# PSoC 4 Lab 2: LED Dimming Lab Manual

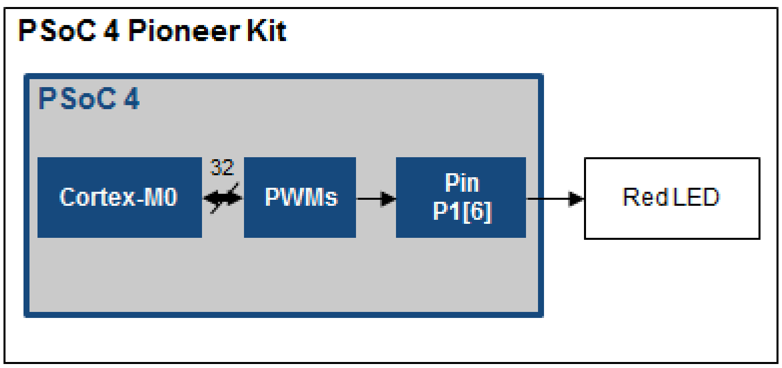
## Objectives

Modulate an LED on your PSoC 4 Pioneer Kit using a PWM Component

| Requirements | Details |
|---|---|
| Hardware | CY8CKIT-042 PSoC 4 Pioneer Kit |
| Software | PSoC Creator 3.0 or Newer |
| Firmware | PSoC4Lab 2 Template |
| Components used | PWM (TCPWM Block), Clock and Pin Components |

## Block Diagram

Figure 1: Lab 2 Block Diagram




## Theory

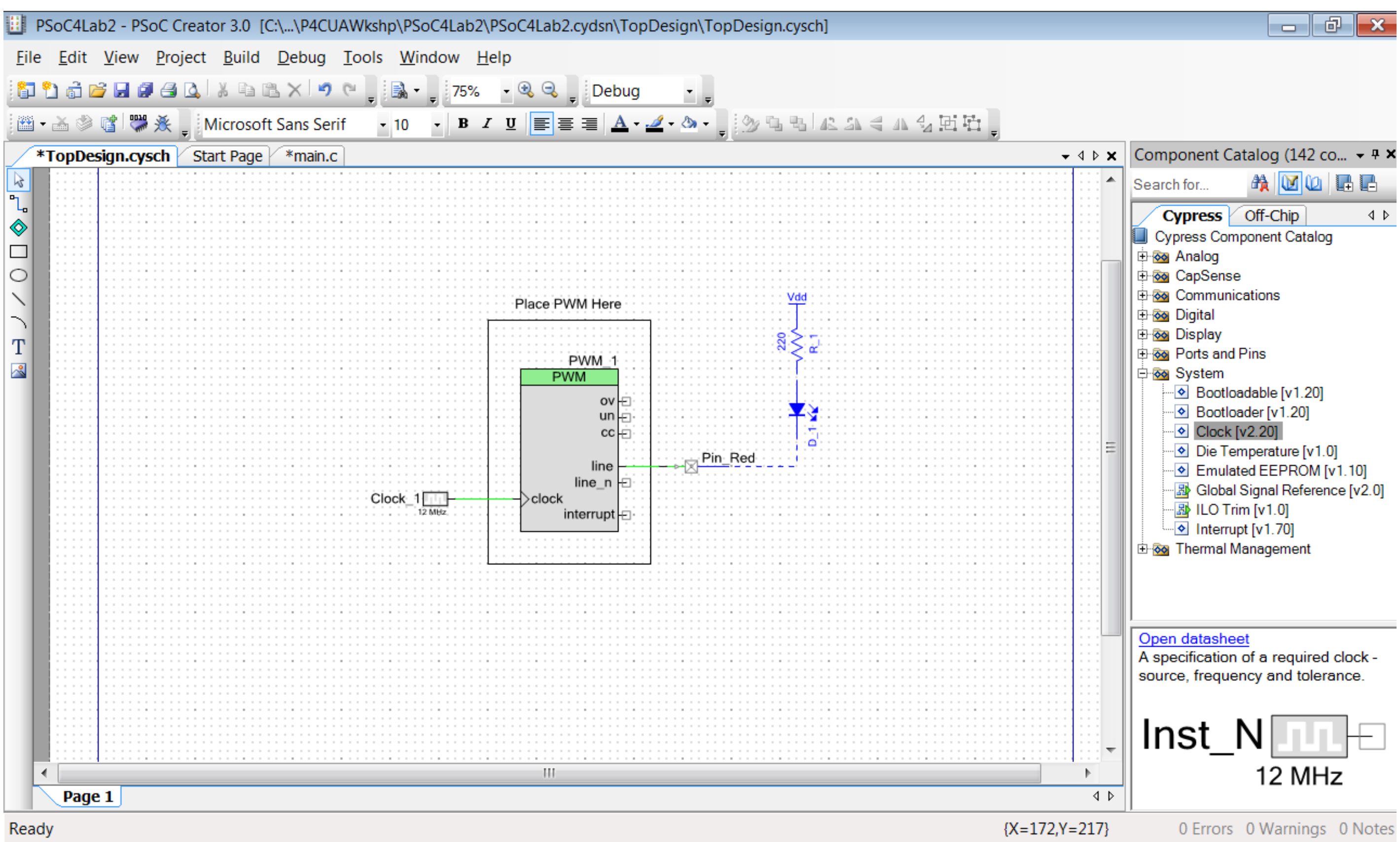
The goal of this lab is to learn the basics of the PSoC 4 fixed-function Timer/Counter/PWM (TCPWM) Components by implementing a PWM driven LED whose intensity varies over time. A project is created, Components are placed and configured, pins are assigned, and code is written. The project is then programmed onto the development board and observed.

The LED is connected to P1[6], and is lit when the pin sinks current. This means that driving the pin output low turns the LED on. To control the pin's behavior, we will use a PWM Component, which allows control of the dedicated TCPWM hardware using a Component Configuration tool and high-level Component APIs. The PWM's compare value is incremented by the Cortex-M0 CPU at regular intervals, resulting in LED intensity that changes smoothly over time.

**Procedure: Firmware**

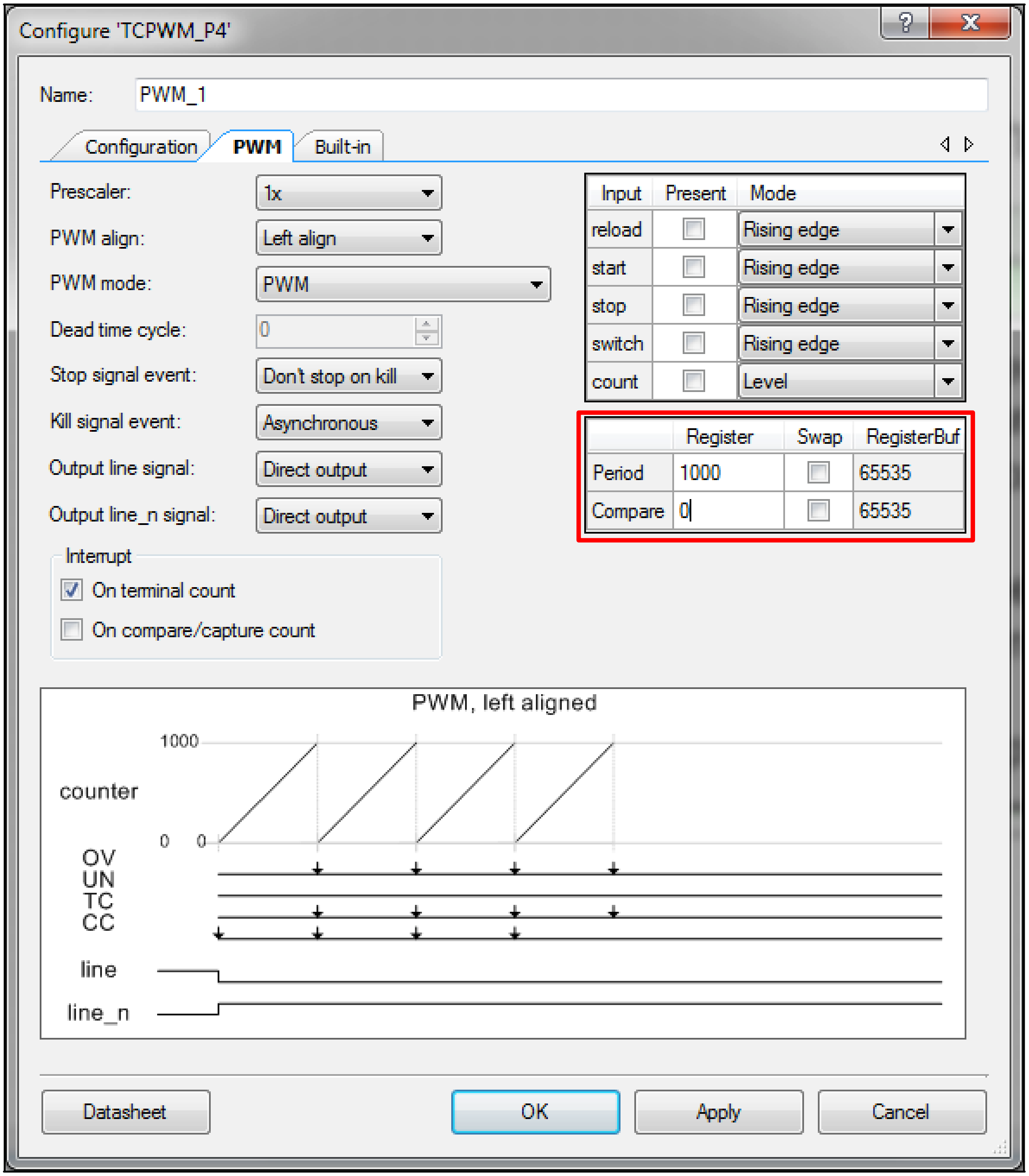
1. Close any open workspace files.
2. Open PSoC4Lab2 workspace by double-clicking on the “PSoC4Lab2.cywrk” file in the PSoC4Lab2 directory.
3. Open the project’s schematic by double-clicking on the “TopDesign.cysch” file in the Workspace Explorer. Note that in this schematic, we’ve included off-chip annotation Components to show how the red LED is connected to the pin.
4. Open the Pin_Red Component Configuration tool by double-clicking on the Component in the schematic. Check the “HW Connection” checkbox to enable hardware control of the pin. Click the “OK” button to apply changes and close the window.
5. In the Component Catalog, under the “Digital -> Functions” category, select the “PWM (TCPWM mode)” Component, and drag it into the schematic. Place it in the box labeled “Place PWM here.” At this point, please ensure to correctly line up the “line” terminal with the “Pin_Red” Pin Component.
6. In the Component Catalog, under the “System” category, select the Clock Component, and drag it into the schematic. Place it such that its output is connected to the clock input of the PWM. The result is shown in Figure 4.

Figure 4: Schematic With PWM and Clock Components Placed



7. Open the PWM's Component Configuration tool by double-clicking on it. Select the PWM tab to edit PWM-specific behavior. Change the "Period" value to 1000, and the "Compare" value to 0. Press the "OK" button. The PWM Configuration tool is shown in Figure 5.

Figure 5: PWM Configuration Window with Period and Compare Set



8. In the “Workspace Explorer”, double-click the “main.c” file to open it in the code editor.
9. Replace the “Change1” line with the PWM start API, shown in Code 1.

Code 1: Lab 2 PWM Start API Code

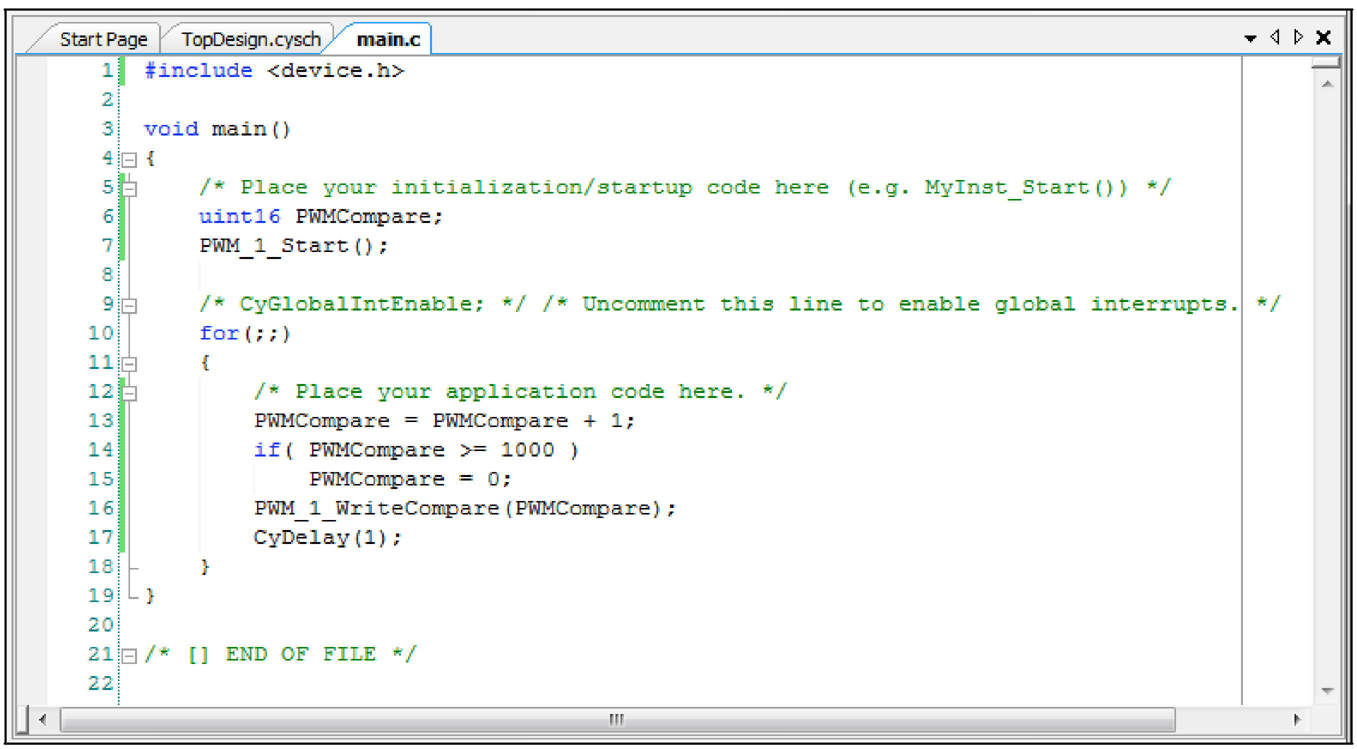
```
PWM_1_Start();   // Starts the PWM hardware block
```

10. Replace the “Change2” line with the PWM WriteCompare API, shown in Code 2. The entire “main.c” should look like that shown in Figure 6.

Code 2: Lab 2 PWM Write Compare API Code

```
PWM_1_WriteCompare(PWMCompare); //Updates PWM compare value
```

Figure 6: Lab 2 Solution “main.c”



11. Press the “Program” button on the PSoC Creator toolbar to build the project and program your kit. At this point, your red LED should start displaying a sawtooth-shaped brightness waveform at 1 Hz.

**Procedure: PSoC 4 Pioneer Kit Hardware Setup**

No hardware setup is required for this project. The red LED is connected to P1[6] with a copper trace.

**Conclusion**

You have successfully implemented variable duty-cycle PWM drive of the red LED on your kit. This technique can be applied to a number of different transducers. The TCPWMs may also be used for many other purposes, such as motor control or precise timing and counting of digital signals.

**Stretch Goals**

1. Implement a different brightness waveform.
    a. We have implemented a sawtooth waveform by incrementing the compare value until it rolls over.
    b. Try creating a triangle waveform by incrementing to a limit, then decrementing back down to zero.
    c. Try implementing a lookup table to create an arbitrary waveform like a sine wave.
2. Drive multiple LEDs to create a combination of colors.
    a. We are driving the red LED out of the tri-color array using P1[6]. The green LED is attached to P0[2]. The blue LED is attached to P0[3].
    b. Add two more PWMs to control the other colors, and vary their compare values along with that of the red LED to mix the colors together.

**Document Revision History**

| Revision | By | Description |
|---|---|---|
| ** | MAXK | First Release |
| *A | GUL | Updated formatting |
| *B | PKX | Updated for stand alone lab |