Using the Parallel Printer Port

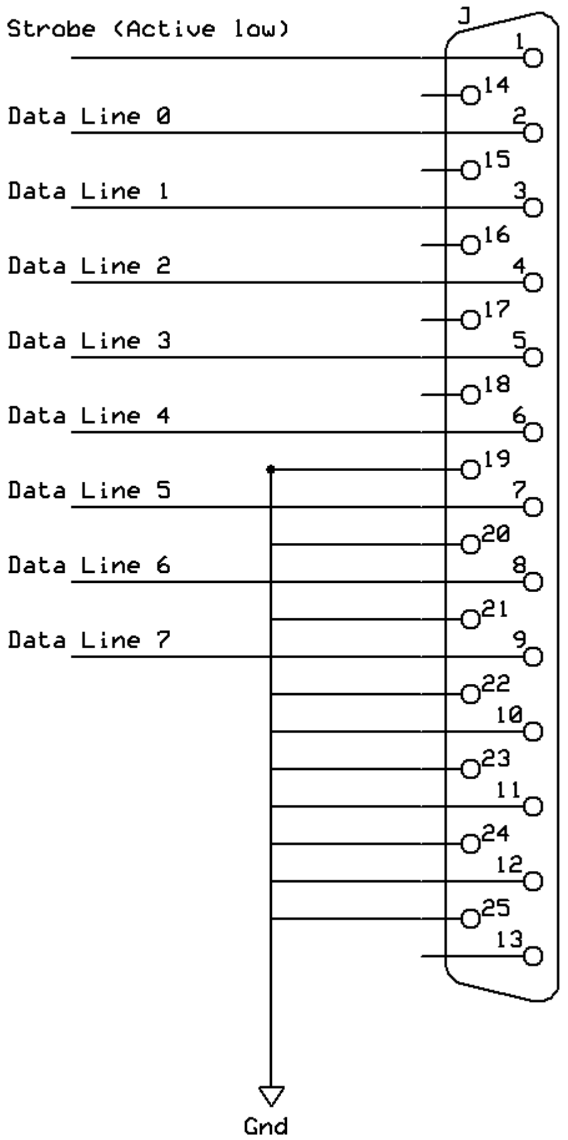
Originally Published in QL Today Vol 16, Issue 4, Jun-August 2012

In part 4 of my series, I2C Interface for QL Emulators, I have shown how you can use an LCD Display using the PCF8574 device with an RS232 serial to I2C converter. Which provides an 8 bit parallel I/O port. However you can have a simple 8 bit output only port from your QXL, QPC2, Qemulator and Super Gold Card equipped QL's. So a LCD display can be used in a similar manner. Or for that matter any other application, that requires a simple output port. This is what I will cover in this article and also expand a little more on using LCD displays, which can also be applied to I2C users. I must point out I have not been able to get this to work with Qlay or QL2K. I have to say I have not spent a great deal of time looking at these two emulators in this regard, so that is not to say, with some work they cannot be made to work. But on the systems I have tested and know to work, I think this will cover most QL/SMSQ system users.

Let me start with QXL, QPC2 and Qemulator users, since this is the simplest. Since in all these cases we are using a PC's hardware, which works in a slightly different manner to the Super Gold Cards printer port. In so far as implementing a parallel output port is concerned.

First your PC must have a parallel printer port, this will be a 25 way female D-Type connector on your machine. If you do not have one, then a USB to parallel converter may work. However if you use one of these, do ensure that in the Device Manager of your PC operating system, under the Ports (Com & LPT) shows 'LPT1'. Not all do.

So for QXL, QPC2 and Qemulator users you only need the following interface.

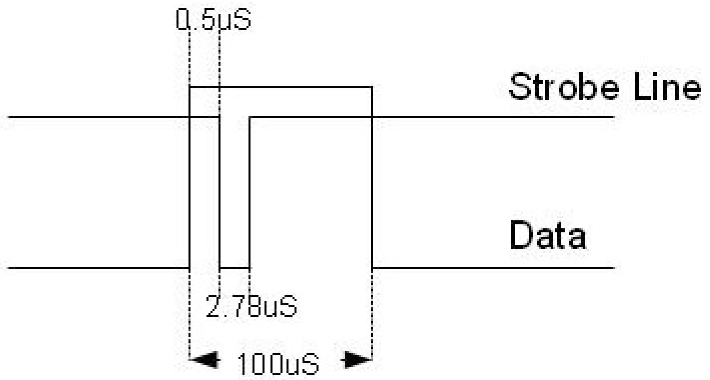


I have shown the strobe line, but in the case of PC based systems this is not required. The grounding of the various control lines gives us latched outputs. The following listing will make the data lines go up and down. If you put an LED, anode to the data line and via a resistor of say 470ohm to ground, the LED will flash. To enable the 'PAR' device to work you must ensure TK2 is running.

```
10 TK2_EXT
20 OPEN#3;par
30 REPeat loop
40 PRINT#3;CHR$(255);
50 PAUSE 25
60 PRINT#3;CHR$(0);
70 PAUSE 25
80 END REPeat loop
```

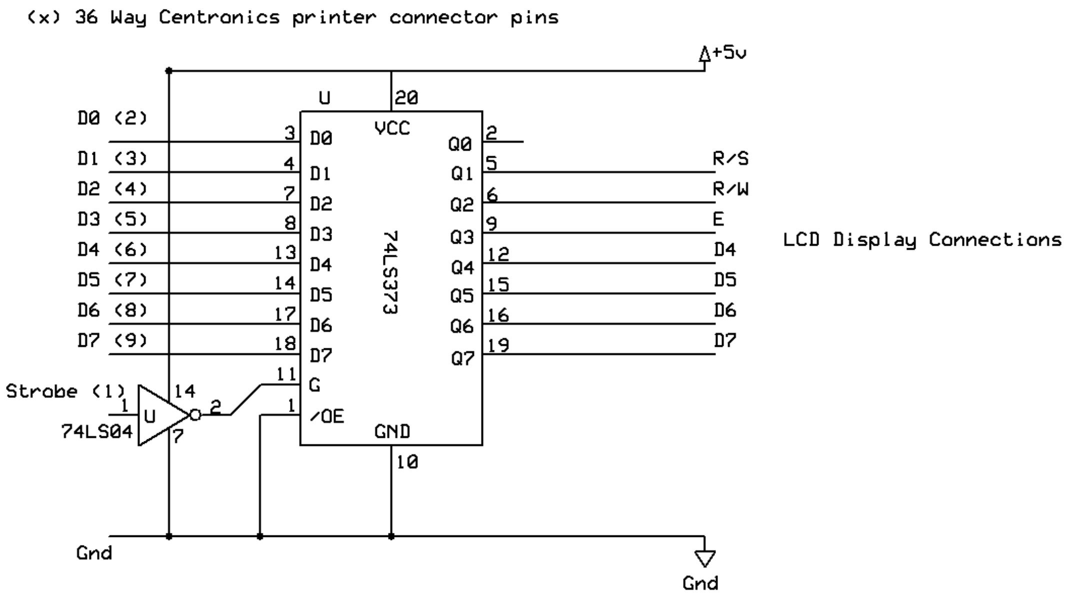
If you a Super Gold Card user then things get a little more complicated. The parallel printer port on the Super Gold Card only has the data lines and the strobe line. It does not have the other control lines that a standard PC parallel printer port has. So will not provide latched output that the PC parallel port can. However having the strobe line we only need to add a latching circuit.

The signal timing from the SGC is shown below:-



Output from Super Gold Card

So the data lines are set, and the flip flops in the latch are then clocked by the strobe line. This latches the output from the flip flops low or high as required and hold that state until another data and strobe event take place. The circuit to achieve this is shown below.



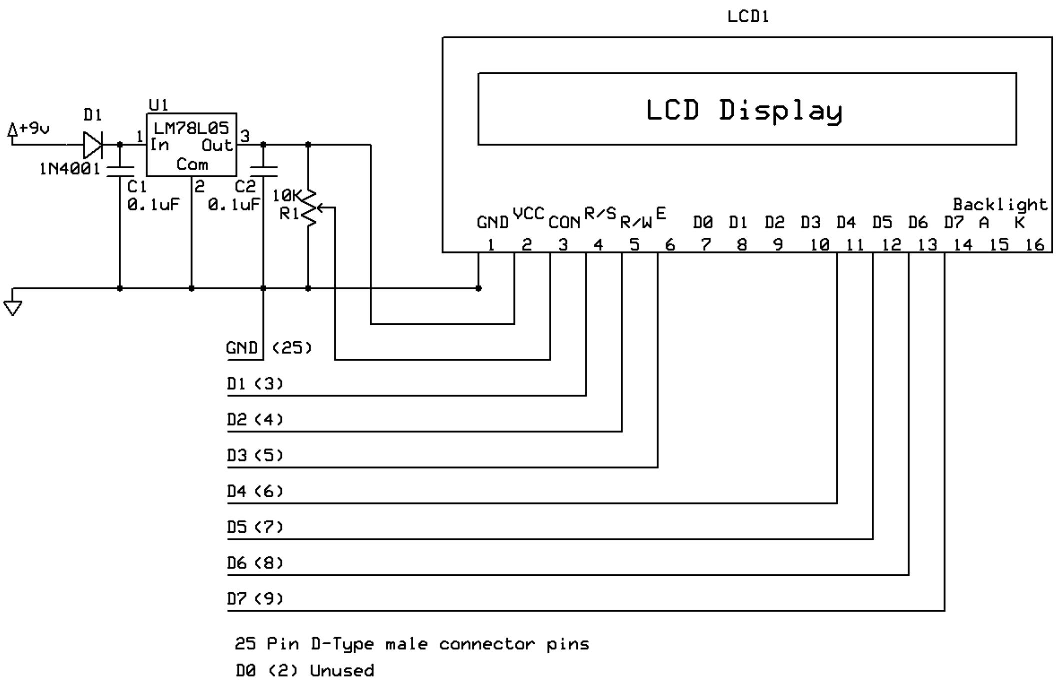
The 74LS373 contains 8 flop flops with a common clock (G), so that any data pattern will be latched on to the output when trigged by the strobe signal. The strobe signal is inverted, hence the 74LS04 inverter chip. The 74LS04 has 6 inverters, in this case only one is used.

I have shown the 36 pin centronics connector pin numbers so you can use the printer cable that came with your SGC. You will need a female centronics connect to be able to use this. These are available from the usual electronic component suppliers If you look in the SGC manual it shows the IDE pin header connections. Which is another way of connecting to the SCG.

No power is available from parallel ports, unlike the USB to I2C converters, so arrangements have to be made to provide 5V to this circuit. When we move on to driving LCD displays, you will see such an arrangement.

To test this circuit the same listing as shown above for the PC parallel port can be used. Again you can connect LED(s) in the same way. So anode to the output data line (Q0-Q7), then cathode to ground via a resistor (470 ohm).

So now to connecting a LCD display.



I have shown the 25 way D-Type connector pin numbers if you are using a PC parallel port, and the data line numbers (D1 to D7) for the SGC latch circuit. So if you are using a PC, combine this circuit with the 25 way D-Type circuit at the beginning of this article.

As you may see in the diagram above, there is a 5V regulator(U1), this is used to supply the LCD display itself but can be also used to supply 5V to the SGC latch circuit. You can take the power for this from pin 3 (out) of U1.

So that deals with the hardware what about the software. Find below a listing which is common to all the following routines to show what can be done with a LCD display. This is not too different from the listing in the last I2C article, but has been changed for parallel port use.

```
10 REMark Parallel LCD Experiments common routines
20 REMark EPE Feb 1997, How to use intelligent LCD's
30 init
40 init_LCD
50 clear_LCD
900 PAUSE
910 clear_LCD
920 PRINT "End"
940 PRINT#3;CHR$(0);
950 CLOSE#3
960 STOP
970 :
1000 DEFine PROCedure init
1010 rs=2:rw=4:en=8:REMark rs is register select, rw is read/write (only needed for using
display ram)
1020 OPEN#3;par
1030 CLS
1040 PRINT#3;CHR$(0);:REMark to set all parallel data lines to 0
1050 FOR r=1 TO 3:REMark This FOR NEXT loop is to ensure the LCD is in it initial mode
which is 8 bit interface mode, after power up. This does not reset the LCD Display.
```

```
1060 load_LCD 48,0
1070 NEXT r
1080 PRINT "Ensured LCD Display in 8 bit interface mode."
1090 PRINT#3;CHR$(0);:REMark to set all parallel lines to 0
1100 END DEFine init
1110 :
1120 DEFine PROCedure init_LCD
1130 PRINT#3;CHR$(0);:REMark to set all parallel line to 0
1140 load_LCD 32,0:REMark Sets LCD to 4 bit interface mode
1150 PRINT "Set LCD to 4 bit interface mode."
1160 load_LCD 32,0:REMark LCD Function set to 2 line mode, 1st Nibble
1170 load_LCD 128,0:REMark LCD Function set to 2 line mode, 2st Nibble
1180 REMark Note, be careful not to set D4(16) when addressing the Function Set register,
since this will return the LCD to 8 bit interface mode.
1190 PRINT "Function Set to 2 line mode."
1200 load_LCD 0,0:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, this is
the Display On/Off & Cursor register, 1st Nibble
1210 load_LCD 240,0:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, this is
the Display On/Off & Cursor register, 2nd Nibble
1220 PRINT "Set LCD to Display On, Cursor On and Cursor Blinking."
1230 load_LCD 0,0:REMark Sets LCD to Character Entry mode, Increment Cursor Position
No Display shift, 1st nibble.
1240 load_LCD 96,0:REMark Sets LCD to Character Entry mode, Increment Cursor Position
No Display shift, 2st nibble.
1250 PRINT "Set LCD to Increment Cursor Position and No Display shift."
1260 END DEFine init_LCD
1270 :
1280 DEFine PROCedure LCD_message(message$)
1290 mlen=LEN(message$)
1300 FOR mc=1 TO mlen
1310 ms$=message$(mc)
1320 ms=CODE(ms$)
1330 nib1=(INT(ms/16))
1340 nib2=ms-(nib1*16)
1350 nib1=(nib1*16):REMark 1st nibble
1360 nib2=(nib2*16):REMark 2nd nibble
1370 load_LCD nib1,rs
1380 load_LCD nib2,rs
1390 PRINT ms$;" ASCI Character Number:";ms;" First Nibble:";nib1;" Second
Nibble:";nib2
1400 NEXT mc
1410 END DEFine LCD_message
1420 :
1430 DEFine PROCedure move_second_line
1440 load_LCD 192,0:REMark Move to start of second line 1st nibble.
1450 load_LCD 0,0:REMark Move to start of second line 2st nibble.
1460 PRINT "Moved to second line on LCD"
1470 END DEFine move_second_line
1480 :
1490 DEFine PROCedure load_LCD(lcd_data,rsm)
1500 PRINT#3;CHR$(rsm);
1510 PRINT#3;CHR$(en+rsm);
1520 PRINT#3;CHR$(lcd_data+en+rsm);
1530 PRINT#3;CHR$(lcd_data+rsm);
1540 REMark PAUSE:REMark Use this to step though the workings on the LCD.
```

```
1550 END DEFine load_LCD
1560 :
1570 DEFine PROCedure more_characters
1580 PRINT "Press any key to see more of the character set           "
1590 PAUSE
1600 clear_LCD
1610 END DEFine more_characters
1620 :
1630 DEFine PROCedure clear_LCD
1640 load_LCD 0,0:REMark clear LCD and return cursor to start, 1st nibble.
1650 load_LCD 16,0:REMark clear LCD and return cursor to start, 2st nibble.
1660 load_LCD 128,0:REMark reset display address to 0, 1st nibble.
1670 load_LCD 0,0:REMark reset display address to 0, 1st nibble.
1680 PAUSE 10:REMark giving time for display to respond to the last commands
1690 END DEFine clear_LCD
32000 DEFine PROCedure UPDATE
32010 SAVE win1_parallel_LCD_common_bas
32020 PRINT "Update Complete"
32030 END DEFine UPDATE
```

Having now got the common routine which is commented, so you can follow what is going on. Now the first experiment which will have the display, display “QLToday Forever” across two lines of the display. Assuming you are using a display with at least 2 lines.

```
60 :
70 LCD_message "QLToday"
80 move_second_line
90 LCD_massage "Forever"
100 :
```

Now in the second experiment, here we will make the LCD display all the character set. This is not the most compact program. But I have written this deliberately so you see what is going on. That some characters use standard ASCI code, but others do not. So shows you how to display the Chinese and Greek characters. Now I guess not many will need Chinese, but the symbols and Greek are useful.

```
60 LCD_message " !#$%&'()*+,-/:;"
70 move_second_line
80 LCD_message "<=>?0123456789"
90 more_characters
100 LCD_message "ABCDEFGHIJKLMNOP"
110 move_second_line
120 LCD_message "QRSTUVWXYZ"
130 more_characters
140 LCD_message "abcdefghijklmnop"
150 move_second_line
160 LCD_message "qrstuvwxyz"
170 more_characters
180 PRINT "Now the other characters"
190 FOR c=91 TO 96:LCD_message CHR$(c):NEXT c
200 move_second_line
210 FOR c=123 TO 127:LCD_message CHR$(c):NEXT c
220 more_characters
230 PRINT "Chinese Characters":REMark Now the LCD will display Chinese characters
240 FOR c=160 TO 175:LCD_message CHR$(c):NEXT c
```

```
250 move_second_line
260 FOR c=176 TO 191:LCD_message CHR$(c):NEXT c
270 more_characters
280 FOR c=192 TO 207:LCD_message CHR$(c):NEXT c
290 move_second_line
300 FOR c=208 TO 223:LCD_message CHR$(c):NEXT c
310 more_characters
320 PRINT "Greek Characters":REMark Now the LCD will display Greek Characters"
330 FOR c=224 TO 239:LCD_message CHR$(c):NEXT c
340 move_second_line
350 FOR c=240 TO 255:LCD_message CHR$(c):NEXT c
360 PRINT "All characters have now been displayed, press any key to clear LCD display and end program."
```

This third experiment demonstrates display scrolling.

```
60 a$="This demonstrates the LCD display scroll":REMark this line is 40 characters long, the capacity of one line on the LCD display.
70 LCD_message a$
80 mlen=LEN(a$)
90 PAUSE 100:REMark sets the delay before the display scrolls.
100 FOR c=1 TO (mlen-16):REMark assuming a 16 character per line display.
110 load_LCD 16,0:REMark Set scrolling one character position to the left, 1st nibble
120 load_LCD 128,0:REMark Set scrolling one character position to the left, 2nd nibble
130 PAUSE 25:REMark Sets the speed the display scrolls.
140 NEXT c
```

The forth and last experiment demonstrates setting up user defined graphics. The first 7 locations of the character set are user definable. This routine just addresses this first character with a stick man. Again I have written this deliberately so you see what is going on.

```
60 load_LCD 128,0:REMark set display address to 0, 1st nibble
70 load_LCD 0,0:REMark set display address to 0, 2nd nibble.
80 FOR a=0 TO 7
90 LCD_message CHR$(a)
100 NEXT a
105 PAUSE
110 load_LCD 64,0:REMark set CGRAM address to 0, 1st nibble
120 load_LCD 0,0:REMark set CGRAM address to 0, 2nd nibble.
130 load_LCD 0,rs:REMark set character 1 address 0, 1st nibble
140 load_LCD 224,rs:REMark set character 1 address 0, 2nd nibble.
150 load_LCD 16,rs:REMark set character 1 address 1, 1st nibble
160 load_LCD 16,rs:REMark set character 1 address 1, 2nd nibble.
170 load_LCD 0,rs:REMark set character 1 address 2, 1st nibble
180 load_LCD 224,rs:REMark set character 1 address 2, 2nd nibble
190 load_LCD 0,rs:REMark set character 1 address 3, 1st nibble
200 load_LCD 64,rs:REMark set character 1 address 3, 2nd nibble
210 load_LCD 16,rs:REMark set character 1 address 4, 1st nibble
220 load_LCD 16,rs:REMark set character 1 address 4, 2nd nibble
230 load_LCD 0,rs:REMark set character 1 address 5, 1st nibble
240 load_LCD 64,rs:REMark set character 1 address 5, 2nd nibble
250 load_LCD 0,rs:REMark set character 1 address 6, 1st nibble
260 load_LCD 160,rs:REMark set character 1 address 6, 2nd nibble
270 load_LCD 16,rs:REMark set character 1 address 7, 1st nibble
280 load_LCD 16,rs:REMark set character 1 address 7, 2nd nibble
```

```
285 PAUSE
290 clear_LCD
300 FOR a=0 TO 7
310 LCD_message CHR$(a)
320 NEXT a
```

You should see the display, display rubbish to start with, this is normal. Once the routine has loaded the user defined graphic, the routine will display the first 7 character locations again, but the first one will now show the stick man.

As I said before, these routines will work with the I2C LCD common routine published in my last article.

Until next time have fun.

References

Farnell
http://uk.farnell.com/jsp/home/homepage.jsp

RS
http://uk.rs-online.com/web/

PCF8574(A) Data Sheet
http://www.nxp.com/documents/data_sheet/PCF8574.pdf
http://focus.ti.com/lit/ds/symlink/pcf8574.pdf

How to use intelligent LCD's Part 1
Everyday Practical Electronics, February 1997
Downloadable from http://www.wizard.org/auction_support/lcd1.pdf

How to use intelligent LCD's Part 2
Everyday Practical Electronics, March 1997
Downloadable from http://www.wizard.org/auction_support/lcd2.pdf

Character LCD Displays – Part 1
http://www.protostack.com/blog/2010/03/character-lcd-displays-part-1/

Hitachi LDC Driver chip, advanced stuff in here. But does show all the capabilities of LCD displays that use this chip.
http://www.sparkfun.com/datasheets/LCD/HD44780.pdf