```
        processor   pic16F84    ; Processor type is an assembler directive
        radix   dec                         ; decimal is now default input radix
    ; for another radix use, binary b'00000100', hexadecimal h'FF'

    ; Memory Bounds     ( use the double underscore )
        __maxram        h'2F'   ; highest register file address used in pic16c84
        __badram        h'07'   ; register 7 does not exist
;
;-----------------------------------------------------------------------
; Definitions and the naming of bits and registers goes in here
        #define w       0       ; destination is W reg if dest = 0
        #define f       1       ; destination is RegisterFile if dest = 1
        #define c       0       ; Carry flag is status bit zero
        #define dc      1       ; Digit Carry flag is status bit one
        #define z       2       ; Zero flag is status bit two
;
        cblock  0       ; Bank 0 special register file names
        indf, tmr0, pcl, status, fsr, porta, portb, undef, eedata, eeaddr, pclath, intcon
        endc
;
        cblock  0       ; Bank 1 special register file names
        indf, optreg , pcl, status, fsr, trisa, trisb, undef, eecon1, eecon2, pclath,
intcon
        endc

bank0   macro
                bcf     status,5        ; clear bit 5 in status reg to select bank zero
    endm
;
bank1   macro
                bsf     status,5        ; set bit 5 in status reg to select bank one
        endm

;-----------------------------------------------------------------------
; Variables are stored in the register file, starting at location  0C hex
        cblock  0CH     ; start of 36 available bytes of ram in register file
                status          ; users variables here please
        endc                    ; note that a total of only 36 bytes of ram are available
        ; don't forget to initialise critical variables before enabling interupts
;
; Power up code starts here following reset of chip
        org     0       ; Reset vector
```

```
        goto    startup
;
startup  ; Main program gets to continue here, chained from goto instruction at location
zero

;-----------------------------------------------------------------------
Cycle                           ; expect it back here in 100 usec
    ; time is now 0 usec
        movf    PortA, w        ; read the state from port
        andlw   h'07'       ; mask high five bits, keep only three lowest bits
    movwf   state       ; save state of machine for this cycle

    btfss   state, 2    ; If state(bit2) NOT 1 then turn off motor
    bcf     PortB, 0    ; turn off motor, ( might have been 100% on )

    btfsc   state, 2    ; If state(bit2) NOT 0 then
        bsf     PortB, 0    ; turn on motor, PWM is a go

    call    delay25     ; 25 usec

    ; time is now 25 usec, turn off if PWM=25%
    movf    state, w    ; get cycle state
    xorlw   4           ; if state = b'100'
    btfsc   status, z   ; then
    bcf     PortB, 0    ; turn off motor

    call    delay25     ; 25 usec

    ; time is now 50 usec, turn off if PWM=50%
    movf    state, w    ; get cycle state
    xorlw   5           ; if state = b'101'
    btfsc   status, z   ; then
    bcf     PortB, 0    ; turn off motor

    call    delay25     ; 25 usec

    ; time is now 75 usec, turn off if PWM=75%
    movf    state, w    ; get cycle state
    xorlw   6           ; if state = b'110'
    btfsc   status, z   ; then
    bcf     PortB, 0    ; turn off motor
```

```
    call    delay25     ; 25 usec

    ; time is now 100 usec, stay on even if PWM=100%, to avoid an off glitch
    Goto    cycle       ; do state machine again

; end of code file
```