# LAB 1 REPORT
# BLINKING LED

**Director: Dr. Ahmed Rafaey Hussein**

**Group Members: Mohammadamin Saburruhmonfared & Chaoao Shi**

**June 9, 2016**

## OBJECTIVES

1. Learn how to use PSoC Creator to implement and debug PSoC designs
2. Implement a simple blinking LED designProcedure

## 1. Block diagram

Set up the project in PSoC Creator so that it follows the block diagram shown in Figure 1.

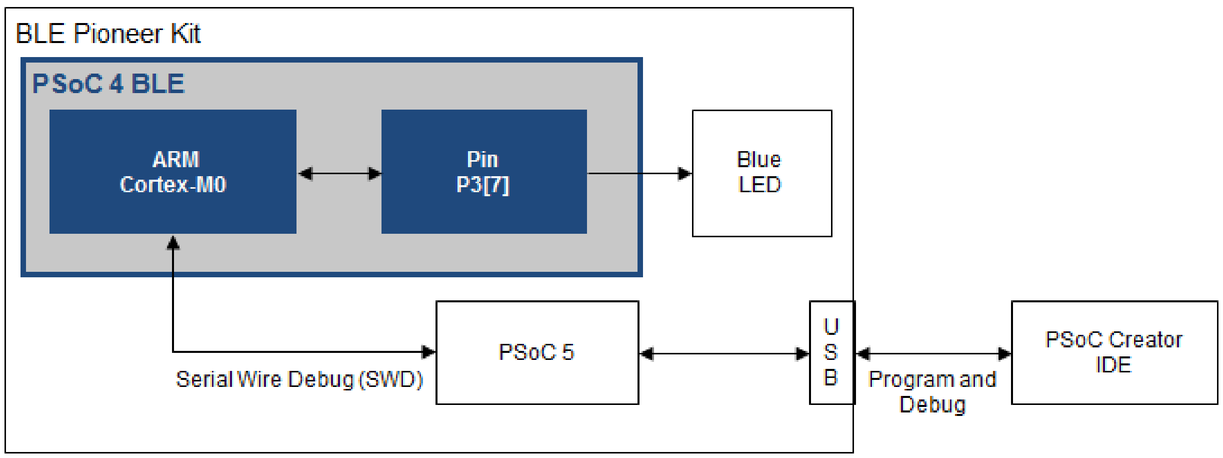


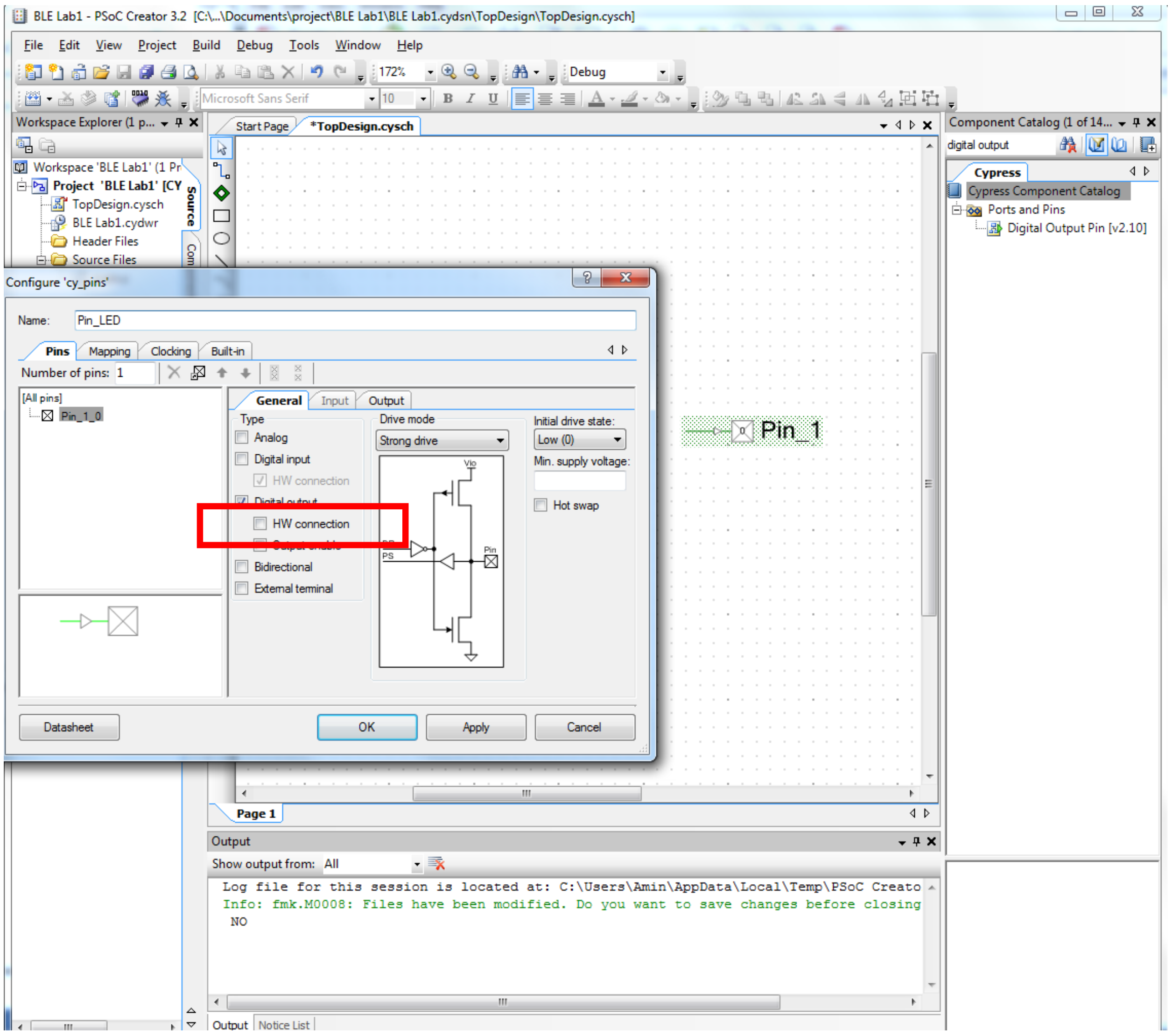
**Figure 1: Block Diagram for Lab 1**

## 2. Procedure

We did the following steps:

1. Create new project
2. Select **PSoC 4100 BLE / PSoC 4200 BLE Design** as a design template
3. go to schematic editor, Drag and drop **Digital Output Pin** from the left side software window

4. configure the component as mentioned in lab manual


BLE Lab1 - PSoC Creator 3.2 [C:\...\Documents\project\BLE Lab1\BLE Lab1.cydsn\TopDesign\TopDesign.cysch]
File Edit View Project Build Debug Tools Window Help
Workspace Explorer
Start Page
*TopDesign.cysch
Component Catalog
Cypress Component Catalog
Ports and Pins
Digital Output Pin [v2.10]
Configure 'cy_pins'
Pins Mapping Clocking Built-in
General Input Output
Datasheet OK Apply Cancel
Pin_1
Page 1
Output
Notice List

Name: Pin_LED

Number of pins: 1

[All pins] Pin_1_0

Type: Analog ☐; Digital input ☐; HW connection ☑; Digital output ☑; HW connection ☐; Output enable ☐; Bidirectional ☐; External terminal ☐

Drive mode: Strong drive

Initial drive state: Low (0)

Min. supply voltage:

Hot swap ☐

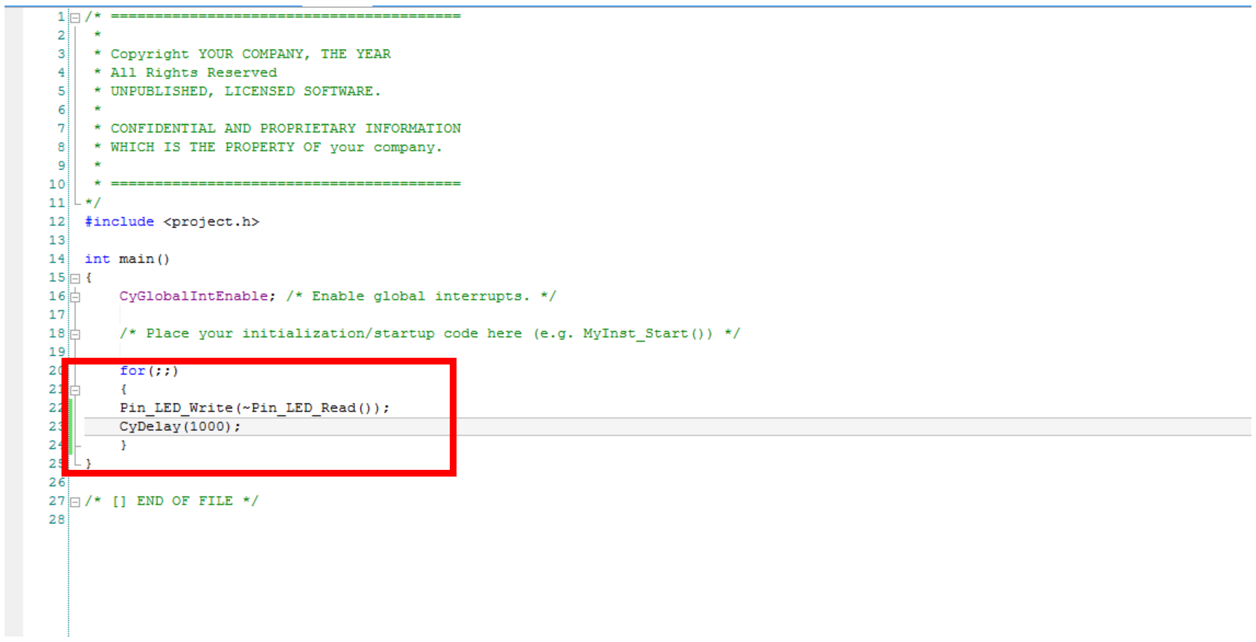
Component Catalog search: digital output

Show output from: All

```
Log file for this session is located at: C:\Users\Amin\AppData\Local\Temp\PSoC Creato
Info: fmk.M0008: Files have been modified. Do you want to save changes before closing
 NO
```

Figure 2: Schematic and Component Configuration

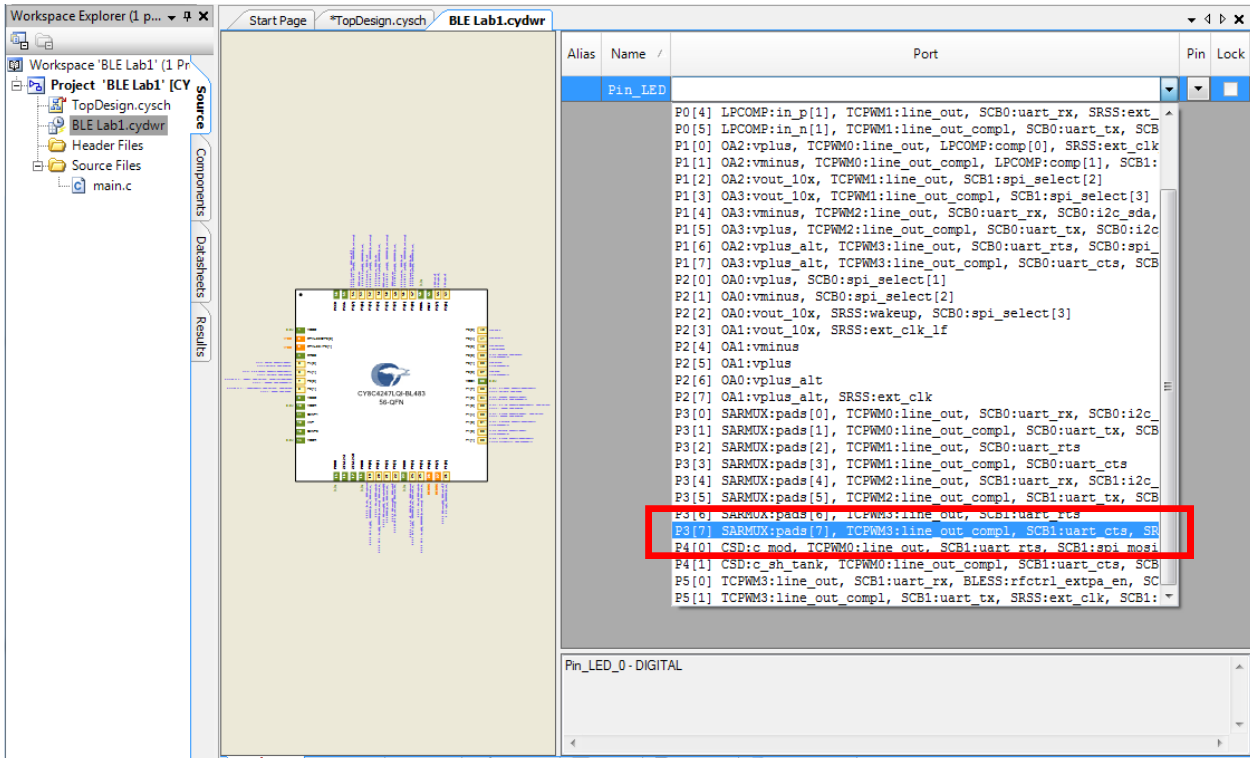
5. From the **Workspace Explorer** double-click **main.c** to open the source, and the write the C code shown below inside the for(;;).



```c
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>

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

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

    for(;;)
    {
    Pin_LED_Write(~Pin_LED_Read());
    CyDelay(1000);
    }
}

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

**Figure 3: Code Editor Showing main.c**

6. Now we are done with configuring the component. We are going to configure the DWR file, which is a Design Wide Resources. So I open BLE Lab1.cydwr, and set PIN_LED to P3[7] as shown in Figure 13.



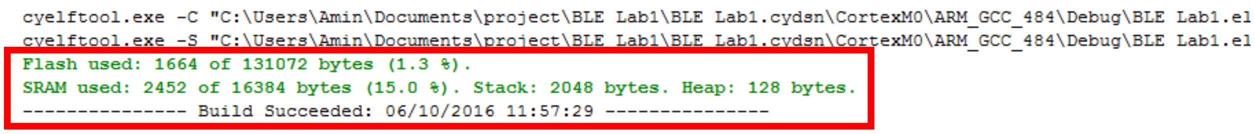
| Alias | Name | Port | Pin | Lock |
|---|---|---|---|---|
|  | Pin_LED |  |  |  |

```
P0[4] LPCOMP:in_p[1], TCPWM1:line_out, SCB0:uart_rx, SRSS:ext_
P0[5] LPCOMP:in_n[1], TCPWM1:line_out_compl, SCB0:uart_tx, SCB
P1[0] OA2:vplus, TCPWM0:line_out, LPCOMP:comp[0], SRSS:ext_clk
P1[1] OA2:vminus, TCPWM0:line_out_compl, LPCOMP:comp[1], SCB1:
P1[2] OA2:vout_10x, TCPWM1:line_out, SCB1:spi_select[2]
P1[3] OA3:vout_10x, TCPWM1:line_out_compl, SCB1:spi_select[3]
P1[4] OA3:vminus, TCPWM2:line_out, SCB0:uart_rx, SCB0:i2c_sda,
P1[5] OA3:vplus, TCPWM2:line_out_compl, SCB0:uart_tx, SCB0:i2c
P1[6] OA2:vplus_alt, TCPWM3:line_out, SCB0:uart_rts, SCB0:spi_
P1[7] OA3:vplus_alt, TCPWM3:line_out_compl, SCB0:uart_cts, SCB
P2[0] OA0:vplus, SCB0:spi_select[1]
P2[1] OA0:vminus, SCB0:spi_select[2]
P2[2] OA0:vout_10x, SRSS:wakeup, SCB0:spi_select[3]
P2[3] OA1:vout_10x, SRSS:ext_clk_lf
P2[4] OA1:vminus
P2[5] OA1:vplus
P2[6] OA0:vplus_alt
P2[7] OA1:vplus_alt, SRSS:ext_clk
P3[0] SARMUX:pads[0], TCPWM0:line_out, SCB0:uart_rx, SCB0:i2c_
P3[1] SARMUX:pads[1], TCPWM0:line_out_compl, SCB0:uart_tx, SCB
P3[2] SARMUX:pads[2], TCPWM1:line_out, SCB0:uart_rts
P3[3] SARMUX:pads[3], TCPWM1:line_out_compl, SCB0:uart_cts
P3[4] SARMUX:pads[4], TCPWM2:line_out, SCB1:uart_rx, SCB1:i2c_
P3[5] SARMUX:pads[5], TCPWM2:line_out_compl, SCB1:uart_tx, SCB
P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart_rts
P3[7] SARMUX:pads[7], TCPWM3:line_out_compl, SCB1:uart_cts, SR
P4[0] CSD:c_mod, TCPWM0:line_out, SCB1:uart_rts, SCB1:spi_mosi
P4[1] CSD:c_sh_tank, TCPWM0:line_out_compl, SCB1:uart_cts, SCB
P5[0] TCPWM3:line_out, SCB1:uart_rx, BLESS:rfctrl_extpa_en, SC
P5[1] TCPWM3:line_out_compl, SCB1:uart_tx, SRSS:ext_clk, SCB1:
```

Pin_LED_0 - DIGITAL

**Figure 4: DWR configuration**

7. Build & Program: Then we build the application to generate source code files, which will auto-complete the API names, variables and macro. Figure 14 shows the build is successful.



```
cyelftool.exe -C "C:\Users\Amin\Documents\project\BLE Lab1\BLE Lab1.cydsn\CortexM0\ARM_GCC_484\Debug\BLE Lab1.el
cyelftool.exe -S "C:\Users\Amin\Documents\project\BLE Lab1\BLE Lab1.cydsn\CortexM0\ARM_GCC_484\Debug\BLE Lab1.el
Flash used: 1664 of 131072 bytes (1.3 %).
SRAM used: 2452 of 16384 bytes (15.0 %). Stack: 2048 bytes. Heap: 128 bytes.
--------------- Build Succeeded: 06/10/2016 11:57:29 ---------------
```

**Figure 5: Successful Project Build**

The program log for this lab is shown in Figure 6.

```
KitProg version Expecting 2.16, but found 2.10. Please use PSoC Programmer GUI to upgrade firmware.
Programming started for device: 'PSoC 4200 BLE CY8C4247LQI-BL483'.
Device ID Check
Erasing...
Programming of Flash Starting...
Protecting...
Verify Checksum...
Device 'PSoC 4200 BLE CY8C4247LQI-BL483' was successfully programmed at 06/09/2016 19:04:35.
```

**Figure 6: Successful Programmed to Kit**

8. Results: The figure 7 shows the result on the board.

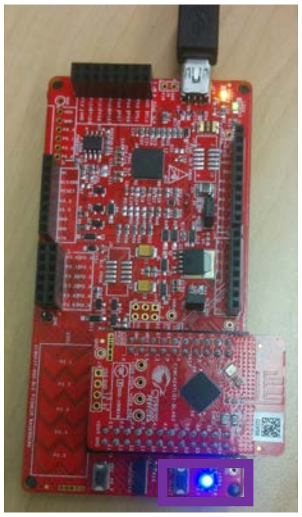


**Figure 7: Board Demonstration**

## ADDITIONAL EXERCISES

### 1. Change the color of blinking LED to red and green.

For changing the color of blinking LED to red and green instead of blue, all the steps are exactly the same as the previous experience, but in DWR configuration phase, instead of setting the PIN_LED to P3[7]. We need to set the pin to:

1. For changing the color to green the PIN_LED should set to P3[6].
2. For changing the color to red the PIN_LED should set to P2[6].

The results shows the demonstration results on the board on fig 8.

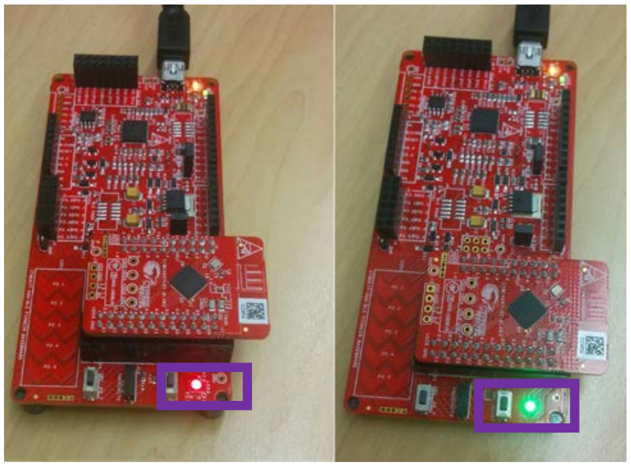
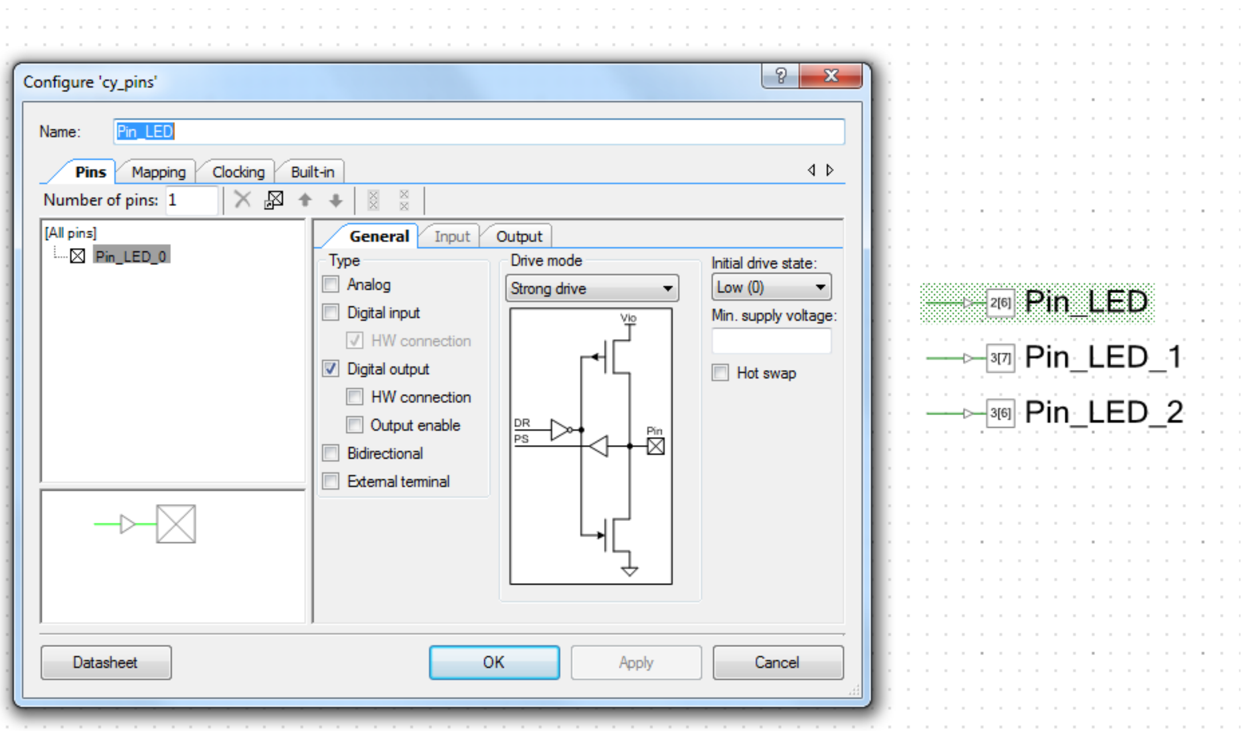


**Figure 8: Board Demonstration**

## 2. Constant white color from the RGB LED.

For getting the constant white on the LED, we need to redesign the project.

We need import three LEDs to schematic editor with the same configuration we did for the first experience. The schematic and Component Configuration are shown in the fig 12.



Configure 'cy_pins'

Name: Pin_LED

Pins | Mapping | Clocking | Built-in

Number of pins: 1

[All pins]
Pin_LED_0

General | Input | Output

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

Drive mode: Strong drive

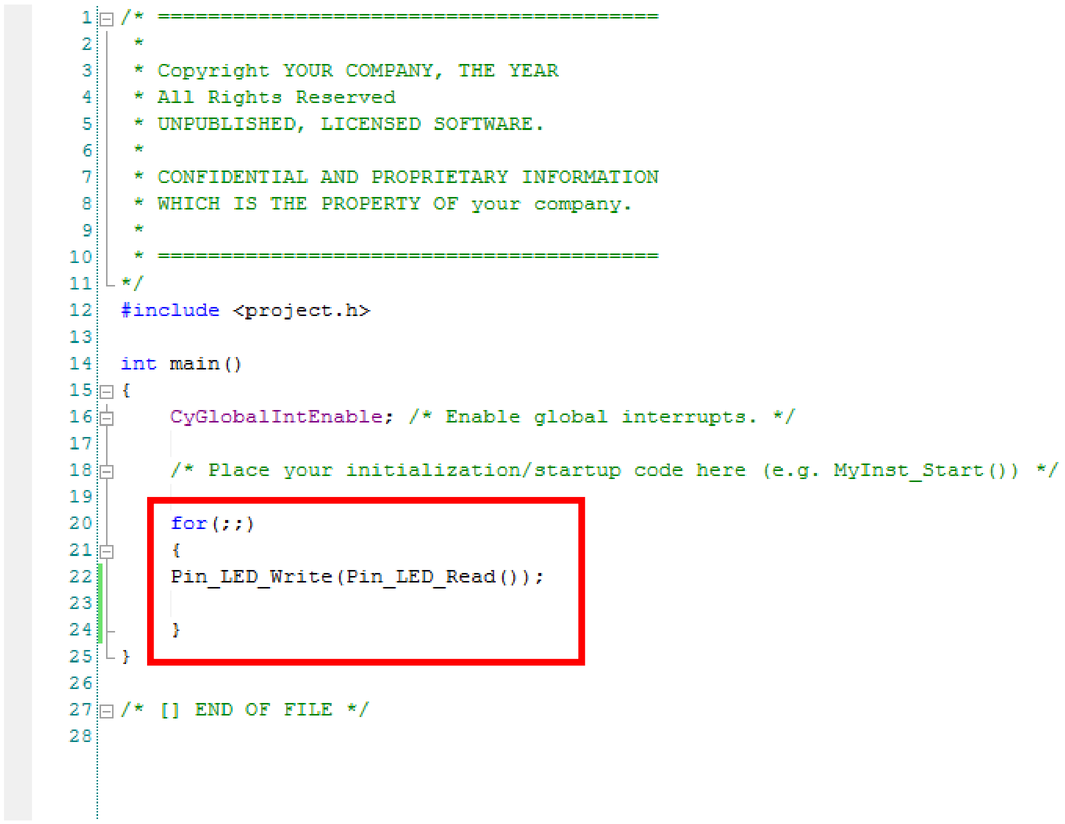
Initial drive state: Low (0)

Min. supply voltage:

- [ ] Hot swap

**Figure 9: Schematic and Component Configuration**

Because we need the constant LED, the C code also should be modified as shown in Fig 10.



```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>

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

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

    for(;;)
    {
    Pin_LED_Write(Pin_LED_Read());

    }
}

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

**Figure 10: Code Editor Showing main.c**

The results are demonstrated on fig 11.

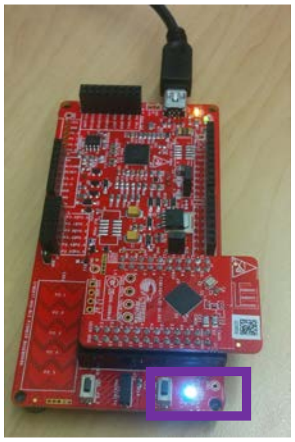


**Figure 11: Board Demonstration**

## 3. Use a PWM Component (hardware) to achieve the same result..

For redesigning the blinking blue LED ,We did the following steps:

1. Create the new project
2. Drag and drop Clock, PWM and Digital output pin to the schematic editor as shown in fig 12.

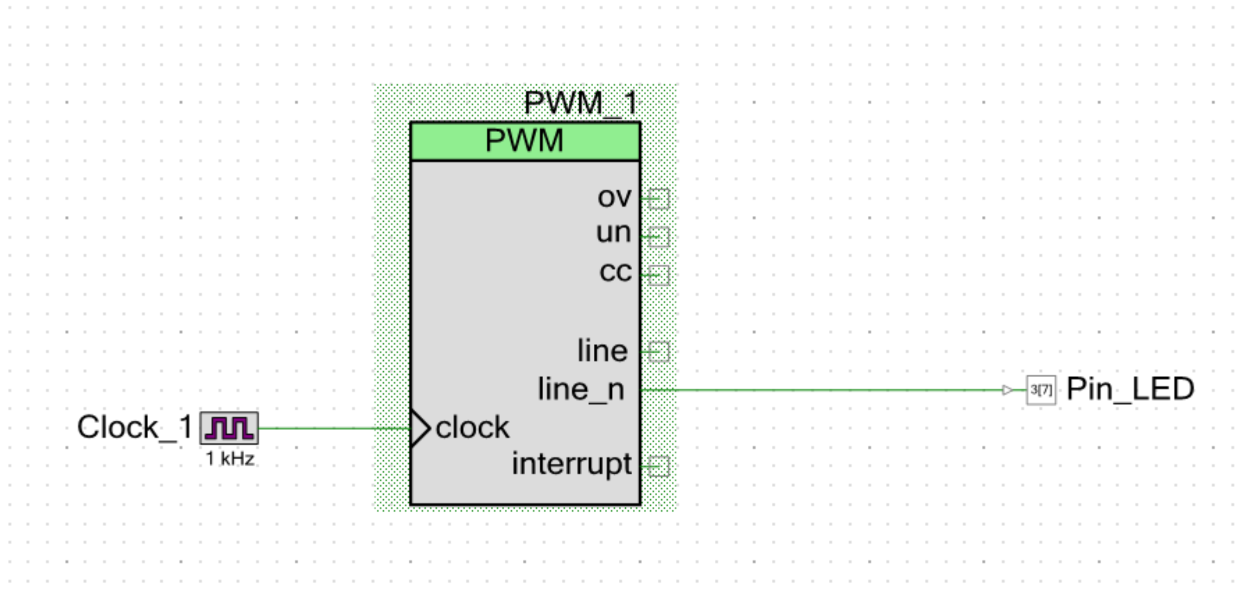
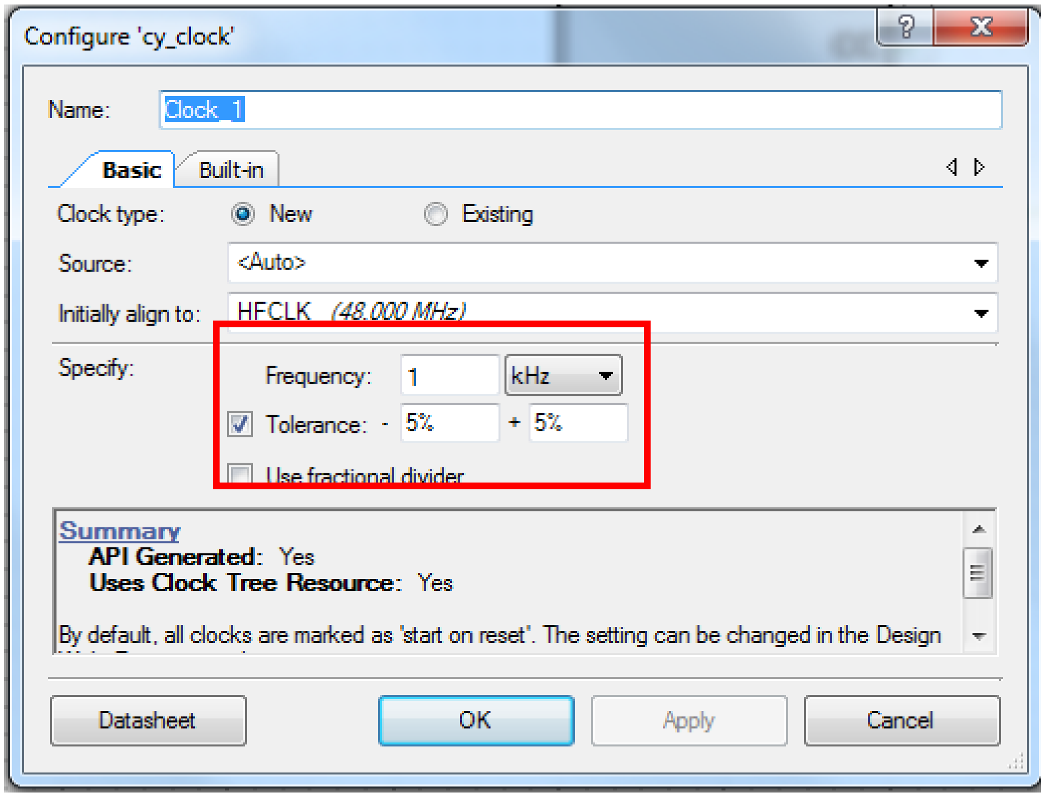


**Figure 12: Schematic**

3. These components are configured as shown in fig 13, fig 14 and fig15.



Configure 'cy_clock'

Name: Clock_1

Clock type: [✓ New] Existing

Source: <Auto>

Initially align to: HFCLK *(48.000 MHz)*

Specify:

Frequency: 1 kHz

[✓] Tolerance: - 5% + 5%

[ ] Use fractional divider

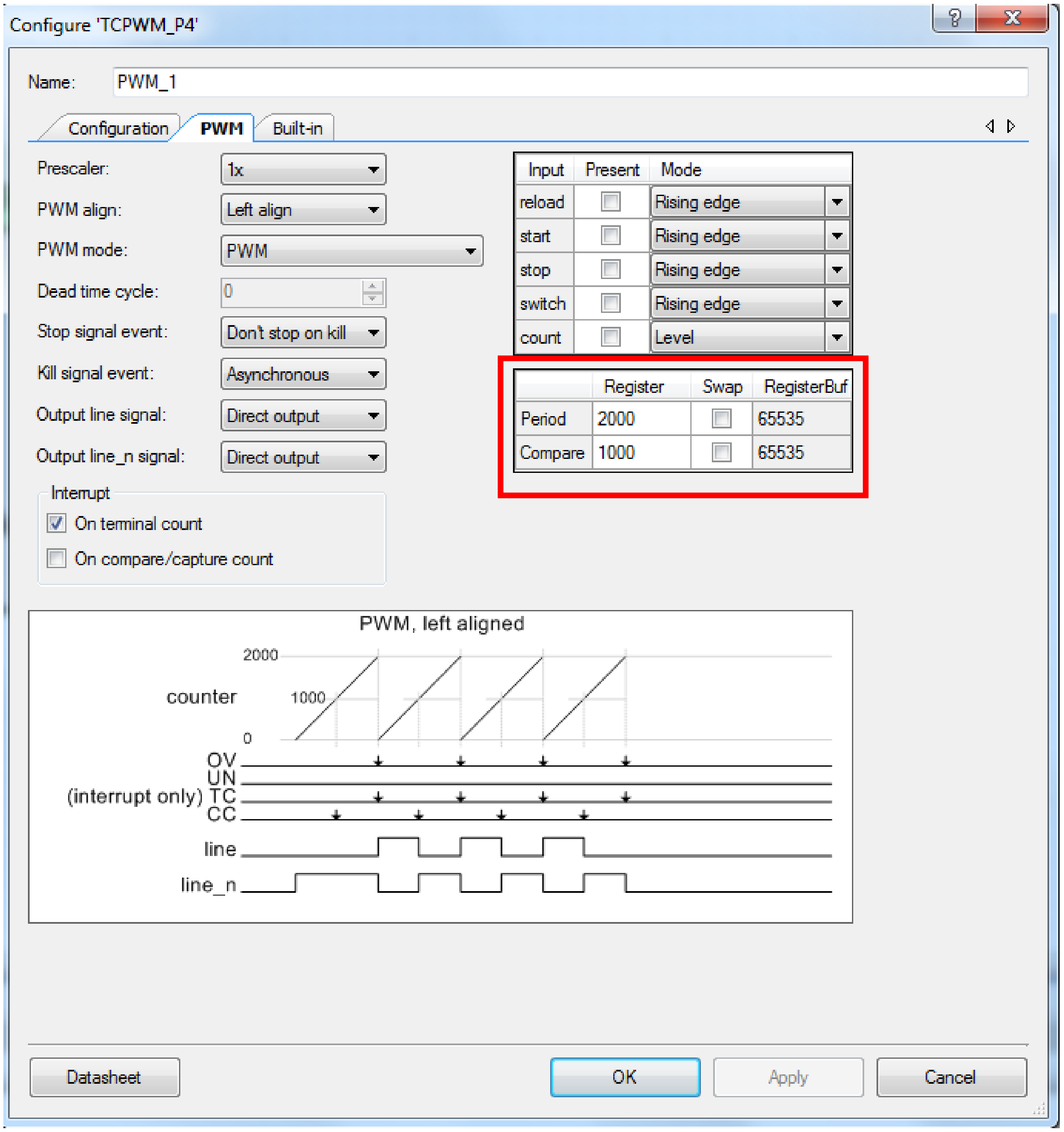
Summary

**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

Figure 13: Clock Configuration



Configure 'TCPWM_P4'

Name: PWM_1

Configuration | **PWM** | Built-in

| | |
|---|---|
| Prescaler: | 1x |
| PWM align: | Left align |
| PWM mode: | PWM |
| Dead time cycle: | 0 |
| Stop signal event: | Don't stop on kill |
| Kill signal event: | Asynchronous |
| Output line signal: | Direct output |
| Output line_n signal: | Direct output |

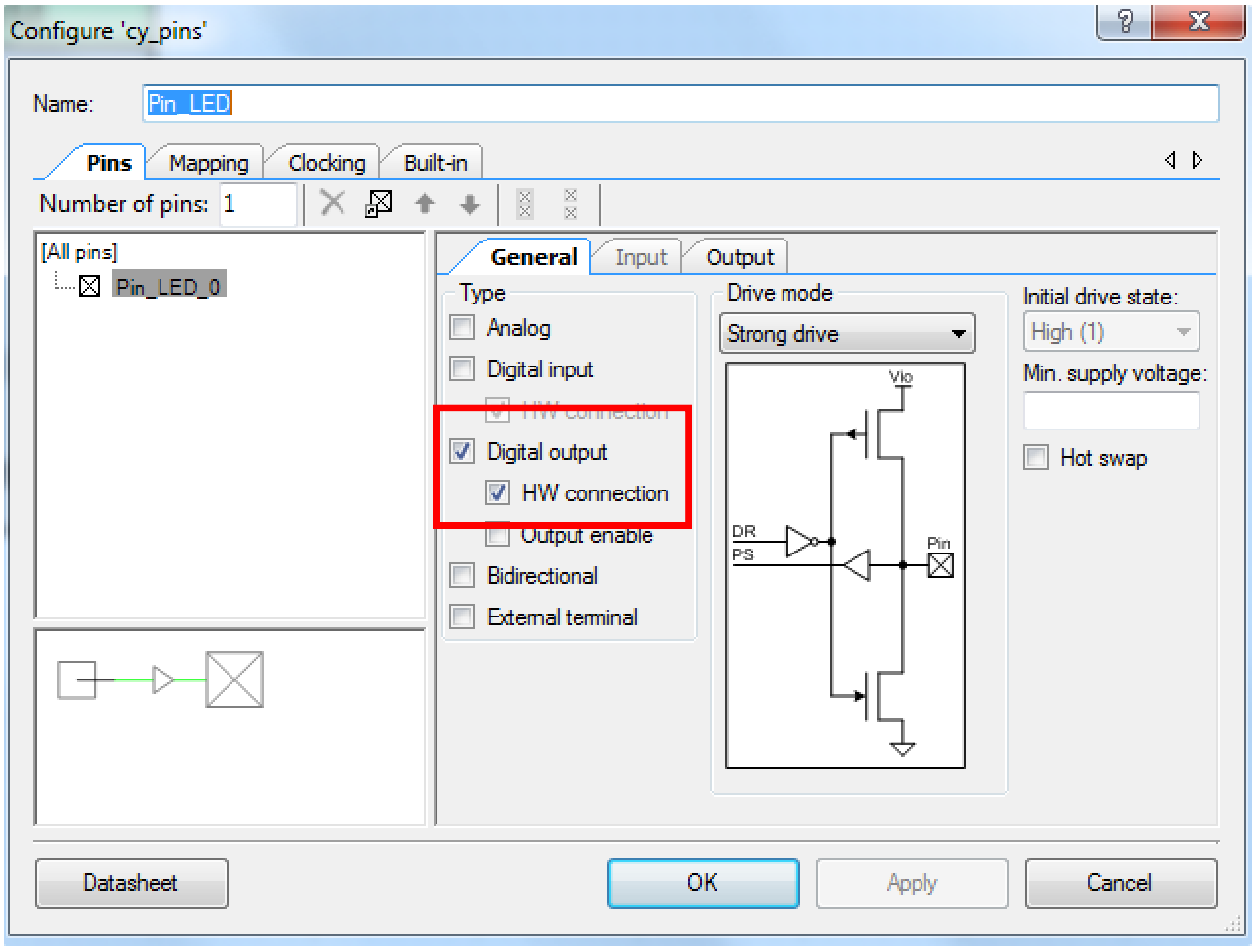
Interrupt
- [x] On terminal count
- [ ] On compare/capture count

| Input | Present | Mode |
|---|---|---|
| reload | ☐ | Rising edge |
| start | ☐ | Rising edge |
| stop | ☐ | Rising edge |
| switch | ☐ | Rising edge |
| count | ☐ | Level |

| | Register | Swap | RegisterBuf |
|---|---|---|---|
| Period | 2000 | ☐ | 65535 |
| Compare | 1000 | ☐ | 65535 |

Datasheet | OK | Apply | Cancel

**Figure 14: PWM Configuration**



Configure 'cy_pins'

Name: Pin_LED

Pins | Mapping | Clocking | Built-in

Number of pins: 1

[All pins]
Pin_LED_0

General | Input | Output

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

Drive mode: Strong drive

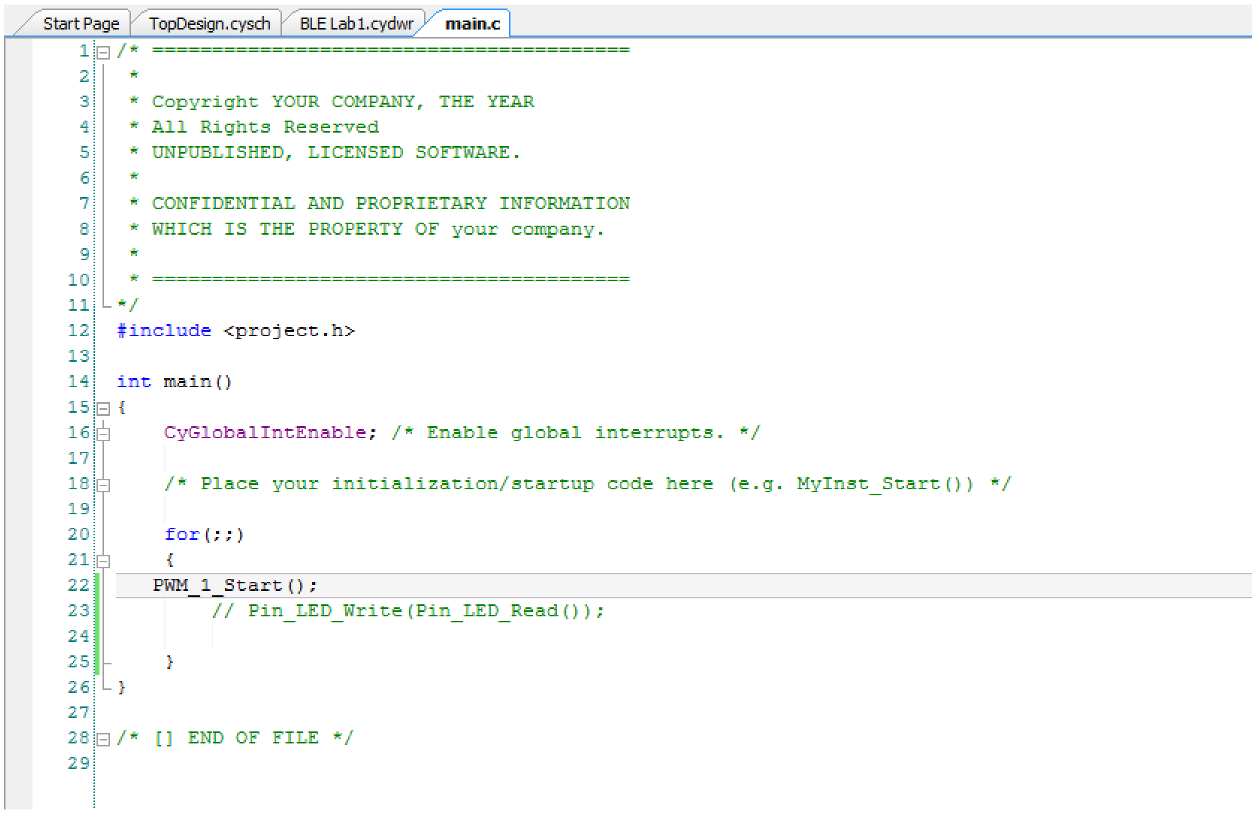
Initial drive state: High (1)

Min. supply voltage:

- [ ] Hot swap

**Figure 15: Pin Configuration**

4. The C code is also modified as demonstrated in Fig 16.



```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>

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

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

    for(;;)
    {
   PWM_1_Start();
        // Pin_LED_Write(Pin_LED_Read());

    }
}

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

**Figure 16: Code Editor Showing main.c**

5. Set PIN_LED to P3[7].

The results are demonstrated on fig 17.

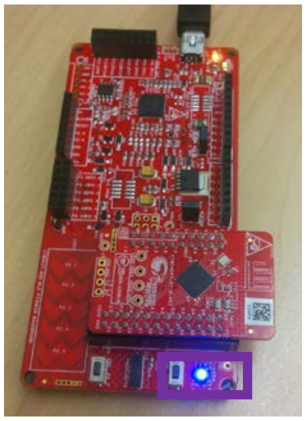


**Figure 17: Board Demonstration**

## CONCLUSIONS

From Lab 1, we learned how to work with PSoC creator as well as how to make the board “CY8CKIT-042-BLE”to work. In addition, the additional works on this lab gave me opportunity to work with PWM block.