Interfacing to a Graphics Module with a Toshiba T6963C Controller

Introduction:
The Toshiba T6963C LCD controller is one of the most popular controllers for use in small graphics displays. For this reason it is used in a number of Hantronix LCD modules from 128x128 to 240x128 pixels. This class of module is most commonly used to display a mixture of text and graphics in small devices and instruments. It is usually interfaced directly with an embedded 8-bit microprocessor. This application note describes the use of this controller with a number of Hantronix graphics modules. The example used here is based on an 8-bit Intel I8051 micro-controller but applies to almost any micro-controller.

Functional Description:
The T6963C uses a simple 8-bit data bus to transfer all instructions and data to and from the display. After power is applied a hardware reset is issued. The controller must then be initialized with a series of commands to set up the various modes and options available to the user. Once initialized the display is ready to accept the data to be displayed. This data can be in the form of bit-mapped graphical data or text data in pseudo ASCII format. The internal character generator is biased at -20h. which means the user must subtract 20h from the ASCII code before sending it to the display.

In the bit-mapped mode each byte of data represents 8 pixels on the display. See Figure #1. Text data can be written in pseudo ASCII format and will be displayed as a 5x7 dot character. The graphical and text data are written to separate, user assigned, areas, or pages, in the display RAM within the display module. The user can then choose one of three ways to mix the text and graphics pages on the display or the text or graphics page can be displayed alone.

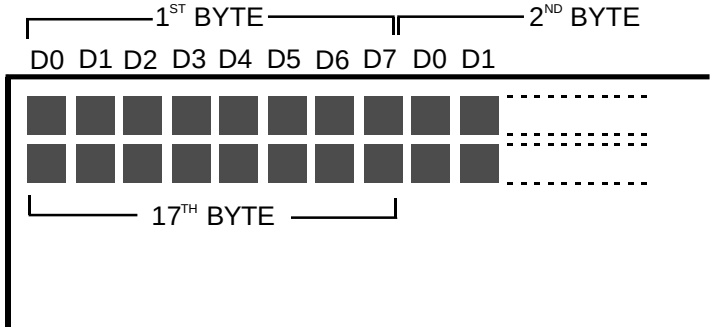


Figure #1 Memory map of a 128x128 display

Schematic:
The 80C51 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 2.

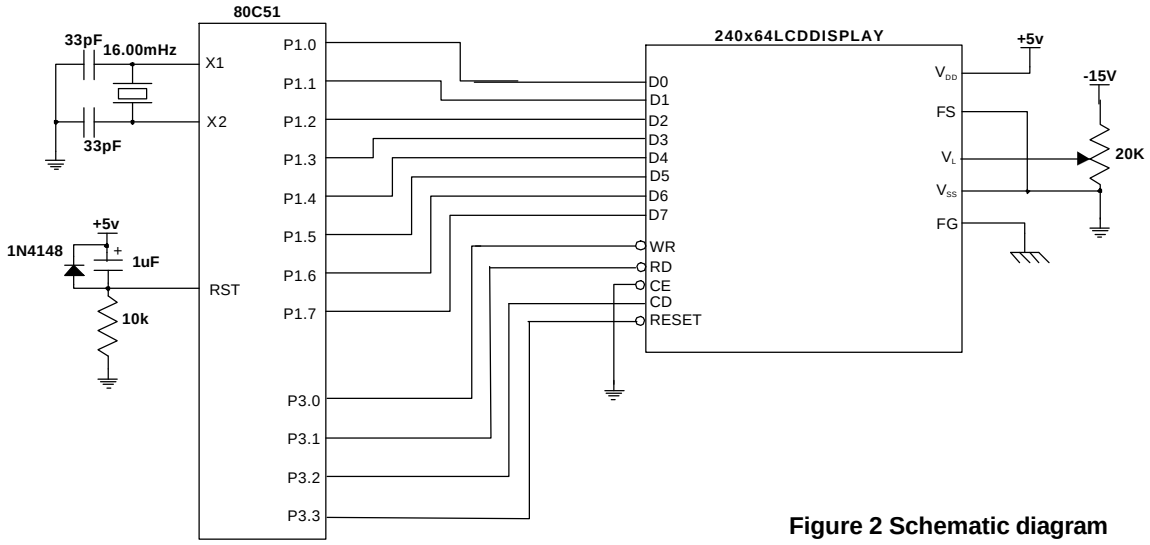


Figure 2 Schematic diagram

Software:
The sample program here is written in 8051 assembly language and is designed to work with the hardware shown in Figure 2. It is based on a 240x64 display but is applicable to any Hantronix graphics display with the T6963C controller.

The program first resets the display. This reset must last a minimum of 2µS. It then sends a series of command bytes and parameters to the LCD controller to initialize it. The T6963C requires that the parameters, if any, are sent in the data mode and must precede the command byte, which is sent in the command mode.

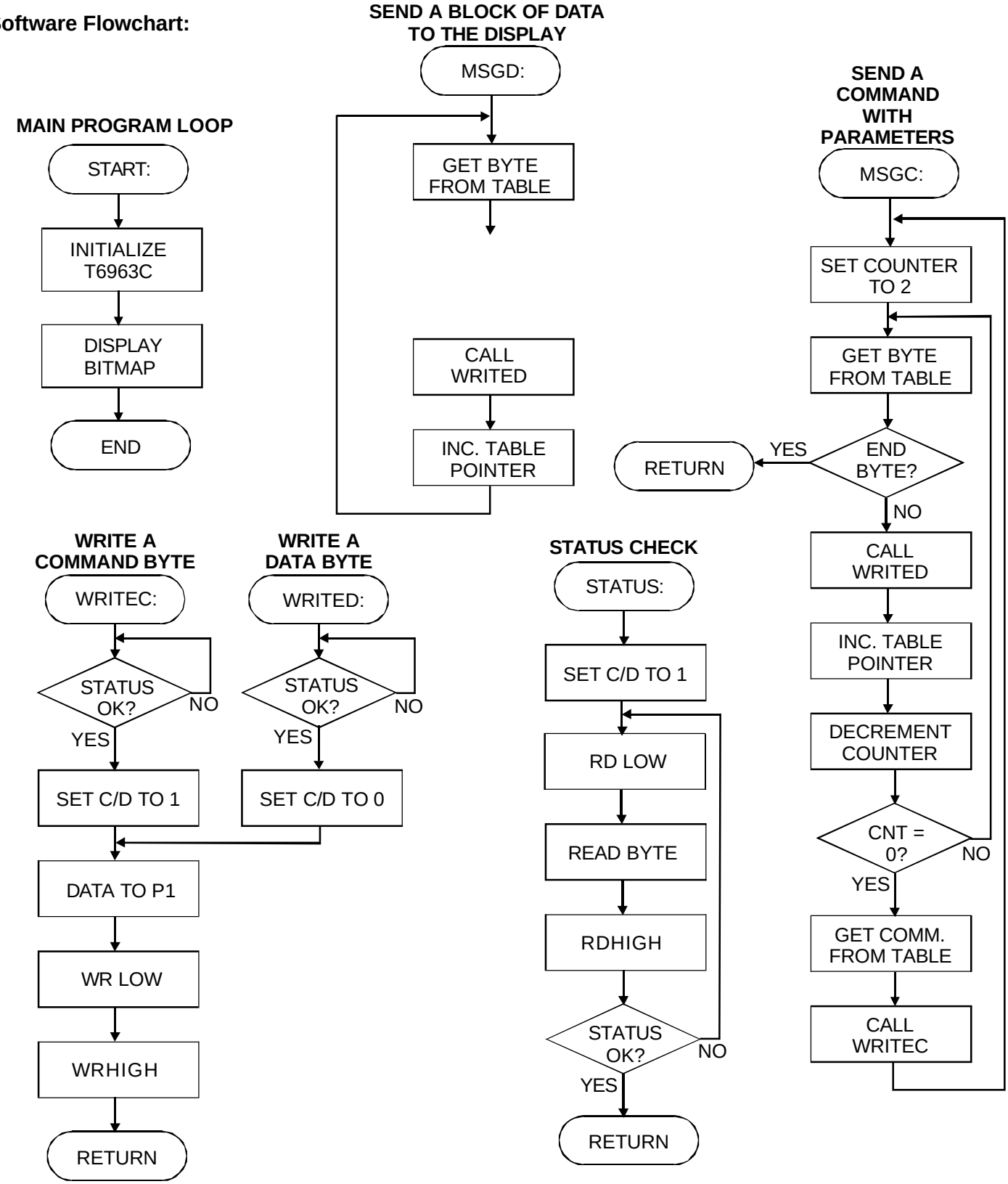
The controller is initialized with the graphics page at 0000-077fh. This is 1920 bytes which will accommodate a full screen of data. No text is displayed in this example but the controller is initialized for text operation to illustrate the process. The text page starts immediately following the graphics page at location 0780h. All Hantronix graphics displays that have a built in T6963C controller also have 8k of built-in display RAM.

The FS (Font Select) line is set to a zero in this example which will place the 5x7 pixel character in a 8x8 pixel field. This will produce 30 characters per line with 8 lines per display ($\frac{240}{8}=30$). The total number of characters possible in this mode is 240. A one on the FS line will produce an 6x8 pixel field for each character.

After initialization the controller is set in the auto write mode and the display is filled with a bit-mapped graphic.

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

Software Flowchart:



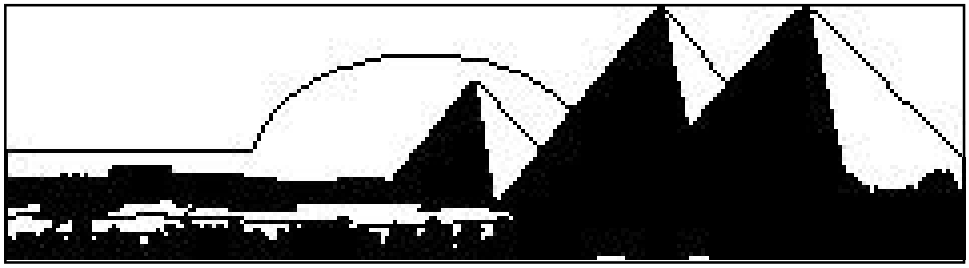
Initialization:
Before the LCD controller can accept or display data or text it must be initialized. This is usually done immediately after the system is powered up. The following chart lists the initialization commands and the parameters that accompany them along with a brief explanation of the function of each. Where a parameter is different for a display that differs in resolution from the 240x64 example, the alternate values are also listed.

Initialization bytes:

COMMAND	CODE	PARAMETERS*	FUNCTION
TEXT HOME ADDRESS	40h	80h, 07h	START OF THE TEXT STORAGE AREA IN DISPLAY RAM 0780h
TEXT AREA	41h	1eh, 00h (240x128, 240x64)	LINE LENGTH IN CHARACTERS $\frac{240}{8}$ = 30 = 1eh FOR AN 8 BIT CHARACTER CELL (FS=0). (28h FOR FS=1)
		14h, 00h (160x128)	LINE LENGTH IN CHARACTERS $\frac{160}{8}$ = 20 = 14h FOR AN 8 BIT CHARACTER CELL (FS=0). (1ah FOR FS=1)
		10h, 00h (128x128)	LINE LENGTH IN CHARACTERS $\frac{128}{8}$ = 16 = 10h FOR AN 8 BIT CHARACTER CELL (FS=0). (15h FOR FS=1)
GRAPHIC HOME ADDRESS	42h	00h, 00h	START OF THE GRAPHICS STORAGE AREA IN DISPLAY RAM 0000h
GRAPHIC AREA	43h	1eh, 00h (240x128, 240x64)	LINE LENGTH IN PIXELS $\frac{240}{8}$ = 30 = 1eh
		14h, 00h (160x128)	LINE LENGTH IN PIXELS $\frac{160}{8}$ = 20 = 14h
		10h, 00h (128x128)	LINE LENGTH IN PIXELS $\frac{128}{8}$ = 16 = 10h
MODE SET	81h	———	"EXOR" TEXT MODE
ADDRESS POINTER SET	24h	00h, 00h	LOCATION OF NEXT DATA WRITE (OR READ) IN THIS CASE THE BEGINNING OF THE GRAPHICS AREA
DISPLAY MODE SET	98h	———	TEXT OFF, GRAPHICS ON

*** ADDRESSES AREFORMATTEDAS [LSB,MSB]**

Displayed image:



Software:

```
$mod51
; ************************************************
; *                                              *
; *          T6963 Application Note V1.0         *
; *                                              *
; ************************************************

; The processor clock speed is 16MHz.
; Cycle time is .750mS.
; Demo software to display a bit-mapped
; graphic on a 240x64 graphics display
; with a T6963C LCD controller.



        org     000h
        ljmp    start           ;program start

        org 100h

start:

; Initialize the T6963C

        clr     p3.3            ;hardware reset
        nop
        nop
        setb    p3.3
        mov     dptr,#msgi1     ;initialization bytes
        lcall   msgc

; Start of regular program

; Display graphic
        mov     dptr,#msgi2     ;set auto mode
        lcall   msgc
        mov     dptr,#msg1      ;display graphic
        lcall   msgd
        sjmp    $

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

; MSGC sends the data pointed to by
; the DPTR to the graphics module
; as a series of commands with
; two parameters each.

msgc:
        mov     r0,#2           ;# of data bytes
msgc2:
        clr     a
        movc    a,@a+dptr       ;get byte
        cjne    a,#0a1h,msgc3   ;done?
        ret
msgc3:  mov     r1,a
        lcall   writed          ;send it
        inc     dptr
        djnz    r0,msgc2
        clr     a
        movc    a,@a+dptr       ;get command
        mov     r1,a
        lcall   writec          ;send command
        sjmp    msgc            ;next command

; MSGD sends the data pointed to by
; the DPTR to the graphics module.

msgd:
        clr     a
        movc    a,@a+dptr       ;get byte
        cjne    a,#0a1h,msgd1   ;done?
        ret
msgd1:
        mov     r1,a
        lcall   writed          ;send data
        inc     dptr
        sjmp    msgd

; WRITEC sends the byte in R1 to a
; graphics module as a command.

writec:
        lcall   status          ;display ready?
        setb    p3.2            ;c/d = 1
writec1:
        mov     p1,r1           ;get data
        clr     p3.0            ;strobe it
        setb    p3.0
        ret

; WRITED sends the byte in R1 to the
; graphics module as data.

writed:
        lcall   status          ;display ready?
        clr     p3.0            ;c/d = 0
        sjmp    writec1

; STATUS check to see that the graphic
; display is ready. It won't return
; until it is.

status:
        setb    p3.2            ;c/d=1
        mov     p1,#0ffh        ;P1 to input
        mov     r3,#0bh         ;status bits mask
stat1:
        clr     p3.1            ;read it
        mov     a,p1
        setb    p3.1
        anl     a,r3            ;status OK?
        clr     c
        subb    a,r3
        jnz     stat1
        ret
```

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

; Initialization bytes for 240x64
msgi1:
        db      80h,07h,40h     ;text home address
        db      1eh,00,41h      ;text area
        db      00,00,42h       ;graphic home address
        db      1eh,00,43h      ;graphic area
        db      00,00,81h       ;mode set
        db      00,00,24h       ;address pointer set
        db      00,00,98h       ;display mode set
        db      0a1h


msgi2:
        db      00,00,0b0h      ;auto mode
        db      0a1h

;240x64 Bitmap graphic data
;Only the first 8 bytes are shown here
;The real graphic consists of 1920 bytes
;of binary data.

msg1:
        db      00h,00h,00h,00h,00h,00h,00h,00h
        db      0a1h

        end
```