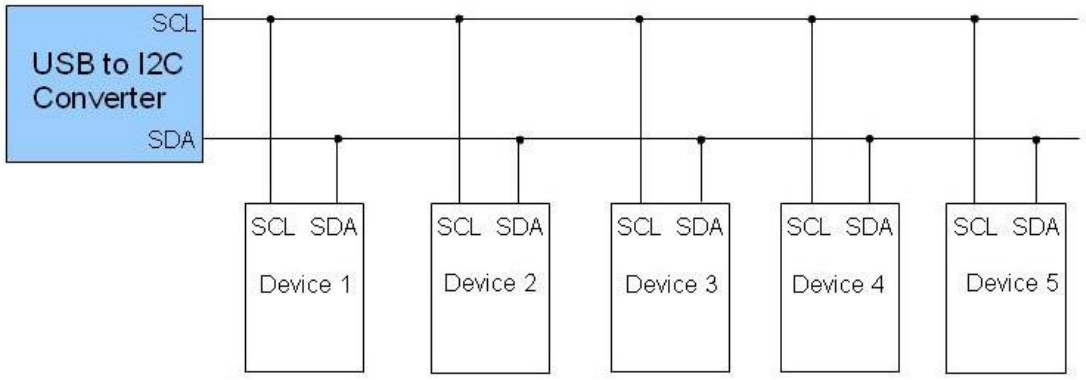
I2C Interface for QL Emulators Part 2

Originally published in QL Today Vol 16, Issue 1, Sept-Nov 2011

In part one of this series we looked at some of the basics of using the I2C bus and the ByVac BV4221 USB to I2C converter. Also I covered the first I2C interface using the PCF8574 parallel device. This time we will look at using the PCF8591 analogue to digital (DA) and digital analogue converter device. The DS1307 RTC (Real Time Clock). The PCF8570 256 x 8 RAM and the DS1803 Dual digital potentiometer we will look at another time.

Last time we really only looked at the I2C in a very simple way, the BV4221 converter driving one device, the parallel (PCF8574) device. However you can have up to 254 devices connected to the I2C bus. So my test/experimental board has one of each of the devices covered in the series of articles. The diagram below shows how this works.

Any number up to 254 devices can be connected to an I2C bus. There being only three connections required. Ground (GND), SDA (Serial Data) and SCL (Serial Clock). An example diagram is shown below to show the connections for multiple devices.

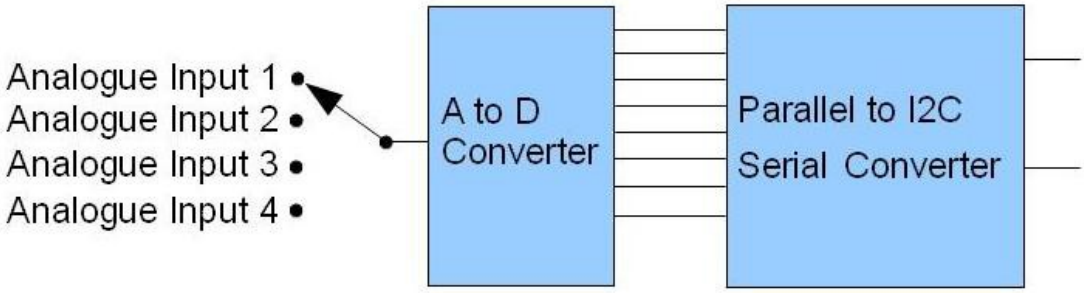


Now what I have not shown is the fact that ‘pull up’ resistors or termination resistors if you prefer, are required on the SDA and SCL lines. The reason I have not shown these is they are not required when using the BV4221 converter, since these resistors are already fitted within the converter. The reason for the resistors in the first place is the SCL and SDA are open drain sources. So needs a resistor from the line to VCC power line to complete the circuit. There are reasons that this was done with the I2C protocol which go beyond the scope of these articles. This issue is covered in the I2C book featured below.

So the above diagram shows, that you just connect the SCL and SDA lines to all the devices you wish to connect. So you will see that from now on I will not be showing the BV4221 converter in the following circuit diagrams. Just refer back to part one of this serial of articles to see how the BV4221 to any device(s) is connected.

Now we will look at the A/D (Analogue to Digital) and D/A (Digital to Analogue) converter device PCF8591.

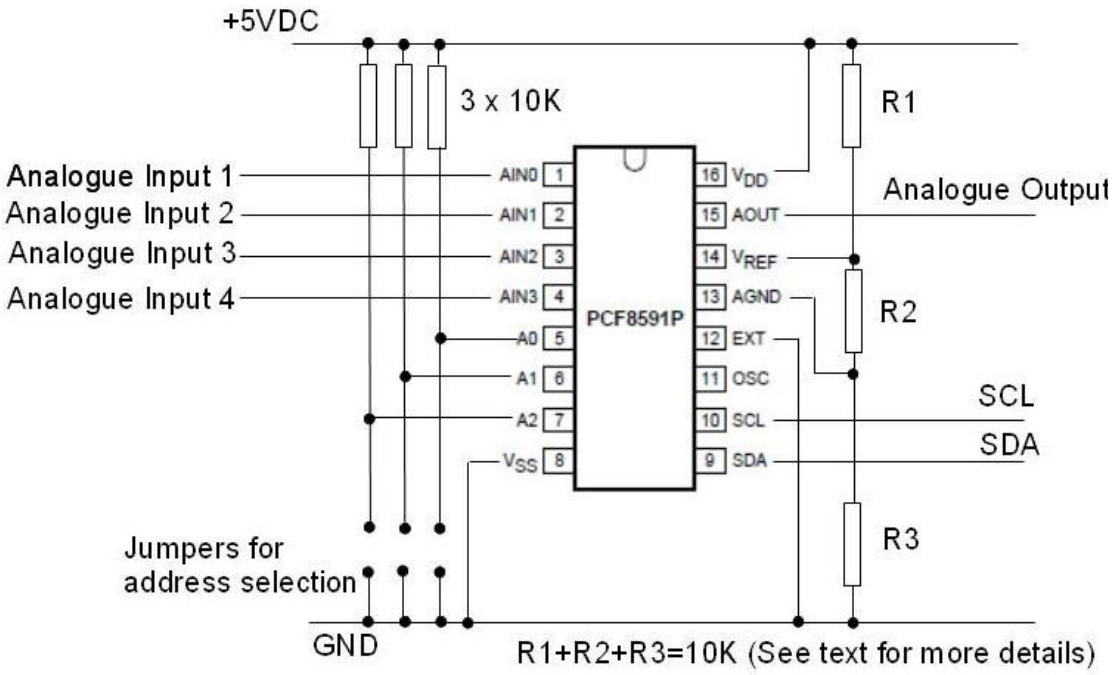
The PCF8591 has a single A(nalogue) to D(igital) converter. But as you see it has 4 analogue inputs. There is a 4 x 1 switch on the input of the device which you can select which input you wish to use via the I2C bus protocol for this device. So for example you could have 4 temperature sensors. One connected to each input. Then just select each device one by one, each time you select a sensor take a measurement and then move on to the next sensor and again take a measurement. Working though all the sensors connected and going back the first one and start the process all over again.



Simplified Block Diagram of the A/D side of the PCF8591

The converter is not that fast. The maximum I2C data bus speed is 100kHz, but this is controlled by the master device in our case the QL emulator. So several factors have to be taken into account, for example the baud rate selected from the QL. If you are running a baud rate of 9600 then clearly you cannot drive the I2C bus at 100kHz. Also the number of devices that you are communicating with on the I2C bus in a given application will limit the bandwidth available to the A/D converters. So the sort of applications for this device have to be non time critical. So fine for measuring temperature or voltage several times per second. However not fast enough to record audio signals for example, were you need to measure at a fairly high and consistent rate, remember CD's are recorded at 44.1 kHz and 16 bits for example. Also not the resolution of this device is 8 bits so that could be a limiting factor for this sort of application as well.

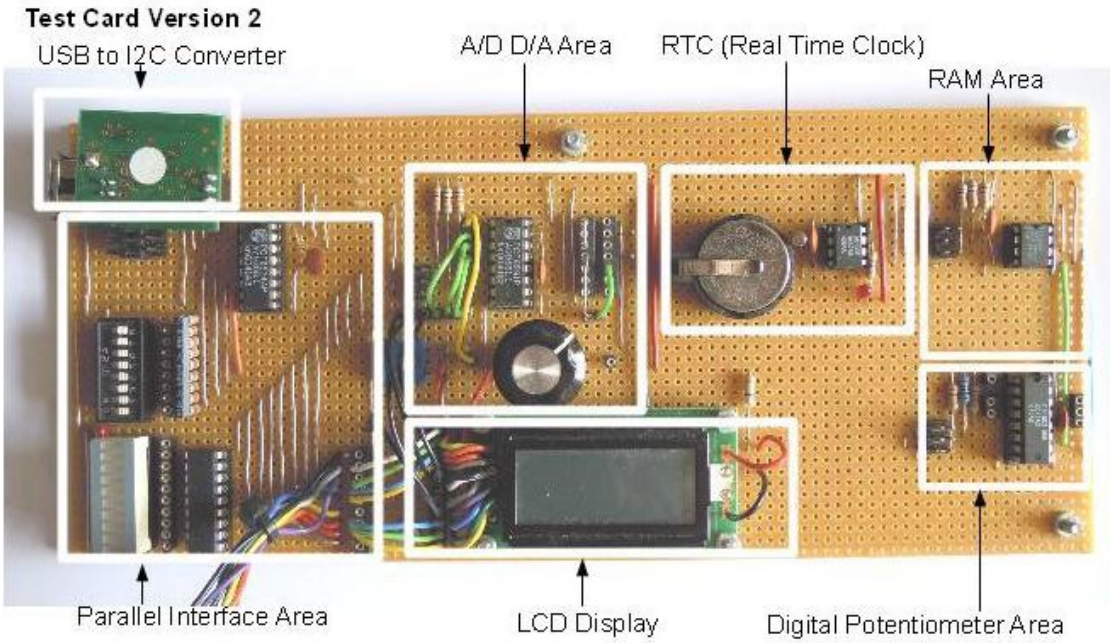
I should point out that Simon Goodwin did produce some software that did deal with audio in a very limited way. Not Hi Fi that is for sure. However this was using the Minerva ROM solution, which will work faster than using the USB to I2C converter. You also have to take into account we are working with an emulator, so we do have to take the PC hardware/operating system into account. It does get very complicated at this level from a timing point of view for this type of application. Which is why I have not explored it.



Above you will find the complete circuit of the A/D and D/A converter.

Another detail to take into account is the voltage range of the input is adjustable. Using resistors R1, R2 and R3. These act as a potential divider to set the maximum and minimum voltages. Please note the maximum input voltage can be no higher than 5 volts. Also note the combined resistance of all three resistors should be 10K ohm.

To make things easy to start with, if you connect pin 14 (Vref) to the 5V rail and connect pin 13 to GND and omit R1, R2 and R3. This sets the converter up to work the range 0V to +5V. Then connect a 10K linear potentiometer, the top connection to the 5V rail, the bottom to GND and the slider to one of the inputs (pins 1 to 4). This will provide you an analogue input to experiment with. In the photo of my test board you will see a control knob. That is the potentiometer I used for testing.



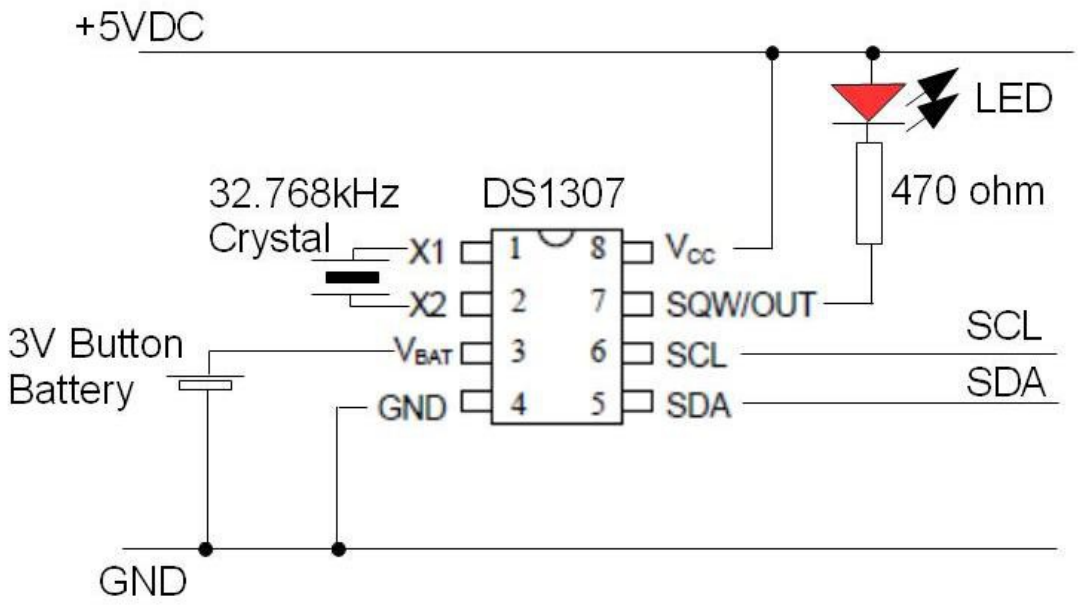
The software for the A/D and D/A device

```
10 CLS
20 BAUD 115200
30 ram=174:REMark PCF8570 address, all address links open.
40 parallel1=126:REMark PCF8574A address, all address links open
50 parallel2=78:REMark PCF8574 address, all links open
60 adda=158:REMark PCF8591 address, all address links open
70 rtc=208:REMark DS1307 real time clock, one fixed address with this device.
80 digpot=94:REMark DS1803 Digital Poteniometer, all link open
90 OPEN#3;ser2ir:REMark i=ignor hardware handshake, r=raw data
100 PRINT#3;CHR$(13);:REMark Carriage Return to set the baud rate in the USB to
I2C converter, required on first pass to initialise USB to I2C converter.
110 print_reply
120 PRINT
130 PRINT#3;"V";CHR$(13);:REMark Command to USB to I2C coverter for
firmware version.
140 PRINT "Return USB Converter Version Number:-";
150 print_reply:REMark Prints version number reply from USB to I2C converter
160 PRINT#3;"D";CHR$(13);:REMark Sets USB to I2C coverter to receive decimal
numbers, default is hex numbers.
170 PRINT "Decimal Mode Selected"
180 print_reply:REMark returns a device address.
190 DAC
195 AT 10,10:PRINT "        "
200 ADC
300 AT 10,10:PRINT "End         ":CLOSE#3:STOP
310 :
1000 DEFine PROCedure print_reply
1005 c$=""
1010 REPeat loop
1020 a$=INKEY$(#3)
```

```
1030 b$=a$
1040 PRINT b$;
1050 IF a$=">" THEN EXIT loop
1060 END REPeat loop
1070 END DEFine print_reply
1080 :
1090 :
1100 DEFine PROCedure non_print_reply
1110 REPeat loop
1120 a$=INKEY$(#3)
1140 IF a$=">" THEN EXIT loop
1150 END REPeat loop
1160 END DEFine non_print_reply
1170 :
1200 DEFine PROCedure extract_read_data
1205 c$=""
1210 REPeat loop
1220 a$=INKEY$(#3)
1230 b$=a$
1235 c$=c$&b$
1250 IF a$=CHR$(32) THEN slice:EXIT loop
1260 END REPeat loop
1290 END DEFine extract_read_data
1300 :
1310 DEFine PROCedure slice
1320 l=LEN(c$)
1330 IF l<6 THEN GO TO 1350
1340 c$=c$(6 TO l)
1350 END DEFine slice
1360 :
2000 DEFine PROCedure DAC
2010 adcontrol=64:REMark Sets PCF8591 to analogue output enabled, four single
ended inputs, autoincrement 'Off' and AD channel=0
2020 FOR a=0 TO 255
2030 PRINT#3;"s-";adda;" ";adcontrol;" ";a;" p";CHR$(13);
2040 AT 10,10:print_reply:PRINT a
2050 PAUSE 5
2060 NEXT a
2070 END DEFine DAC
2080 :
3000 DEFine PROCedure ADC
3010 adcontrol=64:REMark Sets PCF8591 to analogue output enabled, four single
ended inputs, autoincrement 'Off' and AD channel=0
3020 REPeat input_loop
3030 PRINT#3;"s-";adda+1;" g-1 p";CHR$(13);
3040 AT 10,10:extract_read_data:PRINT "     ":AT 10,10:PRINT c$;
3045 REMark FOR b= 1 TO 200000:NEXT a
3050 IF INKEY$=" " THEN EXIT input_loop
3060 END REPeat input_loop
3070 END DEFine ADC
```

The above listing just tests the A/D input and D/A output. To check the output connect a volt meter set to read 5V or above, to pin 15 (AOUT) and GND, when you run the code you will see the output voltage rise. Once the D/A test is finished, then it will test the input, adjust the potentiometer and you should the number displayed on your QPC screen change between 0 and 255.. This is the device working at it's simplest.

Now we will take a brief look at the RTC (Real Time Clock)



As you will see from the above diagram this is very simple. The first thing to note is that there are no address links, this device has only one address, so only one of these devices can be used on a single bus.

You will also note there is a button battery. This keeps the device running when the power it off, so keeps time when not in use. The LED is optional when fully powered up, this blinks once per second.

The software for the RTC.

```
10 REMark RTC (Real Time Clock) DS1307 test routines
20 init
30 set_rtc
40 OPEN#3;ser2ir:REMark i=ignor hardware handshake, r=raw data
50 PRINT#3;CHR$(13);:REMark Carriage Return to set the baud rate in the USB to
I2C converter, required on first pass to initialise USB to I2C converter.
60 print_reply:PRINT "Reply from sending CR."
70 PRINT
80 PRINT#3;"V";CHR$(13);:REMark Command to USB to I2C coverter for
firmware version.
90 PRINT "Return USB Converter Version Number:-";
100 extract_read_data:PRINT d$:print_reply:REMark Prints version number reply
from USB to I2C converter
```

```
110 PRINT
120 PRINT#3;"D";CHR$(13);:REMark Sets USB to I2C coverter to receive decimal
numbers, default is hex numbers.
130 PRINT "Decimal Mode Selected"
140 print_reply:REMark returns a device address.
150 PRINT
160 PRINT "Writing Clock Set Data"
170 write_rtc
180 PRINT "Writing RAM Area Data"
190 write_rtc_ram
200 PRINT "Both Clock Set Data and RAM Data read from device"
210 read_rtc
220 PRINT "From the first data byte the number should match between what was
writen to the device and what is read from the device."
230 REPeat loop_time
240 read_rtc_data
250 display_time_date
260 PAUSE 25
270 END REPeat loop_time
280 AT 10,10:PRINT "End         ":CLOSE#3:STOP
290 :
1000 DEFine PROCedure init
1010 CLS
1020 BAUD 115200
1030 ram=174:REMark PCF8570 address, all address links open.
1040 parallel1=126:REMark PCF8574A address, all address links open
1050 parallel2=78:REMark PCF8574 address, all links open
1060 adda=158:REMark PCF8591 address, all address links open
1070 rtc=208:REMark DS1307 real time clock, one fixed address with this device.
1080 digpot=94:REMark DS1803 Digital Potentiometer, all link open
1090 DIM tdata(7)
1100 DIM days$(7,3)
1110 RESTORE
1120 FOR a=1 TO 7
1130 READ d$
1140 days$(a)=d$
1150 NEXT a
1160 DATA "Mon","Tue","Wed","Thu","Fri","Sat","Sun"
1170 END DEFine init
1180 :
1190 DEFine PROCedure print_reply
1200 c$=""
1210 REPeat loop
1220 a$=INKEY$(#3)
1230 b$=a$
1240 c$=c$&b$
1250 PRINT b$;
1260 IF a$=">" THEN EXIT loop
1270 END REPeat loop
1280 END DEFine print_reply
```

```
1300 :
1310 DEFine PROCedure non_print_reply
1320 c$=""
1330 REPeat loop
1340 a$=INKEY$(#3)
1350 b$=a$
1360 c$=c$&b$
1370 IF a$=">" THEN EXIT loop
1380 END REPeat loop
1390 END DEFine non_print_reply
1400 :
1410 DEFine PROCedure extract_read_data
1420 d$=""
1430 REPeat data_loop
1440 a$=INKEY$(#3)
1450 b$=a$
1460 d$=d$&b$
1470 IF a$=CHR$(10) THEN EXIT data_loop
1480 END REPeat data_loop
1490 END DEFine extract_read_data
1500 :
1510 DEFine PROCedure set_rtc
1520 PRINT
1530 INPUT;"Year (Last two digits only):-";rtcyear$
1540 INPUT;"Month (number 01 to 12)    :-";rtcmonth$
1550 INPUT;"Date (01 to 31)            :-";rtcdate$
1560 INPUT;"Day (Number ie 1=Mon)      :-";rtcday$
1570 INPUT;"12/24 Hour Clock (Number)  :-";rtchours$
1580 IF rtchours$="12" THEN INPUT;"AM/PM     :-";rtcap$
1590 INPUT;"Hour (01 to 24)            :-";rtchour$
1600 INPUT;"Minutes (01 to 59)         :-";rtcmins$
1610 INPUT;"Seconds (01 to 59)         :-";rtcsecs$
1620 INPUT;"Pulse output enabled  (Y/N):-";rtcpulseen$
1630 INPUT;"Pulse Frequency (1=1Hz, 2=4.096kHz, 3=8.192kHz,
4=32.768kHz:-";rtcfreq
1640 PRINT
1650 drtcyear=rtcyear$(1):urtcyear=rtcyear$(2)
1660 rtcyear=(drtcyear*16)+urtcyear
1670 drtcmonth=rtcmonth$(1):urtcmonth=rtcmonth$(2)
1680 rtcmonth=(drtcmonth*16)+urtcmonth
1690 drtcdate=rtcdate$(1):urtcdate=rtcdate$(2)
1700 rtcdate=(drtcdate*16)+urtcdate
1710 rtcday=rtcday$
1720 IF rtchours$="12" THEN rtchourt=64
1730 IF rtchours$="24" THEN rtchourt=0
1740 IF rtchours$="12" AND (rtcap$=="AM" OR rtcap$=="PM") THEN rtcap=0
1750 drtchour=rtchour$(1):urtchour=rtchour$(2)
1760 rtchour=((drtchour*16)+rtchourt+rtcap)+urtchour
1770 drtcmins=rtcmins$(1):urtcmins=rtcmins$(2)
1780 rtcmins=(drtcmins*16)+urtcmins
```

```
1790 drtcsecs=rtcsecs$(1):urtcsecs=rtcsecs$(2)
1800 rtcsecs=(drtcsecs*16)+urtcsecs
1810 rtcpulseen=0
1820 IF rtcpulseen$=="Y":rtcpulseen=144
1830 rtcfreq=rtcfreq-1
1840 rtcpulse=rtcpulseen+rtcfreq
1850 PRINT "Setting details to be transmitted to device, this is in Hex form"
1860 PRINT "Year:-";rtcyear
1870 PRINT "Month:-";rtcmonth
1880 PRINT "Date:-";rtcdate
1890 PRINT "Day:-";rtcday;" ";days$(rtcday)
1900 PRINT "Hours:-";rtchour
1910 PRINT "Minutes:-";rtcmins
1920 PRINT "Seconds:-";rtcsecs
1930 PRINT "Pulse Output:-";rtcpulse
1940 END DEFine set_rtc
1950 :
2000 DEFine PROCedure write_rtc
2010 PRINT#3;"s-";rtc;" 0 ";rtcsecs;" ";rtcmins;" ";rtchour;" ";rtcday;" ";rtcdate;"
";rtcmonth;" ";rtcyear;" ";rtcpulse;" p";CHR$(13);:REMark the first number after the
s-ram is the starting word address, the remaining numbers are data to be loaded
incrementing the word address for each data item sent.
2020 non_print_reply
2030 END DEFine write_rtc
2040 :
2050 DEFine PROCedure read_rtc
2060 PRINT#3;"s-";rtc;" 0 p";CHR$(13);:REMark the first number after the s-174 is
the starting word address, when reading data this set the start word address.
2070 non_print_reply
2080 PRINT#3;"s-";rtc+1;" g-57 p";CHR$(13);:REMark g-9 means it will read 9 data
words in this example.
2090 extract_read_data:non_print_reply:PRINT d$
2100 END DEFine read_rtc
2110 :
2120 DEFine PROCedure write_rtc_ram
2130 PRINT "Writing Ram Data Only ";
2140 FOR ramd=8 TO 56
2150 dd=RND(0 TO 255):REMark Generating a random number to load into the
RAM
2160 PRINT#3;"s-";rtc;" ";ramd;" ";dd;" p";CHR$(13);:REMark the first number
after the s-ram is the starting word address, the remaining numbers are data to be
loaded incrementing the word address for each data item sent.
2170 non_print_reply
2180 PRINT dd;" ";
2190 NEXT ramd
2200 PRINT
2210 END DEFine write_rtc_ram
2220 :
3000 DEFine PROCedure read_rtc_data
3010 FOR a=0 TO 7
```

```
3020 PRINT#3;"s-";rtc;" ";a;" p";CHR$(13);:REMark the first number after the s-174 is the starting word address, when reading data this set the start word address.
3030 non_print_reply
3040 PRINT#3;"s-";rtc+1;" g-1 p";CHR$(13);:REMark g-9 means it will read 9 data words in this example.
3050 extract_read_data:non_print_reply
3060 d=d$
3070 d1=INT(d/16):d2=d-(16*d1)
3080 REMark PRINT dh$;" ";d;" ";d1;" ";d2
3090 tdata(a)=(d1*10)+d2
3100 REMark PRINT tdata(a)
3110 NEXT a
3120 END DEFine read_rtc_data
3130 :
3140 DEFine PROCedure display_tdata
3150 FOR a=0 TO 7
3160 PRINT tdata(a);" ";
3170 NEXT a
3180 PRINT
3190 END DEFine display_tdata
3200 :
3210 DEFine PROCedure display_time_date
3220 AT 42,10:PRINT tdata(2);":";tdata(1);":";tdata(0);"    "
3230 AT 43,10:PRINT days$(tdata(3));" ";tdata(4);"/";tdata(5);"/20";tdata(6)
3240 END DEFine display_time_date
```

This listing sets and reads the RTC.

That is it for this issue, until next time have fun.

For those who would like to delve deeper in to the I2C protocol and other devices that can be used, a new book has appeared. It is from the publishers of the Elektor magazine called Mastering the I2C Bus, by Vincent Himpe. It is not a cheap book at £29.50 plus £6.00 for post and packing, prices as at time of writing. But takes you though from the very basics of the I2C bus to quiet advanced projects. It is full of interesting ideas. Also there is a project to provide a USB to I2C converter, this offers an alternative to the ByVac product I have been using in this series. I have not, as yet tried this converter but plan to do so, I will report back in a future article. For more information and to order a copy go to www.elektor.com.

References

http://www.byvac.com/bv3/index.php?route=product/product&product_id=88
(Please note, I used the original V1 of the BV4221, ByVac now supply V2 which also has a SPI interface. The commands are the same, so the programs listed in the article should still work.)

http://www.byvac.com/bv3/index.php?route=product/category&path=44

PCF8570 Ram Data Sheet
http://www.nxp.com/documents/data_sheet/PCF8570.pdf

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

PCF8591 Data Sheet
http://www.nxp.com/documents/data_sheet/PCF8591.pdf

DS1307 RTC (Real Time Clock)
http://datasheets.maxim-ic.com/en/ds/DS1307.pdf

DS1803 Digital Potentiometer Data Sheet
http://datasheets.maxim-ic.com/en/ds/DS1803.pdf

I2C Tutorials
http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html
http://www.i2c.byvac.com/ar_foundation.php

TF Services I2C manual
http://www.dilwyn.me.uk/docs/manuals/index.html

Advanced I2C information, but still worth a read to understand I2C protocols
http://www.nxp.com/documents/user_manual/UM10204.pdf