

BLE IPSP Router and Node 1.0

Features

- BLE Internet Protocol Support Profile (IPSP) in Router and Node role
- Logical Link Control and Adaptation Protocol (L2CAP) data transfer
- Deep Sleep mode support
- LED status indication

Development Kit Configuration

Default CY8CKIT-042 BLE Pioneer Kit configuration.

General Description

This example project demonstrates the Internet Protocol Support Profile operation of the Bluetooth Low Energy (BLE) PSoC Creator Component. The example project requires CY8CKIT-042-BLE Pioneer Kit.

This example demonstrates how to setup an IPv6 communication infrastructure between two devices over a BLE transport using L2CAP channel. Creation and transmission of IPv6 packets over BLE is not part of this example.

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

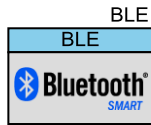
BLE IPSP Router Project Configuration

The router example project consists of the following components:

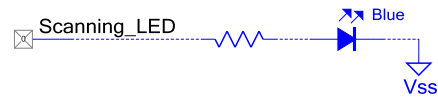
- Bluetooth Low Energy (BLE)
- UART
- Watchdog timer (WDT) that is used as a general timer

The top design schematic is shown in Figure 1.

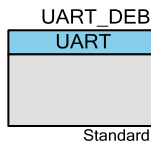
Figure 1. Top design schematic
BLE IPSP Router Example Project



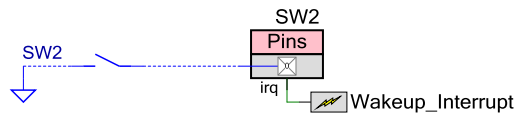
BLE component is configured as IPSP router



The blue LED is used to indicate that the device is scanning.



UART is used for transmitting the debug information.



The button is used to wake the device up from the hibernate mode.

The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and is used to wake a device from the hibernate mode.

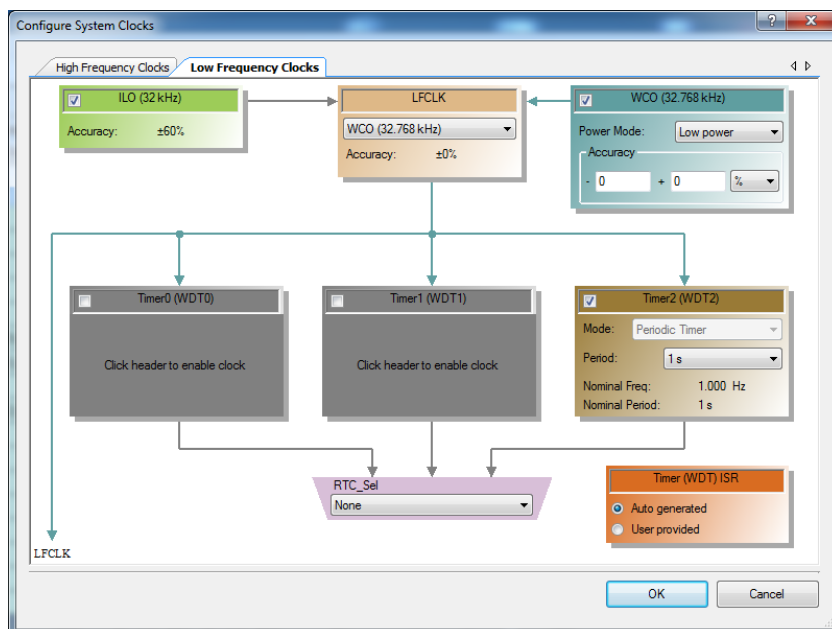
UART

This component is used for printing debug information and scan commands from a terminal.

Watch Dog Timer (WDT)

The WDT timer works over low power Deep Sleep mode, therefore it is used as a general timer for LED indication and simulation purpose. WDT Timer2 is configured in the Low Frequency Clocks tab of the Clocks configuration in the Design Wide Resources (DWR).

Figure 2. WDT Timer2 settings



Bluetooth Low Energy (BLE)

The BLE component is configured as Internet Protocol Support Profile in Router role operation. L2CAP MTU size is set to 1280.

Figure 3. GATT settings

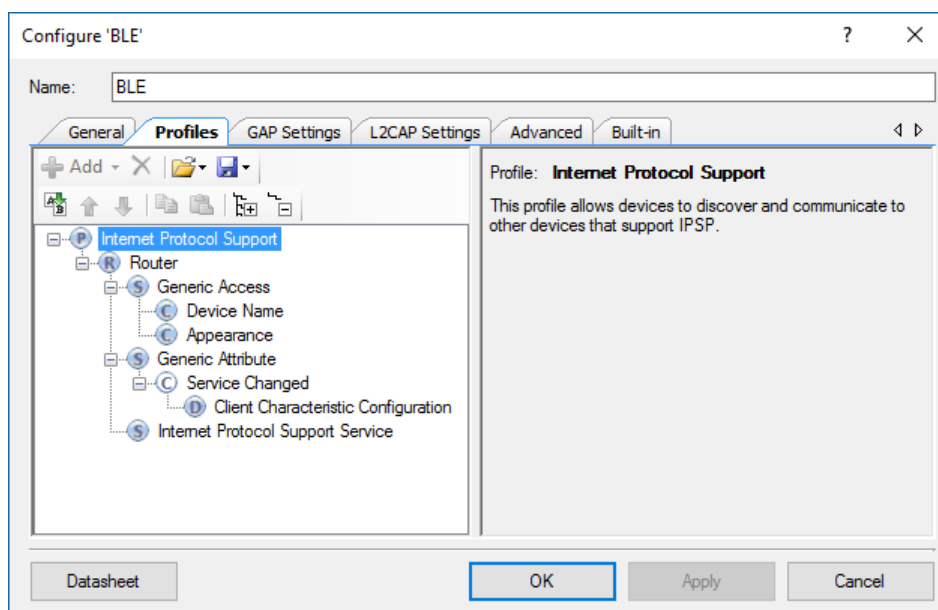


Figure 4. GAP settings

Configure 'BLE'

Name: BLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

General

- Central role
- Scan settings
- Connection parameters
- Security

Device address

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

☐ 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:

Appearance: Unknown

Attribute MTU size (bytes): 23

Link layer max TX payload size (bytes): 27

Link layer max RX payload size (bytes): 27

Adv/Scan TX power level (dBm): 3

Connection TX power level (dBm): 3

☐ Enable Link Layer Privacy

Restore Defaults

Datasheet OK Apply Cancel

Figure 5. GAP settings -> Scan settings

Configure 'BLE'

Name: BLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

General

- Central role
- Scan settings**
- Connection parameters
- Security

Discovery procedure: Limited

Scanning state: Active

Filter policy: All

☒ Duplicate filtering

Scan parameters

Fast scan parameters:

Scan window (ms): 30

Scan interval (ms): 30

☒ Scan timeout (s): 180

☐ Slow scan parameters:

Scan window (ms): 1125

Scan interval (ms): 1280

☒ Scan timeout (s): 150

Restore Defaults

Datasheet OK Apply Cancel

Figure 6. Security settings

The screenshot shows the 'Configure BLE' dialog box with the 'Security' tab selected. The 'Name' field is set to 'BLE'. The left sidebar shows a tree view with 'Security' selected. The main area contains the following settings:

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

Buttons at the bottom: Datasheet, OK, Apply, Cancel.

Figure 7. L2CAP Settings

The screenshot shows the 'Configure BLE' dialog box with the 'L2CAP Settings' tab selected. The 'Name' field is set to 'BLE'. The left sidebar shows a tree view with 'L2CAP Settings' selected. The main area contains the following settings:

Setting	Value
<input checked="" type="checkbox"/> Enable L2CAP logical channels	
Number of L2CAP logical channels:	1
Number of PSMs:	1
L2CAP MTU size (bytes):	1280
L2CAP MPS size (bytes):	1280

Buttons at the bottom: Datasheet, OK, Apply, Cancel.

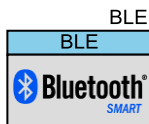
BLE IPSP Node Project Configuration

The node example project consists of the following components:

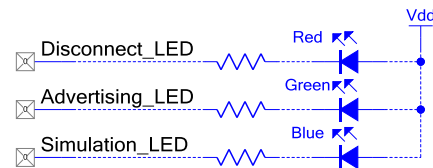
- Bluetooth Low Energy (BLE)
- UART
- Watchdog timer (WDT) that is used as general timer

The top design schematic is shown in Figure 8.

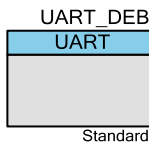
Figure 8. Top design schematic
BLE IPSP Node Example Project



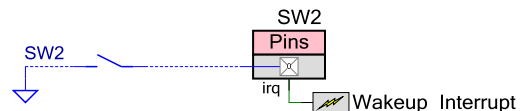
BLE component is configured as IPSP node



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 events simulation.



UART is used for transmitting the debug information.



The button is used to wake the device from the hibernate mode.

The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and is used to wake a device from the hibernate mode.

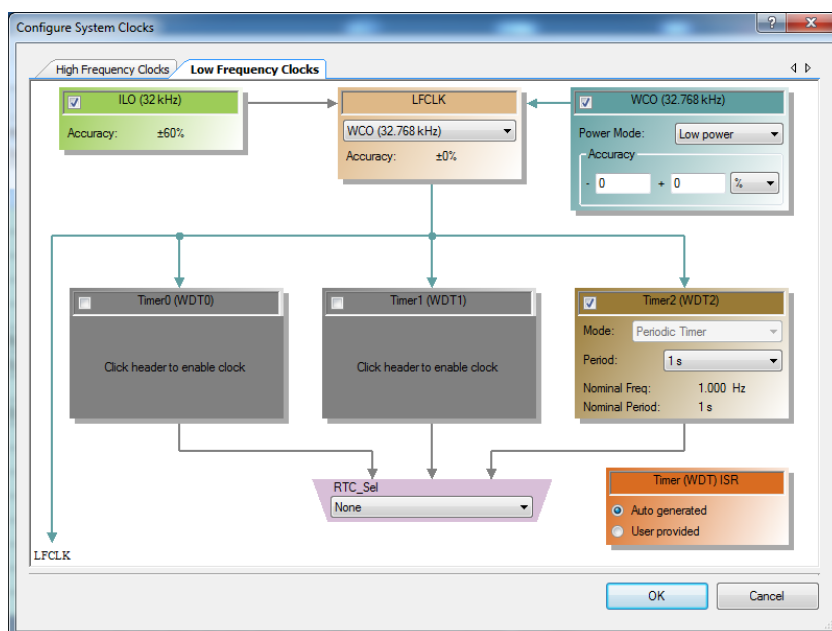
UART

This component is used for printing debug information.

Watch Dog Timer (WDT)

The WDT timer works over low power Deep Sleep mode, therefore it is used as a general timer for LED indication and simulation purpose. WDT Timer2 is configured in the Low Frequency Clocks tab of the Clocks configuration in the Design Wide Resources (DWR).

Figure 9. WDT Timer2 settings



Bluetooth Low Energy (BLE)

The BLE component is configured as Internet Protocol Support Profile in Node role operation. Node Advertisement packet is configured to connectable undirected advertising type with complete local name and IPSS service UUID. L2CAP MTU size is set to 1280.

Figure 10. GATT settings

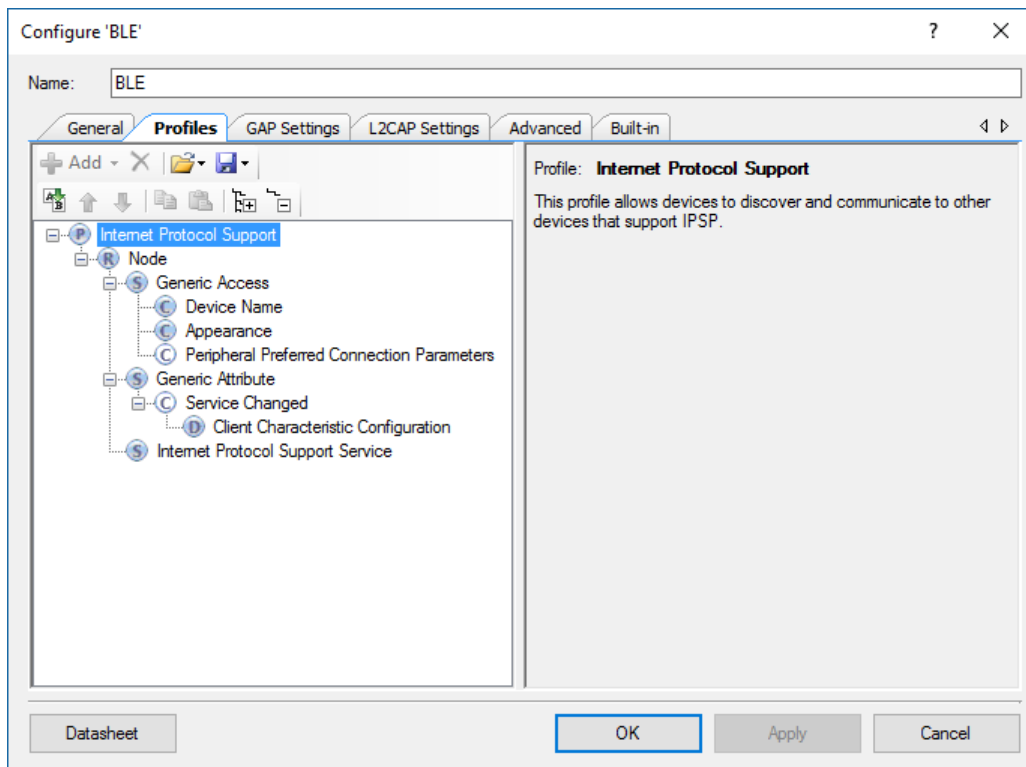


Figure 11. GAP settings

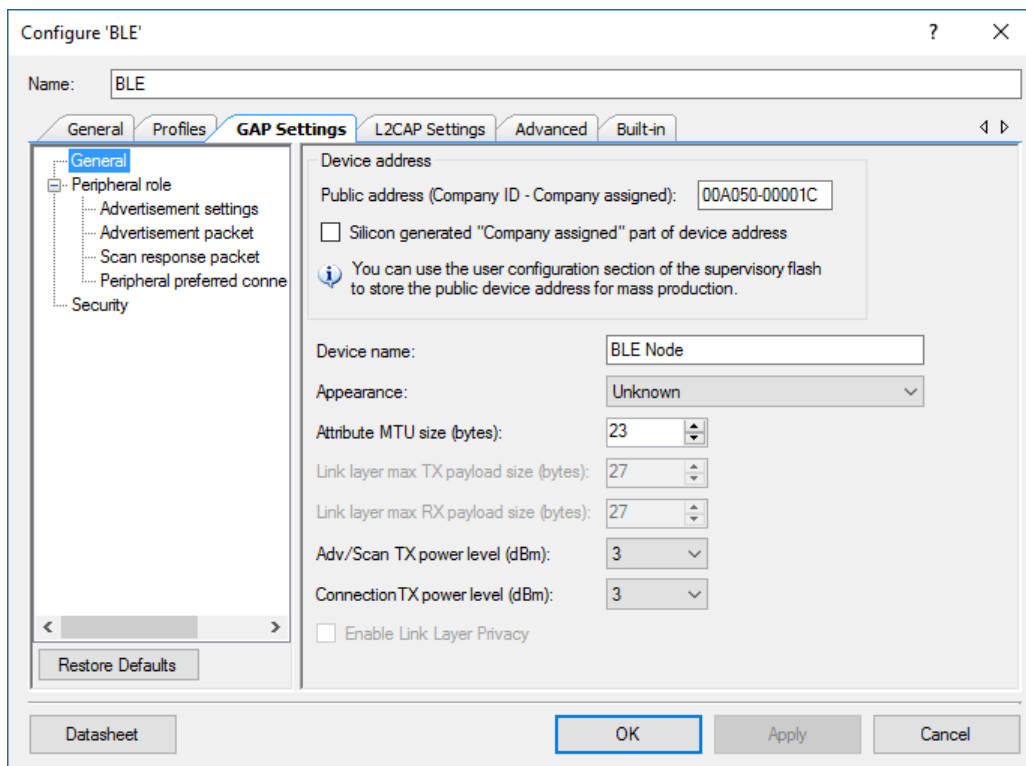


Figure 12. GAP settings -> Advertisement settings

Configure 'BLE'

Name: BLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

Advertisement data settings:

Name	Value
<input checked="" type="checkbox"/> Flags	
<input checked="" type="checkbox"/> Limited discoverable mode	
<input checked="" type="checkbox"/> BR/EDR not supported	
<input checked="" type="checkbox"/> Local Name	
Local name	Complete
<input type="checkbox"/> TX Power Level	
<input type="checkbox"/> Slave Connection Interval Range	
<input checked="" type="checkbox"/> Service UUID	
<input checked="" type="checkbox"/> Internet Protocol Support Service	
<input type="checkbox"/> Service Solicitation	
<input type="checkbox"/> Service Data	
<input type="checkbox"/> Service Manager TK Value	
<input type="checkbox"/> Appearance	
<input type="checkbox"/> Public Target Address	
<input type="checkbox"/> Random Target Address	
<input type="checkbox"/> Advertising Interval	
<input type="checkbox"/> LE Bluetooth Device Address	
<input type="checkbox"/> LE Role	
<input type="checkbox"/> URI	
<input type="checkbox"/> 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	0x09	[3]
<<Local Name>>	0x09	[4]
'B'	0x42	[5]
'L'	0x4C	[6]
'E'	0x45	[7]
''	0x20	[8]
'N'	0x4E	[9]
'o'	0x6F	[10]
'd'	0x64	[11]
'e'	0x65	[12]
AD Data 3: << Complete list of 16-bit UUIDs available>>		
Length	0x03	[13]
<< Complete list of 16-bit UUIDs available>>	0x03	[14]
Service: Internet Protocol Support Service		
[0]	0x20	[15]

Restore Defaults

Datasheet

OK Apply Cancel

Figure 13. Security settings

Configure 'BLE'

Name: BLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

Security mode: Mode 1

Security level: Unauthenticated pairing with encryption

Strict pairing: No

I/O capabilities: No Input No Output

Keypress notifications: No

Bonding requirement: Bonding

Encryption key size (bytes): 16

Restore Defaults

Datasheet

OK Apply Cancel

BLE IPSP Router Project Description

The project demonstrates the BLE component functionality configured as a Router. For operation the example project uses a callback function `AppCallBack()` for receiving the generic and L2CAP events from the BLE Stack.

To start the example project operation, build it and program into development kit BLE Dongle. Right after the startup, the BLE, UART, and ISR components are initialized. After the initialization the BLE component begins its operation that can be seen on the user LED which starts blinking with a blue color. This indicates that the device has started scanning. After 180 seconds timeout, if no peripheral device has been connected, the Router stops scanning. User LED is turned off indicating the disconnection state and the system enters into the hibernate mode. Press the user button on BLE Dongle (SW2) to wake up the system and start re-scanning.

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

The example project uses UART component for displaying debug information and also for sending commands through the Terminal emulator app. Commands are the procedures which a user can perform. The list of the commands is shown below:

Command	Description
'z'+'#'	Select specific peer device, where '#' – sequence number from advertising packet (0-7). Default value: 0.
'c'	Send connection request to selected peer device.
'd'	Send disconnect request to peer device.
'v'	Cancel connection request.
's'	Start discovery procedure.
'1'	Send Data packet to Node though IPSP channel

The above list is prompted to Terminal emulator when 'h' is entered in the app.

To connect to the Node device, select a specific device by pressing 'z' + device number listed along with the advertising report in the terminal and send a connection request to the device ('c') when the device is advertising.

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

When GAP connection is established, after `CYBLE_EVT_GATT_CONNECT_IND` event, Router automatically initiates an L2CAP LE credit based connection with a PSM set to `LE_PSM_IPSP`.

Use '1' command to generate and send first Data packet to Node though IPSP channel. Sent data will be compared with the received data in response packed after `CYBLE_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 and "Wraparound failed" message will indicate a failure.

The Router updates the LE credits dynamically, when the credit count goes below the low mark (CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND event), to allow continuous transfer of data between the Node and the Router.

LE IPSP Node Project Description

The project demonstrates the BLE component functionality configured as a Node.

For operation the example project uses a callback function - AppCallBack() for receiving the generic and L2CAP events from the BLE Stack. CyBle_GappStartAdvertisement() API is called after CYBLE_EVT_STACK_ON event to start advertising with the packet shown in Figure 12.

To start the example project operation, build it and program into baseboard of development kit. Right after the startup, the BLE, UART, and ISR components are initialized. After the initialization the BLE component begins its operation that can be seen on the RGB LED which starts blinking with a green color. This indicates that the device has started advertising. After 180 seconds timeout, if no central device has been connected, the Node stops advertising, a red LED is turned on indicating the disconnection state and the system enters into the hibernate mode. Press the mechanical button on CY8CKIT-042 BLE (SW2) to wake up the system and start re-advertising.

When a Client has connected successfully, both red and green LEDs are turned off. Blinking blue LED indicates data packet transaction.

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

The Node automatically accept L2CAP LE credit based connection request with a PSM set to LE_PSM_IPSP after CYBLE_EVT_L2CAP_CBFC_CONN_IND event.

Received data after CYBLE_EVT_L2CAP_CBFC_DATA_READ event are automatically wrapped back to the Router.

The Node updates the LE credits dynamically, when the credit count goes below the low mark (CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND event), to allow continuous transfer of data between the Node and the Router.

Expected Results

The BLE IPSP Router project is intended to work in pair with BLE IPSP Node. Two projects send log messages through UART. After starting, the Router project logs Advertising and Scan response reports from 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. Use this number after 'z' command to select required Node if multiple Node devices are available. Press 'c' to connect to Node. Press '1' to start wraparound test. Blue LED on Node indicates data transfer process. In case of wraparound

data validation failure Blue LED will stop blinking and “Wraparound failed” message will appear in Router UART log.

The example log is shown below:

```
BLE IPSP Router Example Project
Stack Version: 2.1.0.11
Bluetooth On, StartScan with addr: 00a05000001d
CYBLE_EVT_GAPC_SCAN_START_STOP, state: 3
Advertisement report: eventType = 0, peerAddrType - 0, peerBdAddr - 0: 00a05000001c, rssi - -29 dBm
CYBLE_EVT_GAPC_SCAN_START_STOP, state: 5
GAPC_END_SCANNING
EVT_GATT_CONNECT_IND: 0, 4
L2CAP channel connection request sent.
EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms
CYBLE_EVT_L2CAP_CBFC_CONN_CNF: bdHandle=4, ICid=64, responce=0, connParam: mtu=1280, mps=1280, credit=1000
-> CyBle_L2capChannelDataWrite #0
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #1
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #2
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #3
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #4
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #5
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #6
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #7
<- EVT_L2CAP_CBFC_DATA_READ: ICid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #8
```

```

BLE Node Example Project
Stack Version: 2.1.0.11
Bluetooth On, StartAdvertisement with addr: 00a05000001c
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 3
EVT_GATT_CONNECT_IND: 0, 4
EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms
EVT_L2CAP_CBFC_CONN_IND: bdHandle=4, lCid=64, psm=35, connParam mtu=1280, mps=1280, credit=1000 SUCCESSFUL
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0

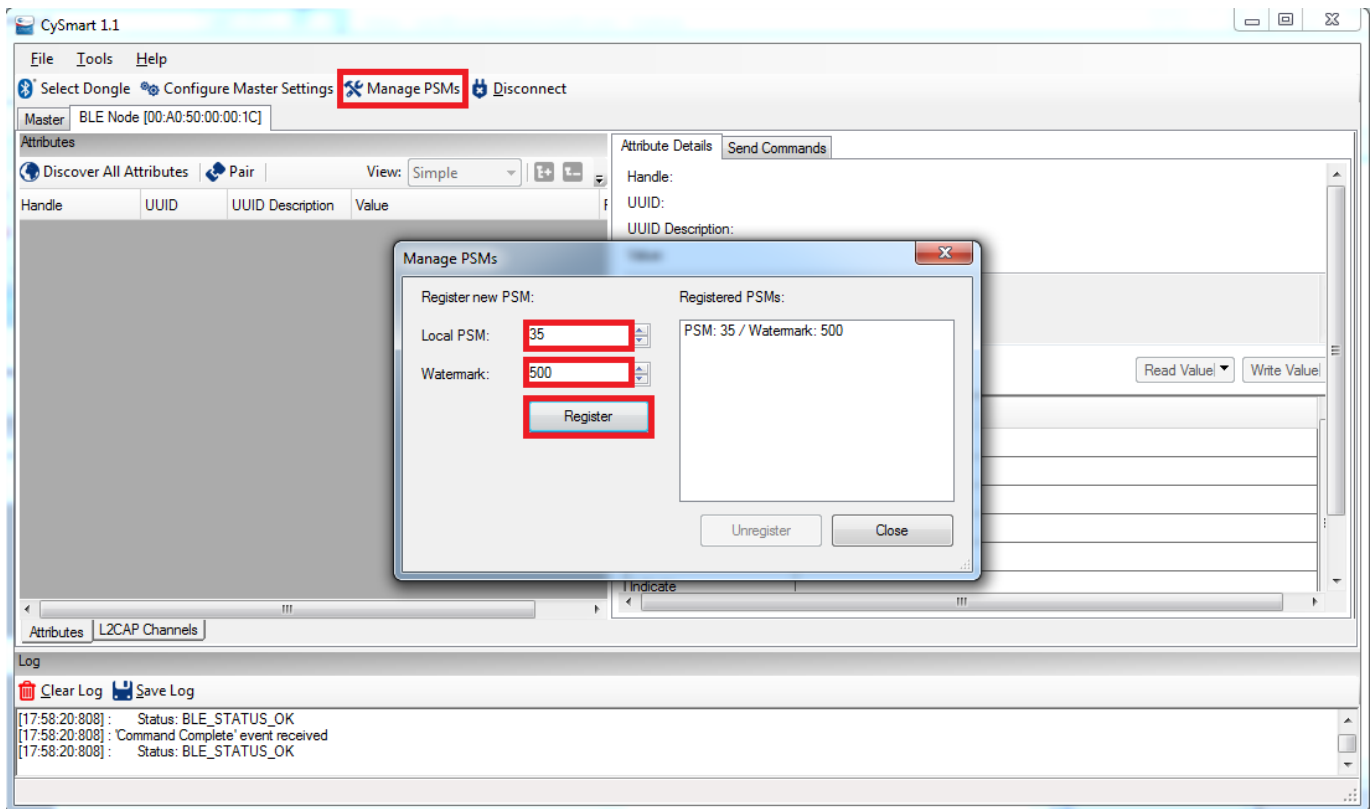
```

You can use CySmart app on a Windows PC BLE-compatible device as Client for connection to Node.

To use CySmart Windows application as Client:

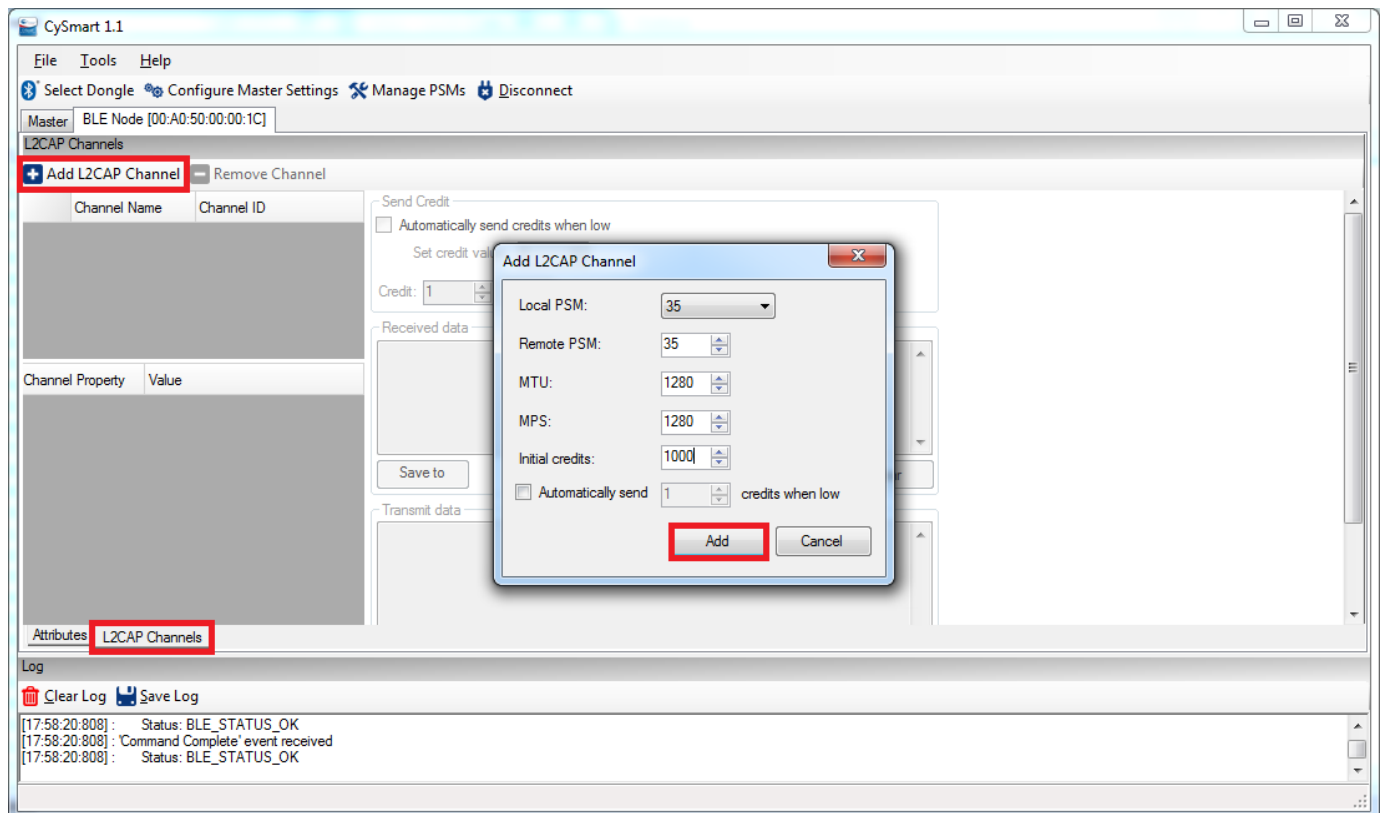
- Connect the CySmart BLE dongle to a USB port on the PC.
- Launch CySmart app and select connected dongle in dialog window.
- Reset development kit to start advertising by pressing SW1 button.
- Click **Start Scan** button to discover available devices.
- Select **BLE Node** in the list of available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

Figure 14. CySmart Manage PSM window



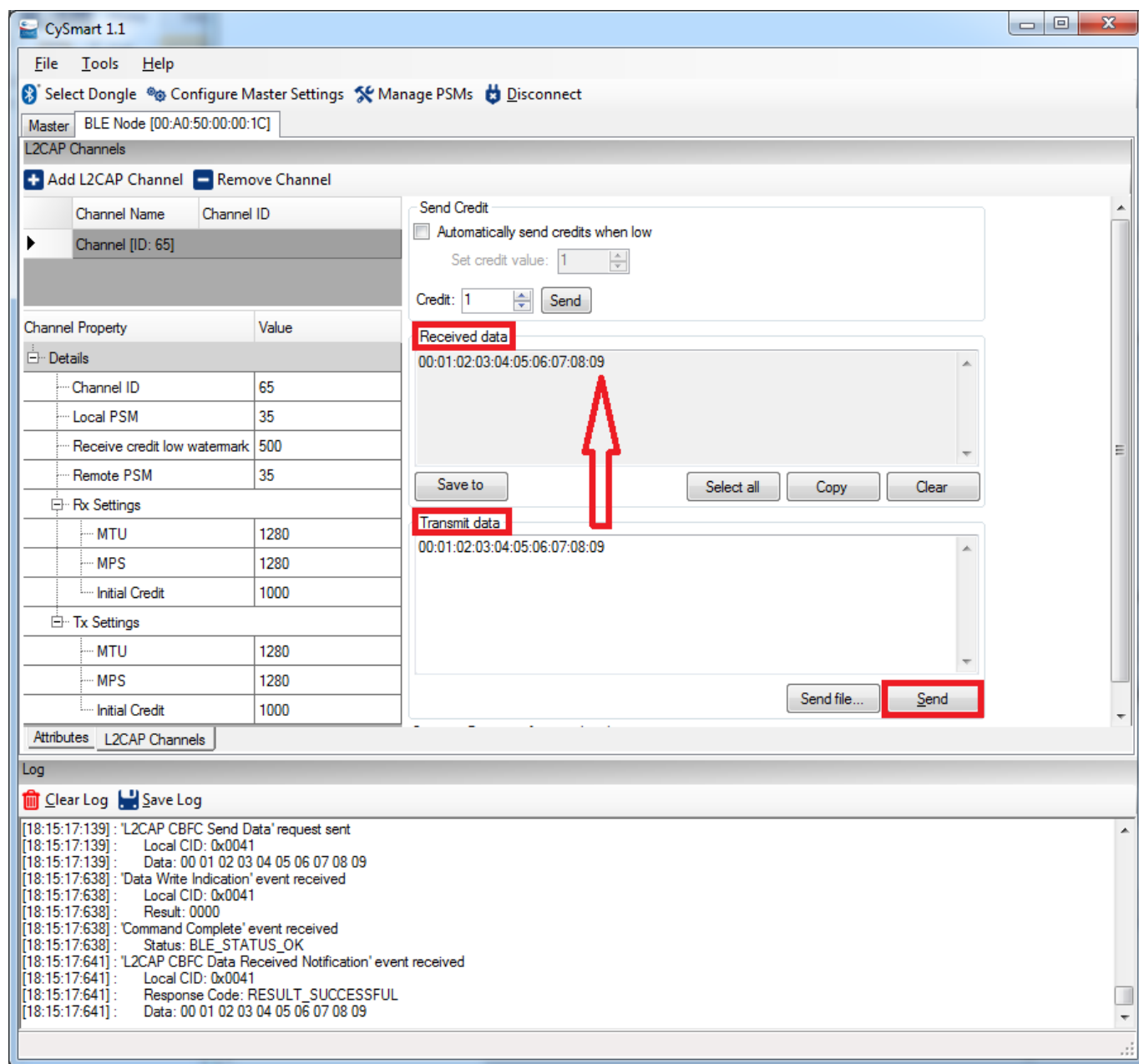
- Select **L2CAP Channels** tab and press **Add L2CAP Channel** to create L2CAP channel. In “Add L2CAP Channel” window select **Local PSM: 35**, enter **Remote PSM: 35**, **MTU: 1280**, **MPS: 1280**, **Initial credits: 1000** and press **Add** button.

Figure 15. CySmart Add L2CAP Channel window



- Now L2CAP channel is ready to transmit and receive data. Enter some data into **Transmit data** area and press **Send** button. The same data will appear in **Receive data** area

Figure 16. CySmart L2CAP channels window.



If you have problems with usage of CySmart app, refer to [CySmart User Guide](#).

CySmart mobile app does not have IPSP profile support.

Cypress Semiconductor Corporation, 2009-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.