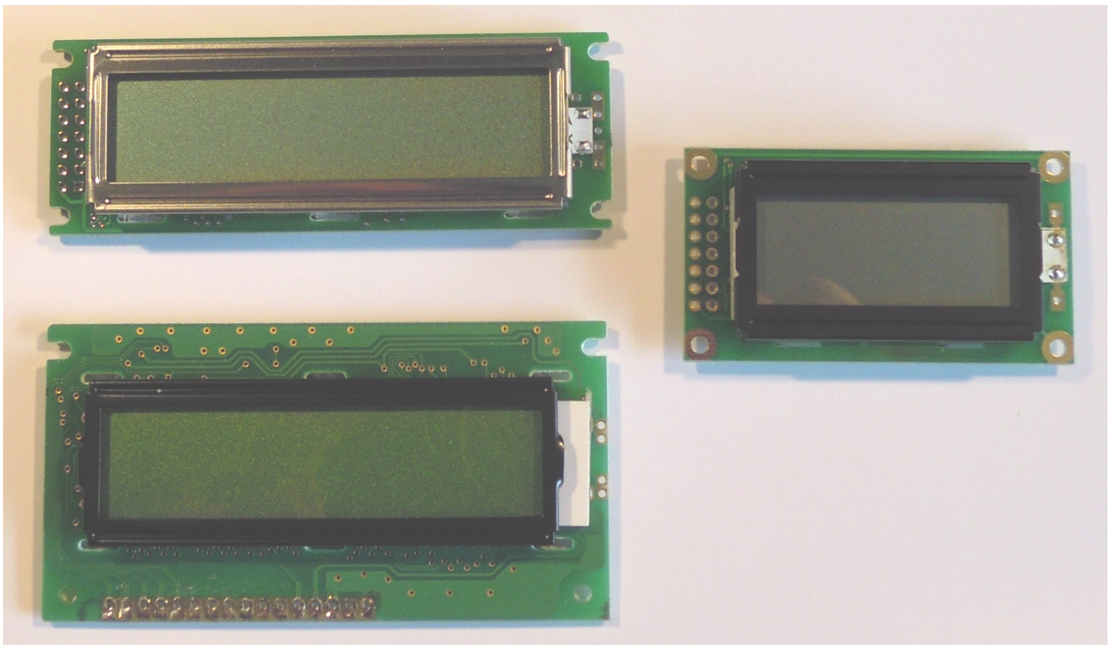
I2C Interface for QL Emulators Part 4

Originally Published in QL Today Vol 16, Issue 3, March-May 2012

In the first three parts of this series we have looked at the principles of the I2C bus and some the devices that can be used. Such as parallel and AD/DA interfaces, RTC (Real Time Clock), RAM (Random Access Memory) and a digital potentiometer. This time will we look a practical use for the PCF8574 parallel device we looked at in part one of this series.

In this article I am going to show you how you can drive a LCD (Liquid Crystal Display). In part one, I showed a circuit to be able to demonstrate input and outputs to and from the PCF8574 device. In this application we only need to use the output function of the PCF8574 since we are only going to drive the LCD display at a simple level. However the input/output capability of the PCF8574, can fully exploit all the available functions of the LCD displays such as these. See the HD44780.pdf in the references below, for further information on this.

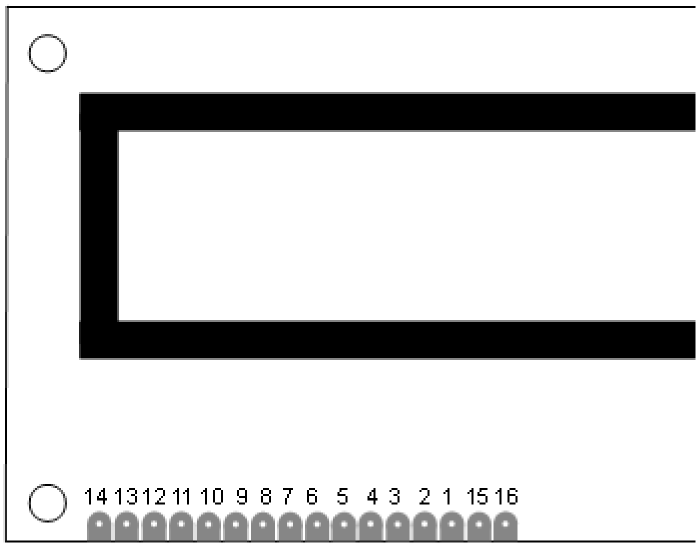
LCD's (Liquid Crystal Display's) come in many different sizes, for example common ones are 8 characters by 2 lines, 16 Characters by 2 lines, 16 characters by 4 lines, 20 character by 1 line, 20 character by 2 lines and 40 character by 2 lines. They are easily available from the likes of Farnell and RS via mail order or via their web sites. There are other suppliers as well.

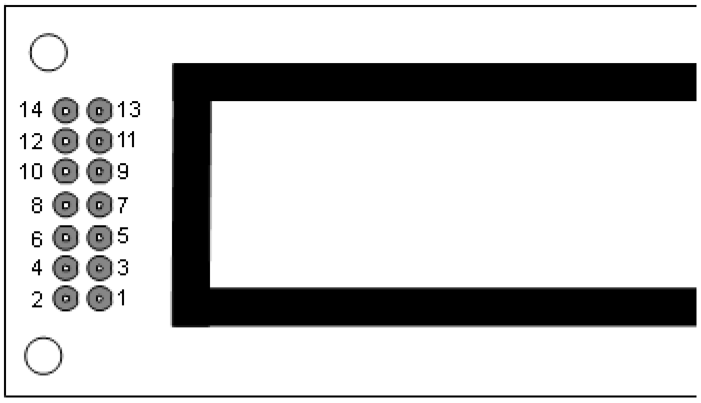


Some example LCD Display's, note the different connection layouts on the two 16 x 2 displays on the left.

One of the big advantages with these displays is that they have a standard interface and protocol using the Hitachi HD44780 driver chip. So any software you write will work with all versions of these displays. The only consideration is how many characters you wish to display at any one time. Which should govern the size of display you use. LCD’s come with or without back lighting, it is your choice which to use. However back lit versions do tend to be more expensive. Also do check the display you are using has the HD44780 chip. Most do, but worth double checking. One other word of caution, some modules require as low as –7V on their Vee (Contrast) pins, these versions will not work in this circuit, worth double checking the datasheet for the device

The standard connection layout’s of the displays is shown below. There are some variations out there, so again do check the datasheet for the LCD you are using. But these are the common ones.



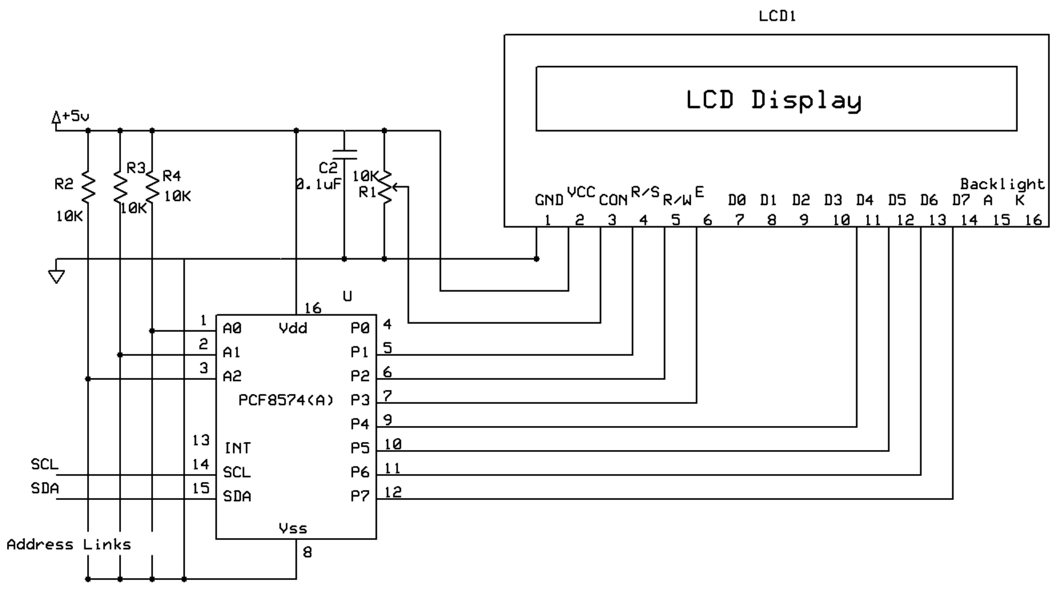


The pin out functions for all LCD types is show below:-

Pin no	Name	Function
1	Vss	Ground 0V
2	Vdd	+ve Supply (5V)
3	Vee	Contrast
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Data Bit 0

8	D1	Data Bit 1
9	D2	Data Bit 2
10	D3	Data Bit 3
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7
15	BLA (If Fitted)	Back Light Anode
16	BLC (If Fitted)	Back Light Cathode

Below you will see the circuit diagram for the I2C to LCD project. As usual do refer to part one which shows the connections to the BV4221 USB to I2C converter. The BV4221 provides the 5V supply, SCL, SDA and GND connections required. So the circuit is self powered from the BV4221 USB to I2C converter.



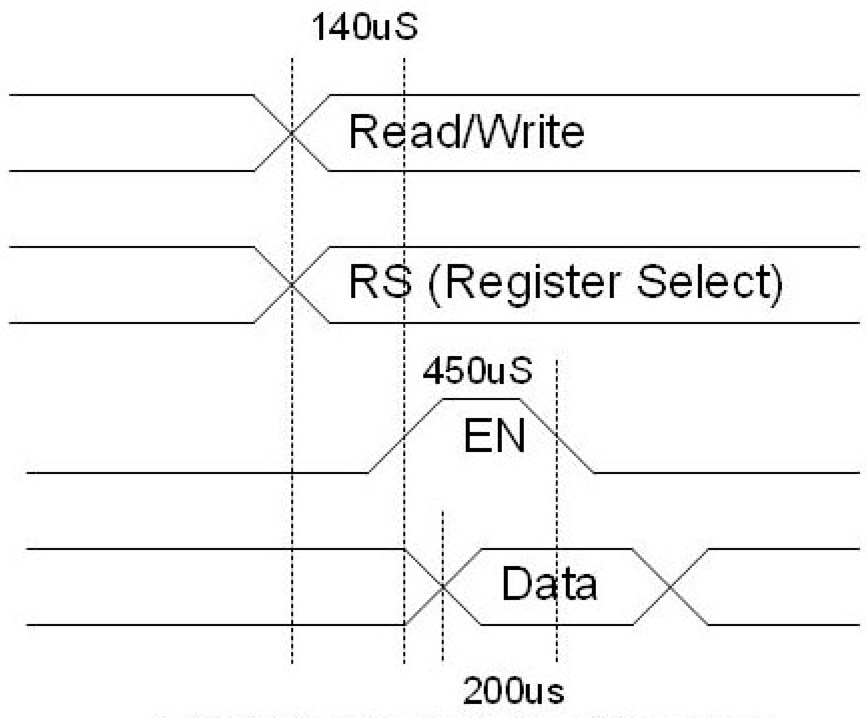
The circuit above is designed to exploit the full capabilities of the LCD module. However in this example we will only be sending data to the module, not reading any data from the module. So plenty of room here for you to experiment.

The potentiometer R1 is the display contrast control, this can be a preset type, since once set, it does not need to be touched often. Note: with multi line displays, this may need to be adjusted if you change the number of lines to be displayed. So for example if is a two line display and you change it to one line operation from two line operation then the contrast will need adjusting. Best to start off with the pot turned the GND end of the travel to start with, then adjust for best contrast once you have run the program. You may see the top line of the display go dark at switch on, this is normal until the LCD display has been initialised, this will be explained later. Do not adjust until the display has been initialised and is displaying the characters you have sent it.

You will see from the above circuit diagram that we are using 7 bits from the PCF8574 to control and send data to the LCD module. These LCD modules have two modes of operation with regards to data. The first being 8 bit words sent in one go. This requires all LCD data pins to be driven. That would then require two PCF8574, one for LCD control and a second to provide the data. Not the most efficient way. In the LCD's second mode of operation, we can use one PCF8574, the lower 4 bits being used (actually only 3 are used, there is a spare pin), for LCD control and the upper 4 bits for data. The data is now sent as two 4 bit nibbles, providing the 8 bit word the LCD requires.

The spare line from the PCF8574, could via a transistor (not shown) could be used to control the back light of the LCD, if you use a LCD module fitted with one. Do not try the drive the back light directly from the PCF8574, since a fairly high current is required.

There is a process that your software needs to perform to get the LCD display working. First the LCD display needs to be initialised. See line 1120 (LCD_init) onwards in the listing below. The first process is to tell the LCD Display to work in 4 bit mode, which is command,'32'. This is sent as a single 8 bit word, since this is D5 bit =1 and the rest of the bits are 0, this is one of the reasons bit 0 to bit 4 has to be grounded so they really are 0. Next we have to set the LCD to accept 8 bits (as 2 nibbles, in this case) and that the display, in this case, is a two line one. The numbers sent are now split into two, the first nibble that is the last 4 bits then the second nibble which is the first 4 bits. All commands and data have to be sent this way from now on. The next command sent set the LCD display to, display ON, cursor ON and Cursor blinking. The last command to initialise the display is Entry mode, Cursor increment and No display shift. All the Commands are shown in the table below. After much experimenting and referring to the data sheets, the data is sent in a given sequence so at to replicate the timing diagram shown below.



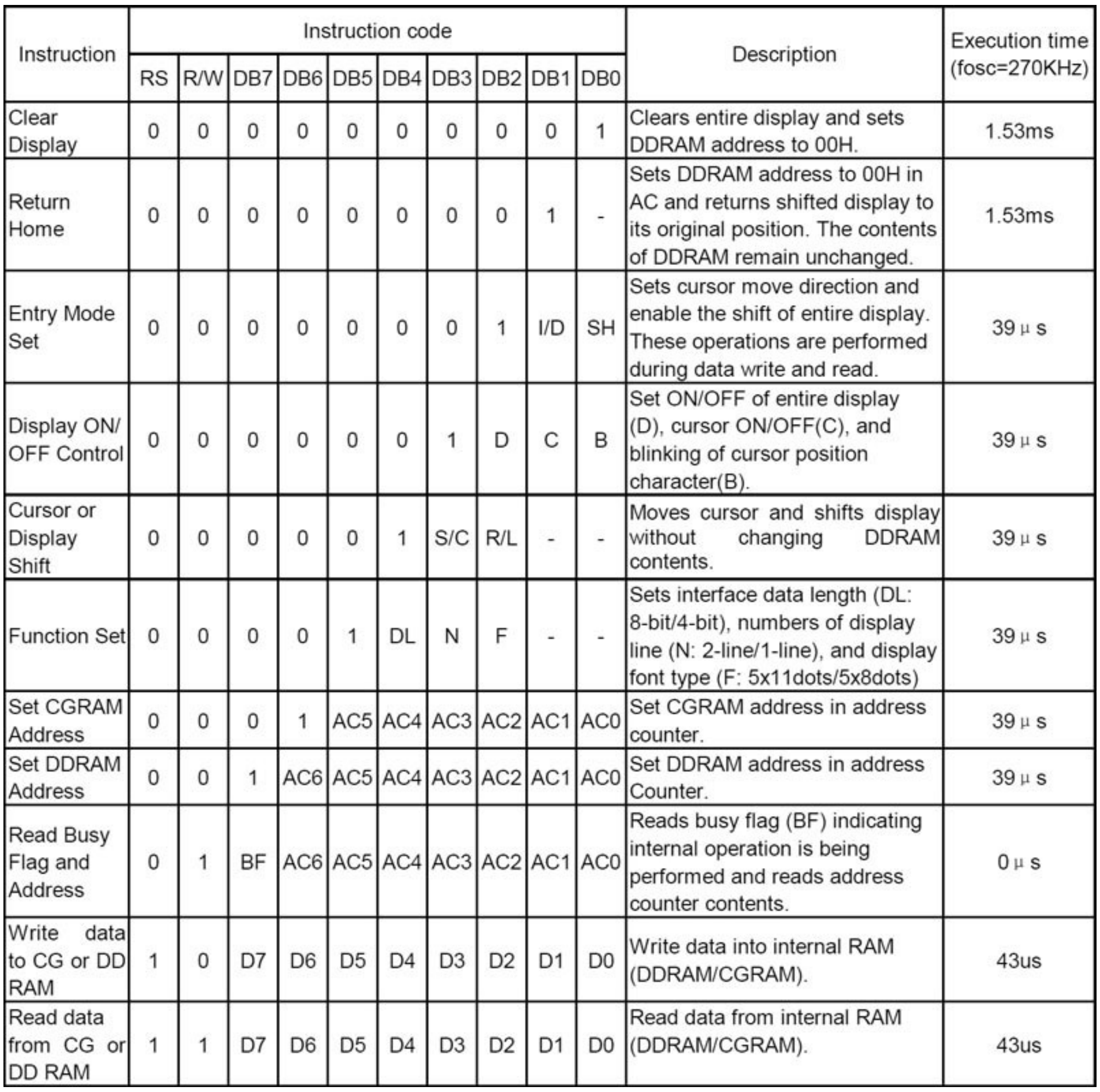
LCD Display Timing Diagram

Now the timing is not important, in fact the timings shown are minimum values as described in data sheets. However the order in which the control and data lines are delivered are important.
So commands and data is sent as follows:-

Command or Data	EN Line	RS Line
0	0	1
0	1	1
Data 1st Nibble	1	1
Data 1st Nibbel	0	1

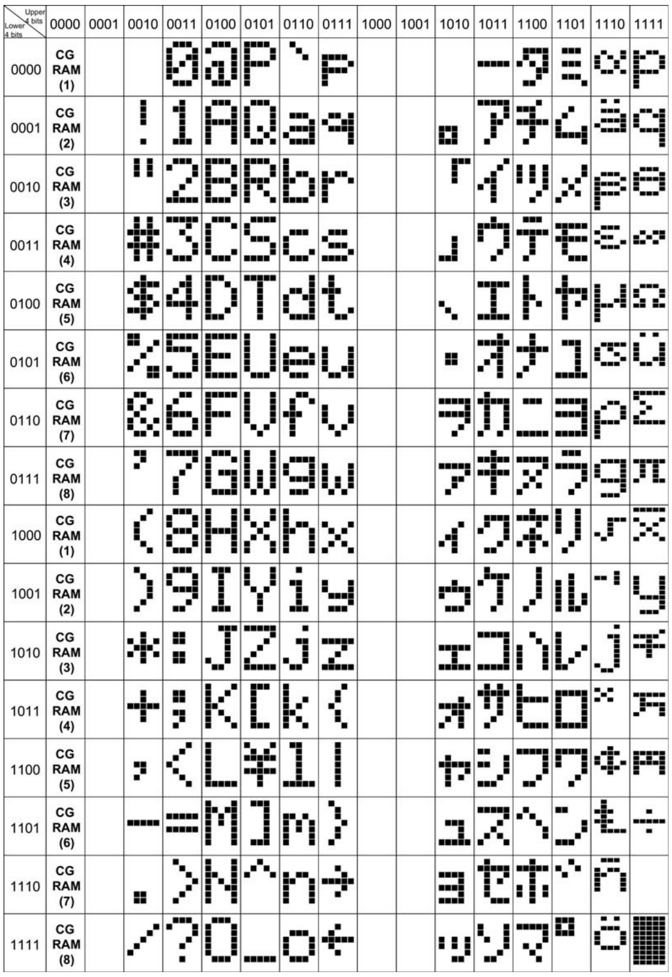
Repeat for the second data nibble. The action of the EN line falling to zero(0), transfers the data within the LCD display.

The enable line is P3 from the PCF8574, so 8, (variable=en) is added to the data number to transfer the data in the LCD display. All the control codes are show in the table below.



Instruction	Instruction code										Description	Execution time (fosc=270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address to 00H.	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Sets DDRAM address to 00H in AC and returns shifted display to its original position. The contents of DDRAM remain unchanged.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Sets cursor move direction and enable the shift of entire display. These operations are performed during data write and read.	39 μ s
Display ON/ OFF Control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF of entire display (D), cursor ON/OFF(C), and blinking of cursor position character(B).	39 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Moves cursor and shifts display without changing DDRAM contents.	39 μ s
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line), and display font type (F: 5x11dots/5x8dots)	39 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μ s
Write data to CG or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data from CG or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Now the text can be sent to the LCD Display is this done in the LCD_message procedure. The entire character set is shown below.



Upper 4 bits / Lower 4 bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
0001	CG RAM (2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
0010	CG RAM (3)		"	2	B	R	b	r			「	イ	ツ	メ	β	θ
0011	CG RAM (4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
0100	CG RAM (5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
0101	CG RAM (6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
0110	CG RAM (7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
0111	CG RAM (8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
1000	CG RAM (1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
1001	CG RAM (2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
1010	CG RAM (3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
1011	CG RAM (4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
1100	CG RAM (5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
1101	CG RAM (6)		-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
1110	CG RAM (7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
1111	CG RAM (8)		/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█

The following listing is just to demonstrate how to drive a LCD display in a basic manner. It has been deliberately slowed down so you can see the display respond to the commands. Clearly in normal operation you do not need to do this. The listing is fully commented so you should be able to follow what is going on.

```
10 REMark Parallel LCD Experiment 1
20 REMark
30 init
40 init_LCD
50 clear_LCD
60 :
70 LCD_message "QLToday"
80 move_second_line
90 LCD_message "Forever"
100 :
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
1700 :
32000 DEFine PROCedure UPDATE
32010 SAVE win1_parallel_LCD_test1_bas
32020 PRINT "Update Complete"
32030 END DEFine UPDATE
```

If all goes well you should get a display like this.

So now time for you to experiment with the initialisation as well as the characters. Also you can program the LCD Display with your own characters, do read the reference documents on how to do this. There is lots of information on the web about these clever displays. So lots to have fun with, like scrolling displays. This article is just to get you going.

Next time we will take a break from the I2C interface and will look at using the parallel printer port on QXL, QPC2, Qemulator and Super Gold card QL's, as a simple output port and show more example of how to drive LCD displays.

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