CE212741 – IPSP Router and Node with PSoC 6 MCU with BLE Connectivity

Objective

This example demonstrates the operation of the Internet Protocol Support Profile (IPSP) with the Bluetooth Low Energy (BLE_PDL) Component.

Overview

This example demonstrates how to set up the IPv6 communication infrastructure between two devices (CY8CKIT-062 PSoC 6 BLE Pioneer Kits) over a BLE transport using the L2CAP channel. Creation and transmission of IPv6 packets over the BLE is not part of this example.

The example consists of two projects: IPSP Router (GAP Central) and IPSP Node (GAP Peripheral). The router sends generated packets with different content to the node in a loop and validates them with the data packet received afterward. The node wraps the received data coming from the router back to the router.

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 Bluetooth Low Energy (BLE) Connectivity (PSoC 6 BLE) parts

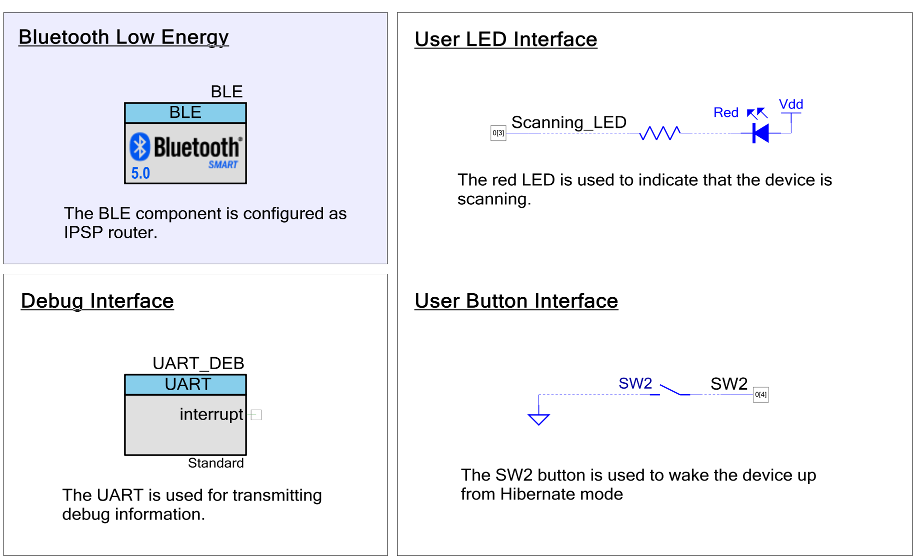
Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

Design

BLE IPSP Router

Figure 1 shows the top design schematic of the project.

Figure 1. BLE IPSP Router Code-Example Schematic



The project demonstrates the BLE_PDL Component functionality configured as a router.

The project uses the `AppCallBack()`callback function to receive generic and L2CAP events from the BLE Stack.

After a project startup, the BLE_PDL, UART, and ISR Components are initialized. When the BLE_PDL Component begins its operation, the user LED starts blinking. This indicates that the device has started scanning. After a 180-second timeout, if no Peripheral device has been connected, the router stops scanning. The user LED is turned OFF indicating the disconnection state; the system then enters Hibernate mode.

Advertising packets that are received during the scanning procedure from peripheral devices are parsed and filtered. Only packets with the IPSS service UUID are handled and shown in the debug terminal with the device sequence number as a candidate to connect.

After connection to the node device, the IPSP protocol multiplexer for L2CAP is registered and the initial Receive Credit Low Mark for Based Flow Control mode is set after the `CY_BLE_EVT_STACK_ON` event.

When the GAP connection is established, after the `CY_BLE_EVT_GATT_CONNECT_IND` event, the router automatically initiates an L2CAP LE credit-based connection with a PSM set to `LE_PSM_IPSP`.

The project allows sending data packets to the node through the IPSP channel. Sent data is compared with the received data in the response packet after the `CY_BLE_EVT_L2CAP_CBFC_DATA_READ` event. When no failure is observed, a new packet is generated and sent to the node automatically. Otherwise, the transfer is stopped; the “Wraparound failed” message indicates a failure.

The Router updates the LE credits dynamically, when the credit count goes below the low mark (`CY_BLE_EVT_L2CAP_CBFC_RX_CREDIT_IND` event), to allow continuous transfer of data between the node and router.

The project uses the UART Component for displaying debug information and for sending commands through the terminal emulator application. Table 1 lists the commands which you can use to perform procedures:

Table 1. List of Commands

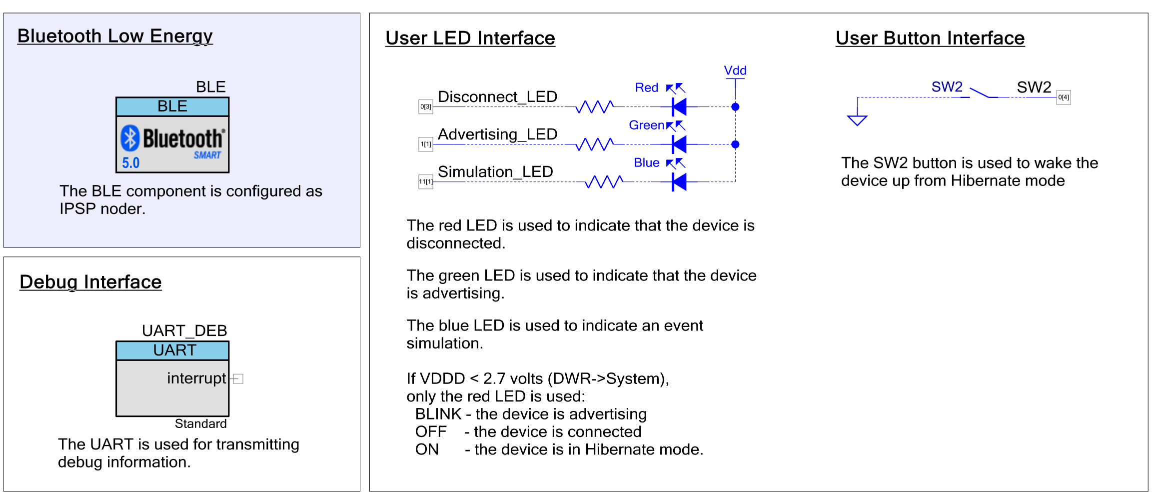
Command	Description
'z'+'#'	Select a specific peer device, where ‘#’ is a sequence number from the advertising packet (0-7). The default value is 0.
'c'	Send a connection request to the selected peer device.
'd'	Send a disconnect request to the peer device.
'v'	Cancel the connection request.
's'	Start the discovery procedure.
'1'	Send the data packet to the Node through the IPSP channel

The terminal emulator application lists these commands when you enter ‘h’ in the application.

BLE IPSP Node

Figure 2 shows the top design schematic of the project.

Figure 2. BLE IPSP Node Code Example Schematic



The project demonstrates the BLE_PDL Component functionality configured as a node.

The project uses the `AppCallBack()`callback function to receive generic and L2CAP events from the BLE Stack. The `CY_BLE_GappStartAdvertisement()` function is called after the `CY_BLE_EVT_STACK_ON` event to start advertising with the packet shown in Figure 13.

After project startup, BLE_PDL, UART, and ISR Components are initialized. When the BLE_PDL Component begins its operation, the RGB LED starts blinking with green color. This indicates that the device has started advertising. After a 180-second timeout, if no Central device has been connected, the node stops advertising; the red LED is turned ON indicating the disconnection state; and the system enters Hibernate mode.

When a Client has connected successfully, the red and green LEDs are turned OFF. The blinking blue LED indicates a data packet transaction.

The IPSP protocol multiplexer for L2CAP is registered and the initial Receive Credit Low Mark for Based Flow Control mode is set after the `CY_BLE_EVT_STACK_ON` event.

The node automatically accepts the L2CAP LE-credit-based connection request with a PSM set to `LE_PSM_IPSP` after the `CY_BLE_EVT_L2CAP_CBFC_CONN_IND` event.

After the `CY_BLE_EVT_L2CAP_CBFC_DATA_READ` event, received data is automatically wrapped back to the router.

The node updates the LE credits dynamically, when the credit count goes below the low mark (`CY_BLE_EVT_L2CAP_CBFC_RX_CREDIT_IND` event), to allow continuous transfer of data between the node and the router.

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, 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 `DEBUG_UART_ENABLED` to `DISABLED` in the *common.h* file.

LED Behavior for VDDD Voltage Smaller than 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 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 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.

Important to remember:

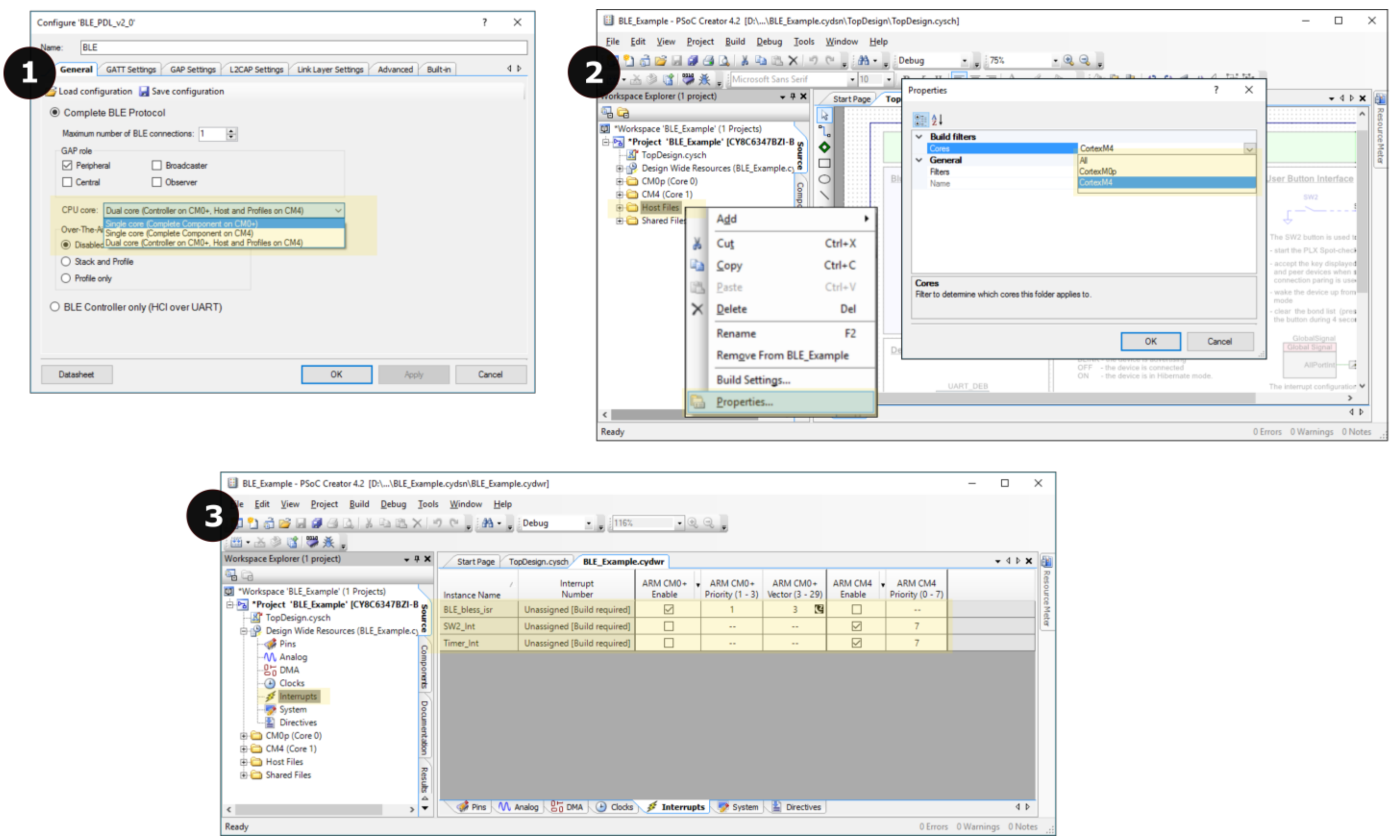
- All application host-files must be run on the host core.
- The BLE Subsystem (BLESS) interrupt must be assigned to the core where the controller runs.

- All additional interrupts (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 the 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. It should be:
   - 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 the BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to 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 3. Steps for Switching the CPU Cores Usage



Hardware Setup

The code example was created for the CY8CKIT-062 PSoC® 6 BLE Pioneer Kit.

BLE IPSP Router

Table 2 lists the pin assignment and connections required on the development board for the supported kits.

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]	
Scanning_LED	P0[3]	The red color of the RGB LED
SW2	P0[4]	

BLE IPSP Node

Table 3 lists the pin assignment and connections required on the development board for the supported kits.

Table 3. 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

BLE IPSP Router

Table 4 lists the PSoC Creator Components used in the BLE IPSP Router project as well as the hardware resources used by each Component.

Table 4. PSoC Creator Components List

Component	Hardware Resources
BLE	1 BLE, 1 Interrupt
UART_DEB	1 SCB
SW2	1 pin
Wakeup_Interrupt	1 interrupt
Scanning_LED	1 pins

Parameter Settings

Bluetooth Low Energy (BLE_PDL)

Figure 4. General Settings

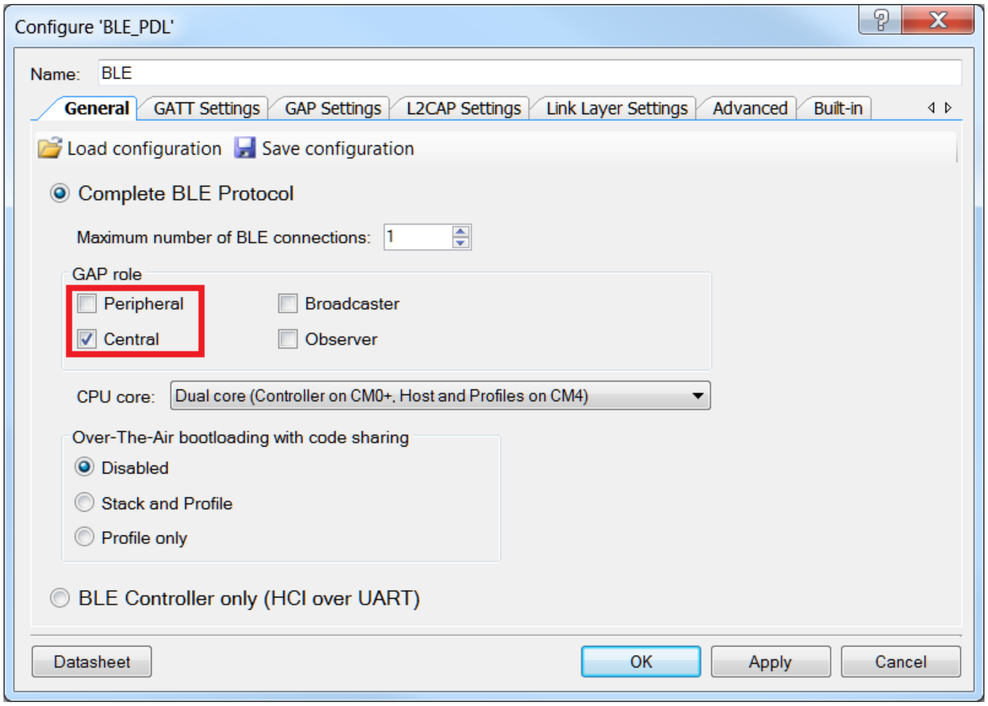


Figure 5. GATT Settings

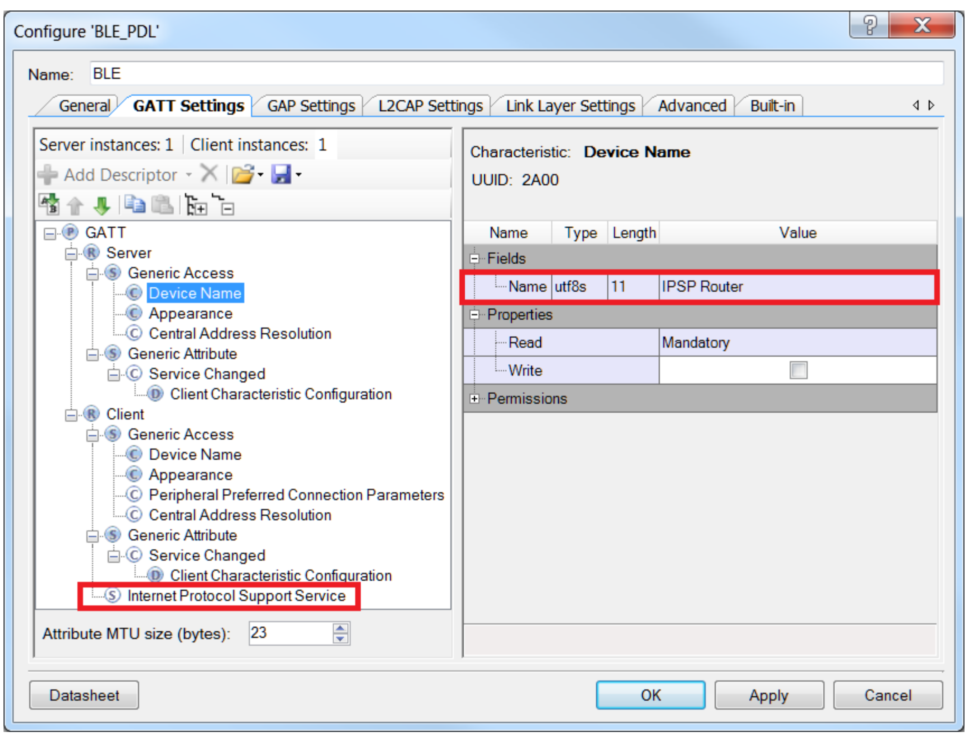


Figure 6. GAP Settings

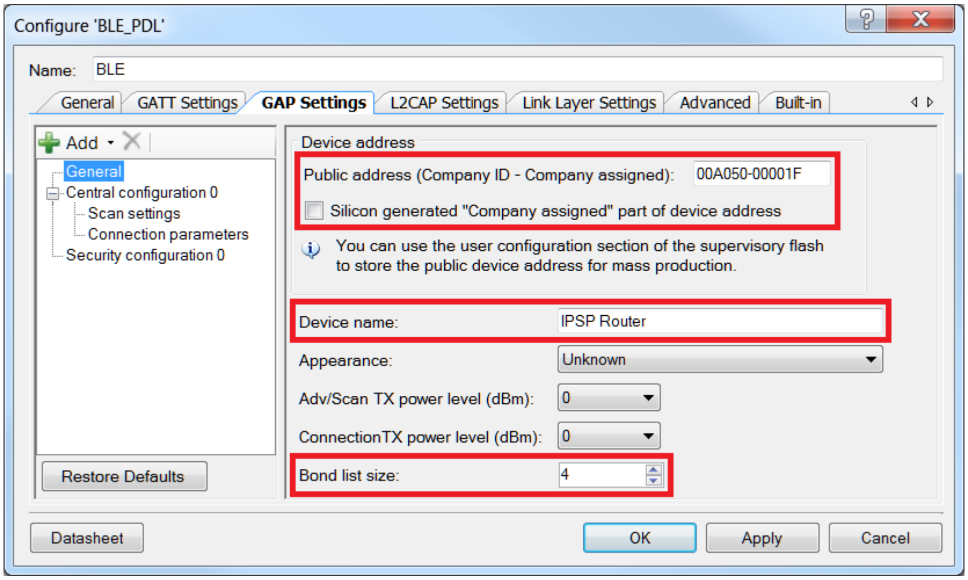


Figure 7. GAP Settings > Scan Settings

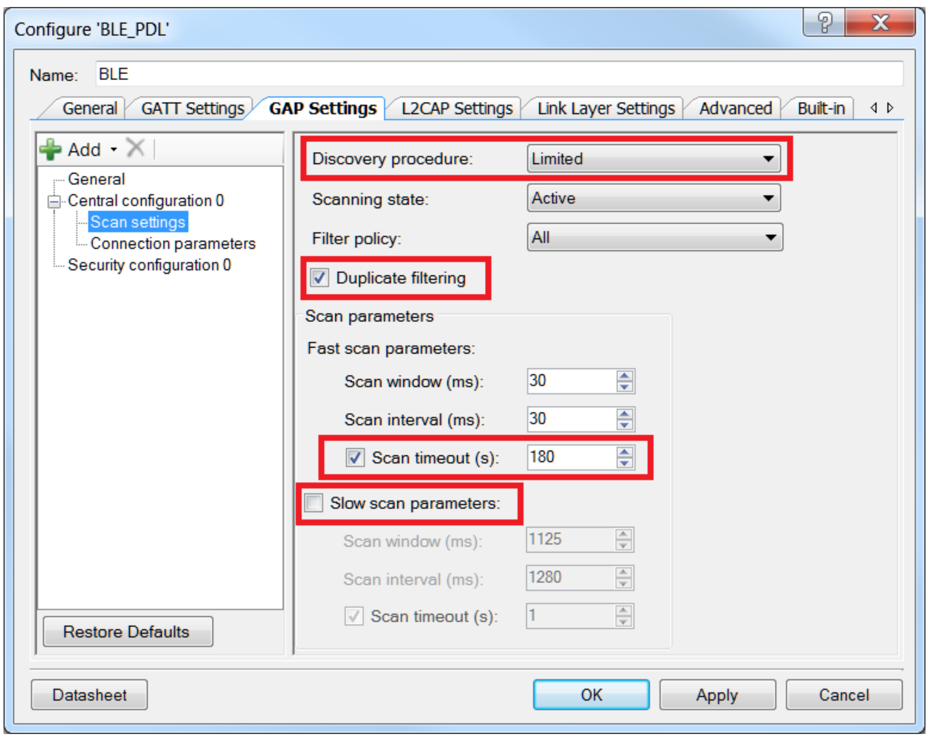
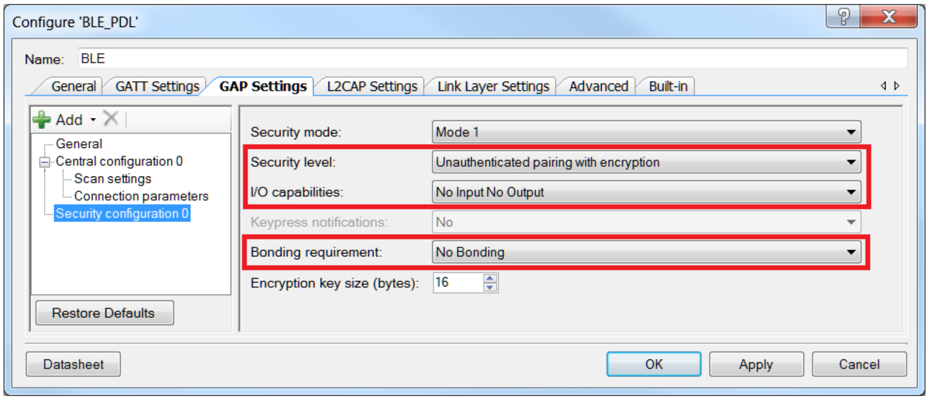


Figure 8. Security Settings



BLE IPSP Node

Table 5 lists the PSoC Creator Components used in the BLE IPSP Node project as well as the hardware resources used by each Component.

Table 5. PSoC Creator Components List

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

Parameter Settings

Bluetooth Low Energy (BLE_PDL)

Figure 9. General Settings

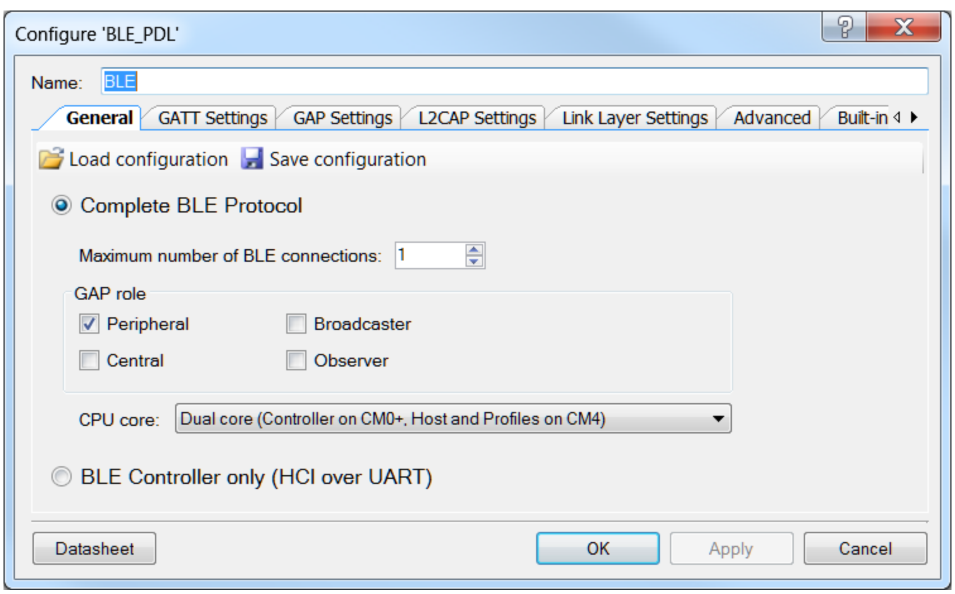


Figure 10. GATT Settings

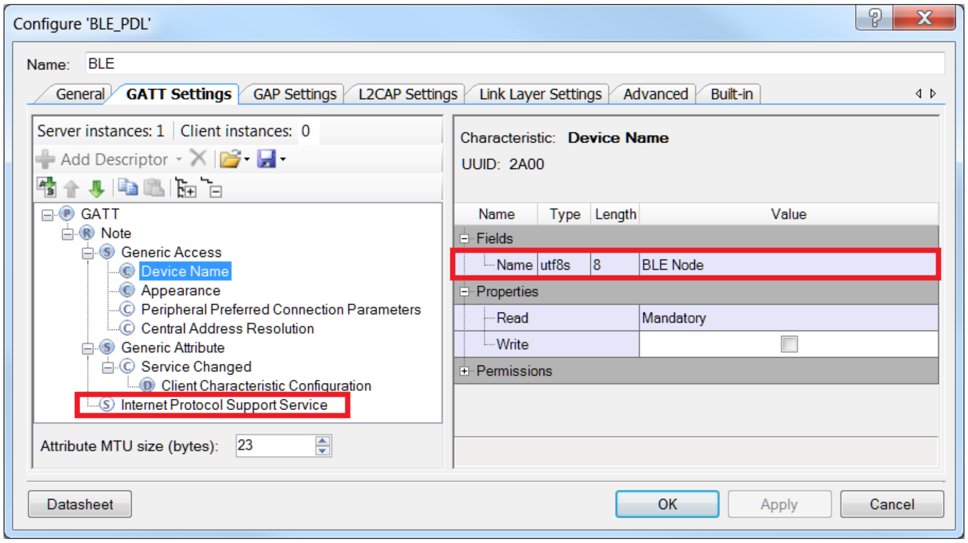


Figure 11. GAP Settings

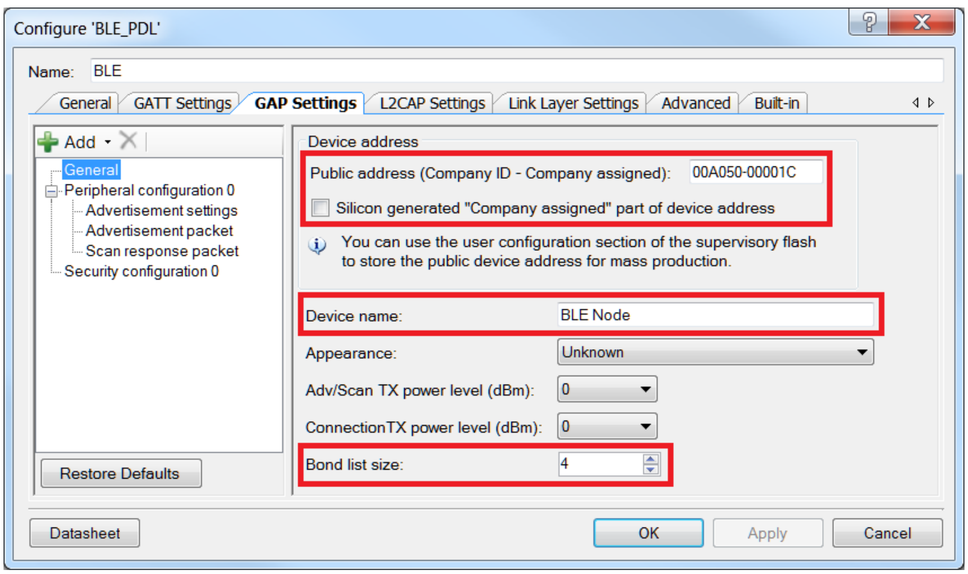


Figure 12. GAP Settings > Advertisement settings

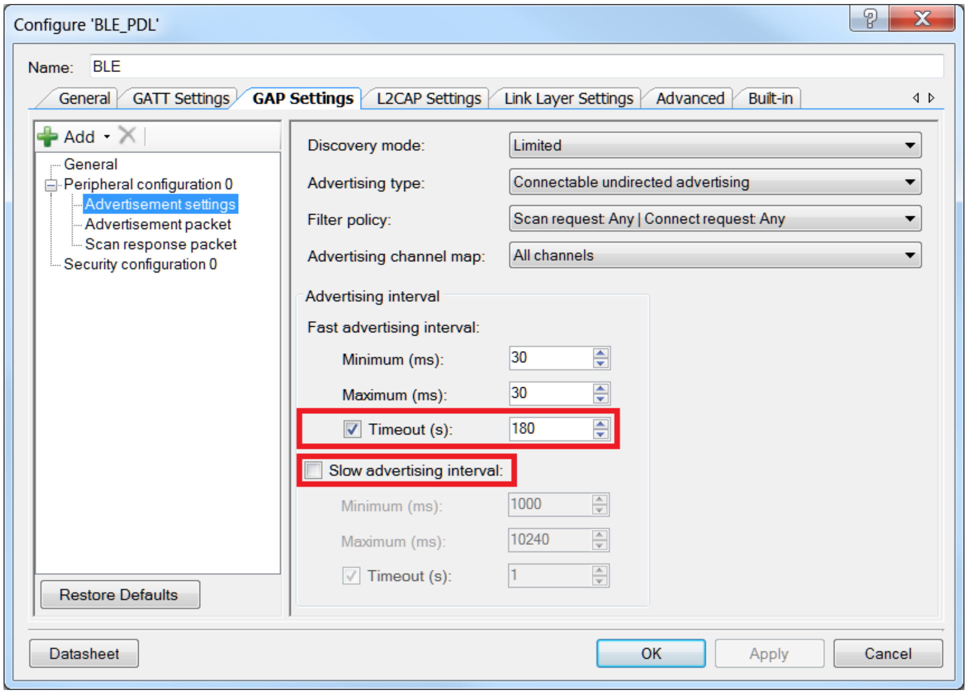


Figure 13. GAP Settings > Advertisement Packet

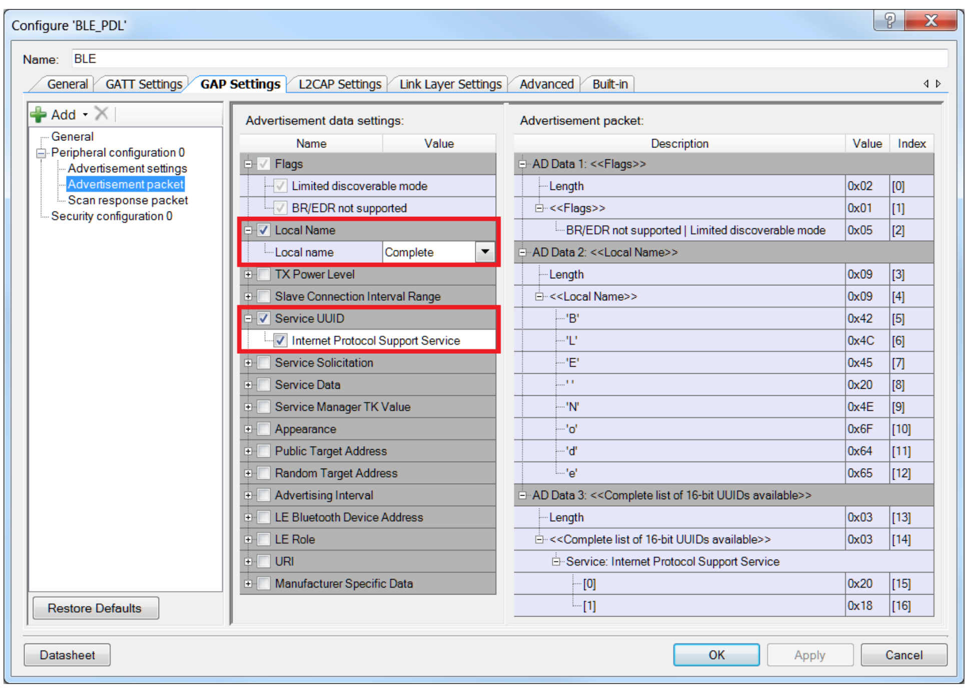
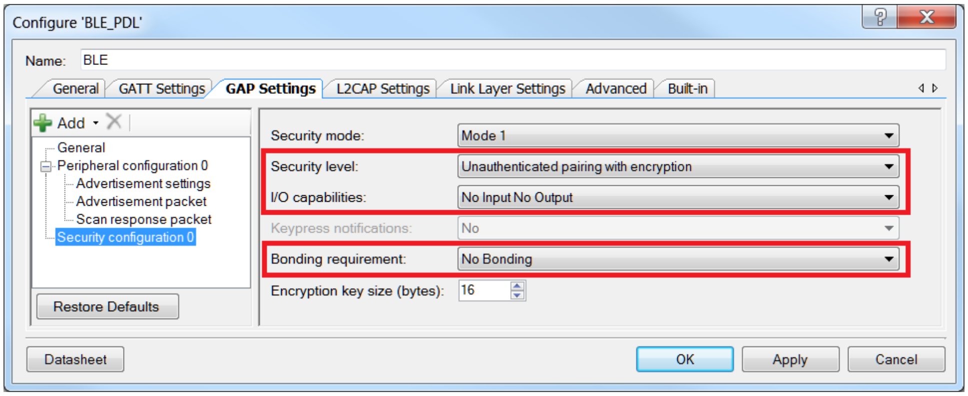


Figure 14. Security Settings



Operation

The BLE IPSP Router project is intended to work in association with the BLE IPSP Node project.

1. Build and program the BLE IPSP Router and Node projects into two CY8CKIT-062 PSoC 6 BLE Pioneer Kits.
2. After a start, two projects send log messages through the UART. The router project logs Advertising and Scan response reports from the node; for example:

   Advertisement report: eventType = 0, peerAddrType - 0, peerBdAddr - #0: 00a05000001C, rssi - -58 dBm

   where #0 is a sequence number of the node device.
3. Use the sequence number from the Advertisement report after the '**z**' command to select the required node if multiple node devices are available.
4. Press '**c**' to connect to the node.
5. Press '**1**' to start the wraparound test.

The blue LED on the node indicates a data transfer process. If wraparound data validation fails, the blue LED stops blinking and the "Wraparound failed" message appears in the Router UART log.

Example logs:

BLE IPSP Router Example Project

```
BLE Stack Version: 5.0.0.718
CY_BLE_EVT_STACK_ON, StartAdvertisement
Bluetooth On, StartScan with addr: CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a05000001f
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAPC_SCAN_START_STOP, state: 2
CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE

Advertisement report: eventType = 0, peerAddrType - 0, peerBdAddr - 0: 00a05000001c, rssi - -47 dBm
c
CY_BLE_EVT_GAPC_SCAN_START_STOP, state: 0
GAPC_END_SCANNING
CY_BLE_EVT_GATT_CONNECT_IND: 0, 0
```

```
L2CAP channel connection request sent.

CY_BLE_EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms

CY_BLE_EVT_L2CAP_CBFC_CONN_CNF: bdHandle=0, lCid=64, responce=0, connParam: mtu=1280, mps=1280, credit=1000
```

1

```
-> Cy_BLE_L2CAP_ChannelDataWrite #0

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite #1

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite #2

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite #3

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite #4

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite #5

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

…
```

BLE Node Example Project

```
BLE Stack Version: 5.0.0.718


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: 00a05000001c

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


CY_BLE_EVT_GATT_CONNECT_IND: 0, 0

CY_BLE_EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms

CY_BLE_EVT_L2CAP_CBFC_CONN_IND: bdHandle=0, lCid=64, psm=35,connParam mtu=1280, mps=1280, credit=1000
SUCCESSFUL


<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278

-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0

<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
```

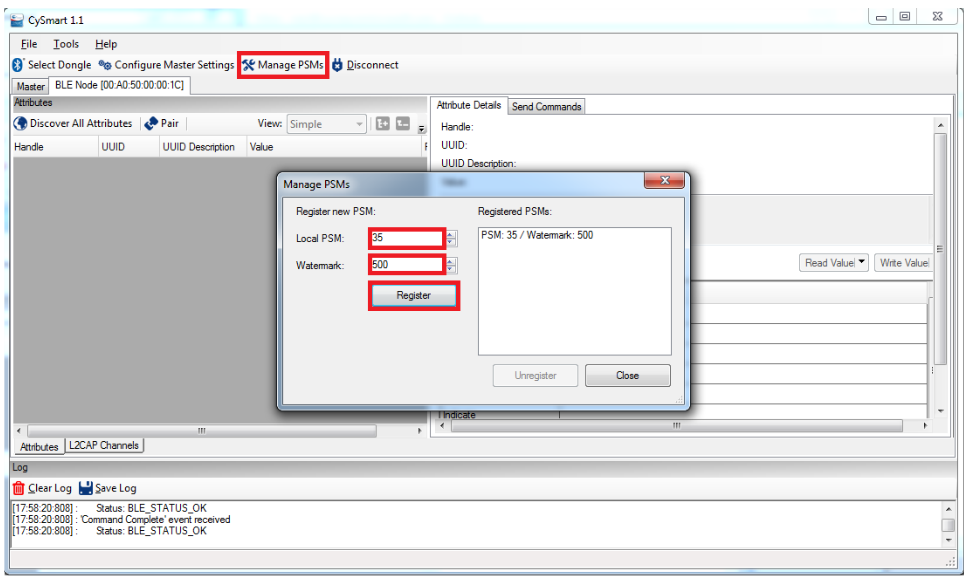
```
-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> Cy_BLE_L2CAP_ChannelDataWrite API result: 0
…..
```

You can use the CySmart application on a Windows PC BLE-compatible device as a Client to connect to the node.

Do the following to use the CySmart Windows application as a Client:

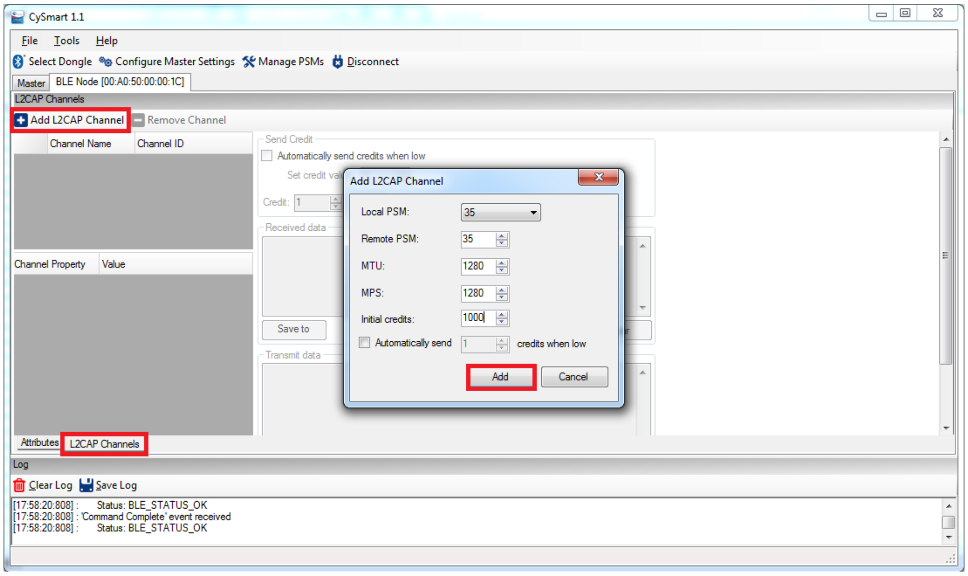
1. Connect the CySmart BLE dongle to a USB port on the PC.
2. Launch the CySmart application and select the connected dongle in the dialog window.
3. Reset the development kit to start advertising by pressing the **SW1** button.
4. Click the **Start Scan** button to discover available devices.
5. Select **BLE Node** in the list of available devices and connect to it.
6. Click the **Manage PSMs** button. In the Manage PSMs window, enter the following. Click **Register**, and then close the window.
   - Local PSM: 35(LE_PSM_IPSP),
   - Watermark: 500

Figure 15. CySmart Manage PSM Window



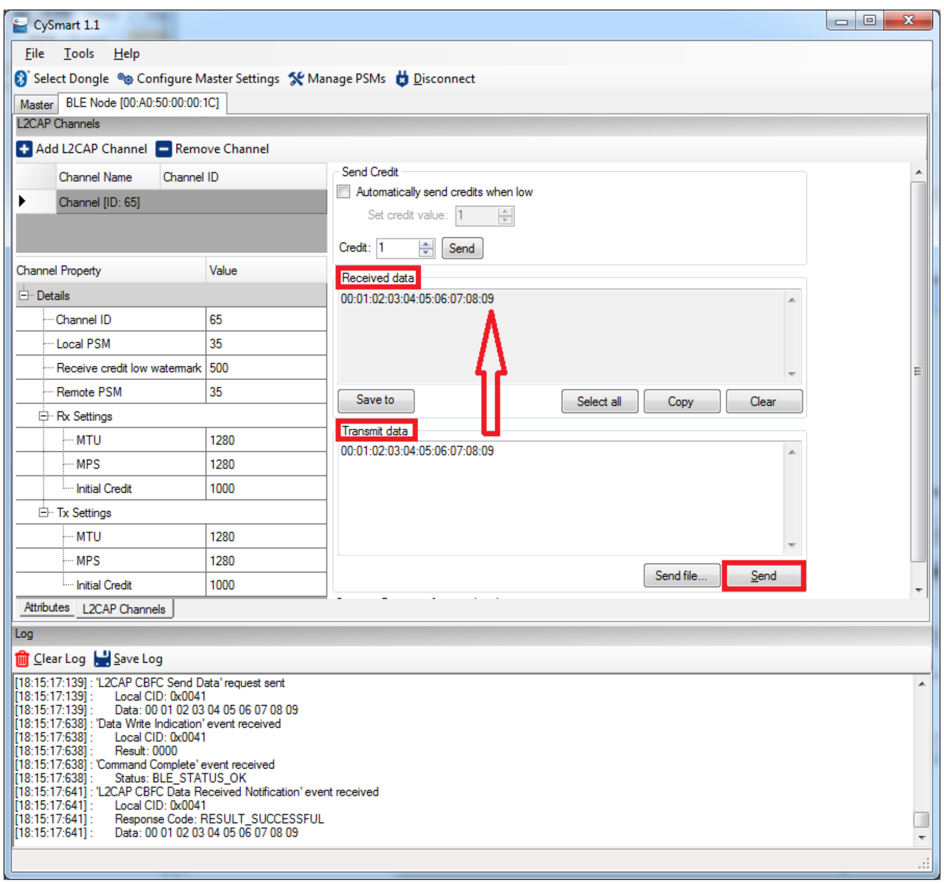
7. Select the **L2CAP Channels** tab and click **Add L2CAP Channel** to create an L2CAP channel. In the Add L2CAP Channel window, enter the following details and click **Add**:
   - Local PSM: 35
   - Remote PSM: 35
   - MTU: 1280
   - MPS: 1280
   - Initial credits: 1000

Figure 16. CySmart Add L2CAP Channel Window



8. Now, the L2CAP channel is ready to transmit and receive data. Enter some data into the Transmit data area and click **Send**. The same data will appear in the Receive data area.

Figure 17. CySmart L2CAP Channels Window



For more information about the CySmart Central Emulation tool, refer to CySmart User Guide.

Note: The CySmart mobile application does not have IPSP profile support.

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 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: CE212741 – IPSP Router and Node with PSoC 6 MCU with BLE Connectivity

Document Number: 002-12741

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5846404	NPAL	11/20/2017	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

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