Objective

This example demonstrates the use of PSoC 4 BLE as both BLE Client and Server.

Overview

This code example uses the BLE component to configure PSoC 4 BLE as Client and Server. "Client and server-peripheral" example work as Client and server with GAP role: Peripheral. It acts as Client for Device Information Service (DIS) and acts as server for Tx power level service. Example "Client and Server-Central" acts as client and server with GAP role: Central. It acts as a client for Tx power level service and server for DIS. Using UART commands we can implement the function like enabling notifications, disabling notifications, read characteristics etc.

Requirements

Tool: PSoC Creator 4.0 Update 1, Serial Terminal emulator such as TeraTerm or Putty

Programming Language: C (GCC 4.9 – Included with PSoC Creator)

Associated Parts: All PSoC 4 BLE parts and PROC BLE parts.

Related Hardware: CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit

Hardware Setup

BLE Pioneer Kit has the necessary hardware connections and dongle that are required for this example project. If you are using your own hardware, then connect the LED to P3.7, P3.6, P2.6 of PSoC4 BLE.

PSoC Creator Schematic

Figure1(a). PSoC Creator Schematic of "Client and Server-Peripheral" example project

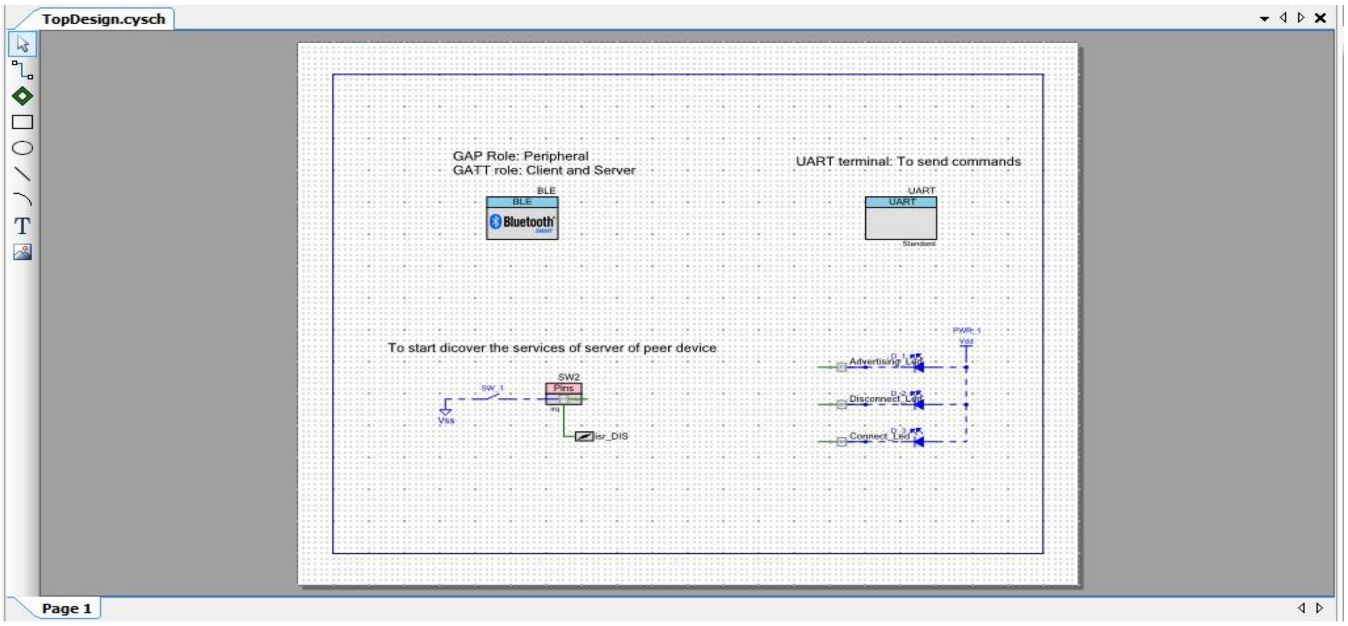
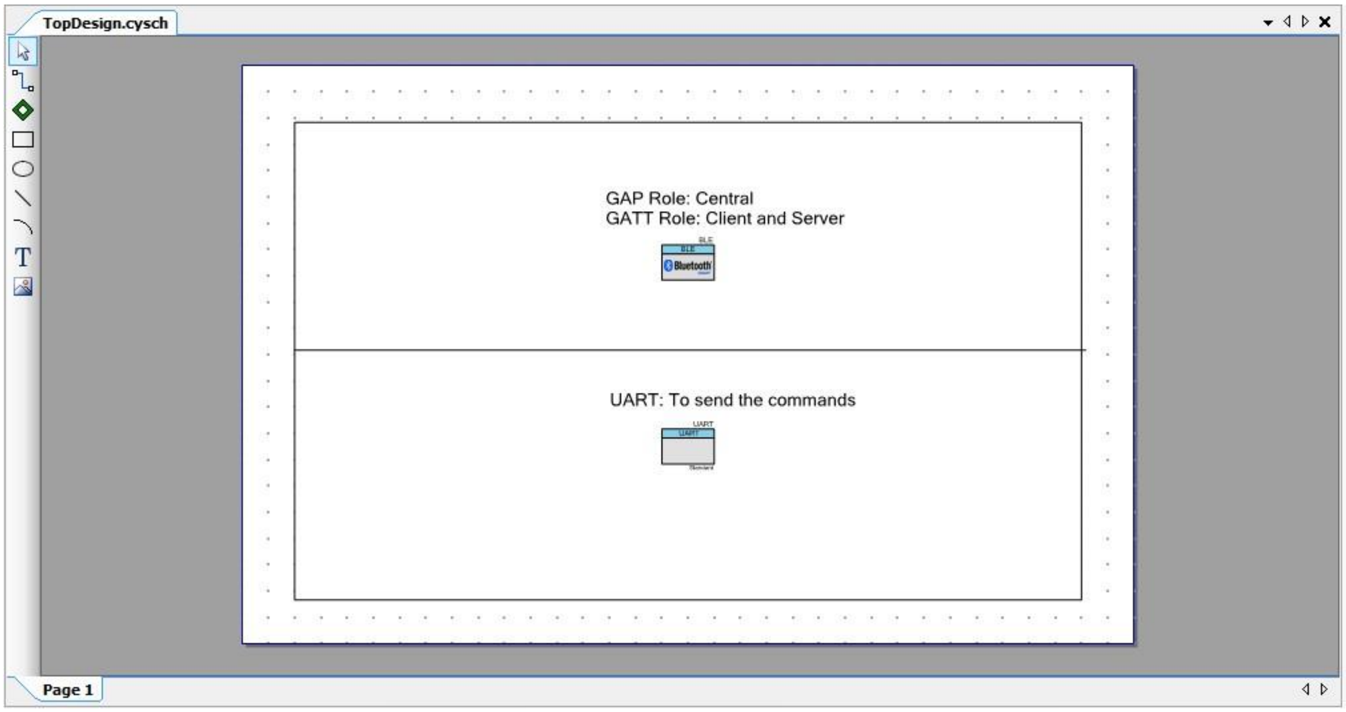
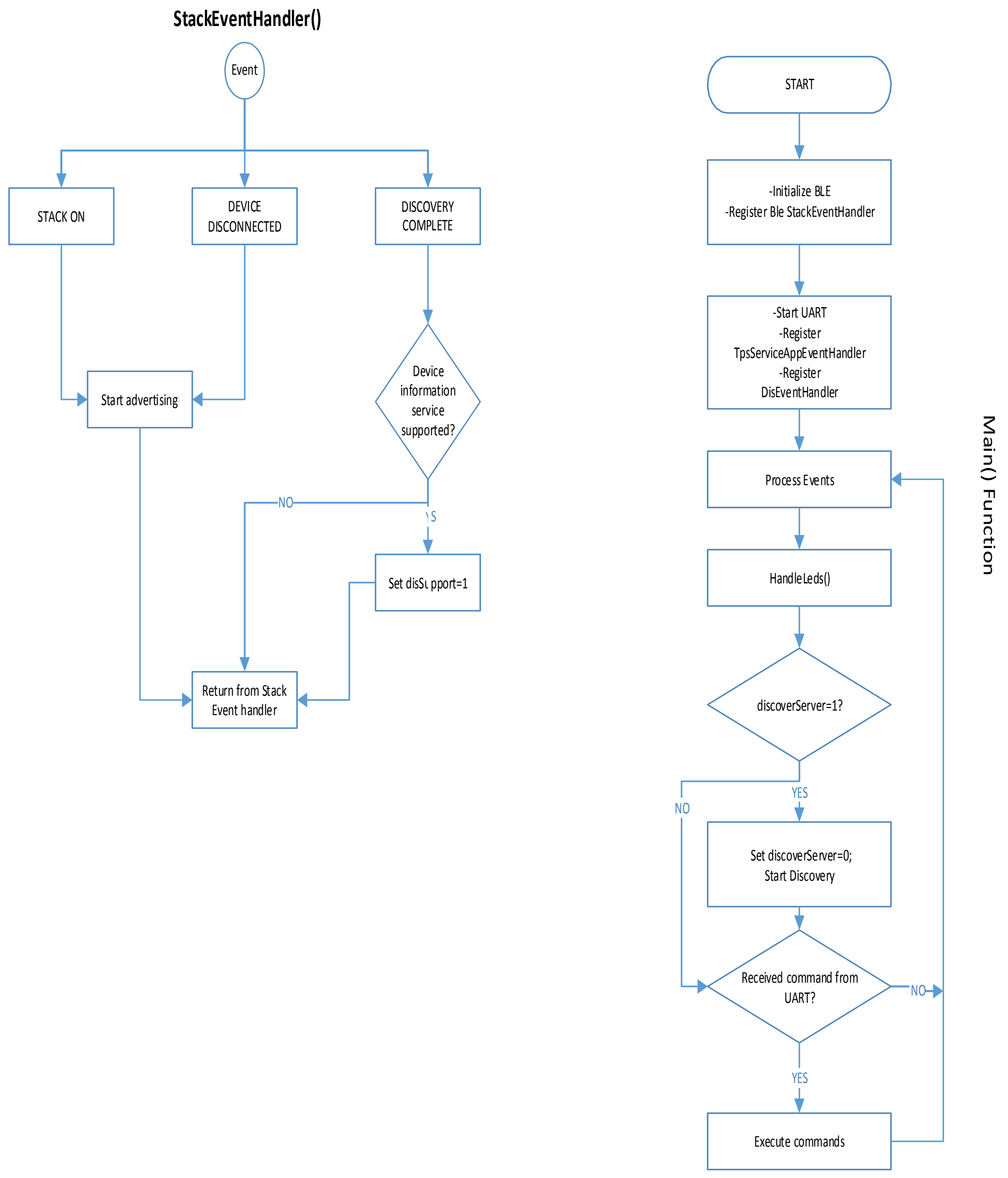


Figure 1(b) PSoC creator schematics of “Client and Server-Central” example project



Operation

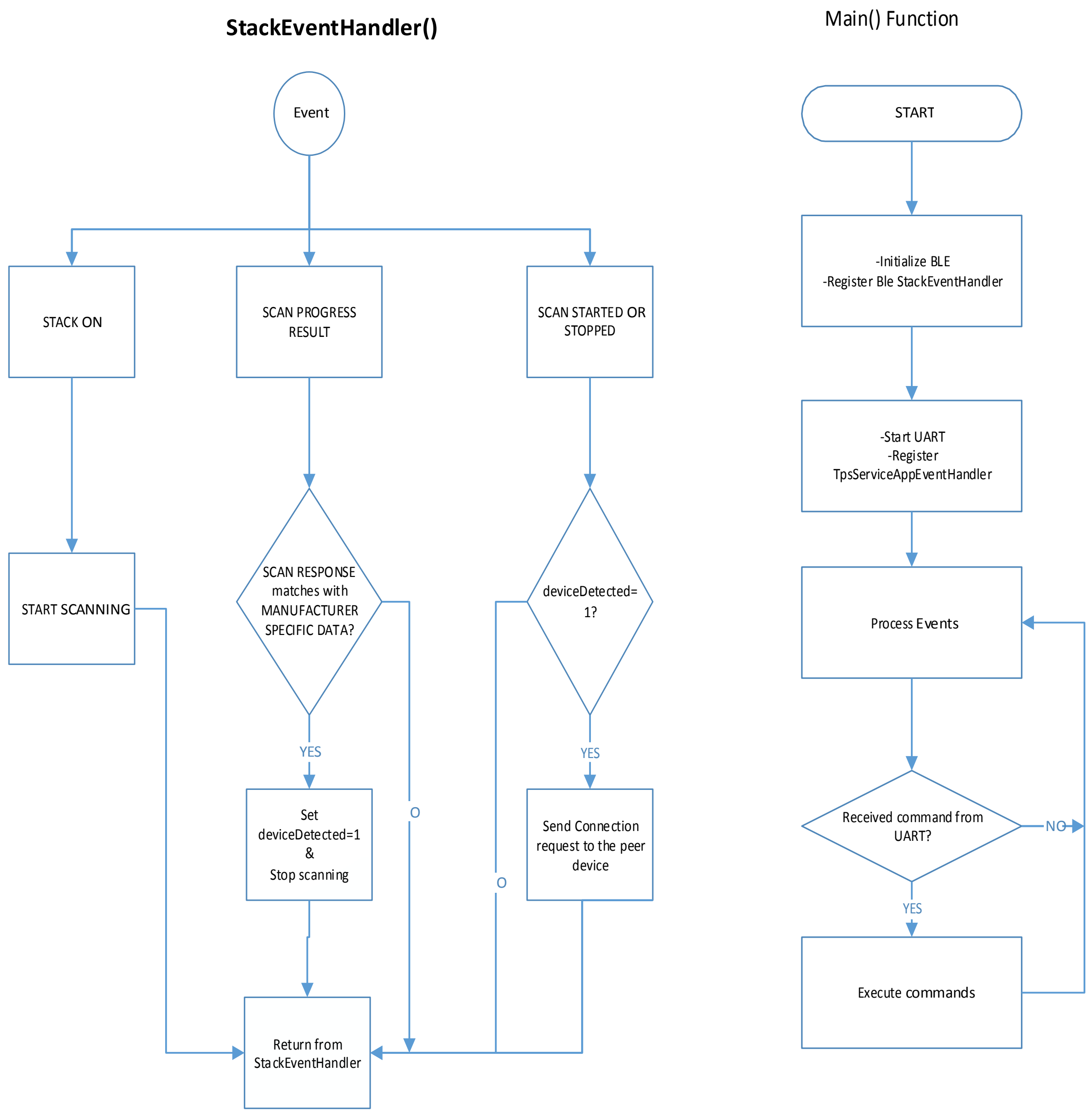
Figure 2. Firmware Flow (Client and Server-Peripheral)



1. **main() function**: This is the function which performs the initialization of the BLE stack, executes the necessary routines to process the BLE events and maintain the connection. In the initial section of the *main()* function, the API *CyBle_Start(StackEventHandler)* is called to start the BLE Component and register a callback to the stack event handler. The APIs CyBle_DisRegisterAttrCallback (DISEventHandler) and CyBle_TpsRegisterAttrCallback (TpsServiceAppEventHandler) are used to register a callback to the service related handler. Note that the callback function can assume any name – in this project, we used StackEventHandler. Once the system is initialized, *main()* function continuously operates in a *while(1)* loop executing *CyBle_ProcessEvents()*, executes UART commands and send notifications, read characteristic etc. CyBle_ProcessEvents processes the events received by the BLE stack and enables application layer to use them and take the appropriate action. Commands from UART terminal
   a. 'n'-> send the notification about the Tx power level to the Client.
   b. 'd'-> decrease the Tx power level.
   c. '1'-> read the **Manufacturer Name** characteristic of the peer server that supports DIS service.
   d. '2'-> read the **MODEL NUMBER** characteristic of the peer server that support DIS service.
   e. '3'-> read the **SERIAL NUMBER** characteristic of the peer server that supports DIS service
   f. '4'-> read the **HARDWARE Rev** characteristics of the peer server that supports DIS service
   g. '5'-> read the **FIRMWARE Rev** characteristic of the peer server that supports DIS service
   h. '6'-> read the **SOFTWARE Rev** characteristic of the peer server that support DIS service
   i. '7'-> read the **SYSTEM ID** characteristic of the peer server that supports DIS service
   j. '8'-> read the **IEEE 11073-20601** characteristic of the peer server that supports DIS service
   k. '9'-> read the **PNP-ID** characteristic of the peer server that supports DIS service.

2. **StackEventHandler() function**: This function handles the common events generated for the BLE Stack.
   a. *CYBLE_EVT_STACK_ON* is received when the Stack is initialized and turned ON.
   b. *CYBLE_EVT_GAP_DEVICE_DISCONNECTED* will occur when the BLE connection gets disconnected and its starts to advertise again.
   c. CYBLE_EVT_GATTC_DISCOVERY_COMPLETE: will occur when device completes the discovery of the services of the peer server. It checks for the DIS service on the peer server and enables the UART commands to read the appropriate characteristics.
3. **HandleLeds()** function: This function is called in the while(1) in main().The function TURNs ON or TURN OFF the LEDS based on the current state of the device
4. **CY_ISR(DIS_Interrupt)** function: This function is used to handle the interrupt routine when SW2 is pressed. In this function variable `discoverServer=1` to start discover the services of the peer server.

Figure 2(b) Client and Server-Central


StackEventHandler()
Main() Function
Event
STACK ON
SCAN PROGRESS RESULT
SCAN STARTED OR STOPPED
START SCANNING
SCAN RESPONSE matches with MANUFACTURER SPECIFIC DATA?
deviceDetected= 1?
YES
YES
O
O
Set deviceDetected=1 & Stop scanning
Send Connection request to the peer device
Return from StackEventHandler
START
-Initialize BLE
-Register Ble StackEventHandler
-Start UART
-Register TpsServiceAppEventHandler
Process Events
Received command from UART?
NO
YES
Execute commands

5. **main() function**: This is the function which performs the initialization of the BLE stack, executes the necessary routines to process the BLE events and maintain the connection. In the initial section of the *main()* function, the API *CyBle_Start(StackEventHandler)* is called to start the BLE Component and register a callback to the stack event handler. The API CyBle_TpsRegisterAttrCallback (TpsServiceAppEventHandler) is used to register the callback to the TPS service related handler. Note that the callback function can assume any name – in this project, we used StackEventHandler. Once the system is initialized, *main()* function continuously operates in a *while(1)* loop executing *CyBle_ProcessEvents()* and executes UART commands. CyBle_ProcessEvents processes the events received by the BLE stack and enables application layer to use them and take the appropriate action. Commands from UART terminal
   a. ‘e’-> enable the notifications for Tps service.
   b. ‘d’-> to disable the notifications for Tps service.

   c. ‘r’-> to read the Tx power level characteristic.
6. **StackEventHandler() function**: This function handles the common events generated for the BLE Stack.
   a. *CYBLE_EVT_STACK_ON* is received when the Stack is initialized and turned ON.
   b. *CYBLE_EVT_GAP_DEVICE_DISCONNECTED* will occur when the BLE connection gets disconnected and its starts to scan again.
   c. CYBLE_EVT_GATTC_DISCOVERY_COMPLETE: will occur when device completes the discovery of the services of the peer server. It checks for the TPS(Tx power level service) service on the peer server

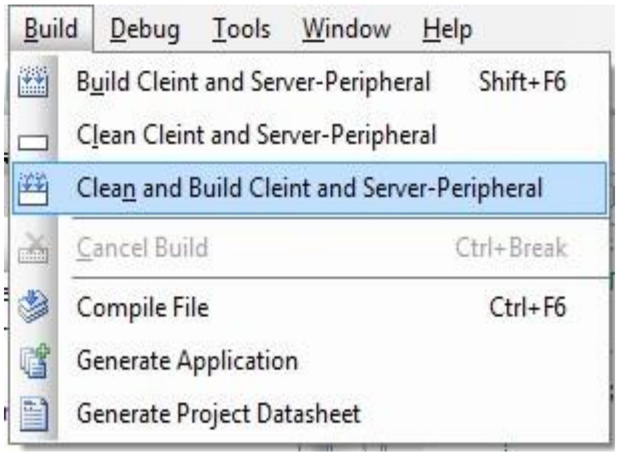
Build and Program the PSOC4 BLE pioneer kit

This section shows how to build the project and program the PSoC 4 BLE device. If you are using a development kit with a built-in programmer (BLE Pioneer Kit, for example), connect the BLE Pioneer Baseboard to your computer using the USB Standard-A to Mini-B cable. .For other kits, refer to the kit user guide.

If you are developing on your own hardware, you need a hardware debugger, for example, a Cypress CY8CKIT-002 MiniProg3.

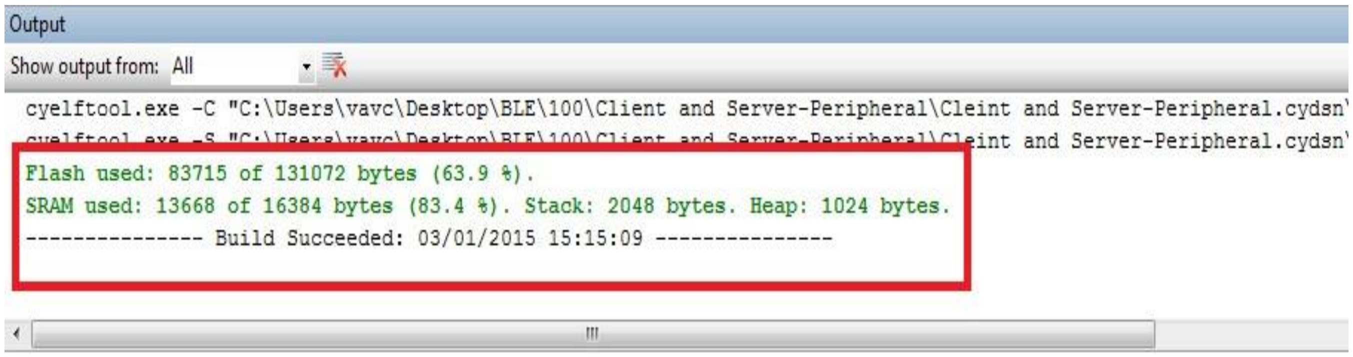
1. On PSoC Creator, select **Build** > **Clean and Build Client and Server- Peripheral**, as shown in Figure 3.

Figure 3 . Build Project


Build
Debug
Tools
Window
Help
Build Cleint and Server-Peripheral Shift+F6
Clean Cleint and Server-Peripheral
Clean and Build Cleint and Server-Peripheral
Cancel Build Ctrl+Break
Compile File Ctrl+F6
Generate Application
Generate Project Datasheet

2. On successful build, total flash and SRAM usage is reported, as shown in Figure 4.

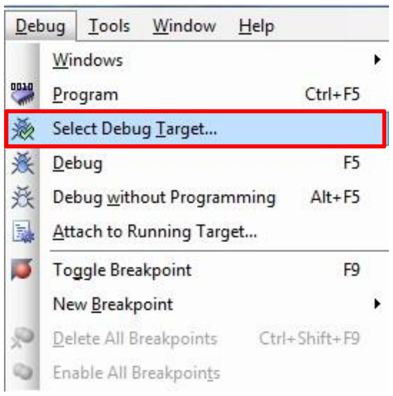
Figure 4. Build Succeeded



```
cyelftool.exe -C "C:\Users\vavc\Desktop\BLE\100\Client and Server-Peripheral\Cleint and Server-Peripheral.cydsn'
cyelftool.exe -S "C:\Users\vavc\Desktop\BLE\100\Client and Server-Peripheral\Cleint and Server-Peripheral.cydsn'
Flash used: 83715 of 131072 bytes (63.9 %).
SRAM used: 13668 of 16384 bytes (83.4 %). Stack: 2048 bytes. Heap: 1024 bytes.
--------------- Build Succeeded: 03/01/2015 15:15:09 ---------------
```

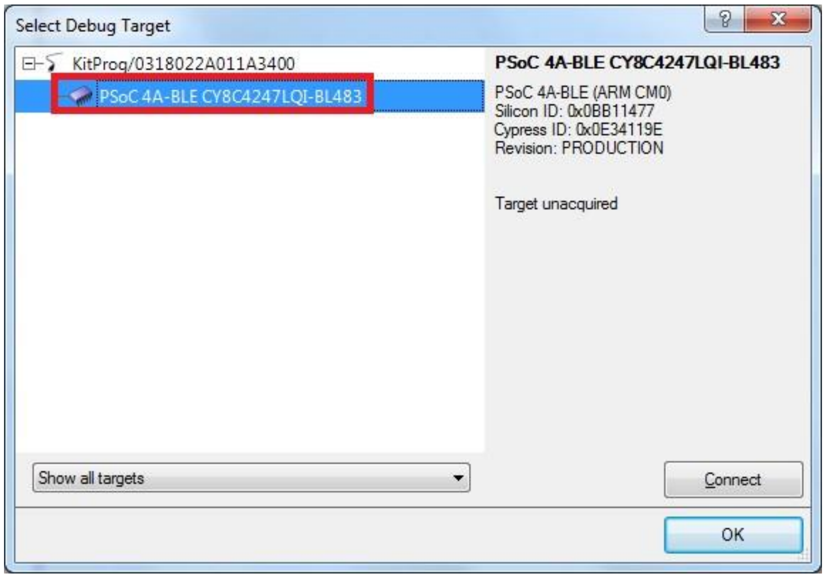
3. Select **Debug** > **Select Debug Target**, as shown in Figure 5.

Figure 5. Selecting Debug Target



4. In the Select Debug Target dialog box, click Port Acquire, and then click Connect, as shown in Figure 6. Click OK to close the dialog box.

Figure 6. Connecting to a Device


Select Debug Target

KitProg/0318022A011A3400
PSoC 4A-BLE CY8C4247LQI-BL483

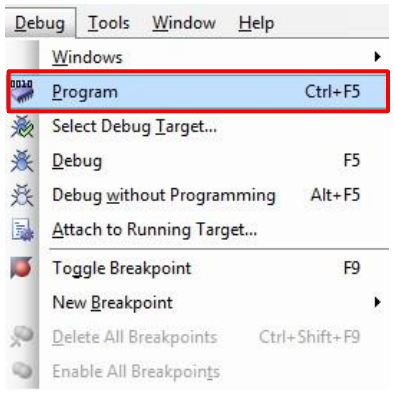
PSoC 4A-BLE CY8C4247LQI-BL483
PSoC 4A-BLE (ARM CM0)
Silicon ID: 0x0BB11477
Cypress ID: 0x0E34119E
Revision: PRODUCTION

Target unacquired

If you are using your own hardware, make sure the Port Setting configuration under Select Debug Target window for your programming hardware is configured as per your setup.

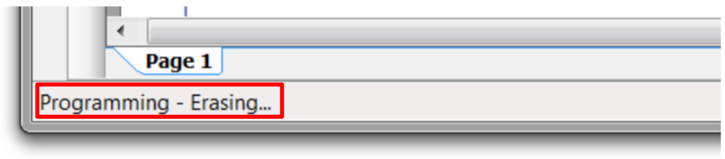
5. Select **Debug** > **Program** to program the device with the project, as shown in Figure 7.

Figure 7. Programming the Device



You can view the programming status on the PSoC Creator status bar (lower-left corner of the window), as shown in Figure 8.

Figure 8. Programming Status



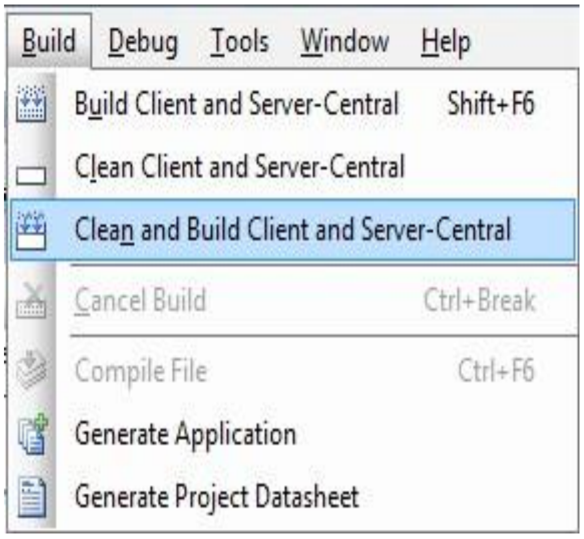
Build and Program the Dongle:

This section shows how to build the project and program the dongle. Connect the dongle directly to the USB port.

.

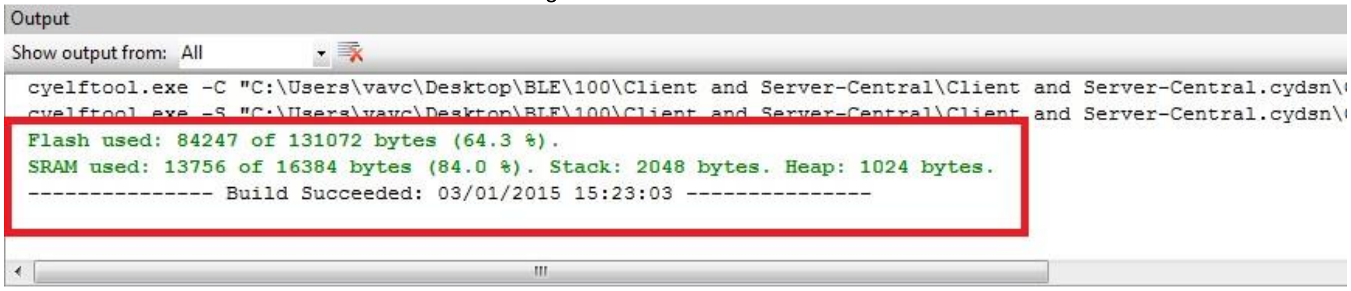
1. On PSoC Creator, select **Build** > **Clean and Build Client and Server- Central**, as shown in Figure 3

Figure 9.Build project


Build
Debug
Tools
Window
Help
Build Client and Server-Central Shift+F6
Clean Client and Server-Central
Clean and Build Client and Server-Central
Cancel Build Ctrl+Break
Compile File Ctrl+F6
Generate Application
Generate Project Datasheet

2. On successful build, total flash and SRAM usage is reported, as shown in Figure 4

Figure 10. Build Succeeded



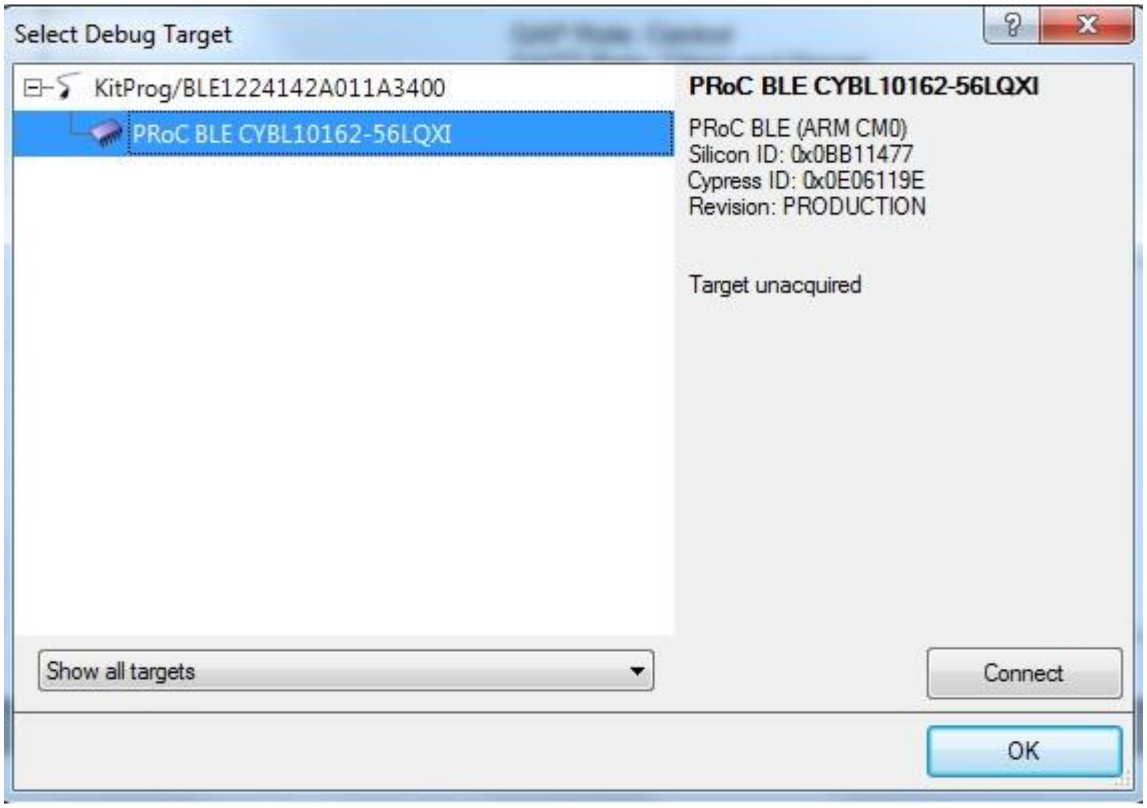
Output

Show output from: All

```
cyelftool.exe -C "C:\Users\vavc\Desktop\BLE\100\Client and Server-Central\Client and Server-Central.cydsn\
cyelftool.exe -S "C:\Users\vavc\Desktop\BLE\100\Client and Server-Central\Client and Server-Central.cydsn\
Flash used: 84247 of 131072 bytes (64.3 %).
SRAM used: 13756 of 16384 bytes (84.0 %). Stack: 2048 bytes. Heap: 1024 bytes.
--------------- Build Succeeded: 03/01/2015 15:23:03 ---------------
```

3. Select **Debug** > **Select Debug Target**, as shown in Figure 5.
4. In the Select Debug Target dialog box, click Port Acquire, and then click Connect, as shown in Figure 6. Click OK to close the dialog box.

Figure 11. . Connecting to a dongle



Select Debug Target

KitProg/BLE1224142A011A3400
PRoC BLE CYBL10162-56LQXI

PRoC BLE CYBL10162-56LQXI
PRoC BLE (ARM CM0)
Silicon ID: 0x0BB11477
Cypress ID: 0x0E06119E
Revision: PRODUCTION

Target unacquired

Show all targets

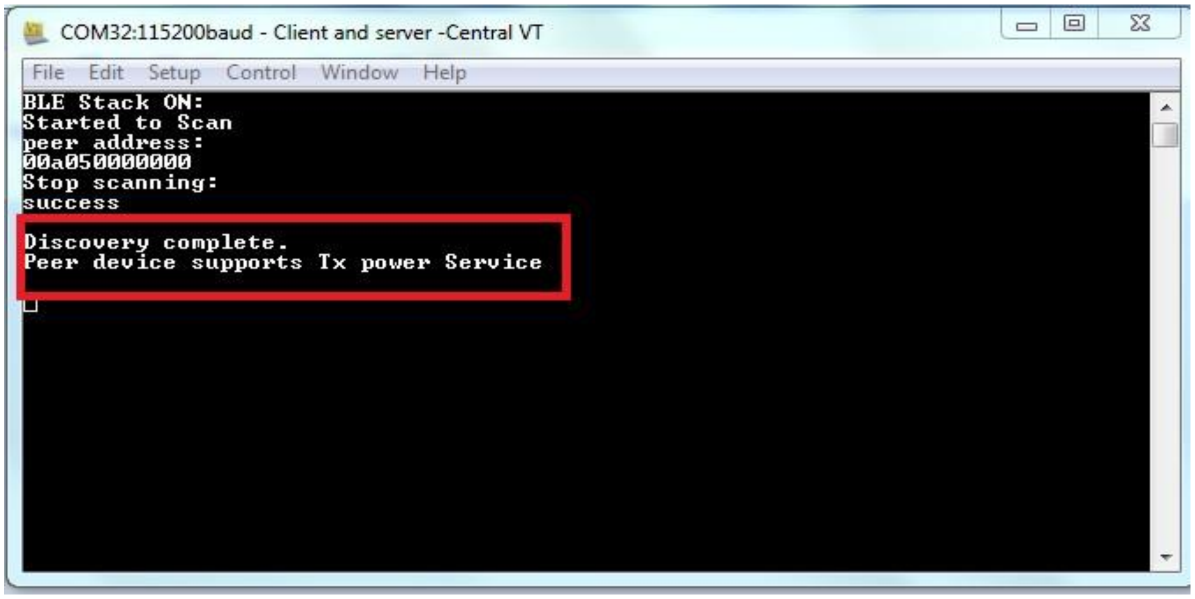
If you are using your own hardware, make sure the Port Setting configuration under Select Debug Target window for your programming hardware is configured as per your setup.

5. Program the dongle as shown in Figure 7.

Steps to Test the project:

1. Open UART Terminal and the select the COM port (related to PSOC4 BLE device) and use the following settings
   I. Baud rate:115200
   II. Data rate: 8 bit
   III. Parity: none
   IV. Stop: 1 bit
   V. Flow control : none
2. With the same settings mentioned above, open another UART terminal and select the COM port (related to the Dongle).
3. After programming the Psoc4 BLE it starts to advertise.
4. Program the dongle and it starts to scan. Among the received scan response packet it searches for the manufacturer specific data. If the scan response data matches then it will connect automatically.
5. After the connection is established the dongle starts to discover the services. It searches for the Tx power level service .The following things will get displayed on the UART terminal related to dongle after the discovery is completed(If the peer device supports Tx power level service).

Figure 12. UART terminal (related to dongle)

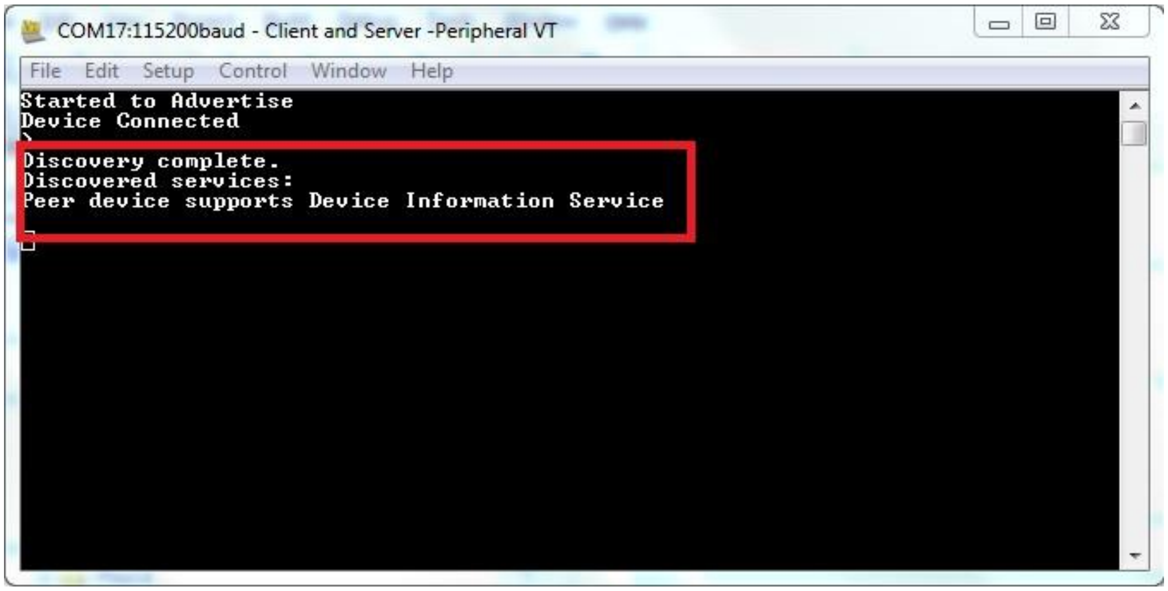

COM32:115200baud - Client and server -Central VT
File Edit Setup Control Window Help

```
BLE Stack ON:
Started to Scan
peer address:
00a050000000
Stop scanning:
success

Discovery complete.
Peer device supports Tx power Service
```

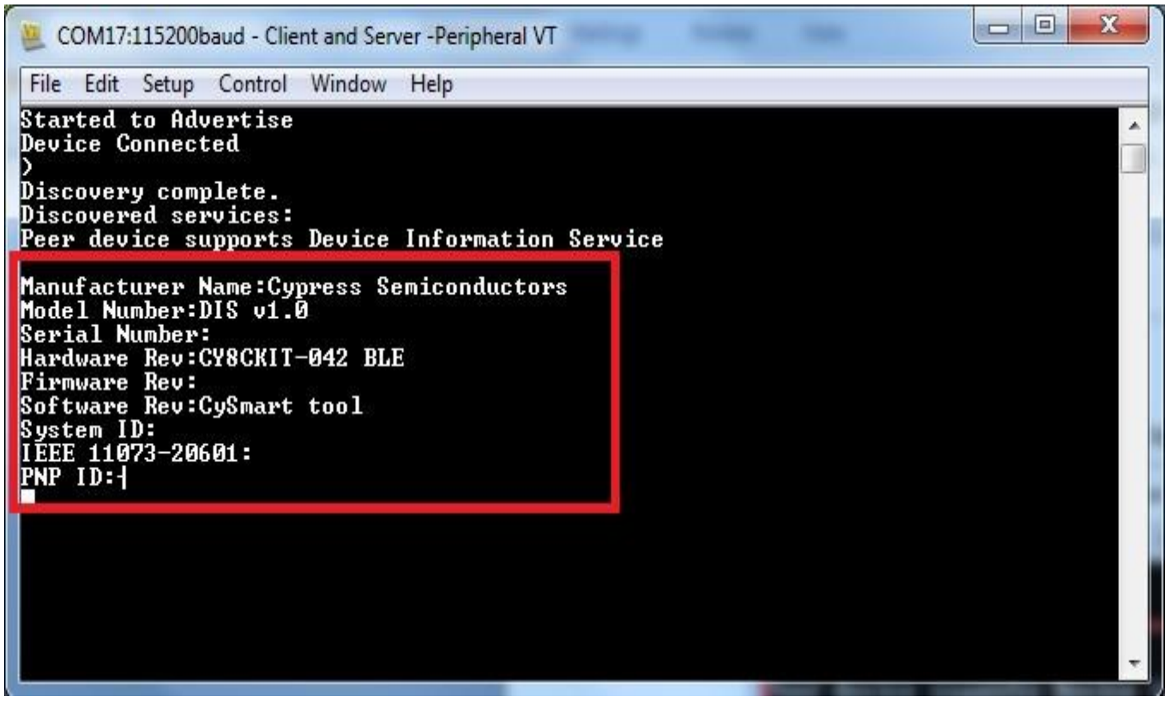
6. Now press SW2 on the pioneer kit to start discovery of services of peer server. It searches for the DIS service and display the following things on UART terminal related to PSOC4 BLE(as shown in Figure 6) after discovery is completed(if peer device supports DIS service)

Figure 13.UART terminal (related to PSOC4 BLE)


COM17:115200baud - Client and Server -Peripheral VT
File Edit Setup Control Window Help

```
Started to Advertise
Device Connected
>
Discovery complete.
Discovered services:
Peer device supports Device Information Service
```

7. From the UART terminal of PSOC4 BLE send ‘1’ to ‘9’ to read the characteristics related to DIS. The following messages will be displayed on UART terminal related to PSOC4 BLE after reading the characteristics.


COM17:115200baud - Client and Server -Peripheral VT
File Edit Setup Control Window Help

```
Started to Advertise
Device Connected
>
Discovery complete.
Discovered services:
Peer device supports Device Information Service

Manufacturer Name:Cypress Semiconductors
Model Number:DIS v1.0
Serial Number:
Hardware Rev:CY8CKIT-042 BLE
Firmware Rev:
Software Rev:CySmart tool
System ID:
IEEE 11073-20601:
PNP ID:
```

8. From the UART terminal of PSOC4 BLE send the command ‘d’ to decrease the Tx power level. The current power level after decrement will be displayed on UART terminal. .
9. From the UART terminal of dongle enable the notifications by sending ‘e’.
10. Send ‘r’ from UART terminal of dongle to read the characteristic TX power level.

11. To send the notifications from PSOC4 side, send the command ‘n’ from the UART terminal of PSCO4 BLE.

Related Documents

Table 1 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component / user module datasheets.

Table 1. Related Documents

Document	Title	Comment
AN91267	Getting Started with PSoC 4 BLE	Provides an introduction to PSoC 4 BLE device that integrates a Bluetooth Low Energy radio system along with programmable analog and digital resources.
AN91445	Antenna Design Guide	Provides guidelines on how to design an antenna for BLE applications.