AN-005 - Interfacing Graphic LCD Modules to the DOS Stamp™

Topic

This application note describes how to interface a graphic LCD module to the DOS Stamp™.

In this application note the following items are used:

- 1 DOS Stamp embedded computer (Bagotronix)
- 1 240 x 128 monochrome graphic LCD module with built-in T6963C controller (Densitron)
- 1 DOS Stamp JP2 cable
- 1 piece of experimenter's prototyping board with solder pads
- Various electronic parts as shown in Figure 1
- +5V and -20V power supply

The schematic is shown in Figure 1. A photo of the experimental setup is shown in Figure 2.

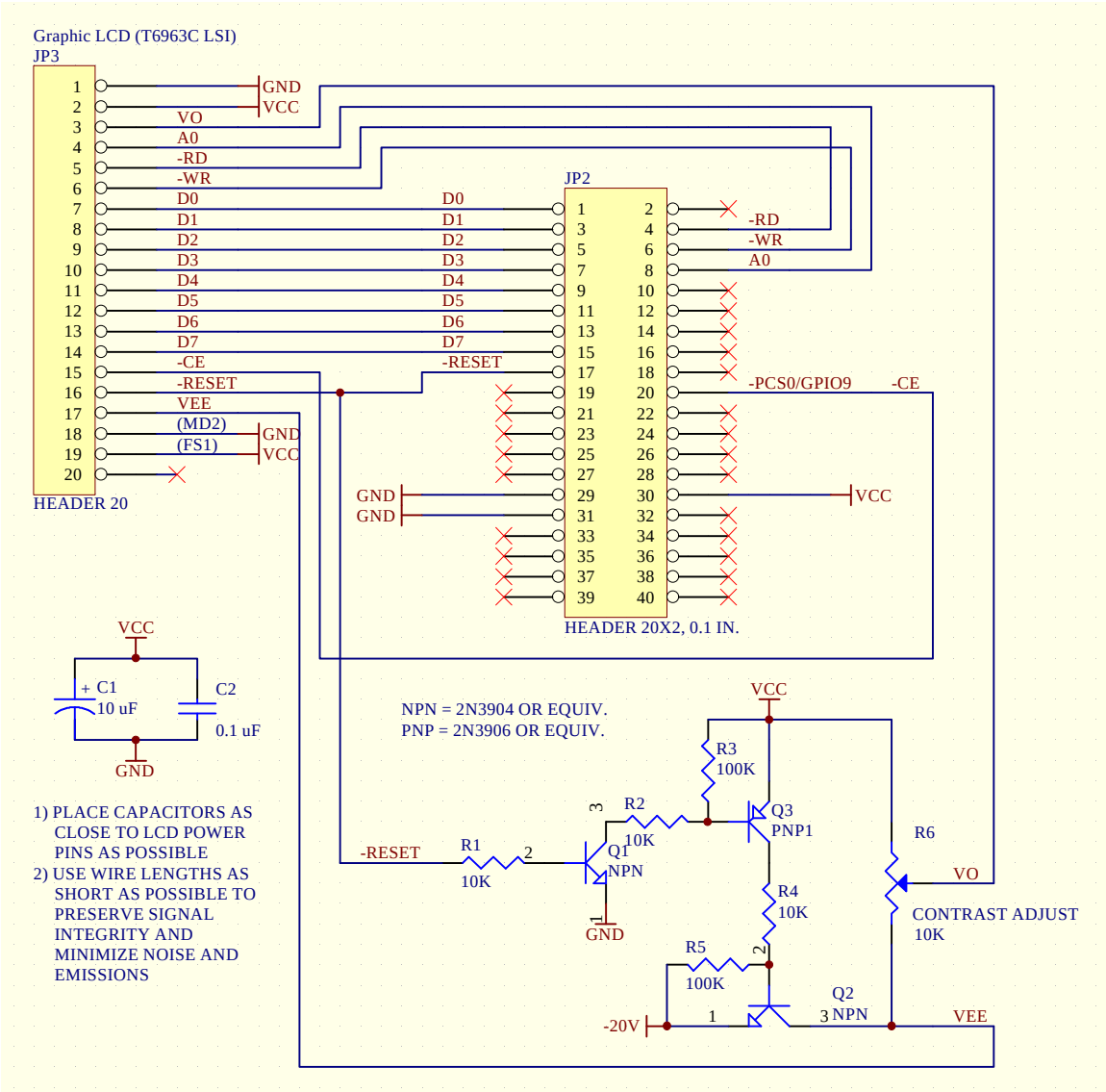

Figure 1

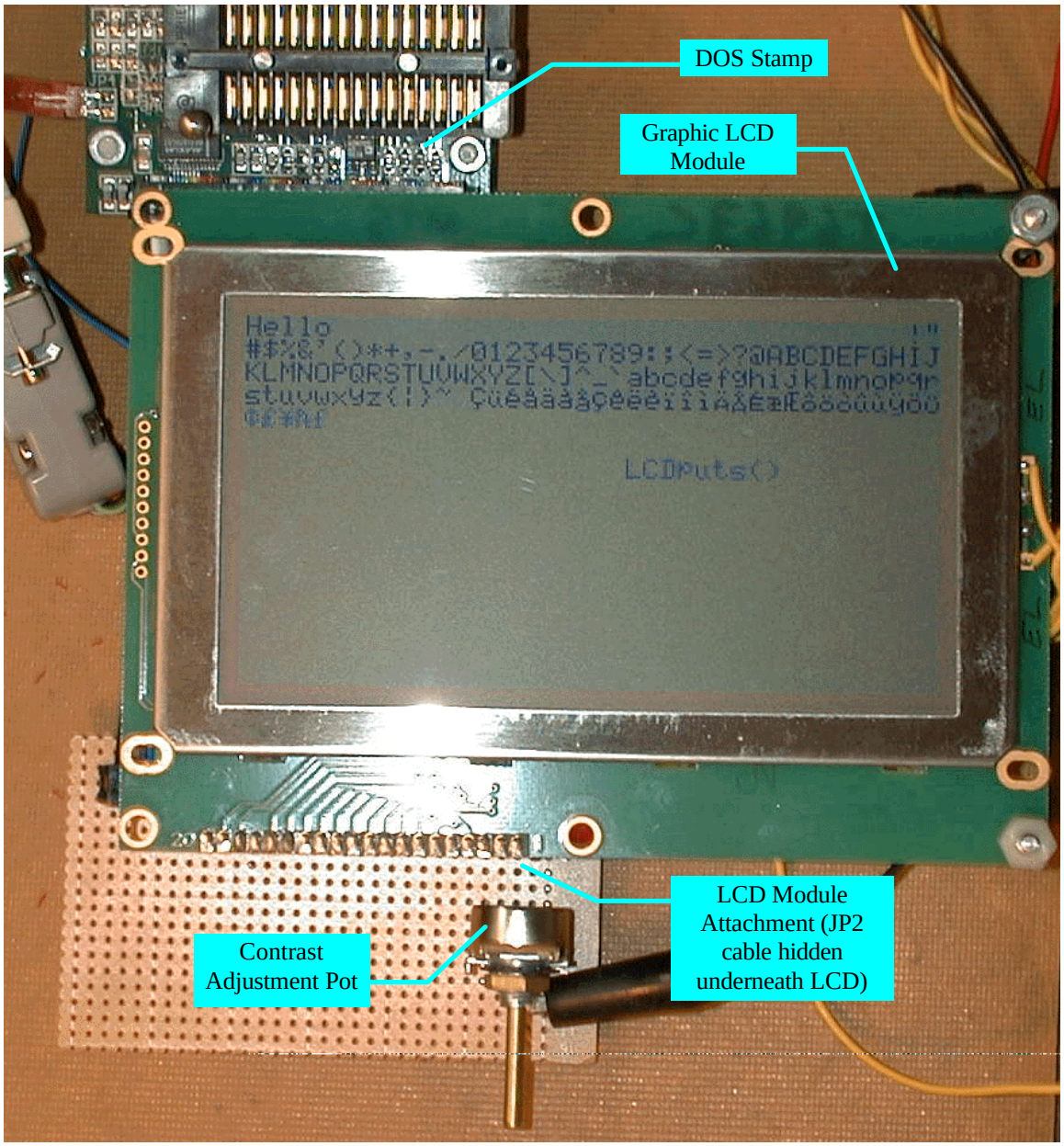

Figure 2

Theory of Operation

The LCD module connects to the DOS Stamp via the Simple Bus Interface (SBI). For more information on the SBI, refer to the DOS Stamp User's Manual. The SBI allows easy connection of digital devices to the DOS Stamp.

The SBI peripheral chip select bit 0 (-PCS0) is connected to the LCD module chip enable input -CE. Note that for -PCS0/GPIO9 (JP2 pin 20) to function as -PCS0, it must be configured by software to "normal function" (see DSGLCD.C test program). This must be done before any software interaction with the LCD module. The -RESET output of the DOS Stamp is used to reset the LCD upon powerup.

For the LCD module to work, it must be supplied with three voltages: +5V for logic (VCC), -20V for bias (VEE), and a variable (-10V to -20V, VO) voltage for contrast. Also, to avoid damage to the LCD, VCC must be applied before VEE is applied, and VEE must be removed before VCC is removed. Q1, Q2, and Q3 comprise a voltage switch for VEE. When the DOS Stamp is first powered up, -RESET is held low by the voltage monitor chip on the DOS Stamp. Once VCC is stable, -RESET is driven high, which turns on Q1, Q3, and Q2, applying VEE to the LCD module. When VCC falls below acceptable levels, -RESET is driven low by the voltage monitor chip. This causes Q1, Q3, and Q2 to turn off, removing VEE from the LCD module.

Due to the high speed nature of the digital signals used in the DOS Stamp, it is necessary to observe a few precautions when designing circuitry that connects to the SBI. Keep wire lengths to a minimum - 6 to 8 inches (including JP2 cable) is the maximum distance the SBI signals should be run if no termination, ground plane, or impedance control is to be used. Use heavy wires (22 AWG or better) or traces (>= 25 mil) for the VCC and GND paths. Place C1 and C2 decoupling capacitors as close to the LCD VCC and GND pins as possible. If these precautions are not observed, the LCD module may not work properly.

The Software

The graphic LCD module software was written in Borland C v3.0. Other C compilers and versions can be used, although some minor modifications may be required. The program consists of the following files, compiled and linked to produce a 16-bit DOS EXE:

- DSGLCD.C
- DOSSTAMP.C

Using the TRANSFER.EXE utility, the program DSGLCD.EXE was transferred to the DOS Stamp

The source file for the DSGLCD.EXE program is listed here and is also available with the other source files in ZIPped form from the Bagotronix website.

Program Listing

```
/*      DOS Stamp Graphic LCD Test Program
        (C)1999 Bagotronix Inc.
        Author: Ivan Baggett

        Written in Borland C v3.0.
        Use this program as an example on how to operate the DOS S   tamp with a
        graphic LCD module based on the T6963C controller.
*/

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

#include "stdinc.h"
#include "dosstamp.h"

#define PCS0    16              // PCS0# function is on PIO16 pin
#define LCD_BASE        PCS0_BASE       // base I/O address of LCD
#define LCD_STATUS      (LCD_BASE+1)
#define LCD_DATA        LCD_BASE
#define LCD_COMMAND     (LCD_BASE+1)

#define COMMAND 1
#define DATA 0

char inittbl[][2] = {
        {       COMMAND,        0x80    },
        {       DATA,           0               },
        {       DATA,           0               },
        {       COMMAND,        0x42    },
        {       DATA,           0x28    },
```

```
        {       DATA,           0               },
        {       COMMAND,        0x43    },
        {       DATA,           0               },
        {       DATA,           0x17    },
        {       COMMAND,        0x40    },
        {       DATA,           0x28    },
        {       DATA,           0               },
        {       COMMAND,        0x41    },

        {       DATA,           0               },
        {       DATA,           0x17    },
        {       COMMAND,        0x24    },
        {       DATA,           0x24    },
        {       COMMAND,        0xc0    },
        {       DATA,           0x25    },
        {       COMMAND,        0xc0    },
        {       DATA,           0x2e    },
        {       COMMAND,        0xc0    },
        {       COMMAND,        0x94    }
};

BOOLEAN done = FALSE;


WORD waitforkey (void) {
        WORD c;

        while (!kbhit());
        if (!(c = getch()))
                c = getch();
        return c;
}


UWORD LCDstatus (void) {
        return ((UWORD)inportb (LCD_STATUS));
}


void LCDwrite (BOOLEAN op, BYTE c) {
        WORD u;

        while ((LCDstatus() & 0x0003) != 0x0003);       // wait for LCD
        if (op == COMMAND)
                outportb (LCD_COMMAND, c);      // command
        else
                outportb (LCD_DATA, c);         // data
}


void LCDputc (BYTE c) {
        LCDwrite (DATA, c - 0x20);      // the char adjusted for CGROM offset
        LCDwrite (COMMAND, 0xc0);       // data write autoincrement
}


int LCDputs (const char *s) {
        char *t;

        t = (char *) s;                 /* copy pointer */
        while (*t)
                LCDputc (*t++);
        return 1;
}


int main (void) {
        WORD c, i;
        UWORD u;

        // set up PIO16 as Simple Bus Interface chip select PCS0#
        disable();
        PIOconfig (PCS0, PIOCFG_NORMAL, 1);
        enable();
```

```
        // LCD initialize
        for (i = 0; i < sizeof inittbl / sizeof inittbl[0]; i++) {
                LCDwrite (inittbl[i][0], inittbl[i][1]);
        }

        // clear display RAM
        LCDwrite (DATA, 0);
        LCDwrite (DATA, 0);
        LCDwrite (COMMAND, 0x24);               // address pointer set
        LCDwrite (COMMAND, 0xb0);               // data auto write
        for (i = 0; i < 8192; i++) {
                while ((LCDstatus() & 0x0008) != 0x0008);       // wait for LCD aut o write
                LCDwrite (DATA, 0);
        }
        LCDwrite (COMMAND, 0xb2);               // auto reset

        // go to text home
        LCDwrite (DATA, 0);
        LCDwrite (DATA, 0x17);
        LCDwrite (COMMAND, 0x24);               // address pointer set

        // Hello
        LCDputc ('H');
        LCDputc ('e');
        LCDputc ('l');
        LCDputc ('l');
        LCDputc ('o');
        // All characters test
        for (i = 0; i < 256; i++) {
                LCDputc ((BYTE)i);
        }

        // string test
        LCDputs (" LCDputs()");

        printf ("\n\nType to LCD, <space> = done");
        done = FALSE;
        while (!done) {
                while (!kbhit());
                        c = getch ();
                if (!c)
                        getch();
                if (c == ' ')
                        done = TRUE;
                LCDputc ((BYTE)c);
        }

        return 0;
}
```