CE215118 – BLE Multi-Master Single Slave with PSoC 6 MCU with BLE Connectivity

Objective

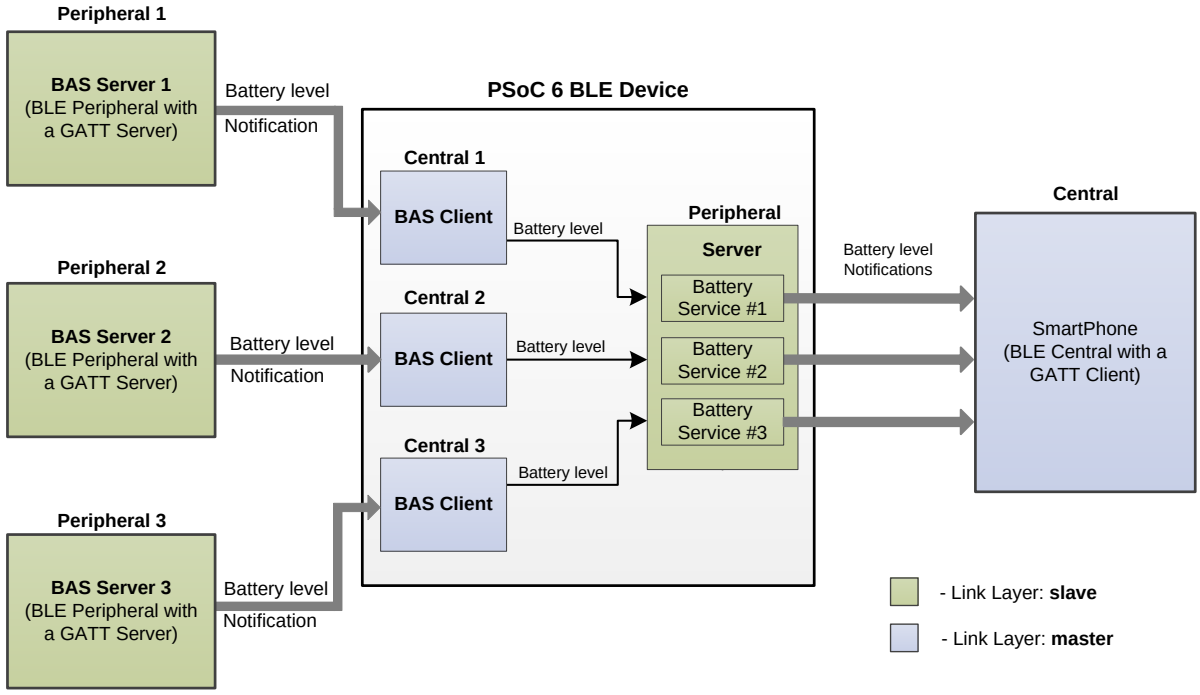
This example demonstrates how to configure the PSoC® 6 MCU with Bluetooth Low Energy (BLE) Connectivity device in simultaneous Multiple Master and Single Slave modes of operation.

Overview

The BLE Multi-Master Single Slave project is used in a pair with the CE215119 BLE Battery Level code examples for PSoC 6 MCU or PSoC 4 devices to demonstrate the operation in simultaneous Multiple Master and Single Slave modes. The Multi-Master Single Slave project uses three BLE Central connections and one Peripheral connection:

- The Central is configured as a Generic Attribute Profile (GATT) Client with a Battery Service that can communicate with a peer device in the Generic Access Profile (GAP) Peripheral and GATT Server roles. Use the existing CE215119 BLE Battery Level code examples for PSoC 6/PSoC 4 devices or an application that can simulate a GATT Server with a Battery Service as a peer device.
- The Peripheral is configured as a GATT Server with three Battery services. This configuration represents the battery level of the three Peripherals that the device is connected to. Figure 1 shows a block diagram of the Multi Master Single Slave.

Figure 1. Multi-Master Single-Slave



Requirements

Tool: PSoC Creator™ 4.2 or later

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

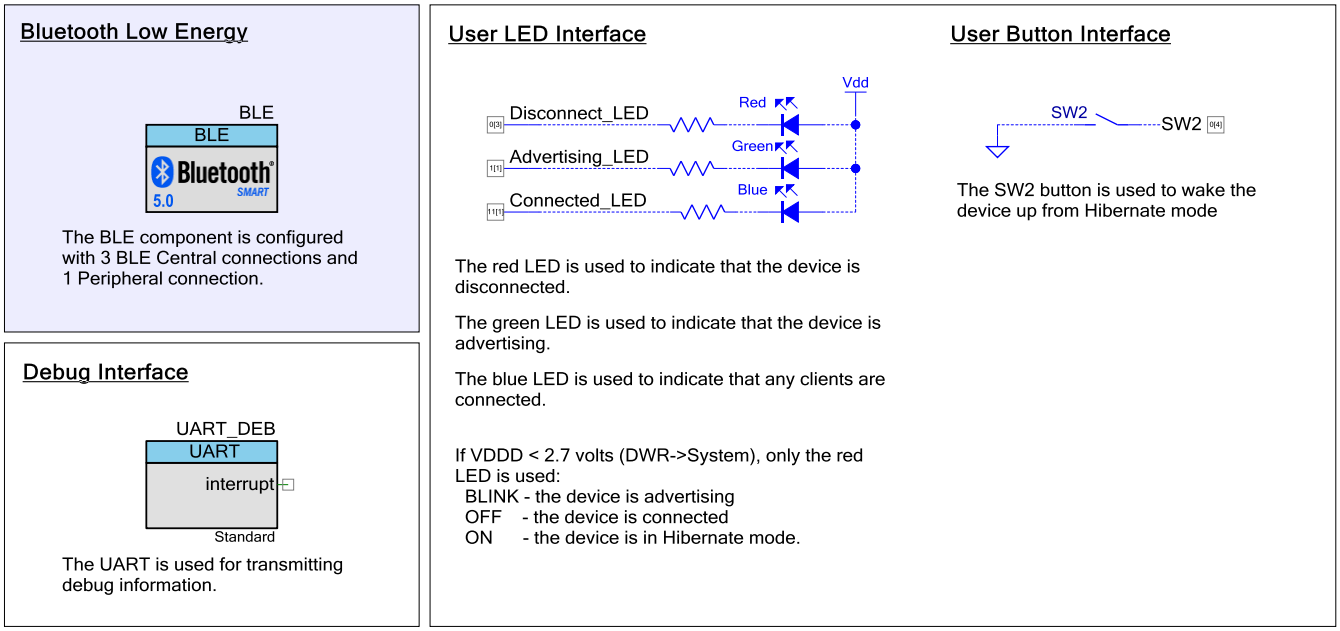
Associated Parts: All PSoC 6 MCU with BLE Connectivity (PSoC 6 BLE) parts

Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

Design

Figure 2 shows the top design schematic.

Figure 2. BLE Multi Master Single Slave Code Example Schematic



The project demonstrates how to configure the PSoC 6 BLE device in simultaneous Multiple Master and Single Slave modes of operation.

After the startup, the device initializes the BLE Component. To operate, the Component requires several callback functions in order to receive events from the BLE Stack. `AppCallBack()` is used to receive general BLE events. `BasCallBack()` is used to receive events specific to the service's attribute operations. The list of the terminal commands is in Table 1.

Table 1. Terminal Commands List

Command	Description
1	Stop scanning.
2	Stop advertising.
s	Start scanning for BLE devices.
c	Send a connect request to the peer device.
d	Send a disconnect request to the peer device.
p	Print the list of the connected devices.

These commands are prompted to the Terminal emulator when "h" is entered in the application.

Design Considerations

Using UART for debugging

Download and install a serial port communication program. Freeware such as Bray's Terminal and PuTTY are available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the device manager program in your PC, find a COM port that the kit is connected to, and note the port number.
3. Open the serial port communication program and select the COM port noted in Step 2.

4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the Putty configuration window. The default settings: Baud rate – 115200, Parity – None, Stop bits – 1 and Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.
5. Start communicating with the device as explained in the Operation section.

UART debugging can be disabled by setting the `DEBUG_UART_ENABLED` to `DISABLED` in the *common.h* file.

LED Behavior for VDDD Voltage < 2.7 V

If the VDDD 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.

Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core/ 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+ core will be used.
- **Single core (Complete Component on CM4)** – only CM4 core will be used.
- **Dual core (Controller on CM0+, Host and Profiles on CM4**) – both cores will be used: CM0+ for the Controller and CM4 for the Host and Profiles.

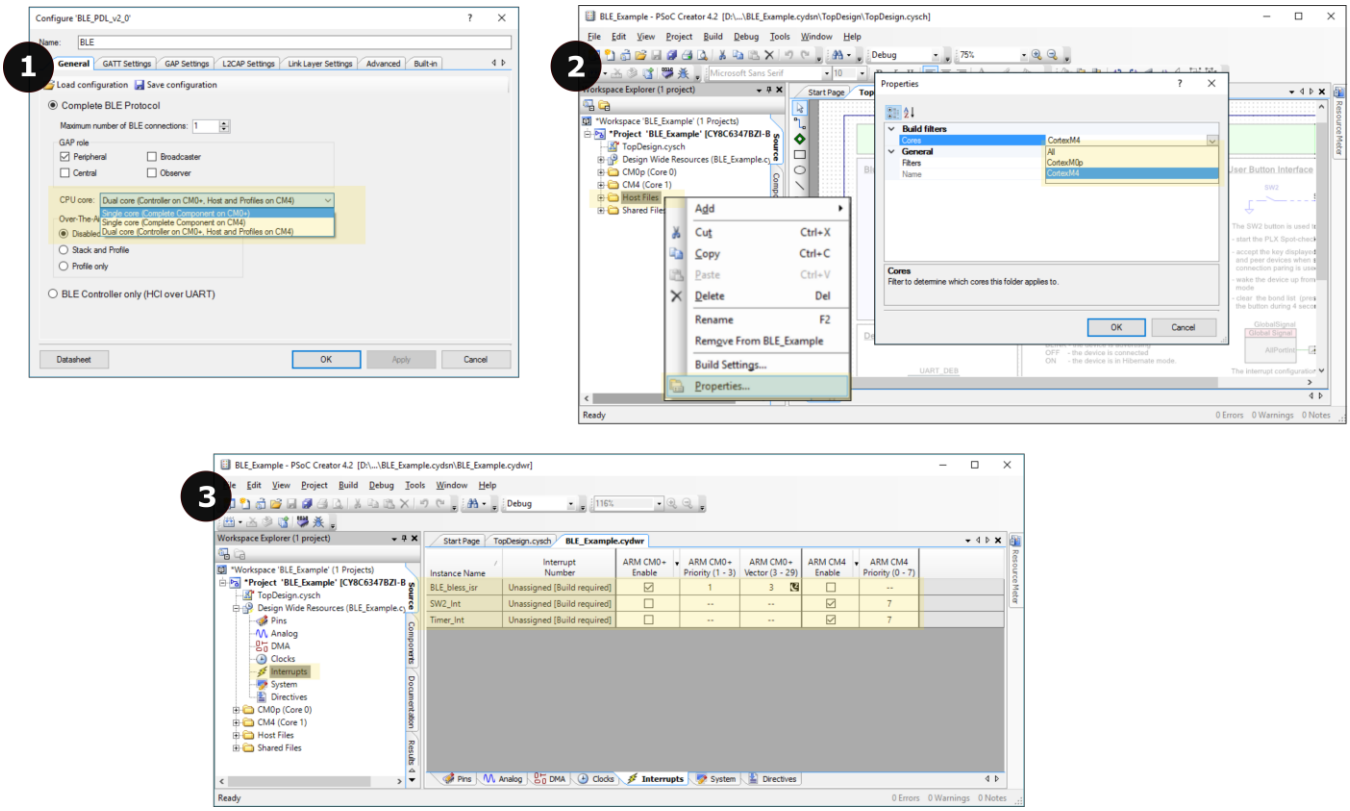
The BLE examples' structure allows easy switching between different CPU cores options. Keep in mind the following:

- 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, MCWDT, etc.) used in the example must be assigned to the host core.

Do the following to switch the CPU Cores usage:

1. In the BLE Component Customizer **General** tab, select appropriate CPU core option.
2. Change the cores Properties to CortexM4 or CortexC0p for the project folder Host Files in dependence of which CPU core was chosen in Step 1 as follows:
   - 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**
3. Assign BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to appropriate core in **DWR** > **interrupts** tab as follows:
   - 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 3. Steps for Switching the CPU Cores Usage



Hardware Setup

The code example was created for the CY8CKIT-062 PSoC 6 BLE Pioneer Kit. Pin assignment and connections required on the development board for the supported kits are in Table 2.

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

Components / User Modules

Table 3 lists the PSoC Creator Components used in this example as well as the hardware resources used by each Component.

Table 3. PSoC Creator Components List

Component	Hardware Resources
UART_DEB	1 SCB
BLE	1 BLE, 1 Interrupt
SW2	1 pin
Wakeup_Interrupt	1 interrupt
Disconnect_LED, Advertising_LED, Connect_LED	3 pins

Parameter Settings

The BLE Component in the GAP Central role is configured as a BAS GATT Client with the settings shown below. The Peripheral has three instances of the BAS GATT Server that correspond to the battery states that the three Central devices are connected to. The BLE Component that represents the Multi-Master Single Slave is configured as follows:

- Public Device Address: 00A050-000200
- Device name: BLE MMSS Example
- Security Level: Unauthenticated pairing with encryption

Figure 4. General Settings

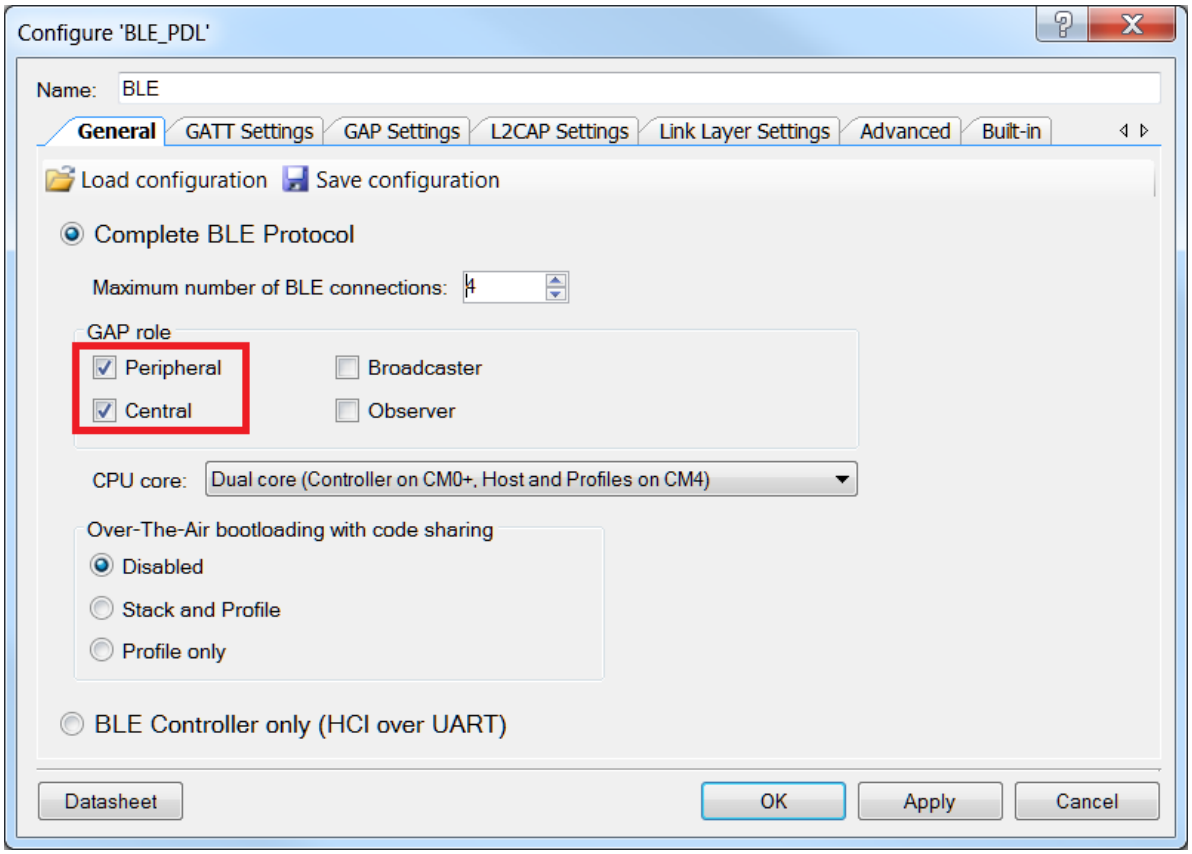


Figure 5. GATT Settings

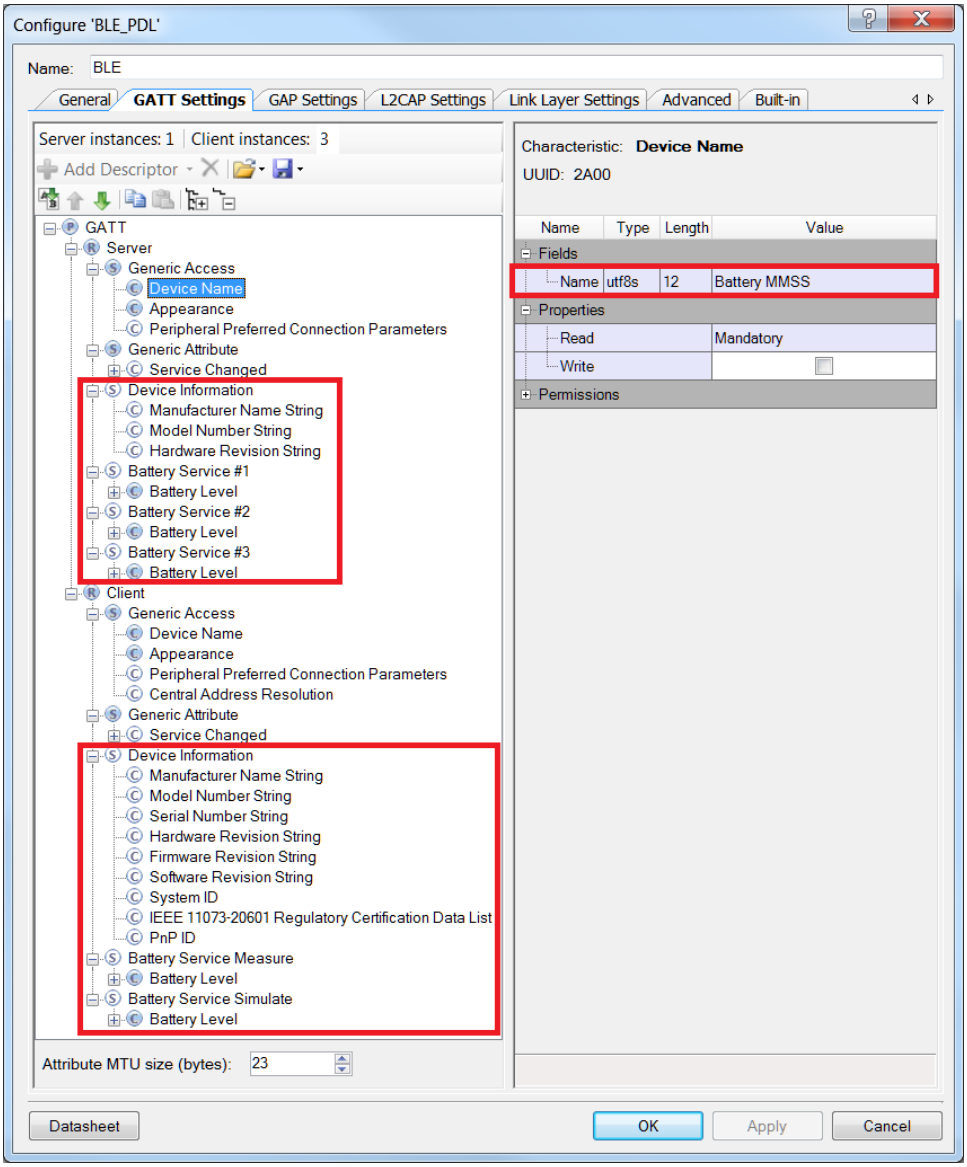


Figure 6. GAP Settings

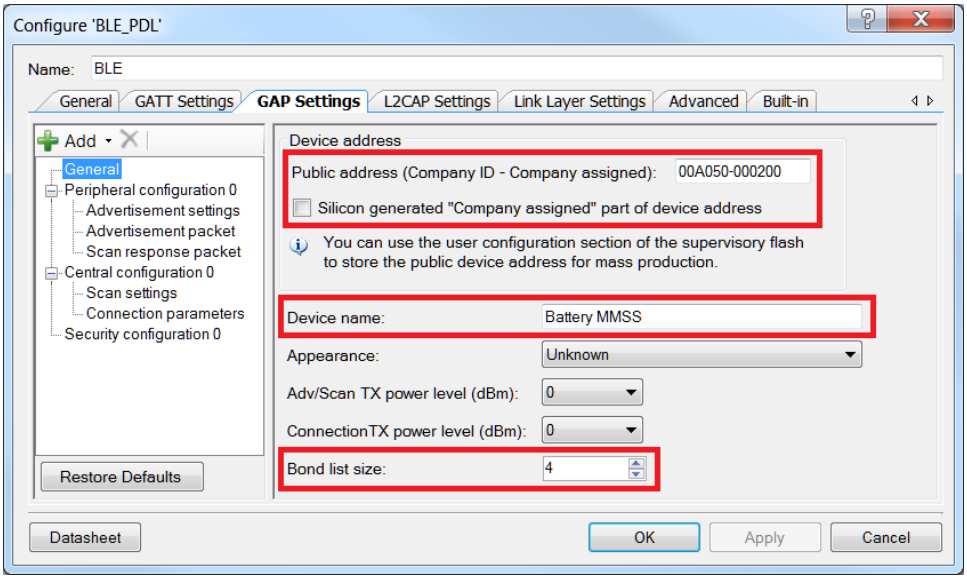


Figure 7. GAP Settings: Advertisement Settings

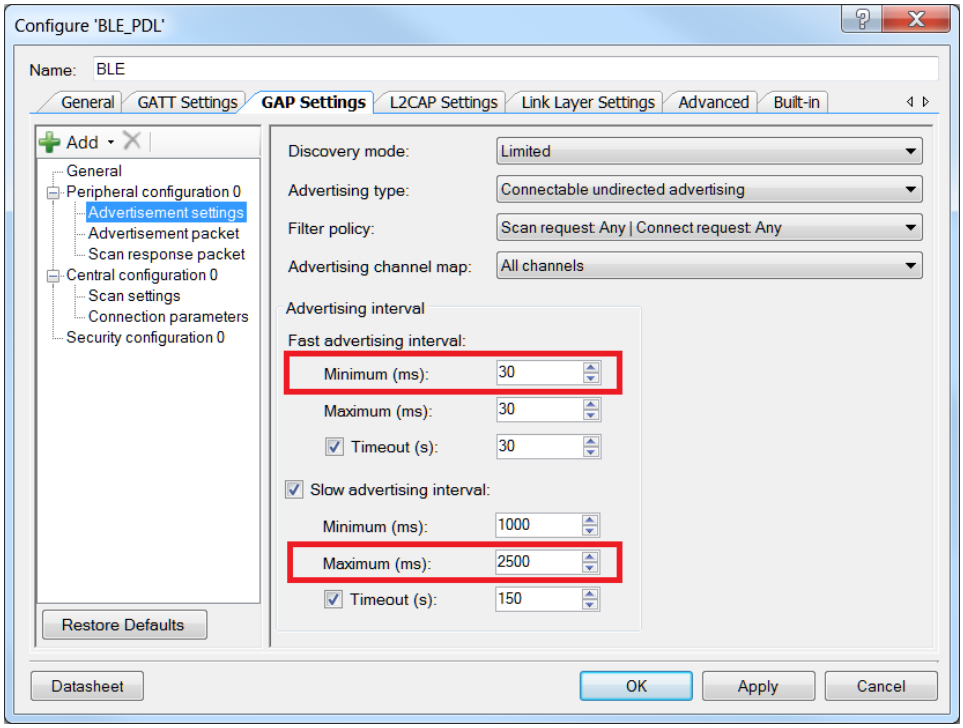


Figure 8. GAP Settings > Advertisement Packet

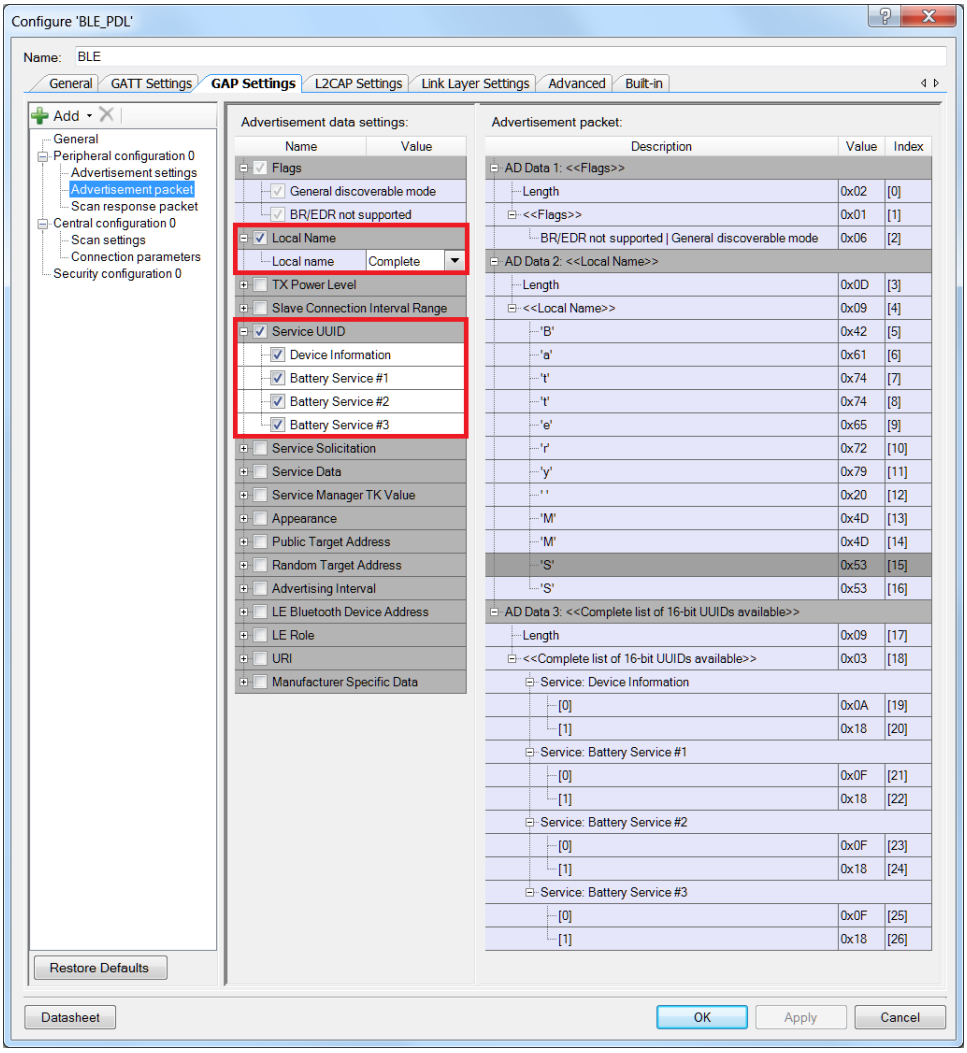


Figure 9. GAP Settings > Scan settings

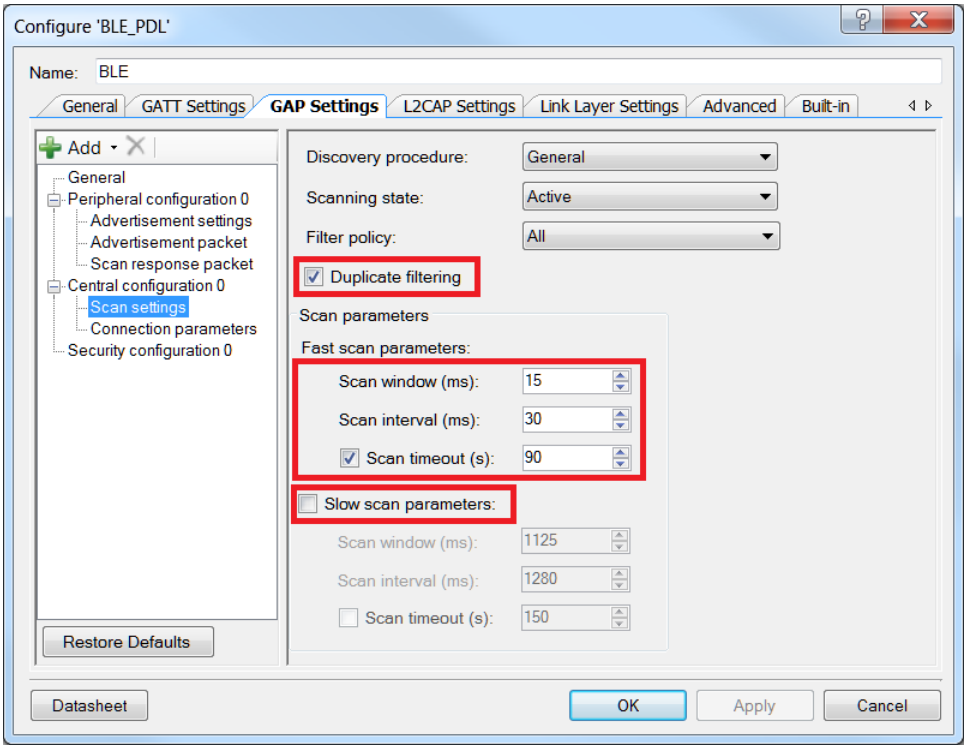
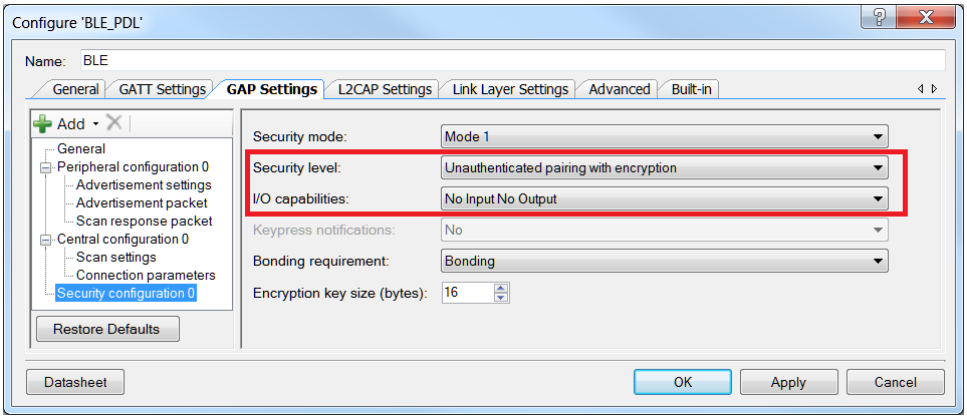


Figure 10. Security Settings



Operation

1. Build and program the BLE Multi Master Single Slave project into CY8CKIT-062 PSoC® 6 BLE Pioneer Kit.
2. Build and program the BLE Battery Level (BAS GATT Server) project into the three CY8CKIT-062 PSoC 6 BLE Pioneer Kits or CY8CKIT-042 PSoC 4 Pioneer Kits. Enable the silicon-generated “Company assigned” part of the device address option in the GAP settings of the BLE Component to have different device addresses in three devices.
3. Run a serial port communication program for the BLE Multi-Master Single Slave project.
4. In the program’s window, press ‘**s**’ to start scanning for advertising devices. When the scan report from the device with the Battery Service UUID in the advertisement data is received, the device address is automatically appended to the peerAddr list. Press ‘**c**’ to perform connection, and then select the number that corresponds to the device with the needed address. To connect to another device, press ‘**c**’ again and select a device for connection. Repeat this step three times to connect to three devices.

The output on the Client's serial port communication program may appear as follows:

	BLE Multi Master Single Slave Example BLE Stack Version: 5.0.0.718 CY_BLE_EVT_STACK_ON, StartAdvertisement Please select operations: '1' -- stop scanning '2' -- stop advertising 's' -- start scanning for BLE devices 'c' -- send connect request to peer device 'd' -- send disconnect request to peer device 'p' -- print list of connected devices CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a050000200 CY_BLE_EVT_SET_TX_PWR_COMPLETE CY_BLE_EVT_SET_TX_PWR_COMPLETE
Press 's'	**s** **Start scanning for BLE devices….** CY_BLE_GapcStartScan API Success CY_BLE_EVT_GAPC_SCAN_START_STOP GAPC_START_SCANNING ADV type: 0x0 address: 61d98fec1464, rssi - -69 dBm, data - 02 01 06 02 0a fe 11 15 d0 00 2d 12 1e 4b 0f a4 99 4e ce b5 31 f4 05 79 03 02 b2 fe --- **uuid: BAS SERVICE - YES, added to the connect list** ADV type: 0x0 address: **00a050aex50e**, rssi - -42 dBm, data - 02 01 06 0e 09 42 61 74 74 65 72 79 20 4c 65 76 65 6c 07 03 0a 18 0f 18 0f 18 --- --- **uuid: BAS SERVICE - YES, added to the connect list** ADV type: 0x0 address: **00a050aea52e**, rssi - -47 dBm, data - 02 01 06 0e 09 42 61 74 74 65 72 79 20 4c 65 76 65 6c 07 03 0a 18 0f 18 0f 18 --- ADV type: 0x0 address: 65800bfa6a70, rssi - -73 dBm, data - 02 01 06 02 0a fe 11 15 d0 00 2d 12 1e 4b 0f a4 99 4e ce b5 31 f4 05 79 03 02 b2 fe ADV type: 0x0 address: 00a050252401, rssi - -78 dBm, data - 02 01 02 1b 09 50 65 72 66 6f 72 6d 61 63 65 20 4d 65 61 73 75 72 65 6d 65 74 20 41 50 50 20 …

Press 'c'	**c** **Connect to a device…** Device 1: 00a050aea50e Device 2: 00a050aex52e Device 3: 00a050aea55e Device 4: 00a050aea56e Select a device for connection: (1..4):
Press '1'	**1** CY_BLE_EVT_GAPC_SCAN_START_STOP Scan complete! Connecting to the device (address - 00a050aea50e) CY_BLE_EVT_GATT_CONNECT_IND: 3, 13 CY_BLE_EVT_GAP_DEVICE_CONNECTED: 0 CY_BLE_GapAuthReq API Success CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: bdHandle=13, security=1, bonding=0, ekeySize=10, err=0 CY_BLE_EVT_GAP_AUTH_COMPLETE: bdHandle=13, security=1, bonding=0, ekeySize=10, err=0 Start Discovery ENCRYPT_CHANGE: 1 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 0x180f has range from 0x13 to 0x17 service with UUID 0x180f has range from 0x18 to 0x1c service with UUID 0x180a has range from 0xc to 0x12 Enable Notification CY_BLE_BascSetCharacteristicDescriptor() successful.
Press 'c'	**c** **Connect to a device…** Device 1: 00a050aea50e [CONNECTED] Device 2: 00a050aea52e Device 3: 00a050aea55e Device 4: 00a050aea56e

	Select a device for connection: (1..4):

Press '2'	**2**
	
	Connecting to the device (address - 00a050aea52e)
	CY_BLE_EVT_GATT_CONNECT_IND: 2, 12
	CY_BLE_EVT_GAP_DEVICE_CONNECTED: 0
	CY_BLE_GapAuthReq API Success
	CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: bdHandle=12, security=1, bonding=0, ekeySize=10, err=0
	CY_BLE_EVT_GAP_AUTH_COMPLETE: bdHandle=12, security=1, bonding=0, ekeySize=10, err=0
	
	Start Discovery
	ENCRYPT_CHANGE: 1
	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 0x180f has range from 0x13 to 0x17
	  service with UUID 0x180f has range from 0x18 to 0x1c
	  service with UUID 0x180a has range from 0xc to 0x12
	
	Enable Notification
	CY_BLE_BascSetCharacteristicDescriptor() successful.

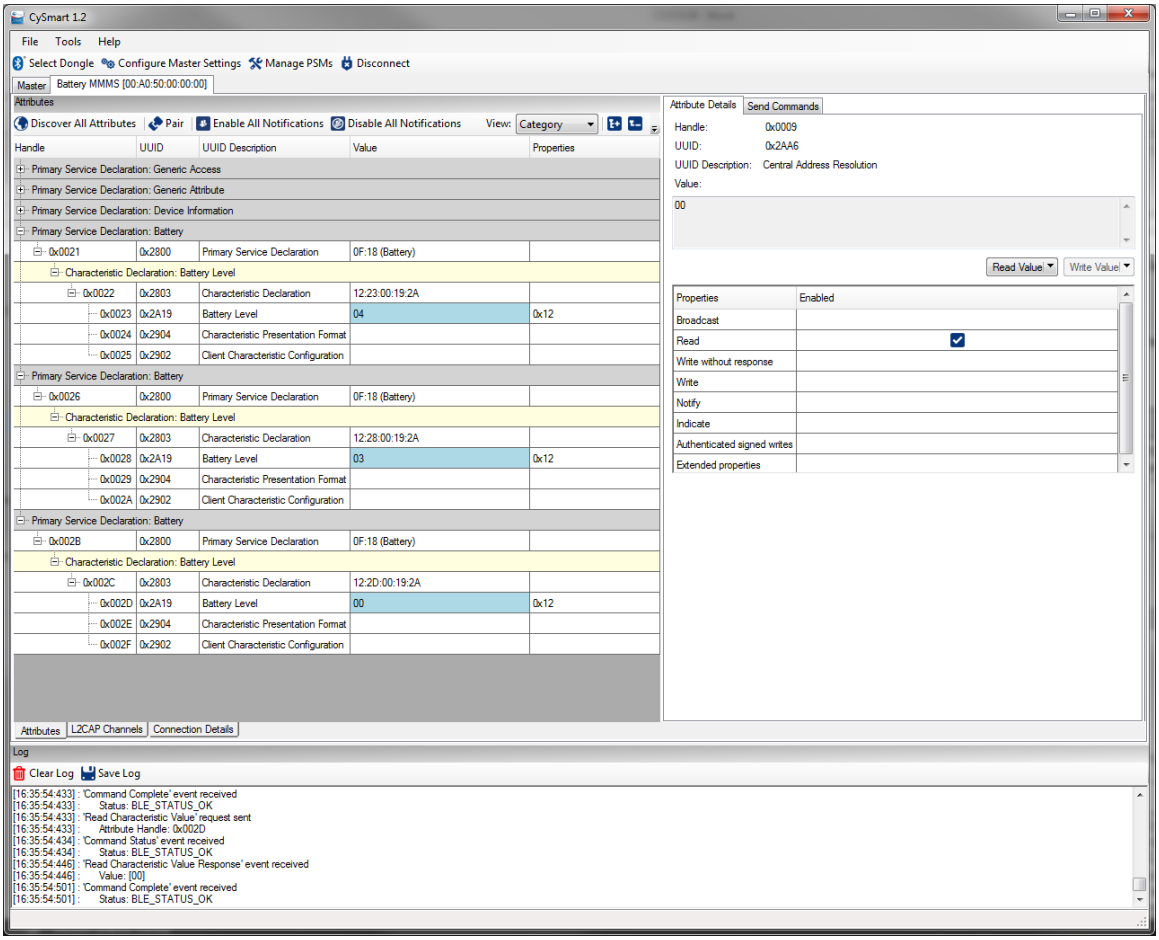
Notifications	CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 5 [attId:3, bdHandle:13] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 5 [attId:2, bdHandle:12] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 6 [attId:3, bdHandle:13] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 6 [attId:2, bdHandle:12] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 7 [attId:3, bdHandle:13] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 8 [attId:2, bdHandle:12]
Press 'p'	**p** --- **Connected devices list:** 1. address: 00a050aea50e bdHandle:13 attId:3 CLIENT 2. address: 00a050aea52e bdHandle:12 attId:2 CLIENT --- CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 8 [attId:3, bdHandle:13] CY_BLE_EVT_BASC_NOTIFICATION: Battery Level: 9 [attId:2, bdHandle:12]

5. Open the CySmart Central Emulation tool and select the connected dongle in the **Select BLE Dongle Target** window.

- Click **Start Scan** to discover available devices.
- Select **Battery MMSS** in the list of available devices and connect to it.
- Respond **Yes** to a pairing request received from the peer device. The output appears as follows (press '**p**' to check that the Server is connected).
- Click **Discover All Attributes**, and then **Enable All Notifications**. Observe the received characteristic values (see Figure 11).

	CY_BLE_EVT_GATT_CONNECT_IND: 1, 11 CY_BLE_EVT_GAP_DEVICE_CONNECTED: 1 CY_BLE_GapAuthReq API Success CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 3 CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: bdHandle=11, security=1, bonding=0, ekeySize=10, err=0 CY_BLE_EVT_GAP_AUTH_COMPLETE: bdHandle=11, security=1, bonding=0, ekeySize=10, err=0 ENCRYPT_CHANGE: 1
Press 'p'	**p** ---------- **Connected devices list:** 1. address: 00a050aea50e bdHandle:13 attId:3 CLIENT 2. address: 00a050aea52e bdHandle:12 attId:2 CLIENT 3. address: 00a050bddd9a bdHandle:11 attId:1 SERVER ----------

Figure 11. CySmart Central Emulation Tool (Windows)



6. Press '**d**' to disconnect the device.

Press 'd'	**d** --- **Connected devices list:** 1. address: 00a050aea50e bdHandle:13 attId:3 CLIENT 2. address: 00a050aea52e bdHandle:12 attId:2 CLIENT 3. address: 00a050bddd9a bdHandle:11 attId:1 SERVER --- Select a device for disconnect: (1..3):
Select Device for Disconnection	
Press '3' Disconnect SERVER	**3** Select a device for disconnect:2 CY_BLE_GapDisconnect param: bdHandle:11, reason:13 CY_BLE_GapDisconnect API Success CY_BLE_EVT_GATT_DISCONNECT_IND: attId=1, bdHandle=11 Disconnect peripheral StartAdvertisement start CY_BLE_EVT_GAP_DEVICE_DISCONNECTED, reason: 0

Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with BLE Connectivity, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-Core CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – BLE Test and Debug Tool		CySmart is a BLE host emulation tool for Windows PCs. The tool provides an easy-to-use GUI to enable the user to test and debug their BLE 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 Programmable System-on-Chip (PSoC®)		PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual (TRM)
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		

Document History

Document Title: CE215118 – BLE Multi-Master Single Slave with PSoC 6 MCU with BLE Connectivity

Document Number: 002-15118

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5968176	NPAL	11/20/2017	Initial release

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

Cypress Developer Community

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

Technical Support

cypress.com/support

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