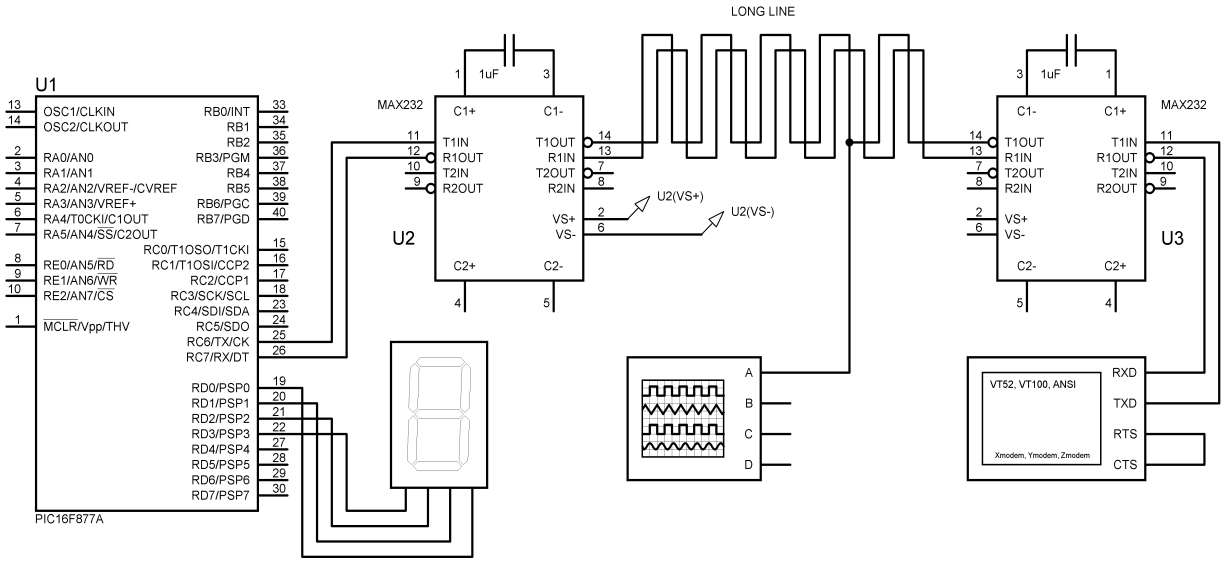
Interfacing PIC Microcontrollers

USART2 Schematic



- This application demonstrates basic serial communication using the PIC USART port
- The virtual terminal allows ASCII input to the port from the users keyboard and displays characters returned from the PIC
- The serial line is connected via RS232 transceivers that boost the line voltage
- The data received from the virtual terminal is also displayed on a BCD 7 segment LED

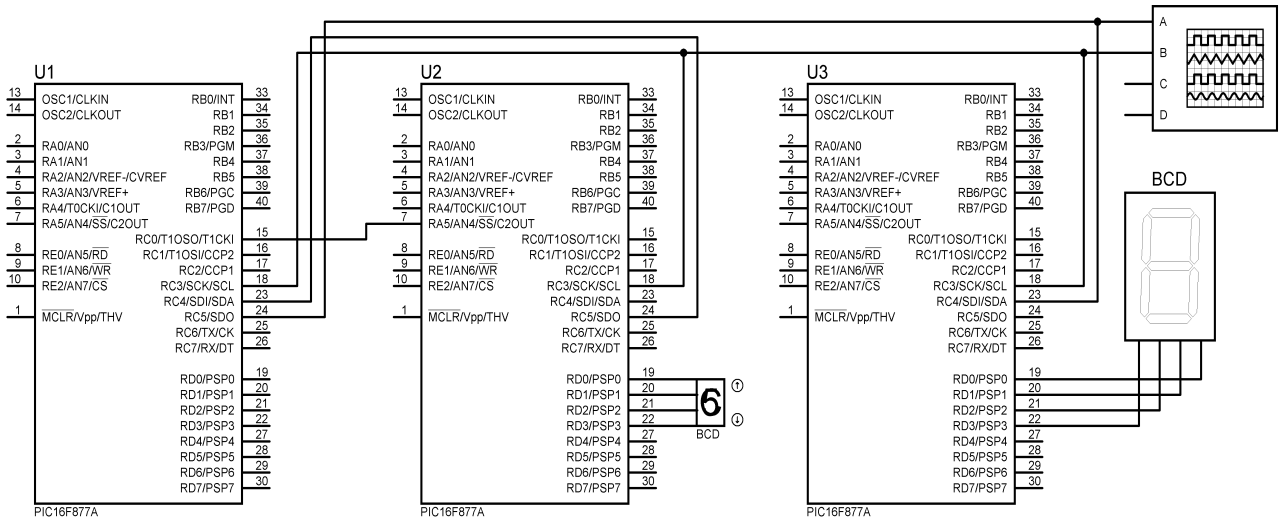
USART2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       USART2.ASM      MPB     23-01-13
;.....................................................................
;
;       Test RS232 communications using the
;       USART Asynchronous Transmit and Receive
;
;       The Proteus Virtual Terminal allows ASCII characters
;       to be displayed, and generated from the computer keys.
;       The program outputs a fixed message to the display
;       from a table, and then displays numbers input from the
;       terminal on a BCD 7-segment LED display.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877        ; define MPU
        __CONFIG 0x3731                 ; XT clock (4MHz)

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

        INCLUDE "P16F877A.INC"; Standard register labels

        Point   EQU     020
        Inchar  EQU     021

; Initialise ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Place machine code
        NOP                     ; Required for ICD mode

        BANKSEL TRISD           ; Select bank 1
        CLRF    TRISD           ; Display outputs
        BCF     TXSTA,TX9       ; Select 8-bit transmission
        BCF     TXSTA,TXEN      ; Disable transmission initially
        BCF     TXSTA,SYNC      ; Asynchronous mode
        BSF     TXSTA,BRGH      ; High baud rate

        MOVLW   D'25'           ; Baud rate counter value ..
        MOVWF   SPBRG           ; .. for 9600 baud, 4MHz clock
        BSF     TXSTA,TXEN      ; Enable transmission

        BANKSEL RCSTA           ; Select bank 0
        BSF     RCSTA,SPEN      ; Enable serial port


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

        CALL    write           ; Display message on terminal
readin  CALL    read            ; Get number input from terminal
        GOTO    readin          ; Keep reading until reset
```

```
; SUBROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Write message to terminal.....................................

write   CLRF    Point           ; Table pointer = 0
next    MOVF    Point,W         ; Load table pointer
        CALL    mestab          ; Get character
        CALL    sencom          ; Output to terminal
        INCF    Point           ; Point to next
        MOVLW   D'14'           ; Number of characters + 1
        SUBWF   Point,W         ; Check pointer
        BTFSS   STATUS,Z        ; Last character done?
        GOTO    next            ; No - next
        RETURN                  ; All done


; Read input numbers from terminal..............................

read    BSF     RCSTA,CREN      ; Enable reception
waitin  BTFSS   PIR1,RCIF       ; Character received?
        GOTO    waitin          ; no - wait

        MOVF    RCREG,W         ; get input character
        MOVWF   Inchar          ; store input character
        MOVLW   030             ; ASCII number offset
        SUBWF   Inchar,W        ; Calculate number
        MOVWF   PORTD           ; display it
        RETURN                  ; done


; Transmit a character .........................................

sencom  MOVWF   TXREG           ; load transmit register
waitot  BTFSS   PIR1,TXIF       ; sent?
        GOTO    waitot          ; no
        RETURN                  ; yes


; Table of message characters...................................

mestab  ADDWF   PCL             ; Modify program counter
        RETLW   'E'             ; Point = 0
        RETLW   'N'             ; Point = 1
        RETLW   'T'             ; Point = 2
        RETLW   'E'             ; Point = 3
        RETLW   'R'             ; Point = 4
        RETLW   ' '             ; Point = 5
        RETLW   'N'             ; Point = 6
        RETLW   'U'             ; Point = 7
        RETLW   'M';            ; Point = 8
        RETLW   'B'             ; Point = 9
        RETLW   'E'             ; Point = 10
        RETLW   'R'             ; Point = 11
        RETLW   ':'             ; Point = 12
        RETLW   ' '             ; Point = 13


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

Interfacing PIC Microcontrollers

SERSPI2 Schematic



- This application demonstrates serial communications using the PIC SPI port
- Three MCUs are used, a master, slave transmitter and slave receiver
- Test data is input from a thumbwheel switch to the slave transmitter and displayed on the the slave receiver
- The master genrates a clock signal and controls the data transfer
- Each MCU has its own test program

SERSPI2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       SPIM2.ASM      MPB     24-01-13
;.......................................................................
;
;       SPI Master program
;
;       Outputs clock to slave transmitter, receives BCD data
;       and sends it to slave receiver for display
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877      ; define MPU
        __CONFIG 0x3731               ; XT clock (4MHz)
        INCLUDE "P16F877.INC" ; Standard register labels

Store   EQU     020

; Initialise ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Place machine code
        NOP                     ; Required for ICD mode

        BANKSEL TRISC
        BCF     TRISC,5         ; Serial data (SDO) output
        BCF     TRISC,3         ; Serial clock (SCK) output
        BCF     TRISC,0         ; Slave select (SS) output
        CLRW    SSPSTAT         ; Default clock timing

        BANKSEL PORTD
        BSF     PORTC,0         ; Reset slave transmitter
        CLRF    SSPCON          ; SPI master mode, clock = 1MHz
        BSF     SSPCON,SSPEN    ; Enable SPI mode

; Main loop ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

again   BCF     PORTC,0         ; Enable slave transmitter
        MOVWF   SSPBUF          ; Rewrite buffer to start clock
waitin  BTFSS   PIR1,SSPIF      ; wait for SPI interrupt
        GOTO    waitin          ; for data recieved

        BCF     PIR1,SSPIF      ; clear interrupt flag
        MOVF    SSPBUF,W        ; read SPI buffer
        MOVWF   Store           ; store BCD value
        BSF     PORTC,0         ; Disable slave transmitter
        MOVWF   SSPBUF          ; Reload SPI buffer

waits   BTFSS   PIR1,SSPIF      ; wait for SPI interrupt
        GOTO    waits           ; for data sent
        BCF     PIR1,SSPIF      ; clear interrupt flag
        GOTO    again           ; repeat main loop

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

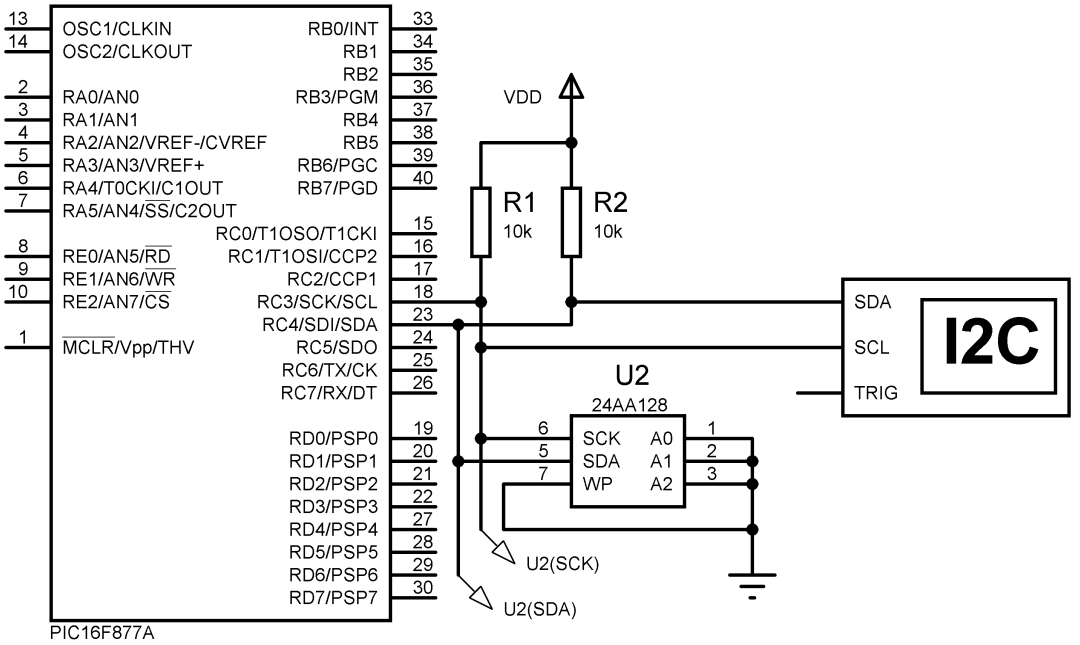
Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       SPIT2.ASM       MPB     24-01-13
;...........................................................
;
;       SPI Slave Transmitter program
;       Waits for !SS and transmits switch BCD data
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877        ; define MPU
        __CONFIG 0x3731         ; XT clock (4MHz)
        INCLUDE "P16F877A.INC"; Standard register labels

; Initialise ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Place machine code
        NOP                     ; Required for ICD mode

        BANKSEL TRISC
        BCF     TRISC,5         ; Serial data output
        CLRW    SSPSTAT         ; Default clock timing

        BANKSEL PORTD
        MOVLW   B'00000100'     ; SPI slave mode with SS
        MOVWF   SSPCON          ; SPI clock = 1MHz
        BSF     SSPCON,SSPEN    ; Enable SPI mode

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


start   MOVF    PORTD,W         ; Read BCD switch
        MOVWF   SSPBUF          ; Write SPI buffer
wait    BTFSS   PIR1,SSPIF      ; wait for SPI interrupt
        GOTO    wait
        BCF     PIR1,SSPIF      ; clear interrupt flag
        GOTO    start           ; repeat main loop

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

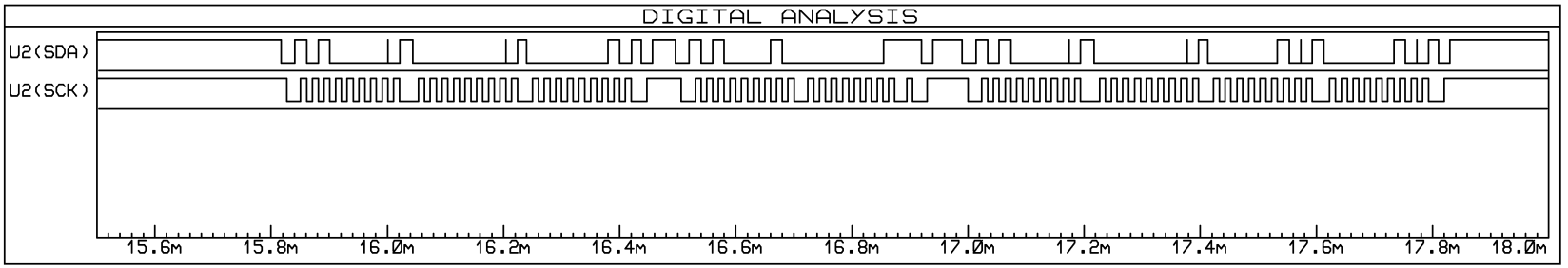
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       SPIR2.ASM       MPB     23-01-13
;...............................................................
;
;       SPI Slave Receiver program
;       Waits for BCD data sent from master and displays it
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877        ; define MPU
        __CONFIG 0x3731                 ; XT clock (4MHz)
        INCLUDE "P16F877A.INC"; Standard register labels

; Initialise ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Place machine code
        NOP                     ; Required for ICD mode

        BANKSEL TRISD
        CLRF    TRISD           ; Display outputs
        CLRF    SSPSTAT         ; Default clock timing

        BANKSEL PORTD
        MOVLW   B'00000101'     ; SPI slave mode, SS disabled
        MOVWF   SSPCON          ; SPI clock = 1MHz

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

        BSF     SSPCON,SSPEN    ; Enable SPI mode
wait    BTFSS   PIR1,SSPIF      ; wait for SPI interrupt
        GOTO    wait

        MOVF    SSPBUF,W        ; get data
        MOVWF   PORTD           ; and display
        BCF     PIR1,SSPIF      ; clear interrupt flag
        GOTO    wait            ; repeat main loop

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

Interfacing PIC Microcontrollers

SERI2C2 Schematic



- This application demonstrates serial data transmission using the PIC I^2C port
- The program strores test data in a serial memory chip
- The data analyser allows the test data to be picked up from the data line and displayed for system debugging (see below)
- The protocol uses software addressing



SERI2C2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       I2C2.ASM              MPB     24-01-13
;
;       Test program for 24AA128 I2C 16k byte serial
;       memory with P16F877A (4MHz XT)
;       Demonstrates single byte write and read
;       with 10-bit address.
;
;       Write data from                       0x20
;       High address                  0x21
;       Low address                   0x22
;       Read data back to             0x23
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A
        __CONFIG 0x3FF1
        INCLUDE "P16F877A.INC"

; Data, address & control registers ;;;;;;;;;;;;;;;;;;;;

SenReg  EQU     0x20            ; Send data store
HiReg   EQU     0x21            ; High address store
LoReg   EQU     0x22            ; Low address store
RecReg  EQU     0x23            ; Receive data store
ConReg  EQU     0x24            ; Control byte store
Temp    EQU     0x25            ; Scratchpad location

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Program start address

        NOP                       ; ICPD location
         CLRF   SenReg          ; Zeroise data
        CLRF    HiReg           ; Zeroise high address
        CLRF    LoReg           ; Zeroise low address
        GOTO    begin           ; jump to main program
```

```
;-----------------------------------------------------
; SUBROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Wait for interrupt flag SSPIF for send/recive done ...

wint    NOP                     ; BANKSEL has no address
        BANKSEL PIR1            ; Select bank
        BCF     PIR1,SSPIF      ; reset interrupt flag
win     NOP
        BTFSS   PIR1,SSPIF      ; wait for..
        GOTO    win             ; ..transmit done
        RETURN                  ; done


; Send a byte ...................................

send    NOP                     ; Select..
        BANKSEL SSPBUF          ; .. bank
        MOVWF   SSPBUF          ; Send address/data
        CALL    wint            ; Wait until sent
        RETURN                  ; done

;-----------------------------------------------------
; Routines to send start, control, address, data, stop ..
;.....................................................

sencon  NOP                     ; GENERATE START BIT
        BANKSEL PIR1
        BCF     PIR1,SSPIF      ; Clear interrupt flag
        BANKSEL SSPCON2         ; select register page
        BSF     SSPCON2,ACKSTAT ; Set acknowledge flag
        BSF     SSPCON2,SEN     ; Generate start bit
        CALL    wint            ; wait till done
        MOVF    ConReg,W        ; SEND CONTROL BYTE
        CALL    send            ; Memory ID & chip address
        RETURN                  ; done

;.....................................................

senadd  NOP
        BANKSEL SSPCON          ; SEND ADDRESS BYTES
        MOVF    HiReg,W         ; load address high byte
        CALL    send            ; and send
        MOVF    LoReg,W         ; load address low byte
        CALL    send            ; and send
        RETURN

;.....................................................

sendat  MOVF    SenReg,W        ; Load data
        CALL    send            ; and send
        RETURN                  ; done
```

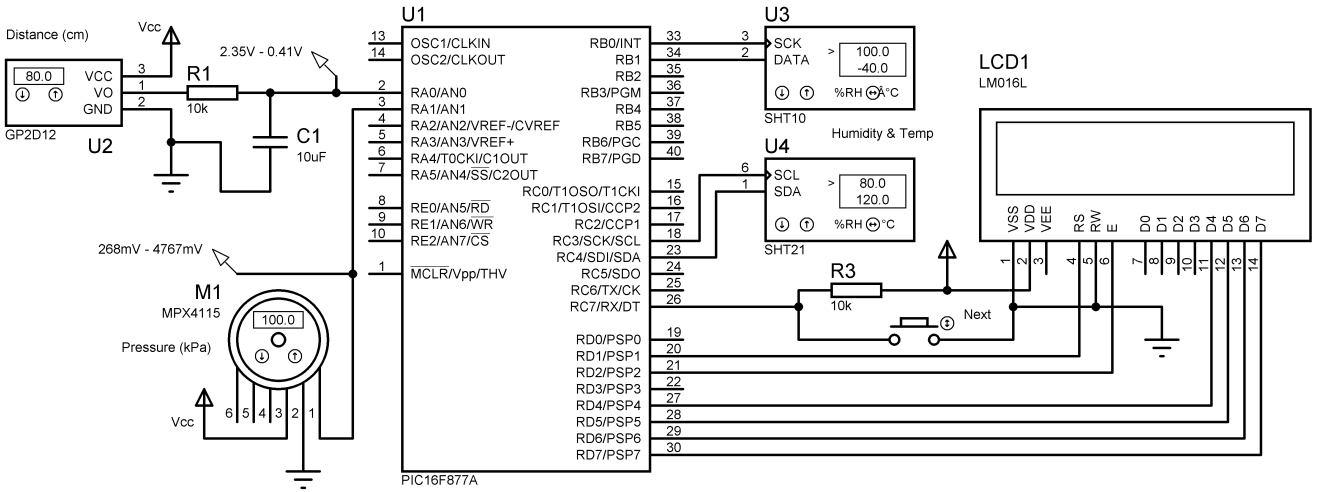
SERI2C2 Source Code

```
;.....................................................

senstop NOP
        BANKSEL SSPCON2         ; GENERATE STOP BIT
        BSF     SSPCON2,PEN     ; Generate stop bit
        CALL    wint            ; wait till done
        RETURN                  ; done

;.....................................................

senack  NOP
        BANKSEL SSPCON2
        BSF     SSPCON2,ACKDT   ; Set ack. bit high
        BSF     SSPCON2,ACKEN   ; Initiate ack.sequence
        CALL    wint            ; Wait for ack. done
        RETURN                  ; done

;.....................................................

wait    NOP
        BANKSEL TMR0            ; WAIT FOR WRITE DONE
        MOVLW   d'156'          ; Set starting value
        MOVWF   TMR0            ; and load into timer
        BANKSEL INTCON          ; 64 x 156us = 10ms
        BCF     INTCON,T0IF     ; Reset timer out flag
wem     BTFSS   INTCON,T0IF     ; Wait 10ms
        GOTO    wem             ; for timeout
        BANKSEL TMR0            ; default bank
        RETURN                  ; Byte write done....

;-----------------------------------------------------
; Initialisation sequence ..............................

init    NOP
        BANKSEL SSPCON2         ;
        MOVLW   b'01100000'     ; Set ACKSTAT,ACKDT bits
        MOVWF   SSPCON2         ; Reset SEN,ACK bits
        MOVLW   b'10000000'     ;
        MOVWF   SSPSTAT         ; Speed & signal levels
        MOVLW   0x13            ; Clock = 50kHz
        MOVWF   SSPADD          ; Load baud rate count-1
        BANKSEL SSPCON          ;
        MOVLW   b'00101000'     ;
        MOVWF   SSPCON          ; Set mode & enable
        BCF     PIR1,SSPIF      ; clear interrupt flag

; Initialise TIMER0 for write delay ...............

        BANKSEL OPTION_REG      ;
        MOVLW   B'11000101'     ; TIMER0 setup code
        MOVWF   OPTION_REG      ; Internal clock,1/64
        BANKSEL TMR0
        RETURN
```

```
;-----------------------------------------------------
; Write a test byte to given address ................

writeb  MOVLW   0xA0            ; Control byte for WRITE
        MOVWF   ConReg          ;
        CALL    sencon          ; Send control byte
        CALL    senadd          ; Send address bytes
        CALL    sendat          ; Send data byte
        CALL    senstop         ; Send stop bit
        CALL    wait            ; Wait 10ms for write
        RETURN

;-----------------------------------------------------
; Read the byte from given address ...................

readb   MOVLW   0xA0            ; Control byte to WRITE
        MOVWF   ConReg          ; address to memory
        CALL    sencon          ; Send control byte
        CALL    senadd          ; Send address bytes
        CALL    senstop         ; Stop

        MOVLW   0xA1            ; Control byte to READ
        MOVWF   ConReg          ; data from memory
        CALL    sencon          ; Send control byte
        BANKSEL SSPCON2
        BSF     SSPCON2,RCEN    ; Enable receive mode
war     BTFSS   SSPSTAT,BF      ; Check ...
        GOTO    war             ; for read done
        CALL    senack          ; send NOT acknowledge
        CALL    senstop         ; send stop bit

        MOVF    SSPBUF,W        ; Read receive buffer
        MOVWF   RecReg          ; and store it
        RETURN


; MAIN PROGRAM        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

begin   CALL    init            ; Initialise for I2C
next    CALL    writeb          ; write the test byte
        CALL    readb           ; and read it back
        INCF    SenReg          ; next data
        INCF    LoReg           ; next location
        BTFSS   STATUS,Z        ; end of memory block?
        GOTO    next            ; no, next location
        INCF    HiReg           ; next block

        MOVF    HiReg,W         ; copy high address byte
        MOVWF   Temp            ; store it
        MOVLW   0x40            ; Last block = 3F
        SUBWF   Temp            ; Compare
        BTFSS   STATUS,Z        ; Finish if block = 40xx
        GOTO    next            ; next memory block..
        SLEEP                   ; .. unless done

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

Interfacing PIC Microcontrollers

ICSENS2 Schematic

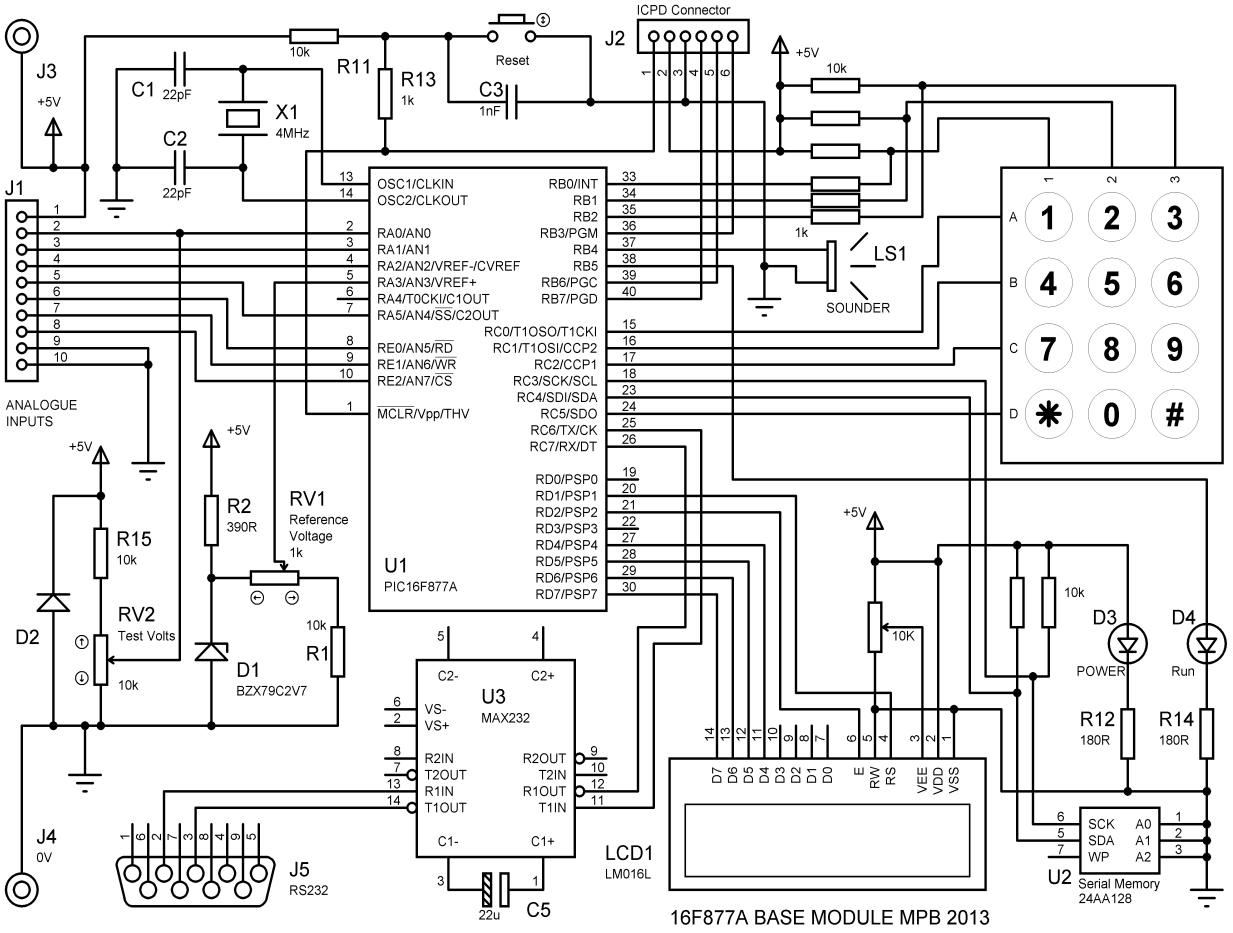


Note that a test program is not available for this project, but may be developed based on the source code examples for analogue and serial inputs provided

- This schematic shows a range of integrated sensors for measuring distance, temperature, humidity and pressure
- The reflective distance sensor produces an analogue output voltage
- The pressure sensor also produces a dc output voltage, proportional to the absolute pressure
- The SHT10 sensor produces a serial data output in a non-standard format which must be decoded in software
- The SHT21 sensor produces output data in I2C format

Interfacing PIC Microcontrollers

BASE2 Schematic



- This application incorporates range of interfaces in a general purpose controller board
- These include a basic keypad, aphanumeric LCD, serial expansion flash memory, RS232 interface, and provision for digital and analogue external connections
- The test program exercises all the features of the board, using included routines previously developed (only the main program is listed below)

BASE2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:              Interfacing PICs
;       Source File Name:     BASE2.ASM
;       Devised by:           MPB
;       Date:                 28-01-13
;       Status:               Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Program to exercise the 16F877 BASE module
;       with 8-bit analogue input, LCD, keypad
;       and serial memory
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A       ; Clock = XT 4MHz
        __CONFIG 0x3731         ; Standard fuse settings
        INCLUDE "P16F877A.INC"  ; standard register labels

; User register label allocation ;;;;;;;;;;;;;;;;;;;;;;;;;;;

; GPR 20 - 2A  local variables
; GPR 30 - 32  KEYPAD subroutine
; GPR 60 - 65  SERMEM serial memory driver
; GPR 70 - 75  LCDIS display driver
; GPR 77 - 7A  CONDEC BCD conversion routine

LCDport EQU    08      ; assign LCD to Port D
LCDdirc EQU    88      ; data direction register

Temp    EQU    20      ; temp store
Tabin   EQU    21      ; Table pointer

; Keypad I/O pins

    #DEFINE RowA    PORTC,0
    #DEFINE RowB    PORTC,1
    #DEFINE RowC    PORTC,2
    #DEFINE RowD    PORTC,5

    #DEFINE Col1    PORTB,0
    #DEFINE Col2    PORTB,1
    #DEFINE Col3    PORTB,2
```

```
;-------------------------------------------------------------
; MAIN PROGRAM
;-------------------------------------------------------------

        CODE    0               ; Default start address
        NOP                     ; required for ICPD

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

        BANKSEL TRISA           ; Select bank 1
        MOVLW   B'11001111'      ; Port B code for
        MOVWF   TRISB           ; keypad inputs & sounder
        MOVLW   B'10011000'     ; Port C code for
        MOVWF   TRISC           ; keypad, memory & USART
        CLRF    TRISD           ; Display port is output

        BANKSEL PORTA           ; Select bank 0
        CLRF    PORTD           ; Clear display outputs
        CLRF    HiReg           ; select memory page 0
        CLRF    LoReg           ; select first location
        CALL    inimem          ; initialise serial memory
        CALL    inid            ; Initialise the display

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

start   CLRW                    ; Select AN0 input
         BSF     PORTB,5
        CALL    adin            ; read analogue input
        CALL    condec          ; convert to decimal
        CALL    putdec          ; display input
        CALL    store           ; store input in memory

        BCF     PORTB,5
         CALL   putkey          ; Fixed message
        CALL    keyin           ; scan keypad
        CALL    send            ; display key
        GOTO    start           ; and again
```

BASE2 Source Code

```
;-------------------------------------------------------------
; SUBROUTINES
;-------------------------------------------------------------
; Display input test voltage on top line of LCD
;-------------------------------------------------------------

putdec  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 ---------------------------------

        MOVLW   030             ; load ASCII offset
        ADDWF   Huns            ; convert hundreds to ASCII
        ADDWF   Tens            ; convert tens to ASCII
        ADDWF   Ones            ; convert ones to ASCII

; Display voltage on line 1 -------------------------------

        CALL    volmes          ; Display text on line 1

        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

        RETURN                  ; done

; Store voltage in serial memory --------------------------

store   BSF     SSPCON,SSPEN    ; Enable memory port
        MOVF    ADRESH,W        ; Get voltage code
        MOVWF   SenReg          ; Load it to write
        CALL    writmem         ; Write it to memory
        INCF    LoReg           ; Next location
        BCF     SSPCON,SSPEN    ; Disable memory port
        RETURN                  ; done
```

```
;-------------------------------------------------------------
; Display key input on bottom line of LCD
;-------------------------------------------------------------

putkey  BCF     Select,RS       ; set display command mode
        MOVLW   0C0             ; code to home cursor
        CALL    send            ; output it to display
        BSF     Select,RS       ; and restore data mode
        CALL    keymes
        RETURN                  ; done

;-------------------------------------------------------------
; Display fixed messages
;-------------------------------------------------------------

volmes  CLRF    Tabin           ; Zero table pointer
next1   MOVF    Tabin,W         ; Load table pointer
        CALL    mess1           ; Get next character
        MOVWF   Temp            ; Test data...
        MOVF    Temp,F          ; ..for zero
        BTFSC   STATUS,Z        ; Last letter done?
        RETURN                  ; yes - next block
        CALL    send            ; no - display it
        INCF    Tabin           ; Point to next letter
        GOTO    next1           ; and get it

; -------------------------------------------------------------

keymes  CLRF    Tabin           ; Zero table pointer
next2   MOVF    Tabin,W         ; Load table pointer
        CALL    mess2           ; Get next character
        MOVWF   Temp            ; Test data...
        MOVF    Temp,F          ; ..for zero
        BTFSC   STATUS,Z        ; Last letter done?
        RETURN                  ; yes - next block
        CALL    send            ; no - display it
        INCF    Tabin           ; Point to next letter
        GOTO    next2           ; and get it

;-------------------------------------------------------------
; Text strings for fixed messages
;-------------------------------------------------------------

mess1   ADDWF   PCL             ; Set table pointer
        DT      "Volts = ",0    ; Text for display

mess2   ADDWF   PCL             ; Set table pointer
        DT      "Key = ",0      ; Text for display
```

BASE2 Source Code

```
;-------------------------------------------------------
; INCLUDED ROUTINES
;-------------------------------------------------------
; KEYPAD DRIVER
;       Scans 3x4 key pad once
;       Returns keynumber 1-12 with 0 = no key
;       CALL keyin, returns with key in W
;       * = 10, 0 = 11, # = 12

        INCLUDE "KEYPAD2.INC"
;-------------------------------------------------------
; LCD DRIVER
;       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)
;
        INCLUDE"LCD2.INC"
;
;-------------------------------------------------------
; Convert 8 bits to 3 digit decimal
;
;       Receives 8-bits in W
;       Returns BCD diits in 'huns','tens','ones'
;
        INCLUDE"CONDEC2.INC";
;
;-------------------------------------------------------
; Read selected analogue input
;
;       Receives channel number in W
;       Returns 8-bit input in W
;
        INCLUDE"ADIN2.INC"
;
;-------------------------------------------------------
; SERIAL MEMORY DRIVER
;       Write high address into 'HiReg' 00-3F
;       Write low address into 'LoReg' 00-FF
;       Load data send into 'SenReg'
;       Read data received from 'RecReg'
;
;       To initialise call 'inimem'
;       To write call 'writmem'
;       To read call 'readmem'
;
        INCLUDE "SERMEM2.INC"
;
;-------------------------------------------------------
        END                     ; of source code
;-------------------------------------------------------
```