

Objective

This example demonstrates how to use PSoC 4 BLE/PRoC BLE device as SPI-BLE Peripheral device.

Overview:

This code example uses a custom BLE profile with two custom service to demonstrate the SPI-BLE peripheral functionality. One custom service is used to notify the data written by the SPI master to the client and the second one is used to update the SPI TX FIFO when the client writes to the GATT DB of the peripheral device

Requirements:

<i>Programming Language</i>	: C (GCC 4.8.4)
<i>Associated Parts</i>	: PSoC 4 BLE, PRoC BLE
<i>Required software</i>	: PSoC Creator 3.1 SP2 , PSoC Programmer 3.22.3 , CySmart PC application
<i>Required hardware</i>	: CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit , SPI master device
<i>Optional hardware</i>	: Bluetooth sniffer

Project Description:

This project demonstrates the following.

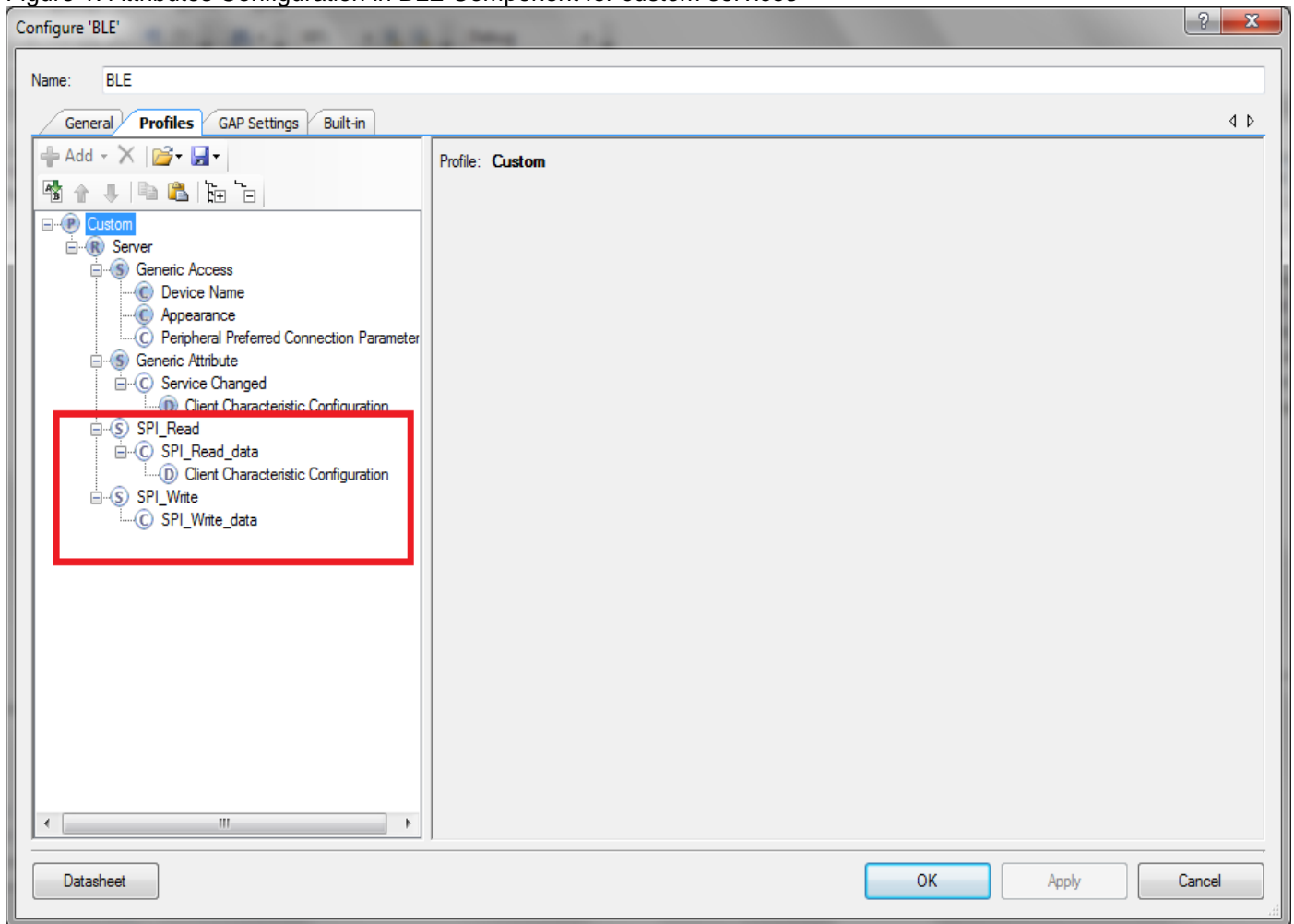
- SPI-BLE bridge implementation – Peripheral role
- Connection with any central device
- Two custom services in a single profile
- Low power implementation for coin-cell operation

The BLE profile in this project consists of two custom services: SPI_Read and SPI_Write. The SPI_Read service consists of one custom characteristics which is used to send the SPI data written by the SPI master to the client as notifications. This simulates the SPI write in case of wired SPI.

The second custom service SPI_Write consists of one custom characteristics through which the GATT client can write data to the peripheral device. The peripheral device in turn update the data written by the GATT client to the SPI TX FIFO registers for the SPI master to read. This simulates the SPI slave updating its TX FIFO registers for the master to read.

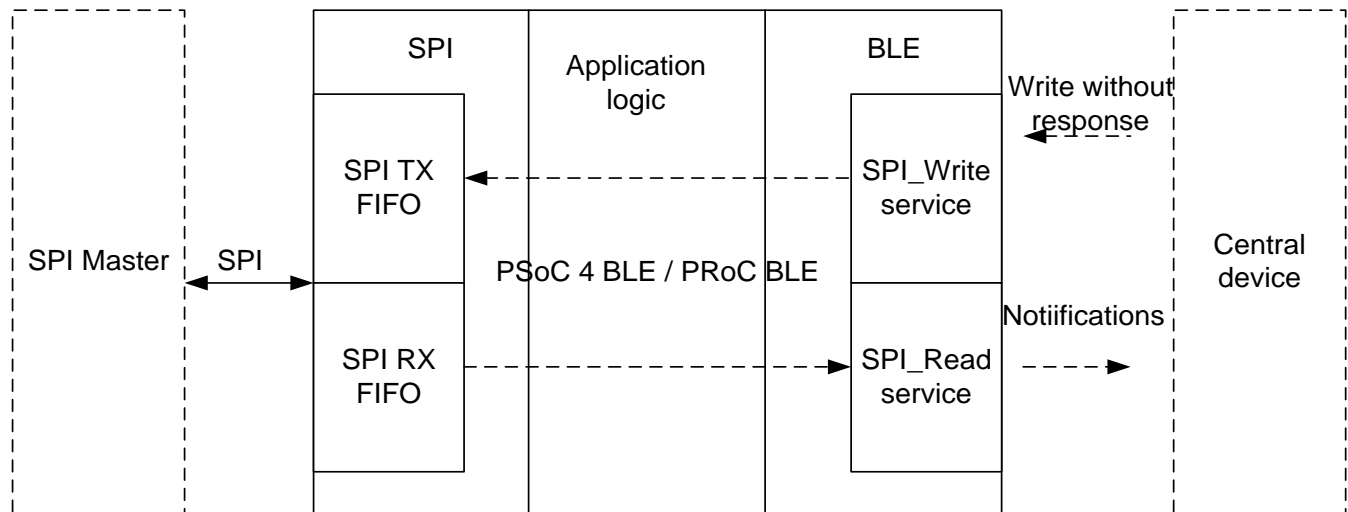
These properties for the custom service/characteristics are configured in the BLE component under the Profiles tab, as shown in Figure 1 below

Figure 1. Attributes Configuration in BLE Component for custom services



A simple block diagram of the implementation is shown in the figure 2.

Figure 2. Block diagram of the implementation



The project consist of the following files:

Main.c/h

These files contain the main function, which is the entry point and execution of the firmware application. It also contains function definition for initialization of the system.

App_BLE.c/h

These files contain all the macros and function definitions related to BLE communication and operation. It contains the event callback function definition that is registered with the BLE component startup and used by the component to send BLE-related events from the BLE stack to the application layer for processing. It contains a method to send notifications to the GATT client device and process the Write commands on the SPI_Read characteristic by the GATT client device.

Low_power.c/h

These files contain the function to handle low-power mode. This function is continuously called in the main loop and is responsible for pushing the BLE hardware block (BLESS) as well as the CPU to Deep Sleep mode as much as possible. The wakeup source is either the BLE hardware block Link Layer internal timer or the interrupt from the I2C address match. This allows for very low power mode implementation and operation using a coin cell.

App_SPI.c/h

These files contain the function to handle the SPI read and write activity.

Config.h

This file has macros to enable or disable – Low power mode implementation and LED indication

Additionally there are LED indications to show the state of the device.

BLUE LED – Device is in advertisement state

RED LED – Device is in disconnected state

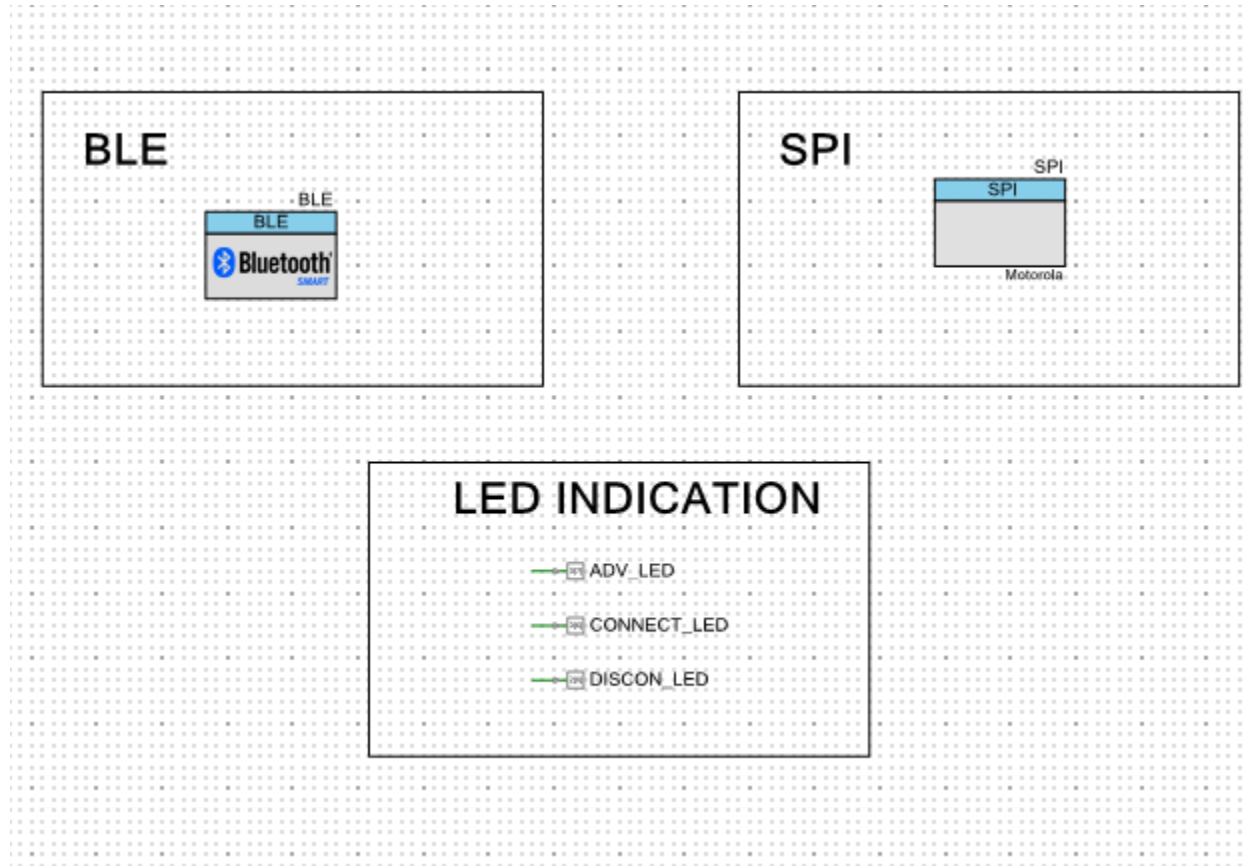
GREEN LED – Device is in connected state

To measure the best power consumption number, disable the LED indication.

The SWD pin are configured as GPIO to get the lowest possible current consumption number.

The top design of the project is shown in the figure 3 below.

Figure 3 Top Design for SPI_BLE Bridge Project



Hardware Connections

Connect the SPI lines of the SPI master to that of the PSoC 4 BLE / PRoC BLE.

The pin assignment for this project is in **SPI_BLE_Bridge.cydwr** in the Workspace Explorer, as shown in [Figure 4](#)

[Figure 4](#) Pin Selection for SPI_BLE Project

Alias	Name /	Port	Pin	Lock
\SPI:miso_s\	P0[1] LPCOMP:in_n[0], TCPWM0:line_out_compl, SCB1:uart_tx, SCB1:i2c_scl, SCB1:spi_miso	▼	20	▼ <input checked="" type="checkbox"/>
\SPI:mosi_s\	P0[0] LPCOMP:in_p[0], TCPWM0:line_out, SCB1:uart_rx, SCB1:i2c_sda, SCB1:spi_mosi	▼	19	▼ <input checked="" type="checkbox"/>
\SPI:sclk_s\	P0[3] TCPWM1:line_out_compl, SCB1:uart_cts, LPCOMP:comp[1], SCB1:spi_clk	▼	22	▼ <input checked="" type="checkbox"/>
\SPI:ss_s\	P0[2] TCPWM1:line_out, SCB1:uart_rts, LPCOMP:comp[0], SCB1:spi_select[0]	▼	21	▼ <input checked="" type="checkbox"/>
ADV_LED	P3[7] SARMUX:pads[7], TCPWM3:line_out_compl, SCB1:uart_cts, SRSS:ext_clk_1f	▼	54	▼ <input checked="" type="checkbox"/>
CONNECT_LED	P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart_rts	▼	53	▼ <input checked="" type="checkbox"/>
DISCON_LED	P2[6] OAO:vplus_alt	▼	43	▼ <input checked="" type="checkbox"/>

Verify Output

The project can be verified by using the CySmart Central Emulation Tool and BLE Dongle.

CySmart Central Emulation Tool

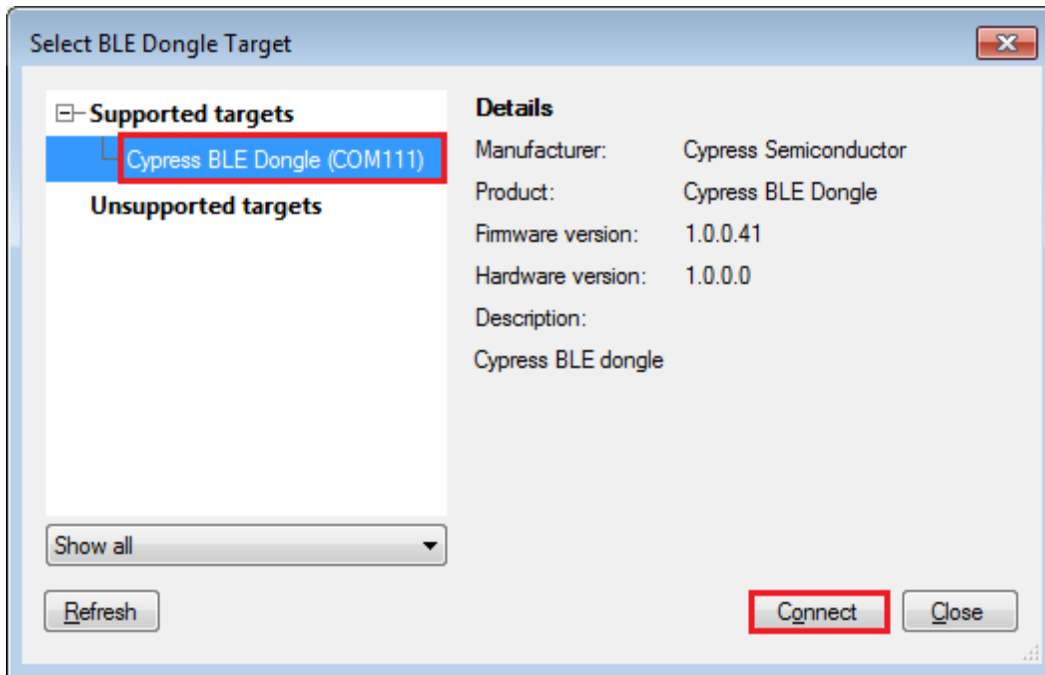
To verify the SPI_BLE_Bridge project using the CySmart Central Emulation Tool, follow these steps:

Note: Refer [CySmart BLE Host Emulation tool](#) to learn how to use the tool.

1. Connect the BLE Dongle to one of the USB ports on the PC and wait for the dongle to enumerate.
2. Start the CySmart Central Emulation Tool on the PC by going to **Start > All Programs > Cypress> CySmart <version> > CySmart <version>**. You will see a list of BLE Dongles connected to it.

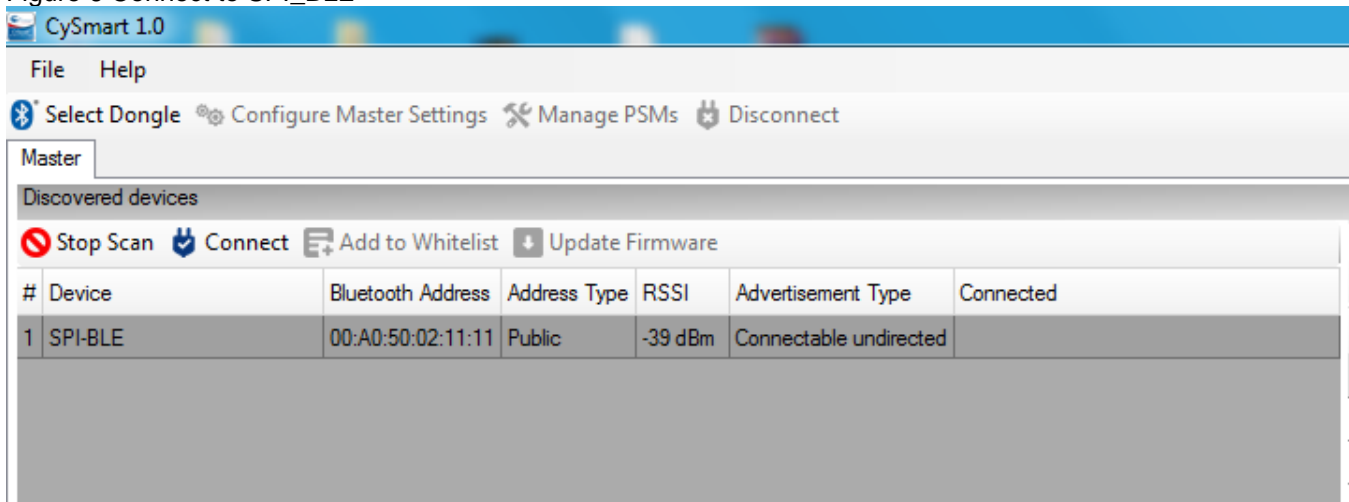
If no BLE Dongle is found, click **Refresh**. Select the BLE Dongle and click **Connect**.

Figure 5 Connect to BLE Dongle



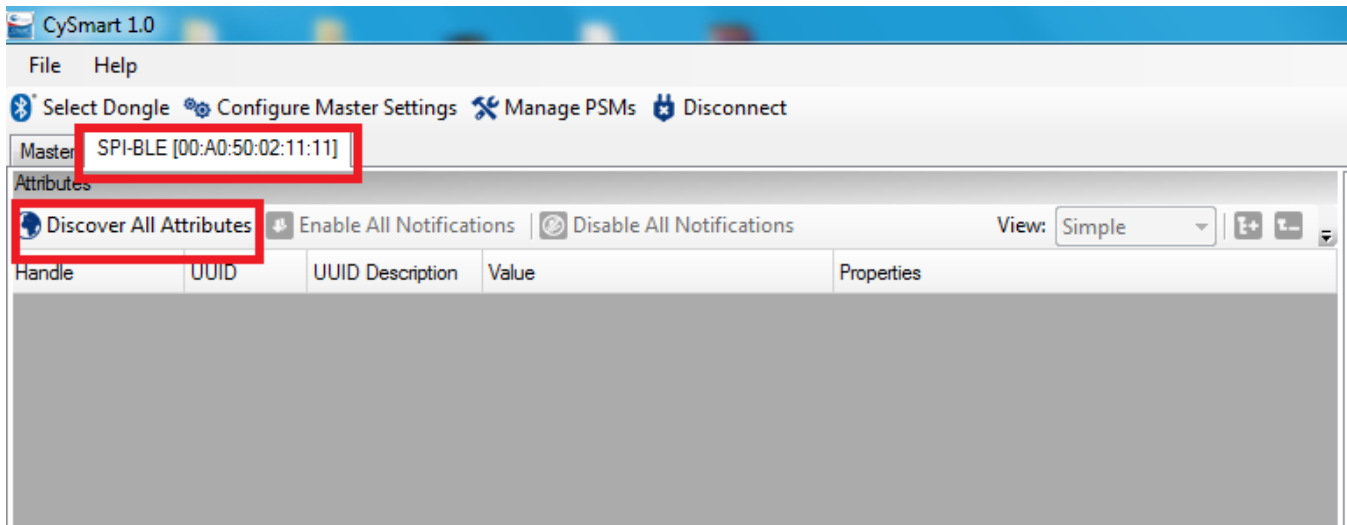
3. Connect the PSoC 4 BLE/PRoC BLE module on the J10 and J11 headers on the BLE Pioneer Kit.
4. Connect the SPI lines of the SPI master to that of PRoC-BLE or PSoC 4 BLE
5. Power the BLE Pioneer Kit through the USB connector **J13**.
6. Program the BLE Pioneer Kit with the SPI_BLE_Bridge project.
7. After successful programming the BLUE LED will glow indicating that the device is in advertisement mode.
8. On the CySmart Central Emulation Tool, click **Start Scan** to see the list of available BLE peripheral devices.
9. Double-click the **SPI_BLE** device to connect, or click **SPI_BLE** and then Click **Connect**. The GREEN LED on the device will start glowing indicating it is connected to the Client.

Figure 6 Connect to SPI_BLE



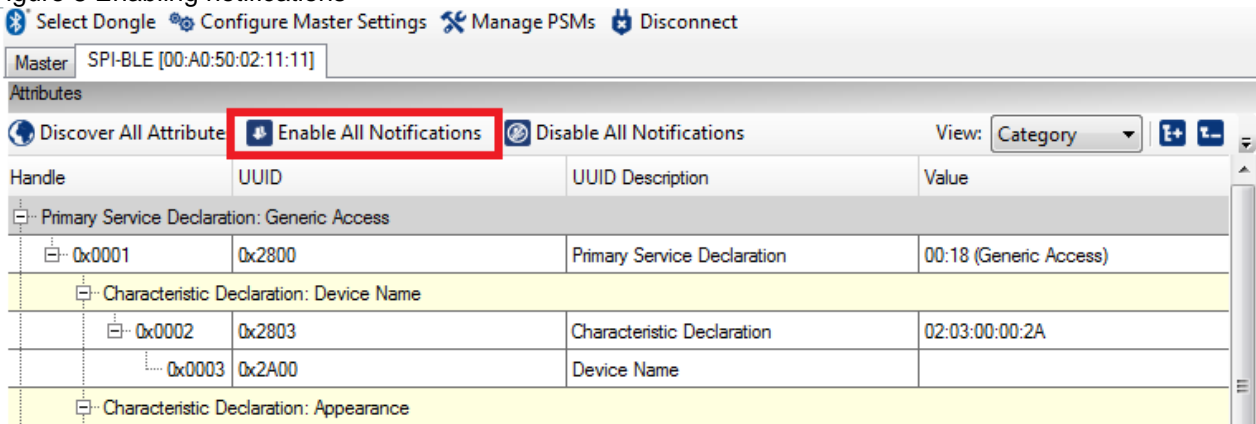
10. Click **Discover All Attributes** to find all attributes supported.

Figure 7 Discover all attributes of SPI_BLE



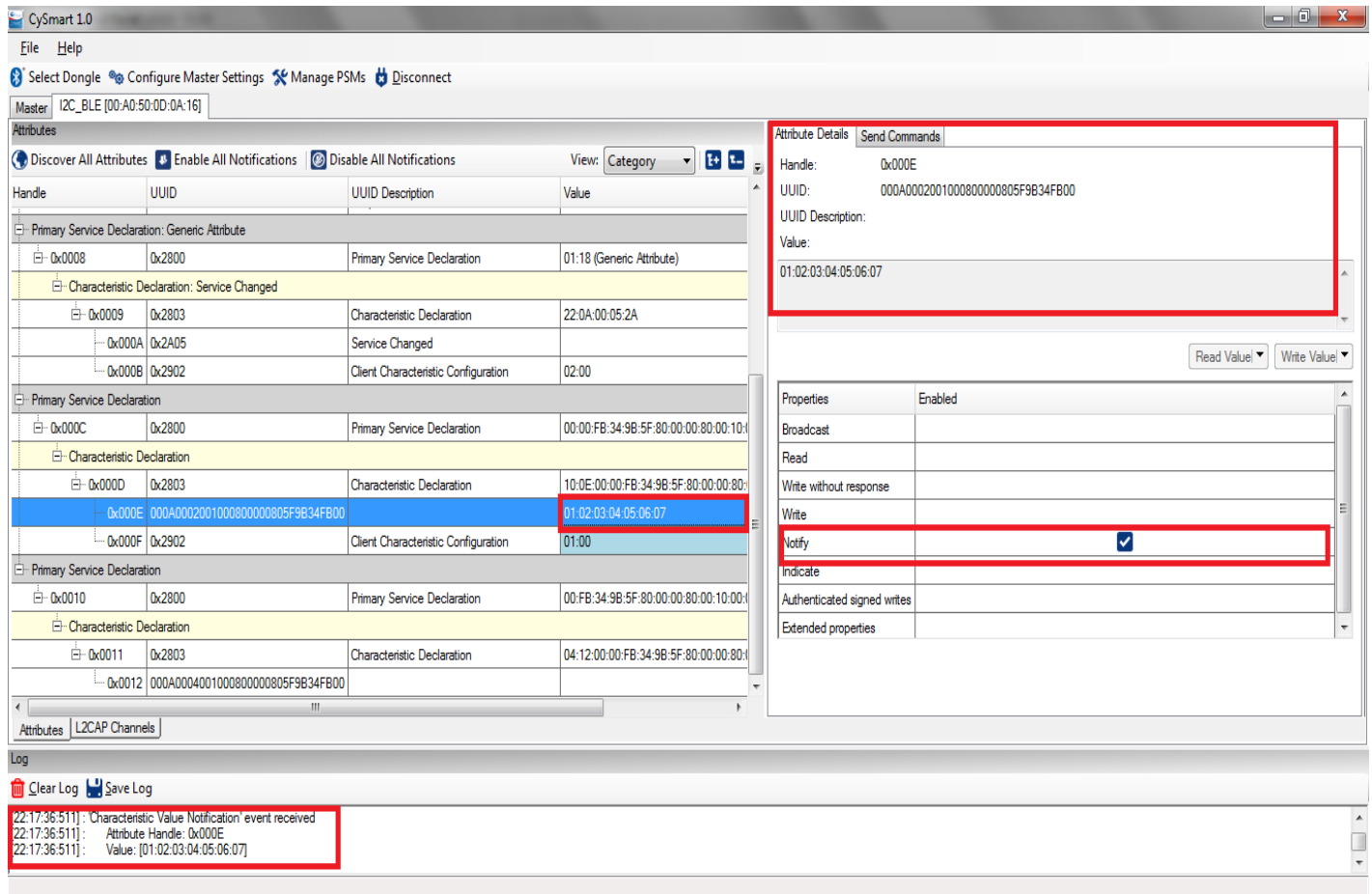
11. Click on **Enable All Notifications** to enable all notifications.

Figure 8 Enabling notifications



12. Write data from the SPI master to PSoC 4 BLE or PSoC 4 BLE. The data written by the SPI master is sent as notifications to the client device

Figure 13 Check notifications received

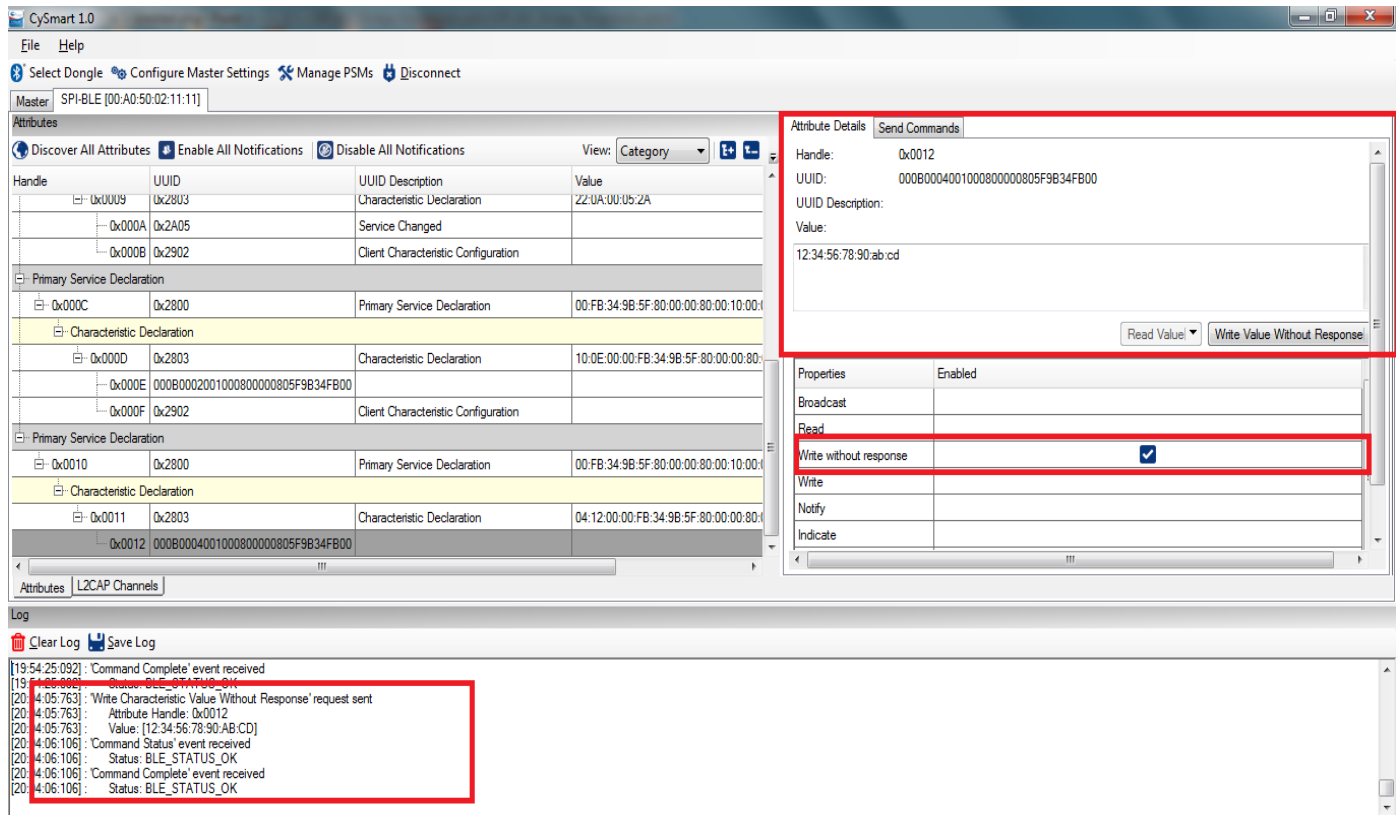


The screenshot shows the CySmart 1.0 application window. The main area displays a list of BLE attributes for a device with address I2C_BLE [00:A0:50:0D:0A:16]. The attributes are organized into categories like Primary Service Declaration and Characteristic Declaration. A specific attribute with handle 0x000E and UUID 000A0002001000800000805F9B34FB00 is highlighted in blue. The value for this attribute is 01:02:03:04:05:06:07. To the right, the 'Attribute Details' panel shows the same information. At the bottom, the 'Log' panel shows a message: 'Characteristic Value Notification' event received, Attribute Handle: 0x000E, Value: [01:02:03:04:05:06:07].

Handle	UUID	UUID Description	Value
0x0008	0x2800	Primary Service Declaration	01:18 (Generic Attribute)
0x0009	0x2803	Characteristic Declaration	22:0A:00:05:2A
0x000A	0x2A05	Service Changed	
0x000B	0x2902	Client Characteristic Configuration	02:00
0x000C	0x2800	Primary Service Declaration	00:00:FB:34:9B:5F:80:00:00:80:00:10:00:00:00:00
0x000D	0x2803	Characteristic Declaration	10:0E:00:00:FB:34:9B:5F:80:00:00:80:00:00:00:00
0x000E	000A0002001000800000805F9B34FB00		01:02:03:04:05:06:07
0x000F	0x2902	Client Characteristic Configuration	01:00
0x0010	0x2800	Primary Service Declaration	00:FB:34:9B:5F:80:00:00:80:00:10:00:00:00:00
0x0011	0x2803	Characteristic Declaration	04:12:00:00:FB:34:9B:5F:80:00:00:80:00:00:00
0x0012	000A0004001000800000805F9B34FB00		

13. On CySmart select the handle 0x0012 and perform write without response to write to peripheral GATT DB

Figure 14 Write data to GATT DB of peripheral



The screenshot shows the CySmart 1.0 application interface. The 'Master' tab is selected, and the device 'SPI-BLE [00:A0:50:02:11:11]' is connected. The 'Attributes' tab is active, displaying a list of characteristics. The 'Attribute Details' panel on the right shows the details for the selected characteristic (Handle: 0x0012). The 'Write without response' checkbox is checked. The 'Log' panel at the bottom shows the command history, including the 'Write Characteristic Value Without Response' request.

Handle	UUID	UUID Description	Value
0x0009	0x2803	Characteristic Declaration	ZZ:0A:00:05:2A
0x000A	0x2A05	Service Changed	
0x000B	0x2902	Client Characteristic Configuration	
Primary Service Declaration			
0x000C	0x2800	Primary Service Declaration	00:FB:34:9B:5F:80:00:00:80:00:10:00:00:00:00:00:00
Characteristic Declaration			
0x000D	0x2803	Characteristic Declaration	10:0E:00:00:FB:34:9B:5F:80:00:00:80:00:00:00:00:00
0x000E	000B00002001000800000805F9B34FB00	Characteristic Declaration	
0x000F	0x2902	Client Characteristic Configuration	
Primary Service Declaration			
0x0010	0x2800	Primary Service Declaration	00:FB:34:9B:5F:80:00:00:80:00:10:00:00:00:00:00:00
Characteristic Declaration			
0x0011	0x2803	Characteristic Declaration	04:12:00:00:FB:34:9B:5F:80:00:00:80:00:00:00:00:00
0x0012	000B00004001000800000805F9B34FB00	Characteristic Declaration	

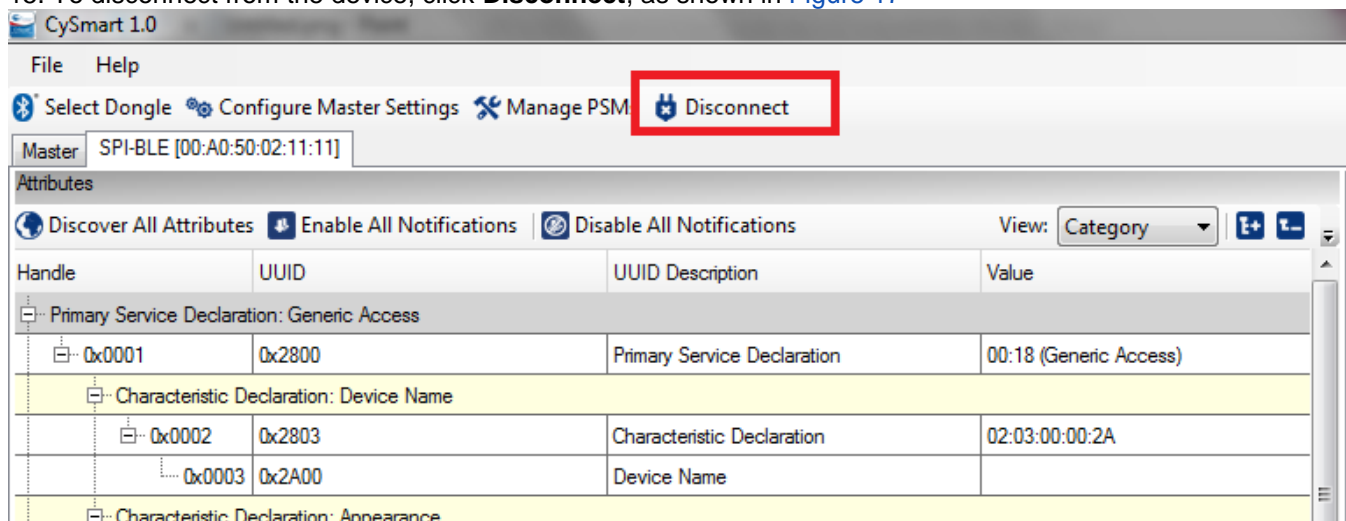
Log:

```

[19:54:25:092] : Command Complete' event received
[19:54:25:092] : Status: BLE_STATUS_OK
[20:4:05:763] : 'Write Characteristic Value Without Response' request sent
[20:4:05:763] : Attribute Handle: 0x0012
[20:4:05:763] : Value: [12:34:56:78:90:AB:CD]
[20:4:06:106] : Command Status' event received
[20:4:06:106] : Status: BLE_STATUS_OK
[20:4:06:106] : Command Complete' event received
[20:4:06:106] : Status: BLE_STATUS_OK
  
```

14. Read data from the PProC-BLE/PSoC 4 BLE device. The data read by the SPI master will be the same as the data written by the client.

15. To disconnect from the device, click **Disconnect**, as shown in Figure 17



The screenshot shows the CySmart 1.0 application interface. The 'Master' tab is selected, and the device 'SPI-BLE [00:A0:50:02:11:11]' is connected. The 'Attributes' tab is active, displaying a list of characteristics. The 'Disconnect' button is highlighted in red.

Handle	UUID	UUID Description	Value
Primary Service Declaration: Generic Access			
0x0001	0x2800	Primary Service Declaration	00:18 (Generic Access)
Characteristic Declaration: Device Name			
0x0002	0x2803	Characteristic Declaration	02:03:00:00:2A
0x0003	0x2A00	Device Name	
Characteristic Declaration: Appearance			

16. After disconnection the RED LED will glow for 2 seconds and the device will start advertising as indicated by BLUE LED turning on. You can connect it back following the steps mentioned above.