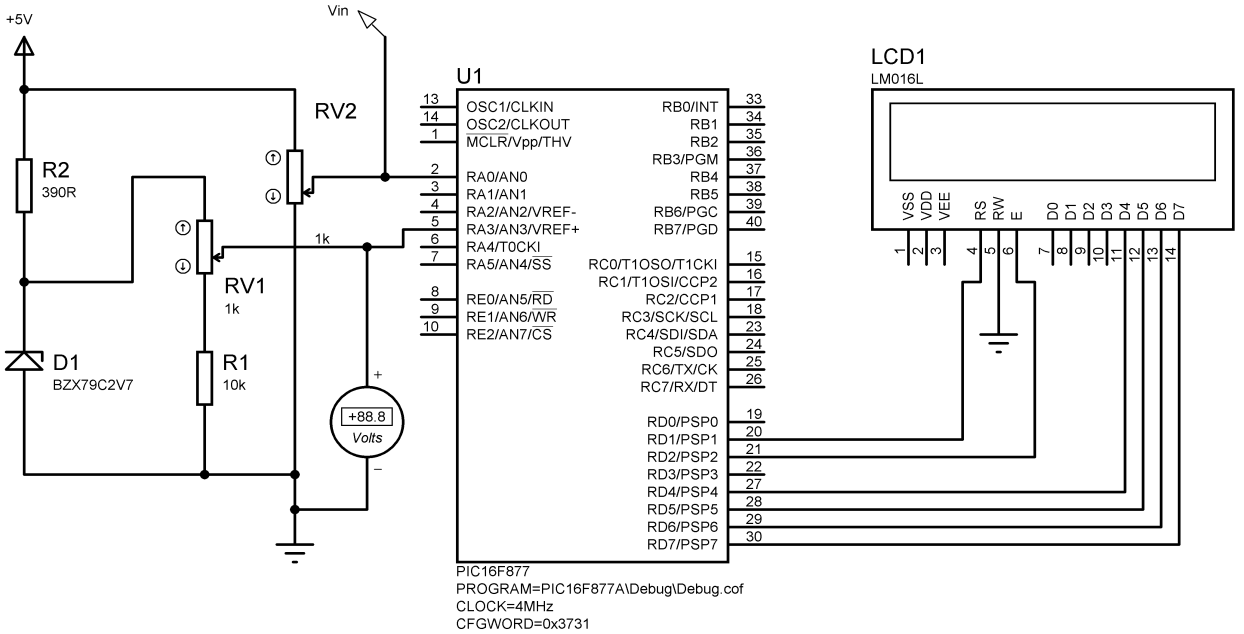
Interfacing PIC Microcontrollers

ADC8BIT2 Schematic



- This application demonstrates analogue input sampling
- A manually adjusted test voltage 0-5V is provided at AN0 input
- A reference voltage of 2.56V is provided at VREF+, setting the conversion range and resolution (10mV per bit)
- The internal analogue to digital converter produces an 8-bit code representing the voltage
- This is converted to 3 digit BCD and each digit converted to ASCII for output to the LCD

ADC8BIT2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:                ADC8BIT2
;       Devised by:             MPB
;       Date:                   14-01-13
;       Status:                 Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Demonstrates simple analogue input
;       using an external reference voltage of 2.56V
;       The 8-bit result is converted to BCD for display
;       as a voltage using the standard LCD routines.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A
;       Clock = XT 4MHz, standard fuse settings
        __CONFIG 0x3731

;       LABEL EQUATES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        #INCLUDE "P16F877A.INC"      ; standard labels

; GPR 70 - 75 allocated to included LCD display routine

count   EQU     30      ; Counter for ADC setup delay
ADbin   EQU     31      ; Binary input value
huns    EQU     32      ; Hundreds digit in decimal value
tens    EQU     33      ; Tens digit in decimal value
ones    EQU     34      ; Ones digit in decimal value

; PROGRAM BEGINS ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Default start address
        NOP                     ; required for ICD mode

; Port & display setup.....................................

        BANKSEL TRISC           ; Select bank 1
        CLRF    TRISD           ; Display port is output
        MOVLW   B'00000011'     ; Analogue input setup code
        MOVWF   ADCON1          ; Left justify result,
                                ; Port A = analogue inputs

        BANKSEL PORTC           ; Select bank 0
        CLRF    PORTD           ; Clear display outputs
        MOVLW   B'01000001'     ; Analogue input setup code
        MOVWF   ADCON0          ; f/8, RA0, done, enable

        CALL    inid            ; Initialise the display

; MAIN LOOP ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

start   CALL    getADC          ; read input
        CALL    condec          ; convert to decimal
        CALL    putLCD          ; display input
        GOTO    start           ; jump to main loop

; SUBROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Read ADC input and store ...............................

getADC  BSF     ADCON0,GO       ; start ADC..
wait    BTFSC   ADCON0,GO       ; ..and wait for finish
        GOTO    wait
        MOVF    ADRESH,W        ; store result high byte
        RETURN

; Convert input to decimal ...............................

condec  MOVWF   ADbin           ; get ADC result
        CLRF    huns            ; zero hundreds digit
        CLRF    tens            ; zero tens digit
        CLRF    ones            ; zero ones digit

; Calclulate hundreds.....................................

        BSF     STATUS,C        ; set carry for subtract
        MOVLW   D'100'          ; load 100
sub1    SUBWF   ADbin           ; and subtract from result
        INCF    huns            ; count number of loops
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub1            ; no, carry on

        ADDWF   ADbin           ; yes, add 100 back on
        DECF    huns            ; and correct loop count

; Calculate tens digit....................................

        BSF     STATUS,C        ; repeat process for tens
        MOVLW   D'10'           ; load 10
sub2    SUBWF   ADbin           ; and subtract from result
        INCF    tens            ; count number of loops
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub2            ; no, carry on

        ADDWF   ADbin           ; yes, add 100 back on
        DECF    tens            ; and correct loop count
        MOVF    ADbin,W         ; load remainder
        MOVWF   ones            ; and store as ones digit

        RETURN                  ; done
```

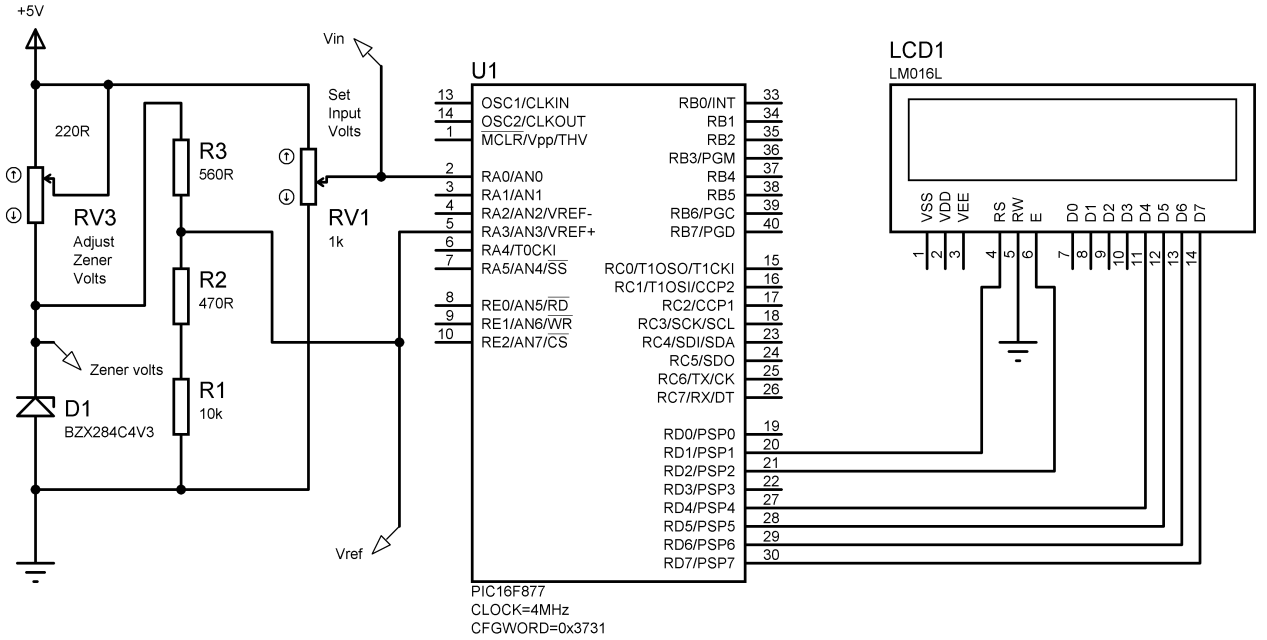
ADC8BIT2 Source Code

```
; Output to display.....................................

putLCD  BCF     Select,RS       ; set display command mode
        MOVLW   080             ; code to home cursor
        CALL    send            ; output it to display
        BSF     Select,RS       ; and restore data mode

; Convert digits to ASCII and display.....................

        MOVLW   030             ; load ASCII offset
        ADDWF   huns            ; convert hundreds to ASCII
        ADDWF   tens            ; convert tens to ASCII
        ADDWF   ones            ; convert ones to ASCII

        MOVF    huns,W          ; load hundreds code
        CALL    send            ; and send to display
        MOVLW   '.'             ; load point code
        CALL    send            ; and output
        MOVF    tens,W          ; load tens code
        CALL    send            ; and output
        MOVF    ones,W          ; load ones code
        CALL    send            ; and output
        MOVLW   ' '             ; load space code
        CALL    send            ; and output
        MOVLW   'V'             ; load volts code
        CALL    send            ; and output
        MOVLW   'o'             ; load volts code
        CALL    send            ; and output
        MOVLW   'l'             ; load volts code
        CALL    send            ; and output
        MOVLW   't'             ; load volts code
        CALL    send            ; and output
        MOVLW   's'             ; load volts code
        CALL    send            ; and output

        RETURN                  ; done
```

```
; INCLUDED ROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Include LCD driver routines
;
        #INCLUDE "LCD.INC"
;       Contains routines:
;       inid:   Initialises display
;       onems:  1 ms delay
;       xms:    X ms delay
;               Receives X in W
;       send:   Sends a character to display
;               Receives: Control code in W (Select,RS=0)
;                         ASCII character code in W (RS=1)

        END     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Interfacing PIC Microcontrollers

ADC10BIT2 Schematic



- This application demonstrates high resolution 10-bit ADC input handling
- The reference voltage is 4.096V, giving a conversion factor of 4mV per bit
- The ADC binary output of 0 – 1024 is converted into 4 digit BCD
- This is converted to ASCII and sent to the LCD
- The measurement precision is better than 0.1%

ADC10BIT2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:              ADC10BIT2
;       Devised by:           MPB
;       Date:                 14-01-13
;       Status:               Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Demonstrates 10-bit voltage measurement
;       using an external reference voltage of 4.096V,
;       giving 4mV per bit, and an resolution of 0.1%.
;       The result is converted to BCD for display
;       as a voltage using the standard LCD routines.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877
;       Clock = XT 4MHz, standard fuse settings
        __CONFIG 0x3731

;       LABEL EQUATES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        INCLUDE "P16F877A.INC"
        ; standard register labels

;-----------------------------------------------------------
; User register labels
;-----------------------------------------------------------
; GPR 20 - 2F allocated to included LCD display routine

count   EQU     30      ; Counter for ADC setup delay
ADhi    EQU     31      ; Binary input high byte
ADlo    EQU     32      ; Binary input low byte
thos    EQU     33      ; Thousands digit in decimal
huns    EQU     34      ; Hundreds digit in decimal value
tens    EQU     35      ; Tens digit in decimal value
ones    EQU     36      ; Ones digit in decimal value

;-----------------------------------------------------------
; PROGRAM BEGINS
;-----------------------------------------------------------

        CODE    0               ; Default start address
        NOP                     ; required for ICD mode

;-----------------------------------------------------------
; Port & display setup

        BANKSEL TRISD           ; Select bank 1
        CLRF    TRISD           ; Display port is output
        MOVLW   B'10000011'     ; Analogue input setup code
        MOVWF   ADCON1          ; Right justify result,
                                ; Port A = analogue inputs
                                ; with external reference

        BANKSEL PORTD           ; Select bank 0
        CLRF    PORTD           ; Clear display outputs
        MOVLW   B'01000001'     ; Analogue input setup code
        MOVWF   ADCON0          ; f/8, RA0, done, enable

        CALL    inid            ; Initialise the display

;-----------------------------------------------------------
; MAIN LOOP
;-----------------------------------------------------------

start   CALL    getADC          ; read input
        CALL    con4            ; convert to decimal
        CALL    putLCD          ; display input
        GOTO    start           ; jump to main loop
```

ADC10BIT2 Source Code

```
;-------------------------------------------------------------
; SUBROUTINES
;-------------------------------------------------------------
; Read ADC input and store
;-------------------------------------------------------------

getADC  MOVLW   007             ; load counter
        MOVWF   count
down    DECFSZ  count           ; and delay 20us
        GOTO    down

        BSF     ADCON0,GO       ; start ADC..
wait    BTFSC   ADCON0,GO       ; ..and wait for finish
        GOTO    wait
        RETURN

;-------------------------------------------------------------
; Convert 10-bit input to decimal
;-------------------------------------------------------------

con4    MOVF    ADRESH,W        ; get ADC result
        MOVWF   ADhi            ; high bits
        BANKSEL ADRESL          ; in bank 1
        MOVF    ADRESL,W        ; get ADC result
        BANKSEL ADRESH          ; default bank 0
        MOVWF   ADlo            ; low byte

; Multiply by 4 for result 0 - 4096 by shifting left.........

        BCF     STATUS,C        ; rotate 0 into LSB and
        RLF     ADlo            ; shift low byte left
        BTFSS   STATUS,C        ; carry out?
        GOTO    rot1            ; no, leave carry clear
        BSF     STATUS,C        ; rotate 1 into LSB and
rot1    RLF     ADhi            ; shift high byte left

        BCF     STATUS,C        ; rotate 0 into LSB
        RLF     ADlo            ; rotate low byte left again
        BTFSS   STATUS,C        ; carry out?
        GOTO    rot2            ; no, leave carry clear
        BSF     STATUS,C        ; rotate 1 into LSB and
rot2    RLF     ADhi            ; shift high byte left

; Clear BCD registers.....................................

clrbcd  CLRF    thos            ; zero thousands digit
        CLRF    huns            ; zero hundreds digit
        CLRF    tens            ; zero tens digit
        CLRF    ones            ; zero ones digit


; Calclulate thousands low byte ..........................

tholo   MOVF    ADhi,F          ; check high byte
        BTFSC   STATUS,Z        ; high byte zero?
        GOTO    hunlo           ; yes, next digit

        BSF     STATUS,C        ; set carry for subtract
        MOVLW   0E8             ; load low byte of 1000
        SUBWF   ADlo            ; and subtract low byte
        BTFSC   STATUS,C        ; borrow from high bits?
        GOTO    thohi           ; no, do high byte
        DECF    ADhi            ; yes, subtract borrow

; Calculate thousands high byte...........................

thohi   BSF     STATUS,C        ; set carry for subtract
        MOVLW   003             ; load high byte of 1000
        SUBWF   ADhi            ; subtract from high byte
        BTFSC   STATUS,C        ; result negative?
        GOTO    incth           ; no, inc digit and repeat
        ADDWF   ADhi            ; yes, restore high byte

; Restore remainder when done ............................

        BCF     STATUS,C        ; clear carry for add
        MOVLW   0E8             ; load low byte of 1000
        ADDWF   ADlo            ; add to low byte
        BTFSC   STATUS,C        ; carry out?
        INCF    ADhi            ; yes, inc high byte
        GOTO    hunlo           ; and do next digit

; Increment thousands digit and repeat....................

incth   INCF    thos            ; inc digit
        GOTO    tholo           ; and repeat
```

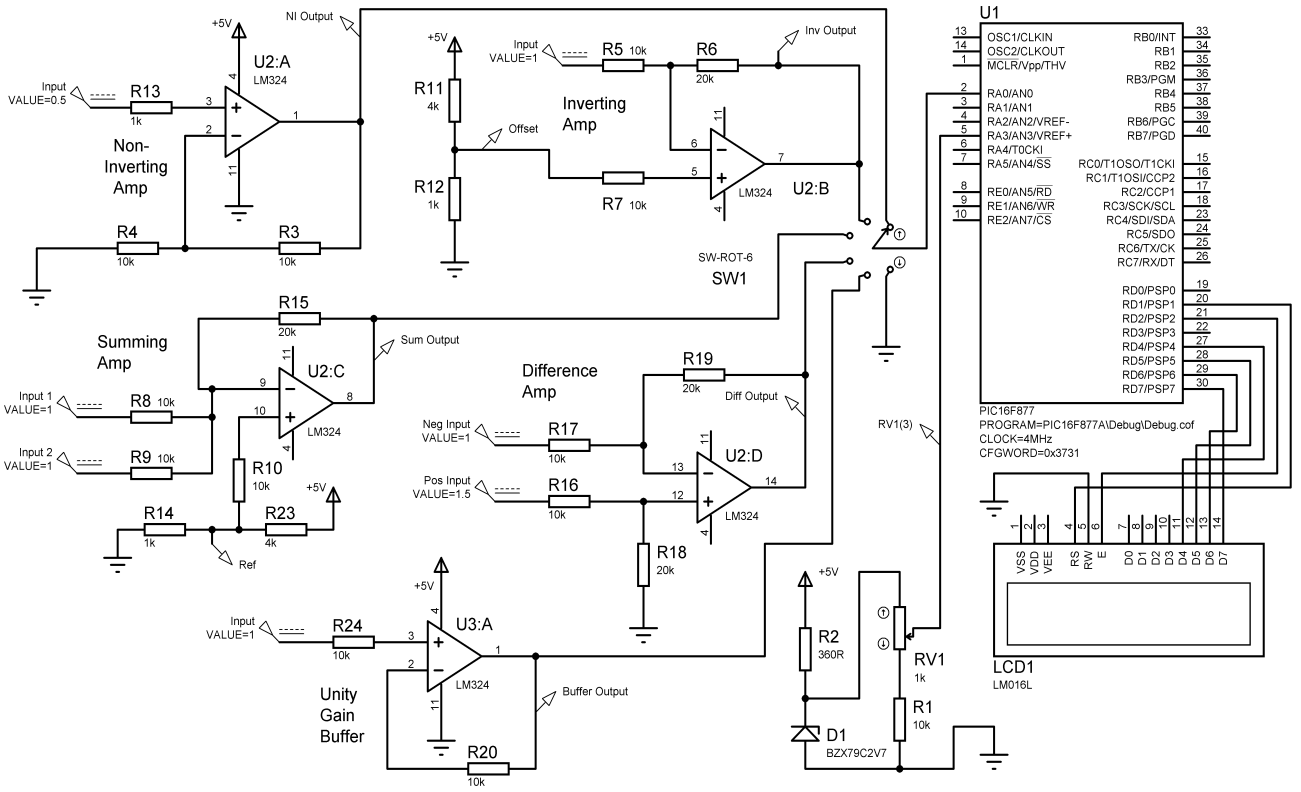
ADC10BIT2 Source Code

```
; Calclulate hundreds ....................................

hunlo   MOVLW   064             ; load 100
        BSF     STATUS,C        ; set carry for subtract
        SUBWF   ADlo            ; and subtract low byte
        BTFSC   STATUS,C        ; result negative?
        GOTO    inch            ; no, inc hundreds & repeat

        MOVF    ADhi,F          ; yes, test high byte
        BTFSC   STATUS,Z        ; zero?
        GOTO    remh            ; yes, done
        DECF    ADhi            ; no, subtract borrow
inch    INCF    huns            ; inc hundreds digit
        GOTO    hunlo           ; and repeat

remh    ADDWF   ADlo            ; restore onto low byte


; Calculate tens digit....................................

subt    MOVLW   D'10'           ; load 10
        BSF     STATUS,C        ; set carry for subtract
        SUBWF   ADlo            ; and subtract from result
        BTFSS   STATUS,C        ; and check if done
        GOTO    remt            ; yes, restore remainder
        INCF    tens            ; no, count number of loops
        GOTO    subt            ; and repeat


; Restore remainder.....................................

remt    ADDWF   ADlo            ; yes, add 10 back on
        MOVF    ADlo,W          ; load remainder
        MOVWF   ones            ; and store as ones digit

        RETURN                  ; done


;---------------------------------------------------------
; Output to display
;---------------------------------------------------------

putLCD  BCF     Select,RS       ; set display command mode
        MOVLW   080             ; code to home cursor
        CALL    send            ; output it to display
        BSF     Select,RS       ; and restore data mode
```

```
; Convert digits to ASCII and display.....................

        MOVLW   030             ; load ASCII offset
        ADDWF   thos            ; convert thousands to ASCII
        ADDWF   huns            ; convert hundreds to ASCII
        ADDWF   tens            ; convert tens to ASCII
        ADDWF   ones            ; convert ones to ASCII

        MOVF    thos,W          ; load thousands code
        CALL    send            ; and send to display
        MOVLW   '.'             ; load point code
        CALL    send            ; and output
        MOVF    huns,W          ; load hundreds code
        CALL    send            ; and send to display
        MOVF    tens,W          ; load tens code
        CALL    send            ; and output
        MOVF    ones,W          ; load ones code
        CALL    send            ; and output
        MOVLW   ' '             ; load space code
        CALL    send            ; and output
        MOVLW   'V'             ; load volts code
        CALL    send            ; and output
        MOVLW   'o'             ; load volts code
        CALL    send            ; and output
        MOVLW   'l'             ; load volts code
        CALL    send            ; and output
        MOVLW   't'             ; load volts code
        CALL    send            ; and output
        MOVLW   's'             ; load volts code
        CALL    send            ; and output

        RETURN                  ; done

;---------------------------------------------------------
; INCLUDED ROUTINES
;---------------------------------------------------------
; Include LCD driver routine
;
        INCLUDE"LCD.INC"
;
;       Contains routines:
;       init:   Initialises display
;       onems:  1 ms delay
;       xms:    X ms delay
;               Receives X in W
;       send:   sends a character to display
;               Receives: Control code in W (Select,RS=0)
;                         ASCII character code in W (RS=1)
;
;---------------------------------------------------------
        END                     ; of source code
;---------------------------------------------------------
```

Interfacing PIC Microcontrollers

AMPS2 Schematic



- This application demonstrates a range of op-amp interface configurations using the low resolution voltage display (0-2.55V)
- The operating characterisitics of the inverting, non-inverting, summing, difference and unity gain buffer are illustrated by the simulated output values
- The exact gain, output voltage offset and external currents associated with each configuration using the LM324 op-amp can be evaluated

AMPS2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:                AMPS2
;       Devised by:             MPB
;       Date:                   14-01-13
;       Status:                 Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Displays input from different amplifier types
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A
;       Clock = XT 4MHz, standard fuse settings
        __CONFIG 0x3731

;       LABEL EQUATES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        INCLUDE "P16F877A.INC"
;       standard register labels

;-----------------------------------------------------------
; User register labels
;-----------------------------------------------------------
; GPR 20 - 2F allocated to included LCD display routine

count   EQU     30      ; Counter for ADC setup delay
ADbin   EQU     31      ; Binary input value
huns    EQU     32      ; Hundreds digit in decimal value
tens    EQU     33      ; Tens digit in decimal value
ones    EQU     34      ; Ones digit in decimal value

;-----------------------------------------------------------
; PROGRAM BEGINS
;-----------------------------------------------------------

        CODE    0               ; Default start address
        NOP                     ; required for ICD mode

;-----------------------------------------------------------
; Port & display setup

        BANKSEL TRISC           ; Select bank 1
        CLRF    TRISD           ; Display port is output
        MOVLW   B'00000011'     ; Analogue input setup code
        MOVWF   ADCON1          ; Left justify result,
                                ; Port A = analogue inputs

        BANKSEL PORTC           ; Select bank 0
        CLRF    PORTD           ; Clear display outputs
        MOVLW   B'01000001'     ; Analogue input setup code
        MOVWF   ADCON0          ; f/8, RA0, done, enable

        CALL    inid            ; Initialise the display

;-----------------------------------------------------------
; MAIN LOOP
;-----------------------------------------------------------

start   CALL    getADC          ; read input
        CALL    condec          ; convert to decimal
        CALL    putLCD          ; display input
        GOTO    start           ; jump to main loop

;-----------------------------------------------------------
; SUBROUTINES
;-----------------------------------------------------------
; Read ADC input and store ................................

getADC  BSF     ADCON0,GO       ; start ADC..
wait    BTFSC   ADCON0,GO       ; ..and wait for finish
        GOTO    wait
        MOVF    ADRESH,W        ; store result, high 8 bits
        RETURN

;-----------------------------------------------------------
; Convert input to decimal

condec  MOVWF   ADbin           ; get ADC result
        CLRF    huns            ; zero hundreds digit
        CLRF    tens            ; zero tens digit
        CLRF    ones            ; zero ones digit

; Calclulate hundreds.....................................

        BSF     STATUS,C        ; set carry for subtract
        MOVLW   D'100'          ; load 100
sub1    SUBWF   ADbin           ; and subtract from result
        INCF    huns            ; count number of loops
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub1            ; no, carry on

        ADDWF   ADbin           ; yes, add 100 back on
        DECF    huns            ; and correct loop count

; Calculate tens digit....................................

        BSF     STATUS,C        ; repeat process for tens
        MOVLW   D'10'           ; load 10
sub2    SUBWF   ADbin           ; and subtract from result
        INCF    tens            ; count number of loops
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub2            ; no, carry on

        ADDWF   ADbin           ; yes, add 100 back on
        DECF    tens            ; and correct loop count
        MOVF    ADbin,W         ; load remainder
        MOVWF   ones            ; and store as ones digit

        RETURN                  ; done
```

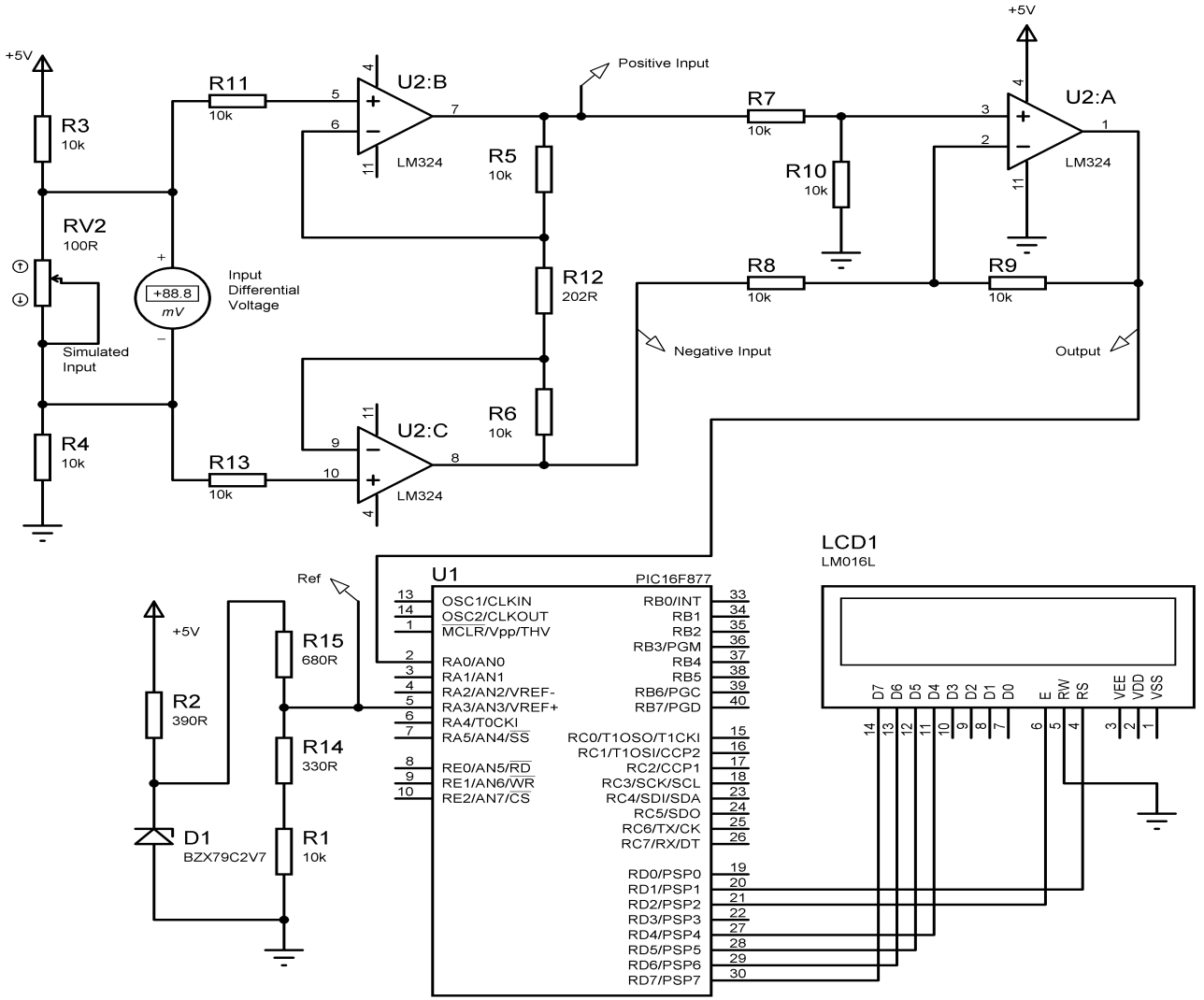
AMPS2 Source Code

```
;-------------------------------------------------------------
; Output to display

putLCD BCF    Select,RS      ; set display command mode
       MOVLW  080            ; code to home cursor
       CALL   send           ; output it to display
       BSF    Select,RS      ; and restore data mode

; Convert digits to ASCII and display.........................

       MOVLW  030            ; load ASCII offset
       ADDWF  huns           ; convert hundreds to ASCII
       ADDWF  tens           ; convert tens to ASCII
       ADDWF  ones           ; convert ones to ASCII

       MOVF   huns,W         ; load hundreds code
       CALL   send           ; and send to display
       MOVLW  '.'            ; load point code
       CALL   send           ; and output
       MOVF   tens,W         ; load tens code
       CALL   send           ; and output
       MOVF   ones,W         ; load ones code
       CALL   send           ; and output
       MOVLW  ' '            ; load space code
       CALL   send           ; and output
       MOVLW  'V'            ; load volts code
       CALL   send           ; and output
       MOVLW  'o'            ; load volts code
       CALL   send           ; and output
       MOVLW  'l'            ; load volts code
       CALL   send           ; and output
       MOVLW  't'            ; load volts code
       CALL   send           ; and output
       MOVLW  's'            ; load volts code
       CALL   send           ; and output

       RETURN                ; done
```

```
;-------------------------------------------------------------
; INCLUDED ROUTINES
;-------------------------------------------------------------
; Include LCD driver routine
;
       INCLUDE "LCD.INC"
;      Contains routines:
;      init:  Initialises display
;      onems: 1 ms delay
;      xms:   X ms delay
;             Receives X in W
;      send:  sends a character to display
;             Receives: Control code in W (Select,RS=0)
;                       ASCII character code in W (RS=1)
;
;-------------------------------------------------------------
       END                   ; of source code
;-------------------------------------------------------------
```

Interfacing PIC Microcontrollers

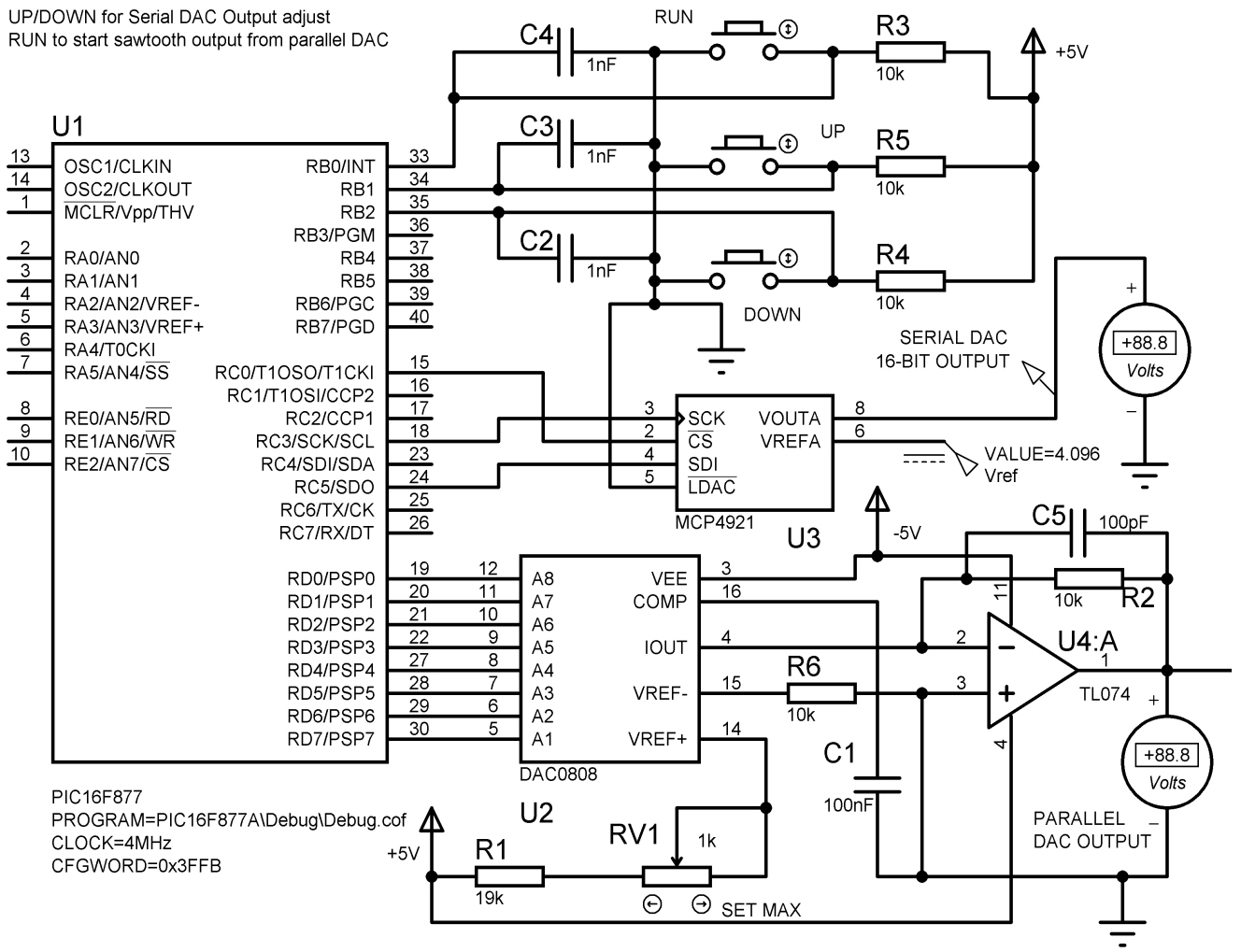
INSTAMP2 Schematic



- This application demonstrates an interface for an instumentation amplifier
- A small differential input voltage up to 25mV is fed to an input stage with a gain of 100
- The output differential unity gain amplifier gives a single ended output of 0 - 2.5V
- The display MCU and LCD use the same firmware as the application AMPS2

Interfacing PIC Microcontrollers

DACS2 Schematic



- This application demonstrates analogue output
- The parallel DAC receives an 8-bit input and produces a proportionate analogue output current that is converted to to voltage in the output buffer
- The serial DAC receives an SPI serial data input containing a 16-bit code that it converts to a corresponding anlogue voltage output
- The outputs are incremented and decremented via the push buttons

DACS2 Source Code

```
;*************************************************
;       DACS2.ASM      MPB     18-01-13
;
;       Test program for parallel
;       and serial D/A Converters
;       DAC0808 & MCP4921
;
;       Updated for VSM v8
;
;*************************************************

        PROCESSOR 16F877A
        INCLUDE "P16F877A.INC"
        __CONFIG 0X3731

Hibyte  EQU     020     ; SPI data high byte
Lobyte  EQU     021     ; SPI data low byte

        CODE 0          ; Load at default range
        NOP             ; for ICD operations

; Initialise parallel and serial ports -------------

        BANKSEL TRISD
        CLRF    TRISD           ; Parallel port
        BCF     TRISC,5         ; Serial data
        BCF     TRISC,3         ; Serial clock
        BCF     TRISC,0         ; Chip select
        CLRF    SSPSTAT         ; default SPI mode

        BANKSEL PORTD
        CLRF    PORTD           ; zero PDAC
        CLRF    SSPCON          ; default SPI mode

        MOVLW   B'00111001'     ; Initial SDAC data
        MOVWF   Hibyte          ; and store
        MOVLW   B'11111111'
        MOVWF   Lobyte

; Check buttons ------------------------------------

up      BTFSC   PORTB,1         ; Test UP button
        GOTO    down            ; and jump if off
        INCF    PORTD           ; Increment PDAC
        INCF    Hibyte          ; Increment SDAC
waitup  BTFSS   PORTB,1         ; Wait for..
        GOTO    waitup          ; button release

down    BTFSC   PORTB,2         ; Test DOWN button
        GOTO    spi             ; and jump if off
        DECF    PORTD           ; Decrement PDAC
        DECF    Hibyte          ; Decrement SDAC
waitdo  BTFSS   PORTB,2         ; Wait for..
        GOTO    waitdo          ; button release

; Send 16-bit data to SDAC via SPI port ------------

spi     BSF     SSPCON,SSPEN    ; Enable SPI port

        BCF     PORTC,0         ; Enable SDAC chip
        MOVF    Hibyte,W        ; Get high data
        MOVWF   SSPBUF          ; and send it
waithi  BTFSS   PIR1,SSPIF      ; Wait for..
        GOTO    waithi          ; SPI interrupt
        BCF     PIR1,SSPIF      ; Reset interrupt

        MOVF    Lobyte,W        ; Get low data
        MOVWF   SSPBUF          ; and send it
waitlo  BTFSS   PIR1,SSPIF      ; Wait for..
        GOTO    waitlo          ; SPI interrupt
        BCF     PIR1,SSPIF      ; Reset interrupt

        BSF     PORTC,0         ; Disable SDAC chip

; Run output loop until reset ----------------------

        BTFSC   PORTB,0         ; Test run button
        GOTO    up              ; and repeat loop

run     INCF    PORTD           ; Increment PDAC
        GOTO    run

        END ;----------------------------------------
```