PSoC 4 BLE Lab2: Setup a BLE Connection

Group: Gloria Bauman and Kervins Nicolas

Due: September 27th, 2016

Description

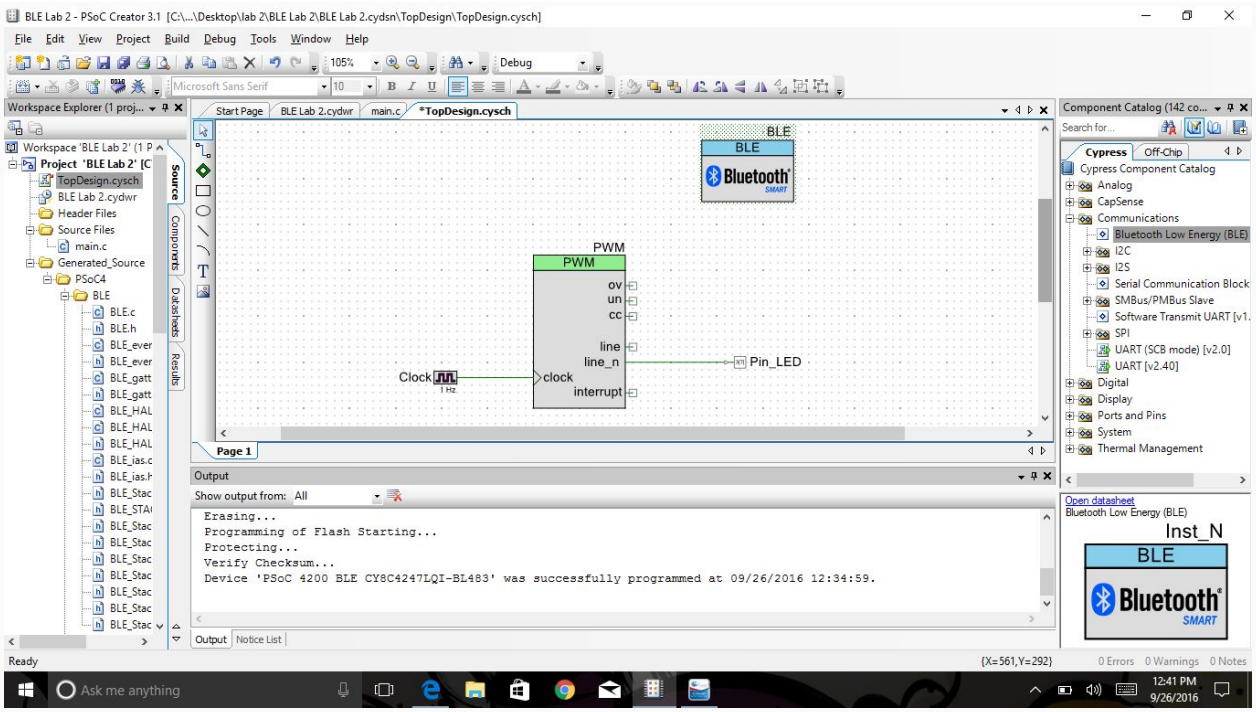
This lab introduces you to the Bluetooth Low Energy feature of PSoC 4 BLE. It helps you create your first BLE application by implementing a BLE Standard Find-Me Profile.

Objectives

1. Learn how to use the BLE Component
2. Implement a standard BLE Find Me Profile with the Immediate Alert Service (IAS)
3. Learn how to use the CySmart BLE Test and Debug Tool to debug BLE designs

Process

1) Under the “TopDesign.cysch” from the “Workspace Explorer” drag and place a “Bluetooth Low Energy” Component from the “Component Catalog” under the communications category.



2) Configure the component to the appropriate parameters and settings.

- Under the “General” Tab
    - Profile: “Find Me”
    - Profile Role: “Find ME Target (GATT Server)”
    - GAP role: “Peripheral”
- Under “Profiles Tab”
    - No Changes
- Under “GAP Settings” Tab
    - “General”
        - Check “Silicon generated “Company assigned” part of the device address”
        - Device name: “BLE Lab 2”
        - Appearance: “Generic Tag”
    - “Peripheral role” - Advertisment Setting
        - Discovery mode: “General”

        - Advertising Type: “Connections undirected advertising”
        - Filter Policy: “ Scan request Any| Connect request Any”
        - Advertising channel maps: “All Channels”
        - Uncheck “Slow advertising interval”
    - “Peripheral role”- Advertisement Package
        - Check “Service UUID”
            - Check “immediate alert”
        - Check “appearance”
    - “Peripheral role”- Scan response packet
        - Check “Local Name”
            - Local name = “complete”
    - “Security”
        - I/O Capabilities: “No input No Output”
        - Bonding requirement: “No Bonding”

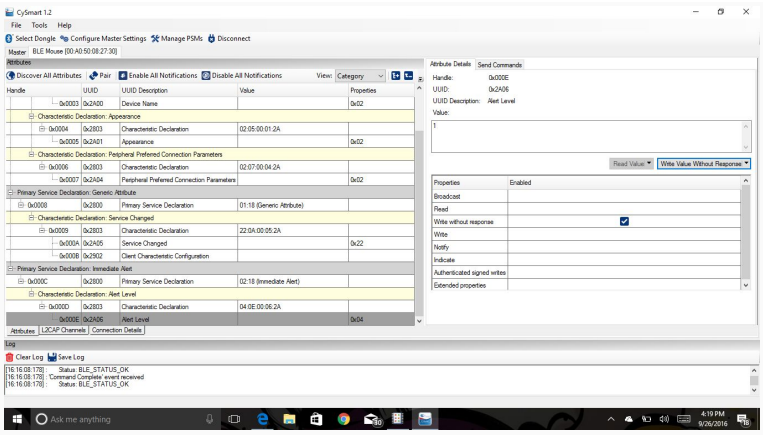
3) After configuring the BLE component build and debug your program.

4) Testing your program. Open “CySmart 1.0” and select “BLE Dongle Target”. Next select “Cypress BLE Dongle (COMxx), and connect.

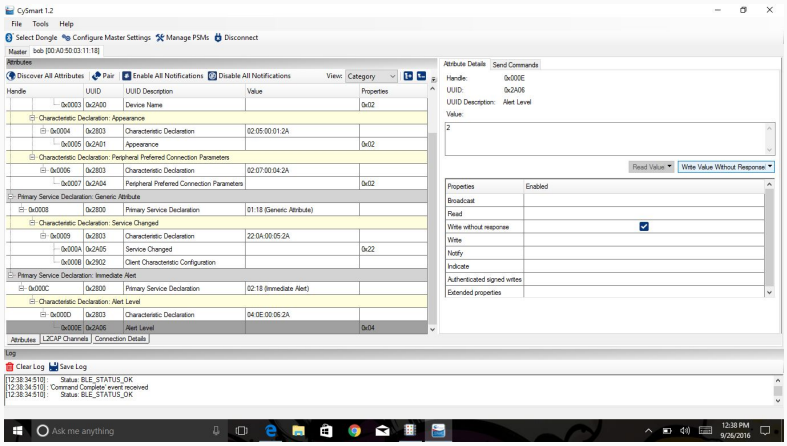
5) Start the scan to find your BLE device. Then click on device name to see “Advertisment data” and “Scan response data” packets, when that is accomplished press “Connect”.

6) Once the tool opens a new tab for the connected device, select “Discover All Attributes.” Next Locate the “Alert Level” attribute for the “Immediate Alert Service.”

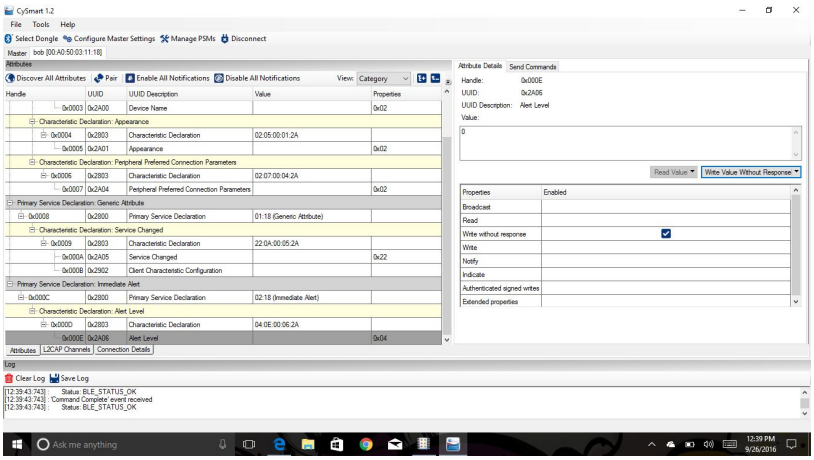
- Write a value of 1 to start the blinking red light



- Value of 2 to keep LED on always

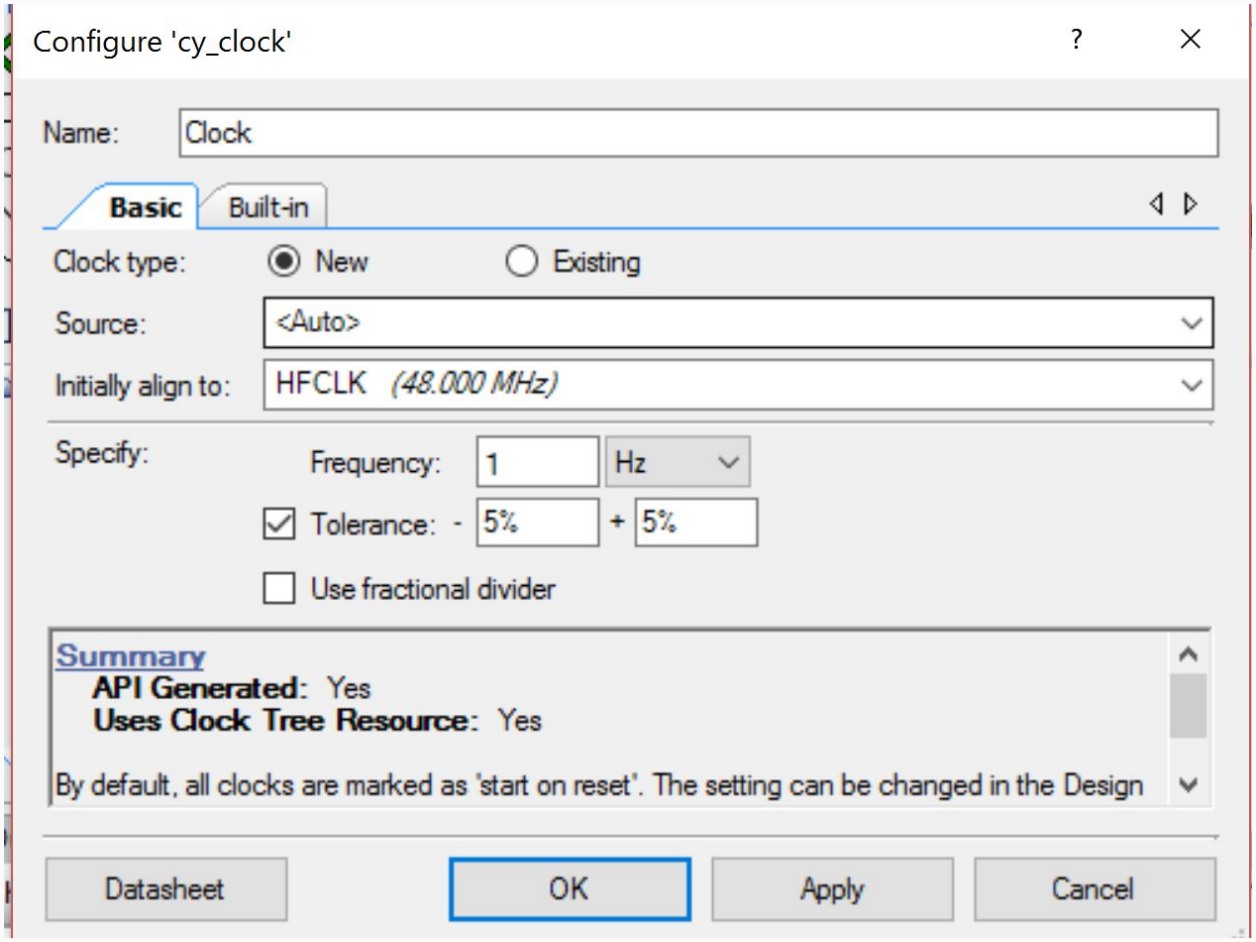


- Value of 0 to turn off LED

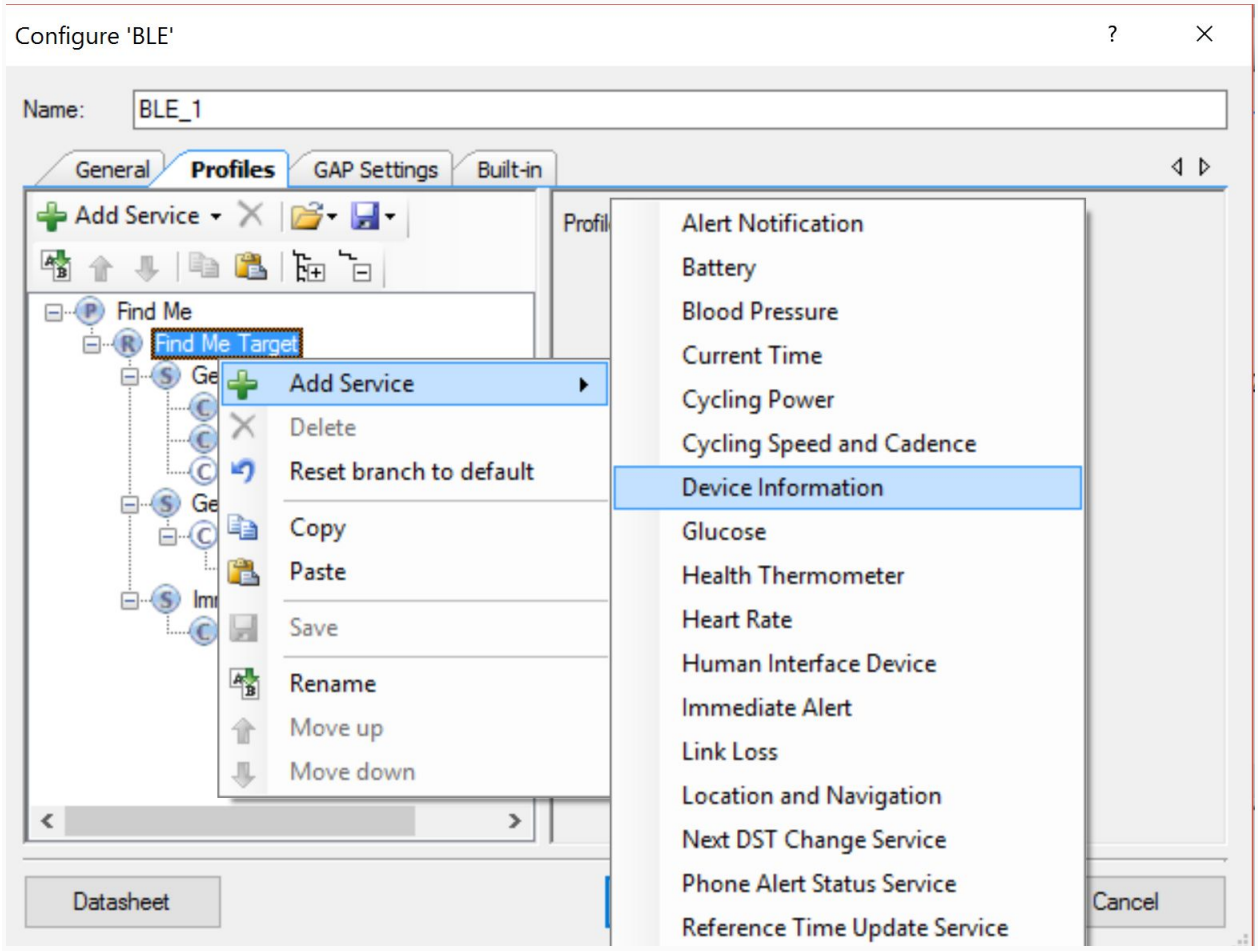


Additional Exercises

1. Configure the PWM Component's Period parameter value to change the LED blink rate to 1-Hz.
    - Under the "TopDesign.cysch" tab select the PWM clock component to configure it.
    - Specify the frequency and change it from 1 kHz to 1 Hz, then click "OK"



2. Add the Device Information Service (DIS) to the Find Me Profile.
   Hint: Additional Services can be added by right-clicking the Find Me Target in the BLE Component
   Configuration Tool, and selecting Add Service.
    - Under the "TopDesign.cysch" tab select the BLE component to configure it.
    - Under the "Profiles" tab, right-click the "Find Me Target" and select "Add Service" then "Device Information."



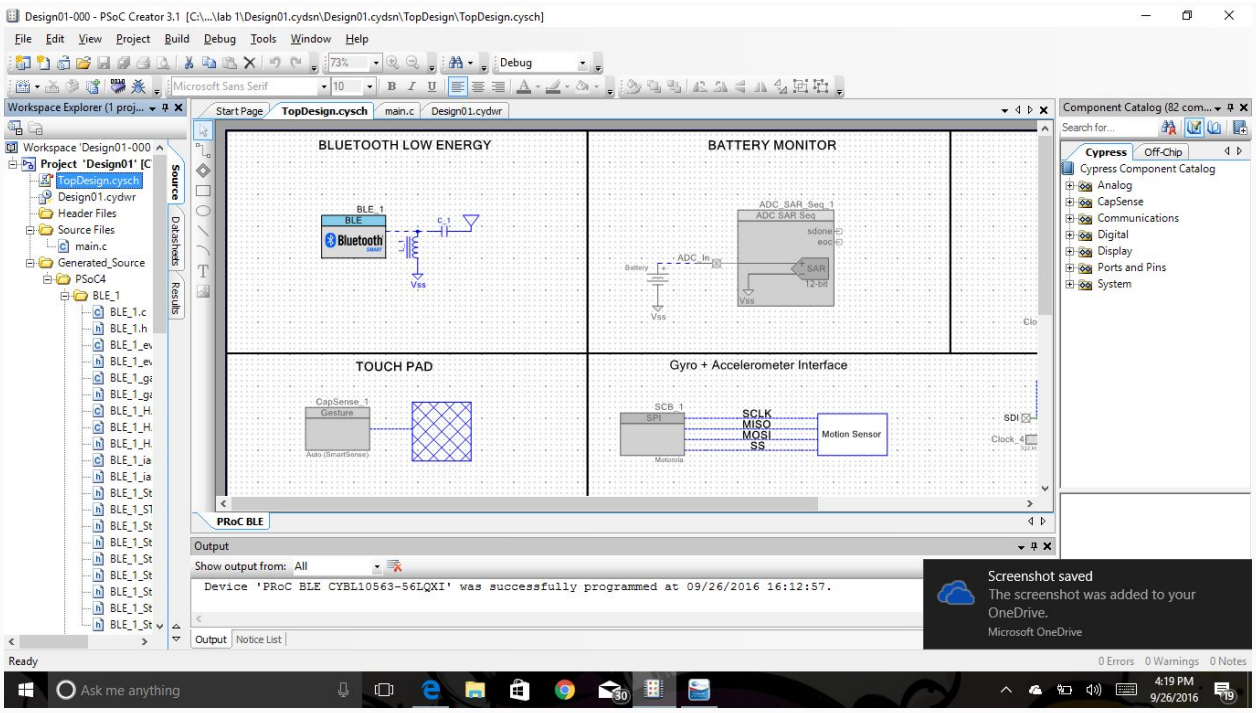
3. Repeat this lab with a PRoC BLE device.

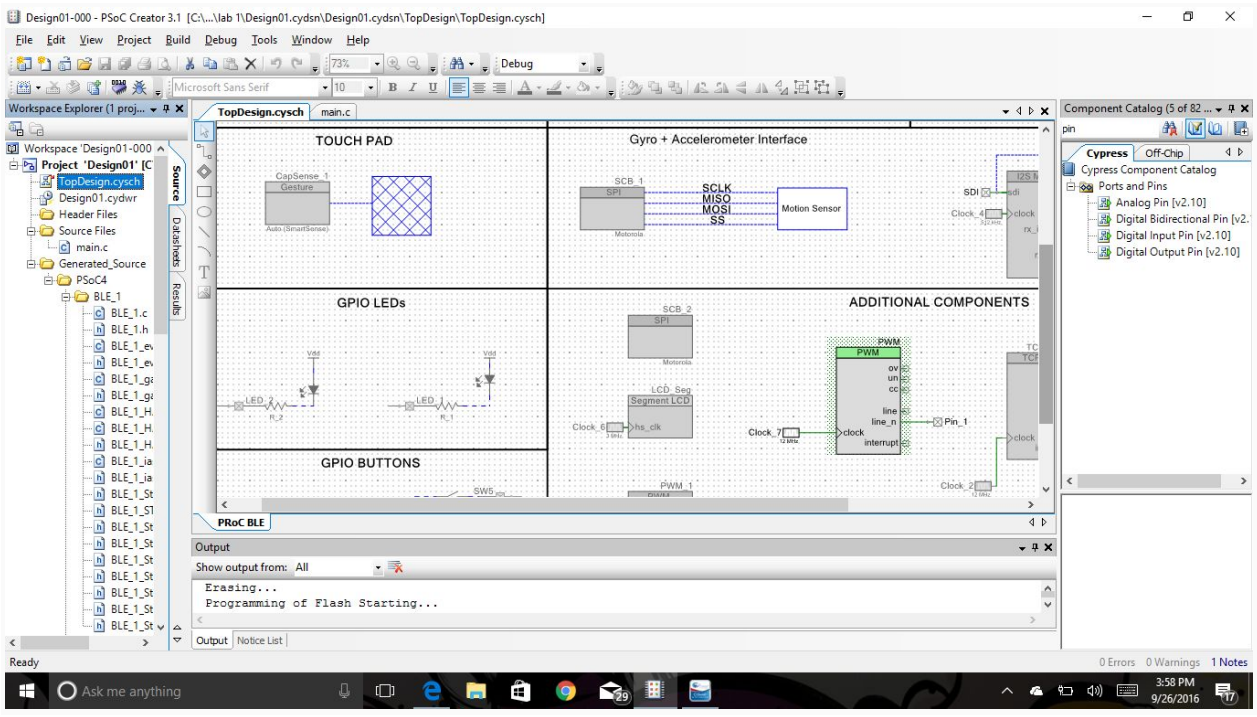
   Hints:

   a. Create a New Project using the PRoC BLE device: CYBL10563-56LQXI.

   - Plug in PRoC BLE device: CYBL10563-56LQXI (Red circuit board)

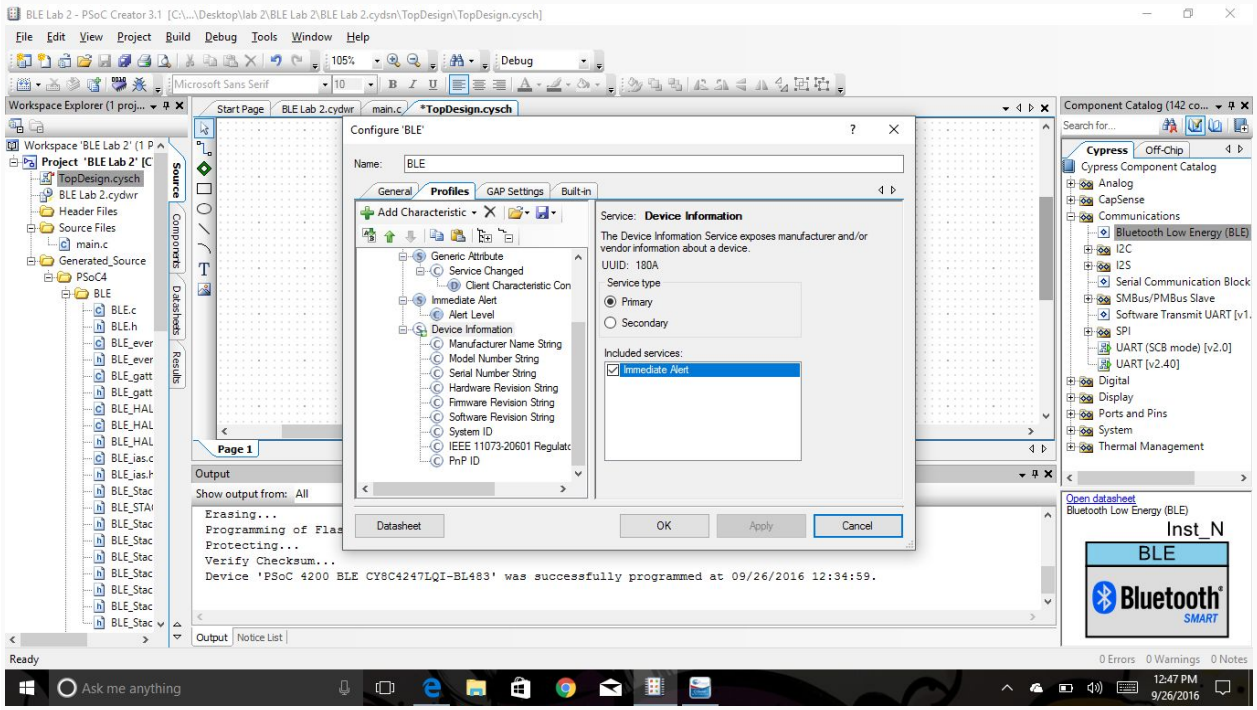
b. Disable the unused Components by right-clicking on the Component and selecting the Disable option.

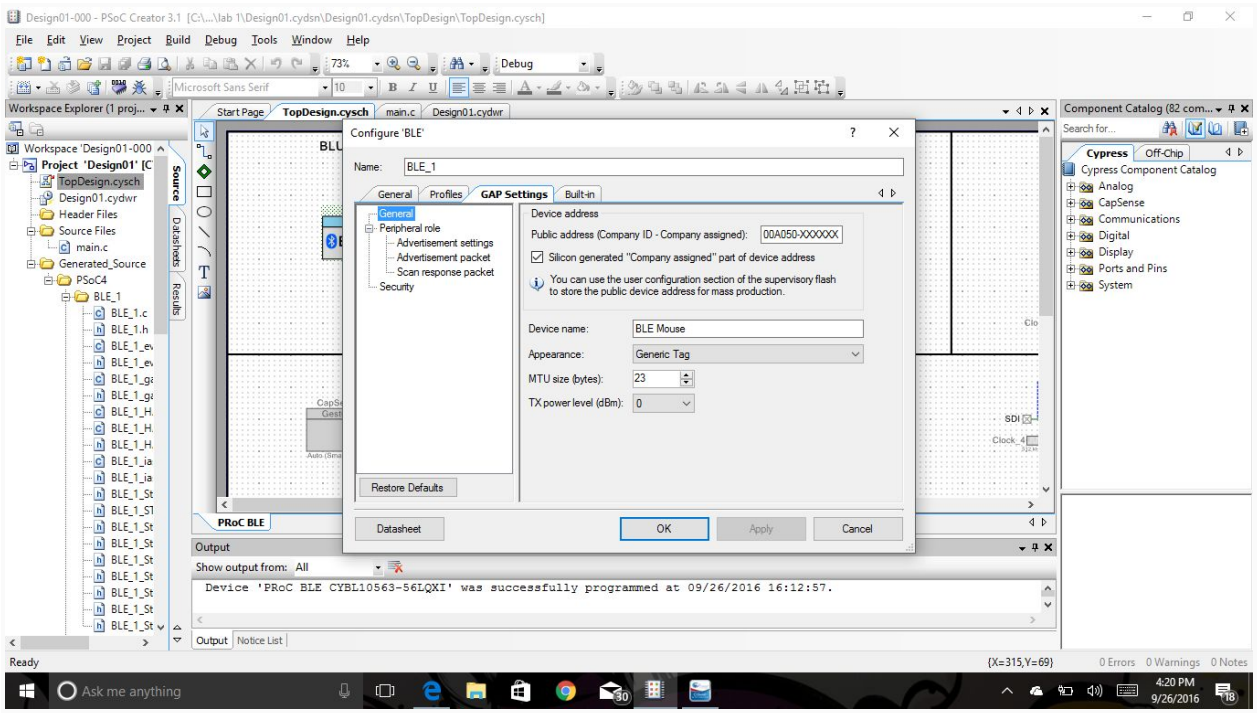


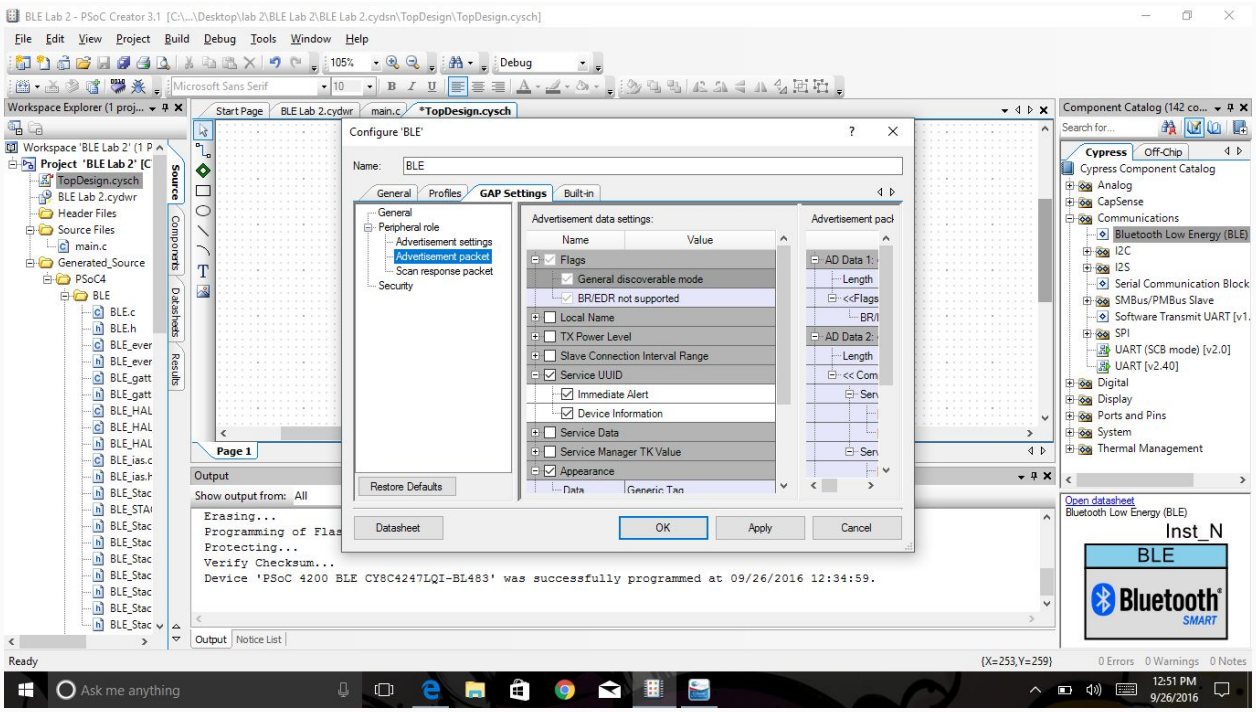


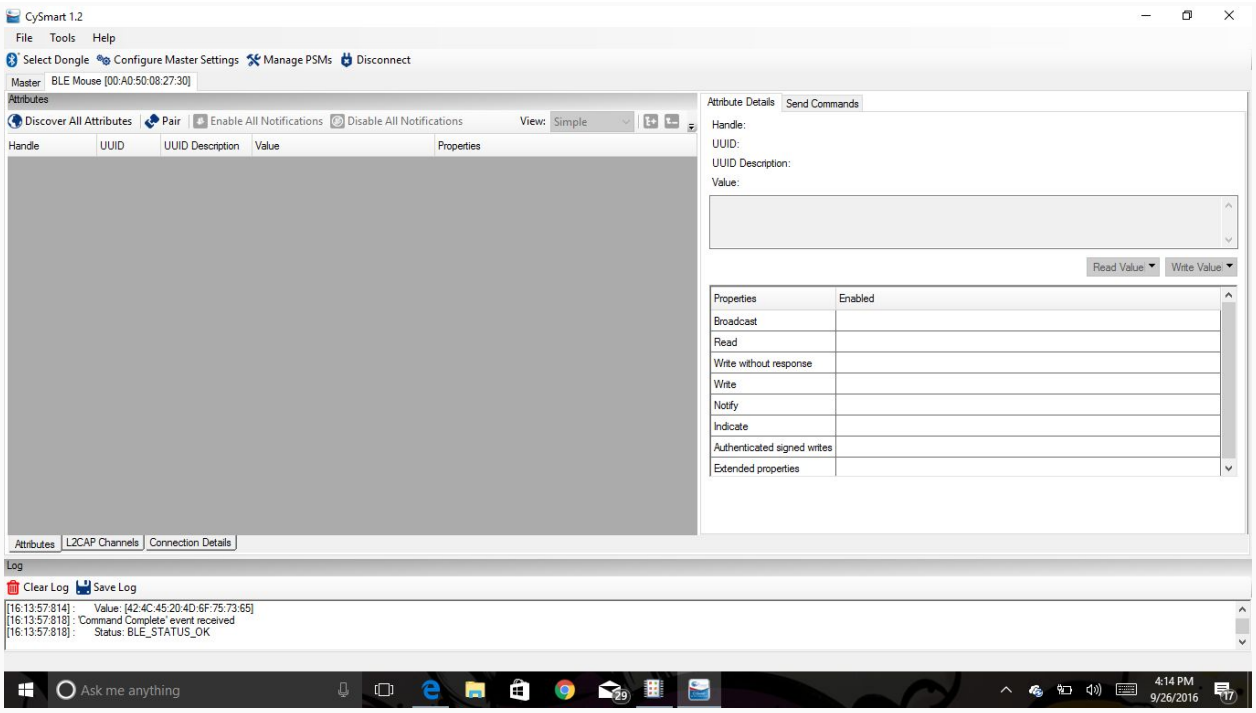
c. Copy over the main.c firwmware from the PSoC 4 BLE lab 2 template.

*See code at bottom


BLE Lab 2 - PSoC Creator 3.1
Configure 'BLE'
Name: BLE
Profiles
GAP Settings
Built-in
Add Characteristic
Generic Attribute
Service Changed
Client Characteristic Con
Immediate Alert
Alert Level
Device Information
Manufacturer Name String
Model Number String
Serial Number String
Hardware Revision String
Firmware Revision String
Software Revision String
System ID
IEEE 11073-20601 Regulato
PnP ID
Service: Device Information
The Device Information Service exposes manufacturer and/or vendor information about a device.
UUID: 180A
Service type
Primary
Secondary
Included services:
Immediate Alert
Datasheet
OK
Apply
Cancel
Erasing...
Programming of Flas
Protecting...
Verify Checksum...
Device 'PSoC 4200 BLE CY8C4247LQI-BL483' was successfully programmed at 09/26/2016 12:34:59.
Open datasheet
Bluetooth Low Energy (BLE)
Inst_N
BLE
Bluetooth
SMART
Ready
12:47 PM
9/26/2016


Design01-000 - PSoC Creator 3.1
Configure 'BLE'
Name: BLE_1
General
Profiles
GAP Settings
Built-in
Peripheral role
Advertisement settings
Advertisement packet
Scan response packet
Security
Device address
Public address (Company ID - Company assigned):
00A050-XXXXXX
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:
BLE Mouse
Appearance:
Generic Tag
MTU size (bytes):
23
TX power level (dBm):
0
Restore Defaults
Datasheet
OK
Apply
Cancel
Device 'PRoC BLE CYBL10563-56LQXI' was successfully programmed at 09/26/2016 16:12:57.
Ready
{X=315,Y=69}
4:20 PM
9/26/2016


BLE Lab 2 - PSoC Creator 3.1
File Edit View Project Build Debug Tools Window Help
Configure 'BLE'
Name: BLE
General Profiles GAP Settings Built-in
Advertisement data settings:
Flags
General discoverable mode
BR/EDR not supported
Local Name
TX Power Level
Slave Connection Interval Range
Service UUID
Immediate Alert
Device Information
Service Data
Service Manager TK Value
Appearance
Restore Defaults
Datasheet
OK
Apply
Cancel
Erasing...
Programming of Flas
Protecting...
Verify Checksum...
Device 'PSoC 4200 BLE CY8C4247LQI-BL483' was successfully programmed at 09/26/2016 12:34:59.
Component Catalog
Bluetooth Low Energy (BLE)
Inst_N
BLE
Bluetooth SMART
Ready
12:51 PM
9/26/2016


CySmart 1.2
File Tools Help
Select Dongle Configure Master Settings Manage PSMs Disconnect
Master BLE Mouse [00:A0:50:08:27:30]
Attributes
Discover All Attributes Pair Enable All Notifications Disable All Notifications View: Simple
Handle UUID UUID Description Value Properties
Attribute Details Send Commands
Handle:
UUID:
UUID Description:
Value:
Read Value Write Value
Properties Enabled
Broadcast
Read
Write without response
Write
Notify
Indicate
Authenticated signed writes
Extended properties
Attributes L2CAP Channels Connection Details
Log
Clear Log Save Log
[16:13:57:814] : Value: [42:4C:45:20:4D:6F:75:73:65]
[16:13:57:818] : 'Command Complete' event received
[16:13:57:818] : Status: BLE_STATUS_OK
4:14 PM
9/26/2016

Code:

```
/*******************************************************************************
* File Name: main.c
*
* Version: 1.0
*
* Description:
*  This is the source code for the PSoC 4 BLE lab 2 - Setting up a Connection.
*
* Hardware Dependency:
*  CY8CKIT-042 BLE Pioneer Kit
*
********************************************************************************
* Copyright 2014, Cypress Semiconductor Corporation.  All rights reserved.
* You may use this file only in accordance with the license, terms, conditions,
* disclaimers, and limitations in the end user license agreement accompanying
* the software package with which this file was provided.
```

```
*******************************************************************************/
#include <project.h>
/***************************************
*           API Constants
***************************************/
#define NO_ALERT        (0u)
#define MILD_ALERT              (1u)
#define HIGH_ALERT              (2u)
#define NO_ALERT_COMPARE   (0u)
#define MILD_ALERT_COMPARE (250u)
#define HIGH_ALERT_COMPARE (500u)
/***************************************
*           Function Prototypes
***************************************/
void StackEventHandler(uint32 event, void* eventParam);
void IasEventHandler(uint32 event, void* eventParam);
void HandleAlertLEDs(uint8 status);

/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
*  Main function.
*
* Parameters:
*  None
*
* Return:
*  None
```

```
*
*******************************************************************************/
int main()
{
        CyGlobalIntEnable;
        /* Start the BLE component and register StackEventHandler function */
        CyBle_Start(StackEventHandler);

        /* Start the PWM component */
        PWM_Start();
        /* Register IAS event handler function */
        CyBle_IasRegisterAttrCallback(IasEventHandler);

        while(1)
        {
        /* Process all the pending BLE tasks. This single API call to
        * will service all the BLE stack events. This API MUST be called at least once
        * in a BLE connection interval */
        CyBle_ProcessEvents();
        }
}
/*******************************************************************************
* Function Name: StackEventHandler
********************************************************************************
*
* Summary:
*  This is an event callback function to receive events from the BLE Component.
*
* Parameters:
*  uint8 event:     Event from the CYBLE component
```

```
*  void* eventParams: A structure instance for corresponding event type. The
*                     list of event structure is described in the component
*                     datasheet.
*
* Return:
*  None
*
*******************************************************************************/
void StackEventHandler(uint32 event, void *eventParam)
{
        switch(event)
        {
        /* Mandatory events to be handled by Find Me Target design */
        case CYBLE_EVT_STACK_ON:
        case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
        /* Start the BLE fast advertisement. */
        CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
        break;

        default:
                break;
        }
}
/*******************************************************************************
* Function Name: IasEventHandler
********************************************************************************
*
* Summary:
*  This is an event callback function to receive events from the BLE Component,
*  which are specific to Immediate Alert Service.
```

```
*
* Parameters:
*  uint8 event:     Write Command event from the CYBLE component.
*  void* eventParams: A structure instance of CYBLE_GATT_HANDLE_VALUE_PAIR_T
*                     type.
*
* Return:
*  None
*
*******************************************************************************/
void IasEventHandler(uint32 event, void *eventParam)
{
            uint8 alertLevel;


            /* Alert Level Characteristic write event */
            if(event == CYBLE_EVT_IASS_WRITE_CHAR_CMD)
            {
            /* Extract Alert Level value from the GATT DB using the
  * CYBLE_IAS_ALERT_LEVEL as a parameter to CyBle_IassGetCharacteristicValue
  * routine. Store the Alert Level Characteristic value in "alertLevel"
  * variable */
            CyBle_IassGetCharacteristicValue(CYBLE_IAS_ALERT_LEVEL, sizeof(alertLevel),
&alertLevel);


            /*Based on alert Level level recieved, Drive LED*/
            HandleAlertLEDs(alertLevel);
            }
}
/*******************************************************************************
* Function Name: HandleAlertLEDs
*******************************************************************************
```

```
*
* Summary:
*  This function drives the LED based on the alert level
*
* Parameters:
*  uint8 status:    Alert level
*
* Return:
*  None
*
*******************************************************************************/
void HandleAlertLEDs(uint8 status)
{
        /* Update Alert LED status based on IAS Alert level characteristic. */
        switch(status)
        {
        case NO_ALERT:
        PWM_WriteCompare(NO_ALERT_COMPARE);
        break;
        case MILD_ALERT:
        PWM_WriteCompare(MILD_ALERT_COMPARE);
        break;

        case HIGH_ALERT:
        PWM_WriteCompare(HIGH_ALERT_COMPARE);
        break;
        }
}

/* [] END OF FILE */
```

Conclusion:

This Lab enabled us to expand our knowledge of the PsoC program and the BLE components. By learning how to configure each component accordingly we were able to press a button on one BLE device, which then sent a signal to alert the other BLE device.