

```

/*****
/* Program      : BUZZER.C
/* Function     : Buzzer Control Procedures
/* Author      : John F. Fitter B.E.
/*
/* Rev No.    Rev date    Test date    Test platform    Description
/* -----    -
/*      01     6jun98          PIC16C77-ME      Original
/*
/*          Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved
/*****

#define _BUZZER_C

#include <stdio.h>
#include <stdlib.h>
#include <commddefs.h>
#include "main.h"
#include "buzzer.h"
#include "lcd44780.h"

/*****
/* Procedures to drive the buzzer.
/* These procedures have been written to make use of the counter which is used to operate
/* the backlight. The reason for this is that if the buzzer were driven with delay routines
/* the processing loop would be stopped for the duration of the buzzing.
/*****

// This procedure sets up the buzzer to begin beeping. nbeeps is the number of times to
// beep. The on and off times are 100ms. The durations are only approximate and are tied to
// the millisecond interrupt.

void beep(unsigned char nbeeps) {

    if(flag_buzzen) { // check if the buzzer is enabled
        buz_beeps = nbeeps; // set the number of beeps
        buz_elaps = 10; // set the buzzer 10mS counter
        flg_buzzing = true; // flag buzzer is on
        buzzer_on(); // make it happen
    }
}

// This procedure manages the buzzer while it is beeping. It must be called regularly otherwise
// the buzzer will operate continuously. Normally this procedure is called every millisecond
// to enable other processes to run concurrently with buzzer operation. For exclusive buzzer
// operation this procedure must be called from a tight loop and it's return value used for
// loop termination. The procedure returns false when beeping is finished or not enabled.

unsigned char update_buzzer() {

    if(flag_buzzen) { // do nothing if buzzer not enabled
        if(buz_beeps) { // finished beeping ?
            if(!buz_elaps) { // no, finished buzzing or pausing ?
                buz_elaps = 10; // finished buzzing/pausing, reset timer
                if(flag_buzzing) {
                    --buz_beeps; // decrement beep counter
                    flg_buzzing = false; // flag as not buzzing
                    buzzer_off(); // and turn off the buzzer
                }else {
                    flg_buzzing = true; // flag as buzzing
                    buzzer_on(); // and turn on the buzzer
                }
            }
        }else{ // must be finished beeping
            buzzer_off(); // turn off buzzer (precautionary)
            return false; // return false - finished
        }
        return true; // return true - not finished
    }
    return false; // return false - buzzing not enabled
}

// ***** EOF BUZZER.C *****

```