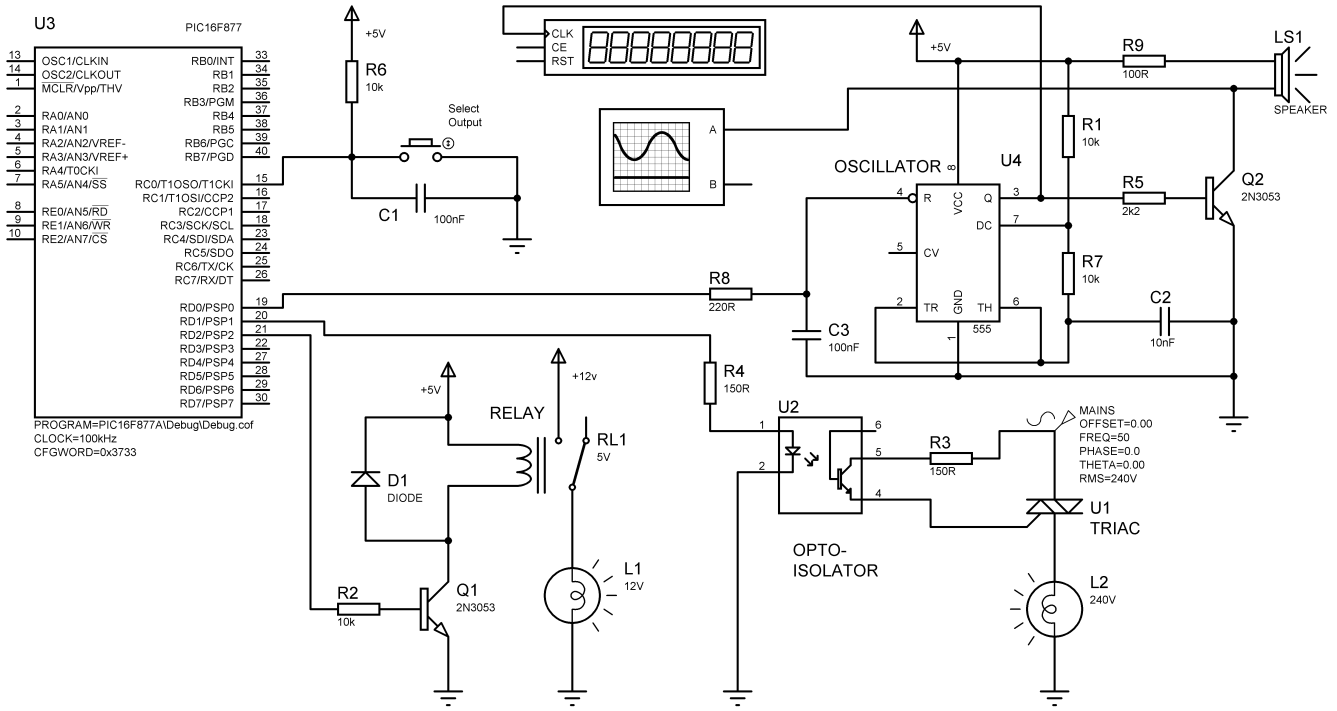
Interfacing PIC Microcontrollers

POWER2 Schematic



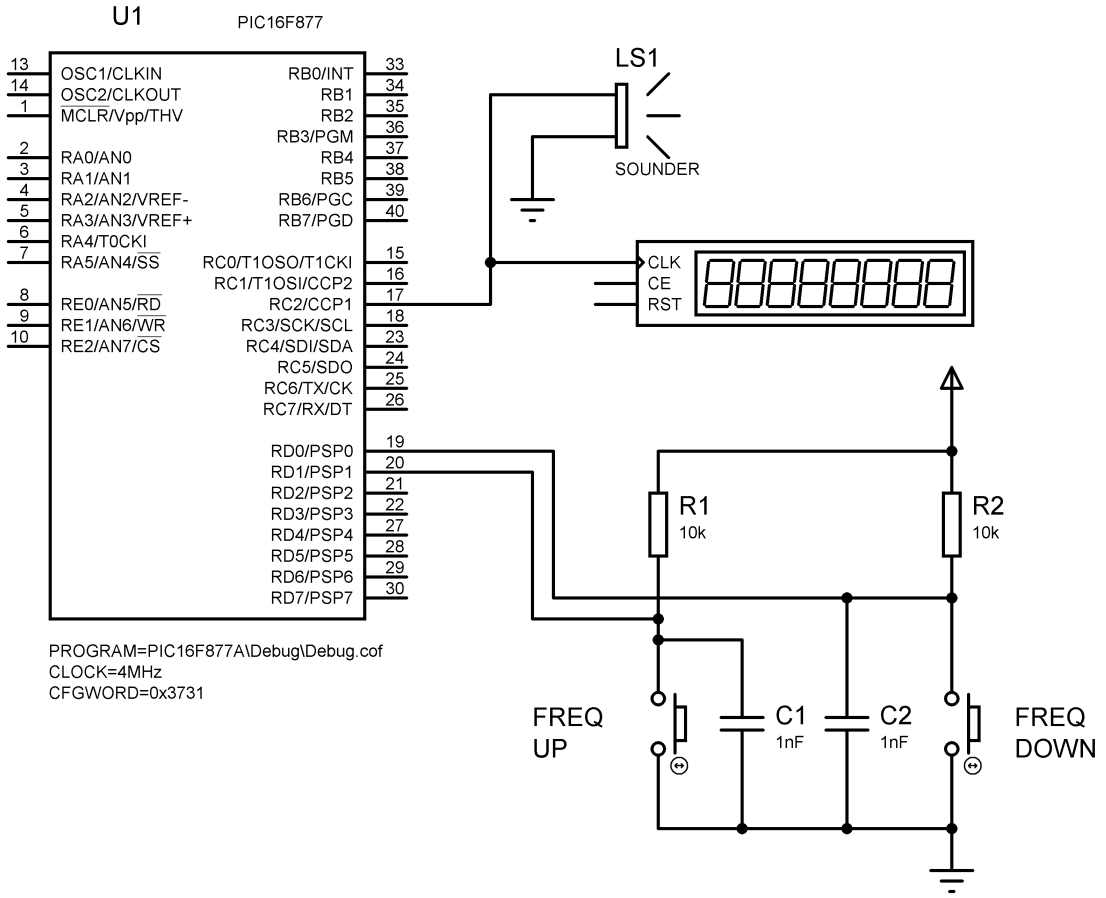
- This application demonstrates output interfacing for power loads
- The relay interface shows simple, low speed load switching
- The oscillator represents an external dedicated hardware interface that is simply switched on and off
- The opto-isolator and triac demonstrates digital power control, where the ac load current can be controlled

POWER2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:        Interfacing PICs Ed2
;       Source File:    POWER2.ASM
;       Author:         M. Bates
;       Date:           21-01-13
;       ..................................................
;
;       Step through power outputs using button
;       1. DC switch using relay
;       2. Mains AC switch using triac
;       3. Speaker output from relay
;       ..................................................
;
;       Processor:      PIC 16F877A
;       Clock:          RC = 100kHz
;       Inputs:         none
;       Outputs:        PortD
;
;       WDTimer:        Disabled
;       PUTimer:        Enabled
;       Interrupts:     Disabled
;       Code Protect:   Disabled
;
; ************************************************************

        PROCESSOR 16F877        ; Define MCU type
        __CONFIG 0x3733         ; Set config fuses (RC clock)

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

PORTC   EQU     07              ; Port C Data Register
PORTD   EQU     08              ; Port D Data Register
TRISD   EQU     88              ; Port D Direction Register
STATUS  EQU     03              ; Carry flag register

; ************************************************************

        CODE    0

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

        BANKSEL TRISD           ; Bank 1
        CLRF    TRISD           ; Outputs
        BANKSEL PORTD           ; Bank 0

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

start   CLRF    PORTD           ; Clear Port D Data
        BSF     PORTD,3         ; Select no output
        BCF     STATUS,0        ; Clear carry bit

next    BTFSC   PORTC,0         ; Button on?
        GOTO    next            ; no
wait    BTFSS   PORTC,0         ; yes - off?
        GOTO    wait            ; no

        RRF     PORTD           ; yes - move bit right
        BTFSS   STATUS,0        ; last output?
        GOTO    next            ; no - next output
        GOTO    start           ; yes - restart

        END                     ; terminate source code
```

Interfacing PIC Microcontrollers

PULSE2 Schematic



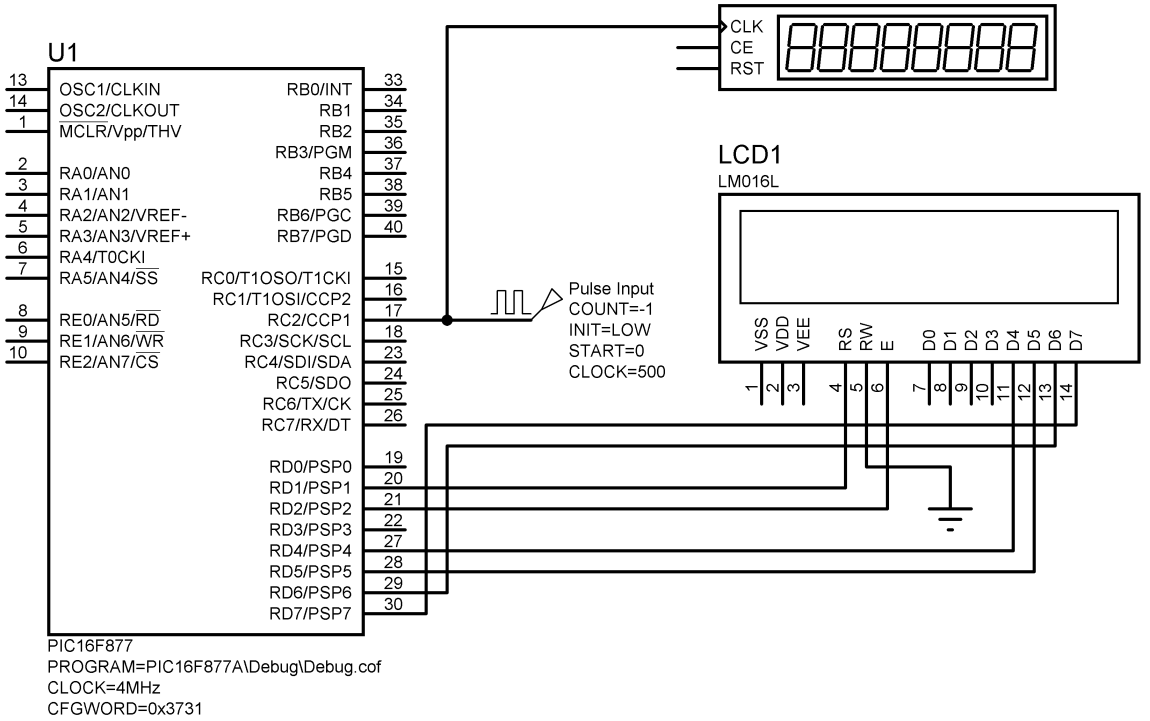
- This application demonstrates the generation of a timed output pulse
- A hardware timer is used in compare mode to generate the required mark/space ratio
- The use of the timer interrupt is also demostrated

PULSE2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       PULSE2.ASM     MPB            12-01-13
;
;       Generates timed output interval using Timer 1
;       in compare mode
;
;        Updated for VSM v8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877A       ; Select MCU
        __CONFIG 0x3731         ; Clock = XT 4MHz

;       LABEL EQUATES  ..................................

        INCLUDE "P16F877A.INC"; Standard register labels

Count   EQU     20              ; software timer

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

        CODE    0               ; Place machine code
        NOP                     ; for ICD mode
        GOTO    init            ; Jump over ISR vector

        ORG     4               ; ISR vector address
        GOTO    isr             ; run ISR

init    NOP
        BANKSEL TRISC           ; Select bank 1
        MOVLW   B'11111011'     ; RC2 = output
        MOVWF   TRISC           ; Initialise display port
        MOVLW   B'00000100'     ; Timer1 interrupt..
        MOVWF   PIE1            ; ..enable

        BANKSEL PORTC           ; Select bank 0
        CLRF    PORTC           ; Clear output
        MOVLW   B'11000000'     ; Peripheral interupt..
        MOVWF   INTCON          ; ..enable
        MOVLW   B'00001000'     ; Compare mode..
        MOVWF   CCP1CON         ; ..set output on match
        MOVLW   027             ; Initial value..
        MOVWF   CCPR1H          ; .. for high byte (10ms)
        MOVLW   010             ; Initial value..
        MOVWF   CCPR1L          ; .. for low byte (10ms)
        MOVLW   B'00000001'     ; Timer1 enable..
        MOVWF   T1CON           ; with internal clock (1MHz)

        GOTO    start           ; Jump to main program

;       SUBROUTINES.....................................

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

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


;       INTERRUPT SERVICE ROUTINE.......................

;       Reset interrupt, check buttons, generate 1ms pulse

isr     CLRF    PIR1            ; clear interrupt flags
        CLRF    TMR1H           ; clear timer high..
        CLRF    TMR1L           ; ..and low byte

        BTFSC   PORTD,0         ; dec frequency button?
        GOTO    other           ; no
        INCFSZ  CCPR1H          ; yes, inc period, zero?
        GOTO    other           ; no
        DECF    CCPR1H          ; yes, step back

other   BTFSC   PORTD,1         ; inc frequency button?
        GOTO    wait            ; no
        DECFSZ  CCPR1H          ; yes, inc period, zero?
        GOTO    wait            ; no
        INCF    CCPR1H          ; yes, step back

wait    CALL    onems           ; wait 1ms
        BCF     CCP1CON,3       ; clear output
        BSF     CCP1CON,3       ; re-enable timer mode

        RETFIE                  ; return to main program


;-------------------------------------------------------------
;       Main program
;-------------------------------------------------------------
start   GOTO    start           ; wait for timer interrupt
        END                     ; of source code
```

Interfacing PIC Microcontrollers

TIMIN2 Schematic



- This application demonstrates the use of a hardware timer to measure the period of an input pulse waveform
- The timer is set up in capture mode and the input transition used to trigger the capture operation
- The input period is calculated converted from a 16-bit result into microseconds and displayed on the LCD
- The interrupt routine simply resets the timer

TIMIN2 Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       TIMIN2.ASM     MPB           12-01-13
;
;       Measure input period using Timer1 16-bit capture
;       and display in microseconds, signal input CCP1
;
;        Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

       PROCESSOR 16F877A
       __CONFIG 0x3731

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

       INCLUDE "P16F877A.INC"; Standard register labels

;      Local label equates.....................................

Hibyte EQU     020
Lobyte EQU     021

Tents  EQU     022
Thous  EQU     023
Hunds  EQU     024
Tens   EQU     025
Ones   EQU     026


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

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

       ORG     4             ; Interrupt vector adress
       GOTO    ISR           ; jump to service routine

init   NOP
       BANKSEL TRISD         ; Select bank 1
       CLRF    TRISD         ; Initialise display port
       CLRF    PIE1          ; Disable peripheral interrupts

       BANKSEL PORTD         ; Select bank 0
       CLRF    PORTD         ; Clear display outputs

       MOVLW   B'11000000'   ; Enable..
       MOVWF   INTCON        ; ..peripheral interrupts
       MOVLW   B'00000100'   ; Capture mode:
       MOVWF   CCP1CON       ; ..every falling edge
       MOVLW   B'00000001'   ; Enable..
       MOVWF   T1CON         ; ..Timer 1

       GOTO    start         ; Jump to main program

; INTERRUPT SERVICE ROUTINE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ISR    CLRF    TMR1L
       CLRF    TMR1H
       BCF     PIR1,CCP1IF   ; Reset interrupt flag
       RETFIE

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

       INCLUDE "LCD.INC"     ; Include display routines

;----------------------------------------------------------------
; Convert 16 bit binary result to 5 digits
;----------------------------------------------------------------

conv   MOVF    CCPR1L,W      ; Get high byte
       MOVWF   Lobyte        ; and store
       MOVF    CCPR1H,W      ; Get low byte
       MOVWF   Hibyte        ; and store

       MOVLW   06            ; Correction value
       BCF     STATUS,C      ; prepare carry flag
       ADDWF   Lobyte        ; add correction
       BTFSC   STATUS,C      ; and carry
       INCF    Hibyte        ; in required

       CLRF    Tents         ; clear ten thousands register
       CLRF    Thous         ; clear thousands register
       CLRF    Hunds         ; clear hundreds register
       CLRF    Tens          ; clear tens register
       CLRF    Ones          ; clear ones register

; Subtract 10000d (2710h) and count ..........................

sub10  MOVLW   010           ; get low byte to sub
       BSF     STATUS,C      ; get ready to subtract
       SUBWF   Lobyte        ; sub 10h from low byte
       BTFSC   STATUS,C      ; borrow required?
       GOTO    sub27         ; no - sub high byte

       MOVF    Hibyte,F      ; yes - check high byte
       BTFSS   STATUS,Z      ; zero?
       GOTO    take1         ; no - take borrow

       MOVLW   010           ; yes - load low byte to add
       BCF     STATUS,C      ; get ready to add
       ADDWF   Lobyte        ; restore low byte
       GOTO    subE8         ; next digit
```

TIMIN2 Source Code

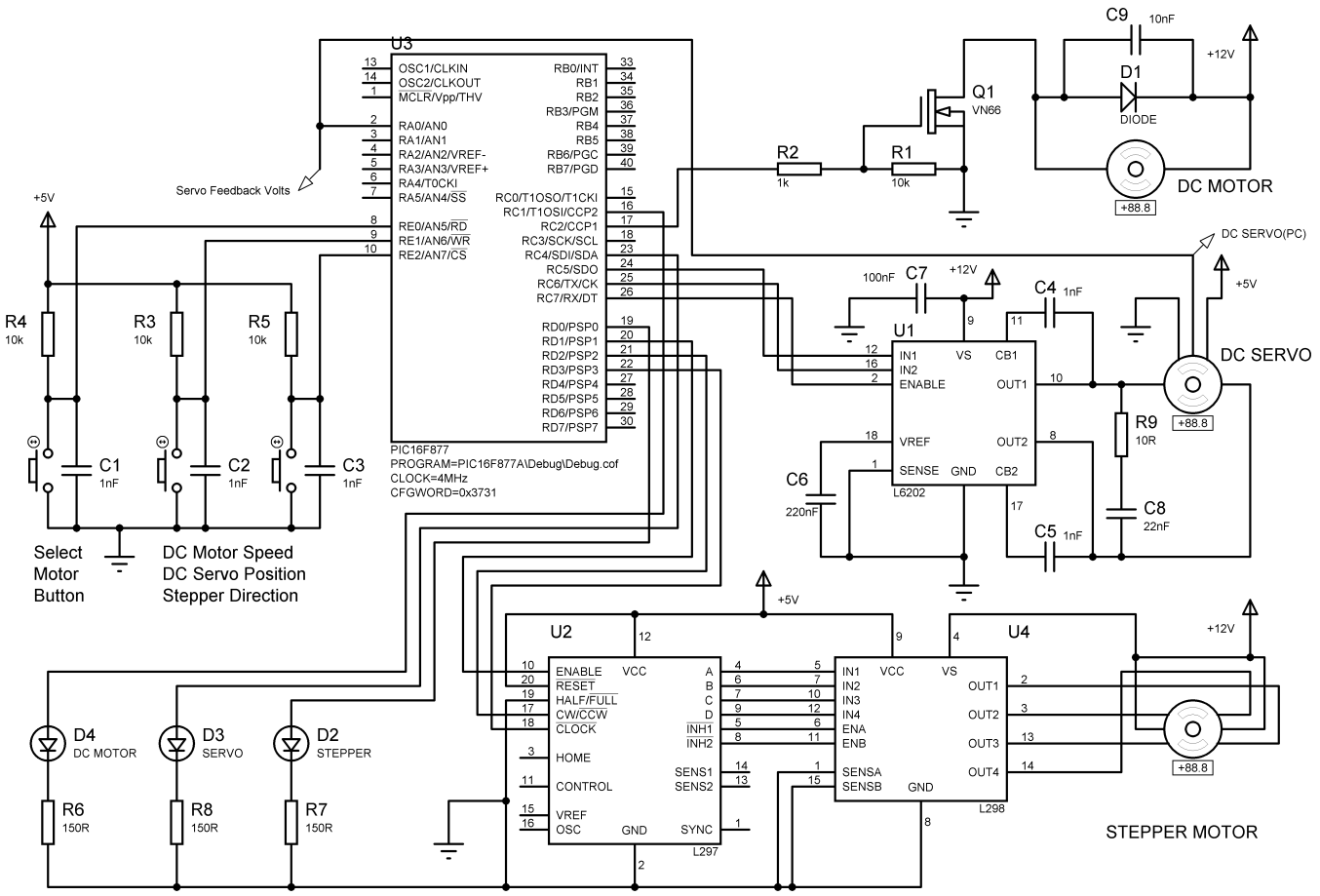
```
take1   DECF    Hibyte          ; take borrow

sub27   MOVLW   027             ; get high byte to sub
        BSF     STATUS,C        ; get ready to subtract
        SUBWF   Hibyte          ; sub from high byte
        BTFSS   STATUS,C        ; borrow taken?
        GOTO    done1           ; yes - restore remainder
        INCF    Tents           ; no - count ten thousand
        GOTO    sub10           ; sub 10000 again

done1   MOVLW   010             ; restore..
        BCF     STATUS,C        ; get ready to add
        ADDWF   Lobyte          ; restore low byte
        BTFSC   STATUS,C        ; Carry into high byte?
        INCF    Hibyte          ; yes - add carry to high byte
        MOVLW   027             ; restore..
        ADDWF   Hibyte          ; ..high byte

; Subtract 1000d (03E8) and count.............................

subE8   MOVLW   0E8             ; get low byte to sub
        BSF     STATUS,C        ; get ready to subtract
        SUBWF   Lobyte          ; sub from low byte
        BTFSC   STATUS,C        ; borrow required?
        GOTO    sub03           ; no - do high byte

        MOVF    Hibyte,F        ; yes - check high byte
        BTFSS   STATUS,Z        ; zero?
        GOTO    take2           ; no - take borrow

        MOVLW   0E8             ; load low byte to add
        BCF     STATUS,C        ; get ready to add
        ADDWF   Lobyte          ; restore low byte
        GOTO    sub64           ; next digit

take2   DECF    Hibyte          ; take borrow

sub03   MOVLW   03              ; get high byte
        BSF     STATUS,C        ; get ready to subtract
        SUBWF   Hibyte          ; sub from high byte
        BTFSS   STATUS,C        ; borrow taken?
        GOTO    done2           ; yes - restore high byte
        INCF    Thous           ; no - count ten thousand
        GOTO    subE8           ; sub 1000 again

done2   MOVLW   0E8             ; restore..
        BCF     STATUS,C        ; get ready to add
        ADDWF   Lobyte          ; restore low byte
        BTFSC   STATUS,C        ; Carry into high byte?
        INCF    Hibyte          ; yes - add carry to high byte
        MOVLW   03              ; restore..
        ADDWF   Hibyte          ; ..high byte

        ; Subtract 100d (064h) and count.............................

sub64   MOVLW   064             ; get low byte
        BSF     STATUS,C        ; get ready to subtract
        SUBWF   Lobyte          ; sub from low byte
        BTFSC   STATUS,C        ; borrow required?
        GOTO    inchun          ; no - inc count

        MOVF    Hibyte,F        ; yes - check high byte
        BTFSS   STATUS,Z        ; zero?
        GOTO    take3           ; no - take borrow

        MOVLW   064             ; load low byte to add
        BCF     STATUS,C        ; get ready to add
        ADDWF   Lobyte          ; restore low byte
        GOTO    subA            ; next digit

take3   DECF    Hibyte          ; take borrow

inchun  INCF    Hunds           ; count hundred
        GOTO    sub64           ; sub 100 again

; Subtract 10d (0Ah) and count, leaving remainder.................

subA    MOVLW   0A              ; get low byte to sub
        BSF     STATUS,C        ; get ready to subtract
        SUBWF   Lobyte          ; sub from low byte
        BTFSS   STATUS,C        ; borrow required?
        GOTO    rest4           ; yes - restore byte
        INCF    Tens            ; no - count one hundred
        GOTO    subA            ; and repeat

rest4   ADDWF   Lobyte          ; restore low byte
        MOVF    Lobyte,W        ; copy remainder..
        MOVWF   Ones            ; to ones register

        RETURN                  ; done
```

TIMIN2 Source Code

```
;-----------------------------------------------------------------
; Display period in microseconds
;-----------------------------------------------------------------

disp    BSF     Select,RS       ; Set display data mode

        MOVLW   'T'             ; Time period
        CALL    send            ; Display it
        MOVLW   ' '             ; Space
        CALL    send            ; Display it
        MOVLW   '='             ; Equals
        CALL    send            ; Display it
        MOVLW   ' '             ; Space
        CALL    send            ; Display it

; Supress leading zeros.....................................

        MOVF    Tents,F         ; Check digit
        BTFSS   STATUS,Z        ; zero?
        GOTO    show1           ; no - show it

        MOVF    Thous,F         ; Check digit
        BTFSS   STATUS,Z        ; zero?
        GOTO    show2           ; no - show it

        MOVF    Hunds,F         ; Check digit
        BTFSS   STATUS,Z        ; zero?
        GOTO    show3           ; no - show it

        MOVF    Tens,F          ; Check digit
        BTFSS   STATUS,Z        ; zero?
        GOTO    show4           ; no - show it

        MOVF    Ones,F          ; Check digit
        BTFSS   STATUS,Z        ; zero?
        GOTO    show5           ; no - show it

; Display digits of period..................................

show1   MOVLW   030             ; Load ASCII offset
        ADDWF   Tents,W         ; Add digit value
        CALL    send            ; Display it

show2   MOVLW   030             ; Load ASCII offset
        ADDWF   Thous,W         ; Add digit value
        CALL    send            ; Display it

show3   MOVLW   030             ; Load ASCII offset
        ADDWF   Hunds,W         ; Add digit value
        CALL    send            ; Display it

show4   MOVLW   030             ; Load ASCII offset
        ADDWF   Tens,W          ; Add digit value
        CALL    send            ; Display it

show5   MOVLW   030             ; Load ASCII offset
        ADDWF   Ones,W          ; Add digit value
        CALL    send            ; Display it

; Show fixed characters.....................................

        MOVLW   ' '             ; Space
        CALL    send            ; Display it
        MOVLW   'u'             ; micro
        CALL    send            ; Display it
        MOVLW   's'             ; secs
        CALL    send            ; Display it
        MOVLW   ' '             ; Space
        CALL    send            ; Display it
        MOVLW   ' '             ; Space
        CALL    send            ; Display it

; Home cursor ..............................................

        BCF     Select,RS       ; Set display command mode
        MOVLW   0x80            ; Code to home cursor
        CALL    send            ; Do it
        RETURN                  ; done

;-----------------------------------------------------------------
; MAIN LOOP
;-----------------------------------------------------------------
start   CALL    inid            ; Initialise display
        BANKSEL PIE1            ; Select Bank 1
        BSF     PIE1,CCP1IE     ; Enable capture interrupt
        BANKSEL PORTD           ; Select Bank 0
        BCF     PIR1,CCP1IF     ; Clear CCP1 interrupt flag

loop    CALL    conv            ; Convert 16 bits to 5 digits
        CALL    disp            ; Display period in microsecs
        GOTO    loop

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

Interfacing PIC Microcontrollers

MOTORS2 Schematic



- This application demonstrates various simple motor interfaces
- The DC motor speed is controlled by PWM via a single ended MOS FET drive
- The DC position servo is controlled via a bidirectional IC FET bridge driver, with dc feedback voltage
- The stepper motor is controlled via a dedicated chip driving four separate phases in sequence
- Each is selected in turn via the push button inputs

MOTORS2 Source Code

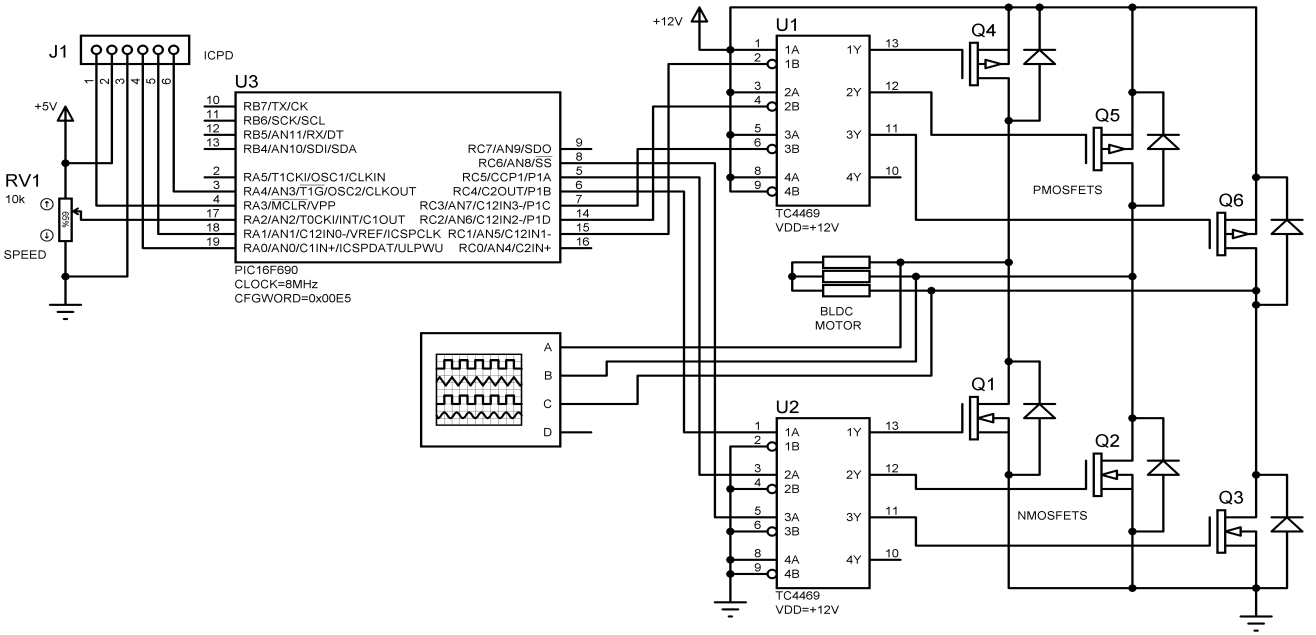
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Project:               Interfacing PICs Ed2
;       Source File Name:      MOTORS2.ASM
;       Devised by:            MPB
;       Date:                  21-01-13
;       Status:                Updated for VSM v8
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       Demonstrates DC, SERVO & STEPPER MOTOR control
;       Select motor and direction using push button inputs
;       - DC Motor PWM speed control
;       - DC Servo position control
;       - Stepper direction control
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

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

        INCLUDE "P16F877A.INC"
        ; standard register labels

;-------------------------------------------------------------
; User register labels
;-------------------------------------------------------------

Count1  EQU     20              ; delay counter
Count2  EQU     21              ; delay counter
Target  EQU     22              ; servo target position

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

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

;-------------------------------------------------------------
; Port & PWM setup

init    NOP
        BANKSEL TRISB           ; Select control registers
        CLRF    TRISC           ; Output for dc motors
        CLRF    TRISD           ; Output for stepper
        MOVLW   B'00000010'     ; Analogue input setup code
                                ; PortA = analogue inputs
                                ; Vref = Vdd
        MOVWF   ADCON1          ; Port E = digital inputs
        MOVLW   D'249'          ; PWM = 4kHz
        MOVWF   PR2             ; TMR2 preload value

        BANKSEL PORTB           ; Select output registers
        CLRF    PORTC           ; Outputs off
        CLRF    PORTD           ; Outputs off
        MOVLW   B'01000001'     ; Analogue input setup code
        MOVWF   ADCON0          ; f/8, RA0, done, enable
        MOVLW   D'128'          ; intial servo position
        MOVWF   Target

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

but0    BTFSC   PORTE,0         ; wait for select button
        GOTO    but0

        MOVLW   B'00001100'     ; Select PWM mode
        MOVWF   CCP1CON         ;
        MOVLW   D'128'          ; PWM = 50%
        MOVWF   CCPR1L          ;

but1    BTFSS   PORTE,0         ; wait for button release
        GOTO    but1
        CALL    motor           ; check for speed change
        BTFSC   PORTE,0         ; wait for select button
        GOTO    but1
        MOVLW   B'00000000'     ; deselect PWM mode
        MOVWF   CCP1CON         ;
        CLRF    PORTC           ; switch off outputs

but2    BTFSS   PORTE,0         ; wait for button release
        GOTO    but2
        CALL    servo           ; move servo cw or ccw
        BTFSC   PORTE,0         ; wait for select button
        GOTO    but2
        CLRF    PORTC           ; switch off servo

but3    BTFSS   PORTE,0         ; wait for button release
        GOTO    but3
        CALL    step            ; output one step cycle
        BTFSC   PORTE,0         ; wait for select button
        GOTO    but3
        CLRF    PORTD           ; disable stepper outputs
        GOTO    but0            ; start again
```

MOTORS2 Source Code

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

; Change dc motor speed by one step and wait 1ms
; to debounce and control rate of change.....................

motor   BSF     PORTC,1         ; switch on motor LED

        BTFSS   PORTE,1         ; inc speed?
        INCF    CCPR1L          ; yes
        MOVLW   D'248'          ; max speed?
        SUBWF   CCPR1L,W
        BTFSS   STATUS,Z
        GOTO    lower           ; no
        DECF    CCPR1L          ; yes - dec speed

lower   BTFSS   PORTE,2         ; dec speed?
        DECFSZ  CCPR1L          ; yes - min speed?
        GOTO    done            ; no
        INCF    CCPR1L          ; yes - inc speed

done    CALL    onems           ; 1ms debounce
        RETURN

; Move servo 10 bits cw or ccw...............................

servo   BSF     PORTC,4         ; switch on servo LED
        BSF     PORTC,7         ; enable drive chip
        BTFSC   PORTE,1         ; move forward?
        GOTO    rev             ; no

wait1   BTFSS   PORTE,1         ; yes- wait for button..
        GOTO    wait1           ; ..release
        MOVLW   D'10'           ; add 10...
        ADDWF   Target          ; ..to servo target position
        BSF     PORTC,5         ; move..
        BCF     PORTC,6         ; .. forward

getfor  CALL    getADC          ; get position
        BSF     STATUS,C        ; set carry flag
        MOVF    Target,W        ; load position
        SUBWF   ADRESH          ; compare with target
        BTFSS   STATUS,C        ; far enough?
        GOTO    getfor          ; no - repeat

        BCF     PORTC,5         ; yes - stop
        MOVLW   D'250'          ; wait 250ms ..
        CALL    xms             ; .. before next step
rev     BTFSC   PORTE,2         ; move reverse?
        RETURN                  ; no

wait2   BTFSS   PORTE,2         ; yes- wait for button..
        GOTO    wait2           ; ..release
        MOVLW   D'10'           ; yes - sub 10 from...
        SUBWF   Target          ; .. servo target position
        BCF     PORTC,5         ; move ..
        BSF     PORTC,6         ; ..reverse

getrev  CALL    getADC          ; get position
        BSF     STATUS,C        ; set carry flag
        MOVF    Target,W        ; load position
        SUBWF   ADRESH          ; compare with target
        BTFSC   STATUS,C        ; far enough?
        GOTO    getrev          ; no - repeat

        BCF     PORTC,6         ; yes - stop
        MOVLW   D'250'          ; wait 250ms ..
        CALL    xms             ; .. before next step
        RETURN

; Output one cycle of stepper clock..........................

step    BSF     PORTD,0         ; switch on stepper LED
        BSF     PORTD,1         ; enable stepper drive

        BTFSS   PORTE,1         ; test cw button
        BSF     PORTD,2         ; select clockwise
        BTFSS   PORTE,2         ; test ccw button
        BCF     PORTD,2         ; select counter-clockwise

        BSF     PORTD,3         ; clock high
        MOVLW   D'25'           ; load delay time
        CALL    xms
        BCF     PORTD,3         ; clock low
        MOVLW   D'25'           ; load delay time
        CALL    xms

        RETURN

; Stepper software delay ....................................

xms     MOVWF   Count2          ; receive x ms in W
down2   CALL    onems
        DECFSZ  Count2
        GOTO    down2
        RETURN

onems   MOVLW   D'249'          ; delay one millisec
        MOVWF   Count1
down1   NOP
        DECFSZ  Count1
        GOTO    down1
        RETURN

; 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

;-------------------------------------------------------------
        END                     ; of source code
;-------------------------------------------------------------
```

Interfacing PIC Microcontrollers

BDLC2 Schematic



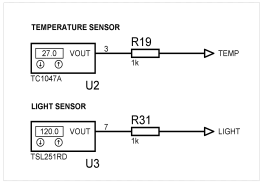
- This application demonstrates the drive required by the typical BLDC motor
- The system uses the PIC 16F690 MCU
- Three MOSFET half bridge current drive stages are interfaced via level shifting CMOS gates
- The BLDC phase windings are represented by simple resistors to simplify the display obtained on the virtual scope
- The output codes switch on the bidirectional phases in sequence to produce a rotating magnetic field

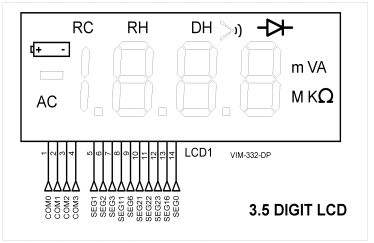
BDLC2 Source Code

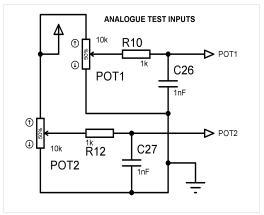
```
;
;       BLDC2.ASM       MPB     21-01-13
;
;       BLDC motor drive
;        Speed control adjust between 2 -20 Hz
;
;       MCU = 16F690
;        Internal clock = 8MHz
;
;*****************************************************************
;
;       Switching sequence
;
;       FET     Q6  Q5  Q4  Q3  Q2  Q1
;       OUT     RC6 RC5 RC4 RC3 RC2 RC1
;
;       CA      0   0   1   1   0   0
;       CB      0   1   0   1   0   0
;       AB      0   1   0   0   0   1
;       AC      1   0   0   0   0   1
;       BC      1   0   0   0   1   0
;       BA      0   0   1   0   1   0
;
;*****************************************************************

        PROCESSOR 16F690      ; Specify MCU for assembler
        INCLUDE "P16F690.INC" ; Standard labels
        __CONFIG 0x00E5       ; MCLR, PWRTE enabled, Internal Clock

;       Labels ..................................................

HICO    EQU     020
LOCO    EQU     021

;       Locate program start

         CODE    0

; Initialise registers.....................................

        BANKSEL TRISC         ; Select Bank 1
        MOVLW   B'10000001'
        MOVWF   TRISC         ; Initialise RC1-6 for output
        MOVLW   B'00100000'   ; Analogue input setup code
        MOVWF   ADCON1        ; Fosc/32

        BANKSEL PORTC         ; Select bank 0
        MOVLW   B'00000000'
        MOVWF   PORTC         ; Drives off
        MOVLW   B'00001001'   ; Analogue input setup code
        MOVWF   ADCON0        ; f/8, AN2, done, enable

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

start   MOVLW   B'00011000'   ; Output driver codes
        MOVWF   PORTC
        CALL    readin

        MOVLW   B'00101000'
        MOVWF   PORTC
        CALL    readin

        MOVLW   B'00100010'
        MOVWF   PORTC
        CALL    readin

        MOVLW   B'01000010'
        MOVWF   PORTC
        CALL    readin

        MOVLW   B'01000100'
        MOVWF   PORTC
        CALL    readin

        MOVLW   B'00010100'
        MOVWF   PORTC
        CALL    readin

        GOTO    start         ; repeat output loop

; SUBROUTINE ..............................................

readin  BSF     ADCON0,1      ; start ADC..
wait    BTFSC   ADCON0,1      ; ..and wait for finish
        GOTO    wait
        MOVF    ADRESH,W      ; store result

         MOVWF  HICO          ; load delay count
slow     MOVLW   d'249'       ; start 1000 cycle loop
        MOVWF   LOCO          ; = 500us
fast     NOP                  ; fill loop to 4 cycles
         DECFSZ  LOCO         ; low count
         GOTO    fast         ; loop
        DECFSZ  HICO          ; high count
        GOTO    slow          ; loop = ADC x 1000 cycles

        RETURN                ; from variable delay

;.........................................................
        END                   ; Terminate assembler
;*************************************************************
```

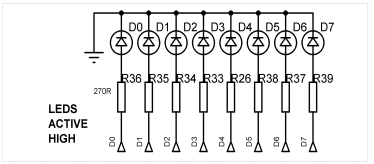
Interfacing PIC Microcontrollers

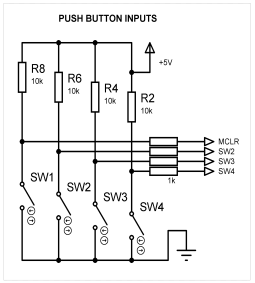
MECH2 Schematic

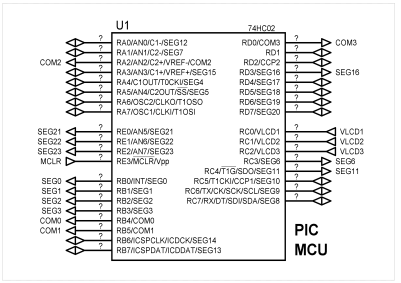


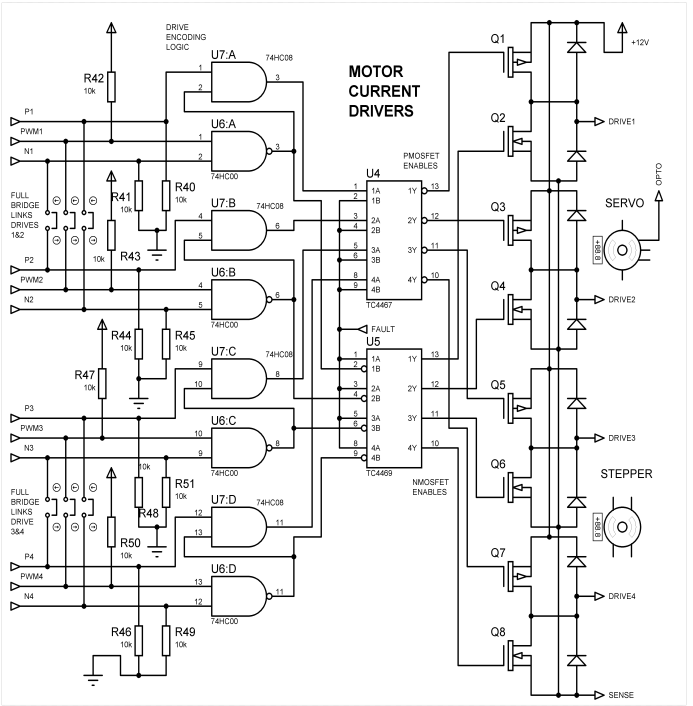


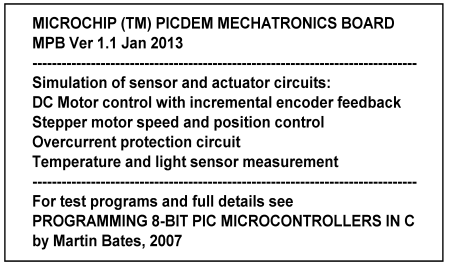


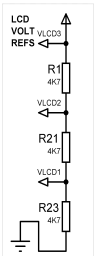


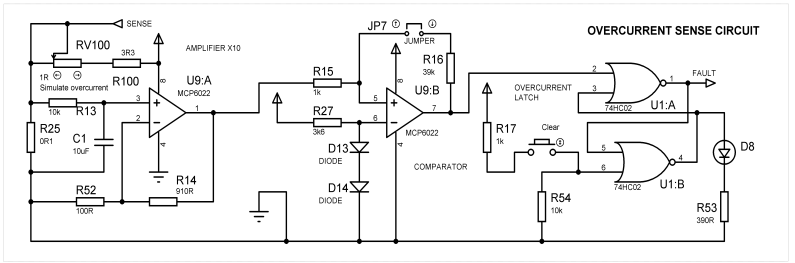












- This schematic shows the Microchip PICDEM mechatronics board designed to drive DC or stepper motors
- Four half bridge drivers are controlled by a discrete logic system that facilitate PWM, full bridge and half bridge drive schemes
- The PIC 16F917 MCU also operates a dedicated status LCD
- The board also incorporates temperature and light sensors and overcurrent protection
- Demo C programs are available