ECE 414 Project 5
Square Wave Signal Generator

Vijay Natarajan and Charles Wolfe

August 19, 2013

For the final project of the semester, a square wave generator was designed. A simple square wave generator can be made with a operational amplifiers, two SPI controlled digital potentiometers, and a PIC microcontroller. The square wave signal is generated by a square wave relaxation oscillator circuit. The frequency of the oscillator is based on the feedback resistance in the circuit, so the frequency can be controlled by using the one digital potentiometer as the feed back resistance in the circuit. Then, in order to control voltage of the signal, the output of the oscillator is fed into a voltage divider that uses the other digital potentiometer.

1 Components:

Relaxation Oscillator: A relaxation oscillator is an oscillating circuit in which the circuit attempts to reach an equilibrium point, but whenever the circuit reaches its equilibrium point, the system is disturbed and the circuit has to reattempt to reach equilibrium. The circuits need to relax to a stable point is what causes it to oscillate, which is why it is called a relaxation oscillator. The single supply relaxation oscillator circuit is shown in figure 1.

As shown in the figure, the non-inverting input of the operation amplifier is biased at the half the supply voltage, or at the midpoint between the rails.There is also a feed back resistance back to the non-inverting input. This resistance will move the equilibrium point of the system to a little bit above or below the midpoint voltage. The inverting input of the opamp is connected to a capacitor connected to ground and a feedback resistor, creating an RC circuit between the output and the inverting input.

When the circuit is turned on, the inverting input will be at 0 volts since the capacitor hasn't been charged, and the non-inverting has some positive voltage above 2.5 volts. Since

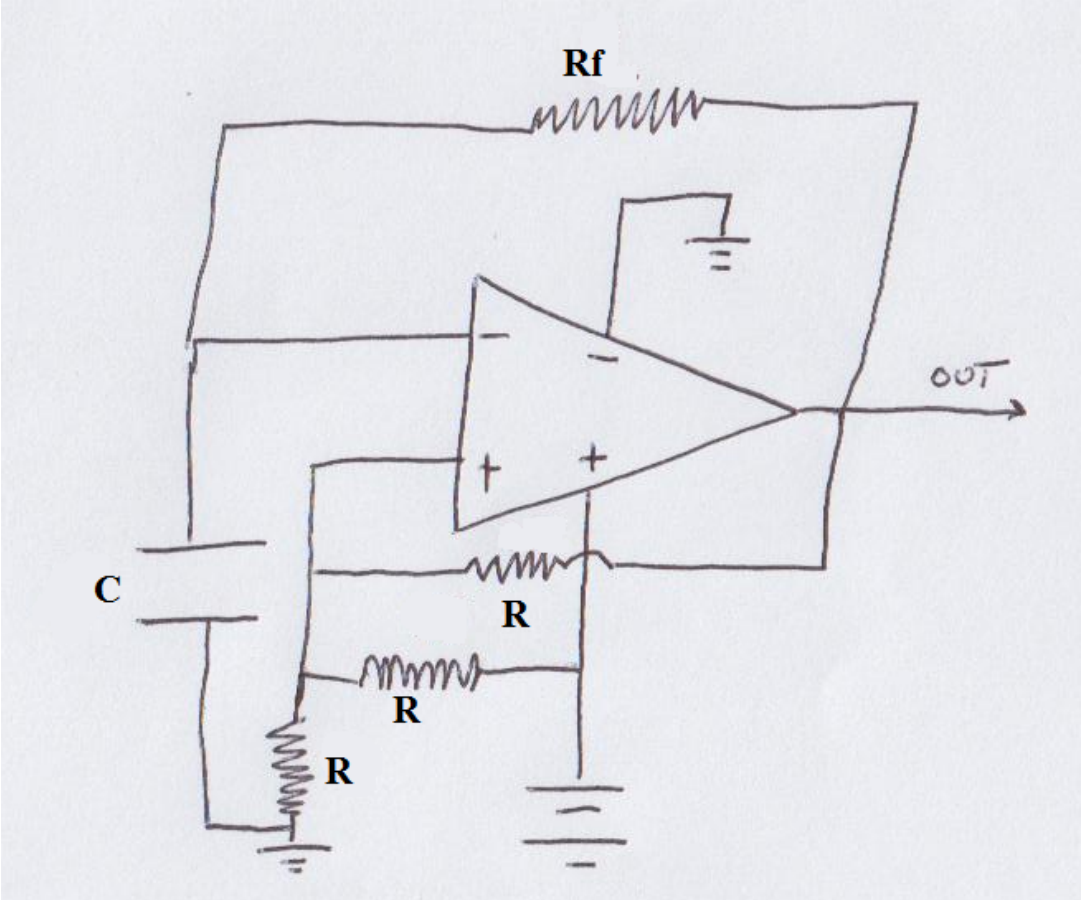

Figure 1: Single Supply Relaxation Oscillator

the non-inverting input has a higher potential than the inverting input, the differential gain of the op-amp will rail the output to the positive rail, which will raise the non-inverting voltage above the midpoint voltage and will charge the capacitor connected to the inverting input. Now the capacitor wants to charge to equal to output voltage, and while its charging it will pass the non-inverting voltage level, which is the equilibrium point of the system. Once this happens, since the inverting input now has a higher potential than the non-inverting input, the differential gain of the op-amp will rail the output to the negative rail, which lowers the non-inverting voltage below the midpoint voltage, and the inverting input capacitor will now start discharging until it discharges passed the equilibrium point and oscillates back to the positive rail.

Within this circuit, three different oscillating signals are created. The output signal is a square wave oscillating between the positive rail and the negative rail. The non-inverting

signal is a square wave oscillating between a potential above the midpoint voltage and a potential below the midpoint voltage. The inverting signal is a triangle wave oscillating with peak voltages approximately equal to the potentials at the non-inverting input. Only the output signal is used for this project. The Frequency of all the signals can be characterized by the equation $f = \frac{1}{2 \ln 3 R_f C}$. Therefore, by adjusting R_f, the frequency of the oscillator can be increased or decreased. This resistance can be controlled by using a digital potentiometer.

Digital Potentiometer: The digital potentiometer used to control the frequency and voltage lever of the oscillator is the MCP4241. The MCP4241 is a dual SPI digital pot, having two separate digital potentiometers within the IC. The IC acts as a slave on an SPI bus; the value of the potentiometers can be set by sending a control message over the SPI bus from some master device. The digital potentiometers used in the circuit has a max resistance of $10k\Omega$. When trying to set a value to either potentiometer, first a control message is sent to select which potentiometer is to be written two, and then a 7 bit number is written to the SPI line, which determines the resistance of the Potentiometer. The resistance of potentiometer is characterized by the equation $R = \frac{10k\Omega N}{128} + 75\Omega$, where N is the 7 bit number written to the SPI bus.

PIC16F690(micro-controller) The micro-controller used to control the square wave signal generator is a PIC16F690. The PIC16F is a 16 bit processor that has 256 bytes of programmable rom, in which a program can be writen to. In order to program the PIC micro-controller, A tool called a PICkit is used. The PICkit tool is a usb device that can write a program to the programmable rom within the PIC. The first step in programming the PIC is wiring up the PICkit. Pin 1 on the PICkit goes to RA3, which supplies the programming voltage needed to flash the programmable rom, pin 2 goes to vdd or 5 volts, pin 3 is wired to ground, pin 4 goes to RA0, which supplies the program data that needs to be flashed, and pin 5 goes to RA1, which acts as the clock for the flashing process. Once the PIC is wired to the PICkit, the program for the micro-controller will be written in MicroC. In this project, the PIC needs to utilize two different communication protocols. The first communication protocol used is UART, an asynchronous, peer to peer serial data protocol which is used to collect the users controls from the PC, and the second protocol used is SPI, a synchronous, master/slave serial data protocol which is used to set the resistance values of the digital potentiometers.

UART serial communication is an asynchronous data communication protocol. In this case, the protocol utilized for serial transmission is the RS-232 serial protocol. In order for

asynchronous data transfer to work, all the machines connected to the data bus must be configured to read the data at the same baud rate. The baud rate used was 9600 bits/sec. When a message needs to be sent over the data bus, the writing device will first write a start bit to the bus. This will inform the other devices that a message is about to be sent over the data bus. When the writing device is done sending its message, it will write a few stop bits to the bus, informing the other devices that the message being sent is complete. For this project, the UART bus will be used to read control information sent by the user, such as increasing or decreasing the frequency or voltage level of the square wave being generator, and will also write the frequency being produced back to the screen for the user to read.

SPI serial communication is a synchronous data communication protocol. Unlike UART, SPI is a master slave communication protocol, meaning that there is some master device that needs to query the slave device before data can be written to the bus by either the slave or master device. SPI communication uses 4 wires, a data in and a data out lines, a chip selection line, and a clock that synchronizes the data transfer. When a master wants to query a slave device, it first needs to select the slave device it wants to use using the chip select line, and then must send a control message to the slave device. In this project, since there is only one slave device, its chip select line can always be active. When the PIC tries to write to the digital potentiometer, it first sends a control message saying it wants to write to the device, then it sends the data that needs to be written to the Potentiometer in order to change its resistance.

2 Design and Specifications:

The design of the square wave signal generator is fairly simple. The feedback resistance to the inverting input of the relaxation oscillator will be on digital pot and the output of the relaxation oscillator will be connected to a voltage divide with the first resistor being the other digital pot. The schematic for signal generator is shown in figure. Based on the attributes of the digital potentiometers, specifications for the signal generation can be place. For frequency, the bigger the feedback resistance is, the lower the frequency of the oscillator is going to be. Therefore, since the max resistance of the digital pot is $10k\Omega$, the circuit can be designed for a minimum frequency, and the range of frequencies can be found. For this project, a minimum frequency of 1 kHz was chosen because this was the lowest frequency the circuit could oscillate at while still producing a clean square wave. At any lower of a frequency, there was a lot more distortion. In order to create a 1 kHz signal, the capacitor value was calculated to $C = \frac{1}{2 \ln 3 * 10k\Omega * 1000Hz} = 0.046\mu F$. A capacitor of $0.047\mu F$ was used

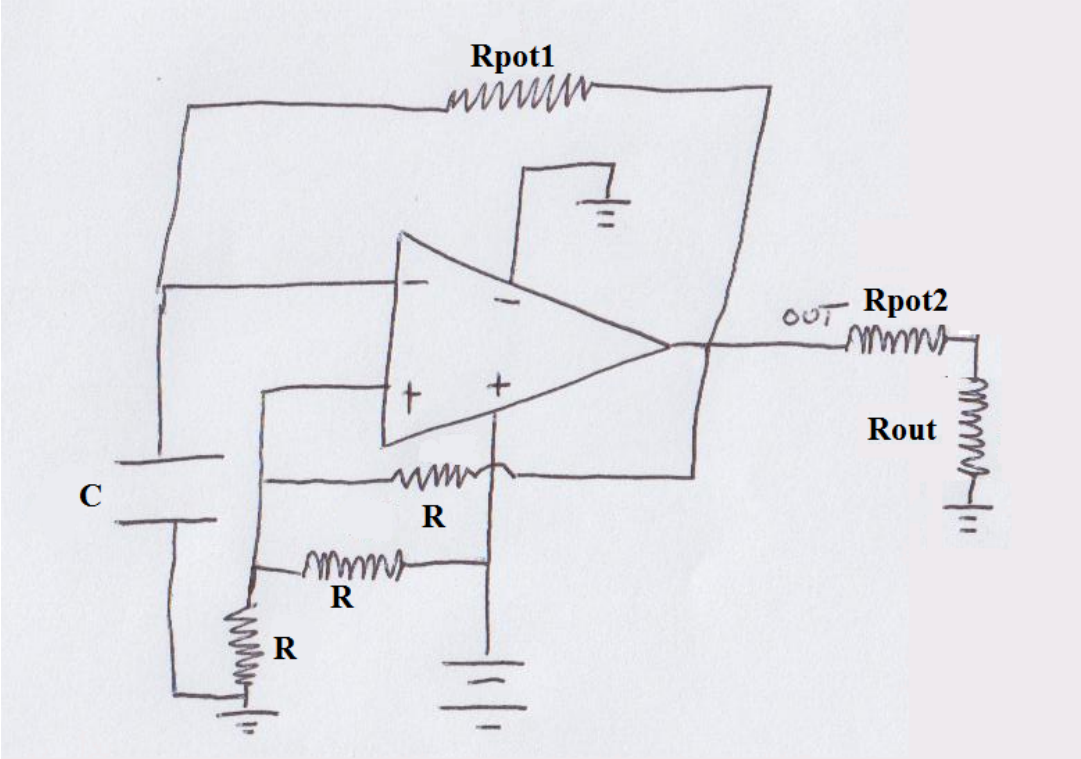

Figure 2: Square Wave Signal Generator

in the physical circuit, which makes the minimum frequency around $f = \frac{1}{2 \ln 3 * 10k\Omega * 0.047\mu F} = 968Hz$. Since the minimum resistance of the digital pot is 75Ω, the maximum frequency should be about $f = \frac{1}{2 \ln 3 * 75\Omega * 0.047\mu F} = 129kHz$.

For voltage level, theoretically, the signal should be oscillating from rail to rail, or 0 to 5 volts in this case. However, in the physical circuit, the output has to be at least a diode drop away from the rail voltages, so it can be assumed that the signal will oscillate from 0.7 - 4.3 volts, or have a peak to peak voltage of 3.6 volts. By connecting the output of the oscillator to a voltage divider circuit like what is shown in the figure ,the peak to peak voltage can be attenuated by the equation $V_{attenuated} = V_{out} \frac{R_{out}}{R_{out} + R_{pot2}}$. Therefore, the smaller the R_{out} is, the more the signal can be attenuated. For this project, an R_{out} value of $2.2k\Omega$ was chosen. This means the max output voltage will be $3.6V * \frac{2200\Omega}{2275\Omega} = 3.5V$ and the minimum output voltage will be $3.6 * \frac{2200\Omega}{12200\Omega} = 650mV$. Therefore the expected specifications for the square wave generator are that the voltage will be from .65 to 3.5 volts and the frequency will range from about 1 to 129 kHz.

3 Physical Circuit and Problems

After building the circuit, a lot of different problems were seen. Though the circuit could act as a square wave signal generator, in which a user could set the frequency and amplitude of the square wave being generated, the circuit wasn't able to meet the specification calculated above. The first problem was with the voltage level. The Op-Amp that was used(LM741) needs to operate a rail voltage of at least 6 volts. However, the usb Uart connection that is powering the circuit only provides 5 volts. Though the opamp was still operational, the peak voltages were farther than a diode drop from the rails. The max peak to peak voltage for the physical circuit was 2.5 volts and the min was 500 mV. As for the frequency, the circuits minimum frequency was at 1kHz. However when the frequency was raised over 10 kHz, the slew rae of the opamp was too slow to oscillate like a sqaure wave, the the signal began to attenuate. The rise and fall times of the waves were $8\mu s$, which is too big for any square signal over 10 kHz. Therefore the frequency range for the physical model(without any voltage attenuation) was from 1 kHz to 10 kHZ. Shown in figures 3 and 4 is the output of the signal generator, and figures 5 and 6 show the C code written for the PIC.

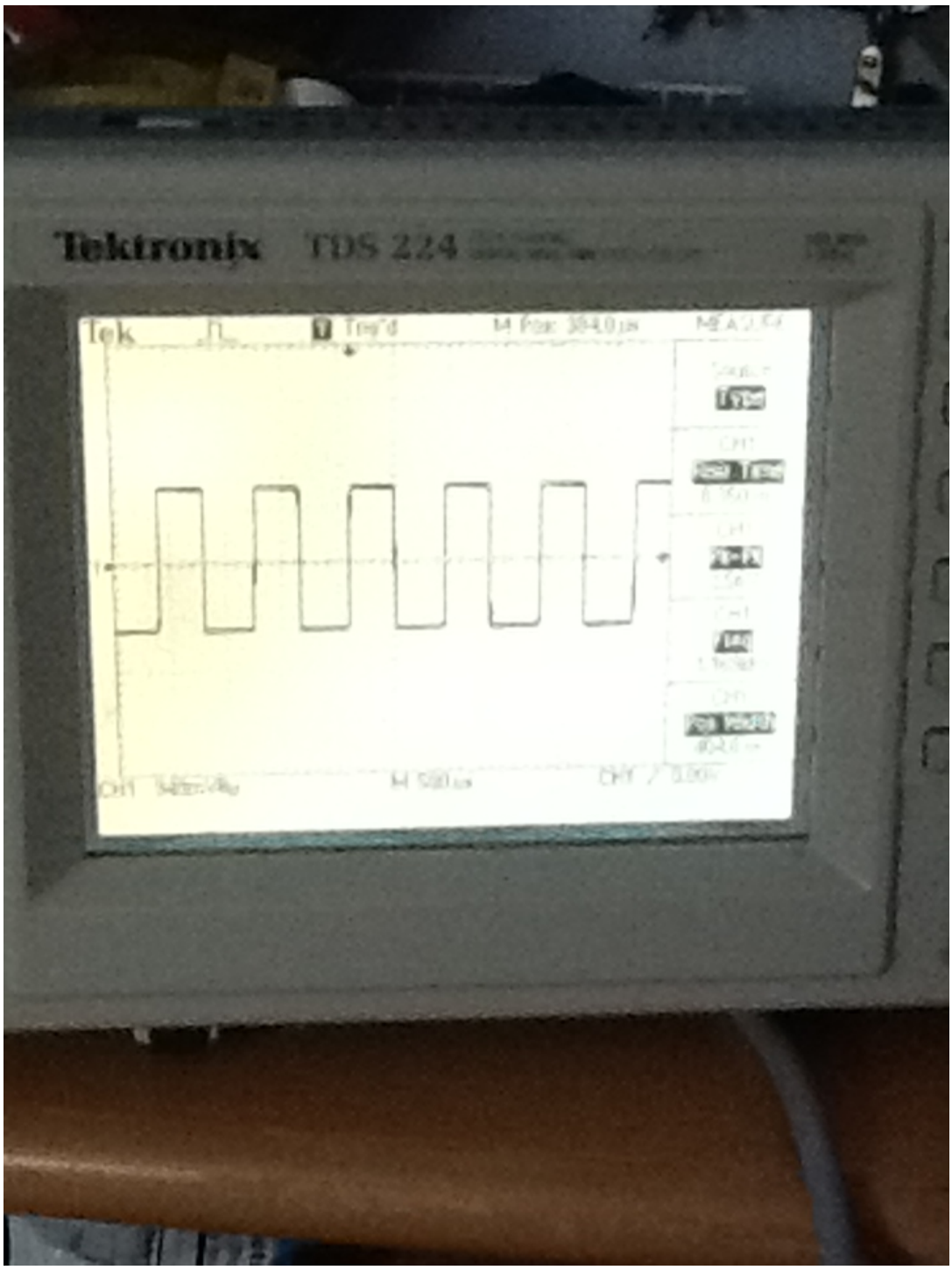

Figure 3: output of signal generator(1)

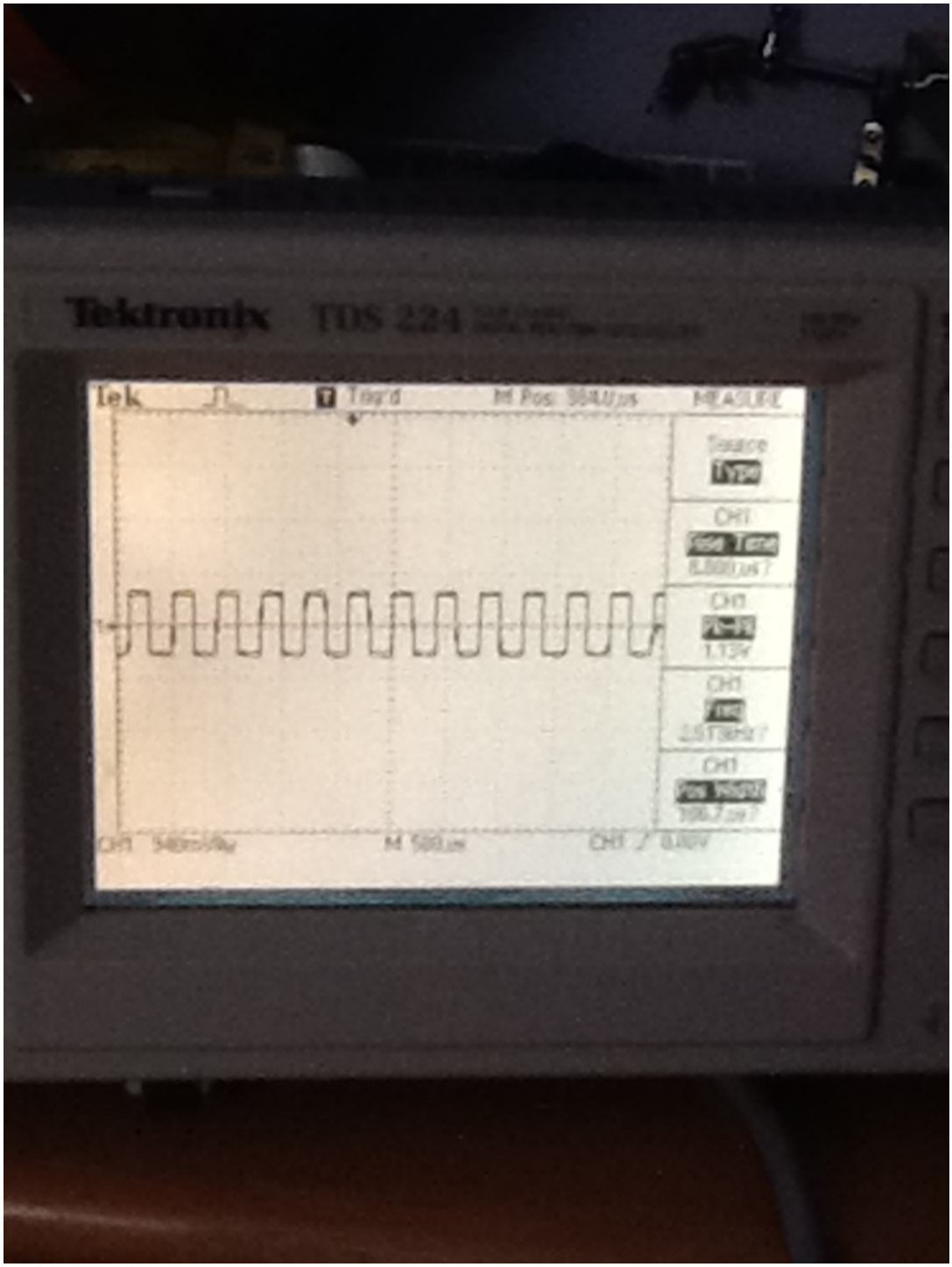

Figure 4: output of signal generator(2)

```
1: /************************************************************
2: *****Single supply signal generator**************************
3: *****By Vijay Natarajan and Charles Wolfe*******************
4: ************************************************************/
5: //control message for digital pot
6: #define WRITEWIPE0 0x00
7: #define WRITEWIPE1 0x10
8: //inits for soft spi - needed for microc functions
9: sbit SoftSpi_SDI at RB4_bit;
10: sbit SoftSpi_SDO at RC7_bit;
11: sbit SoftSpi_CLK at RB6_bit;
12:
13: sbit SoftSpi_SDI_Direction at TRISB4_bit;
14: sbit SoftSpi_SDO_Direction at TRISC7_bit;
15: sbit SoftSpi_CLK_Direction at TRISB6_bit;
16:
17: int WiperValue1 = 128;
18: int WiperValue0 = 0;
19:
20: void set_pot(int value, int control) {
21:  //this function writes the value to the pot by first
22:  //sending the control message, then the value that
23:  //needs to be writen
24:      Soft_SPI_Write(control);
25:      Soft_SPI_Write(value);
26: }
27: void PrintAString(char string[]) {
28:       int k;
29:      for(k=0;string[k]!=0;k++) // Recall: String is an array terminated by zero
30:             Uart1_Write(string[k]);
31: }
32: float calcFrequency(int wiper) {
33:     float freq;
34:     if(wiper > 118) freq= 974.2+7.3455*(129-wiper);
35:     else if(wiper > 108) freq=969.4788+8.2788*(129-wiper);
36:     else if(wiper > 98) freq= 938.0636+9.8182*(129-wiper);
37:     else if(wiper > 88) freq= 906.1394+10.824*(129-wiper);
38:     else if(wiper > 78) freq= 789.6909+13.673*(129-wiper);
39:     else if(wiper > 68) freq= 601.3273+17.345*(129-wiper);
40:     else if(wiper > 58) freq= 396.1636+20.727*(129-wiper);
41:     else if(wiper > 48) freq= 28.151*(129-wiper)-127.7394;
42:     else if(wiper > 38) freq= 39.890*(129-wiper)-1078.273;
43:     else if(wiper > 28) freq= 58.370*(129-wiper)-2765.206;
44:     else if(wiper > 18) freq= 97*(129-wiper)-6672.4;
45:     return freq;
46:    }
47: char crlf[] = "\r\n";
48: char prompt[] = "Frequency:press a to raise,z to lower. Voltage:press s to raise,x
    x to lower\r\n";
49: char freqValue[] = "Frequency = ";
50: char error[] = "Invalid Entry\r\n";
51: char temp[10];
52: void main() {
53:       char input;
54:       TRISC = 0xFF;
55:       TRISB = 0xFF;
56:       TRISA = 0xFF;
57:       ANSEL = 0x00;
58:       ANSELH = 0x00;
59:       C1ON_bit = 0;
60:       C2ON_bit = 0;
61:       Soft_SPI_Init();
```


Figure 5: C code for signal generator(page 1)

```
62:       Uart1_Init(9600);
63:       INTCON = 0;
64:       set_pot(WiperValue1,WRITEWIPE1);
65:       set_pot(WiperValue0,WRITEWIPE0);
66:       Delay_ms(2000);
67:       while(1){
68:                PrintAString(prompt);
69:                while(!Uart1_Data_Ready());
70:                input = Uart1_Read();
71:                if(input == 'z') {
72:                   if(WiperValue1<128) WiperValue1++;
73:                    set_pot(WiperValue1,WRITEWIPE1);
74:                }
75:                else if(input == 'a') {
76:                   if(WiperValue1>6) WiperValue1--;
77:                   set_pot(WiperValue1,WRITEWIPE1);
78:                 }
79:                else if(input == 'x') {
80:                   if(WiperValue0<128) WiperValue0++;
81:                    set_pot(WiperValue0,WRITEWIPE0);
82:                }
83:                else if(input == 's') {
84:                   if(WiperValue0 > 0) WiperValue0--;
85:                   set_pot(WiperValue0,WRITEWIPE0);
86:                }
87:                else PrintAString(error);
88:
89:                PrintAstring(freqValue);
90:                InttoStr((int)calcfrequency(WiperValue1),temp);
91:                PrintAString(temp);
92:                PrintAstring(crlf);
93:       }
94: }
```


Figure 6: C code for signal generator(page 2)