ECE 3720

Microcomputer Interfacing Laboratory

Section 005

Aaron Bruner

Date Performed: 01 / 27 / 2021

Lab 1

ABSTRACT:

A lab designed to show the methods of programming, compiling, and running code for the PIC32MX150F128D microcontroller using the MPLAB X IDE software and an NI ELVIS II board.

**INTRODUCTION:**

The goal for lab 1 is to demonstrate the method of preparing our development environment along with testing our equipment. To perform our lab, we required a variety of materials including:

* A breadboard
* Wires
* Resistor ( x1 15kΩ & x1 1kΩ )
* Capacitor ( x1 100nF & x1 10µF )
* PIC32MX150F128B
* Analog Discovery 2 (AD2)
* Digilent WaveForms
* MPLAB Snap Debugger
* MPLAB X IDE (Programming Software)
* PIC32 Datasheet

Once we have the breadboard setup for this lab we won’t need to set it up again for future labs. However, we may need to add on to what we already have or modify the original state.

The AD2 is going to be a partial power source and our output LEDs. Instead of using actual LEDs we will use the WaveForms software and simulate the LED output. And partial power source means that we will power the PIC32 chip with the AD2 but not the MPLAB Snap Debugger. It will be powered off of a Micro USB cord.

**EXPERIMENTAL PROCEDURES:**

The circuit was setup exactly as the documentation explained. First, with a clear breadboard you want to seat your PIC32 chip somewhere near the center of the breadboard. This will give us enough room to add other elements to our board. After this we want to collect other materials such as resistors and capacitors from the provided lab materials. Resistors and capacitors were preselected by the lab TA along with how to connect them. A 15kΩ resistor should be in series with a 1kΩ resistor connected on one side to the power rail and the other to pin 1. Note that the 15kΩ resistor should be closer to the power rail when placing the resistors. Capacitators were applied to pin 20 and 1. The pin 1 capacitor is 100 nF and should be attached as a note between the two resistors and the other end connected to ground. The pin 20 capacitor is 10µF, the longer end should be connected to pin 20 and the other end should be in ground.

There was no differentiation between materials used. Wires were selected based on what was provided as lab materials. Similarly, the wires to the AD2 were not specific and have no defining factors over other wires.

Pin 4 (B0) through 7 (B3) were connected to the analog ports on the AD2 to display the output. These pins correspond to the 4 least-significant bits of Port B.

The code for lab 1 was provided. A simple delay function was added that can change how fast the change in LEDs is. The time it takes to change the number displayed is what delay is designed to do. The main function sets all B pins to be output using TRISB = 0. Then, using a while loop forever we will write the current output to all of the B pins using LATB. Next, we’ll increment the count variable and ensure that we’re not going over 15. We call the delay function and then return to the top. Once our counter reaches 16 we set the count back to 0 and start from the beginning.

The lab also tasked us with modifying the provided code to count down to 0 instead of counting up to 15. This is a simply modification, to begin we will change the initial count to be 15 instead of zero. Then, we want to change the counter to be a decrement operation instead of a increment. Lastly, we need to flip our if statement to be if(count < 0) count = 15; that way we’re going back to the top once we hit the bottom.

After programming the microchip we can see using the AD2 software that it does indeed count down from 15 to 0.

**RESULTS and DISCUSSION:**

After programming the circuit, you can notice that even after unplugging the MPLAB Snap Debugger the microchip still performs the count up/down. We know it is still performing this function from the display LEDs in the AD2 software.

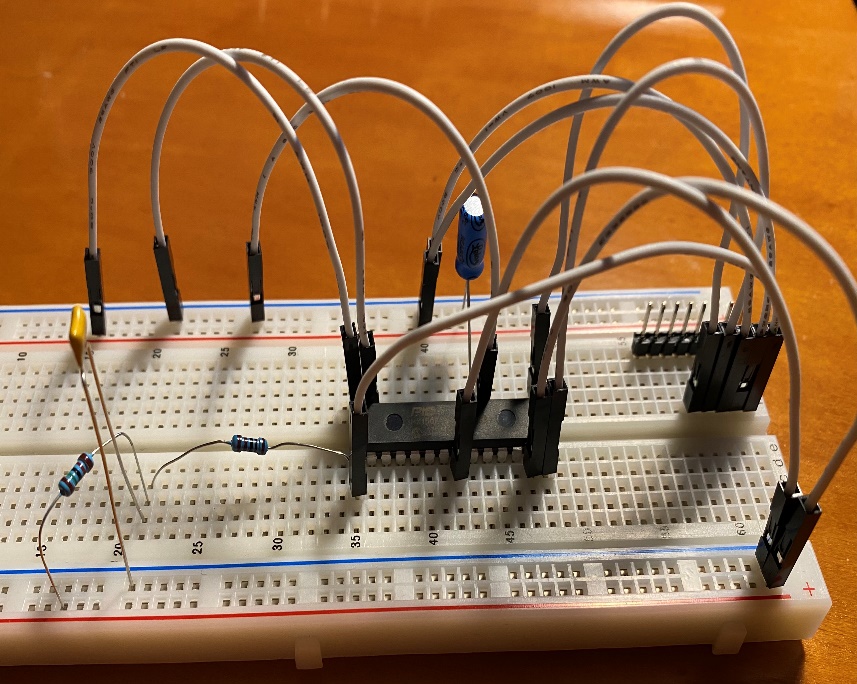
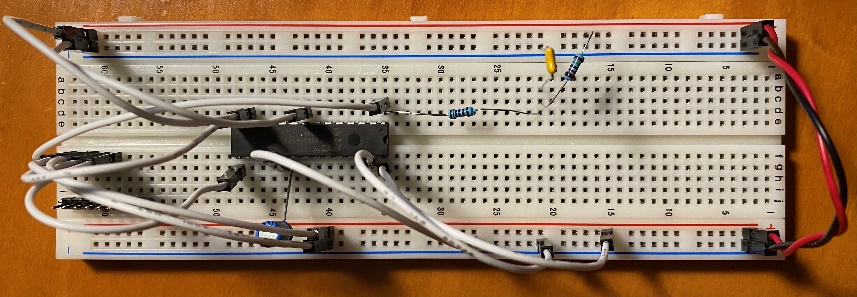
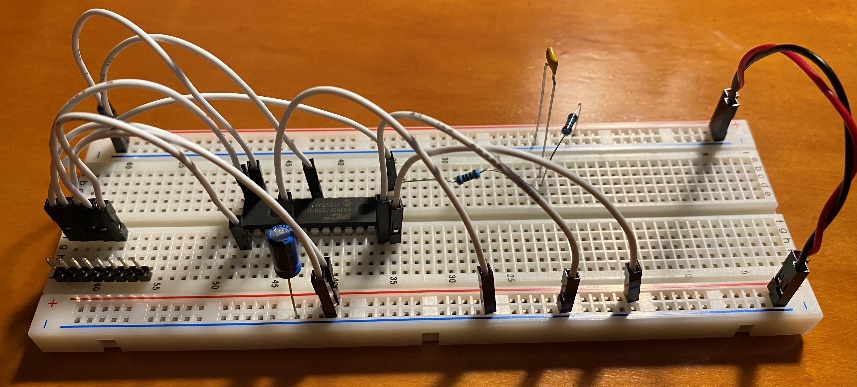
One issue that arose was that the MPLAB Snap Debugger’s active LED was not lit even though it was plugged in and detected. This was simply fixed by using the MPLAB X IDE software to reinstall the firmware on the chip to return it to full functionality. Once completing the steps it tells you to do it worked just fine.

This week’s methods of utilizing the IDE to program in and the Snap Debugger to program the PIC32 chip are critical for the rest of the semester. Without the knowledge gained in this first lab I am unsure if it would be possible to complete the other labs.

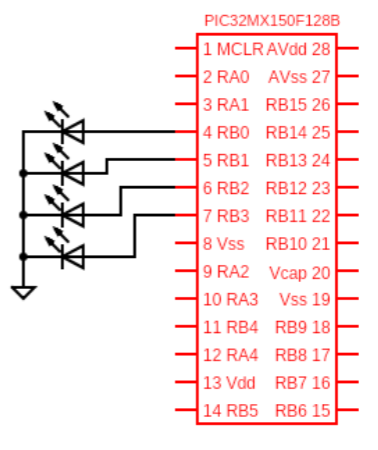
**CONCLUSION:**

To conclude, lab 1 demonstrates some fundamental techniques for setting up your development environment to program more complex microchips with even more complex programs. Going forward the labs will begin to build on each other and towards the end of the semester the limitations as to what can be made will practically be endless.

**FIGURES AND TABLES:**



**Figure 1: Image of breadboard with circuit from Figure 2**

****

**Figure 2: Wiring for lab 1 (Pin connections described in experimental procedures)**

**CODE:**

CODE FOR COUNTING DOWN:

#include <xc.h>

//Simple delay w/o timers because of lab order

void delay(){

int i, j;

for(i = 0; i < 400; i++)

for(j = 0; j < 400; j++);

}

main(){

int count = 0;

TRISB = 0; // Set all B pins to output

while(1){

LATB = count; // Write count to output pins

count++;

if(count > 15) // If the count is 15 then we go back to the bottom of 0

count = 0;

delay();

}

}

CODE FOR COUNTING UP:

#include <xc.h>

//Simple delay w/o timers because of lab order

void delay(){

int i, j;

for(i = 0; i < 300; i++) // By changing the 300 variable you can change how fast the

for(j = 0; j < 300; j++); // LEDs change numbers

}

main(){

int count = 15;

TRISB = 0; // Set all B pins to output

while(1){

LATB = count; // Write count to output pins

count--;

if(count < 0) // If count is 0 then we go back to the top

count = 15;

delay();

}

}