```
;*****************************************************************************
;
;File Name:     Alarm16F676.asm
;Author:        N. Fitch
;Date:          Oct 21, 2009.
;Version:       2.0
;Description:   Turn on two LEDs and buzz the buzzer, then delay and repeat
;                       Bit 0 on port C (RC0) is used as an output connected to the buzzer
;                       Bit 1 on port C (RC1) is used as an output connected to LED#1
;                       Bit 2 on port C (RC2) is used as an output connected to LED#2

;       Numbers on the right hand edge correspond to command notes and explanations that appear
;       at the bottom of this document.  As this is an introductory PIC chip example code,
;       a minimum of chip programming commands has been used (only 10).  The entire Microchip
;       "assembly" programming language consists of only about 40 such commands.

;*****************************************************************************
;  SOFTWARE ASSEMBLY INFORMATION                                                (1)
;*****************************************************************************
        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 and variables to make the code more readable.                   (2)
;*****************************************************************************
#define         BUZZER          PORTC,0         ; define symbols to refer to LED and buzzer I/O ports
#define         LED1            PORTC,1         ; For instance, BUZZ now refers to bit 0 of PORTC
#define         LED2            PORTC,2
#define         CTris           B'11111000'     ; bit pattern to config. LED and buzzer I/O pins to outputs
#define         Bank0           h'00'           ;                                       (3)
#define         Bank1           h'80'
#define         CMoff           B'00000111'     ; bit pattern to turn analog comparator off


#define         BUZZcounter     h'20'   ; buzzer loop counter lives at address 20        (4)
                                        ;(h'xx' means in hexadecimal)
#define         BUZZnumber      d'200'  ; number of times the buzzer will buzz each main loop cycle
                                        ; (d'xx' means decimal value)
                                        ; default values give 1ms per entire period so d'200' gives
                                        ; 200ms of "buzzing", then 200ms of "silence", repeated.
#define         InnerCounter    h'22'   ; delay counter storage location
#define         InnerCount      d'163'          ; d'163' makes each delay 500.0us long.
;*****************************************************************************

;Reset Vector
;*****************************************************************************
        ORG     0x000           ; processor reset vector, execution starts here on power up
        nop                             ; required by in circuit debugger       (5)
        goto    Init            ; go to beginning of program                    (6)

;Initialization
;*****************************************************************************
Init
        banksel Bank1           ; go to bank 1 to reach TRIS registers                  (7)
        movlw   CTris           ; get bit pattern to set Port C so bits 0,1,2 are outputs
        movwf   TRISC           ; set it                                                (8)
        clrf    ANSEL           ; set I/O pins (PORTA and PORTC) to digital, not analog (8a)
        banksel Bank0           ; back to bank 0 to reach I/O ports

        movlw   CMoff           ; putting the bit value CMoff into CMCON special register
        movwf   CMCON           ; turns off the comparator (part of making the I/O ports digital)

        call    LED1off         ; turn off the LEDs and the buzzer                      (9)
        call    LED2off
        call    BUZZoff
        call    CorrectOscillator       ; The PIC chip's internal RC oscillator doesn't
                                        ; actually run at 4MHz without a correction value.
```

```
;*****************************************************************************
;
;Main program
;*****************************************************************************
;
Mainloop
        call    LED1on              ; turn on the LED#1
        call    LED2on              ; turn on LED#2

        movlw   BUZZnumber          ; put number of buzzer cycles desired in W register          (10)
        movwf   BUZZcounter         ; store number of cycles in the counter variable             (11)
loop1                               ; The buzzer is buzzing during loop 1
        call    BUZZon              ; BUZZZZZZZZZ
        call    delay               ; call delay subroutine (leave buzzer on for a while)
        call    BUZZoff             ; No BUZZZZZZZ
        call    delay               ; leave buzzer off for a while
        decfsz  BUZZcounter,f       ; decrement loop counter, skip next command if done          (12)
        goto    loop1               ; loop if counter not zero yet
                                    ; else continue

        call    LED1off             ; turn off LED#1
        call    LED2off             ; turn off LED#2

        movlw   BUZZnumber          ; put number of cycles desired in W register (again)
        movwf   BUZZcounter         ; store number of cycles in the counter variable

loop2                               ; The buzzer is not buzzing during loop 2
        call    delay               ; delay called twice above (loop1) for each loop sequence
        call    delay               ; replicate that here, but without ever toggling the buzzer on/off.
        decfsz  BUZZcounter,f       ; decrement loop counter, skip if done
        goto    loop2               ; loop if counter not zero yet
                                    ; else continue

        goto    Mainloop        ; repeat forever
;*****************************************************************************
;


;Subroutines
;*****************************************************************************
;
LED1off
        bcf         LED1        ; turn off LED#1                                    (13)
        return
LED1on
        bsf         LED1        ; turn on LED#1
        return
LED2off
        bcf         LED2        ; turn off LED#2
        return
LED2on
        bsf         LED2        ; turn on LED#2
        return
BUZZoff
        bcf         BUZZER      ; turn off buzzer I/O pin
        return
BUZZon
        bsf         BUZZER      ; turn on the buzzer
        return

delay                           ; This delay is exactly (!) 500 us long.
        movlw   InnerCount
        movwf   InnerCounter
InnerLoop
        decfsz  InnerCounter,f  ;
        goto    InnerLoop
        nop
        return              ; return from subroutine when "OuterCounter" is empty.

CorrectOscillator
        call    0x3FF       ; this special function register contains the oscillator correction value
        movwf   OSCCAL      ; now the oscillator should really be running at 4MHz.
        return

        END                 ; end of code, this has to be here for your code to build correctly.
```

```
; NOTES
;(1) The configuration section loads files to be included (that have pre−determined
;    definitions, like what PORTC corresponds to on the actual device) with the
;    #include command.  It also accesses various chip options, like what to do on
;    a brownout event, what code to protect, various timers etc.
;
;(2) User defined symbols and variables make the code easier to read.  The #define
;    command defines the first argument as the second argument.  Thus when you refer
;    to "BUZZER" in the main program, the chip knows that you mean bit 0 of PORTC.
;
;(3) The 16F676 has two "banks" of file registers in the chip.  One shortcoming of
;    Microchip's design is that the user must explicitly change between banks to access
;    file registers that live there.  Hence to write to the TRISC special function
;    register, you have to change to the bank in which it lives (bank1 in this case).
;
;(4) The file register that holds the current buzz counter is at address h'20'.  This
;    is the first user−accessible file register in bank 0.  Registers below that address
;    are special function registers.  See the chip's datasheet.
;
;(5) "nop" is a command that means "no operation".  It just kills a little time.
;
;(6) A "goto" command branches (jumps) to the label that follows it.  No return information
;    is stored.
;
;(7) The "banksel" command selects the specified bank.
;
;(8) The TRISA and TRISC registers control the behavior of the input/output pins for
;    PORTA and PORTC respectively.  Writing 00001111 to the TRISA register makes the most
;    significant (bits 7,6,5,and 4) outputs and the least significant bits inputs.
;    These I/O pins also can function in analog mode (for Digital to Analog conversion etc.).
;    You will notice in the code that the analog mode is explicitly turned off.
;
;(8a) The "clrf" command clears the specified file register (makes all bits 0's).
;
;(9) The "call" command branches to a label in the code just like a "goto" does, but it
;    also stores a return address so that when a "return" is executed, the processor
;    jumps back to the next command after the "call".
;
;(10) The "movlw" command moves a literal (8 bits of data) into the working file register (W).
;     The literal that is moved into W is the following 8−bit value.
;
;(11) The "movwf" command moves the contents of W into the specified file register.
;
;(12) The "decfsz" command decrements the value of a file register, skipping the NEXT
;     line of code (usually a looped goto) if the result is zero (low state=0V).
;     This is useful for delay loops and counting loops.  The ",f" specification
;     following the register name instructs the chip to leave the decremented
;     result in that file register.  There is also a "decfss" which skips the
;     next instruction in the code if the tested bit is set (high).
;
;(13) The "bcf" command clears (makes low) the specified bit.
;     It is followed by "register address,bit#".
;(14) the "bsf" command sets (makes high) the specified bit.
```