Writing Software for T6963C based Graphic LCDs

(Adapted from the Toshiba 6963C Data Sheet)

1. Introduction

So you have a graphics LCD module with a T6963C controller, connected to a Microcontroller, and want to get software written to run it. The document assumes that you have a basic idea of what the LCD module will do and what you want it to do. Reading the .PDF version of the Toshiba Data Sheet on the T6963C is recommended, if this document is unclear. (don't read the original data sheet, as it is very poor)

2. Basics of Writing to, and Reading from the T6963C

For all of the following, so long as your microcontroller takes more than 200ns to do an instruction, you arn't going to violate the specs. (For microcontroller faster than this, see page 152 of the T6963C data sheet, 'Bus timing characteristics.')

2.1 **Write data** - The data to be written should be set on D0 - 7 and
C/\D taken low,
/WR taken low (/RD should be high)
/CE pulsed low for greater than 80ns

2.2 **Read data** - C/\D take low,
/RD take low (/WR should be high)
/CE take low
After 150ns read the data on D0-7
/CE take high

2.3 **Write Command** - To Write a Command - The command should be set on D0 - 7 and
C/\D taken high,
/WR taken low (/RD should be high)
/CE pulsed low for greater than 80ns

2.4 **Read Status** - To check the Status of the T6963C controller -
C/\D take high,
/RD take low (/WR should be high)
/CE take low
After 150ns read the data on D0-7
/CE take high
Note that there is no such thing as reading an command instruction.

3. What does status checking means.

3.1 Before every data read, data write, or command write, the status must be checked. For the majority of commands, this only requires checking that STA0 and STA1 are set to '1' at the same time. Where other bits of the status need to be checked, it will be noted against the command in later sections.

As no timings are given for command execution, delaying each command and data until the last is known to have finished so that the status does not need to be checked, (as used frequently on character LCD modules) is more difficult. The Toshiba data sheet states it is not possible, but I have seen code that uses this method with the T6963C.

When the check status byte is read, the bit meanings are as follows:-

MSB

D7	D6	D5	D4	D3	D2	D1	D0
STA7	STA6	STA5	X	STA3	STA2	STA1	STA0

STA0 - 0 = command being executed, 1 = ready for new data and command.
STA1 - 0 = internal data read/write occurring, 1 = ready for new data and command.
STA2 - 0 = internal automode data read occurring, 1 = ready for next data read.
STA3 - 0 = internal automode data write occurring, 1 = ready for next data write.
X - not used.
STA5 - 1 = controller capable of operation. [1]
STA6 - 1 = screen peek/copy error (address not in graphic RAM), 0 = no error.
STA7 - 0 = blinking areas of display off, 1 = blinking areas of display on. [2]

Notes; [1] - I don't know what use this bit has. It is ignored in all software I've seen.
[2] - I believe this is the meaning of this bit.

4. Writing commands with one, two or no data byte.

4.1 Where a command requires no data, do the following:-

Check Status
Write Command

4.2 Where a command requires one data byte, do the following:-

Check Status
Write Data
Check Status
Write Command

4.3 Where a command requires two data byte, do the following:-

Check Status
Write First Data byte
Check Status
Write Second Data byte
Check Status
Write Command

4.4 In the case of sending more data bytes than is required, only the last 1 or 2 are used. This is useful if you have stored all the required instructions to set up the display as a lookup table, each instruction having 2 data bytes in the table (even if it does not need 2 data bytes), and use the write with two data bytes routine. Commands that don't need the data will ignore it. Commands only need one byte will use the last byte, and ignore the first.

5. Initialising the display.

Full descriptions of all the commands are given in Appendix A. Only brief description as required, are given here.

5.1 To set-up the **graphics** part of the display controller, if you are going to use graphics, or are going to use TEXT ATTRIBUTE mode (which uses the graphics area to store the attribute data):-

5.1.1 Set 'GRAPHICS HOME ADDRESS' to the address that you want to start the graphics RAM from. If you want the graphics area of memory before the text area, or are just using the graphics area, set it to the start of memory, otherwise set it to an area out the way of the text RAM area.

		e.g. for start at 0000h	e.g. for start at 0200h
status check			
1st data	Low addr byte	00h	00h
status check			
2nd data	High addr byte	00h	02h
status check			
Command		42h	42h

5.1.2 Set 'GRAPHICS AREA SET' to how many bytes later you want the next line to start at. Note that this can be the same as the number of bytes to fill one line of the screen, for most efficient use of the RAM, or it can be larger, to make the calculation of the addresses easier to implement in your routines. (If you make less than the number of bytes to fill one line, you get some weird results!)

For example for a 240 bit (30 bytes if font set to 8x8) you could use 30, or you could use 32, which is easier to use in calculations.

		for 30 (1Eh)	for 32 (20h)
status check			
1st data	N^{o} of bytes later	1Eh	20h
status check			
2nd data	Always 00h	00h	02h
status check			
Command		43h	43h

5.2 To set-up the **text** part of the display controller, if you are going to use text:-

5.2.1 Set 'TEXT HOME ADDRESS' to the address that you want to start the text RAM from. If you want the text area of memory before the graphics area, or are just using the

text area, set it to the start of memory, otherwise set it to an area out the way of the graphics RAM area.

		e.g. for start at 0000h	at 0800h
status check			
1st data	Low addr byte	00h	00h
status check			
2nd data	High addr byte	00h	08h
status check			
Command		40h	40h

5.2.2 Set 'TEXT AREA SET' to how many bytes later you want the next line to start at. Note that this can be the same as the number of columns on the screen, for most efficient use of the RAM, or it can be larger, to make the calculation of the addresses easier to implement in your routines. (If you make less than the number of bytes to fill one line, you get some weird results!)

For example for a 240 bit (30 columns if font set to 8x8) you could use 30, or you could use 32, which is easier to use in calculations.

		for 30 (1Eh)	for 32 (20h)
status check			
1st data	N^{o} of bytes later	1Eh	20h
status check			
2nd data	Always 00h	00h	00h
status check			
Command		41h	41h

5.3 To set-up the display:-

5.3.1 Set 'MODE SET' to logical OR, EXOR or AND of the text and graphics as required. To use only the text, but with extra display options, there is also TEXT ATTRIBUTE mode (note in this mode, there is no graphics display - the attribute data is in the graphics area instead.) MODE SET also selects whether text characters 00h-7F come from the controller's built in character generator ROM, or from RAM i.e. user-defined. (characters 80h-FFh always come from the RAM).

MSB

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	CG	MD2	MD1	MD0

MD0-2 [1]
000 = OR mode.
001 = XOR mode.
010 = AND mode.
100 = TEXT ATTRIBUTE mode.
CG 0 = Internal ROM CG, 1 = RAM CG

Notes; [1] - I don't know what setting MD0-2 to 011, 101, 110 or 111 would do! - I'll have to try it sometime.

For example, for OR mode, and internal CG ROM = 1000 0 000 = 80h

status check		
Command		80h

5.3.2 Set 'DISPLAY MODE' to set whether the graphic, text or both displays are on, and to set the cursor on or off, and cursor blink on or off

MSB

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	GRPH	TEXT	CUR	BLK

GRPH 1 = graphics display on. [1]
TEXT 1 = text display on.
CUR 1 = cursor displayed.
BLK 1 = cursor blink on.

Notes; [1] - If using TEXT ATTRIBUTE mode, this bit must be set as well as TEXT

For example, graphics and text display on, with cursor, no blink = 1001 1110 = 9Eh

status check		
Command		9Eh

5.4 If you've followed the above steps, now you should have a display full of rubbish. If not, check the contrast level, the wiring and the software.

5.5 Display data write and read commands

5.5.1 The ADDRESS POINTER is used for pointing in memory, before writing / read ing text byte, graphic bytes, attribute bytes or user-character bytes (depends where it is pointing, and whether graphics or attribute mode is on).
To set the ADDRESS POINTER to a memory address, ready for reading or writing bytes, e.g. to set it to address 0123h

status check		
1st data	Low addr byte	23h
status check		
2nd data	High addr byte	01h
status check		
Command		24h

5.5.2 To then send a byte of data, there are 3 command, depending whether you want the pointer address to decrease, stay the same or increase.

Write then decrease address pointer command is C2h
Write, no change to address pointer command is C4h
Write then increase address pointer command is C0h

For example if the address pointer is set to an address in the text RAM area, to display an ‘A’ (character 21h - note the character set is not ASCII, but ASCII - 20hex) and move the address pointer to the next address:-

status check		
Data	byte to send	21h
status check		
Command		C0h

5.5.3 Similarly, to read a byte of data from the display RAM, again there are 3 command, depending whether you want the pointer address to decrease, stay the same or increase.

Read then decrease address pointer command is C3h
Read, no change to address pointer command is C5h
Read then increase address pointer command is C1h

For example if the address pointer is set to an address in the text RAM area, to read the code of the character at that address and move the address pointer to the next address:-

status check	
Command	C1h
status check	
Read data	

5.5.4 To save time, e.g. for writing or reading the full screen, DATA AUTO mode is available, where the command is written only one, to set ‘auto mode data read’ or ‘auto mode data write’. After this, no commands are required for each data byte read or written. When finished, ‘auto mode reset’ needs to be sent to return the controller to normal operation. Note that during auto mode the status check between writing data bytes is of STA2, and between reading bytes is of STA3. STA0 & 1 are not valid until after the auto mode reset .

Data Auto Write Set is B0h
Data Auto Read Set is B1h
Auto Mode Reset is B2h

For example

Reading data:-		Writing data:-	
status check		status check	
Command	B1h	Command	B0h
status check of STA3		**status check of STA2**	
Read data ‘1’		Write data ‘1’	XXh
. .		. .	
. .		. .	
status check of STA3		**status check of STA2**	
Read data ‘n’		Write data ‘n’	XXh
status check of STA3		**status check of STA2**	
Command	B2h	Command	B2h

5.5.4 To set or reset an individual bit, use the ‘BIT SET/RESET’ command. Only one bit can be set at a time.

MSB

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	S/R	B2	B1	B0

S/R 1 = Set the bit, 0 = Reset the bit

B2-B0 000 - Bit 0 (Right hand bit of byte)
001 - Bit 1
010 - Bit 2
011 - Bit 3
100 - Bit 4
101 - Bit 5
110 - Bit 6
111 - Bit 7 (Left hand bit of byte)

For example, to set bit 3

status check
Command FBh

5.6 Cursor Commands

5.6.1 Cursor Pattern Select - To select the number of line high the cursor is, from the bottom of the character, use:-

1 Line Cursor is A0h
2 Line Cursor is A1h
3 Line Cursor is A2h
4 Line Cursor is A3h
5 Line Cursor is A4h
6 Line Cursor is A5h
7 Line Cursor is A6h
8 Line Cursor is A7h

For example, for a 4 line cursor,

status check
Command A4h

5.6.2 To set the position of the cursor onscreen, use ‘CURSOR POINTER SET’

For example for the cursor at 24h across, 03h down

status check
1st data Xadrs 24h
status check
2nd data Yadrs 03h
status check
Command 21h

5.7 Character Generator Commands

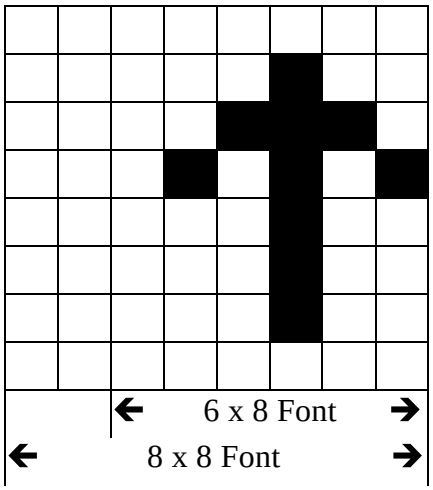
5.7.1 To set the address at which the character generator RAM starts from, the ‘OFFSET REGISTER SET’ command is used. If 8K (2000h) of RAM is fitted to the module, only the first four offset data numbers are usable.

Data	CG RAM hex address range
00000	0000 - 07FFh
00001	0800 - 0FFFh
00010	1000 - 17FFh
00011	1800 - 1FFFh
:	: :
11110	F000 - F7FFh
11111	F800 - FFFFh

For example to set the CG RAM area to 1800 - 1FFFh, so that character 80h takes up locations 1C00h to 1C07h.

status check		
1st data	Offset Data	03h
status check		
2nd data	Always 00h	00h
status check		
Command		22h

5.7.2 Then to set character number 80h to a user defined character, say ‘↑‘, as drawn below



= 0000 0000b = 00h
= 0000 0100b = 04h
= 0000 1110b = 0Eh
= 0001 0101b = 15h
= 0000 0100b = 04h
= 0000 0100b = 04h
= 0000 0100b = 04h
= 0000 0000b = 00h

Assuming the CG RAM area is set to 1800 - 1FFFh, set the address pointer to location 1C00h (1800h + 80h x 8) as per section 5.5.1, and write the above 8 bytes using “write then increase” (section 5.5.2) or “data auto write” (section 5.5.4). Then setting the address pointer to an address in the text area, and writing ‘80h’ will display the ↑ character.