**Improving QL Emulator I/O, Oscilloscope**

Originally Published in QL Today Vol 14, Issue 4, June-August 2010

In my first article on improving the QL emulator I/O published in volume 12, issue 3 of QL Today I showed how you can get parallel inputs and outputs via the RS232 port on your PC. As promised in that article here is the second part dealing with the oscilloscope project. This, in the event took me longer than I expected, sorting out routines to carry out the FFT (Fast Fourier Transform). More on this later

The hardware for this project was originally published in the August and September 2007 issues of Everyday Practical Electronics magazine. The project has two PCB's one handles the analogue processing and contains the analogue to digital converters. There are two channels on this card so you have a two channel oscilloscope as a result. It is not impossible to change the entire project to deal with more channels. However you would need to have some PIC programming experience to achieve this. 8 channels should be possible. The basic hardware, with additional input cards can support this. The second card contains the memory and a PIC processor to take the streams of data produced by the analogue to digital converters in to a RS232 stream. There is a lot more data here to be handled, compared to the parallel I/O project. Because of the higher data rates involved, 57,600 baud is required. A PC fitted with a RS232 port or if your PC does not have one then you can use a USB to RS232 converter. However please note, most QL original hardware will not run at these speeds. The analogue to digital converters are able to handle signals up to 40KHz. Not very high by todays standards but for simple audio type projects it is good enough. The signal processing is 8 bit so is not CD quality. Also it is more a storage oscilloscope. I will explain. There is also a mode you can put the hardware into which will make the oscilloscope a 8 bit digital analyser, this aspect I will not be covering in this article, but is not difficult to achieve. As I have said before we are supposed to be tinkerers. It is not my aim to give you a total solution but give you ideas.

A storage oscilloscope does what is say on the tin, it stores a snap shot in time of a waveform for you to study in you own time. This also means it is not a real time device. In the case of modern digital oscilloscopes this is very much the case. It takes time for the signal processing to take place, and the computer, in this case a QL emulator, to display the results. Two point to make here, one is, it is not a real time device, that is what you see now did not happen now. Also what you get is also not continuous, it is a snap shot in time, over a fixed time span. So you cannot use this as a continuous waveform streaming device, this is a limitation of this hardware solution. Even higher data rates would be required to achieve this and I do not think even a QL emulator could keep up with this. Do not under estimate how much processing power is required for even relatively low frequencies that audio has (20KHz). Taking in to account you need to sample a signal at twice the original signal frequency. That is why an audio CD runs at 44.1KHz, it is just over what is required to give you a 20KHz bandwidth, which is the limit of human hearing. The main problem is you have to process on a continuous basis. Remember we have to acquire the data, process it and display (output) it. It all takes time. If you can solder then you can make this project. The PCB and the components are easily available. The PCB's from Everyday Practical Electronics(1) publishers and most of the components from Farnell(2) for example, but there are others places such as RS, Rapid etc. You will have to have the ability to program a PIC (Microcontroller chip), You can use the same power supply that you may have used for the serial controller card.

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 boards 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 resistor colour codes to follow. Do watch for the position and orientation of components. However there is one error. The part list: Note IC15 should be a 79L05 (not 78L05), the circuit is correct. Also some individual setup of PC's can prevent the V2 PC Scope from working with its serial control. This can be cured by installing the entire EPE Serial software from the download area of

the EPE web site at; ftp;//ftp.epemag.wimborne.co.uk/pub/PICS/SerialOCX/
Accessible via the Downloads section of their home page www.epemag.co.uk.
Download all the files into a temporary folder, make sure you have no other applications running (including virus checkers, email clients, Visual Basic ect and run the SETUP.exe and follow the prompts. This will install a copy of the OCX; and all its sub-components and correctly install and register them. The OCX software has been tested with Windows 98, 2000, ME and XP. You may only need to do this with the original EPE Visual Basic software.
The listing is not a final program but it does give you a basis to work from. It will display both channels and show you a spectrum display from channel one. At start up you can select simulate, enter "S" at the prompt, so you do not need the hardware connected to give you an idea of what can be achieved. If you do have the hardware the enter 'R' for Real, then you will be asked which serial port you are using. If you look at the my program listing it will give you a good idea how to control the card. But here is some additional information to help you.

Send "G" 'GO' tell the PIC to get ready to send
Receive "R" Wait for a'R' to be sent from PIC (Indicates Ready)
Send "B" Tell the PIC to send a block (256 samples, 8 block per channel, 16 in total)
Send "S" Tell the PIC to get ready to receive setting data (Gain, Bias and AC/DC coupling)
Receive "V" Valid Baud, could not get this to work on my own code or the original Visual Basic program. But is not important unless you wish to change the baud rate. Not a good idea anyway.

No LF and/or CR on "G","R","B" or "S" commands
Data returned dies have LF and CR terminators

You need to send some other data after the "S" start command. In the following order. AC/DC coupling & X10 attenuator this is a single number see below. The second character to be sent is for Gain 'Channel A' followed by Bias 'Channel A', Gain 'Channel B', Bias 'Channel B',Mode (See below) and then "G" to end the string.
The data for the gain and bias controls are from 0 to 255
The bias setting is an offset value so for example it can deal with positive and negative input values, so for example to display an AC signal such a sine wave it would be best to set the bias to 127 so the analogue to digital converter is works at it's mid range, so the AC single will swing either side of this value.
Gain settings can be anything you like from 0 to 255. But as a guide the following values will give you some known gain settings.

23=/10
48=/5
84=/2
127=X1
170=X2
213=X5
233=X10

The first piece of data after the "S" start command set the AC/DC coupling and the attenuator. This is all controlled as a 8 bit word in to a 8 bit output port in the hardware as follows:-

1=Channel A X10 attenuator (Bit 1)
2=Channel A AC/DC coupling (Bit 2)
4=Channel B AC/CD coupling (Bit 3)
8=Channel B X10 attenuator (Bit 4)

Just add together the numbers you require, so for example send '12', will set the AC/DC coupling to DC and the X10 attenuator to 'On' again for channel B.

The mode command near the end of the string is as follows:-

0=Oscilloscope Mode
1=Spectrum Mode
2=Digital Analyser Mode
Note: In this listing the spectrum mode is not used, but dies need to be sent.

Another aspect I have included in the listing is FFT (Fast Fourier Transform) so that you can have spectrum displays. This is more a library of routines. I claim no originality for these they are transcribed from the routines in the "The Scientist and Engineers guide"(4). This also goes into details of how Fast Fourier Transforms work so I will not repeat here. It is not perfect but does give a good idea what can be done in terms of signal processing and display on the QL platform.

**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) "The Scientist and Engineer's Guide to Digital Signal Processing", Chapter 12. http://www.dspguide.com/pdfbook.htm


**Listing**

```
10 init
12 IF sr$=="s" THEN load_arrays
20 IF sr$=="r" THEN start
30 IF sr$=="r" THEN start_data 6,127,50,127,50,0
50 main_loop
60 CLOSE#4
1000 DEFine PROCedure init
1010 OPEN#3;con
1020 COLOUR_PAL
1030 WINDOW#3;1024,768,0,0:INK#3;1:PAPER#3;0:CLS#3
1040 WINDOW#1;1024,768,0,0:INK#1;1:PAPER#1;2:CLS#1
1050 SCALE#3;1039,0,0
1060 DIM chand$(16,256)
1070 DIM chand(16,256)
1080 BANNER 1,(SCR_XLIM/2)-100,50,4,1,4,"Virtual Scope"
1090 INBANNER 1,20,100,1,4,1,4,"Simulated or Real (S or R):"
1100 IF in$<>"S" AND in$<>"s" AND in$<>"R" AND in$<>"r" THEN GO TO 1090
1110 sr$=in$
1120 IF sr$=="r" THEN INBANNER 1,2,150,1,4,1,4,"Which Serial Port is the V2Scope Hardware Connected (1 to 8):"
1130 IF sr$=="r" AND (in$<"1" OR in$>"8") THEN INBANNER 1,2,150,1,4,1,4,"Which Serial Port is the V2Scope Hardware Connected (1 to 8):":GO TO 1130
1140 IF sr$=="r" THEN SerPort%=in$
```

```
1150 IF sr$=="r" THEN init2
1160 END DEFine init
1170 REMark
*************************************************************************
1180 DEFine PROCedure load_arrays
1190 amp=120:freq=pass
1200 FOR xx=1 TO 1024
1210 y2=127+SIN(RAD(freq)*(xx))*amp
1220 y2$=y2
1230 IF xx>=1 AND xx<=256 THEN block1=1:block2=9:xx2=xx
1240 IF xx>=257 AND xx<=512 THEN block1=2:block2=10:xx2=xx-256
1250 IF xx>=513 AND xx<=768 THEN block1=3:block2=11:xx2=xx-512
1260 IF xx>=769 AND xx<=1024 THEN block1=4:block2=12:xx2=xx-768
1270 IF xx>=1025 AND xx<=1280 THEN block1=5:block2=13:xx2=xx-1024
1280 IF xx>=1281 AND xx<=1538 THEN block1=6:block2=14:xx2=xx-1280
1290 IF xx>=1539 AND xx<=1792 THEN block1=7:block2=15:xx2=xx-1538
1300 IF xx>=1793 AND xx<=2048 THEN block1=8:block2=16:xx2=xx-1792
1310 chand$(block1,xx2)=CHR$(y2$)
1320 chand$(block2,xx2)=CHR$(y2$)
1330 NEXT xx
1340 END DEFine load_arrays
1350 REMark
*************************************************************************
1360 DEFine PROCedure BANNER(Ch%,Xx%,Yy%,Sz%,Ink%,Pap%,f$)
1370  LOCal l%,X%,Y%,W%,H%
1380  l%=LEN(f$)
1390 SELect ON Sz%
1400  =1:W%=7*(l%+1):H%=19
1410  =2:W%=8*(l%+1):H%=19
1420  =3:W%=12*(l%+1):H%=19
1430  =4:W%=16*(l%+1):H%=19
1440  =5:W%=7*(l%+1):H%=30
1450  =6:W%=8*(l%+1):H%=30
1460  =7:W%=12*(l%+1):H%=30
1470  =8:W%=16*(l%+1):H%=30
1480 END SELect
1490  IF Xx%<0 THEN X%=(512-W%)/2:ELSE X%=Xx%:END IF
1500  IF Yy%<0 THEN Y%=(256-H%)/2:ELSE Y%=Yy%:END IF
1510  BLOCK#Ch%;W%,H%,X%+6,Y%+4,0
1520  BLOCK#Ch%;W%,H%,X%,Y%,0
1530  BANNER_TXT Ch%,Xx%,Yy%,Sz%,Ink%,Pap%,f$
1540 END DEFine BANNER
1550 REMark
*************************************************************************
************************************
1560 DEFine PROCedure BANNER_TXT(Ch%,Xx%,Yy%,Sz%,Ink%,Pap%,f$)
1570  LOCal l%,X%,Y%,W%,H%
1580  l%=LEN(f$)
1590 SELect ON Sz%
1600  =1:W%=7*(l%+1):H%=19:CSIZE#Ch%;0,0
1610  =2:W%=8*(l%+1):H%=19:CSIZE#Ch%;1,0
1620  =3:W%=12*(l%+1):H%=19:CSIZE#Ch%;2,0
```

```
1630  =4:W%=16*(l%+1):H%=19:CSIZE#Ch%;3,0
1640  =5:W%=7*(l%+1):H%=30:CSIZE#Ch%;0,1
1650  =6:W%=8*(l%+1):H%=30:CSIZE#Ch%;1,1
1660  =7:W%=12*(l%+1):H%=30:CSIZE#Ch%;2,1
1670  =8:W%=16*(l%+1):H%=30:CSIZE#Ch%;3,1
1680 END SELect
1690  IF Xx%<0 THEN X%=(512-W%)/2:ELSE X%=Xx%:END IF
1700  IF Yy%<0 THEN Y%=(256-H%)/2:ELSE Y%=Yy%:END IF
1710  BLOCK#Ch%;W%-4,H%-2,X%+2,Y%+1,Pap%
1720  OVER#Ch%;1:INK#Ch%;0
1730  CURSOR#Ch%;X%+5,Y%+4:PRINT#Ch%;f$;
1740  CURSOR#Ch%;X%+7,Y%+4:PRINT#Ch%;f$;
1750  CURSOR#Ch%;X%+5,Y%+6:PRINT#Ch%;f$;
1760  CURSOR#Ch%;X%+7,Y%+6:PRINT#Ch%;f$;
1770  INK#Ch%;Ink%
1780  CURSOR#Ch%;X%+6,Y%+5:PRINT#Ch%;f$;
1790 END DEFine BANNER_TXT
1800 REMark
**********************************************************************************
**************************************
1810 DEFine PROCedure INBANNER(Ch%,Xx%,Yy%,Ex%,Sz%,Ink%,Pap%,f$)
1820  LOCal l%,X%,Y%,W%,H%
1830  l%=LEN(f$)
1840 SELect ON Sz%
1850  =1:W%=7*(l%+1+Ex%):H%=19
1860  =2:W%=8*(l%+1+Ex%):H%=19
1870  =3:W%=12*(l%+1+Ex%):H%=19
1880  =4:W%=16*(l%+1+Ex%):H%=19
1890  =5:W%=7*(l%+1+Ex%):H%=30
1900  =6:W%=8*(l%+1+Ex%):H%=30
1910  =7:W%=12*(l%+1+Ex%):H%=30
1920  =8:W%=16*(l%+1+Ex%):H%=30
1930 END SELect
1940  IF Xx%<0 THEN X%=(512-W%)/2:ELSE X%=Xx%:END IF
1950  IF Yy%<0 THEN Y%=(256-H%)/2:ELSE Y%=Yy%:END IF
1960  BLOCK#Ch%;W%,H%,X%+6,Y%+4,0
1970  BLOCK#Ch%;W%,H%,X%,Y%,0
1980  INBANNER_TXT Ch%,Xx%,Yy%,Ex%,Sz%,Ink%,Pap%,f$
1990 END DEFine INBANNER
2000 REMark
**********************************************************************************
****************************************
2010 DEFine PROCedure INBANNER_TXT(Ch%,Xx%,Yy%,Ex%,Sz%,Ink%,Pap%,f$)
2020  LOCal l%,X%,Y%,W%,H%
2030  l%=LEN(f$)
2040 SELect ON Sz%
2050  =1:W%=7*(l%+1+Ex%):H%=19:CSIZE#Ch%;0,0
2060  =2:W%=8*(l%+1+Ex%):H%=19:CSIZE#Ch%;1,0
2070  =3:W%=12*(l%+1+Ex%):H%=19:CSIZE#Ch%;2,0
2080  =4:W%=16*(l%+1+Ex%):H%=19:CSIZE#Ch%;3,0
2090  =5:W%=7*(l%+1+Ex%):H%=30:CSIZE#Ch%;0,1
2100  =6:W%=8*(l%+1+Ex%):H%=30:CSIZE#Ch%;1,1
```

```
2110  =7:W%=12*(l%+1+Ex%):H%=30:CSIZE#Ch%;2,1
2120  =8:W%=16*(l%+1+Ex%):H%=30:CSIZE#Ch%;3,1
2130 END SELect
2140  IF Xx%<0 THEN X%=(512-W%)/2:ELSE X%=Xx%:END IF
2150  IF Yy%<0 THEN Y%=(256-H%)/2:ELSE Y%=Yy%:END IF
2160  BLOCK#Ch%;W%-4,H%-2,X%+2,Y%+1,Pap%
2170  OVER#Ch%;1:INK#Ch%;0
2180  CURSOR#Ch%;X%+5,Y%+4:PRINT#Ch%;f$;
2190  CURSOR#Ch%;X%+7,Y%+4:PRINT#Ch%;f$;
2200  CURSOR#Ch%;X%+5,Y%+6:PRINT#Ch%;f$;
2210  CURSOR#Ch%;X%+7,Y%+6:PRINT#Ch%;f$;
2220  INK#Ch%;Ink%
2230  CURSOR#Ch%;X%+6,Y%+5:PRINT#Ch%;f$;
2240 SELect ON Sz%
2250 =1:=5:CURSOR#Ch%;X%+6+(l%*6),Y%+4
2260 =2:=6:CURSOR#Ch%;X%+6+(l%*8),Y%+4
2270 =3:=7:CURSOR#Ch%;X%+6+(l%*12),Y%+4
2280 =4:=8:CURSOR#Ch%;X%+6+(l%*16),Y%+4
2290 END SELect
2300 INK#Ch%;1:INPUT#Ch%;in$
2310 END DEFine INBANNER_TXT
2320 REMark
*****************************************************************************
*********************************************
2330 DEFine PROCedure init2
2340 OPEN#3;con
2350 COLOUR_PAL
2360 WINDOW#3;1024,768,0,0:INK#3;1:PAPER#3;0:CLS#3
2370 WINDOW#1;1024,768,0,0:INK#3;1:PAPER#3;2:CLS#3
2380 SCALE#3;1039,0,0
2390 ws:PRINT#1
2400 BAUD SerPort%,57600
2410 SER_ROOM SerPort%,4500
2420 SER_BUFF SerPort%,5,4500
2430 END DEFine init2
2440 REMark
*****************************************************************
2450 DEFine PROCedure start
2460 SerPort$=SerPort%
2470 OPEN#4;"ser"&SerPort$&"i"
2480 END DEFine start
2490 REMark
*****************************************************************
2500 DEFine PROCedure start_data (in%,ga%,ba%,gb%,bb%,mod%)
2510 REMark in%=input channel+AC/DC selection; ga%=input gain a;ba%=bias a;gb%=input
gain b;bb%=bias a:mod%=spectrum/scope/digital
2520 PRINT#3;"Awaiting Reply"
2530 PRINT#4;"S";
2540 a$=INKEY$(#4)
2550 IF a$="" THEN GO TO 2540
2560 IF a$="S" THEN PRINT#4;CHR$(in%);CHR$(ga%);CHR$(ba%);CHR$(gb%);CHR$(bb
%);CHR$(mod%);"G";
```

```
2570 REPeat in_loop
2580 a$=INKEY$(#4)
2590 IF a$="R" THEN EXIT in_loop
2600 END REPeat in_loop
2610 END DEFine start_data
2620 REMark
*****************************************************************************
2630 DEFine PROCedure main_loop
2640 pass=1
2650 CLS#3
2660 REPeat loop
2670 SER_CLEAR SerPort%
2680 IF sr$=="r" THEN PRINT#4;"G";
2690 IF sr$=="r" THEN read_real_data
2700 AT#3;75,150:PRINT#3;"Pass ";pass;" complete"
2710 IF sr$=="s":AT#3;0,115:PRINT#3;"Press space bar to continue to next pass/frequency"
2720 pass=pass+1
2730 IF sr$=="s" THEN load_arrays
2740 REMark test_data
2750 BLOCK#3;1024,190,0,15,10
2760 BLOCK#3;1024,190,0,207,10
2770 BLOCK#3;1024,190,0,399,10
2780 channel_grid 2,1022,764
2790 channel_grid 2,760,504
2800 channel_grid 2,500,244
2810 AT#3;2,0:INK#3;1:PRINT#3;"Channel 1"
2820 AT#3;21,0:PRINT#3;"Channel 2"
2830 AT#3;40,0:PRINT#3;"FFT"
2840 scan_display_one
2850 scan_display_two
2860 spectrum_init
2870 The_FFT
2880 scan_fft
2890 IF sr$=="S" THEN PAUSE
2900 END REPeat loop
2910 END DEFine main_loop
2920 REMark
*****************************************************************************
2930 DEFine PROCedure channel_data
2940 FOR bloc=1 TO 16
2950 PRINT "Block Receiving:";bloc
2960 PRINT#4;"B";
2970 PAUSE 1
2980 INPUT#4,a$
2990 chand$(bloc)=a$
3000 NEXT bloc
3010 END DEFine channel_data
3020 REMark
*****************************************************************************
3030 DEFine PROCedure scan_display_one
3040 last=128:voffset=765
3050 FOR xp=1 TO 1024
```

```
3060 IF xp<=256 THEN bloc=1:xx=xp
3070 IF xp<=512 AND xp>256 THEN bloc=2:xx=xp-256
3080 IF xp<=768 AND xp>512 THEN bloc=3:xx=xp-512
3090 IF xp>768 THEN bloc=4:xx=xp-768
3100 REMark LINE#3;x,744 TO x,1000
3110 y1$=chand$(bloc,xx)
3120 y1=CODE(y1$)
3130 INK#3,3
3140 IF xp=1 THEN POINT#3,xp,y1+voffset
3150 IF xp>1 THEN LINE#3;xp-1,last TO xp,voffset+y1
3160 last=voffset+y1
3170 INK#3;1
3180 NEXT xp
3190 END DEFine scan_display_one
3200 REMark
*****************************************************************************
3210 DEFine PROCedure scan_display_two
3220 last=128:voffset=505
3230 FOR xp=1 TO 1024
3240 IF xp<=256 THEN bloc=9:x1=xp
3250 IF xp<=512 AND xp>256 THEN bloc=10:x1=xp-256
3260 IF xp<=768 AND xp>512 THEN bloc=11:x1=xp-512
3270 IF xp>768 THEN bloc=12:x1=xp-768
3280 REMark LINE#3;x,484 TO x,1000
3290 y1$=chand$(bloc,x1)
3300 y1=CODE(y1$)
3310 INK#3,6
3320 IF xp=1 THEN POINT#3,xp,y1+voffset
3330 IF xp>1 THEN LINE#3;xp-1,last TO xp,voffset+y1
3340 last=voffset+y1
3350 INK#3;1
3360 NEXT xp
3370 END DEFine scan_display_two
3380 REMark
*****************************************************************************
3390 DEFine PROCedure channel_grid(col%,top%,bottom%)
3400 INK#3;col%
3410 FOR a=0 TO 5
3420 LINE#3;0,bottom%+(a*51) TO 1024,bottom%+(a*51)
3430 NEXT a
3440 FOR a=0 TO 25
3450 LINE#3;a+(a*40),bottom% TO a+(a*40),top%
3460 NEXT a
3470 END DEFine channel_one_grid
3480 REMark
*****************************************************************************
3490 DEFine PROCedure channel_two_grid
3500 INK#3;2
3510 FOR a=0 TO 5
3520 LINE#3;0,484+(a*51) TO 1024,484+(a*51)
3530 NEXT a
3540 FOR a=0 TO 25
```

```
3550 LINE#3;a+(a*40),484 TO a+(a*40),740
3560 NEXT a
3570 END DEFine channel_two_grid
3580 REMark
*************************************************************************
3590
 DEFine PROCedure read_real_data
3600 FOR b=1 TO 16
3610 PRINT#4;"B";
3620 REMark PAUSE 3:rem sometimes depending on hardware, a dealey is required to give time for the PIC to respond
3630 INPUT#4;a$
3640 REMark PRINT#1;"B ";bloc1;" Lenght ";LEN(a$):REMark test line only
3650 IF b=1 THEN chand$(b)=a$(2 TO 257):REMark to lose the 'R' and CHR$ 10 out of the string)
3660 IF b>1 THEN chand$(b)=a$(1 TO 256):REMark to lose the CHR$ 10 out of the string
3670 NEXT b
3680 END DEFine read_real_data
3690 REMark
*************************************************************************
3700 DEFine PROCedure ws
3710 LOCal xp
3720 xp=xx+290
3730 REMark WINDOW xlim,ylim,0,0:PAPER 0:CLS
3740 REMark WINDOW#0,xlim,ylim-28,0,28:PAPER#0,0
3750 REMark WINDOW#2,xlim,ylim-28,0,28
3760 INK#1;1:OVER#1;1:CSIZE#1;2,3
3770 FOR n=1 TO 10
3780 CURSOR#1;n+50,10+n
3790 PRINT#1;"---V2Scope---"
3800 END FOR n
3810 CURSOR#1;n+50,10+n
3820 INK#1;0:PRINT#1;"---V2Scope---"
3830 CSIZE#1;0,0:INK#1;1
3840 OVER#1;0
3850 END DEFine ws
3860 REMark
*************************************************************************
3870 DEFine PROCedure spectrum_init
3880 N%=1024:REMark Number of samples-must be a "power of 2" value
3890 DIM IMX(N%)
3900 DIM REX(N%)
3910 DIM samples1(N%)
3920 DIM samples2(N%)
3930 load_sample_array
3940 FOR nn=1 TO 1024
3950 IMX(nn)=samples1(nn)
3960 NEXT nn
3970 END DEFine spectrum_init
3980 REMark
*************************************************************************
3990 DEFine PROCedure load_sample_array
```

```
4000 FOR nn=1 TO 1024
4010 IF nn<=256 THEN t1$=chand$(1,nn):t2$=chand$(9,nn)
4020 IF nn>256 AND nn<=512 THEN t1$=chand$(2,nn-256):t2$=chand$(10,nn-256)
4030 IF nn>512 AND nn<=768 THEN t1$=chand$(3,nn-512):t2$=chand$(11,nn-512)
4040 IF nn>768 AND nn<=1024 THEN t1$=chand$(4,nn-768):t2$=chand$(12,nn-768)
4050 IF nn>1024 AND nn<=1280 THEN t1$=chand$(5,nn-1024):t2$=chand$(13,nn-1024)
4060 IF nn>1280 AND nn<=1536 THEN t1$=chand$(6,nn-1280):t2$=chand$(14,nn-1280)
4070 IF nn>1536 AND nn<=1792 THEN t1$=chand$(7,nn-1536):t2$=chand$(15,nn-1536)
4080 IF nn>1792 AND nn<=2048 THEN t1$=chand$(8,nn-1792):t2$=chand$(16,nn-1792)
4090 t1=CODE(t1$)
4100 t2=CODE(t1$)
4110 samples1(nn-1)=t1
4120 samples2(nn-1)=t2
4130 NEXT nn
4140 END DEFine load_sample_array
4150 REMark
**************************************************************************
4160 DEFine PROCedure The_FFT
4170 REMark Upon entry, N% contains the number of points in the DFT, REX[] and
4180 REMark IMX[] contains the real and imaginary parts of the input. Upon return,
4190 REMark REX[] and IMX[] contain the DFT outputs. All signals run from 0 to N%-1.
4200 :
4210 REMark PI=3.14159265:rem Set constants
4220 NM1%=N%-1
4230 ND2%=N%/2
4240 M%=INT(LOG10(N%)/LOG10(2))
4250 J%=ND2%
4260 :
4270 FOR I%=1 TO N%-2:REMark Bit reversal sorting
4280 IF I%>=J% THEN GO TO 4350
4290 TR=REX(J%)
4300 TI=IMX(J%)
4310 REX(J%)=REX(I%)
4320 IMX(J%)=IMX(I%)
4330 REX(I%)=TR
4340 IMX(I%)=TI
4350 K%=ND2%
4360 IF K%>J% THEN GO TO 4400
4370 J%=J%-K%
4380 K%=K%/2
4390 GO TO 4360
4400 J%=J%+K%
4410 NEXT I%
4420 :
4430 FOR l%=1 TO M%:REMark Loop for each stage
4440 LE%=INT(2^l%)
4450 LE2%=LE%/2
4460 UR=1
4470 UI=0
4480 SR=COS(PI/LE2%):REMark Calculate sine & cosine values
4490 SI=SIN(PI/LE2%)
4500 FOR J%=1 TO LE2%:REMark Loop for each sub DFT
```

```
4510 JM1%=J%-1
4520 FOR I%=JM1% TO NM1% STEP LE%:REMark Loop for each butterfly
4530 IP%=I%+LE2%
4540 TR=REX(IP%)*UR-IMX(IP%)*UI:REMark Butterfly calculation
4550 TI=REX(IP%)*UI+IMX(IP%)*UR
4560 REX(IP%)=REX(I%)-TR
4570 IMX(IP%)=IMX(I%)-TI
4580 REX(I%)=REX(I%)+TR
4590 IMX(I%)=IMX(I%)+TI
4600 NEXT I%
4610 TR=UR
4620 UR=TR*SR-UI*SI
4630 UI=TR*SI+UI*SR
4640 NEXT J%
4650 NEXT l%
4660 :
4670 END DEFine The_FFT
4680 REMark
*****************************************************************************
4690 DEFine PROCedure IFFT
4700 REMark INVERSE FAST FOURIER TRANSFORMATION
4710 REMark Upon entry, N% contains the number of points in the IDFT, REX[] and
4720 REMark IMX[] contain real and imaginary parts of the complex frequency domain.
4730 REMark Upon return, REX[] andIMX[] contain the complex time domain
4740 REMark All signals run from 0 to N%-1.
4750 :
4760 FOR K%=0 TO N%-1:REMark Change the sign of IMX[]
4770 IMX(K%)=-IMX(K%)
4780 NEXT K%
4790 :
4800 The_FFT:REMark Calculate forward FFT
4810 :
4820 FOR I%=0 TO N%-1:REMark Divide the time domain by N% and change the sign of IMX[]
4830 REX(I%)=REX(I%)/N%
4840 IMX(I%)=-IMX(I%)/N%
4850 NEXT I%
4860 :
4870 END DEFine IFFT
4880 REMark
*****************************************************************************
4890 DEFine PROCedure FFT_Real
4900 REMark FFT FOR REAL SIGNALS
4910 REMark Upon entry, N% contains the number of points in the DFT, REX{} contains
4920 REMark the real input signal, while values in IMX[] are ignored. Upon return
4930 REMark REX[] and IMX[] contain DFT out. All signals run from 0 to N%-1.
4940 :
4950 NH%=N%/2-1:REMark Separate even and odd points
4960 FOR I%=0 TO NH%
4970 REX(I%)=REX(2*I%)
4980 IMX(I%)=REX(2*I%+1)
4990 NEXT I%
5000 :
```

```
5010 N%=N%/2:REMark Calculate N%/2 point FFT
5020 The_FFT
5030 N%=N%*2
5040 :
5050 NM1%=N%-1:REMark Even/odd frequency domain decomposition
5060 ND2%=N%/2
5070 N4%=N%/4-1
5080 FOR I%=1 TO N4%
5090 IM%=ND2%-I%
5100 IP2%=I%+ND2%
5110 IPM%=IM%+ND2%
5120 REX(IP2%)=(IMX(I%)+IMX(IM%))/2
5130 REX(IPM%)=REX(IP2%)
5140 IMX(IP2%)=-(REX(I%)-REX(IM%))/2
5150 IMX(IPM%)=-IMX(IP2%)
5160 REX(I%)=(REX(I%)+REX(IM%))/2
5170 REX(IM%)=REX(I%)
5180 IMX(I%)=(IMX(I%)-IMX(IM%))/2
5190 IMX(IM%)=-IMX(I%)
5200 NEXT I%
5210 REX(N%*3/4)=IMX(N%/4)
5220 REX(ND2%)=IMX(0)
5230 IMX(N%*3/4)=0
5240 IMX(ND2%)=0
5250 IMX(N%/4)=0
5260 IMX(0)=0
5270 :
5280 REMark PI=3.14159265:rem Complete the last FFT stage
5290 l%=INT(LOG10(N%)/LOG10(2))
5300 LE%=INT(2^l%)
5310 LE2%=LE%/2
5320 UR=1
5330 UI=0
5340 SR=COS(PI/LE2%)
5350 SI=-SIN(PI/LE2%)
5360 FOR J%=1 TO LE2%
5370 JM1%=J%-1
5380 FOR I%=JM1% TO NM1% STEP LE%
5390 IP%=I%+LE2%
5400 TR=REX(IP%)*UR-IMX(IP%)*UI
5410 TI=REX(IP%)*UI+IMX(IP%)*UR
5420 REX(IP%)=REX(I%)-TR
5430 IMX(IP%)=IMX(I%)-TI
5440 REX(I%)=REX(I%)+TR
5450 IMX(I%)=IMX(I%)+TI
5460 NEXT I%
5470 TR=UR
5480 UR=TR*SR-UI*SI
5490 UI=TR*SI+UI*SR
5500 NEXT J%
5510 END DEFine FFT_Real
5520 REMark
```

```
**************************************************************************
5530 DEFine PROCedure IFFT_Real
5540 REMark INVERSE FFT FOR REAL SIGNALS
5550 REMark Upon entry, N% contains the number of points in the IDFT, REX[] and
5560 REMark IMX[] contain the real and imaginary parts of the frequency domain running from
5570 REMark index 0 to N%/2. The remaining samples in REX[] and IMX[] are ignored.
5580 REMark Upon return, REX[] contains the real time domain, IMX[] contains zeros.
5590 :
5600 FOR K%=(N%/2+1) TO (N%-1):REMark Make frequency domain symmerical
5610 REX(K%)=REX(N%-K%)
5620 IMX(K%)=-IMX(N%-K%)
5630 NEXT K%
5640 :
5650 FOR K%=0 TO N%-1
5660 REX(K%)=REX(K%)+IMX(K%):REMark Add real and imaginary parts together
5670 NEXT K%
5680 :
5690 FFT_Real:REMark Calculate forward real DFT
5700 :
5710 FOR I%=0 TO N%-1:REMark Add real and imaginare parts together and divide the time
domain by N%
5720 REX(I%)=(REX(I%)+IMX(I%))/N%
5730 IMX(I%)=0
5740 NEXT I%
5750 :
5760 END DEFine IFFT_Real
5770 DEFine PROCedure DFT_by_Correlation
5780 REMark Upon entry, N% contains the number of points in the DFT, and
5790 REMark XR[] and XI[] contain the real and imaginary parts of the time domain
5800 REMark Upon retrun, REX[] and IMX[] contain the frequecy domain data.
5810 REMark All signals run from 0 to N%-1
5820 :
5830 REMark PI=3.14159265:rem Sets constants
5840 :
5850 FOR K%=0 TO N%-1:REMark Zero REX[] and IMX{}, so they can be used as accumulators
during the correlation
5860 REX(K%)=0
5870 IMX(K%)=0
5880 NEXT K%
5890 :
5900 FOR K%=0 TO N%-1:REMark Loop for each value in frequecy domain
5910 FOR I%=0 TO N%-1:REMark Correlate with the complex sinusoid,SR & SI
5920 :
5930 SR=COS(2*PI*K%*I%/N%):REMark Calculate complex sinusoid
5940 SI=-SIN(2*PI*K%*I%/N%)
5950 REX(K%)=REX(K%)+XR(I%)*SR-XI(I%)*SI
5960 IMX(K%)=IMX(K%)+XR(I%)*SI+XI(I%)*SR
5970 :
5980 NEXT I%
5990 NEXT K%
6000 :
6010 END DEFine DFT_by_Correlation
```

```
6020 REMark
*************************************************************************
6030 DEFine PROCedure NFG
6040 REMark NEGATIVE FRREQUENCY GENERATION
6050 REMark This subroutine creates the complex frequency domian from the real frequency
domain
6060 REMark Upon entry to this subroutine, N% contains the number of point in the signals, and
6070 REMark REX[] and IMX[] contains the real frequency domoan in samples 0 to N%/2.
6080 REMark On return, REX[] and IMX[] contain the complex frequency domian in sample 0 to
N%-1.
6090 :
6100 FOR K%=(N%/2+1) TO (N%-1)
6110 REX(K%)=REX(N%-K%)
6120 IMX(K%)=-IMX(N%-K%)
6130 NEXT K%
6140 :
6150 END DEFine NFG
6160 REMark
*************************************************************************
6170 REMark
*************************************************************************
6180 DEFine PROCedure scan_fft
6190 AT#3;70,150:PRINT#3;"Scanning FFT"
6200 last=128:voffset=244
6210 FOR xp=1 TO 1024
6220 INK#3,6
6230 y1=REX(xp)/80
6240 IF y1<0 THEN y1=0
6250 REMark PRINT#3;xp;" ";y1
6260 IF xp=1 THEN POINT#3,xp,y1+voffset:xp2=xp
6270 IF xp>1 THEN LINE#3;xp2,last TO xp*10,voffset+y1:xp2=xp*10
6280 last=voffset+y1
6290 INK#3;1
6300 NEXT xp
6310 AT#3;70,150:PRINT#3;"           "
6320 END DEFine scan_fft
6330 REMark
*************************************************************************
6340 DEFine PROCedure test_data
6350 FOR count=1 TO 256
6360 REMark a$=chand$(10,count)
6370 PRINT "Chan 1 ";count;" ";CODE(chand$(1,count));" ";CODE(chand$(2,count));"
";CODE(chand$(3,count));" ";CODE(chand$(4,count));" ";CODE(chand$(5,count));"
";CODE(chand$(6,count));" ";CODE(chand$(7,count));" ";CODE(chand$(8,count));" Chan 2
";CODE(chand$(9,count));" ";CODE(chand$(10,count));" ";CODE(chand$(11,count));"
";CODE(chand$(12,count));" ";CODE(chand$(13,count));"
 ";CODE(chand$(14,count));" ";CODE(chand$(15,count));" ";CODE(chand$(16,count))
6380 NEXT count
6390 END DEFine test_data
6400 REMark
*************************************************************************
6410 DEFine PROCedure test_data_2
```

```
6420 FOR b=1 TO 16
6430 PRINT chand$(b)
6440 NEXT b
6450 END DEFine test_data_2
6460 REMark
*****************************************************************
32000 DEFine PROCedure update
32010 SAVE win1_v2scope_prog5_bas
32020 PRINT "Update Complete"
32090 END DEFine update
```