Lab 2:

LED's, Square Wave, Button, Pins

Microcomputer Systems

ELC 343 - L1

Nikita Eisenhauer and Jacob Levine

Submission: 9/24/17

Introduction

In this lab students were to build upon the knowledge they had previously acquired on the functioning and uses of the PSOC Creator. Students were to gain experience in wiring hardware pins of the PSOC board, outputting digital signals to those pins and coding the main logic processor of the board in C. Specifically, students were asked to output the highest frequency square wave possible with the board and measure it using an oscilloscope. In addition an LED was programmed to blink at a specified rate and a separate LED programmed to alter its state based on push-button input. The concepts of infinite loops, blocking delays and resistive pull-ups were also introduced during this lab.

Procedure

This lab's procedure consisted of writing code to generate a square wave with the highest frequency possible, and outputting that signal to an LED.

```
// code for cycling led at maximum possible speed
LED_Write( !LED_Read() );
```

Then the code was modified by enabling a delay period in order to make the LED flash at a rate of 60 hertz.

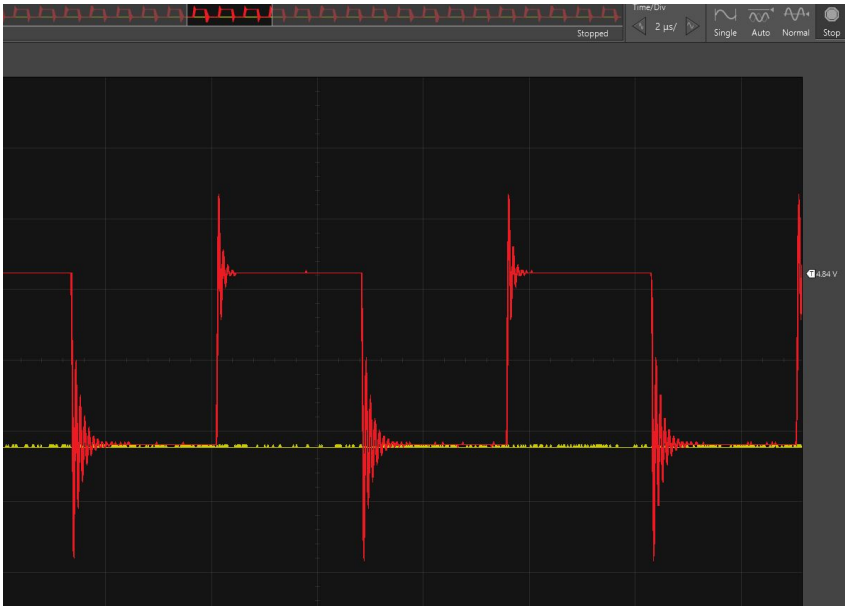
```
LED_Write( !LED_Read() );
// delay to adjust led cycle frequency
CyDelay(500);
```

This was followed by enabling a button toggle to activate another LED.

```
if( !SwitchInput_Read() )
{
LED2_Write(1);
}
else
{
LED2_Write(0);
}
```

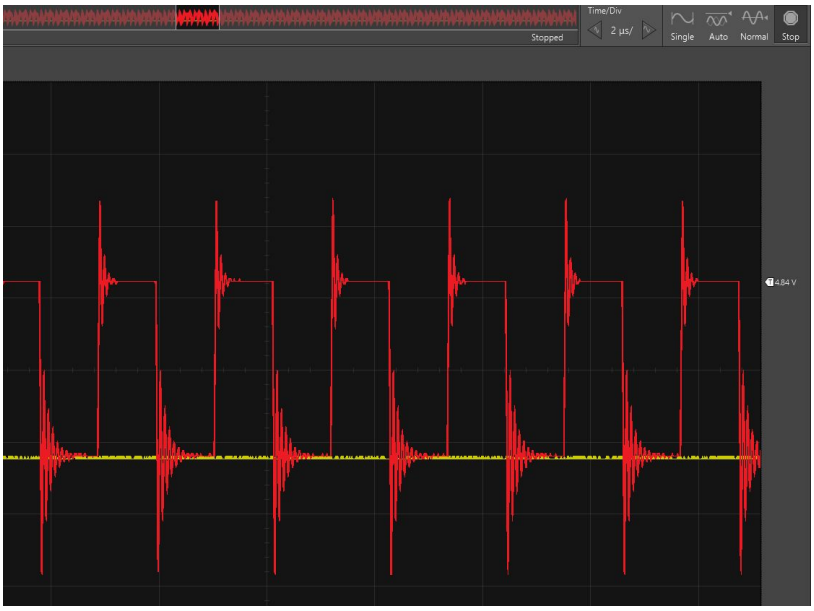
Results

When attempting to generate a high-speed square wave, we were able to obtain a period of approximately 5.6 µs without enabling any optimization during the compiling stage. A scope of this output is shown below.

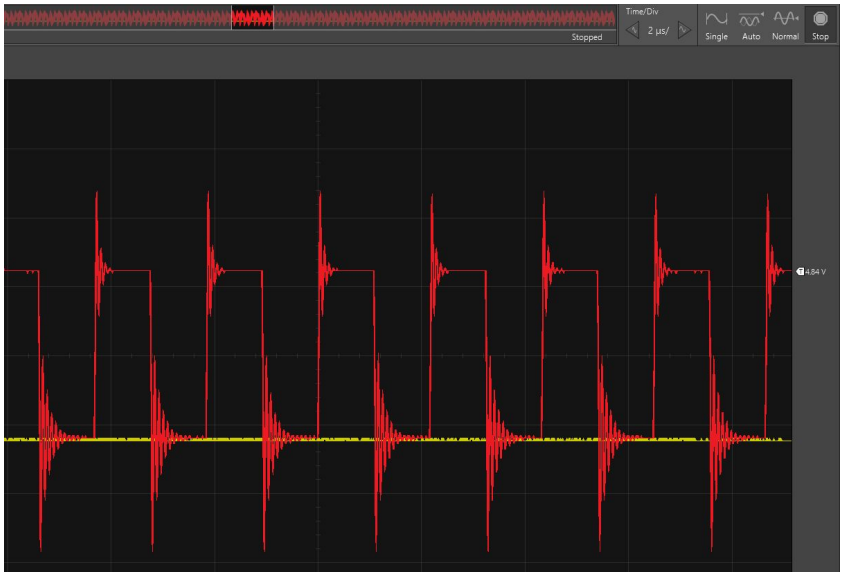


No Optimization

Next, we instructed the compiler to optimize for speed, followed by a general “maximum” optimization. Speed optimization resulted in a period of 4.2 µs, a significant drop of 1.4 µs from the unoptimized wave. Maximum optimization gave a similar result. These results are shown respectively, below.



Speed Optimized



Maximum Optimized

Finally, when introducing the manual delay to achieve a 60 hertz flash, we discovered that the button enabled LED was also affected by the delay. This is due to the fact that the code was written in a serial manner, doing both the button polling and cycling the LED sequentially. This emphasizes the importance of using parallelism in code designed to do two tasks when possible.

Discussion

Using the PSOC Creator's optimization options had a noticeable effect upon the maximum switching frequency the board could implement upon its LED's. A very slight microsecond difference in the period of the maximum vs speed optimized signals was present, however compared to the unoptimized option both decreased the switching period by 2 microseconds. Time critical PSOC programs could benefit greatly by the increased computation capacity such an optimization could afford.

The CyDelay() function allowed for somewhat accurate repetition rates for controlling square wave pulses and the timing of various pins, by forcing the board to ignore all user input and essentially do nothing for a specific number of clock cycles. However, due to the blocking nature of the CyDelay() function, all other functions running after it in code will also not be run until the set number of clock cycles has passed. In essence CyDelay() postpones all further execution of code by the amount specified by the user.

Due to the permanent ground connection wired to the output end of the push button used in the second half of the lab, a pull-up resistor configuration had to be utilized. When the push button was open (it's natural state), the input pin to the button would equal the supplied voltage and so the board would read a logical 1. When the push button was pressed, the input pin would be connected to ground and the voltage would be driven to 0, hence the board would read a logical 0. Therefore, in code the value of the input pin was inverted when used in the if statements in order to properly control the LED output to represent when the button was pressed or not pressed.

Conclusion

Overall, code optimization and thorough research were found to be the most important aspects of this lab. The deceiving nature of the CyDelay() function could easily catch any first time user of the PSOC suite off guard and without knowledge of the specific ground wiring scheme employed by the pushbuttons on the board, one would most likely encounter unexpected program execution when pressing them. Nevertheless students were demonstrated the possibilities and flexibility of the PSOC Creator program and its ease of which it allows for ideas and designs to be implemented into physical hardware.