

ECE 381 – Lab 0 – PSoC4 Blinking LED

Objectives:

- Learn how to use PSoC Creator to create a simple project
 - Learn how to compile & upload a simple project to the PSoC4 we are using
 - Familiarize yourself with basic Digital GPIO
1. Launch PSoC Creator from the Start Menu (Or install it first, it's located in Labs/PSoC_Software_and_Manuals/)
 2. Go to “File” → “New” → “Project”
 3. Select Target Device, select the PSoC 4 and pick the PSoC 4200M in the drop down window and click “Next”.

Create Project - CY8C4247AZI-M485

Select project type
Choose the type of project – design, library, or workspace.

Design project:


- ☐ Target kit:
- ☐ Target module:
- ☒ Target device: PSoC 4 PSoC 4200M
- ☐ Library project
- ☐ Workspace


Next > Cancel

4. Create an Empty schematic and click Next

Create Project - CY8C4247AZI-M485 ? X

Select project template
Choose a schematic template or start your design with a kit or example project.

 **Code example**
Choose from our library of code examples.

 **Pre-populated schematic**
Start with typical MCU functions (like UART, ADC, etc.).

☐ **Empty schematic**
Create a full custom design by adding functionality from the component catalog.

< Back Next > Cancel

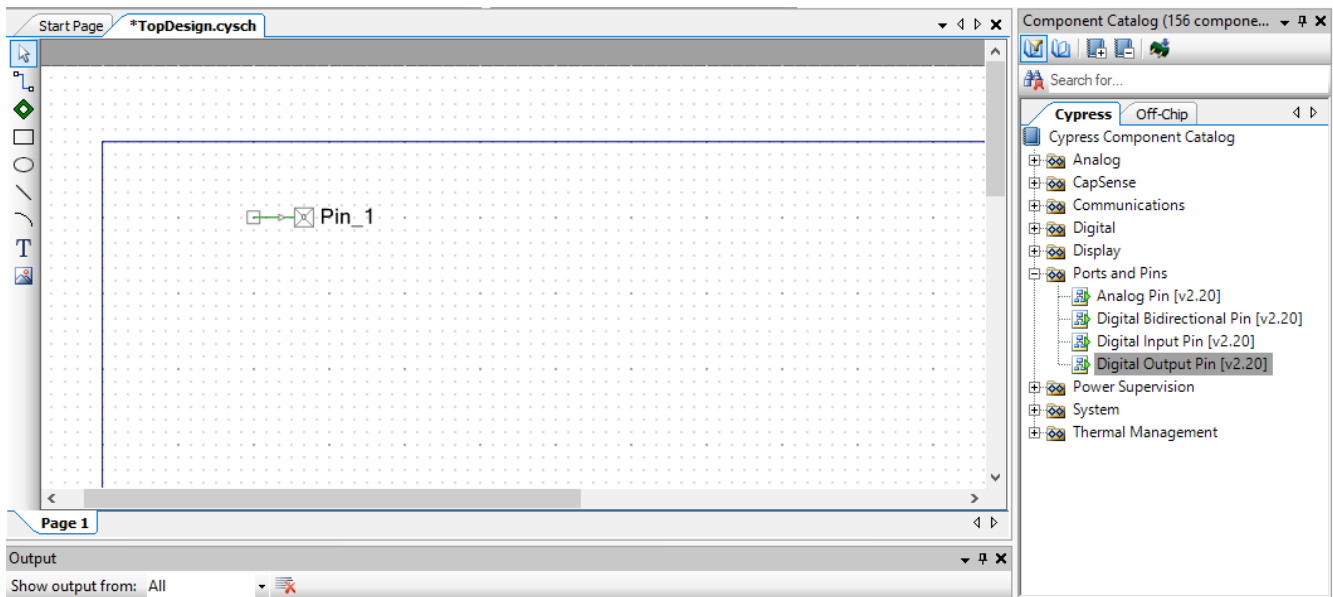
5. Select “Create new workspace” from the drop down, change “Workspace name:” to “blink_led” , “Location:” to “Desktop\ECE381_Labs” (Create if it doesn’t exist. If using your own PC, you can keep the default Documents\PSoC Creator as the root instead of Desktop), and “Project name:” to “blink_led” and click Finish. **DON’T USE YOUR Z: DRIVE OR A FLASH DRIVE!!!** The build process makes many small files and it takes considerably longer on these. There is also a bug that requires double compilation+uploading on a non-local drive. Use a directory on the C: drive only. A PSoC Creator project is organized in a single Workspace which can incorporate many projects. By default on creation, only one project is created within one workspace. New projects can either be created in new workspaces or added to existing workspaces. If multiple projects are in a single workspace, only the one selected as the active project will be built/compiled/programmed on the PSoC. I’ll leave it up to you to organize your lab projects as you wish (Personally, I typically use one project per workspace per lab, but that’s mostly so I don’t get confused with forgetting to set the active project correctly)

The screenshot shows a 'Create Project' dialog box with the following fields and values:

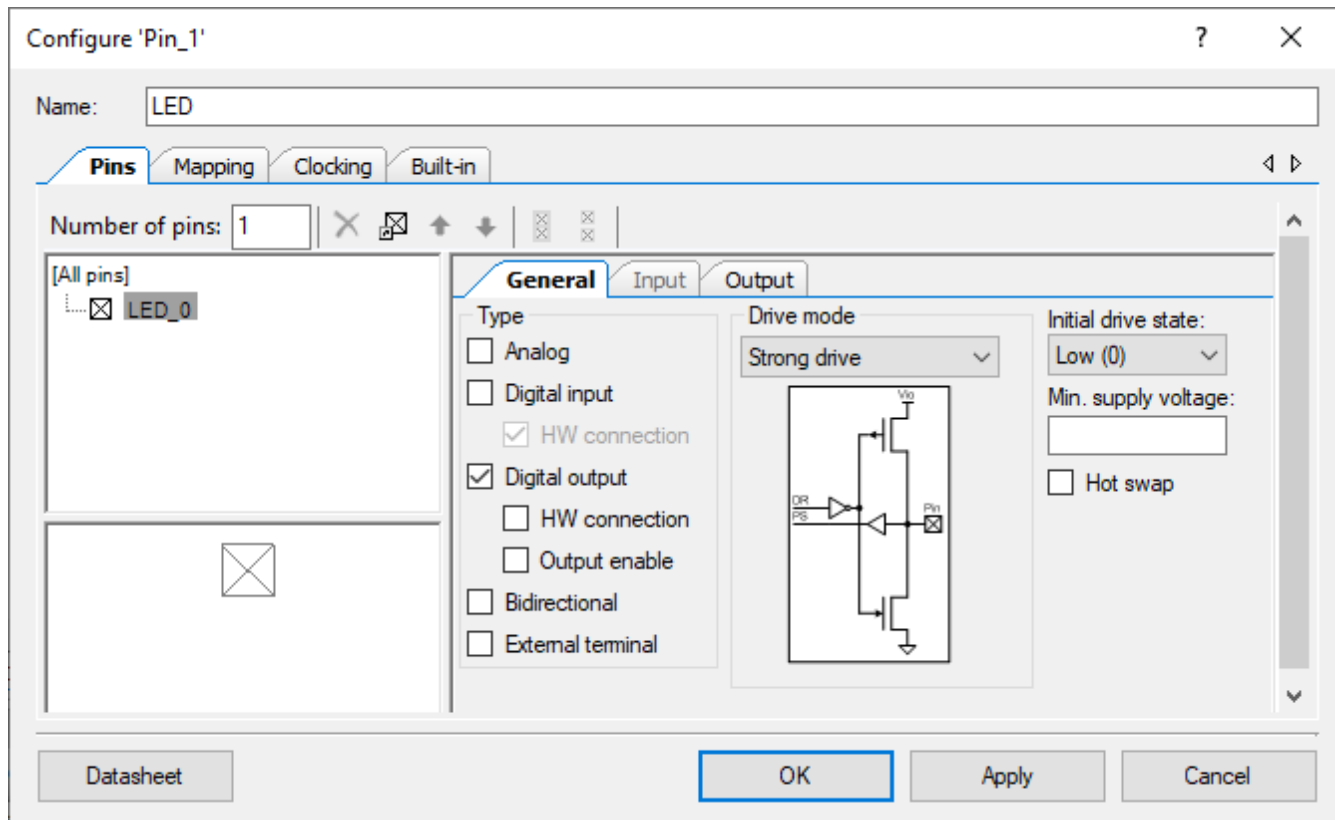
- Workspace: Create new workspace (dropdown menu)
- Workspace name: blink_led
- Location: C:\Users\tyork\Desktop\ECE381_Labs
- Project name: blink_led

At the bottom, there are three buttons: '< Back', 'Finish' (highlighted with a blue border), and 'Cancel'.

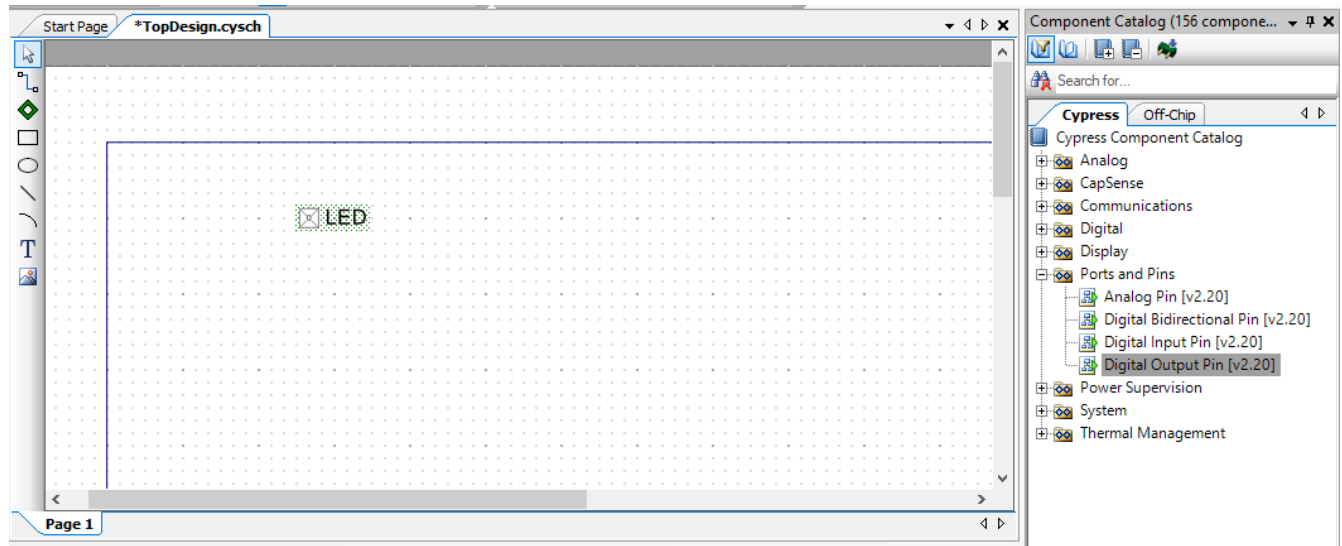
6. You should now see the main screen of the PSoC project. The schematic window is for placing User Modules. Since we want to blink an LED, we need to place a Digital Output Pin User Module. Click on “Ports and Pins”, click on the “Digital Output Pin” to highlight it, and drag it over to the schematic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



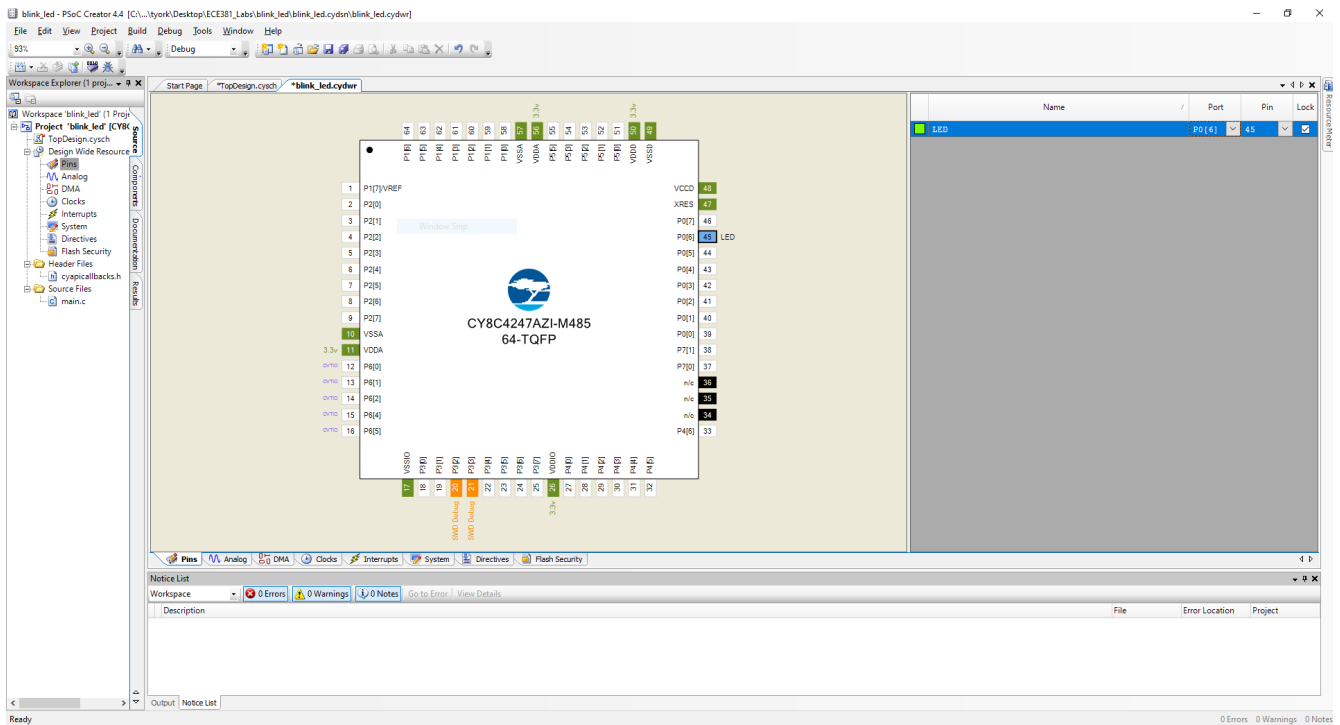
7. Now, right click on Pin_1 and click “Configure”. This brings up a dialog that allows you to configure various aspects of the pin. The most important option is the “Drive mode”. This should be set to “Strong drive” if you want the pin to be an output. The “Digital output” box should also be checked. 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 LED pin. We will use these features later on in the course, but for now, since we will drive LED straight from the C code, we will uncheck “HW Connection”. Your final configuration should look like this:



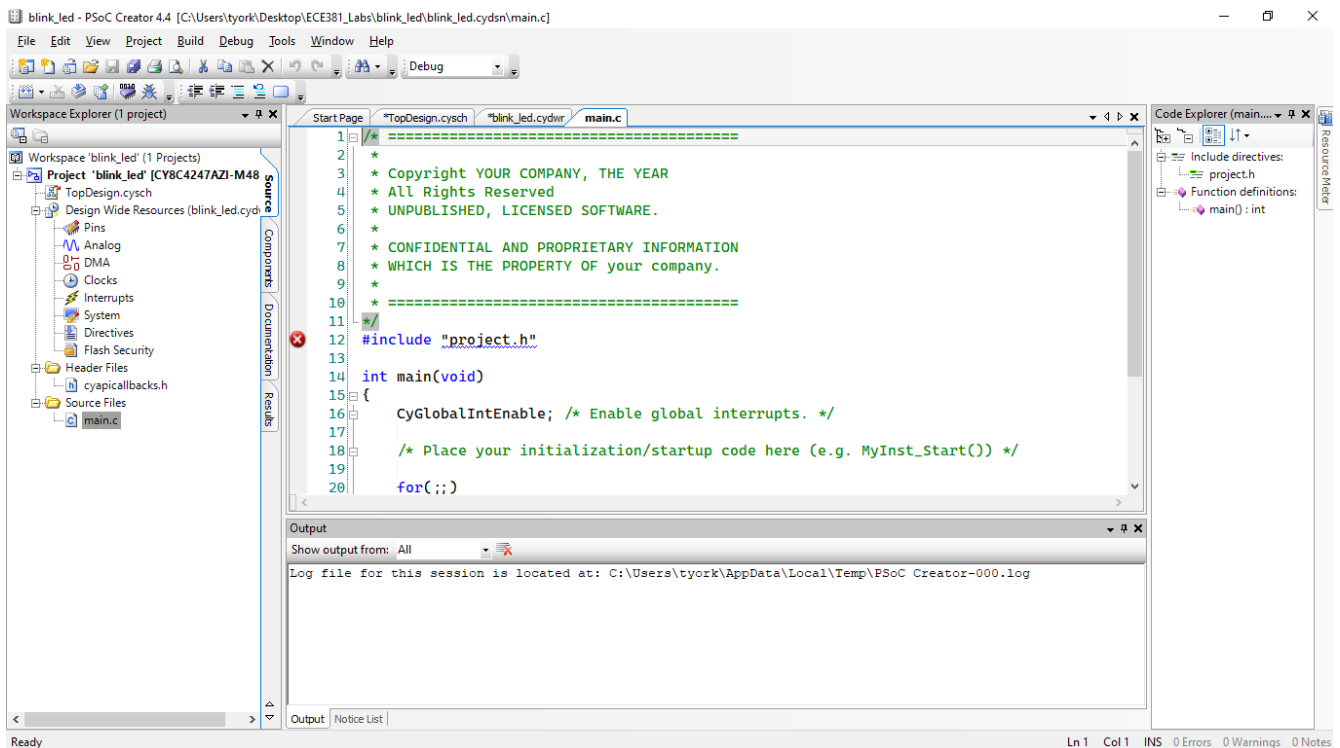
8. Click OK and the schematic screen should only show the pin now with the name LED.



9. Next, we need to assign a physical pin to the digital output pin named LED that we have placed. In the Workspace Explorer frame, double click on the Pins icon under Design Wide Resources. This should bring up the hardware configuration view for the actual PSoC 4200M (CY8C4247AZI-M485) 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 LED. There is a 3-color LED on our CY8CKIT-044, and checking the back shows the cathode (through a resistor) of the Red LED is on P0[6], the Green LED on P2[6], and the Blue LED on 6[5] respectively. 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 P0[6] for the Port (we'll use Red here, any of the 3 will work).



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 built anything yet, so project.h doesn’t exist.



10. 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” → “Generate Application” to build the framework of the project. This step doesn’t compile any code, but it does generate the backend APIs for each module (and other stuff that is needed like bootloaders, Cypress specific APIs, etc.). It might take a bit, but you should see a “Build Succeeded” message in the Output frame at the bottom of the screen.

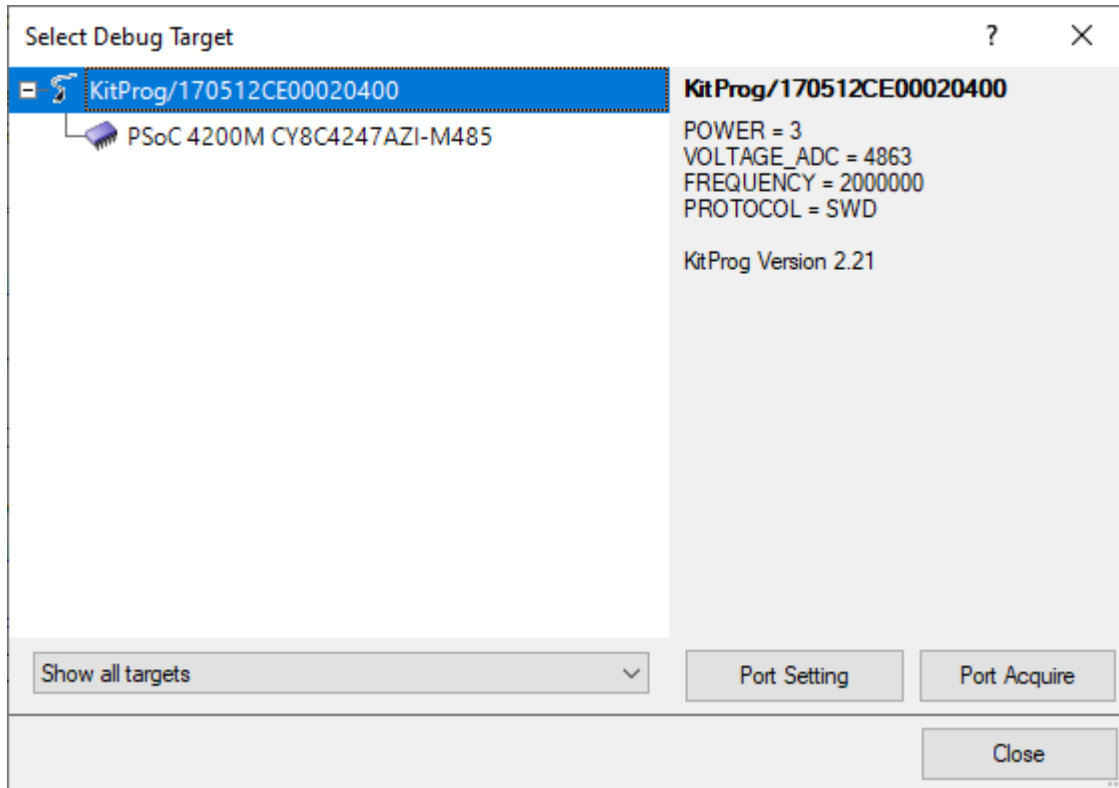
11. We can now program the blinking LED. There are two ways to do this, and I will illustrate both:

```
/* =====  
 *  
 * Author: Timothy York  
 * Course: ECE381, Lab0  
 *  
 * Description: Blink the LED on the  
 * PSoC Dev Kit.  
 *  
 * =====  
 */  
  
#include "project.h"  
  
int main(void)  
{  
    CyGlobalIntEnable; /* Enable global interrupts. */  
  
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */  
    int ix = 0; //Variable for counting  
  
    for(;;)  
    {  
        /* Place your application code here. */  
        //LED_DR |= LED_MASK; Write 1 to the LED using the Drive Register  
        LED_Write(1); //Write 1 to the LED using the HAL  
        for(ix = 0; ix < 3000000; ix++); //Delay loop 1  
        // LED_DR &= ~LED_MASK; Write 0 to the LED the Drive Register  
        LED_Write(0); //Write 0 to the LED using the HAL  
        for(ix = 0; ix < 3000000; ix++); //Delay loop 2  
    }  
}
```

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 PSoC4 Datasheet PDF for the Digital Input/Output/Bidirectional Pin specifics). Using `LED_Write(value)` writes a specific value to the pin(s) and takes care of all *masking* necessary so the values of other pins on the port are undisturbed. The API only works if the pin is NOT HW Connected (driven by a User Module), so it works here.

The alternate method is commented out. The alternative works by directly assigning the data register (`LED_DR`) associated with the 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). See the included AN86439 PDF for Lab01 to see the difference). We'll get to bitmasking and memory-mapping in the next lab.

12. 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 a PSoC4 as a debugging target. Click on it and then click Connect, it should say Target acquired. IF YOU GET AN “ARM Debug Device” or similar, it’s likely the PSoC needs a firmware upgrade. See the Appendix at the end of this lab for instructions on how to do it.



13. Now select “Debug” → “Program” (Or press Ctrl+F5) to both compile the code and program the device. If all goes well you should see the red (or whichever color you picked) LED blink on the PSoC dev board. Congratulations! You have now completed your first ECE 381 lab. In the future, you don’t have to select the debug target every time, just hit Ctrl+F5 if the PSoC is connected to do everything in one go!
14. During the demo, try showing me/TA how to:
1. Use a different color LED
 2. Change the blink speed

REFERENCES:

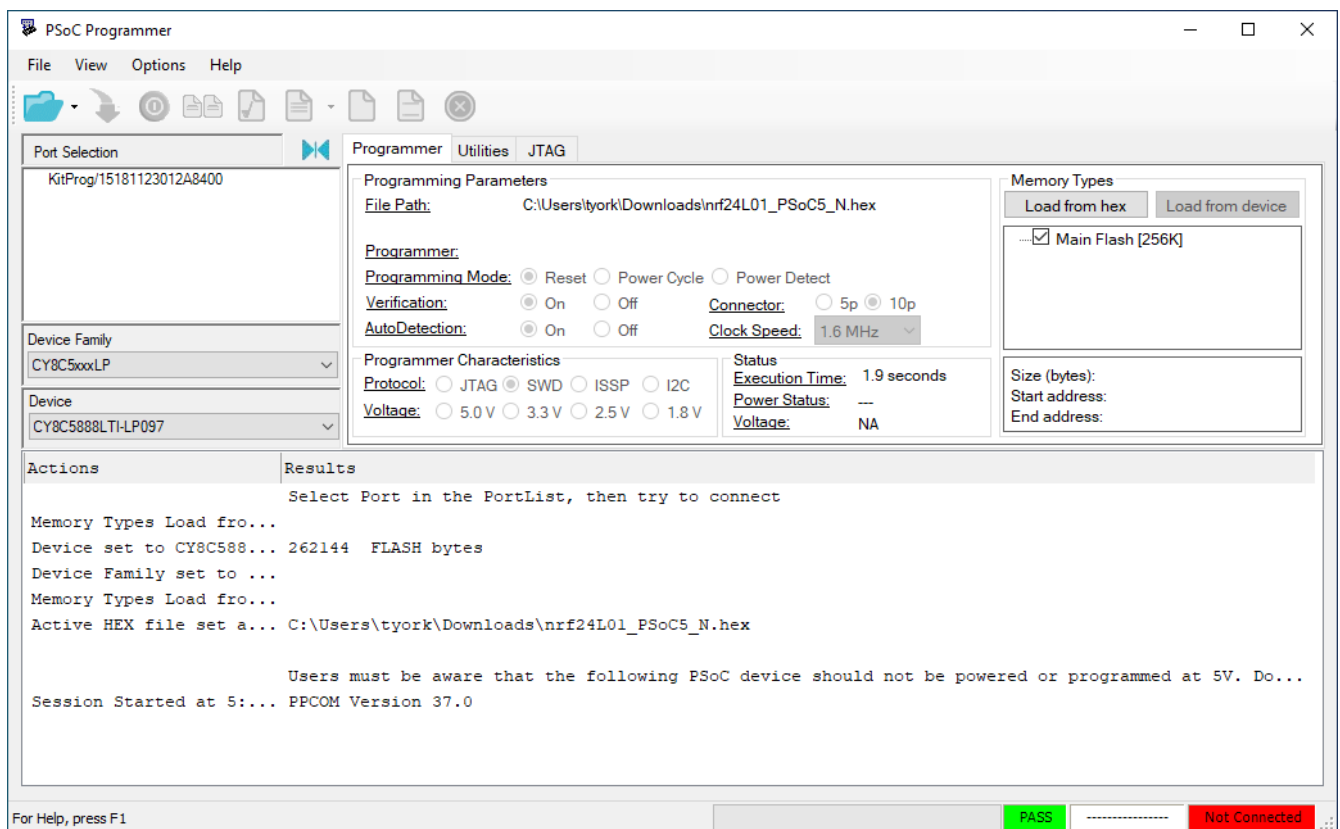
- PSoC4 4200M Datasheet
(Labs/PSoC_Software_and_Manuals/PSoC_4200M_Family_Datasheet_Programmable_System-on-Chip_PSoC.pdf)

- PSoC4 Technical Reference Manual
(Labs/PSoC_Software_and_Manuals/PSoC_4200M_Architecture_TRM.pdf)
- CY8CKIT-044 Prototyping Kit Guide
(Labs/PSoC_Software_and_Manuals/CY8CKIT-043_Prototyping_Kit_Guide.pdf)

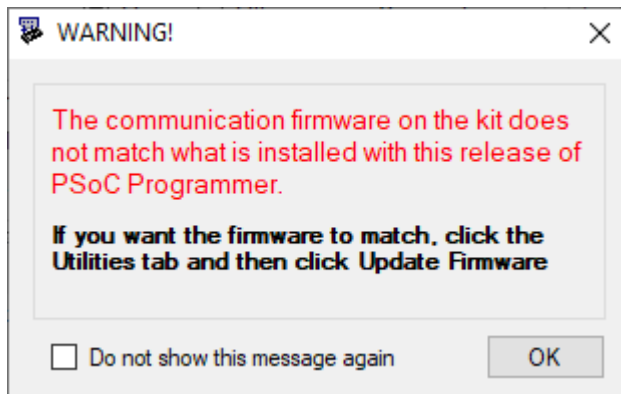
APPENDIX: FIRMWARE UPGRADE

Sometimes newer versions of PSoC programmer are issued, and along with them new versions of the firmware for the KITPROG PSoC on our CY8KIT-044 eval boards. The firmware is basically the program that runs on this PSoC which is closest to the large USB port. This is the one that handles flashing/debugging the 2nd PSoC in the middle of the CY8KIT-044 which is the one that runs your actual code. If you see warnings about firmware being out of date, you can attempt to upgrade it through the following steps:

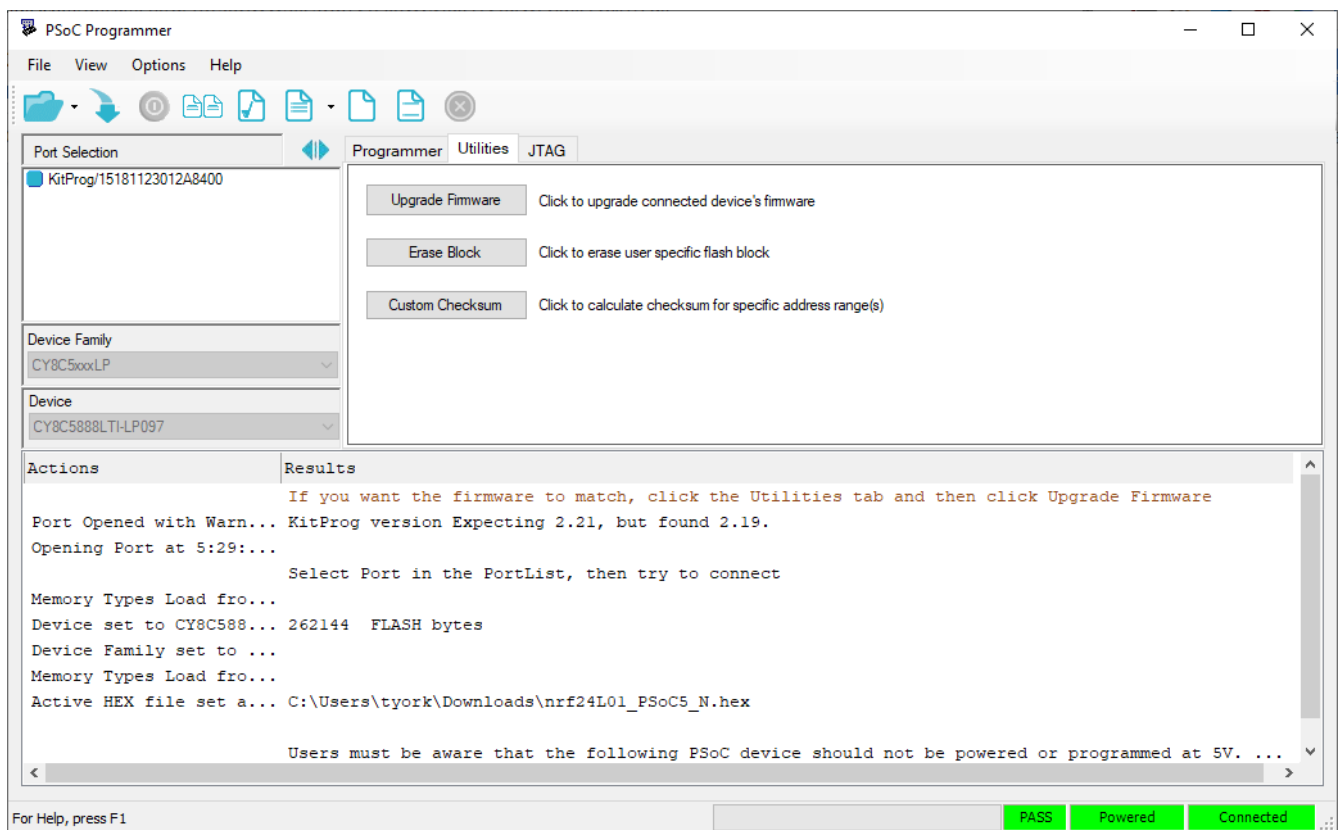
1. Close all open PSoC Creator windows. This is necessary since only one program can have access to the KITPROG device, and we will need the PSoC Programmer software in the next step to have this access.
2. Run *PSoC Programmer*. In Windows, it is usually under the Cypress folder in the Start menu (or typing PSoC Programmer will bring it up). PSoC Programmer is a separate program from PSoC Creator.
3. In PSoC Programmer, the KITPROG device should appear in a pane on the left under Port Selection. Click on it:



4. You may see a warning like this (which is fine at this point, since we are attempting to upgrade it), so just click OK.



5. Click on the Utilities tab and click the Upgrade Firmware button:



6. Wait for it to complete. ***DO NOT REMOVE POWER UNTIL IT IS DONE!!!*** If the *PASS*, *Powered*, and *Connected* bars show up green after the progress bar goes through twice, the firmware should be updated. You can now close PSoC Programmer, especially if you are going back to using PSoC Creator. If Programmer is running before Creator, the device will not appear in Creator until Programmer is closed (and vice versa).