Lab Report 1 - Taylor Rainwater, Triston Luzanta

**Overview**

The objective of this lab assignment was as a refresher on the concepts of bit registers, program functions, and to become familiarized with lab equipment and software. The requirement was to create a program which would blink at a rate of 8 Hz with a 25% low duty cycle.

**Measurement**

For this lab we were given a basic program which contains a wait function that takes an integer argument of ticks and waits for that many clock cycles. The program then uses that wait function to control the high and low pulse widths of port 0 pin 22 on the LPC1769 microcontroller making the LED blink as specified. Below is the base code:

#ifdef \_\_USE\_CMSIS

#include "LPC17xx.h"

#endif

#include <cr\_section\_macros.h>

#include <stdio.h>

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

#define FIO0PIN (\*(volatile unsigned int\*) 0x2009c014)

void wait\_tick(int ticks)

{

volatile int count;

for (count=0; count<ticks; count++){

}

}

int main(void) {

FIO0DIR |= (1<<22);

while(1) {

FIO0PIN |= (1<<22);

wait\_tick(30);

FIO0PIN &= ~(1<<22);

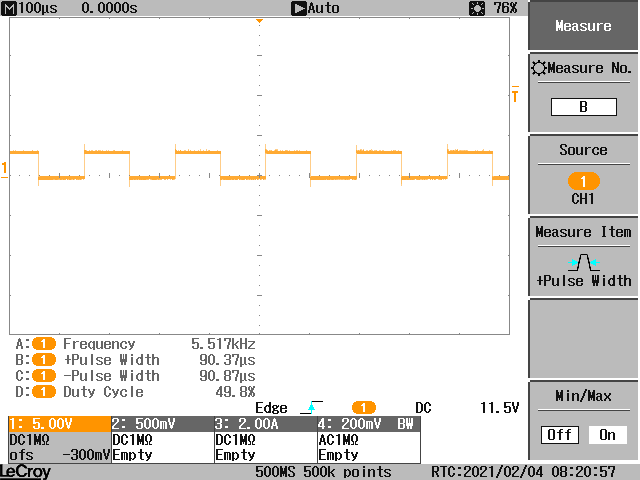
wait\_tick(30);

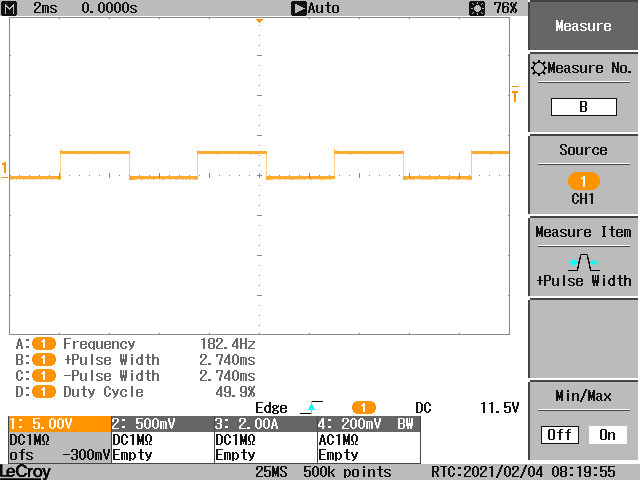
}

return 0 ;

}

Firstly we ran the code with an argument of 30 ticks and measured the pulse width using an oscilloscope as shown below.



Using an argument of 30 ticks resulted in a pulse width of 90.37𝛍s. Next we changed the argument of the wait functions to 1000 and again measured the pulse width.

This time the pulse width was 2.74ms.

**Calculation**

With these measurements we were able to calculate the number of ticks necessary to create the desired signal. At 8 Hz and 25% low duty cycle, this is 93.75 ms high and 31.25 ms low.

Since the wait function follows a linear relationship of ticks to wait time in the form of

We were able to compute the number of ticks needed to create the desired high and low time.

Plugging in the numbers that we measured previously into the equation we obtain:

Eq 1.

Eq 2.

From here we found *m* using algebra:

Then plug *m* into Eq1. to obtain *b*:

With these values we can calculate the number of ticks needed in the wait argument to create the desired high and low time.

Since *x* represents the number of ticks, we restructured the format so that:

We then unplugged in our numbers for *m* and *b*, with *y* being the desired time.

**Code**

Once we obtained the values for the wait argument, we augmented the program to output the desired signal, as shown below.

/\*

===============================================================================

Name : test\_1.c

Author : Triston Luzanta, Taylor Rainwater

Version :

Copyright : $(copyright)

Description : main definition

===============================================================================

\*/

#ifdef \_\_USE\_CMSIS

#include "LPC17xx.h"

#endif

#include <cr\_section\_macros.h>

#include <stdio.h>

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

#define FIO0PIN (\*(volatile unsigned int\*) 0x2009c014)

void wait\_tick(int ticks)

{

volatile int count;

for (count=0; count<ticks; count++){

}

}

int main(void) {

FIO0DIR |= (1<<22);

while(1) {

FIO0PIN |= (1<<22);

wait\_tick(34338);

FIO0PIN &= ~(1<<22);

wait\_tick(11444);

}

return 0 ;

}

**Results**

Below is the resulting output from our code, the frequency is a perfect 8 Hz and 25% low duty cycle.



