Graphics LCD Test System Description

Overview

The graphics LCD test system project served to evaluate 128 x 64 pixel graphics LCDs and develop a basic user interface for subsequent use in a turbocharger control product. Six different graphics LCDs were evaluated and as part of this project firmware was developed for an Atmel ATMega644P microcontroller along with hardware interfaces. Not much information or code examples are available for interfacing graphics LCDs to Atmel AVR microcontrollers, so we hope that publishing the project details will prove helpful to others.

The complete project file can be downloaded from:

www.daytona-sensors.com/download/Graphics_LCD_Test_System_Project.zip

Hardware

The LCD test system schematic, available as a PDF file, shows the hardware details. Sheet 1 shows the ATMega644 microcontroller (MCU) and associated circuitry including ISP and JTAG ports. The ATMega644 is powered from +5V. Sheets 2-7 show the interface to the various LCDs and a Bourns encoder for user input. Only one LCD is connected at a given time. All of the LCDs except the Lumex part utilize a SPI interface. The Lumex LCD uses a parallel interface (see additional comments in the next section). Each LCD has an associated LED backlight driver based on a Linear Technology LT3590 buck mode constant current converter and with the exception of the 5V Lumex LCD, a 3V or 3.3V power supply based on a Microchip MCP1801T or National LM3480 voltage regulator. For the LCD displays that use 3V or 3.3V power and SPI interface, a MC74VHCT50A is used as a level shifter for the 5V SPI signals from the Atmel MCU.

The Bourns encoder is similar to other industry standard parts with quadrature output and momentary push switch feature. Note that the Bourns encoder signals are filtered with RC networks to reduce switch contact bounce related noise.

The hardware was assembled on a 3M ACE Board P/N 318 (Digi-Key P/N 922318). Surface mount parts such as the voltage regulators were mounted on Surfboards that are also available from Digi-Key. We used a Bellin Dynamic Systems SO14 P/N 514 prototyping adapter (Mouser P/N 853-B514) for the MC74VHCT50A and a MSOP8 P/N 523 prototyping adapter (Mouser P/N 853-B513) for the LT3590. The LT3590 SC-70 pattern didn't fit perfectly, but we were able to make it work. The various LCDs were mounted on separate boards supported on standoffs. Depending on the LCD, the boards typically consisted of standard .1"x.1" perf boards with plated thru hole pads. Some of the LCDs had flex ribbon cables that required SMT type connectors. The SMT connectors were mounted on a small section of Twin Industries P/N 8100-SMT10 0.5mm SMT adapter board (Digi-Key P/N 438-1032). Other components required by the LCDs such as bypass and power supply capacitors were also mounted on these boards in close proximity to the LCDs. All the boards with LCDs using SPI have a 10 pin header for signal and power connections (pin numbers are given on the schematic). The project file includes pictures of each LCD display.

Graphics LCDs

We tested 128 x 64 pixel graphics LCDs from six manufacturers: Displaytech, Electronic Assembly, Lumex, Newhaven, Optrex and Tianma. All these displays are transflective (for readability in direct sunlight) with white LED backlighting. All except the Lumex part use SPI interface with a total of 5 signal and control lines coming from the MCU. The Lumex part has an older generation 8-bit parallel interface that requires total of 14 data and control lines. This uses up a significant amount of the available port pins on the MCU and is a significant disadvantage in any practical design. The older parallel interface was also surprisingly slow, requiring almost 10 msec to write the entire display versus about 1.7 msec for all the LCDs using SPI. An additional disadvantage of the Lumex part was the lack of a contrast control command. For our evaluation, we used a trimpot for contrast adjustment. For a real world design additional circuitry such as a D/A or digital pot would have been required, chewing up more valuable MCU pins. Based on the poor performance and huge interface requirement, the Lumex part was immediately disqualified. In our opinion any new product design should use a graphics LCD with SPI.

The project file includes datasheets for the various LCDs and associated controller ICs.

Firmware

Firmware for the AVR MCU is written in assembly language. The project file includes the firmware. The main program initializes the MCU and then executes a continuous loop that resets the watchdog timer and waits for a debounce timer flag to be set by the timer2 overflow ISR every 8 msec. When the flag is set, the encoder signals are read and debounced (ENC_DEB) and a user interface state machine (UI_SM) is called. The user interface state machine makes calls to other routines for displaying and editing data and parameters. All graphics and text strings are written to a 1K SRAM graphics buffer corresponding to 128 x 64 pixels and entire SRAM graphics buffer is then written to the LCD.

A separate include file with graphic LCD subroutines is used for each LCD. Each graphics include file (such as Mega_LCD_Optrex.inc for the Optrex LCD) has subroutines for initializing the LCD, adjusting contrast (not applicable to the Lumex part), clearing a SRAM graphics buffer, writing text strings and bitmapped graphics to the SRAM buffer, and writing the SRAM graphics buffer to the LCD. Data is included for 6x8 and 8x16 fonts that support ASCII characters $20 (space) through $7F (□).

The initialization sequence is unique for each LCD and is based on the manufacturer's recommendations and controller IC datasheet. The Electronic Assembly, Newhaven, Optrex, and Tianma datasheets included flowcharts showing the recommended initialization. Displaytech had initialization info on their website. But getting this info for the Lumex LCD was like pulling teeth, with both the datasheet and corporate websites being brain dead.

With the exception of the unique initialization sequences and the Lumex LCD that requires a special routine for writing the SRAM graphics buffer to the LCD, all the remaining subroutines are common. In general, supporting a new graphics LCD with SPI should only require editing the initialization sequence.

The main program also has an RS-232 communications routine that allows reading back a firmware ID, setting data display values and reading back user entered parameter values. RS-232 communications is at 56000 baud, the highest baud rate that the AVR MCU can accurately generate with a 16 mHz oscillator. Refer to additional notes on RS-232 communications.

Additional include files are Mega_math.inc (math routines), Mega644P_EEPROM.inc and Convert.inc. Convert.inc contains binary to BCD and BCD to ASCII conversion subroutines that are useful for display of numeric data. The Bin2SToAscD subroutine is of particular utility as it converts a 16-bit signed binary number to a 7 character right justified ASCII coded decimal with decimal point selection and appropriate leading zeros. This allows display of numbers ranging from -3.2768 to 3.2768 or -32767 to 32768. The signed binary value $FFFE with 3 digits decimal point selection is correctly displayed as -0.002.

User Interface

The user interface has an initial splash screen that displays our company logo. After a two second delay, a data screen displays four lines of data. The user can rotate the encoder clockwise to scroll eight data values (Data0 – Data7). Pushing the encoder switches to the parameter entry screen that allows entry of eight parameters (Param0 – Param7). When a parameter is highlighted, pushing the encoder allows changing the parameter. Pushing the encoder again saves the edited parameter value to EEPROM and then allows scrolling (rotate encoder clockwise) to the next parameter (push to select) or returning to data display (push to select).

Initial data values including decimal point position are set in the main program. An RS-232 command can be used to change the data values. Initial parameter values are read from EEPROM. Parameter values can be read out by an RS-232 command. All data and parameter values are stored as 16-bit signed binary numbers. Data and parameter names (labels) are stored as text strings in the main program.

This very basic user interface was designed to demonstrate the capabilities of the graphics LCDs. It is relatively intuitive. Naive users were told to try rotating and pushing the encoder and most had no difficulty figuring it out.

Software

Two Windows programs were used as part of the project: font generation software from Electronic Assembly and the graphics conversion shareware program bmp2asm.

The FontEditor program from Electronic Assembly (EA) is supplied with sample fonts on a USB flash drive (Mouser P/N 790-EAUSBSTICK-FONT). This program can also be used to import Windows fonts and generate font files. FV (vertical) fonts were exported as C header files. The header information was then removed with a text editor and the raw comma delimited data imported into Excel to be converted into formatted hex data. A sample Excel file Font_Table_6X8.xls is included in the firmware folder. Saving the Excel spreadsheet as a formatted text space delimited .prn file allows the font data to easily be copied and pasted into AVR code. Note that using any of the EA fonts requires purchasing the FontEditor program.

The bmp2asm program can be downloaded at http://www.piclist.com/techref/microchip/bmp2asm.htm

We use bmp2asm to generate graphics data such as the Daytona Sensors corporate logo that appears on the splash screen. Start by using Photoshop or similar graphics editing software to make a bitmap (no grayscale) .bmp graphics file. Note that the number of vertical pixels must be a multiple of eight. Use bmp2asm to convert the .bmp file to .asm format. Select $ as prefix. Then edit the .asm file with WordPad or similar text editor. Remove the header line. Replace all instances of “dt” with “.DB” The edited .asm file will have one .DB ------- data row for each 8 pixel high row of the graphic image and can easily be copied and pasted into AVR code.

RS-232 Communications

We use an in-house developed Visual Basic program, VbHexterm, for testing and debugging RS-232 communications. VbHexTerm allows sending and receiving ASCII or hex data. VBHexTerm is included in the project file. After installing the program, refer to Help for details. Commands that can be used to communicate with the graphics LCD test program include:

ASCII Mode

Sending <CTRL>P will read back the ASCII firmware ID (<CTRL>P is hex $10)

Hex Mode

(Note that in hex mode bytes are sent by entering the two digit hex code and hitting carriage return, the $ prefix character is not required)

Set LCD contrast: $20 followed by contrast byte (look at the LCD initialization for a rough idea of what the value should be)

Set data value: $22 followed by data_num, dataL, and dataH bytes where data_num is 0-7 for display data0 to data7 and dataH:dataL is a 16-bit signed binary number

Read parameter value: $23 followed by param_num. Reads back paramL and paramH where param_num is 0-7 for param0 to param7 and paramH:paramL is a 16-bit signed binary number.

The RS-232 interface (U5) works with most PC serial ports. Newer laptops do not include an RS-232 port, so an RS-232 to USB adapter or interface is required. Our products use the FTDI Chip FT232RL for this purpose (refer to http://www.ftdichip.com for details). The FT232RL requires very few external components and we highly recommend this part. For prototyping, test systems, and low volume products, DLP Designs makes a RS-232 to USB interface module P/N DLP-USB232M-G that uses a similar FTDI chip (Digi-Key P/N 813-1018).

As mentioned before, RS-232 communications is at 56000 baud, the highest baud rate that the AVR MCU can accurately generate with a 16 mHz oscillator. Some RS-232 to USB chips, such as the PL-2303 from Prolific do not readily support 56000 Baud. Many commercially available, low cost RS-232 to USB adapters (such as devices sold at Radio-Shack and Best Buy) use the Prolific chip and do not work at 56000 baud.

Conclusions

The Graphics LCD Analysis spreadsheet (refer to PDF file in project folder) shows the results of our testing including total cost of the various LCDs. The Optrex LCD had the best display quality and moderate cost. It also offered the advantage of convenient mounting and having connectors for all signals. This is an important consideration. Rework is much more difficult for LCDs such as the Displaytech and EA units that have many PCB pins. An additional advantage of the Optrex LCD is that it can be spaced slightly above the PCB assembly allowing components to be located underneath.