Interfacing a Hantronix 320 x 240 Graphics Module to an 8-bit Microcontroller

Introduction:
Due to its thin profile, light weight, low power consumption and easy handling, liquid crystal graphic display modules are used in a wide variety of applications. The 320 x 240 (¼ VGA) LCD display is very popular in a number of different computing environments. It is for this reason that a controller is not included on the module.

Possible choices of controllers include an embedded 8-bit microcontroller with an LCD controller, such as the Epson/S-MOS SED1335 or the OKI MSM6255/6355. Some embedded microcontrollers, such as the National NS486SXF, have built-in LCD controllers and will interface directly to the display.

For PC based embedded controllers like the Intel 386/486EX, a VGA controller chip, such as the Chips and Technology F65545 or the Vadem VG-660, is the best choice. If the display is to be run directly from a PC, a number of VGA cards are available that will operate with this display. A number of single board computers are available with LCD display outputs.

This application note will deal with one of the most popular application environments, the 8-bit embedded microcontroller. The example detailed here is based on a Phillips 87C751 micro-controller driving an Epson/S-MOS SED1335 LCD controller.

Functional Description:
The Hantronix 320 x 240 series of displays have an industry standard 4-bit parallel interface. This interface requires the controller to continuously refresh the display and to maintain the video display RAM.

Before the display can be used the microcontroller must first send a series of initialization bytes to the LCD controller to set up its operational parameters and to describe the display to the controller.

Once initialized the application microcontroller can send text or graphic data to the LCD controller where it will be formatted and stored in the display RAM. Coincident with these RAM updates the LCD controller is continuously reading data from the display RAM, serializing it and sending it to the display. The application microcontroller doesn't have direct access to the display RAM and must send all data and commands to the LCD controller chip.

Schematic:
The 87C751 microprocessor is connected to the LCD controller chip via parallel I/O ports in this example. It could also be connected to the processor's data bus and be mapped into the processor's data memory area. See figure 1.

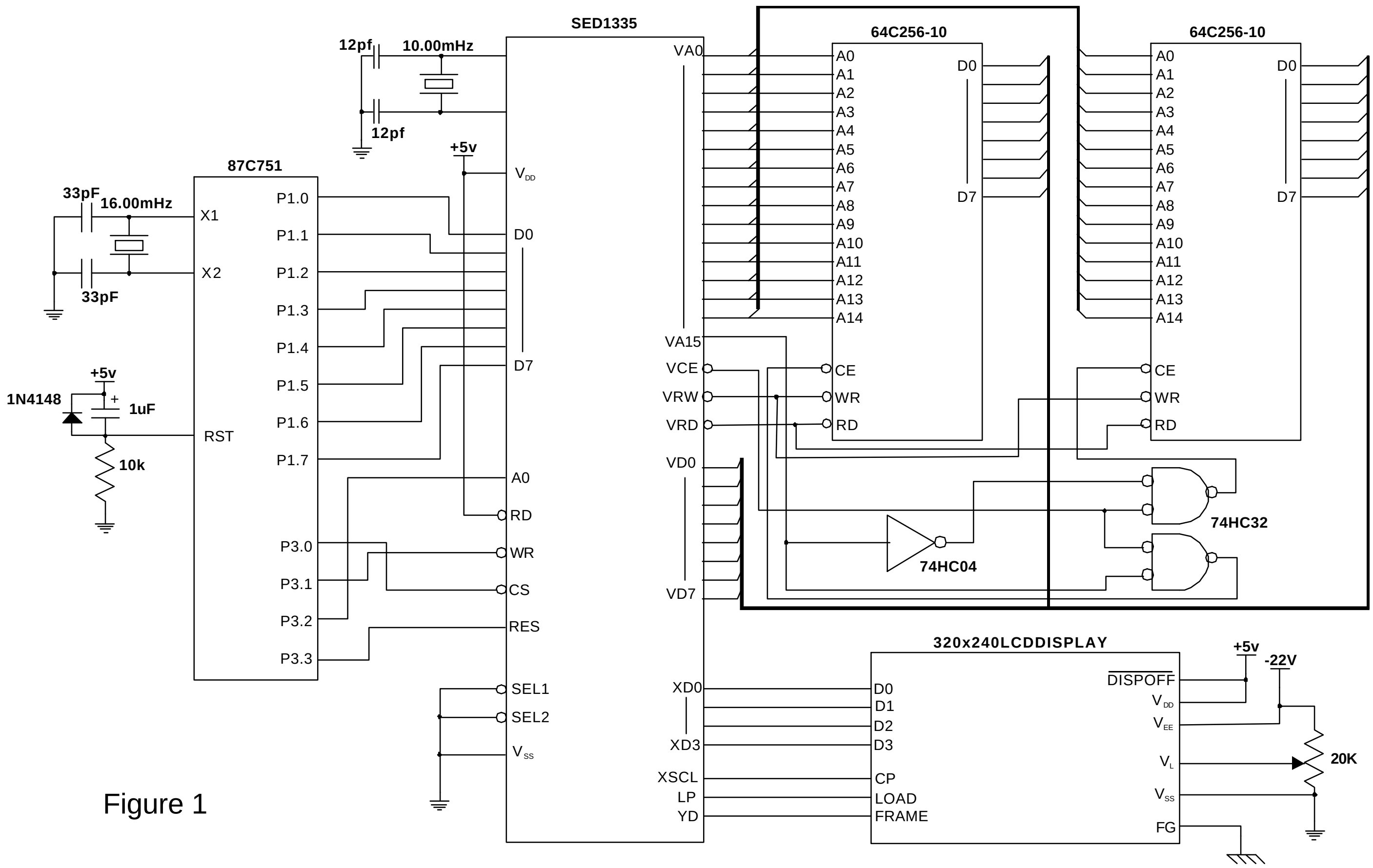


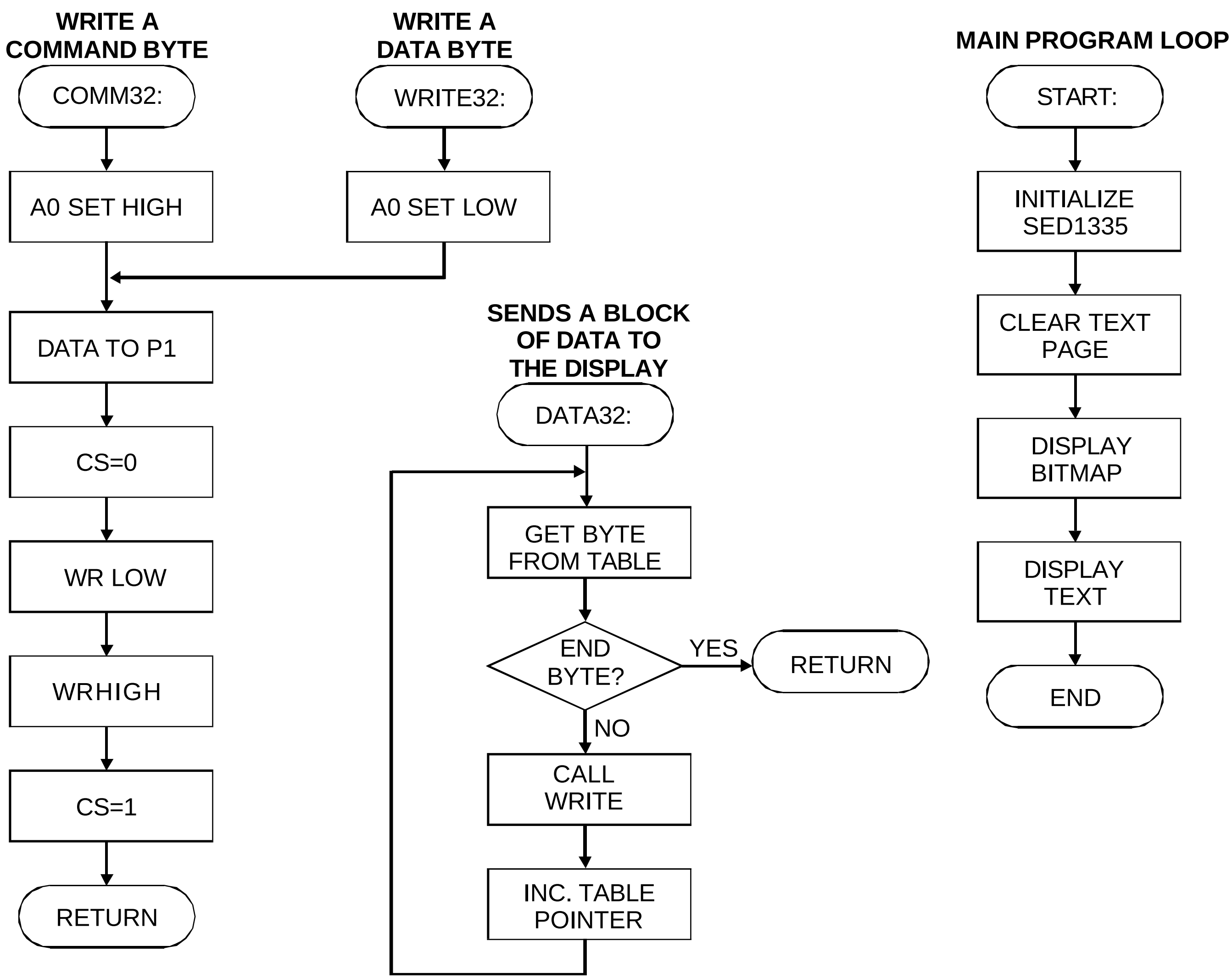
Figure 1

Software:
The sample program here is written in 8051 assembly code and is designed to work with the hardware shown in Figure 1. It first sends a series of command bytes followed by the appropriate parameters to the LCD controller to initialize it. The controller is initialized with one text page at memory location 0000-04afh and one graphics page at 4b0h-2a2fh. This will allow for 1200 text characters arranged as 30 lines of 40 columns each. The graphics page is 9600 bytes in size to accommodate a full screen of data. The display mode is set with both screens on and the text overlaying the graphics in the "exclusive or" mode.

The text area of memory is then cleared by storing 20h, a space character, in all 1200 locations. The graphics page is then filled with the image of a bonsai tree. Four lines of text are then displayed.

The code example is not written to be efficient but to be as simple to follow as possible.

Software Flowchart:

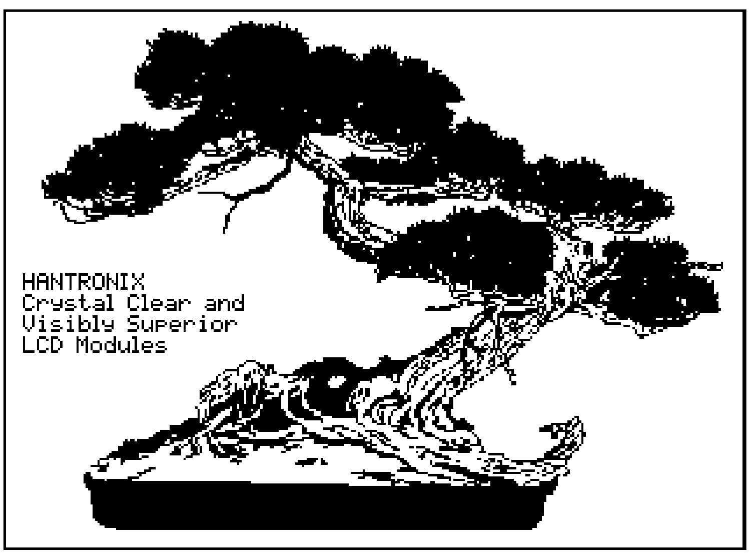


Initialization:
Before the LCD controller can accept or display data or text it must be initialized. This is usually done immediately after the system is powered up. The following chart lists the initialization commands and the parameters that accompany them along with a brief explanation of the function of each.

Initialization bytes:

COMMAND	CODE	PARAMETER	FUNCTION
SYSTEM SET	40h	30h	LCD PANEL HARDWARE SETUP
		87h	CHARACTER WIDTH [7] IN PIXELS
		07h	CHARACTER HEIGHT [7+1] IN PIXELS
		27h	ADDRESS RANGE FOR 1 TEXT LINE
		39h	LINE LENGTH IN CHARACTERS [40-1=39]
		efh	NUMBER OF LINES PER FRAME [240]
		28h	HORIZONTAL ADDRESS RANGE (TEXT) [40]
		0	
SCROLL	44h	0	SETS THE SCROLL START ADDRESS AND THE NUMBER OF LINES PER SCROLL BLOCK
		0	
		efh	
		b0h	
		04h	
		efh	
		0	
		0	
		0	
		0	
CURSOR FORM	5dh	04h	CURSOR FORM AND SIZE [BLOCK, 4 PIXELS WIDE, 6 PIXELS HIGH]
		86h	
CURSOR DIRECTION	4ch		CURSOR DIRECTION IN AUTO WRITE MODE [RIGHT]
HORIZONTAL SCROLL RATE	5ah	00h	HORIZONTAL SCROLL RATE, [1] PIXEL AT A TIME
OVERLAY	5bh	01h	TEXT/GRAPHICS OVERLAY MODE [EXOR]
DISPLAY ON/OFF	59h	16h	DISPLAY ON/OFF [ON]

Displayed image:



Software:

```
$MOD751

; ***************************************************
; *                                                 *
; *         HDM3224 Application Note V1.0           *
; *                                                 *
; ***************************************************

; The processor clock speed is 16MHz.
; Cycle time is .750mS.
; Demo software to display a bonsai
; tree bitmap image and 4 lines of
; text on a 320 x 240 LCD.

        org     00h
        ljmp    start           ;program start

        org 100h

; Initialize the 32241
; Text page 0000h 04afh
; Graphics page 04b0h 2a2fh

start:

        mov     r1,#40h         ;system set
        lcall   comm32
        mov     dptr,#msg1      ;ss param
        lcall   data32
        mov     r1,#44h         ;scroll
        lcall   comm32
        mov     dptr,#msg2      ;scroll param
        lcall   data32
        mov     r1,#5dh         ;csr form
        lcall   comm32
        mov     dptr,#msg3      ;csr param
        lcall   data32
        mov     r1,#4ch         ;csrdir
        lcall   comm32
        mov     r1,#5ah         ;hdot scr
        lcall   comm32
        mov     dptr,#msg18     ;hdot param
        lcall   data32
        mov     r1,#5bh         ;overlay
        lcall   comm32
        mov     dptr,#msg4      ;ovrly param
        lcall   data32
        mov     r1,#59h         ;disp on/off
        lcall   comm32
        mov     dptr,#msg5      ;disp param
        lcall   data32

; clear the text page
        lcall   clrtext

; display bitmap
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg6
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg12
        lcall   data32

; display text
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg7
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg14
        lcall   data32
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg8
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg15
        lcall   data32
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg9
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg16
        lcall   data32
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg10
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg17
        lcall   data32
        sjmp    $               ;stop

;**************************************************
;SUBROUTINES

; comm32 sends the byte in R1 to the
; 32241 display as a command

comm32:
        setb    p3.2            ;a0=1=command
comm321:
        mov     a,r1            ;get data byte
        mov     p1,a
        clr     p3.0            ;CS the display
        clr     p3.1            ;strobe
        setb    p3.1
        setb    p3.0
        ret

; write32 sends the byte in R1 to the
; 32241 display as a data byte.

write32:
        clr     p3.2            ;a0=0=data
        sjmp    comm321

; data32 sends the message pointed to
; by the DPTR to the 32241 display.

data32:
        clr     a               ;get the byte
        movc    a,@a+dptr
        cjne    a,#0a1h,data321;done?
        ret
```

```
data321:
        mov     r1,a
        lcall   write32         ;send it
        inc     dptr
        sjmp    data32          ;next byte

; Clear text RAM on the 3224
clrtext:
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg13     ;cursor param
        lcall   data32
        mov     r1,#42h         ;mwrite
        lcall   comm32
        mov     dptr,#msg11     ;all spaces
        lcall   data32
        mov     r1,#46h         ;set cursor
        lcall   comm32
        mov     dptr,#msg6
        lcall   data32
        ret

;*************************************************
; TABLES AND DATA

; Initialization parameters for 3224.

msg1:
        db      30h,87h,07h,27h ;system set
        db      39h,0efh,28h,0h,0a1h

msg2:
        db      0,0,0efh,0b0h   ;scroll
        db      04h,0efh,0,0
        db      0,0,0a1h

msg3:
        db      04h,86h,0a1h    ;csr form

msg4:
        db      01h,0a1h        ;overlay param
msg5:
        db      16h,0a1h        ;disp on/off

msg6:
        db      0b0h,04h,0a1h   ;set cursor to
                                ;graphics page
msg7:
        db      31h,2h,0a1h     ;set cursor
                                ;text page
                                ;1st line

msg8:
        db      59h,2,0a1h      ;2nd line

msg9:
        db      81h,2,0a1h      ;3rd line

msg10:
        db      0a9h,2,0a1h     ;4th line

; 1200 spaces for text page clear
; The following table is not listed
; here, except for the first 8 bytes,
; but consists of 1200 bytes
; all of which are 20h

msg11:
        db      '        '
        db      01ah

msg18:  db      0,01ah          ;hscr param

; 320x240 bonsai tree graphic
; The following table is not listed
; here. It consists of 9600 bytes
; which constitute a full screen
; bit map image of a bonsai tree.
; You may add a few bytes before the
; 01ah termination byte for testing
; puposes or include a complete
; bitmap image

msg12:
        db      01ah

msg13:
        db      0,0,01ah        ;set cursor
                                ;to text page

msg14:
        db      'HANTRONIX'
        db      0a1h

msg15:
        db      'Crystal Clear and'
        db      0a1h

msg16:
        db      'Visibly Superior'
        db      0a1h

msg17:
        db      'LCD Modules'
        db      0a1h

        end
```