MikroPascal ST7565 package:

Intro

The Sitronix ST7565 family of LCD controllers are available in a lot of commercially available LCD modules, including the DOGM, Displaytech, and Newhaven modules. The controller integrates a lot of functions which were performed with external analog circuitry on older units, making the initial setup more complicated, since a lot of internal voltage regulation has to be set up for each manufacturer and model of controller. This library attempts to provide a basic set of primitive procedures which can adapt to any model of ST7565 based LCD module.

Goals

This project provides for basic driving of an LCD module through the parallel interface, and the 8080 variant. This does not attempt to support either SPI or 6800-parallel interface. Adaptations to these interfaces are welcome. This library provides basic cursor positioning and byte writing, and also complete frame buffer writing. No attempt will be made to furnish basic graphic functions such as line, circle, box, etc.

In the future, I intend to add text routines.

This was developed and tested in mikroPascal for PIC32, but should be easily portable to PIC24 and 18, and with some adjustment probably PIC16 and AVR. Tested variants for these platforms are welcome.

Background

The PDF for the Sitronix controller chip can be found here:
http://www.newhavendisplay.com/app_notes/ST7565.pdf

There is also useful information from DOGM:

http://www.lcd-module.com/eng/pdf/grafik/dogm128e.pdf

Hardware

The 8080 interface requires the C86 line to be connected to ground. The other five control lines must be interfaced to the microcontroller. Those lines are named:

6800 Nomenclature	8080 Nomenclature	Library Variable
A0	A0	STA0

RES	RES	STRS
CS1	CS1	STCS
E	/RD	STRD
R/W	/WR	STWR

Declare the variables in the right column to be 'at' the appropriate pin as in the example.

Also, the data bits (D0-D7) are set up to be in scrambled order, in case the data bits on the LCD don't correspond exactly to a PIC port. If there is a direct correspondence between the LCD data port and a PIC port, the code can be simplified, and will run faster on an 8-bit PIC. There is no noticeable slowing from using scrambled bits on a PIC32.

Procedures

STInit(array[20] of byte);

The general procedure for using a ST7565 based controller is to first initialize it, and then write to the LCD RAM. The initialization in this routine is done in two parts; the basic hardware reset, and then a series of initialization commands. The basic opcodes are all one byte, but some commands are two-byte. The actual sequence will vary from one LCD module to another, and so these codes need to be placed into this array of byte to assure the correct initialization. A number of opcodes are predefined as constants.

The intent of this procedure is to allow all the particulars for a given model of display module to be embedded into the array. Then the procedure simply needs to be called with the correct array as its argument in order to set all the correct settings of the particular model. Note that some setting may have to be dynamically adjusted, so a subsequent call to STWriteCode will be needed after the call to STInit.

This library was tested on the Displaytech xxx, and the array is included. It is hoped that as users will submit tested initialization arrays after successfully testing.

The string size of 20 was chosen as a high number for initial settings. Normal strings typically run more like 12 or 14. The remaining elements must be padded with NOP instructions. Upon startup, these opcodes are all executed in order. Order is important for 2-byte instructions!

Unfortunately, there is no substitute for understanding the ST7565 instruction set and the specifications for the individual model of display module in order to create the initialization string. In addition to getting the voltage generator and contrast voltage settings right, the initialization string can set the display for 6 or 12 o'clock position, and other such settings that you might want to adjust. If, on the other hand, a furnished initialization string works with your module, you can simply use it, and move on to drawing graphics. The only tested string for this package is for the Displaytech xxx, though the one for DOGM should work, as it was copied directly from their literature.

STWriteCode(opcode : byte);

This primitive procedure writes one control byte.

STWriteByte(outport : byte);

This primitive procedure writes one data byte to the display. The column counter is automatically incrimented.

STWriteImage(image : ^const byte);

This paints the entire 128x64 screen with an image in constant (flash) memory. See example for usage.

STSetPage(arg : byte);

Sets the page to 0-7. Pages on the ST7565 work identically to the KS107 (existing GLCD library)

STSetCol(arg : byte);

Sets the column to 0-127. Columns on the ST7565 work identically to the KS107 (existing GLCD library)

STWriteBuffer(buffer : ^byte);

This paints the entire 128x64 screen with an image in RAM (framebuffer).

Acknowledgment:

This code sample taken from the CCS forum was invaluable in creating the basic I/O routines:

http://www.ccsinfo.com/forum/viewtopic.php?p=154055#154055

Janni provided a lot of help on the forum, including pointers on pointers.