```
                                                        /*
 ------------------------------------------------------------
 * Program to control a T6963C-based 240x64 pixel LCD display
 * written in Microsoft Quick C
 *
 * ------------------------------------------------------------
 */
 #include <stdio.h>
 #include <stdlib.h> // rand()
 #include <conio.h>  // inp() outp() kbhit()
 #include <string.h> // strlen()
 #include <graph.h>  // _settextposition(row,column) ie, (y,x)
 #include <math.h>   // cos(),sin()
 #include <time.h>
 /* ------------------------------------------------------------------
 * Pin numbers refer to pins on PC's DB25 parallel port connector.
 * Recall that SEL (pin 17), LF (14), and STROBE (1) control ouputs
 * are inverted, but Init (16) is true.
 *
 * LCD pin numbering for TOSHIBA TLX-711A-E0 (uses T6963C controller)
 *
 * LCD Pin ----- PC Port Pin Status Reg. bit
 * ------------------------------------------------------------------------------
 * C/D (8) <--> (17) /SEL 3
 * /WR (5) <--> (16) Init 2
 * /RD (6) <--> (14) /LF 1
 * /CE (7) <--> (1) /Strobe 0
 * ------------------------------------------------------------------------------
 * D0 (11) <--> (2) D0
 * D1 (12) <--> (3) D1
 * D2 (13) <--> (4) D2
 * D3 (14) <--> (5) D3
 * D4 (15) <--> (6) D4
 * D5 (16) <--> (7) D5
 * D6 (17) <--> (8) D6
 * D7 (18) <--> (9) D7
 * GND (2) <--> (25) GND
 * ------------------------------------------------------------------------------
 * FG (1) frame ground
 * +5V (3) LCD logic supply
 * -7.8V (4) LCD display contrast
 * FS (19) font select
 * RST (10) active low
 */

 // define statement ***********************************************************

 #define CEHI outp(pcont, (inp(pcont) & 0xfe) ) // take PC 1 HI
 #define CELO outp(pcont, (inp(pcont) | 0x01) ) // take PC 1 LO
 #define RDHI outp(pcont, (inp(pcont) & 0xfd) ) // take PC 14 HI
 #define RDLO outp(pcont, (inp(pcont) | 0x02) ) // take PC 14 LO
 #define WRHI outp(pcont, (inp(pcont) | 0x04) ) // take PC 16 HI
 #define WRLO outp(pcont, (inp(pcont) & 0xfb) ) // take PC 16 LO
 #define CDHI outp(pcont, (inp(pcont) & 0xf7) ) // take PC 17 HI
 #define CDLO outp(pcont, (inp(pcont) | 0x08) ) // take PC 17 LO
 #define DATAIN outp(pcont, (inp(pcont) | 0x20) )  // 8bit Data input
 #define DATAOUT outp(pcont, (inp(pcont) & 0xdf) ) // 8bit Data output


 /* ----- Definitions concerning LCD internal memory ------ */
 #define G_BASE 0x0200 // base address of graphics memory
 #define T_BASE 0x0000 // base address of text memory
 #define BYTES_PER_ROW 40 // how many bytes per row on screen
```

```
/* This will be 30 with 8x8 font, 40 with 6x8 font, for both
Text & Graphics modes. Font selection by FS pin on LCD module:
FS=High: 6x8 font. FS=Low: 8x8 font. */
/* ------------------------------------------------------------ */

#define sgn(x) ((x)>0?1:-1)
#define frand() ((float)rand()/RAND_MAX)
#define UI unsigned int
#define UL unsigned long

// prototypes ********************************************************************

void delay(UL d);       // delay proportional to "d" value
void dput(int byte);    // write data byte to LCD module
int dget(void);         // get data byte from LCD module
int sget(void);         // check LCD display status pbrt
void cput(int byte);    // write command byte to LCD module
void lcd_setup();       // make sure control lines are at correct levels
void lcd_init();        // initialize LCD memory and display modes
void lcd_print(char *string);   // send string of characters to LCD
void lcd_clear_graph();         // clear graphics memory of LCD
void lcd_clear_text();          // clear text memory of LCD
void lcd_xy(int x, int y);      // set memory pointer to (x,y) position (text)
void lcd_setpixel(int column, int row); // set single pixel in 240x64 array

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


#define BASE 0x378 // base address for parallel port LPT1:
UI pdata = BASE; // par.port data register
UI pstatus = BASE+1; // par.port status register
UI pcont = BASE+2; // par.port control register
#define home() dput(T_BASE%256);dput(T_BASE>>8);cput(0x24); // upper-left

#define XMAX 239 // limits of (x,y) LCD graphics drawing
#define XMIN 0
#define YMAX 63
#define YMIN 0
#define PI 3.1415926536


// MaIN ***************************************************************************


//*********************************************************************************
void main()
{
        int d1;         // data from port
        int s1;         // status register value
        int c1;         // control register value
        int i;          // generic counter
        int c;          // character to write to display
        float x,y;      // coordinates on graphics screen
        float xinc,yinc;
        float r,theta;          // polar coords. for point dV
        float theta_inc;
        unsigned int cc;
        char string[320];       // string to print to display
        char tmpbuf[128];       // time buffer

        _clearscreen(_GCLEARSCREEN); // Clears PC's display screen, not LCD
        printf("LCD control program. jpb 5/3/97\n");
        printf("<esc> to quit.\n");
```

```
lcd_setup(); // make sure control lines are at correct levels
printf("Setup lcd sucessfully.\n");
lcd_init(); // initialize LCD memory and display modes
printf("Initialized lcd sucessfully.\n");
do
{
        // outer loop: keypress exits
        /* --- display characters available from LCD's ROM CharGen ------ */
        lcd_clear_text();
        cput(0x97); // Graphics & Text ON, cursor blinking
        for (c=0;c<10;c++)
        {
                lcd_xy(0,0); // write text from upper left corner
                /*
                //for (i=0;i<320;i++)
                //{
                        //cc = (unsigned int) (c+i)%0x7f; // display all
characters
                        //dput(cc); cput(0xc0); // write char, inc ptr.
                //}
                */

                strcpy(string, "AZ Displays, Inc.");
                lcd_print(string);
                lcd_xy(0,1); // first character, second line
                lcd_print("Complete LCD Solutions");
                lcd_xy(5,2);
                lcd_print("75 Columbia, Aliso Viejo, CA-92656");
                lcd_xy(0,3);
                lcd_print("www.azdisplays.com");
                lcd_xy(10,5);
                lcd_print("sales@azdisplays.com");
                lcd_xy(0,7);
                lcd_print("AZ Displays");
                /* Set time zone from TZ environment variable. If TZ is not
set,
                * PST8PDT is used (Pacific standard time, daylight savings).
                */
                tzset();

                /* Display DOS-style date and time, in refresh loop */
                for (i=0;i<500;i++)
                {
                        lcd_xy(22,0);
                        _strtime( tmpbuf );
                        lcd_print( tmpbuf ); // DOS-style time
                        lcd_print(" "); // intervening space
                        _strdate( tmpbuf );
                        lcd_print( tmpbuf ); // DOS-style date
                        if (kbhit()) break;
                }
                // delay(400000); // about 0.5 sec on 75 MHz Pentium

        }

        if (kbhit())
                break;


        /* ------ bouncing line graphics "screensaver" demo ---------- */

        lcd_clear_graph(); // fill graphics memory with 0x00
        cput(0x98); // Graphics ON, Text OFF
```

```
                x=(XMAX-XMIN)/2;
                y=(YMAX-YMIN)/2;
                r=0.3;
                theta = 2*PI*frand();
                theta_inc = (0.01 * frand()) - 0.005;
                for (c=0;c<12000;c++)
                {
                        lcd_setpixel((int)x,(int)y); // draw pixel on LCD screen
                        xinc = r*cos(theta);
                        yinc = r*sin(theta);
                        theta += theta_inc;
                        x += xinc;
                        y += yinc;
                        if (x>XMAX) {x=XMAX; theta = PI-theta;}
                        if (y>YMAX) {y=YMAX; theta = -theta;}
                        if (x<XMIN) {x=XMIN; theta = PI-theta;}
                        if (y<YMIN) {y=YMIN; theta = -theta;}
                        if (kbhit())
                                break;
                        // delay(1000); // delay calibrated on 75 MHz Pentium

                } // end for(c)


                if (kbhit())
                        getch();

        } while (!kbhit());

} // end main()



/* Block writes would, I think, run faster if you used the DATA AUTO
mode, eg command 0xB0. I didn't bother.
*/

// lcd_clear_graph ***********************************************************

// ***************************************************************************
void lcd_clear_graph() // clear graphics memory of LCD
{
        int i;
        dput(G_BASE%256);
        dput(G_BASE>>8);
        cput(0x24); // addrptr at address G_BASE
        for (i=0;i<2560;i++)
        {
                dput(0); cput(0xc0); // write data, inc ptr.
        } // end for(i)

} // end lcd_clear_graph()


// lcd_clear_text ************************************************************

// ***************************************************************************
void lcd_clear_text()
{
        int i;
        dput(T_BASE%256);
        dput(T_BASE>>8);
```

```
        cput(0x24); // addrptr at address T_BASE
        for (i=0;i<320;i++) {
        dput(0); cput(0xc0); // write data, inc ptr.
        } // end for(i)

} // lcd_clear_text()


// lcd_print ***************************************************************

// *************************************************************************
void lcd_print(char *string) // send string of characters to LCD
{
        int i;
        int c;
        for (i=0;i<strlen(string);i++)
        {
                c = string[i] - 0x20; // convert ASCII to LCD char address
                if (c<0)
                        c=0;
                dput(c);
                cput(0xc0); // write character, increment memory ptr.
        } // end for
} // end lcd_string


// lcd_setpixel ************************************************************

// *************************************************************************
void lcd_setpixel(int column, int row) // set single pixel in 240x64 array
{
        int addr; // memory address of byte containing pixel to write
        addr = G_BASE + (row*BYTES_PER_ROW) + (column/6);
        dput(addr%256); dput(addr>>8); cput(0x24); // set LCD addr. pointer
        cput(0xf8 | (5-(column%6)) ); // set bit-within-byte command

} // end lcd_setpixel()


// lcd_xy ******************************************************************

// *************************************************************************
void lcd_xy(int x, int y) // set memory pointer to (x,y) position (text)
{
        int addr;
        addr = T_BASE + (y * BYTES_PER_ROW) + x;
        dput(addr%256); dput(addr>>8); cput(0x24); // set LCD addr. pointer

} // lcd_xy()

// delay *******************************************************************

// *************************************************************************
void delay(UL d) // delay proportional to "d" value
{
        UL i;
        double a;
        a = 1.000;
        for (i=0;i<d;i++)
        {
                a = a / 1.001;
        }
} // end delay()
```

```
/* ========================================================================
* Low-level I/O routines to interface to LCD display
* based on four routines:
*
* dput(): write data byte
* cput(): write control byte
* dget(): read data byte (UNTESTED)
* sget(): read status
*
* If you are confused about outp() and inp() then look at the sample
* code to see how do they work
* outp(0x29A, 0x69); // outp(unsigned int port, int value);
* will write the byte-sized value 0x69 to port 0x29A
*
*
* value = inp(0x76B);
* will just assign the variable "value" the byte/word receive from port 0x76B.
========================================================================*/


// lcd_setup ***********************************************************

// *********************************************************************
void lcd_setup() // make sure control lines are at correct levels
{
        CEHI; // disable chip
        RDHI; // disable reading from LCD
        WRHI; // disable writing to LCD
        CDHI; // command/status mode
        DATAOUT; // make 8-bit parallel port an output port

} // end lcd_setup()


// lcd_init ************************************************************

// *********************************************************************
void lcd_init() // initialize LCD memory and display modes
{
        dput(G_BASE%256);
        dput(G_BASE>>8);
        cput(0x42); // set graphics memory to address G_BASE
        dput(BYTES_PER_ROW%256);
        dput(BYTES_PER_ROW>>8);
        cput(0x43); // n bytes per graphics line
        dput(T_BASE%256);
        dput(T_BASE>>8);
        cput(0x40); // text memory at address T_BASE
        dput(BYTES_PER_ROW%256);
        dput(BYTES_PER_ROW>>8);
        cput(0x41); // n bytes per text line
        cput(0x80); // mode set: Graphics OR Text, ROM CGen
        cput(0xa7); // cursor is 8 lines high
        dput(0x00);
        dput(0x00);
        cput(0x21); // put cursor at (x,y) location
        cput(0x97); // Graphics & Text ON, cursor blinking
        // (For cursor to be visible, need to set up position)

} // end lcd_init()
```

```
// sget(): read status

// ***************************************************************************
int sget(void) // get LCD display status byte
{
        int lcd_status;
        DATAIN; // make 8-bit parallel port an input
        CDHI; // bring LCD C/D line high (read status byte)
        RDLO; // bring LCD /RD line low (read active)
        CELO; // bring LCD /CE line low (chip-enable active)
        lcd_status = inp(pdata); // read LCD status byte
        CEHI; // bring LCD /CE line high, disabling it
        RDHI; // deactivate LCD read mode
        DATAOUT; // make 8-bit parallel port an output port
        return(lcd_status);

} // sget()

// dput ********************************************************************
// dput(): write data byte
// write data byte to LCD module over par. port
// assume PC port in data OUTPUT mode

// ***************************************************************************
void dput(int byte)

{
        do
        {

        } while ((0x03 & sget()) != 0x03); // wait until display ready
        CDLO;
        WRLO; // activate LCD's write mode
        outp(pdata, byte); // write value to data port
        CELO; // pulse enable LOW > 80 ns (hah!)
        CEHI; // return enable HIGH
        WRHI; // restore Write mode to inactive
        // using my P5/75 MHz PC with ISA bus, CE stays low for 2 microseconds
} // end dput()


// dget ********************************************************************

// ***************************************************************************
int dget(void) // get data byte from LCD module
{
        int lcd_byte;
        do
        {

        } while ((0x03 & sget()) != 0x03); // wait until display ready
        DATAIN; // make PC's port an input port
        WRHI; // make sure WRITE mode is inactive
        CDLO; // data mode
        RDLO; // activate READ mode
        CELO; // enable chip, which outputs data
        lcd_byte = inp(pdata); // read data from LCD
        CEHI; // disable chip
        RDHI; // turn off READ mode
        DATAOUT; // make 8-bit parallel port an output port
        return(lcd_byte);
```

```
} // dget()

// cput ************************************************************************

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

void cput(int byte) // write command byte to LCD module
// assumes port is in data OUTPUT mode
{
        do
        {

        } while ((0x03 & sget()) != 0x03); // wait until display ready

        outp(pdata, byte);      // present data to LCD on PC's port pins
        CDHI;                   // control/status mode
        RDHI;                   // make sure LCD read mode is off
        WRLO;                   // activate LCD write mode
        CELO;                   // pulse ChipEnable LOW, > 80 ns, enables LCD I/O
        CEHI;                   // disable LCD I/O
        WRHI;                   // deactivate write mode

} // cput()
```