SWEN-563/CMPE-663/EEEE-663

Real-Time and Embedded Systems

Project 1 - Feb. 13, 2017

Zachary Weeden | zdw7287@rit.edu

Overview:

We were given the task of sampling measurements from an oscilloscope and then build a histogram of said measurements, enumerating duplicate samples displaying the findings of 1000 samples. We were required to implement a Power On Self Test where a smoke-esque test was ran, verifying that a pulse was seen within 100milliseconds on initialization. User interaction was also required in this project, thus we made use of the UART serial connection. If the POST failed, users were notified and prompted if they wish to run the initial test again. On a passing POST, the sampling of measurements would then take place.

Prior to sampling, the user was prompted if they wanted to change the bounds of which the histogram would show, the default being 950 microseconds to 1050 microseconds. The test would then run for 1000 samples. A timer was enabled and then disabled on the detection of an input capture on the GPIO pin. This process repeated for each of the 1000 samples and results were then tallied and then displayed.

Areas of Focus:

Zachary Weeden: Responsible for entirety of Project 1 (see [https://github.com/zweed4u/Real-Time-and-Embedded-Systems/](https://github.com/zweed4u/Real-Time-and-Embedded-Systems/tree/master/Projects))

Dinesh A Bashkaran: N/A

Analysis/Design:

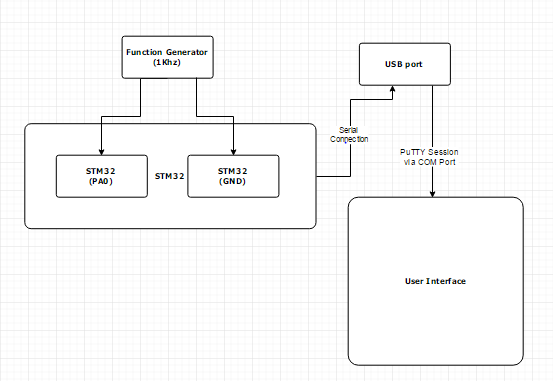
With the STM32 and provided components, I wrote in C the functions and source files needed to configure a Timer and GPIO pin to indicate when a rising edge input was captured on the GPIO pin.

I isolated the Timer2 configuration and helper functions in their own file. TIMER.c and TIMER.h respectively. In the C source, wrapped bit operations on registers were given useful names for use throughout the rest of the project.

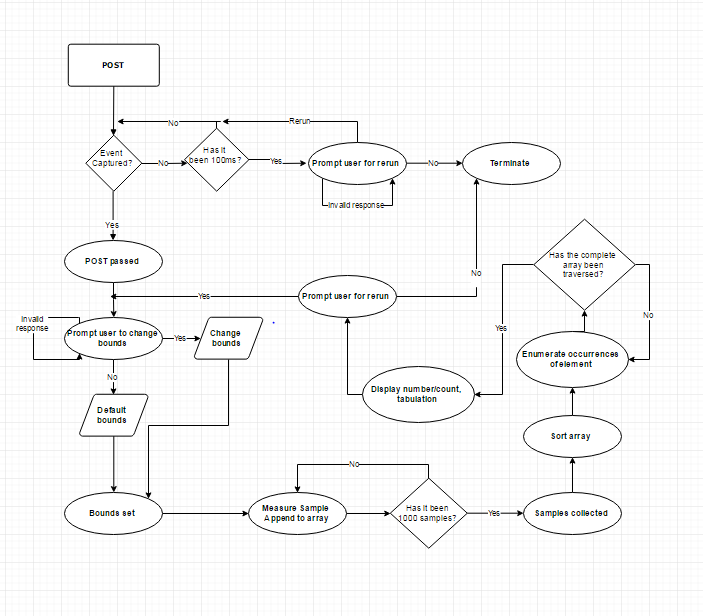
Another asset of this project included was the input\_pa0\_test source. I modified this code from the original source received from Professor Kiser. This file served as the conduit for getting the GPIO pin configured as well as the timer.

More borrowed resources included a binary search and a sorting method I implemented later in the program. I isolated both of these functionalities as well. The sorting file made the array of measurements in ascending order which allowed for binary search enumeration of elements very efficient. It should be noted that this was an aggregation of snippets found online to accomplish a similar task.

These, in conjunction with my main source and other supplied files such as the UART files, comprised the first project’s codebase.



This is the Hardware Block Diagrams of components used



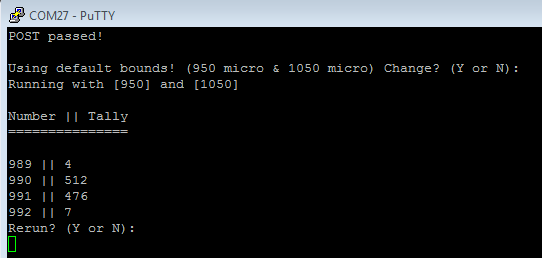
This is the general program flow of my design.

Test Plan:

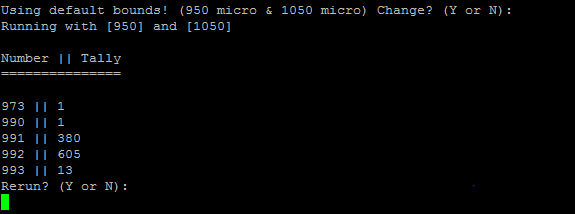
The POST test acted as a validity test in that it ensured GPIO pin functionality in receiving a signal as well as the proper configuration of the Timer. Using an exclusive POST function, I began a timer and made a variable to keep track of when it was initialized. I then entered an indefinite loop waiting for an input capture signal to break out of the loop. This indicated that a rising edge was seen and therefore the POST was successful. There was a condition inside the infinite loop to ensure that we had not exceeded the defined macro of 100,000 microseconds (100 milliseconds). If this was the case then the test had failed due to timing out. A failed POST sent control to a FAIL function in which the user could choose to run the POST again via UART prompt. The FAIL and the POST function returned an integer value. 0 for a failing, non-rerunable POST and 1 for a passing POST. A variable in main was declared as the return value of the POST function which allowed either the main program to continue operation or to terminate due to a failing POST.

Project Results:

We expected to get many values around 1000 for our sample measured times. With the function generator hanging around 998.05 Hz, the STM32 board received the following:

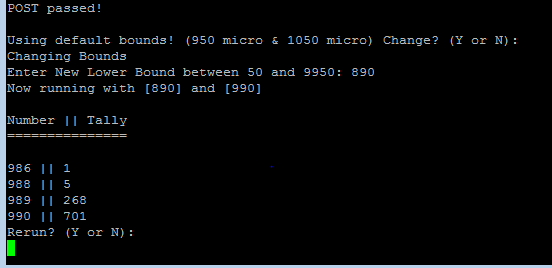


This histogram makes sense, however the summation of the tallies is off. This is due to a measurement falling outside of the default bounds. The tally summation yields 999 out of 1000. The 1 sample missing can be attributed to the first wave being ill-measured. More specifically, the first pulse measured was not the true full pulse and began measuring mid-wave. Running several more tests yield the full 1000 sample and measured times.



Here the 1st sample is noticeably smaller than the rest of the measurements. This is due to the nature of captured samples aforementioned. This is most likely a pulse that has run awry but no so much as to escape the scope of 950 microseconds to 1050.

Keeping the function generator at 999.20Hz, we adjust the bounds so slightly to see the skew of the histogram. The bounds are then changed to 890 and 990 microseconds respectively. The sum of this tally yields a greater difference to 1000. 975 counts of pulses out of 1000. The 25 missing samples are missing due to them being out of the scope of the user defined bounds of 890 to 990.



Overall the consistency was adequate. Expectations were high and albeit small, a spread of 4 different wave periods was unexpected.

Lessons Learned:

This project required user interaction over the serial connection. Prompting the user deemed to be quite the feat as the UART\_Read() method is an interrupt and triggers with any input. This made entering a series of keystrokes (setting bounds) difficult. The received byte was continuously monitored ensuring that “\r” (0xD, hex ascii) wasn’t the last character.

Along with the manipulating of user input, organization of arrays was crucial in that, preparing the data for display required much work. A data structure such as a hash table or dictionary would have been extremely useful in this project.

A final thought on the matter of setting bits in registers; it proved to be very tedious. The reference manual and reading over the various uses of registers and their respective bits/flags was a crucial part in the success of this project in configuring the timer used to measure the period of the wave.