PSoC 4 BLE Lab #1: Blinking LED

Group: Gloria Bauman and Kervins Nicolas

**Due: September 27th, 2016**

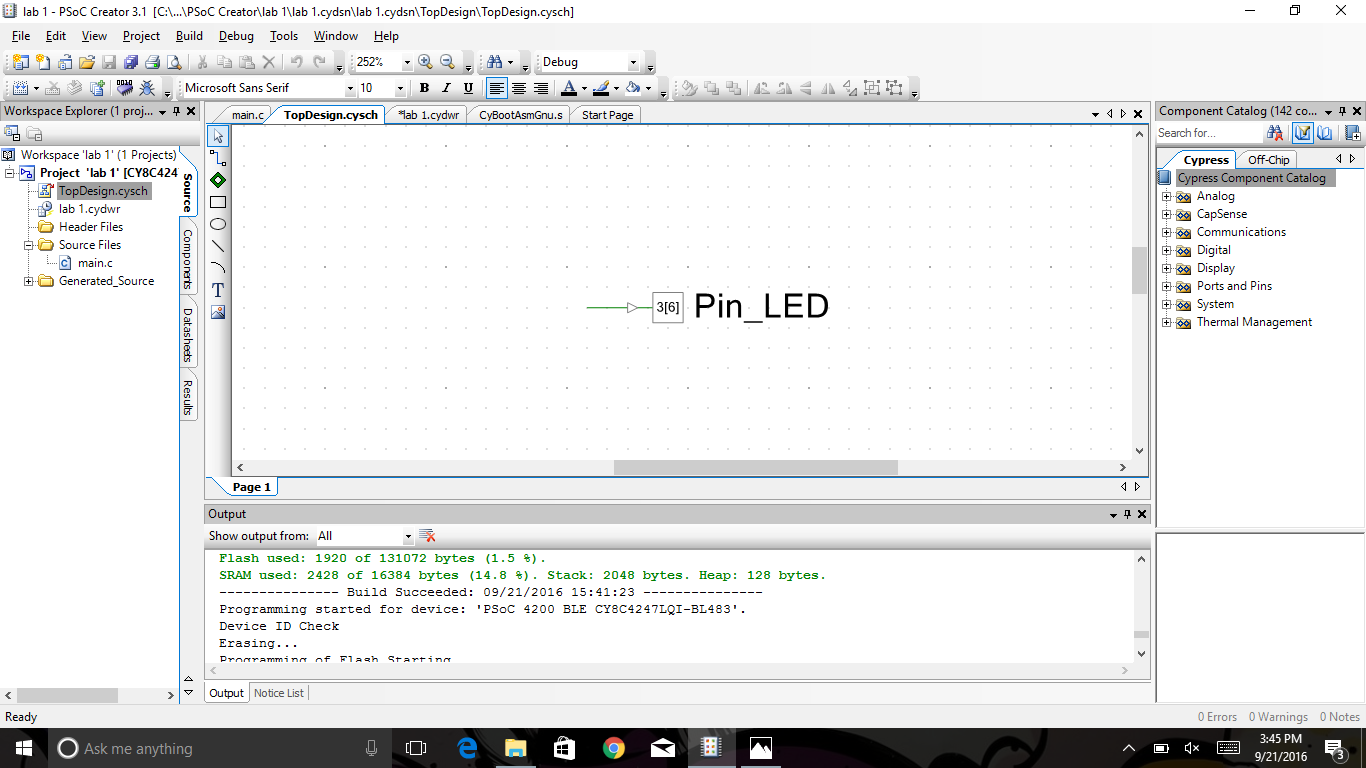
**Description**

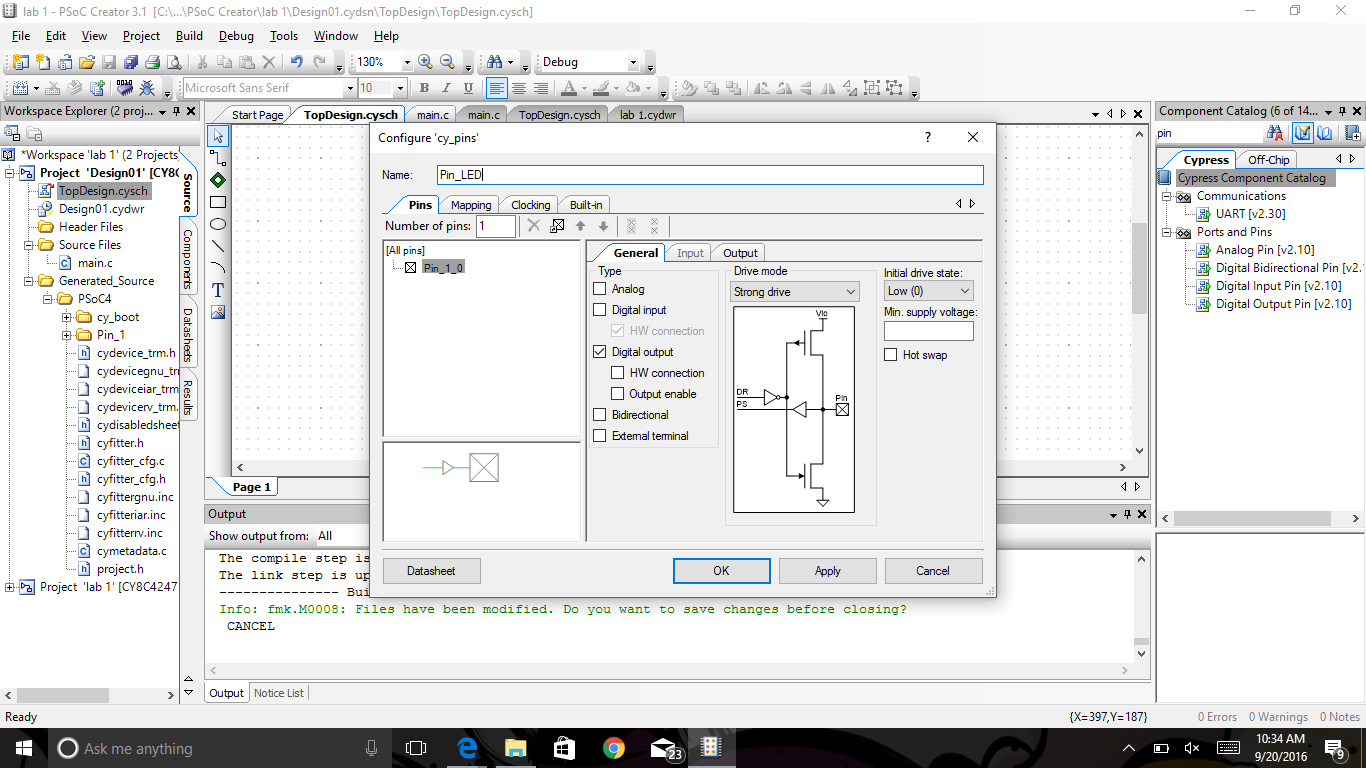
This Lab introduces you to the PSoC Creator IDE for developing and programming applications on the PSoC 4 BLE chip. It also introduces you to the Bluetooth Low Energy (BLE) Pioneer Kit.

**Objectives**

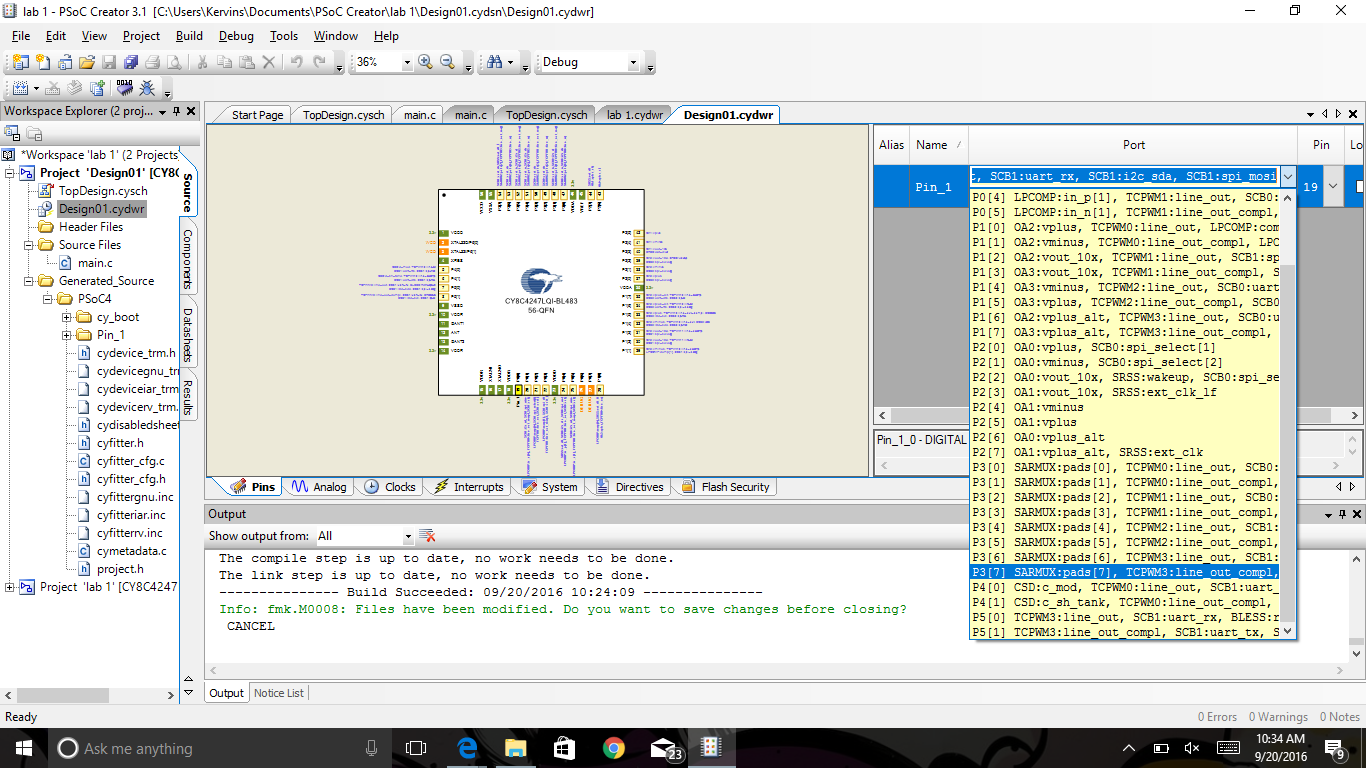
1. Learn how to use PSoC Creator to implement and debug PSoC designs
2. Implement a simple blinking LED design

**Process**

1. Opened a “New Project” and placed an “output\_pin” on the schematic diagram. Renamed the “Pin\_0” to “Pin\_LED.”
2. Configured “Pin\_LED” so that there is no hardware attached to it.

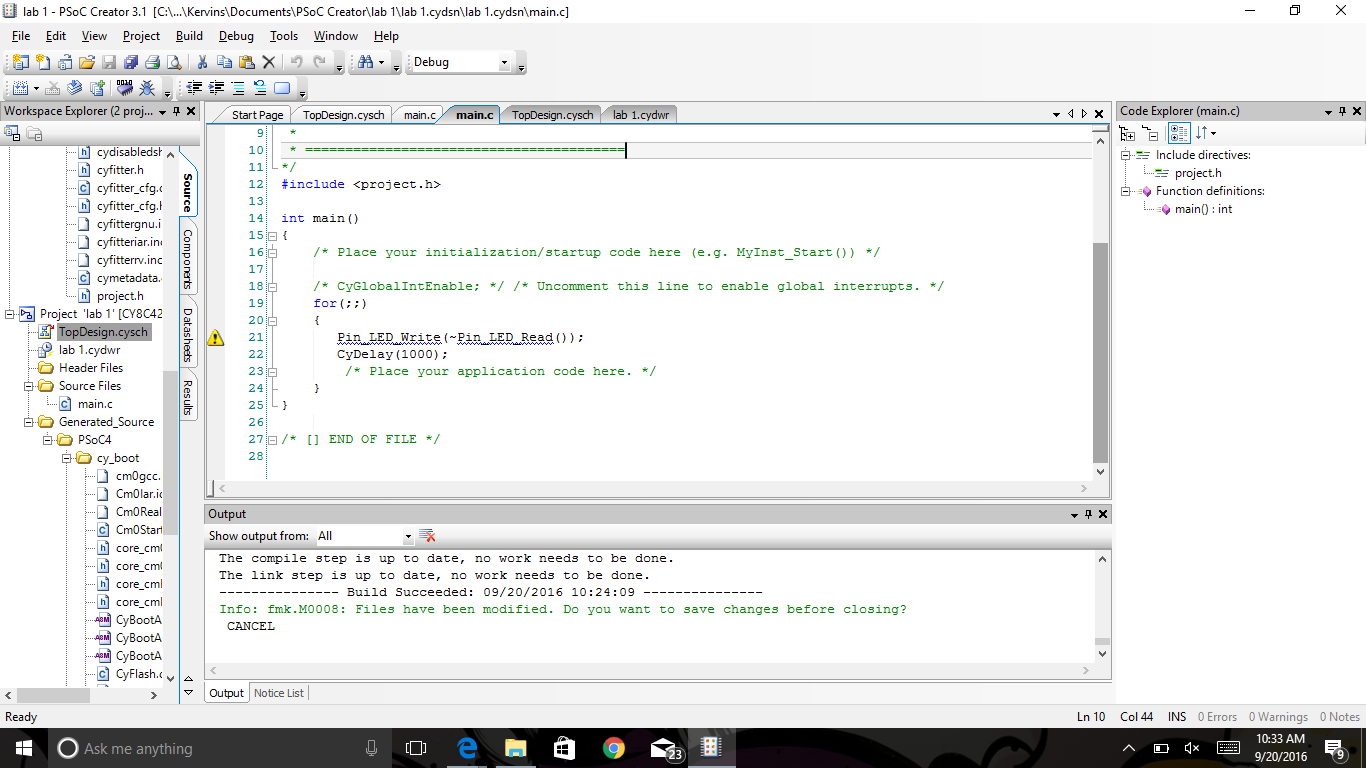


1. Assign “Pin\_LED” to ‘P3[7]” in the “Pins” tab.

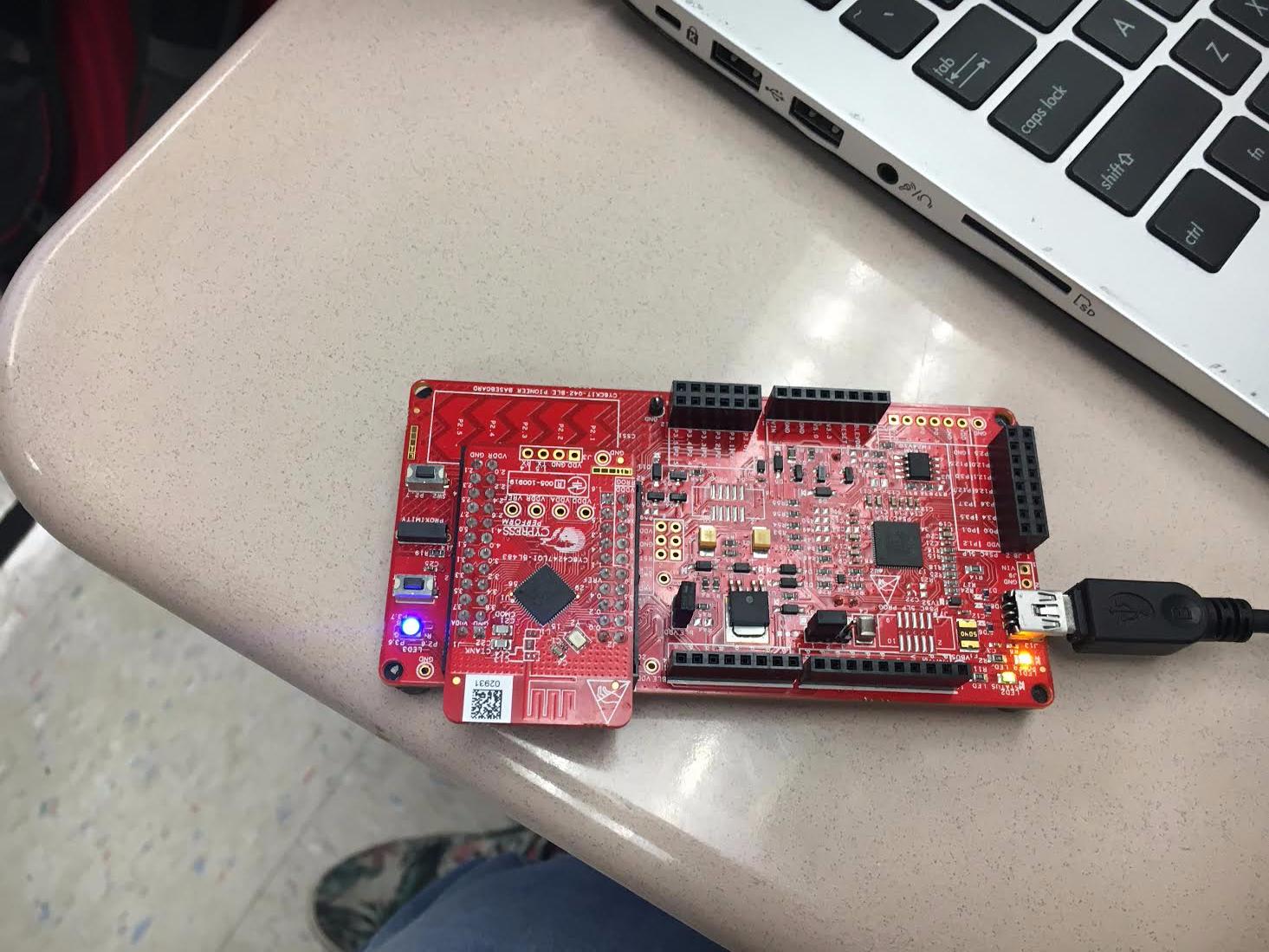
4) Open “main.c” to access the source file in the code editor. Then add in the code inside the “for(::)” 

“Pin\_LED\_Write(~Pin\_LED\_Read()); //Toggle Pin State”

“CyDelay(1000); //System Delay of 1 second”

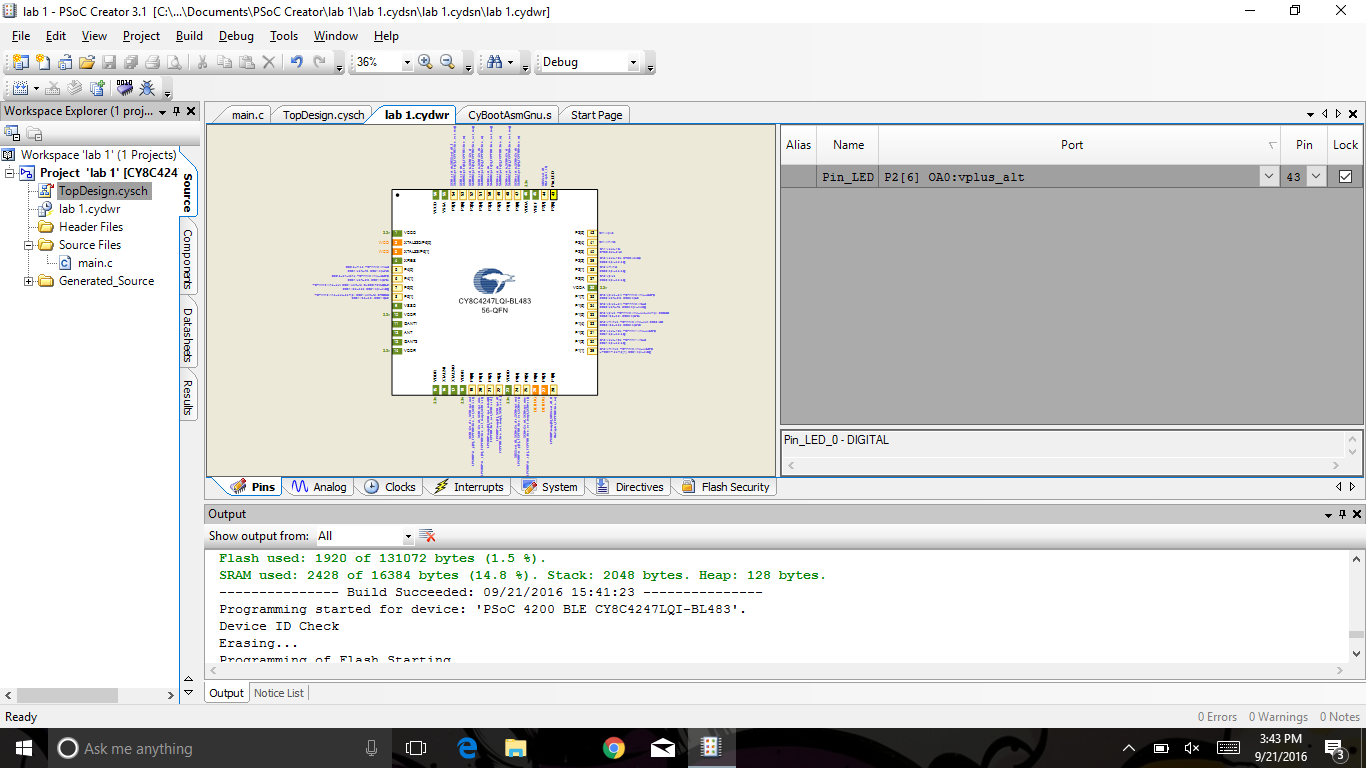


5) Build and Debug your program. After the program is debugged the blue LED on the kit blinks on and off every second.

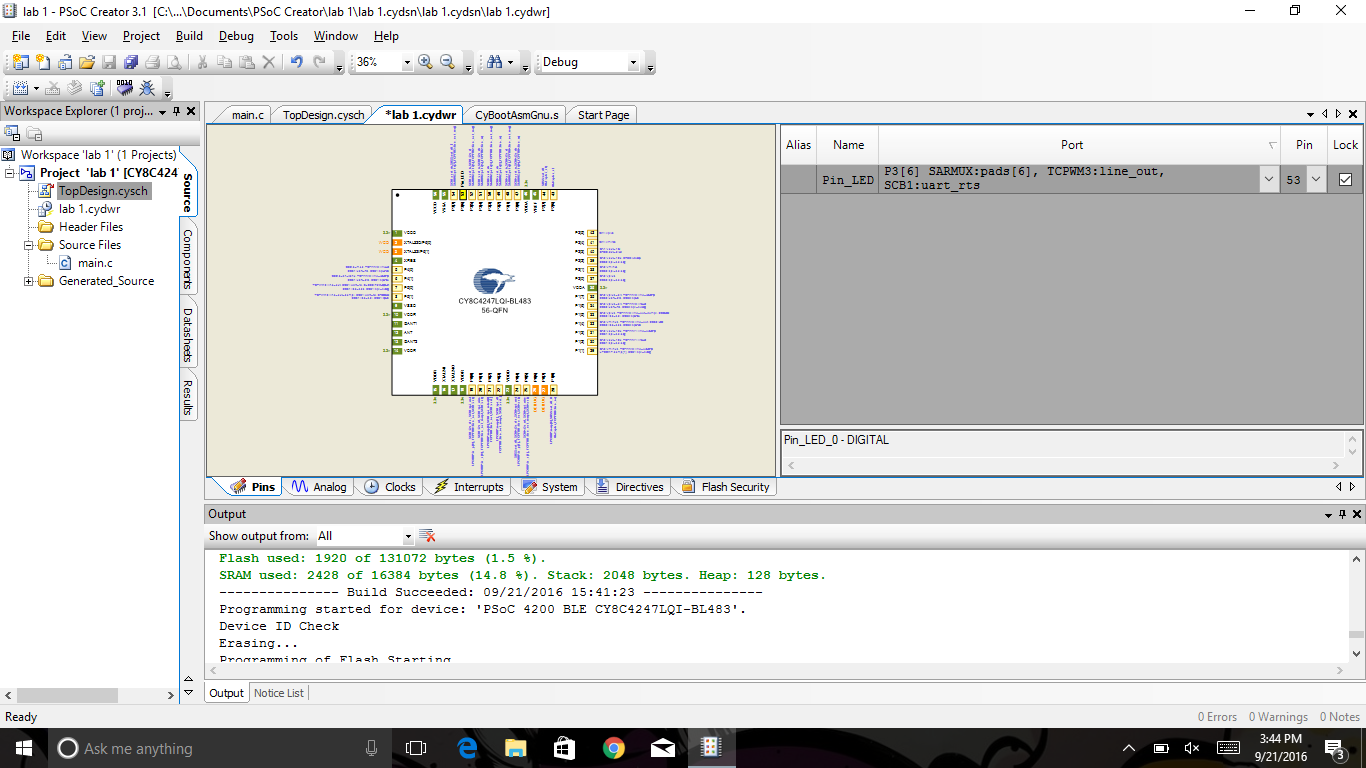


**Additional Exercises**

1. Instead of Blue LED, blink Red or Green LED.
   1. Hint: the LED pin-outs are printed on the board.

* Change the pin LED outputs







1. Instead of blinking the LED, generate a constant white color from the RGB LED

- Combine all of the pins together

- Created 3 pin: Pin\_LED

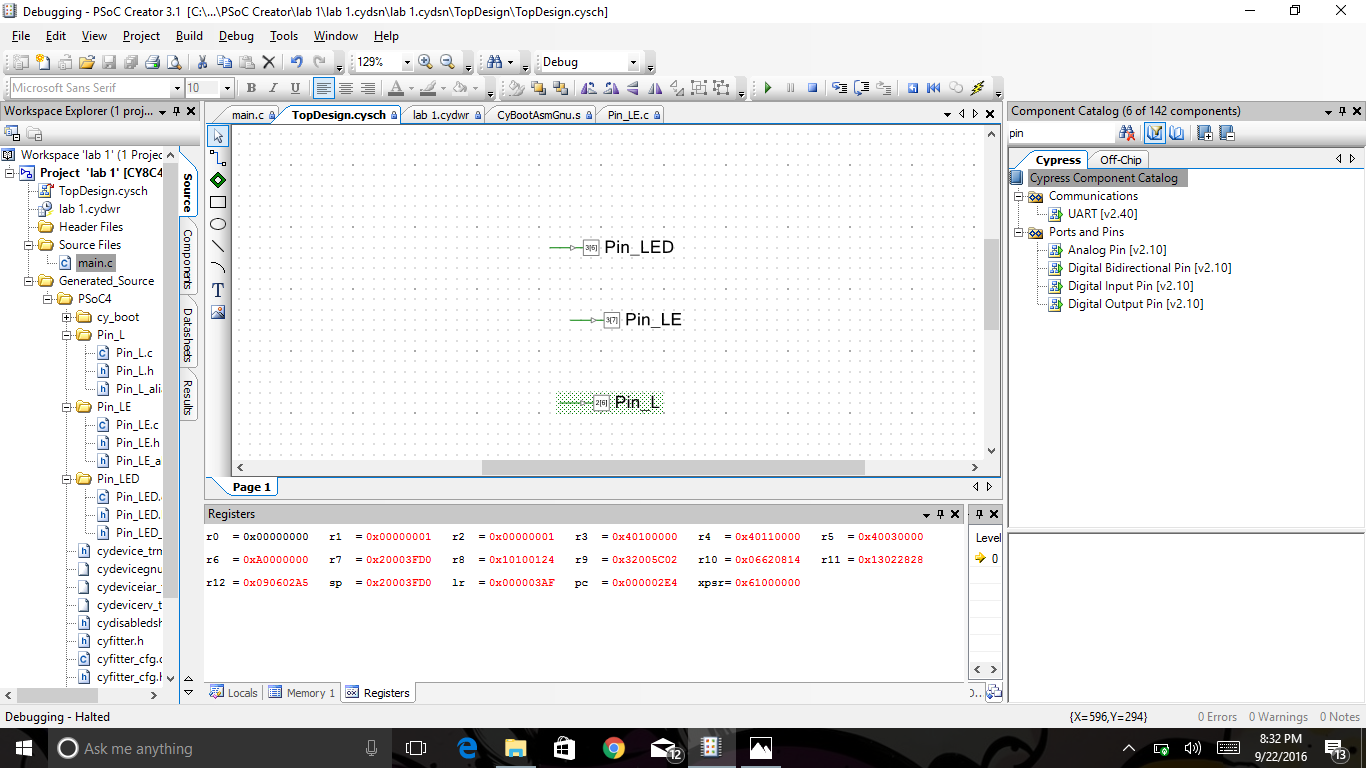
Pin\_LE

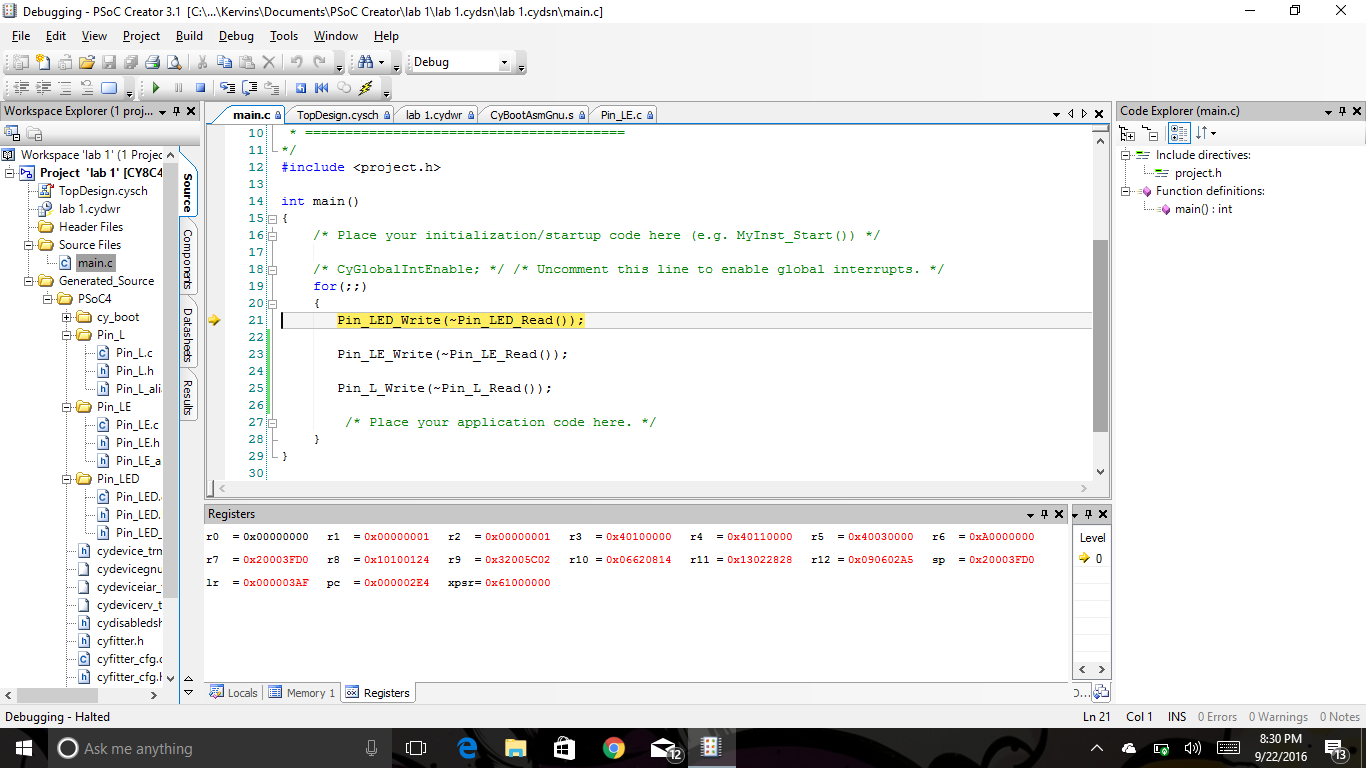
Pin\_L

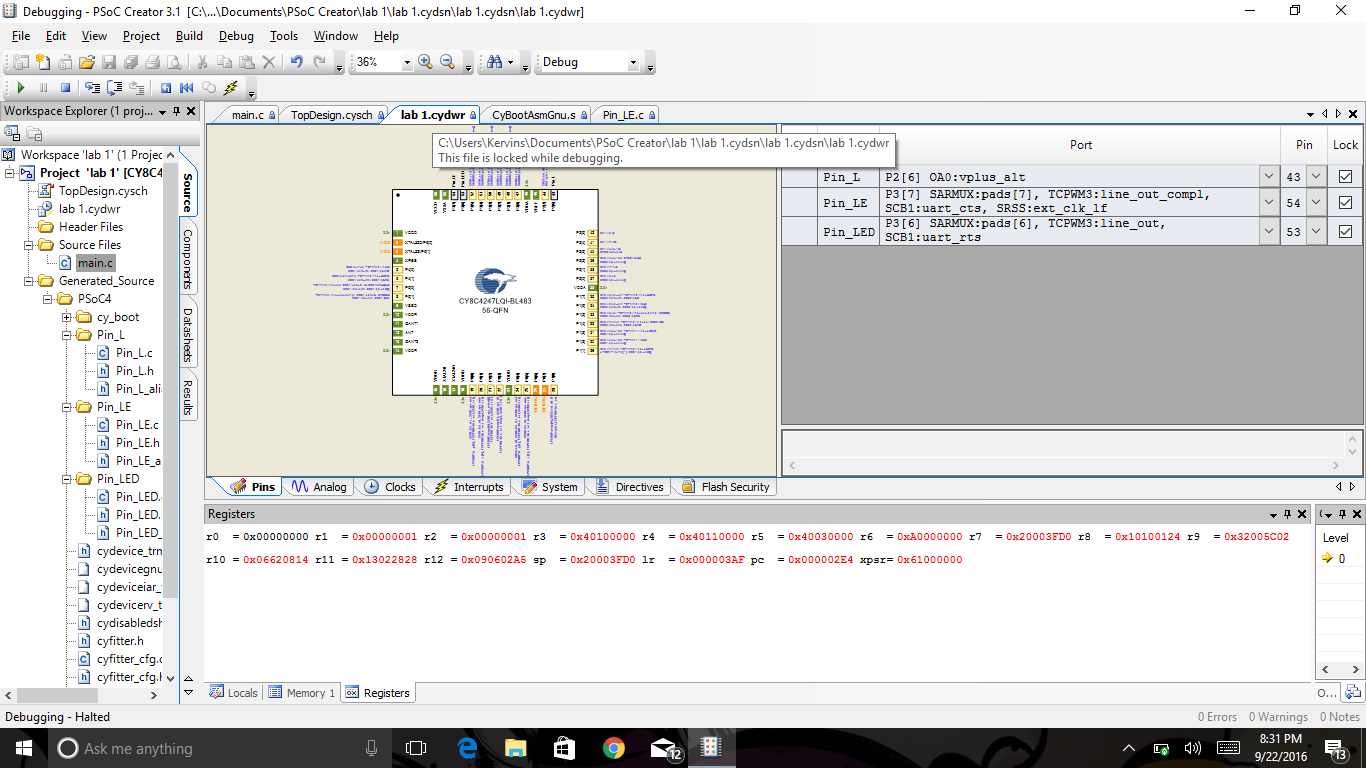
- Assigned the pins to a specific output

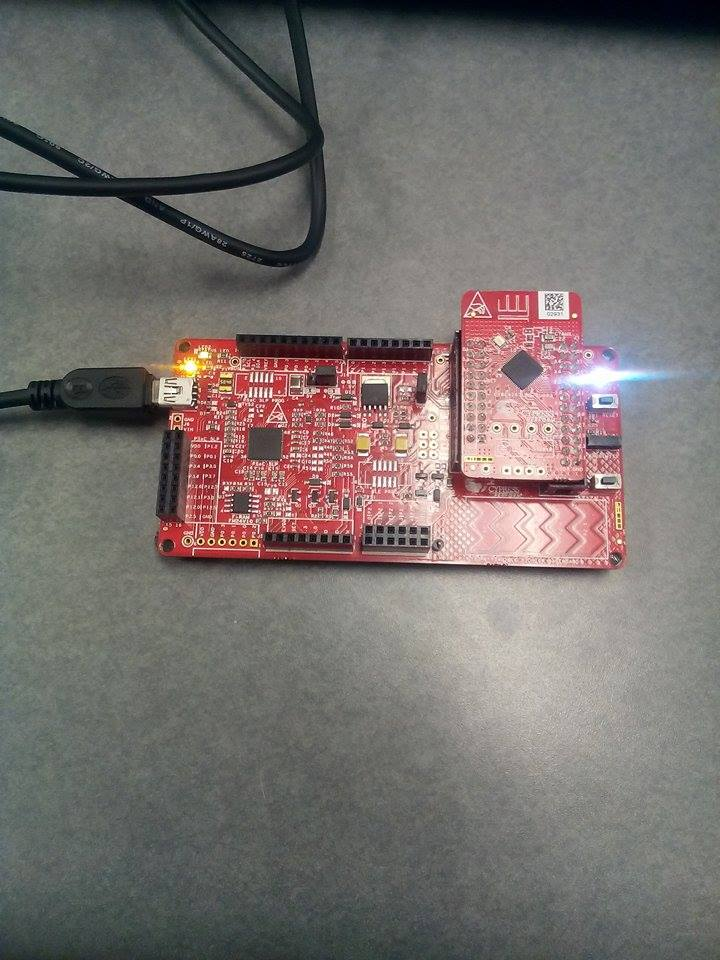
- Wrote code in main.c

Pin\_”name”\_Write(~Pin\_”name”\_Read());



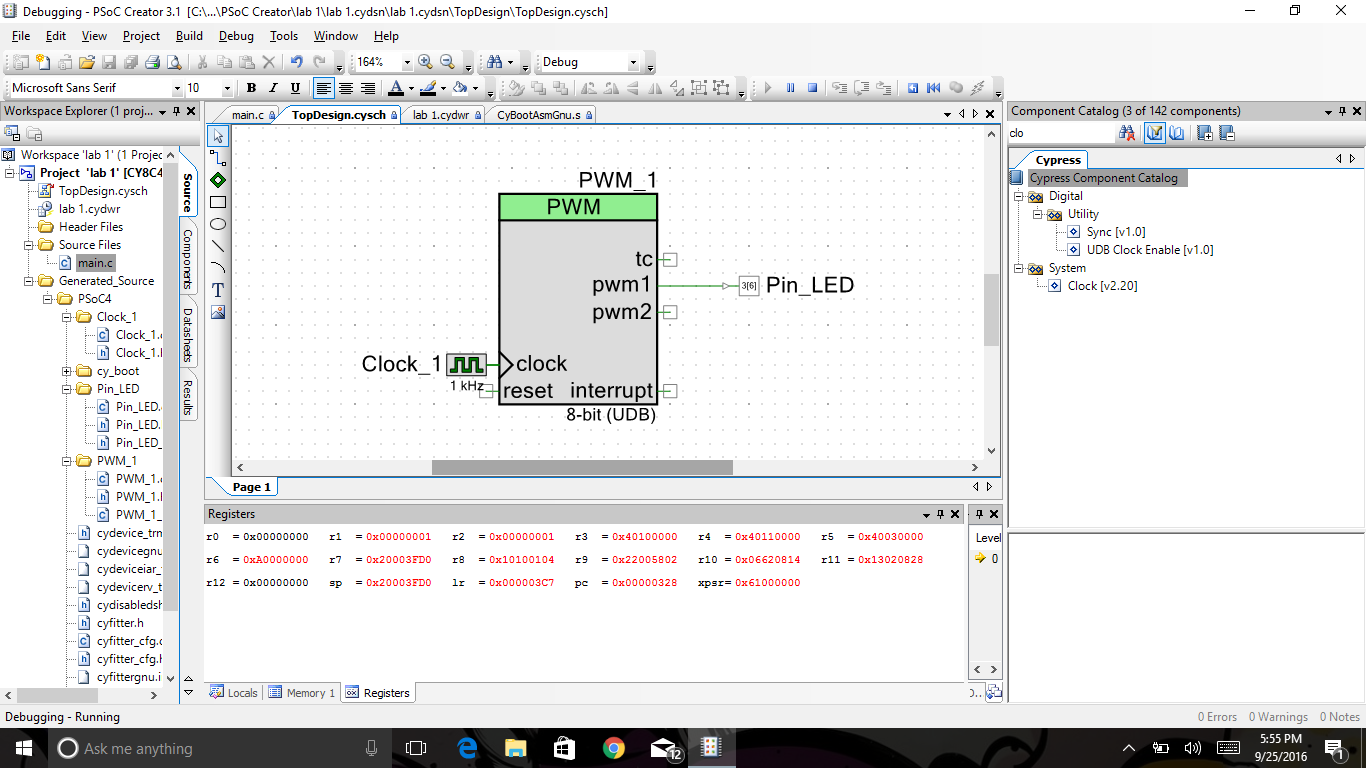




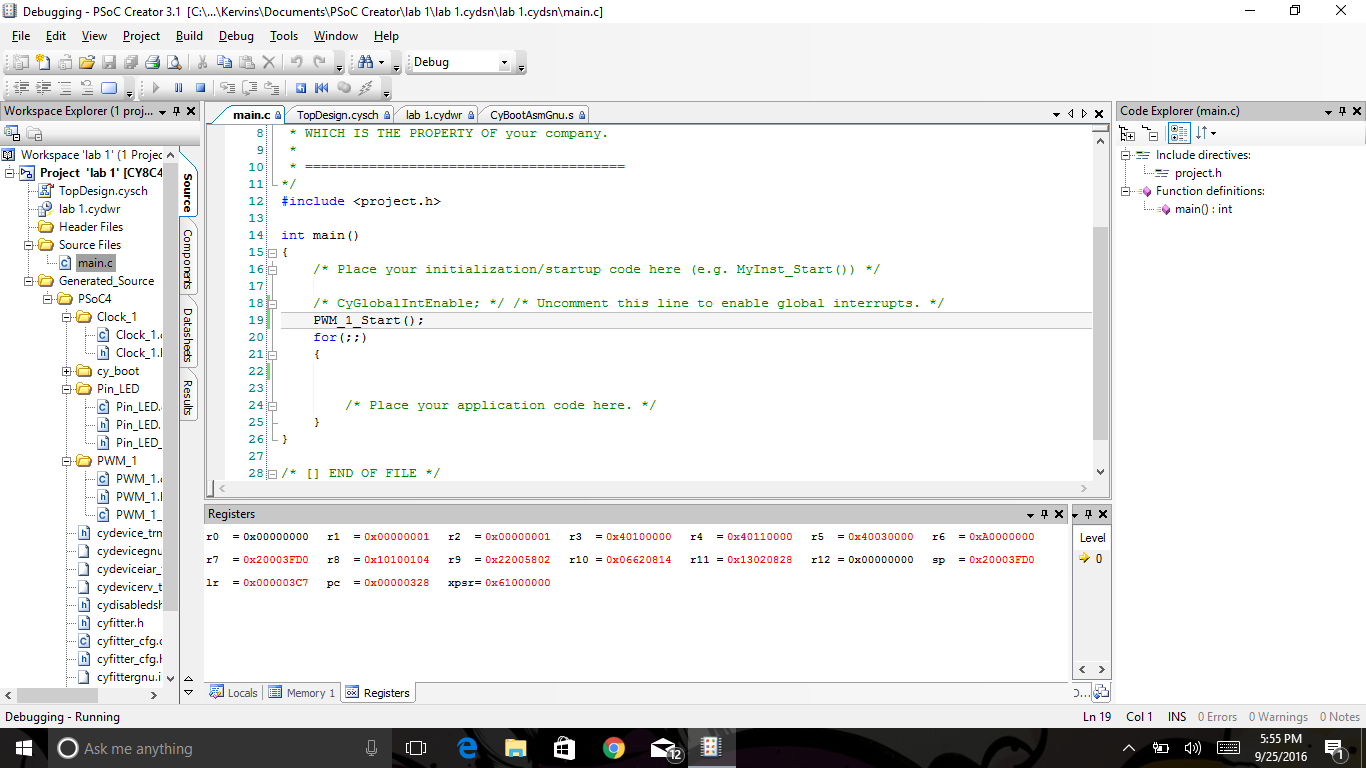


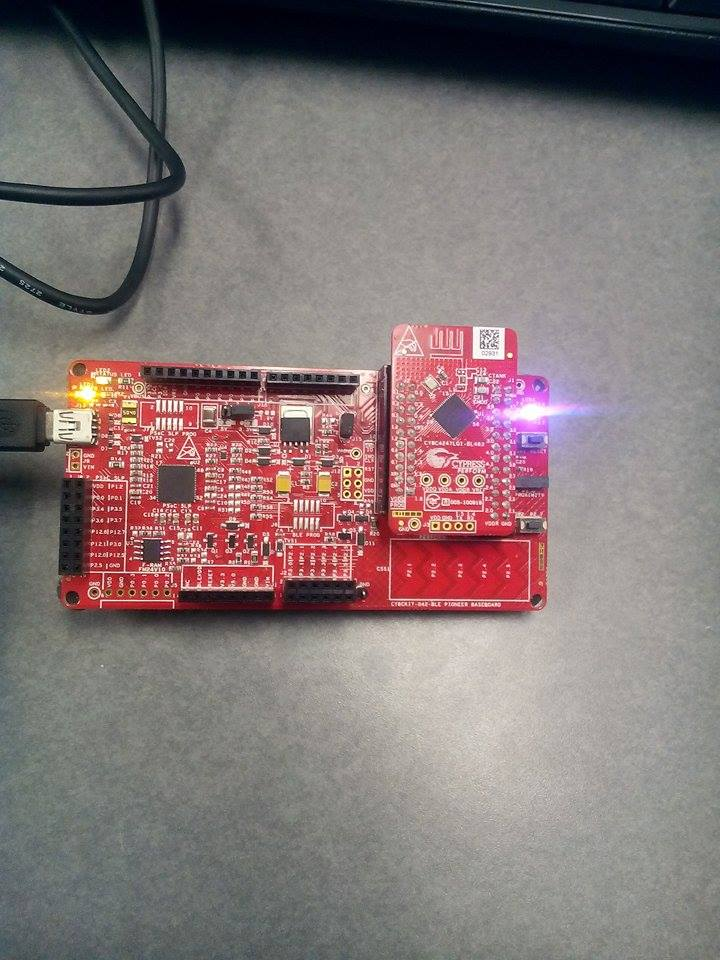
1. Instead of using firmware to blink the LED, use a PWM Component (hardware) to achieve the same result.
   1. Hint: Pick a PWM component from the Cypress Component Catalog. Read the Component data sheet for details on how to configure its parameters. Remember to start the PWM Component in main.c by using the <Component\_name>\_Start() API. Remember to enable the HW Connection on the pin Component so you can control it via the PWM instead of firmware.

* On the search PWM
* Place it on TopDesign.cysch
* Place a clock of 1 kHz and a Pin output
* In this case it was Pin\_LED



* Then finished by implementing the code in the main.c





Code:

/\* ========================================

\*

\* 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()

{

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

/\* CyGlobalIntEnable; \*/ /\* Uncomment this line to enable global interrupts. \*/

PWM\_1\_Start();

for(;;)

{

/\* Place your application code here. \*/

}

}

/\* [] END OF FILE \*/

Conclusion:

This is a basic project for beginner Psoc users to get familiar with the system. By learning how to implement pins into this project, we were able to configure the LED’s on the board into what we wanted. Basic coding for the project, debugging and building are all skills that we learned to use for upcoming laboratory work.