Assignment 1: Lab Report

Digital Design Lab

Sarah Brown

I. Introduction

A. Overview

This assignment focuses on implementing a simple function to blink a built-in LED. By completing this assignment students work to familiarize themselves with the lab equipment and MCUXpresso IDE. In addition, this lab serves to show students how to compile code and use the IDE's debugger.

B. Objectives

The objective of this assignment was to modify provided code to make the built-in LED on the board blink at 5 Hz with a high time of 20 ms and a low time of 180 ms. These values should be within a margin of error of 0.5% of their targets.

C. Demonstratables

Show the built-in LED blinking at 5 Hz with the high time and low time shown on the oscilloscope within the acceptable error range.

II. Solution

The wait_ticks function can be modeled as the equation of a line in the format of y = mx + b. By collecting data from two different circumstances, this relationship can be found. The initially provided code (see appendix) inputs 30 into the wait_ticks function. By measuring the high time of the square wave on port 0 bit 22, we gather a datapoint of y for a corresponding x. For a x input of 30 ticks, the y output is 90.70 microseconds.

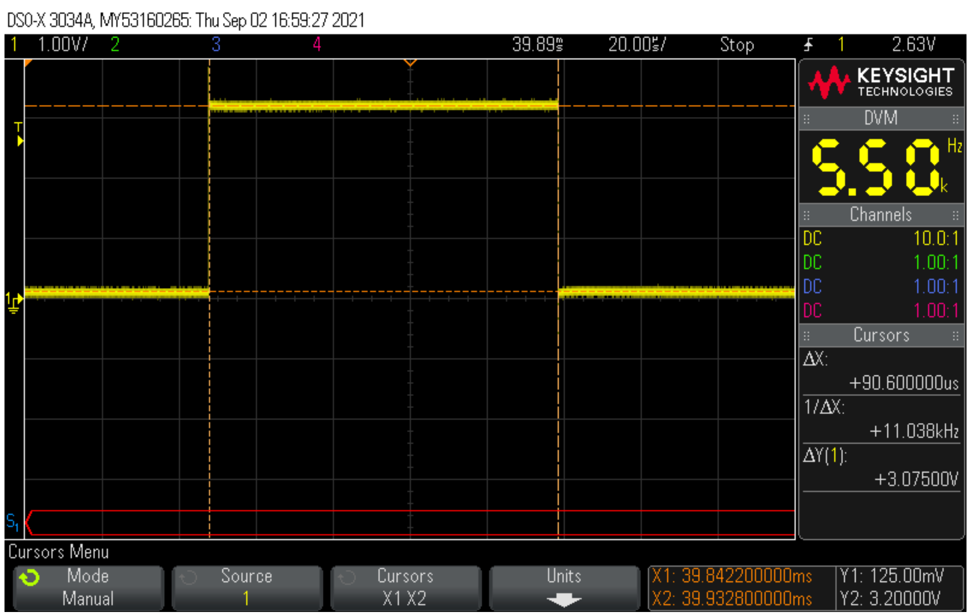

Figure 1: Screenshot of oscilloscope output of original code showing a high time of 90.60 microseconds. Measurement performed with manual cursors.

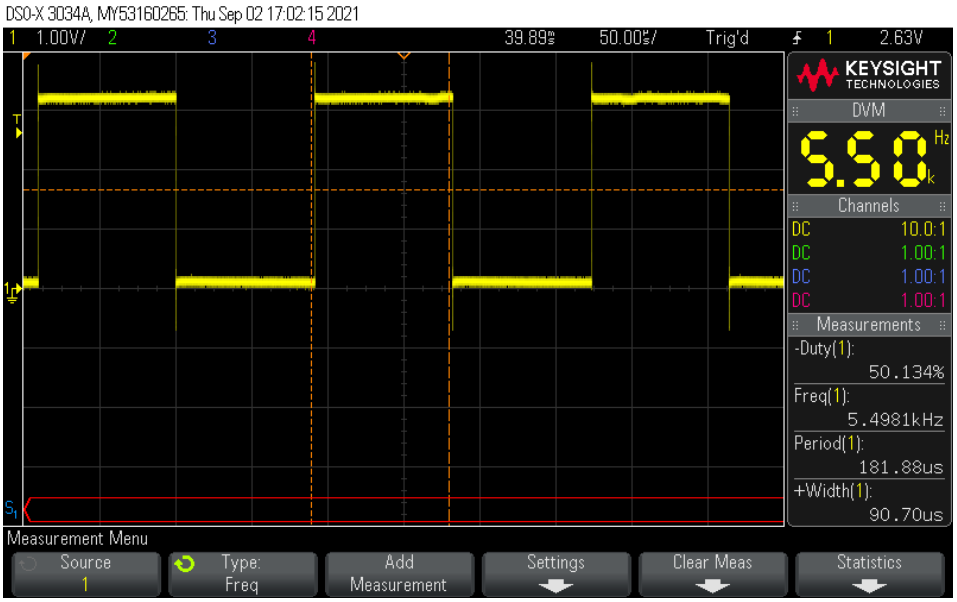

Figure 2: Screenshot of oscilloscope output of original code showing a high time of 90.70 microseconds. Measurement performed with automatic oscilloscope measurements.

Next, the code is changed (see appendix) so that the x value is 10000 ticks. Port 0 bit 22 can then again be measured to obtain the corresponding y value for this x value. For a x input of 10000 ticks, the y output is 27.349 milliseconds.

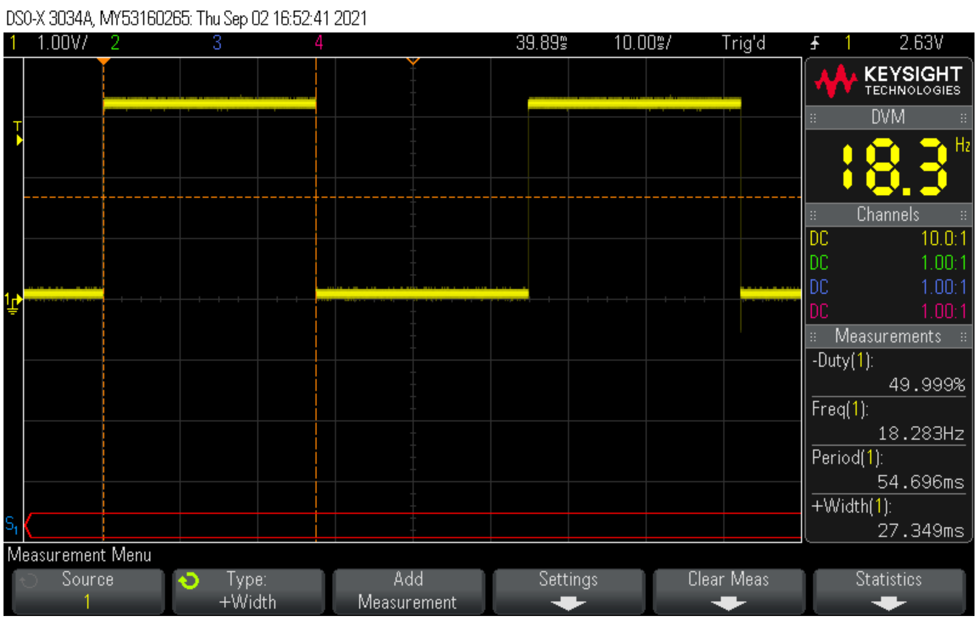

Figure 3: Screenshot of oscilloscope output of changed code showing a high time of 27.349 milliseconds.

Trials	Input Parameter (ticks)	LED High Time (seconds)
1	30	90.70 * 10^-6
2	10000	27.349 * 10^-3

Table 1: Data collected from the two trials.

With these two different values for x and y, m and b can be computed for the equation y = mx+b. These calculations were done by inputting values into excel and using a linear trendline and by solving y=mx+b manually.

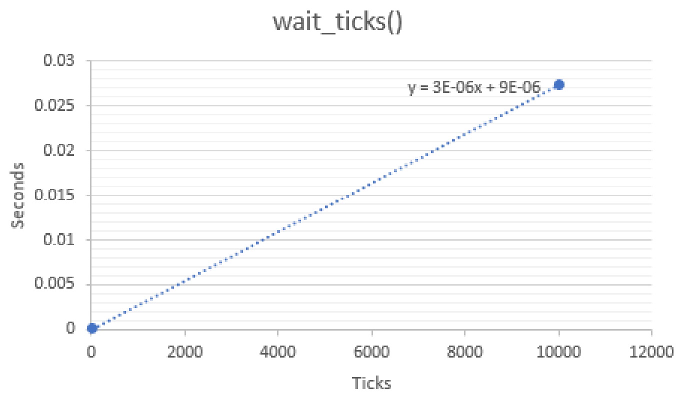

Figure 4: A graph created in Excel showing the equation for the wait_ticks() function.

Y = mx + b
27.349*10^-3 = m*10000 + b -> 27.349*10^-3 - m*10000 = b
90.70 *10^-6 = m*30 + b -> 90.70 *10^-6 = m*30 + 54.696*10^-3 - m*10000
-> 90.53 *10^-6 - 27.349*10^-3 = - 9970m
-> -0.02725847 = -9970m -> m = 2.73*10^-6
b = 27.349*10^-3 - m*10000 = 54.696*10^-3 - 2.73*10^-6*10000 = 8.5*10^-6

y = 2.73*10^-6* x + 8.5*10^-6

This equation can then be solved for x to determine the number of ticks required to obtain a certain time.

Solving x for y = 20*10^-3 gives x ≈ 7322.9
Solving x for y = 180*10^-3 gives x ≈ 65930.95

As the wait_ticks function requires an integer input, the x variable for the high time will be 7323 and the x variable for the low time will be 65931. The code is modified with these variables to change the high and low times to the required values (see appendix).

	Expected Time	Input Ticks	Allowed Error (0.5%)
High Time	20 ms	7323 ticks	+/- 0.1 ms
Low Time	180 ms	65931 ticks	+/- 0.9 ms

Table 2: Derived values and allowed error for the required high and low times.

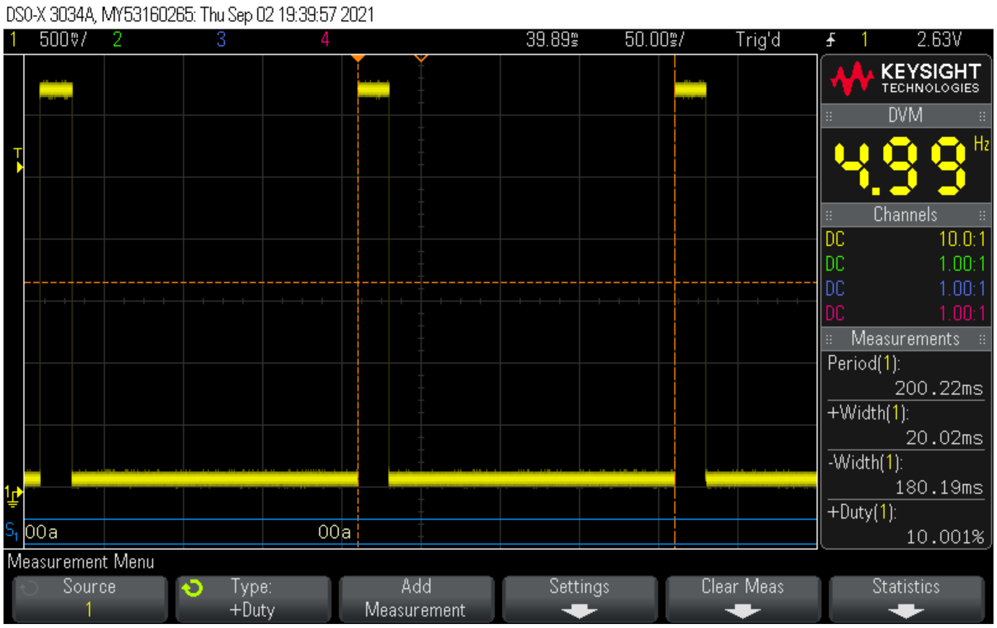

Figure 5: Screenshot of oscilloscope output of final code showing a high time of 20.02 ms and a low time of 180.19 ms

	Expected Time	Allowed Error (0.5%)	Measured Error
High Time	20.02 ms	+/- 0.1 ms	0.1%
Low Time	180.19 ms	+/- 0.9 ms	0.1%

Table 3: Measured values and error for the high and low times.

The mathematical error can be calculated with the following:

$$Error = \left|\frac{Actual - Expected}{Expected}\right| \cdot 100\%$$

$$High\ Time\ Error = \left|\frac{20.02-20}{20}\right| \cdot 100\% = 0.1\%$$

$$Low\ Time\ Error = \left|\frac{180.19-180}{180}\right| \cdot 100\% = 0.1\%$$

Both the high and low times are within the allowed error range and thus satisfy the requirements of the assignment.

III. Conclusion

This lab assignment focused on familiarizing the student with the lab equipment and the MCUXpresso IDE. By changing sample code and taking different data samples it was possible to determine the linear relationship in the wait_ticks function. This function can now be used in future lab assignments as it has a known relationship.

Appendix

- Original code

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>

#define FIO0DIR (*(volatile unsigned int *)0x2009c000)
#define FIO0PIN (*(volatile unsigned int *)0x2009c014)

void wait_ticks(int ticks) {
        volatile int count;
        for (count=0; count<ticks; count++) {
                //do nothing
        }
}

int main(void) {
        FIO0DIR |= (1<<22); // configure port 0 bit 22 as output

    while(1) {
        FIO0PIN |= (1 << 22); // make port 0 bit 22 go high
        wait_ticks(30);
        FIO0PIN &= ~(1 << 22); // make port 0 bit 22 go low
        wait_ticks(30);
    }
    return 0 ;
}
```

- Code with 10000 instead of 30

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>

#define FIO0DIR (*(volatile unsigned int *)0x2009c000)
#define FIO0PIN (*(volatile unsigned int *)0x2009c014)

void wait_ticks(int ticks) {
        volatile int count;
        for (count=0; count<ticks; count++) {
                //do nothing
        }
}

int main(void) {
        FIO0DIR |= (1<<22); // configure port 0 bit 22 as output

    while(1) {
        FIO0PIN |= (1 << 22); // make port 0 bit 22 go high
        wait_ticks(10000);
        FIO0PIN &= ~(1 << 22); // make port 0 bit 22 go low
        wait_ticks(10000);
    }
    return 0 ;
}
```

- Final code to blink at 5 Hz

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>

#define FIO0DIR (*(volatile unsigned int *)0x2009c000)
```

```c
#define FIO0PIN (*(volatile unsigned int *)0x2009c014)

void wait_ticks(int ticks) {
        volatile int count;
        for (count=0; count<ticks; count++) {
                //do nothing
        }
}

int main(void) {
        FIO0DIR |= (1<<22); // configure port 0 bit 22 as output

    while(1) {
        FIO0PIN |= (1 << 22); // make port 0 bit 22 go high
        wait_ticks(7323);
        FIO0PIN &= ~(1 << 22); // make port 0 bit 22 go low
        wait_ticks(65931);
    }
    return 0 ;
}
```