**Project 1**

Histogram of Input Capture Events

By: Aliana Tejeda ([axt3290@g.rit.edu](mailto:axt3290@g.rit.edu)) and Scott Avery ([sra2270@g.rit.edu](mailto:sra2270@g.rit.edu))

February 10, 2019

**Areas of Focus:**

|  |  |
| --- | --- |
| **Team Member** | **Duties** |
| Aliana | Project Leader, UI Developer |
| Scott | Input Capture developer |

**Design:**

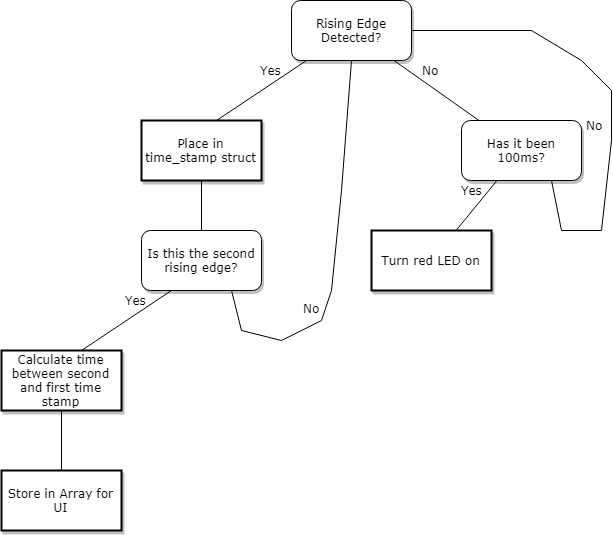
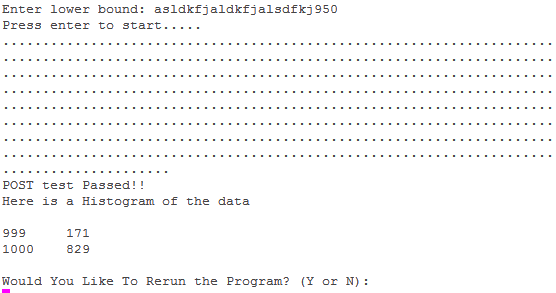
****

Figure 1: Logic of Main Program

**Our approach:**

1. **Timer Capture**
   1. *Initialize Timer 2 for input capture on channel 1 (PA0)*
      1. test to make sure PA0 sees a pulse
      2. create struct to save count once capture interrupt is triggered
         1. test to make sure two capture events are being saved to time\_stamp struct
         2. test math to ensure the correct values are saved into delta\_time
   2. *Enable Timer 2 interrupt on update events*
      1. once update event is triggered turn red LED on to signify POST fail
      2. turn off timer 2
      3. once update event is triggered turn green LED on to signify POST success
      4. turn off timer 2
   3. *Housekeeping*
      1. After a post, turn off the timer and clear it for the next run
      2. Make sure there is a way for each function to talk to each other
         1. Respected variables are set to the correct value for program continuation.
         2. return to the correct functions in order in response to situation
      3. Have a catch all if any errors occur
2. **User Interface**
   1. *Startup* 
      1. Do not run the program immediately
      2. Give the user the ability to edit the bounds or continue the program
      3. Display to the user that the program is running
         1. chose dots for this
      4. run the POST test
         1. let the user know that the POST has failed
            1. give the user the chance to rerun the POST
   2. *Output*
      1. make sure 1000 values have been saved to delta\_time
      2. output histogram
      3. Clear and understandable instructions to user
   3. *Main Program*
      1. ensure correct and valid inputs are entered into terminal
      2. Give the user the ability to run again or edit bounds
      3. allow user to seamlessly go through the different stages of the program by the click of a button.
      4. Allow for the user to exit the program if they wish to

**Test Plan:**



**Figure 3**: Rejects non-digit characters

**Figure 3** shows how letters are rejected from the lower bound input but are not rejected from being outputted to the USART. This could cause some confusion to the user and should be addressed for clear operation.



**Figure 4**: Rejects lower bound less than 50

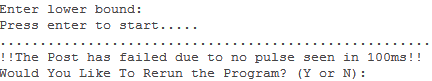


**Figure 5**: Rejects lower bound greater than 9950

**Figures 4** and **5** show an example of the result of the verification process. Hitting ENTER here will restart the entire application again. This could be improved to just prompt for new bounds. Also the bounds are never output to the terminal to allow a user to verify that these are correct.

**POST:**

Timer 2 generates an interrupt if 100ms with no rising edge is detected the red LED is turned on.

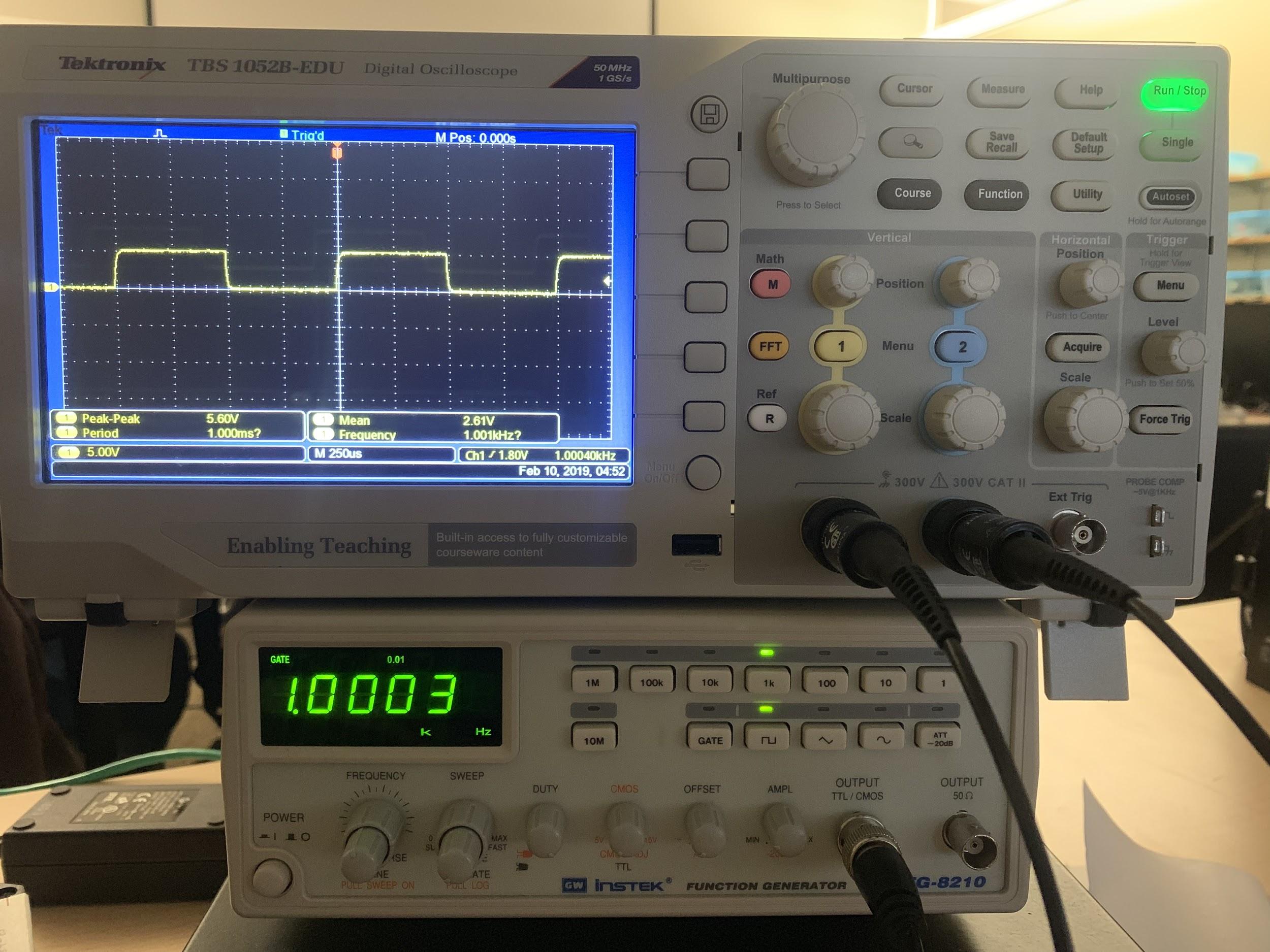


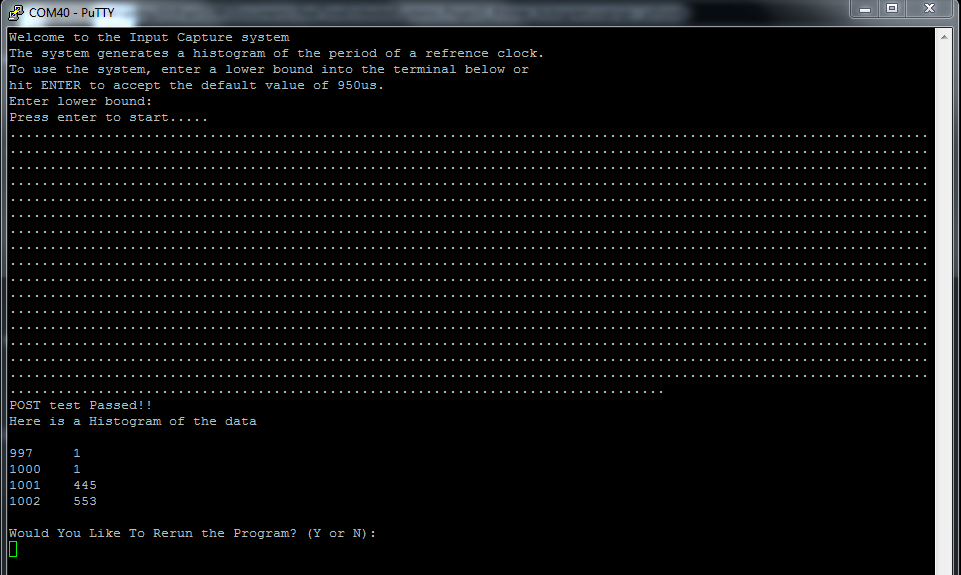
**Figure 6**: POST failed

**Figure 6** shows the terminal output of a failed POST. This happens when a rising edge is not seen by pin PA0. This will occur if no output waveform is input through PA0 , if the signal has a rising edge after 0.1s, or the frequency is to fast to the board to analyze.

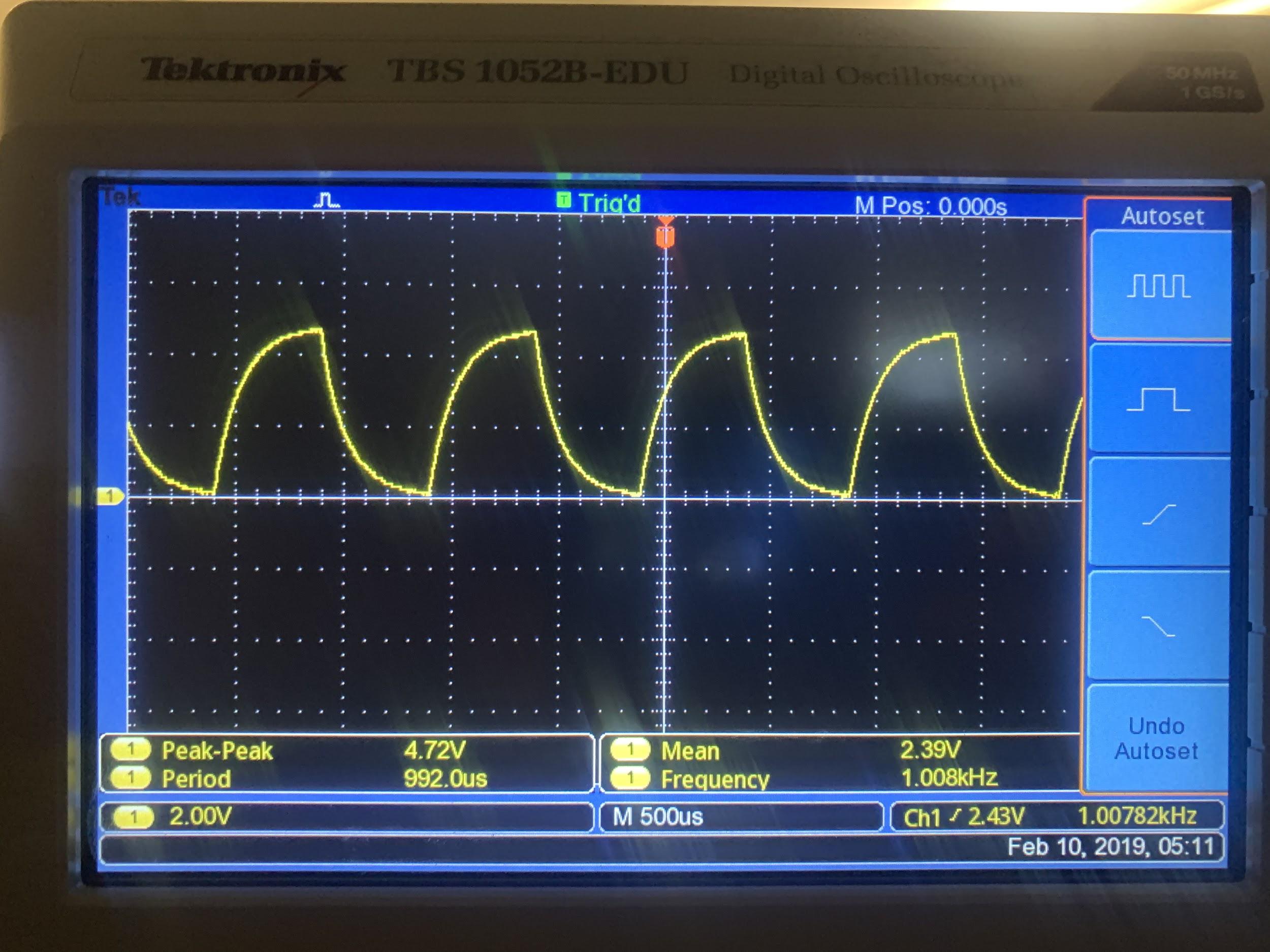
**Testing With 1000 Hz from Function Generator**

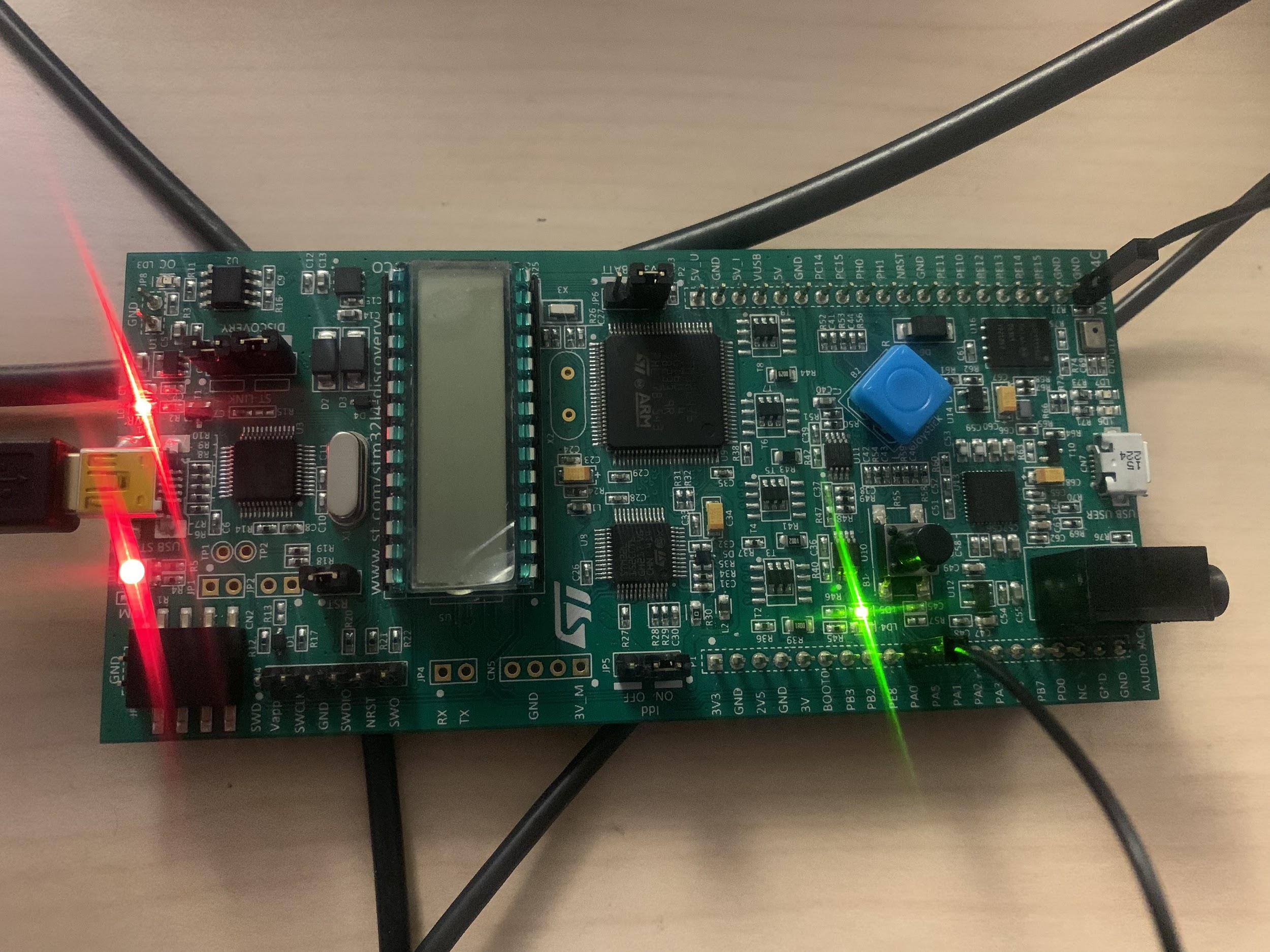
****

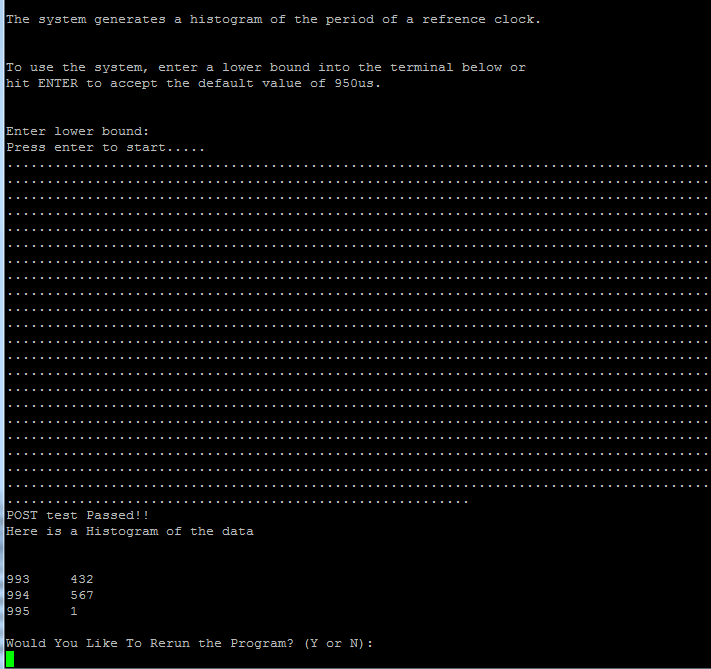




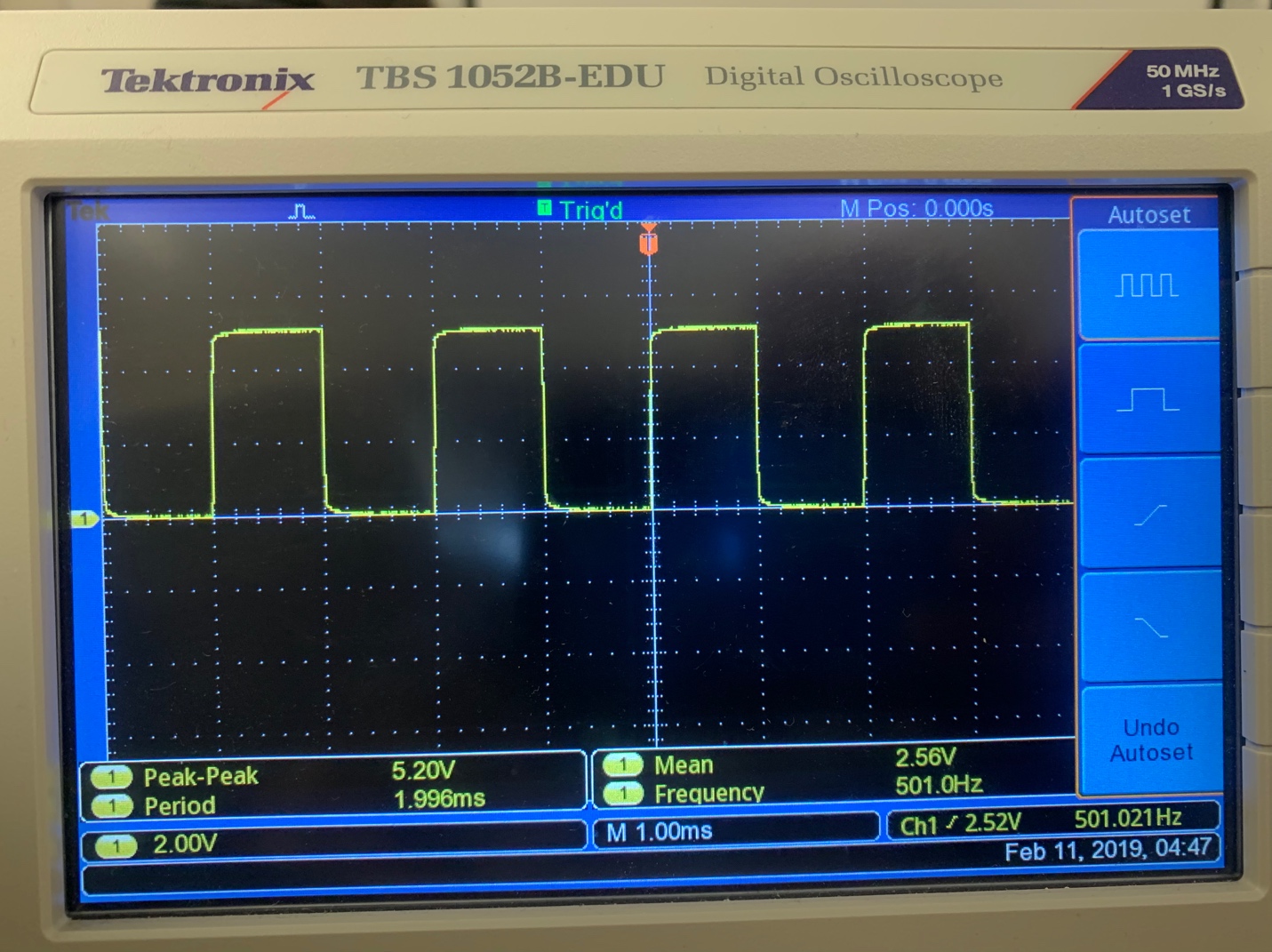
**Testing With 1000 Hz from Oscilloscope Test Signal**

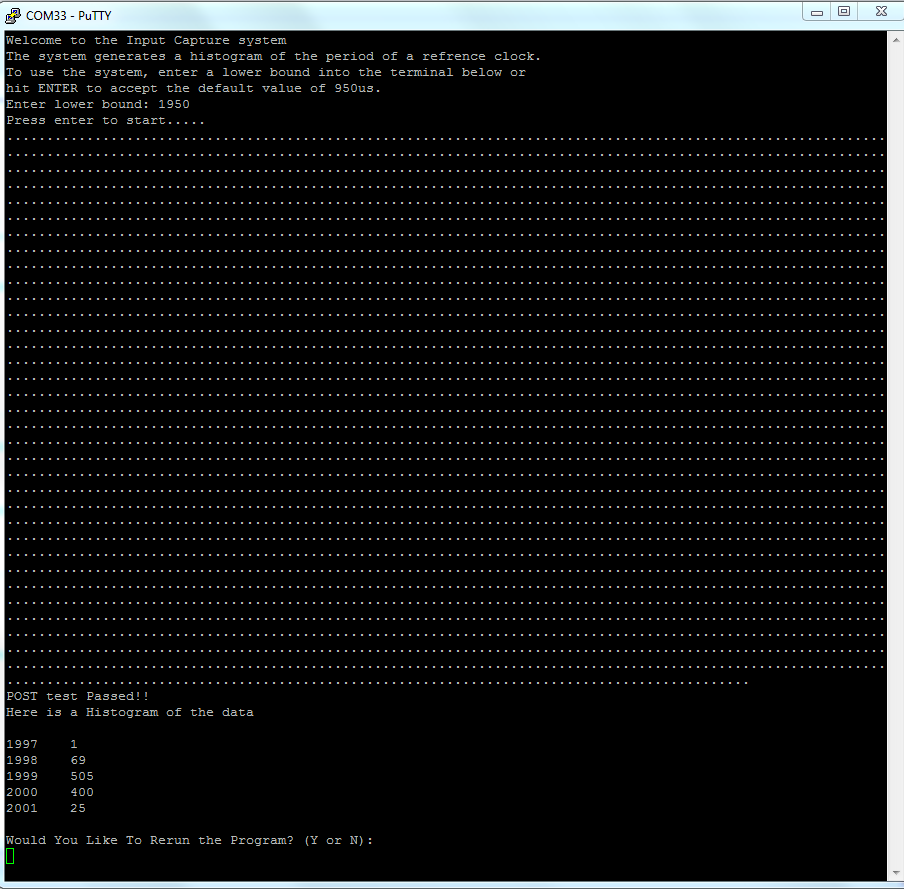
****





**Testing With 500 Hz from Function Generator**

****

****

**Results:**

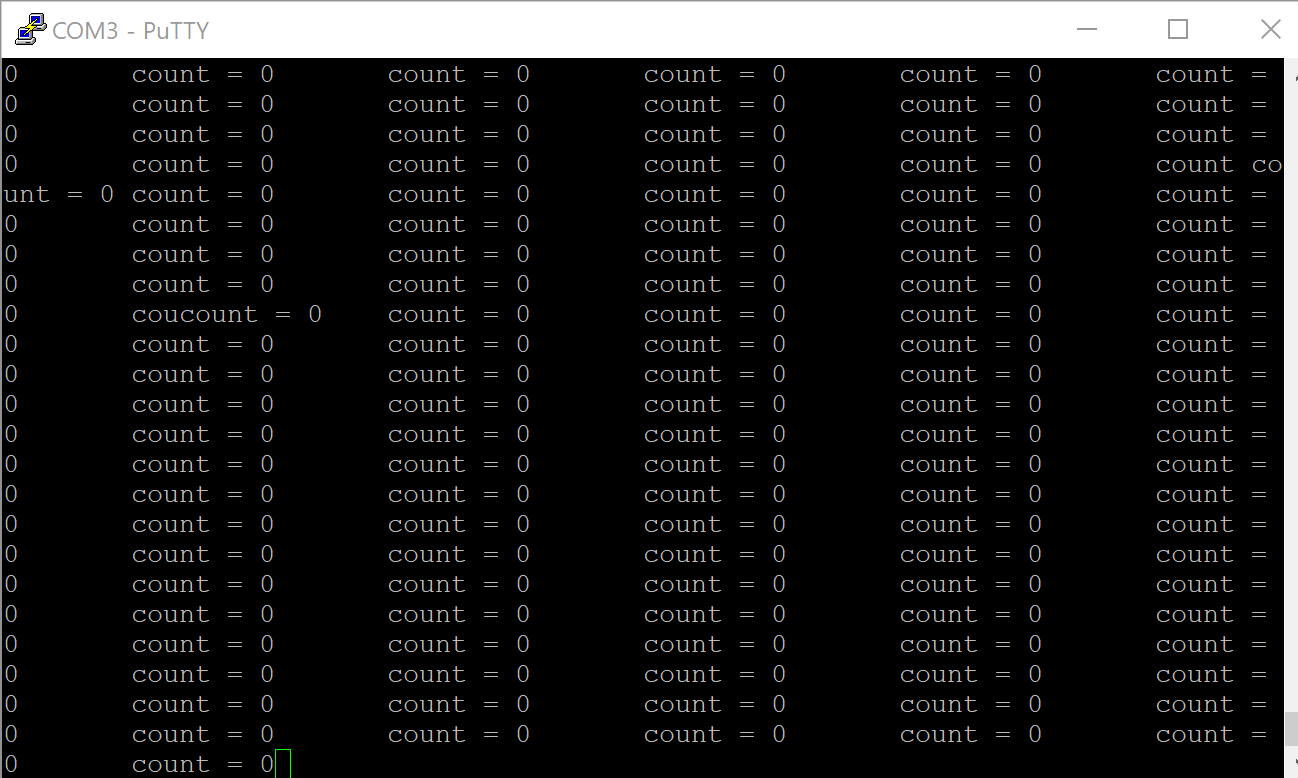
The system is able to accurately read an input signal and report back the time between two rising edges, but there are limitations. An input signal of 100kHz is not readable by the system. The reason for this is most likely due to the schmitt trigger on the pin. The fast oscillations are leveled out and input capture on timer 2 is never triggered.

A signal of 100 Hz is readable of the system but takes about three and half minutes to complete due to measuring two rising edges for each of the 1000 delta times taken.

**Lessons Learned:**

Upon startup of the project, code for hardware and software configuration were all set up in the main file of the project. After further development, we learned that for the project to run as smoothly as possible, it would be best to separate our hardware and software configurations. This allowed for future debugging to be less stressful than it was before.

A example of such is when we were debugging the issue of our timer not counting. A snippet of such is shown below.



**Figure 7**: Timer2 - No Count

**Figure 7** shows the PuTTY terminal showing a count of zero for Timer2. Many hours were spent on trying to figure out where we went wrong in terms of configuring the timer. In the end we learned that, in order to write and configure a timer, you have to clear the CCRxE bit first.

Another lesson we learned is that when it comes to sorting the array after a POST, using the regular brute force sorter is more stable than using a recursive one. A sorting algorithm that loops through the array twice is more stable when stack space is limited, than a sorter that would compare two buckets then give a verdict. Quicksort put too much stress on the stack, and caused it to fail, while bubble sort is slow but sufficient and stable, because most of the values in the array are the same value.

**Conclusion:**

To improve the design of this application the time\_stamp struct would have to change such that the measurements taken would be dela\_time[i]-delta\_time[i-1] with i > 0. This would improve the speed of our application but would sacrifice perciscion. The application takes 2\*ARRAY\_SIZE of measurements which limits its scalability but allows for precise calculations of time between rising edges. Modular design allowed for easy testing and debugging in the long run allowing for easy integration and correction for the various stages of the project.