Lab Report 3 - Taylor Rainwater, Triston Luzanta

**Objectives**

The objective of this lab assignment is to configure the LPC1769 to generate a 9 Mhz square wave with a 50% duty cycle using both software and hardware methods. Each method will require different techniques to generate the square wave. For the hardware method, we will be utilizing a 8 Mhz crystal along with a clock multiplier/divider to raise the frequency to 9 Mhz. As for software, the microcontroller has built in oscillators in itself meaning that we just have to program the signal.

**Software Techniques:**

* Using the 4 Mhz internal RC oscillator
* Using PLL0 method
* 2M/N \* SYSCLK\* 1/K = 9Mhz
  + Where SYSCLK = 4 Mhz
  + Let N = 1
  + Let M = 54
    - Solve for K = 48
  + 2M \* SYSCLCK = 432 Mhz within range 275 Mhz<= 432 Mhz<= 532 Mhz

Although we did not get the chekoff for this technique. We would like to at least post the source code to give our idea on how to implement the signal.

/\*

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

Name : Assignment\_3.c

Author : Taylor Rainwater and Triston Luzanta

Version :

Copyright : $(copyright)

Description : This code is to configure the microcontroller to generate a 9 Mhz signal with at least a 50% duty cycle

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

\*/

#ifdef \_\_USE\_CMSIS

#include "LPC17xx.h"

#endif

#include <cr\_section\_macros.h>

// Defining Registers

#define PLL0FEED (\*(volatile unsigned int\* )0x400FC08C) // Feed Register PLL0

#define PLL0CON (\*(volatile unsigned int\* )0x400FC080) // Control Register PLL0

#define CLKSRCSEL (\*(volatile unsigned int\* ) 0x400FC10C) // Clock Source Select Register

#define PLL0CFG (\*(volatile unsigned int \*) 0x400FC084 ) // Configuration Register PLL0

#define PLL0STAT (\*(volatile unsigned int \*) 0x400FC088 ) // Status Register PLL0

#define CCLKCFG (\*(volatile unsigned int \*) 0x400FC104 ) // CPU Clock Configuration Register, Clock Divider

#define CLKOUTCFG (\*(volatile unsigned int \*) 0x400FC1C8 ) // Clock Output Configuration Register, Power control

#define PINSEL3 (\*(volatile unsigned int \*) 0x4002C00C) //

// Feed sequence

void feedseq()

{

PLL0FEED = 0xAA;

PLL0FEED = 0x55;

}

// Driver Program

int main(void)

{

// So by default Internal RC oscillator as the PLL0 clock source

// Select P.127

PINSEL3 |= (1<<22);

PINSEL3 &= ~(1<<23);

// We will select the CPU clock as the CLKOUT source

CLKOUTCFG = 0;

// Enable the CLKOUT control

CLKOUTCFG |= (1<<8);

// The CCLCKCFG register controlst he division of the PLL0 output and so we will divide by our K value which will be 48

CCLKCFG = 47; // CCLKSEL + 1 where 47 is CCLKSEL

// Do Steps for PPL0 configuration steps

// Disconnect PLL0 with one feed sequence if PLL0 is already connected

PLL0CON &= ~(1<<1);

feedseq();

// Disable PLL0 and then feed

PLL0CON &= ~(1<<0);

feedseq();

// Wait for PLL0 to be disabled

while( (PLL0STAT & (1<<24) ) ){

}

// Write to the PLL0CFG and make it effective with one feed sequence. The PLL0CFG can only // be updated when PLL0 is disabled.

// Feed in our Multiplication of M, M - 1

PLL0CFG = 53;

feedseq();

// Enable PLL0 with one feed sequence

PLL0CON |= (1 << 0);

feedseq();

// Wait for PLL0 to achieve lock by monitoring the PLOCK0 bit in the PLL0STAT register, or // using the PLOCK0 interrupt, or wait for a fixed time when the input clock to PLL0 is slow

// Wait for PLL0 to be locked

while ( !(PLL0STAT & (1<<26)) )

{}

// Connect PLL0 with one feed sequence

PLL0CON |= (1 << 1);

feedseq(); }

**Hardware Technique**

**1.** Crystal oscillator:

To begin, hardware for the crystal oscillator circuit is designed. The crystal assumes 32pF load capacitance CL. This means that C1 and C2 must be chosen to fit Eq 1.

Eq1. C\_L = ((C\_i + C\_1)C\_2) / (C\_1+C\_i+C\_2 ) + C\_stray

Where Ci is the input capacitance of the inverter Ci = 3pF and Cstray was assumed to be 5pF.

To solve this we first assumed C1 = C2 = C and solved for C. this yielded Eq2.

Eq2. 32 = (C^2 + 3C) / (2C + 3) + 5

From this we obtain C = 52.4pF. Plugging this value into Eq1. for C1 we obtain C2 = 52.66pF. Finally rounding to standard capacitor values we get C1 = C2 = 51pF.

To decide the value of Rs in the crystal circuit we create an LPF with cutoff frequency equal to 8MHz.

Eq3.f = 1 / (2πRC)

Where f = 8MHz, R = Rs, and C = 51pF. Plugging in these values we obtain Rs = 390 ohms. Finally we selected Rbias = 1Mohm.

**2.** Divider and Multiplier Values:

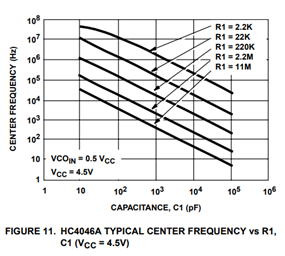
Next the base frequency of the oscillator must be multiplied and divided to create a CLKout = 9MHz signal according to Eq4.

Eq4. CLKout = CLKin M / N

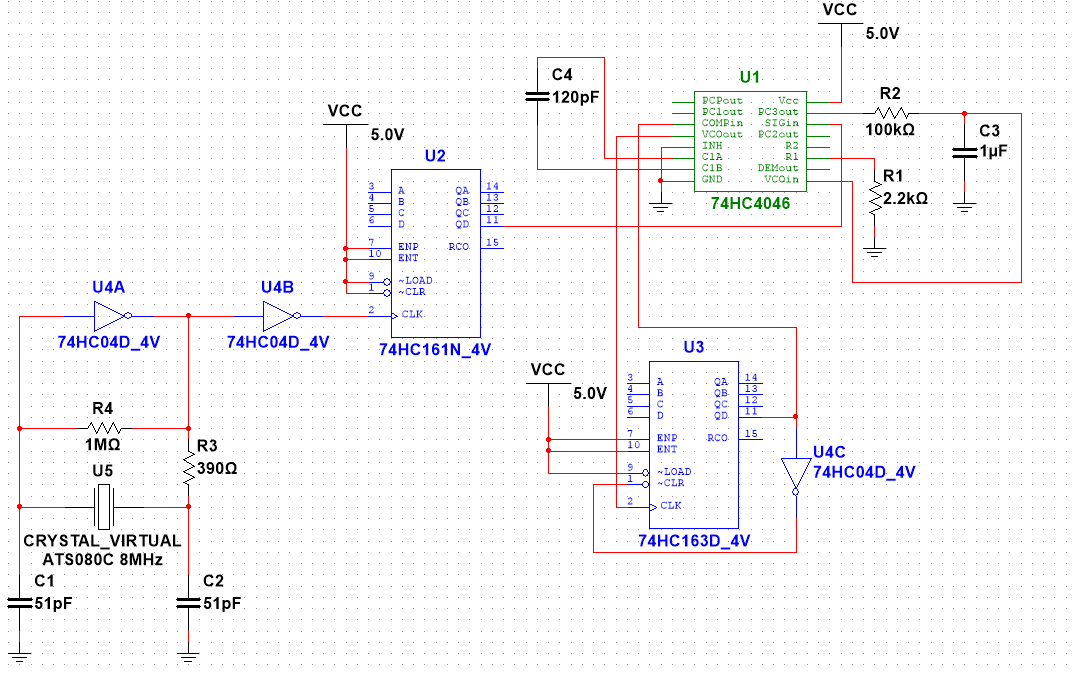
Since the oscillator frequency CLKin = 8MHz, we decided the best way to create 9MHz would be to make M = 8 and N = 9. To do this we used a basic 4 bit counter clock divider to divide the oscillator circuit into a 1MHz signal, and a PLL circuit to multiply the 1MHz signal to a 9MHz signal.

**3.** Phase Locked Loop Circuit

To begin we chose the capacitor and resistor values for the VCO block in the PLL. We used a 5V power supply for our circuit as it gave a greater range to the VCO. Using the graph below from the 74HC4046 datasheet, we determined that a 2.2K resistor and a 100pF capacitor would be best suited, though we used a 120pF capacitor as that was the closest value we had access to. The values were determined by selecting 10^7 from the Y axis as that is closest to the value of 9MHz and seeing where on the X axis corresponds to the 2.2Kohm line.



For the LPF section we once again used Eq3. to create a cutoff frequency of about 1Hz. We selected a 1uF capacitor and obtained the resistor value to be about 100Kohms. Finally we used a 4 bit counter to create a divide by 9 in the feedback portion of the PLL.

Below is the final circuit.

Below are the output waveforms for the hardware.

