Networking

Both the QL and Spectrum support a simple local area network – a kind of high-speed cassette interface which can link up to 64 computers at once. The QL and Thor have networking hardware built in, but Spectrum users need an add-on interface, like Sinclair's Interface 1 or the popular Disciple disk interface, from Rockfort.

This month we're going to explore Sinclair's network and explain how you can use it to link Spectrums and QLs. This is a feat Sinclair itself pronounced impossible, after initial advertising to the contrary.

Networked computers are joined in a chain by network leads, electrically identical to a single cassette lead but usually longer. Network leads are very simple – just two 3.5 mm mono jack plugs connected with a bit of twin-core wire. Officially 3 amp mains flex is most suitable, but in practice you can get by with almost anything. Cheap speaker cable seems to work fine over short distances, and Sinclair reckoned bell wire worked OK in lengths up to 100 metres.

It doesn't matter which network socket you use at the end of a chain, but you should be careful to leave one socket unused at either end of the network. A switch in each unused socket hooks up a terminating resistor, and the network won't be reliable unless a socket at each end is unused, so that both ends of the line are terminated.

Sometimes the switch in the socket gets bent upwards by repeated plugging and un-plugging, so it's best to use the same socket consistently if you don't like pulling your computer apart and bending the contacts periodically. Some users give up on the switch and use a dummy plug with a 330 ohm resistor inside, at both ends of the network. This alternative approach seems to work fine.

Transmission

Data is sent over the network in 'packets' of up to 255 characters each. Each packet has a prefix to show which machine (or network station) it is addressed to.

Packets for network station '0' are received by every machine that is listening. These are called 'broadcasts', and come in useful if you want to load the same software or data into several machines at once.

If all the QLs on a network have QJump's *SuperToolkit 2* fitted, they can share devices automatically, with no need for users to synchronise machines or agree what that are going to send. A background task monitors the network port and relays system calls over the net, so each machine can access all devices on the network.

Accessing

If you want to use a device on another machine, you just prefix the name with 'N' and the station number, followed by an underscore and the device name – so OPEN #3,"N2_SER" is enough to link channel #3 to the serial port on network station 1. Apart from the OPEN parameter, all the normal commands work, as if the device was on the same machine.

Sadly there's no background file server for the Spectrum, although a simple Basic print and file server is listed in Andrew Pennell's classic *Master your ZX Microdrive*, published by Sunshine Books. If you're interested in writing a Spectrum server, this is a good start.

The pitfall is the Spectrum operating system, which cannot support shared files and devices because it is not designed to multi-task. Former Sinclair employees say that a multi-tasking Z80 version of QDos was developed and tested for the Spectrum and Pandora – the progenitor of the Z88 – but it never reached the market.

Protocols

The QL and Spectrum use the same network protocol, but hardware quirks mean that success is not guaranteed; we've had mixed results with our QLs, and the Thor XVI doesn't seem happy to talk to the Spectrum. We'll try this on MGT's SAM as soon as we get a machine to experiment with.

The example programs are based on listings from the late lamented techies' magazine *Your Spectrum*. The code uses the

Continued on page 97

Continued from page 94

Spectrum as a print buffer, but it's a general-purpose routine to transfer files of any type from the QL to Spectrum RAM, so you could use it for many other purposes. If you're bored with the QL's BEEP, why not send strings for the Spectrum 128 to PLAY, or digitised sounds to be squirted out of a Spectrum sampler or SpecDrum?

Most applications involve using the Spectrum as a 'slave processor', so that the QL can be used for something else while the Spectrum crunches its data. Our example prints a file while the QL gets on with something else. This is called 'spooling', which stands for Simultaneous Peripheral Operation On-Line (S.P.O.O.L). This sounds to us like an explanation made up after the event, but it's a good mnemonic.

It's easy to do the whole job on the QL, if you're a programmer with a compiler capable of producing multi-tasking code, expansion memory and an appropriate QL printer interface but if you've already got a Spectrum and interfaces hanging around, this is an alternative that works well and gives the QL – or networked Spectra – access to Spectrum add-ons without tying up the processor of the networked machine.

Network speed

Sinclair's network is potentially quite fast. Data blocks are transmitted at around 7K a second, but the control signals that allow several machines to share the network slow things down, and there are some incompatibilities of timing between the Spectrum and QL.

We find that it takes about five seconds to transfer 6K of data from the QL to the Spectrum. Much of this delay is probably due to the character-handling routines in the Spectrum ROM, which were written before *Interface 1* was finished.

A separate system call is needed to fetch each character, and the code winds through the Spectrum's streams and channels for a while before extracting the byte from the buffer. This is the reason why sequential disk files on the Disciple, Discovery and Plus D can't handle more than about 1.5K a second.

You could probably go much faster by PEEKing the network packet (a block of up to 255 characters) directly from the Spectrum buffer. MGT's SAM uses a stream system similar to the Spectrum's, for compatibility, but designers Bruce Gordon and Andy Wright promise a 'block-handling' system call to fetch and store data at full speed. Having decided to print blocks of data from the QL, using the Spectrum as a buffer, the problem splits into two parts. First we must move the data to the Spectrum's memory; then it must be output, character by character, to the printer.

You can copy a file from QL microdrive or disk to the network with this standard command:

```
C      O      P      Y
MDV1_YOUR_FILE,NETO_1
```

The MOVE command transfers Spectrum data-files over the network:

```
MOVE "m";1;"YOUR_FILE" TO
"n";1
```

This is fine if you've got the required data in a file ready for transmission – but no good if it's in memory or you want to edit the data together from several places. In that case you need to explicitly open a channel from the QL to the network – then you can PRINT data to the channel, like a file or any other device.

For example, three simple statements are needed to send the current QL program listing over the network:

```
OPEN #4,NETO_1 LIST #4
CLOSE #4
```

The first statement opens a channel for output to network device 1. Then the listing is directed to that channel. You could use PRINT #4 to send text or numbers, instead of LIST. Finally the network channel is closed. This ensures that the last few characters of the file are sent.

If you're keen, you can bundle those three statements into a PROCedure called, say, LLIST to save typing them repeatedly. If you're transmitting data from another Spectrum, the equivalent commands are:

```
OPEN #4;"n";1 LIST #4 CLOSE
#4
```

At the receiving station, things can get a little more tricky. You can't use INPUT to read lines because the QL uses CHR$ 10 (line feed) to mark the end of each line, rather than the Spectrum's CHR$ 13 (carriage return). This means you have to resort to INKEY$ to read characters, one by one, from the network.

Listing 1 shows a simple *ZX Basic* program to read data into the Spectrum from the QL. An OPEN is used, as normal, to associate channel 4 with data from the network. Then INKEY$ fetches characters one by one. They are POKEd into reserved memory. About 40K of data can be stored on a 48K Spectrum, and 7K on a 16K machine (they still exist). In the interests of speed and simplicity, there is no check for overflow.

Once all of the data has been received, the Spectrum stops with an 'End of file' report. We have

```
  5>REM Simple QL-ZX Spooler,
  6 REM Computer Shopper 7/89
  7:
 10 CLEAR #
 20 CLEAR 24999
 30 OPEN #4;"n";1
 40 LET m=25000
 50:
 60 LET a$=INKEY$ #4
 70 IF a$="" THEN GO TO 60
 80 POKE m,CODE a$
 90 LET m=m+1
100 GO TO 60
110:
120 PRINT m-25000;" bytes."
130 FOR p=25000 TO m-1
140    LET c=PEEK p
150    REM Convert LF to CR
160    IF c=10 THEN LET c=13
170    LPRINT CHR$ c;
180 NEXT p
190 STOP
```

Listing 1

```
 10 CLEAR #
 20 CLEAR 24949
 60 FOR p=24950 TO 24982
 70    READ d
 80    POKE p,d
 90 NEXT p
100 RANDOMIZE USR 24950
110:
120 LET m=PEEK 24983+256*PEEK 24984
125 PRINT m-25000;" bytes."
130 FOR p=25000 TO m-1
140    LET c=PEEK p
150    REM Convert LF to CR
160    IF c=10 THEN  LET c=13
170    LPRINT CHR$ c;
180 NEXT p
190 STOP
195:
200 DATA 62,1,50,214,92,207,45
210 DATA 33,168,97,34,151,97
220 DATA 221,34,81,92,205,230
230 DATA 21,183,40,250,42,151
240 DATA 97,119,35,34,151,97,24
250 DATA 240
```

Listing 2

Continued on page 99

Continued from page 97

```
Computer Shopper July 1989, QL-ZX network print spooler

                    ORG   24950           Start address

 30       START:    LD    A,1             Station 1
                    LD    (D_STR1),A      Select source
                    RST   8               Call Shadow ROM
                    DEFB  45              Open network
 40                 LD    HL,BASE         Reset POINTR
                    LD    (POINTR),HL
                    LD    (CURCHL),HL     Select channel

 60       DO_CHR:   CALL  INP_AD          Get a character
 70                 OR    A               Check code
                    JR    Z,DO_CHR        Keep trying
                    LD    HL,(POINTR)     Get pointer
 80                 LD    (HL),A          Store character
 90                 INC   HL              Advance POINTR
                    LD    (POINTR),HL
100                 JR    DO_CHR          Loop till EOF

          POINTR:   DC.W  0               -> Next byte

          BASE:     END                   Start of buffer

Listing 3 - Spectrum code to read the ZX network
```

Listing 3

not trapped the error, as the original program was designed to work with the ZX *LPRINT* printer interface, which requires that you throw a switch to select it in place of Sinclair's Interface 1.

Type GO TO 120 once you're ready to print. The Spectrum prints out the characters it has received, while the other machine gets on with other things. If you have trouble typing 'blind' you can add this line to the program:

```
25 LET g=120
```

and use 'GO TO g' (all on one key) to start printing. In the printing loop an IF statement is used to convert CHR$ 10 into CHR$ 13. Some printers don't require this.

Listing 1 works, but it is irritatingly slow – the Basic program reads the network character by character and takes ages to read the entire file. The output part is also slow because it is in Basic, but that does not matter much as the Spectrum has nothing else to do at that stage, and can keep up with most printers even in ZX BASIC.

Faster transfer

Listing 2 does the same thing as **Listing 1**, but uses machine code to read the network, and consequently works much faster. The corresponding Z80 code in **Listing 3** reads the network at a rate in excess of 1K per second. The line numbers cross-reference the code against **Listing 1**.

The next data address is kept at 24983 and 24984, so line 120 can locate the end of the data. As before, you should type GO TO 120 on the Spectrum to start printing once the QL cursor has re-appeared.

The code assumes that the QL is net station number 1. This is the default, and works fine if you're just linking one Spectrum and one QL. Change the source station number – the second item of DATA on line 200 – if you are using a network of more than two machines. In this case you must give each station a distinct number.

It's possible to set up the spooler to run automatically, if you don't want to swap TV or monitor leads as you work. If you store the Spectrum program on microdrive under the name **RUN** you can set the spooler up by pressing 'R' and Enter in 48K BASIC, once both computers are turned on. The equivalent file name for a Disciple disk interface is AUTOLOAD.

Next packet

When space permits we'll look at networking the other way – from the Spectrum to the QL. Sinclair Research gave up and decided this was impossible, but we've had some success despite the usual hickups.

We'd be very interested to hear from anyone who has tried networking Spectrums or QLs with the Disciple disk interface, which includes a pair of network ports. Theoretically these are Sinclair-compatible, but it is hard to know what that means when the Spectrum code contains errors on transmission and reception that happen to cancel out, and some QLs have dodgy network hardware.

We have not been able to get CST's Thor XVI to talk to the Spectrum at all, over the network. Has anyone else managed this, or found the reason?

If you do get in touch, please let us know the exact version of the hardware you're using, so we don't mislead anyone, and give us software examples if you're doing anything out of the ordinary.

Converts all

by Sid Martin and Timothy Green

In QL Corner this month reports on TexTidy, CST's Thor and SuperBasic compatible compilers

QL TexTidy

TexTidy converts text files between different formats used by three operating systems: QDos, CP/M and MsDos. PDQL supplies the program on disk or cartridge with a 10-page A4 manual, printed in condensed black 'NLQ' letters on purple paper. It's legible, but not particularly easy to read.

TexTidy can translate *QL Quill* documents into MsDos *Quill* format, for export with *DiscOver, Media Manager* or alternatives. You can then edit the file with Psion's *PC Four* package on IBM-compatible computers.

It can also convert QL or MsDos *Quill* files into *Wordstar* format, to suit other computers or emulators. These conversions preserve the format of the document – including variations like **bold** and underlined text.

TexTidy can read *WordStar* files from QL disks; they must be transfered with another utility, or written by a CP/M or PC emulator on the QL. The resultant file includes all the words from the original document, but formatting information is lost. You must load the file into *Quill* with IMPORT rather than LOAD.

The final option strips control codes from *Quill* files, among others, so you can load them into text editors like *ED* or *DevPac QL* – or into the *SuperBasic* interpreter.

It's possible to get *Quill* to read and write plain Ascii anyway if you load with IMPORT and save with PRINT, using a page length of zero and a printer_dat file with no control codes in it. *TexTidy* saves you the trouble, although you have to run it before loading each file, after saving from *Quill*.

The conversion from *Quill* to text format collapses justified lines, but keeps the original column limits. It makes sure there's a new-line character at the end of every line, and leaves one space at the end of each line in the middle of a paragraph. You have the choice of QDos or CP/M/MsDos end of line characters.

TexTidying

The program is a 53K task, started with EXEC or EXEC_W. You can format, view and erase QDos files from inside *TexTidy*, so there's no need to use EXEC unless you need to issue other commands. The program displays a cursor at all times – but sometimes it displays two, and you have to find the right one after switching to other tasks before you can regain control.

Menu-driven

TexTidy is controlled by picking options from simple menus. The arrow or letter keys pick a particular option; then you press Enter to select it. This works well.

The directory display uses standard extensions: _EXP for export, _TXT for plain text, _WS for *WordStar* and _DOC for *Quill*. The display distinguishes between MsDos and QL *Quill* formats, which use the same extension, but cannot be loaded by the other program without conversion. Psion is strange like that.

Limitations

TexTidy only works with documents and text files – you can't use it to 'tidy' *Archive, Abacus* or *Easel* datafiles unless you EXPORT the data first. Annoyingly, Psion provides no way to export the formulae from *Abacus*, so you can't use *TexTidy* to move a spreadsheet design to another package.

TexTidy recognises the common programming language extensions: _C, _PAS, _ASM and _BAS. It won't convert to and from a single directory, which could be annoying if you've got microdrives and don't use directories at all – but it will copy to and from different directories on a single device.

Performance

We found a minor bug in the 'convert' option, which prints a warning if you press Enter without selecting any files, and returns you to the main screen without the list of keys at the bottom of the display. At this point you can't select anything unless you press Escape and re-enter the option.

The disk grinds furiously for a few seconds when first you read the directory. *TexTidy* must load the start of each data file to check what it contains. This process is much faster the second time, when the disk information can be read from Slave Blocks (*Shopper* 12).

The screen display is neat but sluggish. *TexTidy* was compiled with Metacomco's bugged C compiler, so it won't work with screen accelerators and gets upset by Ultrasoft's *Toolkit III*. It seems that the program was once known as *FileTidy*, as that's what it calls itself when it asks if you want to quit the task.

Conversion speed is tolerable, but not fast. *TexTidy* took 80 seconds to translate a 42K *Quill* file into plain text, reducing the size by about 20 per cent in the process. A 2K file was converted in about 8 seconds. You can stop conversion by pressing a key while *TexTidy* is not preoccupied with the disk drive, but this usually leaves a partly-written file lying around.

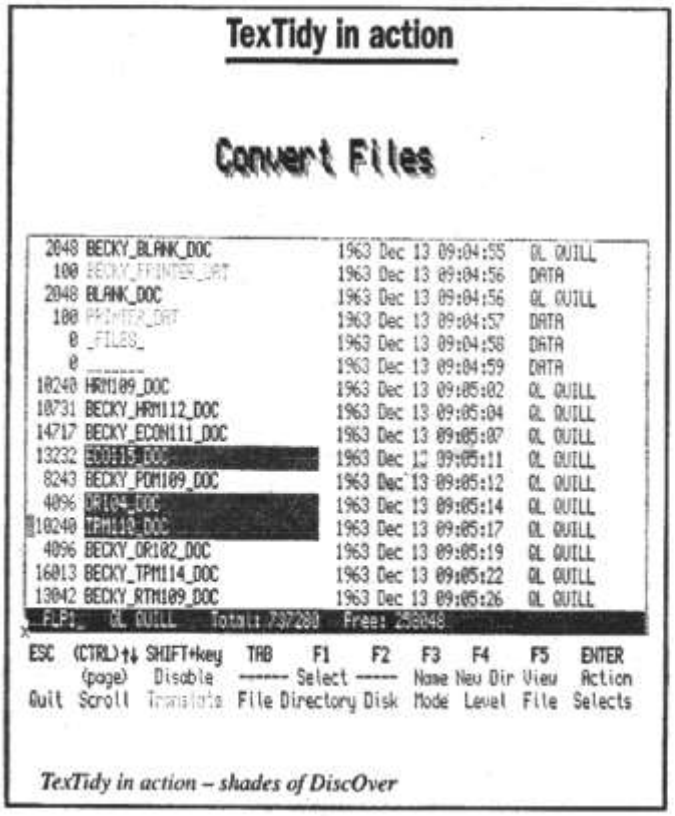
Verdict

At £10 with plentiful documentation *TexTidy* is a cheap and painless solution to some file-conversion problems.

Under Scrutiny

QL TexTidy £10
PDQL, Unit 1, Heaton House, Birmingham B1 3BZ (021) 200 2313
Converts text files between QDos, CP/M and MsDos formats.
Hisoft Basic ST £79.95; Hisoft Amiga Basic £99.95
ST & Amiga DevPac and Basic compilers
Hisoft, The Old School, Greenfield, Bedford MK45 5DE (0525) 718181
QL Basic compilers
Digital Precision, 222 The Avenue, Chingford, London E4 9SE (01) 527 5493
ST QL Emulator
Strong Computer Systems, Bryn Cottage, Peniel, Carmarthen, Dyfed SA32 7DJ. (0267) 231246

TexTidy in action



TexTidy in action – shades of DiscOver

68000 SuperBasic

Shopper correspondent D A Elgee has heard reports that HiSoft's ST and Amiga compilers are compatible with *QL SuperBasic*, and feels this is too good to be true. Well, it is true in principle, as HiSoft's 68000 compilers are derived from

Supercharge – the first *SuperBasic* compiler on the QL.

In 1986 HiSoft bought a license to convert the *Supercharge* design for the Atari ST. Since then it has developed the program in all directions. It has spawned *HiSoft Basic*, *Power Basic* and *First Basic* on the ST, and *HiSoft Basic* for the Amiga.

The first ST compiler

The QL compiler was used to create the first ST compiler, via a new library. This compiled itself in ever-expanding versions until it could handle Microsoft's Basics and *Atari Basic* – but its own source remained in modified *SuperBasic*, so the compiler supports QL goodies like SELect, named REPeat loops (with NEXT and EXIT), procedures, functions, and reference parameters.

There are a few snags, from the *SuperBasic* programmer's point of view. Some commands are slightly different, because of clashes between *QL* and *MBasic* keywords, but it's easy enough to transliterate these.

FOR loops follow the original Basic rules, so EXIT and loop epilogues are not supported. *HiSoft Basic* uses LEFT$ and RIGHT$ rather than QL string slicing, but conversion is simple. There are lots of new data-types – dynamically allocated strings of any length, 32-bit integers and 16 digit double precision floats.

QL Toolkit commands are not available – but many of them are machine-specific anyway. You can call ST and Amiga libraries from Basic, and there are lots of new keywords, especially on the Amiga version.

You can use GET and PUT to access the blitter, PALLETTE and PAINT for colour graphics, Object (sprite) control commands, plus SAY and TRANSLATE$ for speech synthesis. Events can be trapped with ON..BREAK, ON..COLLISION, ON..ERROR, ON..MENU, ON..MOUSE and ON..TIMER.

Documentation

The Amiga version has a 350-page manual with a six- page index. The ST *HiSoft Basic* manual is even bigger at 400 pages, with six contents pages but no index as such. There's loads of library information, absent from the 'budget' £40 *Power Basic* manual.

Basic bundles

Another cut-down version of *HiSoft Basic* forms part of Atari's latest sofware bundle. *First Basic* has almost all the features of *HiSoft Basic*, including the full *GEM* library, but it does not generate stand-alone code. You must load and compile your programs, at about 2,000 lines a minute, every time you want to run them.

First Basic drops you back into the editor as soon as it finds an error, whereas *Hisoft* and *Power Basic* find all the errors in one go. The bundled version lacks optimisation controls – it always produces 'safest' code – but it's still a big step up from *Atari Basic*, in terms of speed and syntax. *First Basic* comes in the new bundle with every 520ST, but you're still stuck with *Atari Basic* if you buy a 1040 or Mega-ST.

Conversion to the ST removed the multi-tasking features of *Supercharge*, but the Amiga version of *HiSoft Basic* multi-tasks and shares libraries between tasks. It can't LINK_LOAD or share compiled Basic variables, procedures or functions, unlike tasks compiled with *Turbo* on the QL.

John Silk of PDQL

Origins

The history of these compilers forked back in 1986, when HiSoft started work based on *Supercharge*, the stand-alone QL compiler which preceeded *Turbo*. Both *Turbo* and *HiSoft Basic* started out from the *Supercharg e* design, but they went in radically different directions.

A third QL Basic compiler, *Q-Liberator*, arrived in 1986. *Q-Lib* is re-entrant, easy to use and very forgiving of poor structure, but it relies heavily on Sinclair ROM routines, limiting its speed and conversion potential.

Turbo is a logical extension of Sinclair *SuperBasic* and the *QDos* programming environment, while *HiSoft Basic* is compatible with several different dialects and more portable between machines – for instance it uses the *MBasic* FIELD scheme for file-handling, and compiles *MBasic* and *QuickBasic* programs from the PC with minimal alterations.

Basic advice

You can trace compiled code with *DevPac ST* or *DevPac Amiga* – also based on a QL original – but you need to be a good assembler programmer to see the wood from the trees. It's hard work, and these compilers won't allow interpreted development, so don't chuck your QL away unless you're happy with the pidgin Basic interpreters supplied with the ST and Amiga, or every program you write works first time...

CST THOR NEWS

Reports from several *Shopper* readers confirm that CST has the Thor XVI super-QL back in production after its move to Denmark. The first Danish-made machines have been delivered to customers, and the new version of the operating system, ARGOS, is reportedly much faster than the original QL-based code.

We say *reportedly* because we have also heard rumours of a few new bugs in the ROM; CST boss David Oliver has promised us a copy of the new ROM as soon as it is 'stable' so we hope to be able to report again soon.

Late in April CST was in Moscow, persuing the long-awaited deal to supply thousands of Thors to the Russian education market. The company is now talking directly to the Government, rather than to middle-men; the first order is said to comprise 3,800 machines.

Legacy of the QL

The 16-bit software market owes a massive debt to the QL, which introduced thousands of programmers to the 68000 processor family. The QL was in production months before design work on the ST started, yet QL programs are still being converted and getting good reviews when they arrive, years after their first launch, on ST and Amiga. Besides *Devpac* and *HiSoft Basic*, the QL brought AmigaDos authors MetaComCo into the 68000 world; their *Assembler, C* and *BCPL* compilers started life on the QL.

The QL is hardly revered as a games machine, but it introduced the world to games like *Psion Chess*, Magnetic Scrolls' *The Pawn* and Pyramide's 3D *Wanderer*, recently re-launched by Elite.

The QL's 68008 processor is slower than the 68000 in the ST and Amiga, but code is totally compatible; that's what made it easy to convert *Devpac QL* for the ST and Amiga. A single QL source file could be used to generate all three versions; the first code was moved from the QL to the target machines via RS-232 ports.

QL code seems blisteringly fast and economical when you run it on American 68000 machines. It's easy for programmers starting out on the ST and Amiga to squander power and RAM, whereas QL tasks have resources to spare when they turn up on newer machines.

The basic QL has only about 85K of free memory. Standard machines can expand this to 724K, and clones like the Thor XVI and Atari QL Emulator can handle several megabytes – but the large market for 128K QL programs means that QL tasks must be both concise and efficient.

Releases

New QL programs are still released regularly, with some 120,000 UK-made machines in circulation, plus tens of thousands made under licence by Samsung in Korea, and new production from clone-makers like CST and ABC. These machines have improved hardware and sell well, particularly in mainland Europe.

Programmers carry on writing for QDos because of its advantages over AmigaDos or the ST's TOS; it is well- documented, simple to call, easy to extend and can multi-task almost anything – an advantage you must use to appreciate.

The QL was sold as a general purpose machine, and, unlike its near-compatibles, it has never been pigeon-holed into the games and music market. There's loads of Public Domain QL software, available from active user groups like QLAF, Quanta and QL-SUB.

At the moment conversions to American 68000 machines are running about three years behind the QL marketr. The hardware emulator runs at full ST speed, about twice as fast as an expanded QL, yet it supports all the QL features except microdrives (ahem!) and the network.

QL programs multi-task blindingly quickly, are directly supported in the UK, and cost much less than their ST counterparts. If you're desperate for 512 colours, you can still run ST programs at any time; it's wise to use a genuine Atari monitor if you intend to use both the QL and ST composite video outputs, as both have non-standard timing requirements.

True QL clones like the Thor XVI and ABC's promised Enigma are still attractive if you need several QDos machines. They support Sinclair's network, which lets up to 64 machines communicate and share devices, connected only by some bell-wire and earphone plugs. The original QL may no longer be in production, but the good ideas go rolling on.