MICROPROCESSOR II AND EMBEDDED SYSTEM DESIGN

Lab Number: 1

Date: 02/09/2016

Student Name: Trever Wagenhals

Student ID: 01425986

Group Name: Steel Elephants!

PERSONAL CONTRIBUTION TO THE LAB ASSIGNMENT

Stephen and I assisted each other in creating the rough draft of the code (pseudo). I verified it after he typed it up, helping with the minor bugs that needed to be fixed, and some commenting confusion. I also assisted Nicole with the schematic setup, giving her a rough layout of the pin configuration. There was some confusion across why it was set up in a certain way, and so I explained the reasoning to her.

PURPOSE OF THE LAB

The purpose of this lab was to program a PICkit and use a microprocessor to control if an LED is lit or not based on the voltage passed through a photo-resistor. Through this, embedded microcontroller design, analog to digital conversion, and sensor circuitry should be better understood.

INTRODUCTION OF THE LAB AND REPORT

Because of how heavily almost all of the electronics around us rely on sensors to operate, it is necessary to understand them. To gain a better understanding of designing embedded microcontrollers with sensors, this lab involved designing a light intensity sensor controlled by a PIC microcontroller. This device should be able to:

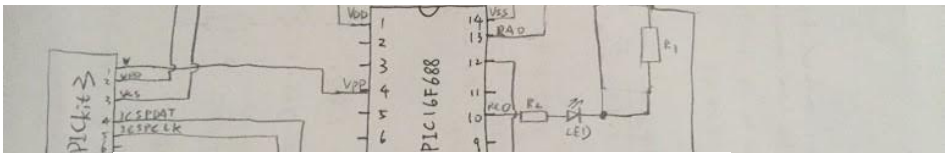
1. Convert light intensity information to a variable voltage
2. Obtain an analog input signal through AN0
3. Convert analog to digital through the internal ADC of the PIC
4. Turn on the LED when in a dark area and turn on the LED when in a light area.

MATERIALS, DEVICES AND INSTRUMENTS

- PIC Kit3 Microcontroller with USB cable

- Laptop with MPlab and 8bit compiler, for programming purposes
- 10k Ohm Photo-resistor
- 10k Ohm Resistor
- 200 Ohm Resistor
- 0.1 uF Capacitor
- Breadboard with wires
- PIC16F688 microprocessor
- Lab instruction and assistance as given in pdf format
- Multimeter

SCHEMATICS



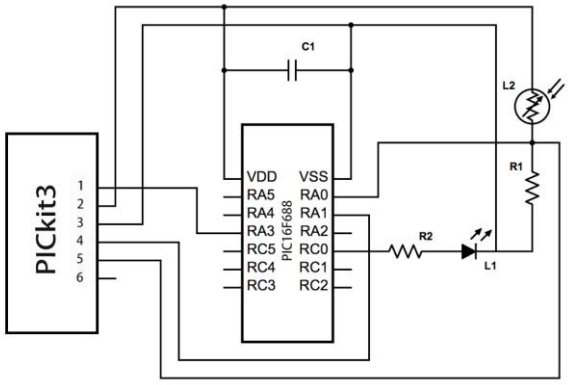
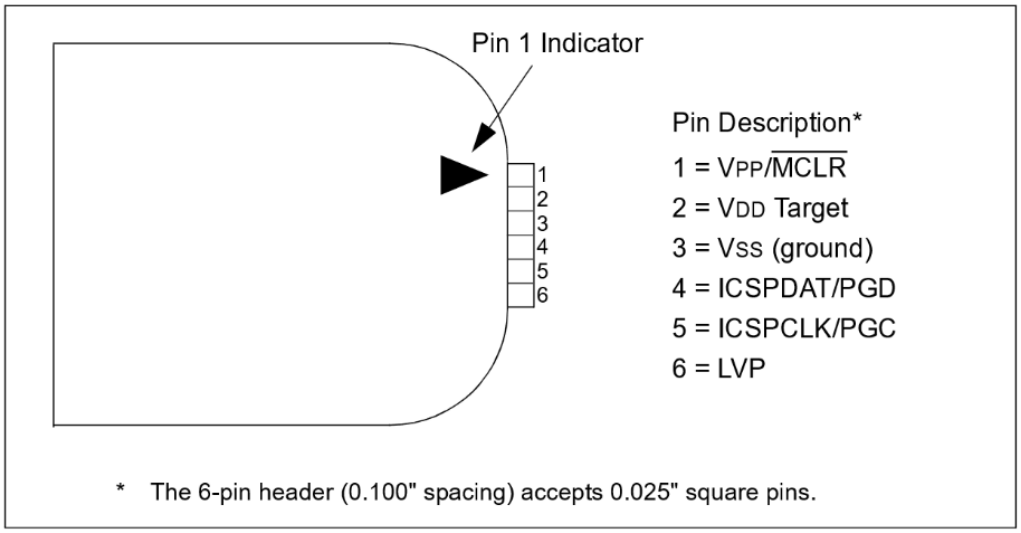
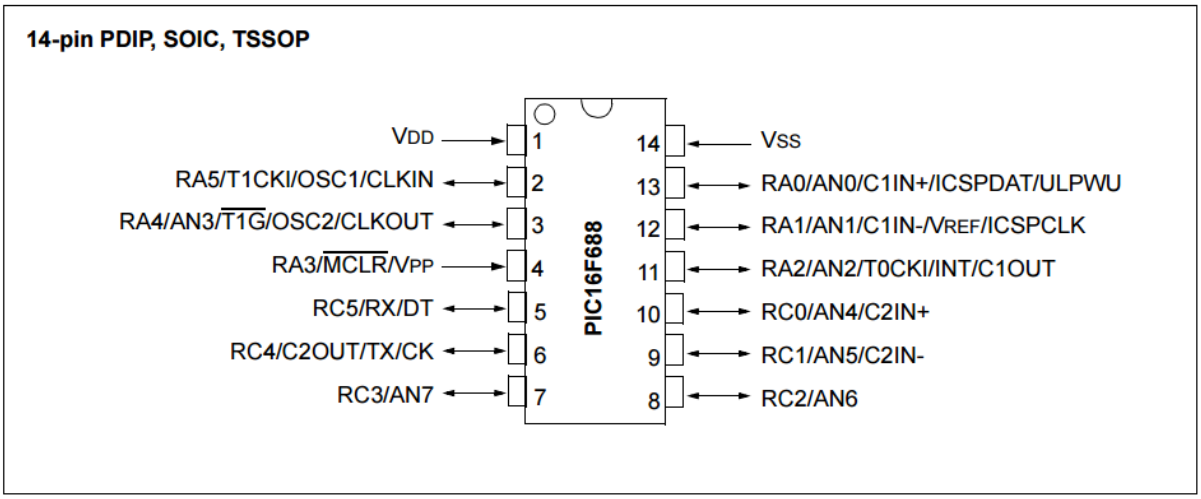


FIGURE 1-2: PICKIT™ 3 PROGRAMMER CONNECTOR PINOUT



Pin Diagram (PDIP, SOIC, TSSOP)



LAB METHODS AND PROCEDURE

The first steps in order to start moving towards completion of this lab were to identify all of the pins on the PIC microcontroller and on the microprocessor. The pin indicators for the PIC was given in the PIC box, allowing quick identification. The microprocessor pin configuration was given in pdf format and was required to study before beginning the project.

Once the pins were identified, it was possible to begin implementing the circuitry on the breadboard. Before implementation, we decided to test all of our components. In order to test them, we had to use the C program example that was supplied to use to load the PIC through our laptop so that it can start supplying a voltage. We initially tried the 5V to see if the laptop we were using would support it. It initiated, but the measurement was only 4.25V right from the start. Setting up

our photo-resistor and LED, we confirmed that the LED was functional and the voltage drop across the photo-resistor varied based on the lighting.

Given the schematic of the microprocessor and the hints, we knew that RA0 should be set as an input with the given code. RA0 was pin 13 and was tied to pin 4 of the controller to implement input. Being told that we could pick any of the pins below pin 12 for output, we just chose pin 10. Using similar code as the snippet given for RA0 input, we made RC0 output. Through the hint given, we used the ANSEL register to set AN0 to analog.

We turned the ADC on through setting GO equal to 1. Once the conversion was turned on, we could then read the value that was stored to the ADC result high register and the ADC result low register. By shifting the high register and adding the low register, we can get the whole ADC value, which was a conversion from the analog voltage value to a digital value. This value was then returned and used in our LED function to determine if the LED should be on or off based on the value. This part involved a lot of guess and check work, where we had to adjust which values set RC0 to either 0 or 1. Having a higher than necessary value will cause the LED to be on even if it is in a light environment. Having a lower than necessary value will cause the LED will have the LED stay off even if it was in a dark environment. Eventually, the value we needed was found for consistency.

TROUBLE IN THE LAB AND HOW DID YOU SOLVE IT

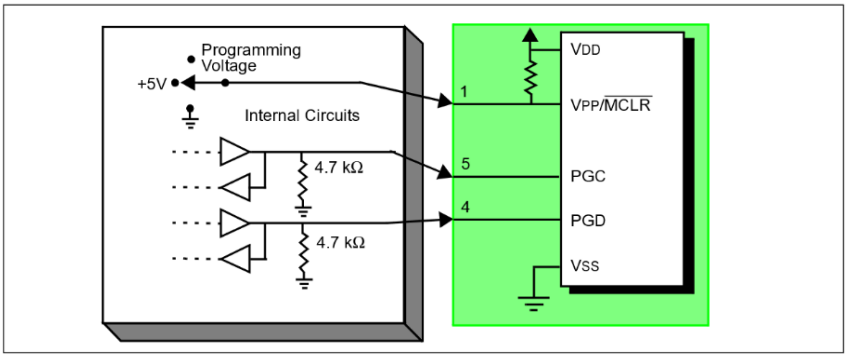
Initially, knowing which pins of the PIC did what was an issue. After digging through the supplied pdf, a great picture showing a basic setup of the PIC to a circuit to allow for programming was given that helped steer us in the right direction. The picture that helped us is posted below.

Reading through the hints, it was clear that a better understanding of the registers was needed. In the PIC16F866 pdf, PORTA and the TRISA registers are explained in detail in section 4.1 that allowed use to understand how to implement it. Continuing reading also explained the ANSEL register that was mentioned in section 4.2.1 and section 4.3.

When we were implementing the high and low values of the ADC register, we knew that the we need the combination of the two values in some way to then use to set our range of values for our LED. At first, we did not shift the high register over and simply added the register values, resulting in really low numbers that did not allow for high customization of the digital value to get the right sensitivity. It took us a few attempts to realize that we needed to shift the high register over 8 bits (due to the length of the ADC being 10 bits) to get a value that was appropriate to create an accurate sensor.

When we started writing our pseudo code, we knew that we were going to use uint based on other examples we saw. So, we defined it and attempted to execute it later, and for some reason the compiler kept sending error codes at us. Because this was early in our execution, we did not realize that we had not yet converted our analog value to a digital value properly, and thus it was not able to execute our uint command. Once we realized that we had to set GO to 1 to start the conversion, we managed to get resolve our issues when addressing the value in the high and low registers. This part was mainly resolved through assistance of the professor, as well as a few classmates.

FIGURE 2-6: PROPER CONNECTIONS FOR PROGRAMMING



RESULTS AND CONCLUSION

By the end of the lab, the circuit was working as anticipated. As a finger was placed over the photo-resistor to limit light to the circuit, the LED would turn right on. The opposite was true when the finger was moved and a higher light intensity was introduced to turn the LED back off. With a better understanding of how to use PIC controller, reading the pins and registers of microprocessors, basic programming through the PIC, and becoming familiar with a photo-resistor, we can now implement more advanced sensors in the future.