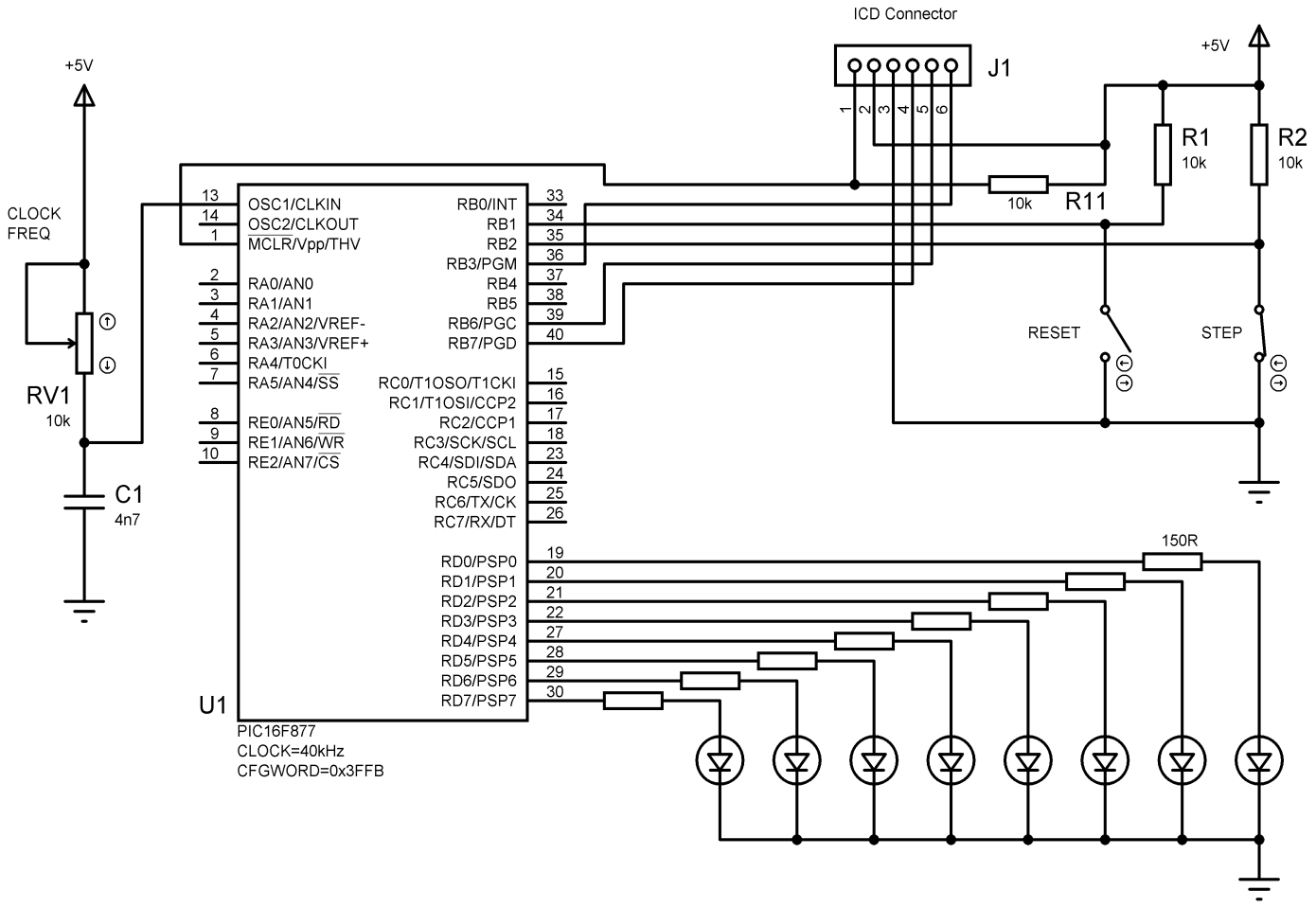
Interfacing PIC Microcontrollers

LED2 Schematic



- Application LED2 demonstrates the simplest forms of input and output
- Simple manual switched inputs produce logic high and low inputs
- The LEDs light up when the output is high. The PIC outputs can produce up to 25mA
- The simple RC clock components control the MCU operating frequency
- The ICD connector allows the PIC programming module to be attached for machine code downloading

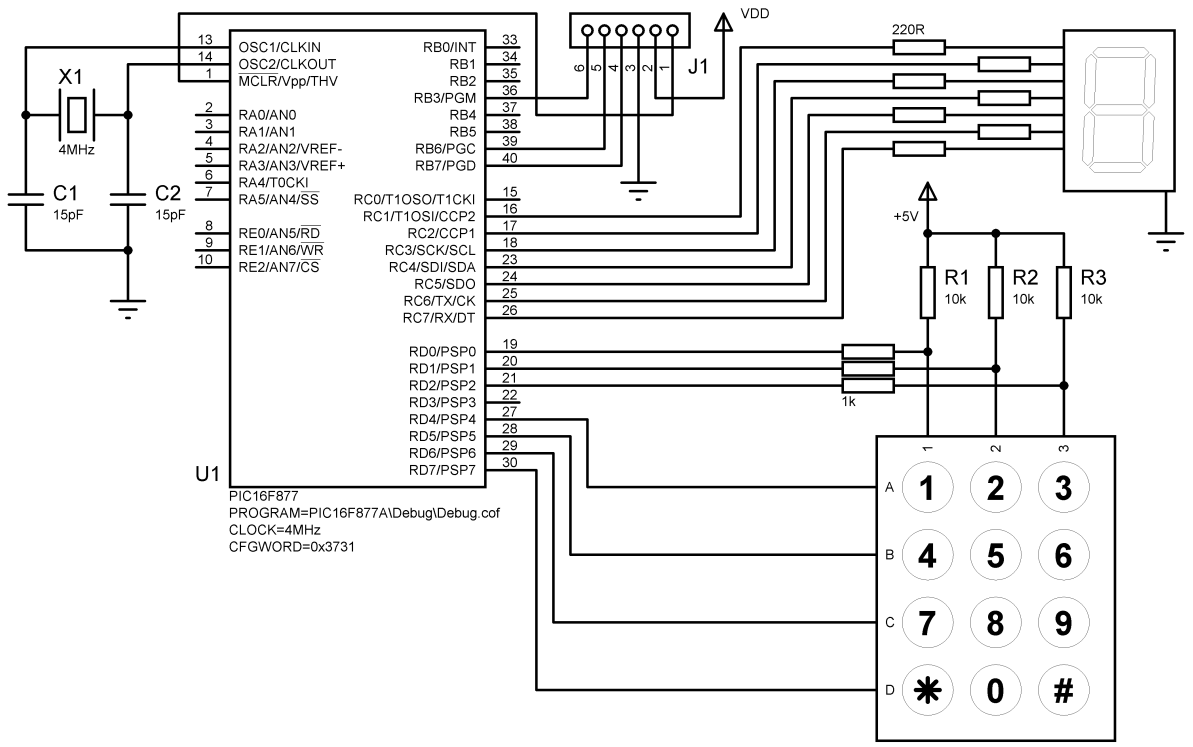
LED2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Source File:          LED2.ASM
;       Author:               MPB
;       Date:                 2-1-13
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Slow output binary count is stopped, started
;       and reset with push buttons.
;
;       Processor:      PIC 16F877A
;
;       Hardware:       Prototype
;       Clock:          RC = 40kHz
;       Inputs:         Port B: Push Buttons
;                       RB1, RB2 (active low)
;       Outputs:        Port D: LEDs (active high)
;
;       WDTimer:        Disabled
;       PUTimer:        Enabled
;       Interrupts:     Disabled
;       Code Protect:   Disabled
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

       PROCESSOR 16F877      ; Define MCU type
       __CONFIG 0x3733       ; Set config fuses

; Register Label Equates..................................

PORTB   EQU     06     ; Port B Data Register
PORTD   EQU     08     ; Port D Data Register
TRISD   EQU     88     ; Port B Direction Register
Timer   EQU     20     ; GPR used as delay counter

; Input Bit Label Equates ................................

Inres   EQU     1      ; 'Reset' input button = RD0
Inrun   EQU     2      ; 'Run' input button = RD1
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


         CODE    0            ; Program code start address

; Initialise Port B (Port A defaults to inputs)............

        BANKSEL  TRISD        ; Select bank 1
        MOVLW    b'00000000'  ; Port B Direction Code
        MOVWF    TRISD        ; Load the DDR code into F86
        BANKSEL  PORTD        ; Select bank 0
        GOTO     reset        ; Jump to main loop


; 'delay' subroutine .......................................

delay   MOVWF  Timer           ; Copy W to timer register
down    DECFSZ Timer           ; Decrement timer register
        GOTO   down            ; and repeat until zero
        RETURN                 ; Jump back to main program


; Start main loop ..........................................

reset   CLRF   PORTD          ; Clear Port B Data

start   BTFSS  PORTB,Inres    ; Test reset button
        GOTO   reset          ; and reset Port B if pressed
        BTFSC  PORTB,Inrun    ; Test run button
        GOTO   start          ; and repeat if n pressed

        INCF   PORTD           ; Increment output at Port B
        MOVLW  0FF             ; Delay count literal
        CALL   delay          ; Jump to subroutine 'delay'
        GOTO   start           ; Repeat main loop always

        END                    ; Terminate source code
```

Interfacing PIC Microcontrollers

KEYPAD2 Schematic



- Application KEYPAD2 demonstrates simple keypad and display interfacing
- The keypad has switches arranged in rows and columns, scanned by outputting a logic low on each row and reading the each column input in turn
- A suitable seven segment code is selected output to the LED display
- The crystal clock produces a more precise clock frequency
- No clock components are needed if using 16F887 when the internal clock is selected

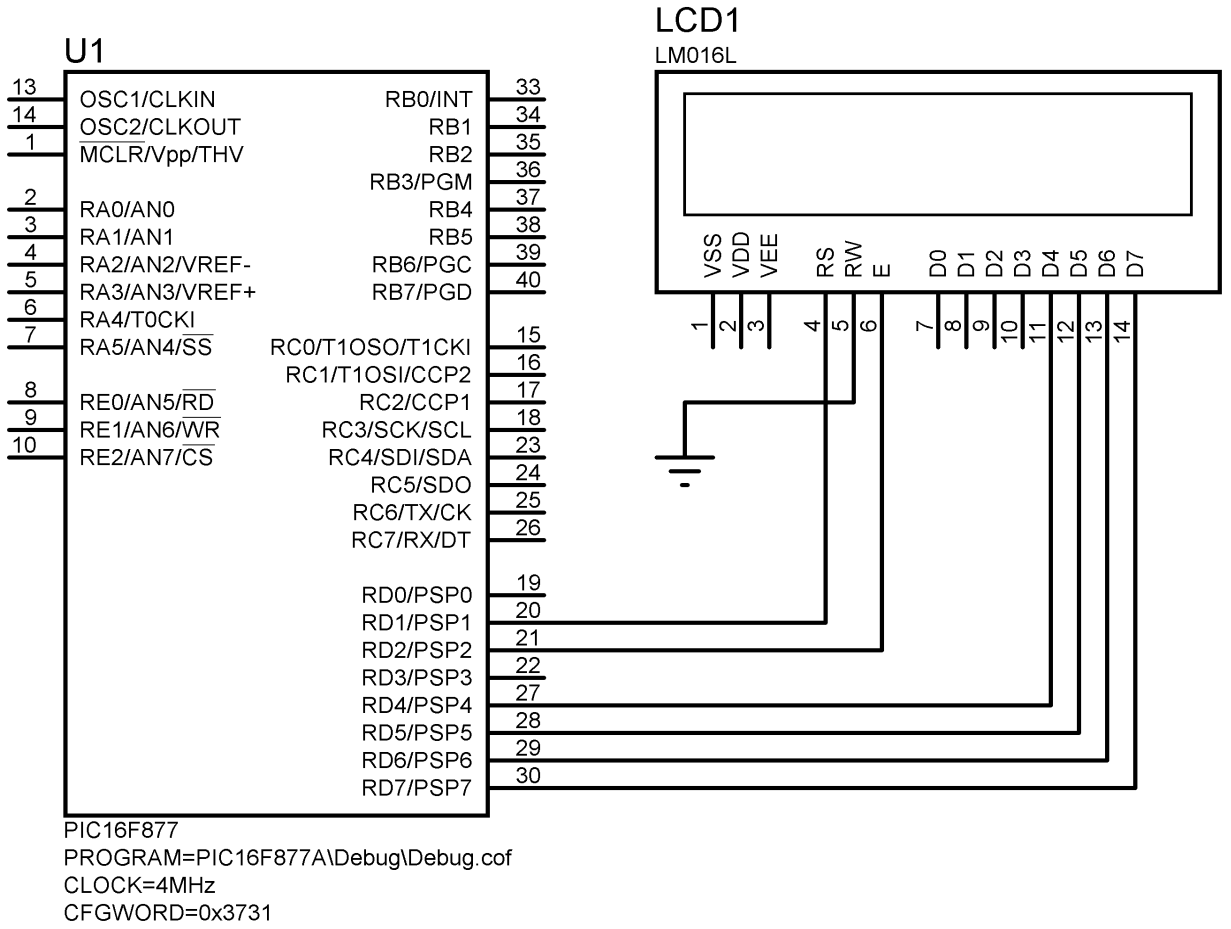
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       KEYPAD2.ASM    MPB    11-01-13
;
;       Reads keypad and shows digit on display
;       Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

               PROCESSOR 16F877

PCL     EQU     002             ; Program Counter
STATUS  EQU     003             ; Status register
PORTC   EQU     007             ; 7-Segment display
PORTD   EQU     008             ; 3x4 keypad
Timer   EQU     020             ; Delay counter

TRISC   EQU     087             ; Data direction
TRISD   EQU     088             ; registers

Key     EQU     021             ; Count of keys

; Code start.............................................

        CODE    0               ; code start
        NOP                     ; ICPD location

; Initialise ports.......................................

        BANKSEL TRISC           ; Display
        CLRW                    ; all outputs
        MOVWF   TRISC           ;
        MOVLW   B'00000111'     ; Keypad
        MOVWF   TRISD           ; bidirectional

        BANKSEL PORTC           ; Display off
        CLRF    PORTC           ; initially
        MOVLW   B'11111111'     ; Set all outputs
        MOVWF   PORTD           ; to keypad high
        GOTO    main            ; jump to main

; Subroutines............................................

delay   CLRF    Timer
count   DECFSZ  Timer
        GOTO    count
        RETURN

; Check a row of keys ...................................

row     INCF    Key             ; Count first key
        BTFSS   PORTD,0         ; Check key
        GOTO    found           ; and quit if on

        INCF    Key             ; and repeat
        BTFSS   PORTD,1         ; for second
        GOTO    found           ; key

        INCF    Key             ; and repeat
        BTFSS   PORTD,2         ; for third
        GOTO    found           ; key
        GOTO    next            ; go for next row

; Scan the keypad........................................

scan    CLRF    Key             ; Zero key count
        BSF     STATUS,0        ; Set Carry Flag
        MOVLW   B'11111111'     ; Set all outputs
        MOVWF   PORTD           ; to keypad high
        BCF     PORTD,4         ; Select first row
        GOTO    row             ; Check first row

next    BSF     PORTD,3         ; Set fill bit
        RLF     PORTD           ; Select next row
        BTFSC   STATUS,0        ; 0 into carry flag?
        GOTO    row             ; if not, next row
        GOTO    scan            ; if so, start again

found   CALL    delay           ; debounce delay
        BTFSS   PORTD,0         ; await button release
        GOTO    found
        BTFSS   PORTD,1
        GOTO    found
        BTFSS   PORTD,2
        GOTO    found
        RETURN                  ; quit with key count
```

KEYPAD2 Source Code (continued)

```
; Display code table.....................................

table   MOVF    Key,W          ; Get key count
        ADDWF   PCL            ; and calculate jump
        NOP                    ; into table
        RETLW   B'00001100'    ; Code for '1'
        RETLW   B'10110110'    ; Code for '2'
        RETLW   B'10011110'    ; Code for '3'
        RETLW   B'11001100'    ; Code for '4'
        RETLW   B'11011010'    ; Code for '5'
        RETLW   B'11111000'    ; Code for '6'
        RETLW   B'00001110'    ; Code for '7'
        RETLW   B'11111110'    ; Code for '8'
        RETLW   B'11001110'    ; Code for '9'
        RETLW   B'10010010'    ; Code for '*'
        RETLW   B'01111110'    ; Code for '0'
        RETLW   B'11101100'    ; Code for '#'

; Output display code....................................

show    CALL    table          ; Get display code
        MOVWF   PORTC          ; and show it
        RETURN

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

; Read keypad & display....

main    CALL    scan           ; Get key number
        CALL    show           ; and display it
        GOTO    main           ; and repeat
        END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Interfacing PIC Microcontrollers

LCD2 Schematic



- Application LCD2 demonstrates the connection of an LCD alphanumeric display
- It can operate with a 4 or 8 bit input. In this case, the ASCII text codes are input as 2x4 bit nibbles
- The program generates a fixed message and a calculated incrementing count
- The simulation does not need the clock components to be included in the schematic

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       LCD2.ASM       MPB     11-01-13
;
;       Outputs fixed and variable characters
;       to 16x2 LCD in 4-bit mode
;        Updated for Proteus VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

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

        INCLUDE "P16F877A.INC"; Standard register labels

Timer1  EQU     20              ; 1ms count register
TimerX  EQU     21              ; Xms count register
Var     EQU     22              ; Output variable
Point   EQU     23              ; Program table pointer
Select  EQU     24              ; Copy of RS bit
OutCod  EQU     25              ; Temp store for output code

RS      EQU     1               ; Register select output bit
E       EQU     2               ; Display enable

; Program code ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CODE    0               ; Place machine code
        NOP                     ; for ICD mode

        BANKSEL TRISD           ; Select bank 1
        CLRW                    ; All outputs
        MOVWF   TRISD           ; Initialise display port
        BANKSEL PORTD           ; Select bank 0
        CLRF    PORTD           ; Clear display outputs

        GOTO    Start           ; Jump to main program
```

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

; 1ms delay with 1us cycle time (1000 cycles).............

Onems   MOVLW   D'249'          ; Count for 1ms delay
        MOVWF   Timer1          ; Load count
Loop1   NOP                     ; Pad for 4 cycle loop
        DECFSZ  Timer1          ; Count
        GOTO    Loop1           ; until Z
        RETURN                  ; and finish


; Delay Xms, X received in W .............................

Xms     MOVWF   TimerX          ; Count for X ms
LoopX   CALL    Onems           ; Delay 1ms
        DECFSZ  TimerX          ; Repeat X times
        GOTO    LoopX           ; until Z
        RETURN                  ; and finish


; Generate data/command clock siganl E ..................

PulseE  BSF     PORTD,E         ; Set E high
        CALL    Onems           ; Delay 1ms
        BCF     PORTD,E         ; Reset E low
        CALL    Onems           ; Delay 1ms
        RETURN                  ; done


; Send a command byte in two nibbles from RB4 - RB7 ........

Send    MOVWF   OutCod          ; Store output code
        ANDLW   0F0             ; Clear low nybble
        MOVWF   PORTD           ; Output high nybble
        BTFSC   Select,RS       ; Test RS bit
        BSF     PORTD,RS        ; and set for data
        CALL    PulseE          ; and clock display register
        CALL    Onems           ; wait 1ms for display

        SWAPF   OutCod          ; Swap low and high nybbles
        MOVF    OutCod,W        ; Retrieve output code
        ANDLW   0F0             ; Clear low nybble
        MOVWF   PORTD           ; Output low nybble
        BTFSC   Select,RS       ; Test RS bit
        BSF     PORTD,RS        ; and set for data
        CALL    PulseE          ; and clock display register
        CALL    Onems           ; wait 1ms for display
        RETURN                  ; done
```

LCD2 Source Code (continued)

```
;       Table of fixed characters to send ..................

Line1   ADDWF   PCL             ; Modify program counter
        RETLW   'C'             ; Pointer = 0
        RETLW   'O'             ; Pointer = 1
        RETLW   'N'             ; Pointer = 2
        RETLW   'S'             ; Pointer = 3
        RETLW   'T'             ; Pointer = 4
        RETLW   ':'             ; Pointer = 5
        RETLW   '0'             ; Pointer = 6
        RETLW   '1'             ; Pointer = 7
        RETLW   '2'             ; Pointer = 8
        RETLW   '3'             ; Pointer = 9
        RETLW   '4'             ; Pointer = 10
        RETLW   '5'             ; Pointer = 11
        RETLW   '6'             ; Pointer = 12
        RETLW   '7'             ; Pointer = 13
        RETLW   '8'             ; Pointer = 14
        RETLW   '9'             ; Pointer = 15

Line2   ADDWF   PCL             ; Modify program counter
        RETLW   'V'             ; Pointer = 0
        RETLW   'A'             ; Pointer = 1
        RETLW   'R'             ; Pointer = 2
        RETLW   'I'             ; Pointer = 3
        RETLW   'A'             ; Pointer = 4
        RETLW   'B'             ; Pointer = 5
        RETLW   'L'             ; Pointer = 6
        RETLW   'E'             ; Pointer = 7
        RETLW   ' '             ; Pointer = 8
        RETLW   '='             ; Pointer = 9
        RETLW   ' '             ; Pointer = 10
```

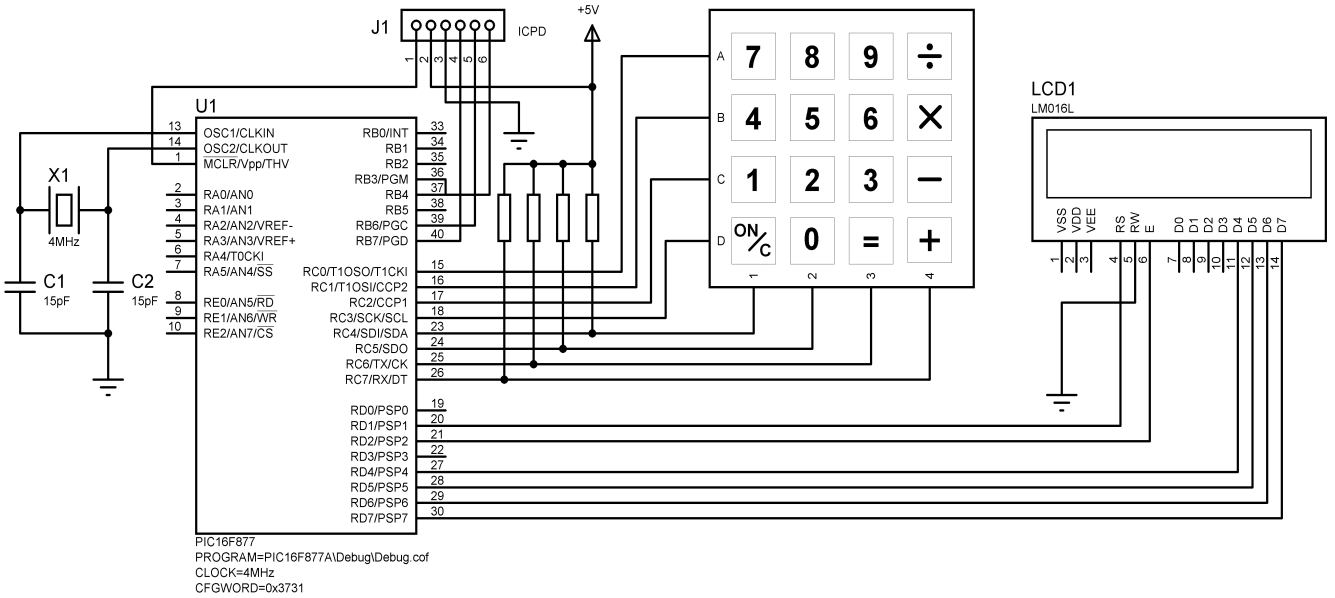
```
; Initialise the display.................................

Init    MOVLW   D'100'          ; Load count for 100ms delay
        CALL    Xms             ; and wait for display start
        MOVLW   0F0             ; Mask for select code
        MOVWF   Select          ; High nybble not masked

        MOVLW   0x30            ; Load initial nibble
        MOVWF   PORTD           ; and output it to display
        CALL    PulseE          ; Latch initial code
        MOVLW   D'5'            ; Set delay 5ms
        CALL    Xms             ; and wait
        CALL    PulseE          ; Latch initial code again
        CALL    Onems           ; Wait 1ms
        CALL    PulseE          ; Latch initial code again
        BCF     PORTD,4         ; Set 4-bit mode
        CALL    PulseE          ; Latch it

        MOVLW   0x28            ; Set 4-bit mode, 2 lines
        CALL    Send            ; and send code
        MOVLW   0x08            ; Switch off display
        CALL    Send            ; and send code
        MOVLW   0x01            ; Code to clear display
        CALL    Send            ; and send code
        MOVLW   0x06            ; Enable cursor auto inc
        CALL    Send            ; and send code
        MOVLW   0x80            ; Zero display address
        CALL    Send            ; and send code
        MOVLW   0x0C            ; Turn on display
        CALL    Send            ; and send code

        RETURN                  ; Done
```

LCD2 Source Code (continued)

```
; Send the fixed message to the display.....................

OutMes  CLRF    Point           ; Reset table pointer
        BSF     Select,RS       ; Select data mode

Mess1   MOVF    Point,W         ; and load it
        CALL    Line1           ; Get ASCII code from table
        CALL    Send            ; and do it
        INCF    Point           ; point to next character
        MOVF    Point,W         ; and load the pointer
        SUBLW   D'16'           ; check for last table item
        BTFSS   STATUS,Z        ; and finish if 16 done
        GOTO    Mess1           ; Output character code

        MOVLW   0xC0            ; Move cursor to line 2
        BCF     Select,RS       ; Select command mode
        CALL    Send            ; and send code
        CLRF    Point           ; Reset table pointer
Mess2   MOVF    Point,W         ; and load it
        CALL    Line2           ; Get fixed character
        BSF     Select,RS       ; Select data mode
        CALL    Send            ; and send code
        INCF    Point           ; next character
        MOVF    Point,W         ; Reload pointer
        SUBLW   D'11'           ; and check for last
        BTFSS   STATUS,Z        ; Skip if last
        GOTO    Mess2           ; or send next
        RETURN                  ; done
```

```
; Output variable count to display (0-9) endlessly..........

OutVar  CLRF    Var             ; Clear variable number
        MOVLW   0X30            ; Load offset to be added
        ADDWF   Var             ; to make ASCII code (30-39)

Next    MOVF    Var,W           ; Load the code
        BSF     Select,RS       ; Select data mode
        CALL    Send            ; and send code

        MOVLW   0xCB            ; code to move cursor back
        BCF     Select,RS       ; Select command mode
        CALL    Send            ; and send code
        MOVLW   D'250'          ; Load count to wait 250ms
        CALL    Xms             ; so numbers are visible

        INCF    Var             ; Next number
        MOVF    Var,W           ; Load number
        SUBLW   0x3A            ; Check for last (10=A)
        BTFSS   STATUS,Z        ; and skip if last
        GOTO    Next            ; or do next number
        GOTO    OutVar          ; Repeat from number Z


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

Start   CALL    Init            ; Initialise the display
        CALL    OutMes          ; Display fixed characters
        GOTO    OutVar          ; Display an endless count

        END                     ; of source code
```

CALC2 Schematic



- This application demonstrates the basic principles of numerical processing using the keypad and LCD display
- It accepts two single digit integer inputs and performs an add, subtract, multiply or divide operation
- A two digit result is provided, with the division producing a result with an integer remainder
- The LCD driver routine is included as a separate source code file derived from the application LCD2 above

CALC2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       CALC2.ASM      MPB     12-01-13
;
;       Simple calculator
;       Single digit input, two digit results
;       Integer handling only
;        Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877
        __CONFIG 0x3731         ; Clock = XT 4MHz

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

        INCLUDE "P16F877A.INC"

Char    EQU     30      ; Display character code
Num1    EQU     31      ; First number input
Num2    EQU     32      ; Second number input
Result  EQU     33      ; Calculated result
Oper    EQU     34      ; Operation code store
Temp    EQU     35      ; Temporary register for subtract
Kcount  EQU     36      ; Count of keys hit
Kcode   EQU     37      ; ASCII code for key
Msd     EQU     38      ; Most significant digit of result
Lsd     EQU     39      ; Least significant digit of result
Kval    EQU     40      ; Key numerical value

RS      EQU     1       ; Register select output bit
E       EQU     2       ; Display data strobe

; Program begins ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

        BANKSEL TRISC           ; Select bank 1
        MOVLW   B'11110000'     ; Keypad direction code
        MOVWF   TRISC           ;
        CLRF    TRISD           ; Display port is output

        BANKSEL PORTC           ; Select bank 0
        MOVLW   0FF             ;
        MOVWF   PORTC           ; Set keypad outputs high
        CLRF    PORTD           ; Clear display outputs
        GOTO    start           ; Jump to main program

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

start   CALL    inid            ; Initialise the display
        MOVLW   0x80            ; position to home cursor
        BCF     Select,RS       ; Select command mode
        CALL    send            ; and send code

        CLRW    Char            ; ASCII = 0
        CLRW    Kval            ; Key value = 0
        CLRW    DFlag           ; Digit flags = 0

scan    CALL    keyin           ; Scan keypad
        MOVF    Char,1          ; test character code
        BTFSS   STATUS,Z        ; key pressed?
        GOTO    keyon           ; yes - wait for release
        GOTO    scan            ; no - scan again

keyon   MOVF    Char,W          ; Copy..
        MOVWF   Kcode           ; ..ASCIIcode
        MOVLW   D'50'           ; delay for..
        CALL    xms             ; ..50ms debounce

wait    CALL    keyin           ; scan keypad again
        MOVF    Char,1          ; test character code
        BTFSS   STATUS,Z        ; key pressed?
        GOTO    wait            ; no - rescan
        CALL    disout          ; yes - show symbol

        INCF    Kcount          ; inc count..
        MOVF    Kcount,W        ; ..of keys pressed
        ADDWF   PCL             ; jump into table
        NOP
        GOTO    first           ; process first key
        GOTO    scan            ; get operation key
        GOTO    second          ; process second symbol
        GOTO    calc            ; calculate result
        GOTO    clear           ; clear display

first   MOVF    Kval,W          ; store..
        MOVWF   Num1            ; first num
        GOTO    scan            ; and get op key

second  MOVF    Kval,W          ; store..
        MOVWF   Num2            ; second number
        GOTO    scan            ; and get equals key
```

CALC2 Source Code (continued)

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

; Include LCD driver routine

        INCLUDE "LCD.INC"


; Scan
keypad ..............................................

keyin   MOVLW   00F             ; deselect..
        MOVWF   PORTC           ; ..all rows
        BCF     PORTC,0         ; select row A
        CALL    onems           ; wait output stable

        BTFSC   PORTC,4         ; button 7?
        GOTO    b8              ; no
        MOVLW   '7'             ; yes
        MOVWF   Char            ; load key code
        MOVLW   07              ; and
        MOVWF   Kval            ; key value
        RETURN

b8      BTFSC   PORTC,5         ; button 8?
        GOTO    b9              ; no
        MOVLW   '8'             ; yes
        MOVWF   Char
        MOVLW   08
        MOVWF   Kval
        RETURN

b9      BTFSC   PORTC,6         ; button 9?
        GOTO    bd              ; no
        MOVLW   '9'             ; yes
        MOVWF   Char
        MOVLW   09
        MOVWF   Kval
        RETURN

bd      BTFSC   PORTC,7         ; button /?
        GOTO    rowb            ; no
        MOVLW   '/'             ; yes
        MOVWF   Char            ; store key code
        MOVWF   Oper            ; store operator symbol
        RETURN

rowb    BSF     PORTC,0         ; select row B
        BCF     PORTC,1
        CALL    onems

        BTFSC   PORTC,4         ; button 4?
        GOTO    b5              ; no
        MOVLW   '4'             ; yes
        MOVWF   Char
        MOVLW   04
        MOVWF   Kval
        RETURN

b5      BTFSC   PORTC,5         ; button 5?
        GOTO    b6              ; no
        MOVLW   '5'             ; yes
        MOVWF   Char
        MOVLW   05
        MOVWF   Kval
        RETURN

b6      BTFSC   PORTC,6         ; button 6?
        GOTO    bm              ; no
        MOVLW   '6'             ; yes
        MOVWF   Char
        MOVLW   06
        MOVWF   Kval
        RETURN

bm      BTFSC   PORTC,7         ; button x?
        GOTO    rowc            ; no
        MOVLW   'x'             ; yes
        MOVWF   Char
        MOVWF   Oper
        RETURN
```

CALC2 Source Code (continued)

```
rowc    BSF     PORTC,1         ; select row C
        BCF     PORTC,2
        CALL    onems

        BTFSC   PORTC,4         ; button 1?
        GOTO    b2              ; no
        MOVLW   '1'             ; yes
        MOVWF   Char
        MOVLW   01
        MOVWF   Kval
        RETURN

b2      BTFSC   PORTC,5         ; button 2?
        GOTO    b3              ; no
        MOVLW   '2'             ; yes
        MOVWF   Char
        MOVLW   02
        MOVWF   Kval
        RETURN

b3      BTFSC   PORTC,6         ; button 3?
        GOTO    bs              ; no
        MOVLW   '3'             ; yes
        MOVWF   Char
        MOVLW   03
        MOVWF   Kval
        RETURN

bs      BTFSC   PORTC,7         ; button -?
        GOTO    rowd            ; no
        MOVLW   '-'             ; yes
        MOVWF   Char
        MOVWF   Oper
        RETURN
```

```
rowd    BSF     PORTC,2         ; select row D
        BCF     PORTC,3
        CALL    onems

        BTFSC   PORTC,4         ; button C?
        GOTO    b0              ; no
        MOVLW   'c'             ; yes
        MOVWF   Char
        MOVWF   Oper
        RETURN

b0      BTFSC   PORTC,5         ; button 0?
        GOTO    be              ; no
        MOVLW   '0'             ; yes
        MOVWF   Char
        MOVLW   00
        MOVWF   Kval
        RETURN

be      BTFSC   PORTC,6         ; button =?
        GOTO    bp              ; no
        MOVLW   '='             ; yes
        MOVWF   Char
        RETURN

bp      BTFSC   PORTC,7         ; button +?
        GOTO    done            ; no
        MOVLW   '+'             ; yes
        MOVWF   Char
        MOVWF   Oper
        RETURN

done    BSF     PORTC,3         ; clear last row
        CLRF    Char            ; character code = 0
        RETURN
```

CALC2 Source Code (continued)

```
; Write display ......................................

disout  MOVF    Kcode,W         ; Load the code
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send code
        RETURN

; Process operations .................................

calc    MOVF    Oper,W          ; check for add
        MOVWF   Temp            ; load input op code
        MOVLW   '+'             ; load plus code
        SUBWF   Temp            ; compare
        BTFSC   STATUS,Z        ; and check if same
        GOTO    add             ; yes, jump to op

        MOVF    Oper,W          ; check for subtract
        MOVWF   Temp
        MOVLW   '-'
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    sub

        MOVF    Oper,W          ; check for multiply
        MOVWF   Temp
        MOVLW   'x'
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    mul

        MOVF    Oper,W          ; check for divide
        MOVWF   Temp
        MOVLW   '/'
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    div
        GOTO    scan            ; rescan if key invalid
```

```
; Calclate results from 2 input numbers ...................

add     MOVF    Num1,W          ; get first number
        ADDWF   Num2,W          ; add second
        MOVWF   Result          ; and store result
        GOTO    outres          ; display result


sub     BSF     STATUS,C        ; Negative detect flag
        MOVF    Num2,W          ; get first number
        SUBWF   Num1,W          ; subtract second
        MOVWF   Result          ; and store result

        BTFSS   STATUS,C        ; answer negative?
        GOTO    minus           ; yes, minus result
        GOTO    outres          ; display result

minus   MOVLW   '-'             ; load minus sign
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send symbol

        COMF    Result          ; invert all bits
        INCF    Result          ; add 1
        GOTO    outres          ; display result


mul     MOVF    Num1,W          ; get first number
        CLRF    Result          ; total to Z
add1    ADDWF   Result          ; add to total
        DECFSZ  Num2            ; num2 times and
        GOTO    add1            ; repeat if not done
        GOTO    outres          ; done, display result
```

CALC2 Source Code (continued)

```
div     CLRF    Result          ; total to Z
        MOVF    Num2,W          ; get divisor
        BCF     STATUS,C        ; set C flag
sub1    INCF    Result          ; count loop start
        SUBWF   Num1            ; subtract
        BTFSS   STATUS,Z        ; exact answer?
        GOTO    neg             ; no
        GOTO    outres          ; yes, display answer
neg     BTFSC   STATUS,C        ; gone negative?
        GOTO    sub1            ; no - repeat
        DECF    Result          ; correct the result
        MOVF    Num2,W          ; get divisor
        ADDWF   Num1            ; calc remainder

        MOVF    Result,W        ; load result
        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send result

        MOVLW   'r'             ; indicate remainder
        CALL    send
        MOVF    Num1,W
        ADDLW   030             ; convert to ASCII
        CALL    send
        GOTO    scan

; Convert binary to BCD ..................................

outres  MOVF    Result,W        ; load result
        MOVWF   Lsd             ; into low digit store
        CLRF    Msd             ; high digit = 0
        BSF     STATUS,C        ; set C flag
        MOVLW   D'10'           ; load 10

again   SUBWF   Lsd             ; sub 10 from result
        INCF    Msd             ; inc high digit
        BTFSC   STATUS,C        ; check if negative
        GOTO    again           ; no, keep going
        ADDWF   Lsd             ; yes, add 10 back
        DECF    Msd             ; inc high digit
```

```
; display 2 digit BCD result ............................

        MOVF    Msd,W           ; load high digit result
        BTFSC   STATUS,Z        ; check if Z
        GOTO    lowd            ; yes, dont display Msd

        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send Msd

lowd    MOVF    Lsd,W           ; load low digit result
        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send Msd

        GOTO    scan            ; scan for clear key


; Restart ................................................

clear   MOVLW   01              ; code to clear display
        BCF     Select,RS       ; Select data mode
        CALL    send            ; and send code
        CLRF    Kcount          ; reset count of keys
        GOTO    scan            ; and rescan keypad


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

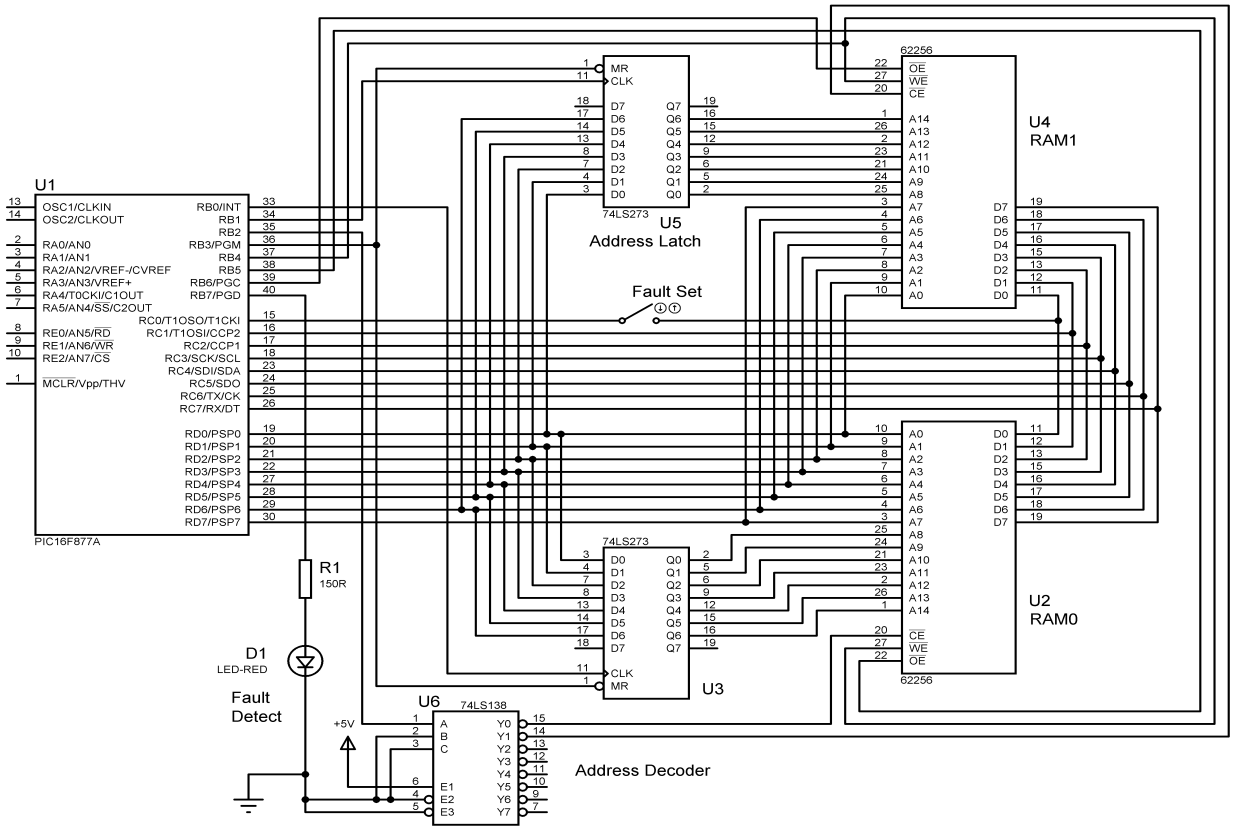
CALC2 Source Code (continued)

```
;-----------------------------------------------------------------
;       LCD.INC        MPB     19-12-05
;
;       Include file to operate 16x2 LCD display
;       Uses GPR 70 - 75
;
;       Final version
;-----------------------------------------------------------------

Timer1  EQU     70              ; 1ms count register
TimerX  EQU     71              ; Xms count register
Var     EQU     72              ; Output variable
Point   EQU     73              ; Program table pointer
Select  EQU     74              ; Used to set or clear RS bit
OutCod  EQU     75              ; Temp store for output code

RS      EQU     1               ; Register select output bit
E       EQU     2               ; Enable display input


;-----------------------------------------------------------------
;       1ms delay with 1us cycle time (1000 cycles)
;-----------------------------------------------------------------
onems   MOVLW   D'249'          ; Count for 1ms delay
        MOVWF   Timer1          ; Load count
loop1   NOP                     ; Pad for 4 cycle loop
        DECFSZ  Timer1          ; Count
        GOTO    loop1           ; until Z
        RETURN                  ; and finish

;-----------------------------------------------------------------
;       Delay Xms
;       Receives count in W, uses Onems
;-----------------------------------------------------------------
xms     MOVWF   TimerX          ; Count for X ms
loopX   CALL    onems           ; Delay 1ms
        DECFSZ  TimerX          ; Repeat X times
        GOTO    loopX           ; until Z
        RETURN                  ; and finish

;-----------------------------------------------------------------
;       Generate data/command clock siganl E
;-----------------------------------------------------------------
pulseE  BSF     PORTD,E         ; Set E high
        CALL    onems           ; Delay 1ms
        BCF     PORTD,E         ; Reset E low
        CALL    onems           ; Delay 1ms
        RETURN                  ; done
```

```
;-----------------------------------------------------------------
;       Send a command byte in two nibbles from RB4 - RB7
;       Receives command in W, uses PulseE and Onems
;-----------------------------------------------------------------
send    MOVWF   OutCod          ; Store output code
        ANDLW   0F0             ; Clear low nybble
        MOVWF   PORTD           ; Output high nybble
        BTFSC   Select,RS       ; Test RS bit
        BSF     PORTD,RS        ; and set for data
        CALL    pulseE          ; and clock display register
        CALL    onems           ; wait 1ms for display

        SWAPF   OutCod          ; Swap low and high nybbles
        MOVF    OutCod,W        ; Retrieve output code
        ANDLW   0F0             ; Clear low nybble
        MOVWF   PORTD           ; Output low nybble
        BTFSC   Select,RS       ; Test RS bit
        BSF     PORTD,RS        ; and set for data
        CALL    pulseE          ; and clock display register
        CALL    onems           ; wait 1ms for display
        RETURN                  ; done

;-----------------------------------------------------------------
;       Initialise the display
;       Uses Send
;-----------------------------------------------------------------
inid    MOVLW   D'100'          ; Load count for 100ms delay
        CALL    xms             ; and wait for display start
        MOVLW   0F0             ; Mask for select code
        MOVWF   Select          ; High nybble not masked

        MOVLW   0x30            ; Load initial nibble
        MOVWF   PORTD           ; and output it to display
        CALL    pulseE          ; Latch initial code
        MOVLW   D'5'            ; Set delay 5ms
        CALL    xms             ; and wait
        CALL    pulseE          ; Latch initial code again
        CALL    onems           ; Wait 1ms
        CALL    pulseE          ; Latch initial code again
        BCF     PORTD,4         ; Set 4-bit mode
        CALL    pulseE          ; Latch it

        MOVLW   0x28            ; Set 4-bit mode, 2 lines
        CALL    send            ; and send code
        MOVLW   0x08            ; Switch off display
        CALL    send            ; and send code
        MOVLW   0x01            ; Code to clear display
        CALL    send            ; and send code
        MOVLW   0x06            ; Enable cursor auto inc
        CALL    send            ; and send code
        MOVLW   0x80            ; Zero display address
        CALL    send            ; and send code
        MOVLW   0x0C            ; Turn on display
        CALL    send            ; and send code

        RETURN                  ; Done

;-----------------------------------------------------------------
```

Interfacing PIC Microcontrollers

PARMEM2 Schematic



- This application demonstrates parallel memory expansion
- A pair of 32k RAM chips are used that have 8-bit data I/O ports and 15-bit address inputs
- Address latches are used to store the address high byte so that only 8 PIC data lines are needed
- The address decoder selects the RAM chip to be accessed
- A switch in the LSB data line simulated a faulty data access

PARMEM2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       PARMEM.ASM      MPB     Ver:2.0         18-02-13
;...............................................................
;
;       Parallel memory system
;       PIC 16F877A operates with expansion memory
;       RAM = 2 x 62256 32kb
;        Updated for VSM v8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A     ; define MPU
        __CONFIG 0x3731               ; XT clock
        INCLUDE "P16F877A.INC"; Standard register labels

ConReg  EQU     06      ; Port B = Control Register
DatReg  EQU     07      ; Port C = Data Register
AddReg  EQU     08      ; Port D = Address Register
HiAdd   EQU     20      ; High address store

CLK0    EQU     0       ; RAM0 address buffer clock
CLK1    EQU     1       ; RAM1 address buffer clock
SelRAM  EQU     2       ; RAM select bit
ResHi   EQU     3       ; High address reset bit
WritEn  EQU     4       ; Write enable bit
OutEn0  EQU     5       ; Output enable bit RAM0
OutEn1  EQU     6       ; Output enable bit RAM1
LED     EQU     7       ; Memory error indicator

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

         CODE    0             ; Place machine code

        NOP                    ; Required for ICD mode

        BANKSEL TRISB          ; Select bank 1
        CLRF    TRISB          ; Control output bits
        CLRF    TRISC          ; Data bus initially output
        CLRF    TRISD          ; Address bus output

        BANKSEL AddReg         ; Select bank 0
        CLRF    DatReg         ; Clear outputs initially
        CLRF    AddReg         ; Clear outputs initially
        BCF     ConReg,CLK0    ; RAM0 address buffer clock
        BCF     ConReg,CLK1    ; RAM1 address buffer clock
        BCF     ConReg,SelRAM  ; Select RAM0 initially
        BCF     ConReg,ResHi   ; Reset high address latches
        BSF     ConReg,OutEn0  ; Disable output enable RAM0
        BSF     ConReg,OutEn1  ; Disable output enable RAM1
        BSF     ConReg,WritEn  ; Disable write enable bit
        BCF     ConReg,LED     ; Switch of error indicator
```

```
; MAIN LOOP ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

start   CALL    write           ; test write to memory
        CALL    read            ; test read from memory
        SLEEP                   ; shut down

; Write checkerboard pattern to both RAMs ;;;;;;;;;;;;;;;;;;;;;;

write   BSF     ConReg,ResHi    ; Enable address latches
nexwrt  MOVLW   055             ; checkerboard test data
        MOVWF   DatReg          ; output on data bus
        CALL    store           ; and write to RAM
        MOVLW   0AA             ; checkerboard test data
        MOVWF   DatReg          ; output on data bus
        CALL    store           ; and write to RAM
        BTFSS   ConReg,ResHi    ; all done?
        RETURN                  ; yes - quit
        GOTO    nexwrt          ; no - next byte pair

; Check data stored ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

read    NOP                     ; required for label
        BANKSEL TRISC           ; select bank 1
        MOVLW   0FF             ; all inputs..
        MOVWF   TRISC           ; ..at Port C

        BANKSEL ConReg          ; select default bank 0
        BSF     ConReg,ResHi    ; Enable address latches
        BCF     ConReg,SelRAM   ; select RAM0
        BCF     ConReg,OutEn0   ; set RAM0 for output
        CALL    nexred          ; check data in RAM0
        BSF     ConReg,SelRAM   ; select RAM0
        BCF     ConReg,OutEn1   ; set RAM0 for output
        CALL    nexred          ; check data in RAM0
        RETURN                  ; all done

; Load test data and check data ...............................

nexred  MOVLW   055             ; load even data byte
        CALL    test            ; check data
        MOVLW   0AA             ; load odd data byte
        CALL    test            ; check data
        BTFSS   ConReg,ResHi    ; all done?
        RETURN                  ; yes - quit
        GOTO    nexred          ; no - next byte pair
```

PARMEM2 Source Code (continued)

```
; Write data to RAM .....................................

store   BCF     ConReg,SelRAM  ; no - Select RAM0
        BCF     ConReg,WritEn  ; negative pulse ..
        BSF     ConReg,WritEn
        BSF     ConReg,SelRAM  ; Select RAM1
        BCF     ConReg,WritEn  ; negative pulse ..
        BSF     ConReg,WritEn  ; ..write bit
        INCF    AddReg         ; next address
        BTFSC   STATUS,Z       ; last address?
        CALL    inchi          ; yes - increment high address
        RETURN                 ; no - next byte

; Test memory data ......................................

test    MOVF    DatReg,F       ; read data
        SUBWF   DatReg,W       ; compare data
        BTFSS   STATUS,Z       ; same?
        BSF     ConReg,LED     ; no - switch on LED
        INCF    AddReg         ; yes - next address
        BTFSC   STATUS,Z       ; last address in block?
        CALL    inchi          ; yes - increment high address
        RETURN                 ; no - continue

; Select next block of RAM ..............................

inchi   INCF    HiAdd          ; next block
        BTFSC   STATUS,Z       ; all done?
        GOTO    alldon         ; yes
        MOVF    HiAdd,W        ; no - load high address
        MOVWF   AddReg         ; output it
        BSF     ConReg,CLK0    ; clock it into latches
        BSF     ConReg,CLK1
        BCF     ConReg,CLK0
        BCF     ConReg,CLK1
        CLRF    AddReg         ; reset low address to zero
        RETURN                 ; block done
alldon  BCF     ConReg,ResHi   ; reset address latches
        RETURN                 ; all blocks done

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