Introduction to Assembly Language Programming with the PIC Microcontroller

Matthew Beckler
beck0778@umn.edu
EE2361 Lab 007

February 1, 2006

Abstract

In this lab, we are introduced to many aspects of working with the PIC Microcontroller. In building and programming a simple LED blinking circuit, the MPLAB IDE, processor flashing, assembly language programing, and basic PIC circuit setup are all introduced.

1 Introduction

The objective of this lab is to gain a basic understanding and skill set for working with the PIC microcontroller. We will be using the PIC to flash a LED which we can then use to calculate the internal instruction clock. We will accomplish these goals through the following steps:

1. Construct basic PIC circuit
2. Familiarization with the MPLAB environment
3. Flashing of simple program to the PIC
4. Installation and troubleshooting of flashed PIC
5. Analysis of source program
6. Timing of LED flashes
7. Calculation of instruction clock

The source code for this project, which was provided by our Professor, is included on the next page.

```
        list    p=18F452        ;simply sets the processor type being used
        include p18f452.inc     ;include file for our processor, needed for
                                ;  predefined things such as "PORTC"
        radix   decimal         ;this specifies that default radix is decimal
                                ;to override below in code,
                                ;  use b'01..' for binary, or 0xF2.. for hex
        cblock  0               ;define variable address location names (starting at 0)
        count2, count1, count0
        bits
        endc

        org     0               ;reset vector...program execution starts here
                                ;  (program address 0)
        goto    main

        org     8               ;high priority interrupt vector (can ignore for now)
        retfie

        org     0x18            ;low priority interrupt vector (can ignore for now)
        retfie

main:
        clrf    TRISC           ;set all bits of port C to output
                                ;  (0 for 'O'utput, 1 for 'I'nput)
        clrf    PORTC           ;and set them all to 0
        clrf    count0
        clrf    count1
oloop:
        clrf    count2
loop:
        incf    count0,f        ;4 clocks
        bnz             loop            ;8 clocks if taken, 4 else
                                        ;  (loop is 256*12 - 4 clocks)
        incf    count1,f        ;4 clocks
        bnz             loop            ;8 clocks if taken, 4 else
                                        ;  (loop is 256*(3068+12) - 4 clocks)
        incf    count2,f        ;4 clocks
        btfss   count2,4        ;8 clocks if bit 4 of count3 is 1, 4 else
        goto    loop            ;8 clocks (this instruction is skipped when count2 has
                                ;been incremented 16 times, which occurs when the above
                                ;loops have executed 16*(788476+12)-4 clocks
        incf    bits,f          ;4 clocks
        movf    bits,w          ;4 clocks
        movwf   PORTC           ;4 clocks
        goto    oloop           ;8 clocks
        end
```

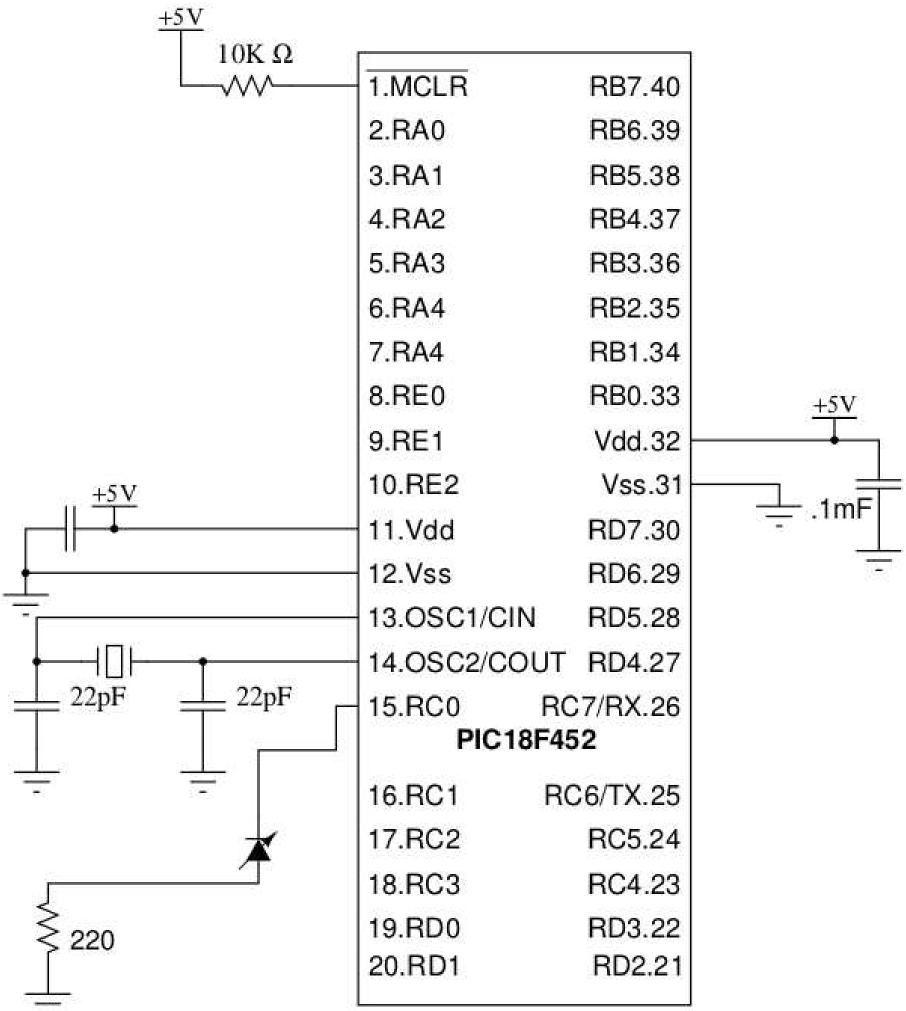

Figure 1: Circuit Schematic

2 Data

When the experiment hardware and software are properly setup, the LED starts blinking. We were instructed to try both a 5V and 3V supply, to observe any changes in the timing of the blinks. Our method of timing was to count the number of transitions, both **on** to **off**, and **off** to **on**, that occurred in a 30 second window. We then divided the number of transitions by 30 to obtain the frequency of blinking. The results are summarized in the table:

Voltage	Transitions per 30 seconds	Frequency (hz)
3V	90	3.0
5V	96	3.2

As shown, there was not a significant difference between operation at 3V when compared to 5V. In the rest of this experiment, we used the value of 3hz for the LED transition frequency.

To calculate the instruction clock and the oscillation clock, we need to compare the number of instructions executed to the frequency of blinks on the LED. This allows us to 'see' into the code's execution. For this source code, we determined the number of oscillations required to be approximately 12,615,828. This is based on the fact that most of the instructions require four oscillations, but some require eight or twelve. Taking the number of instruction and dividing by the time per transition, we can determine the frequency of the basic oscillator:

$$\frac{12,615,828\,instructions}{\frac{1}{3}\,second} = 12,615,828 \cdot 3 = 37,847,484\,hz$$

This is quite close to the correct value of approximately 40 mhz.

Since each instruction takes four clock cycles, the frequency of the instruction clock is $\frac{1}{4}$ the previous value. This value, 9,461,871 hz, is approximately the value of the external crystal oscillator.

3 Conclusion

The desired outcome of this lab was successful, in that we learned everything we were supposed to. Since the basic crystal and power supply section of the PIC circuit are mostly the same between labs, we are well prepared for the next laboratory investigation next week. The initial discomfort with assembly language programming has been overcome through the study of well commented code, and we are eagerly awaiting the next lab.