Improving QL Emulator I/O (Input/Output) Capabilty

Originally published in QL Today March-May 2008

We are supposed to be tinkers, and having seen Hugh Rooms articles on GPS (QL Today Volume 11, issues 2,3 and 4) which involves external equipment in the form of the GPS receiver. I have been playing with Hugh's program with two types of GPS receiver, and I am planning to do a follow up article of my experiences on this subject. However I thought it may be a good idea to look at other forms of interfaces that we could use that may have a wider interest.

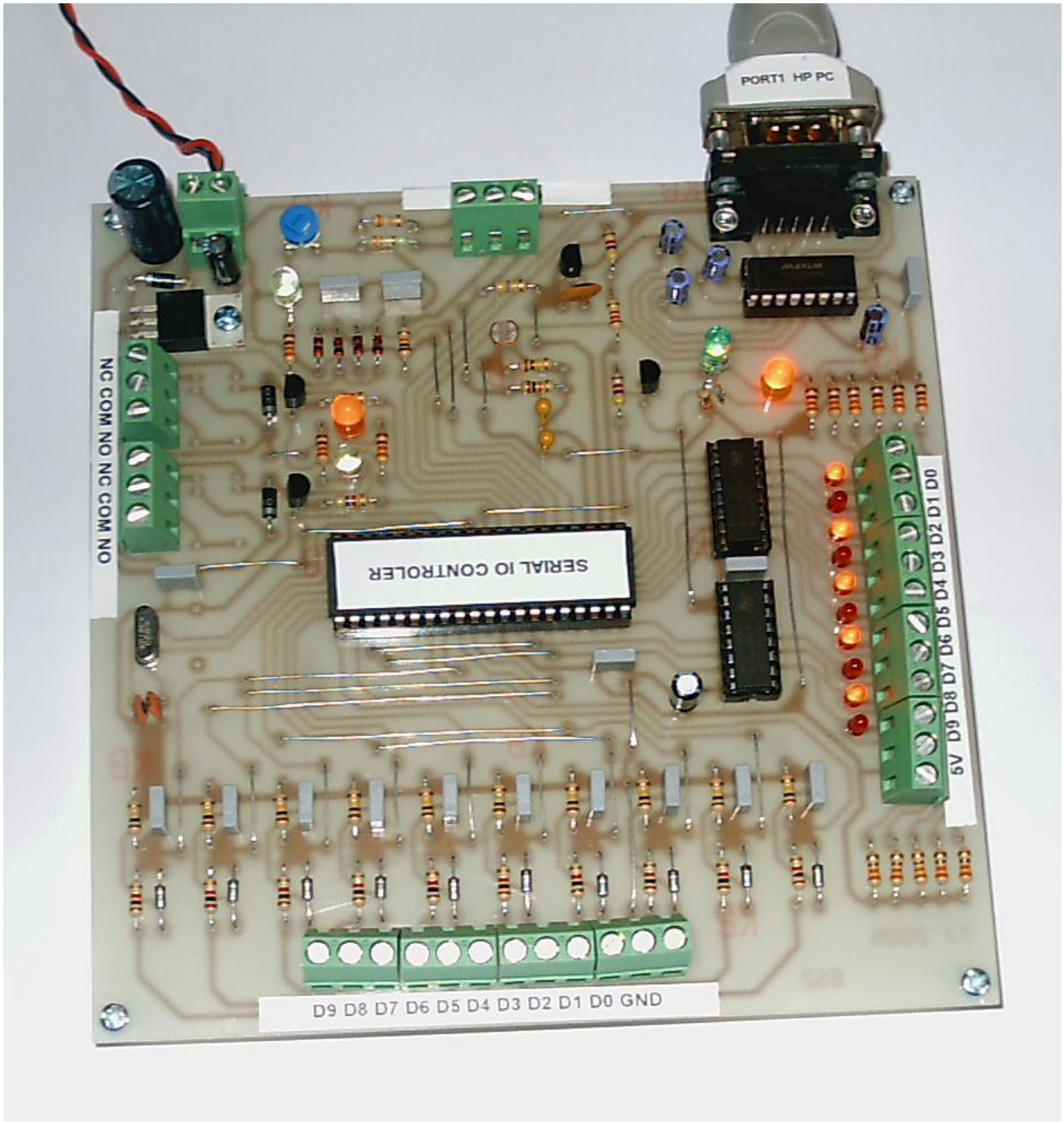
Even if you don't build/use them they may still be of interest. In the last few months there have been two projects published in Everyday Practical Electronics magazine which could be of interest. One being an oscilloscope project which would also make a fairly fast, by QL standards, (upto 40Khz) analogue to digital converter, it also has two channels. So could be used for stereo audio applications. This project is published in two parts, in the August and September 2007 issues of the magazine. The other project of use is the Serial I/O Controller project in the Jan 2008 issue of said magazine. In both cases these projects use the standard RS232 interface to the computer.

I have chosen the Serial I/O Controller project first since it is the simplest of the two projects. I will cover the oscilloscope in a future article. The other reason for going with the Serial I/O Controller is it will work with all QL systems, since it communicates at a low baud rate of 2400 Baud. The oscilloscope project needs higher than 9600 Baud rate which not all QL system can support. The original article is not totally correct in saying it will work at 9600 Baud rate, more on that when I cover that project.

If you can solder then you can make this project. The PCB and the components are easily available. The PCB from Everyday Practical Electronics(1) publishers and most of the components from Farnell(2) for example, but there are other places such as RS, Rapid etc. If you do not have the ability to program a PIC (Microcontroller chip), then the PIC needs to be programmed, you will need a PIC programmer for this. The cost of the board project is about £50.00 this will vary depending were you purchase, so please do not hold me to this, it is just a guide. You will also need to add some form of power supply, the board needs between 9-15VDC with 100mA capability should be more than enough, my test board takes 25mA at idle, with all the LED's lit and relays closed it rises to about 50mA, so one of these wall plugin low voltage type units would be good and safe. There is a safety diode on the supply input, so if you do connect to the power input incorrectly you will not do any damage. Also you will need a 9 way D Type serial cable, neither of these items is included in my guide price for the project. You do not need many tools either, side cutters to cut the component leads, small pliers to bend component leads, soldering iron 15W, solder, good light, and magnifying glass so you can check all your soldered joints and also check for any shorts across PCB tracks. Also a small screw driver. A cheap digital multimeter, some are as low as £5 would also be useful. For example build the board, but do not insert the PIC, or other IC's except the input voltage regulator (REG1), power up the card, the LED (Light Emitting Diode) LED4 should light up, if not disconnect the power and check for shorts and the polarity of the LED. With the multimeter you can check the voltage output from REG1 is 5V. If the LED is lit and you have 5V things are looking very good. The instructions in the original EPE article are fairly good, I built mine from the instructions to make sure they were accurate for this article, and did not find any problems and my board worked first time. However I must come clean and say I am an electronics engineer by training so I do have an advantage, but I tried to look at this as if an inexperienced constructor. They even give you the resistor colour codes to follow. The most important diagram is the one on page 18 which gives you all the component positions and orientations, components you need to watch for in this regard are some capacitors, diodes, transistors, LED's (note the short leg on an LED's is K(Cathode), and the IC's(Chips), watch for the notches or dot's, the pin next to a dot is pin 1, which will be the same end as a notch. You will also need some 24SWG tinned copper wire

for the wire links on the board of which there are 34. I strongly advise that you use IC sockets, there are 4 required, 1 40 pin and 3 16 pin. Again watch the orientation, there is notch at one end of these and this should line up with the IC(chip) notch, they are clearly shown in the EPE diagram. See my picture below this may help. Please note I have put an LED in place of the Buzzer for testing purposes, the Buzzer was driving me round the bend while testing. Also, I have at the stage this picture was taken not fitted the relays either. As you can see the board does work, this picture was taken with the board powered and being driven from my test program.

CAUTION THIS BOARD IS NOT MAINS RATED, SO DO NOT CONNECT ANY MAINS VOLTAGES DIRECTLY TO THE BOARD, EVEN THE RELAYS. MAINS CAN AND DOES KILL. YOU HAVE BEEN WARNED.



All the required PIC software and PC Visual Basic based software is available free of charge from the EPE web site. Also the source code is included so you see how the project work. Please note you may need the SerialOCX software also downloadable from the EPE web site, for the Visual Basic applications to work on your PC. This is due to the way some of Microsoft systems work, in some cases access is needed to be given to the serial hardware within the PC. If you are not going to use the Visual Basic software then it is not required.

With the QL hardware, there have been over the years a number of options for getting external signals in and out of the QL. The Minerva, I think was the best, with the I2C serial communication capability. These were available from Tony Firshman of TF Services(3) sadly no longer trading, I still have and use mine. Producing extra inputs or outputs just means adding the appropriate chip (IC). In fact up 256 I/O chips could be used on one serial line. Each chip having it's own address, selectable by setting pins high or low on the chips themselves. These I/O chips come in various types. The most useful to us are the parallel digital(PCF8574) and analogue (PCF8591) chips. TF Services did supply pre built units in there own cases and DIP switches for setting the address. The only disadvantage is you cannot run I2C over long distances, a metre or so is the limit. But we lost the ability to use I2C with emulator based systems. However all is not lost. The Serial I/O Controller gives us a 10 bit parallel input and 10 bit parallel output, 4 A/D(analogue to digital) converters, of which one is connected to an LDR (Light Dependant Resistor), the second to a LM335Z temperature sensor. The third input can measure voltages from 0 to 25 volts DC and the fourth measures voltages from 0 to 5 volts DC. However these can be changed to take other inputs with simple additions and minor hardware changes, like, for example, a pressure sensor. So one has the makings of a weather station.

If you need more capability than one board can supply, then you can use more than one board, it just depends how many RS232 serial ports you have on your system. A lot of PC's these days do not have any RS232 ports, however you can get fairly cheaply USB to RS232 converters, there are also parallel ones available as well if you only require 8 parallel outputs. QPC for example can support up to 8 serial ports. So you could have a system with up to 80 parallel input lines, 80 parallel output lines, 16 relays, 8 light sensors and 8 temperature sensors and 16 analogue inputs using 8 Serial I/O Controller boards. Not as many as a I2C based system but I think this would satisfy most people. Also the oscilloscope project that can be used on another RS232 port at the same time as the Serial I/O Card, the limitation being the 8 RS232 ports.

On the Serial I/O Controller there is also an on board buzzer and two relays. With the temperature sensor and the relays we have the making of a thermostat type control system. Using the QL's internal clock we can combine this with a time schedule. We can also log temperature and light information against time and create a log to analyse later. This is using the board as designed, no changes required to the published hardware.

In the original EPE article details are given as to the serial protocol to drive the board. With this and the listing below you should be able to get things going quickly. The list gives you procedures for accessing all the features of the board, I/O, temperature, light (LDR), relays, buzzer and the A/D converters.

```
100 REMark saved as win5_SerialCon_test
110 REMark Serial IO Controller test program V1, by Ian Burkinshaw 6 Jan 2008
120 BAUD 2400
130 OPEN#5;ser1ir
140 OVER 0:CLS:CSIZE 1,3
150 FOR c%=0 TO 1023
160 read_ldr
170 read_temp
180 read_ana1
190 read_ana2
200 read_10bit
210 set_output c%
220 display_data
230 drive_relay 1,0
240 drive_relay 2,0
250 drive_relay 1,1
260 drive_relay 2,1
270 drive_buzzer 0
280 drive_buzzer 1
290 NEXT c%
300 CLOSE#5:CSIZE 0,0:STOP
900 REMark ************************************************
1000 DEFine PROCedure set_output(cc%)
1010 c1%=0:c2%=0
1020 a$="a"&CHR$(8)
1030 PRINT#5;a$;
1040 REMark delay before sending next command
1050 PAUSE 2
1060 a$="a"&CHR$(cc%)
1070 PRINT#5;a$;
1080 REMark delay before sending next command
1090 PAUSE 1
1100 REMark Routine for bits 9 and 10
1110 IF cc%>=256 THEN c1%=1
1120 IF cc%>=512 AND cc%<=767 THEN c1%=0
1130 IF cc%>=512 THEN c2%=1
1140 a$="a"&CHR$(c1%)&CHR$(c2%)
1150 PRINT#5;a$;
1160 END DEFine set_output
1170 REMark ************************************************
1180 DEFine PROCedure read_temp
1190 a$="a"&CHR$(1)
1200 PRINT#5;a$;
1210 temp$=""
1220 read_data
1230 temp$=a1$
1240 END DEFine read_temp
1250 REMark ************************************************
1260 DEFine PROCedure read_ldr
1270 a$="a"&CHR$(2)
1280 PRINT#5;a$;
1290 ldr$=""
```

```
1300 read_data
1310 ldr$=a1$
1320 END DEFine read_ldr
1330 REMark ************************************************
1340 DEFine PROCedure read_ana1
1350 a$="a"&CHR$(4)
1360 PRINT#5;a$;
1370 ana1$=""
1380 read_data
1390 ana1$=a1$
1400 END DEFine read_ana1
1410 REMark ************************************************
1420 DEFine PROCedure read_ana2
1430 a$="a"&CHR$(3)
1440 PRINT#5;a$;
1450 ana2$=""
1460 read_data
1470 ana2$=a1$
1480 END DEFine read_ana2
1490 REMark ************************************************
1500 DEFine PROCedure read_10bit
1510 a$="a"&CHR$(7)
1520 PRINT#5;a$;
1530 bit$=""
1540 read_data
1550 bit$=a1$
1560 END DEFine read_10bit
1570 REMark ************************************************
1580 DEFine PROCedure read_data
1590 REMark first synchronise to synchronisation character 'a', also will keep running loop until
character appears, so no PAUSE command required wait for the PIC to do it's job.
1600 REPeat loop1
1610 a$=INKEY$(#5)
1620 IF a$="a" THEN EXIT loop1
1630 END REPeat loop1
1640 a1$=""
1650 REMark extract data until termination character '@'
1660 REPeat loop2
1670 a$=INKEY$(#5)
1680 IF a$="@" THEN EXIT loop2
1690 a1$=a1$&a$
1700 END REPeat loop2
1710 END DEFine read_data
1720 REMark ************************************************
1730 DEFine PROCedure display_data
1740 AT 0,0:PRINT "LDR Data : ";ldr$;"    "
1750 AT 1,0:PRINT "Temp Data : ";temp$;"    "
1760 REMark Next line displays the data and voltage conversion for analogue input 1, reads 0 to 5
VDC
1770 AT 2,0:PRINT "Analogue Data 1 : ";ana1$;"    ";ana1$/204.6;" Volts                "
1780 REMark Next line displays the data and voltage conversion for analogue input 2, reads 0 to 25
VDC
```

```
1790 AT 3,0:PRINT "Analogue Data 2 : ";ana2$;"   ";ana2$/40.92;" Volts              "
1800 AT 4,0:PRINT "10 Bit Input Data : ";bit$;"   "
1810 AT 5,0:PRINT "10 Bit Output Data : ";c%;"   "
1820 END DEFine display_data
1830 REMark ***********************************************
1840 DEFine PROCedure drive_relay(rn,nf)
1850 a$="a"&CHR$(5)
1860 PRINT#5;a$;
1870 PAUSE 2
1880 a$="a"&CHR$(rn)&CHR$(nf)
1890 PRINT#5;a$;
1900 END DEFine drive_relay
1910 DEFine PROCedure drive_buzzer (nf)
1920 a$="a"&CHR$(6)
1930 PRINT#5;a$;
1940 PAUSE 2
1950 a$="a"&CHR$(nf)
1960 PRINT#5;a$;
1970 END DEFine drive_buzzer
```

You will see from the listing that there are PAUSE statements, these are very important. Since after sending a command to the board the PIC has to process the command before it can read the next command or command data, so you have to give the board time to do this. The PAUSE command works in frames per second, in our case here in Europe there 50 frames per second. Or put another way PAUSE 1 equals 40ms, the processing time for most commands is about 50 ms, so PAUSE 2 is about right. It does not matter if the delay is longer, only if it is too short problems occur. The other issue to take into account is that RS232 communication supports only 8 bits at a time, not ten. So for example setting the parallel outputs requires you to send one word, to represent the first 8 bits then two further words that are either 1 or 0 as required to represent bits 9 and 10. Look at the procedure SET_OUTPUT, variable c1% is bit 9 and c2% is bit 10. The IF statements then sort out when they should be set, depending on the input variable cc%. To use this procedure, 'SET_OUTPUT [variable]', the variable can be between 0 and 1023. Note there is no error trapping if you exceed this value. If you run the program as is you should see the Serial IO Controllers output LED's count in a binary fashion. Lines 100 to 290 with procedure 'DISPLAY_DATA' are just to test and show basic operation for testing and as an example. Reading data back from the board, the procedure 'READ_DATA' looks for the synchronising character after the required delay from sending a command to then return data, the character which is 'a' ASCI code 97. Then the data value itself is sent. The data stream is terminated with the character "' ASCI code 64.

Hope that gives you some ideas and that if you are brave enough to try it, that you have fun. It may make a good school project as well. May be you could write an article for QL Today about how you use this project. Now let me get the oscilloscope card going.

Sources and References

I have no connection with any of the organizations below. They are just what I have used in the preparation of this article.

(1) "Everyday Practical Electronics" magazine http://www.epemag.co.uk
(2) "Farnell" source of electronic components http://www.farnell.com
(3) "TF Services" source of Minerva and I2C products. http://www.firshman.co.uk