

```

;*****
;File Name: Alarm16F676.asm
;Author: J. Price
;Date: 9 Aug 2004
;Version: 1.00
;Description: Turn on two LEDs and buzz the buzzer, then delay and repeat
;
;*****
;Hardware:
;
;PIC 16F676 with 2 LEDs and piezo buzzer
;
; 1 VDD +5V
; 14 VSS ground
; 13 RA0 ICSPDAT serial programming data
; 12 RA1 ICSPCLK serial programming clock
; 4 VPP +13V programming voltage
;
; 8 RC2 red LED
; 9 RC1 red LED
; 10 RC0 piezo buzzer
;
;*****

list p=16F676 ; list directive tells assembler which processor
#include <p16f676.inc> ; get file containing processor specific symbol definitions

__CONFIG _CP_OFF & _WDT_OFF & _BODEN & _PWRTE_ON & _INTRC_OSC_NOCLKOUT & _MCLRE_OFF & _CPD_OFF

; '__CONFIG' directive is used to embed a processor configuration word within the .asm file.
; The labels following the directive are located in the p16f676.inc file.
; See the data sheet for information on configuration word settings.

;*****
;Define symbols to make the code more readable
;*****

#define Bank0 0x00 ; starting address of memory bank 0
#define Bank1 0x80 ; starting address of memory bank 1

#define BUZZ PORTC,0 ; define symbols to refer to LED and buzzer I/O ports
#define LED1 PORTC,1
#define LED2 PORTC,2
#define CTris B'11111000' ; bit pattern to config. LED and buzzer I/O pins to outputs
#define toggle B'00000001' ; mask to toggle buzzer bit of port C

```

```

#define CMoff      B'00000111'      ; bit pattern to turn analog comparator off

;*****
;Define storage area for variables, data memory starts at hex location 20
;*****

        cblock 0x20
        counter      ; buzzer loop counter
        CountH      ; timer variable
        CountL      ; timer variable
        endc

;*****
;Reset Vector
;*****
        ORG      0x000      ; processor reset vector, execution starts here on power up
        nop      ; required by in circuit debugger
        goto     Init      ; go to beginning of program

;*****
;Interrupt Vector
;*****
        ORG      0x004
        return     ; execution starts here when there is an interrupt
;*****
;Initialization
;*****
Init
        call     0x3FF      ; retrieve factory clock frequency calibration value
                                ; comment this instruction if using simulator, ICD2, or ICE2000
        banksel Bank1      ; go to bank 1 to reach TRIS and OSCAL registers
        movwf   OSCCAL     ; update osc. calibration register with factory cal value
        movlw  CTris      ; get bit pattern to set Port C so bits 0,1,2 are outputs
        movwf  TRISC      ; set it
        clrf   ANSEL      ; set I/O pins to digital, not analog
        banksel Bank0     ; back to bank 0 to reach I/O ports
        movlw  CMoff      ; get bit pattern to turn analog comparator off
        movf   CMCON      ; turn comparator off

;*****
;Main program
;*****
        bcf    BUZZ      ; turn off buzzer I/O pin
        bcf    LED1      ; turn off LED1
        bcf    LED2      ; turn off LED2

Mainloop

```

```

        bsf    LED1        ; turn on LED1
        bsf    LED2        ; turn on LED2

        movlw  D'200'      ; put number of buzzer cycles desired in W register
        movwf  counter     ; store number of cycles in the counter variable
Buzzloop1
        movlw  toggle      ; get the toggle mask
        xorwf  PORTC,f     ; XOR mask with Port C, this toggles the buzzer ouput
        call   timer       ; call timer subroutine
                          ; 0.5 ms delay for 1 kHz buzzer frequency
        decfsz counter,f   ; decrement loop counter, skip if done
        goto  Buzzloop1   ; loop if counter not zero yet
                          ; else continue

        bcf    LED1        ; turn off LED1
        bcf    LED2        ; turn off LED2

        movlw  D'200'      ; put number of cycles desired in W register
        movwf  counter     ; store number of cycles in the counter variable
loop2
        call   timer       ; 0.5 ms delay
        decfsz counter,f   ; decrement loop counter, skip if done
        goto  loop2       ; loop if counter not zero yet
                          ; else continue

        goto  Mainloop    ; repeat forever

;*****
;Subroutines
;*****

;*****
;timer
;*****
;
; If you carefully count the number of instructions that are executed by this
; routine, you can show that the timer constants (TenMSH and TenMSL)
; control the total time delay as follows:
;
; delay = 4*(4/fosc) + (outer loop time)=
;        =(4/fosc)*[4+TenMSH*[5+3*TenMSL]]=(4/fosc)*[4+5*TenMSH+3*TenMSH*TenMSL]
;
; if TenMSH = D'1', TenMSL = D'164', 4/fosc = 1 us
; then delay = 501 us
;

#define TenMSH    D'1'

```

```

#define TenMSL      D'164'

timer
    movlw TenMSH      ; set outer timer loop count
    movwf CountH      ; outer loop overhead is 2 instructions

; outer loop time=TenMSH*[5*(4/fosc)+(inner loop time)]

SD10
    movlw TenMSL      ; set inner timer loop
    movwf CountL

; inner loop time=[3*TenMSL * (4/fosc)]

SD20
    decfsz CountL,f    ; inner loop countdown
    goto SD20         ;

    decfsz CountH,f    ; outer loop countdown
    goto SD10         ; reset inner loop after each outer loop count

    return            ; return from subroutine

    END                ; end of code

```