ELEC-2220

Computer Systems

FROM: Basil Moledina

TO: Dr. Victor Nelson

DATE: April 29th, 2022

Final Project Briefing and Documentation

Description and Summary of the Final Product

For the final project, a program is to be written for the STM32F411VETx microcontroller so that it becomes a programmable function generator (PFG). The program must not only create a digital waveform, but it must also be user-adjustable. It can either generate sawtooth waves or triangle waves, and the period of the wave may be set to [OFF, 0.5s, 1.0s, 1.5s, 2.0s, 2.5s, 3.0s, 3.5s] . Pressing the button once in two seconds changes the period and pressing it twice switches the wave type / pattern. This allows for 15 different possible waves to be generated (including OFF / no wave). Besides using uVision's Logic Analyzer tool to graph the waveform, LEDs must also be used to output information about the wave. The blue LED toggles after every period as another way of showing the wave's duration, and the rest are used to show which period has been selected (i.e., 0.5 s is Period 1, so the LEDs would display the number 1 in binary).

The PFG utilizes several hardware devices and controllers that have been discussed throughout the course. GPIO ports and controllers are used to drive LEDs and read the board's button. This allows for an even wider user interface, furthering interactivity and device status information to the end user. Extending the capabilities of GPIO ports are external interrupts that allow multitasking and edge detection. This relieves great stress on the CPU when it comes to counting button presses and debouncing. Arguably the most essential device to the FPG, however, is the built-in timer and its interrupts. This is what is responsible for accurately dictating when to plot a new data point and counting how much time has elapsed after a button press. Without it, period and frequency may not fall within a required range (i.e., real time applications), thus making it intolerable for many applications.

The mathematics of the timer is noteworthy because of its simplicity and the burden it alleviates. One requirement of the project is that there be 100 plots per period. To find out how often to display a new data point, the period can be divided by the 100 plots. Simplifying this ratio for T = 0.5 s gives 5 ms until the next plot. As it so happens, every other period to be displayed is a multiple of 5 ms. This means that to adjust the period, wait an additional 5 ms. From the program's perspective, this means wait an extra timer interrupt (if your timer is set to trigger every 5 ms) in some while loop. Because of this property, the timer only needs to be set once, and the autoreload value and prescaler never need to be modified again.

The actual sawtooth and triangle shapes lend themselves to this project. The sawtooth is effectively a line that resets back to 0 at some point, and the triangle waveform is just a piecewise function consisting of two lines. Because all these shapes break down into something linear, this allows a "graphing technique." Since the sawtooth is a line that goes all the way from the start to the end of the waveform, it's slope or "rise over run" can be taken (max value / 100 plots) . DACvalue effectively acts as a y-value, some variable that changes with respect to time, and the slope can be added to the current DACvalue to get the next plot on the line. The same method is utilized for the triangle wave, except that since the wave is piece-wise, the slope would be (max / 50 plots). Since the DACvalue changes at the same rate – only now it's decreasing on the latter part of the triangle wave - the same slope can be used to generate this second half of triangle, only DACvalue will be subtracted by the slope to get the next plot value.

Control Structure Flow Diagram

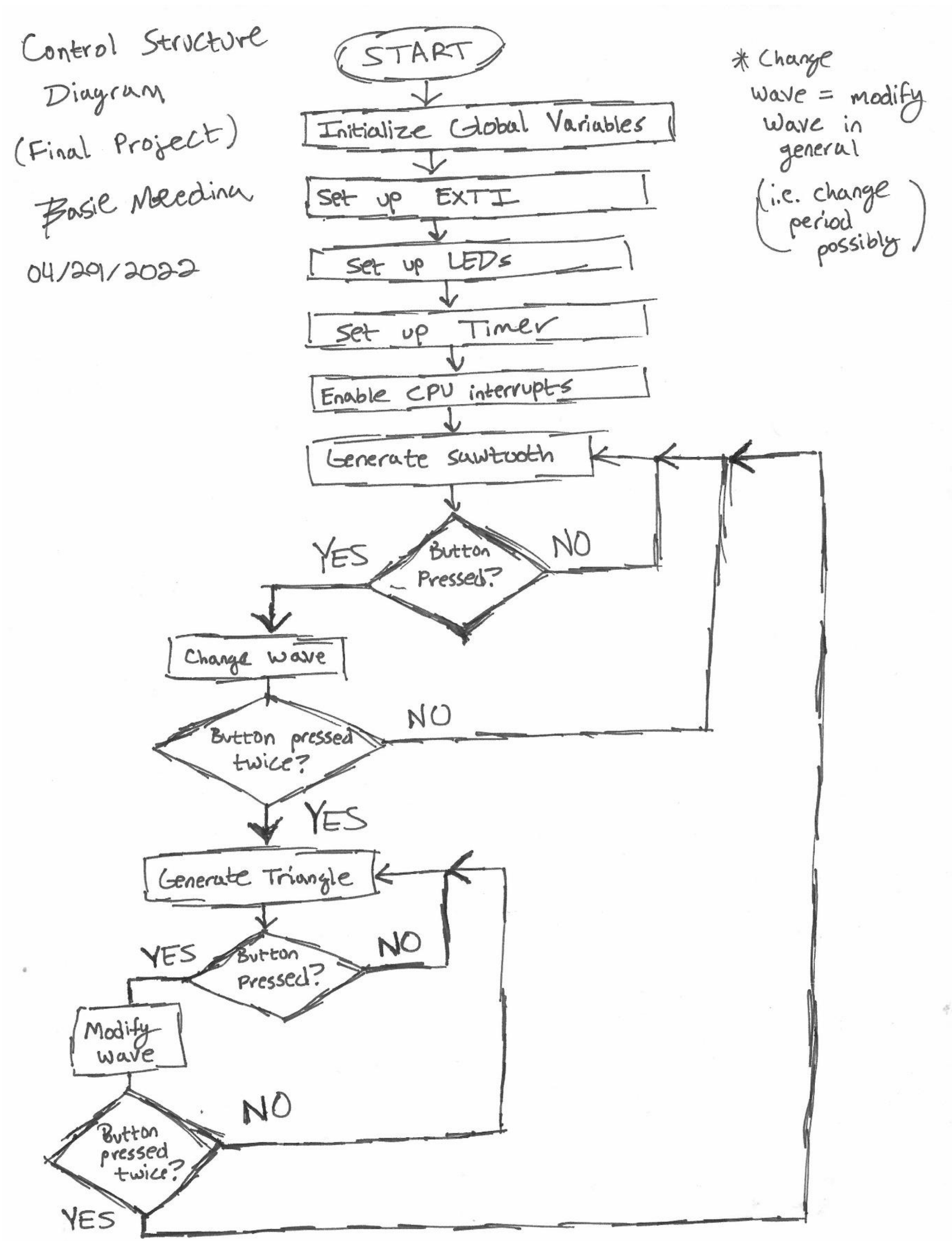

Figure 1: Control Structure Diagram of the PFG software

Results and Graphics

What follows in this section are some snapshots of the PFG in action. Waveforms are graphically displayed in the Logic Analyzer, with data (DACvalues) ranging from [0, 4095] to represent 0 to 3V. Figures 2 through 5 show that the program is capable of generating the two waveforms and multiple periods for each. For reference, though, both waveforms will use the same two periods so comparisons can be made more easily.

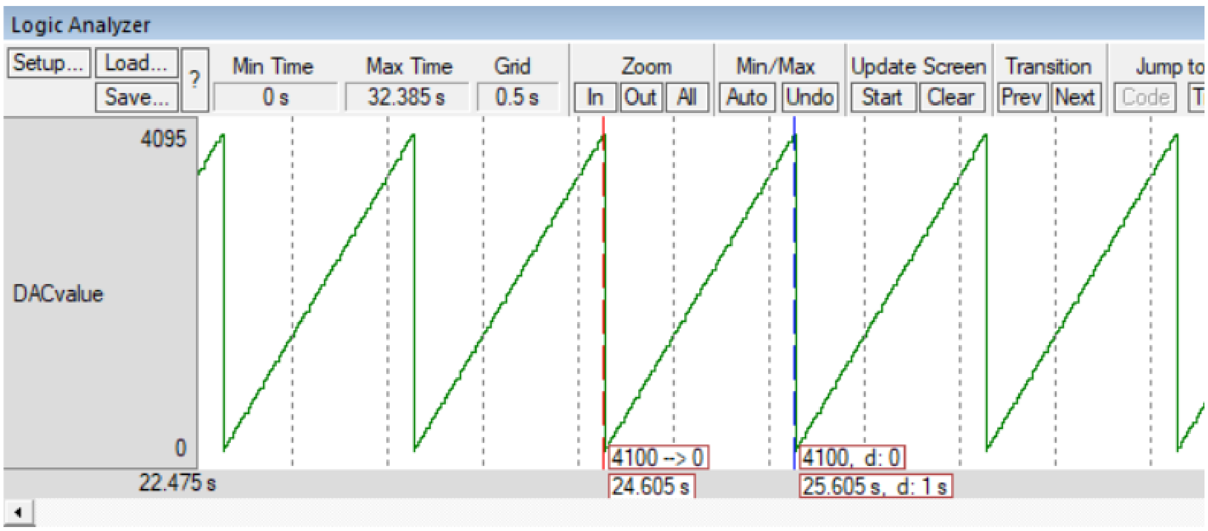

Figure 2: Sawtooth Waveform, T = 1 second

Figure 3: Sawtooth Waveform, T = 2.5 seconds

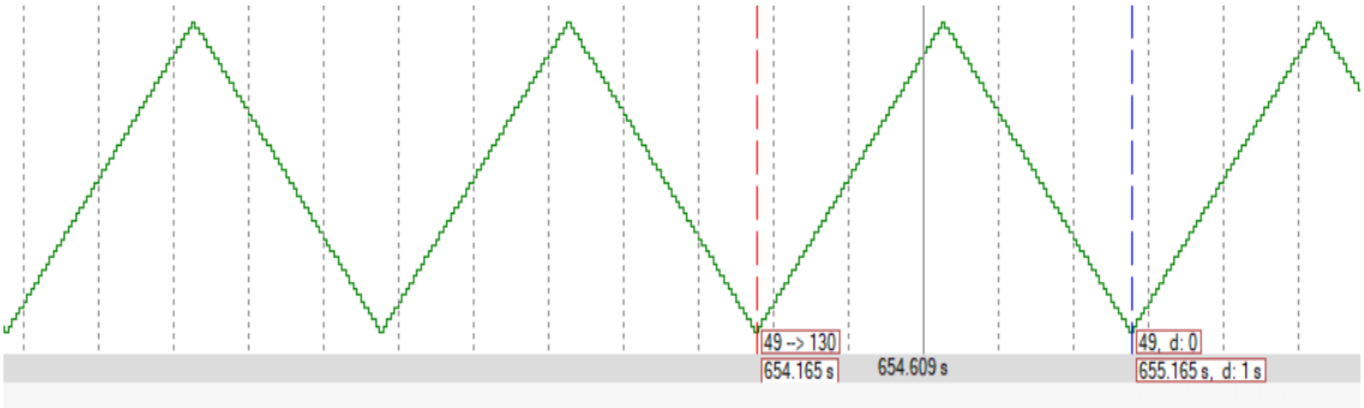

Figure 4: Triangle Waveform, T = 1 Second

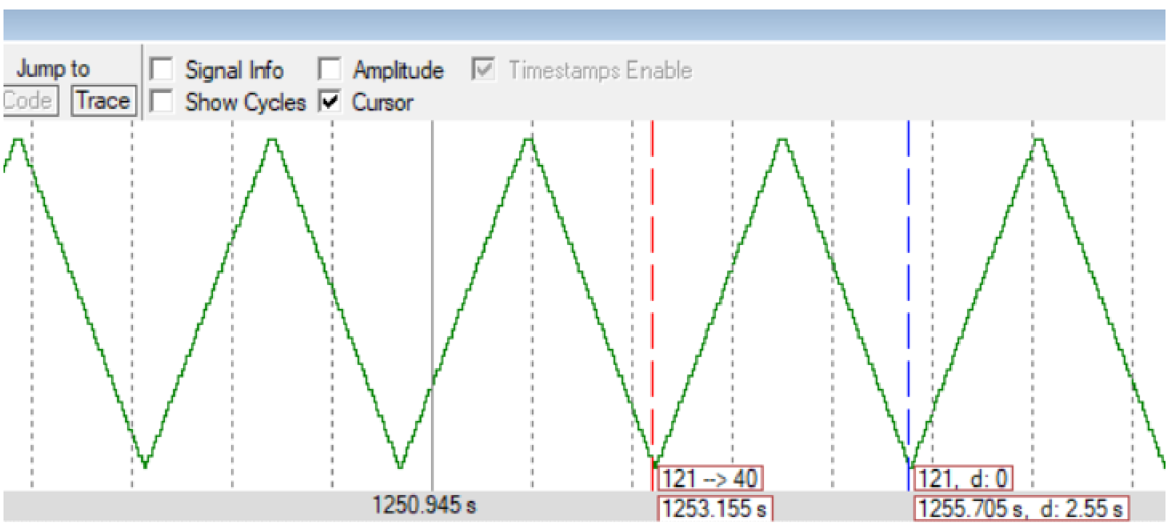

Figure 5: Triangle Waveform, T = 2.5 seconds