TABLE OF CONTENT

PIC Controlled Oscilloscope

The aim of this assignment is to design and build a simple oscilloscope using the Microchip PIC 16F877 Micro controller, the 2 potentiometers on the analogue module that connects to the DSX Kit, the DSX 9 pin serial port and LCD (Liquid Crystal Display).

The potentiometers on the analogue module will provide the frequency (X-axis) and amplitude (Y-axis) controllers.

The positive (left) potentiometer is used to move the Y-axis, while the negative (right) potentiometer is used to move the X-axis.

Below are 2 diagrams that show what the output looks like.

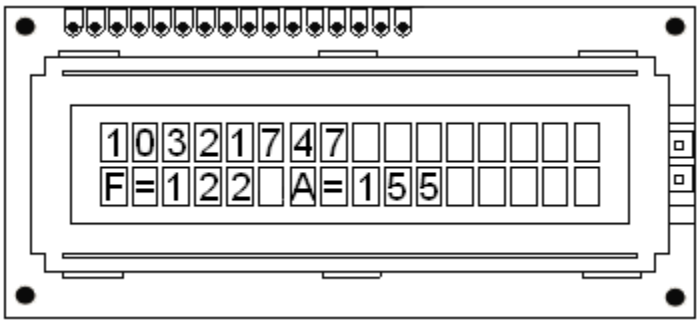


Figure 1.0: LCD

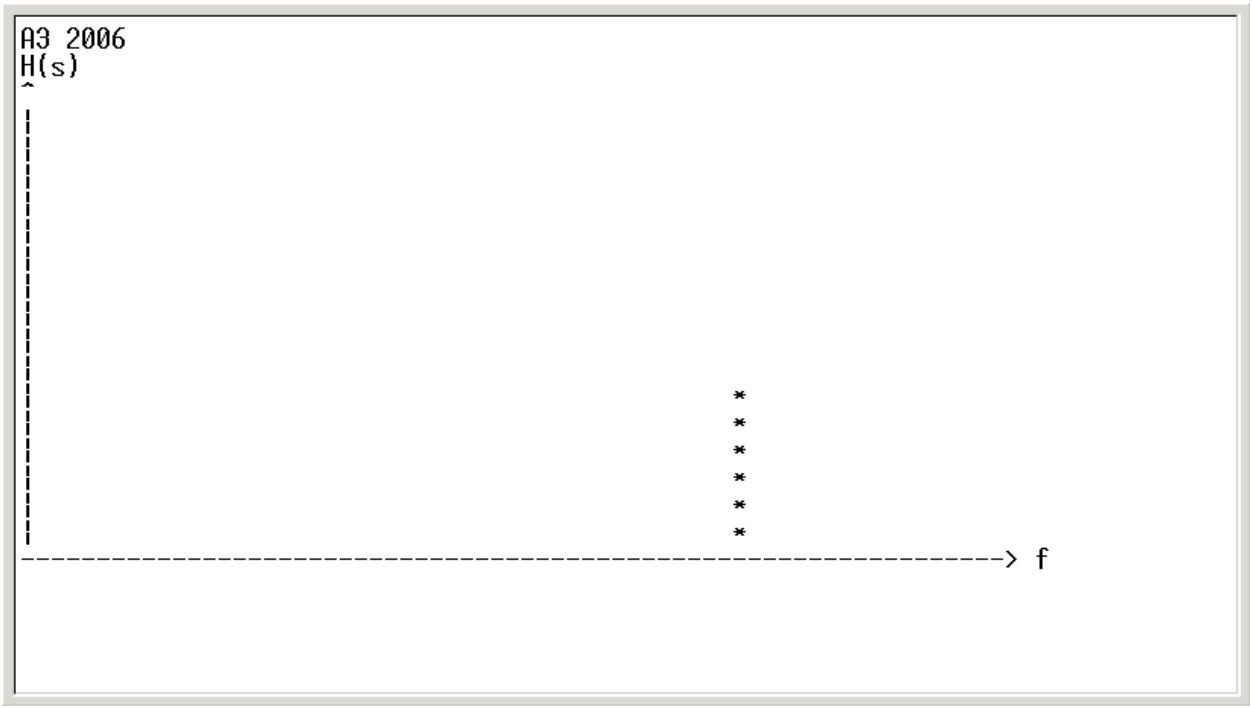


Figure 1.1: Graph on Hyper Terminal

Operation of the Assembly Code

The following pseudo code demonstrates the operation of the assembly code.

```
BEGIN
      Setup ports
      Clear interrupt flags registers
      Turn off interrupts

      Initialize LCD
            Set to ram bank 0
            Set PORTE to receive commands
            Power up
            Set LCD to 8 bits
            Set 8 bit interface 2 line, 5x7 character format
            Clear all DDRAM and set cursor to address 01
            Display ON

            Set PORTE to receive data
            Write student number
            Write "F=" and "A="

      Initialize Hyper Terminal
            Set to ram bank 0
            Send the following text; char by char
                  A3 2006
                  H(s)
                  ^
            Temp = 16
            Send "|" then carriage return
                  Temp = Temp - 1
                  Go to Send while Temp not = 0

            Temp = 65
            Send "-" then carriage return
                  Temp = Temp - 1
                  Go to Send while Temp not = 0

            Send >
            Send space
            Send f


      LOOP:

      Convert analogue signal to digital
            Set to channel 2
            Acquire time; at least 20µs
            Start conversion (set GO/DONE)
            Poll GO/DONE to be cleared
            Write result to register (only high byte ADDRESH)

            Set to channel 3
            Repeat process for Amplitude
```

```
      Scale Data (From 8bit to 6bit (Freq.) & to 4bit (Amp.))
            Temp = 2
            Shift
                  Right shift scaled Frequency
                  Clear bit 7
                  Temp = Temp - 1
                  If Temp not = 0 then go to Shift
            Store scaled frequency in user define register

            Temp = 4
            Repeat process for Amplitude to be scaled


      If Frequency changed then
      Update LCD

      If Amplitude changed then
      Update LCD

      If scaled Frequency changed then
      Update Hyper Terminal

      If scaled Amplitude changed then
      Update Hyper Terminal

      Set old scaled frequency to scaled frequency
      Set old scaled amplitude to scaled amplitude

      GOTO: LOOP
END
```

The following pseudo code demonstrates the operation of the LCD Update

```
BEGIN
      Set PORTE to receive commands
      Move cursor to 3rd position 2nd line of the LCD
      Convert frequency (binary number) to BCD
      Set PORTE to receive data

      Send hundreds
      Send tens
      Send units

      Set PORTE to receive commands
      Move cursor to 9th position 2nd line of the LCD
      Convert amplitude (binary number) to BCD
      Set PORTE to receive data

      Send hundreds
      Send tens
      Send units
END
```

NOTE: hundreds, tens and units are saved in a user define register during the conversion process

The flow chart demonstrates the Binary to BCD conversion

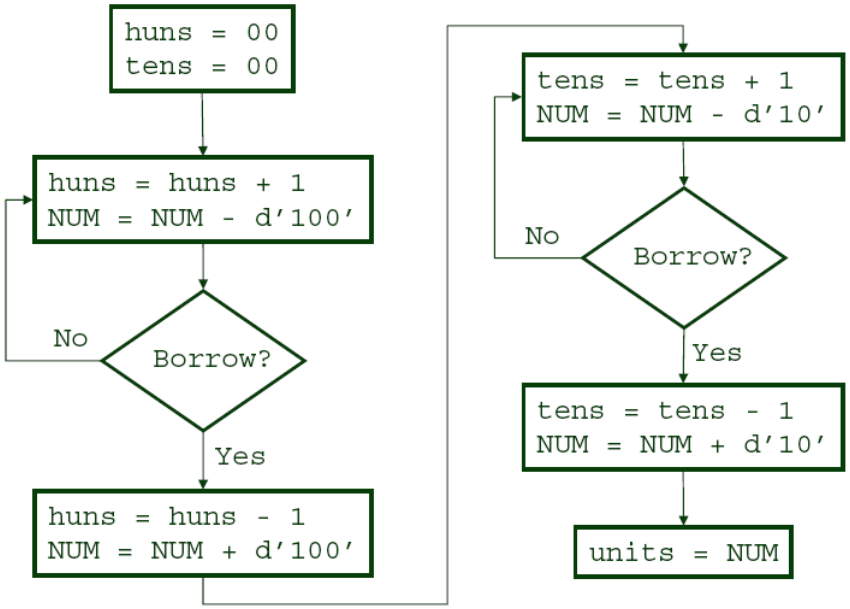


Figure 2.0: Binary to BCD (provided in Lecture PIC3)

The following pseudo code demonstrates data being sent to the LCD in 8 bit mode

```
BEGIN
      Clear enable pin of PORTE
      Delay 1ms
      Write data to PORTD (LCD data pins)
      Enable pulse pin on PORTE
      Clear enable pin of PORTE
END
```

This particular pseudo code demonstrates data being sent to Hyper Terminal

```
BEGIN
      Move value from W register to transmit register (TXREG)
      Check
            Check if GO/DONE set
            If not, go to Check
END
```

Last but not least the pseudo code for updating Hyper Terminal

```
BEGIN
      Move cursor to top-left corner of Hyper Terminal
      Move cursor down 18 times

      Temp = old scaled frequency + 1
      Move forward
            Move cursor forward
            Temp = Temp - 1
            If Temp not = 0 then
            Go to Move forward

      Temp = old scaled amplitude + 1
      Erase char
            Erase value
            Move cursor back once (backspace)
            Move cursor up
            Temp = Temp – 1
            If Temp not = 0 then
            Go to erase char


      Move cursor to top-left corner
      Move cursor down 18 times

      Temp = scaled frequency + 1
      Move forward
            Move cursor forward
            Temp = Temp – 1
            If Temp not = 0 then
            Go to Move forward

      Temp = scaled amplitude + 1
      Add star
            Add star to represent the amplitude value
            Move cursor back once (backspace)
            Move cursor up
            Temp = Temp – 1
            If Temp not = 0 then
            Go to add star
END
```

USART

The Universal Synchronous Asynchronous Receiver Transmitter also known as a Serial Communications Interface (SCI).

The following settings are outlined in the specification sheet and are to be used

- 57600bps
- 8bit
- No parity
- No flow control
- 1 stop bit

The USART is configured with the following settings:

PORTC is cleared to initialize it and TRISC<TX> is set to enable USART asynchronous transmission.

Bit BRGH (TXSTA<2>) is set to enable transmission.

The SPBRG register controls the period of free running 8-bit timer. The register is set to decimal 20. The value is taken from Table 10-4 on page 100 of the PIC16F87X Data sheet, using BRGH = 1, F_{osc} = 20 MHz and 57.6K Baud rate.

Bit SYNCH (TXSTA<4>) is cleared to enable asynchronous mode.

Bit TXEN (TXSTA<5>) is set to enable transmission.

Bit SPEN (RCSTA <7>) is set to enable the serial port.

To transmit data over the serial port, TXREG is loaded with the data. However there needs to be a delay of at least 20µs between each sent character to transfer the data correctly.

This is calculated using the following formula:

Baud rate: 57.6K

1 / 57.6K = 17.36µs

However in my design, I poll TXIF (PIR1<4>) to see if it has been set.
Once it has been set the next character can be sent.

Analog to Digital Converter

The built-in module in the PIC16F877 Micro controller converts the analogue data received from the 2 potentiometers to digital.

The potentiometers are connected to pins `AN2` (V_{ref-}) and `AN3` (V_{ref+}) as specified in the Analogue Peripheral Module Schematics.

The voltage references V_{ref+} and V_{ref-} for the A/D converter need to be set to V_{dd} and V_{ss}.

Pins `AN2` and `AN3` are analogue. Pins `RE2 – RE0` need to be digital to interface with the LCD.

By looking at the `PCFG3:PCFG0 (ADCON1 <3:0>)` A/D Port Configuration bits on page 112 of the PIC16F87X Data sheet, it can be seen that there is only one option available. That is, bits `PCFG3:PCFG0` are set to `0010`.

So `PORTE` is set to digital and `PORTA` is set to analogue.

Furthermore, the A/D module produces a 10 bit output, bit `ADFM (ADCON1<7>)` is cleared to left-justify to measure the 8 most significant bits from the A/D module high register – `ADRESH`.

For the A/D converter to meet its specified accuracy, the charge holding capacitor must be allowed to fully charge to the input channel voltage. The Acquisition time is calculated using the following formula.

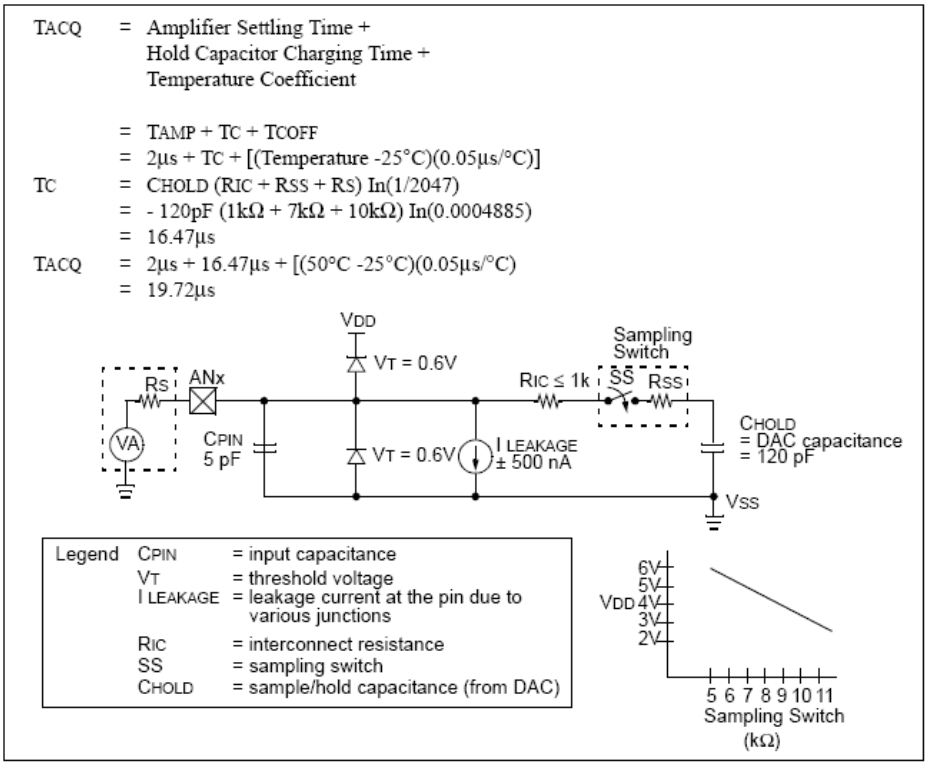


Figure 3.0: Acquisition time

It can be seen that at least 20µs need to pass before trying to do the conversion. In my design I use 1ms to re-use the delay sub routine that is used throughout the program. Also the human eye can't tell the difference between 20µs and 1ms nor is this time of great importance so the cause is justified.

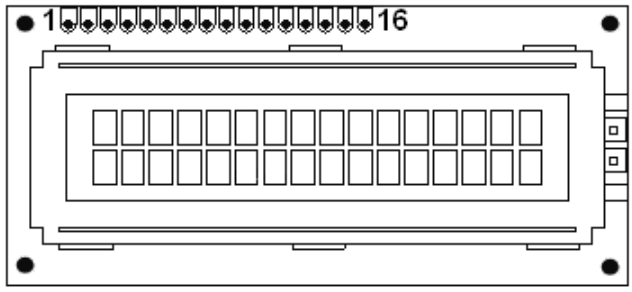
Under the section *Operation of the Assembly Code*, the process to convert analogue signal to digital is presented as pseudo code.

In this particular design the bit `GO/DONE (ADCON0<2>)` is polled to check if conversion is complete. When it clears, the conversion is complete and the value in `ADRESH` is stored in a user defined register.

Liquid Crystal Display (LCD)

The LCD displays the student number, frequency and amplitude, as shown in *Figure 1.0*. The frequency and amplitude range is from 0 to 255. The analogue peripheral module equivalent values are 0V to 5V.

The figure shows the address of each block on the LCD.



Mechanical Drawing.

Characters	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
First line	00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH
Second line	40H	41H	42H	43H	44H	45H	46H	47H	48H	49H	4AH	4BH	4CH	4DH	4EH	4FH
Characters	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figure 1.1: Display Data Ram Access Map

The following table shows how the LCD is connected to the PIC.

LCD PIN	Connection	Description
16		
15		
	PORT D	All hardwired
14	RD7/PSP7	LCD Data Line 7 hardwired
13	RD6/PSP6	LCD Data Line 6 hardwired
12	RD5/PSP5	LCD Data Line 5 hardwired
11	RD4/PSP4	LCD Data Line 4 hardwired
10	RD3/PSP3	LCD Data Line 3 hardwired
9	RD2/PSP2	LCD Data Line 2 hardwired
8	RD1/PSP1	LCD Data Line 1 hardwired
7	RD0/PSP0	LCD Data Line 0 hardwired
	PORTE	All hardwired
6	RE2/CS/AN7	LCD E – Enable hardwired
5	RE1/WR/AN6	LCD R/W – Read / Write hardwired
4	RE0/RD/AN5	LCD RS – Register Select hardwired
3	Vo	Voltage For LCD (acts as contrast)
2	VCC	Supply Voltage For Logic
1	GND	Ground

Table 1.0: LCD Connections

The LCD is set as an 8 bit interface 2 lines, 5x7 character format.

Display shifting is disabled and cursor shifting is used instead.

Display is set to ON.

Once the frequency or amplitude is converted to BCD it's ready to be transferred to the LCD.

To transfer the number 255; 2 needs to be sent first, then 5 followed by 5.

However before transferring the value it needs to be changed into the correct ASCII char to represent the actual value. This is done by adding H '30' to the value.

So if we have a value of 2 we add H '30' to get H '32' which represents decimal 2.

It's also worth mentioning the following:

LCD E – Enable Line is used to initiate the actual transfer of commands or character data between the module and data lines.

LCD R/W – Read/Write Line is pulled low in order to write commands or data to the module.

LCD RS – Register Select Line. When cleared (low), data bytes transferred to the display are treated as commands. By setting (high) RS, character data is transferred to the LCD.

Troubleshooting

During the program development minor problems were encountered but did not require the use of a software based debugger.

One such problem was after I got my LCD working I wrote the code for Hyper Terminal and tested it separately which worked fine. But after putting the two together the LCD would not update the frequency and amplitude smoothly, it was very slow compared to how it functions on its own.

My first assumption was that I must be doing something wrong when checking if the frequency or amplitude changed. After close inspection I saw that it was fine. It then hit me that it's the delay sub routine used after each character is sent to Hyper Terminal, I was using a 1ms delay. A lot of chars are sent to the hyper terminal so the ms's add up so by the time I get to update the LCD time passes.

The issue was solved by not using a delay but by checking the bit `TXIF (PIR1<4>)`. It's polled until its set; once set another character can be sent.

Another problem was a silly typing mistake, which I found later out. This also happened when I combined the Hyper Terminal together with the LCD code. The frequency was showing up fine but the amplitude would only show garbage on the LCD. Common sense said that I must be corrupting my user defined register that holds the value for amplitude. So I started disabling non critical sub routines and found out that the sub routine that scales the data was causing the problem.

After spending half an hour trying to figure out what the problem was and in the process resisting not to throw the assignment in garbage I spotted the mistake. I have a loop which is called Scale_Amplitude and a user defined register Scaled_Amplitude. The silly spelling mistake was during the step when I clear bit 7. I was clearing Scale_Amplitude instead of Scale**d**_Amplitude.

A 3rd mistake I made was during the update to hyper terminal. Everything worked fine except it wouldn't erase. The fact was I was setting my old scaled amplitude to the current scaled amplitude without realising it, so each time I wanted to erase data it would erase exactly the number of time the new amplitude is and write the new values, making it look like it wasn't erasing. This mistake was easily fixed by removing the code that was setting the old scaled frequency to current scaled frequency, which was used earlier only for testing purposes.

I have been programming for many years now and even though assembly code is new to me, I have learned to debug a lot of coding problems without the need of a software based debugger. This assignment is a clear example of that. The specification sheet does not specifically ask to show evidence of debugging using the MPLAB debugger so I believe my way of debugging is just as good.

Assembly Code

```
;-------------------------------------------------------------------------------------------------------------------
; author                : Beno Gorancic
; last modified         : 05/06/2006
;
; most of the code used in this program has been adopted from code made available
; to students on UTS Online under the subject 48441 - Introductory Digital Systems
;-------------------------------------------------------------------------------------------------------------------


            LIST      P=16F877, F=INHX8M
            #include <P16F877.inc>


;-------------------------------------------------------------------------------------------------------------------
; CONFIGURATION WORD
;-------------------------------------------------------------------------------------------------------------------
            ; code protect off, debug off, program memory write protection off,
            ; data EE memory code protection off, low voltage programming off,
            ; brown out detection on, watch dog timer off,
            ; high speed(>4MHz) xtal program write on
            __config _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_ON &
_WDT_OFF & _HS_OSC & _PWRTE_ON


;-------------------------------------------------------------------------------------------------------------------
; MEMORY EQUATES
;-------------------------------------------------------------------------------------------------------------------
Bank0Ram            equ         H'20'


;-------------------------------------------------------------------------------------------------------------------
; MEMORY ALLOCATIONS
;-------------------------------------------------------------------------------------------------------------------
            cblock      Bank0Ram                    ; beginning of access RAM
                        Temp
                        LCD_Temp_Buffer
                        Axis_Temp_Value
                        DelayA
                        DelayB
                        DelayC
                        huns
                        tens
                        units
                        Frequency
                        Amplitude
                        Old_Frequency
                        Old_Amplitude
                        Scaled_Frequency
                        Scaled_Amplitude
                        Old_Scaled_Frequency
                        Old_Scaled_Amplitude
            endc


;-------------------------------------------------------------------------------------------------------------------
; VECTORS
;-------------------------------------------------------------------------------------------------------------------
            org         0x0000              ; reset vector
            goto        MainLine

            org         0x0004              ; interrupt vector
            goto        IntService
```

```
;-------------------------------------------------------------------------------------------------------------------
; IntService - INTERRUPT SERVICE ROUTINE
;-------------------------------------------------------------------------------------------------------------------
IntService          ; empty routine, just in case an interrupt occurs
        retfie      ; return from interrupt


;-------------------------------------------------------------------------------------------------------------------
; MainLine - MAIN PROGRAM
;-------------------------------------------------------------------------------------------------------------------
MainLine
        call        Initialize
        call        Startup_LCD
        call        Startup_HyperTerminal


;-------------------------------------------------------------------------------------------------------------------
; MainLoop - MAIN PROGRAM LOOP
;-------------------------------------------------------------------------------------------------------------------
MainLoop
        call        Analog_to_Digital
        call        Scale_Data

        movfw       Frequency                   ; set value in W register to Frequency
        subwf       Old_Frequency,W             ; subtract Frequency from Old_Frequency; store in W
        btfss       STATUS,Z                    ; if W = 0, STATUS = 1...
        call        Send_to_LCD                 ; then skip this operation

        movfw       Amplitude
        subwf       Old_Amplitude,W
        btfss       STATUS,Z
        call        Send_to_LCD

        movfw       Scaled_Frequency
        subwf       Old_Scaled_Frequency,W
        btfss       STATUS,Z
        call        Send_to_HyperTerminal

        movfw       Scaled_Amplitude
        subwf       Old_Scaled_Amplitude,W
        btfss       STATUS,Z
        call        Send_to_HyperTerminal

        movfw       Scaled_Frequency
        movwf       Old_Scaled_Frequency
        movfw       Scaled_Amplitude
        movwf       Old_Scaled_Amplitude

        goto        MainLoop


;-------------------------------------------------------------------------------------------------------------------
; Initialize - INITIALIZATION ROUTINE
;-------------------------------------------------------------------------------------------------------------------
Initialize
        bcf         STATUS,RP0
        bcf         STATUS,RP1              ; bank 0 selected
        clrf        PIR1
        clrf        PIR2                    ; clear interrupt flags registers
        clrf        PORTA
        clrf        PORTB
        clrf        PORTC
        clrf        PORTD
        clrf        PORTE                   ; clear ports to output all 0's
        clrf        INTCON                  ; ensure all interrupts are turned off
        bsf         RCSTA,SPEN              ; serial port enabled
        bsf         STATUS,RP0              ; set ram bank 1
        clrf        TRISA
        clrf        TRISB
        clrf        TRISC
```

```
        clrf    TRISD
        clrf    TRISE           ; pins on ports A to E are outputs
        clrf    PIE1            ; disable interrupts
        movlw   H'3F'
        movwf   TRISA           ; port A pins all inputs
        bsf     TRISC,6         ; enable USART asynchronous transmit
        movlw   H'02'
        movwf   ADCON1          ; left justified, port E digital, port A analogue, Vref+ = Vdd, Vref- = Vss
        movlw   H'14'
        movwf   SPBRG           ; Fosc = 20Mhz, BRGH = 1, baud rate = 57.6K,  we get: SPRGH value of D'20'
        bsf     TXSTA,TXEN      ; transmission enabled
        bcf     TXSTA,SYNC      ; asynchronous mode
        bsf     TXSTA,BRGH      ; high speed baud rate
        bcf     STATUS,RP0      ; set ram bank 0

        return


;-----------------------------------------------------------------------------------------------------------------
; Analog_to_Digital - ANALOG TO DIGITAL CONVERTER
;-----------------------------------------------------------------------------------------------------------------
Analog_to_Digital
        movlw   H'91'
        movwf   ADCON0              ; Fosc/32, channel 2, GO/DONE 0, A/D converter on
        movlw   .1
        call    Delay               ; delays 1ms before conversion
        bsf     ADCON0,GO_DONE      ; ADC routine

Wait_Frequency
        btfsc   ADCON0,GO_DONE      ; check if conversion is over
        goto    Wait_Frequency      ; if not goto Wait_Frequency
        movfw   ADRESH              ; move high byte to W
        movwf   Frequency           ; move high byte to Frequency
        movlw   H'99'
        movwf   ADCON0              ; Fosc/32, channel 3, GO/DONE 0, A/D converter on
        movlw   .1
        call    Delay               ; delays 1ms before conversion
        bsf     ADCON0,GO_DONE      ; ADC routine

Wait_Amplitude
        btfsc   ADCON0,GO_DONE      ; check if conversion is over
        goto    Wait_Amplitude      ; if not goto Wait_Amplitude
        movfw   ADRESH              ; move high byte to W
        movwf   Amplitude           ; move high byte to Amplitude

        return


;-----------------------------------------------------------------------------------------------------------------
; Binary_to_BCD - CONVERTS A BINARY NUMBER TO A BINARY-CODED-DECIMAL
;-----------------------------------------------------------------------------------------------------------------
Binary_to_BCD
        clrf    huns            ; clear hundreds
        clrf    tens            ; clear tens
        clrf    units           ; clear units

; do subtract hundred and counting while no borrow is generated
Loop_h
        incf    huns,F          ; record one hun subtracted
        addlw   -D'100'         ; subtract decimal hundred
        btfsc   STATUS,C        ; if a borrow (C==0) then exit Loop_h
        goto    Loop_h
        decf    huns,F          ; correct for one subtract
        addlw   D'100'          ; add 100 to residue

; do subtact ten and counting while no borrow is generated
Loop_t
        incf    tens, F         ; record one ten subtracted
        addlw   -D'10'          ; subtract decimal ten
        btfsc   STATUS, C       ; if a borrow (C==0) then exit Loop_t
```

```
        goto    Loop_t
        decf    tens, F         ; correct for one subtract
        addlw   D'10'           ; add 10 to residue
        movwf   units           ; save the remainder in units

        movlw   H'30'           ; hex ascii code for number 0
        addwf   huns,F          ; add H'30' to get correct ascii value
        addwf   tens,F          ; add H'30' to get correct ascii value
        addwf   units,F         ; add H'30' to get correct ascii value

        return


;-----------------------------------------------------------------------------------------------------------
; Startup_LCD - LCD STARTUP ROUTINE
;-----------------------------------------------------------------------------------------------------------
Startup_LCD
        bcf     STATUS,RP0
        bcf     STATUS,RP1              ; set ram bank 0

        ; the following sequence is the recommended power up sequence for an LCD as stated by the chipset
manufacturer
        bcf     PORTE,0                 ; instruction register select
        movlw   .25
        call    Delay                   ; delay for LCD PowerUp
        movlw   .30
        call    LCDWrite8               ; set LCD to 8 bits
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   .30
        call    LCDWrite8               ; set LCD to 8 bits
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   .30
        call    LCDWrite8               ; set LCD to 8 bits
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   .56
        call    LCDWrite8               ; 8 bit interface 2 line, 5x7 character format
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   .56
        call    LCDWrite8               ; 8 bit interface 2 line, 5x7 character format
        movlw   .5
        call    Delay                   ; delay for LCD

        ; write user specific configuration data
        movlw   H'6'
        call    LCDWrite8               ; display shifting disabled, use cursor shifting instead
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   H'1'
        call    LCDWrite8               ; clear all DDRAM and set cursor to pos 1
        movlw   .5
        call    Delay                   ; delay for LCD
        movlw   H'C'
        call    LCDWrite8               ; display ON
        bsf     PORTE,0                 ; set to data register

        ; write data to the LCD "10321747"
        movlw   H'31'
        call    LCDWrite8
        movlw   H'30'
        call    LCDWrite8
        movlw   H'33'
        call    LCDWrite8
        movlw   H'32'
        call    LCDWrite8
        movlw   H'31'
        call    LCDWrite8
```

```
        movlw   H'37'
        call    LCDWrite8
        movlw   H'34'
        call    LCDWrite8
        movlw   H'37'
        call    LCDWrite8

        ; move cursor to 1st position 2nd line of the LCD
        bcf     PORTE,0                 ; set instruction register
        movlw   H'C0'
        call    LCDWrite8
        bsf     PORTE,0                 ; set data register

        ; write data to the LCD "F="
        movlw   'F'
        call    LCDWrite8
        movlw   '='
        call    LCDWrite8

        ; move cursor to 7th position 2nd line of the LCD
        bcf     PORTE,0                 ; set instruction register
        movlw   H'C6'
        call    LCDWrite8
        bsf     PORTE,0                 ; set data register

        ; write data to the LCD "A="
        movlw   'A'
        call    LCDWrite8
        movlw   '='
        call    LCDWrite8

        return


;-----------------------------------------------------------------------------------------------------------------
; Send_to_LCD - SENDS DATA TO THE LCD
;-----------------------------------------------------------------------------------------------------------------
Send_to_LCD
        bcf     PORTE,0                 ; set instruction register
        movlw   H'C2'                   ; move cursor to 3rd position 2nd line of the LCD
        call    LCDWrite8

        movfw   Frequency
        call    Binary_to_BCD

        bsf     PORTE,0                 ; set data register
        movfw   huns
        call    LCDWrite8               ; send hundreds
        movfw   tens
        call    LCDWrite8               ; send tens
        movfw   units
        call    LCDWrite8               ; send units

        bcf     PORTE,0                 ; set instruction register
        movlw   H'C8'                   ; move cursor to 9th position 2nd line of the LCD
        call    LCDWrite8

        movfw   Amplitude
        call    Binary_to_BCD

        bsf     PORTE,0                 ; set data register
        movfw   huns
        call    LCDWrite8               ; send hundreds
        movfw   tens
        call    LCDWrite8               ; send tens
        movfw   units
        call    LCDWrite8               ; send units

        return
```

```
;-----------------------------------------------------------------------------------------------------------------
; LCDWrite8 - WRITE TO THE LCD IN 8 BIT MODE
;-----------------------------------------------------------------------------------------------------------------
LCDWrite8
        movwf   LCD_Temp_Buffer         ; temporally store data to be written to the LCD
        bcf     PORTE,2                 ; clear the enable pin
        movlw   .1
        call    Delay                   ; 1ms delay for LCD timing
        movf    LCD_Temp_Buffer,W
        movwf   PORTD                   ; write stored data to PORTD ie LCD data pins
        bsf     PORTE,2                 ; pulse enable pin
        nop
        bcf     PORTE,2
        return


;-----------------------------------------------------------------------------------------------------------------
; Delay - DELAYS A SET AMOUNT OF ms; COUNTS CYCLES BASED ON 20MHZ CLOCK
;-----------------------------------------------------------------------------------------------------------------
Delay
        movwf   DelayA
OuterLoop
        movlw   .29
        movwf   DelayB
MiddleLoop
        movlw   .42
        movwf   DelayC
InnerLoop
        nop
        decfsz  DelayC,F
        goto    InnerLoop
        decfsz  DelayB,F
        goto    MiddleLoop
        decfsz  DelayA,F
        goto    OuterLoop
        return


;-----------------------------------------------------------------------------------------------------------------
; Scale_Data - SCALES DOWN THE FREQUENCY AND AMPLITUDE; USED FOR HYPER TERMINAL
;-----------------------------------------------------------------------------------------------------------------
Scale_Data
        movfw   Frequency               ; set value in W register to Frequency
        movwf   Scaled_Frequency        ; move value in W register to Scaled_Frequency
        movlw   H'02'
        movwf   Temp                    ; moves H'02' from W register to Temp

; changes the 8 bit frequency to 6 bit
Scale_Frequency
        rrf     Scaled_Frequency        ; right shift Scaled_Frequency
        bcf     Scaled_Frequency,7      ; clear bit 7
        decfsz  Temp                    ; decrements Temp and checks if 0...
        goto    Scale_Frequency         ; if it's 0 then skip goto

        movfw   Amplitude               ; set value in W register to Amplitude
        movwf   Scaled_Amplitude        ; move value in W register to Scaled_Amplitude
        movlw   H'04'
        movwf   Temp                    ; moves H'04' from W register to Temp

; changes the 8 bit amplitude to 4 bit
Scale_Amplitude
        rrf     Scaled_Amplitude        ; right shift Scaled_Amplitude
        bcf     Scaled_Amplitude,7      ; clear bit 7
        decfsz  Temp                    ; decrements Temp and checks if 0...
        goto    Scale_Amplitude         ; if it's 0 then skip goto

        return
```

```
;-------------------------------------------------------------------------------------------------------------------
; Startup_HyperTerminal - HYPER TERMINAL STARTUP ROUTINE
;-------------------------------------------------------------------------------------------------------------------
Startup_HyperTerminal
        bcf STATUS,RP0
        bcf STATUS,RP1                  ; select bank 0

        movlw   H'41'                   ; moves ascii char to W reg;    A
        call    Send_Char
        movlw   H'33'                   ;                               3
        call    Send_Char
        movlw   H'20'                   ;                               space
        call    Send_Char
        movlw   H'32'                   ;                               2
        call    Send_Char
        movlw   H'30'                   ;                               0
        call    Send_Char
        movlw   H'30'                   ;                               0
        call    Send_Char
        movlw   H'36'                   ;                               6
        call    Send_Char
        movlw   H'0D'                   ;                               carriage return
        call    Send_Char
        movlw   H'0A'                   ;                               new line
        call    Send_Char
        movlw   H'48'                   ;                               H
        call    Send_Char
        movlw   H'28'                   ;                               (
        call    Send_Char
        movlw   H'73'                   ;                               s
        call    Send_Char
        movlw   H'29'                   ;                               )
        call    Send_Char
        movlw   H'0D'                   ;                               carriage return
        call    Send_Char
        movlw   H'0A'                   ;                               new line
        call    Send_Char
        movlw   H'5E'                   ;                               ^
        call    Send_Char
        movlw   H'0D'                   ;                               carriage return
        call    Send_Char
        movlw   H'0A'                   ;                               new line
        call    Send_Char

        movlw   D'16'
        movwf   Axis_Temp_Value

; insert '|' 16 times
Vertical_Axis
        movlw   H'7C'                   ;                               |
        call    Send_Char
        movlw   H'0D'                   ;                               carriage return
        call    Send_Char
        movlw   H'0A'                   ;                               new line
        call    Send_Char
        decfsz  Axis_Temp_Value         ; decrement Axis_Temp_Value...
        goto    Vertical_Axis           ; skip goto if 0

        movlw   D'65'
        movwf   Axis_Temp_Value

; insert '-' 65 times
Horizontal_Axis
        movlw   H'2D'                   ;                               -
        call    Send_Char
        decfsz  Axis_Temp_Value         ; decrement Axis_Temp_Value...
        goto    Horizontal_Axis         ; skip goto if 0

        movlw   H'3E'                   ;                               >
        call    Send_Char
```

```
        movlw   H'20'                   ;                       space
        call    Send_Char
        movlw   H'66'                   ;                       f
        call    Send_Char

        return


;-------------------------------------------------------------------------------------------------------------
; Send_Char - SENDS CHARACTER TO HYPER TERMINAL
;-------------------------------------------------------------------------------------------------------------
Send_Char
        movwf   TXREG                   ; move value from W register to transmit register
        nop
Check
        btfss   PIR1,TXIF               ; checks if GO/DONE is set
        goto    Check                   ; if not goto Check

        return


;-------------------------------------------------------------------------------------------------------------
; Send_to_HyperTerminal - SENDS DATA TO HYPER TERMINAL
;-------------------------------------------------------------------------------------------------------------
Send_to_HyperTerminal

; move cursor home (top-left corner of Hyper Terminal)
        movlw   H'1B'                   ; moves ascii char to W reg;    ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'48'                   ;                       H
        call    Send_Char

; move cursor down
        movlw   H'1B'                   ;                       ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'31'                   ;                       1
        call    Send_Char
        movlw   H'38'                   ;                       8
        call    Send_Char
        movlw   H'42'                   ;                       B
        call    Send_Char

        movfw   Old_Scaled_Frequency    ; set value in W register to Old_Scaled_Frequency
        movwf   Temp                    ; move value in W register to Temp
        movlw   .1
        addwf   Temp                    ; add value in W register to Temp

; move cursor forward
Move_Forward
        movlw   H'1B'                   ;                       ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'43'                   ;                       C
        call    Send_Char

        decfsz  Temp                    ; decrement Temp...
        goto    Move_Forward            ; if it's 0 then skip goto

        movfw   Old_Scaled_Amplitude
        movwf   Temp
        movlw   .1
        addwf   Temp
```

```
; erase current value
Erase_Value
        movlw   ' '                     ;                       ' '
        call    Send_Char
        movlw   H'08'                   ;                       backspace
        call    Send_Char

; move cursor up
        movlw   H'1B'                   ;                       ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'41'                   ;                       A
        call    Send_Char

        decfsz  Temp
        goto    Erase_Value

; move cursor home
        movlw   H'1B'                   ; moves ascii char to W reg;    ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'48'                   ;                       H
        call    Send_Char

; move cursor down
        movlw   H'1B'                   ;                       ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'31'                   ;                       1
        call    Send_Char
        movlw   H'38'                   ;                       8
        call    Send_Char
        movlw   H'42'                   ;                       B
        call    Send_Char

        movfw   Scaled_Frequency
        movwf   Temp
        movlw   .1
        addwf   Temp

; move cursor forward
Move_Forward_New
        movlw   H'1B'                   ;                       ESC
        call    Send_Char
        movlw   H'5B'                   ;                       [
        call    Send_Char
        movlw   H'43'                   ;                       C
        call    Send_Char
        decfsz  Temp
        goto    Move_Forward_New

        movfw   Scaled_Amplitude
        movwf   Temp
        movlw   .1
        addwf   Temp

; send '*' char
Add_Star
        movlw   H'2A'                   ;                       *
        call    Send_Char
        movlw   H'08'                   ;                       backspace
        call    Send_Char
```

```
; move cursor up
Move_Up
        movlw   H'1B'           ;               ESC
        call    Send_Char
        movlw   H'5B'           ;               [
        call    Send_Char
        movlw   H'41'           ;               A
        call    Send_Char

        decfsz  Temp
        goto    Add_Star

        return

        END
```

References

ANSI Escape Sequence
http://adm.lacitec.on.ca/~ymicha/mcours/micro1/escape.html, last accessed 6/6/06

ASCII Table
http://www.lookuptables.com, last accessed 06/06/2006

Figure 1.0: LCD
"LCD Connections and Description" PDF file from UTS Online under subject: 48441

Figure 2.0: Binary to BCD (provided in Lecture PIC3)
"Lecture 10 Capture/Compare/PWM modules" 48441 – Introductory Digital Systems, University of Technology, Sydney

Figure 3.0: Acquisition time
"Microchip PIC16F87X Data Sheet", page 114

Figure 1.1: Display Data Ram Access Map
"LCD Connections and Description" from UTS Online under subject: 48441 - Introductory Digital Systems

Table 1.0: LCD Connections
"LCD Connections and Description" from UTS Online under subject: 48441 – Introductory Digital Systems

"LCD Connections and Description" PDF file from UTS Online made available to students under the subject 48441 - Introductory Digital Systems

"*TRULY* LCD Module Product Specifications" PDF file

"Microchip PIC16F87X Data Sheet" PDF file

Most of the code used in the program has been adopted from code made available to students on UTS Online under the subject 48441 - Introductory Digital Systems. Some of the more extensively used assembly files:

- LCDExample.asm
- adconv.asm
- USART1.asm

Also some code has been adopted from code provided in the lecture notes.