

```

/*****
/*
/*----- L E D C -----*/
/* Task : Sets the various bits in the BIOS keyboard flag, causing the LEDs on the AT keyboard to flash. */
/*-----*/
/* Author : Michael Tischer */
/* Developed on : 08/22/88 */
/* Last update : 04/07/95 */
/*-----*/
/* Memory model : SMALL */
*****/

/*== Add include files =====*/

#include <stdio.h>
#include <dos.h>
#include <bios.h>

/*== Macros =====*/

#ifndef MK_FP /* Was MK_FP already defined? */
#define MK_FP(seg, ofs) ((void far *)\
                        ((unsigned long) (seg)<<16|(ofs)))
#endif

/*-- BIOS_KBF creates a pointer to the BIOS keyboard flag -----*/
#define BIOS_KBF ((unsigned far *) MK_FP(0x40, 0x17))

#define TICKS(ms) ((ms*10+549) / 550 ) /* Convert millis. to ticks */

/*== Constants =====*/

#define SCRL 16 /* SCROLL LOCK bit */
#define NUML 32 /* NUM LOCK bit */
#define CAPL 64 /* CAPS LOCK bit */
#define INS 128 /* INSERT bit */

#ifdef __TURBOC__ /* Definitions for TURBO C */
#define GetBiosTime(x) ( x = biostime( 0, NULL ) )
#else /* Definitions for Microsoft C Compiler */
#define GetBiosTime(x) ( _bios_timeofday( _TIME_GETCLOCK, &x) )
#endif

/*****
/* Delay: Waits a certain length of time. */
/* Input : PAUSE = The number of ticks to wait. */
/* Output : None */
/* Info : One tick = 1/18.2 seconds */
*****/

void delay( unsigned int pause )
{
    long curtime, /* Current time */
        trgttime; /* Target time */

    if ( pause ) /* Pause not equal to 0? */
    { /* No */
        GetBiosTime( trgttime );
        trgttime += (long) pause; /* Target time elapsed */

        do /* Wait loop, get current time */
            GetBiosTime( curtime );
        while ( curtime < trgttime ); /* Time elapsed? */
    } /* Yes --> End function */
}

/*****
* Function : S E T _ F L A G *
**-----**
* Description : Sets individual bits or flags in the BIOS keyboard flag. *
* Input parameters : FLAG = The bit or flag to be set. *
* Return value : None *
*****/

```

```

*****/

void set_flag( unsigned flag )
{
    union REGS regs;                /* Store the processor registers */

    *BIOS_KBF |= flag;    /* Set the specified bits in the keyboard flag */
    regs.h.ah = 1;        /* Function no.: Character ready? */
    int86(0x16, &regs, &regs);    /* Call BIOS keyboard interrupt */
}

/*****
*   Function           : C L R _ F L A G
**-----**
*   Description        : Clears bits or flags in BIOS keyboard flag.
*   Input parameters   : FLAG = Bit or flag to be cleared.
*   Return value       : None
*****/

void clr_flag( unsigned flag )
{
    union REGS regs;                /* Store the processor registers */

    *BIOS_KBF &= ~flag;    /* Mask bits in BIOS keyboard flag */
    regs.h.ah = 1;        /* Function no.: Character ready? */
    int86(0x16, &regs, &regs);    /* Call BIOS keyboard interrupt */
}

/*****
**                               **
**                               **
*****/

void main()
{
    unsigned i;                    /* Loop counter */

    printf( "LEDC - (c) 1988, 92 by Michael Tischer\n\n");
    printf( "Watch the LEDs on your keyboard!\n");

    for (i=0; i<10; ++i)          /* Run through the loop 10 times */
    {
        set_flag( CAPL );                /* Turn CAPS on */
        delay( TICKS(100) );            /* Wait 100 milliseconds */
        clr_flag( CAPL );                /* Turn CAPS off again */
        set_flag( NUML );                /* Turn NUM LOCK on */
        delay( TICKS(100) );            /* Wait 100 milliseconds */
        clr_flag( NUML );                /* Turn NUM off again */
        set_flag( SCRL );                /* Turn SCROLL LOCK on */
        delay( TICKS(100) );            /* Wait 100 milliseconds */
        clr_flag( SCRL );                /* Turn SCROLL LOCK off again */
    }

    for (i=0; i<10; ++i)          /* Run through the loop 10 times */
    {
        set_flag(CAPL | SCRL | NUML);    /* All three flags on */
        delay( TICKS(500) );            /* Wait 500 milliseconds */
        clr_flag(CAPL | SCRL | NUML);    /* All three flags off */
        delay( TICKS(500) );            /* Wait 500 milliseconds */
    }
}

```