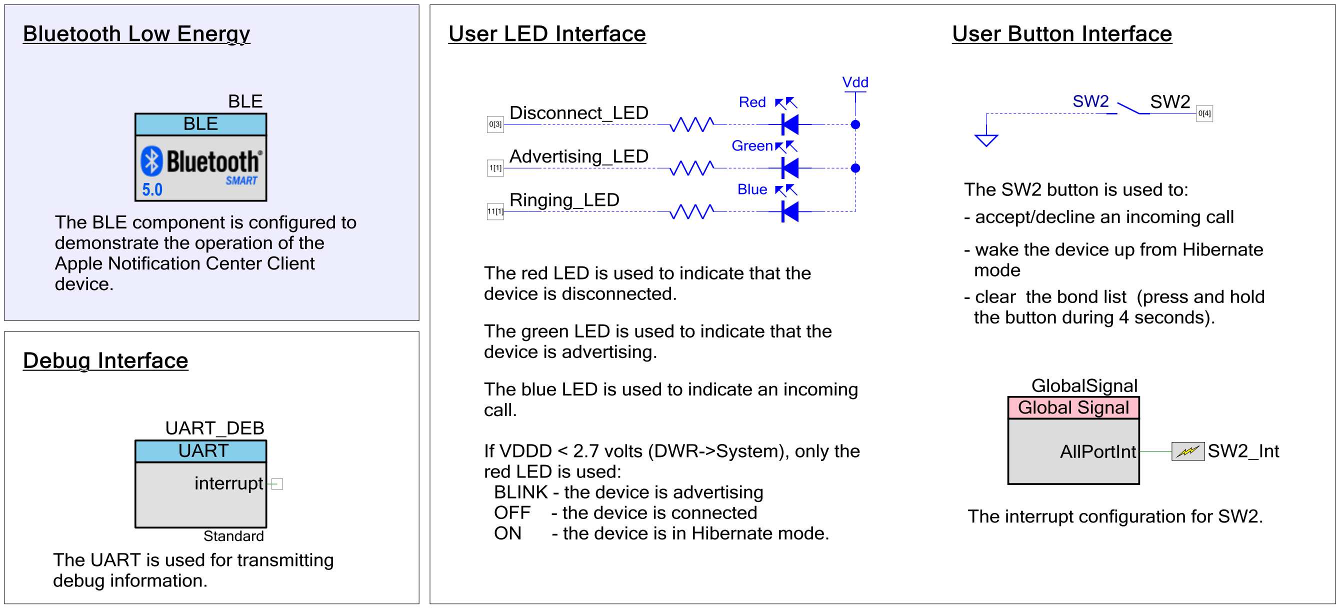
13. Use the UART debug port to view verbose messages:
    a. The code example ships with the UART debug port enabled. To disable it, set the macro `DEBUG_UART_ENABLED` in *common.h* to `DISABLED` and rebuild the code.
    b. The output of the debug serial port looks like the sample below:

```
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: 0, 18( ms), 0, 48
CY_BLE_EVT_GATTS_XCNHG_MTU_REQ, final mtu= 2050
CY_BLE_EVT_GAP_AUTH_REQ: security=0x3, bonding=0x1, ekeySize=0x10, err=0x0
CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: security:2, bonding:1, ekeySize:10, authErr 0
ENCRYPT_CHANGE: 1
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: security: 0x1, bonding: 0x1, ekeySize: 0x10, authErr 0x0
StartDiscovery
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ, attHandle: 3
The discovery is complete.
The discovered services:
Service with UUID 0x1800 has handle range from 0x1 to 0x5
Service with UUID 0x1801 has handle range from 0x6 to 0x9
Service with UUID 0x0 has handle range from 0x23 to 0x2c.
The peer device supports ANS.

Notification Source characteristic CCCD write request: 0x01
Data Source characteristic CCCD write request: 0x01
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ, attHandle: 3
```

Design and Implementation

Figure 2 shows the top design schematic.

Figure 2. BLE Apple Notification Service Code Example Schematic


Bluetooth Low Energy
BLE
BLE
Bluetooth SMART 5.0
The BLE component is configured to demonstrate the operation of the Apple Notification Center Client device.
Debug Interface
UART_DEB
UART
interrupt
Standard
The UART is used for transmitting debug information.
User LED Interface
Vdd
Disconnect_LED
Red
Advertising_LED
Green
Ringing_LED
Blue
The red LED is used to indicate that the device is disconnected.
The green LED is used to indicate that the device is advertising.
The blue LED is used to indicate an incoming call.
If VDDD < 2.7 volts (DWR->System), only the red LED is used:
BLINK - the device is advertising
OFF - the device is connected
ON - the device is in Hibernate mode.
User Button Interface
SW2
SW2
The SW2 button is used to:
- accept/decline an incoming call
- wake the device up from Hibernate mode
- clear the bond list (press and hold the button during 4 seconds).
GlobalSignal
Global Signal
AllPortInt
SW2_Int
The interrupt configuration for SW2.

The project demonstrates the functionality of the BLE component configured as a BLE Apple Notification Center Service Client.

After a startup, the device initializes the BLE Component. In this project, two callback functions are required for the BLE operation. Callback function `AppCallBack()` is required to receive generic events from the BLE Stack and the service-specific callback function `AncsCallBack()` is required for Apple Notification Center 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 shown in Figure 7. The BLE Component stops advertising as soon as a 180-second advertising period expires.

Pin Assignments

Table 1 lists the pin assignments and connections required on the development board for the supported kits.

Table 1. Pins 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
Ringing_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 Apple Notification Center Client device.	See the Parameter Settings 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 Ringing_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 can 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

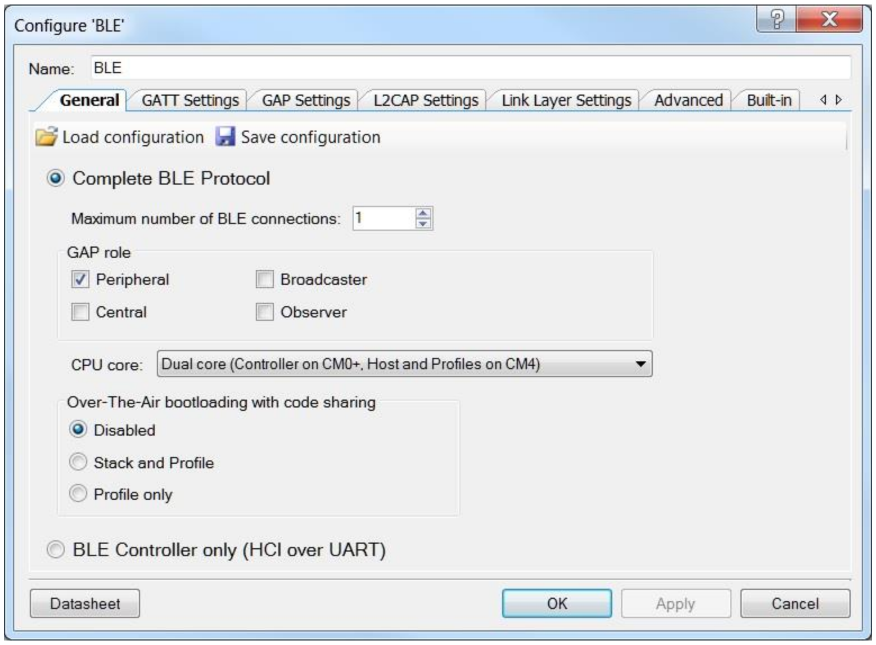
For information on the hardware resources used by a Component, see the Component datasheet.

Parameter Settings

The BLE Component is configured as an Apple Notification Center Client in the GAP Peripheral role. The BLE Component is also configured to have:

- Public Device Address: 00A050-00001E
- Device name: ANCS Client
- Appearances: Generic Watch
- Security Level: Unauthenticated pairing with encryption
- I/O capabilities: No Input No Output
- Bonding requirements: Bonding

Figure 3. General Settings



Configure 'BLE'

Name: BLE

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

Load configuration | Save configuration

(●) Complete BLE Protocol

Maximum number of BLE connections: 1

GAP role

[✓] Peripheral | [ ] Broadcaster

[ ] Central | [ ] Observer

CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)

Over-The-Air bootloading with code sharing

(●) Disabled

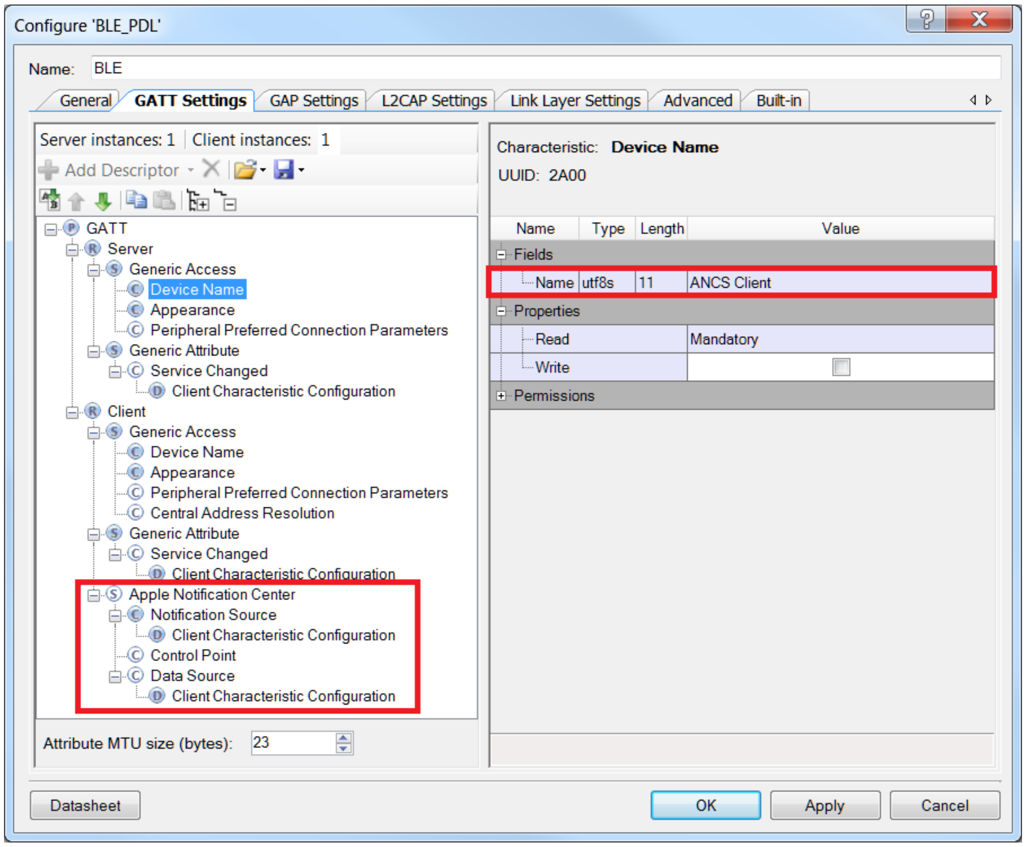
( ) Stack and Profile

( ) Profile only

( ) BLE Controller only (HCI over UART)

Datasheet | OK | Apply | Cancel

Figure 4. GATT Settings



Configure 'BLE_PDL'

Name: BLE

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

Server instances: 1 | Client instances: 1

Add Descriptor

- GATT
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
  - Client
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
      - Central Address Resolution
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
    - Apple Notification Center
      - Notification Source
        - Client Characteristic Configuration
      - Control Point
      - Data Source
        - Client Characteristic Configuration

Attribute MTU size (bytes): 23

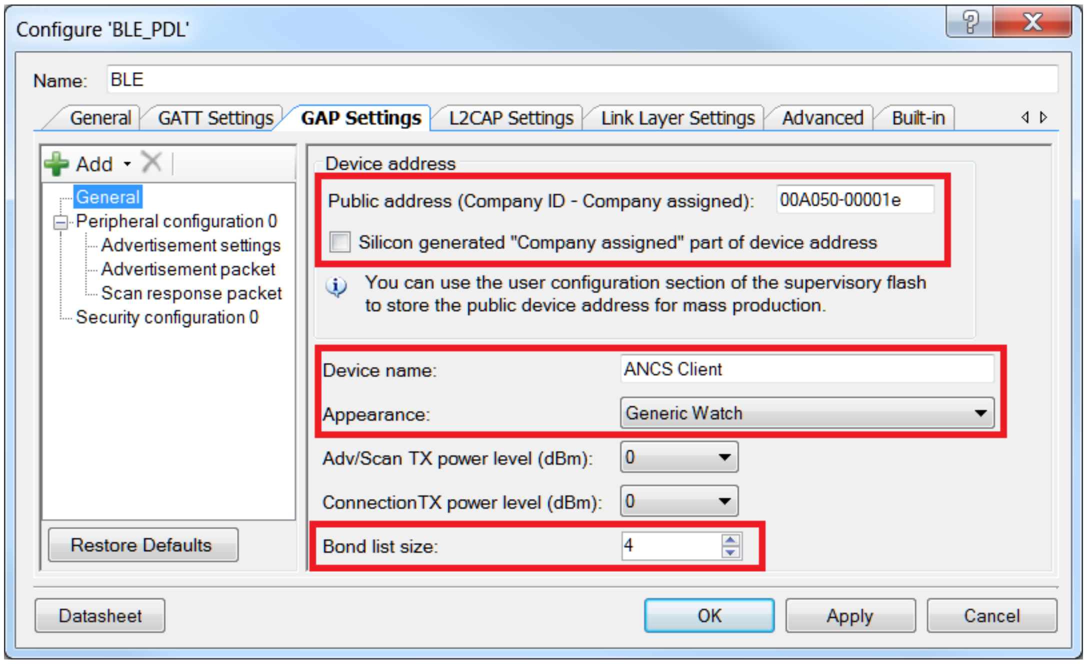
Characteristic: **Device Name**

UUID: 2A00

Name	Type	Length	Value
Fields			
Name	utf8s	11	ANCS Client
Properties			
Read			Mandatory
Write			[ ]
Permissions			

Datasheet | OK | Apply | Cancel

Figure 5. GAP Settings



Configure 'BLE_PDL'

Name: BLE

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

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

Security configuration 0

Device address

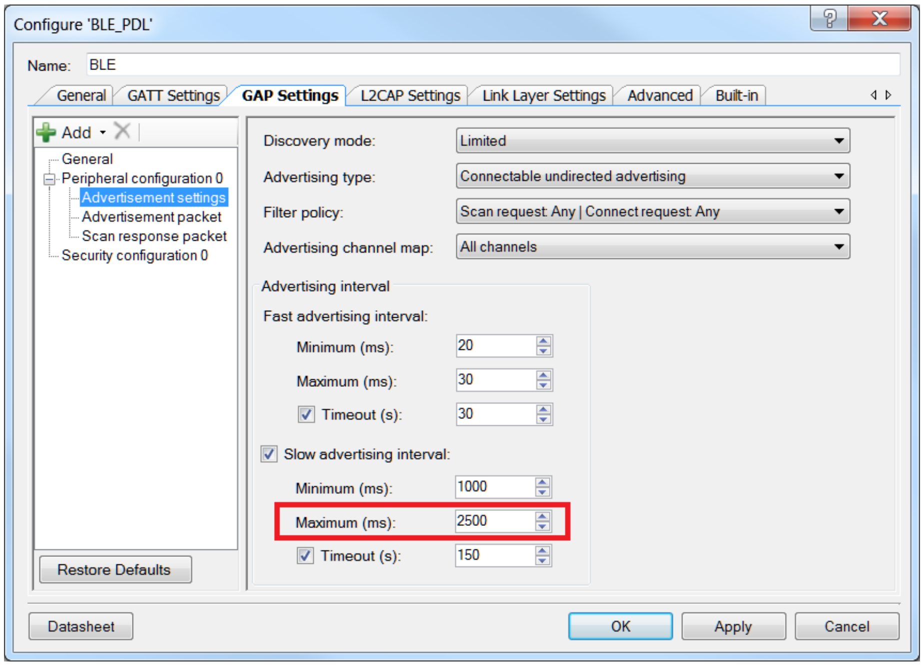
Public address (Company ID - Company assigned): 00A050-00001e

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name:	ANCS Client
Appearance:	Generic Watch
Adv/Scan TX power level (dBm):	0
ConnectionTX power level (dBm):	0
Bond list size:	4

Figure 6. GAP Settings → Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

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

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

Security configuration 0

Discovery mode:	Limited
Advertising type:	Connectable undirected advertising
Filter policy:	Scan request: Any | Connect request: Any
Advertising channel map:	All channels

Advertising interval

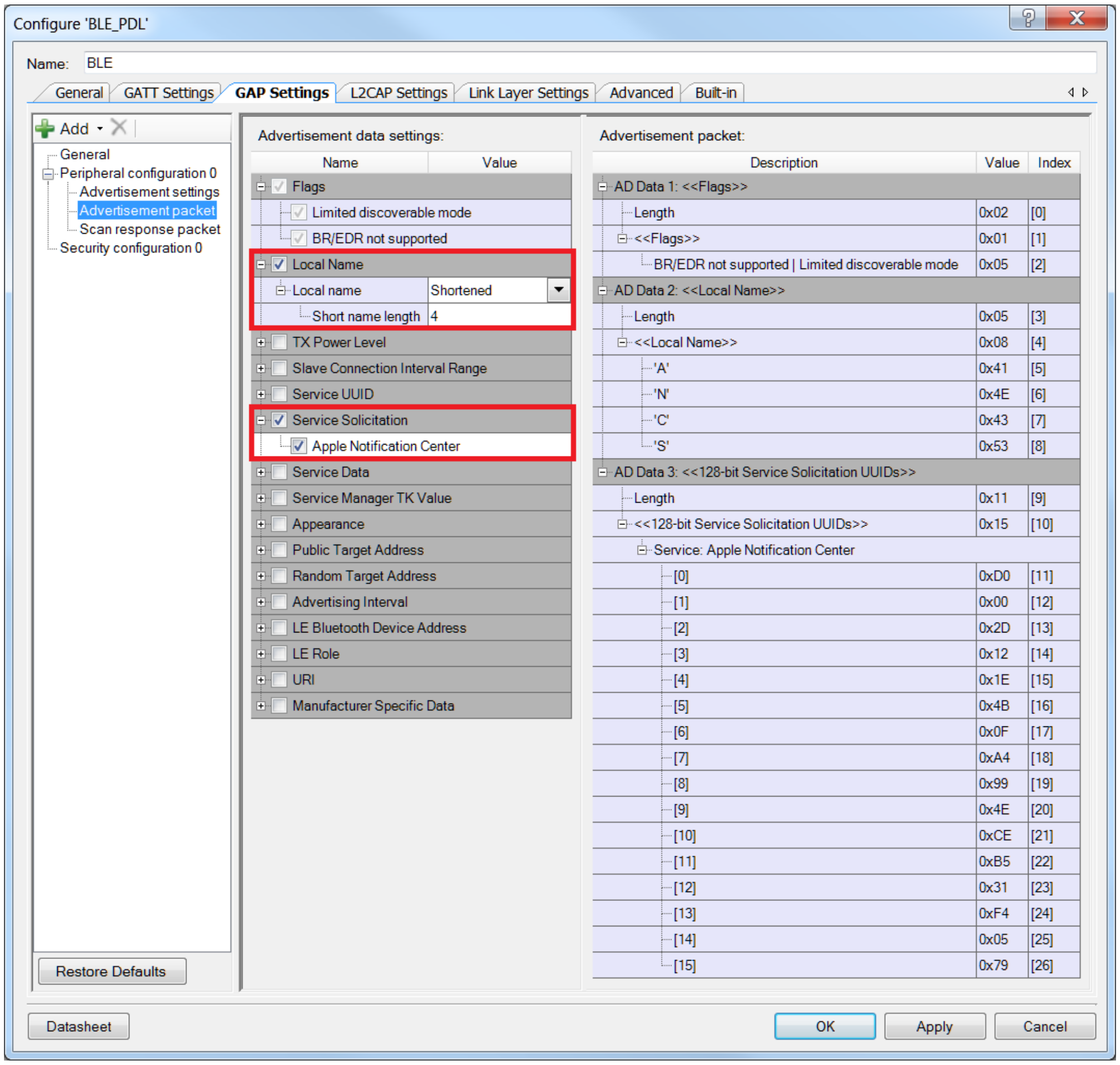
Fast advertising interval:

Minimum (ms):	20
Maximum (ms):	30
☑ Timeout (s):	30

☑ Slow advertising interval:

Minimum (ms):	1000
Maximum (ms):	2500
☑ Timeout (s):	150

Figure 7. GAP Settings → Advertisement Packet



Name: BLE

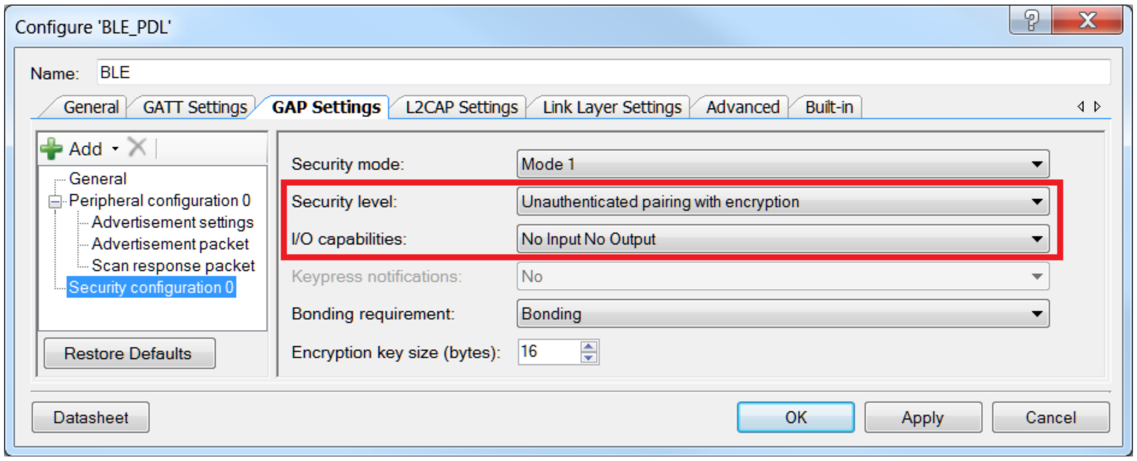
Advertisement data settings:

Name	Value
✓ Flags	
✓ Limited discoverable mode	
✓ BR/EDR not supported	
✓ Local Name	
Local name	Shortened
Short name length	4
TX Power Level	
Slave Connection Interval Range	
Service UUID	
✓ Service Solicitation	
✓ Apple Notification Center	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	
Advertising Interval	
LE Bluetooth Device Address	
LE Role	
URI	
Manufacturer Specific Data	

Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported | Limited discoverable mode	0x05	[2]
AD Data 2: <<Local Name>>		
Length	0x05	[3]
<<Local Name>>	0x08	[4]
'A'	0x41	[5]
'N'	0x4E	[6]
'C'	0x43	[7]
'S'	0x53	[8]
AD Data 3: <<128-bit Service Solicitation UUIDs>>		
Length	0x11	[9]
<<128-bit Service Solicitation UUIDs>>	0x15	[10]
Service: Apple Notification Center		
[0]	0xD0	[11]
[1]	0x00	[12]
[2]	0x2D	[13]
[3]	0x12	[14]
[4]	0x1E	[15]
[5]	0x4B	[16]
[6]	0x0F	[17]
[7]	0xA4	[18]
[8]	0x99	[19]
[9]	0x4E	[20]
[10]	0xCE	[21]
[11]	0xB5	[22]
[12]	0x31	[23]
[13]	0xF4	[24]
[14]	0x05	[25]
[15]	0x79	[26]

Figure 8. GAP Settings → Security



Name: BLE

Security mode: Mode 1
Security level: Unauthenticated pairing with encryption
I/O capabilities: No Input No Output
Keypress notifications: No
Bonding requirement: Bonding
Encryption key size (bytes): 16

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.

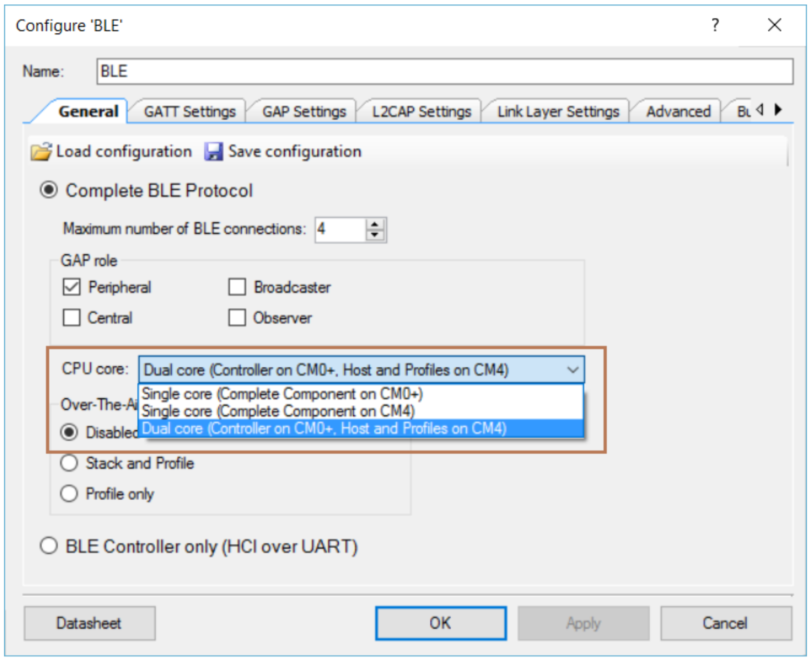
Important to remember:

- 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 9. Select CPU Core


Configure 'BLE'

Name: BLE

General | GATT Settings | GAP Settings | L2CAP Settings | Link Layer Settings | Advanced | Bu

Load configuration | Save configuration

(●) Complete BLE Protocol

Maximum number of BLE connections: 4

GAP role
[✓] Peripheral [ ] Broadcaster
[ ] Central [ ] Observer

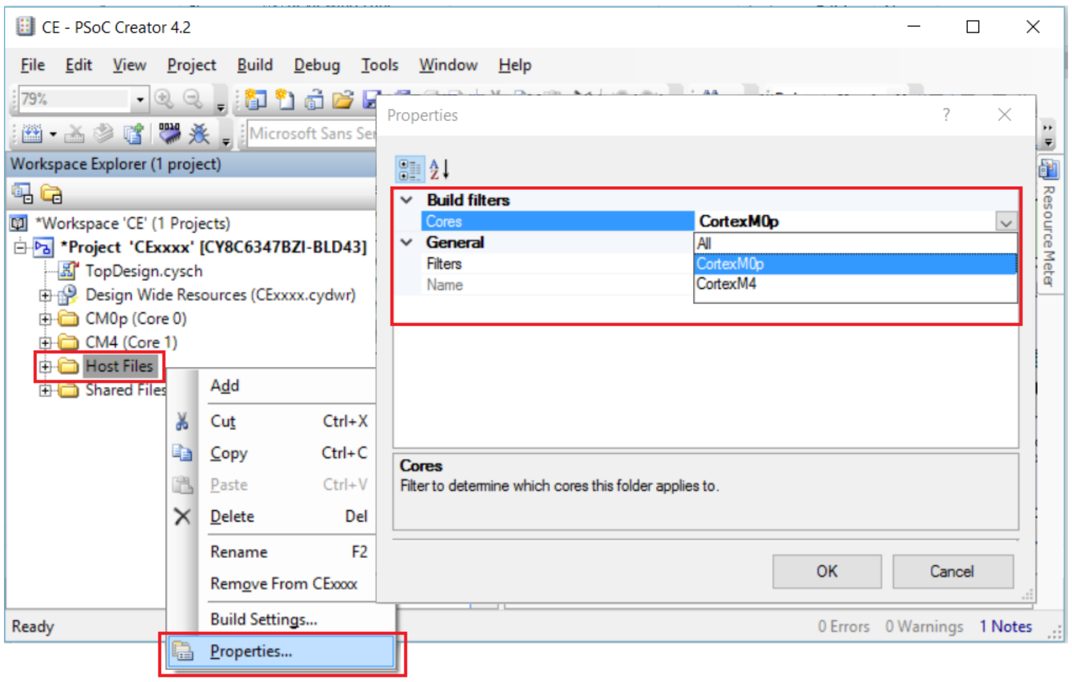
CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)
- Single core (Complete Component on CM0+)
- Single core (Complete Component on CM4)
- Dual core (Controller on CM0+, Host and Profiles on CM4)

Over-The-Ai
(●) Disabled
( ) Stack and Profile
( ) Profile only

( ) BLE Controller only (HCI over UART)

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 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



Build filters

Cores	CortexM0p

Cores options: All, CortexM0p, CortexM4

General

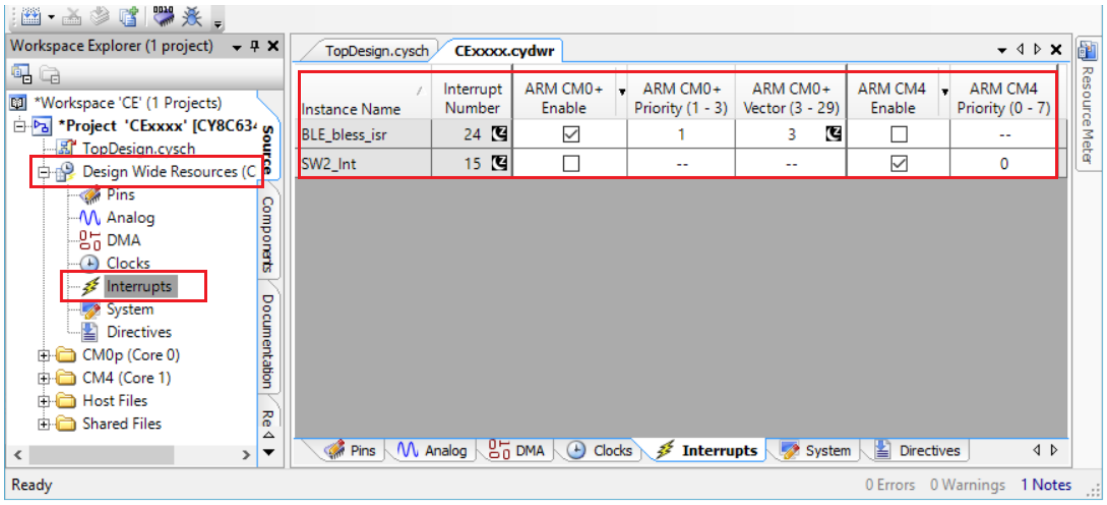
Filters	
Name	

Cores
Filter to determine which cores this folder applies to.

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 11. Assign Interrupts


Workspace Explorer (1 project)
*Workspace 'CE' (1 Projects)
*Project 'CExxxx' [CY8C63
TopDesign.cysch
Design Wide Resources (C
Pins
Analog
DMA
Clocks
Interrupts
System
Directives
CM0p (Core 0)
CM4 (Core 1)
Host Files
Shared Files
TopDesign.cysch
CExxxx.cydwr
Pins Analog DMA Clocks Interrupts System Directives
Ready
0 Errors 0 Warnings 1 Notes
Source Components Documentation Resource Meter

Instance Name	Interrupt Number	ARM CM0+ Enable	ARM CM0+ Priority (1 - 3)	ARM CM0+ Vector (3 - 29)	ARM CM4 Enable	ARM CM4 Priority (0 - 7)
BLE_bless_isr	24	✓	1	3	☐	--
SW2_Int	15	☐	--	--	✓	0

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: CE217632 - BLE Apple Notification Client with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17632

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6086741	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

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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