Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (617) 461-3881, FAX: (617) 461-3010, EMAIL: dsp_applications@analog.com

How to Interface an LCD to the 21xx and 2106x Family DSP's

Last Modified: *9/21/1997*

Overview

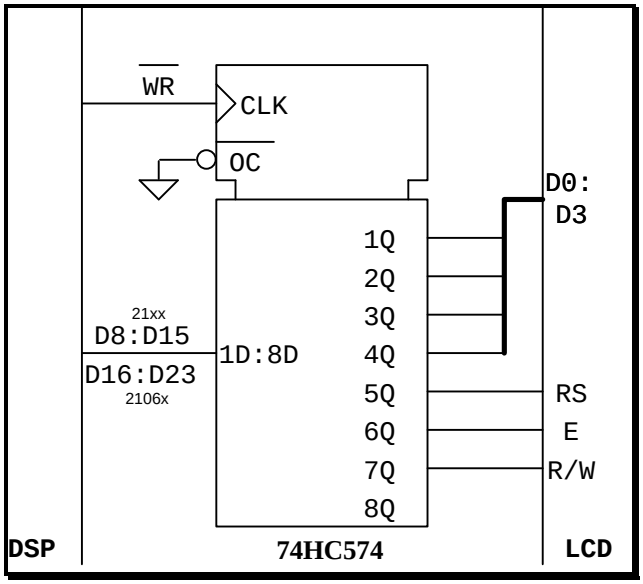
This document contains example code and hardware to interface an Optrex dot-matrix type LCD module to both the ADSP21xx and ADSP2106x family digital signal processors. Optrex brand displays come in a wide range of display sizes (16 characters x 1 line to 40 characters by 4 lines) and can be easily interfaced to both family of DSP's. Below is list of features:

- 192 kinds of displayable characters, numerals, symbols and special characters stored in internal ROM
- Programmable character RAM
- Various display functions include: clear display, cursor to home, on/off cursor, blink character, shift display, shift character, read/write display
- Compact and lightweight design
- Low power consumption

Hardware Interface

Because the micro-controller within the LCD display is much slower than the DSP, the minimum timing specs on certain control signals in the LCD extend far past the maximum timing specs of the DSP. To compensate for this, a CMOS octal D-type flip-flop is used between the two. Both the control signals and data lines pass through the flip-flop so they can be held as long as necessary.

This design is the most simple and cheap, requiring only one octal flip-flop (74HC574). It will work for both the ADSP21xx family and ADSP2106x family DSP's. The limitations on such a simple design is that we cannot read back data from the LCD and all data transfers are 4 bits wide (vs. 8 bits). Most applications, however, would not necessitate the ability to read back from the LCD or write 8-bits at a time. Below is a simple schematic.



Software

Each OPTREX LCD has an initialization sequence that must be performed upon power up. This configures the display mode and data width (4 or 8 bit). When entering any 8-bit LCD data into a data buffer within the DSP, it is important to remember that we are writing 4 bits at a time and the most-significant nibble is written first. The sub-routine LCD_INIT listed on the following pages writes the initialization nibbles to the LCD from a buffer in memory called InitCodes.

When sending initialization or configuration data to the LCD, the RS line is low. To send the bytes 0x86, 0xC2, 0xFF for example, our data buffer in the DSP would look like this: 0x8, 0x6, 0xC, 0x2, 0xF, 0xF. When sending characters, on the other hand, the RS line (bit 5) is high. To send the same data as before as character data, the data buffer in the DSP would look like this: 0x18, 0x16, 0x1C, 0x12, 0x1F, 0x1F. Initialization and configuration data must be stored with bit 5 cleared and character data must be stored with it set.

The following 2 pages contain example code for both the ADSP21xx and ADSP2106x family DSP's.

Related Websites:
http://www.cis.uoguelph.ca/~mac/lcd.html
http://www.avnet.se/avnet_times/september96/optrex.html

ADSP 21xx Code

LCD_INIT

Inputs : n-length buffer in program memory

Description : Writes buffer of initialization data to LCD.

```
lcd_init:
          call delay_15ms;       /* delay after power up */
          i7 = ^InitCodes; l7 = %InitCodes; m7 = 1;
          cntr = %InitCodes-1;
          DO end_1 Until CE;
                  ay0 = pm(i7,m7);
end_1:            CALL lcd_write;            /* output command to display */
          rts;
```

LCD_WRITE

Inputs : 4 LSB's of ay0 hold nibble data to be sent to LCD

Description : performs a nibble-write to the LCD. Sets and clears all necessary control signals and takes care of all timing issues. Assume LCD is located in external memory space at location 0x0.

```
lcd_write:
          ar = 0;                /* ensure that the E (enable) is low */
          dm(0x0) = ar;
          ar  = ar or ay0;     /* set up the data, RS, and RW */
          dm(0x0) = ar;
          nop; nop;        /* delay at least 60ns before E goes high */
          ax0 = b#0000000000100000;        /* ar = setbit LCD_E_BIT of ar */
          af = pass ar; ar = ax0 or af;
          dm(0x0) = ar;         /* E pin is high now */
          cntr = 6;                /* pause for enable pulse width */
          do data_hold until CE;
data_hold:        nop;
          ar = ax0 xor af;       /* ar = clrbit LCD_E_BIT of ar */
          dm(0x0) = ar;         /* return E to low */
          ar = 0;
          dm(0x0) = ar;
          call delay_15ms;/* for subsequent writes, need this delay */
          rts;
```

DELAY_15MS

Description : Waits for 15ms with 40Mhz DSP clock speed.

```
delay_15ms:
          cntr = 150;
          do end_15 until CE;
                  cntr = 3333;
                  do end_16 until ce;
end_16:           nop;
end_15:   nop;
          rts;
```

ADSP 2106x Code

LCD_INIT

Inputs : n-length buffer in program memory

Description : Writes buffer of initialization data to LCD.

```
lcd_init:
          call delay_15ms;       /* delay after power up */
          b8 = InitCodes; l8 = @InitCodes; m8 = 1;
          lcntr = @InitCodes-1;
          DO end_1 Until LCE;
                  r1 = pm(i8,m8);
end_1:            CALL lcd_write;             /* output command to display */
          rts;
```

LCD_WRITE

Inputs : 4 LSB's of r1 hold nibble data to be sent to LCD

Description : performs a nibble-write to the LCD. Sets and clears all necessary control signals and takes care of all timing issues. Assume LCD is located in external memory space at location 0x400000.

```
lcd_write:
          r0 = 0;                 /* ensure that the E (enable) is low */
          dm(0x400000) = r0;
          r0  = r0 or r1;     /* set up the data, RS, and RW */
          dm(0x400000) = r0;
          nop; nop;        /* delay at least 60ns before E goes high */
          r2 = b#0000000000100000;        /* ar = setbit LCD_E_BIT of ar */
          r3 = pass r0; r0 = r2 or r3;
          dm(0x400000) = r0;          /* E pin is high now */
          lcntr = 6;                  /* pause for enable pulse width */
          do data_hold until LCE;
data_hold:        nop;
          r0 = r2 xor r3;       /* ar = clrbit LCD_E_BIT of ar */
          dm(0x400000) = r0;          /* return E to low */
          r0 = 0;
          dm(0x400000) = r0;
          call delay_15ms;/* for subsequent writes, need this delay */
          rts;
```

DELAY_15MS

Description : Waits for 15ms with 33Mhz DSP clock speed.

```
delay_15ms:
          lcntr = 150;
          do end_15 until LCE;
                  lcntr = 4000;
                  do end_16 until LCE;
end_16:           nop;
end_15:   nop;
          rts;
```