```
;> PIC version of Astable oscillator
VARSTART__   EQU     0x8
VAREND__     EQU     0x1f
LOCORE__     EQU     0
HICORE__     EQU     0x3ff

;> Use the W Register as a destination for results
WREG         EQU     0

;> Use the FILE Register as a destination for results
FILE         EQU     1

INDIR        EQU     0
RTCC         EQU     0x1
PC           EQU     0x2
PCL          EQU     0x2

;> F3 Reg is STATUS Reg.
STATUS       EQU     0x3
FSR          EQU     0x4

;>Oscillator calibration
OSC_CAL      EQU     0x5

;> I/O Port Assignments
PORT_B       EQU     0x6

;> I/O Port Assignments
PORTB        EQU     0x6
PORT         EQU     0x6
GPIO         EQU     0x6
RB0          EQU     0
RB1          EQU     0x1
RB2          EQU     0x2
RB3          EQU     0x3
RB4          EQU     0x4
RB5          EQU     0x5

;> Carry Bit is Bit0 of F3
CARRY        EQU     0
C            EQU     0
DCARRY       EQU     0x1
DC           EQU     0x1
;> Bit 2 of F3 is Zero Bit
Z_bit        EQU     0x2
Z            EQU     0x2
LSB          EQU     0
MSB          EQU     0x7
TRUE         EQU     0x1
YES          EQU     0x1
FALSE        EQU     0
NO           EQU     0

;> GPIO / Port B assignments
;> Bit 0 = LED 1 [ LSB  ]
;> Bit 1 = LED 2 [ NMSB ]
;> Bit 2 = LED 3
;> Bit 3 = SWITCH
;> Bit 4 = LED 4
;> Bit 5 = LED 5 [ MSB  ]

;> Register constants
;> Output control register

Astable      EQU     0x7
Period       EQU     0x8
```

```
;> 1 Sec Count accumulation register
One_Sec         EQU     0x9

;> Set Bit 0 for 1/2 second period
HALF_SEC        EQU     0

;> Set Bit 1 if count OK
SEC_SET         EQU     0x1

;> Set bit 2 if 5 second point
FIVE_SET        EQU     0x2

;> 2mS count
TWO_MS          EQU     0x2

;> reset loop count every 8 changes
LOOP            EQU     0x3

;> 1/2 second
HALF_SECOND     EQU     0x5

;> 16mS point count is 8 = half cycle
SIXTEEN_MS      EQU     0x8

ONE             EQU     0x1

;> Five seconds
FIVE            EQU     0x5

;> Count 10
TEN             EQU     0xa

;> 0x32 = 50 decimal
FIFTY           EQU     0x32
HUNDRED         EQU     0x64

;> LED 1 on GPIO 0
LED_1           EQU     0
;> LED 2 on GPIO 1
LED_2           EQU     0x1
;> LED 3 on GPIO 2
LED_3           EQU     0x2
;> LED 4 on GPIO 4
LED_4           EQU     0x4
;> LED 5 on GPIO 5
LED_5           EQU     0x5

;> Start main code

        ORG     0

;> set osc value - On start up this PIC sets the oscillator value in the WREG
;> which is placed immediately in the Oscillator calibration register
        movwf   OSC_CAL

;> Set up the prescaler etc
        movlw   0xC7
        option

;> Although we are not using the watch dog facility here we include for reference
;> Look it up in the data sheet
        clrwdt
```

```
;> Jump over any other code or unused memory to the start of the program code.
        goto START_UP

;> Our program code starts at location hex 0x40
        org     0x40

START_UP
;> Bits 3 of GPIO is input only
        movlw   0x8
        tris        GPIO

;> Clear the outputs to OFF
        clrf    GPIO

;> We are now going to clear the internal memory before using it
;> Start at the end of our working registers
        movlw   VAREND__
        movwf   FSR

DH1
;> clear the register pointed to by FSR
        clrf    INDIR

;> Adjust for next register to clear then check to see if we have finished
        decf    FSR,1
        movf    FSR,0

;> There are only 31 registers total in this device
        andlw   0x1f

;> Have we reached bottom yet
        addlw   0xf9
        btfss   STATUS,Z

;> If NOT at bottom loop and clear next otherwise skip next instruction
        goto DH1

;> Set up One second timer variables [ 10 * 100mS ]
        movlw   TEN
        movwf   One_Sec

;> set base period 50 * 2mS == 100mS
        movlw   0x32
        movwf   Period

;> Preset LED_1 ON
        bsf     GPIO,LED_1

;> show LED_1 on for first cycle
        movlw   0x1
        movwf   Astable

;> Start of Main - Our program loops from here to END continuously
main

;> First check for our 2mS time period - Is it OVER ?
;> If it is NOT over jump back to Main
        btfss   RTCC,TWO_MS
        goto        Main

;> Otherwise Reset 2mS counter
        movlw   0
        movwf   RTCC
```

```
;> Subtract 1 from the Period register - If it goes to ZERO the Z [ Zero ]
;> flag in the STATUS register is set for us to test
        decf    Period, FILE

;> If we have reached ZERO then skip the next instruction
        btfss   STATUS, Z

;> Not ZERO so jump to loop back to Main until ready
        goto    END_LOOP_MAIN

;* Reset 100mS period
        movlw   0x32
        movwf   Period

;> Now test for the end of our ONE second period
        decf    One_Sec, FILE
        btfss   STATUS,Z
        goto    END_LOOP_MAIN

; Reset the One second timer
        movlw   TEN
        movwf   One_Sec

;> Now we need to know which LED is ON - So subtract ONE from register Astable
;> If it is ZERO after subtracting ONE then LED 1 was ON So go do LED 2
;> NOTE that we subtract 1 from the number in Astable but we put the result
;> in to WREG so the number in Astable remains the same as it was before
        movlw   0x1
        subwf   Astable, WREG
        btfss   STATUS, Z_bit
        goto    SET_LED_1

SET_LED_2
;> First set Astable to 2 which tells us that LED 2 is now ON, NOT LED 1
        movlw   0x2
        movwf   Astable

;> Now switch ON LED 2
        bsf     GPIO,LED_2

;> Now switch OFF LED 1
        bcf     GPIO,LED_1

;> Jump to program loop
        goto    END_LOOP_MAIN

SET_LED_1
;> Set Astable to 1 telling us LED 1 is now ON
        movlw   0x1
        movwf   Astable

;> Now switch ON LED 1
        bsf     GPIO,LED_1

;> Now switch OFF LED 2
        bcf     GPIO,LED_2

END_LOOP_MAIN
        goto    main

   END
```