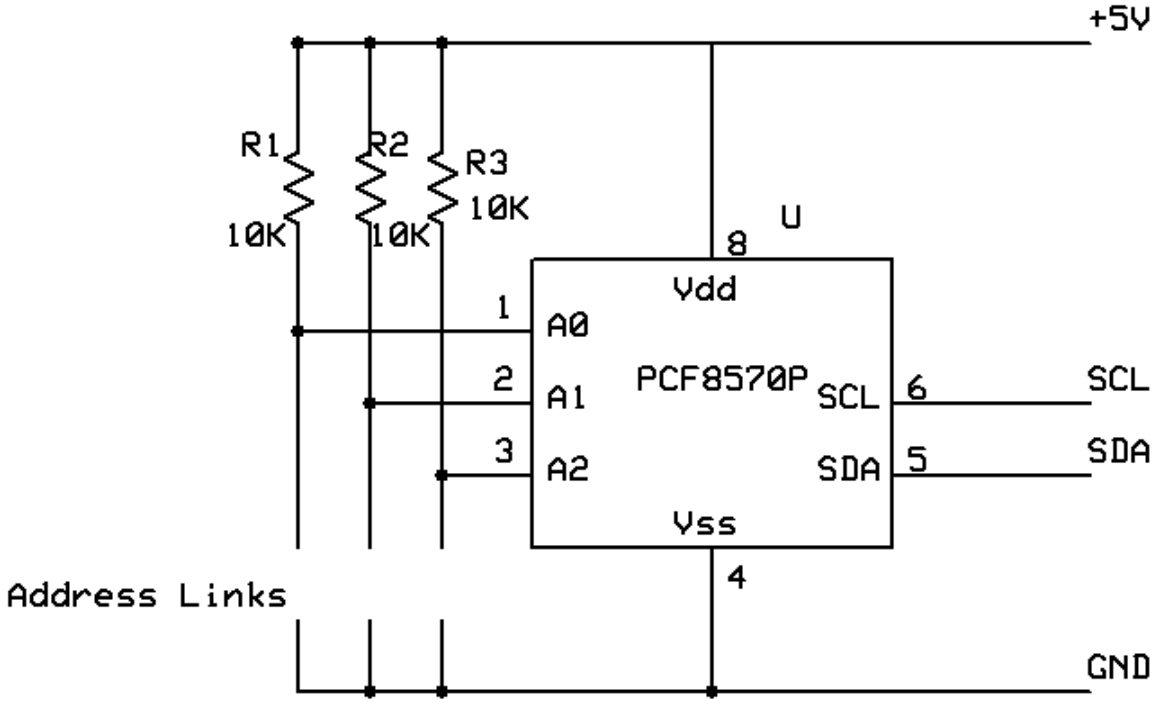
I2C Interface for QL Emulators Part 3

Originally Published in QL: Today Vol 16, Issue 2, Dec-Feb 2011/2012

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 in the first part the I2C interface using the PCF8574 parallel device. In part two we looked at using the PCF8591 analogue to digital (DA) and digital analogue converter device and the DS1307 RTC (Real Time Clock). This time we will look at the PCF8570 256 x 8 RAM and the DS1803 Dual digital potentiometer.

As can be seen from the circuit below, the PCF8570 RAM is very simple. Just power, GND and the SCL and SDA I2C bus lines not forgetting the address pull up resistors and links if required. As we have seen before address links are only required if more than one device of this type is required in your application. The PCF8570 is a 256 x 8 bit RAM (Random Access Memory), put another way it stores 256, 8 bit words. Now that may not seem much these days, but it can have it's uses. For example, you may wish to store some variables while being able to reset your QPC session, then load those variables back again. The PCF8570 is volatile memory, that means if you remove the power from the device it will forget every thing stored. However if the device is powered with a battery or an alternative power supply, when the QPC/QL is turned off then the device would retain all the data. The device has a low standby current of 15uA, so a battery could last for years. As you can see from the circuit diagram the PCF8570 has address links, please see part one of this series for the address ranges.



The short test example program below tests each of the 256 memory locations. The REM statements explain what is going on in the program.

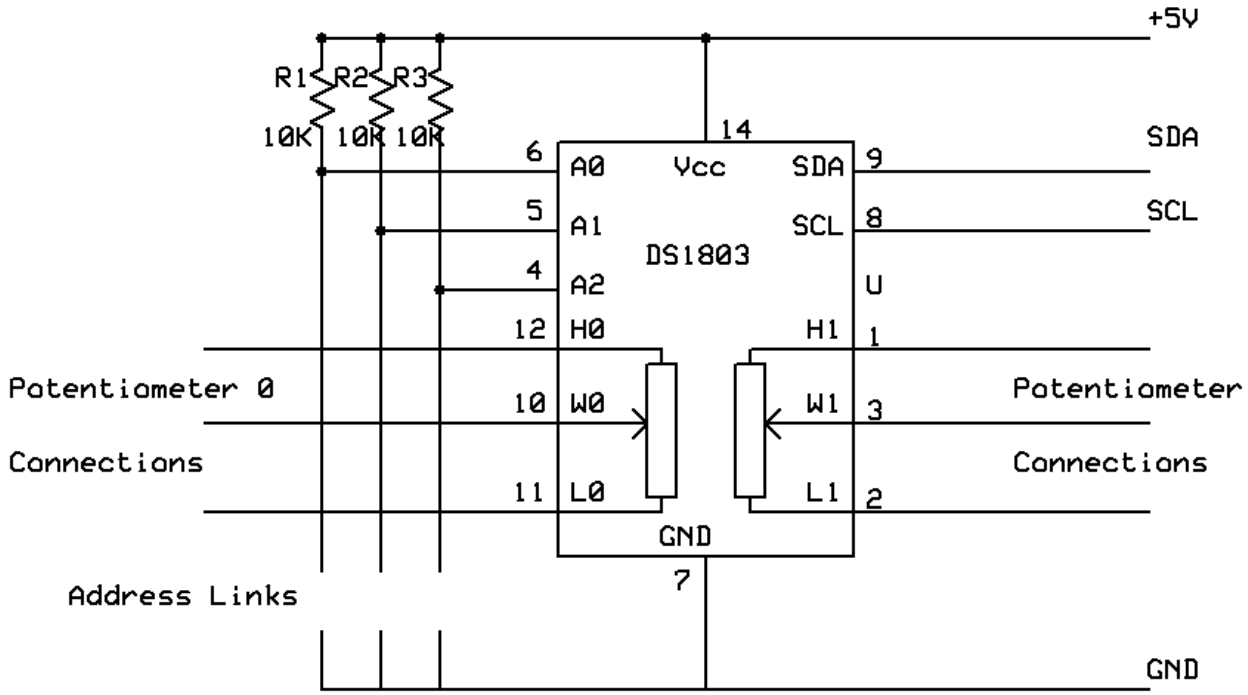
```
10 REMark RAM PCF8570P test routines
20 init
40 OPEN#3;ser2ir:REMark i=ignor hardware handshake, r=raw data
50 PRINT#3;CHR$(13);:REMark Carriage Return to set the baud rate in the USB to
I2C converter, required on first pass to initialise USB to I2C converter.
```

```
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$(13);:REMark Sets USB to I2C converter to receive decimal
numbers, default is hex numbers.
130 PRINT "Decimal Mode Selected"
140 print_reply:REMark returns a device address.
150 PRINT
180 PRINT "Writing RAM Area Data"
190 write_ram_data
200 PRINT "RAM Data read from device"
210 read_ram_data
220 PRINT "From the first data byte the number should match between what was
writen to the device and what is read from the device."
280 AT 10,10:PRINT "End         ":CLOSE#3:STOP
290 :
1000 DEFine PROCedure init
1010 CLS
1020 BAUD 115200
1030 ram=174:REMark PCF8570 address, all address links open.
1040 parallel1=126:REMark PCF8574A address, all address links open
1050 parallel2=78:REMark PCF8574 address, all links open
1060 adda=158:REMark PCF8591 address, all address links open
1070 rtc=208:REMark DS1307 real time clock, one fixed address with this device.
1080 digpot=94:REMark DS1803 Digital Potentiometer, all link open
1090 DIM tdata(7)
1100 DIM days$(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
2110 :
2120 DEFine PROCedure write_ram_data
2140 FOR ramd=0 TO 255
2150 dd=RND(0 TO 255):REMark Generating a random number to load into the RAM
2160 PRINT#3;"s-";ram;" ";ramd;" ";dd;" p";CHR$(13);:REMark the first number
after the s-ram is the starting word address, the remaining numbers are data to
be loaded incrementing the word address for each data item sent.
2170 non_print_reply
2180 PRINT dd;" ";
2190 NEXT ramd
2200 PRINT
2210 END DEFine write_ram_data
2220 :
3000 DEFine PROCedure read_ram_data
3010 FOR a=0 TO 255
3020 PRINT#3;"s-";ram;" ";a;" p";CHR$(13);:REMark the first number after the s-
ram(174) is the starting word address, when reading data this set the start word
address.
3030 non_print_reply
3040 PRINT#3;"s-";ram+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 PRINT d;" ";
3110 NEXT a
3120 PRINT
3130 END DEFine read_ram_data
3140 :
```

The final device we will look at in the series is the DS1803 dual potentiometer. This is a very interesting device. It has two full independently controlled via I2C interface potentiometers. There are 3 versions of this device, DS1803-10 which the potentiometers have the value 10K ohms, DS1803-50 which is 50K ohms and DS1803-100 which is 100K ohms. End potentiometer can be incremented in 256 steps, each step increasing the resistance of the potentiometer by the same

amount, so is linear in operation. To make a logarithmic potentiometer then this is done in your software code, however to achieve this the number of control steps will reduce. For an audio application I was working on I found 26 steps worked quiet well. One thing you must ensure is that any voltage applied to the potentiometer connections must not exceed the range 0V to +5V from the device ground. There is not any electrical isolation between the control logic, I2C bus or the potentiometer elements with in the device hence the caution required. As can be seen from the circuit the DS1803 is very simple. Just power, GND and the SCL and SDA I2C bus lines not forgetting the address pull up resistors and links if required. As we have seen before address links are only required if more than one device of this type is required in your application. As you can see, from the circuit diagram the DS1803 has address links, please see part one of this series for the address ranges. The software listing assumes the links are open.




```
10 REMark Digital Potentiometer DS1803 test routines
20 init
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 converter to receive decimal
numbers, default is hex numbers.
130 PRINT "Decimal Mode Selected"
140 print_reply:REMark returns a device address.
150 PRINT
160 potval0=128
```

```
170 potval1=255
180 PRINT "Writing Pot 0 Data"
190 write_pot_0_data
200 PRINT "Reading Pot 0 Data"
210 read_pot_0_data
220 PRINT "Writing Pot 1 Data"
230 write_pot_1_data
240 PRINT "Reading Pot 1 Data"
250 read_pot_1_data
280 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 Poteniometer, 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
2000 :
2010 DEFine PROCedure write_pot_0_data
2020 PRINT#3;"s-";digpot;" ";169;" ";potval0;" p";CHR$(13);
2030 non_print_reply
2040 END DEFine write_pot_0_data
2050 :
2060 DEFine PROCedure write_pot_1_data
2070 PRINT#3;"s-";digpot;" ";170;" ";potval1;" p";CHR$(13);
2080 non_print_reply
2090 END DEFine write_pot_1_data
2100 :
3000 DEFine PROCedure read_pot_0_data
3010 PRINT#3;"s-";digpot+1;" g-1 p";CHR$(13);
3020 extract_read_data:non_print_reply
3030 d=d$
3040 PRINT d;" ";
3050 PRINT
3060 END DEFine read_pot_0_data
3070 :
3080 DEFine PROCedure read_pot_1_data
3090 PRINT#3;"s-";digpot+1;" g-2 p";CHR$(13);
3100 extract_read_data:non_print_reply
3110 d=d$
3120 PRINT d$;" First number is the value for pot 0 the sencond number is for pot
1"
3130 PRINT
3140 END DEFine read_pot_1_data
3150 :
```

That concludes our quick overview of some devices that can be controlled using the I2C bus and how to use the I2C bus. Next time we start to look at some applications, first being, driving LCD alpha numeric displays.

References

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

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