Gabriel Emerson
ELEC 3040
Due 10/15/21

Draft Experimentation Report

This report designs, tests, and validates the functionality of the stopwatch timer circuit and program. This experiment creates a simple circuit that will act as a stopwatch by using the keypad for counting up when start/stop is pushed and reset when clear is pressed.

There are a few states we should monitor to really understand the data being represented to us. First are the values of the stopwatch as the timer is enabled. This is crucial to knowing how the stopwatch counts (by what variable and at what time). This is also used when the count is cleared and everything is set back to 0. We will use this data to see what the value of the stopwatch should be and what it currently is (this includes previous state, current state, and next state). The next state to monitor is button state after either button has been pressed. This data will help us to show what the button is doing before and after the button is pressed. These are the main two parts of this experiment, if we can gather the information about the keypad (count state at the timer) and the buttons states themselves, we can then monitor all states of the lab and verify how each part of the circuit works. We should lastly look to verify the timing of our stopwatch to ensure we are counting up a number at approximately once per second.

In order to effectively test our design, we will use 5 DIO plugins to view the current state of count (essentially the number our stopwatch is currently at) and also view the state of the keypad button press. First connect DIO[0-3] to the output pins that show the value of the stopwatch count (these are pins PB3-PB6 respectively). Then connect another DIO (does not matter which one) to the input signal from the keypad so that we may view when a button has

been pressed (I prefer to use DIO[15]). After setting up these DIO pins we are ready to view the direct inputs and outputs of the stopwatch on the Waveforms Static I/O viewer. When opening the viewer we should note that the stopwatch output can be viewed in binary LED forms on DIO[0-3] and the input signal can be viewed on DIO[15]. We should lastly connect our oscilloscope to the counter output to view the timing delay between each iteration of count.

Now that we have the stopwatch setup and ready to run, it is time to test it. Start the program and view the contents of the static I/O viewer. The stopwatch count should be at 0 (the count in viewer will appear in binary since it is 4 LED's, this means 0 will be shown as 0000). The input signal, DIO[15], should be 1 (high/on) to show that the input is working. This is important to remember that the input LED is on because the configuration of the keypad is that it sees the button being pressed and pulls the signal low, this means it should be high until a button is pressed. After verifying the state of the input and outputs before starting the stopwatch, we should then press the start button to view the behavior of what happens next. When we press the start button, we should view the input LED turn off, then back on. We should then see the output LED's begin to count up in binary form. The output LED's should be counting up in the same order shown in Table A. After the first second, the input should be back to 1 since no button is being pressed, and the count should be 0001 (same as decimal 1). At this point the Static I/O viewer should look like the one shown in Figure 1. Now as the stopwatch counts up, we can view the oscilloscope to verify the correct timing between each count. This should be right around 1 Hz frequency on the scope. The pulse with values is shown in Figure 2. Now all that is left to test is our reset button. After pressing the start/stop button we should notice the counting come to a complete stop. After this step the reset button can be used. Pressing the reset button will reset the

entire stopwatch back to its original state as if the program is just being ran for the first time. We have now checked all possibilities for the stopwatch program and can say that the stopwatch program has been tested fully and runs successfully.

When stopwatch is counting up, the numbers should be in this order before looping back to the beginning.

Decimal	**Binary**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Table A – Values for stopwatch program

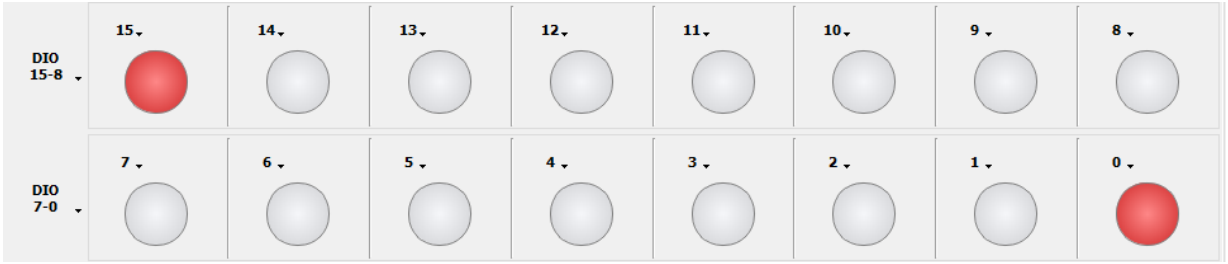

Figure 1 – Static I/O in Waveforms after start and 1 second pass

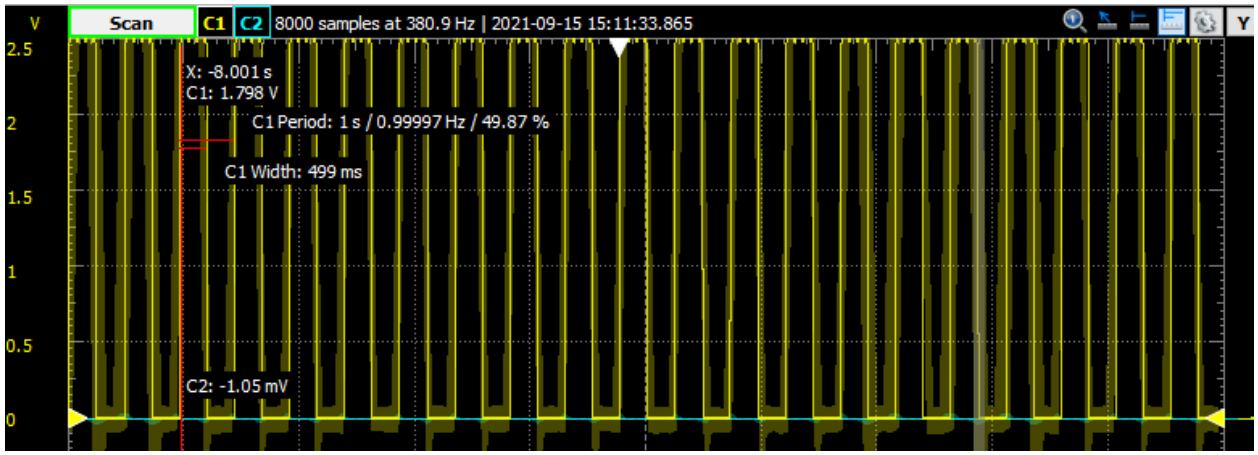

Figure 2 – Oscilloscope measuring the time between counting each step for stopwatch

The data we have received from the testing of our stopwatch shows the values of the input (keypad press) and the output LED's counting up from 0 to 15, then it starts over at 0 again. From the Figures above we can also see that the value of the keypad-pressed input is 1 until a button is pushed, then it will go low. We can also see the timing of each count increment pulsing at ~1Hz, which is exactly how often it should. Since we have verified both the keypad functions and the count output is working, we can now verify the overall functionality of the complete experiment.

In conclusion, we have shown that our test of the stopwatch program works according to plan. This was a very basic stopwatch program which made the design and testing both straightforward from the beginning. However, we could possibly build off this in the future by trying to count in different increments, or even using more buttons on the keypad to do different things on the stopwatch (such as the overlapping feature used on most stopwatches). The next step in testing our stopwatch is to try all different buttons from the keypad to see if it will change any settings in the program itself. For instance, we could check if the reset button will adjust the

value of the counter while its counting. This should do nothing since reset should only work when the stopwatch is stopped, but this and maybe other buttons could affect the values of the stopwatch. This could be built off as well to change the different increments of count, to make the duty cycle of the count change according to the button that was pressed.

References

STMicroelectronics. “STM32 Nucleo-32 Boards (MB1180) – User-Manual.” Nov. 2018, https://www.st.com/resource/en/user_manual/dm00231744-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf. Accessed 15 Oct. 2021.