Frequency Counter

EECE 283

March 29, 2002

Raveen Kumaran

11782992

Table of Contents

1.0 Introduction

The objective of this project will be to build a working frequency counter that will count frequencies as low as 10Hz to frequencies as high as a few megahertz. The counter should be reliable and as accurate as possible and able to handle frequencies of varying types of waveforms including sine, square and triangular as well as amplitudes of both large and small magnitudes. The result should then be displayed in a steady and clear form so as to be easily read by anyone.

Frequency counters are useful tools. Currently, the only tool in the lab capable of such a task is an oscilloscope, but while an oscilloscope is capable of displaying more details of a waveform such as its magnitude, it is big and heavy and therefore not exactly conducive as a tool to be used in the field. The proposed frequency counter, while limited in its functions when compared to an oscilloscope, is small in size and therefore more convenient to use when conducting testing, which requires moving from place to place. Alternatively, the frequency counter could serve as a low cost alternative for hobbyists without access to an oscilloscope.

The principle of frequency measurement is quite simple. The device will measure the time difference between two peaks, most likely by measuring the number of peaks in a given period of time and calculating the time for one cycle. The frequency is then obtained by finding the inverse of that value. Figure 1.1 shows a simple sine wave with cursors indicating the time axis values corresponding to the peaks followed by a measurement of the frequency.

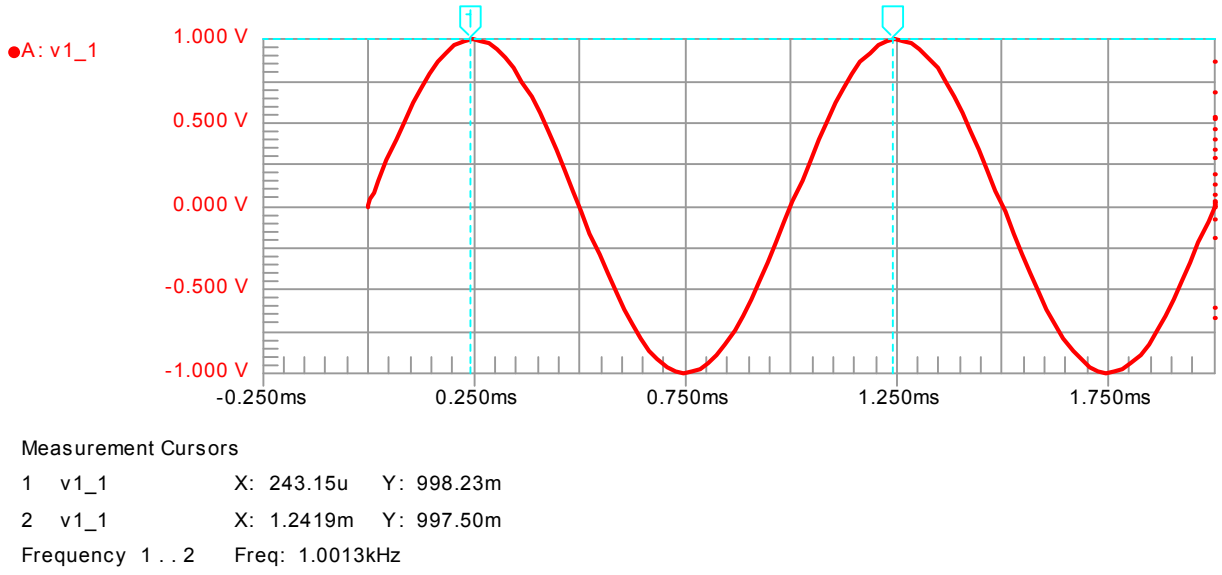


Figure 1.1 Measurement of the Frequency

2.0 Proposed Solution

The frequency counter will be implemented primarily through the use of a PIC16F84-A micro-controller that will count input pulses and display the result on a LCD.

There are 4 main sections to the device and they are described as follows:

2.1 Input Signal Adjuster

The voltage levels of the input signal must be converted to appropriate levels (0V = low , 5V = high) to serve as digital signals for the NAND gates. In order to achieve this, the signal is passed through a DC blocker and a limiter. An N channel Junction FET and a PNP BJT are used to amplify the signal to 0-5V levels. As such, digital pulses are generated with the same frequency as the input signal. This will allow for a greater tolerance of input signals with varying DC offset levels as well as larger amplitudes.

2.2 74HC132 Quad 2-1 NAND Gates with Schmitt Trigger

The adjusted input signal is then passed through a series of NAND gates before being fed into the PIC16F84-A. One of the outputs of the PIC16F84-A is used as feedback to the series of NAND Gates to turn the signal on or off. The 74HC132 is used because it is more tolerant to signal noise due to the hysteresis properties of the Schmitt Trigger. As a result, the frequency counted will be more accurate.

2.3 PIC16F84-A Microcontroller

The final input signal will be received by the microcontroller, which will then count the number of pulses in a specific time period. That value is then sent to a Hitachi LCD. The PIC microcontroller is used because of its simple instruction set, sufficient I/O pins (13 pins) and low cost (under $10).

2.4 Hitachi LCD with HD44780 Controller Chip

This 20x4 character LCD is used to display the frequency measurement. The device is chosen over standard 7 segment displays in order to provide a more professional appearance for the frequency counter since it is able to display ASCII characters which will be used to display units as well as error messages.

3.0 Input Signal Adjuster

3.1 DC Blocker and Limiter Circuit

The first two sections of the input signal adjuster are the DC blocker and the limiter circuit. The circuit and waveforms are shown in Figure 3.1 and Figure 3.2 respectively.

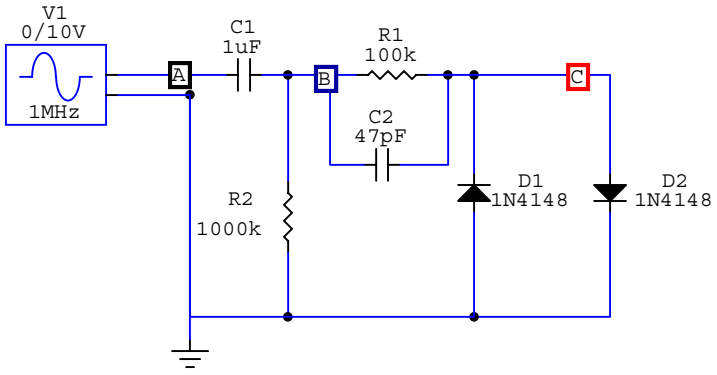


Figure 3.1
DC Blocker and
Limiter Circuit

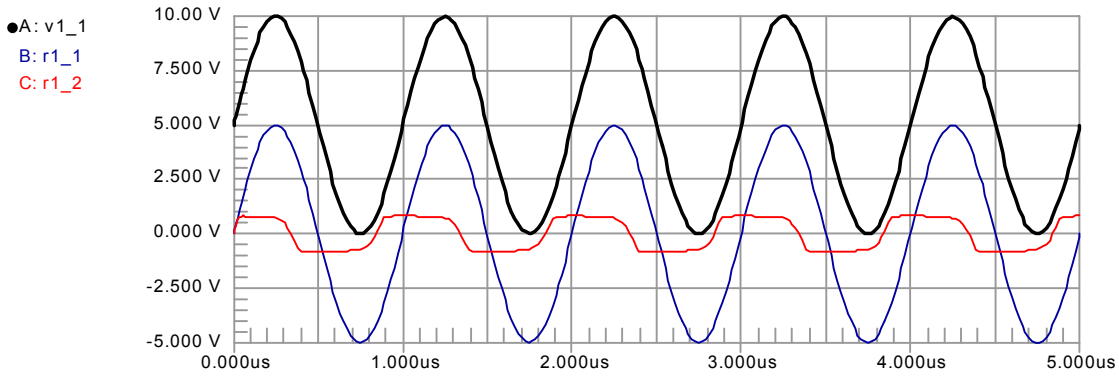


Figure 3.2: Input and Output Waveforms

Point A displays a sine wave input signal of frequency 1Mhz with DC offset of 5V and amplitude 5V. The DC blocker is essentially a high pass filter consisting of C1 and R2. Large values of capacitance and resistance are chosen so that the cutoff frequency is low. In this case, w_c is equal to 1. This allows most frequencies to pass but the DC component of the signal is removed. Point B shows the signal with its 5V DC offset removed.

The signal then passes through a limiter circuit consisting of R1, C2, D1 and D2. Through the use of the resistor and two rectifying diodes, the peaks of the signal voltage are clipped off and instead clamped to an upper and lower threshold. The output voltage oscillates between these two thresholds as shown by Point C. A small capacitance C2 is added to the limiter circuit to improve gain at higher frequencies.

3.2 N Channel JFET and PNP Type BJT

The signal from the DC blocker and limiter circuit is then used by an N channel JFET and a PNP BJT to amplify the signal to 0-5V levels. The entire circuit and corresponding waveforms are shown in Figures 3.3 and 3.4 respectively.

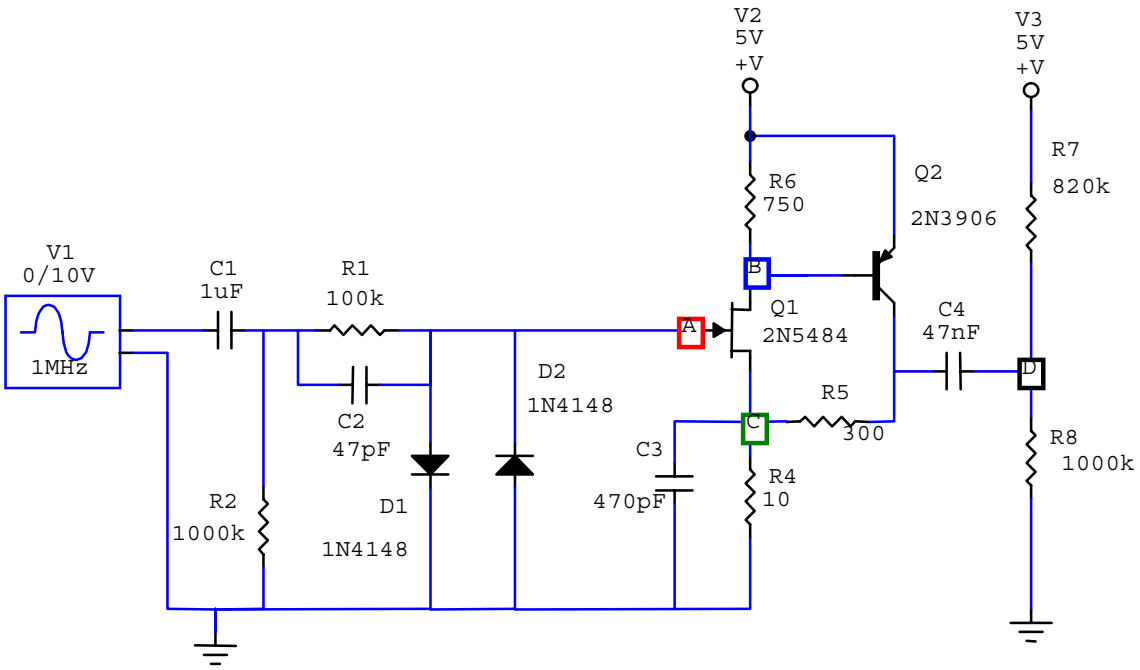


Figure 3.3: Entire Input Signal Adjuster Circuit

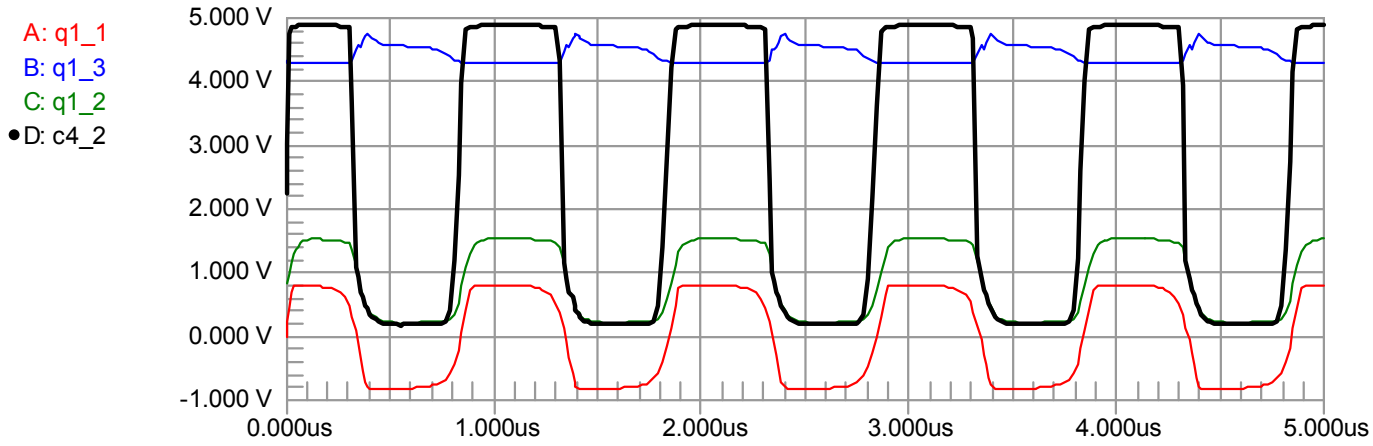
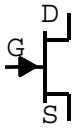
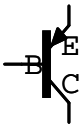


Figure 3.4: Voltage Waveforms at Points A, B, C and D



The voltage at Point A has been clamped by the limiter circuit but still oscillates at the same frequency as the input signal. This voltage becomes the Gate voltage for the Junction Field Effect Transistor (JFET) Q1 and it will be used to control the Drain – Source current. Q1 is configured as a common source amplifier biased by resistor R4. JFETs are voltage sensitive unlike BJT transistors, which are current sensitive. In the case of Q1, which is an N channel, the Drain – Source current increases as the Gate voltage increases. As this occurs, more current flows across resistors R6 and R4. This results in a higher voltage drop across those resistors, which is shown by Point B and C in Figure 3.4.



At this point the Base – Collector voltage of PNP transistor Q2 (configured as a common emitter amplifier) is larger than the threshold voltage set by the Emitter – Base junction. As such, Q2 is switched to active mode and Point D goes high. Resistor R5 is used as a load resistor for the collector output.

The amplified output of Q1 and Q2 is coupled through C4 to the Schmitt Trigger NAND gate. The actual peak to peak amplitude obtained at point D is slightly larger than 3 volts (The waveform in figure 3.4 is merely a Circuit Maker simulation). As such, the voltage is offset with the use of resistors R7 and R8. This raises the output waveform to the range of slightly under 1V to over 4V, which is adequate for triggering high and low levels for the NAND input.

The 2N5484 JFET was chosen because it was the only N-channel RF amplifier available. An N-channel JFET switch (J108) was tested as well but it lacked sufficient gain. The 2N3906 was chosen simply because it was the only PNP BJT available. Values for R4, R5 and R6 were chosen by trial to achieve the maximum gain. The maximum current ratings for Q1 and Q2 were observed during this process.

3.3 Results

From this circuit any input signal (sine, square or triangle so far) with DC offsets and within tolerable amplitude levels (up to 20 Vpp amplitude from a function generator were tested) will be converted into a square wave output with voltage levels 0V to 5V. This is important in two ways: First it protects later components (such as the microcontroller) from high voltages. Second, it allows for the implementation of digital logic through the use of NAND gates. Even though the amount of power that could be delivered is unknown, resistors rated at 0.5W were used since they were adequate for testing conducted with the bench function generator.

4.0 Quad 2 Input NAND with Schmitt Trigger

After the voltage levels of the signal have been adjusted, the signal is passed through a NAND Gate network that has two main purposes:

1) Depending on the state of the microcontroller, the NAND Gate network will feed the signal or prevent it from getting to one of the microcontroller inputs.
2) The fast switching properties of the 74HC132 allow for cleaner pulses (sharper rise/fall times)

The NAND Gate network circuit shown in Figure 4.1 is implemented by using a 74HC132 that has 4 NAND Schmitt Triggers. A truth table for a single 2 input NAND gate is shown below in Table 4.1. A truth table for the network is shown in Table 4.2

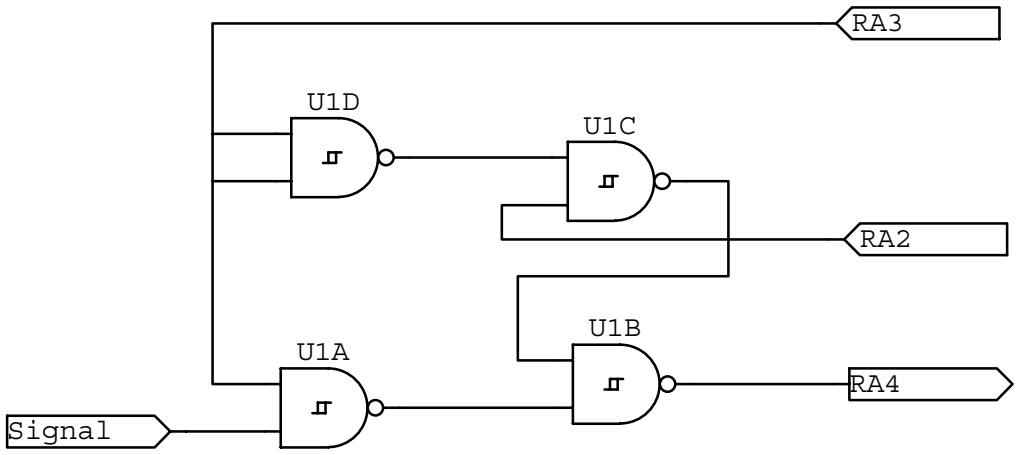


Figure 4.1 NAND Gate Circuit

Input A	Input B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Table 4.1: Truth Table for NAND Gate

Signal	RA3	RA2	OUTPUT (RA4)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 4.2: Truth Table for Circuit in Figure 4.1

In Figure 4.1, the signal input pin represents the signal that was adjusted by the circuit in section 3. The output pin RA4 is the output of the network, which goes to the PIC16F84. The input pins RA2 and RA3 provide feedback from the microcontroller that will control the output of the NAND Gate network.

At the beginning, both RA2 and RA3 are low which means the output is low. RA2 is then made high, which in turn switches the output high. This toggles the PIC16F84 pin that will receive the input signals. When counting is to begin, RA3 is made high and therefore the output will be a clean 0V – 5V square wave signal with the same frequency as the input signal but with smaller rise and fall times. The input and output waveforms are displayed in Figure 4.2, where B denotes the input signal and A, the cleaner output signal.

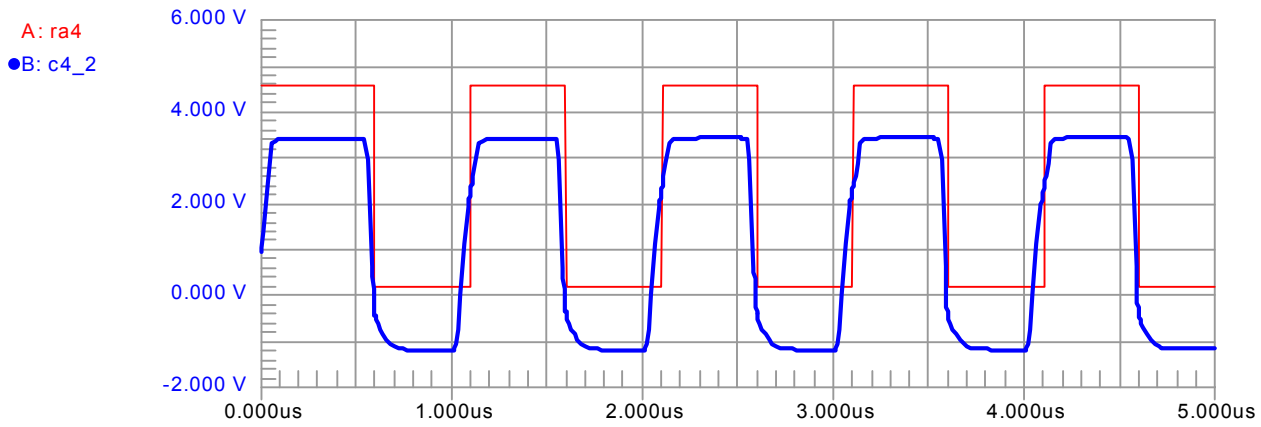


Figure 4.2: Input and Output Waveforms of the NAND Gate Network

4.1 Results

From the Input Signal Adjuster (Section 3.0), the modified input signal is passed to the PIC16F84 via a NAND Schmitt Trigger network that enables or disables the signal. The fast switching properties of the 74HC132 allow for a clean output signal to be generated and its Schmitt Trigger property prevent against inaccurate switching due to signal noise.

Note: The 74HC132 is a high speed CMOS IC. It was chosen over other 132s to ensure compatibility with the PIC16F84, which is CMOS as well.

5.0 PIC16F84-A Microcontroller

5.1 Introduction

The PIC16F84-A microcontroller functions as the core unit in the frequency counter. It receives and clocks the input signal, determines the frequency, determines how it will display that value (units, decimal places, etc) and sends it to the LCD. The pinout diagram for the PIC16F84-A is shown in Figure 5.1 below.

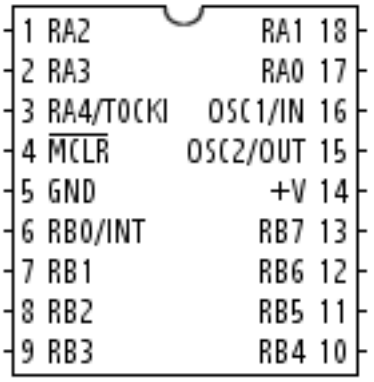


Figure 5.1: PIC16F84-A Pinout Diagram

The PIC16F84-A is a 14-bit RISC microcontroller built by Microchip Technology. It only has 35 instructions, which makes it easier to program than most microcontrollers. It has 13 I/O lines divided into 2 Ports, Port A (RA0 – RA4) and Port B (RB0 – RB7). RA4 can also serve as timer or Real Time Clock Counter (RTCC) which it does, for this application. The PIC16F84-A also supports interrupts, that is, it can break from regular programming upon a signal change at its inputs. One of its more versatile properties is EEPROM Flash memory, which allows it to be reprogrammed many times. The PIC16F84-A in normal mode operation is shown in Figure 5.2.

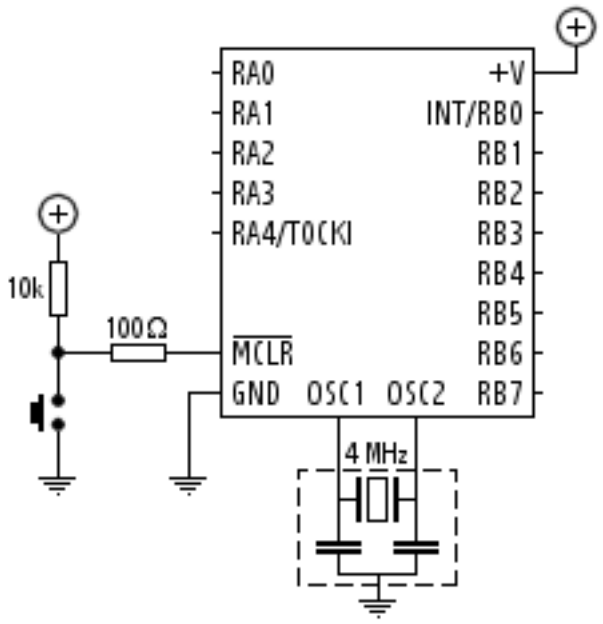


Figure 5.2: PIC16F84-A in Normal Operation Mode

The clock for PIC16F84-A is a 4 MHz crystal oscillator, which is connected between pins 15 and 16 with two capacitors of 22 pF each connecting both pins to ground. Since each instruction takes 4 clock pulses, the PIC16F84-A can perform one million instructions per second. The best source voltage to use would be a stabilized +5V.The _MCLR pin (no 4) must be pulled up to 5V in order for it to function in normal mode. When _MCLR is grounded, operation is reset until _MCLR goes high again.

The PIC16F84-A is programmed using the JDM hardware programmer designed by Jens Dyekjær Madsen together with IC-PROG programming software. Assembly code for the frequency counter was written, tested and simulated using MPLAB. The code was then assembled into hex by MPASM. Both software packages were provided by Microchip Technology.

5.2 Frequency Measurement

In this project, the input signal enters through pin RA4, which has been configured to be a RTCC. The signal is passed through the 74HC132 when RA3 is made high by the PIC16F84-A for a specific gate time, either 0.1 sec or 1 sec. For higher frequencies, the lower gate time is used in order to obtain a better average value. After the gate time elapses, RA3 is made low and the input signal ceases from entering RA4. At this point, the number of pulses (measured by counting the number of falling edges) that were clocked by RA4 during the gate time are stored and ready to be processed.

Initially, the PIC16F84-A begins with a gate time of 0.1 seconds. It executes a subroutine to test if the measured frequency is in the MHz range. If so, it calculates the appropriate position of the decimal and sends that value as well as the units “MHz” to the LCD. If not, it executes a different subroutine to test if the measurement is in the KHz range. Just as before, if the measurement is indeed in the KHz range, it determines the location of the decimal and appends the units “KHz” to the reading before sending it to the LCD. If not, it executes yet another similar subroutine but for the “Hz” range instead.

Note that any one of the above subroutines can be accessed from each other. That is, if the current reading is 100.5 KHz and the frequency is increased to 1.5 MHz, the “KHz” subroutine will be exited and the “MHz” subroutine will be executed. These subroutines are looped through continuously until the processor is reset or power is turned off. If the input signal is removed, the LCD will just display the units Hz with no reading. The full code is provided in Appendix B.

6.0 Hitachi LCD with HD44780 Controller Chip

The LM044L LCD is used to display the frequency measured by the PIC16F84-A. The LM044L is a display module consisting of a 20x4 character LCD screen that has a built in HD44780 controller chip. The HD44780 is an industry standard capable of receiving ASCII characters for display. It features 14 pins:

Pin	Symbol	Level	I/O	Function
1	Vss	-	-	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)
3	Vee	-	-	Contrast adjust
4	RS	0/1	I	0 = Instruction input 1 = Data input
5	R/W	0/1	I	0 = Write to LCD module 1 = Read from LCD module
6	E	1, 1-->0	I	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)
8	DB1	0/1	I/O	Data bus line 1
9	DB2	0/1	I/O	Data bus line 2
10	DB3	0/1	I/O	Data bus line 3
11	DB4	0/1	I/O	Data bus line 4
12	DB5	0/1	I/O	Data bus line 5
13	DB6	0/1	I/O	Data bus line 6
14	DB7	0/1	I/O	Data bus line 7 (MSB)

Figure 6.1: Pin Information for HD44780 Controller Chip

The controller chip is configured to run in 8 bit mode which means that all 8 I/O pins must be used. This is accomplished by utilizing Port B of the PIC16F84-A (which has 8 I/O lines) entirely. On the other hand, the first 3 pins of Port A (RA0- RA2) are used to control RS, R/W and E. In order to write a character to the LCD, the 8 bit address of the location must be placed by the microcontroller on Port B. An address map is shown in Figure 6.2.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	← Character position (dec.)
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	← Row0 DDRAM address (hex)
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	← Row1 DDRAM address (hex)
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	← Row2 DDRAM address (hex)
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	← Row3 DDRAM address (hex)

Figure 6.2: Address Locations for the 20x4 Character LCD

Next, the RS and R/W must be set low and bit 7 of Port B must be set high. When Enable is later set high, the address is written to the LCD and the cursor is moved to that position. After that, the code of the intended character is written to Port B of the microcontroller. A table of character codes is displayed below in Figure 6.3. To write the selected character on the LCD, RS must be set high, R/W set low and Enable set high.

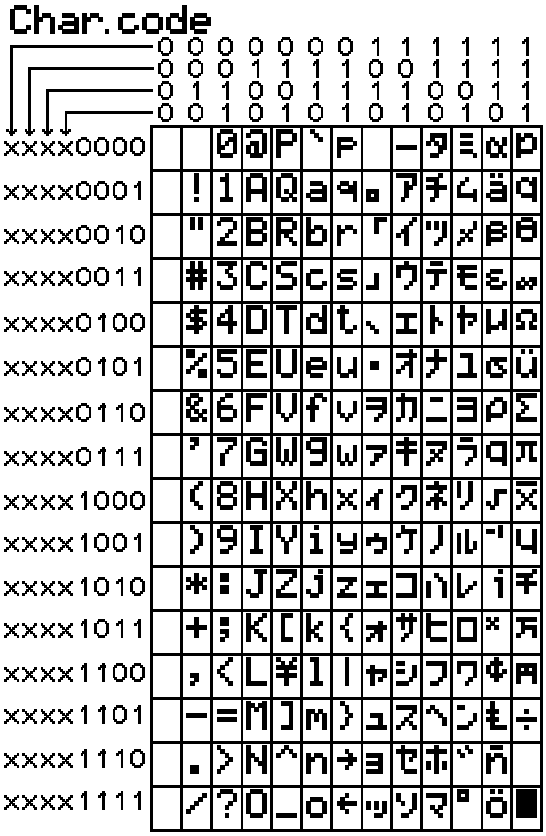


Figure 6.3: Character Codes for the HD44780

7.0 Results

The circuit was initially constructed, tested and debugged on a breadboard. Testing conducted using the bench function generator was extremely successful, with the circuit able to measure frequencies encompassing the entire frequency (1 Hz to 4 MHz) and voltage (1 to 20 Vpp) range of the function generator. After this successful and reliable operation, the final circuit was soldered on veroboard to make for a more tidy and permanent device. To incorporate mobility, power was provided in the form of a 9 Volt battery which was regulated via a 7805 to provide the steady 5 Volts required by the device. A LED in series with a 100 Ohm resistor was placed between power lines as an indicator of the device being on or off.

One problem that developed after soldering involved the Input Signal Adjuster, specifically the gain from Q1 and Q2 (Figure 3.3) was now less responsive to higher frequencies. As such at higher frequencies, insufficient gain was produced to activate the high and low levels of the NAND gate with Schmitt Trigger. Tweaking the bias resistances led to an improvement in frequency response, where measurements of frequencies up to 1 MHz could now be made. This was deemed adequate enough, as the repetitive switching of components on a soldered veroboard circuit was time consuming and potentially damaging to the other components. This problem most likely stems from the inherent differences in resistance and capacitance between the lines of the breadboard and veroboard, especially when the bias resistances are relatively small (R4 is 10 Ohms).

The Gantt chart below details approximate hours spent on each section of the project:

Task	Time (hours)
Proposal, Initial Design	3
Purchase of Supplies	2
Input Signal Adjuster	8
Quad 2 Input NAND Gate with Schmitt Trigger	5
PIC16F84-A	30
Hitachi LCD with HD44780 Controller Chip	7
Breadboard Testing	10
Soldering	14
Final Testing and Debugging	14

8.0 Conclusion

At the end of this project a considerable wealth of knowledge regarding microcontrollers, LCDs as well as the principles of frequency measurement were gained. The finished product was a successful and accurate frequency measuring device that was economical as well as convenient in terms of mobility. Through the use of the input adjustment circuit, input waveforms of varying voltage amplitude as well DC offsets were able to be processed and their frequencies measured. The project entailed many hours, mostly spent learning the operation and potential of the microcontroller as well as methods of interfacing it with other devices. However, it was worth the effort as the frequency counter is an extremely useful tool. A diagram of the final circuit in its entirety is displayed in Figure 8.1.

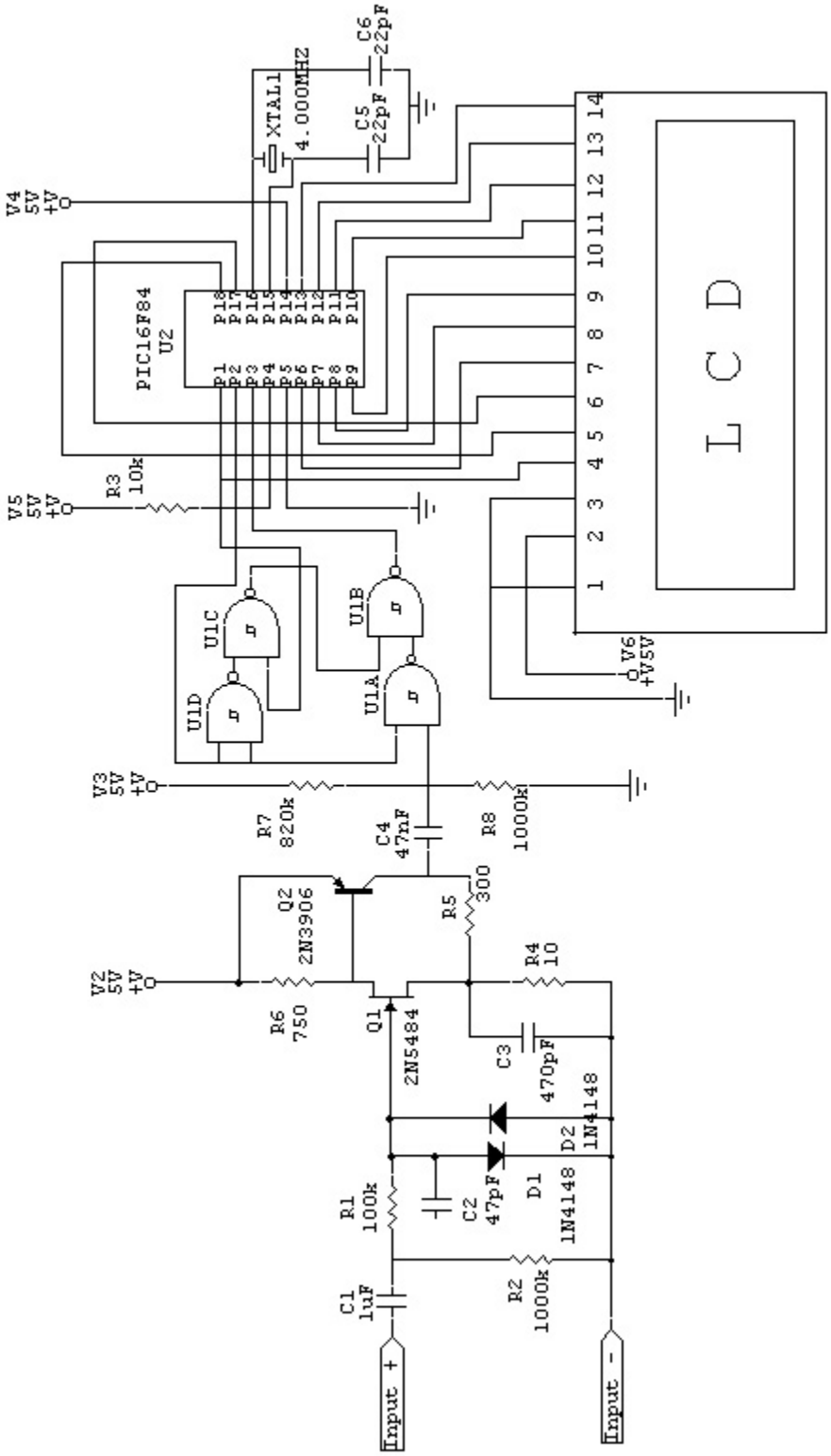


Figure 8.1: Full Circuit for Frequency Counter

References

1. PIC16F84-A Microcontroller Datasheet

   http://www.microchip.com/download/lit/pline/picmicro/families/16f8x/35007b.pdf

2. 74HC132 Quad 2 Input NAND with Schmitt Trigger Datasheet

   http://atrey.karlin.mff.cuni.cz/~clock/twibright/ronja/datasheets/74hc132.pdf

3. HD44780 Instruction Set http://www.hkrmicro.com/course/hd44780.html

4. 7805 5V Regulator Datasheet http://www.national.com/pf/LM/LM7805C.html

5. 2N5484 Datasheet http://www.fairchildsemi.com/pf/2N/2N5484.html

6. 2N3906 Datasheet http://www.fairchildsemi.com/pf/2N/2N3906.html

7. A.S Sedra and K.C Smith. Microelectronic Circuits. 4th Edition. Oxford University Press, New York. 1998. pp 447-452.

8. MPASM User's Guide

   http://www.microchip.com/download/tools/picmicro/code/mpasm/33014g.pdf

9. JDM Programmer http://www.jdm.homepage.dk/newpic.htm

10. IC-PROG Prototype programmer http://www.ic-prog.com/

11. MPLAB Manual

    http://www.microchip.com/download/tools/picmicro/devenv/manual/51025e.pdf

12. Ouwehand, Peter. How to Control a HD44780 Based LCD.

    http://home.iae.nl/users/pouweha/lcd/lcd.shtml

13. Knott, Graham. The FET Amplifier.

    http://ourworld.compuserve.com/homepages/g_knott/elect247.htm

14. Simple Low Cost Frequency Meter. http://www.madlab.org/kits/frqmeter.html

Appendix A: List of Parts

Component	Description	Quantity
PIC16F84-A	Microcontroller	1
74HC132	Quad NAND w Schmitt Trigger	1
LM044L	20x4 LCD with HD44780 controller chip	1
FOX4.0	4 MHz Crystal Oscillator	1
7805	5 Volt Regulator	1
2N5484	N Channel JFET RF Amplifier	1
2N3906	PNP General Purpose Amplifier	1
1N4148	General Purpose Silicon Diode	2
1 MW	Resistor	2
820 KW	Resistor	1
100 KW	Resistor	1
10 KW	Resistor	1
750 W	Resistor	1
300 W	Resistor	1
100 W	Resistor	1
10 W	Resistor	1
1 mF	Capacitor	1
47 nF	Capacitor	1
470 pF	Capacitor	1
47 pF	Capacitor	1
22 pF	Capacitor	2
LED	Diode	1

Note: All resistances are rated at 0.5 W

Appendix B: FreqCount.ASM

```
;*****************************************************************************
;                       FREQUENCY COUNTER

;
;*****************************************************************************
;
;
;watchdog disabled
;
            list        P=16F84

;=========================================================
;
;     Configuration Bits
;
;=========================================================

_CP_ON                  EQU    H'000F'
_CP_OFF                 EQU    H'3FFF'
_PWRTE_ON               EQU    H'3FF7'
_PWRTE_OFF              EQU    H'3FFF'
_WDT_ON                 EQU    H'3FFF'
_WDT_OFF                EQU    H'3FFB'
_LP_OSC                 EQU    H'3FFC'
_XT_OSC                 EQU    H'3FFD'
_HS_OSC                 EQU    H'3FFE'
_RC_OSC                 EQU    H'3FFF'
;-------------------------------------------------------

ind         equ         0h
rtcc        equ         1h
pc          equ         2h
status      equ         3h
fsr         equ         4h
port_a      equ         5h
port_b      equ         6h
;port_c     equ         7h
c           equ         0h
dc          equ         1h
z           equ         2h
pd          equ         3h
to          equ         4h
MSB         equ         7h
LSB         equ         0h
;
cnt         equ         2h
rs          equ         2h
rw          equ         1h
e           equ         0h
o           equ         7h
;
count1      equ         2ch
count2      equ         2dh
in_reg      equ         2eh
addcnt      equ         2fh
gate        equ         0Ch
cnt1        equ         0Dh
cnt2        equ         0Eh
cnt3        equ         0Fh
calc1       equ         10h
calc2       equ         11h
calc3       equ         12h
sum1        equ         13h
sum2        equ         14h
sum3        equ         15h
rtcc2       equ         16h
;-------------------------------------------------
;we have to set the configuration bits
```

```
;               __config a & b & c
;       _rc_osc, _xt_osc, _hs_osc, _lp_osc oscillator type
;       _wdt_on, _wdt_off watchdog timer
;       _cp_on, _cp_off code protect
;       _pwrte_on, _pwrte_off power up timer enable

;               __config  _xt_osc & _wdt_off & _pwrte_on & _cp_off

;-------------------------------------------------
;-------------------------------------------------
;-------------------------------------------------
                org     0
                goto    start

;
int_del         movlw   0x05                    ;delay 5.000 ms (4 MHz clock)
                movwf   count1
d1              movlw   0xA5
                movwf   count2
d2              decfsz  count2    ,f
                goto    d2
                decfsz  count1    ,f
                goto    d1
                retlw   0x00
;
lcd_out         movwf   port_b                  ;load data into port_b
                movlw   b'00000000'             ;define port_b as output
                tris    port_b
                bsf     port_a,rs  ;rs = data
                bcf     port_a,rw ;r/w = write
                bsf     port_a,e   ;toggle enable
                bcf     port_a,e
                movlw   b'11111111'             ;define port_b as input
                tris    port_b
                bcf     port_a,rs  ;rs = instruction
                bsf     port_a,rw ;r/w = read
                bsf     port_a,e   ;enable high
                movf    port_b,w   ;get address counter
                movwf   addcnt
                bsf     addcnt,7
                bcf     port_a,e   ;enable low
out1            bsf     port_a,e   ;enable high
                btfss   port_b,7   ;test busy flag
                goto    out2
                bcf     port_a,e   ;enable low
                goto    out1
out2            bcf     port_a,e   ;enable low
                goto    shift
;
inst            movwf   port_b                  ;load instruction into port_b
                movlw   b'00000000'             ;define port_b as output
                tris    port_b
                bcf     port_a,rs  ;rs = instruction
                bcf     port_a,rw ;r/w = write
                bsf     port_a,e   ;toggle enable
                bcf     port_a,e
                movlw   b'11111111'             ;define port_b as input
                tris    port_b
                bsf     port_a,rw ;r/w = read
inst1           bsf     port_a,e   ;enable high
                btfss   port_b,7   ;test busy flag
                goto    inst2
                bcf     port_a,e   ;enable low
                goto    inst1
inst2           bcf     port_a,e   ;enable low
                retlw   0x00
;
;-----------------------------------
; LCD 20x4
```

```
shift   btfss   addcnt,0  ;shift to opposite side of display?
        retlw   0x00
        btfss   addcnt,1
        retlw   0x00
        btfsc   addcnt,2
        retlw   0x00
        btfsc   addcnt,3
        retlw   0x00
        btfss   addcnt,4
        retlw   0x00
        retlw   0x00
                                ;movlw   0x27   ; used  to be 39
                                ;addwf   addcnt,f
                                ;bsf     addcnt,7
                                ;movf    addcnt,w
                                ;goto    inst
;
;-------------------------------------
sub     bcf     status,o  ;clear overflow bit
        movf    calc1,w         ;subtract calc1 from cnt1
        subwf   cnt1    ,f
        btfsc   status,c
        goto    sb1
        movlw   0x01            ;borrow from cnt2 if overflow
        subwf   cnt2    ,f
        btfsc   status,c
        goto    sb1
        subwf   cnt3    ,f              ;borrow from cnt3 if cnt2 overflow
        btfss   status,c
        bsf     status,o  ;set overflow bit if result is negative
sb1     movf    calc2,w         ;subtract calc2 from cnt2
        subwf   cnt2    ,f
        btfsc   status,c
        goto    sb2
        movlw   0x01            ;borrow from cnt3 if cnt2 overflow
        subwf   cnt3    ,f
        btfss   status,c
        bsf     status,o  ;set overflow bit if result is negative
sb2     movf    calc3,w         ;subtract calc3 from cnt3
        subwf   cnt3    ,f
        btfss   status,c
        bsf     status,o  ;set overflow bit if result is negative
        retlw   0x00
;-------------------------------------
add     movf    calc1,w         ;add calc1 to cnt1
        addwf   cnt1    ,f
        btfss   status,c
        goto    ad1
        incfsz  cnt2    ,f              ;add to cnt2 if cnt1 overflow
        goto    ad1
        incf    cnt3    ,f              ;add to cnt3 if cnt2 overflow
ad1     movf    calc2,w         ;add calc2 to cnt2
        addwf   cnt2    ,f
        btfsc   status,c
        incf    cnt3    ,f              ;add to cnt3 if cnt2 overflow
        movf    calc3,w         ;add calc3 to cnt3
        addwf   cnt3    ,f
        retlw   0x00
;-------------------------------------
cnvt    movlw   0x07            ;7 digits in display
        movwf   count1
        movlw   0x19            ;set fsr for MSB in display
        movwf   fsr
        movlw   0x2F            ;one less that ASCII "0"
cnvt0   movwf   ind
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    cnvt0
        movlw   0x0F            ;load "1,000,000" in calc1-3
        movwf   calc3
```

```
        movlw   0x42
        movwf   calc2
        movlw   0x40
        movwf   calc1
cnvt1   call    sub             ;subtract number from count
        incf    19      ,f              ;increment 1,000,000's register
        movlw   0x3A
        xorwf   19,w
        btfsc   status,z
        goto    overflow
        btfss   status,o        ;check if overflow
        goto    cnvt1
        call    add             ;add back last number
        movlw   0x01            ;load "100,000" in calc1-3
        movwf   calc3
        movlw   0x86
        movwf   calc2
        movlw   0xA0
        movwf   calc1
cnvt2   call    sub             ;subtract number from count
        incf    1A      ,f      ;increment 100,000's register
        btfss   status,o        ;check if overflow
        goto    cnvt2
        call    add             ;add back last number
        clrf    calc3           ;load "10,000" in calc1-3
        movlw   0x27
        movwf   calc2
        movlw   0x10
        movwf   calc1
cnvt3   call    sub             ;subtract number from count
        incf    1B      ,f      ;increment 10,000's register
        btfss   status,o        ;check if overflow
        goto    cnvt3
        call    add             ;add back last number
        movlw   0x03            ;load "1,000" in calc1-3
        movwf   calc2
        movlw   0xE8
        movwf   calc1
cnvt4   call    sub             ;subtract number from count
        incf    1C      ,f      ;increment 1,000's register
        btfss   status,o        ;check if overflow
        goto    cnvt4
        call    add             ;add back last number
        clrf    calc2           ;load "100" in calc1-3
        movlw   0x64
        movwf   calc1
cnvt5   call    sub             ;subtract number from count
        incf    1D      ,f      ;increment 100's register
        btfss   status,o        ;check if overflow
        goto    cnvt5
        call    add             ;add back number
        movlw   0x0A            ;load "10" in calc1-3
        movwf   calc1
cnvt6   call    sub             ;subtract number from count
        incf    1E      ,f      ;increment 10's register
        btfss   status,o        ;check if overflow
        goto    cnvt6
        call    add             ;add back last number
        movf    cnt1,w          ;put remainder in 1's register
        addwf   1F      ,f
        incf    1F      ,f
        retlw   0x00
;-----------------------------------
count   movlw   b'00110111'     ;rtcc = ext, 1/256
        option
        movlw   b'00010000'     ;define port_a as output
        tris    port_a
        bcf     port_a,3
        bcf     port_a,2
        clrf    cnt3
```

```
        clrf    rtcc
        clrf    rtcc2
        bsf     port_a,2  ;toggle rtcc pin
        bcf     port_a,2
        movf    gate,w            ;get gate time
        movwf   count1
        bsf     port_a,3  ;start count
fr4     movlw   0xFA
        movwf   count2
        goto    fr6
fr5     nop
        nop
        nop
        nop
        nop
        nop
fr6     movf    rtcc,w            ;test for rtcc rollover (12)
        subwf   rtcc2     ,f
        btfss   status,z
        goto    fr7
        nop
        goto    fr8
fr7     btfsc   status,c
        incf    cnt3      ,f
fr8     movwf   rtcc2
        nop
        nop
        nop
        decfsz  count2    ,f
        goto    fr5
        decfsz  count1    ,f
        goto    fr4
        bcf     port_a,3  ;stop count
        movf    rtcc,w            ;get rtcc count
        movwf   cnt2
        subwf   rtcc2     ,f                ;test for rtcc rollover
        btfss   status,c
        goto    fr9
        btfss   status,z
        incf    cnt3      ,f
fr9     clrf    cnt1              ;set to get prescaler count
fr10    decf    cnt1      ,f
        bsf     port_a,2  ;toggle rtcc pin
        bcf     port_a,2
        movf    rtcc,w            ;test if rtcc has changed
        xorwf   cnt2,w
        btfsc   status,z
        goto    fr10
        retlw   0x00
;
;*****************************************************************************
;                           START
;*****************************************************************************
;
start   clrf    port_a            ;instruction, write, enable low
        movlw   b'00010000'
        tris    port_a
        clrf    port_b
        movlw   b'00000000'
        tris    port_b
        call    int_del
        call    int_del
        call    int_del
        movlw   0x38              ;initialize display
        call    inst              ;movwf   port_b
                                  ;bsf     port_a,e   ;toggle enable
        call    int_del
        movlw   0x38              ;bcf     port_a,e
        call    inst              ;bsf     port_a,e   ;toggle enable
        call    int_del
```

```
        movlw   0x38            ;bcf    port_a,e
        call    inst            ;bsf    port_a,e   ;toggle enable
        call    int_del
        movlw   0x38
        call    inst
        call    int_del
                                ;bcf    port_a,e
        movlw   0x38            ;function
        call    inst
        call    int_del

        movlw   b'00000110'     ;entry mode
        call    inst

        movlw   b'00001100'     ;display on, cursor off
        call    inst
        movlw   b'00000001'     ;clear display
        call    inst

        movlw   0xC0
        call    inst

        movlw  0x20             ;space
        call    lcd_out
        movlw  0x46             ;F
        call    lcd_out
        movlw  0x72             ;r
        call    lcd_out
        movlw  0x65             ;e
        call    lcd_out
        movlw  0x71             ;q
        call    lcd_out
        movlw  0x75             ;u
        call    lcd_out
        movlw  0x65             ;e
        call    lcd_out
        movlw  0x6E             ;n
        call    lcd_out
        movlw  0x63             ;c
        call    lcd_out
        movlw  0x79             ;y
        call    lcd_out
        movlw  0x20             ;space
        call    lcd_out
        movlw  0x43             ;C
        call    lcd_out
        movlw  0x6F             ;o
        call    lcd_out
        movlw  0x75             ;u
        call    lcd_out
        movlw  0x6E             ;n
        call    lcd_out
        movlw  0x74             ;t
        call    lcd_out
        movlw  0x65             ;e
        call    lcd_out
        movlw  0x72             ;r
        call    lcd_out

        movlw   0xD4
        call    inst
;-----------------------------------
mhz     movlw   0x14            ;0.1 sec gate
        movwf   gate
        call    count
        call    cnvt            ;convert binary to BCD
        movlw   0x30            ;test if "0"
        xorwf   19,w
        btfss   status,z
```

```
        goto    mhz1
        movlw   0x30            ;test if "0"
        xorwf   1A,w
        btfsc   status,z
        goto    khz1
mhz1    movlw   0xD4            ;set display address
        call    inst
        movlw   0x02            ;output first 2 characters
        movwf   count1
        movlw   0x19            ;MSD of freq
        movwf   fsr
mhz2    movlw   0x30            ;test if "0"
        xorwf   ind,w
        btfss   status,z
        goto    mhz3
        movlw   0x20            ;change preceeding "0's" to "space"
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    mhz2
        goto    mhz4
mhz3    movf    ind,w
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    mhz3
mhz4    movlw   0x2E            ;"."
        call    lcd_out
        movlw   0x05            ;output last 5 characters
        movwf   count1
mhz5    movf    ind,w
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    mhz5
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x4D            ;"M"
        call    lcd_out
        movlw   0x48            ;"H"
        call    lcd_out
        movlw   0x7A            ;"z"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        goto    mhz
;-----------------------------------
khz     movlw   0x14            ;0.1 sec gate
        movwf   gate
        call    count
        call    cnvt            ;convert binary to BCD
        movlw   0x30            ;test if 0
        xorwf   19,w
        btfss   status,z
        goto    mhz1
        movlw   0x32            ;test if < 2
        subwf   1A,w
        btfsc   status,c
        goto    mhz1
        movlw   0x30            ;test if "0"
        xorwf   1A,w
        btfss   status,z
        goto    khz1
        movlw   0x30            ;test if "0"
        xorwf   1B,w
        btfsc   status,z
        goto    xkhz
khz1    movlw   0xD4            ;set display address
```

```
        call    inst
        movlw   0x05            ;output first 5 characters
        movwf   count1
        movlw   0x19            ;MSD of freq
        movwf   fsr
khz2    movlw   0x30            ;test if "0"
        xorwf   ind,w
        btfss   status,z
        goto    khz3
        movlw   0x20            ;change preceeding "0's" to "space"
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    khz2
        goto    khz4
khz3    movf    ind,w
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    khz3
khz4    movlw   0x2E            ;"."
        call    lcd_out
        movf    ind,w           ;output last 2 characters
        call    lcd_out
        incf    fsr     ,f
        movf    ind,w
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x4B            ;"K"
        call    lcd_out
        movlw   0x48            ;"H"
        call    lcd_out
        movlw   0x7A            ;"z"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        goto    khz
;-----------------------------------
xkhz    movlw   0xC8            ;1 sec gate
        movwf   gate
        call    count
        call    cnvt            ;convert binary to BCD
        movlw   0x30            ;test if 0
        xorwf   19,w
        btfss   status,z
        goto    khz
        movlw   0x32            ;test if < 2
        subwf   1A,w
        btfsc   status,c
        goto    khz
        movlw   0x30            ;test if 0
        xorwf   1A,w
        btfss   status,z
        goto    xkhz1
        movlw   0x30            ;test if 0
        xorwf   1B,w
        btfsc   status,z
        goto    hz0
xkhz1   movlw   0xD4            ;set display address
        call    inst
        movlw   0x04            ;output first 4 characters
        movwf   count1
        movlw   0x19            ;MSD of freq
        movwf   fsr
xkhz2   movlw   0x30            ;test if "0"
        xorwf   ind,w
        btfss   status,z
```

```
        goto    xkhz3
        movlw   0x20            ;change preceeding "0's" to "space"
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    xkhz2
        goto    xkhz4
xkhz3   movf    ind,w
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    xkhz3
xkhz4   movlw   0x2E            ;"."
        call    lcd_out
        movf    ind,w           ;output last 3 characters
        call    lcd_out
        incf    fsr     ,f
        movf    ind,w
        call    lcd_out
        incf    fsr     ,f
        movf    ind,w
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x4B            ;"K"
        call    lcd_out
        movlw   0x48            ;"H"
        call    lcd_out
        movlw   0x7A            ;"z"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        movlw   0x20            ;"space"
        call    lcd_out
        goto    xkhz
;-----------------------------------
hz      movlw   0xC8            ;1 sec gate
        movwf   gate
        call    count
        call    cnvt            ;convert binary to BCD
        movlw   0x30            ;test if "0"
        xorwf   19,w
        btfss   status,z
        goto    xkhz1
        movlw   0x30            ;test if "0"
        xorwf   1A,w
        btfss   status,z
        goto    xkhz1
        movlw   0x32            ;test if < 2
        subwf   1B,w
        btfsc   status,c
        goto    xkhz1
hz0     movlw   0xD4            ;set display address
        call    inst
        movlw   0x07            ;output first 7 characters
        movwf   count1
        movlw   0x19            ;MSD of freq
        movwf   fsr
hz1     movlw   0x30            ;test if "0"
        xorwf   ind,w
        btfss   status,z
        goto    hz2
        movlw   0x20            ;change preceeding "0's" to "space"
        call    lcd_out
        incf    fsr     ,f
        decfsz  count1  ,f
        goto    hz1
        goto    hz3
hz2     movf    ind,w
        call    lcd_out
```

```
            incf    fsr     ,f
            decfsz  count1  ,f
            goto    hz2
hz3         movlw   0x20            ;"space"
            call    lcd_out
            movlw   0x48            ;"H"
            call    lcd_out
            movlw   0x7A            ;"z"
            call    lcd_out
            movlw   0x20            ;"space"
            call    lcd_out
            movlw   0x20            ;"space"
            call    lcd_out
            movlw   0x20            ;"space"
            call    lcd_out
            movlw   0x20            ;"space"
            call    lcd_out
            goto    hz
;-----------------------------------
overflow    movlw   0x01            ;clear display
            call    inst
            movlw   0xD6            ;display address
            call    inst
            movlw   0x4F            ;"O"
            call    lcd_out
            movlw   0x76            ;"v"
            call    lcd_out
            movlw   0x65            ;"e"
            call    lcd_out
            movlw   0x72            ;"r"
            call    lcd_out
            movlw   0x66            ;"f"
            call    lcd_out
            movlw   0x6C            ;"l"
            call    lcd_out
            movlw   0x6F            ;"o"
            call    lcd_out
            movlw   0x77            ;"w"
            call    lcd_out
            movlw   0x02            ;cursor at home
            call    inst
            goto    mhz
;-----------------------------------
            end
```