Project 1: Simple GPIO

As you do this project, most sections have a set of questions in the Project 1 Gradescope assignment. Make sure you answer them as you go along because you will change the code and lose what you were supposed to put in Gradescope. READ THE ENTIRE ASSIGNMENT BEFORE YOU GET STARTED!!!

Introduction

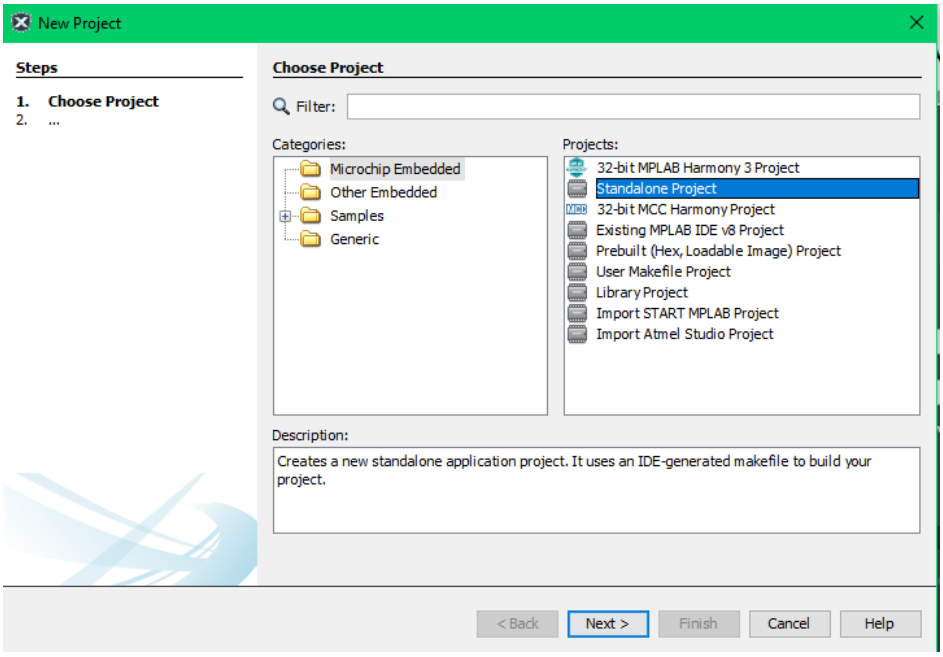
In this project, you are going to build code for the BASYS-MX3 board which will play with the 8 leds and 8 toggle switches the board provides. Before we can do anything, we need to know how those components are connected to the PIC. Open the BASYS-MX3 Reference Manual and find exactly which pins we will be using. As a clue, you want to look at the pinout description. That gives you which pin on the PIC is connected to each thing on the BASYS-MX3 board. You can use those pin names and the PIC documentation to figure out to which port/pin it is referring. For example, RA0 is Port A pin 0.

Answer the question in Gradescope to record the results of your research.

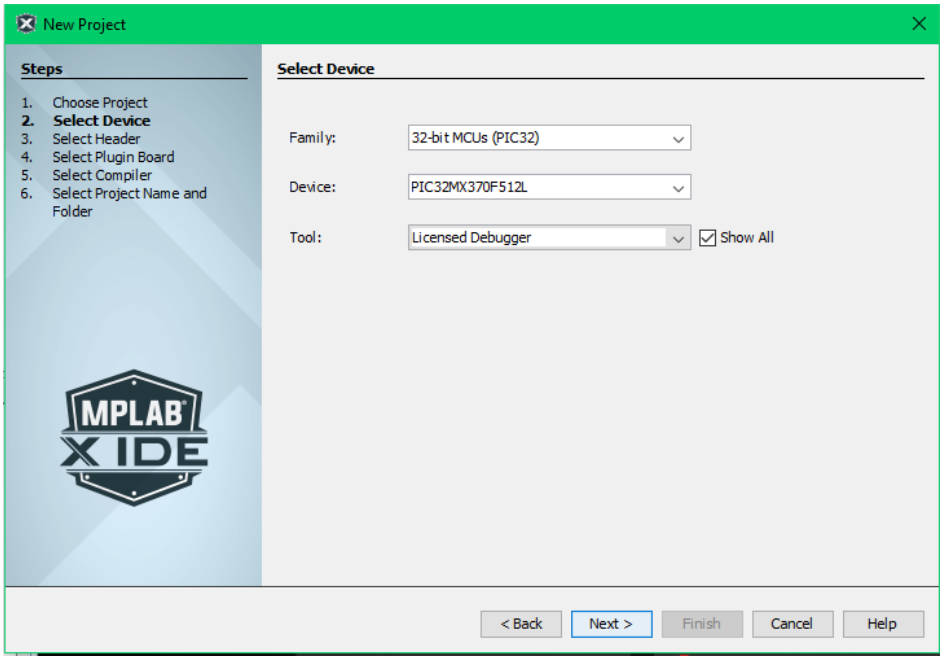
Please note: the BASYS-MX3 comes with a layer of software the can go between our software and the board. That layer totally obfuscates what is really happening, so I want you to develop that layer on your own. Do NOT use their source code for ANYTHING!!!! You need to learn to read the device documentation because, in real life, you'll be the one writing the layer that interacts with the hardware.

Create your project

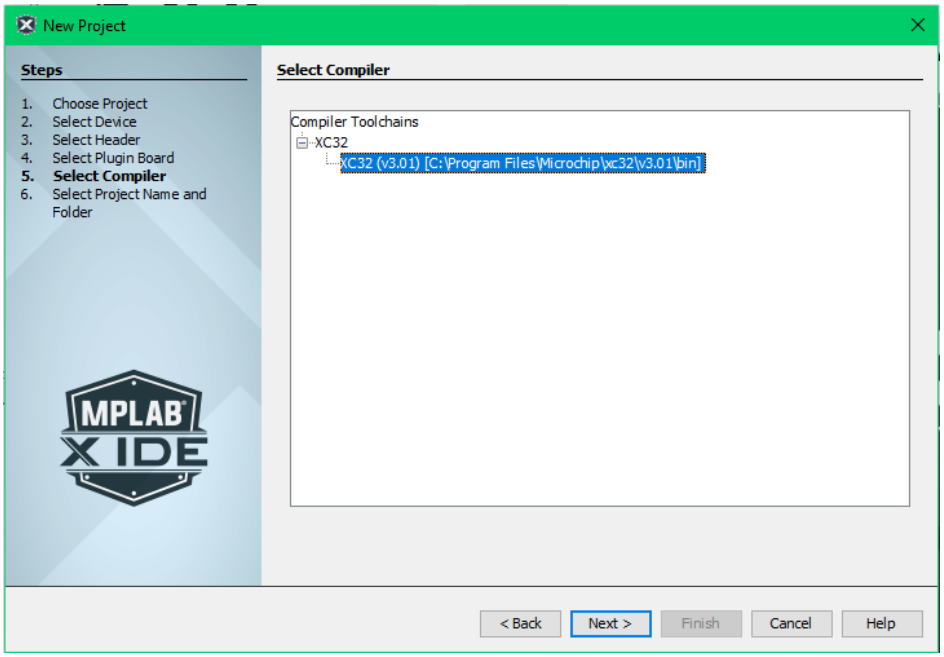
In the MPLab X IDE, create a new project. When you see this screen,



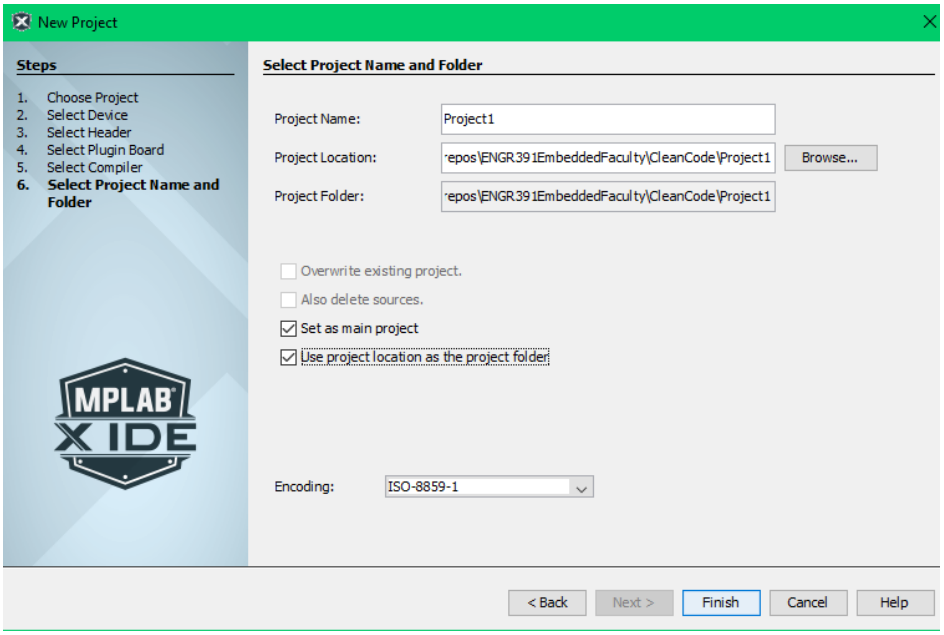
select "Standalone Project" and click Next.



On this screen, pick the processor that is on the Basys board (the one that I've selected in that picture) and pick "Licensed Debugger" for the tool. If you don't see that option, click "Show All" and it should appear. Click Next.



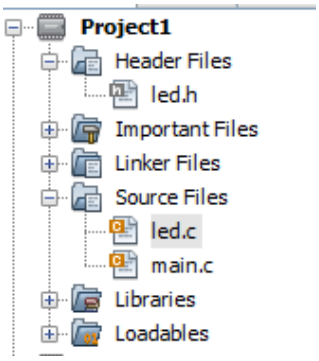
Select the cross-compiler (I think there will only be one option). Click Next.



Now you have to tell it how to structure your projects. For me, I create a folder for each project. That is the "Project Location" and then I set "Use Project Location as the project folder" because that prevents it from making another subfolder. Click Finish

Lay out some architecture

We want to separate the code that is interacting with the hardware from the bigger picture code (design principle: separate what changes (the app) from what stays the same (the hardware)). Therefore, start with two .c files: main.c that will just contain your main function (for now at least) and leds.c that will contain functions that interact with the hardware. Following good C design principles, that means that you should have an leds.h file which should be included in both of your .c files. The .c files should be in your source folder and the .h file should be in your includes file. Read that carefully and set up those three files. Remember that good .h files have header guards (if you don't know that term, google is your friend). When you are finished, your project should look like this:



Put a screen shot of your project set up into GradeScope.

Light some lights

To start, lets just make the leds cycle on and off. For this, you'll need this functionality in your leds.c file:

- A function that intializes the LEDs
- A function that toggles a given LED

Once you've built that, make your main function initialize the LEDs and then loop forever toggling them in sequence. You'll probably need to slow this down to see it happening. You can use this as a simple delay function:

```
1 void DELAY(unsigned int count) {
2         unsigned int i, j;
3         for (i = 0; i < count; i++) {
4                 for (j = 0; j < 100000; j++);
5         }
6 }
```

Put a video of your working system in GradeScope.

Answer the other questions for this section in GradeScope.

Add the switches

Now that we know how to play with the LEDs, let's play with the switches, too. Modify your system so that each switch controls the matching LED (Switch 0 controls LED 0, ...).

To do this correctly, you will need to do the following:

- Add functions in your leds.c/h files to turn on and turn off a given LED
- Create a switches.c file and the corresponding switches.h file and add them to your project
- Make functions in your switches.c/h files to initialize the switches and read from a given switch
- Modify your main function to loop through the switches and setting the LEDs based on the values you read from the switches.

Put a video of your working system in GradeScope.

Answer the other questions for this section in GradeScope.