An EMAP Publication

SINCLAIR'S SUPERMICRO

QL USER

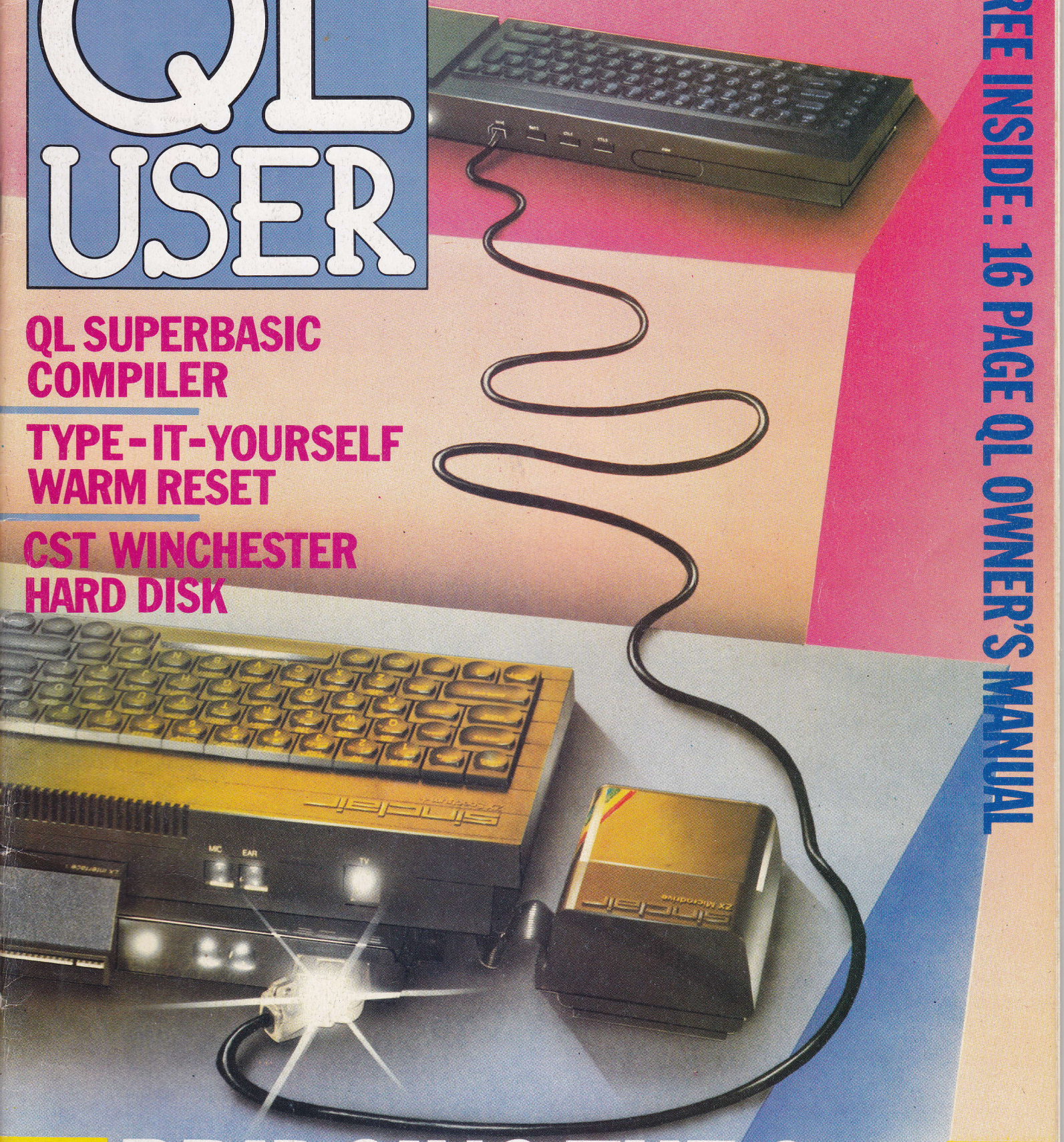
Every Month £1 October 1985
Germany DM6.00

FREE INSIDE: 16 PAGE QL OWNER'S MANUAL

QL SUPERBASIC COMPILER

TYPE-IT-YOURSELF WARM RESET

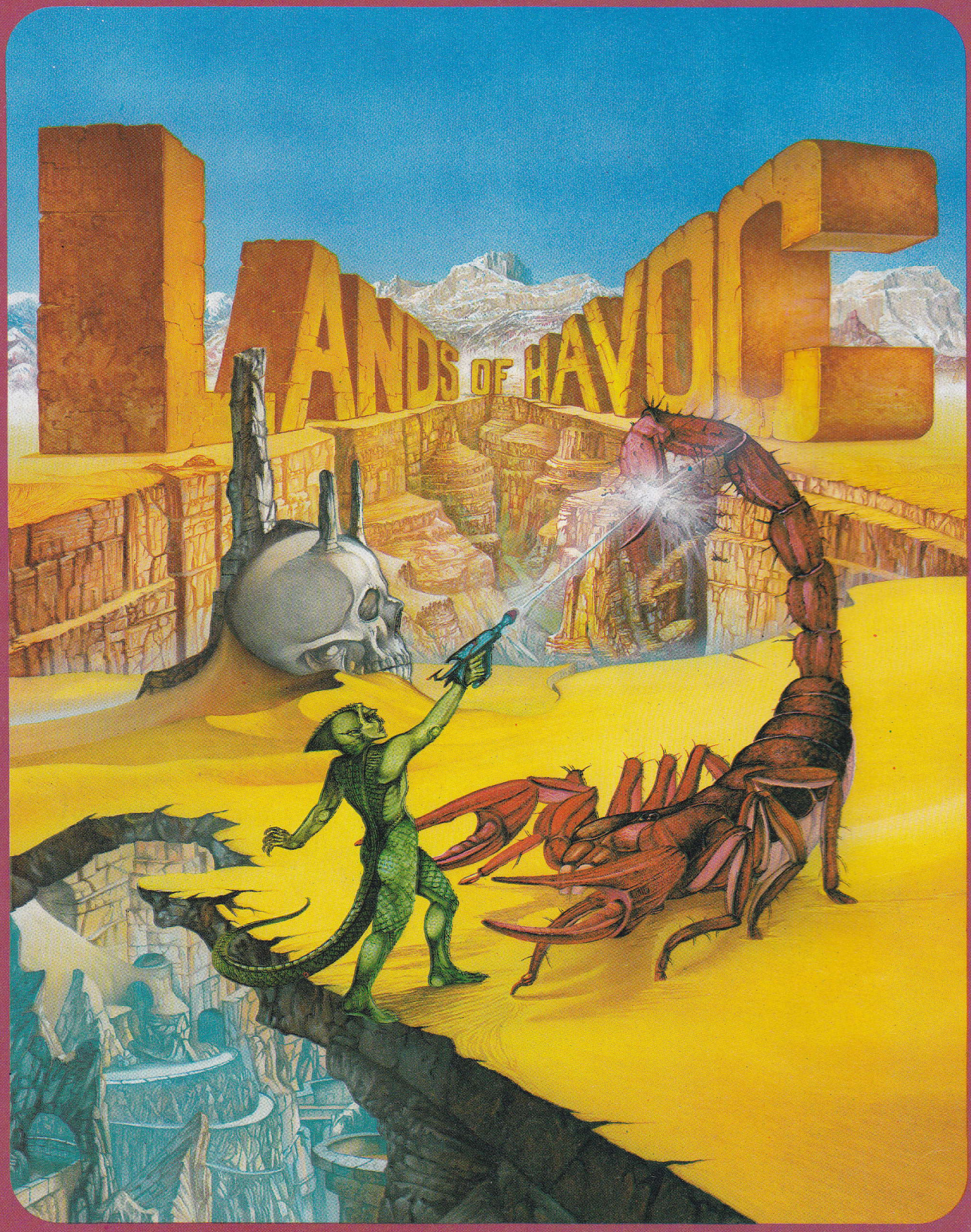
CST WINCHESTER HARD DISK



BRIDGING THE GAP
SPECTRUM CONNECTIONS

COMMODORE 64/ SINCLAIR QL

2,000 Screen Arcade Adventure



MICRODEAL

QL USER

October 1985

Editor	Paul Coster BSc
Assistant Editor	Paolo Baccanello
Editorial Assistant	Shirley Eborn
Art Editor	Mike Spiller
Technical Consultant	Adam Denning
Advertising Manager	Phil Baker
Advert Production	Yvonne Moyser
Production Assistant	Serena Hadley
Publisher	Neil Wood

Contributors

Mike James, Nicky Trevett, Adam Denning, Marcus Jeffery, James Lucy, Richard Cross, Simon Goodwin, I Robinson, L A Privett, Alan Turnbull, Paul Hickling, Ian Stewart, A Didcock and Silhouette.

Cover photography – Rob Brimson
Illustration – Stephen Dew
Microdrive Exchange – System Design

Competitions

Following our announcement in the August edition on the complexities of the Inside Out competition, many of you wrote in asking us to check your entry against the overall winner's list.

As a result, we were able to select the second prizewinner – he is Paul Gelling from Tewkesbury in Gloucestershire. Paul receives an Insider disk interface board from Silicon Express.

For those of you who missed August's *QL User*, the winner was Peter Skiba from Blackpool. He wins the Silicon Express disk system and Insider board.

QL User,
Priory Court, 30-32 Farringdon Lane,
LONDON EC1R 3AU.

Telephone 01-251 6222

Published and Distributed by EMAP Business & Computer Publications.

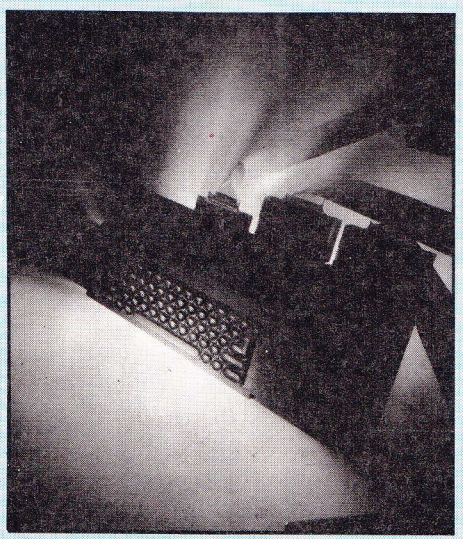
Subscriptions, backnumbers and reprint information from: Carl Dunne (Magazine Services) on 01-251 6222 Ext 2429.

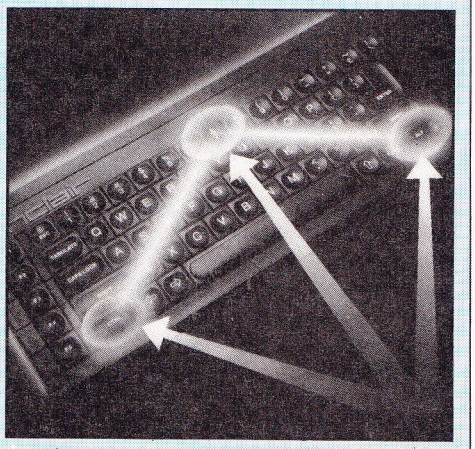
U.S. Sales Agent – Motorsport, RR1, Box 200D, Jonesburg, MO 63351.

Typesetting by Crawley Composition Ltd, Stephenson Way, Three Bridges, Crawley.

Printing by Riverside Press.

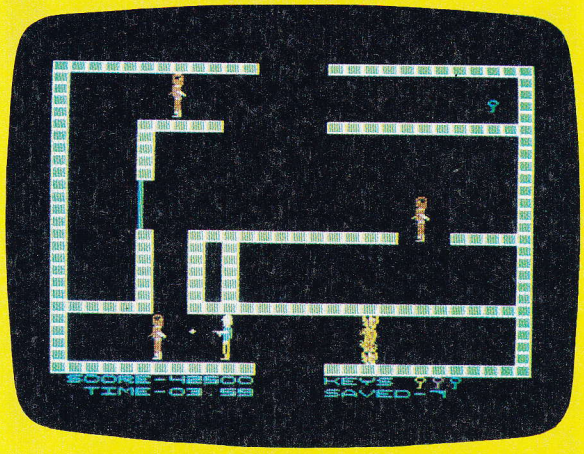
CONTENTS



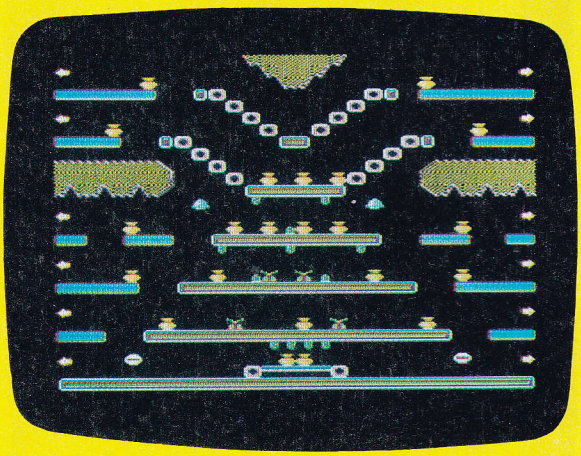


FREE!

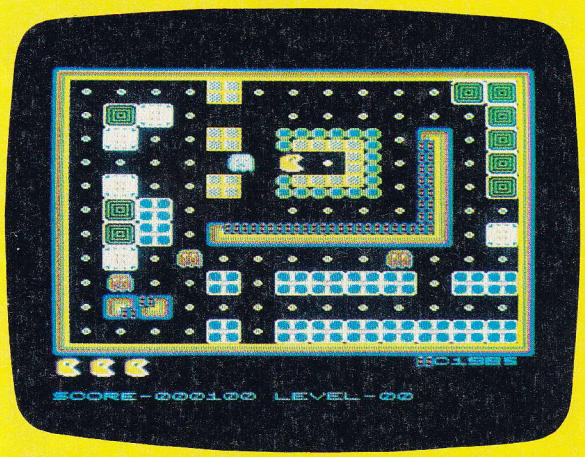
Four free games for every QL owner!



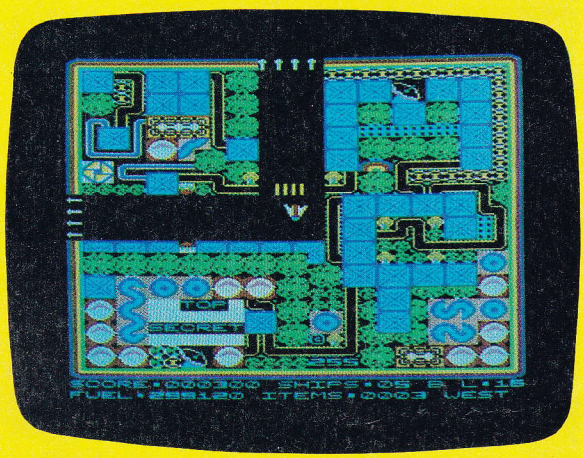
M – COSMIC. Your job is to save your friends, deep-frozen in an enemy space-ship. A mission of cool nerve and firm courage.



M – TREASURE. Deep underground there are bags of gold for you to collect – and man-eating spiders to stop and slay you.



M – CRUNCHER. Stay clear of the jaws that chase you through the maze, avoiding hazards and building points as you go.



M – METROPOLIS. Fast-moving excitement as you steer through screen after screen of incredible dangers and rich rewards.

MEDIC now offer every Sinclair QL owner four brilliant new games – absolutely free! Just send three formatted blank microdrive cartridges for these four best-ever games for the QL, testing your skill, speed and concentration with colourful screens and inventive displays.

All games 100% machine code, with flicker-free multi-colour multi-screen graphics. Three full cartridges of fast-moving instantly-responsive action.

WHY? Medic offer you this fantastic games value FREE, because they want you to see the quality of Medic software, and as their way of showing their committment to this superb machine.

Fill in and post the coupon TODAY to make sure of your MEDIC free games.

To: Medic Datasystems Limited,
Hackwood Lane, Cliddesden,
Hants. RG25 2NH.

Please send me my four free MEDIC games.
I enclose three blank formatted cartridges, plus £2.50 to cover postage, packing and recording.
(Cheque or P.O. U.K. sterling or Girocheques).

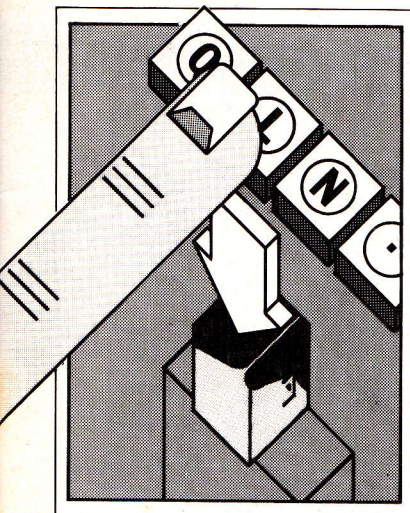
Name_____

Address_____

_____

Telephone Code_____ Number_____

Please complete all details and write clearly.



NEWS

The latest software, hardware and information on the QL product front.

Massive Cuts

As of 2nd September Sinclair Research will have cut the price of the QL by half, from £399 to £199.95. According to Jane Boothroyd, UK Sales and Marketing manager, the massive reduction is 'in line with reduced manufacturing costs'. Industry sceptics, however, would view this as a last ditch attempt to save the QL in the light of fierce competition from Atari, Amstrad and Commodore. Whatever the reason, there can be little doubt that the new pricing, coming as it does immediately prior to the peak selling Christmas period, will send shudders through the industry and hopefully the competition.

At under £200, the QL now makes a mockery of any distinction between home computers destined for entertainment use and those for serious use in small businesses or education. With 128K RAM, 32-bit architecture, 2×100K microdrives it can outperform any mass market games playing machine currently available. Furthermore its award winning bundled software means that users will have access to the so-called 'essential' applications (ie, word processing, database and spreadsheet operations) at no extra cost.

The new pricing would also seem to indicate that Sinclair now see the QL as the logical successor to the Spectrum. This view would seem to have been anticipated by independent peripheral manufacturers who have been quick to support the machine. Users are now able to pick and choose between a variety of competitively priced expansion options and need not necessarily depend upon those marketed by Sinclair themselves.

The same cannot be said of games software houses. Whilst the QL is well supported in terms of business and development software, there are few games of note running on it. Despite the obvious signs that 8-bit processors such as the Z80 and 6502 have had their day none of the big names have been prepared to risk a move across the 68000.

Given the QL's new price and the fact that Dixon's alone are, we understand, stocking 30,000 machines, it will be interesting to see whether this is still the case after Christmas. Who said the QL was finished?!

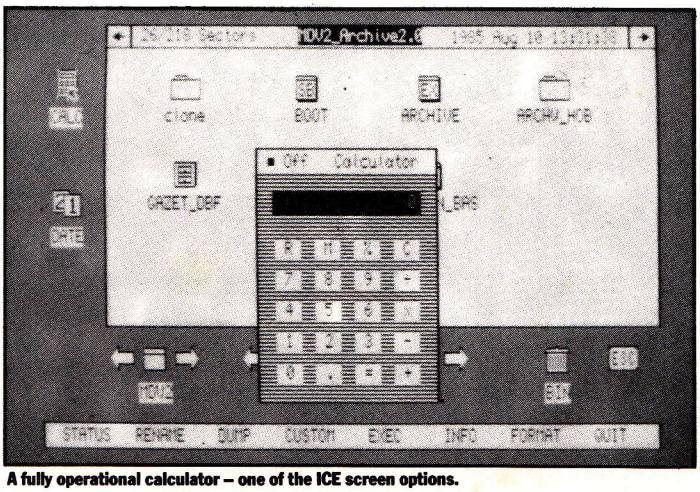
MG Extinct

The MG ROM available on overseas models of the QL is destined never to appear in the UK. Sinclair are understood to be developing an even better ROM for home consumption. This is just as well as the MG has found few supporters overseas. Apparently an 'improvement' in a line drawing algorithm has meant that points drawn on the screen do not appear where they are supposed to.

ICEd Over

Eidersoft's Icon Control Environment or ICE for short, is the QL's answer to GEM. Designed as a user-friendly 'front end' for QDOS, ICE allows you to control the QL's various functions with childlike ease.

Similar to GEM it cocoons the user from the vagaries of the operating system in a protective graphics shell. This takes as its theme the idea that a parallel can be drawn between operating a computer and running a busy office. Switch on your QL and the screen depicts a tidy little workdesk with little figures or 'icons' dotted over it representing devices that can be linked to the QL, various files in storage and such odds and ends as a wastepaper basket, calculator and calendar. All you have to do is select an operation from a control panel at the foot of the display and point to the appropriate item or items on the screen and ICE will take care of the rest. In this way keyboard input is reduced to an absolute minimum and the system is virtually idiot-proof.



A fully operational calculator – one of the ICE screen options.

Without an elaborate system of pull down windows ICE is very much less sophisticated than GEM. However, such a comparison is largely academic as without drastic modification the QL could not possibly support all 90K of GEM. ICE, on the other hand occupies next to no memory as it sits in a 16K EPROM that plugs into a socket on the back of the QL.

ICE will cost £49.95 and will be marketed along with a microdrive cartridge containing a number of useful utilities colourfully described as ICING. The most interesting of these is QTASK a program that permits users to flick back and forth instantly between programs loaded into memory simply by pressing CTRL+F3.

Psion Update

The latest versions of Psion's software are putting in an appearance in Europe. The differences between version 2.00 and 2.03 are:

1. Memory allocation problems have been ironed out so that you may re-use your programs without getting an error message and being compelled to reset the QL.

2. Quill incorporates an option to import a file by line or paragraph.

3. In addition to the current Epson screen dump, Easel will cater for nine other printers and will allow for standard output to Epson and HWP plotters.

4. Archive now include an option to design your own printed forms using SEDIT.

5. A facility to load from and save to the Network device has been implemented.

6. The algorithms controlling use of memory with Quill have been improved so that it is now possible to create 12K documents on Quill before the program overlays it to microdrive.

Further information from Psion – 01-723 9408

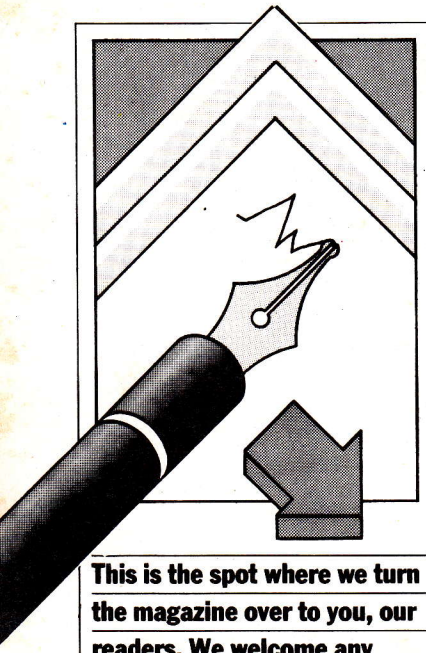
RAMbling On

Silicon Express have come up with one of those simple ideas that you wish you'd thought of yourself. Simply prise the lid off your QL, yank out the 16 64K DRAMs installed courtesy of Sinclair and plug in a similar number of the very latest 256K chips.

If you did it yourself, you would have transformed your QL into the most expensive doorstop you've ever had the privilege to buy. And, having invalidated the warranty you would be stuck with it for the rest of its non-biodegradable life.

However, if the microsurgeons at Silicon Express perform the operation using a special desoldering machine, you get a fourfold increase in memory, a pristine QL and a 90 day warranty for everything but the microdrives. As the latter are the things most likely to go wrong with the QL the upgrade would seem inadvisable for those with brand new machines. For the rest however, at a third of the price of a conventional upgrade yet identical in performance, it's ideal.

OPEN CHANNEL



This is the spot where we turn the magazine over to you, our readers. We welcome any comments, criticisms or anecdotes about either the QL or QL User. The address to send your letters is: Open Channel, QL User, Priory Court, 30-32 Farringdon Lane, London EC1R 3AU

Dead End Game

I have bought a copy of the program QL Cavern. The game works well until you have collected all 395 diamonds then nothing happens. After spending one hour getting this far I went through most of the caverns looking for a possible exit. None was to be found.

I have rung up Sinclair to ask but with no luck. If any of your readers have found the exit, or if there isn't one, I would like to know. Is poor BJ condemned to a life of wandering through empty caverns?
P D Riggs
Witley, Surrey

Having collected all 395 diamonds in QL Cavern, what do you do to end the game? I have tried several things, but to no avail. Please can anybody help!
B Moffat
Glenrothes, Fife

Half Way Happy

To be truthful I am a button pusher. New to computing, so I've no idea what's what! I have difficulty trying to get programs published in other magazines to work. This I guess is due to typesetting errors and the like. However, I'm elated. For the first time I struck gold with two program listings in my first ever copy of QL User (August edition). Yes both *Miners* and *Type Right* work!!! But I had the same old 'no success' problems with *Mouse Maze* and *Qthello*. There must be some way to ensure that program listings are 100% accurate. Then us button pushers out here will get the satisfaction we deserve.

So just two questions, why are listings inaccurate and what can be done to resolve the problem. Maybe good old quality control would help?
Steve Redhead
Tokyo

We repeat this time and time again. Once the programs have been tested, the listings come hot from the QL and are virtually photographed (PMT'd) into the magazine. No typesetting whatsoever occurs. For this reason us pen pushers are confident that button pushers like yourself have made mistakes in typing in the programs. By giving tips as to how to debug your programs either over the phone or in the magazine we ceaselessly endeavour to help you learn the error of your ways. However, if further proof is required as might well be the case with long listings, then simply get hold of a copy of the program from our Microdrive Exchange – it won't cost you an arm and leg. Incidentally, if you read the instructions for 'Maze Mouse" you'll see that "Maze Drawer" has to be typed in first and saved to mdv1_.

Sound Barrier

A few months ago there was an article about making sounds on your QL. I think there should be a section for people to send in their ideas from new sounds. Here are a few suggestions.

BEEP 0,0,67,475,0	(alarm clock)
BEEP 0,0,9,45,79	(alarm)
BEEP 0,0,2,43,1	(shaver)
BEEP 0,0,53,52,4	(Helicopter)

Alan Russel
Edinburgh

Only the tone deaf would appreciate the QL's sound making capability. One voice with a limited range sounding out through a tinny speaker is hardly going to impress. Nor is a BEEP command whose usage is largely experimental and often confusing. Consequently we welcome any suggestions. However, before you send in your contributions take a look at the Composer program, listed in the PROGS, to see just how much can be done with so little.

Question Time

By the end of September I intend to buy a printer. Can you advise me which is better for the QL. The Brother M-1009 or an EPSON? Which corresponds most closely with the capacities of the QL such as underlining, super and sub script, graphics etc.

Secondly, when you open a channel to the screen you use the device scr_. When can you (or must you) use the device con_? I don't understand the difference between them.

And finally, why can I not enter machine code programs into my QL? I have tried the CALL command but it is ineffective! Help!
Marc Delterne
Belgium

Epson's range, in particular the new LX-80, comes out on top. Setting up the appropriate drivers for Psion's packages is a doddle and more importantly all versions of Easel will work with them. However, you should bear in mind that Epson printers are more expensive and do not come with an RS232 interface as standard.

Next, opening a channel to a console device is necessary if input from the keyboard is to be directed down that channel. For example, the command INPUT #1,a$ will fail unless you have previously entered OPEN #1,con_. Opening a channel to the console is more costly in terms of memory as a type-ahead keyboard buffer (128 bytes default) is automatically set up.

Incidentally, closing the QL's default console channel (CLOSE #0) and redirecting input via another channel provides a very good way of preventing a SuperBasic program from being listed once it has been run.

In answer to your last question, you need an assembler program to type in machine code routines. The CALL command simply allows you to execute the routines once they have been loaded into memory.

Instant Recall

I am only proficient in SuperBasic and would like to write a program which draws large figures on to a screen. Currently the graphics take about 5 seconds to draw but I need them to appear instantaneously (well almost).

Obviously the way to do this is with a machine code routine which is beyond me. However my limited contact with other micros (PET, RML) has shown that pictures can be built up in screen memory and the screen then unblanked. The time it takes to do this is unimportant. Can this be done on the QL? If so tell me how?
Paul Sander
London E11

The routine to build up pictures in memory is written into the firmware of the machines you mention. Unfortunately, in this respect the QL's video display is exceedingly crude. Neither this routine nor a number of others vital for smooth animation are written into the firmware. These must be added by the machine code programmer and will execute at very much slower speeds. For this reason games on the QL invariably redraw entire screens rather than scroll from one side to the other.

Two Into One

It seems that most add-on memories and floppy disk interfaces use the main I/O port on the left hand side of the QL. Is it possible to use both extra memory and a disk interface simultaneously?
Alun Phillips
Maidstone, Kent

Peripherals manufacturers have come up with a number of different solutions to this problem. Technology Research include on board DRAMs, up to a maximum of 128K, on their disk interfaces. PCML, we understand will also be following suit providing up to 256K on their interface board.

If you wish to move beyond 256K you have three choices. The first, is to have the DRAMs installed internally within your QL by Silicon Express giving you a total of 512K. Surprisingly, this is by far the cheapest option but invalidates your warranty.

The second alternative, is to purchase additional memory and disk interface separately and connect them up to the QL using CST Plus Four expansion motherboard with separate power supply. This will give you a total of 640K RAM on the QL. However, this method is very expensive and is only advisable if you are contemplating attaching further devices to the QL such as a Winchester or additional floppy drives. Furthermore you should consult CST prior to purchasing extra memory to find out which units are compatible.

The final solution is to purchase Medic's multipurpose expansion system. Here, their disk drives provide the additional power necessary to run the extra 512K of RAM.

Whatever your choice, you should appreciate that if extra memory is to be used to the full you will require a RAM disk driver.

BASIC Misunderstanding

I have found yet another extremely strange function hidden inside the depths of my QL. After powering up the machine (JM), try typing OPEN #0,scr_. In my case the computer fills up the top window (in TV mode) with the error message 'bad parameter'.

Also, if you do the same with windows 1 and 2, it takes two lines the default TV size and changes the default colour of window 1. Even though the size change of one of the windows is documented in the manual, I do not understand the other two changes. Is there any explanation for this?

Incidentally, I have also found how to print an arrow pointing to the right as the User Guide has the wrong keying. Instead of CTRL [, type CTRL].

J Alderson
Studham, Beds

None of these discoveries can be considered anomalies. All you are doing is altering the defaults for various devices. The parameter error crops up simply because you have failed to close channel zero. To confirm this enter:

```
10 CLOSE    #0
20 OPEN     #0,scr_
25 PRINT    #0, "Hello"
30 CLOSE    #0
40 OPEN     #0,con_
```

You will see that everything works fine. As for the changing the window sizes you should be aware that there are three sets of defaults. Those for TV and Monitor which automatically set up windows for channels 0,1 and 2 and a third which may be applied to any window defined. This default is particularly useful as it provides a means of getting round the unwieldy WWxHHaXXxYY extension to the OPEN command. Instead use either OPEN #n,con_ or OPEN #n,scr_ followed by WINDOW #n,WW,HH,XX, YY. As the latter function uses integer arguments as opposed to a single string it is very much easier to manipulate.

'An astonishing deduction Holmes, I don't know how you do it.'

Viglen

offer the best alternative to the microdrive

Speeding up your data access with Viglen disc drives means instant reliable access every time

Specially designed to fit neatly into the case of the Sinclair QL, the Q-Disc Interface Board and companion disc drives are colour-matched to compliment your QL computer.

5¼ inch Drives	Drive on its own	Drive with Q Disk Interface
400K Single drive	£99	£199
800K Dual drive	£179	£249
800K Single drive	£139	£247
1.6M Dual drive	£269	£329
3½ inch Drives		
800K Single drive	£158	£258
1.6M Dual drive	£287	£387

The above drives are compatible and ready to plug in and use with all currently available QL disc interfaces.

When bought with the disc interface they are ready to plug in and use.

Prices correct at time of going to press. Offers subject to availability. Even lower prices with other QL interfaces.

Visit our showrooms
Open Monday – Friday 9.30 – 6.00 Saturday 9.30 – 4.00

Unit 7, Trumpers Way
Hanwell W7 2QA

Carriage
Please add £12.00 for carriage. Orders are usually despatched to you within twenty-four hours of receipt.

Viglen are also major suppliers to educational and government establishments and welcome further enquiries and orders.

Post to: VIGLEN COMPUTER SUPPLIES.
Unit 7, Trumpers Way, Hanwell W7 2QA.
Credit card holders may order by telephone on 01-843 9903.

Please send me _____

I enclose Cheque/P.O. for £_____ incl. carriage.
Cheques payable to Viglen Computer Supplies.

Name: _____

Address: _____

_____

_____

I prefer to pay by ACCESS/BARCLAYCARD (delete one).

Card No: _____

Credit Cards valid only if signed by card holder. Address must be the same as card holder.

Signature: _____

·THE· SUPERBASIC COMPILATION

Simon Goodwin previews the program that sets out to revolutionise the QL's built-in language

Back in 1984, it sounded as if SuperBasic would be as much of a 'Quantum Leap' as the QL itself. We were promised a powerful, multi-tasking, extendable language, which would run at constant speed regardless of program size.

However, what appeared was a well-designed but hastily written BASIC, with a stack of bugs which took Sinclair a year to fix. More importantly, the language wouldn't multi-task and 'benchmark' timings were slow – and got slower as the program size increased. The displayed precision was only seven digits which is limiting for business use (though can be coded around).

In July 1984, as the first 'working' QLs became available, Digital Precision started work on *SuperCharge* – an automatic 'compiler' to translate SuperBasic into fast machine-code.

The aim was to fix all the problems, without affecting the power of the language or its expandability.

SuperCharge's main requirement was that it should compile the vast majority of existing SuperBasic programs, without alteration. It should be able to be run on all versions of the QL, with the minimum amount of memory. And it should be fast, in terms of speed of operation as well as the generated code.

SuperBasic is slow because it has to 'look-up' every command, line and variable in a program, whenever it is found. That's why it is called an interpreter – every dot and comma is laboriously checked, over and over again, as the program runs. By contrast, *SuperCharge* performs the appropriate action, with no 'look-up' at all and is thus a much faster process.

The *SuperCharge* compiler has two main components – a 'parser', to analyse the meaning of the original SuperBasic program, and a 'code-generator' to generate corresponding machine code. There is also a demo program and set of 'add-on' procedures for task-control and error-trapping.

The parser was first written in SuperBasic consisting of about 2,500 lines without a single GO TO or GO SUB. Naturally, its first major challenge was to compile itself! The parser reads the program to be compiled from the QL's memory. The results of its analysis are stored in part of the QL's display RAM, so that no temporary files are needed. This makes compilation very fast, and leaves up to 40K (on a standard QL) for the program text to be compiled.

The code-generator is a small program which loads 'on top of' the parser. It is written in machine-code so that it can manipulate bits and bytes at top speed; the author was Gerry Jackson, who previously wrote Digital Precision's *Super-Forth*.

The code-generator selects the machine-code routines which will make up the compiled task. There is no 'library overhead' – routines are only included if they are needed. The code is written to a disk or microdrive file, so it can be loaded with EXEC. Compiled programs load fast, since they don't have to be 'tokenised' by SuperBasic. They are also protected against prying eyes, as they can't be LISTed, saving you the 'pirates worry'.

There are a few differences between compiled and interpreted programs, besides being a lot faster than the originals. The first is that numbers are displayed to a full nine digits of precision – good news for millionaires and those who work in Lire or Yen!

Non-standard commands, such as those built in to disk systems, will work in compiled programs, so long as the commands are still loaded when the program is run. *SuperCharge* automatically adapts to different ROMs and command-sets – it spends a moment before each run, searching out the routines it needs.

The compiler only bans the few commands (such as ED and RENUM) which rely on the presence of the interpreter's data-structures. This is because if it had to keep track of program source and the name list it would be little faster than the interpreter.

The other trade-offs are that you can't have more than 32,767 elements in any 'slice' of an array. Strings and integers must always be marked with dollar or per cent signs, and only functions may return values – though you can use global variables as usual. The parameters of GO TO, GO SUB and DATA must be constants (not expressions).

These trade-offs are needed to make *SuperCharge* compact and fast. If you make a mistake the compiler prints a message in plain English, showing the exact point of your error. You can have a full compilation listing, on any device, if you wish.

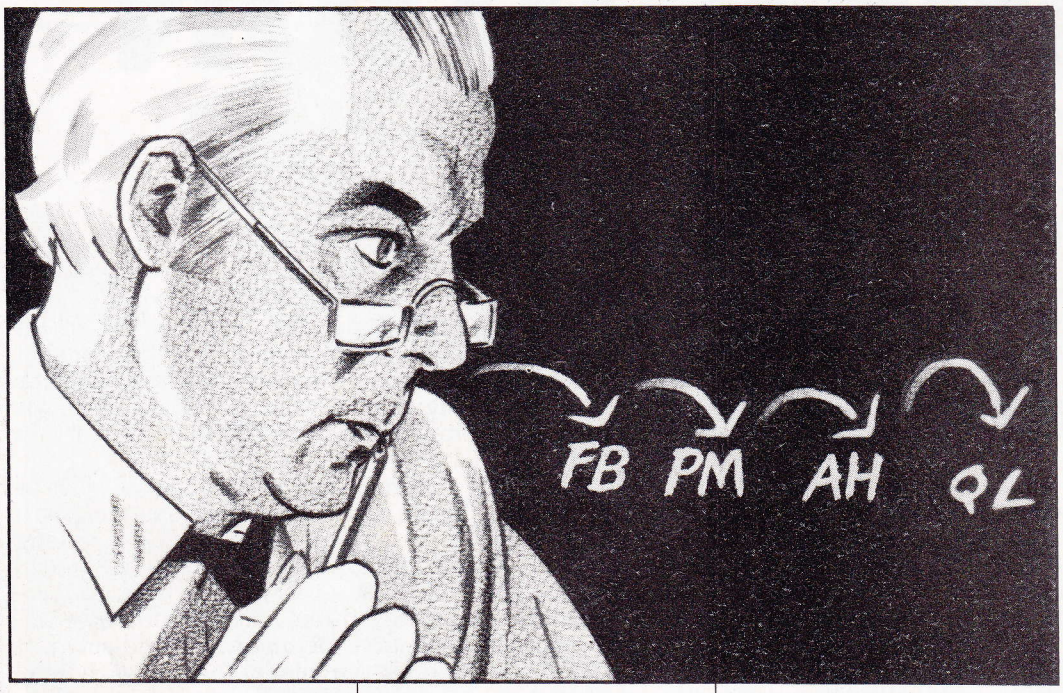
A detailed manual of almost 100 A4 pages accompanies the compiler. It includes notes on the trade-offs, and advice on how to get the best from the product.

Preliminary timings indicate that compiled programs will run on average nine times faster than conventional SuperBasic programs. This increases fourfold when floating point arithmetic is excluded.

Readers should note that because of the exciting possibilities opened up by a SuperBasic compiler, *QL User* felt it important to preview *SuperCharge* before its official launch. For this reason we have allowed the product's designer, Simon Goodwin, to present the article. Benchmarks and a full review will appear in a subsequent issue of *QL User* when production versions are made available.

ROMBLINGS

A Historical Perspective



Alan Turnbull's investigations reveal that the Quantum Leap was not a single bound but a series of short hops

In January 1984, the so-called Quantum Leap was made by Sinclair Research – the QL was launched with 128K RAM on board, two Microdrives built-in, a bundle of four comprehensive software packages included and the facility to multi-task machine code programs using user-defined screen windows.

It was obvious that with so much to offer for £400, the QL was an ambitious project and it was unfortunate that it was launched with so much media hype in January 1984 when it wasn't ready for despatch until six months later!

Naturally, machines were rushed-out to try and meet each deadline and this generated a string of Read-Only Memory (ROM) releases as each batch of QLs was made slightly better than the previous.

The ROMs are actually named after Sinclair Research engineers and the two-letter codes can be examined by typing PRINT VER$ at the QL's keyboard. The firmware has so far been through four metamorphisms – the first release was called *FB* which presumably meant 'Full of Bugs'. The line editor would not let you edit a line it had flagged as 'bad' – you had to type it all in again! Most of the commands didn't work and all you could do with 'FB' was to play and dream about what it should be like.

Next came a hurredly botched version – *PM* – which Sinclair Research probably hoped would stay in office for at least five months if not five years! The line editor was better but the Psion packages were still taking minutes rather than seconds to load. And then, they rarely worked.

But after that, Sinclair Research released what it described as the final version of the QL's firmware – *AH* – which certainly brought a sigh of relief! The Psion software would now load in around 30-40 seconds and the machine did not crash so often. It was by no means the final version as I shall soon show you.

Ever true to its word, out came another ROM – *JM* – actually named after John Mathieson and this apparently made Microdrive handling better. One major bug was still present though – the QL would only recognise one rather than 16 plug-in peripheral cards. The code to search for them had been written incorrectly – in such a way as to only test for one card, then give up!

The latest version of the QL's ROM is *JS* which has 25 new keywords added to allow error trapping in conjunction with the construct 'WHEN ERRor' which is now implemented. Sinclair Research insist that these features are provisional and

Listing 1

```
00100 * SuperBASIC extensions to provide:
00110 * ---------------------------------
00120 *
00130 * 1) A new function QDOS$ to return the QDOS release number
00140 * 2) A command CALL to replace the faulty (pre-'JS') ROM-based version.
00150 *
00160 * COPYRIGHT (c) April 1985, Alan Turnbull, B.Sc.
00170 *
00180 *
00190            LEA      EXTENSIONS(PC),A1       ; Point to list of extensions
00200            MOVEA.W  $110,A0                 ; Vector for linking-in extensions
00210            JSR      (A0)                    ; Call routine
00220            RTS                              ; Return to SuperBASIC
00230 *
00240 * Table of extension definitions
00250 *
00260 EXTENSIONS: DC.W    1                       ; Number of procedures
00270            DC.W     CALL-*                  ; Offset of start of routine
00280            DC.B     4                       ; Length of name
00290            DC.B     'CALL'                  ; Name of procedure
00300            DC.W     0                       ; End of procedures
00310            DC.W     1                       ; Number of functions
00320            DC.W     QDOS-*                  ; Offset of start of routine
00330            DC.B     5                       ; Length of name
00340            DC.B     'QDOS$',0               ; Name of function with padding
00350            DC.W     0                       ; End of functions
00360 *
00370 * Run-time modules
00380 *
00390 QDOS:      CMPA.L   A3,A5                   ; Were any parameters provided?
00400            BNE      ERR_BP                  ; Yes, report 'Bad parameter' error
00410            MOVEQ    #6,D1                   ; Reserve 6 bytes on arithmetic stack
00420            MOVEA.W  $11A,A0                 ; Required vector
00430            JSR      (A0)                    ; Call routine
00440            MOVE.L   $0058(A6),A1            ; Get arithmetic stack pointer
00450            SUBQ.W   #6,A1                   ; Make room for string
00460            MOVE.W   #4,$00(A6,A1.L)         ; Put length of string on stack
00470            MOVEQ    #0,D0                   ; Get system information
00480            TRAP     #1                      ; Call routine
00490            MOVE.L   D2,$02(A6,A1.L)         ; Put QDOS release code on stack
00500            MOVE.L   A1,$0058(A6)            ; Reset arithmetic stack pointer
00510            MOVEQ    #1,D4                   ; Signal string result
00520            RTS                              ; Return to caller: successful
00530 ERR_BP:    MOVEQ    #-15,D0                 ; Signal 'Bad parameter'
00540            RTS                              ; Report error
00550 CALL:      MOVEA.W  $118,A0                 ; Vector to get long word integer
00560            JSR      (A0)                    ; Call routine
00570            BNE      CALL_RET                ; Report error if unsuccessful
00580            LSL.L    #2,D3                   ; Get number of long word parameters
00590            BEQ      ERR_BP                  ; Report error if no parameters
00600            ADD.L    D3,$0058(A6)            ; Reset arithmetic stack
00610            MOVE.L   $00(A6,A1.L),-(A7)      ; Put start address of routine on stack
00620            MOVEM.L  $04(A6,A1.L),D1-D7/A0-A5 ; Place parameters in processor registers
00630            MOVEQ    #-15,D0                 ; Preset error flag
00640 CALL_RET:  RTS                              ; Either return with error or jump to CALL
```

Listing 2

```
100 REMark SuperBASIC program to implement assembly language in Listing 1
110 REMark        COPYRIGHT (c) August 1985, Alan Turnbull, B.Sc.
120 :
130 LET reserved_address=RESPR(256)
140 LET address=reserved_address
150 RESTORE
160 REPeat read_and_store_data
170   IF EOF THEN EXIT read_and_store_data
180   READ machine_code_byte
190   POKE address,machine_code_byte
200   LET address=address+1
210 END REPeat read_and_store_data
220 LET rom_ver$=VER$
230 IF rom_ver$(1 TO 2)='FB' OR rom_ver$(1 TO 2)='PM' THEN PRINT "Return QL to Sinclair":STOP
240 IF rom_ver$(1 TO 2)='AH' OR rom_ver$(1 TO 2)='JM' THEN POKE_W reserved_address+156,458
250 IF rom_ver$(1 TO 3)='JSU' THEN POKE_W reserved_address+156,462
260 IF rom_ver$(1 TO 2)<>'JS' AND rom_ver$(1 TO 2)<>'MG' THEN PRINT "New ROM - contact me throug
h QL User":STOP
270 CALL reserved_address
280 STOP
290 :
300 DATA  67, 250,   0,  42,  65, 250,   0, 114,  35,  72
310 DATA   0,   0,  65, 250,   0, 110,  35,  72,   0,   4
320 DATA  65, 250,   0, 102,  35,  72,   0,  20,  65, 250
330 DATA   0,  94,  35,  72,   0,  28, 114,   0, 112,   7
340 DATA  78,  65,  78, 117,   0,   0,   0,   0,   0,   0
350 DATA   0,   0,   0,   0,   0,  94,   0,   0,   0,  94
360 DATA   0,   0,   0,  94,   0,   0,   0,   0,   0,   0
370 DATA   0,  94,   0,   0,   0,   0,   0,   0,   0,  94
380 DATA   0,   0,   0,  94,   0,   0,   0,  94,   0,   0
390 DATA   0,  94,   0,   0,   0,  94,   0,   0,   0,  94
400 DATA   0,   0,   0,  94,   0,   0,   0,  94,   0,   0
410 DATA   0,  94,   0,   0,   0,  94,   0,   0,   0,  94
420 DATA  80, 143,  78, 115,  70, 252,  39,   0,  42, 121
430 DATA   0,   2, 128,  32,  65, 249,   0,   2,   0,   0
440 DATA  32,  60,   0,   0, 132, 127,  66,  24,  81, 200
450 DATA 255, 252,  32, 124,   0,   0,   1, 204,  78, 144
```

liable to change at any time. Anyway, the main bug is corrected so that the QL can now recognise 16 peripheral cards although you still need the as yet unreleased QL Motherboard. Also, the number base conversion so-called 'utility' vectors which were actually not much use at all now work!

Version *JS* (which I am now the proud owner of – I graduated from an *AH*) was released in February 1985 and contains QDOS version 1.10 which follows on from 1.02 and 1.03 and contains an extra entry to **TRAP #$01** with **DO=$24**. This new trap allows the console messages including the error reports to be re-vectored and also allows the QL's character set to be altered for use in foreign countries. In fact the new SuperBASIC command **TRA** which is short for 'translate' simply calls this **TRAP**.

Not only the firmware has changed regularly either. The hardware has had problems – repaired version *AH* QLs now come back with discrete components soldered across connections on the main board and the customised Uncommitted Logic Arrays (ULAs). No doubt there are decoding problems which Sinclair Research has kept quiet about. Recently, Psion has got its act together with the release of proper versions of its bundled software.

All these changes in the state of the QL package have had an effect. The counter assistant in the computer department of my local branch of a well-known chemists told me recently that he was getting people coming in demanding QLs with version D12 hardware, version *JS* firmware and version 2.00 Psion software before they would part with their money! He said this had meant he had to tighten-up the shop's quality control. This freedom of information on Sinclair products perhaps upsets the shops but it is ultimately good news for the customer.

So where does the QL stand now? Actually, it is still not finished and Sinclair Research plan to release version 2.00 of QDOS soon. I deduced this by having a sneak look through the code of *QL Toolkit* – it has a test for the release code of QDOS and an assembly instruction such as:

CMPI.L #'2.00',D2

is a bit of a giveaway!

Version 2.00 of QDOS will have two extra entries to **TRAP #$03** – with **DO=$4A** and **DO=$4B**. The first one will allow the renaming of a directory based file and the second will truncate a file by chopping off the portion between the file pointer and the end of file. Also, certain **TRAP #$02** operations are upgraded – **TRAP#$02** with **DO=$01** and **D3=$03** will open a file for overwriting as promised in the QDOS Manual and **TRAP #$02** with **DO=$02** (the *CLOSE* operation) will 'datestamp' a file by making an entry in the update date in the file's header as also indicated in the QDOS Manual.

Listing 1 shows an assembly language program to provide a new SuperBASIC function – QDOS$ – which returns the current QDOS release number as a 4-character string in the format 'n.nn'. Also, a correction of the infamous CALL bug is provided by simply designing a SuperBASIC extension with the same name as the command in ROM – the one in RAM gets linked-in to the system later and replaces the ROM definition. This method can be used to re-define all the keywords in the ROM if you so wish!

Listing 2 provides a SuperBASIC program to implement the code in *Listing 1*.

In early versions (pre-version 1.10 QDOS) of the QL, the CALL command will fail when used from within large SuperBASIC programs because word rather than long word addressing is used in indexes. My bug correction simply copies the ROM definition but changes the **.W** indexes to **.L**.

Listing 3 shows a useful table of addresses. The run-time module address of each keyword in the three main QL releases is listed to aid your understanding of your particular version of the QL.

Meanwhile, keep your eyes peeled for version 2.00 of QDOS.

Perhaps the two letter code returned from VER$ will be *FF* – Finally Finished!

Listing 3

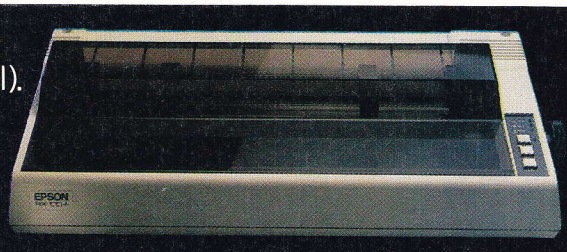
Keyword Run-time module addresses for each main ROM release.
COPYRIGHT (c) April 1985, Alan Turnbull, B.Sc.

Keyword	'AH' ROM (QDOS v1.02)	'JM' ROM (QDOS v1.03)	'JS' ROM (QDOS v1.10)
PRINT	28586	28662	30376
RUN	30232	30322	32164
STOP	30334	30424	32266
INPUT	28584	28660	30374
WINDOW	30646	30736	32638
BORDER	30684	30774	32676
INK	28364	28440	30154
STRIP	28368	28444	30158
PAPER	28372	28448	30162
BLOCK	30660	30750	32652
PAN	28406	28482	30196
SCROLL	28410	28486	30200
CSIZE	24756	24828	26274
FLASH	26026	26098	27564
UNDER	26020	26092	27558
OVER	26048	26120	27586
CURSOR	24792	24864	26310
AT	24806	24878	26324
SCALE	26100	26172	27638
POINT	26118	26190	27656
LINE	26136	26208	27674
ELLIPSE	26160	26232	27698
CIRCLE	26160	26232	27698
ARC	26240	26312	27778
POINT_R	26122	26194	27660
TURN	30416	30506	32408
TURNTO	30408	30498	32400
PENUP	30474	30564	32466
PENDOWN	30478	30568	32470
MOVE	30492	30582	32484
LIST	28036	28112	29824
OPEN	25926	25998	27464
CLOSE	25892	25764	27430
FORMAT	25714	25786	27252
COPY	25740	25812	27278
COPY_N	25744	25816	27282
DELETE	25570	25642	27110
DIR	25576	25648	27116
EXEC	25246	25318	26764

EXEC_W	25250	25322	26768	LOG10	30908	30998	32900
LBYTES	25360	25432	26886	SIN	30914	31004	32906
SEXEC	25414	25486	26940	SQRT	30920	31010	32912
SBYTES	25418	25490	26944	TAN	30926	31016	32918
SAVE	25964	26036	27500	DEG	30932	31022	32924
MERGE	30270	30360	32202	RAD	30938	31028	32930
MRUN	30280	30370	32212	RND	31010	31100	33002
LOAD	30312	30402	32244	INT	31110	31200	33108
LRUN	30318	30408	32250	ABS	30970	31060	32962
NEW	30330	30420	32262	PI	31096	31186	33090
CLEAR	30220	30310	32152	PEEK	31134	31224	33132
OPEN_IN	25930	26002	27468	PEEK_W	31142	31232	33140
OPEN_NEW	25934	26006	27472	PEEK_L	31152	31242	33150
CLS	28402	28478	30192	RESPR	31186	31276	33186
CALL	24540	24612	26058	EOF	31220	31310	33224
RECOL	29536	29626	31436	INKEY$	31274	31364	33294
RANDOMISE	29318	29408	31128	CHR$	31360	31450	33382
PAUSE	28490	28566	30280	CODE	31476	31570	33502
POKE	28526	28602	30316	KEYROW	31614	31708	33646
POKE_W	28534	28610	30324	BEEPING	31208	31298	33208
POKE_L	28540	28616	30330	LEN	31456	31548	33480
BAUD	24308	24380	25826	DIMN	31516	31610	33542
BEEP	24368	24440	25886	DAY$	31690	31784	33722
CONTINUE	30404	30494	32336	DATE	31596	31690	33622
RETRY	30394	30484	32326	DATE$	31684	31778	33716
READ	25200	25272	26718	FILL$	31378	31468	33400
NET	28336	28412	30126	VER$	31258	31348	33266
MODE	28308	28384	30096	ERR_NC	-	-	33808
RENUM	29628	29714	31524	ERR_NJ	-	-	33806
DLINE	28006	28082	29794	ERR_OM	-	-	33804
SDATE	25006	25078	26524	ERR_OR	-	-	33802
ADATE	24986	25058	26504	ERR_BO	-	-	33800
LINE_R	26140	26212	27678	ERR_NO	-	-	33798
ELLIPSE_R	26164	26236	27702	ERR_NF	-	-	33796
CIRCLE_R	26164	26236	27702	ERR_EX	-	-	33794
ARC_R	26244	26316	27782	ERR_IU	-	-	33792
AUTO	29582	29672	31482	ERR_EF	-	-	33790
EDIT	29578	29668	31478	ERR_DF	-	-	33788
FILL	25990	26062	27528	ERR_BN	-	-	33786
WIDTH	30624	30714	32616	ERR_TE	-	-	33784
REPORT	-	-	32120	ERR_FF	-	-	33782
TRA	-	-	32344	ERR_BP	-	-	33780
ACOS	30860	30950	32852	ERR_FE	-	-	33778
ACOT	30866	30956	32858	ERR_XP	-	-	33776
ASIN	30872	30962	32864	ERR_OV	-	-	33774
ATAN	30878	30968	32870	ERR_NI	-	-	33772
COS	30884	30974	32876	ERR_RO	-	-	33770
COT	30890	30980	32882	ERR_BL	-	-	33768
EXP	30896	30986	32888	ERNUM	-	-	33882
LN	30902	30992	32894	ERLIN	-	-	33896

With an Epson RX100+ this would never have hap

Up to 233 characters per line; NLQ (optional); 100 cps;
sheet feeder (optional); Parallel (standard); Serial and IEEE (optional).



RX100+. Fast. Very good print. Very, very, very wide. £399 (+VAT).

EPSON

MEDIC. The Definitive

NOW–what the Macintosh does in monochrome – the QL does in colour!

M-PAINT — an amazing new program from MEDIC gives you total freedom to draw, design, sketch or paint. You can create any image, from a freehand scribble to flowcharts and technical drawings. Thousands of different effects, stipples, shadings, using a full palette of options, cursor-key or mouse-driven. Runs on the standard (unexpanded) Sinclair QL. Easy permanent print-outs. M-PAINT brings you complete design freedom – in brilliant colour. Order your copy now for immediate delivery.

New M-Paint from Medic
£30.00 inc. VAT (Plus £1 p&p)

(Please write clearly)

To: Medic Datasystems Limited,
Hackwood Lane, Cliddesden, Hants RG25 2NH

Please send me ☐ copies M-PAINT colour graphics program on microdrive at £30.00 ea. inc. VAT, plus £1.00 p&p.
I enclose cheque/P.O. value £ ..... (add £1.00 for p&p, irrespective of number of programs ordered).

NAME .....

ADDRESS .....

.....

TELEPHONE Code..... No .....

MEDIC offer you immediate delivery of a complete expansion system for your QL. Choose the amount of extra memory you need, plus options of disk interface with parallel port and printer spooler, and modem – all contained simultaneously inside the MEDIC single-unit systemcartridge.

The MEDIC system unleashes the full power and performance of your Sinclair QL for business.

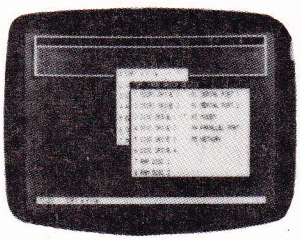
Extra memory, for speed and easier program and file handling, starts at £169.95. Complete systems, consisting of 1 Mb disk drive, disk interface, parallel port, integral power supply and all cables, from £249.95. Even higher performance can be achieved by adding extra MEDIC memory to your disk system, utilising RAM-disk procedures available with MEDIC free software.

You can upgrade your systemcartridge at any time, adding up to 512k extra memory, modem, and up to 4 disk drives.

MEDIC – the reliable QL expansion system that *guarantees* you more than any other system.

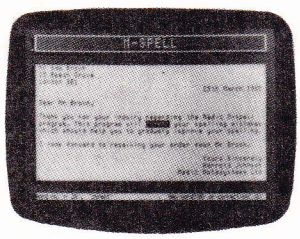
FREE SOFTWARE

Nine performance-raising programs, free with every MEDIC disk drive system.
(Extra memory recommended.)

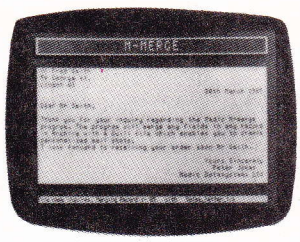


M-DESK

M-TRANSFER



M-SPELL



M-MERGE

M-DESK Macintosh-type single-key depression user interface, for instant program switching and utilities selection **M-BASE** turns Archive into a menu-driven database with single-key commands **M-ACCOUNTS** fully integrated sales, purchase, nominal ledgers, and stock control **M-KEY** single-key entry of user-defined text in any program **M-SPELL** spelling checker **M-MERGE** personalised mailshots **M-SQUEEZE** file compression **M-BOOT** sets up RAM disks in memory then automatically loads pre-defined files and programs **M-TRANSFER** microdrive – disk routine.

QL Expansion System.

QL User, August 1985:
"Medic have set out to manufacture the definitive expansion system and have, to a large extent, succeeded."

FREE – microdrive dust covers. Simply fill in and post the coupon

...beats all other systems on speed, on disk capacity, on versatility, on extras ... and on sheer value!

Order form

☐ Please send me the MEDIC brochure and copies of press reviews, free.

To order, please tick the items you require.
All prices include VAT. Add £10 p&p. (£2.95 p&p for orders under £125)
All orders despatched in strict rotation, delivery guaranteed within 28 days.

Extra memory
MEDIC systemcartridge with extra memory
256K £169.95 ☐

Disk interface
(if you already have disk drives) £129.95 ☐

Dust cover
+ 3 adjustable feet £14.95 ☐

Joystick £9.95 ☐

Disk pack
Ten 3½in disks in rigid plastic storage box £60.00 ☐

Modem*
Add to disk interface or complete system price (not sold separately, since connects through systemcartridge) £120.00 ☐
*not shipped until BT approved

Complete systems
1Mb disk drive, MEDIC systemcartridge inc. disk interface, parallel port, integral power supply, all cables
+ FREE SOFTWARE £249.95 ☐

2Mb double disk drive, MEDIC systemcartridge inc. disk interface, parallel port, integral power supply, all cables
+ FREE SOFTWARE £399.95 ☐

Complete systems + extra memory
Add to complete system price 64K £50.00 ☐
128K £80.00 ☐
256K £110.00 ☐
512K £200.00 ☐

I enclose cheque/P.O. payable to Medic Datasystems Ltd. in payment for the items ticked above, plus £10 p&p (£2.95 p&p for orders under £125), total £ _____

Signed _____

Name _____

Address _____

_____ Telephone _____

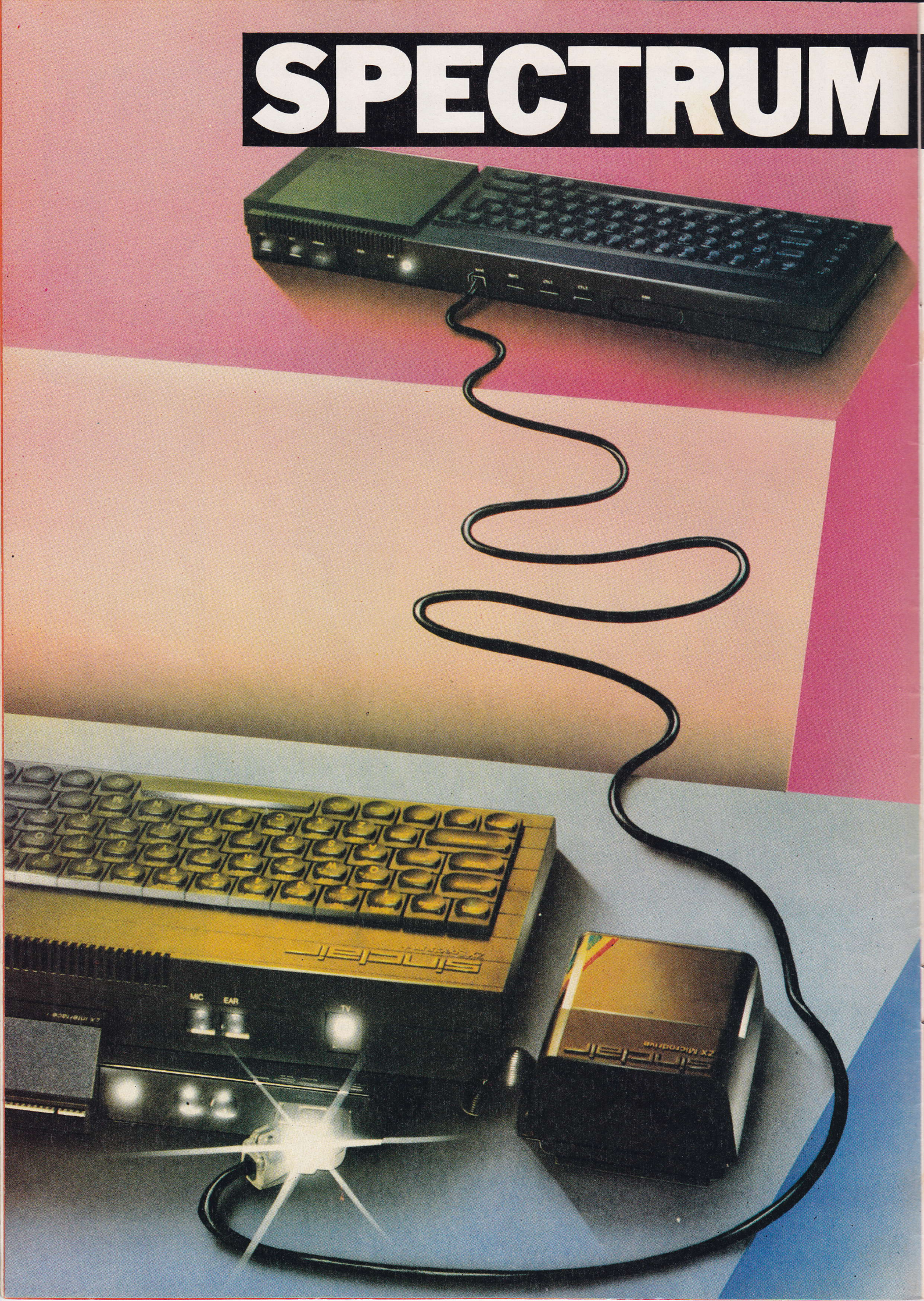
Please write clearly

MEDIC

Medic Datasystems Limited
Hackwood Lane, Cliddesden
Hants RG25 2NH
Telephone 0256 460692

System now available from stock for immediate delivery. Dealer enquiries invited.

SPECTRUM



CONNECTIONS

In the first in a series that links the Spectrum to the QL L A Privett reveals the secrets to a trouble free transfer.

When the QL was launched it was claimed that the machine would "communicate with the Spectrum using the Network facility". An attractive proposition to many thousands of Sinclair users, who having spent many hours programming the Spectrum, were looking for a compatible upgrade over to which they could port their BASIC programs.

Well, there was an element of truth in the claim! The QL can communicate but not via the Network port. The necessary firmware has not yet been implemented (Turkish version of the MG ROM excepted). Simple commands such as **SAVE Neto_1** at the QL end, and **LOAD *"n";0** at the Spectrum end simply do not work. Nor do a variety of possible permutations.

However, in common with many computers the QL does incorporate an RS232C port and at the Spectrum end, a similar port is available, providing you own a Sinclair *Interface 1*. So the idea of transferring Spectrum BASIC programs to the QL, updating them free from any attribute problems and other delights that Superbasic gives us, is still feasible. Indeed, several programs have appeared in various magazines that have shown considerable promise in this area.

Unfortunately, when these programs were typed in and RUN, they had one of two things in common – either parts of the program did not work or data was corrupted in the course of transfer. In the latter case, this invariably meant that the QL would freeze up so that you had to hit RESET button and lose everything. In our case replacing both QL (now with a JM ROM) and Spectrum interface reduced the incidence of corruption but failed to eliminate it altogether. The programs remained unworkable so we decided to go it alone and write our own program (called *Spectrum_bas*)

Although the program (listed below) is designed for one-way communication from Spectrum to QL, such is the nature of the RS232 link that two way communication is possible. This would require a similar program to be written for the Spectrum and would enable one to send programs, data and so on from the QL to a file on the ZX Microdrive. However, as files transferred would be of a DATA type and trying to load them into the Spectrum as programs would fail with the error *wrong file type*, the only benefit to be derived here would be access to cheap and fairly reliable cassette storage. When microdrive cartridges were £5 a shot this might well have been worthwhile. Today when microdrives are equally reliable, very much faster and cheaper in terms of storage capacity such a course of action is no longer justified.

Onto the program itself. It has been designed for use with a JM version of the QL and interface 1 (version 1). We mention this because later versions of the interface have had some slight improvements made regarding the network facility or so we are led to believe. If you wish to know which version you have you enter:

CLOSE#1:PRINT PEEK 23729

directly into the Spectrum, a value of 0 indicates that a version 1 ROM has been installed and anything else refers to a later version. Either way as the RS232 connection has not been altered this program should work without hitch.

Connectors to the RS232 ports at both ends are non-standard. A suitable lead must therefore be made up. Most QL users will have received a free Printer lead from Sinclair. This may be adapted to fit interface 1. Remove the connector at the printer end and replace it with what is known as a 9 pin 'D' type male plug. Wiring up instructions are given below. The colours on the left refer to the cables that come from the new telephone type plug (supplied by Sinclair) that fits the various ports at the back of the QL. The Spectrum numbers are the pin numbers on the D type plug and will need to be soldered into place. An electrician should be able to do this if you do not feel confident enough to risk burnt fingers and blobs of solder all over the living room carpet as previous experience has shown.

ser1	*RS 232*
GREEN	*2*
WHITE	*3*
RED	*4*
BLUE	*5*
BLACK	*7*

Once correctly wired and connected, the lead may be tested as follows:

Step 1: ENTER the command **COPY SER1 TO SCR_** on your QL

Step 2: Run the following program on your Spectrum

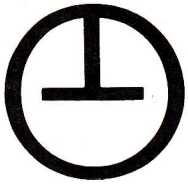
```
10 FORMAT "T";9600
20 OPEN #4;"T"
30 LIST #4
40 CLOSE #4
```

Result: The program given in step 2 will be listed on the QL's screen.

Having satisfied yourself that the lead works you should be in a position to use our *Spectrum_bas* program. This will enable you to transfer your favourite Spectrum BASIC programs to a file on the QL's microdrives and then, at your leisure, correct any incompatibilities between Spectrum BASIC and QL SuperBasic using the QL's built-in editor. Incompatibilites fall into two categories.

First, there are the lines highlighted with the keyword MIStake at their start. Easily identified, these are errors where a command is the same in both dialects but it's usage differs. Keywords LINE, AT and FORMAT fall into this category.

The second incompatibility is where a command has not been implemented in SuperBasic. Spectrum keywords such as PLOT, LPRINT, MERGE, BIN and DRAW fall into this category. Such errors materialise only when the program is RUN and generate the error message 'bad name' as the QL's interpreter classes them as unde- ▶



fined procedures. Whilst not immediately apparent they are easily rectified by defining your own 'conversion' procedures within the body of your program. For example,

LPRINT "This is just an example"

would be acceptable on the QL provided the following procedure has been added:

```
DEFine procedure LPRINT (text$)
OPEN #7, serlz
PRINT #7, text$
CLOSE #7
END DEFine
```

The procedure for using *spectrum_bas* is as follows:

Step 1: Reset both Spectrum and QL

Step 2: [Spectrum] Load the program you wish to transfer across to the QL.

Step 3: [QL] Load and run (LRUN) spectrum_bas program ensuring that there is a formated cartridge in mdv2_.

Step 4: [Spectrum] On the QL's screen you will be prompted to enter commands to control transmission using interfacel. These commands relate to FORMAT, BAUD rate, Binary and Text transmission channels. They should be typed in on your Spectrum as a single continuous line with each statement separated by a colon.

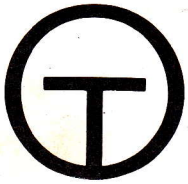
Step 5: [QL] On the QL press the space bar. This will cause mdv2_ to start whirring as the QL readies itself to transfer information recieved via the RS232 port to a file called tempfile opened on mdv2_.

Step 6: [QL] When transmission has ended, you will be prompted to enter an arbitrary filename. When you have done so a permanent file will be opened in that name followed by the postfix *_bas*. For example, if you enter *myfile*, the program creates a file called *myfile_bas*. After this the contents of *tempfile* will be verified and then copied across to *myfile_bas*.

Step 7: [QL] You may now LOAD or RUN the permanent file (ie *myfile_bas*) just as you would any other SuperBasic program. However, for reasons given earlier it is very likely that the program will require debugging using the QL's built-in SuperBasic editor to remove incompatibilities between the two BASICs.

If we examine *spectrum_bas* in detail we see that it breaks down into three main procedures. The first, **Instruct** sets the screen size

```
100 REMark ******************************
110 REMark ***    QL User 1985         ***
120 REMark ** Spectrum_bas by L Privett **
130 REMark ***       L.A.PRIVETT.      ***
140 REMark ******************************
150 :
160 INSTRUCT
170 START
180 CORRECT
190 STOP
200 :
210 DEFine PROCedure INSTRUCT
220 MODE 4:WINDOW 512,220,0,0:BORDER 15,0:CLS:INK
4:CSIZE 2,0
230 PRINT\\'   SPECTRUM BASIC LOADER'\\\
240 PRINT\'PLACE DESIRED CARTRIDGE IN MDV2 SLOT.'
250 PRINT\'PRESS A KEY TO CONTINUE.':PAUSE
260 END DEFine
270 :
280 DEFine PROCedure CORRECT
290 INPUT#0;'PLEASE ENTER NAME FOR PERMANENT FILE
';FILE$
300 OPEN_IN #4,mdv2_TEMPFILE
310 OPEN_NEW #5,'MDV2_'&FILE$&'_BAS'
320 CLS#0
330 AT 4,4:PRINT'Please wait while Mdv2 is accesse
d'\'    It may take awhile ......'
340 LET l=1
350 REPeat TRANSFER
360  IF EOF(#4) THEN EXIT TRANSFER
370  LET SPECTRUM$=INKEY$(#4)
380  LET SPECTRUM=CODE(SPECTRUM$)
390  SELect ON SPECTRUM
400  =32 TO 191
410   PRINT#5,SPECTRUM$;
420  =13
430   PRINT#5,CHR$(10);
440   AT#1,10,4
450   PRINT#1,'Line ';l;' completed '
460   l=l+1
470  =192 TO 255
480   PRINT#5,' ::: ';
490  END SELect
500 END REPeat TRANSFER
510 :
520 PRINT\'    NOW CLOSING FILES'
530 CLOSE#4
540 CLOSE#5
550 pa
560 PRINT\'    NOW DELETING TEMPFILE'\
570 DELETE mdv2_TEMPFILE:pa
580 PRINT\'    PROCEEDURE COMPLETED'
590 pa:CLS:DIR MDV2_
600 END DEFine CORRECT
610 :
620 DEFine PROCedure pa
630 FOR f=1 TO 10:PAUSE 50:NEXT f
640 END DEFine
650 :
660 DEFine PROCedure START
670     CLS
680     CSIZE 1,0:INK 4
690     PRINT'ENTER AS DIRECT COMMANDS ON THE ZX S
PECTRUM.'\\:INK 7
700     PRINT'FORMAT "B";9600'\\
710     PRINT'OPEN#4;"T" : OPEN#5;"B"'\\
720     PRINT'LIST#4 : PRINT#5; CHR$ (26)'\\
730     PRINT'CLOSE#4 : CLOSE#5'\\\:INK 2
740     PRINT'PRESS A KEY WHEN READY.'\\:PAUSE:INK
 4
750     PRINT'NOW OPENING FILE ON MDV2'
760     PRINT'AND COPYING PROGRAM ONTO'\'TEMPFILE.
'\\
770     COPY serlz TO mdv2_TEMPFILE:INK 7
780     pa
790     PRINT 'COPYING FINISHED.'\'PRESS A KEY TO
CONTINUE.'
800     PAUSE : CLS
810 END DEFine
820 :
```



and tells you where to put the cartridge (no rude comments please) that is to contain the program. After this, the **Start** procedure first tells you what commands to enter into the Spectrum and then opens a file (*Tempfile*) on mdv2_. This copies the listing sent by the Spectrum on the serial link to Tempfile. It will close the file automatically when the QL has received a 'End of File' code (CHR$ (26)) or if the QL is stopped using CRTL & SPACE, this is the reason the second channel is opened on the Spectrum to send this code as an EOF is not normally sent on the "T" channel (at least not on ours).

The last procedure **Correct** after asking for a file name, reopens *Tempfile* and transfers its contents to a permanent file. In the course of the repeat loop *transfer* it weeds out newline errors replacing the Spectrum's Line Feed & Carriage Return code (CHR$(13)) with the corresponding code on the QL (CHR$(10)). Other checks could be inserted here such as replacing CHR$(39) (a Spectrum separator) with CHR$(92) (a QL separator).

The procedure **correct** will display which lines are currently being converted (PRINT #1, 'LINE ';1; ' COMPLETED '). Users converting large files should be prepared to wait two or more minutes (depending on size of file) for this to be completed. Once finished, Tempfile is deleted and a directory of the cartridge is produced. Thereafter the permanent file can then be loaded as a Basic program and the other errors or mistakes that might occur can be edited using the Superbasic editor.

Whilst the transfer process adopted in *spectrum_bas* may seem unnecessarily long-winded they do ensure virtually trouble-free conversion. The longest program that we managed to transfer successfully across to the QL was a complex 25K inventory control containing innumerable data statements. The process took two hours including debugging using the QL's editor. Had the program been typed in manually it would have taken at least two days to reproduce it on the QL.

Connectors: D range 9 way plug 68p, D range 9 way protective cover 95p, available from Maplin Electronic Supplies Ltd 0702-552911

Next month Rob Miles explains how to transfer your favourite arcade screens from Spectrum to QL.

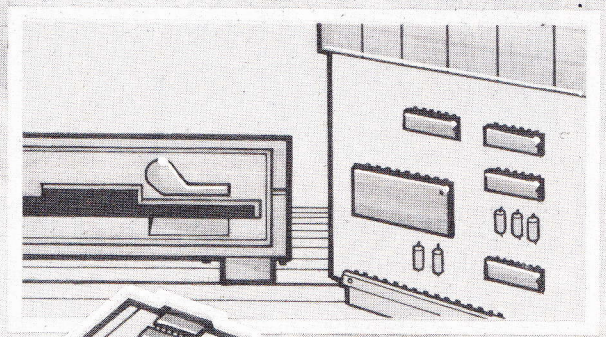
QL Future Growth

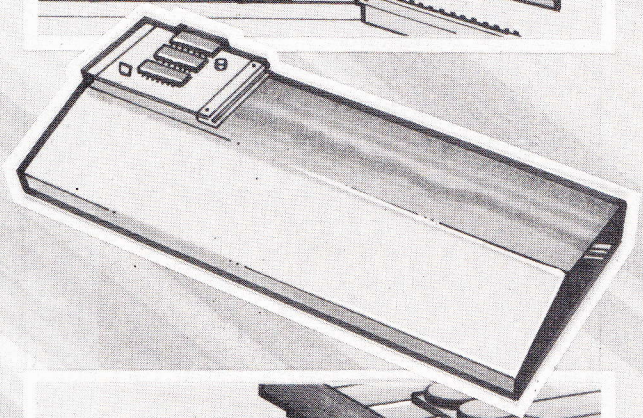
New horizons for your microcomputer from CAMBRIDGE SYSTEMS TECHNOLOGY, the dedicated specialists in expansion peripherals for the Sinclair QL.

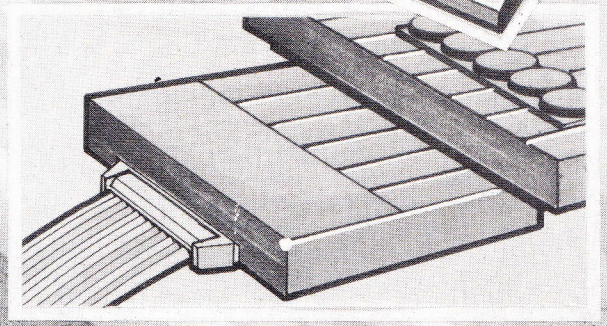
CST who were the first on the market with a disc drive controller, a Centronics port and a fully operational IEEE-488 interface, now offer the Q+4 multi-way expansion module. With four fully-buffered ports, the Q+4 is fully compatible with QL add-ons and features a controller ROM functioning with any version of the QL operating system. Built into a rugged matching case, the Q+4 is designed to sit beneath the computer.

The CST Q-disc is the first controller to allow standard disc drives to be connected to the Sinclair QL, via the QL expansion port. The Q-disc offers extensive file handling and random access facilities plus an essential utility disc and a comprehensive manual.

The Q-488 is a fully implemented IEEE-488 interface which permits the Sinclair QL to communicate with scientific and industrial equipment offering extensive help facilities plus comprehensive error checking.







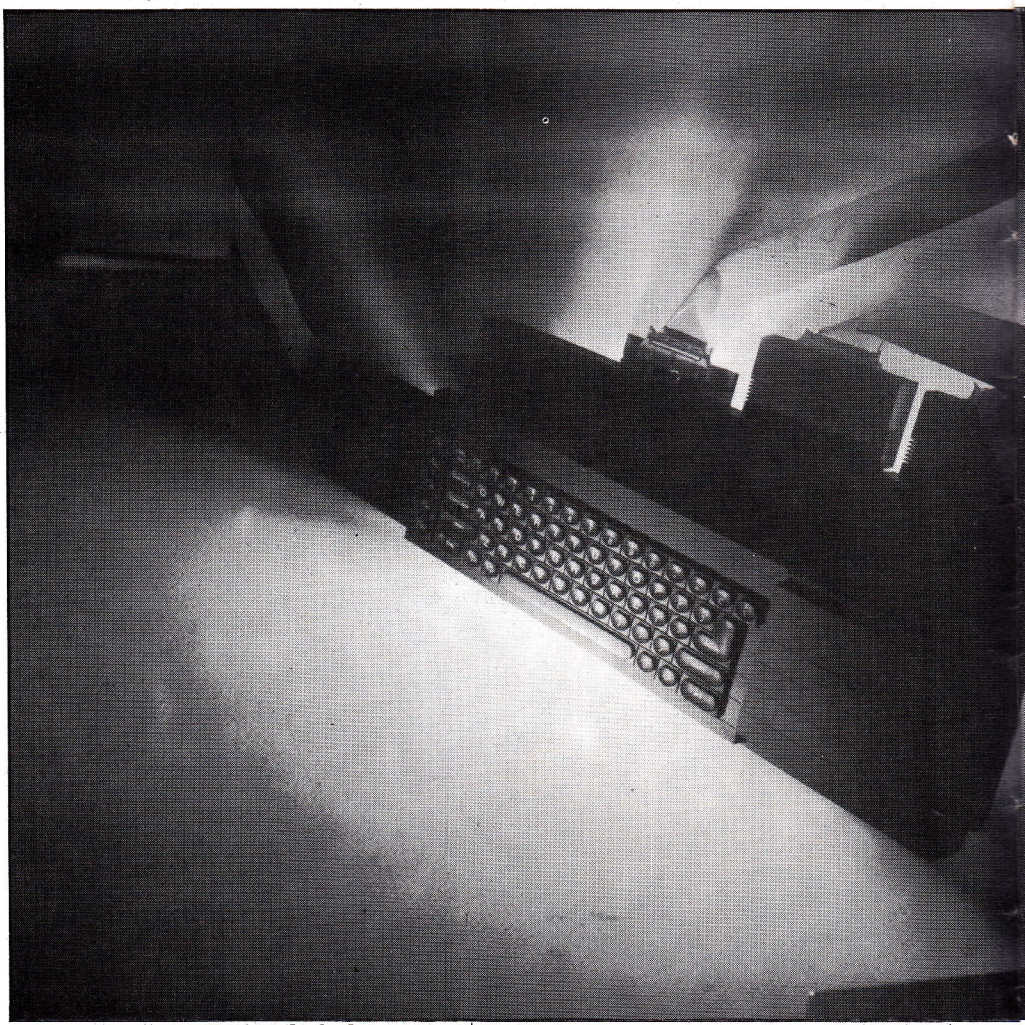
½ Megabyte internal RAM upgrade now available – **ONLY £150**
10 Megabyte Winchester disk now available.
See review in this issue

CST
CAMBRIDGE SYSTEMS TECHNOLOGY

30 Regent Street Cambridge
Telephone: Cambridge (0223) 323302

HARDWARE

Close on the heels of their Q-disc interface and Plus 4 expansion unit, CST have announced the first QL Winchester hard disk system. Paul Hickling previews this technological marvel, as well as Technology Research's disk interface.



Paying around £1,200 for an add-on to a computer that now costs less than £200 may, at first sight, seem more than a little strange. CST took the decision to develop their hard disk drive after market research indicated that some QL owners (at least) would be prepared to part with this sort of money to equip their machine with 10Mb of mass storage. The system demonstrated to us was one of two early production models, housed in an anonymous black box. Final versions will be livened up by the addition of silk screen legends and two LEDs to indicate operating status.

As far as the user is concerned a hard disk drive behaves in the same way as a standard floppy drive; it just offers far more storage. In the case of the basic CST system, some 19,000 sectors provide 10Mb of store (the company will be able to supply custom systems offering storage in excess of this figure).

Hard disk drives use the same magnetic storage systems as employed in floppy disk drives, the much increased storage density is accounted for by the fact that the mechanics of a hard disk unit do not have to allow for the interchangeability of disks. A hard disk, of the type used by CST, is a sealed unit. This allows for greater precision in the construction of the unit and hence the much-improved storage capacity.

The hard disk system is supplied in two parts, the hard disk drive itself and a QL expansion card which houses the system firmware and some additional interface electronics. Although a minimum system could be built around only the QL and the hard disk, it is more than likely that the disk drive will be used as part of a larger system – CST demonstrated the disk drive in conjunction with their Plus 4 expansion system and a floppy disk interface and standard 5¼" disk drives.

Users of floppy disk systems will be familiar with the need to back up data on a regular basis; although not all users are attentive to the need to keep back-up disks. With a hard disk, and up to 10Mb of data on a single disk, the need to back-up is *vital*.

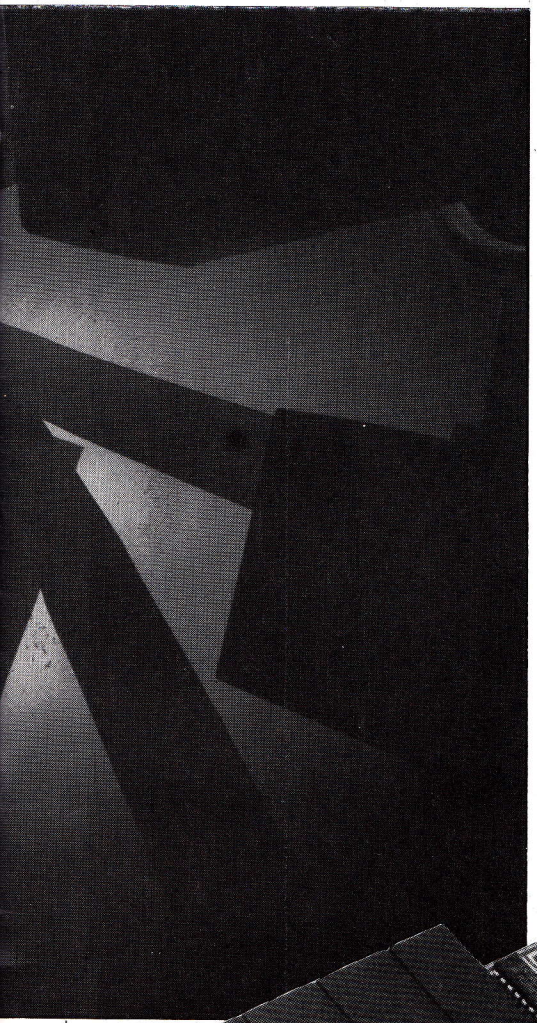
The CST drive is supplied with an intelligent back-up program. This will allow either the entire contents of the disk to be backed up to floppies, or indeed microdrives. This operation, for a disk that held even half its maximum capacity, would be a rather time consuming affair. The system thus provides a facility by which only those files created since the last back-up date will be copied. It is here that the intelligent aspect of the back-up software becomes apparent. When a back-up is initiated, the disk drive first checks the QL clock. If the time read back is arbitrary, the back-up firmware will simply add one second to the time held in its memory and proceed with the process. This avoids the possibility of the clock going back in time.

In addition to the need to make back-ups of the data stored on the disk, the fact that the disk unit is fixed will mean that as its storage capacity is reached, users will have to back-up the data that is not in current use. In most applications though it will take a considerable time before this limitation becomes a problem.

Other statistics relating to the drive are as impressive as its storage capacity. Supporting up to 630 files, the maximum data transfer rate is 1.6uS per byte and means that files are loaded into the QL's memory in times that are significantly faster than those associated with floppy systems.

CST first demonstrated the system at the PCW show and are 'talking' to Sinclair about the possibility of them marketing a badge engineered version of the system.

HORIZONS V



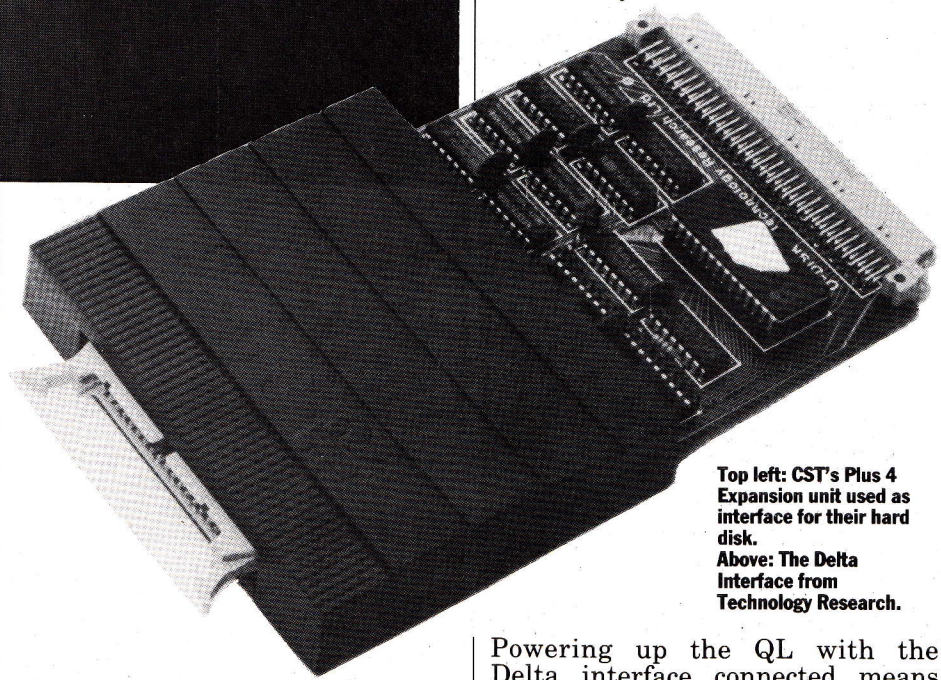
CST are considering enhancing the performance of the system as demonstrated to *QL User* in a number of ways. One of these is the provision of a hierarchical file structure (à la OS9). In view of the number of files catered for within the system this feature would add considerably to its ease of use. However, the CST hard disk system is not likely to appeal to many hobbyist users of the QL, it's more for the business/professional environment.

CST are first to the market with a system that can meet the needs of certain users. Such hardware is not fodder for the mass market, but it will meet a real need for some QL users.

Delta Interface

Technology Research's Delta Interface, could perhaps be thought of as a 'value added' package. The system provides not only a disk interface supporting up to four drives in any of the common sizes (5¼", 3½") or the not so common (3"), but also a Centronics parallel port and a 'RAM disk' driver.

The interface plugs into the expansion slot on the left hand side of the QL, or into an appropriate slot on an expansion system if one is fitted. The manual supplies clear instructions detailing the installation procedure and few users should have any problems in hooking up the Delta system.



Top left: CST's Plus 4 Expansion unit used as interface for their hard disk.
Above: The Delta Interface from Technology Research.

Powering up the QL with the Delta interface connected means both the floppy disks and the RAM disk may be accessed by commands following the standard Sinclair syntax (replacing the mdv device name with either 'flp ' or 'ram').

The firmware supplied with the disk interface has some useful features. The FORMAT utility, for example, is able to detect the physical specification of the drives connected to the interface. An FLP_USE command will allow the default name of the disk system to be altered. The most likely application for this facility is to change the name of the disk system to 'mdv'. This will mean that any program containing calls to the microdrives will now instead look to the disks for the files.

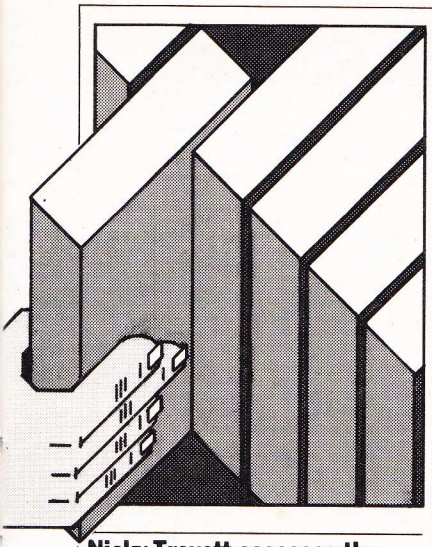
The RAM disk driver supported by the Delta interface allows areas of the QL's memory to be treated in exactly the same way as either microdrives or floppy disks – a RAM disk can recover data far faster than other forms of mass storage as it makes use of semiconductor, rather than the magnetic/mechanical systems.

To create a RAM disk the command FORMAT is used – for example the command FORMAT ram2_80 would create a logical storage device with the name 'ram2' which would provide 80 sectors of storage.

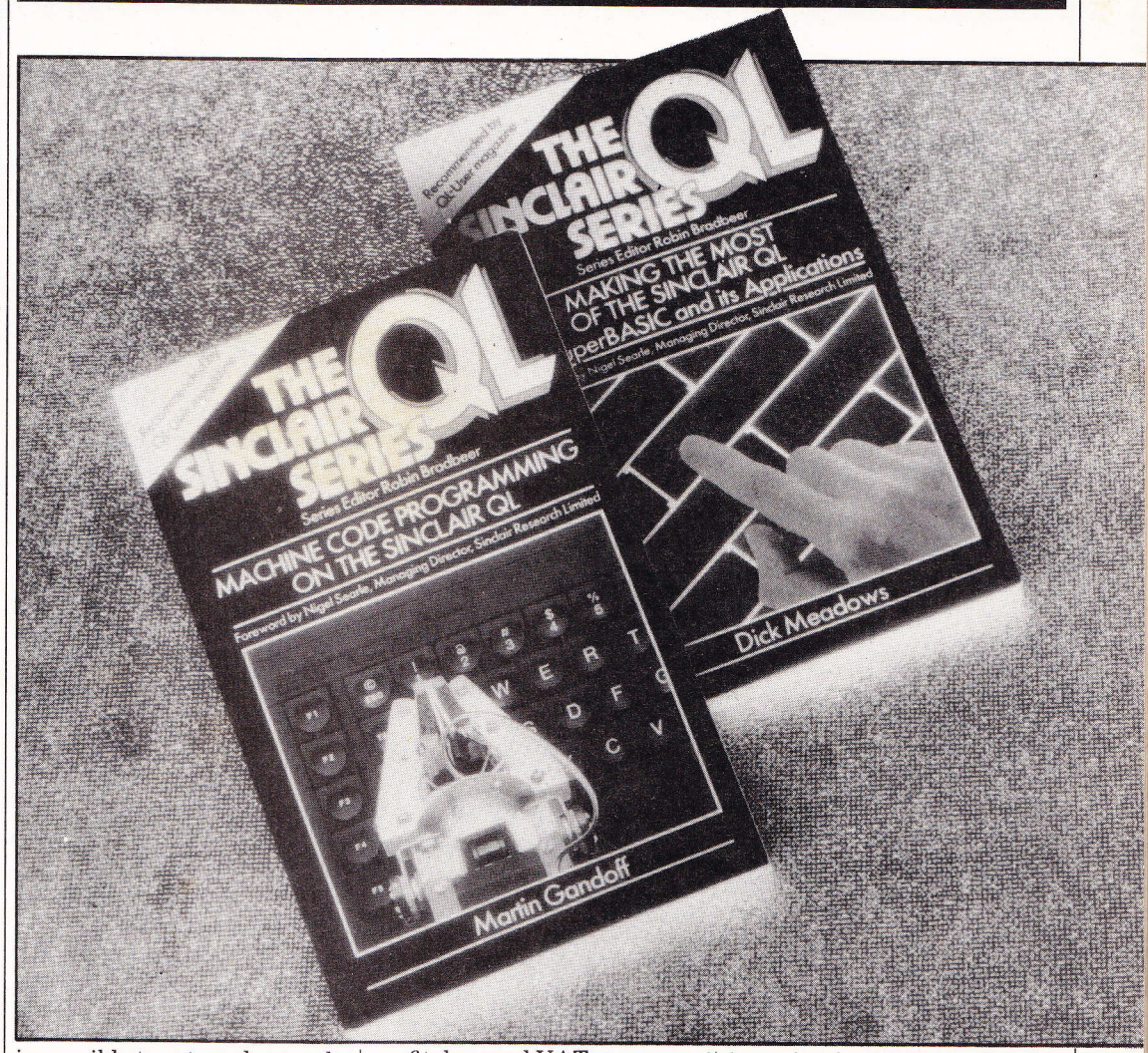
The printer interface that forms part of the Delta board has the device driver name of *par* (device drivers allow all inputs and outputs to be treated in a similar way – the QL's QDOS operating system is not concerned about the physical characteristics of any device connected to it, but relies on device drivers associated with each peripheral to write or read to the logical files manipulated by QDOS. This allows add-ons, such as the Delta interface, to be integrated within the QL's command structure). The default setting associated with it sends a line feed at the end of each line and has a buffer of 80 bytes. While it is only possible to open one print port at any one time, it's possible to have a number of print files pending without typing up the driver. The Delta also allows for up to 128K of extra RAM to be fitted to the QL in 64K byte chunks.

The Delta interface is a versatile unit that supports a well thought out disk interface system together with other features that add to the QL's overall performance. The RAM disk will be of particular interest to any users who run applications software that makes frequent calls to 'disk'. In these cases the RAM disk could considerably increase the system's performance.

BOOKMARKS



Nicky Trevett assesses the latest additions to an acclaimed series.



The thirst for knowledge among QL users is never ending, if the constant stream of new books about the QL is anything to judge by.

In 1984, Hutchinson published its well-regarded Sinclair QL Series, five books covering SuperBasic programming, word processing and introductions to QL computing.

Hutchinson has now launched five more books in the series, edited as before by Robin Bradbeer and all featuring introductions (the same one) written by Nigel Searle, ex-managing director at Sinclair Research. The books deal with graphics, database management, spreadsheets, machine code and SuperBasic programming on the QL, and each costs £7.95.

Déjà Vu

To take perhaps the most unexpected title first, *Making the Most of the QL* is another way of saying Another Book about SuperBasic Programming. As the series already boasts two books about SuperBasic programming, it was difficult at first to see where a third could fit in. In fact, it complements the other two quite well.

Written by Dick Meadows, the book is intended to act as a practical quide to using and programming the QL, claimed to be aimed at both the newcomer and more experienced user.

I wish authors wouldn't do that; the needs of the novice programmer are likely to be quite different from those of the veteran, and it is almost impossible to cater adequately for both in one volume. This particular book I would say is far better suited to the user already familiar with the principles of Basic and SuperBasic.

Its first two chapters offer a competent summary of the essential commands and statements of SuperBasic, with the first chapter including a look at the managing of cartridges, microdrives and printers, and the second covering all the major concepts and techniques. This part of the book could be used as an excellent refresher course in SuperBasic programming and, taken with the comprehensive index, as a useful reference guide.

The remaining five chapters are given over to practical application programming, with numerous examples. One section, for example, provides a series of handy little general-use programs, including converting units and currency exchange, checking bills, working out how to pay off a loan, calculating your income tax, profit, loss and VAT calculations, and so on.

Other chapters deal with graphs and graph plotting, sorting and statistics, science and engineering applications, and solving equations.

The book is lucid and well presented, and should prove ideal reading for anyone who needs to brush up their programming skills and/or put those skills to work.

Low Level Prose

Still on the subject of programming but on a rather different level, Martin Gandoff, who, incidentally, wrote the earlier *Advanced Programming with the Sinclair QL*, has written *Machine Code Programming on the Sinclair QL* for those who would like to take the plunge into this more complex area.

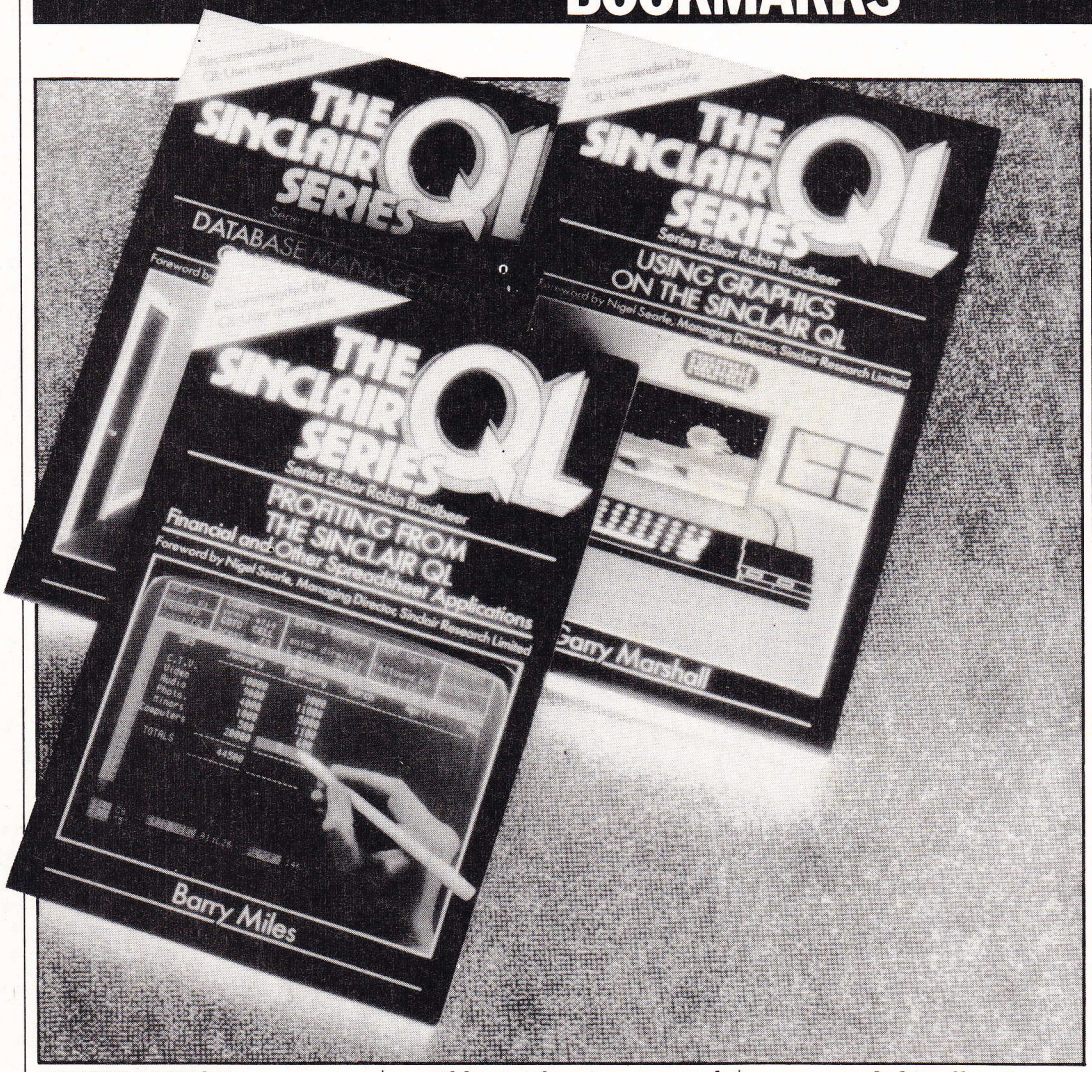
In a long and chatty author's preface, Martin Gandoff ponders the scope of his work, explaining that his aim was to write "a fairly simple book of a reasonable size by missing out some of the difficult bits". This is more or less what he has done, coming up with a readable introduction to machine code, the 68008 chip and QDOS which should enable the reader already familiar with SuperBasic to write machine code programs and sub-routines for applications which are slow or clumsy for a high level language.

The book starts with a simplified look at the workings of the processor which also acts as a painless introduction to 'number systems', moving on to logic operations and machine code format. There is a short chapter on SuperBasic, designed to show in what circumstances you might want to use machine code in a program.

Later chapters deal with 68008 machine code and assembly language, including the Metacomco assembler development kit, addressing modes, instructions, system control, exceptions processing and QDOS, and finally sub-routines, programs and jobs.

The main appendix comprises a summary of 68008 instruction formats, reproduced from the Motorola

BOOKMARKS



M68008 16/32-bit microprocessor programmer's reference manual, and there is a full index.

This is a relaxed and accessible contribution to the growing number of books on the subject of machine code programming, recommended to anyone who wants to get better acquainted with the QL.

In For A Penny . . .

Profiting from the Sinclair QL sounds an irresistible title. In fact, this book by Barry Miles looks at ways of getting the most out of the Abacus package.

It's aimed at both the newcomer to Abacus and the more experienced user, and in this case the approach works quite well. Chapters one to three are given over to introductions to spreadsheets and to the QL, and this bit can be skipped if necessary. Chapters four to twelve work through Abacus itself, and finally there is a lengthy section looking at practical applications – the bit that justifies the title!

Barry Miles adopts a sensible step-by-step approach to Abacus which looks at the package much as a user would see it, starting with loading and moving on to describe what the user should expect to see on the screen. There are actual 'screen shots' to illustrate the text, particularly useful for newcomers.

There should be something for everyone in the final chapter on applications, which include 'general' applications like cost volume profit analysis, and more specialized users, like calculating your cricket batting average! The earlier applications are explained thoroughly, the later ones less so, on the assumption that by the time you reach them you know what you are doing. Other applications that should be of interest to spreadsheet users are depreciation calculations, capital allowances and investment appraisals. All are well illustrated with printer dumps.

Like most of the books in the series, the text is well organized and clearly written, and the book can be recommended to all serious Abacus users.

Practical Primer

As the author of *Database Management on the Sinclair QL* points out, the problem with any powerful database packages is the fact that they are seen as difficult to use by people without extensive programming experience, Author Mike O'Reilly sets out to put the record straight.

He describes his book as a practical primer on using Archive, aimed at QL users who have either found the program hard to get to grips with, or who might not appreciate Archive's full potential. He insists right from the start that "only a smattering" of programming know-how is required to exploit the program.

The first chapter is an introduction to databases in general, explaining the jargon and the basic principles, followed by several chapters on creating an Archive database. Like Barry Miles' book on spreadsheets, a careful step-by-step approach is taken with plenty of pictures of the screen, designed to show exactly what the user should expect to see at each stage.

The rest of the book deals with programming in Archive, the difficult bit, although it turns out not to be so difficult after all. The book assumes very little programming knowledge, although if you already understand SuperBasic procedures you'll find this section easier still. Topics like printing, nested statements, menus, screen design, dynamic screen displays and multiple files are covered in this part of the book.

There is, as usual, a good index, and also a glossary of database and computing terms.

Picture Book

To round off the series, a book devoted to graphics on the QL. Garry Marshall's *Using Graphics on the Sinclair QL* is aimed at both new and experienced users, and sets out to show how graphics can be created on the QL, whether via SuperBasic or by using Easel. This is an extremely sensible way of dealing with the question of graphics, not always adopted by books on Easel which often ignore the role which can be played by SuperBasic.

The book starts with a discussion on the applications of graphics and some thoughts on when you should use SuperBasic, and when Easel is more appropriate. There's a 'Getting started' section, followed by a look at the workings of Easel, including the way it communicates with Abacus and Archive.

Most of the book, however, is given over to graphics creaded using SuperBasic. This includes such matters as line graphics, windows, turtle graphics, 3D graphics, even a section on problem-solving – like finding the way through a maze.

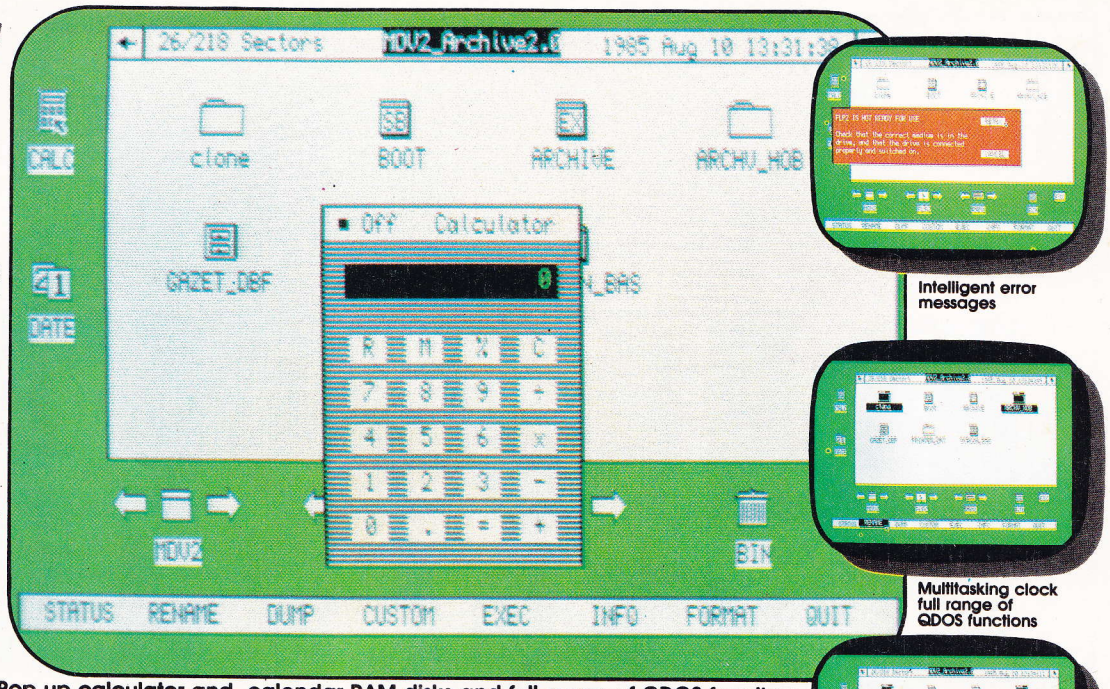
At the back, there's a small but useful chapter summarizing the graphics facilities provided by the QL. There's also the same glossary that appears in *Database Management on the QL* – which, incidentally, contains a particularly apt definition of IBM (Incredibly Big Multinational company!).

ICON

ICE

CONTROLLED

ENVIRONMENT

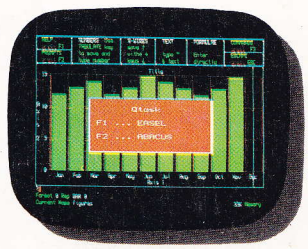


Pop up calculator and calendar RAM disks and full range of QDOS functions

Intelligent error messages

Multitasking clock full range of QDOS functions

Folders allow action on multiple files



Multi-task a number of programs (memory permitting) using CHOice tasking software with mailmerge

ICE is a Fully Icon Based Desktop Manager and Front End for QDOS

ICE makes all of the day to day tedious computer housekeeping (eg. deleting and copying files) you need to do simplicity itself. By using pictures (Icons) to represent the functions you want and the files on your disk or microdrive, all you have to do is point and click, instead of typing reams of mysterious commands. ICE is extremely powerful and can take action on many files at the same time, saving you hours each week! ICE gives you an inbuilt multitasking clock, calendar and calculator for up to the minute efficiency.

ICE also includes the powerful CHOice tasking software that allows you to run several programs simultaneously!

ICE comes on EPROM cartridge that plugs into the cartridge ROM port at the rear of the QL.

ICE uses next to no user RAM! ICE is always available to the user instantly no need to load from microdrive or disk ICE is fully compatible with the Psion™ applications and most QL commercial software ICE is fully compatible with Superbasic programs and can be called from Superbasic ICE is fully compatible with microdrives, disk drives and RAM disks ICE can be used in conjunction with the cursor keys, a joystick or a mouse ICE is multitasking! ICE is only £49.95.

QSPELL

Qspell is a proof reader and spelling checker for QUILL™ with many advanced features.
★26000 word dictionary including 1000 user words."
★Create your own dictionary
★Multitasking Quill Editor."

★Puzzle section allows you to solve crosswords, anagrams and puzzles
★Features fast machine code sort routines
★Fully Menu Driven with Help files
Amazing Value at **£19.95**

QL JOYSTICKS

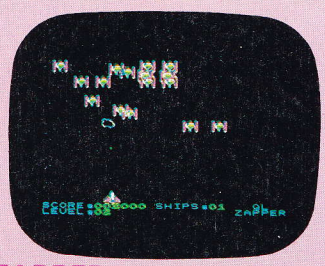
Ideal for ICE or QL Games and Graphics programs
£2 Discount if purchased with program
QL SURESHOT Fully Micro switched. Now only **£19.95**
QL QUICKSHOT Ideal for Games **£13.95**

Designed by John Knight Publicity Ltd, London.

ARCHIVER £18.95

Archiver is a collection of business programs for Psion ARCHIVE™.
★Invoicing ★Stock Control
★Mailing ★Appointments
Archiver lets you use the real power of the Archive Database
You may change the Archiver programs to your own taste easily
Archiver comes with a 48 page user and technical manual.
Please write or telephone for full details
NOW WITH MAILMERGE

★QL ARCHIVE BOOK
by Ian Muarry (356 pages)
Archive made simple! An ideal companion text to ARCHIVER Only **£14.95** from Eidersoft



ZAPPER 68008 Arcade Action

Zap your way through scores of aliens in this multi-level 100% machine code QL arcade game
TV/Monitor
Cursor/Joystick **£10.95**

EIDERSOFT 0708 852647 Telex 8951807
Trade Enquiries Welcome
Psion, Quill and Archive are trademarks of Psion Ltd.

CREDIT CARDS 0708 852647 8am – 8pm
EIDERSOFT THE OFFICE, HALL FARM, NORTH OCKENDON, UPMINSTER, ESSEX RM14 3QH.
Please supply the following items:-

ICE @ £49.95 each ..... + £1.50 P&P
QSPELL @ £19.95 each ..... + £1.00 P&P
ZAPPER @ £10.95 each ..... + £1.00 P&P
ARCHIVER @ £16.95 each ..... + £1.50 P&P
QL SUPERSHOT JOYSTICK @ £19.95 each ..... + £1.50 P&P
QL QUICKSHOT JOYSTICK @ £13.95 each ..... + £1.50 P&P

Name ..... Postcode .....
Address .....
.....
.....

☐ tick if you require disk drive system information
☐ tick if you require ARCHIVER information
I enclose a cheque/PO/card no. for £ .....

PIRATES!

Haul up the Jolly Roger and be prepared to walk the gang plank as Sinclair User does an exposé on the software pirates.

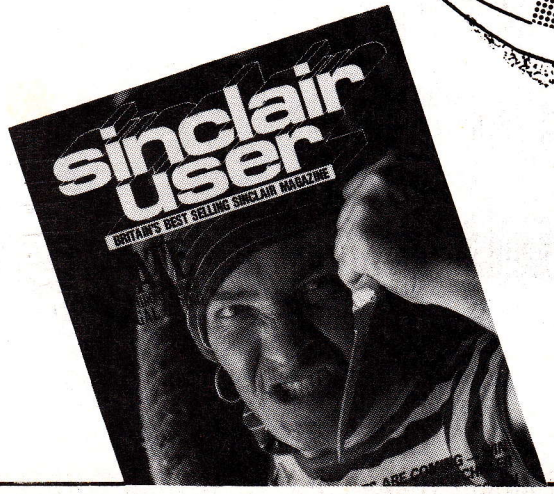
We investigate the cut-throat world of illegal taping, and ask why attempts to prevent it have been so unsuccessful.

Plus that old salty seadog himself, Popeye: A swashbuckling games review.

And the man whose caused more cutlasses to be drawn in anger than Captain Blood; **Gremlin** — with another intriguing insight into the spoils of the computer industry.

All in October's issue of Sinclair User.

On sale **September 18th.** At all newsagents. You'll be washed up without it.



sinclair user

Britain's best selling computer magazine

PRESENTED
FREE WITH

QL
USER

OWNER'S MANUAL

No.1

COMPUTER ANATOMY

The internal electronics inside most computers is beyond the average hacker. A sound knowledge of the basic building blocks isn't . . .

A machine's firmware is software built into the computer which is available as soon as the machine is switched on. With the QL, this means the SuperBasic interpreter, the QDOS operating system, and 'device drivers' to control the screen, serial ports and mdv's.

Firmware is stored inside the QL on a special kind of memory device called a **ROM**. This stands for 'Read-Only Memory'. Once it has been programmed with the software, it holds it internally forever, and even switching off the machine will not remove it. Perhaps this is why software in this form is known as 'firmware' – it is firmly in place!

When you switch your QL on there is a noticeable delay before the microdrives whirr. This is when QDOS takes stock. It goes around the machine, seeing how much memory is available, what peripheral devices are attached and what built-in routines the SuperBasic interpreter is to know about. This process is known as initialisation and once complete QDOS then tries to read a BASIC program called 'BOOT' from microdrive 1 (this isn't strictly true, as we'll see later). From this point on, the machine is in the hands of the SuperBasic interpreter, and by flashing a cursor at us on the screen indicates its presence.

This interpreter has a special significance to the QL. Most other machines of the same calibre as the QL do not enter their version of the BASIC interpreter quite so readily – they have to be told. Instead, they load and run a program known as a *command interpreter*, also known as a *shell* or a *console command processor*. This command interpreter accepts commands typed in by the user and acts upon them, ie, loading a Basic interpreter, a word-processor, or perhaps formatting a disk.

The QL does not have a true command interpreter as the presence of the SuperBasic interpreter makes it unnecessary. SuperBasic is provided with so many built-in procedures and functions that it can very nearly emulate a more usual command interpreter simply by invoking its built-in routines. For example, if we wanted to run a program,

EXEC my_program

and if we wanted to format a microdrive cartridge, we'd type

FORMAT MDV1_myvol

These *command lines* are in fact simply SuperBasic commands. In no way do they give direct access to QDOS. Each command line we type is passed to the SuperBasic interpreter, and if the command at the start of the line is recognised, the interpreter calls the routine which deals with that command. It may well be that such a command routine in turn invokes a particular aspect of QDOS, but that is not implied.

Why bother making this distinction? Simply because it highlights the fact that control over your QL is one step removed. The SuperBasic interpreter acts as the 'middle man'. You communicate with it and it communicates with QDOS. It's a marriage of convenience in so far as SuperBasic is more forgiving but does have its limitations as we shall see.

THE SUPERBASIC INTERPRETER

The SuperBasic interpreter is a program built into the QL's ROM which accepts statements and commands and acts upon them. If each command is preceded by a decimal number within a given range, the number is treated as a 'line number', and the commands are stored in main memory until invoked by the RUN command. Such a group of numbered and stored statements is known as a 'program', so the SuperBasic interpreter is a program which is capable of running other programs.

These programs must follow certain rules if they are to be executed successfully, and these rules form the definition of the SuperBasic language. In much the same way that sentences in English must follow certain pre-defined grammatical rules before they may be universally understood, so each line of a SuperBasic program must follow the rules set down by the designers of the language. Unlike humans, who often are able to discern the meaning of a sentence from its context and general form, the SuperBasic interpreter has no such intuition, and its programs must exactly follow these rules.

When the interpreter encounters a statement (or a line containing statements), it determines what type of statement it is and acts accordingly. In SuperBasic, there are basically four statement types. These are the Procedure call, the Assignment, the Declaration and the Flow of control statement.

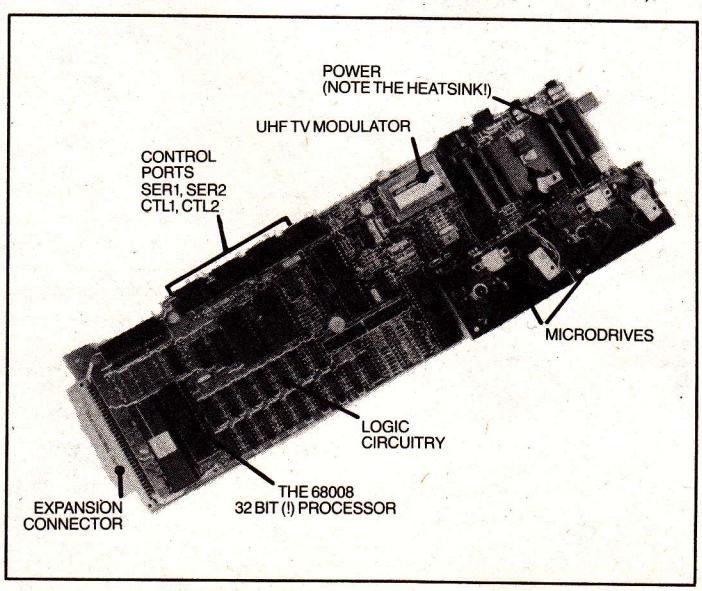
The *procedure call* type of statement occurs whenever the start of a statement consists of a name which the interpreter recognises as a procedure. This procedure may already exist, such as *PRINT*, *CLS* and *LIST*, or it has to be created using the DEFine PROCedure statement. A procedure call causes the code or lines of SuperBasic contained within the relevant procedure to be executed, perhaps with 'parameters' passed to it from the point at which it was called. After this control returns to the point immediately following the procedure call statement. A typical example is

100 PRINT "This is a procedure call"

In this example, *PRINT* is the procedure and the string "This is a procedure call" is its parameter.

An *assignment* occurs when a statement places a value into a variable. The value to be placed in the variable is derived from the *expression* following the '=' sign (which indicates assignment). This expression can be as simple as a number, such as 0 (zero), or it may be as arbitrarily complicated as the application warrants. The expression may involve mathematical operators, like '+' (plus) and '*' (multiply), and it may include *function calls*, which are very similar to procedure calls but return a value. The value of the expression also has a *type*, which indicates how the result is to be treated. If we were assigning a value to a normal variable, the type of the expression would be *floating point*. If we were assigning to a string variable, the type of the expression would be string.

Although the SuperBasic interpreter is fairly lax about these types, it obviously makes little sense to assign a string such as "Silly!" to a normal



variable. Although such a statement would be accepted, as it conforms to the rules of the language, it would cause an error ('Error in expression') when it is executed. A few typical assignments are

```
100  LET a=4
110  x$=a$ & "Another string"
120  a=myfunc (12) -2 DIV 4
```

Notice that the *LET* keyword, which formally introduced an assignment, is entirely optional.

With 'block structured' languages such as C and Pascal the *declaration* statement is a distinct feature. It is used to bring a variable, procedure or function into existence and determines its type, size or parameters. It is quite separate from the assignment statement and without it there can be no assignment. In SuperBasic whilst this is true of procedures and functions, where variable declarations are concerned it applies only to arrays (the DIM statement). With other variables assignment and declaration are one and the same thing so that when we assign a value

```
new_variable = 10
```

even if the variable did not previously exist, it would now.

Additionally, where procedures and functions are concerned the declaration statement marks the start of their definition within the SuperBasic program. So the line

```
1000 DEFine PROCedure MYPROC(a,b)
```

declares *MYPROC* as being a procedure with two parameters (a and b) and also marks the beginning of its definition. The definition consists of all subsequent lines up to the next *END DEFine* statement. The purpose of a declaration is to tell the rest of the program what is meant by a particular word, in this case *MYPROC*, and the definition tells the interpreter what to do when it encounters the word. Typical declaration statements are shown below

```
100  DIM a$(20,3),b(10)
10000  DEFine FuNction SILLY(a)
20000  DEFine FuNction STRFN$(a,b$)
```

The final class of statement is nice and clear cut. It is the *flow of control* statement. Normally, the SuperBasic interpreter executes each statement and each line within a program sequentially (one after the other), but this isn't always what we desire. We may want a certain group of statements to be executed if a particular condition is met, and another group if it is not. We use a flow of control statement to take the appropriate action. In other circumstances, we may want a group of statements to be executed a given number of times, or until a condition is met. Each of these circumstances is catered for by the SuperBasic statements

```
REPeat .. END REPeat
FOR .. END FOR
SELect .. END SELect
IF.. THEN .. ELSE .. END IF
```

with the help of *NEXT* and *EXIT* within *FOR* and *REPeat* loops. REPeat is used to execute a group of statements repeatedly (hence the name!), and the loop may be left by executing an *EXIT* statement. *FOR* is used to execute a group of statements for a certain number of times (it is actually more versatile than this, but we'll leave the full description until later), and the loop may be ended prematurely by executing an *EXIT* statement. *IF* and *SELect* are used to execute certain groups of statements depending on given conditions.

Notice that the SuperBasic interpreter also supports the archaic flow of control statements seen in other implementations of the BASIC language: *GOTO*, *GOSUB* and *ON .. GOTO / GOSUB*. It is very unlikely that you will ever need to use these statements once you understand the power of the more structured statements. Here are some examples of flow of control statements.

```
100 REPeat loop
110     IF EOF(#3):EXIT loop
120     INPUT#3,a$
130     PRINT #;a$(1 to 12)
140 END REPeat loop
200 IF x=4
210     MYPROC
220     x=MYFUNC (x)
230 ELSE
240     OTHERPROC
250     x=0
260 END IF
```

QDOS

QDOS is the QL's operating system, which means that it is a collection of programs which deal with all the low-level aspects of the machine, such as reading and writing data to devices, looking after the memory in the machine, and controlling running programs. QDOS is a *multitasking* operating system, which means that it is capable of running more than one program at a time. Don't be fooled though – it is QDOS which is capable of this, not the SuperBasic interpreter, so it is not possible to have more than one SuperBasic program running at the same time!

In QDOS jargon, each program in the machine is known as a *job*, as each program represents a job which QDOS has to do. When the QL is first switched on, only one job is present. This is the SuperBasic interpreter, which is special in comparision to other jobs that may be running because it cannot be removed. QDOS allows the systems programmer to remove any job from memory except the interpreter. SuperBasic also has a few other special features, but these require a greater understanding of QDOS so they won't be described until later.

QDOS imposes a certain *memory map* upon the system which determines where various parts of the system live. This memory map is shown in *fig 1*, although it may not mean much yet. The operating system is accessed by executing a few members of a 68000 instruction type called *TRAPS*. To make the programmer's task that much easier, QDOS also has a number of *utility routines* which are accessed by getting their addresses from defined locations in memory (known as 'vectors') and then jumping to these addresses.

QDOS performs a variety of tasks. Firstly, we can use it to control the memory resources of the machine. A job can ask QDOS to give it a block of memory for a while, and if it does so QDOS can be relied upon not to give that same block of memory to anyone else until the job tells QDOS that it has finished with it. QDOS can also create new jobs, and subsequently control them by activating them, suspending them, removing them and so on. It can also be told to include

HEX ADDR		DECIMAL ADDR
FFFFC	INTERUPT 7 VECTOR	1048575
FFFFF	ROM EXPANSION	1048572
E0000	PERIPHERAL AND RAM EXPANSION	917504
40000	STANDARD RAM	262144
28000	SCREEN RAM	163840
20000	PERIPHERAL EXPANSION	131072
1C000	I/O HARDWARE	114688
18000	PERIPHERAL EXPANSION	98304
10000	PLUG-IN ROM	65536
0C000	QDOS AND SUPERBASIC ROM	49152
00000		0

PRINTERLAND

ORDERED TODAY – DELIVERED TOMORROW

VISA

Free... WITH ALL PRINTERS: QL USERS PRINTERS GUIDE · Getting the best from your printer is not easy
We supply a **free** booklet with all printers which explains how to obtain all the features available on your printer. Without this you could waste many hours. Buy elsewhere and you'll pay far more and get far less.

■ DOT MATRIX	EX VAT	INC VAT
SHINWA CP A80	£165-00	£189-75
EPSON FX80 F/T+	£314-00	£361-10
EPSON FX100 F/T+	£425-00	£488-75
■ DOT MATRIX PLUS NEAR LETTER QUALITY		
EPSON LX80 NEW	£205-00	£235-75
KAGA TAXAN KP810 SPECIAL OFFER	£235-00	£270-25
CANON 1080A	£240-00	£276-00
CANON 1156 or KAGA 910	£335-00	£385-25
■ DAISY WHEEL		
QUENDATA 1120	£225-00	£258-75
JUKU 6100	£325.00	£373-75
EPSON DX100 SPECIAL OFFER	£356-00	£409-40
■ COLOUR PRINTERS		
EPSON JX-80 SPECIAL OFFER	£450-00	£517-50
CANON PJ1080A	£425.00	£488-75
■ PRINTER INTERFACES		
MIRACLE SYSTEMS	£26-05	£29.95

■ COMPUTERS	EX VAT	INC VAT
SINCLAIR QL ONLY	£334-00	£384-10

■ DISC DRIVES
We stock and use ALL drives and interfaces available including MEDIC, COMPUTAMATE and MICRO PERIPHERALS.
For helpful impartial advice and best prices contact us.
Prices include power supply and interface.

MICRO PERIPHERALS		
3½" SINGLE DRIVE SYSTEM (.75 MBYTE)	£241-00	£277.15
3½" DUAL DRIVE SYSTEM (1.5 MBYTE)	£365-00	£419-75
COMPUTAMATE		
5½" SINGLE DRIVE SYSTEM (.75 MBYTE)	£290-00	£333-50
5½" DUAL DRIVE SYSTEM (1.5 MBYTE)	£399-00	£458-58
MEDIC: PRICES ON APPLICATION.		

■ TYPEWRITER PRINTER COMBINED		
JUKI 2200	£245-00	£281-75
■ MONITORS		
PHILIPS 7502 GREEN	£75-00	£86-25
MICROVITEC CUB 1451/653	£220-00	£253-00
■ RAM EXPANSION BOARDS		
PCML 250K	£150-00	£172-50

COMPLETE PACK SPECIAL OFFER

Package includes: SINCLAIRE QL, MICROVITEC CUB563, PRINTER & ALL LEADS AND INTERFACES

WITH EPSON LX80 F/T	£775-00	£891-25	WITH CANON 1080	£812-00	£933-80
WITH CANON 1156	£900-00	£1035-00	OTHER PRINTERS ON REQUEST		

PRINTERLAND

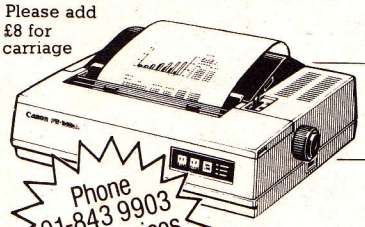
Although mainly mail order, our offices are open 9am to 5-30pm Mon-Fri and 9am to 1pm Saturday. Callers welcome for demonstration collection. DELIVERY-PRINTERS MONITORS (SECURICOR) £10-00 Other £1-00

Unit 27 Estate Buildings, Railway Street, Huddersfield HD1 1JP. Tel. 0484 514105 512037

The best service, the best value in printers from Viglen

Dot Matrix Printers

Please add £8 for carriage



Canon PW1080A/ Taxan Kaga KP810 **£279**

Epson RX80/FT **£275**

Epson FX80 **£389**

Phone 01-843 9903 for best prices on Monitors

Daisy Wheel Printers

NEW LOW PRICES

Juki 6100 **£369**

Quen Data **£247**

Please add £8 for carriage

Please add £25.00 for the Serial to Centronics Interface Cable for the QL.

Viglen are also major suppliers to educational and government establishments and welcome further enquiries and orders.

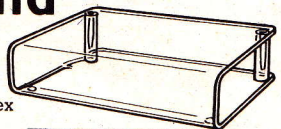
Visit our showroom
Weekdays 9.30 – 6pm.
Saturday 9.30 – 4pm.

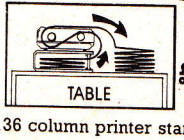
All prices correct at time of going to press

ALL PRICES INCLUDE VAT

Printer Stand

- 80 column dot matrix printers
- Raises printer high enough to put continuous stationery underneath
- Beautifully finished in clear perspex
- Viglen quality every time
- Will accept paper up to 12½" wide
- Non-slip rubber pads





Dimensions:
15" (380mm) wide
12½" (320mm) deep
4" (90mm) high

Please add £3 for carriage

Our price **£16.95** inc. VAT

Also available 136 column printer stand £27.00

Post to: Viglen Computer Supplies.
Unit 7, Trumpers Way, Hanwell W7 2QA.

Credit Card Holders may order by phone. 01-843 9903.

Please send me _____ QU10

I enclose Cheque/P.O. for £ _____ Incl. carriage.

Cheques payable to Viglen Computer Supplies.

Name _____

Address _____

I prefer to pay by ACCESS/BARCLAYCARD
(Delete whichever is not applicable)

Credit Cards valid only if signed by card holder. Address must be the same as card holder.

Card No. _____

Signature _____

BASIC SUPERBASIC

Most micros use BASIC – the far superior QL uses SuperBasic . . .

Say you have a problem and need to write a program to solve it. As you are new to the game you naturally choose to write the program in SuperBasic, but how do you go about it?

The first thing is to carefully work out the problem itself. It probably helps if you write it down. After this split it into logical sections, each of which solves a particular aspect or arrives at a helpful mid-solution. There are highbrow academic names for this sort of program writing, but we like to think of it as little more than applied commonsense.

We'll look at this in detail by considering the problem of how to write a program which counts the words in a file. There's our initial problem: 'how do we count the words in a file?'

This can be broken down quite easily:

```
1) Initialise – prepare the computer to solve our problem
2) Loop until we reach the end of the file:
       get a character
       are we currently within a word?
           If yes:
               does the new character form part of the same
               word?
                   If yes:
                       go back to step 2)
                   Otherwise:
                       increment word count
                       set flag to say 'not in word'
                       go back to step 2)
           Otherwise:
               does the character start a new word?
                   If yes:
                       set flag to say 'in word'
                       go back to step 2)
                   Otherwise:
                       go back to step 2)
3) Print out result
4) End program
```

The first step, '*Initialise*', is a nice bland word which has more to do than it may seem, so let's split it up even further:

1) *Initialise:*
   a) Get name of file to count
   b) Open it for reading
   c) set count to 0
   d) set flag to say 'not in word'

Now, steps a) and b) should also take account of the fact that the filename typed in by the user may not be found, or may be illegal, so we could split this even further if we wanted to. For the moment, though, let's just code step 1):

```
100 REMark Get file from user
110 INPUT 'Name of file to
    count:'!f$
120 REMark Open this file for
    reading
130 OPEN_In#3,f$
140 REMark Set count to 0
150 count=0
160 REMark Set flag to say
    'not in word'
170 inword=0
```

Not so difficult! Step 2 is considerably more involved and it is a good idea to break the problem down even further, perhaps by coding each line at a time.

One area of difficulty we're about to encounter is deciding whether the character we've read from the file is part of a word or not. To be able to work this out, we need to define a word. The most sensible definition is to consider a word as any sequence of one or more 'alphanumeric' characters. What's an alphanumeric character? It's computing jargon for any letter or digit; that is, the characters 'A' to 'Z', 'a' to 'z' and '0' to '9'.

Now, as we need this definition throughout the execution of step two, it would seem like a good thing to put into an independent routine. Such a routine ought to return one value if the character being examined is alphanumeric, and another if not. The values it is to return will mean, as far as we are concerned, TRUE (if it is alphanumeric) and FALSE (if it is not). Computer programmers almost always refer to FALSE as 0, and to TRUE as any value which is not 0, such as 1.

Our independent routine to determine whether a character is alphanumeric or not returns a result to use, so it is known in computing as a *function*. We can define these quite easily in SuperBasic but as usual we must first examine the problem the function is to solve:

```
1) Is the character between 'A'
   and 'Z'?
   If so, return TRUE
2) Is the character between 'a'
   and 'z'?
   If so, return TRUE
3) Is the character between '0'
   and '9'?
   If so, return TRUE
4) Return FALSE
```

Dead easy, this, isn't it:

```
10000 DEFine FuNction
      ISALNUM(char)
10010 IF char>='A' AND char
      <='Z':RETurn 1
10020 IF char>='a' AND char
      <='z':RETurn 1
10030 IF char>='0' AND char
      <='9':RETurn 1
10040 RETurn 0
10050 END DEFine
```

We've called our function *ISALNUM*, as it seems as good a name as any, and we've said that it gets one *parameter*, which is to be called *char* inside the function. We then ask SuperBasic if the character is alphanumeric, with the >= (greater than or equal to) and <= (less than or equal to) *relational operators*. SuperBasic defines that any character which is greater than or equal to 'A' and less than or equal to 'Z' will be a capital letter between 'A' and 'Z', any greater than or equal to 'a' and less than or equal to 'z' will be a

```
180 REMark Loop. . .
190 REPeat loop
200     REMark . . . until end of file
210     IF EOF(#3)=1:EXIT loop
220     REMark Get a character
230     c$=INKEY $(#3,-1)
240     REMark Are we currently within a word?
250     IF inword=1 THEN
260         REMark Yes, does the character form part of a
            word?
270         IF ISALNUM(c$)=1 THEN
280             REMark Yes, go back to step 2)
290             NEXT loop
300         ELSE
310             REMark No, so increment word count . . .
320             count=count+1
330             REMark . . . set flag to say 'not in word' . . .
340             inword=0
350             REMark . . . and go back to step 2)
360             NEXT loop
370         END IF
380     ELSE
390         REMark Does the character start a new word?
400         IF ISALNUM(c$)=1 THEN
410             REMark Yes, set flag to say 'in word'
420             inword=1
430             REMark Go back to step 2)
440             NEXT loop
450         ELSE
460             REMark No, go back to step 2)
470             NEXT loop
480         END IF
490     END IF
500 END REPeat loop
```

small letter between 'a' and 'z', and any greater than or equal to '0' and less than or equal to '9' will be a digit between '0' and '9'.

The function returns its result to the main program with the aid of the RETurn statement, and the rules for functions are such that the use of a function within an expression yields the returned value of that function, so

```
x=ISALNUM('D')
```

will put the value 1 ('TRUE') into the variable 'x', and

```
y=ISALNUM('$')
```

will put the value 0 ('FALSE') into the variable 'y'. So now step two is a little easier to write.

That's step 2 complete, but as we'll see later there's an awful lot of redundant code which has slipped in because the problem was written in a way to suit humans rather than a programming language. When the entire program has been written, we will go through the code looking for bits which don't need to be there.

CHECKPOINTS

1.–We check to see if there are any more characters to read from the file by calling the built-in EOF function. This returns 0 ('FALSE') if there are more characters to read on the specified channel, and 1 ('TRUE') if there are not.

2.–Whenever there are lines starting with 'IF something THEN' which spread over a few lines, we end the entire IF . . THEN . . ELSE sequence with a line saying ELSE IF. This is one of the rules of the SuperBasic programming language which we need to follow if we want our programs to be executed successfully.

3.–Notice that when we use the INKEY$ function to get a character from the file into our variable 'c$', we pass the function two parameters. The first, '#3', refers to the channel from which we want to read data. The second '–1', is known as the *timeout* and tells the function to wait a specified time if there are currently no characters available. The value of –1 means 'wait forever'. We need to specify a timeout in INKEY$ calls because reading data from a disk or a microdrive always takes time, and there may not be any characters available straightaway. By asking it to wait forever, we ensure that every character returned by INKEY$ is valid.

Now we can write step 3. This bit is probably the simplest:

```
510 REMark Print out the
    result
520 PRINT 'There are' !count!
    'words in' !f$
```

Step 4 is the ending of the program. In this instance, the only thing we need to do at the end of the program is to close the file we were reading the characters from:

```
530 REMark End the program
540 CLOSE#3
```

So we now have our complete program. Although you'll find that it works exactly as it should, and really does print out the number of words in a named file, purists will demand that the excesses of line-by-line coding are removed. So here goes.

The first things we can take a look at are the four lines

```
IF EOF (#3)=1:EXIT loop
IF inword=1 THEN
IF ISALNUM(c$)=1 THEN
IF ISALNUM(c$)=1 THEN
```

In all four cases, we're comparing values with 1. SuperBasic adopts much the same conventions as we do in this respect, in that it too will consider 0 as FALSE and anything else as TRUE, so we could rewrite these lines as

```
IF EOF (#3):EXIT loop
IF inword THEN
IF ISALNUM(c$) THEN
IF ISALNUM(c$) THEN
```

and the effect would be just the same. This is because *EOF(#3)*, *inword* and *ISALNUM(c$)* are just as valid expressions as *EOF(#3)=1*, *inword=1* and *ISALNUM(c$)=1*, and take exactly the same values. That is, if *inword* is 0, both *inword=1* and *inword* evaluate to 0, which is FALSE. If inword was equal to 1, both expressions would evaluate to 1, which is TRUE.

The next improvement concerns the proliferation of NEXT loop statements. As each of these occurs within IF . . THEN . . ELSE statements, nothing else could be executed until the line

```
END REPeat loop
```

is met anyway, and END REPeat loop is where the code jumps if we execute a NEXT loop statement. So, part 2 of the program could usefully be:

```
180 REMark Loop . . .
190 REPeat loop
200     REMark . . . until end of file
210     IF EOF (#3):EXIT loop
220     REMark Get a character
230     c$=INKEY$(#3,-1)
240     REMark Are we currently within a word?
250     IF inword THEN
260         REMark Yes, does the character form part of a
            word?
270         IF NOT ISALNUM(c$) THEN
310             REMark No, so increment word count . . .
320             count=count+1
330             REMark . . . set flag to say 'not in word' . . .
340             inword=0
370         END IF
380     ELSE
390         REMark Does the character start a new word?
400         IF INSALNUM(c$) THEN
410             REMark Yes, set flag to say 'in word'
420             inword=1
480         END IF
490     END IF
500 END REPeat loop
```

And subsequently the program could be renumbered. If we then remove all the REMark statements, our program becomes

```
100     INPUT 'Name of file to count:'!f$
110     OPEN_IN#3,f$
120     count=0
130     inword=0
140     REPeat loop
150         IF EOF(#3):EXIT loop
160         c$=INKEY$(#3,-1)
170         IF inword THEN
180             IF NOT ISALNUM(c$) THEN
190                 count=count+1
200                 inword=0
210             END IF
220         ELSE
230             IF ISALNUM(c$) THEN
240                 inword=1
250             END IF
260         END IF
270     END REPeat loop
280     PRINT 'There are' !count! 'words in' !f$
290     CLOSE#3
10000 DEFine FuNction ISALNUM(char)
10010 IF char>='A' AND char <='Z':RETurn 1
10020 IF char>='a' AND char <='z':RETurn 1
10030 IF char>='0' AND char <='9':RETurn 1
10040 RETurn 0
10050 END DEFine
```

This is about as efficient and compact as we can get it, unless we start compressing a few of the IF . . THEN . . ELSE statements into single lines. We are perfectly able to do this, but it greatly reduces the readability of the program. Always try to break your problem down smaller.

SUPERBASIC DATA TYPES

Whenever we use variables in a SuperBasic program, we tell the SuperBasic interpreter what *type* of data it may hold in each variable. We don't necessarily do this deliberately as the system 'defaults' to a particular type. But what is a type and how do we take advantage of them?

It's all to do with data representation; if we're dealing with numbers in an accounting program, we're likely to want to use fairly large numbers which may optionally include a decimal point. If we were writing a program to look after the stock levels in our factory, we'd probably want to use whole numbers, and if we're writing an address book program, we want to deal with strings of characters. These are the three fundamental data types accepted and used by the SuperBasic interpreter.

The first type, numbers which may include decimal points and can take on extremely large values, is known as *floating point*. On the QL, floating point numbers have a range of in the region 10 to the power of ± 615.

The second data type, that comprising whole numbers, is known as the *integer* data type. On the QL, integers can be any whole number between 32768 and −32767.

The last type, comprising of strings of characters, is known as the *string* data type, and strings can be from zero characters long (known as the *null string* ie, "") up as far as 32766 (NOT 32767!) characters long.

The reason for the existence of the three types is that different applications demand different kinds of data, and having to hold all data as one particular type would be very inefficient. QL SuperBasic variables are specified in terms of type by following the name of the variable with a special symbol. If there is no symbol, as in:

variable

then the data type is floating point. As it is implied rather than specified, this is the type adopted by default.

Integer variables are introduced by following the name with a per cent sign, as in:

variable%

and string variables are created by terminating the name in a dollar sign:

variable$

The SuperBasic interpreter exercises something known as *type coercion*, which means converting data from one type into another. For example, we may have a variable, *v*, which holds the number 23. As *v* has no special symbol at the end, it is a floating point variable. This means that its value, 23, is held as a floating point number inside the QL. Nevertheless, we can put the value into an integer variable

x%=v

because the value of *v* (23) also happens to be a valid value for an integer. The SuperBasic interpreter coerces the floating point representation of 23 into its integer representation before the value is placed into x%. This coercion is often very useful, as we don't need to check for operations such as the assignment above, and old BASIC functions such as STR$ and VAL are done away with. To see whether a coercion could be successful, we need only to consider the *generality* of each type. As a string variable can hold an arbitrary string of characters, and as all numbers may be written down as characters (and therefore stored as characters), it follows that a string variable can always be assigned a numeric value. The statement

v$=1234.56

will put the seven characters "1", "2", "3", "4", ".", "5" and "6" into the string variable *v$*. The value of 1234.56 (ie, the number 1234.56) will not be stored, simply its string representation. The string data type is therefore more general than numeric types such as floating point and integers.

Floating point is more general than integer, because every integer may also be represented as a floating point number. This makes the integer data type the least general and string type the most general.

There is a further data type accepted by the SuperBasic interpreter, called the *name* data type. This is used for procedure, variable and function names, and for file and device names. It is comprised of a string of characters, just like the string data type, but is slightly less general than string as not all strings are valid names. It is used whenever you specify a filename, as in

OPEN#3,mdv1_myfile

As you can see, the only difference between it and a normal string is that it is not surrounded in quote marks. It is perfectly permissable to add the quote marks, and thus turn it into a string, but it is generally less convenient to do so. The name data type is a sub-branch or leaf of the string data type.

UN-TYPED DATA

Occasionally you will come across instances of data which don't seem to have a type specified at all. Although you may think that this makes it default to the floating point type, this isn't quite true. Let's take a typical defined function:

```
1000  DEFine FuNction MYFUNC%(a,b)
1010  LOCal c,d$
1020  c=a INSTR b
1030  IF c=0 THEN d$='' ELSE d$='OKAY'
1040  RETurn LEN(d$)
1050  END DEFine
```

If we make a list of all the *identifiers* (names) in this function, we see *MYFUNC%*, *a*, *b*, *c*, *d$*. The function itself, *MYFUNC%*, is terminated in a % sign, which indicates that the function is typed. It returns an integer result. Likewise, the two local variables *c* and *d$* are typed; *c* is a floating point variable and *d$* is a string variable. What about the two formal parameters, *a* and *b*? No type is specified for them, yet they are used as string variables within the function.

This is another rule of the language followed by the SuperBasic interpreter: formal parameters to functions and procedures are un-typed, and the actual type is derived from the usage of the variables within the procedure or function. As we're using them as strings here, it doesn't matter what type the actual parameters are as string is the most general type. If we had used them as integers in the function, we'd get an 'Error in expression' error (or possibly 'Overflow') if the actual values of 'a' and 'b' could not be coerced into integers.

There is nothing to stop us specifying the types of the formal parameters within the function, as in

```
1000  DEFine FuNction MYFUNC%(a$,b$)
```

but as the interpreter still follows its own rules, the type we specify will be over-ridden by the type dictated by the usage.

The reason for having un-typed formal parameters is a little obscure, but as it makes little difference to the way we write programs it doesn't really matter. Some of SuperBasic's rules for typing DO affect us, though, and if we don't know about them it is very easy to get confused. The main point to watch is the 'flow of control' statements

SELect
REPeat
FOR

Each of these has a 'control variable' (or, more accurately, a control identifier) which is specified in the construct:

SELect ON var
REPeat *loop_1*
FOR *x*=0 TO 10

The important thing about these control variables is that in all three cases they MUST be floating point variables. We cannot have

SELect on var%

or

REPeat loop$

The annoying thing is that if we enter such an erroneous line into a program, the SuperBasic interpreter won't complain. Things will go wrong as soon as we run it, though, and the particular error generated may be extremely obscure. So watch out!

3 IN 1 DELTA INTERFACE

A UNIQUE TRANSFORMER FOR QL

The Delta Disk Interface enhances the Quantum Leap and transforms the QL to a truly professional standard. The Delta Disk has a unique 3 in 1 feature. It is a Disk Interface, a Centronic Printer Interface and a Memory Expansion (Optional extra).

■ Professional Quality

The disk drive system has been regarded as a standard feature on the professional systems. It is fast, reliable, economical and flexible. The Delta Disk is designed to meet the most stringent professional standard and quality giving the maximum performance, compatibility and high speed storage.

■ Compatibility

The operating system of Delta Disk is in the EPROM. The operating software is unique and is designed with the full compatibility of the QL software in mind. The Delta Disk can work with any disk drives which BBC Micro uses (Shugart Standard). It works perfectly well with the Microdrive. It means you can run both Microdrive and disk in complete harmony.

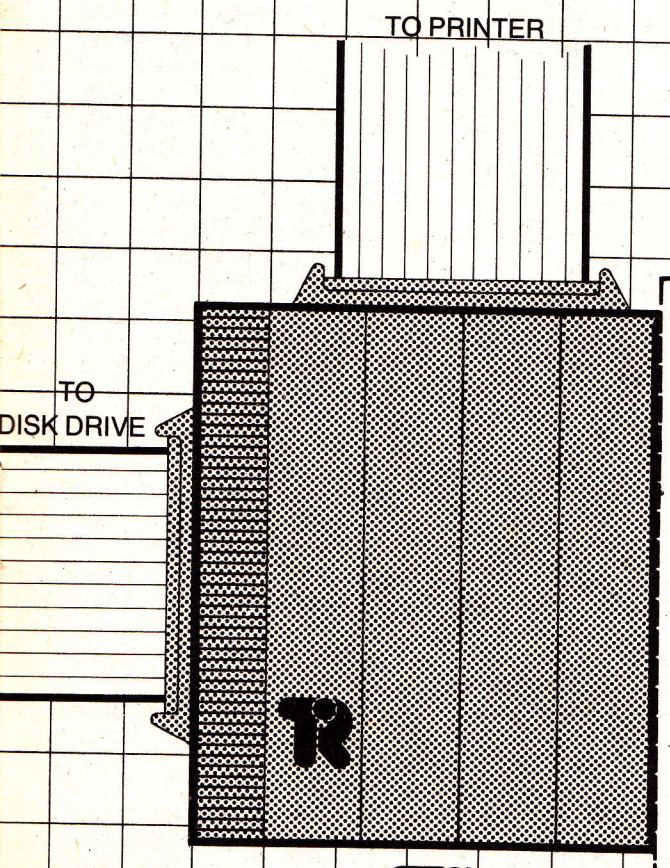
■ Unique 3 in 1 feature

The design of Delta Disk allows total flexibility and upgradibility. It is a Disk Interface, a Centronic Printer Interface and it is also a Memory Expansion up to a total RAM of 128Kbytes. Three models are available:

a. Delta Basic
b. Delta 64 with 64 Kbyte RAM
c. Delta 128 with 128 Kbytes RAM

The Delta Disk requires **no** external power supply. The Delta Disk slots into the left side of your QL to give instant operating compatibility with the disk drive and the printer.

■ The Delta disk from Technology Research, a truly inseparable partner for your QL.



CUT OUT THIS COUPON AND SEND WITH CHEQUE OR POSTAL ORDER TO:
TECHNOLOGY RESEARCH LIMITED
UNIT 18, CENTRAL TRADING ESTATE, STAINES, MIDDLESEX TW18 4UX

PLEASE SEND ME:		QTY		TOTAL
DELTA BASIC	@ £129.50		£4.00 p&p	£
DELTA 64	@ £199.00		£4.00 p&p	£
DELTA 128	@ £249.50		£4.00 p&p	£
				£

NAME _____

ADDRESS _____

_____

_____ POSTCODE _____

*Allow 28 days for delivery. Prices include VAT.

Technology Research Limited

Unit 18, Central Trading Estate, Staines, Middlesex TW18 4UX Telephone Staines (0784) 63547 Telex 896691 TLXIR G

MACHINE CODING

Overcoming the limitations and strictures of SuperBasic is one reason for using machine code. There are, however, many others . . .

Programs written in SuperBasic have one major problem – they are written in a language which the 68008 processor inside the QL cannot immediately understand. The program needs to be interpreted as it is being executed, and this is the job of the built-in SuperBasic interpreter. Now, as the processor is getting the program 'second hand' it executes it at a slower speed than would otherwise be the case. To be able to get the 68008 to work at full speed you have to cut out the middle man and write your programs in the processor's native language, known as machine code. The problem with machine code is that although the processor is eminently capable of understanding it, humans are not.

For this reason, programmers that decide to program the computer in its own language cheat a bit. They write their programs in something known as *assembly language*. Assembly language is a more readable form of machine code, in that every number that represents a machine code instruction is replaced by a short, more easily remembered catchword or *mnemonic*. The program which converts each mnemonic into a numbered instruction is known as an 'assembler'. There are quite a few of these about for the QL now, and it can't be emphasised enough just how important the use of one is if you choose to program in the processor's native tongue.

It must also be stressed that learning machine code is not the sort of thing which may be easily covered in a three-part supplement. A book and a lot of practice is needed, but budding machine code programmers may take comfort in the fact that 68000 (and 68008) assembly language is considerably more powerful and easy to learn than the native codes of other, lesser processors.

No matter what the depth of experience of a machine code programmer, he still needs to know a lot about the insides of the QL itself before any really useful programs can be written. After all, how useful is a program which is unable to write to the screen, read from the keyboard, or talk to the microdrives likely to be? To get this information, the programmer needs to understand the underlying operating system of the hardware, which in this case is the QDOS operating system. At the same time, he will need to know where in memory he may put his programs, what constraints are placed upon them by the host environment, and just what a program may realistically be expected to do.

To learn all this information, yet more books are required. Obviously, the most informative and useful is likely to be that produced by the authors of the operating system or their agents, and indeed the *QL Technical Guide*, published by Sinclair Research, provides all the information a programmer is likely to require. The problem is this information is not always presented in the most accessible form. As the first foray in to machine code in this supplement, then, we'll take a closer look at QDOS. Not from the point of view of what we may ask it to do for us, as that is a topic which needs a gentle introduction, but to see what it expects of a machine code program.

MACHINE CODE PROGRAMS

A machine code program on the QL may fall into three general categories. There is the small section of code which we may choose to write so that it may be called from the SuperBasic interpreter with the CALL procedure. There is the multitasking machine code program, called a *job*, which we would execute from the SuperBasic interpreter with the EXEC or EXEC_W procedures, and there is the resident extension. The last category may also be further sub-divided into extensions to the SuperBasic interpreter, extensions to QDOS, such as more trap-invoked routines to date-stamp microdrive files, rename files or whatever, and extensions to the system as a whole, such as device drivers, function key programmers and resident hardware controllers.

The first group, the CALLed subroutine is the least used as it is the least versatile option. A programmer may write a small subroutine to perform certain mathematical or graphical operations at a higher speed than the SuperBasic interpreter is capable of. Such routines would normally be hidden away in the *resident procedure area* of the machine, as they would then be permanently installed until the machine is reset or switched off. Resident procedure space is taken from the system by asking QDOS to allocate a given number of bytes. Within SuperBasic, this is done with the RESPR function, which is passed as a parameter the number of bytes required and returns the base address of the area allocated. If, for example, we had a 100-byte long code file on microdrive called *mycode*, it could be installed into the system with just two statements:

```
base=RESPR(100)
LBYTES mdv1_mycode,base
```

and *base* would be the address at which the first byte of the code file was loaded. If this also coincided with the start address of the routine, which it generally would, then the subroutine would be executed with a statement along the lines of

```
CALL base
```

Now, as QDOS returns the base address of the area allocated to us, the actual absolute value of this base address is not governable by us, which means that any code we write which is to be placed in the resident procedure area must be *position independent*. This is a concept which is so fundamental to the average 68000 system that every machine code programmer soon gets used to it. Thankfully, the machine code programmer finds that 68000 assembly language is particularly easy to write in a position independent manner.

The main reason that subroutines are not often written in this way is that the SuperBasic interpreter itself is extendable, and the process of adding new procedures and functions to it is made extremely easy. Once a machine code procedure or function has been added to the SuperBasic interpreter's name list, it can be called in exactly the same way as other, resident, procedures and functions. Suppose, for example, that we wrote a procedure in machine code called *MYPROC*, which expects two parameters. It would be called with a statement of this form:

```
MYPROC par1,par2
```

making it indistinguishable from other procedures and functions. If a large number of procedures and functions were to be added, it is obviously far easier to remember the routines by name rather than by base address, which we would need to do if we made each routine a subroutine to be CALLed.

Naturally, there are certain rules and coding restrictions which must be followed when writing routines to be added to the SuperBasic interpreter in this way, but they are fairly minor and easily overcome. We will look at these later.

Writing programs to be executed as independent jobs, which will multi-task, is a simple process, as the restrictions placed upon such code are very few. In fact, there's only one – the code must be position independent, and even this can be overcome by the programmer (if the code for the job is in a ROM, for example).

JOBS

If we take advantage of the multitasking environment provided by QDOS, we must follow its rules and be aware of its expectations. Every multitask-

ing program on the QL is a job, and every job is independent from any other in that it has its own personal copies of the 68008 registers which under normal circumstances are that job's private domain. It is very unusual for one job to consistently alter the register values of another job, but it is quite in order for the job which initiates another job to set up certain values prior to the new job's execution (particularly the stack pointer, as it is quite likely that the starting job will want to put channel IDs and strings on the new job's stack).

Each job is regularly started and stopped by part of QDOS called the *scheduler*, the frequency of activation being determined by a value known as the job's *priority*. This is a number between 0 and 127, with 127 being the highest possible priority. A job with a low priority will be activated less often than one with a higher priority, and one with a priority of 0 will not get activated at all.

Although the frequency of these activations may be controlled by altering a job's priority, the length of the activation period may not, as this is integral to the system. As activation periods (more often known as the *time slices*) are very short in human terms, the user gets the impression that jobs run at the same time, although this is not actually the case.

To maintain this impression, each job must allow itself to be interrupted by the scheduler so that another job may be given its time slice. For this to happen, the job must be running in the 68008's *user mode*, while the scheduler runs in *supervisor mode*. Such a situation will almost always prevail though the author of the job may choose to enter supervisor mode so that a particular operation can be performed without interruption.

Almost all QDOS routines are entered by executing a 68008 TRAP instruction, which automatically enters supervisor mode, so just about every QDOS operation is carried out in supervisor mode. However, as each operation generally takes a very small amount of time, the duration of the activation which occurs in supervisor mode is very small, so other jobs are not disturbed overmuch.

To encourage this state of affairs, QDOS ensures that those routines which will take a considerable amount of time are interruptable. For example, if we ask QDOS to send a large number of bytes to an output device and not to come back until it has finished, it will do this by telling the requisite device driver that it wants to send data, and then *suspending* the asking job. A suspended job is not given its share of time slices, but the thing which it is suspended for (in this case the device driver) will be periodically asked if it has finished yet. When the device driver replies that it has, QDOS *releases* the suspended job which continues where it left off. The upshot of all this is that it is only in very exceptional circumstances that a job will enter supervisor mode, and therefore suspend multitasking, for an extended period. Unfortunately, one of these circumstances occurs whenever a microdrive is accessed, which is why everything stops when a job talks to the microdrive.

QDOS keeps track of all the jobs by maintaining a table in memory, and each job is preceded by a special 104 byte long header which QDOS uses to keep track of things, such as the job's priority, its current status (i.e. whether it is suspended or not, whether it is currently active, and so on). This area is also where QDOS keeps the job's stored registers. When a job gets its timeslice, QDOS retrieves the register values from this store, puts them into their requisite registers, and continues. When the job's timeslice is over, QDOS re-saves the register values in the same area. In this way, the processing of each job is entirely independent. This has the side effect that if one job crashes in a way which is not entirely catastrophic, the other running jobs continue to run.

QDOS imposes a (get ready for it!) *heirarchical tree structure* upon the jobs in the machine. This means that every job is "owned" by another, and that removing ("killing") a job will also kill any jobs below it in the tree.

The SuperBasic interpreter is always job number 0, which places it at the top of the tree, and any job started with the EXEC or EXEC_W commands will be owned by the SuperBasic interpreter. In Unix terminology, these new jobs will be children of the SuperBasic interpreter. A typical job tree is

When a job is killed, any memory allocation it owns is released back to the system, and any open channels it owns are closed (the ownership of a channel is determined when a channel is opened). This is why a tree set up by the QJUMP QL Toolkit commands EX and EW works so effectively. If a typical EW command line is examined: **EW my job,#1 TO other job, myfile TO page,ser1c; 'A job tree'** we can see that it specifies three jobs: *my job, other job, page*. The input to *myjob* is taken from SuperBasic channel #1, and its output is sent along a pipe to *otherjob*. *Otherjob* also gets some information from *myfile*, and its output is sent to *page*. *Page* sends its output to SER1C, and also has a parameter string 'A job tree'.

Now, in order for the whole tree to be dismantled when the process has ended, and in order for all the channels opened for these jobs to be closed, the EW command uses the QDOS job tree structure. It first makes space for *page*, and loads it in. *Page* is therefore owned by the SuperBasic interpreter. It then loads *otherjob*, telling QDOS to make it a child of *page*. Finally, *myjob* is loaded as a child of *otherjob*. The channels are opened for each job and their IDs are put in places where the jobs can find them (on top of their stacks). Now, as the chain of data flows from *myjob* through *otherjob* to *page*, it is *page* which will finish last. When it is killed (by finishing) it also kills *otherjob*, as it owns it. The killing of *otherjob* in turn causes *myjob* to be killed, as it owns that. Thus, the killing of *page* removes the entire tree, meaning that commands like EW do not need to keep track of all the jobs as the process is handled entirely by QDOS.

As we have seen, QDOS is well-equipped to help us in our machine code programming efforts. There is a great deal more that it can do for us, too, but we can't cover all that here.

HIGH-LEVEL PROGRAMMING

Another way of writing programs in machine code, without going through the hassles of having to learn about the 68008, about QDOS and about multitasking, is to write our programs in high-level languages which *compile* into machine code programs. We can write all our programs in a language we are familiar with, such as C, Pascal or BCPL, and then let the compilers of these languages do the hard work of converting them to machine code.

This is a great solution, so long as there is a compiler available for the language with which you're particularly happy with, but it is very rare that such a compiler would also let us get down to talking directly with the system. We would normally have to add little machine code extensions to the compiler, which may be seen as defeating the object. However, as the majority of the program is likely to be written in the high level language, this is often a very popular solution.

Bibliography
QL Technical Guide, Sinclair Research Ltd
QL Toolkit Manual, QJUMP
Advanced QL Machine Code, Adam Denning, Duckworth
The Sinclair QDOS Companion, Andrew Pennel, Sunshine

THE EXPANDING QL

The QL currently finds wide application as a development system. There are, however, less ambitious extensions...

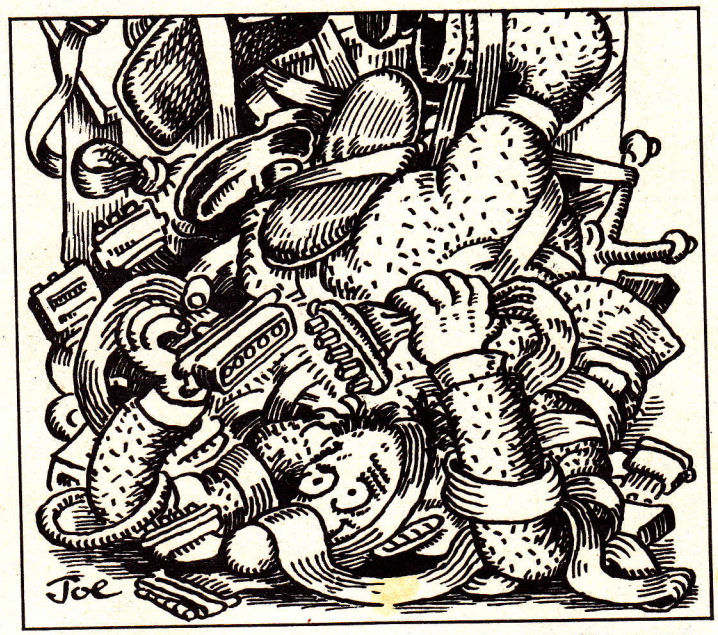
One of the most frustrating things about the QL when it's used for professional work, such as word processing or database management, is the slowness of the microdrives when loading and saving data. The average Quill document of perhaps 2000 words takes far longer to load into the machine than Quill itself does. When you consider that Quill is considerably longer than a 2000 word document, you begin to wonder just who is to blame for this alarming lack of microdrive performance. Whatever, the business user soon finds himself confronted with one option – he must buy a disk interface and disk drives for his QL.

Disk drives offer a number of advantages over microdrives. In particular, they are in general far faster at loading and saving documents, programs and data, and can squash a lot more storage space onto a disk than the microdrive can onto a cartridge. The average QL disk interface with 80-track double sided drives manages a wonderful 720K (over 720000 characters!) per floppy disk.

The disadvantages are of course just as apparent. Disk drives are bulky, noisy and consume a fair amount of power. The units themselves are extremely expensive and the proliferation of types available makes the choice very difficult for the novice. If you have got to the stage where you really believe you cannot do without disk drives, then what do you buy?

Consider first your reasons for wanting disks. Is it for more storage, faster loading or increased reliability? If you are buying disks for only one of these reasons, then your choice is reasonably simple: buy the unit which most nearly matches your ideal. If however, you choose to buy a disk drive for ALL these reasons, you'll find a host of manufacturers and retailers just dying to get hold of your money. First you must choose your disk interface.

Disk interfaces are now made by at least seven different companies, but each offers different facilities. Tony Tebby, the designer of QDOS, has designed a disk *format* which ought to be adopted by all disk interface manufacturers. It isn't, of course, but those that do tend to be more highly thought of than those that don't. Personally, I use the CST/ Computamate QDISC interface, which costs £99 and seems to be one of the fastest units around. It can support the standard 5.25" floppy disks, the new 3.5" micro-floppy disks, and even the horrible 3" format. It is also able to support single and double-sided units with 40 or 80 tracks. Not to mention single and double-density! We'll explain all these terms in a moment.



Other disk interfaces can also support most of these types, so really it is a question of buying your interface (see earlier issues of *QL User* to help you make your choice) and then buying your drive. Maximum storage will be obtained from a drive if it is both double-sided and 80 track. Double-sided means that it uses both sides of the floppy disk for storing information, and it does it without asking you to turn the disk over! A disk drive which is 80-track will store twice as much information on a floppy disk as a 40-track drive, because although it makes use of roughly the same physical area of disk, it divides the sections it uses into smaller widths. The most flexible unit would be one which is capable of being both 40 and 80-track, but it is fairly unlikely that anyone but a programmer would need such versatility.

The drive you buy must also incorporate its own power supply unit, as the QL would be stretched past its limits if it were asked to supply the power to the drive. In the same way as we find two microdrives far more convenient than one, you'll undoubtedly find that having two disk drives is better than one. Most disk drive manufacturers supply single and double-drive units in cases, so two drives take up only a small amount of desk space more than a single drive.

Choosing between the various sizes of floppy disks and drives is a little more difficult, as there are no clear-cut reasons why you should plump for one size in preference to another. 5.25" disk drives and floppy disks are more widely available and generally cheaper than 3.5" versions, but 3.5" disks are more durable and becoming increasingly more standard. They are also considerably smaller.

When I bought drives for my QL I plumped for Cumana 40/ 80-track switchable dual double-sided drives, in the 'new' slimline format. The main reason I decided upon these units was that I was offered some valuable advice by the retailers, and really one drive is much like another. There are a few technical details to sort out, most particularly it is important that the drives you buy have a 'READY' line available in the standard place.

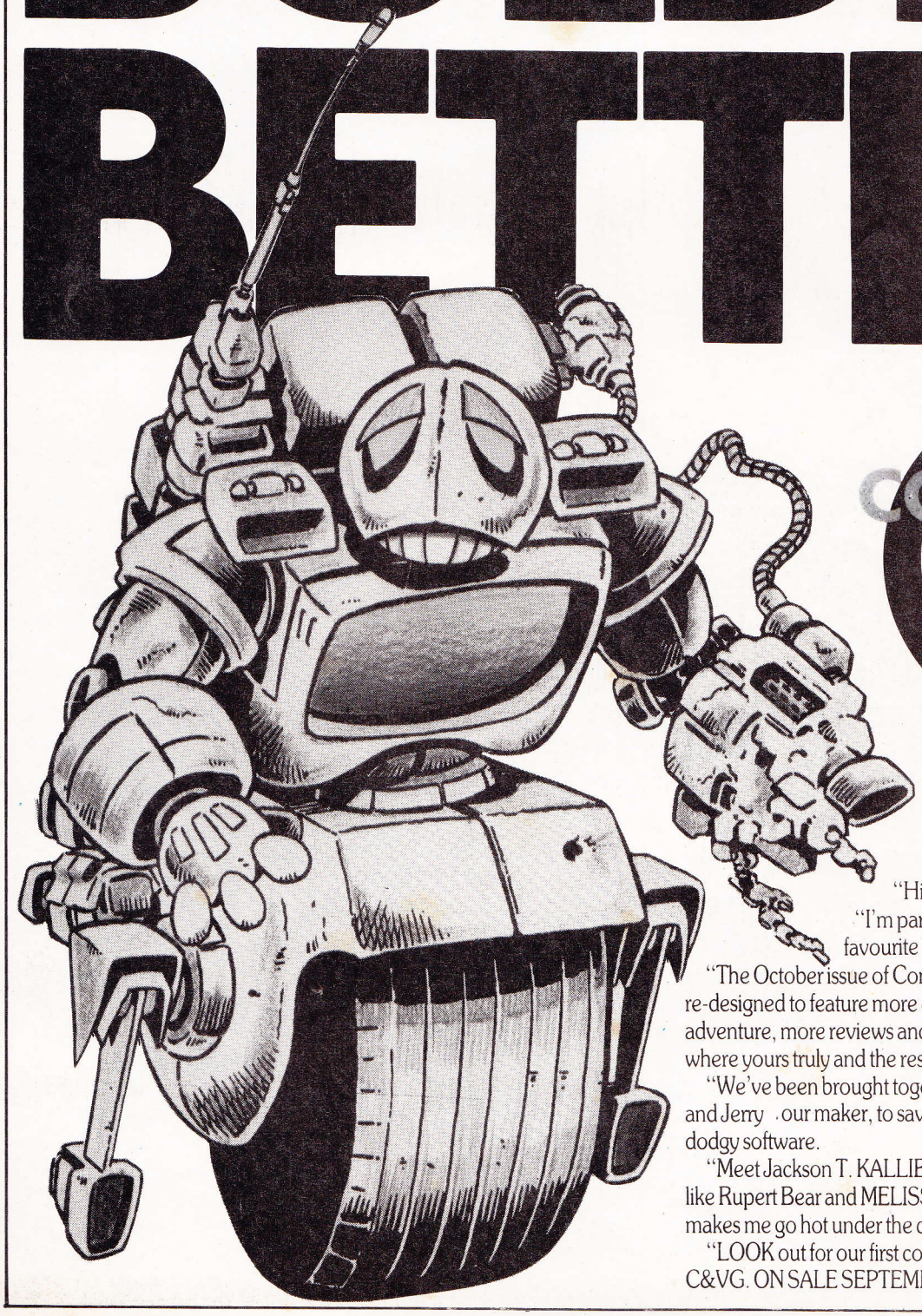
The terms 'double density' and 'single density' refer to the method of data storage used by the interface to record data on the disk. Double density, as its name implies, allows twice as much storage per disk as single density interfaces. It is just about standard now for every interface to use double density, and it is only on outdated machines such as the BBC Micro that you'll find frequent usage of single-density formats.

Your choice should now be narrowed down to being very much "how much do I want to spend?" dependent. Taking all the choices available, the chart of usefulness versus cost looks something like this:

80-track double-sided dual drive
40-track double-sided dual drive
80-track single-sided dual drive
40-track single-sided dual drive
80-track double-sided single drive
40-track double-sided single drive
80-track single-sided single drive
40-track single-sided single drive

The choice between disk sizes is really a personal thing, but otherwise, use your money wisely to get a drive as far up the list as possible, but don't forget that you still need an interface and some floppy disks!

BIGGER BOLDER BETTER



COMPUTER VIDEO GAMES

"Hi I'm Otiss.

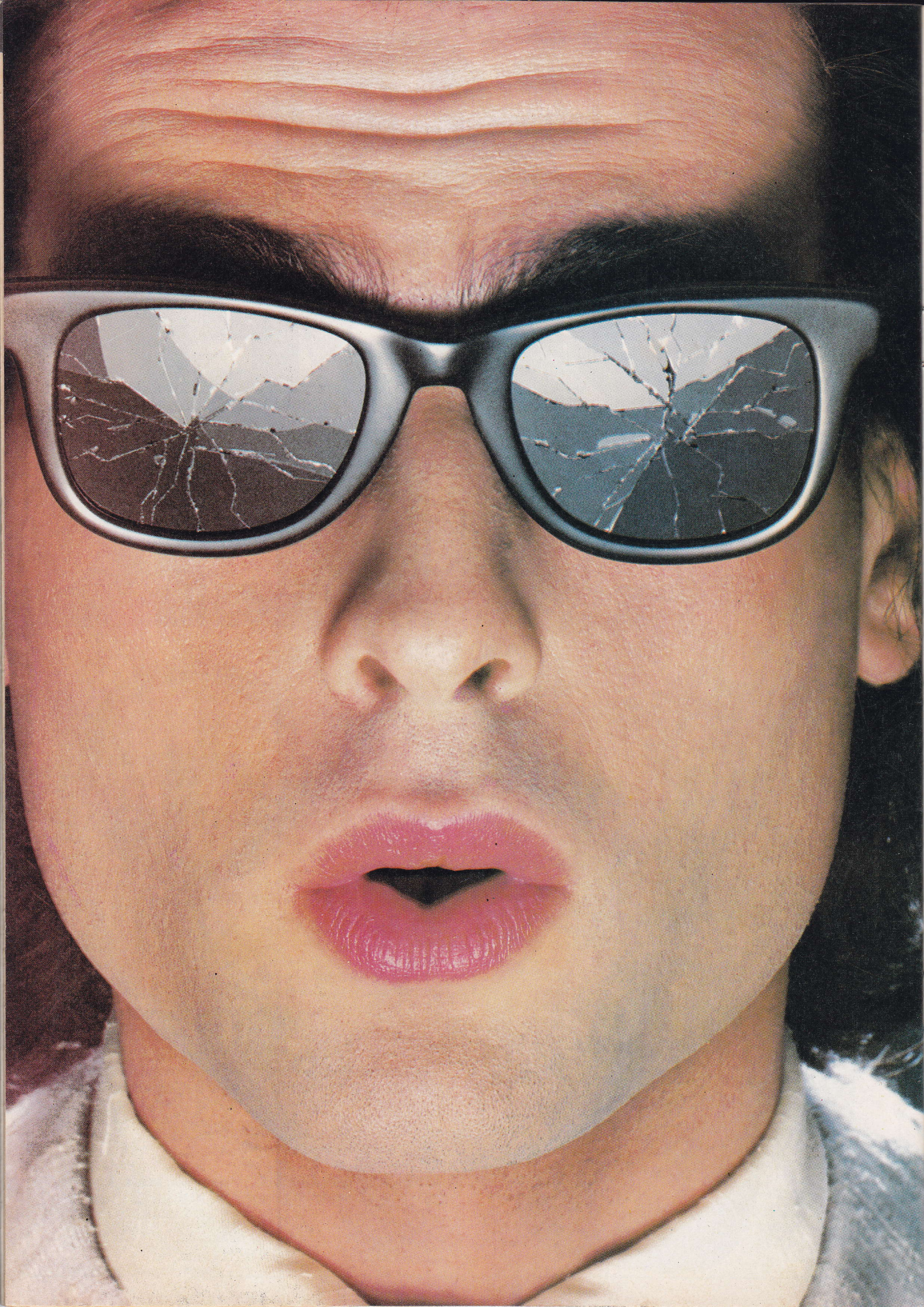
"I'm part of a great new design that makes your favourite games mag even bigger, bolder and better.

"The October issue of Computer & Video Games has been re-designed to feature more colour, bigger screen shots, more adventure, more reviews and the brilliant I.D.E.A.S. Central. That's where yours truly and the rest of the BUG HUNTERS come in.

"We've been brought together by Tim (the esteemed editor of C&VG) and Jerry . our maker, to save *you* from death, destruction and dodgy software.

"Meet Jackson T. KALLIBER . . . my boss, who makes Rambo look like Rupert Bear and MELISSA RAVENFLAME his boss . . . well she makes me go hot under the collar and I'm a robot!

"LOOK out for our first computer adventure in the OCTOBER C&VG. ON SALE SEPTEMBER 16TH. ONLY 95p."



Nice Password.
Shame about the Identity.

It's a unique combination.

Your Special Identity Number and Personal Password. The valuable key to huge databases teeming with activity, set on our Mainframes across the nation.

On Micronet 800, you're a valued individual, adding your own special flavour and personality to the database.

Take our exciting new "Gallery". You control your personal screens for all to see. The intriguing "Chatline" public conversation service gives you freedom to express your views and meet some remarkable people.

All part of a tremendous Communications section that networks you to 60,000 Micronet and Prestel users across the country. Try Teleshopping, or interview celebrities live on "Celebrity Chatline" every Wednesday night.

And there's FREE (& instant) National Electronic Mail, plus International Telex, and the Contact and SwapShop bulletin boards.

Get computer news first on Micronet's daily (and controversial) "Newsflashes" and read up on the latest reviews and courses. Feast from our regularly changing menu of programs to download straight into your micro - absolutely free.

You also get access to Educational Computing's "School Link" and Prestel's huge 300,000 page database, including world news, business & share bulletins and optional homebanking. For only £16.50 per quarter, that's less than the price of a daily paper!

Micronet is unique amongst networks and bulletin boards as it keeps your phone costs very low with special local* rate calls whenever you connect up – that's around 40p for a whole hours entertainment each evening.

The only accessory you need is a Modem, to get the best value for money around in micro communications.

Fill in the coupon for the full facts and send to Micronet 800, 8 Herbal Hill, London EC1R 5EJ. But be warned, Micronet 800 is a 'living' service with ever-expanding features. So maybe you'd be better to call in at your local Micronet 800 Action Station. There are thousands of Micronetters waiting to meet you!

To: MICRONET 800, Durrant House, 8 Herbal Hill, London EC1R 5EJ. Telephone 01-278 3143.
Please send me the full facts about Micronet 800.

Name_____

Make/Model of Micro_____

Address_____

Telephone_____

* For 98% of telephone users.

MEDIC DATASYSTEMS DISK DRIVES

At last, the definitive QL expansion system has arrived:

EXTRA RAM	1 DISK DRIVE	2 DISK DRIVES
–	£249	£399
256K	£359	£509
512K	£449	£595

The above systems use 3.5" 720K Mitsubishi drives, include parallel printer interface, RAM disk and the following **FREE** software: M-DESK, M-BASE, M-ACCOUNTS, M-KEY, M-SQUEEZE, M-SPELL, M-MERGE, M-BOOT, M-TRANSFER, M-COPY.

DISCOUNTS: Deduct **£10** for orders received by the end of October or **£20** if order totals more than £350.

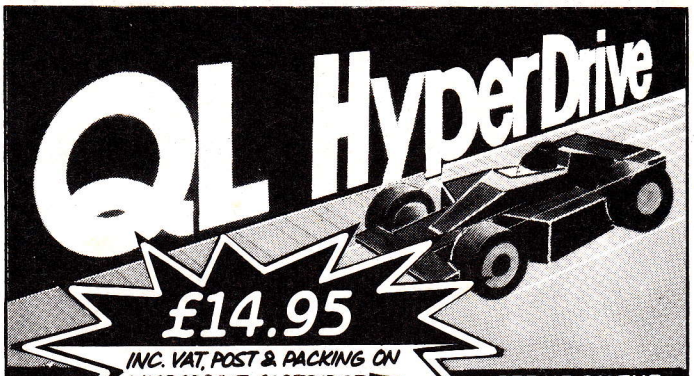
We sell a wide range of other products for the QL at equally **competitive prices**. Just write or telephone for details.

PRICES INCLUDE VAT AND DELIVERY

EXPORT ORDERS WELCOME (NO VAT)

COMPWARE

57 Repton Drive
Haslington, Crewe CW1 1SA
Tel: (0270) 582301



THE QL RACING GAME! FROM ENGLISH SOFTWARE TEL: 061-835 1358

The ENGLISH SOFTWARE Co.
1 North Parade, Parsonage Gardens,
Manchester M60 1BX

ENGLISH SOFTWARE™

It's easy to complain about advertisements. But which ones?

Every week millions of advertisements appear in print, on posters or in the cinema.

Most of them comply with the rules contained in the British Code of Advertising Practice.

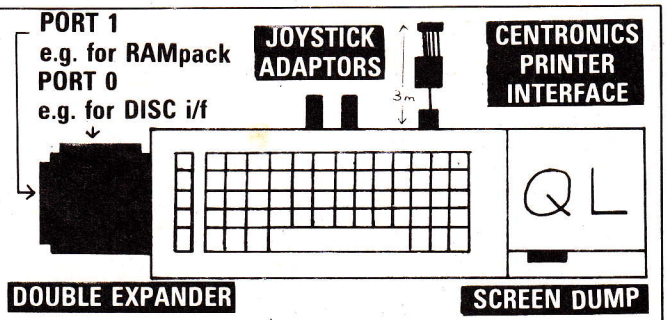
But some of them break the rules and warrant your complaints.

If you're not sure about which ones they are, however, drop us a line and we'll send you an abridged copy of the Advertising Code.

Then, if an advertisement bothers you, you'll be justified in bothering us.

The Advertising Standards Authority.
If an advertisement is wrong, we're here to put it right.

ASA Ltd. Dept 2 Brook House, Torrington Place, London WC1E 7HN



CENTRONICS PRINTER INTERFACE £29.95 inc.
★ 12 month guarantee.
★ Connects QL to any CENTRONICS compatible printer.
★ fully self-contained with 3 metre cable.
★ Plugs directly into SER1 port and directly into printer.
★ Fully compatible with QDOS, SuperBasic, and PSION packages.

SCREEN DUMP £4.99 inc.
★ Copies QL screen image to printer.
★ Supplied on microdrive cartridge.
★ Contains other useful software.

DOUBLE EXPANDER £39.00 inc.
★ Connects up to 2 expansion cards to QL.
★ Ideal for RAMpack and DISC interface combination.

JOYSTICK ADAPTOR £4.99 inc.
★4 Plugs into either CTL1 or CTL2.
★ For use with Atari/Commodore style joysticks.

MOUSE — Call for details.

Please contact your computer dealer or send your order with cheque to:

MIRACLE SYSTEMS LTD
Avondale Workshops
Woodland Way
Kingswood
BRISTOL BS15 1QL
Tel: (0272) 603871 × 210

Sinclair and QL are Trade Marks of SINCLAIR Research Ltd

ACCESS orders welcome

WARM RESET

Alan Turnbull turns an 'intentional' bug to his advantage.

Ever since receiving my QL way back in early 1984, I have been haunted by the <CTRL>, <ALT>, '7' problem – I refer to the way the QL 'crashes' when you press these keys simultaneously. I was convinced it was a ROM bug which had slipped through the net that had been dragged over releases *FB* and *PM*. But, as I reveal in this article, I was to be proved happily wrong and I went on to exploit my findings.

Tim Fountain, a Sinclair Research engineer, assured me that it wasn't a bug at all – indeed, he insisted that it was an intentional feature! I thought he was pulling my leg. But no – he went on to say that the keys **<CTRL>**, **<ALT>** and **'7'** were used by the Sinclair Research design team, when the QL was still on the bench, to simulate a non-maskable interrupt. The keys are sufficiently spaced apart so as not to be pressed accidentally.

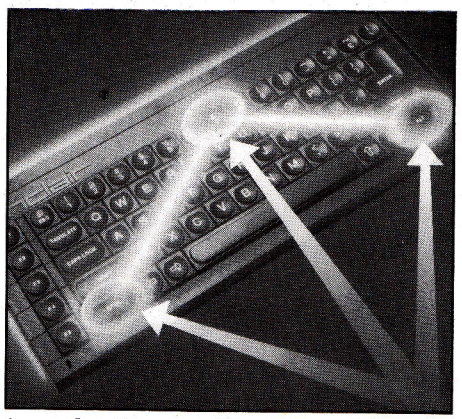
Upon pressing this sequence of keys, the QL's secondary processor (Intellingent Peripheral Controller) – the Intel 8049 – generates a level 7 interrupt internally and this in turn causes the QL's primary processor, the Motorola MC68008, to jump to a non-maskable interrupt server routine normally in ROM. Unfortunately, when the QL was let loose on the public, someone forgot to take out the development facility and hence the bug that is not a bug!

Mr Fountain continued by assuring me that the effect could be disabled using a mechanism within QDOS – the QL's operating system – for redefining the vectors which it does not use for its own purposes.

Sure enough, the system calls to QDOS to redefine the TRAP vectors is **TRAP #1** with D0=7, D1 holding the job ID (which, for the SuperBasic command interpreter, is 0) and A1 pointing to the new vector table. Unfortunately, the QDOS Manual doesn't say much more on the matter so I was left to my own devices with Psion Software Support not able to help me, either.

After much head scratching, visits to the pub, more head scratching and even more visits to the pub, I have finally found-out how to tame the beast and I present my boozy efforts for your delectation – hic!

An assembly language program to set up a new vector table is given in *listing 1*. It makes the QL do a warm restart whenever **<CTRL>**, **<ALT>**, **'7'** are pressed, a privilege violation occurs or an illegal instruction (other than those starting with **$Axxx** or **$Fxxx**) is attempted. Also, when a *CALL* command is



issued with an odd address (thereby generating an address error exception), the machine will not crash but return to the main SuperBasic command interpreter loop. This happens in the ROM anyway but needs to be copied here.

What I mean by a warm restart is essentially the same as that done by the QL when you press the reset key. Except that the RAM is not cleared, although system variables are initialised. This is not quite the same as a 'true' warm restart but does enable you to 'recover' partially by saving part of RAM to Microdrive or disc. So what did you expect – a 'Quantum Leap'?

The routine was developed on a *JS* QL and, unfortunately, the warm restart routine on the QL is not vectored. I therefore provide the other addresses for you – just pick the relevant one and substitute its definition in the assembly listing. Those of you who do not have access to an assembler and wish to enter the code using a BASIC loader should enter *listing 2*. This loader automatically inserts the correct address for the warm restart routine in your particular ROM release.

Any QL owners still with *FB* or *PM* (remember – type **'PRINT VER$'**) should immediately return their machine to Sinclair with a covering note, regardless of guaran-

Listing 1

```
                ********************************************************
                *                                                      *
                * Motorola MC68008 program to provide a 'warm reset'   *
                * facility by re-defining the <CTRL>, <ALT>, '7' effect.*
                * Also, vectors which normally cause the QL to 'crash' are *
                * re-defined to cause a warm system restart instead.   *
                *                                                      *
                * Asseable, save to Microdrive, reserve memory with    *
                * something like LET a=RESPR(256), load code into memory *
                * with something like 'LBYTES mdv1_code,a' and activate *
                * with 'CALL a'.                                       *
                *                                                      *
                * Alternatively, run program in Listing 2.             *
                *                                                      *
                * COPYRIGHT (c) August 1985, Alan Turnbull, B.Sc.      *
                *                                                      *
                *                                                      *
                ********************************************************

BASIC_JOB       EQU     $00                     SuperBASIC job ID
MT_TRAPV        EQU     $07                     TRAP #1 key for TRAP re-direction

WARM_RESTART_AH EQU     $01CA                   Warm restart for  'AH' ROM (QDOS v1.02)
WARM_RESTART_JM EQU     $01CA                   As above but for  'JM' ROM (QDOS v1.03)
WARM_RESTART_JS EQU     $01CC                   As above but for  'JS' ROM (QDOS v1.10)
WARM_RESTART_JSU EQU    $01CE                   As above but for 'JSU' ROM (QDOS v1U10)
WARM_RESTART_MG EQU     $01CC                   As above but for  'MG' ROM (QDOS v1.13)

NEW_DIV_BY_ZERO EQU     $005E
NEW_CHK_INSTR   EQU     $005E
NEW_TRAPV_INSTR EQU     $005E
NEW_TRACE_EXCEPT EQU    $005E
NEW_TRAP_05     EQU     $005E
NEW_TRAP_06     EQU     $005E
NEW_TRAP_07     EQU     $005E
NEW_TRAP_08     EQU     $005E                   Address of 'RTE' in all ROMs
NEW_TRAP_09     EQU     $005E
NEW_TRAP_0A     EQU     $005E
NEW_TRAP_0B     EQU     $005E
NEW_TRAP_0C     EQU     $005E
NEW_TRAP_0D     EQU     $005E
NEW_TRAP_0E     EQU     $005E
NEW_TRAP_0F     EQU     $005E

VIDEO_RAM       EQU     $20000                  Start of video RAM
SV_RAMT         EQU     $28020                  System variable that holds RAM top + 1
SV_STACT        EQU     $28480                  End of fixed system variables

                LEA     TRAP_TABLE,A1           Point to new TRAP vector table
                LEA     NEW_ADDR_ERROR,A0
                MOVE.L  A0,ADDRESS_ERROR-TRAP_TABLE(A1)
                LEA     NEW_ILL_INSTR,A0
                MOVE.L  A0,ILLEGAL_INSTR-TRAP_TABLE(A1)  Load new vectors
```

```
                LEA     NEW_PRIV_VIOL,A0
                MOVE.L  A0,PRIVILEGE_VIOL-TRAP_TABLE(A1)
                LEA     NEW_INT_LEV_7,A0
                MOVE.L  A0,INT_LEV_7-TRAP_TABLE(A1)
                MOVEQ   #BASIC_JOB,D1           Signal SuperBASIC job
                MOVEQ   #MT_TRAPV,D0            Re-define trap table
                TRAP    #1                      Do 'manager' TRAP
                RTS                             Return to SuperBASIC
TRAP_TABLE

ADDRESS_ERROR   DS.L    1
ILLEGAL_INSTR   DS.L    1
                DC.L    NEW_DIV_BY_ZERO
                DC.L    NEW_CHK_INSTR
                DC.L    NEW_TRAPV_INSTR
PRIVILEGE_VIOL  DS.L    1
                DC.L    NEW_TRACE_EXCEPT
INT_LEV_7       DS.L    1
                DC.L    NEW_TRAP_05
                DC.L    NEW_TRAP_06
                DC.L    NEW_TRAP_07
                DC.L    NEW_TRAP_08
                DC.L    NEW_TRAP_09
                DC.L    NEW_TRAP_0A
                DC.L    NEW_TRAP_0B
                DC.L    NEW_TRAP_0C
                DC.L    NEW_TRAP_0D
                DC.L    NEW_TRAP_0E
                DC.L    NEW_TRAP_0F

* [Effectively the same as the corresponding routine in ROM]

NEW_ADDR_ERROR  ADDQ.L  #8,A7                   Skip over extra words on stack
                RTE                             Return from this exception

NEW_ILL_INSTR

NEW_PRIV_VIOL

NEW_INT_LEV_7   MOVE.W  #$2700,SR               Set 'supervisor' mode, mask-out interrupts 1-6 and clear 'user' flags
                MOVEA.L SV_RAMT,A5              Signal value for top of RAM needed by warm restart
                LEA     VIDEO_RAM,A0            Point to start of video RAM
                MOVE.L  #SV_STACT-VIDEO_RAM-1,D0 Number of bytes to top of fixed system variables
AGAIN           CLR.B   (A0)+                   Clear a byte (also provides a delay to stop other lev. 7 interrupts)
                DBRA    D0,AGAIN                Repeat until counter exhausted
                MOVE.L  #WARM_RESTART_JS,A0     Point to warm restart routine in 'JS' ROM
                JSR     (A0)                    Do it!
```

tee validity – these ROM releases and accompanying Psion suites are obsolete and no longer officially supported.

Version *JSU* is the US version of *JS* and *MG* is a slight improvement on *JS* with a few minor bug corrections and better 'foreign language independence'. I do not yet have confirmation of the official release of these ROMs.

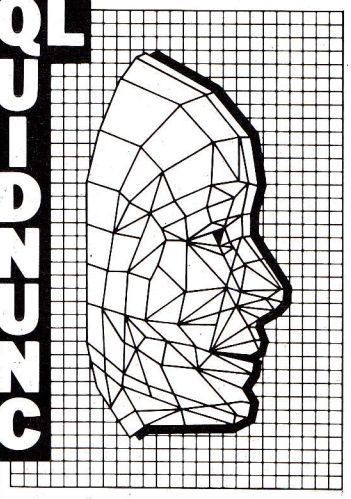
I hope this article has been of interest to those QL owners who thought they had found a major bug and had no chance of correcting it. With a little lateral thinking the problem has not only been eradicated, but a useful addition to the QL's specification has been provided without using a soldering iron, sticky-back plastic or even cardboard tubes!

Listing 2

```
100 REMark  SuperBASIC program to implement MC68008 program in Listing 1
110 REMark          COPYRIGHT (c) April 1985, Alan Turnbull, B.Sc.
120 :
130 LET reserved_address=RESPR(128)
140 RESTORE
150 LET address=reserved_address
160 REPeat read_and_store_data
170   IF EOF THEN EXIT read_and_store_data
180   READ data_item
190   POKE address,data_item
200   LET address=address+1
210 END REPeat read_and_store_data
220 CALL reserved_address
230 STOP
240 :
250 DATA 67,250,0,10,48,120,1,16,78,144,78,117,0,1,0,68,4,67,65,76,76,0,0,0,0,1,
0,12,5,81,68,79,83,36,0,0,0,0,187,203,102,36,114,6,48,120,1,26,78,144,34,110,0,8
8,93,73,61,188,0,4,152,0,112,0,78,65,45,130,152,2,45,73,0,88,120,1,78,117,112,24
1,78,117,48,120,1,24,78,144,102,20,229,139,103,240,215,46,0,88,47,54,152,0,76,24
6,63,254,152,4,112,241,78,117
```

Prestel and Telecom Gold are registered trade marks of British Telecommunications plc.

QL QUIPUNC



More contemptible, contrite communication from the shadows, penned by Silhouette.

With the widely rumoured and recently confirmed price cut upon us, the QL seems set to burst upon the home and dare I say, games market. At such a 'bargain basement' price (£199.95), despite a limited software base (soon to change, no doubt), anyone contemplating a purchase this Christmas would be foolish not to consider the QL. Here's a machine that more than adequately fulfills the role of 'home micro', catering for the home professional (Psion packages), enthusiast (it's CPU and operating system beat the others hands down) and games player (soon, hopefully) and at a price only marginally above the Commodore 64 and Spectrum – the two market leaders. You can keep your Atari's, Amstrads and Amiga's, I'm sticking with the QL!

And now . . . over to Acorn, where the decision to put a 16-bit version of the 6502 (a 2-bit chip if ever there was one) inside its ICL OPD competitor (called, I believe, the Communicator, or is it the Terminator?!) is causing some merriment. Interestingly the 6502 is hardly pushed by the demands placed upon it by a mere terminal program (and View?!). Seems to me that Acorn's upgrading for no reason.

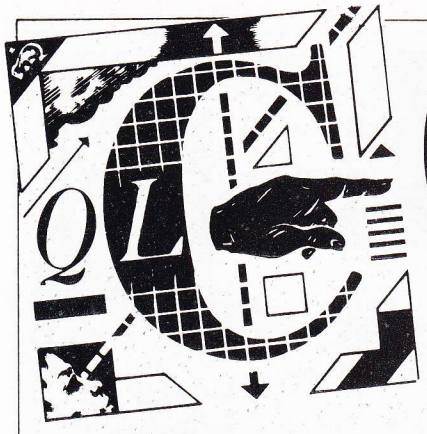
Next – a wind up. When is a PR rep not a PR rep? Answer – when he's being impersonated, for that's what happened to a certain employee of Commodore's PR company. Yes, some unknown character phoned a few software houses around the country, promised them immediate delivery of an Amiga next day, and told them that the machine was to be released at PCW for £495. As you can imagine, this rumour shot around the country at lightning speed, until the real rep heard it. He, apparently, was a mite peeved!

Dr David Potter, founder and Managing Director of Psion, caught pontificating – or is he organising upgrades?

Psion and the ubiquitous David Potter have not been entirely idle of late. It seems they've got enough employees left to update Archive a little (Let's face it, it needed it). The new product is designated V2.30 (or 31) and will doubtless be distributed with new QLs, so let's take a look – the programs (the whole Psion suite of four) can now be executed after each other without the need to reboot. That'll be useful. Quill can now import single lines and paragraphs. Great! Easel is supplied with 10 printer drivers. Should have been done in the first place. Perhaps the company really *does* intend to put the things in ROM?

SILHOUETTE

The Publishers wish to point out that the statements and opinions expressed here do not necessarily reflect those of QL User or its staff. Statements made by Silhouette are purely subjective and should be treated as such.



C SERIES

Continuing our series Adam Denning takes up where most other languages leave off and examines C's most powerful tool – the composite data structure.

The ability to handle complex data types sets C apart from its ancestors. Integers, characters, arrays and real numbers are just a beginning. With the use of an aggregate data type it is possible to hold many items of connected information in a single 'variable'. Unlike an array each component item need not be of the same type, and each may be individually named so that we can refer to it later on.

This mysterious data item is known as the *"structure"*, and anyone who has programmed in Pascal or Algol will know its usefulness. We define it within in C program using the keyword **struct**, like this:

```
struct mystruc{
                char *name
                int i;
                struct mystruc *ptr;
                };
```

What we have in fact defined is simply a 'template' for use later in a program. It is important to note that no storage space is reserved for any data, as no variables have been declared as being this type of structure.

The definition begins with the **struct** keyword which may be followed by any valid C identifier – in this case *mystruc*. This identifier becomes the structure "tag", and is the name with which we declare any variables which we want to be structures of this type. We'll see later that the tag is entirely optional but highly useful.

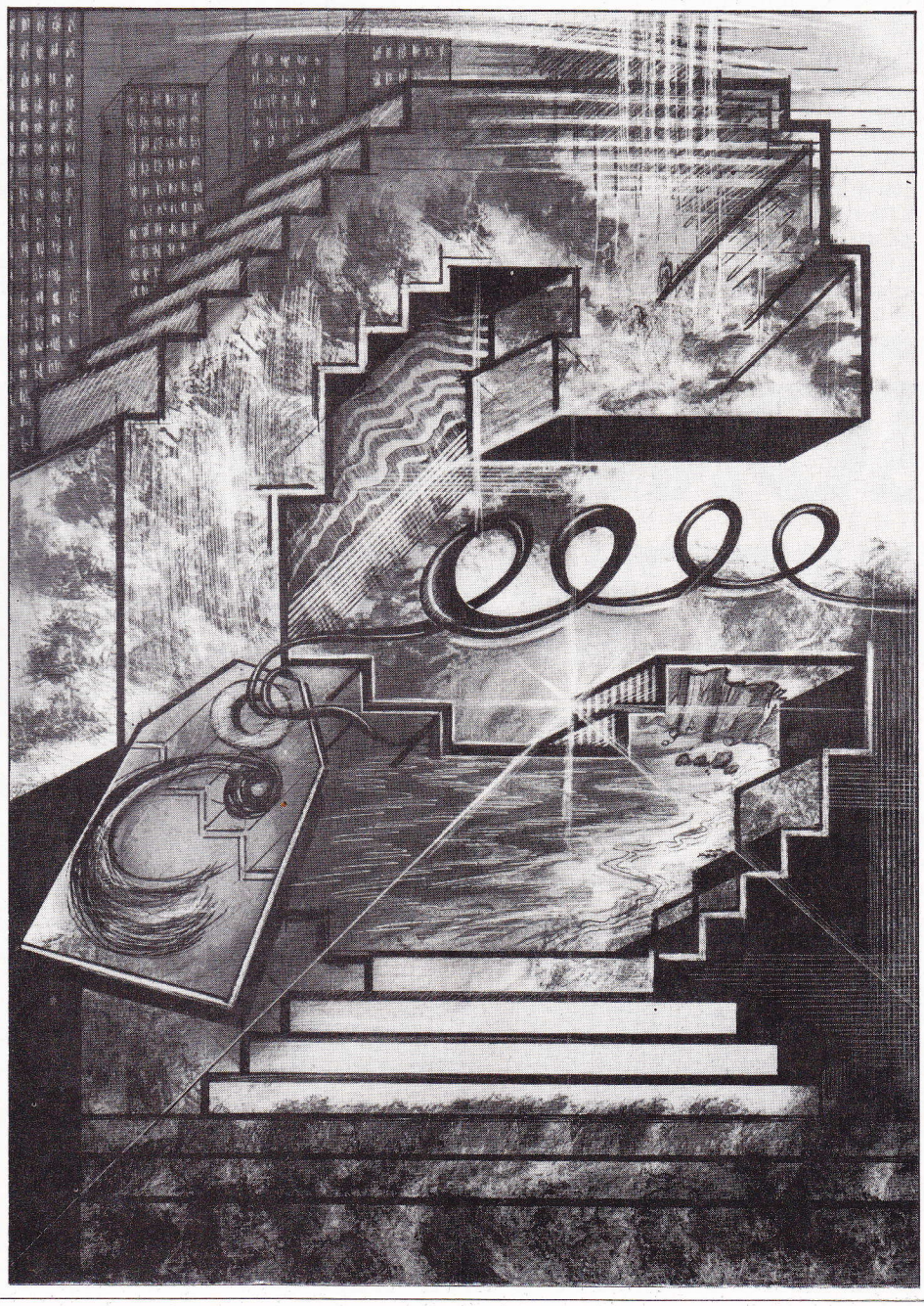
Within the braces ('{', '}') we go on to define each member of the structure. In our example we've chosen, totally arbitrarily, to have a character pointer called *name*, an integer called *i* and a pointer to another *mystruc* type structure called *ptr*. Now, each member of our structure does not itself constitute a variable; the names and types of each member simply pre-warn the compiler of what it can expect to find in a *mystruc* structure whenever it subsequently encounters one.

Notice that each member inside a structure can have any type, including other structures and pointers to structures. Now, our definition tells the compiler all about the '*mystruc*' type of structure, and because we've given the definition a tag, we can refer to that type of structure anywhere else in the program simply by writing **struct mystruc <something>**. This means that we could declare a few variables as being '*mystruc*' structures like this:

```
struct mystruc first,second
[20],*third;
```

This declares *first* as being a straight *mystruc* structure, *second* as being an array of twenty *mystruc* structures, and *third* as being a pointer to a *mystruc* structure. If this statement occurs in a place where variables may be declared then space will be reserved by the compiler for these three variables and they come into existence.

Before we look at how we can access each member of a structure, it is important that we get the method of declaring them sorted out, as it's



quite complicated. The method we used above was to define the structure template with a tag, and use this tag subsequently in real declarations. We could equally have followed the template definition with our declarations:

```
struct mystruc{
                char *name
                int i;
                struct mystruc *ptr;
                } first,second[20],
                *third;
```

This technique is a lot more compact though note that where we declare the variables within our program will determine their scope. That is, if this declaration occurred outside of any function, the variables would be global.

Less frequently used and much less useful, is the following:

```
struct{
      char *name;
      int i;
      struct mystruc *ptr;
      } first,second[20],*third;
```

You probably see the problem straightaway! As the structure was defined without a tag, the compiler has no idea what a 'mystruc' structure is, so it will complain about the **struct mystruc *ptr;** part of the declaration. So, it is good practice to tage each structure definition.

Accessing Structure Members

Now that we have sorted out the ways in which structures may be defined and declared, we can look at how to use them. If we take our identifier *first*, which we declared earlier as being a *mystruc* type structure, we can use the "structure member operator" to get at each member. The symbol for this operator is a full stop, '.' (Pascal devotees will recognise it as being the same as the operator used to refer to individual members on a record.) It is used like this:

structure . member

and, as it refers to a location in which a value may be stored, the expression refers to an lvalue. So, to put a value into a member of a particular structure, we type

first.i=25;

and to get the value of a member, we use

```
if (first.i==25) puts("\nSuccess!");
```

or whatever.

So far so good, but the number of operations which may be applied to structures is limited. The problem is that we cannot pass whole structures to functions and cannot perform ordinary mathematical operations upon a structure as a whole. Certainly we may pass the values of individual members, but if we want a function to act upon a structure we generally need to pass it the address of our structure. And you know what addresses mean – pointers!

As an example, suppose we had a function called *doit* which needs some of the values in our *first* structure to act upon. We could pass all of the necessary values as parameters, but this would be messy:

```
first.i=doit(first.name,first.i);
```

and later on

```
doit(par1,par2)
char *par1;
int par2;
{
return some value ;
}
```

A much easier and far more elegant way would be to pass *doit* the address of **first**, so that it can treat this address as a pointer to a structure:

```
doit(&first);
```

and then

```
doit(par1)
struct mystruc *par1;
{
body of function
}
```

Yet how do we allow the body of *doit* to refer to individual items within the structure? It has the address of a structure, not the structure itself. Our little exercise in pointers last month should help. We could use

(*par1). member

(The parentheses are necessary because '.' has a higher precedence than '*'). This is quite satisfactory and will certainly work, but as structures near enough demand the constant use of pointers, the C language provides another operator to shorten the amount of typing. This is known as the "structure pointer operator", and the symbol used is ->. Using this operator, the expression above would be written as

par1-> member

which is far easier to type! It is important to remember that

(* structure_pointer).member

and

structure_pointer->member

are identical in every respect, so that the -> operator is nothing more than convenient shorthand.

Just as we cannot pass whole structures to functions nor may we return whole structures from them. Once more pointers come to our rescue and we can declare a function as returning a pointer to a structure in the standard way:

```
struct mystruc *myfunc();
```

This declares *myfunc* as being a function which returns a pointer to a *mystruc* type structure. All these pointers, although initially confusing, soon become second nature as they are so fundamental. Remember that a pointer to a structure is no different to any other sort of pointer, so it may be incremented, decremented, added to integers, and so on. This is particularly useful when we're dealing with arrays of structures.

You may remember that our declaration of *first* also included the declaration of *second*, which was an array of twenty *mystruc* structures. This means that second is an identifier which refers to an array, and as it is the array name itself it follows the same rules as other array names. In particular, it is NOT an lvalue. Naturally, we can pass the address of the array to functions and use it in expressions.

In much the same way as we can initialise every element of an array using a pointer and a loop:

```
for (i=0;i<20;i++) *array++=0;
```

so too can we initialise members of structures in an array given a pointer to that array:

```
for (j=0;j<20;j++) (ptr++)->i=0;
```

Having examined all the concepts surrounding structures, let's look at a typical program segment using structures to see if we can work out what it's doing.

Somewhere in our program, we have a structure definition thus:

```
struct term { long elem; char
                op_t; };
```

and one of the functions in our program declares a large array of these structures which it needs to manipulate. Part of the manipulation is carried out by a function called **level5**, which is passed a pointer to the start of the array of structures and an integer which tells **level5** how many elements of the array need to be manipulated. Remember that each element of the array is a *term* type structure. We'll continue the story with the definition of the function:

```
level5(p,num)
struct term *p;
int num;
{
 struct term *q,*r,*s;
 long docalc();
 q=p;
 while (p-q<num)
 {
  while (p->op_t==NULL &&
  p-q<num) ++p;
  if (p-p==num) return;
  r=p+1;
  while (r->op_t==NULL &&
  r-q<num) ++r;
  if (r-q>=num) return;
  s=r+1;
  while (s->op_t==NULL) ++s;
  s->elem=docalc(p->elem,s
  ->elem,r->elem);
  p->op_t=r->op_t=NULL;
  p=s;
 }
}
```

This is nothing like as complicated as it looks! The function contains three local variables *q*, *r* and *s*, which are declared as being pointers to *term* type structures. Notice that a function used by *docalc* is declared as returning long; if this declaration wasn't included, the compiler would expect *docalc* to return an integer.

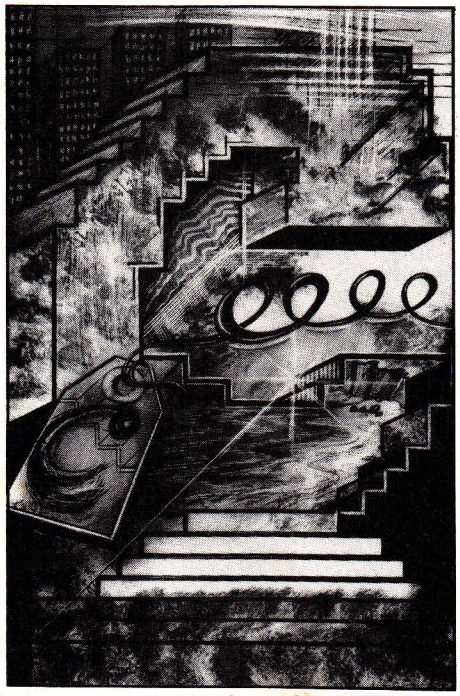
As *p*, *q*, *r* and *s* are all pointers to *term* type structures, it is perfectly in order for us to assign *p* to *q*, which we do with

q=p;

Likewise, now that *p* and *q* point to the same array, subtraction of the pointers, which yields an integer value corresponding to the number of elements between *p* and *q*, is quite legal. So, the expression

(p−q<num)

will be TRUE (1) when the number of elements between *p* and *q* is less than the number of elements to be considered (num). Within the body of the while loop controlled by this expression, we increment *p* (with ++*p*) until the element it points to has an *op_t* member which is not NULL or until *p−q* is equal to *num*. This means that *p* is made to skip over each structure in the array until one is found whose *op_t* member does not have the value NULL. Once this has been found, the address of the next element in the array of structures is put into *r* with



r=p+1;

and *r* is then incremented until it too points to a structure whose *op_t* member is not NULL. The same operation is carried out with *s*. This leaves us with *p* pointing to the first non-NULL structure (actually the first structure in the array with a non-NULL *op_t* member) in the array, *r* pointing to the next, and *s* to the one after that. The *elem* members of these three structures are then passed to the *docalc* function, which performs some operation and returns a value which we put into the *elem* member of the structure in the array pointed to by *s*. After that, the *op_t* members of the structures pointed to by *p* and *r* are set to NULL, and the value of *s* is put into *p*. The while loop then reiterates until its terminating condition becomes FALSE (0).

So, a function which looks fairly complicated actually turns out to be rather simple. This is really the essence of C: a little bit of study soon turns even the most complicated operation into something easily understood, although it must be said that a little help from the author always comes in handy!

Next month, we'll write a program which uses structures for a real purpose, and we'll also look at another related data type called the "union". As Metacomco has just released Lattice C for the QL, we'll also be taking a close look at that to see if it lives up to expectations.

CLASSIFIED

RELEASE THE POWER OF THE QL

QL C £99.95 INC. VAT

New - a professional specification C Compiler for the QL! Designed by LATTICE®, QL C is a complete implementation of the Kernighan and Ritchie definition of C. QL C is endorsed by Sinclair Research; it is the most powerful and fully featured C compiler available for the QL.

- > Complete Kernighan and Ritchie implementation.
- > Proven LATTICE® design, compatible with LATTICE® compilers on IBM PC, etc.
- > True compiler producing native 68000 code.
- > Full floating point arithmetic.
- > Powerful data types such as pointers, arrays, structures, and unions.
- > Comprehensive library of UNIX, QDOS and utility functions.
- > Over 90 detailed error messages.
- > Separate compilation.
- > Macros, conditional compilation, and other pre-processors.
- > Linker to link program modules; will also link to other relocatable binary modules.
- > Chosen by Sinclair Research as the C Compiler for the QL.

Every DEVELOPMENT KIT includes Metacomco's popular screen editor, and a detailed manual. All KITs will operate either on a standard QL or else using QL peripherals such as floppy disks or memory expansion.

Available from W.H. Smith, John Lewis, HMV, Menzies, Dixons and other leading retailers or direct from Metacomco.

METACOMCO

26 Portland Square, Bristol BS2 8RZ.
Tel: Bristol (0272) 428781

Trademarks: UNIX-AT+T Bell Laboratories; QL, QDOS - Sinclair Research Limited.

QL PASCAL £89.95 INC. VAT

Pascal compiler which conforms fully to the ISO 7185 international standard. Approved by Sinclair Research.

Features: fast single pass compilation — no intermediate stages; produces native 68000 code; full ISO 7185 level 0; direct addressing of complete QL address space; easy to use QDOS interfaces; very large sets and arrays.

ASSEMBLER £39.95 INC. VAT

A high specification macro assembler supporting the full Motorola instruction set.

Features: external references; absolute, position independent, and relocatable code; linker; precise error messages; formatted listings; macro expansions; conditional assembly; and a large range of directives.

BCPL £59.95 INC. VAT

A true compiler, ideal for systems programming - writing utilities, games and applications.

Features: generates native 68000 code; run time library includes easy QDOS interfaces; link loader links separately compiled segments; modules can be linked with Pascal or Assembler.

LISP £59.95 INC. VAT

A LISP interpreter for exploring "The language of artificial intelligence".

Features: Turtle graphics; compatible with LISP for the BBC micro; full support of QL features; structure editor; prettyprinter; garbage collector and tracer.

ATARI ST

Ring or write for details!

WE NOW HAVE SOFTWARE AVAILABLE FOR THE ATARI ST.

Phone today or post this coupon to: Metacomco, 26 Portland Square, Bristol BS2 8RZ.

Please send me QL C £99.95 ☐ QL PASCAL £89.95 ☐ ASSEMBLER £39.95 ☐ BCPL £59.95 ☐ LISP £59.95 ☐ MORE INFORMATION ☐ I enclose a cheque for £_____ or debit my ACCESS/VISA No: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Card Expiry Date_____

NAME

ADDRESS

SIGNATURE

POSTCODE TEL NO:

Price includes VAT, postage and packing UK mainland only. Delivery, allow up to 28 days. QL10

SOFTWARE FILE

Marcus Jeffery and Adam Denning combine to give an expert appraisal of what's new on the QL.

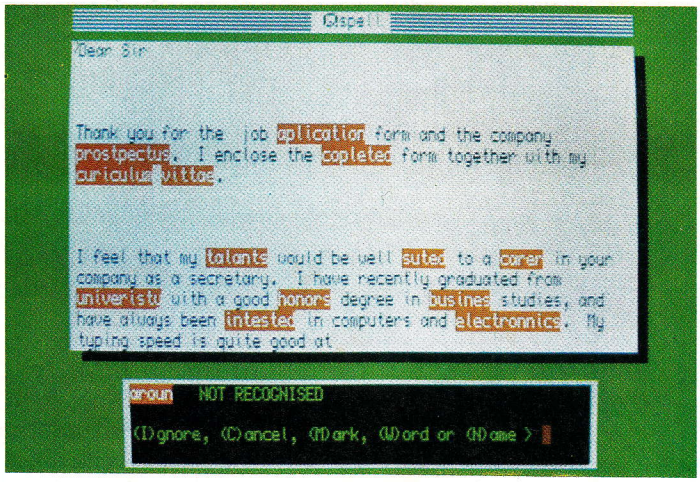
Quick Mate

Qspell, published by Eidersoft, is a spelling-checker and proof-reader, designed for use in conjunction with the Quill wordprocessor. Just the thing for illiterates like wot we journalists are!

You may be surprised to learn that the 70K default dictionary contains about 25,000 words, using data compression techniques. In addition to this, there is room for about 1000 words of your own. These are most likely to be added during proof-reading. If *Qspell* comes across a correct word with which it is not familiar, then you have the option of adding the word to the dictionary, either as a normal word or proper noun. So as not to use too much of the remaining space, *Qspell* only remembers the newly-learnt words whilst the computer is switched on, though you are able to save the dictionary for a permanent record. The master cartridge includes a routine to 'doctor' a copy of Quill, so that it can be used to load and edit *Qspell* proof-read files.

Word freaks will be especially attracted by the dictionary-search facilities available. These include Crossword search which matches words to such inputs as 'm?ga??ne' or just 'maga*' (this finds both the singular and plural versions), Anagram search and Puzzle search. This last option will list all the words which are contained within other words (not unlike a competition or two we've run in the past). Just for fun, we tried it out with words of 15 or more letters, giving it two copies of each letter in the alphabet, and were quite impressed to find 22 words of this length. Three copies of each letter increased this total to 53.

Finally, a couple of routines allow you to compact dictionaries after modifying them and configure the *Qspell* printer drivers. Altogether a well-presented and well documented package. If you have a need for this sort of program, then *Qspell*'s ideal.
Eidersoft



Run Of The Mill

As you probably guessed, given the rather blatant name, *Zapper* is one of your typical "shoot 'em up" games.

Based loosely on the original Galaxians-type of game, the demonstration mode shows a total of eight levels. With your supply of three spaceships, you must zap away at the slow-moving, descending aliens. Having cleared the screen, another set appears which, although a different type, seem to follow very similar patterns. Perhaps the saving feature of the game is that the aliens do move fairly erratically and speed up considerably when there are only a few left on the screen. This makes the game harder than it first appears, though practiced arcaders will, no doubt, find the game overly simple, even with increasing speed. Just in case you do find the game intriguing enough to play for any length of time, there's an extra ship after 10,000 points. I lost interest before discovering whether you get another ship after 20,000.

When shooting the nasties, you have to be fairly accurate with a near-centre hit. It can be quite disconcerting when you're used to getting in that final shot before sliding out from under the attacking alien. On *Zapper* this doesn't work firstly because that last minute shot will probably not be near-centre, and secondly because the control response is too sluggish.

The overall effect is of a game hurriedly thrown together to cash-in on the present lack of available software for the QL. Even the function keys for 'sound on' and 'sound off' are the wrong way round.

If you're particularly interested in zapp-em games, then it's worth considering *Zapper*. However, if you are that keen then this version isn't likely to hold your attention for very long.
Eidersoft

The Incorruptible

What with Sinclair's determination to go on with microdrives, *Cartridge Doctor* must be the product everybody has been waiting for. Essentially a semi-intelligent program to recover lost data from damaged cartridges the package also includes a number of generally useful file utilities.

The main program called *autoclone* allows you to copy from one cartridge to another. Nothing unusual here until you realise that in the course of copying, the routine will automatically extract programs which have been accidentally deleted or piece together those that have been corrupted. A cure then for the 'bad or changed medium' message, or the even more infuriating endlessly running tape associated with damaged cartridges.

Other features include a file patch utility which is used by *autoclone* but may also be accessed manually. A salvage section allows files to be copied in part to a new file. There are also facilities to list directories and transliterate. The latter allows you to copy across a file whilst replacing every occurrence of a given character with another.

If you make use of microdrives this package is virtually indispensable. Certainly, during the period of review, we had cause to use it extensively and at one point found that it saved us over six hours work.
Talent

Operating Tool

The *SuperBasic Extension EPROM* (or *SEE* for short) is an unusual conglomeration of routines (about 70) designed to appeal to *all* QL users *some* of the time. Its obvious advantage over other toolkits is that everything is in

EPROM, rather than taking up valuable RAM.

In many areas *SEE*'s routines overlap with Sinclair Research's own *Toolkit*. Also, similar to that product, the usefulness of the routines depends upon the level at which you are programming the QL. For instance, a large number of routines are devoted to job and data management, stream and file handling and use of machine code.

On the whole, routines tend to be short cuts to using standard QDOS traps. For example, the new command QTRAP allows you to pass and receive information from all three QDOS traps in one go.

For the less able programmer *SEE* provides a number of genuinely useful procedures. On the utility side, the package adds trace, function key definition and hex/decimal conversion facilities. Elsewhere there are additional graphics commands for alternative fonts, cursor control and window definition. In the latter case commands SSAVE, SLOAD and SSHOW enable windows to be refreshed when they have been overwritten and can lead to the creation of Macintosh or GEM style pull-down menus. Finally, games programmers will be interested in HIGHS, SCORE and SETHIGH which allows for the automatic creation and printing of high score tables.

Whilst *SEE* may well be of use as an advanced toolkit it is essentially for those looking to *operate* the QL as opposed to *program* it.

Hi Soft

Modem Moves

The software that comes with the Brightstar modem will cater for the needs of most users. *Qterm*, a rather special terminal emulation package from Datamanagement in York is aimed at the "out and out" communications devotee.

The main purpose of the program is to allow Brightstar users to join and log onto Datamanagement's dedicated QL database/bulletin board called 'Citadel'. As such it offers all the facilities required by this service. The most important of these is the implementation of the Xmodem Ward-Christerpherson data transfer protocols.

These allow file transfer between two modems (both running the software) to be entirely corruption free and consequently thoroughly reliable. A claim that is difficult to believe! However, we put it to the test and transferred from London to York an 86K file. Suffice it to say the transfer was completely successful, taking some twenty minutes at 1200 Baud. We were impressed and the people at the other end who were in dire need of the software were extremely grateful.

Qterm despite its specialised appeal is fairly straightforward to operate. Its various features are accessed via the function keys. It makes good use of the QL's windowing functions and is well documented. In short, a professional product geared for the computer professional.

Datamanagement

Character Change

The *Font Editor* consists of a set of utility programs which allow you to do all the things you ever wanted to do, and a few you probably didn't, with the QL fonts. These are the character sets available on the QL which Sinclair is so fond of calling 'founts'.

The main program (the editor itself) is similar to that expected from the typical sprite designer. You first of all fetch a font, either from a file or a QL window (this will probably be the standard system font to begin with). The entire font will be displayed in the standard mode 8 character size, and a blown-up image of the current character will be displayed in the edit window. A combination of CTRL, the cursor keys and the space bar will allow you to alter this definition to any you may choose. Other options allow you to reverse and inverse, and change characters. There's even a facility which allows you to print out a message in any colour and character size to see how your new character font is shaping up.

In addition to the Font Editor, the cartridge also contains a small SuperBasic program, which calls QDOS and allows you to load and change character fonts in your own programs. This is heavily documented for personal use and supported by two alternative files for system font one and font two.

There is nothing particularly special about this package. Most competent programmers could easily produce something similar. The question to ask, assuming you want to be able to alter the QL fonts, is whether or not it's worth spending the time writing the appropriate routines when everything is provided for only £8.

Saltigrade Software

Perfect Pascal

Metacomco have released yet another winner. This time it's Pascal, and has the distinction of conforming to the ISO standard for that language. This guarantees a full implementation (including pointers, enumerated types, and so on) and makes software written on the QL compatible with many other machines.

The *Pascal Development Kit* is supplied on an EPROM and two microdrive cartridges. These are respectively, the compiler itself, and the screen editor/run-time system. There is also a weighty tome containing basic information on using the kit and a language guide. Not for teaching, certainly, but a useful reference book.

The first thing to learn to use is the full screen editor. This has many features but is easily mastered. A variable size window, shows the current work area within the program, and other areas can be found simply by using the cursor keys, then inserting and deleting as required. Combining the cursor keys with SHIFT and ALT will give you larger word and page jumps. In addition to this there are a number of 'extended' commands to give you even greater control. These allow such operations as splitting and joining lines, searching for and replacing text, setting margins and tab distance, and so on. The function keys are also used. For example, F1 will redraw the window information in case it has been corrupted (say, by another job), whilst F4 will repeat the last extended editor command line.

Upon invoking the compilation program from microdrive you're initially prompted for the input source file. An optional listing file (showing program structure, indentifier usage, and highlighting any compilation errors) and the object code file are then entered. You also need to specify the workspace required (default 20K on a basic machine), and whether range-checking can be omitted. This latter facility allows you to speed up execution time slightly, once you have a program working correctly.

Finally, an additional prompt allows you to include ISO standard extensions. These have been included to make use of the QL-dependant features. To the dedicated programmer, perhaps the most interesting and useful of these additional commands is EXTERNAL. This allows programs written using Metacomco's BCPL compiler or assembler to be linked with a Pascal program. Other extension commands include QTRAP which allows information to be passed to and from the 68000 TRAP instructions and INCLUDE which allows additional program segments to be included in the source program at compilation time, allowing the user to create his own library routines. The INCLUDE statement can be used to integrate a suite of graphic routines supplied in the package, allowing the use of windows, plotting, random numbers, time and date.

Before a compiled program can be finally run, it must be linked to the run-time library system. This is easily done, and Metacomco have stated that though the development kit is copyright, the run-time system can be copied as often as necessary, allowing the final product to be used in any way you wish.

An excellent package, and highly recommended for both Pascal veterans and beginners alike.

Metacomco

SUPPLIERS

Datamanagement, *12 Larch Way, Haxby, York*
Qterm: £19.95

Eidersoft, *The office, Hall Farm, North Ockendon, Upminster, Essex*
Qspell: £19.95
Zapper: £10.95

Hisoft, *13 Goosacre, Cheddington, Leighton Buzzard, Beds*
SuperBasic Extension EPROM: £29.95

Metacomco, *28 Portland Square, Bristol*
QL Pascal Development Kit: £89.95

Saltigrade Software, *31 Royal Terrace, Edinburgh*
Font Editor: £8.00

Talent Computer Systems, *Curran Building, 101 St James Rd, Glasgow*
Cartridge Doctor: £14.95

The Sinclair Vision QL. Designed to perform.

For sheer good looks, superb design and high performance there is nothing to touch the Sinclair Vision QL colour monitor.

Developed in conjunction with Sinclair Research, this beautifully compact unit with its 12″ non-glare tube, 85 column text display and high resolution colour graphics is the perfect partner to the Sinclair QL computer. Comes complete with lead and connector for instant use with the QL.

Take a closer look and you'll see that it also represents amazing value for money.

You'll find the brilliant Sinclair Vision QL at your local dealer, supplied exclusively by DDL. The data distribution company with a product range that is second to none.

Ring us on Ascot (0990) 28921 for the name of your nearest dealer.

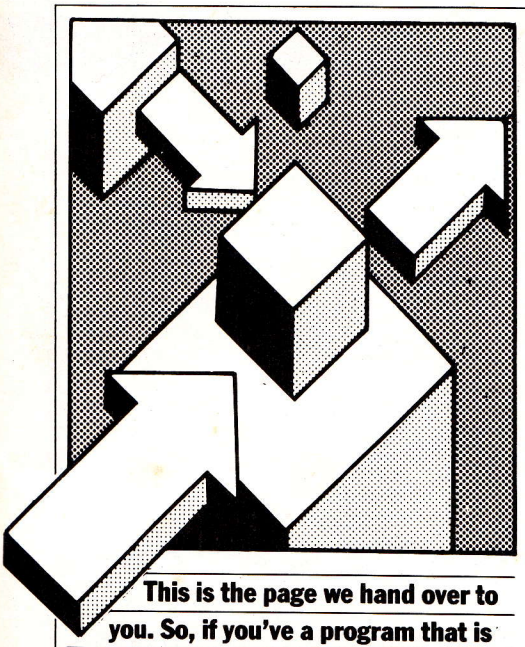
Then you can call in and see for yourself what makes the Sinclair Vision QL colour monitor such a well-designed little performer.

DDL

5 King's Ride Park,
Ascot, Berks. SL5 8BP
Tel: 0990 28921
Telex: 846303 DD LTD G.

THE NEW FORCE IN DISTRIBUTION

THE PROGS



This is the page we hand over to you. So, if you've a program that is worthy of consideration, send it to 'The Progs', QL User, Priory Court, 30-32 Farringdon La, London EC1R 3AU. We pay for everything published at the usual page rates.

Mini Monitor

Richard Cross

Machine code is the only way to get the most out of your QL. However, if you're learning assembler you'll soon discover that whilst your programs may compile without problems it does not follow that they will run as intended. The odds are that you will have made a logic error. There are two ways to track this down. One is to pore over your source code going over the entire program again and again hoping that intuition will make up for inexperience. The other is to use a monitor to track down the problem code whilst the machine code is running.

The following monitor should provide an extremely useful debugging tool. Some 3K in length it will happily multitask on your QL leaving plenty of room for other programs. It can be run by typing "EXEC mdv1_mon_ex" and then pressing CTRL-C. All input must be in hex and preceded by a command letter. Commands are:

R Register dump
C Rn,xxx Store xxx in Register n
K Kill monitor
J xxx Jump to location xxx
A.b xxx Byte by byte, alter memory from location *xxx* onwards. Address, contents in hex, ASCII and a? prompt will display. ASCII characters may be entered provided they are enclosed within quotes – (eg, "*a*") otherwise all values are deemed to be in hex>. Press <ENTER> to keep current values and type to "STOP" return to monitor.
A.w As **A.b** but alters in word size chunks
A.l As **A.b** but alters in longword chunks
D xxx,yyy Will dump lines of memory (Hex and ASCII) in 8 bytes chunks to a specified device. Can be aborted by pressing <SPACE>.
M xxx Will move lump of memory and prompts for destination and size of chunk.
T xxx Will trace the machine code starting from location *xxx*. After each instruction the monitor dumps registers and awaits either <ENTER> to continue or <ESC> to end. Pressing <N> will delay the dump for a specified number of instructions.

Breakpoints can be used by including a trap #5 – #15 in your own programs, which will halt execution and return to the monitor. If you wish to insert them from the monitor use the command *A.w 4E46* wherever desirable.

All 68000 errors are trapped, causing a message to be displayed, registers dumped and a return to the monitor. Finally, note that even running the monitor it is easy to crash the QL by corrupting the system stack by moving without care or jumping to random locations.

Though it would seem fair to assume that most people wanting a monitor will already have an assembler we have nevertheless chosen to publish a SuperBasic hex loader. This has the advantage that it is independent of any variations in mnemonics, addressing modes and directives and, with checksums at the end of each line ensures errors in typing-in are instantly identifiable. Object and (Computer One) Assembler source listings are available from our Microdrive Exchange. Otherwise you'll need to check each line carefully.

```
10 REMark **** QL User - Monitor by R. Cross ****
20 length=2970:f$='mdv1_mon_ex'
25 start=RESPR(length+10):RESTORE :addr=start
30 FOR a=1 TO (length/11)
35 c=0
40 FOR b=1 TO 11
45 READ v:POKE addr,v:addr=addr+1:c=c+v
50 END FOR b
55 READ checksum:IF c<>checksum THEN PRINT 'error
in line ';(a-1)*10+100:STOP
60 END FOR a
65 PRINT 'saving as ';f$:SEXEC f$,start,length,512
:STOP
100 DATA 96,14,4,0,74,251,0,8,109,111,110,777
110 DATA 105,116,111,114,32,97,0,5,86,67,250,983
120 DATA 8,82,114,255,112,7,78,65,69,250,12,1052
130 DATA 76,36,143,52,120,0,198,67,250,8,150,1100
140 DATA 78,146,42,72,52,120,0,198,67,250,8,1033
150 DATA 150,78,146,40,72,69,250,11,210,72,210,130
8
160 DATA 48,0,74,57,0,2,128,52,102,12,114,589
170 DATA 1,116,0,118,255,32,77,112,45,78,67,901
180 DATA 67,250,8,126,97,0,3,246,97,0,6,900
190 DATA 196,67,250,8,166,97,0,3,234,97,0,1118
200 DATA 0,136,69,250,11,158,72,210,48,0,69,1023
210 DATA 250,11,158,52,129,38,72,18,19,12,1,760
220 DATA 0,97,101,10,12,1,0,122,98,4,4,449
230 DATA 1,0,32,12,1,0,84,103,0,0,224,457
240 DATA 12,1,0,74,103,0,0,130,12,1,0,333
250 DATA 82,103,0,0,100,12,1,0,77,103,0,478
260 DATA 3,168,12,1,0,65,103,0,1,2,12,367
270 DATA 1,0,68,103,0,2,92,12,1,0,67,346
280 DATA 103,0,4,6,12,1,0,75,103,66,12,382
290 DATA 1,0,10,103,8,67,250,8,180,97,0,724
300 DATA 3,120,65,250,11,48,76,208,48,0,69,898
310 DATA 250,11,132,46,82,44,124,0,0,0,0,689
320 DATA 96,0,255,112,52,60,0,128,54,60,255,1072
330 DATA 255,67,250,10,144,112,2,78,67,65,250,1300
340 DATA 10,136,78,117,97,0,6,24,96,204,67,835
350 DATA 250,9,190,97,0,3,60,114,255,118,0,1096
360 DATA 112,5,78,65,215,252,0,0,0,2,69,798
370 DATA 250,10,238,114,0,50,18,87,65,32,75,939
380 DATA 97,0,6,228,74,128,102,48,47,4,97,831
390 DATA 0,6,202,40,95,78,148,72,231,128,128,1128
400 DATA 64,192,65,250,11,28,48,128,65,250,11,1112
410 DATA 18,32,188,0,0,0,0,76,223,1,1,539
420 DATA 97,0,6,146,97,0,5,192,96,0,255,894
430 DATA 116,67,250,8,54,97,0,2,226,96,0,916
440 DATA 255,104,215,252,0,0,0,2,69,250,10,1157
450 DATA 152,114,0,50,18,87,65,32,75,97,0,690
460 DATA 6,142,74,128,102,218,47,4,65,250,10,1046
470 DATA 220,32,188,0,0,0,2,97,0,6,106,651
480 DATA 78,64,0,124,128,0,2,124,135,31,40,726
490 DATA 95,78,148,78,64,2,124,7,31,96,0,723
500 DATA 255,142,82,139,12,19,0,46,102,38,82,917
510 DATA 139,12,19,0,76,103,40,12,19,0,108,528
520 DATA 103,34,12,19,0,87,103,32,12,19,0,421
530 DATA 119,103,26,12,19,0,66,103,24,12,19,503
540 DATA 0,98,103,18,32,76,67,250,7,210,96,957
550 DATA 0,255,128,124,4,96,6,124,2,96,2,837
560 DATA 124,1,215,252,0,0,0,2,69,250,10,923
570 DATA 20,114,0,50,18,91,65,32,75,97,0,562
580 DATA 6,10,74,128,102,0,255,86,47,4,18,730
590 DATA 60,0,10,54,60,255,255,32,77,112,5,920
600 DATA 78,67,67,250,9,240,34,151,52,120,0,1068
610 DATA 254,65,250,8,216,78,146,97,0,2,6,1122
620 DATA 97,0,2,16,34,87,65,250,8,200,12,771
630 DATA 6,0,1,103,6,52,120,0,254,96,4,642
640 DATA 52,120,0,250,78,146,42,6,227,141,65,1127
650 DATA 250,8,172,48,133,97,0,1,220,97,0,1026
660 DATA 1,230,65,250,8,158,48,188,0,8,32,988
670 DATA 77,118,255,34,87,52,6,112,7,78,67,893
680 DATA 67,250,6,220,97,0,1,192,32,77,97,1039
690 DATA 0,254,108,12,144,115,116,111,112,103,0,10
75
700 DATA 254,74,12,144,83,84,79,80,103,0,254,1167
710 DATA 64,12,16,0,10,103,50,12,16,0,34,317
720 DATA 103,54,12,16,0,39,103,48,72,193,83,723
730 DATA 129,97,0,5,98,74,128,102,0,254,174,1061
740 DATA 32,87,12,6,0,1,103,10,12,6,0,269
750 DATA 2,103,8,32,132,96,6,16,132,96,2,625
760 DATA 48,132,32,87,209,198,46,136,96,0,255,1239
770 DATA 72,87,65,178,70,102,44,12,6,0,1,637
780 DATA 103,30,12,6,0,2,103,16,24,40,0,336
790 DATA 1,225,140,82,136,24,40,0,1,225,140,1014
800 DATA 82,136,24,40,0,1,225,140,82,136,24,890
810 DATA 40,0,1,96,0,255,174,67,250,6,210,1099
820 DATA 96,0,254,86,84,139,69,250,8,248,114,1348
830 DATA 0,50,18,87,65,107,16,112,0,12,51,518
840 DATA 0,44,0,0,103,18,82,64,178,64,98,651
```

```
850 DATA 242,67,250,6,114,97,0,1,20,96,0,893
860 DATA 253,154,72,231,192,16,32,75,50,0,97,1172
870 DATA 0,4,202,74,128,102,0,254,22,38,68,892
880 DATA 76,223,1,3,65,240,0,1,146,64,83,902
890 DATA 65,97,0,4,178,74,128,102,0,253,254,1155
900 DATA 83,132,47,11,118,255,32,77,112,5,114,986
910 DATA 10,78,67,47,4,65,250,8,238,81,208,1056
920 DATA 67,250,5,136,97,0,0,200,97,0,253,1105
930 DATA 102,34,72,12,17,0,10,103,40,83,129,602
940 DATA 51,65,255,254,114,255,118,2,65,233,255,16
67
950 DATA 254,112,1,78,66,74,128,103,10,32,76,934
960 DATA 52,120,0,204,78,146,96,198,42,72,65,1073
970 DATA 250,8,178,80,208,40,31,67,250,8,84,1204
980 DATA 34,151,65,250,7,64,52,120,0,254,78,1075
990 DATA 146,97,0,0,106,97,0,0,116,34,87,683
1000 DATA 65,250,7,44,52,120,0,254,78,146,97,1113
1010 DATA 0,0,86,34,87,88,137,65,250,7,26,780
1020 DATA 52,120,0,254,78,146,97,0,0,68,97,912
1030 DATA 0,0,78,118,255,34,87,52,60,0,8,692
1040 DATA 112,7,78,67,18,60,0,10,112,5,78,547
1050 DATA 67,80,151,71,250,4,158,112,17,78,65,1053
1060 DATA 8,1,0,6,102,4,81,204,255,156,88,905
1070 DATA 143,65,250,8,66,74,16,103,0,252,170,1147
1080 DATA 32,77,112,2,78,66,96,0,252,160,67,942
1090 DATA 250,6,200,32,77,52,120,0,208,78,146,1169
1100 DATA 78,117,67,250,6,180,96,240,32,76,96,1238
1110 DATA 238,84,139,69,250,7,188,114,0,50,18,1157
1120 DATA 87,65,32,75,97,0,3,178,74,128,102,841
1130 DATA 0,252,254,47,4,67,250,4,190,97,0,1165
1140 DATA 255,220,97,0,252,122,72,193,83,129,97,15
20
1150 DATA 0,3,150,74,128,102,0,252,226,47,4,986
1160 DATA 67,250,4,172,97,0,255,192,97,0,252,1386
1170 DATA 94,72,193,83,129,97,0,3,122,74,128,995
1180 DATA 102,0,252,198,76,223,3,0,193,73,179,1299
1190 DATA 200,98,10,74,132,103,20,18,216,83,132,10
86
1200 DATA 96,246,209,196,211,196,74,132,103,6,19,1
488
1210 DATA 32,83,68,96,246,96,0,252,18,65,250,1206
1220 DATA 7,72,114,0,50,16,65,250,6,184,12,776
1230 DATA 65,0,6,111,0,254,90,12,40,0,68,646
1240 DATA 0,2,103,0,0,36,12,40,0,100,0,293
1250 DATA 2,103,0,0,26,12,40,0,97,0,2,282
1260 DATA 103,0,0,44,12,40,0,65,0,2,103,369
1270 DATA 0,0,34,96,0,254,46,97,0,0,40,567
1280 DATA 97,0,0,62,67,250,7,14,72,130,229,928
1290 DATA 74,35,132,32,0,97,0,2,2,96,0,470
1300 DATA 251,182,97,0,0,12,97,0,0,34,67,740
1310 DATA 250,7,18,96,226,47,8,65,232,0,5,954
1320 DATA 93,65,97,0,2,216,32,95,74,128,102,904
1330 DATA 2,78,117,88,143,96,0,252,28,20,40,864
1340 DATA 0,3,4,2,0,48,12,2,0,7,98,176
1350 DATA 2,78,117,88,143,96,0,253,212,44,79,1112
1360 DATA 79,250,3,72,114,11,72,122,0,218,81,1022
1370 DATA 201,255,250,72,122,0,30,72,122,0,232,135
6
1380 DATA 72,122,0,254,72,122,1,20,72,122,1,858
1390 DATA 42,72,122,1,64,72,122,1,86,46,78,706
1400 DATA 78,117,12,175,0,0,3,8,0,2,103,498
1410 DATA 0,0,100,47,8,65,250,252,8,177,239,1146
1420 DATA 0,6,102,6,32,95,96,0,0,82,32,451
1430 DATA 95,97,0,1,86,97,0,2,54,65,250,747
1440 DATA 6,176,74,144,103,4,83,144,96,58,97,985
1450 DATA 0,1,88,65,250,6,62,32,80,54,60,698
1460 DATA 255,255,112,1,78,67,12,1,0,27,103,911
1470 DATA 20,12,1,0,10,103,28,12,1,0,110,297
1480 DATA 103,28,12,1,0,78,103,22,96,216,92,751
1490 DATA 79,2,124,7,31,67,250,3,254,96,0,913
1500 DATA 0,252,97,0,2,2,78,115,67,250,3,866
1510 DATA 84,65,250,5,254,32,80,52,120,0,208,1150
1520 DATA 78,146,97,0,250,216,65,250,5,110,72,1289
1530 DATA 193,83,129,97,0,1,240,74,128,102,0,1047
1540 DATA 0,10,65,250,6,62,32,132,96,204,65,922
1550 DATA 250,5,212,32,80,67,250,3,98,52,120,1169
1560 DATA 0,208,78,146,96,192,97,0,0,194,92,1103
1570 DATA 79,2,124,7,31,97,0,1,156,97,0,594
1580 DATA 0,202,67,250,3,170,96,0,0,156,97,1041
1590 DATA 0,0,168,92,79,2,124,7,31,97,0,600
1600 DATA 1,130,97,0,0,176,67,250,3,168,96,988
1610 DATA 0,0,130,97,0,0,142,92,79,2,124,666
1620 DATA 7,31,97,0,1,104,97,0,0,150,67,554
1630 DATA 250,3,164,96,0,0,104,97,0,0,116,830
1640 DATA 92,79,2,124,7,31,97,0,1,78,97,608
1650 DATA 0,0,124,67,250,3,178,96,0,0,78,796
1660 DATA 97,0,0,90,92,79,2,124,7,31,97,619
1670 DATA 0,1,52,97,0,0,98,67,250,3,182,750
1680 DATA 96,0,0,52,97,0,0,64,92,79,2,482
1690 DATA 124,7,31,97,0,1,26,97,0,0,72,455
1700 DATA 67,250,3,184,96,0,0,26,80,143,97,946
1710 DATA 0,0,36,92,143,2,124,7,31,97,0,532
1720 DATA 0,254,97,0,0,44,67,250,3,196,65,976
1730 DATA 250,5,14,32,80,52,120,0,208,78,146,985
1740 DATA 96,0,249,208,47,8,65,250,5,82,32,1042
1750 DATA 175,0,10,65,250,5,78,48,175,0,8,814
1760 DATA 32,95,78,117,75,250,4,236,42,85,32,1046
1770 DATA 77,118,255,112,32,78,67,120,0,50,60,969
1780 DATA 0,0,52,4,54,60,255,255,32,77,112,901
1790 DATA 16,78,67,32,4,229,136,71,250,4,218,1105
1800 DATA 46,51,8,0,32,115,8,32,34,72,193,591
1810 DATA 137,8,128,0,0,193,137,34,81,69,250,1037
1820 DATA 4,182,72,210,3,128,44,124,0,0,0,767
1830 DATA 0,67,250,4,168,65,250,3,164,52,120,1143
1840 DATA 0,254,78,146,65,250,3,166,67,250,4,1283
1850 DATA 154,52,120,0,254,78,146,65,250,3,166,128
8
1860 DATA 67,250,4,144,52,120,0,254,78,146,38,1153
1870 DATA 4,6,3,0,48,67,250,3,114,19,67,581
1880 DATA 0,4,19,67,0,16,19,67,0,29,97,318
1890 DATA 0,252,146,82,68,12,4,0,8,102,0,674
1900 DATA 255,122,65,250,3,140,67,250,4,162,52,137
0
1910 DATA 120,0,254,78,146,65,250,3,140,67,250,137
3
1920 DATA 4,148,34,81,52,120,0,254,78,146,65,982
1930 DATA 250,3,160,67,250,4,136,52,120,0,246,1288
1940 DATA 78,146,67,250,3,82,97,0,252,84,78,1137
1950 DATA 117,47,14,77,250,4,46,72,214,63,255,1159
1960 DATA 44,95,67,250,4,92,72,209,192,0,78,1103
1970 DATA 117,77,250,4,26,76,214,63,255,77,250,140
9
1980 DATA 4,74,44,86,78,117,120,0,122,0,209,854
1990 DATA 193,22,32,12,3,0,97,101,10,12,3,485
2000 DATA 0,122,98,4,4,3,0,32,112,255,69,699
2010 DATA 250,0,34,82,128,182,26,103,10,12,0,827
2020 DATA 0,16,102,244,112,255,78,117,235,168,216,
1543
2030 DATA 128,88,133,83,129,74,129,102,204,112,0,1
182
2040 DATA 78,117,48,49,50,51,52,53,54,55,56,663
2050 DATA 57,65,66,67,68,69,70,0,0,0,0,462
2060 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2070 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2080 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2090 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2100 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2110 DATA 0,0,0,0,0,0,0,0,0,0,0,0
2120 DATA 0,0,0,0,0,0,0,0,0,0,9,9
2130 DATA 1,0,0,0,0,1,2,4,1,0,7,16
2140 DATA 1,204,0,132,0,28,0,8,2,1,0,376
2150 DATA 7,1,204,0,42,0,28,0,141,0,50,473
2160 DATA 81,76,32,85,115,101,114,32,77,97,99,909
2170 DATA 104,105,110,101,32,67,111,100,101,32,77,
940
2180 DATA 111,110,105,116,111,114,10,32,32,32,98,8
71
2190 DATA 121,32,82,105,99,104,97,114,100,32,67,95
3
2200 DATA 114,111,115,115,10,10,0,2,62,32,0,571
2210 DATA 31,68,101,118,105,99,101,32,100,101,102,
958
2220 DATA 32,45,32,60,101,110,116,101,114,62,61,83
4
2230 DATA 100,101,102,97,108,116,32,32,62,32,0,782
2240 DATA 0,8,84,111,32,63,32,32,62,32,0,456
2250 DATA 29,78,117,109,98,101,114,32,111,102,32,9
23
2260 DATA 98,121,116,101,115,32,116,111,32,109,111
,1062
2270 DATA 118,101,32,63,32,32,62,32,0,0,4,476
2280 DATA 32,32,63,32,0,40,72,111,119,32,109,642
2290 DATA 97,110,121,32,105,110,115,116,114,117,99
,1136
2300 DATA 116,105,111,110,115,32,98,101,102,111,11
4,1115
2310 DATA 101,32,114,101,116,117,114,110,32,63,32,
932
2320 DATA 62,32,0,22,85,110,114,101,99,111,103,839
2330 DATA 110,105,115,101,100,32,67,111,109,109,97
,1056
2340 DATA 110,100,32,10,0,8,66,97,100,32,72,627
2350 DATA 101,120,10,0,14,66,97,100,32,80,97,717
2360 DATA 114,97,109,101,116,101,114,10,0,40,66,86
8
2370 DATA 97,100,32,115,105,122,101,32,45,32,109,8
90
2380 DATA 117,115,116,32,98,101,32,91,46,108,93,94
9
2390 DATA 32,44,32,91,46,119,93,32,111,114,32,746
2400 DATA 91,46,98,93,32,10,0,20,87,114,111,702
2410 DATA 110,103,32,115,116,114,105,110,103,32,10
8,1048
2420 DATA 101,110,103,116,104,10,0,10,10,60,101,72
5
2430 DATA 115,99,97,112,101,62,10,0,21,66,114,797
2440 DATA 101,97,107,32,112,111,105,110,116,32,69,
992
2450 DATA 120,101,99,117,116,101,100,10,0,0,20,784
2460 DATA 80,114,105,118,105,108,101,103,101,32,86
,1053
2470 DATA 105,111,108,97,116,105,111,110,10,0,37,9
10
2480 DATA 84,114,97,112,32,111,110,32,111,118,101,
1022
2490 DATA 114,102,108,111,119,32,69,120,101,99,117
,1092
2500 DATA 116,101,100,32,45,32,79,118,101,114,102,
940
2510 DATA 108,111,119,10,0,0,28,67,72,75,32,622
2520 DATA 69,120,101,99,117,116,101,100,32,45,32,9
32
2530 DATA 111,117,116,32,111,102,32,114,97,110,103
,1045
2540 DATA 101,10,0,26,65,116,116,101,109,112,116,8
72
2550 DATA 32,116,111,32,100,105,118,105,100,101,32
,952
2560 DATA 98,121,32,122,101,114,111,10,0,37,73,819
2570 DATA 108,108,101,103,97,108,32,105,110,115,11
6,1103
2580 DATA 114,117,99,116,105,111,110,32,99,111,100
,1114
```

```
2590 DATA 101,32,101,110,99,111,117,110,116,101,11
4,1112
2600 DATA 101,100,10,0,0,14,65,100,100,114,101,705
2610 DATA 115,115,32,101,114,114,111,114,10,0,48,8
74
2620 DATA 66,121,101,32,102,114,111,109,32,77,111,
976
2630 DATA 110,105,116,111,114,32,10,32,67,111,110,
918
2640 DATA 116,114,111,108,32,39,67,39,32,102,111,8
71
2650 DATA 114,32,97,110,111,116,104,101,114,32,106
,1037
2660 DATA 111,98,10,10,0,3,32,32,32,0,0,328
2670 DATA 8,0,0,0,0,0,0,0,0,0,39,47
2680 DATA 32,68,110,32,0,0,0,0,0,0,242
2690 DATA 0,32,65,110,32,0,0,0,0,0,239
2700 DATA 0,0,32,40,65,110,41,32,0,0,320
2710 DATA 0,0,0,0,0,10,0,0,79,10,32,131
2720 DATA 32,32,32,32,32,32,32,80,67,32,0,403
2730 DATA 0,0,0,0,0,0,0,32,40,80,67,219
2740 DATA 41,32,0,0,0,0,0,0,0,0,10,83
2750 DATA 10,32,32,32,32,84,32,83,32,32,73,474
2760 DATA 73,73,32,32,32,88,78,90,86,67,32,683
2770 DATA 10,32,83,82,32,0,0,0,0,0,0,239
2780 DATA 0,0,0,0,0,0,0,0,0,0,10,10
2790 DATA 0,0,0,0,0,0,0,0,0,0,0,0
```

Composer II

James Lucy

This is the final part of our composer program. It applies something akin to the Midas Touch to the QL's somewhat limited sound making capability converting discord into harmony. Using this program you can transfer or create your own musical scores, play them back and even amend them, deleting or inserting notes as you go. The program will handle sharps, vary tempo and even specify staccato and legato playing styles.

```
4390 DEFine PROCedure status
4400 CLS#3
4410 AT #3,1,2:PRINT#3,"PAGE"!current_page!!!"NOTE
S"!notenum-1!!!"METRONOME ";metro_mark
4420 END DEFine
4430 DEFine PROCedure sublines
4440 IF p=33 OR p=31 :LINE across,up:LINE_R TO 4,0
 TO -8,0
4450 IF p=36:LINE across,up+1.5:LINE_R TO 4,0 TO -
8,0
4460 IF p=41OR p=38:LINE across,up:LINE_R TO 4,0 T
O -8,0:LINE_R 0,3 TO 8,0
4470 END DEFine
4480 DEFine PROCedure delete_note
4490 INK 4:dn=counter-1
4500 en=(current_page-1)*100+100:IF current_page=p
age:en=en-100+pagenotecount
4510 FOR del=note_number TO en
4520 dn=dn+1
4530 display pitch(del),duration(del),dn
4540 END FOR del
4550 FOR shift=note_number TO notenum
4560 pitch(shift)=pitch(shift+1)
4570 duration(shift)=duration(shift+1)
4580 END FOR shift
4590 notenum=notenum-1
4600 pagenotecount=pagenotecount-1
4610 INK 0:dn=counter-1
4620 FOR redraw=note_number TO en
4630 dn=dn+1
4640  display pitch(redraw),duration(redraw),dn
4650 END FOR redraw
4660 END DEFine
4670 DEFine PROCedure insert_note
4680 INK 4:ino=counter-1
4690 en=(current_page-1)*100+100:IF current_page=p
age:en=en-100+pagenotecount
4700 FOR ins=note_number TO en
4710 ino=ino+1
4720 display pitch(ins),duration(ins),ino
4730 END FOR ins
4740 REPeat check_inserted_note
4750 er=0
4760 input_pitch:check_pitch:IF er:END REPeat chec
k_inserted_note
4770 convert_pitch
4780 input_duration:check_duration:IF er:END REPea
t check_inserted_note
4790 INK 0:display p,d,counter
4800 play p,d
4810 FOR shove_along=notenum+1 TO note_number+1 ST
EP -1
4820 pitch(shove_along)=pitch(shove_along-1)
4830 duration(shove_along)=duration(shove_along-1)
4840 END FOR shove_along
4850 pitch(note_number)=p:duration(note_number)=d
4860 dn=counter-1
4870 FOR redraw=note_number TO en
4880 dn=dn+1
4890 display pitch(redraw),duration(redraw),dn
4900 END FOR redraw
4910 notenum=notenum+1:pagenotecount=pagenotecount
+1
4920 END DEFine
4930 DEFine PROCedure store_music
4940 CLS#0:PRINT#0,"  STORE:  Press 'c' to continu
e : ":y$=INKEY$(-1)
4950 IF y$<>"c" AND y$<>"C" THEN RETurn
4960 REPeat check_saved_music
4970 CLS#0:PRINT#0,"  Enter file name for the mus
ic : (e.g mdv1_opus1)"
4980 PRINT#0,"  Device must exist and must not co
ntain the filename you use !";
4990 INPUT #0,stored_music$
5000 n$=stored_music$&"     ":n$=n$(1 TO 5)
5010 IF n$(1 TO 5)=="mdv1_" OR n$(1 TO 5)=="mdv2_"
:ELSE END REPeat check_saved_music
5020 OPEN_NEW #9,stored_music$
5030 FOR a=0 TO 950:PRINT#9,pitch(a)
5040 FOR a=0 TO 950:PRINT#9,duration(a)
5050 PRINT #9,notenum:PRINT#9,pagenotecount:PRINT#
9,page:PRINT#9,metro_mark
5060 CLOSE #9
5070 REMark
5080 REMark
5090 END DEFine
5100 DEFine PROCedure load_music
5110 CLS#0:PRINT#0,"  LOAD:  Press 'c' to continu
e: ":y$=INKEY$(-1)
5120 IF y$<>"c" AND y$<>"C" THEN RETurn
5130 REPeat check_stored_music
5140 CLS#0:PRINT#0,"  Enter file name of stored m
usic : (e.g. mdv1_opus1)";
5150 INPUT #0,stored_music$
5160 n$=stored_music$&"     ":n$=n$(1 TO 5)
5170 IF n$(1 TO 5)=="mdv1_" OR n$(1 TO 5)=="mdv2_"
 :ELSE END REPeat check_stored_music
5180 CLS :stave
5190 OPEN_IN#9,stored_music$
5200 FOR a=0 TO 950:INPUT#9,pitch(a)
5210 FOR a=0 TO 950:INPUT#9,duration(a)
5220 INPUT#9,notenum:INPUT#9,pagenotecount:INPUT#9
,page:INPUT#9,metro_mark
5230 CLOSE#9
5240 current_page=1
5250 lim=100:IF notenum<100 THEN lim=notenum+1
5260 FOR a=1 TO lim:display pitch(a),duration(a),a
5270 status
5280 editor
5290 END DEFine
5300 DEFine PROCedure sounds
5310 CLS#0:PRINT#0 ,"  SOUND CHANGE : Press 'c' to
 continue"
5320 y$=INKEY$(-1):IF y$<>"C" AND y$<>"c" THEN RET
urn
5330 CLS#0:PRINT#0,"  Press 'm' to return to music
al notes, any other key to change tone"
5340 y$=INKEY$(-1):IF y$=="m" :dur=-1:pitch_2=0:gr
ad_x=0:grad_y=0:wraps=0:fuzzy=0:random=0:RETurn
5350 CLS#0:INPUT#0,"  Duration:  (-32768 to 32767)
 ";dur
5360 INPUT#0,"  Pitch_2 :  (0 to 255) ";pitch_2
5370 INPUT#0,"  Grad_x  :  (-32768 to 15) ";grad_x
5380 INPUT#0,"  Grad_y  :  (-8 to 7)  ";grad_y
5390 INPUT#0,"  Wraps   :  (0 to 32767)  ";wraps
5400 INPUT#0,"  Fuzzy   :  (0 to 15)  ";fuzzy
5410 INPUT#0,"  Random  :  (0 to 15)  ";random
5420 END DEFine sounds
5430 DEFine PROCedure draw_sharp
5440 CURSOR across-4,up+2,0,0
5450 PRINT "#"
5460 END DEFine draw_sharp
5470 DEFine PROCedure draw_legato
5480 IF p<13
5490 LINE across,up-change+13
5500 ARC_R TO 8,0,-PI/2
5510 ELSE
5520 LINE across,up+9
5530 ARC_R TO 8,0,-PI/2
5540 END IF
5550 END DEFine
5560 DEFine PROCedure draw_staccato
5570 FILL 1
5580 IF p<12
5590 CIRCLE across,up+3,1
5600 ELSE
5610 CIRCLE across,up-3,1
5620 END IF
5630 FILL 0
5640 END DEFine
5650 DEFine PROCedure help(intro)
5660 PAPER 4,2,0:CLS: INK 0:CSIZE 3,1:STRIP 1:PAPE
R 4
5670 AT 0,11:UNDER 1:PRINT"HELP":UNDER 0
5680 CSIZE 0,0:PRINT "You reached this screen by t
yping 'HELP' in answer to the PITCH ? prompt."
5690 PRINT\"The pitch of the note entered must be
 from  A to  G, then a to g.  Sharps are entered b
y adding an 'S' or 's'.  Rests are entered using '
R' or 'r'.  The treble clef applies to all stave l
ines."
5700 PRINT\"You may also type'EDIT, DELETE, SAVE,
LOAD, PLAY, TIMBRE or HELP in answer to the PITCH
? prompt."
5710 PRINT\"EDIT - Enters  the  editor.  You  may
```

```
 select a page and can then delete, insert, or cha
nge notes. Pressing 'c'ontinue redraws the last pa
ge of your music.  'P'lay allows any part of the m
usic to be played."
5720 PRINT\"DELETE - Deletes last note drawn.  For
 immediately noticed mistakes."
5730 PRINT\"SAVE - Allows the music to be saved to
 microdrive.  Must be drive 1 or 2 , filename must
 be valid and not yet exist."
5740 INK 7:AT 19,20:PRINT "PRESS ANY KEY TO CONTIN
UE":PAUSE:INK 0
5750 PAPER 4,2,0:CLS:STRIP 1:PAPER 4
5760 PRINT\"LOAD - Deletes any music on the screen
 and loads music from microdrive."
5770 PRINT \"PLAY - Plays whole piece from beginni
ng."
5780 PRINT\"TIMBRE - Allows sound parameters to be
 reset.  Sound effects may then be  played like mu
sic.  Press 'm' to return to music parameters."
5790 PRINT\"HELP - Prints these pages."
5800 PRINT \"The DURATION ? prompt takes notes in
numerical form, assuming a crotchet = 1.  The allo
wed values are .5, .75, 1, 1.5, 2, 3, 4 i.e from s
emiquaver  to semibreve.  A staccato note may be s
pecified by adding an 's' after the number and a l
egato note an 'l' e.g  2l."
5810 PRINT\"Both PITCH and DURATION inputs 'auto r
epeat'- they default to the previous value if 'ENT
ER' is pressed."
5820 PRINT\"No more information."
5830 AT 19,22:INK 7:PRINT"PRESS ANY KEY TO RETURN"
5840 PAUSE:INK 0
5850 IF intro THEN RETurn
5860 PAPER 4:CLS:stave
5870 star=(current_page-1)*100+1:spot=star+99
5880 IF current_page=page THEN spot=star+pagenotec
ount
5890 pp=0:FOR rp=star TO spot:pp=pp+1:display pitc
h(rp),duration(rp),pp
5900 REMark
5910 REMark
5920 REMark
5930 END DEFine help
5940 DEFine PROCedure welcome
5950 MODE 4:WINDOW 512,256,0,0:PAPER 4,2,0:CLS
5960 STRIP 1:PAPER 7:INK 0:CSIZE 2,1:AT 3,8:PRINT
"  Q L  C O M P O S E R  "
5970 AT 6,18:PRINT"  by  "
5980 AT 8,15:PRINT " James Lucy "
5990 AT 10,18:CSIZE 1,0:PRINT " (C) 1985 "
6000 PAUSE 100
6010 PAPER 4,2,0:CLS:PAPER 7:CSIZE 2,0
6020 AT 5,6:PRINT" Press 'i' for information     "
6030 AT 8,6:PRINT" Press 'd' for a demonstration "
6040 AT 11,6:PRINT " Press 's' to start the progra
m"
6050 STRIP 0
6060 REPeat check_choice
6070 cho$=INKEY$(-1)
6080 IF cho$ INSTR "iIsSdD"=0:END REPeat check_cho
ice
6090 IF cho$=="S" :RETurn
6100 IF cho$=="D" :demo
6110 IF cho$=="I"  THEN
6120 WINDOW 448,220,32,16
6130 PAPER 2,7,3:CLS
6140 PRINT\ "  QL COMPOSER accepts musical notes
typed in by the  user, displays them on a stave an
d plays them.  "
6150 PRINT\"  Notes can be inserted, deleted  or c
hanged  and  can be played staccato,legato, or  no
rmally.  The  speed of  playing can be varied; the
  program accepts the standard metronome mark. "
6160 PRINT \"  The program starts by requesting a
metronome mark.  Press enter for the default value
.  A pitch will then be requested, which can be us
ed to enter the  first  note  or  to  enter the co
mmands listed on the help screen."
6170 AT 19,8:STRIP 1:PAPER 7:CSIZE 1,0:PRINT"Press
 any key to view help screen.":PAUSE:STRIP 0
6180 help 1
6190 PAPER 2,7,3:CLS:STRIP 1:PAPER 7:INK 0
6200 AT 10,0:CSIZE 1,0:PRINT" Press 'd' for a demo
nstration, 's' to start the program":STRIP 0
6210 REPeat check_ch
6220 cho$=INKEY$(-1)
6230 IF cho$ INSTR "dsDS"=0:END REPeat check_ch
6240 IF cho$=="s":RETurn
6250 demo
6260 END IF
6270 END DEFine welcome
6280 DEFine PROCedure demo
6290 PAPER 0:CLS:WINDOW 448,200,32,16:PAPER 4:CLS
6300 DIM demop(35),demod(35)
6310 stave:RESTORE :draw_clef 8,85
6320 STRIP 1:PAPER 7:CSIZE 0,0:AT 10,14:PRINT" Ext
ract from Beethoven's Ninth Symphony ":STRIP 0:PAP
ER 4
6330 FOR red=3 TO 32:READ demop(red),demod(red):di
splay demop(red),demod(red),red
6340 FOR red=3 TO 32:play demop(red),demod(red)
6350 DATA 20,1,20,1,19,1,15,1,15,1,19,1,20,1,24,1,
28,1,28,1,24,1,20,1,20,1.5
6360 DATA 24,.5,24,2,20,1,20,1,19,1,15,1,15,1,19,1
,20,1,24,1,28,1,28,1,24,1
6370 DATA 20,1,24,1.5,28,.5,28,2
6380 REMark
6390 REMark
6400 END DEFine demo
6410 DEFine PROCedure pre_initialise
6420 DIM pitch(950):DIM duration(950)
6430 notenum=0:pagenotecount=0:page=1:current_page
=1:metro_mark=160
6440 dur=-1:pitch_2=0:grad_x=0:grad_y=0:wraps=0:fu
zzy=0:random=0
6450 END DEFine pre_initialise
6460 DEFine PROCedure draw_clef(x,y)
6470 LINE x,y+1.5
6480 ARC_R TO 0,4.5,-PI TO 0,-6,-PI TO -3,7,-3*PI/
4
6490 LINE_R TO 5,7:ARC_R TO -2,0,PI
6500 LINE_R TO 0,-18
6510 FILL 1:CIRCLE_R -1,0,1:FILL 0
6520 END DEFine draw_clef
```

Variable Dump

I Robinson

The following utility is invaluable when it comes to debugging BASIC programs. It adds the extra command VAR to SuperBasic. This will list all variables used within a program and their contents to the screen. A hex loader listed below with checksums at the end of each line (to minimise typing-in errors) gives the machine code extension. After then a short test program follows which load the routine and tests it.

```
100 REMark ** QL User 1985 by I Robinson **
110 REMark ** SuperBasic Extension     **
120 REMark ** Variable displayer       **
130 DIM array(5):address=RESPR(598)
140 start=address:CLS:CLS#0:RESTORE
150 READ nwords
160 IF nwords<=0
170   PRINT"Load complete"
180   FOR a=0 TO 49:POKE_L start+380+a*4,0
190   SBYTES mdv1_varext_bin,start,598
200   STOP
210 END IF
220 csum=0:FOR i=0 TO nwords-1:READ array(i):csum=
csum+array(i):NEXT i
230 READ csum2:IF csum<>csum2 THEN PRINT"Checksum
error":PRINT"Correct checksum",csum:PRINT"Line of
data":FOR i=0 TO nwords-1:PRINT,array(i);",";:NEXT
 i:STOP
240 FOR i=0 TO nwords-1:POKE_W address,array(i):ad
dress=address+2:NEXT i
250 GO TO 150
260 DATA 6,17402,12,13432,272,20114,28672,79904
270 DATA 6,20085,1,12,854,16722,0,37674
280 DATA 6,0,0,17402,338,14456,200,32396
290 DATA 6,20116,17402,340,14456,208,20116,72638
300 DATA 6,10862,24,3126,3,-10240,27168,30943
310 DATA 6,3126,1,-10240,26392,3126,3,22408
320 DATA 6,-10239,26406,3126,2,-10239,26468,35524
330 DATA 6,3126,1,-10239,26426,-9220,0,10094
340 DATA 6,8,9326,28,-18995,27338,28674,46379
350 DATA 6,20034,28672,20085,24832,146,3126,96895
360 DATA 6,0,-10240,26368,228,9846,-10236,15966
370 DATA 6,9326,40,-10805,12854,-22528,14456,3343
380 DATA 6,206,20116,24774,24942,3126,0,73164
390 DATA 6,-10240,26368,194,9846,-10236,8814,24746
400 DATA 6,40,-11317,-11314,14456,208,20116,12189
410 DATA 6,24576,-90,24908,3126,0,-10240,42280
420 DATA 6,26368,160,9846,-10236,9326,40,35504
430 DATA 6,-10805,-10802,17402,382,8922,13018,1811
7
440 DATA 6,17402,374,-27698,12040,16890,172,19180
450 DATA 6,-28210,14456,240,20116,17402,160,24164
460 DATA 6,-27698,-28215,13320,8287,30463,17402,13
559
470 DATA 6,146,28679,20035,24576,-168,28690,101958
480 DATA 6,30463,20035,3200,0,0,26402,80100
490 DATA 6,12040,8316,1,1,28673,30463,79494
500 DATA 6,20035,8287,28688,30463,29184,29696,1463
53
510 DATA 6,20035,28706,30463,20035,24784,17025,141
048
520 DATA 6,12854,-10238,8814,32,-11327,16962,17097
530 DATA 6,5174,-26624,13884,-1,-11314,-11268,-301
49
540 DATA 6,0,1,28679,20035,28689,30463,107867
550 DATA 6,29199,20035,20085,17402,38,14456,101215
560 DATA 6,208,20116,24576,-274,1025,7,45658
570 DATA 6,300,100,32,16,12,22081,22541
580 DATA 6,21065,16706,19525,21280,14858,1,93435
590 DATA 6,10784,0,0,0,0,0,10784
600 DATA -1
```

```
5 REMark ** Test program **
10 a=RESPR(598)
20 LBYTES mdv1_varext_bin,a
30 CALL a
40 VAR
```

Pentathlete

A Didcock

Anyone who's played something like *Daley Thompson's Decathlon* will recognise this as being of similar vein. To keep you guessing, however, we won't expand on the way the various events are played – though there are the usual mix of running, jumping and throwing challenges.

```
1 REMark track_boot
3 MODE 8
4 LBYTES mdv1_title_screen,131072
5 MERGE mdv1_UDG_data
6 GO TO 32000
7 MRUN mdv1_TRACK_and_FIELD
10 hiscore=0
20 f_level
40 draw_main_screen
50 draw_power
60 initialise
69 :
90 REPeat game
100  event_1: screen
110  IF NOT qualify THEN end_routine: GO TO 20
120  event_2: screen
130  IF NOT qualify THEN end_routine: GO TO 20
140  event_3: screen
150  IF NOT qualify THEN end_routine: GO TO 20
160  event_4: screen
170  IF NOT qualify THEN end_routine: GO TO 20
180  event_5: screen
190  IF NOT qualify THEN end_routine: GO TO 20
200  next_level
210  IF NOT qualify THEN end_routine: GO TO 20
215  level=level+.5
220 END REPeat game
699 :
700 DEFine PROCedure hi_scores
705 LOCal i
707 hi(event,0)=distance
708 hi$(event,0)=a$
710 IF hi(event,0)>hi(event,1) THEN
720  wr=wr+1
730  FOR i=3 TO 1  STEP -1
740   hi(event,i)=hi(event,i-1)
750   hi$(event,i)=hi$(event,i-1)
760  END FOR i
762  RETurn
764 END IF
766 IF hi(event,0)>hi(event,2) THEN
770  hi(event,3)=hi(event,2):hi$(event,3)=hi$(event,2)
772  hi(event,2)=hi(event,0):hi$(event,2)=hi$(event,0)
774  wr=wr+1
776  RETurn
778 END IF
780 IF hi(event,0)>hi(event,3) THEN
782  wr=wr+1
784  hi(event,3)=hi(event,0):hi$(event,3)=hi$(event,0)
786  RETurn
788 END IF
790 END DEFine
799 :
800 DEFine PROCedure reset
810 STRIP #3,7,0
820 AT #3,1,8:PRINT #3,"                    "
830 INK #3,2: STRIP #3,7
840 AT #3,1,36:PRINT #3,"00 "
850 END DEFine
899 :
900 DEFine PROCedure time (type)
910 SELect ON type
920 ON type=0: ft=DATE: RETurn
930 ON type=1: ut=DATE-ft:ut$=DATE$(ut)
940 END SELect
942 CSIZE 2,0
945 INK 5:STRIP 2:AT 8,35:PRINT ut$(19 TO);".00"
950 END DEFine
999 :
1000 DEFine PROCedure event_1
1005 event=1:CLS #7:draw_shot:reset
1006 pos=2: qual=0
1007 INK #4,7:FOR i=2 TO 4:AT #4,i,14:PRINT #4,"00.00":END FOR i
1009 REPeat shot
1010 PAPER 0:INK 5
1011 foul=0
1015 CSIZE 2,1
1020 x=8: y=3:power=8
1025 AT x,y:PRINT CHR$(128)
1030 INK 0,7:FILL 1:CIRCLE 12,32,.75:FILL 0
1040 CSIZE 2,0:INK 2:PAPER 4:AT 10,3:PRINT "YOUR GO:"
1045 FOR i=100 TO 10 STEP -1:BEEP 300,i:END FOR i
1052 BEEP 1000,5
1055 AT 10,3:PRINT "        ": AT #3,1,power:STRIP #3,0:PRINT #3," "
1056 n=0
1060 FOR shot_loop=1 TO 60
1070  key1=KEYROW(1)
1072  IF key1<>0 THEN BEEP 200,100
1073  IF key1>=64 AND key1<100 THEN EXIT shot_loop
1075  IF key1<>0 AND key1<>key2 AND power<23.5 THEN n=n+1
1076  IF n=INT(level) THEN power=power+1:n=0
1080  IF power>8 AND (key1=0 OR key1=key2) THEN power=power-.2
1090  AT #3,1,power:STRIP #3,0:PRINT #3," ":STRIP #3,7,0:AT #3,1,power+1:PRINT #3," "
1095  FOR i=1 TO 120:END FOR i
1100  key2=key1
1120 END FOR shot_loop
1130 IF shot_loop=60 THEN foul=1
1140 IF NOT foul THEN throw: GO TO 1200
1150 INK 2:STRIP 1:FLASH 1
1160 AT 14,2:PRINT "FOUL":BEEP 0,30,37,9000,7,0,0,0
1170 FLASH 0
1175 AT #4,pos,14:PRINT #4,"-----"
1180 FOR i=1 TO 1000: END FOR i: BEEP
1195 distance=0
1200 pos=pos+1
1210 score=score+INT(distance*10)
1215 INK #4,2
1230 IF score>9 AND score<100 THEN AT #4,0,10:PRINT #4,score
1240 IF score>99 AND score<1000 THEN AT #4,0,9:PRINT #4,score
1245 IF score>999 AND score<10000 THEN AT #4,0,8:PRINT #4,score
1246 IF score>9999 THEN AT #4,0,7:PRINT #4,score
1250 IF NOT foul THEN INK 2:FILL 1:CIRCLE x,y,.8:FILL 0
1270 draw_shot
1280 reset
1282 INK 4
1283 FOR i=1 TO 1500:END FOR i
1285 AT 11,1:PRINT "                    "
1286 AT 13,1:PRINT "                "
1288 STRIP 4:AT 14,2:PRINT "    "
1290 hi_scores
1291 print_hi_scores
1292 power=8
1293 IF pos=5 THEN EXIT shot
1295 END REPeat shot
1296 qualify=qual
1297 END DEFine
1299 :
1300 DEFine PROCedure throw
1305 CSIZE 2,1:STRIP 0:INK 5
1310 AT x,y:PRINT " ":AT x,y+1:PRINT CHR$(129)
1320 BEEP 1000,5
1325 angle=10: STRIP #3,7: INK #3,2
1326 AT #3,1,36:PRINT #3,angle
1330 REPeat key
1340 IF KEYROW(1)=0 OR angle>80 THEN EXIT key
1350 angle=angle+2
1360 AT #3,1,36:PRINT #3,angle
1370 END REPeat key
1375 x=22: y=37: INK 0,7
1376 FILL 1:CIRCLE x,y,1: FILL 0
1377 IF angle<50 THEN a=(angle/10.5)
1378 IF angle>49 THEN a=(1/angle)*200
1379 b=angle/40
1380 n=0
1384 i=0
1385 REPeat launch
1386 i=i+1
1390 x1=x: y1=y
1400 IF i>power*.7 AND NOT n THEN a=a-(a/3): b=-(b*2): n=1
1410 x=x+a: y=y+b
1420 FILL 1:INK 4:CIRCLE x1,y1,1:FILL 0
1430 FILL 1:INK 0,7:CIRCLE x,y,1:FILL 0
1440 IF y<=30 THEN EXIT launch
1450 END REPeat launch
1460 draw_shot
1470 FILL 1:INK 0,7:CIRCLE x,y,1:FILL 0
1475 di=11
1480 FOR i=26 TO x
1490  di=di+1
1510 END FOR i
1520 distance=((di/35)*7)+(INT(RND(0 TO 20))/100)
1530 CSIZE 2,0:INK 2
1540 AT 11,1:PRINT "YOUR THROW WAS ";distance;"m"
1550 IF distance>=15.28 AND NOT qual THEN INK 0: AT 13,1:PRINT "YOU HAVE QUALIFIED":qual=1
1560 IF distance>=10 THEN AT #4,pos,14:PRINT #4,distance: ELSE AT #4,pos,15:PRINT #4,distance
1570 END DEFine
1999 :
2000 DEFine PROCedure event_2
2005 event=2
2010 pos=2:level2=1:lim=.2
2015 CLS #7
2020 INK #4,7:FOR i=2 TO 4:AT #4,i,14:PRINT #4,"00.00":END FOR i
2021 IF INT(level)=3 THEN level2=2
2022 IF INT(level)=2 THEN lim=1
2025 qual=0
2027 REPeat sprint_loop
2030 draw_sprint
2035 time (0):time (1):INK 0:AT 8,30:PRINT "TIME:"
2040 foul=0: tot=1: gm=1: fin=290: power=8
2045 reset
2050 CSIZE 2,1:INK 5:STRIP 2:AT 8,14:PRINT CHR$(130)
2055 co=14
2060 n=0: key1=0: key2=0: char=130: total=380
2065 CSIZE 2,0:INK 2:PAPER 4:AT 10,3:PRINT "GET READY . . ."
2070 le=RND(900 TO 1400)
2075 FOR i=1 TO le:END FOR i
2080 BEEP 1000,1:AT 12,3:PRINT "GO!"
2083 time (0)
2085 PAUSE 40000
2086 AT #3,1,power:STRIP #3,0:PRINT #3," "
2090 AT 10,3:PRINT "                ":AT 12,3:PRINT "    "
2100 REPeat sprint_1
2105  IF char=131 THEN time (1)
2110  CSIZE 2,1:STRIP 2:INK 5
2120  key1=KEYROW(1)
2125  BEEP 300,100
2130  IF key1<>0 AND key1<>key2 AND power<23.5 THEN n=n+1
2135  IF n=level2 THEN power=power+1:n=0
2140  IF power>8 AND (key1=0 OR key1=key2) THEN power=power-lim
2145  AT #3,1,power:STRIP #3,0:PRINT #3," ":STRIP #3,7,0:AT #3,1,power+1:PRINT #3," "
2150  key2=key1
2155  total=total-power
2160  IF char=130 THEN char=131: ELSE char=130
```

THE PROGS

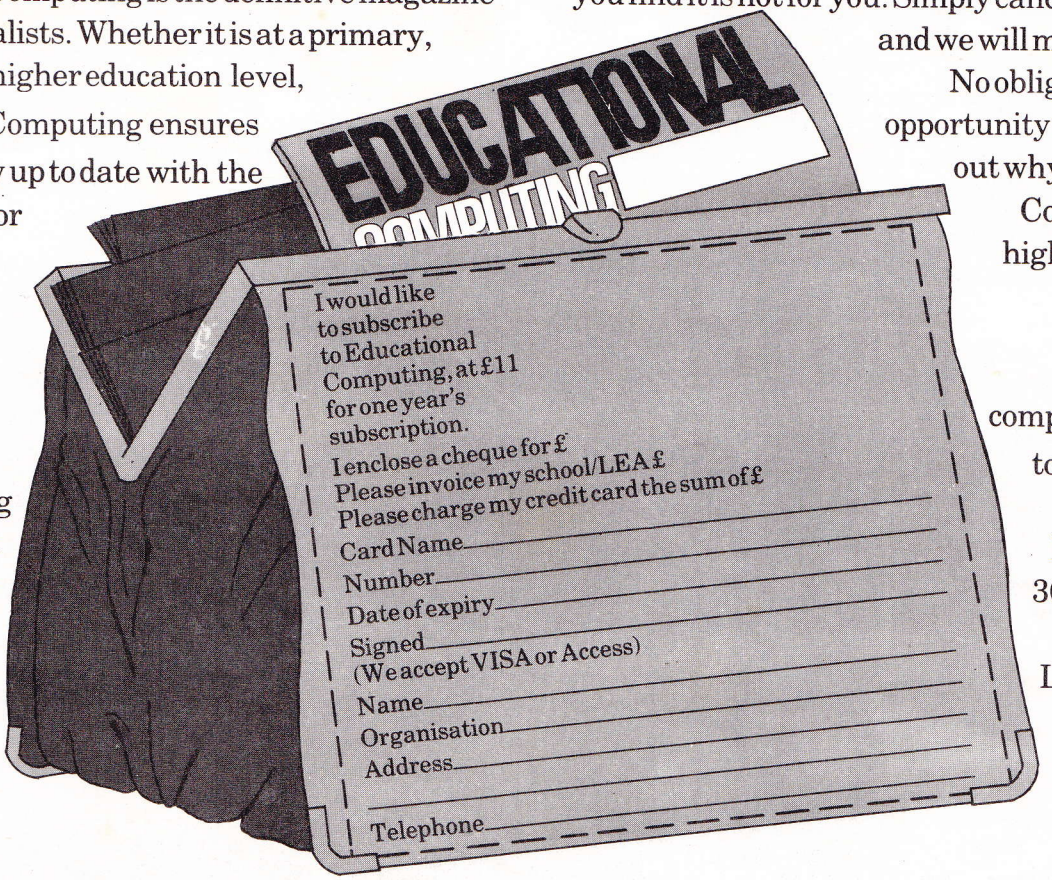
```
2162  IF co>=30 THEN co=0:INK #6,7:POINT #6,732,0: POINT #6,732,100
2165  IF NOT gm THEN AT 8,14:PRINT " ":PAN #6,-power:AT 8,14:PRINT CHR$(char)
2170  IF gm THEN AT 8,14:PRINT " ":PAN #6,-25:AT 8,14:PRINT CHR$(char)
2175  co=co+power
2180  IF total<100 AND tot THEN LINE #6,730,100 TO 680,0: tot=0
2182  IF total<100 THEN fin=fin-power
2183  IF fin<=0 THEN EXIT sprint_1
2185  gm=0
2190 END REPeat sprint_1
2195 tme=ut$(19 TO )+(INT(RND(1 TO 98))/100)
2200 CSIZE 2,0:STRIP 2:INK 5
2205 IF tme<10 THEN AT 8,35:PRINT "0";tme: ELSE AT 8,35:PRINT tme
2210 BEEP 1000,1
2215 IF tme<10 THEN AT #4,pos,14:PRINT #4,"0";tme: ELSE AT #4,pos,14:PRINT #4,tme
2220 pos=pos+1
2225 IF NOT qual AND tme<12.65 THEN qual=1:INK 0:STRIP 4:AT 13,1:PRINT "YOU HAVE QUALIFIED"
2230 score=score+INT((30-tme)*15)
2232 INK #4,2
2235 IF score>9 AND score<100 THEN AT #4,0,10:PRINT #4,score
2240 IF score>99 AND score<1000THEN AT #4,0,9:PRINT #4,score
2245 IF score>999 AND score<10000 THEN AT #4,0,8:PRINT #4,score
2246 IF score>9999 THEN AT #4,0,7:PRINT #4,score
2250 hi_scores_2
2255 print_hi_scores
2260 FOR i=1 TO 1500: END FOR i
2265 AT 13,1:STRIP 4:PRINT "                    "
2270 IF pos=5 THEN EXIT sprint_loop
2275 END REPeat sprint_loop
2280 qualify=qual
2285 END DEFine
2999 :
3000 DEFine PROCedure event_3
3005 event=3:pos=2:CLS #7
3010 INK #4,7:FOR i=2 TO 4:AT #4,i,14:PRINT #4,"00.00":END FOR i
3015 qual=0
3020 REPeat jav_loop
3025 draw_jav
3030 foul=0:power=8:reset:dist=0:me=0
3035 key1=0:key2=0:char=132:n=0:co=0
3040 CSIZE 2,1:STRIP 2:INK 5:AT 9,8:PRINT CHR$(132)
3045 INK 7:LINE 25,26 TO 35,26
3050 CSIZE 2,0:INK 2:PAPER 4:AT 10,3:PRINT "YOUR GO:"
3055 FOR i=100 TO 10 STEP -1:BEEP 300,i:END FOR i
3056 PAUSE 40000
3060 BEEP 1000,5
3065 AT 10,3:PRINT "         ":AT #3,1,power:STRIP #3,0:PRINT #3," "
3067 CSIZE 2,1
3070 FOR i=1 TO 34
3075  key1=KEYROW(1)
3080  BEEP 300,100
3083  IF key1>=64 AND key1<100 THEN EXIT i
3085  IF key1<>0 AND key1<>key2 AND power<23.5 THEN n=n+1
3090  IF n=INT(level) THEN power=power+1:n=0
3095  IF power>8 AND (key1=0 OR key1=key2) THEN power=power-.2
3100  AT #3,1,power:STRIP #3,0:PRINT #3," ":STRIP #3,7,0:AT #3,1,power+1:PRINT #3," "
3105  key2=key1
3110  IF char=133 THEN char=132: ELSE char=133
3115  INK 2:STRIP 2
3120  LINE 25,26 TO 35,26:AT 9,8:PRINT " ":PAN #8,-10:INK 5:AT 9,8:PRINT CHR$(char):INK 7:LINE 25,26 TO 35,26
3124  co=co+1
3125  IF co=10 THEN INK 2:CSIZE 2,0:STRIP 1:AT 19,37:PRINT me+10
3126  IF co=10 THEN LINE #8,1451,100 TO 1371,0:co=0:me=me+10
3127  CSIZE 2,1:INK 5:STRIP 2
3180 END FOR i
3185 IF i=34 THEN foul=1
3186 dist=-((33-i)):power=power*1.14
3190 IF NOT foul THEN throw_jav:GO TO 3200
3191 CSIZE 2,0
3192 distance=0
3193 STRIP 1:INK 2:FLASH 1:AT 14,2:PRINT "FOUL":BEEP 0,30,37,9000,7,0,0,0:FLASH 0
3194 FOR i=1 TO 1000:END FOR i:BEEP
3195 INK #4,7:AT #4,pos,14:PRINT #4,"-----"
3200 pos=pos+1
3205 INK #4,2
3210 IF score>9 AND score<100 THEN AT #4,0,10:PRINT #4,score
3215 IF score>99 AND score<1000 THEN AT #4,0,9:PRINT #4,score
3220 IF score>999 AND score<10000 THEN AT #4,0,8:PRINT #4,score
3221 IF score>9999 THEN AT #4,0,7:PRINT #4,score
3225 hi_scores
3230 print_hi_scores
3235 FOR i=1 TO 1500:END FOR i
3236 STRIP 4
3240 AT 11,1:PRINT "                    ":AT 13,1:PRINT "                    "
3241 AT 14,2:PRINT "     "
3245 IF pos=5 THEN EXIT jav_loop
3250 END REPeat jav_loop
3255 qualify=qual
3260 END DEFine
3299 :
3300 DEFine PROCedure throw_jav
3305 BEEP 1000,5
3310 angle=10:STRIP #3,7:INK #3,2
3315 AT #3,1,36:PRINT #3,angle
3320 REPeat key_jav
3325  IF KEYROW(1)=0 OR angle>80 THEN EXIT key_jav
3330  angle=angle+3
3335  AT #3,1,36:PRINT #3,angle
3340 END REPeat key_jav
3342 BEEP 1000,5
3345 x=25:y=28
3350 h=angle/60: a=angle/12:b=SQRT(100-(a*a)):hh=1:a1=a:ha=0
3355 INK 2:LINE 25,26 TO 35,26:INK 7:LINE x,y TO x+b,y+a
3356 IF angle>50 THEN power=power-(angle/10)
3357 i=1
3358 IF angle<30 THEN power=power-1
3360 REPeat mve_jav
3362 y1=y
3370 PAN #8,-10
3373 co=co+1: dist=dist+1
3374 IF co=10 AND me<90 THEN INK 2:CSIZE 2,0:STRIP 1:AT 19,37:PRINT me+10
3375 IF co=10 THEN LINE #8,1451,100 TO 1371,0:co=0:me=me+10
3380 IF (y+h)+a<64 AND NOT ha THEN y=y+h
3384 IF y+a<28 THEN EXIT mve_jav
3385 INK 4:LINE x,y1 TO x+b,y1+a1:INK 7:LINE x,y TO x+b,y+a
3386 a1=a
3390 IF i>(power*3)*.7 AND hh THEN hh=0:h=-h:a=-a:ha=0
3396 IF i>(power*3)*.5 AND hh THEN ha=1
3397 i=i+1
3399 END REPeat mve_jav
3402 PAN #8,-29
3405 INK 4:LINE x,y1 TO x+b,y1+a1:INK 7:LINE x,y+a TO x+b,y+(a+a)
3410 CSIZE 2,1:INK 2:STRIP 4
3412 distance=dist+(INT(RND(1 TO 98))/100)
3413 CSIZE 2,0
3414 IF distance<1 THEN distance=0:dist=distance
3415 AT 11,1:PRINT "YOU THROW WAS ";distance;"m"
3420 IF distance>70.23 AND NOT qual THEN INK 0:AT 13,1:PRINT "YOU HAVE QUALIFIED":qual=1
3425 score=score+(dist*3)
3430 IF distance<10 THEN INK #4,7:AT #4,pos,14:PRINT #4,"0";distance: ELSE INK #4,7:AT #4,pos,14:PRINT #4,distance
3440 END DEFine
3999 :
4000 DEFine PROCedure event_4
4005 event=4:pos=2:CLS #7:qual=0
4010 INK #4,7:FOR i=2 TO 4:AT #4,i,14:PRINT #4,"00.00":END FOR i
4015 REPeat lo_sprint_loop
4020 draw_lsprint
4025 time (0):time (1):INK 0:AT 8,30:PRINT "TIME:"
4030 foul=0:tot=1:gm=1:fin=290:power=8:limit=23.5
4035 reset
4040 CSIZE 2,1:INK 5:STRIP 2:AT 8,14:PRINT CHR$(130)
4045 co=14:n=0:key1=0:key2=0:char=130:total=740:pow_to=0
4050 CSIZE 2,0:INK 2:PAPER 4:AT 10,3:PRINT "GET READY . . ."
4055 le=RND(1000 TO 1600)
4060 FOR i=1 TO le:END FOR i
4065 BEEP 1000,1:AT 12,3:PRINT "GO!"
4070 time (0)
4075 PAUSE 40000
4080 AT #3,1,power:STRIP #3,0:PRINT #3," "
4085 AT 10,3:PRINT "                ":AT 12,3:PRINT "    "
4090 REPeat lo_sprint_1
4095  IF char=130 THEN time (1)
4100  CSIZE 2,1:STRIP 2:INK 5
4105  key1=KEYROW(1)
4110  BEEP 300,100
4115  IF key1<>0 AND key1<>key2 AND power<limit THEN n=n+1
4117  IF power<limit AND n=INT(level) THEN power=power+1
4120  IF n=INT(level) THEN n=0:pow_to=pow_to+power
4122  IF power>=limit AND power>8 THEN power=power-1
4125  IF power>8 AND (key1=0 OR key1=key2) AND power<limit  THEN power=power-.1666667
4130  AT #3,1,power:STRIP #3,0:PRINT #3," ":STRIP #3,7,0:AT #3,1,power+1:PRINT #3," "
4135  key2=key1:IF total>25 THEN total=total-(power*.95)
4140  IF char=130 THEN char=131: ELSE char=130
4143  IF pow_to>240 AND limit>10 THEN limit=INT(limit-(total/40)):pow_to=0
4145  IF co>=30 THEN co=0:INK #6,7:POINT #6,732,0: POINT #6,732,100
4146  CSIZE 2,1:STRIP 2:INK 5
4150  IF NOT gm THEN AT 8,14:PRINT " ":PAN #6,-power:AT 8,14:PRINT CHR$(char)
4155  IF gm THEN AT 8,14:PRINT " ":PAN #6,-25:AT 8,14:PRINT CHR$(char)
4160  co=co+power
4165  IF total<100 AND tot THEN LINE #6,730,100 TO 680,0: tot=0
4170  IF total<100 THEN fin=fin-power
4175  IF fin<=0 THEN EXIT lo_sprint_1
4180  gm=0
4185 END REPeat lo_sprint_1
4195 tme=ut$(19 TO )+(INT(RND(1 TO 98))/100)
4200 CSIZE 2,0:STRIP 2:INK 5
4205 AT 8,35:PRINT tme
4210 BEEP 1000,1
4215 AT #4,pos,14:PRINT #4,tme
4220 pos=pos+1
4225 IF NOT qual AND tme<21.54 THEN qual=1:INK 0:STRIP 4:AT 13,1:PRINT "YOU HAVE QUALIFIED"
4230 score=score+INT((40-tme)*15)
4232 INK #4,2
4235 IF score>9 AND score<100 THEN AT #4,0,10:PRINT #4,score
4240 IF score>99 AND score<1000 THEN AT #4,0,9:PRINT #4,score
4245 IF score>999 AND score<10000 THEN AT #4,0,8:PRINT #4,score
4246 IF score>9999 THEN AT #4,0,7:PRINT #4,score
4250 hi_scores_2
4255 print_hi_scores
4260 FOR i=1 TO 1500:END FOR i
4265 AT 13,1:STRIP 4:PRINT "                    "
4270 IF pos=5 THEN EXIT lo_sprint_loop
4275 END REPeat lo_sprint_loop
4280 qualify=qual
4285 END DEFine
4999 :
5000 DEFine PROCedure event_5
5005 event=5:pos=2:qual=0:lim=.2
5010 INK #4,7:FOR i=2 TO 4:AT #4,i,14:PRINT #4,"00.00":END FOR i
5011 IF INT(level)=2 THEN lim=.5
5012 IF INT(level)=3 THEN lim=1
5015 REPeat long_jump
5020 draw_jump
5025 foul=0: power=8: n=0:key1=0:key2=0:char=130:x=7:y=2
5030 reset
5035 PAPER 2:INK 1:CSIZE 2,1:AT 7,2:PRINT CHR$(130)
5040 CSIZE 2,0:INK 2:PAPER 4:AT 10,3:PRINT "YOUR GO:"
5045 FOR i=100 TO 10 STEP -1:BEEP 300,i:END FOR i
5046 PAUSE 40000
5050 BEEP 1000,5
5055 AT 10,3:PRINT "         ": AT #3,1,power:STRIP #3,0:PRINT #3," "
5056 CSIZE 2,1:PAPER 2:INK 1
```

We apologise for the small size of this listing, but we wanted to fit it in over two issues – next month sees the remainder.

Start of term! Make sure you have the future of the class in the bag.

Educational Computing is the definitive magazine for educationalists. Whether it is at a primary, secondary or higher education level, Educational Computing ensures you are totally up to date with the applications for computers in teaching.

Find out for yourself. Return this coupon stating how you want to pay. If after reading Educational Computing you find it is not for you. Simply cancel your order, and we will make a refund. No obligation; just an opportunity for you to find out why Educational Computing is so highly respected.



Send your completed coupon to Educational Computing, Priory Court, 30-32 Farringdon Lane, London EC1R 3AU.

A PERFECT COUPLE

Just like ham and eggs and peaches and cream, Commodore User is the perfect companion for your 64 or Vic 20. Every month Commodore User is packed full of the latest new games reviews, special projects and business computing, plus the latest software charts, Tommy's Tips for your tricky computer problems, and much much more. Commodore User, the perfect magazine for Commodore owners, at your newsagent on the first of every month. All for only 95p.

COMMODORE User

INSTANT ACCESS

Each month this directory is updated with new products and information.

If you or your company are currently manufacturing hardware or supplying QL software and would like to be included within this directory, just send details to 'QL User Reference Chart', Dept SE, QL User, Priory Court, Farringdon Lane, EC1R 3AU.

HARDWARE

Monitors

Care Electronics
0923 777155
Philips and other monochrome monitors
Citadel Products Ltd
01 951 1848
Data Distributors Ltd
0990 28921
Kaga, Sinclair Vision
Microvitec PLC
0274 390011
Microworld Computer & Video Centre
0273 671863
Microvitec, Philips, Vision
Opus Supplies Ltd
Redhill 65080
JVC
Printerland
0484 514105/687875
Strong Computer Systems
0267 231246
Microvitec, Philips
Technomatic Ltd
01 208 1177
Microvitec, Kaga
Viglen Computer Supplies
01 843 9903

Printers

Data Distributors Ltd
0990 28921
Datasystems
01 482 1711
Star
MicroPeripherals
0256 473232
Canon
Microworld
0293 545630/0273 6711863
Kaga, Epson, Smith Corona
Printerland
0484 514105/687875
Epson, Brother, Kaga, Canon, Juki
Strong Computer Systems
0267 231246
Brother, Shinwa, Epson, Kaga, Mannesman Tally, Canon, Daisystep, Smith Corona
Technomatic Ltd
01 208 1177
Epson, Kaga, Juki, Brother
Twickenham Computer Centre
01 891 4991
Kaga, Ensign, Canon
Viglen Computer Supplies
01 843 9903

Interfaces

Cambridge Systems Technology
0223 323302
Care Electronics
0923 777155
Computamate Data Products
0782 811711
Miracle Systems Ltd
0272 603871
Printerland
0484 514105/687875
Sigma Research
231 Coldhams Lane, Cambridge
Slave Software
050 846 8866
Q-Disc
SMC Supplies
01-441 1282
Centronics & Epson Serial
Technology Research Ltd
0784 63547
Transform Ltd
089 283 4783

Disk Systems

Computamate Data Products
0782 811711
Compware
0270 582301
CST
0223 323302
Eidersoft
0708 852647
Medic Data Systems Ltd
0256 475244
MicroPeripherals
0256 473232
Printerland
0484 514105/687875
Quest
04215 66488
Silicon Express
0533 374917
Strong Computer Systems
0267 231246

Modems

A>Line Computer Systems
0533 778724
Commpak Data
0792 473697
MicroPeripherals
0256 473232
Modem House
0392 69295
Strong Computer Systems
0267 231246
Tandata
06845 68421
(OEL)

Memory Expansion

Eprom Services
0532 667183
Medic Data Systems Ltd
0256 475244
PCML Ltd
0372 67282/68631
QL+RAM cards
Quest
0421 566488
Simplex Data Ltd
01 575 7531

Extras

A>Line Computer Systems
0533 778724
4-way mains filter/adapter
Action Computer Supplies
01 903 3921
Mains spike eliminator
Anglo Services Ltd
0705 671421
QL Eprom Programmer
Broomspring Specialities
0742 737000
Table-top stands
Classified Product & Services
0930 52204
Leads etc.
Computer Supplies
146 Church Road, Boston, Lincs
Joysticks
Eidersoft
0708 852647
Quicksoft II Joystick and accessories
4 Systems
68 Foxwood Close, Feltham, Middx
Cartridges & box
Management Science Ltd
17 West Hill, London SW18
QL case
Power International
0705 756715
Mains spike eliminator
Sigma Research
231 Coldhams Lane, Cambridge
Joystick
Sinclair Research
0276 685311
SMC Supplies
01-441 1282
Joystick adaptor
Spectrotek
0669 20565
QL repair service
Transform Ltd
089 283 4783
QL dust cover, microdrive storage box, RS232 lead
Viglen Computer Supplies
01 843 9903
Voltmace Ltd
Park Drive, Baldock, Herts
Printer stand

SOFTWARE

Utility Programs

Accountancy Software
Sinclair Research
QL Cash Trader
Adder
0223 277050
Q-Doctor, Assembler
Bedsoft
30 Lansdown Road, Bedford, Beds
Screen Editor
Champagne Computers
Amsterdam 020-149130
QL boek
Computer One
0223 862616
Pascal, Forth, Assembler, Typing Tutor, Monitor
Co-op Soft Ltd
0272 22223
Civil/Structural Engineering
DA Bandoo
81 Mount Pleasant, Wembley
Assembler, Screen Editor
Data Management
0904 760351
SBUTIL, Mbackup, Terminal, Chargen, SBextras, FM
Digital Precision
01 527 5493
QL Super Sprite Generator, Games Designer, Monitor+Dissassembler
Eidersoft
0708 852647
QSPell, Archiver, QL Art
Flite Software Ltd
010 353 7423023
Equate (maths package)
GST Computer Systems
0954 81991
QL Assembler, 68K/OS
Harcourt
Sinclair Research
QL Touch 'n' Go
Hisoft
0582 696421
MonQL
J&D Software
3 Alfred Road, Lowton, Warrington
Metacomco
0272 428781
Assembler, BCPL, Lisp
MicroAPL
01 622 0395
Micrologic Consultants Ltd
57 Station Road
Micro Processor Engineering Ltd
21 Hanley Road, Southwater, Horsham, E Sussex
QL Terminator Emulator
PCS
8 Oak Grove Way, Bridgwater, Somerset
PCS Utilities
Portfolio Software
PO Box No 15, London SW11
Positron Computing
0554 759624
Hi-res screen dump
Printerland
0484 514105/687875
Metacomco Assembler
Psion
01 723 9408/0553
Quill, Abacus, Easel, Archive
QCode
42 Swinburne Road, Abingdon, Oxon
Terminal Emulation, 68000 Assembler/Editor
QJump
Sinclair Research
QL Monitor, QL Toolkit
QSoft
01 499 7417
Agenda
Quest
04215 66488
Business Accounts
Saltgrade Software
31 Royal Terrace, Edinburgh EH7
File Manager, File Editor
Sigma Research
231 Coldhams Lane, Cambridge
Sketchpad
Slave Software
050 846 8866
Arable Farmer Software
Strong Computer Systems
0267 231246
Super Plant Software
097 231246
Plant & gardening software
TDI Software Ltd
0272 742796
USCD Pascal, USCD Fortran 77,

Advanced Development Toolkit, USCD P-System, USCD Prolog
TR Computer Systems
093 924 621
QL Payroll
Triptych
Sinclair Research
QL Decision Maker, QL Entrepreneur, QL Project Planner
WD Software
0534 81392
Xpert Software
0460 42023
Business software

Games

Bedsoft
30 Lansdown Road, Bedford, Beds
Gambler, Beat the Clock
Blain Software
8 Berkeley Close, Staines, Middx
Merry Muncher, Fire Tower, Advance Invaders
Brainstorm
4 Lindsey Close, Cramlington
Westminster Palace
Champagne Computers
Amsterdam 020-149130
DR Q Leap
CP Software
10 Alexandra Road, Harrogate
Bridge Player
Digital Precision
01 527 5493
QL Super Backgammon
Eidersoft
0708 852647
QL Art, Zapper
English Software
061 835 1358
QL Hyperdrive
Equate
2 Ffordd Denwyn, Penyffordd, Chester
Solar Invaders, Wall Breaker, Draughts, Mind Your Path, Statistical Averages, Calendar
Microdeal
0726 68020
Havoc, Frogger, Cuthbert
Intersoft
7 Richmond Road, Exeter, Devon
Executive Adventure
New Horizon Software
Fourwinds, Cwn Lane, Rogerstone, Gwent
Pacman, QBERT, Gold
Peak Electronics
32 Clifton Avenue, Hartlepool, Cleveland
QL Colour Quest
Printerland
0484 513105/687875
Psion Chess
Psion
01 723 9408
Psion Chess
Rodent Software
3 Brookend Crescent, Henley-in-Arden, W Midlands
Adventure Writer, QL Artist, 2×7 Games cartridges
S&B Software
20 St Nicholas Street, Diss, Norfolk
Fantasia Adventure
Shadowsoft
0296 669740
Area Radar Controller, Strategy
Snowsoft
6 Bousefield Crescent, Newton Aycliffe, Durham
Hungry Harry in the Haunted House, Doom Room: Exod. VIII
Sinclair Research
0276 685311
Psion Chess
Summit Software
36 Wood Crescent, Rogerstone, Newport, Gwent
Frogger, Dungeon
Swansoft
164 Vicarage Road, Morriston, Swansea
Space Trek
Talent Computer Systems
041 552 2128
ZKUL, WEST, GraphicQL
WD Software
Hilltop, St Marys, Jersey
WD Morse Tutor

BOOKS

Publishers

Adder
0223 277050
Century
01 434 4241
Collins
01 493 7070
Duckworth
01 485 3484
Ellis Horwood Ltd
0284 789942
Granada
01 493 7070
Harper & Row
01 836 4635
Hutchinson
01 387 2811
Interface
9-11 Kensington High Street, W8
McGraw Hill
0628 23431
MicroPress
0892 39606
Melbourne House
01 940 6064
Prentice Hall
0442 58531
Sunshine
01 437 4343

Book Titles

QL Archive
Murray
(Century £14.95)
Advanced Archive on your Sinclair QL
Davison
Sigma Press £7.95
68000 User Guide
Fleetwood
Sigma Press £8.95
SuperBasic on the Sinclair QL
Alcock
Cambridge University Press £5.95
The Real Thing
Hall
Sigma Press £7.95
Inside the Sinclair QL
Naylor & Rogers
(Sunshine £6.95)
Sinclair QL in Business
Handley
(Newnes Microcomputer Books £9.95)
LISP: A Gentle Introduction to Symbolic Computation
Touretzky
(Harper & Row £11.95)
LISP, The Language of Artificial Intelligence
Berk
(Collins £9.95)

MICRODRIVE EXCHANGE

Each month, for a trial period, this column will contain details of readers' programs that we are able to offer on microdrive.

In return for a small administration charge (per program – including a royalty for the author), we will copy onto blank microdrives any or all of the featured programs.

Each program will be a direct copy of the published listing, or an extended version of that listing where the program in question was too long to print in full (programs for which an abridged version has been published are marked with an asterisk).

It must be stressed that we are not selling the software itself, nor providing any guarantee that it performs any particular function (though we do check every program that is to appear in *QL User*), we are merely offering a service to readers who wish to obtain *QL User* programs on drive rather than by typing them in straight from the page.

HOW TO ORDER

Listed below are programs which have appeared as listings inside *QL User*.

To the right of each program entry is a small box, which you should mark with a bold cross if you want to order that program.

Once you have put a cross next to all the programs you wish to have copied onto microdrive, simply complete the rest of the order form and send it along with your PO/cheque AND BLANK FORMATTED DRIVE to:

QL User, MICRODRIVE EXCHANGE, Priory Court, 30-32 Farringdon Lane, EC1R 3AU.

If you wish us to supply the drive, please add an extra £2.50 for every drive required and mark the order form appropriately.

Please allow 28 days for delivery.

ORDER FORM

Author	Language	Program Name	Price	Issue	Size
Giles Todd	(B)	DIY Assembler	£5.00	Jun/Mar	120 ☐
Converts Assembler source into m/c object code					
Richard Cross	(AO)	Mini Monitor	£3.00	Oct	60 ☐
Pocket sized monitor with comprehensive facilities					
A Didcock	(B)	Connect4	£1.00	Sept	15 ☐
Pit your wits against the QL					
Shergold & Tose	(B)	* Golf	£2.00	May	35 ☐
From fairway to green on 50 different courses of varying difficulty					
Williams & Holliday	(AO)	Paladin	£5.00	Apr	70 ☐
The basis of our games programming series – a space invaders type game written entirely machine code					
Richard Cross	(MB)	Sprite Animation	£2.00	Apr	50 ☐
A subtle blend of machine code and SuperBasic that produces a versatile sprite designer and high speed animator					
Steve Deary	(B)	Pacman	£1.00	Mar	20 ☐
A reasonably fast rendition of the famous arcade favourite					
Andy Carmicheal	(B)	Family Tree	£3.00	Aug	100 ☐
Archive program and database for setting up and displaying large family trees					
James Lucy	(B)	Composer	£3.00	Oct	50 ☐
Compose and play sheet music on the QL					
Mathew Capp	(B)	Miners	£2.00	Aug	30 ☐
A nail biting management simulation that puts you in charge of the NCB					
PJ Smith	(B)	* DIY Adventure	£1.00	Feb	60 ☐
A skeleton framework where you simply have to slot in the details to create your bespoke adventure					
R Green	(B)	Qthello	£1.00	Aug	25 ☐
A 3D version of the well known board game Othello for one or two players					
S J Ackers	(B)	*Touch Type	£4.00	Aug	80 ☐
Touch typing course – 14 lessons, on-screen keyboard, 800+ word vocabulary and WPM readout					

B = SuperBasic, AO = Assembler + Object Code (ready to run), MB = Machine Code + Basic loader

Name_____

Address_____

No of programs ordered ..... **Total cost** £.....

Total sectors (max 200 per drive) .....

No of drives sent .....

No of drives required ..... @ **£2.50 each** £.....

Plus postage & packing £ 0.75

Sub total £.....

Subscribers reduction *(if applicable) @ −10% £.....

Add VAT @ 15% £.....

TOTAL TO BE SENT £.....

Please copy onto microdrive the programs above which I have indicated with a cross. I enclose a cheque/PO to the value of £..... (made payable to *QL User*). I understand that *QL User* only undertakes to SUPPLY these programs (copied onto microdrive) and accepts no liability for their operation as defined by the author. Neither can *QL User* supply additional information about any of the listings other than that originally printed in the magazine (any article reprints required must be ordered and paid for separately at £1 each inclusive of post and packing).

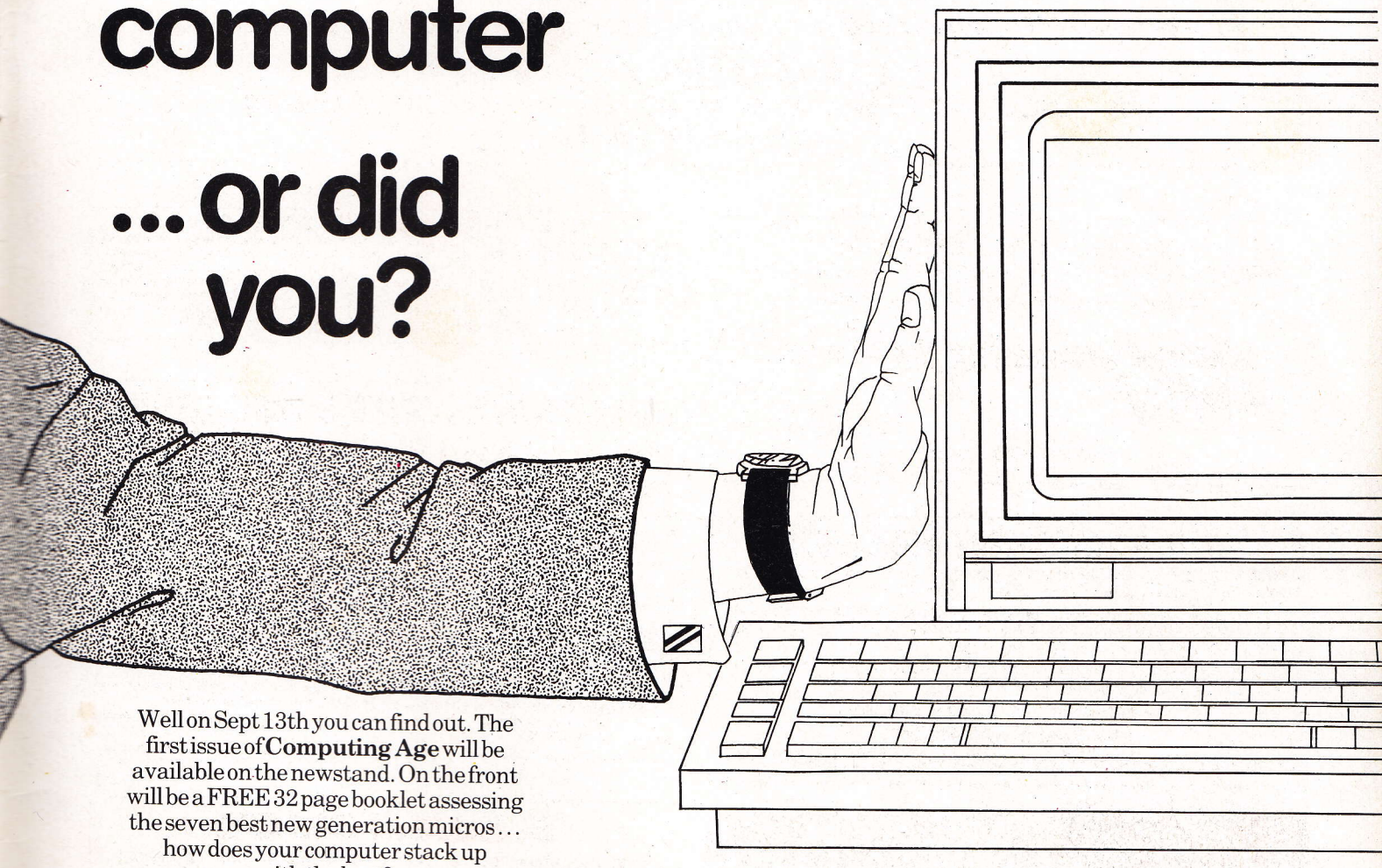
***Subscribers to QL User may deduct 10% off the Sub Total on their orders.**

CLASSIFIED

AD INDEX

You bought the wrong computer

...or did you?

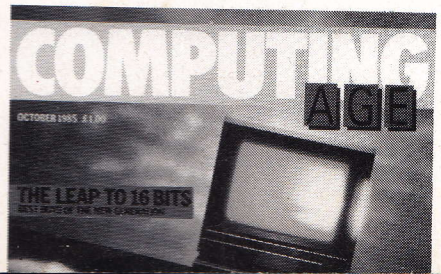


Well on Sept 13th you can find out. The first issue of **Computing Age** will be available on the newstand. On the front will be a FREE 32 page booklet assessing the seven best new generation micros... how does your computer stack up with the best?

Computing Age is a brand new monthly magazine aimed at the serious computer enthusiast.

With the emphasis firmly on applications, new developments and strong communications coverage, the first issue explains how to log onto Telecom Gold; the incredible speed and storage possibilities of compact discs; blowing your EPROM on the BBC; a comparison of new 16 bit machines and what *NOT* to buy this Christmas.

If you're serious about computing, get **Computing Age,** on sale Sept 13th.



Now you can create new dimensions in Computer Graphics.

Create superb colour pictures on your QL with TALENT's outstanding new graphics package. It's supplied on two microdrives – the first holds the master program and a printer dump utility, the second, three demonstration pictures. Backup copies can be made. GRAPHIQL comes with a detailed, clearly-written 60 page instruction manual, outlining the program's many facilities.

- Freehand drawing, 8 colours, optional flash
- Rubber banding, rubber boxes, even rubber circles and ellipses
- Variable size texture definition
- Doodle pad
- Colour and texture fill of any shaped area
- User definable paint brush – any colour or width
- Colour list for full control
- Re-colour facility
- Magnification with panning
- Mirroring and rotation of blocks of screen
- Air-brush effect
- On-line 'help' facility
- Full file-store access
- Printer dump utility.

Text can be included in pictures. The characters can be single or double height with flash and underline. GRAPHIQL pictures can be put into BASIC or assembler programs with the sample routines provided.

Available from selected branches of Boots & WH Smiths.
£34.95 + 50p postage & packing

TALENT
COMPUTER SYSTEMS

Curran Building, 101 St James Road, Glasgow G4 0NS. Tel: 041-552 2128 (24 hour credit card hot-line) Software from Scotland

QL and Microdrive are registered trade marks of Sinclair Research Ltd.