TA Andrew Gerber 10/1/2023

Lab 3 Report

Benjamin Meads
A02323437
Nick Templeton
A02268123

Objectives

In this lab, we were instructed to create a ten-bit binary counter displayed on the bar LED given in the lab kit. Our LEDs needed to increment at a frequency of 2 Hz. We also needed to implement 3 buttons. A green button that will start the counter. A yellow button will pause the counter. Then finally, a red button that will reset the counter and the LED's. Our objective was to become familiar with slowing down the clock to the slowest MSI setting possible. Then to familarize ourselves with more of the GPIO pins. Also to understand the concepts of open-drain methods and pull-up resistors. A large point of this lab was to get our program to run on our microcontroller. Then to interface our microcontroller with the LED display as well as GPIO switches.

Procedure

Before Lab, we drew a schematic of how the LEDs, the resistors, the buttons, and the microcontroller board were supposed to be connected. Our first iteration of the schematic was pretty close, but before the lab began we didn't have a full understanding of how an open drain system worked. We drew our LEDs connected to the ground, instead of being connected directly to the desired pins on the microcontroller board. When lab began, we got a rudimentary version of pseudocode ready to go and started coding. We used PUPDR and OTYPER to configure our 3 pins for the buttons to pull up, and the 10 pins for the LED to open-drain respectively. This took a lot of iterations and logic improvements. When implementing a debouncer to our buttons, we used a small loop of executions to give us a delay of 100ms to properly debounce the button. When reading the button presses from GPIO_IDR, we did it in a counterintuitive way according to one of the TAs. Instead of using bit-masking to retrieve the needed bit from GPIO_IDR, we would store GPIO_IDR into a register, and then we would right-shift the number by the amount needed, depending on if we wanted the data for the green, yellow, or red button. After this, we used the AND procedure to bitwise AND the register with 0x1, to where we were always comparing the result from the GPIO_IDR to a one. The TA I showed this to said it made our code less readable and

harder to understand, but I think it was more against convention than necessarily less readable.

Another part that really tripped us up was the overuse of “bl xxxxxx” and “bx LR”. We used this so much that we then began to pop and push LR to the stack to keep track of the link Register, which worked in some cases but in other cases this resulted in some awful memory errors. To clear this issue up, we simplified the logic we had previously been using, and this made it much easier to manage. Changing the MSI clock to run at 100kHz was actually quite easy, and then we implemented another “do nothing counter loop” to delay our system enough to where it looked like it was operating at 2Hz from the outside looking in. At first, we did the math using how many executions the loop had to find out how many iterations of the loop we needed, but we must have did my math wrong so thus began trial and error. We got our oscilloscope to read 2Hz exactly though, so we are very happy with the turnout.

Results

We were able to get our program to work correctly. We were able to finish it the night before it was due. We were able to get our LED display to increment at exactly a frequency of 2 Hz. We struggled with how the microcontroller moved between our different phases started by the buttons. However, after many hours, we were able to achieve the requirements of the lab. Our green button starts the binary counter. Our yellow button successfully pauses the counter. And our red button resets our counter down to zero.

Figures

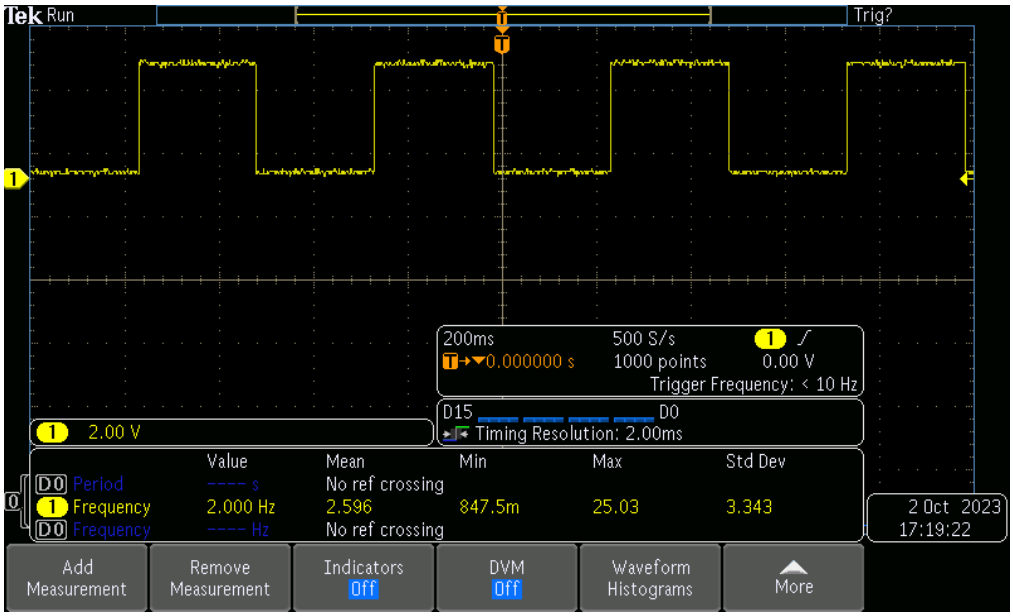

Figure 1: A screenshot of the frequency of our binary counter

Conclusion

This lab was definitely a bit harder than our previous labs. It took quite a bit of time for us to “iron out” our issues and problems. We struggled with the logic of the code and had to go back several times to re-analyze what actually needed to happen line by line. In the end, we were able to get our system to work correctly. Exactly how the guidelines were stated.