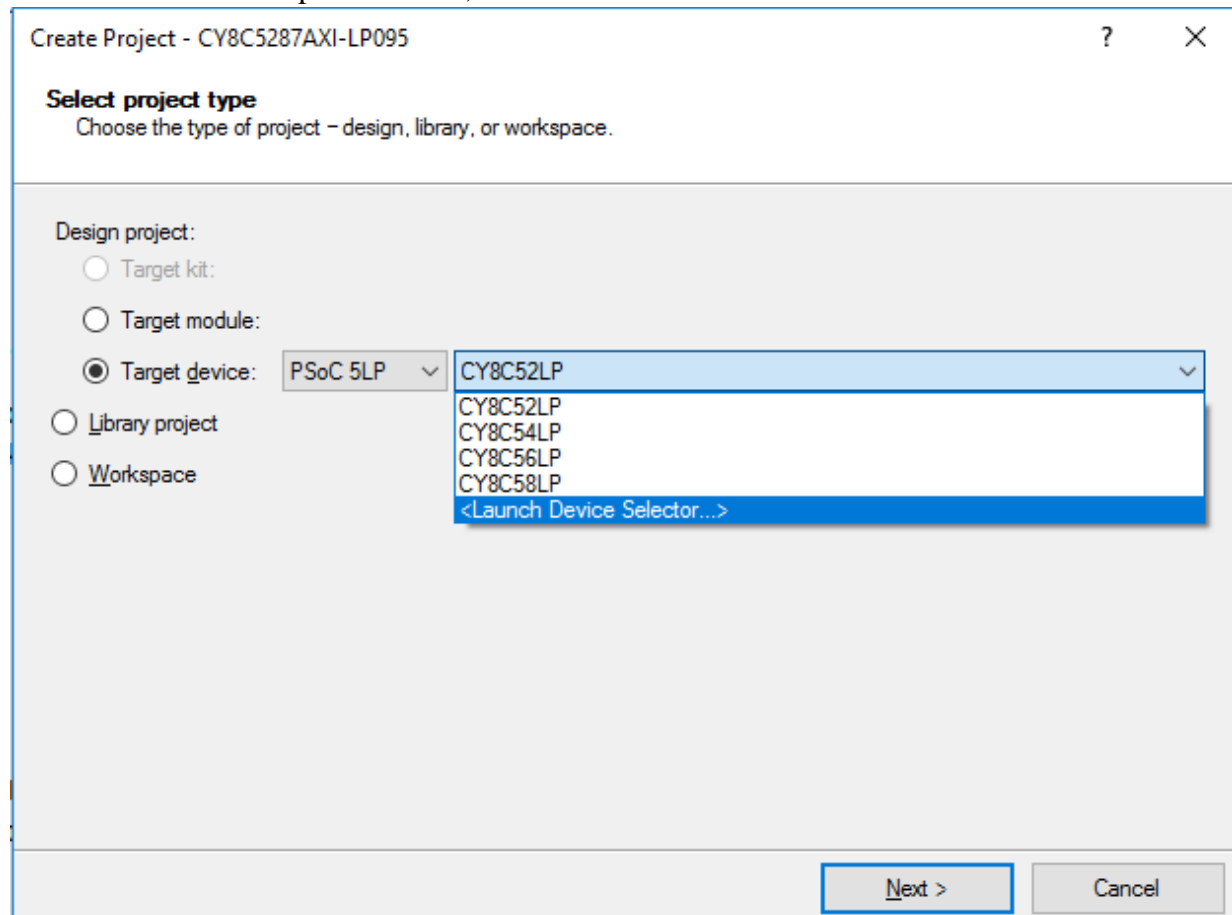


## ECE 381 – Lab 1a – PSoC5 Blinking LED

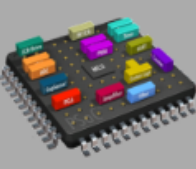
1. Launch PSoC Creator 3.3 from the start menu
2. Go to “File” → “New” → “Project”
3. Select the Target Device radio button and PSoC 5LP in the first dropdown menu
4. In the second dropdown menu, select <Launch Device Selector...>



5. Scroll down to the bottom and select “CY8C5888LTI-LP097” and hit OK

Device Selector ? ×

[View Datasheet](#) [Hide/Show Columns...](#) [Reset to Defaults](#)

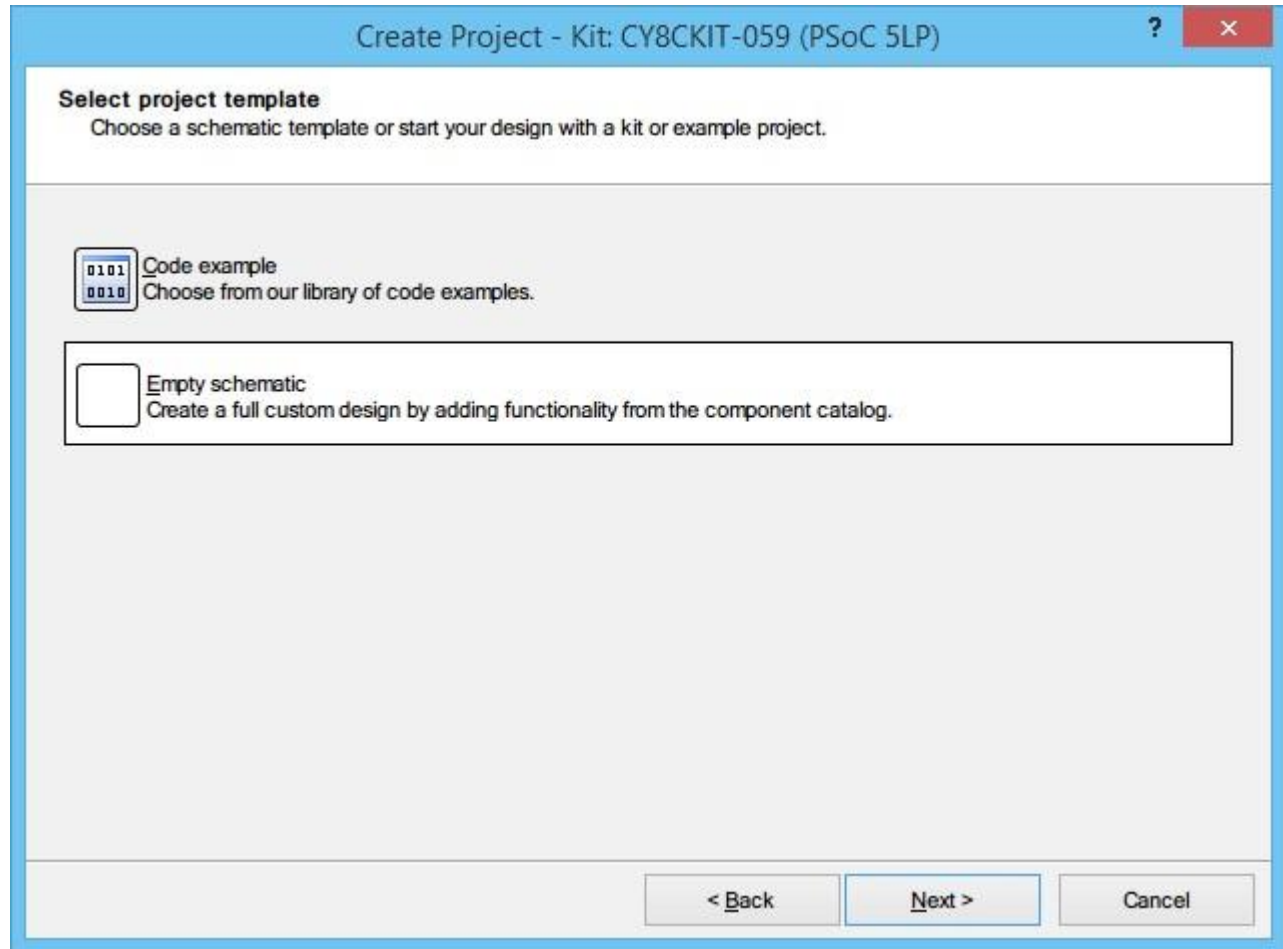


	CPU	Family	Series	Package	Max Frequency (MHz)	Flash (KB)	SRAM (KB)	EEPROM (KB)	IO	CapSense	Bluetooth	LCD Drive	Timer/Counter/PWM	Base Timer	Communication Blocks	UDB
Filters:																
CY8C5888AXI-LP096	ARM CM3	PSoC 5LP	CY8C58LP	100-TQFP	80	256	64	2	72	Y	-	✓	4	-	1	24
CY8C5888AXQ-LP096	ARM CM3	PSoC 5LP	CY8C58LP	100-TQFP	80	256	64	2	72	Y	-	✓	4	-	1	24
CY8C5888FNI-LP210	ARM CM3	PSoC 5LP	CY8C58LP	99-WLCSP	80	256	64	2	72	Y	-	✓	4	-	1	24
CY8C5888FNI-LP214	ARM CM3	PSoC 5LP	CY8C58LP	99-WLCSP	80	256	64	2	72	Y	-	✓	4	-	1	24
<b>CY8C5888LTI-LP097</b>	<b>ARM CM3</b>	<b>PSoC 5LP</b>	<b>CY8C58LP</b>	<b>68-QFN</b>	<b>80</b>	<b>256</b>	<b>64</b>	<b>2</b>	<b>48</b>	<b>Y</b>	<b>-</b>	<b>✓</b>	<b>4</b>	<b>-</b>	<b>1</b>	<b>24</b>
CY8C5888LTQ-LP097	ARM CM3	PSoC 5LP	CY8C58LP	68-QFN	80	256	64	2	48	Y	-	✓	4	-	1	24

< 65 of 65 devices found [Clear Filters](#) >

**OK** **Cancel**

6. Create an Empty schematic and click Next



7. Select “Create new workspace” from the drop down, change “Workspace name:” to “blink\_led” , “Location:” to “Desktop\ECE381\_Labs” (Create if it doesn’t exist), and “Project name:” to “blink\_led” and click Finish.

**Create Project - Kit: CY8CKIT-059 (PSoC 5LP)**

**Create Project**  
Choose a name and location for your design.

Workspace: Create new workspace

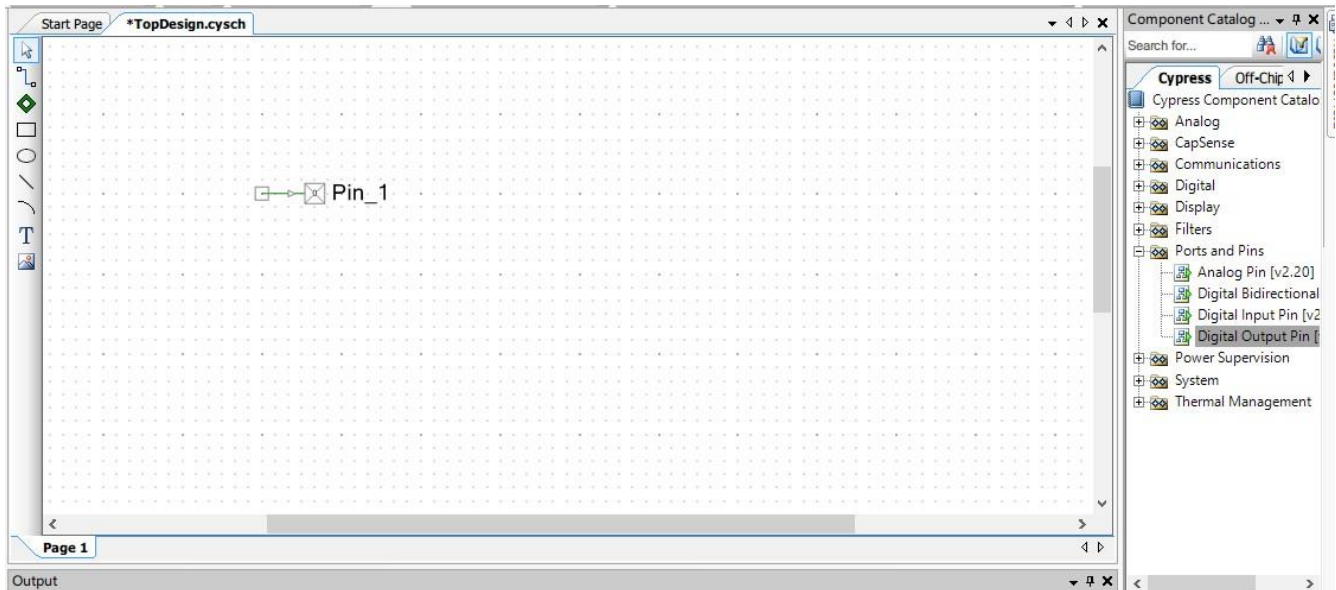
Workspace name: blink\_led

Location: C:\Users\tyork\Desktop\ECE381\_Labs ...

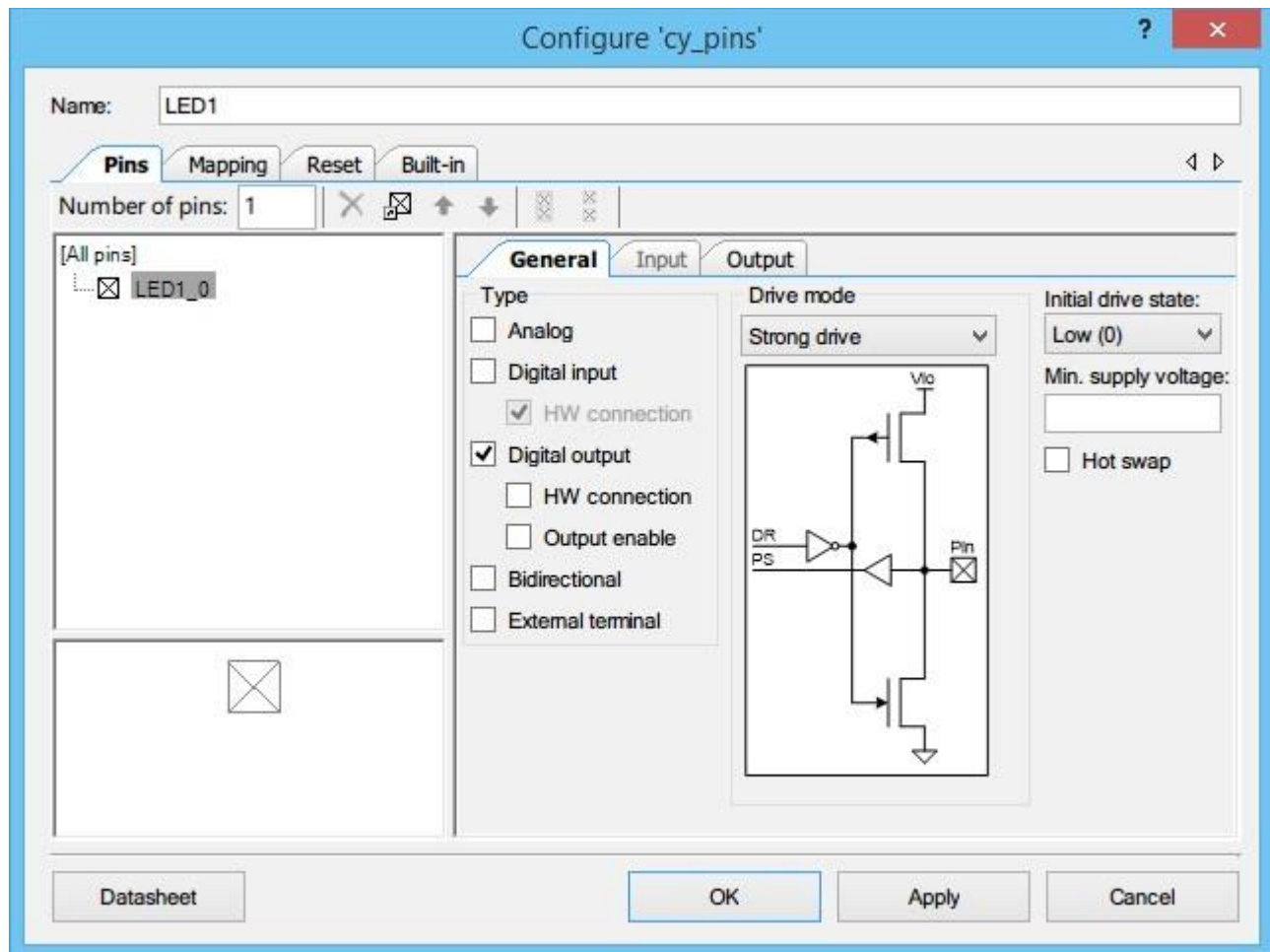
Project name: blink\_led

< Back Finish Cancel

8. You should now see the main screen of the PSoC project. The schematic window is for placing User Modules. Since we want to blink and LED, we need to place a Digital Output Pin User Module. Click on “Ports and Pins”, click on the “Digital Output Pin” to highlight it, and drag it over to the schematic window. After placing it, it is kind of small, so you can zoom in by pressing “CTL++” (Control Key and + key at same time).

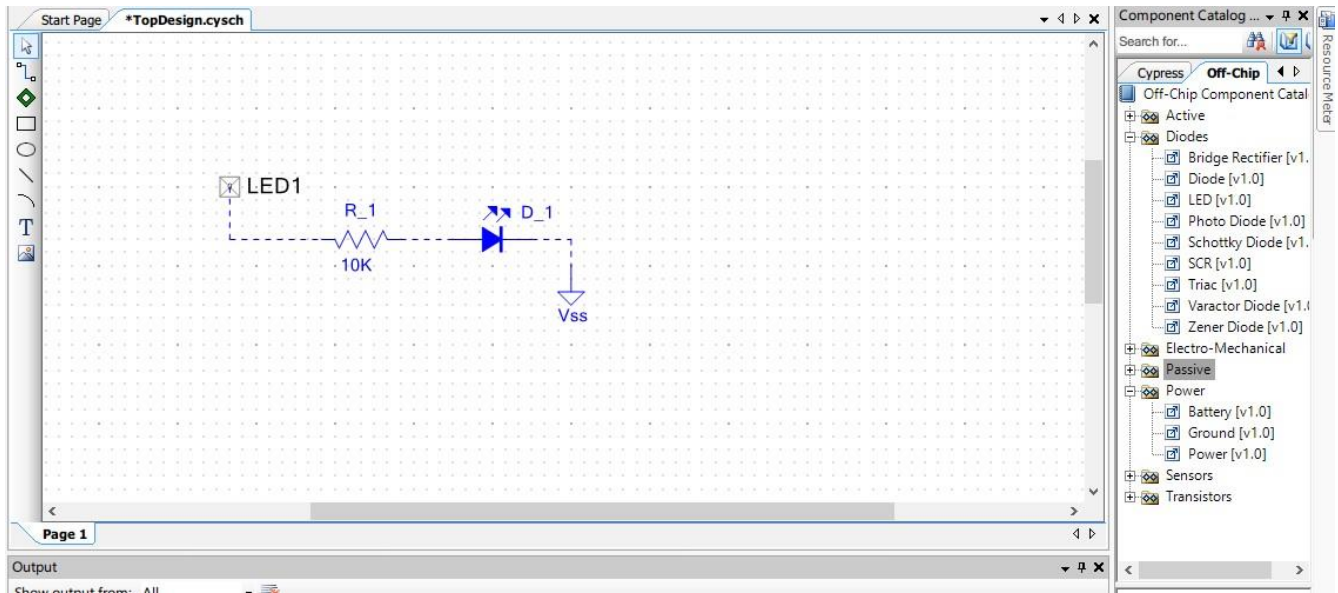


9. Now, right click on Pin\_1 and click “Configure”. This brings up a dialog that allows you to configure various aspects of the pin. The most important option is the “Drive mode”. This should be set to “Strong drive” if you want the pin to be an output. The “Digital output” box should also be checked. The other options that are relevant for this project are under “Digital output”. “HW Connection” allows the pin to be driven by another User Module from the schematic, and “Output enable” adds a controllable output enable pin to the actual LED1 pin. We will use these features later on the course, but for now, since we will drive LED1 straight from the C code, we will uncheck “HW Connection”. Your final configuration should look like this:

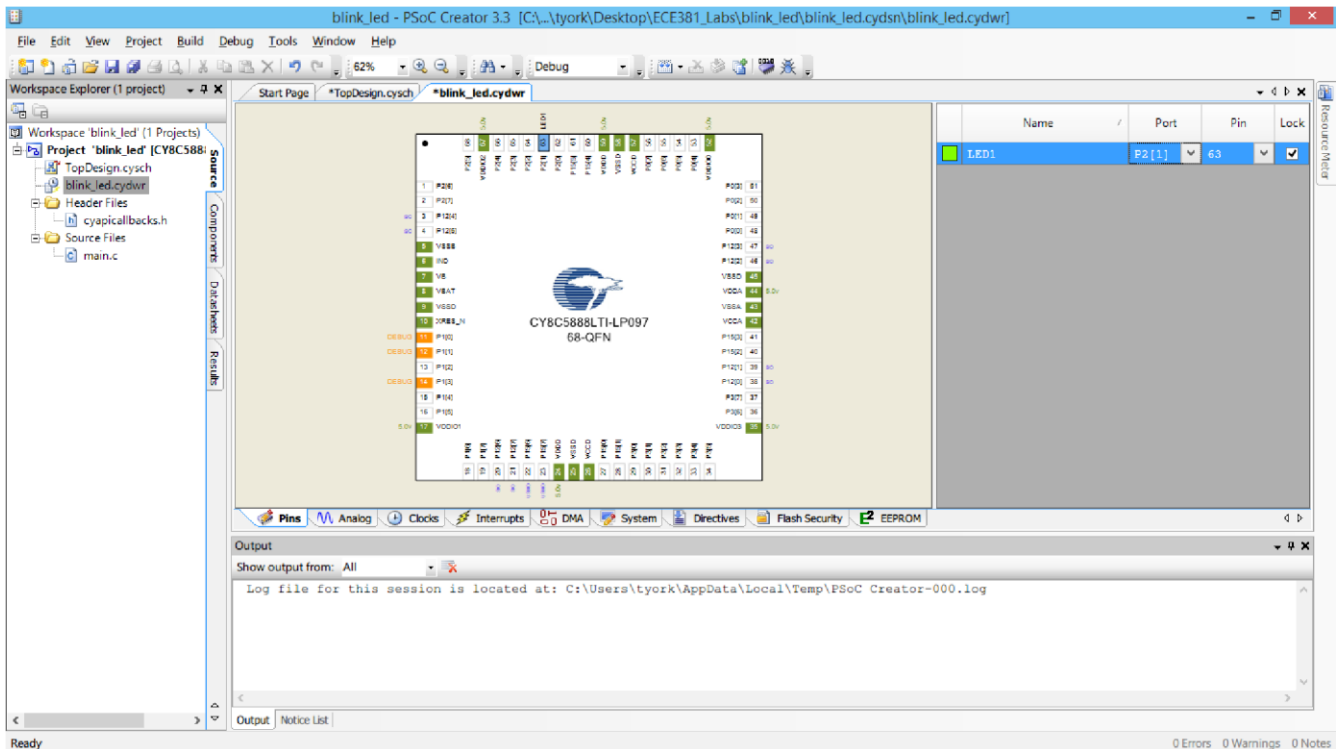


10. Click OK and the schematic screen should only show the pin now with the name LED1. If you want, and this is strictly optional, you can show the external devices connected to the pin. Under the “Component Catalog” frame, click on the “Off-Chip” tab. You can add a resistor

from “Passive”, and LED from “Diodes”, and a Ground from “Power”. Pressing the w key brings up the wire drawing tool to connect them. Again, connecting external components is **strictly optional** and has no impact on the building and compilation of the project. They are only there to provide a clearer schematic and project.

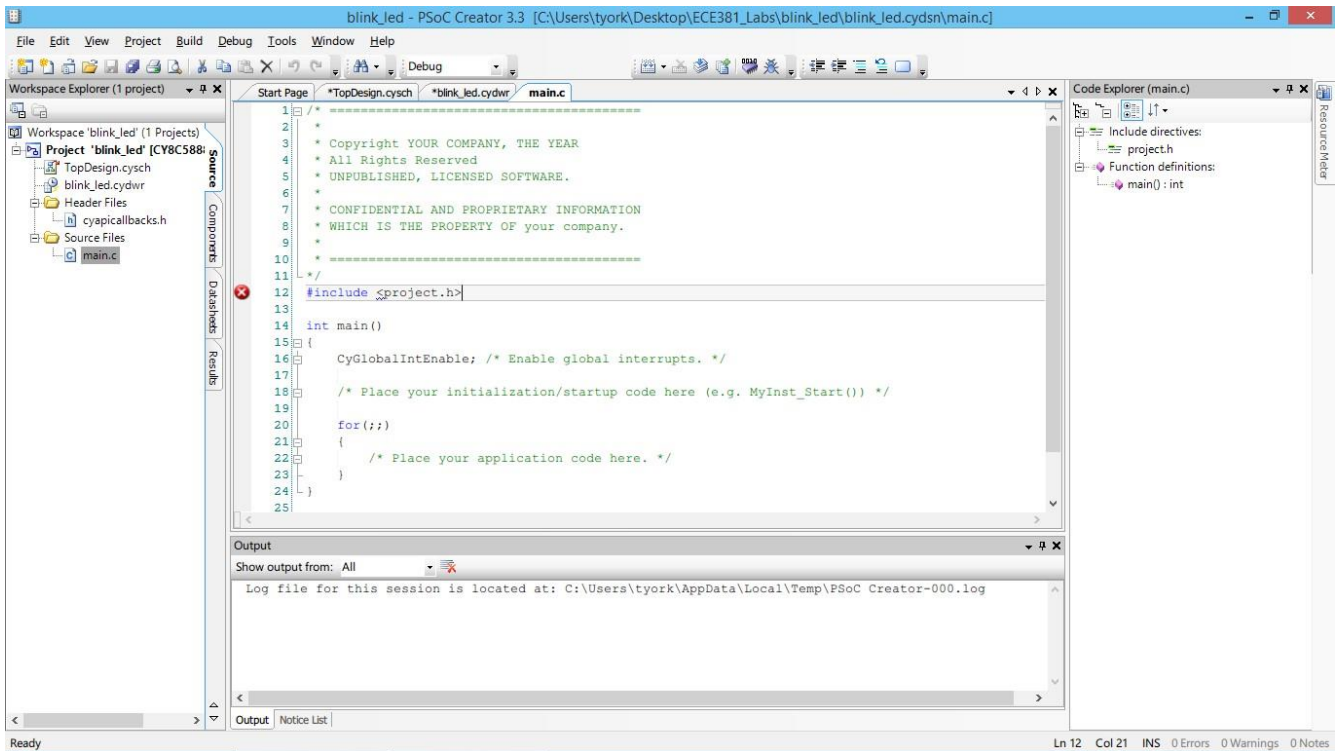


- Next, we need to assign a physical pin to the digital output pin named LED1 that we have placed. In the Window Explorer frame, double click on the “blink\_led.cydwr” file. This should bring up the hardware configuration view for the actual CY8C5888LTI-LP097 chip used in our kit. Under the Pins tab, it will list all of the pins that have been placed in the schematic, so you should see one named LED1. Since we will use the on board LED, we will want to use Pin 1 on Port 2. Clicking on the Port should bring up a drop down menu that displays all pins on the chip as well as any special use for them. Select P2[1] and the fields for the Pin and click the lock tab.





12. That is all of the hardware configuration necessary for this project. We will now go on to writing the program for blinking the LED. On the Workspace Explorer frame, double click “main.c”. This should open up the code editor tab, and display a template for your code. If you see a red X next to the include, that is fine, we haven’t build anything yet.



13. Now, you can write the code as you see fit. However, if you want features like autocomplete for components you placed and named, it would be a good time to build it so that all of the backend files get created. Click “Build” → “Build blink\_led” (or press Shift+F6) to build the project. It might take a bit, but you should see a “Build Succeeded” message in the Output frame at the bottom of the screen.

14. We can now program the blinking LED. There are two ways to do this, and I will illustrate both:

```
/* =====
 *
 * ECE 381 Spring 2017
 * Lab 1a: Blink an LED
 *
 * Dr. Timothy York
 * 24 August 2016
 *
 * =====
 */
#include <project.h>

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

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */    int
    ix = 0; //Declare variables first. This one is for holding a counter value.

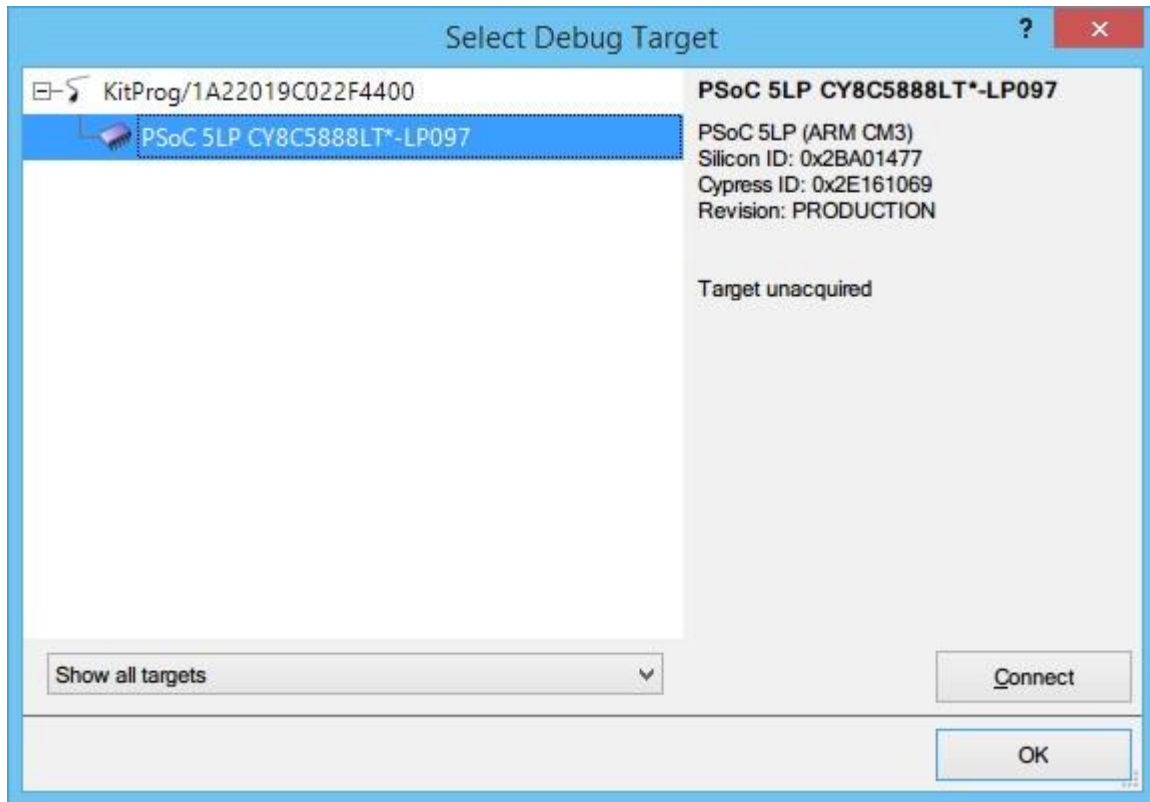
    for(;;)
    {
        /* Place your application code here. */
        //LED1_DR |= LED1_MASK; //Turn LED on by assigning the Port's Data Register
        LED1_Write(1); //Turn LED on by using the Digital Pin API
        //(Only works if Pin NOT HW Connected!)
    for (ix = 0; ix < 300000; ++ix); //Delay loop
        //LED1_DR &= ~LED1_MASK; //Turn LED off by assigning the Port's Data Register
        LED1_Write(0); //Turn LED off by using Digital Pin API (Same warning applies)
        for (ix = 0; ix < 300000; ++ix); //Delay loop
    }
}

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

The API for a Digital Pin contains functions for changing the drive mode, reading the pin value, writing a value to the pin, etc. (See the Datasheet for the Digital Input/Output/Bidirectional Pin). Using LED1\_Write() writes a specific value to the pin and takes care of all masking necessary so the values of other pins on the port are undisturbed. The API only works in the pin is NOT HW Connected (driven by a User Module), so it works here. The alternate method is commented out. It works by directly assigning the data register associated with port. The value assigned is the logical OR of the current value of the data register with a mask that is a 1 at the pin location and 0 elsewhere. Since logical OR with 0 leaves the value unchanged, and logical

OR with 1 is always 1, this also sets the pin to 1 without disturbing other pins. Setting it back to 0 is the logical AND of the inverse of the mask (AND with 1 doesn't change the value, AND with 0 always is 0).

13. Now, on to compiling the code and programming the device. Make sure the CY8KIT is plugged into the USB port. Click on “Debug” → “Select Debug Target”. You should see PSoC 5LP as a debugging target. Click on it and then click Connect, it should say Target acquired.



14. Now select “Debug” → “Program” (Or press F5) to both compile the code and program the device. If all goes well you should see the blue LED blink on the PSoC5 board.  
Congratulations! You have now completed your first ECE 381 lab. Show myself or the TA.

## ECE 381 – Lab 1b – Binary Counter

1. You will now create a new project that implements a simple 8-bit binary counter. Create a new PSoC Creator project named digiDesignerCounter.
2. In the TopDesign.cysch window, place a Digital Output Pin by dragging and dropping. Right click on the pin to bring up the configuration menu. Change the name to be CNTR. Unselect HW Connection. Since we need an 8-bit counter, we *could* place eight separate Digital Output Pins to accomplish this and repeat this. But, there is a much better way! On the configuration screen, there is a text box for the “Number of pins”. By default, it is one, but we can just as easily make it 8, so do so. The individual pins should now have a name like CNTR\_0, CNTR\_1, etc. Click OK.
3. Double-click on the “digiDesignerCounter.cydwr” file in Workspace Explorer tab to open the chip level viewer. Assign the CNTR pins to use Port3. Connect Port3 to the eight green/red LEDs on the Digi-Designer.
4. Build the project and open up the “main.c” file.
5. Write a program that outputs an 8-bit binary counter to the Digi-Designer LEDs. **MAKE SURE GROUND ON YOUR PSOC IS CONNECTED TO THE DIGI-DESIGNER!**