ZBasic

Application Note

AN-220 Interfacing a Parallel LCD

Introduction

Many microcontroller projects require some type of display device in order to indicate system status or provide feedback to the user. There are many choices for such a display but often a small character oriented display, perhaps 2 lines by 16 characters, serves quite well. This type of display is readily available from a variety of sources (e.g. Spark Fun Electronics or Goldmine Electronics).

Typically, the interface to these displays is a parallel interface (4 or 8 data bits and three control lines) but they are also available with a "backpack" that presents a serial interface requiring fewer I/O lines. The raw displays, i.e., those without the additional serial interface, are usually quite a bit less expensive - 2 to 5 dollars vs. 15 to 20 dollars – so it is often desirable to use the less expensive types. This application note illustrates several methods for connecting a parallel LCD to a ZX microcontroller.

Overview of a Typical Parallel LCD

Many of the inexpensive character oriented LCDs that are available support the Hitachi 44780 controller interface. The LCD may or may not actually have the Hitachi controller chip on it – the HD44780 interface has become a de facto standard for small character oriented LCDs. The Hitachi 44780 interface is usually presented on 14 pins arranged in a 2x7 or a 1x14 format with 0.100" spacing. The standard signal assignment for those pins is shown in the table below. Occasionally, you'll find a parallel LCD with 16 pins where the first 14 match the table below.

Table 1: Hitachi 44780 Pinout

Pin	Function	Pin	Function
1	Ground	8	Data Bit 1
2	Power (+5 volts)	9	Data Bit 2
3	Contrast	10	Data Bit 3
4	Register Select (0= command, 1= data)	11	Data Bit 4
5	R/W (0=write, 1=read)	12	Data Bit 5
6	Enable Strobe	13	Data Bit 6
7	Data Bit 0	14	Data Bit 7

The HD44780 interface may be operated in either 8-bit or 4-bit data mode. All of the interface methods presented in this application note use the 4-bit data mode so data bits 0-3 are connected to ground. The HD44780 interface may be operated in "write only" mode or in read-write mode. If the write-only mode is chosen, the R/W signal should be connected to ground. Since this prevents the interface software from reading the display status, delays must be built into the interface code to allow time for the display to process the commands sent. All of the interface methods described in this application note connect the R/W signal to the interface circuitry but the example code can (except for one case) be configured to read the display status or implement the necessary delays.

Communication with the LCD is performed by setting the states of the Register Select (RS), R/W and data signals and then pulsing the Enable strobe. A typical minimum pulse width for the Enable strobe is 250nS. In read mode, the data lines must be read while the Enable strobe is active (high).

The example software that accompanies this application note can be configured for each of the interface methods presented in this application note.

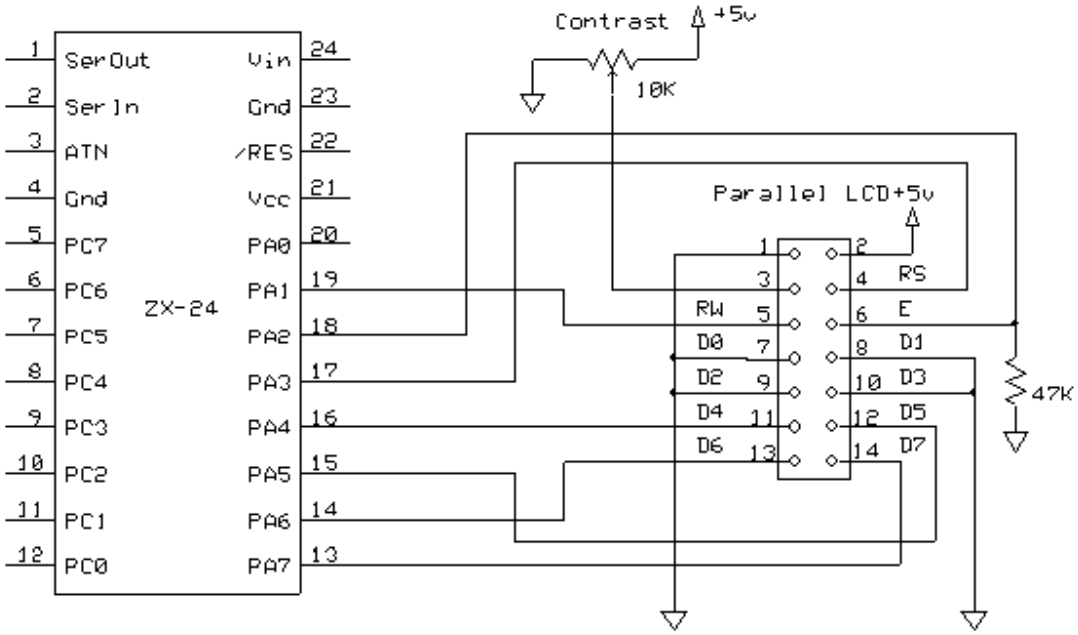
Published June 2008

LCD Initialization

The recommended method for initializing the LCD in 4-bit data mode is described in various documents available on the Internet including at *http://www.myke.com/lcd.htm*. Essentially, a special value is written to the LCD multiples times with intervening delays and then, finally, a command is written to the LCD to enable 4-bit mode. This method is implemented in the example code accompanying this application note.

Example Interface – Direct Connection

The most straightforward method to use to connect a parallel LCD to a ZX microcontroller is to make direct connections between ZX I/O lines and the control/data lines of the LCD. This requires 7 I/O lines when implementing a read-write interface or 6 when implementing a write-only interface. The schematic below illustrates the direct connection method. Note that the contrast signal is derived from a 10K potentiometer that is also connected to +5/ground. The 47K pull-down resistor on the Enable signal may not be required but is included for completeness – it ensures that the Enable signal is held inactive prior to the I/O lines being initialized.



Direct Connection LCD Interface

The example code included with this application note is by default configured for the direct connection. The LCD is initialized by invoking the subroutine InitLCD(). The applicable code in InitLCD() is reproduced below.

```
    ' initialize the LCD interface pins
   Register.PortA = Register.PortA And &H01
   Register.DDRA = Register.DDRA Or &Hfe

   ' delay for power-on initialization
   Call Delay(LCD_POWER_ON_DELAY)

   ' put the display in 4-bit mode
   Call SetBits(Register.PortA, LCD_DATA_MASK, LCD_4BIT_INIT_CMD1)
   For i = 1 to 3
       Call PulseOut(pinE, ePulseWidth, 1)
       Call PulseOut(0, 5.0e-3, 0)
   Next i
   Call SetBits(Register.PortA, LCD_DATA_MASK, LCD_4BIT_INIT_CMD2)
   Call PulseOut(pinE, ePulseWidth, 1)
```


Published June 2008

The first two lines of code configure the 7 I/O lines as outputs while the third implements the delay that is necessary to allow the LCD to initialize itself. The remainder of the code writes the special initialization sequence to put the LCD in 4-bit interface mode.

The remainder of the example code provides several routines for managing the display as shown in the table below.

Table 2: Display Subroutines

Name	Description
DisplayClear	Clears the LCD of all characters.
DisplaySetPos	Sets the cursor to a given row/column position.
DisplayChar	Displays a character at the current cursor position.
DisplayCharAt	Displays a character at a specified row/column.
DisplayStr	Displays a string beginning at the current cursor position.
DisplayStrAt	Displays a string at a specified row/column.
DisplayByteDec	Displays the decimal value of a byte at the cursor position.

Most of the display subroutines eventually call a low level subroutine named lcdSend() that sends either a command or data to the LCD. The core of this routine for the directly connection method is shown in the code excerpt below.

```
 ' set the control lines
Call SetBits(Register.PortA, &H0e, mode)

' send the high 4 bits
Call SetBits(Register.PortA, LCD_DATA_MASK, bval And LCD_DATA_MASK)
Call PulseOut(pinE, ePulseWidth, 1)

' send the low 4 bits
Call SetBits(Register.PortA, LCD_DATA_MASK, Shl(bval, 4) And LCD_DATA_MASK)
Call PulseOut(pinE, ePulseWidth, 1)
```

The first line sets the state of the Register Select line (the 'mode' value having been previously masked down to just that bit). The next two lines output the most significant 4 bits of the command/data value and then pulse the Enable strobe. Finally, the least significant 4 bits of the command data value are output and the Enable strobe pulsed again.

The code excerpt above is surrounded by additional logic that, depending on the configuration, either queries the LCD status before beginning the code sequence or implements the required delay after the command/data value is written.

Example Interface – 3-wire Interface Using a Shift Register

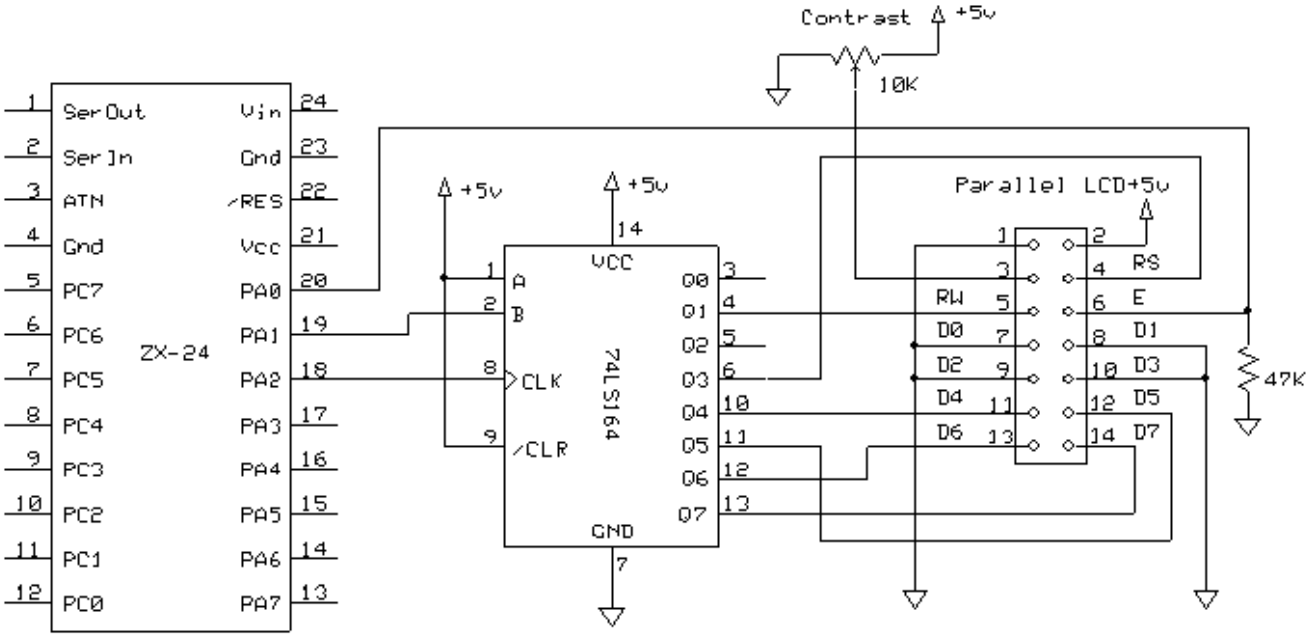
The primary advantage of the 3-wire interface as compared to the direct connection method is that the number of I/O lines required is reduced from seven to three. The downside is that the communication with the display takes somewhat longer because the data must be shifted out serially. The schematic below illustrates this technique.

This circuit uses a shift register to hold the Register Select, R/W and data signals and a directly connected Enable strobe. The Enable strobe cannot be derived directly from the shift register because the bits "ripple" through it during the shift cycle. This would cause unwanted transitions on the Enable signal during the shifting.

The example code for this connection method always shifts out 8 bits to the shift register. By rearranging the assignments of the signals to the shift register outputs so that the least significant 6 outputs are used, the shift count could be reduced by 25% resulting in slightly faster operation. The connection assignment shown was chosen due to its similarity to the other connection methods, resulting in simpler multi-interface code.

Published June 2008

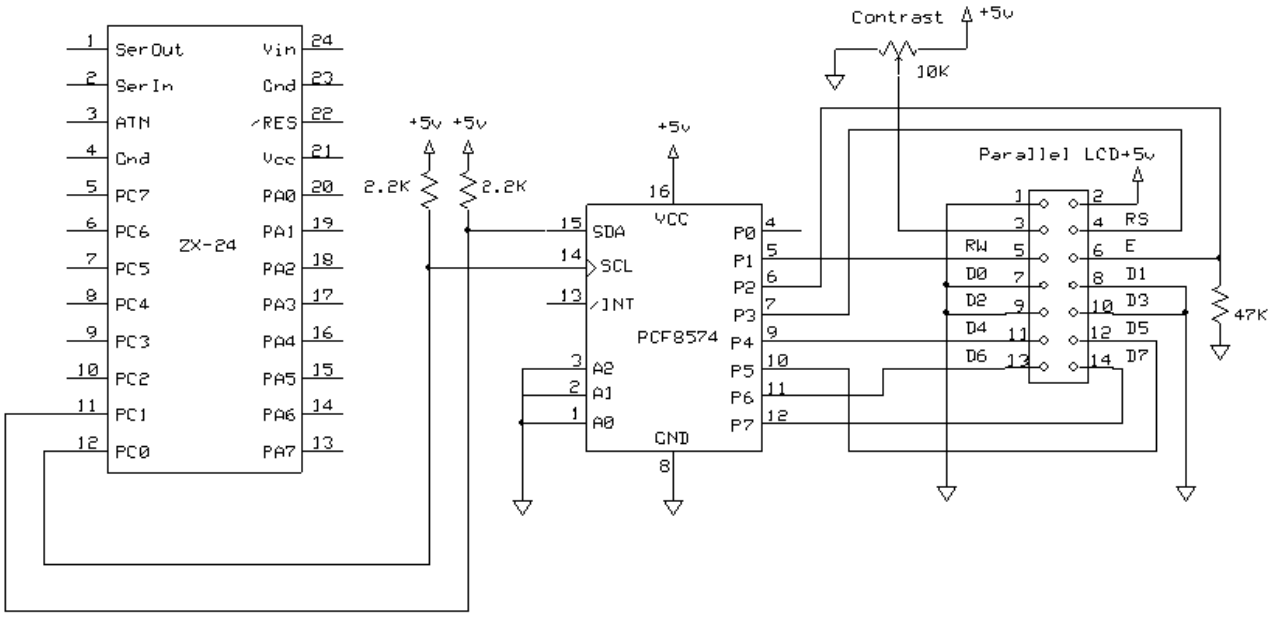
Note that this connection method results in an write-only interface. It is not possible to implement the LCD status checking without employing additional circuitry. Because of this, the R/W connection could be eliminated; reducing to five the number of bits needed in the shift register.



3-wire Serial Interface Using a Shift Register

Example Interface – 2-wire Interface Using an I2C I/O Expander

The primary advantage of using this interface method is a further reduction in the number of ZX I/O lines required – just two. If you are already using other I2C devices in your application the LCD can be added with no net increase in the number of I/O lines used since the SDA and SCL lines can serve multiple devices. The disadvantage of this technique, illustrated in the schematic below, is the relatively slow maximum shift rate that is supported by the I/O Expander used in this example. It is limited to 100KHz maximum.



LCD Interface Using an I2C I/O Expander

Published June 2008

The example code for operating the 2-wire interface is quite similar to that for the 3-wire interface; the ideas are the same but different ZBasic System Library calls are used for initializing the LCD and sending it commands and data.

Software

A key part of this application note is the ZBasic software provided in the associated .zip file. The project is named LCD_4P.pjt (denoting a 4-bit parallel interface). The file LCD_4P.bas contains all of the code for all of the interfacing methods described in this application note with code specific to each interface method contained within conditional constructs. The software is configured for a particular interface method by manipulating the `#define` constructs excerpted below. If the first one is uncommented, the code will be configured for the I2C interface. If the first one is commented out and the second one is not, the code will be configured for the 3-wire serial interface. If both are commented out the code will be configured for the direct connection method. The third `#define` construct controls whether the LCD status will be checked or if delays will be used to allow the LCD time to complete its operations.

```
' Uncomment the following line to configure for the I2C interface.
'#define LCD_I2C

' Uncomment the following line to configure for a 3-wire interface
' using the 74LS164 shift register.
'#define LCD_3WIRE

' Uncomment the following line to implement status checking.
' Otherwise, post-operation delays are used.  The latter method
' allows operation with the R/W line wired to logic zero.  This
' cannot be used with the 3-wire interface.
'#define CHECK_BUSY
```

Author

Don Kinzer is the founder and CEO of Elba Corporation. He has extensive experience in both hardware and software aspects of microprocessors, microcontrollers and general-purpose computers. Don can be contacted via email at dkinzer@zbasic.net.