Course: Summer Industrial Training 2015

Lab Report: EXTRA CREDIT LAB - Home Automation Using PSoC BLE

Report By: Ajinkya Padwad
Abhijit Gokhale

Executive Summary:

The main aim of this project was to implement a basic home automation project using the PSoC BLE to control different types of devices (LEDs in this case) and employ intensity control.

Project Description:

The first step was creating the schematic of the system at the 'Top Design' layout at PSoC Creator. The PWM and BLE modules were added along with the suitable input and output pins from the components menu. Next step was the component configuration after which, the ports were mapped to physical pins. The program code was then compiled into the 'main.c' file and lastly, the interface of the CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit was carried out.

Components used:

Digital Output Pins (LEDs) -

The digital output pins are basically the three LED pins which are software connected to the microcontroller onboard.

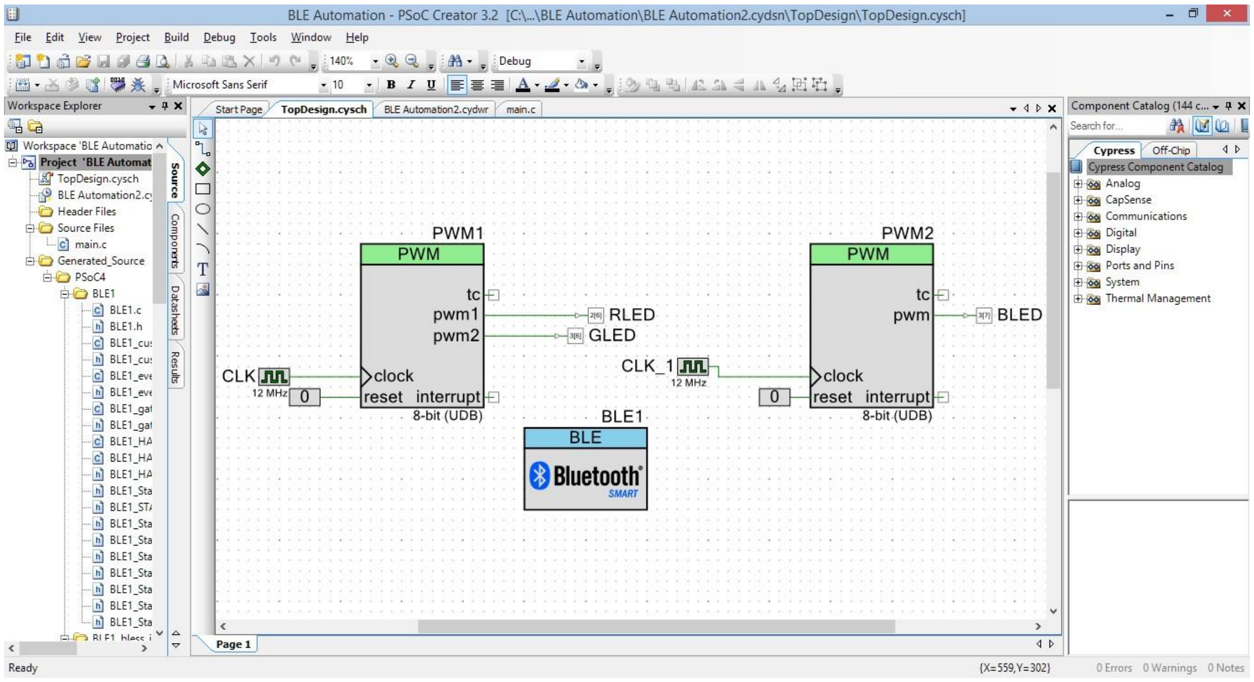
PWM Module -

PWM module is responsible for the duty cycle variation and in turn the intensity control of the LEDs.

Bluetooth Low Energy (BLE) module-

BLE module gives a GUI for the configuration of the GAP and GATT profiles for Bluetooth connectivity.

Project Schematic

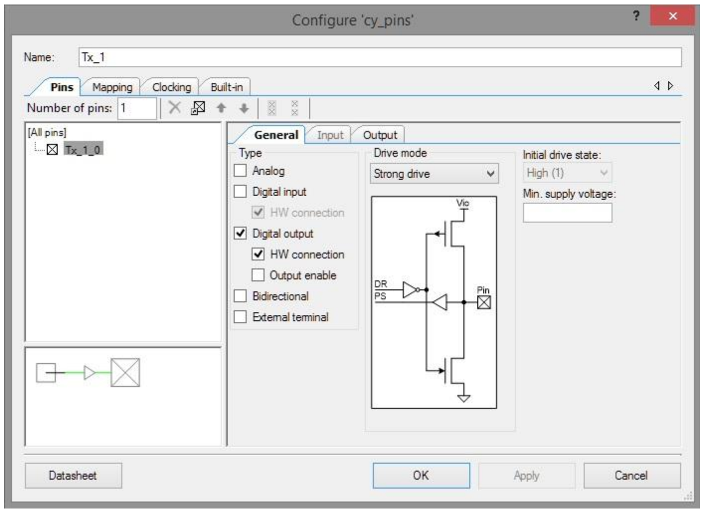


Pin Component Configuration and Pin map:

Digital Output pin -

Drive Mode This parameter configures the pin to provide one of the eight available pin drive modes.

HW Connection parameter determines whether the digital input terminal for an input pin is displayed in the schematic. It may or may not be API controlled depending upon the option being selected.


Configure 'cy_pins'

Name: Tx_1

Number of pins: 1

[All pins]
Tx_1_0

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

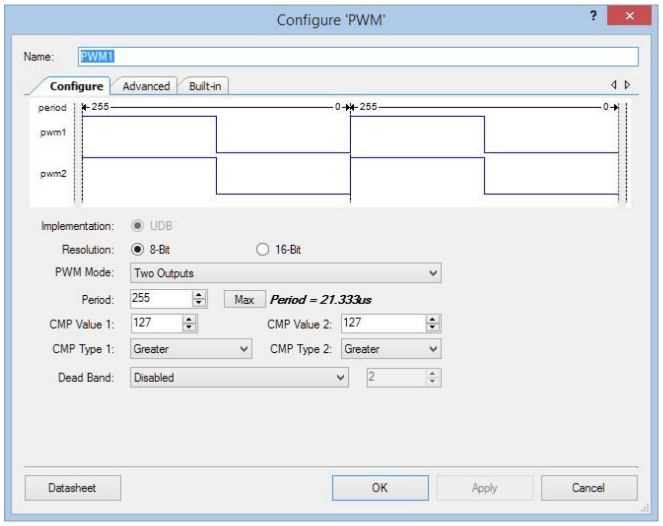
Drive mode: Strong drive

Initial drive state: High (1)

Min. supply voltage:

PWM Module -

PWM Mode parameter defines the overall functionality of the PWM. It can be- One Output – Only a single PWM output or Two Output – Two individually configurable PWM outputs.
CMP Type 1 / CMP Type 2 parameters define the two period counter comparisons that make up the PWM outputs.


Configure 'PWM'

Name: PWM1

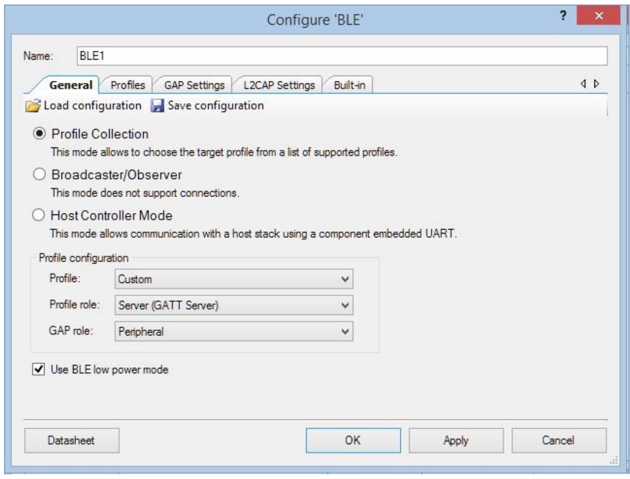
Configure | Advanced | Built-in

Implementation: UDB
Resolution: [✓ 8-Bit] 16-Bit
PWM Mode: Two Outputs
Period: 255 Max *Period = 21.333us*
CMP Value 1: 127 CMP Value 2: 127
CMP Type 1: Greater CMP Type 2: Greater
Dead Band: Disabled 2

Datasheet OK Apply Cancel

Bluetooth Low Energy (BLE) module-

The *Profile role* parameter configuration depends on the chosen Profile, and the Profile role selection affects the GAP role parameter.
The *toolbars* contain navigation options and a means to add or delete Services, Characteristics, and Descriptors.
The *GAP parameters* define the general connection settings required when connecting Bluetooth devices


Configure 'BLE'

Name: BLE1

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

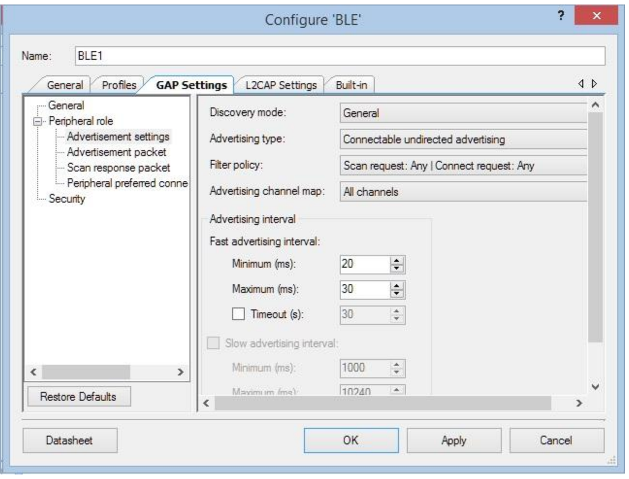
Load configuration Save configuration

[✓ 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

Datasheet OK Apply Cancel


Configure 'BLE'

Name: BLE1

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

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

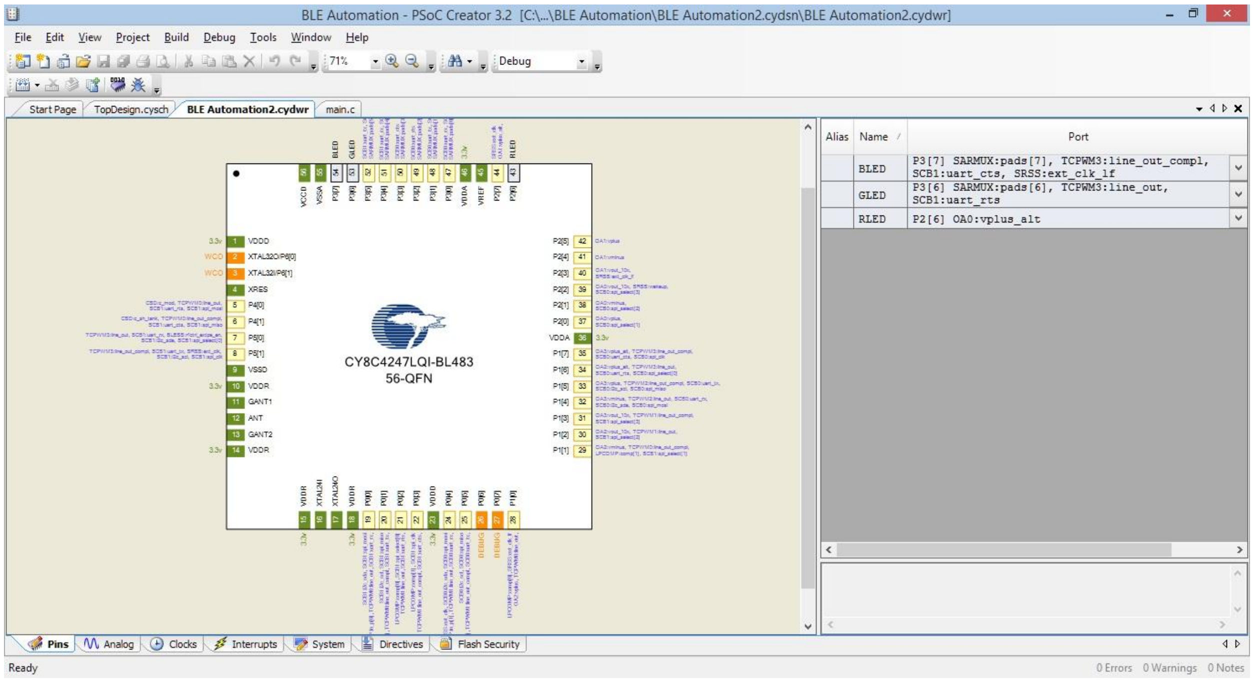
Restore Defaults

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

Datasheet OK Apply Cancel

CYDWR -



Alias	Name	Port
	BLED	P3[7] SARMUX:pads[7], TCPWM3:line_out_compl, SCB1:uart_cts, SRSS:ext_clk_lf
	GLED	P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart_rts
	RLED	P2[6] OA0:vplus_alt

API Description:

Bluetooth API functions -

- CyBle_Start

  This function initializes the BLE Stack, which consists of the BLE Stack Manager, BLE Controller, and BLE Host modules.

- CyBle_Stop

  This function stops any ongoing operation in the BLE Stack and forces the BLE Stack to shut down

- CyBle_ProcessEvents

  This function checks the internal task queue in the BLE Stack, and pending operation of the BLE Stack, if any.

- The GAP APIs allow access to the Generic Access Profile (GAP) layer of the BLE stack.
- The GATT APIs allow access to the Generic Attribute Profile (GATT) layer of the BLE stack.

Output Observed:

The CySmart app for PSoC BLE sends data bits for options selection which is then converted into appropriate signal and the ON/OFF or intensity control of the onboard LEDs is observed.

Test and Debug:

- ✓ Encountered a major problem while configuring the 'custom' profile for this project.
- ✓ Initial drive state of LEDs debugged.

Learning Outcomes:

- ➢ Learnt the various BLE APIs and the process handling.
- ➢ Importance of Bluetooth events and the role of stack handler.

APPENDIX A

Code:

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


void HandleDevice()
{


    if(DevType==0x01) //If ON/OFF Control type
    {
        switch(DevCode)
        {
            case 0x01:RLED_Write(DevParam);break;
            case 0x02:GLED_Write(DevParam);break;
            case 0x03:BLED_Write(DevParam);break;
            default:break;
        }
    }

     else if(DevType==0x02)  // If intensity control type
    {
        switch(DevCode)
        {

        case 0x01:PWM1_WriteCompare1(DevParam);
                  break;
        case 0x02:PWM1_WriteCompare2(DevParam);
                  break;
        case 0x03:PWM2_WriteCompare(DevParam);
                  break;
        default:break;
        }
    }

}


void CustomEventHandler(uint32 event,void*eventParam)

{
   switch(event)
  {
    case CYBLE_EVT_STACK_ON:
    CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
    break;

    case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
    CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
    break;
    case CYBLE_EVT_GATT_CONNECT_IND:
    connectionHandle=*(CYBLE_CONN_HANDLE_T *)eventParam;
    break;

    case CYBLE_EVT_GATTS_WRITE_REQ:
    wrReqParam=(CYBLE_GATTS_WRITE_CMD_REQ_PARAM_T*)eventParam;
    if(CYBLE_HOME_CONTROL_DEVICE_CONTROL_CHAR_HANDLE==wrReqParam-
>handleValPair.attrHandle)
    {
        DevType=(uint8)wrReqParam->handleValPair.value.val[0]; // ON/OFF or
intensity control
        DevCode=(uint8)wrReqParam->handleValPair.value.val[1]; //LED Color
        DevParam=(uint8)wrReqParam->handleValPair.value.val[2];// Control bit
        Reserve=(uint8)wrReqParam->handleValPair.value.val[3]; //Reserved
        HandleDevice();
        CyBle_GattsWriteRsp(connectionHandle);
    }
    break;

    default:break;
  }
}

int main()
{
    CLK_Start();        //Clock started
    PWM1_Start();       // PWM1 Started
    PWM2_Start();       // PWM2 Started

     RLED_Write(0);     // LEDS off initially
     GLED_Write(0);
     BLED_Write(0);
    CyGlobalIntEnable; // Enable global interrupts.
    CyBle_Start(CustomEventHandler);


    for(;;)
    {

        CyBle_ProcessEvents();
    }
}
```