Dario Ugalde
1001268068
CSE 3442

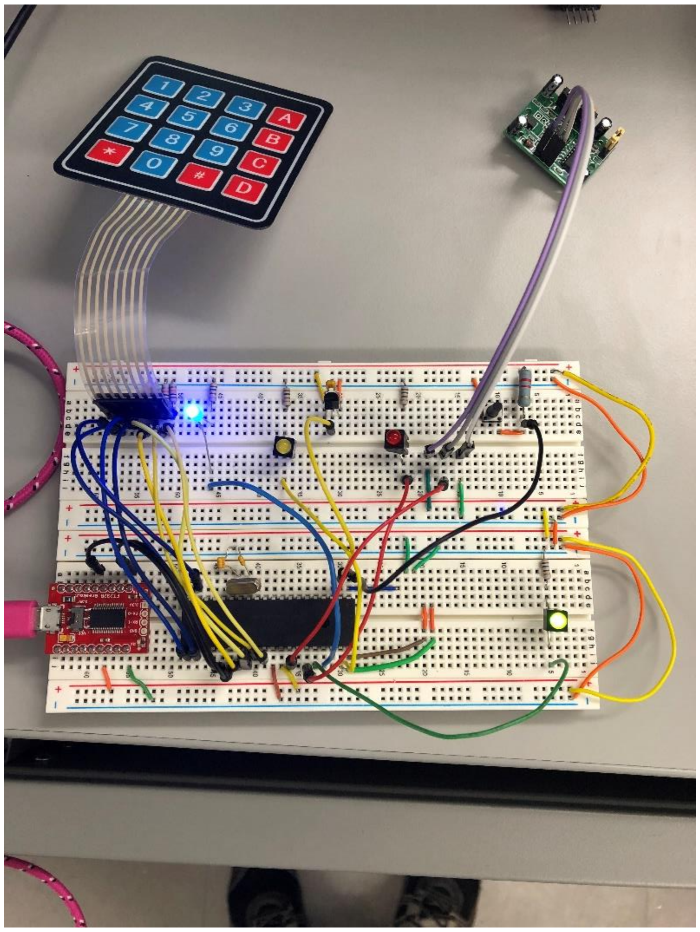
Lab 7: ABET Alarm System

Introduction

The purpose of this lab was to bring together many of the topics covered throughout the students' time in Embedded Systems. The result was to design and implement a hardware/firmware "Alarm System" using bi-directional communication between a PIC18F4520 and a PC. The system was also required to be save previous settings and keep their most recent state even if the PIC were to lose power or be reset through MCLR pin. To create this system, students were required to demonstrate their ability to use EUSART(Enhanced Universal Synchronous Asynchronous Receiver Transmitter), Analog-to-Digital Conversion, Timers, Interrupts, MCLR Reset, External Crystal Oscillator clock source, Digital Input and Output, and ICSP(IN-Circuit Serial Programming) with PICkit 3 and taught in lecture.

Procedure/Results

To start the project, it was necessary to properly wire up a breadboard with all the components provided to use for the project. This task proved to be simple but took a while to get done since every component has to be connected to the appropriate place on the PIC so the system will function correctly.



Above is an example of how a fully wired alarm system could look. Once all the wiring was completed, the next stage of the lab is to determine the configuration settings to have the system interface properly with all the components provided. This was the part that gave me the most trouble out of the entire project. However, because of this struggle I learned just how valuable of a resource a datasheet can be for an electronic device. I ran into many situations where I just didn't know how to get a specific functionality out of the PIC and I never would have known what to do if it wasn't for the datasheet. A specific example of this was when I was first starting out, the PIC would not execute any of my instructions, no matter how many times I recompiled the code or restarted MPLAB, nothing would work. However, during my frustration, I was fiddling with external crystal and for some reason the green LED came on. As soon as that happened, I realized that somewhere there must have been a configuration setting that I had overlooked and sure enough after looking at the PIC configurations and referring to the datasheet, I discovered the PIC was not set to work with a high speed crystal. This situation was one of dozens, by the end of the project, I realized that there are so many nuances and intricacies that are involved in working on these types of projects. However, as frustrating as it was to hit obstacle after obstacle when dealing with these demanding devices I kept looking and trying different approaches to solve each problem I encountered through the project. Once all the configurations were properly working, I began working on the main portion of the project: implementing the alarm functionalities. For each of the main four steps, I tackled them one at a time as they were presented on the lab description. I first brainstormed methods that I could use to approach the requirement and did any necessary calculations to make sure I would have everything I needed to begin coding. Below are the calculations I used to get the serial communication going between the PIC and serial device:

$$\mathbf{F_{osc} = 20MHz}$$

$$\mathbf{SPBRG = (20{,}000{,}000/(64*9600)) - 1}$$

$$\mathbf{= 32.55}$$

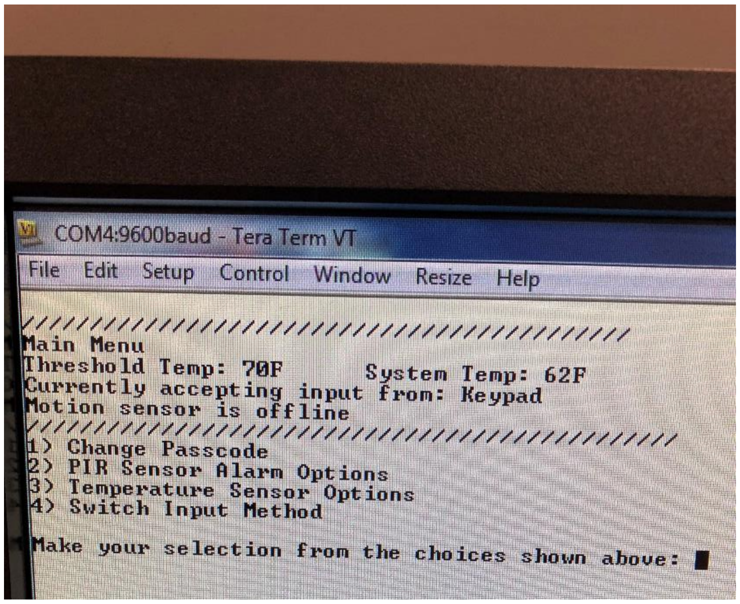
For my code, I had to round up the SPBRG to 33 to get my communication to work right between the two devices. For the interrupts to work properly for the two alarms, I had to set up the two priority vectors, where one was high priority for the motion sensor and the low priority was dedicated to the temperature sensor. To get RB0 to work as an interrupt it was necessary to configure INT0 appropriately and make sure that it was set up as an interrupt, and by default, INT0 happens to be a high priority interrupt so setting that up worked nicely. As far as the temperature sensor's interrupts went, I set up a timer that ran for the longest time possible until it overflowed. I simply set it to the highest settings possible to get the longest time value possible out of the timer as I didn't want to bog down the system with handling temperature calculations every tenth of a second. For the ADC part that went along with that, I referred to the lab we did where we used ADC to calculate voltage. Using the same formula as then, I came up with the following calculation:

$$\left(\frac{\boldsymbol{result}}{\mathbf{1023}}(\boldsymbol{temp}_{+} - \boldsymbol{temp}_{-})\right) + \boldsymbol{temp}_{-}$$

Dario Ugalde
1001268068
CSE 3442

$$\left(\frac{result}{1023}(180)\right) + 50$$

This equation gave me a good enough approximation to room temperature. The requirement to implement for the alarm system was to have the system allow a 4 x 4 keypad be used for input. The instructions on how to achieve this method of input were found on the data sheet of the keypad. Following those instructions, I set up a method that started out by having RD0, RD1, RD2, and RD3 send a signal out to the keypad and poll for a change in RD4, RD5, RD6, and RD7. The was this works is because when a key is pressed the incoming signal shorts out on the button and sends a signal back down to the PIC where it can be read and used to determine exactly what key was pressed based on the combination of bits that are high. This shouldn't have been very difficult to implement but I had some issues where having modified TRISE was causing an anomaly in my PIC and after removing that line of code with TRISE everything worked just fine. I've included a couple of images that show my alarm system's ability to do the required tasks:

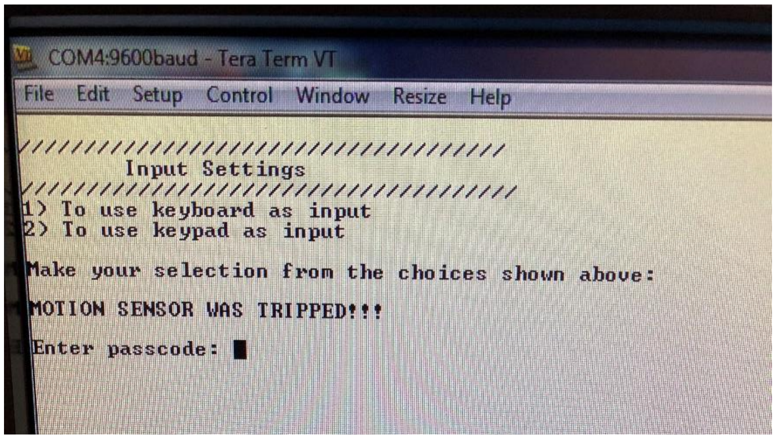

COM4:9600baud - Tera Term VT
File Edit Setup Control Window Resize Help

```
//////////////////////////////////////////
Main Menu
Threshold Temp: 70F      System Temp: 62F
Currently accepting input from: Keypad
Motion sensor is offline
////////////////////////////////////////////
1) Change Passcode
2) PIR Sensor Alarm Options
3) Temperature Sensor Options
4) Switch Input Method

Make your selection from the choices shown above:
```

The system can display the system temperature as read from the sensor using ADC, told the user what the current input mode was and whether or not the motion sensor was available for use.

Dario Ugalde
1001268068
CSE 3442


COM4:9600baud - Tera Term VT
File Edit Setup Control Window Resize Help

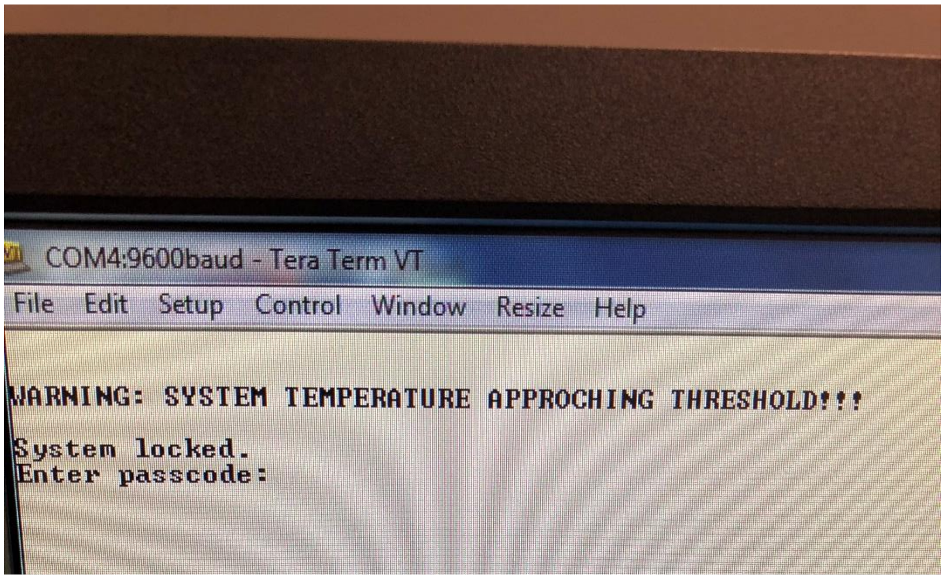
```
//////////////////////////////////////
        Input Settings
//////////////////////////////////////
1) To use keyboard as input
2) To use keypad as input

Make your selection from the choices shown above:

MOTION SENSOR WAS TRIPPED!!!

Enter passcode:
```

System notification when the motion alarm was active.


COM4:9600baud - Tera Term VT
File Edit Setup Control Window Resize Help

```
WARNING: SYSTEM TEMPERATURE APPROCHING THRESHOLD!!!

System locked.
Enter passcode:
```

Here, the system displays a warning if the temperature threshold is exceeded by the system temperature followed by a login prompt.

Conclusion

This lab served to help us demonstrate an understanding of the how to manipulate a microcontroller and utilize the onboard capabilities to implement a system. On top of that we got to work with a different microcontroller than the one used normally in lab to force us to use the datasheet and be able to better understand how to approach not just PIC18F452 and PIC18F4520 but any microcontroller that we might come across in the future.