Benjamin Linam

[Bml42@students.uwf.edu](mailto:Bml42@students.uwf.edu)

EEL 4744L: Microprocessor Applications Laboratory

Lab 9: Input Capture

4/23/2018

**Objective**

The object of this lab is to introduce students to the input capture function of the 68HC11.

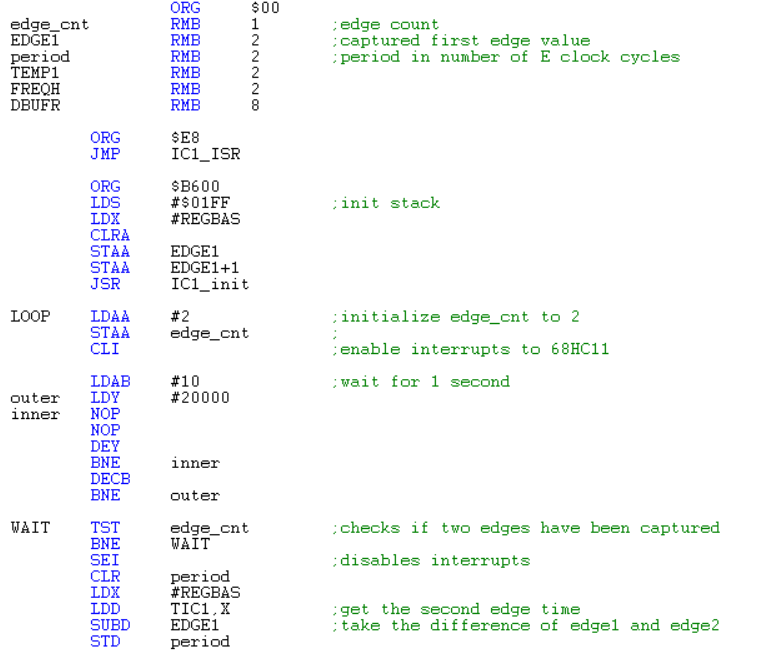
**Introduction/Background/Theory**

The 68HC11 has three input capture functions that interpret input signal transitions such as the rising or falling edges of a wave. The HC11 uses an internal time counter (TCNT) of which time stamps can be recorded and saved for later comparisons. This lab requires use of input capture 1 (IC1) to receive a periodic signal and determine its period and frequency by utilizing the interrupt method where an interrupt will be triggered for each rising edge of the signal for recording time stamps.

**Procedure**

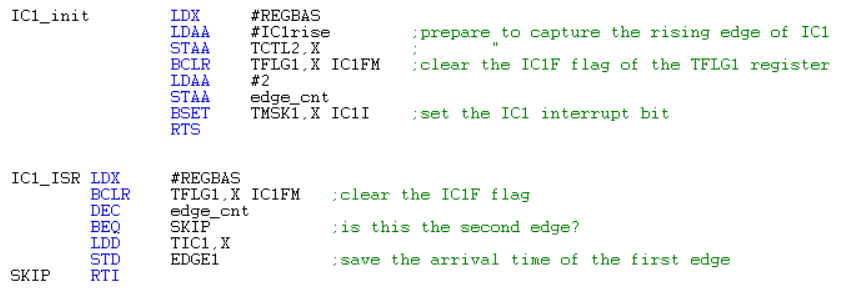
1. Since the program uses the interrupt method, the interrupt vector is initialized, followed by the stack and initialization of IC1 in the IC1\_init subroutine.

2. Figure 1 shows that the program enables interrupts, waits for one second and then tests to see if an input has been received and two rising edges have been captured.



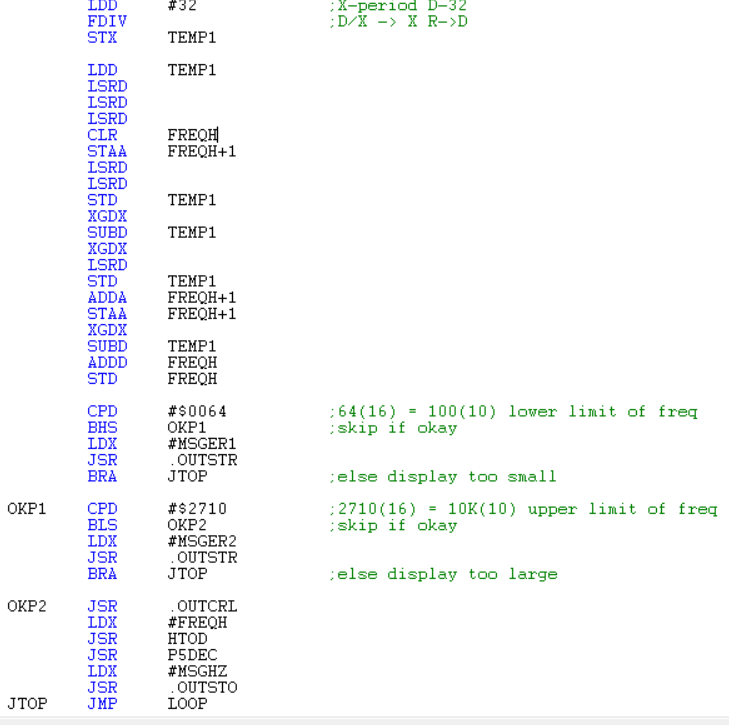
**Figure 1**: ASM code showcasing code controlling initialization of variables, one second delay, and loop used to attain period after two interrupts have occurred.

3. The two rising edges are recorded during the IC1 interrupt which occurs automatically without having to jump to its code. Figure 2 shows the code for the interrupt.



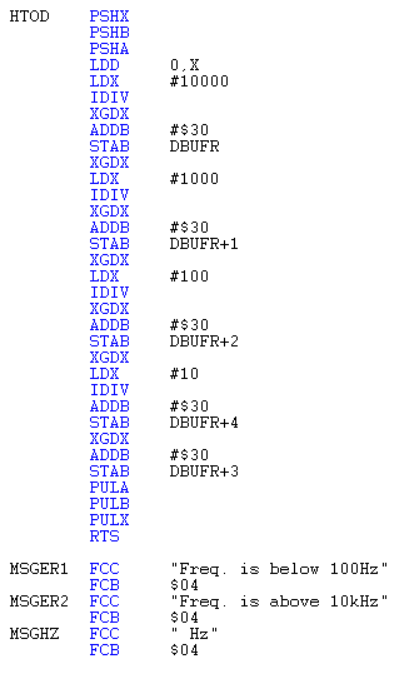
**Figure 2**: ASM code showcasing code initialization of IC1 for rising edges and the IC1 ISR which records timestamps for period calculation

4. Once the period of a signal (in E-clock cycles) is determined, the program utilizes an algorithm which translates the period to frequency as a hexadecimal value. Figure 3 shows that once the frequency is determined, it is compared to a lower limit ($64 which is 100Hz) and an upper limit ($2710 which is 10kHz) and if it is out of range, displays a message that the frequency is too large or too small.



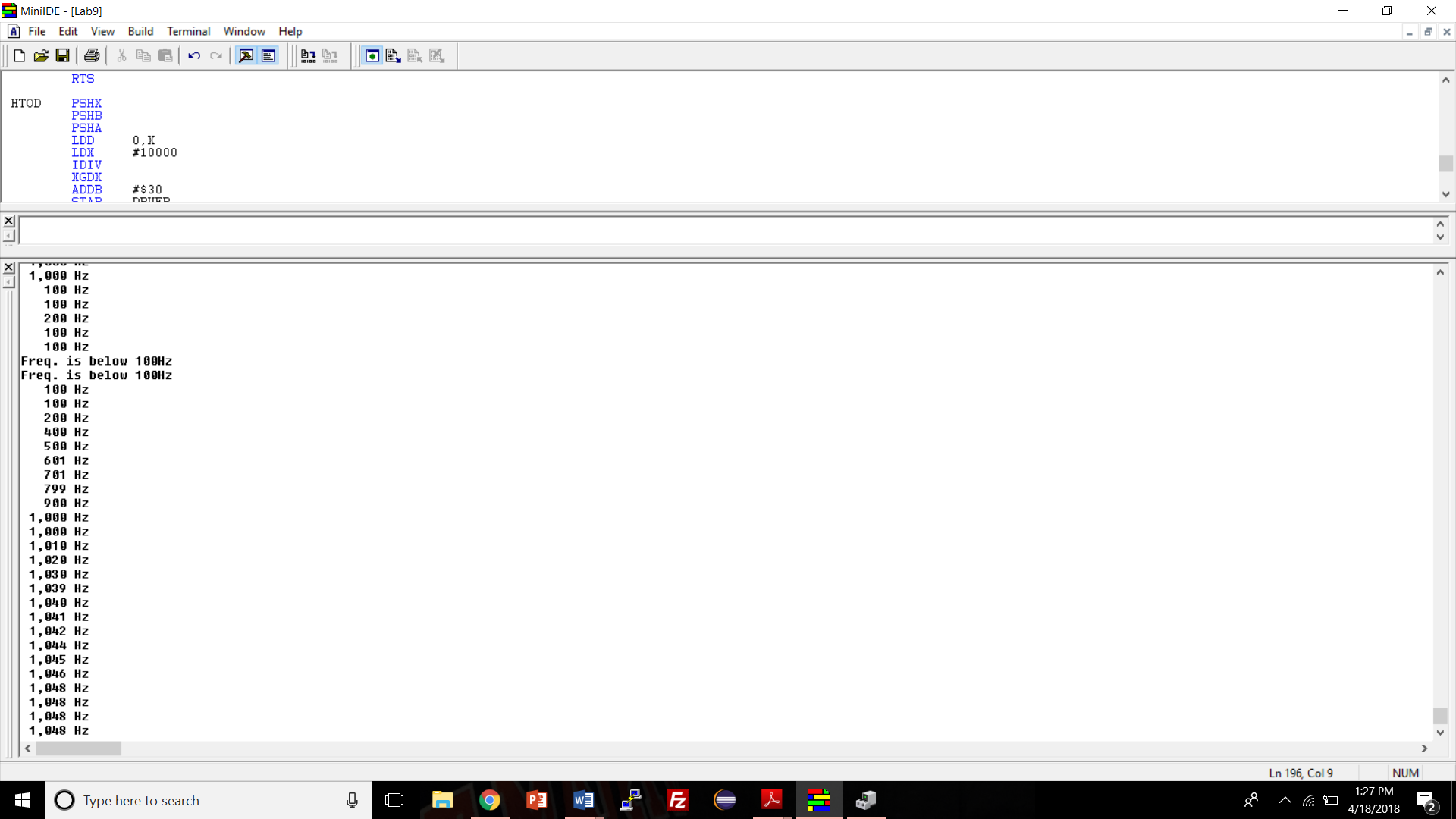
**Figure 3**: ASM code showcasing code which converts period to frequency and displays value to the Buffalo monitor.

5. If the frequency is within the accepted range, a message is printed to the Buffalo monitor displaying the signal’s frequency in hertz. A subroutine is used to convert the previous frequency in hexadecimal to its corresponding decimal value, shown in figure 4.

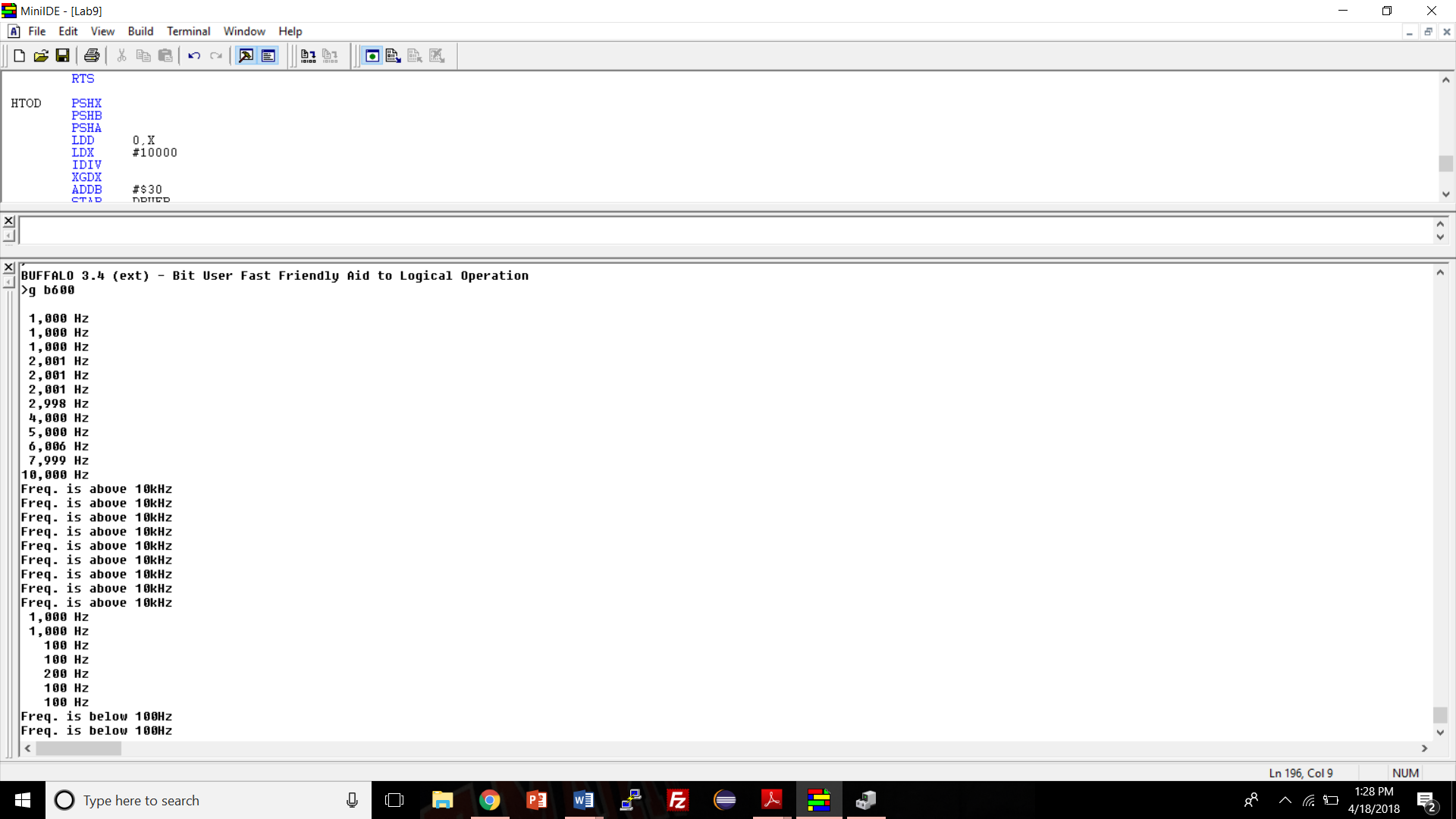


**Figure 4**: ASM code showcasing subroutine used to convert hexadecimal value to decimal frequency.

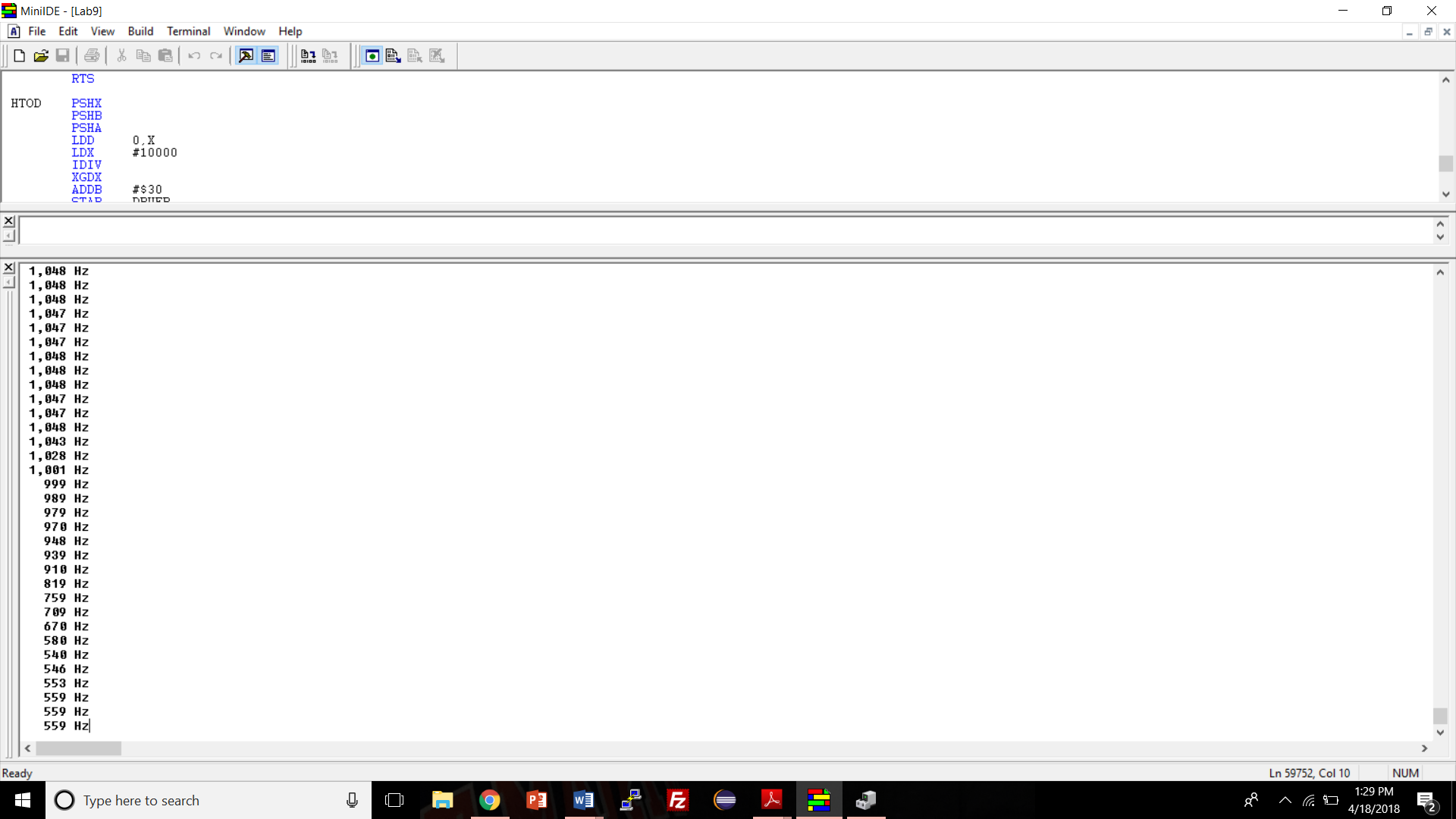
6. Once the frequency value is displayed, the program loops back to the second delay and waits for the next period to be captured. Three screen captures were obtained to show the program displaying correct results shown in figures 5, 6, and **7.**



**Figure 5**: Buffalo monitor displaying low frequencies and low frequency warning message.



**Figure 6**: Buffalo monitor displaying high frequencies and high frequency warning message.



**Figure 7**: Buffalo monitor displaying regular frequencies near neither the upper or lower limits of the range.

**Conclusions**

I encountered two large problems while completing the program for what should have been a relatively easy lab. Firstly, the textbook listed part of the original input capture example incorrectly. It showed that the IC1\_init subroutine accessed the TCTL2 register directly instead of as an offset of REGBAS. (STAA TCTL2 instead of STAA TCTL2, X). I did not notice this typo and left it in my code so the IC1\_ISR never activated. My second problem was encountered when trying to have the Buffalo monitor display frequency values. The first two digits of any frequency would display correctly, but any following digits would be shown incorrectly. I had forgotten to include a # sign in the subroutine which converts the frequencies from hexadecimal to decimal. Besides those problems, I enjoyed working on this lab.