Getting Started with the DMX-973B & SP 5-GFX1

The purpose of this document is to help you get going with Lascar Electronics graphics displays. I have listed useful resources that I have found, before you start print off the command section from the controller IC datasheets which are on the Lascar web site.

The 973B and GFX1 can be controlled from a micro. The display controller IC handles the LCD, including refresh, but it requires data to do anything useful. Data is written as either serial or parallel to the 973B and serial to the GFX1. SERIAL DATA DOES NOT MEAN RS232.

The software examples have mainly been written on the Lascar MDM-1, which is a PIC 16F877 running at 4MHz, there is also some PIC 16C54 & 8051 code. C has generally been used to make reading easy, there is a small code extract written in PIC assembler. The software has been written to make it easily portable to different micro's and languages. Note, on the 973B the BUSY signal is ignored because with code, running at 4Mhz, I do not see the signal

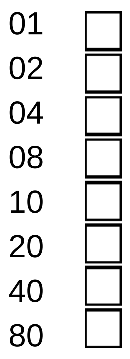
The 973B has a resolution of 100 x 32 pixels and uses the Epson SED1530 (also known as the S1D15300) controller. The GFX1 has a resolution of 128 x 64 pixels and uses a Sunplus SPLC 501C controller. If you look at both datasheets you will see a few additions in the Sunplus but clearly these controllers are based on the Epson standard. Therefore, once you have code for one it is not difficult to produce software for the other. Please read the addendum for the GFX1 that was sent in by Lars Thörnqvist.

Contrast is set by software and is referred to as 'Electronic Volume'. The 973B has 32 levels while the GFX1 has 64.

Initialisation routines are given in the driver software downloads and are important in getting the display to work correctly.

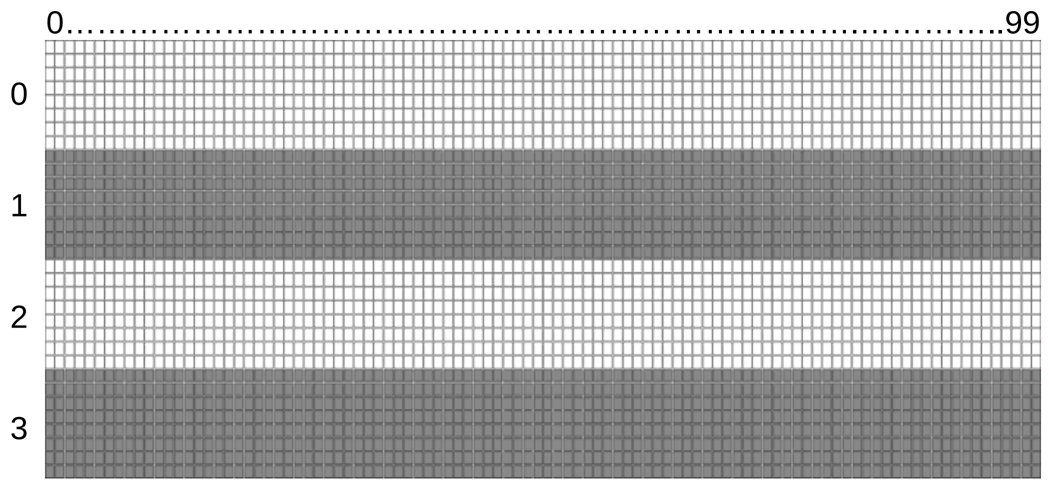
Both graphic displays have columns across the horizontal (x axis) and rows down the vertical (y axis). Imagine horizontal strips running across the display, these are know as Pages. Each Page is 8 bits wide (a byte). Each bit is a pixel, which can either be ON (1) or OFF (0). The zero coordinates (0, 0) are in the top left hand corner.

Bits/Pixels are arranged vertically in a Page as a byte. The byte goes from LSB to MSB as shown below:



DMX 973B LAYOUT

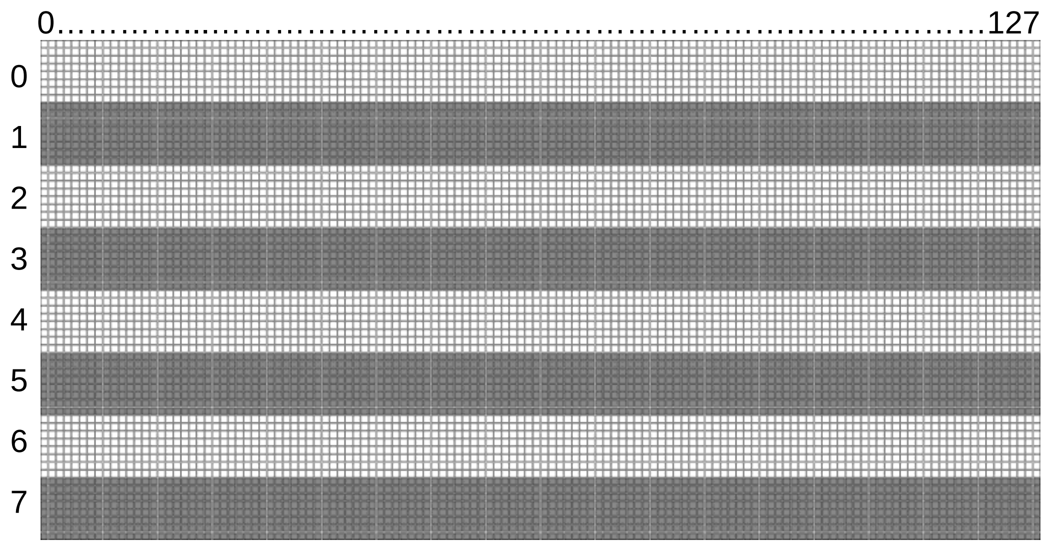
The 973B displays 4 pages and looks like this:



Pages 0- 3 are displayed but 4 – 7 are also available as memory but are not displayed. Also note that the x-axis is offset by 16, this is because internally column 0 of the LCD is connected to column 16 of the controller chip – this made the PCB layout easier.

SP 5-GFX1 LAYOUT

The GFX1 displays 8 pages and looks like this:



In display terms there are two ways of writing to the display. Firstly if you have a picture you will just write to the display, this overwrites anything that is already there. Secondly if you already have something on the display and then you want to display something else in addition then you need to ensure that the original data is not lost. i.e. if the pixel at (x,y) (0, 0) was on then 0x01 would be written to the first byte of Page 0. Now if you want the pixel at (0, 5) to be on it would be no good just writing 0x20 to the same byte as the original data would be lost. Therefore, you have to read the byte, modify the required bit, and then write the data back. Now here is a problem, SERIAL MODE DOES NOT ALLOW READ. Therefore, you have a choice, either, only do byte writes i.e. pictures and text or create a shadow RAM (2 dimensional array). Note that this would have to be 400 bytes for the 973B and 1K for the GFX1. Many micros do not have this amount of free RAM. Of course you may only require part of the display to work this way and so your memory requirements could be reduced.

DISPLAYING FONTS

When I refer to a Page remember that there are 4 displayed on a 973B and each one is 5.25 mm high. On a GFX1 there are 8 displayed and each one is 2 mm high. This means that characters are about 2.5 times smaller on a GFX1 than a 973B. Within the software I have referred to pages as lines i.e. 1L, 2L and 4L.

I have supplied the 5x7 (fixed width and proportional), which is one page wide. A 10x16 fixed width font, which fits into the width of two pages. I also created a 20 x 32 fixed width font for 0 – 9 plus V, which fits into a width of four pages. Another thing to consider with fonts is the size of your data lookup file.

5 x 7	= 7 bytes per character 672 bytes for the full ASCII set
10 x 16	= 20 bytes per character 1920 bytes for the full ASCII set
20 x 32	= 80 bytes per character 7680 bytes for the full ASCII set

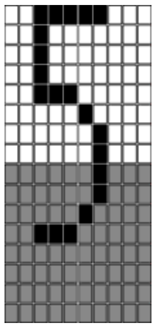
(ASCII set defined as character 32 to 127, but can be greater if you include letters up to 255.)

The way I construct characters was based on the way that other programs create the data. In reality you are free to write them as you wish.

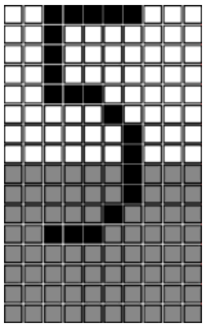
My example represents a 10 x 16 fixed width character. The bytes of data are written in the following order:

	Columns									
Page n	01	03	05	07	09	11	13	15	17	19
Page n+1	02	04	06	08	10	12	14	16	18	20

Plotting in the number 5 you get the following



973B



GFX1

Please note: These images are not to the same scale, but the aspect ratio of the pixels for both the 973B and GFX1 is correct.

This translates into the following data:

```
0x00  0x00  0x00  0x00  0x1F
0x08  0x11  0x08  0x11  0x08
0x21  0x04  0xC1  0x03  0x00
0x00  0x00  0x00  0x00  0x00
```

To save code space (16F877 has 8K of flash) I hold some data in a serial EEPROM, character set can be downloaded as hex data suitable for programming. See the simple EEPROM programmer which is based on the JDM PIC programmer and driven by ICProg PC software. Power is derived from the RS232 port.

The 5x7 font was created using the Charset Editor, written by Javier Martinez, this takes a long time, as each character has to be individually created. The other fonts were created using png2charset written by Nikolai Golovchenko. I have since discovered FontGen written by Bahri Okuroglu, this excellent program needs the following settings (Rotation = 90º & Flip X = Enabled) to allow it to work with the 973B & GFX1. This program accepts a variety of fonts although it is recommended that bitmapped fonts specifically designed for small displays and not full colour, high resolution, displays are used. I have used the associated c code with a different display and found it to be excellent. A selection of bitmap fonts can be found at the dafont web site.

Another program called Fntcvtr is available this converts Fonts or Bitmaps to data but I have not tried it.

If storage space is restricted then further data compression should be considered. First off, variable width fonts waste less space but require more processing. If the character height is 1.5 pages then data could be fitted into 1.5 bytes (3 nibbles) certainly more processing but the data would be reduced by 25%. To save time and memory only create the characters you want to use.

DISPLAYING PICTURES

You can scan, draw or download images. However always remember the resolution and size of your display. Use a picture editor to get the picture to the correct size (100x32 or 128x64), I use Adobe PhotoDeluxe, (bundled with my HP scanner a few years ago) but something like Photoshop or CorelDRAW would do the job. Save it as a black & white bitmap.

If you want to create your own image, or edit an existing image, you can use MS Paint. Set the Attributes to 100 x 32, or 128 x 64, and Black & White. The image is very small therefore select View Zoom Custom 800%.

To produce software I use Bmp2Asm, by John Gerthoffer. This program takes a black & white bitmap and produces data output, each line represents a page. This can be cut and pasted into your software application. To convert the data into a C array then you can run this conversion program from Ron Wintjens.

To help you get going I have posted two bitmaps, a helicopter for the 973B (scanned, rotated by 180° and then touched up in Paint, because the rotor blades broke up) and the world map for the GFX1.

Another excellent program is FastLCD, by I. Bojan. The output is designed for Basic but the data can easily be converted using the Search and Replace function of any good text editor. FastLCD has a graphics editor that allows free draw, lines, fill, text (including rotation), outline shapes, filled shapes, adjustable pen width, symbols and arrows. External bitmaps can also be loaded. The software says version 1.2.0 whereas the website says v1.4.1, also there is no help file. I have not used the program but the data format appears to be correct, select SED output format.

DISPLAYING GRAPHICS

This section applies to the 973B in parallel mode, or, if Shadow Memory is used, to the 973B or GFX1 in serial mode.
Graphics is about drawing lines, curves and other shapes using co-ordinates i.e. draw a circle of 20 pixels diameter, centered at the coordinates 49, 15. Jack Bresenham, established the general principles for use on plotters, back in the early 60's and his algorithms are often referred to. Also look for Wu.

Examples of straight lines and circles have been included in the downloads. A search of the internet will provide many code examples. You should not need to use anything other than lines and circles, I once tried to produce an ellipse using software that I found, it took 20 seconds to appear! Clearly you have to match your requirements to your processing power.

MISCELLANEOUS

The main advantage of using serial data is that it only requires 4 lines (plus 2 for the supply), whereas the parallel uses 12 lines (plus 2 for the supply). The disadvantage of serial is that you can write to LCD memory but not read from it. With the GFX1 it would be possible to multiplex several displays by using the chip select.

The pixels of the 973B are not square, the aspect ratio is approximately 0.65 : 1. Visually this does not look out of place with characters but will need correcting for when using graphics. The GFX1 pixels are square.

I have been asked if an SPI bus can be used on a PIC, Atmel etc with the GFX-1, until recently all I could say was that it looked possible and one customer said he was using it. I have now implemented it on a 8051. The SPI hardware interface is 25 times faster than my Bit Bang code.

I came across this small piece of code, written in Proton Plus Basic, for the GFX1. Go down to the forum entry dated 31st January 2004 from Michael Turner and view newdisplay.bas.

Should you find any errors, omissions or think that something is missing then please contact me.