CE217644 - BLE Time Sync with PSoC 6 MCU with BLE Connectivity

Objective

This example demonstrates the Current Time Service (CTS) in GATT Client and GAP Peripheral role.

Overview

The design demonstrates the Time profile operation of the BLE Component. The Time Sync example uses the BLE Time Profile (configured for the GAP Peripheral role as a Time Client) with one instance of the Current Time Service (CTS) to demonstrate the capability of time synchronization with an external Time Server. The project also contains one instance of the Reference Time Update Service (RTUS) and one instance of the Next DST Change Service (NDCS), but they are not used in the example.

The Time Client operates with other devices that implement the Time Server Profile role. The device uses Limited Discovery mode during, which it is visible for BLE GATT Servers. 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. Refer to the kit guide to ensure the kit is configured correctly.

Connect the BLE Pioneer Kit to the computer's USB port.

LED Behavior

If the V_{DDD} voltage is set to less than 2.7 V in the DWR settings **System** tab, only the red LED is used. The red LED blinks to indicate that the device is advertising. The red LED is OFF when a device is connected to a peer device. When the device is in Hibernate mode, the red LED stays ON.

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

Software Setup

Terminal Tool

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

Operation

For operation, the code example requires a peer Time Server device configured in the GAP Central role. Currently, the Time Server in the GAP Central role is only supported by iOS (iPod, iPhone). Note that in the CySmart™ app also doesn't support the Time Server in the GAP Central role, so you should use the standard iOS menu to connect to the Time Client.

To connect to the Timer Sync 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.

The blue LED is turned ON when the CTS service is discovered on the peer device. The Time Sync enables notifications, reads the Current Time Characteristic and automatically synchronizes the RTC with the time and date received from the BLE Time Server.

The project reads RTC time and date and shows it in the terminal every second. Refer to Time Profile and Current Time Service specifications for more details.

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. Use the UART debug port to view verbose messages. Note that 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.
6. Use a **Settings** > **Bluetooth** iOS menu (iPod or iPhone) to connect to the "**BLE Watch**" (Time Sync Code Example).
7. Observe in the UART terminal that date and time corresponds to the values in the iOS device.
8. The output of the debug serial port looks like the sample below:

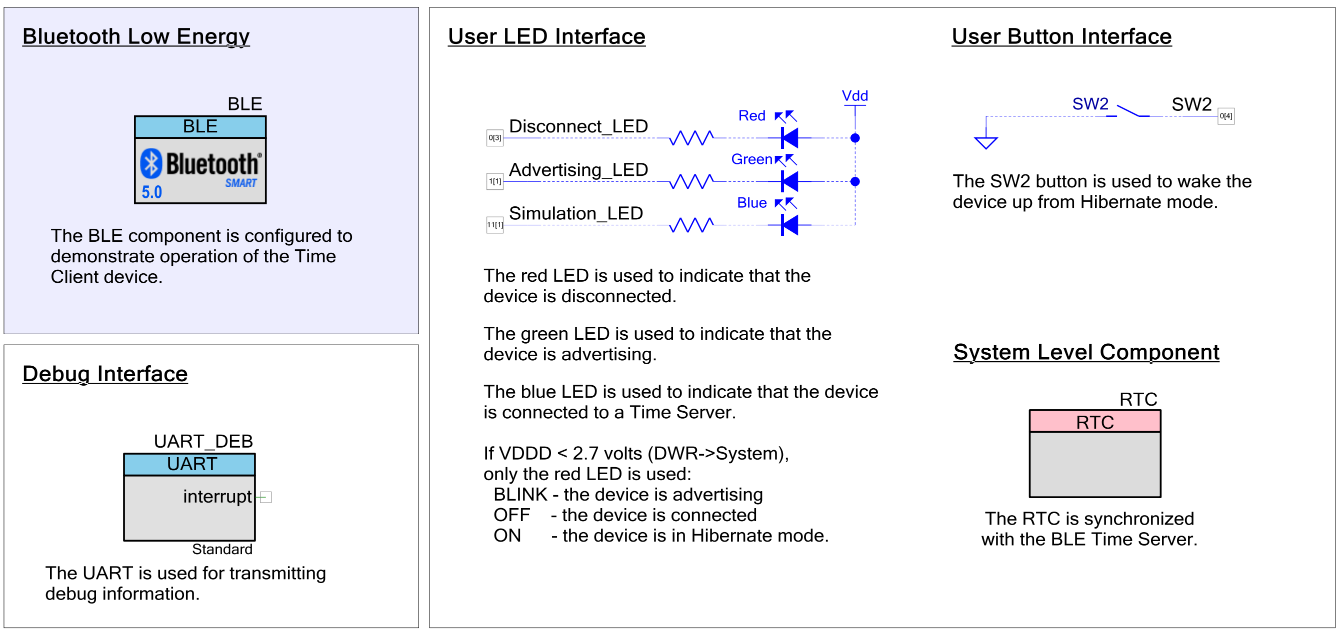
BLE Time Sync code example
Current time: 00:00:00 Sun 01.01.2000
CY_BLE_EVT_STACK_ON
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: public:00a050000005
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE
Current time: 00:00:01 Sun 01.01.2000
CY_BLE_EVT_GATT_CONNECT_IND: 0, 8
CY_BLE_EVT_GAP_DEVICE_CONNECTED: 0, 18(ms), 0, 48
Current time: 00:00:02 Sun 01.01.2000
CY_BLE_EVT_GAP_AUTH_REQ: security=0x2, bonding=0x1, ekeySize=0x10, err=0x0
CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: security:2, bonding:1, ekeySize:10, authErr 0
CY_BLE_EVT_GAP_PASSKEY_DISPLAY_REQUEST. Passkey is: 006990.
Please enter the passkey on your Server device.
CY_BLE_EVT_GATTS_XCNHG_MTU_REQ 0, 8, final mtu= 23
Current time: 00:00:03 Sun 01.01.2000
Current time: 00:00:04 Sun 01.01.2000
Current time: 00:00:05 Sun 01.01.2000
Current time: 00:00:06 Sun 01.01.2000
Current time: 00:00:07 Sun 01.01.2000
ENCRYPT_CHANGE: 1
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: security: 0x2, bonding: 0x1, ekeySize: 0x10, authErr 0x0
Start Discovery
CY_BLE_EVT_PENDING_FLASH_WRITE
Store bonding data, status: 140001, pending: 1
Other event: 0x10005
Store bonding data, status: 140001, pending: 3
Store bonding data, status: 140001, pending: 3
Store bonding data, status: 140001, pending: 3
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ, attHandle: 3
Current time: 00:00:08 Sun 01.01.2000
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
CY_BLE_EVT_GATTC_DISC_SKIPPED_SERVICE
Discovery complete: attId=0, bdHandle=8
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 0x1805 has handle range from 0x18 to 0x1d
Service with UUID 0x1807 has handle range from 0x0 to 0x0
Service with UUID 0x1806 has handle range from 0x0 to 0x0
Enable Current Time Notification, apiResult: 0

Current time: 00:00:09 Sun 01.01.2000
CTS Current Time CCCD was written successfully
Get Current Time char value, apiResult: 0x0
CTS characteristic read response received
Server time: 12:17:21 06.04.2018
Update RTC time
Cy_RTC_SetDateAndTimeDirect API status: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ, attHandle: 3
Current time: 12:17:21 Mon 06.04.2018
Current time: 12:17:22 Mon 06.04.2018
Current time: 12:17:23 Mon 06.04.2018
Current time: 12:17:24 Mon 06.04.2018

Design and Implementation

Figure 1 shows the top design schematic.

Figure 1. BLE Time Sync Code Example Schematic



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

After a startup, the device performs BLE Component initialization. In this project, two callback functions are required for BLE operation. The `AppCallBack()` callback function is required to receive generic events from the BLE Stack; the service-specific callback `CtsCallBack()` is required for receiving events from the Current Time Service. Additional callback functions – `RtusCallBack ()` and `NdcsCallBack()` are optional in the Time profile. They are present only for the for user reference.

The `CY_BLE_EVT_STACK_ON` event indicates a successful initialization of the BLE stack. After this event is received, the Component starts advertising with the packet structure as configured in BLE Component Customizer. The BLE Component stops advertising after the 180-second advertising period expires.

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

While connected to the GATT Client and between the connection intervals, the device is put into Sleep mode.

The Real-Time Clock (RTC) Component is used in the project for keeping track of the time and date by the hardware real-time clock.

Pin Assignments

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

Table 1. Pin Assignment

Pin Name	Development Kit CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	Comment
\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 Time Client device.	See Parameter Settings
Digital Input Pin	SW2	The SW2 button is used to wake the device up from Hibernate mode.	**[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
Real-Time Clock	RTC	This Component is used for keeping track of the time and date by the hardware real-time clock.	**[General tab]** Enable Interrupts

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

Parameter Settings

Figure 2. General Settings

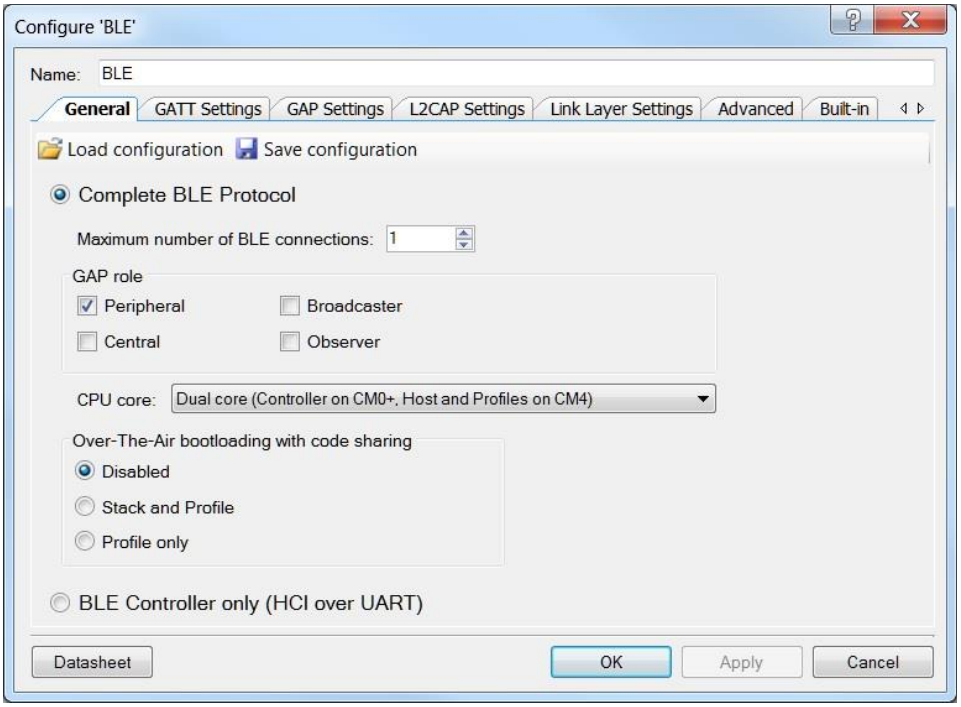


Figure 3. GATT Settings

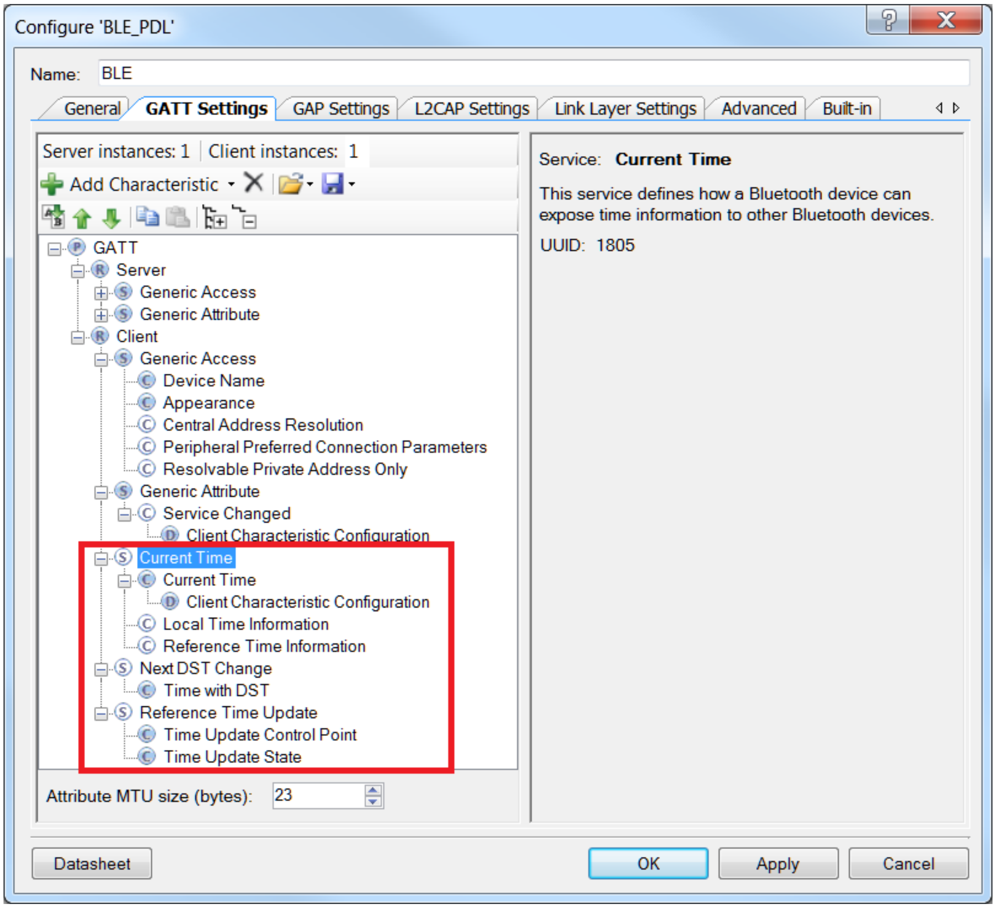


Figure 4. GAP Settings

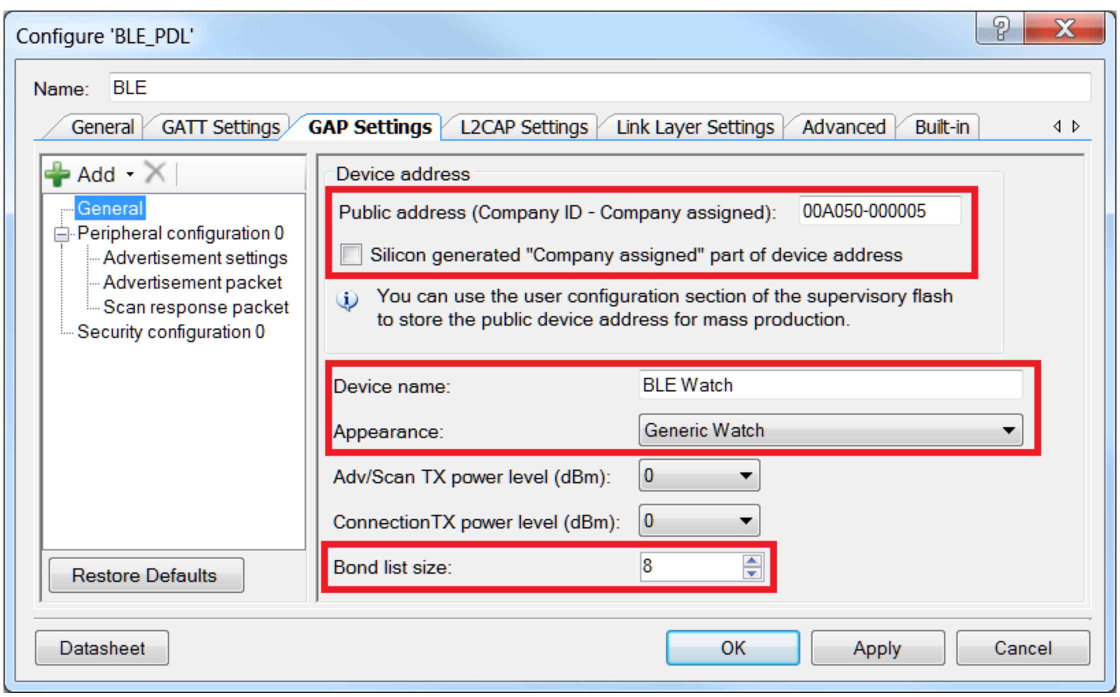


Figure 5. GAP Settings: Advertisement Settings

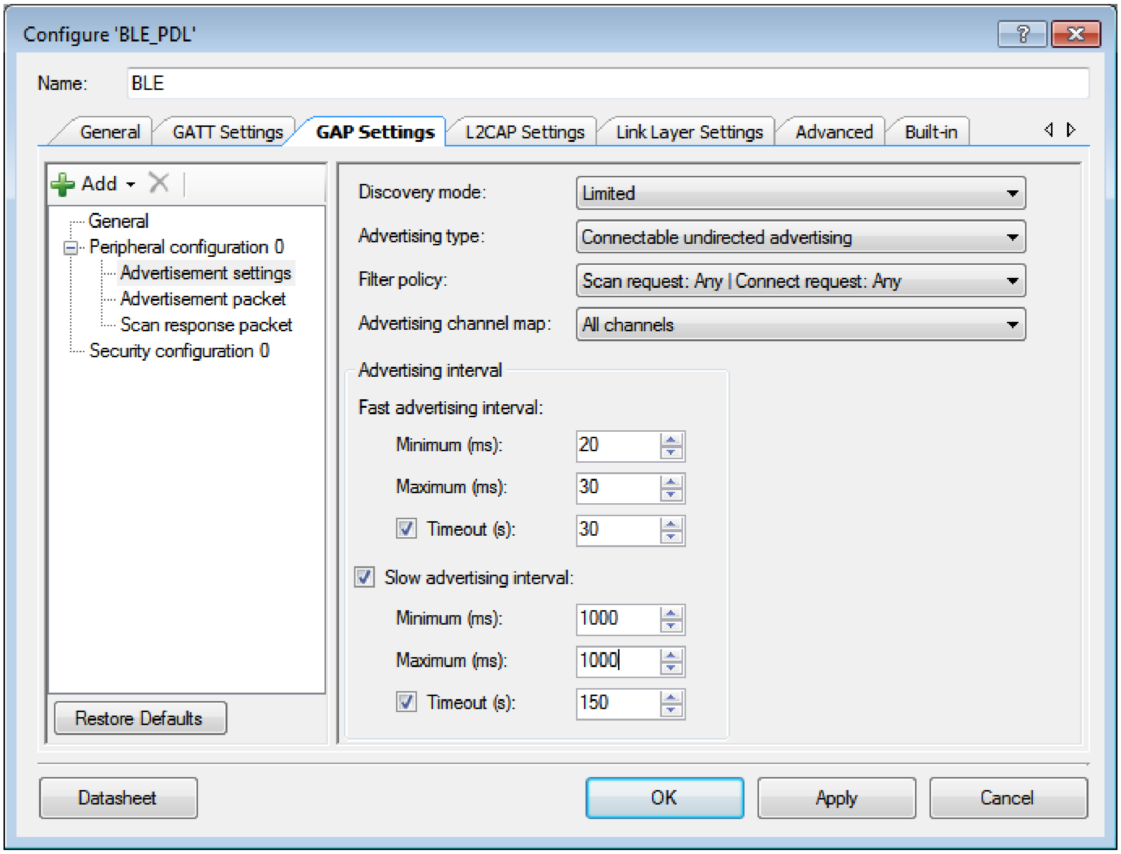


Figure 6. GAP Settings: Advertisement Packet

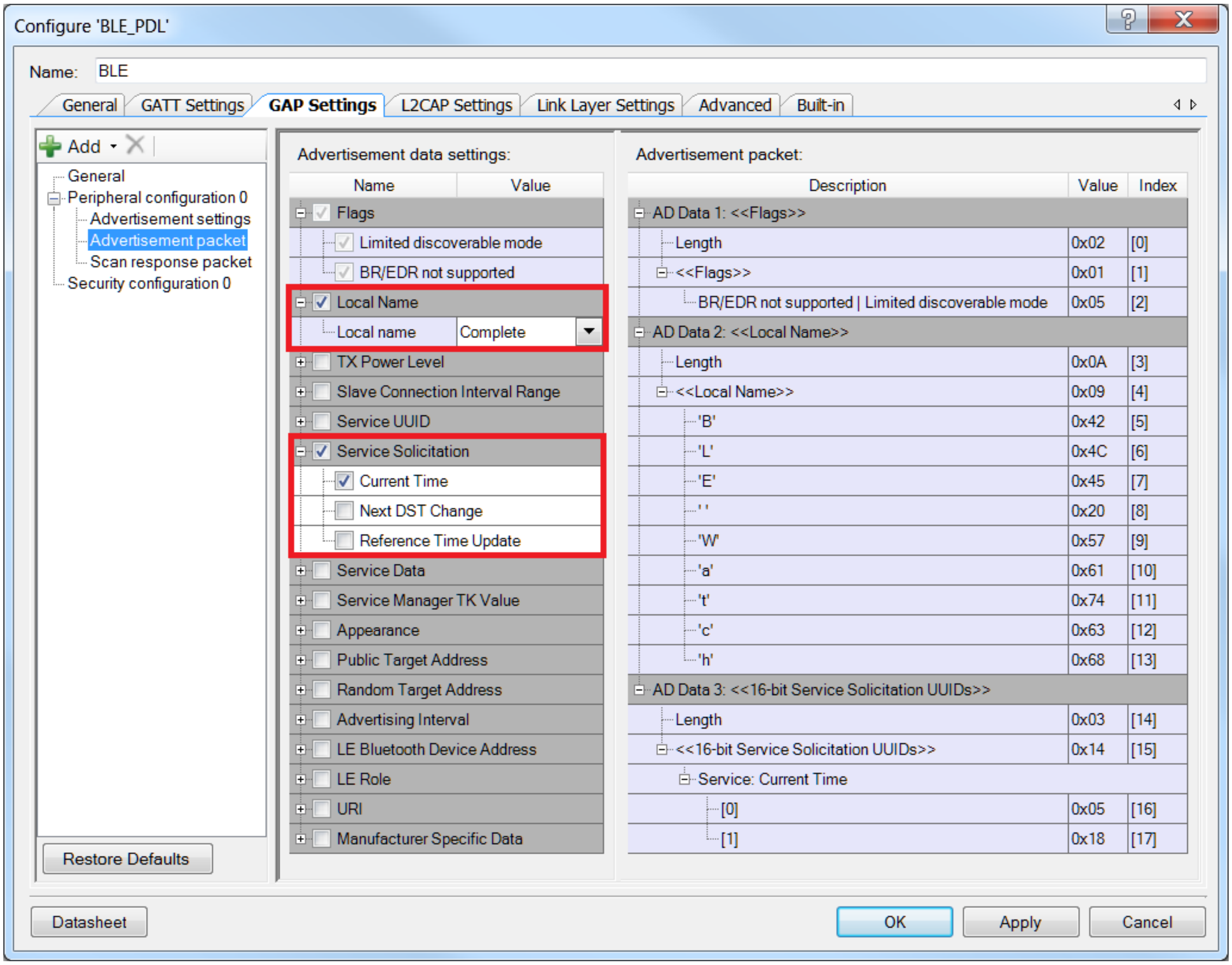
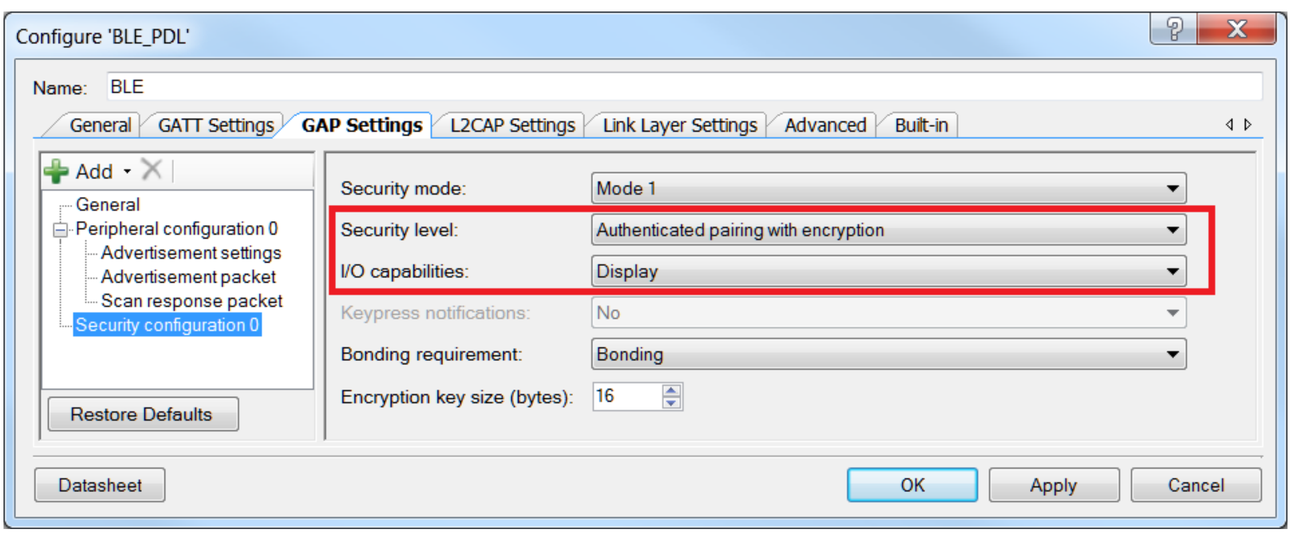


Figure 7. Security Settings



Switching the CPU Cores Usage

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

The BLE component has the CPU Core parameter that defines the cores usage. It can take the following values:

- Single core (Complete Component on CM0+) – only CM0+ will be used.
- Single core (Complete Component on CM4) – only CM4 will be used.
- Dual core (Controller on CM0+, Host and Profiles on CM4) – CM0+ and CM4 will be used: CM0+ for the Controller and CM4 for the Host and Profiles.

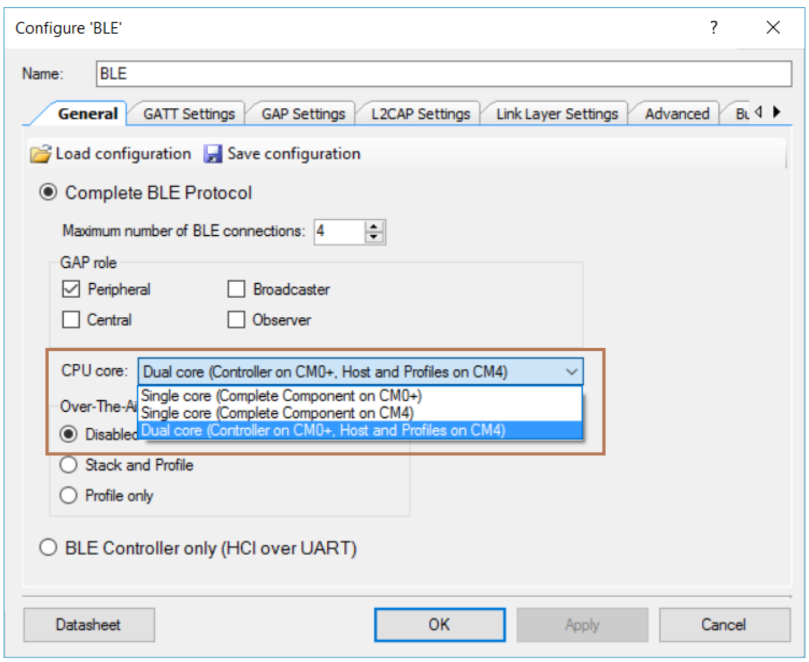
The BLE example structure allows easy switching between different CPU cores options. Important to remember:

- All application host-files must be run on the host core.
- The BLE subsystem (BLESS) interrupt must be assigned to the core where the controller runs.
- All additional interrupts (RTC, etc.) used in the example must be assigned to the host core.

Steps for switching the CPU Cores usage:

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

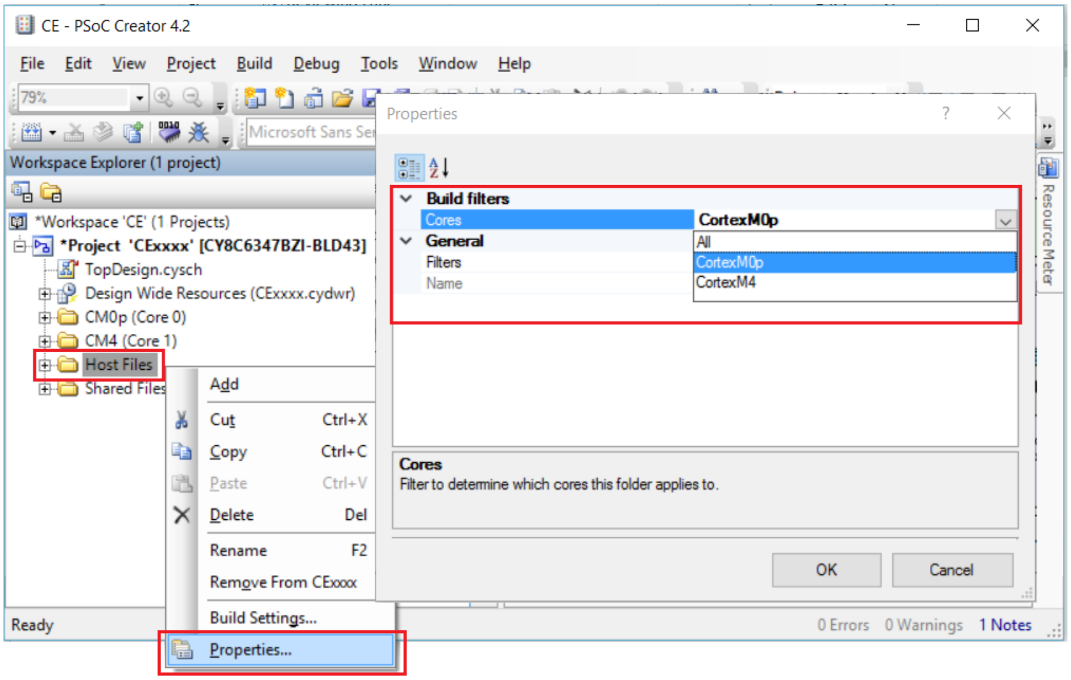
Figure 8. Select CPU Core



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 9.

   - 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

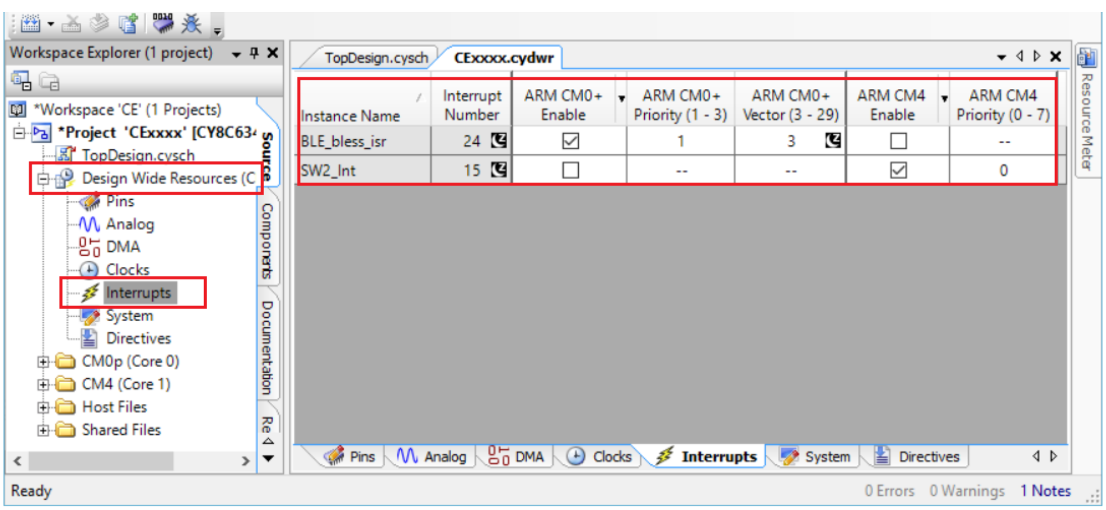
Figure 9. Change Core Properties



3. Assign BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to the appropriate core in **DWR** > **Interrupts** tab:

- for **Single core (Complete Component on CM0+)** option: BLE_bless_isr and peripheral interrupts on **CM0+**
- for **Single core (Complete Component on CM4)** option: BLE_bless_isr and peripheral interrupts on **CM4**
- for **Dual core (Controller on CM0+, Host and Profiles on CM4)** option: BLE_bless_isr interrupt on **CM0+,** other peripheral interrupts on **CM4**

Figure 10. Assign Interrupts



Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 BLE, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – Bluetooth® LE Test and Debug Tool		CySmart is a Bluetooth® LE host emulation tool for Windows PCs. The tool provides an easy-to-use Graphical User Interface (GUI) to enable the user to test and debug their Bluetooth LE peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE_PDL) Component		The Bluetooth Low Energy (BLE_PDL) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
Device Documentation		
PSoC® 6 MCU: PSoC 63 with BLE. Datasheet.		PSoC® 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		

Document History

Document Title: CE217644 - BLE Time Sync with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17644

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6091573	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 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