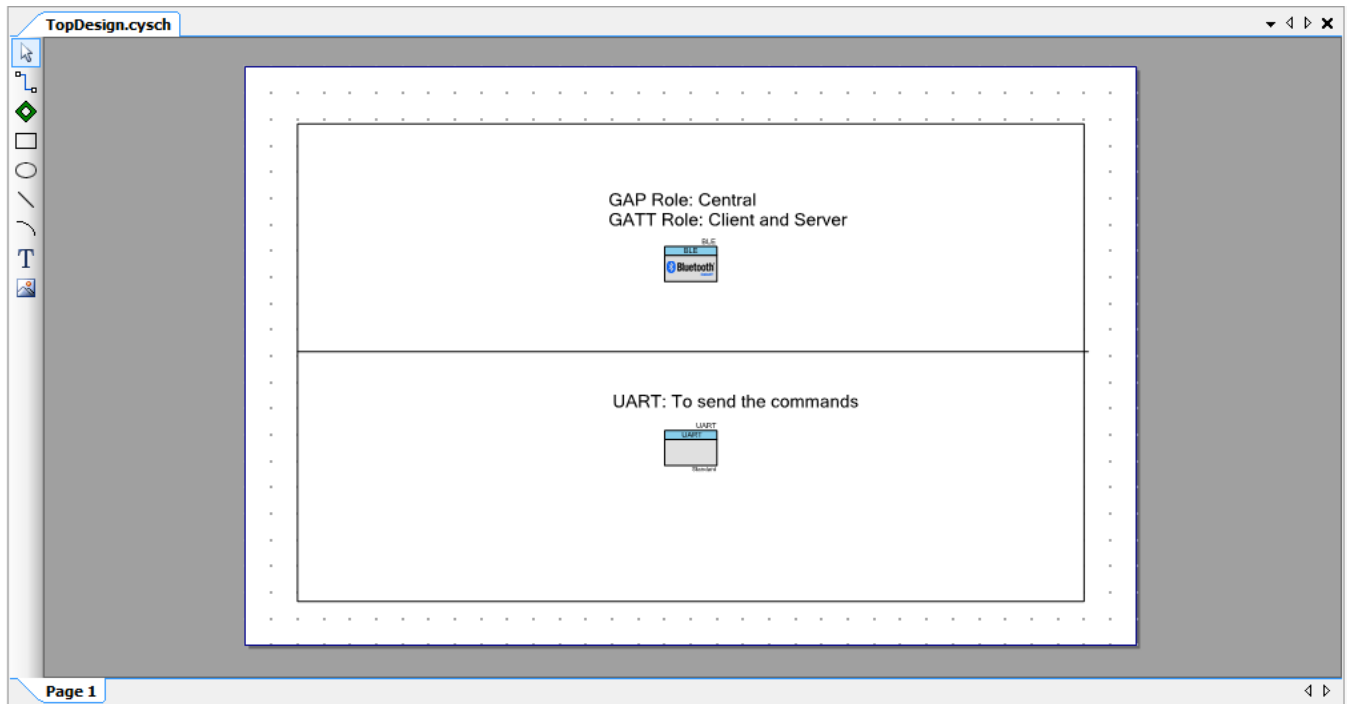


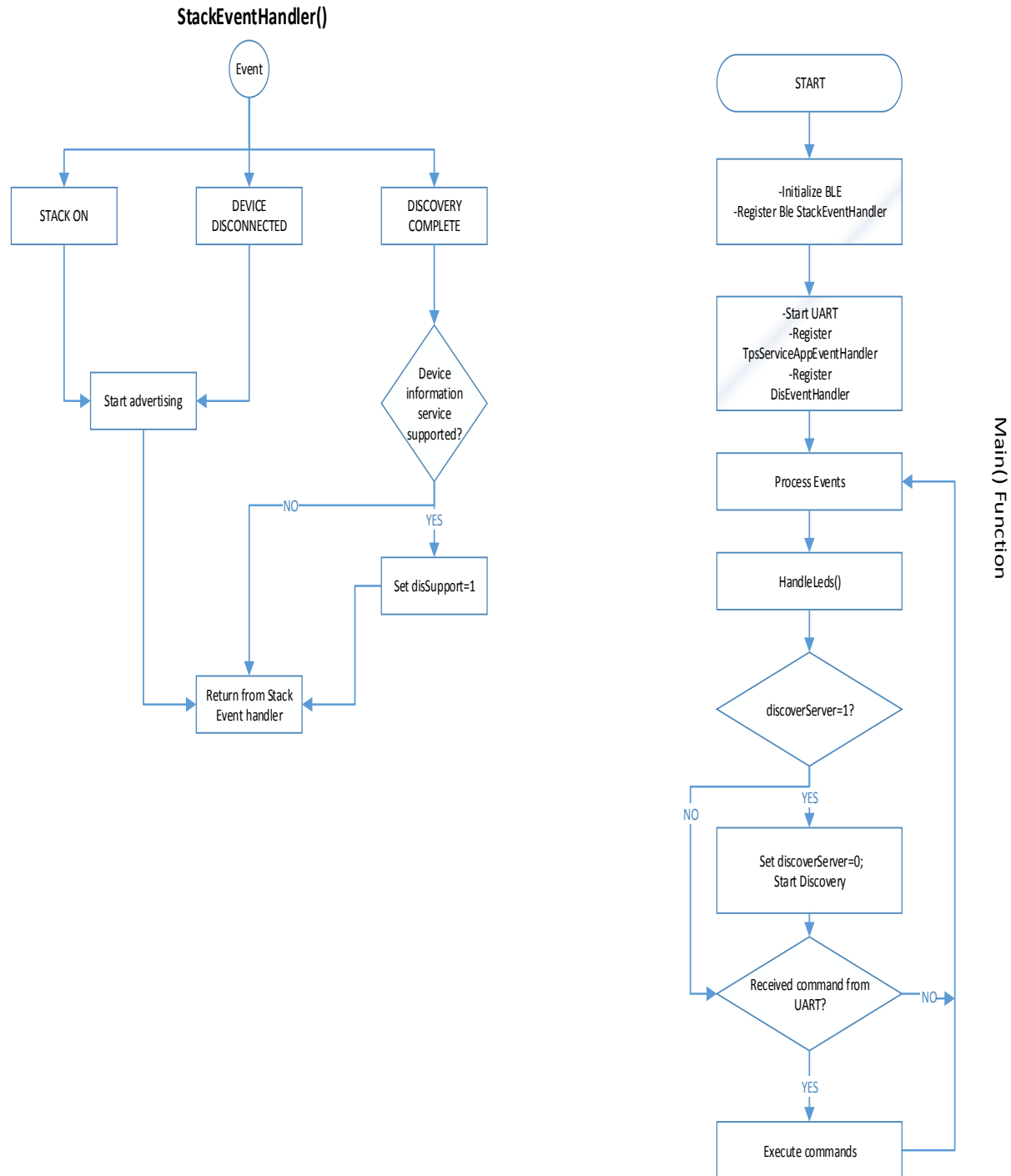
This example demonstrates the use of PSoC4 BLE as both BLE Client and Server.

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



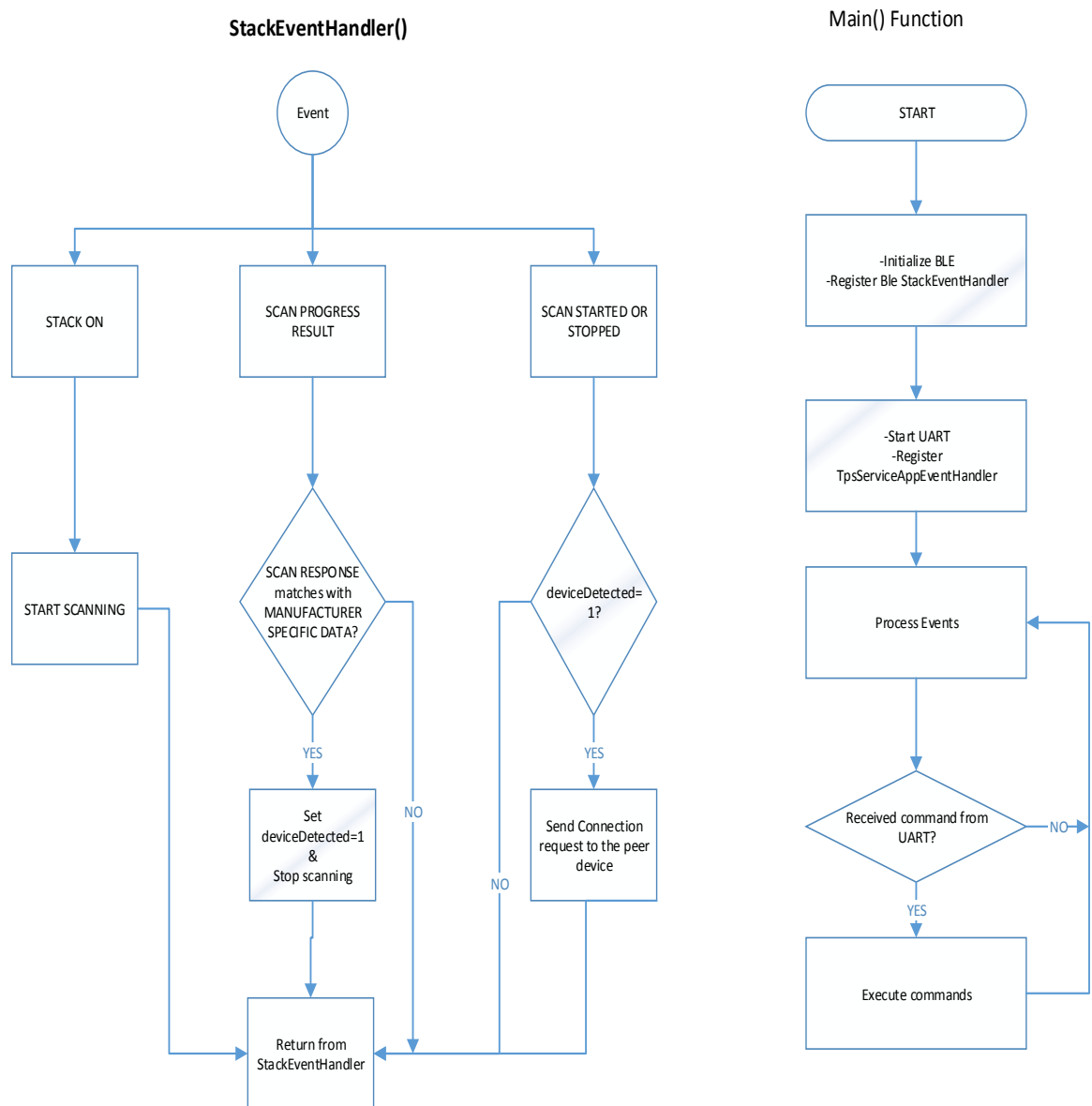
## Operation

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



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

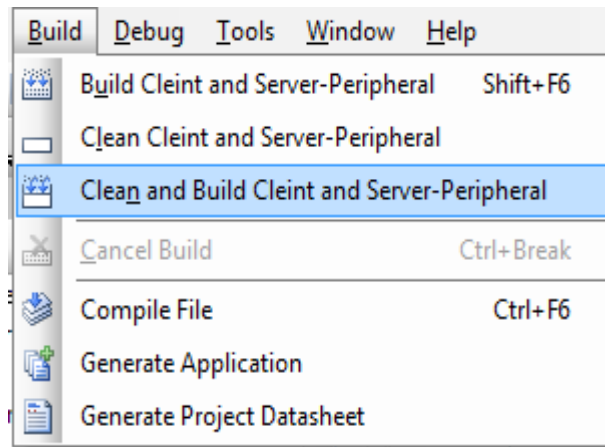
## Build and Program the PSoC4 BLE pioneer kit

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](#).

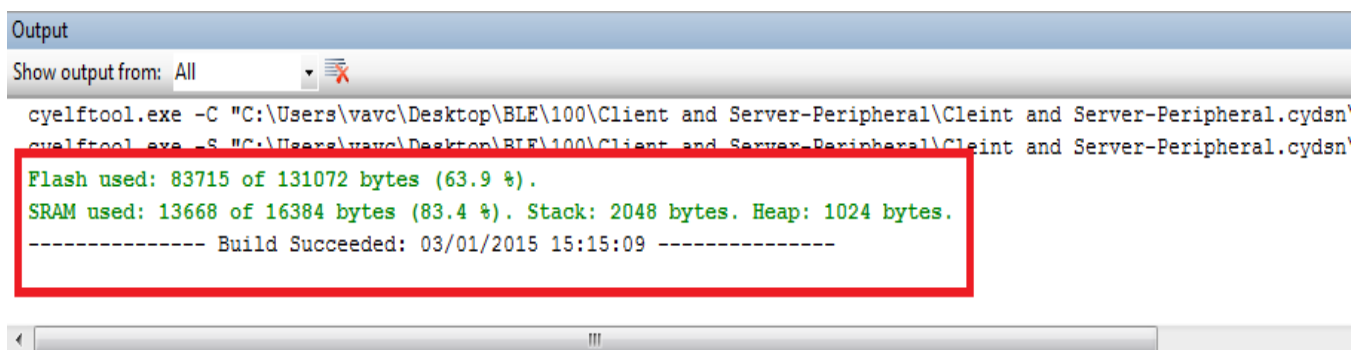
1. On PSoC Creator, select **Build > Clean and Build Client and Server- Peripheral**, as shown in [Figure 3](#).

Figure 3 . Build Project



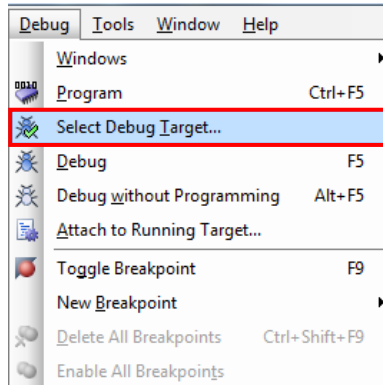
2. On successful build, total flash and SRAM usage is reported, as shown in [Figure 4](#).

Figure 4. Build Succeeded



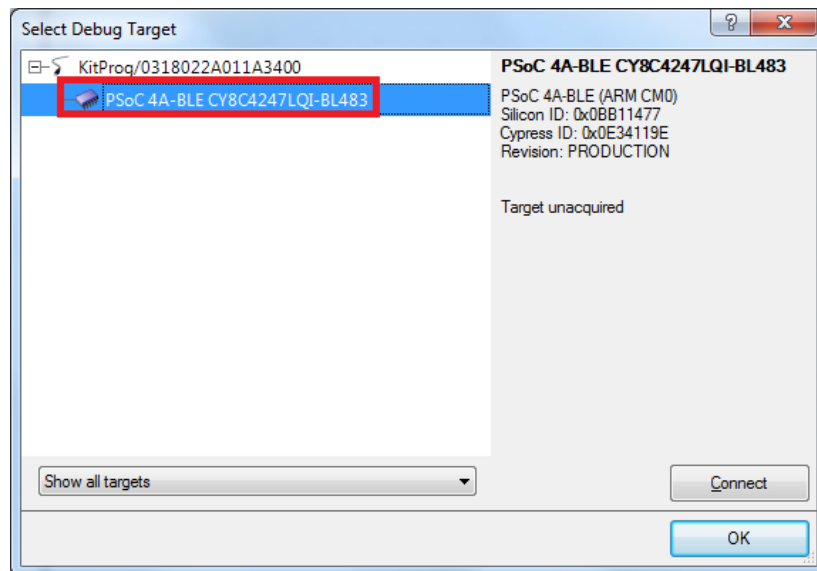
3. Select **Debug > Select Debug Target**, as shown in Figure 5.

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.

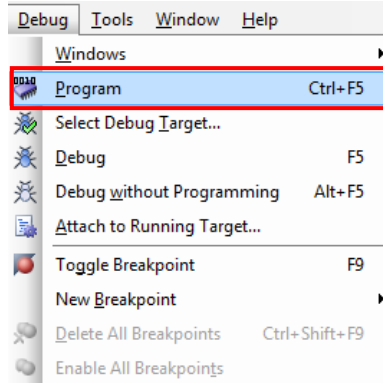
Figure 6. Connecting to a Device



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.

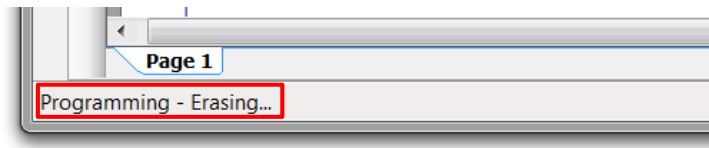
5. Select **Debug > Program** to program the device with the project, as shown in [Figure 7](#).

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

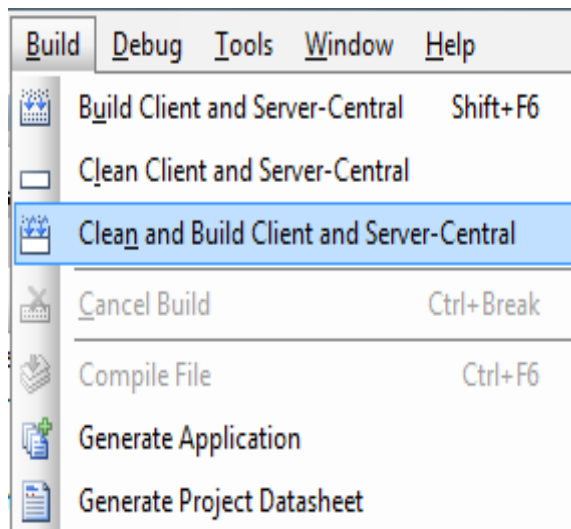


## Build and Program the Dongle:

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





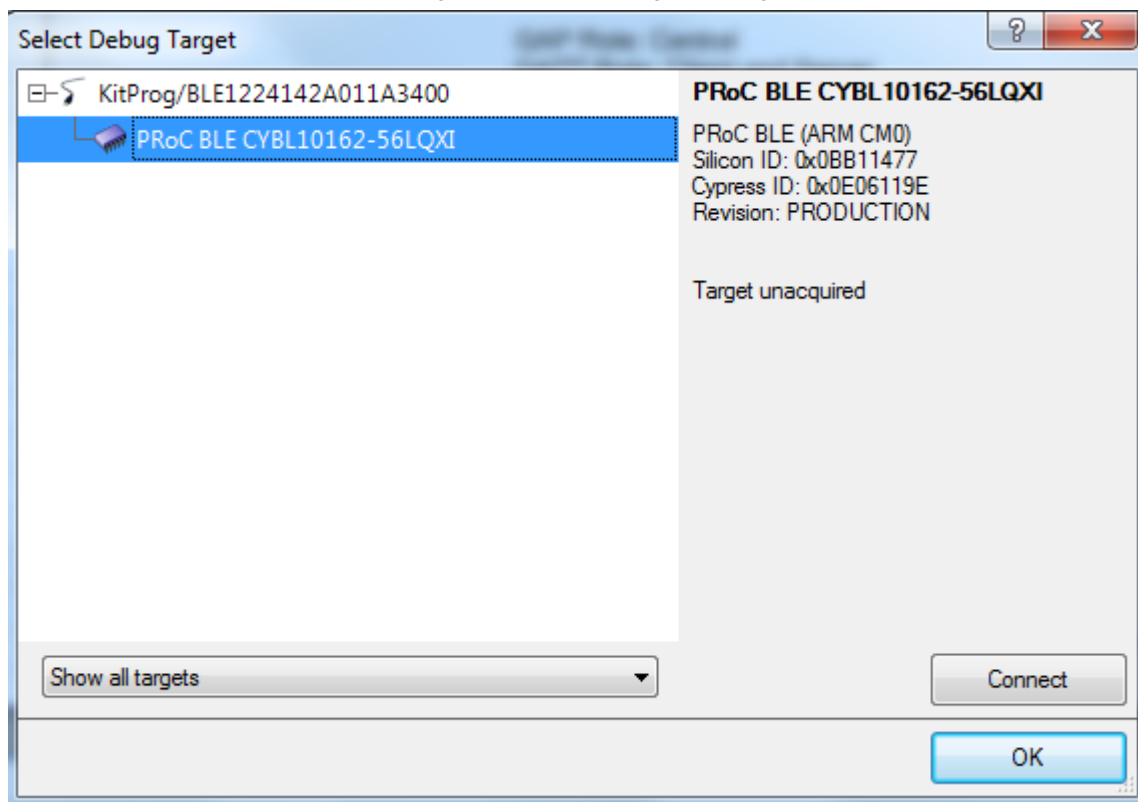
- On successful build, total flash and SRAM usage is reported, as shown in [Figure 4](#)

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 -----
```

- Select **Debug > Select Debug Target**, as shown in [Figure 5](#).
- 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



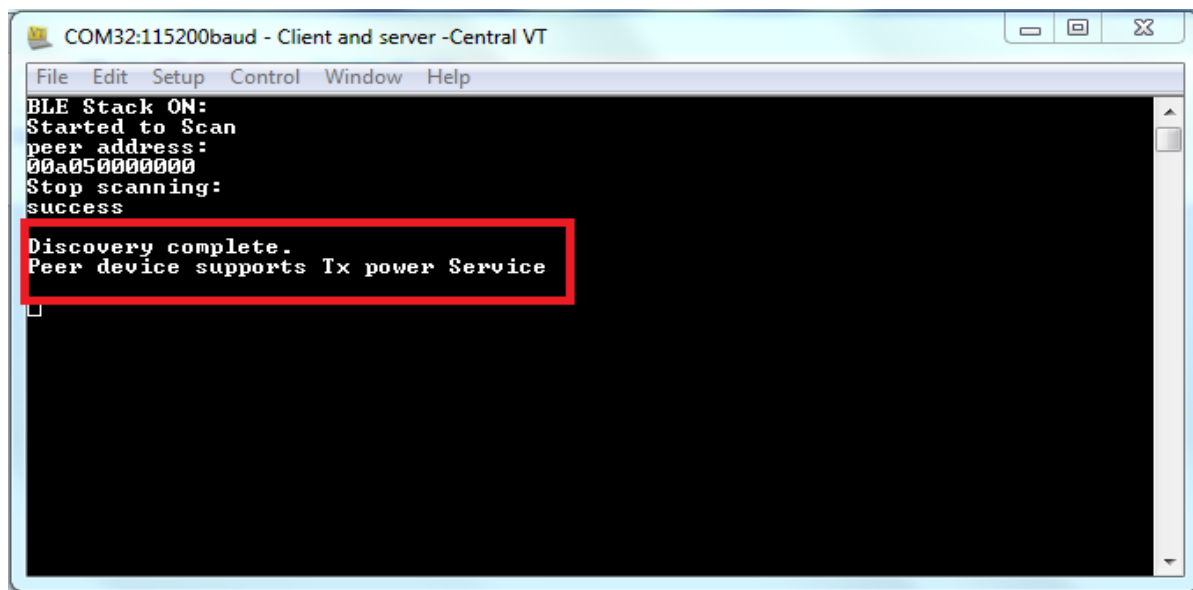
If you are using your own hardware, make sure the Port Setting configuration under Select Debug Target window for your programming hardware is configured as per your setup.

- Program the dongle as shown in [Figure 7](#).

## Steps to Test the project:

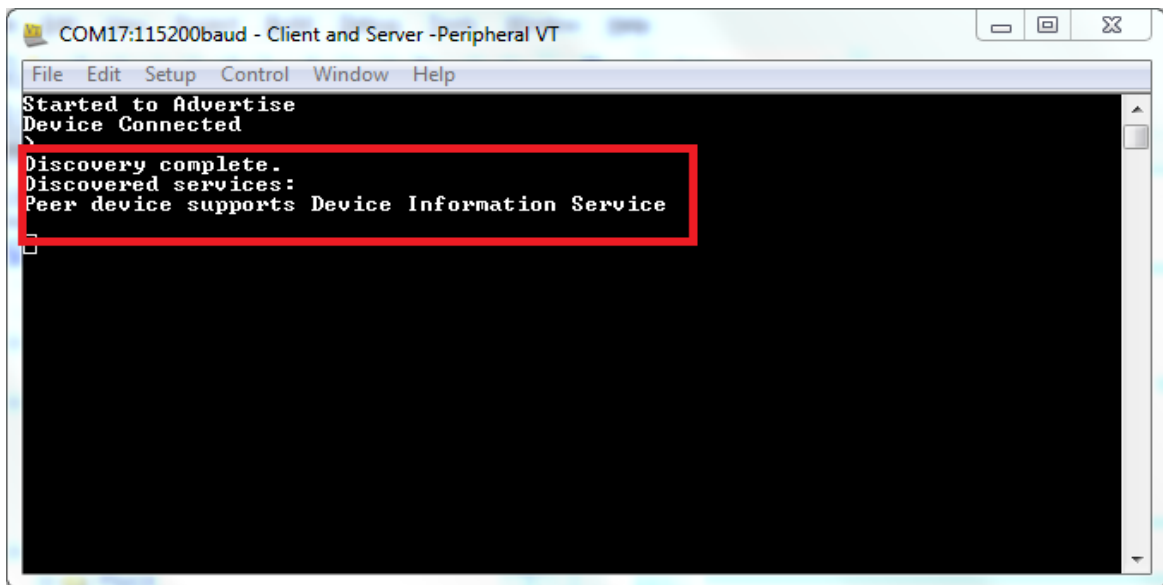
1. Open UART Terminal and select the COM port (related to PSoC4 BLE device) and use the following settings
  - I. Baud rate: 115200
  - II. Data rate: 8 bit
  - III. Parity: none
  - IV. Stop: 1 bit
  - V. Flow control : none
2. With the same settings mentioned above, open another UART terminal and select 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 the dongle after the discovery is completed (If the peer device supports Tx power level service).

Figure 12. UART terminal (related to dongle)

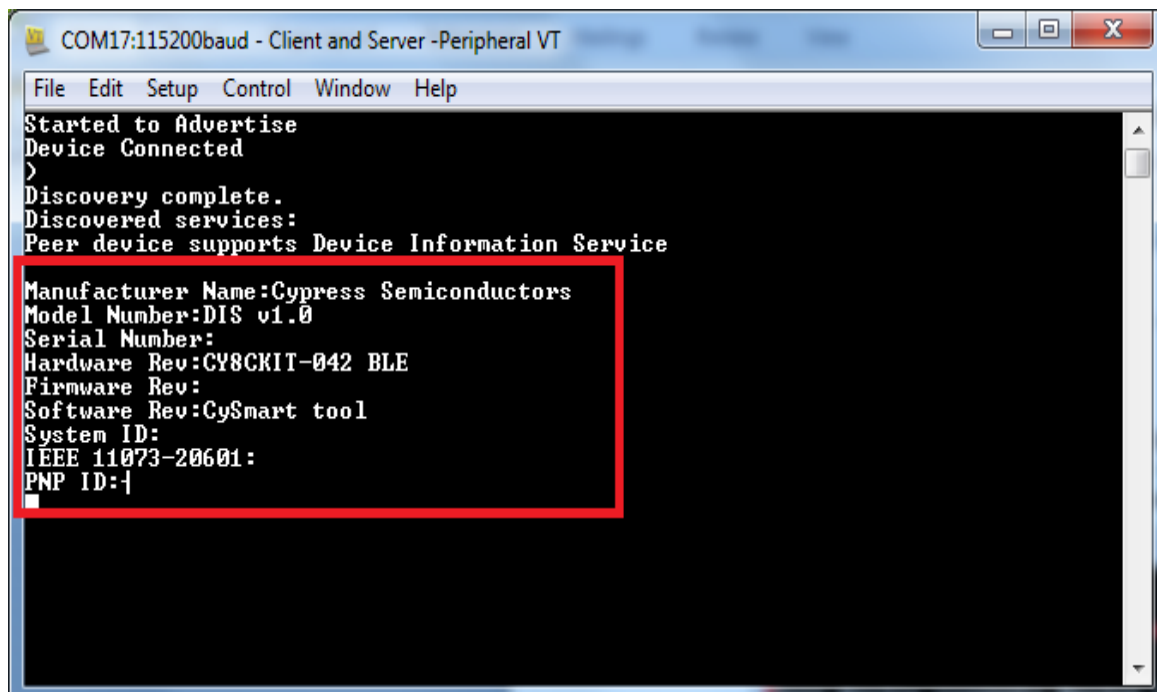


6. Now press SW2 on the Pioneer kit to start discovery of services of peer server. It searches for the DIS service and displays the following things on the UART terminal related to PSoC4 BLE (as shown in [Figure 6](#)) after discovery is completed (if peer device supports DIS service)

Figure 13.UART terminal (related to PSOC4 BLE)



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.



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   |
|-------------------------|---------------------------------|---|
| <a href="#">AN91267</a> | 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. |
| <a href="#">AN91445</a> | Antenna Design Guide            | Provides guidelines on how to design an antenna for BLE applications.   |