Home Automation

Report By:

Pratik Suresh Sheth

Executive Summary:

Objective is to learn Bluetooth Low Energy and switching ON or OFF a device with a Bluetooth handset having CySmart App and also controlling the intensity of the devices like fan regulator etc.

Project Description:

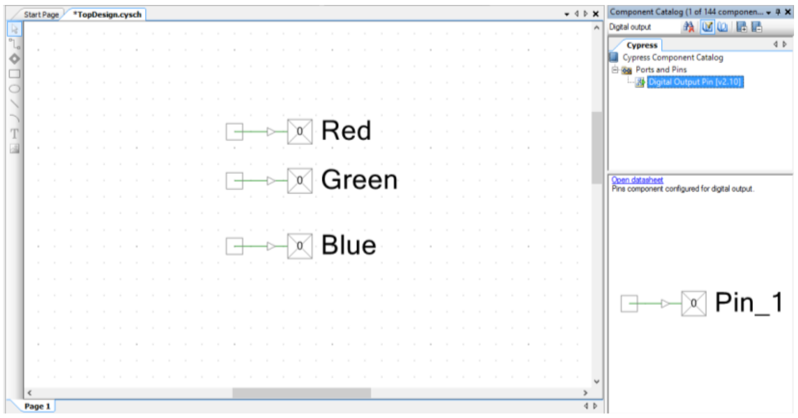
We have created an flow to so that when we give 0x01 the device will only work on ON/OFF mode and when we give 0x02 the device will work on the defined intensity value from the user.

Components Used:

The following are the components used in this project:

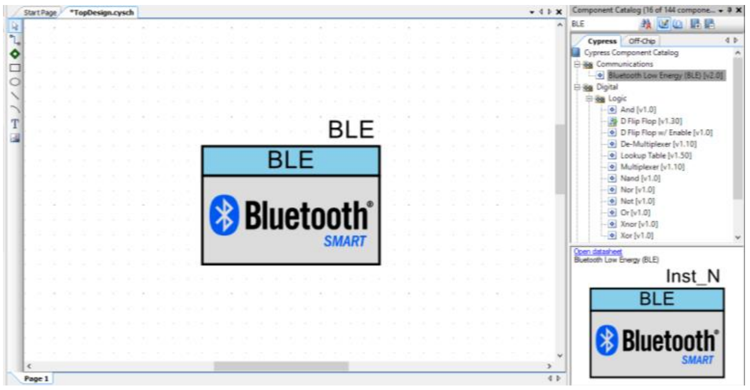
1) Digital Output pin:

We have used 3 digital out pin for 3 LEDs to be connected on respective pins.



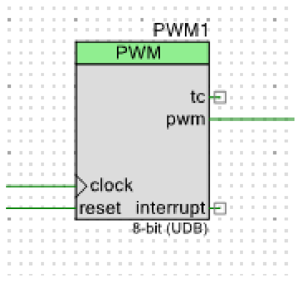
2) Bluetooth Low Energy:

We also have Bluetooth Low Energy Block.

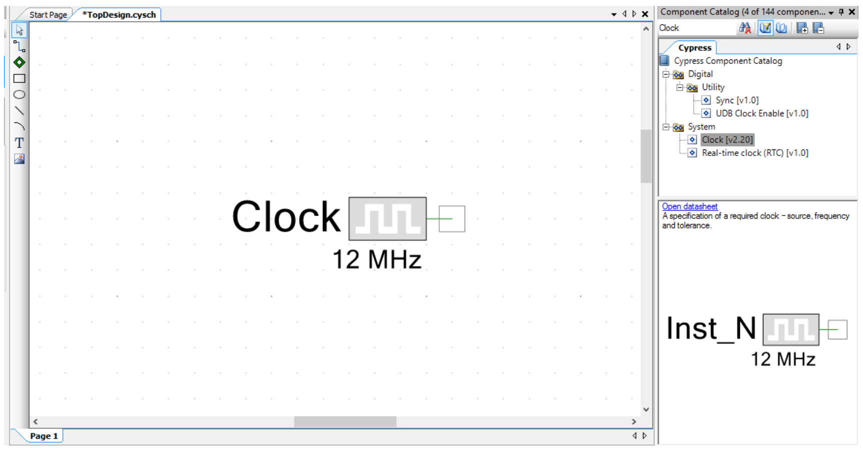


3) PWM:

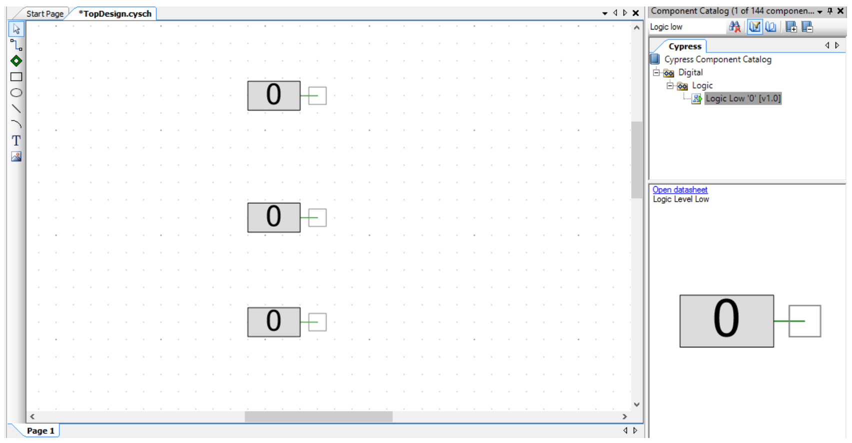
We have used 3 PWM blocks.



4) Clock


Start Page
*TopDesign.cysch
Clock
12 MHz
Page 1
Component Catalog (4 of 144 componen...
Clock
Cypress
Cypress Component Catalog
Digital
Utility
Sync [v1.0]
UDB Clock Enable [v1.0]
System
Clock [v2.20]
Real-time clock (RTC) [v1.0]
Open datasheet
A specification of a required clock – source, frequency and tolerance.
Inst_N
12 MHz

5) Logic Low:

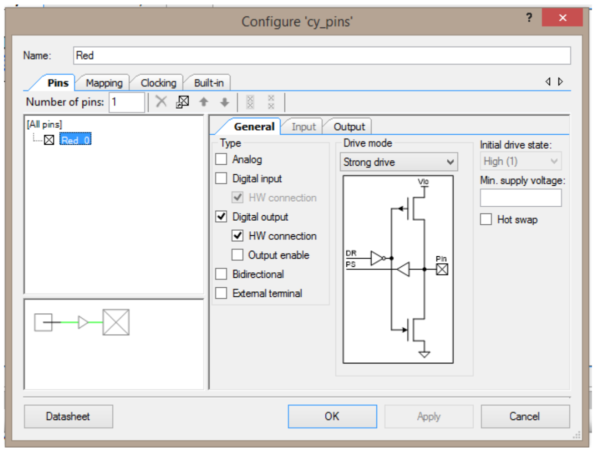


Component Configuration and Pin Map:

The following are the component configuration and pin mapping was done in this project:

1) DigitalOut pin

We selected HW connection and also configured Initial drive state to High (1).


Configure 'cy_pins'

Name: Red

Pins | Mapping | Clocking | Built-in

Number of pins: 1

[All pins]
Red_0

General | Input | Output

Type
- [ ] Analog
- [ ] Digital input
- [✓] HW connection
- [✓] Digital output
- [✓] HW connection
- [ ] Output enable
- [ ] Bidirectional
- [ ] External terminal

Drive mode: Strong drive

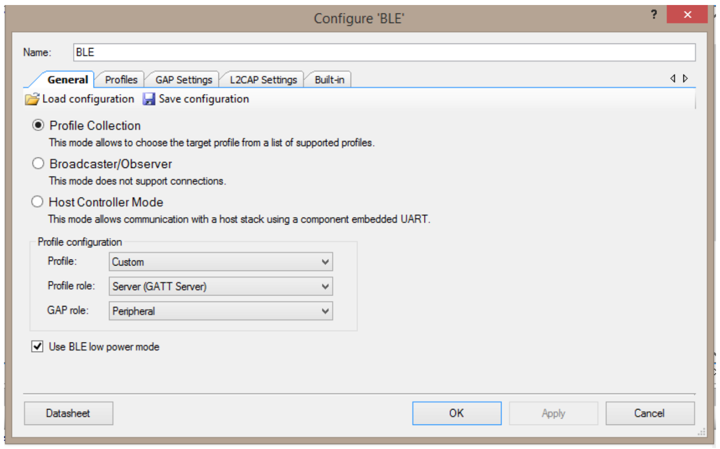
Initial drive state: High (1)

Min. supply voltage:

- [ ] Hot swap

2) Bluetooth Low Energy (BLE):

We changed the name to BLE. Under General Settings we changed the profile to Custom and profile role to Server (GATT Server) and also defined that GAP role as peripheral.


Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

Load configuration | Save configuration

(●) Profile Collection
This mode allows to choose the target profile from a list of supported profiles.

( ) Broadcaster/Observer
This mode does not support connections.

( ) Host Controller Mode
This mode allows communication with a host stack using a component embedded UART.

Profile configuration

Profile: Custom

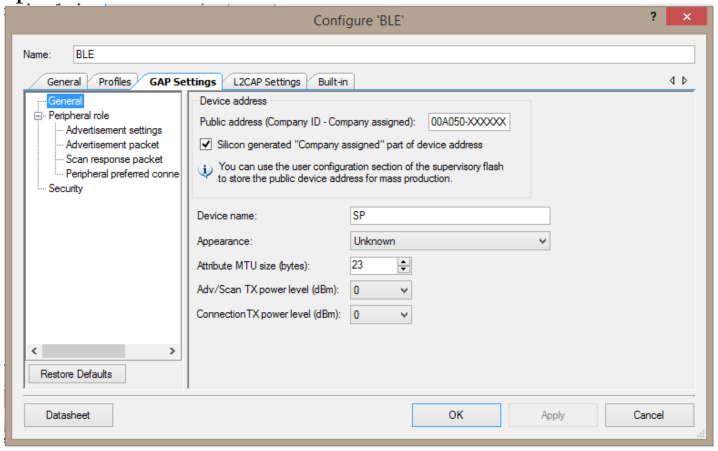
Profile role: Server (GATT Server)

GAP role: Peripheral

- [✓] Use BLE low power mode

In GAP Settings,

General Settings: We assigned a device name and selected appearance to unknown. We kept the transmission power level to 0 dBm.


Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

General
Peripheral role
- Advertisement settings
- Advertisement packet
- Scan response packet
- Peripheral preferred conne
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: SP

Appearance: Unknown

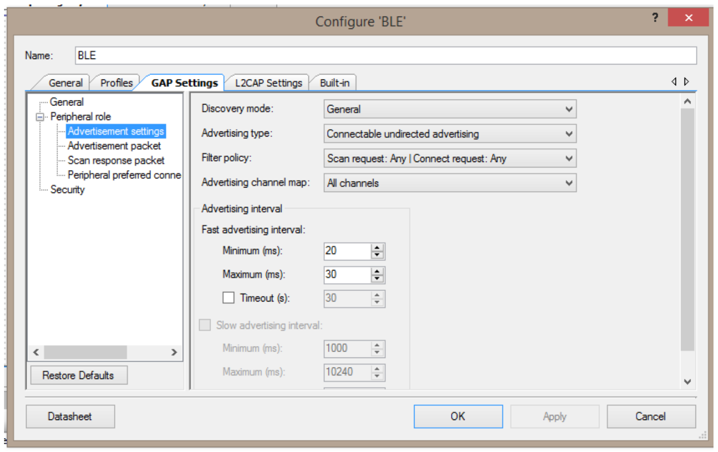
Attribute MTU size (bytes): 23

Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

Peripheral Role:

Advertisement Setting: We assigned fast transmission interval ie Minimum 20ms and maximum of 30ms. So that after every 20ms the peripheral will send 1 advertising packet.


Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

- General
- Peripheral role
  - Advertisement settings
  - Advertisement packet
  - Scan response packet
  - Peripheral preferred conne
- Security

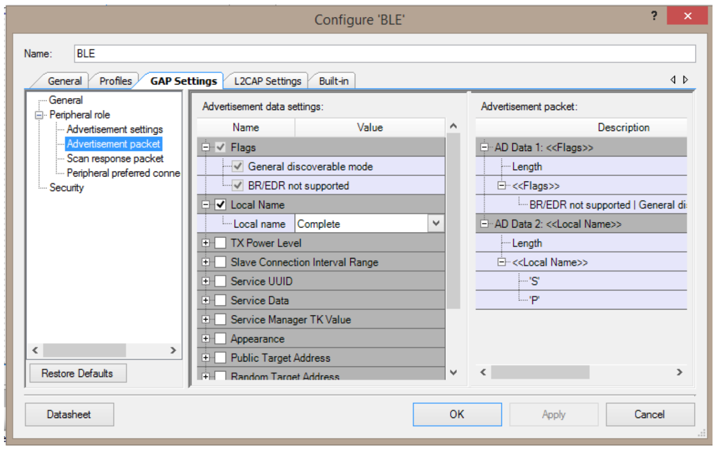
Discovery mode: General
Advertising type: Connectable undirected advertising
Filter policy: Scan request: Any | Connect request: Any
Advertising channel map: All channels

Advertising interval
Fast advertising interval:
Minimum (ms): 20
Maximum (ms): 30
Timeout (s): 30

Slow advertising interval:
Minimum (ms): 1000
Maximum (ms): 10240

Restore Defaults | Datasheet | OK | Apply | Cancel

Advertisement packet: We clicked on Local name so that our device name would be visible on CySmart app.


Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

Advertisement data settings:

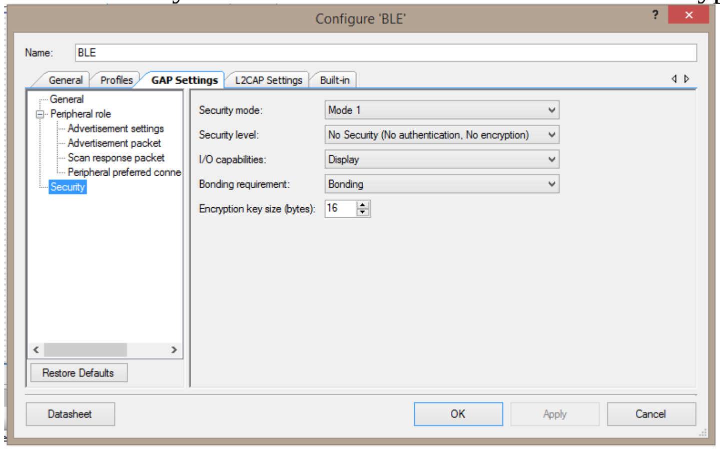
Name	Value
✓ Flags	
✓ General discoverable mode	
✓ BR/EDR not supported	
✓ Local Name	
Local name	Complete
TX Power Level	
Slave Connection Interval Range	
Service UUID	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	

Advertisement packet:

Description
- AD Data 1: <<Flags>>
  - Length
  - <<Flags>>
    - BR/EDR not supported | General di
- AD Data 2: <<Local Name>>
  - Length
  - <<Local Name>>
    - 'S'
    - 'P'

Restore Defaults | Datasheet | OK | Apply | Cancel

Security: We assigned the security level to No authentication and No Encryption.


Configure 'BLE'

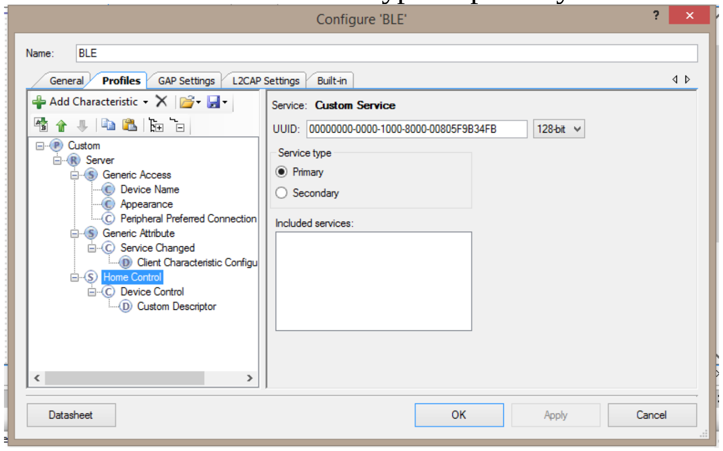
Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

Security mode: Mode 1
Security level: No Security (No authentication, No encryption)
I/O capabilities: Display
Bonding requirement: Bonding
Encryption key size (bytes): 16

Restore Defaults | Datasheet | OK | Apply | Cancel

In Profiles Settings,
We added a Custom service and selected Service type as primary.



Name: BLE

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configu
    - Home Control
      - Device Control
        - Custom Descriptor

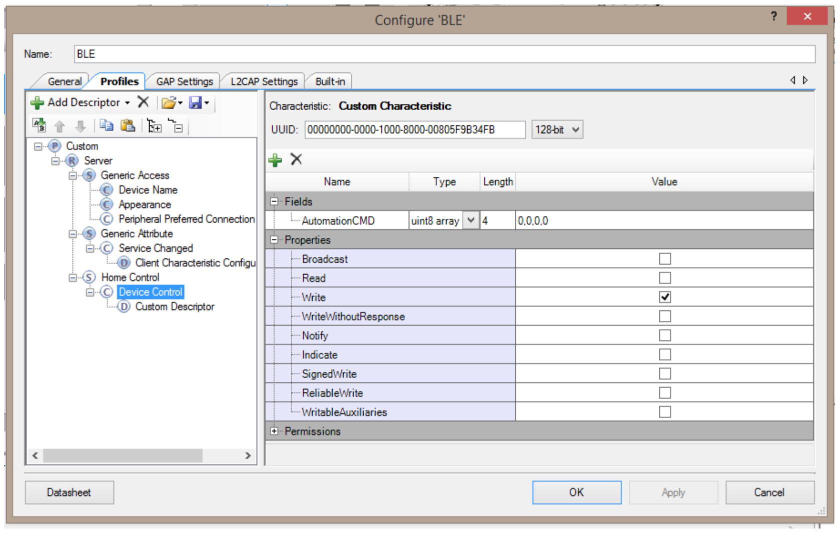
Service: **Custom Service**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Service type: [● Primary] ○ Secondary

Included services:

We also added Custom Characteristics and named as Device Control. We assigned fields as uint8 array and a length of 4. Since we are going to write the data on the board, so we selected properties as write.



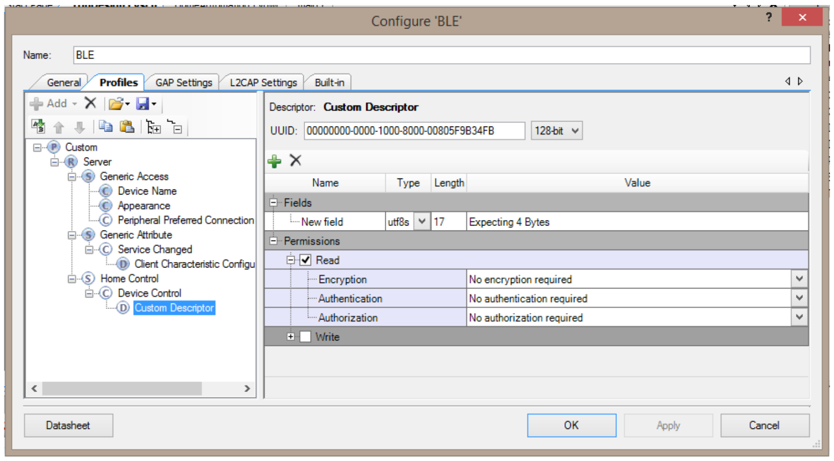
Name: BLE

Characteristic: **Custom Characteristic**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Name	Type	Length	Value
Fields			
AutomationCMD	uint8 array	4	0,0,0,0
Properties			
Broadcast			☐
Read			☐
Write			✓
WriteWithoutResponse			☐
Notify			☐
Indicate			☐
SignedWrite			☐
ReliableWrite			☐
WritableAuxiliaries			☐
Permissions			

We also added Custom Descriptor. We assigned fields as utf8s and a length of 17. In permissions we are going to read the data from user.



Name: BLE

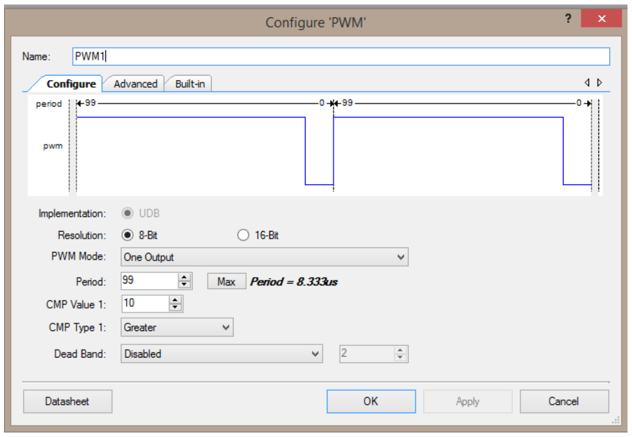
Descriptor: **Custom Descriptor**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Name	Type	Length	Value
Fields			
New field	utf8s	17	Expecting 4 Bytes
Permissions			
✓ Read			
Encryption			No encryption required
Authentication			No authentication required
Authorization			No authorization required
☐ Write			

3) PWM:

We configured PWM as one output mode with period of 99 and compare value 10 and also set the resolution to 8 bit.



Name: PWM1

Implementation: UDB

Resolution: [✓ 8-Bit] 16-Bit

PWM Mode: One Output

Period: 99 Max *Period = 8.333us*

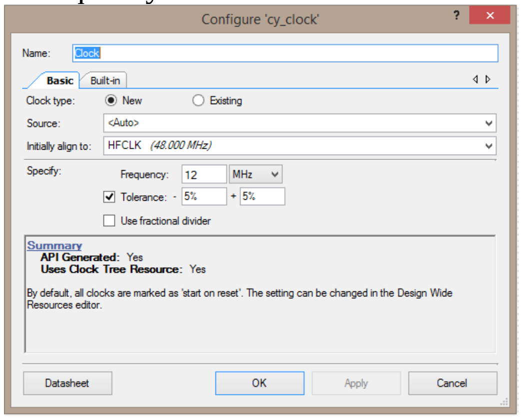
CMP Value 1: 10

CMP Type 1: Greater

Dead Band: Disabled 2

4) Clock

We defined the clock frequency to 12MHz.



Name: Clock

Clock type: [✓ New] Existing

Source: <Auto>

Initially align to: HFCLK *(48.000 MHz)*

Specify: Frequency: 12 MHz

☑ Tolerance: - 5% + 5%

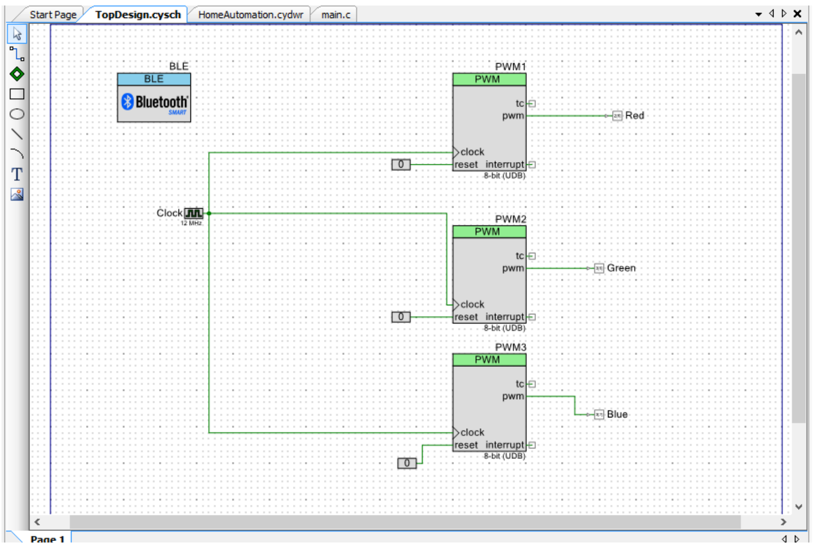
☐ Use fractional divider

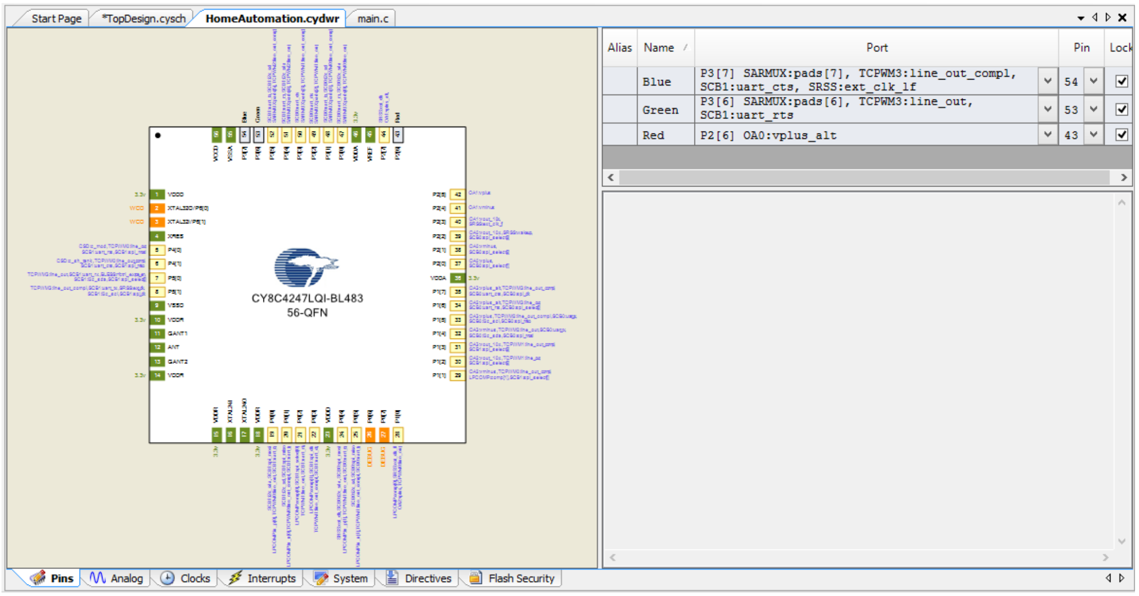
Summary

API Generated: Yes

Uses Clock Tree Resource: Yes

By default, all clocks are marked as 'start on reset'. The setting can be changed in the Design Wide Resources editor.





Alias	Name	Port	Pin	Lock
	Blue	P3[7] SARMUX:pads[7], TCPWM3:line_out_compl, SCB1:uart_cts, SRSS:ext_clk_lf	54	✓
	Green	P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart_rts	53	✓
	Red	P2[6] OA0:vplus_alt	43	✓

API Description:

API stands for Application Programming Interface. It is a higher level code to complete some operation by a single function. The following are the API's used in this project.

1) PWM1_Start(); PWM2_Start(); PWM3_Start();
This instruction is used to initialize the all 3 PWM operations.

2) PWM1_WriteCompare(0); PWM2_WriteCompare(0); PWM3_WriteCompare(0);
This instruction is used to turn off all the three LEDs (RGB).

3) Clock_Start();
This instruction is used to wake up all the devices which are connected to this block.

4) CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
This instruction starts advertising once the device is disconnected.

5) CYBLE_EVT_GATT_CONNECT_IND:
This instruction indicates when a device is connected.

Output Observed:

We have control the devices by simply controlling through the CySmart application. In that we gave the command to switched ON the device (0x01 0x01 0x00 0x00) and for switching OFF the device we gave (0x01 0x01 0x01 0x00) command and for controlling the intensity we gave (0x02 0x01 0xC7 0x00) such that we get 50% duty cycle. We repeated this for controlling different LEDs. In this the first term defines the mode 0x01 - ON/OFF Mode and 0x02 - Control the intensity, the term defines the no of devices connected and the third term defines the intensity value and the fourth term is reserved for future use.

Test and Debug:

We switched OFF all the three LEDs. Then we switched ON Red LED. Then we increased the intensity of light to Full glow (0xFF). Then we switched OFF Red LED and changed the intensity to minimum (0x00). Then we switched ON the Red LED and observed the intensity to be low since it was set to 0x00. We again switched OFF the LED and changed the intensity to maximum and again we switched ON the Red LED and observed the desired output.

APPENDIX A

Code:

```
#include <project.h>
CYBLE_CONN_HANDLE_T connectionHandle;
CYBLE_GATTS_WRITE_REQ_PARAM_T *wrReqParam;
uint8 DevType,DevCode,DevParam,Reserve,p,OFF,q,r;


void HandleDevice(){
   if(DevType==0x01){
      switch(DevCode)
      {
         case 0x01 :
         if (DevParam == 00)
         {
            PWM1_WriteCompare(p);
            OFF=0;
         }
         else
         {
            PWM1_WriteCompare(0);
            OFF=1;
         }
         break;

         case 0x02 :
         if (DevParam == 0)
         {
            PWM2_WriteCompare(q);
            OFF=0;
         }
         else
         {
            PWM2_WriteCompare(0);
            OFF=1;
         }
         break;
```

```
        case 0x03 :
        if (DevParam == 0)
        {
           PWM3_WriteCompare(r);
           OFF=0;
        }
        else
        {
           PWM3_WriteCompare(0);
           OFF=1;
        }
        break;

        default:
        break;
     }
  }

  else if(DevType==0x02) {
     switch(DevCode)
     {
        case 0x01:
        if (OFF == 0)
        {
           p=DevParam;
           PWM1_WriteCompare(p);

        }
        else
        p=DevParam;
        break;

        case 0x02:
        if (OFF == 0)
        {
           q=DevParam;
           PWM2_WriteCompare(q);

        }
        else
        q=DevParam;
        break;

        case 0x03:
        if (OFF == 0)
```

```
          {
             r=DevParam;
             PWM3_WriteCompare(r);
          }
          else
          r=DevParam;
          break;
       }
}
}

void CustomEventHandler(uint32 event, void * eventParam){
 switch(event)
{
   case CYBLE_EVT_STACK_ON: //EVENT CALLED WHEN BLE STACK STARTS
   CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
   break;

   case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
   CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST); //starts advertising once
the device is disconnected
   break;

   case CYBLE_EVT_GATT_CONNECT_IND: //THIS is the statement when someone gets
connected it indicates
   connectionHandle=*(CYBLE_CONN_HANDLE_T *)eventParam; //this will help us to the
format of event parameter
   break;

   case CYBLE_EVT_GATTS_WRITE_REQ:
   wrReqParam = (CYBLE_GATTS_WRITE_REQ_PARAM_T *) eventParam; //For data
   if(CYBLE_HOME_CONTROL_DEVICE_CONTROL_CHAR_HANDLE == wrReqParam-
>handleValPair.attrHandle) //if wrReqParam is passed we need pointer (->) its is like having a
structure inside a structure
   {
      DevType  = (uint8)wrReqParam->handleValPair.value.val[0];
      DevCode  = (uint8)wrReqParam->handleValPair.value.val[1];
      DevParam = (uint8)wrReqParam->handleValPair.value.val[2];
      Reserve  = (uint8)wrReqParam->handleValPair.value.val[3];
      HandleDevice();
      CyBle_GattsWriteRsp(connectionHandle); //giving respone to the user

   }
   break;
```

```
    default: break;

}
}

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    Clock_Start();

    PWM1_Start();
    PWM2_Start();
    PWM3_Start();

    PWM1_WriteCompare(0);
    PWM2_WriteCompare(0);
    PWM3_WriteCompare(0);
    CyBle_Start(CustomEventHandler);

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        CyBle_ProcessEvents();
        /* Place your application code here. */
    }
}

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