

```

/*****
/* Program          : BOOM.C
/* Function         : Boomerang Light Control Program
/* Author          : John F. Fitter B.E.
/* Language        : HiTech C (ANSI C)
/* Platform        :
/* Target          : PIC 12C509 @ 4MHz
/* Development     : 16C01-ME @ 4MHz
/* Target hardware : Boomerang Light Controller Rev A1
/*
/* Version         : 01
/* Revisions       :
/*   Rev No.       :
/*   Rev date      :
/*   Description   :
/*   -----
/*   00            : 27jun98      Original
/*
/* Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved
*****/

// Note: It is the responsibility of each procedure to setup port pin directions prior to use.
//       Procedures are allowed to leave the direction registers in an unknown state.

// This program drives 5 pairs of led's in various sequences. The led's are operated in pulsed
// mode with a duty cycle of 20% and a frequency equal to the speed of the main program loop.
// The led's are pulsed sequentially to provide for even current draw from the battery.
// A display sequence is overlaid on this which is driven from a rom table.

#define _BOOM_C

#include <stdlib.h>
#include <stdio.h>
#include <pic.h>
#include "boom.h"

__CONFIG(FOSC1); // no watchdog

/*****
/* Main program.
*****/

main() {

    unsigned char ndx, n;

    // Initialize port direction registers and data latches.
    TRIS = 8; // GPIO all outputs except GP3
    GPIO = 0; // All led's turned off

#ifdef _DEVELOPMENT
    OSCCAL = 0x60; // set the oscillator calibration
#endif

    // Initialize peripheral hardware.
    OPTION = PS2 | PS0; // prescaler 1:64, active pullups on
    TMR0 = 0; // reset the timer

    // Determine source of reset - powerup or wakeup on port change
    if(TO && PD) { // reset is due to powerup ?
        n = GPIO; // yes, read pins to prevent wakeup
        asm("\tsleep"); // and go to sleep
    }
    // Initialize global variables
    sequence = 0; // reset sequence number
    output_mask = 4; // set output mask to first led
    buttonup = true; // last button state is unpressed
    new_sequence = true; // ensure sequence initialization

    // Main endless loop
    while(true) {
        if(new_sequence) { // check for a sequence change
            count_1s = 200; // set 1S counter
            count_sleep = SLEEPTIME; // set sleep time counter
            index = 0;
            ndx = sequence * 2;
            for(n = 0; n < 5; n++) led[n] = led_on[n][ndx]; // initialize the led's
            new_sequence = false;
            output_pattern = 0xff; // set output pattern to all led's on
        }
    }
}

```

```

GPIO = patt_mask[output_mask] & output_pattern; // set the led's - every loop
if(!output_mask--) output_mask = 4;           // mask for next led - every loop

if(TMR0 > 78 ) {                               // has 5mS (4992uS) elapsed ?
    TMR0 = 0;                                  // yes, reset the timer
    if(!--count_1s) {                          // has 1S elapsed ?
        count_1s = 200;                        // yes, reload the counter
        if(!--count_sleep) {                  // sleep time elapsed ?
            GPIO = 0;                          // yes, turn off the led's
            n = GPIO;                          // read pins to prevent wakeup
            asm("\tsleep");                    // and go to sleep
        }
    }
    for(n = 0; n < 5; n++) {                   // loop for each led
        if(!--led[n]) {                       // has the led timed out ?
            ndx = sequence * 2;                // yes, compute array indices
            if(index & patt_mask[n]) ndx += 1; // first or second timing values ?
            if(output_pattern & patt_mask[n]) { // is the led on ?
                led[n] = led_off[n][ndx];     // yes, setup led off time
                index ^= patt_mask[n];        // and select next timing values
            }else led[n] = led_on[n][ndx];    // led is off, setup on time
            output_pattern ^= patt_mask[n];   // toggle the led status
        }
    }
    if(!GP3 && buttonup) {                     // has button just been pressed ?
        if(++sequence > 9) sequence = 0;     // yes, then select next sequence
        new_sequence = true;                 // flag the sequence change
    }
    buttonup = GP3;                           // save the button state
}
}
}
}
// ***** EOF BOOM.C *****

```