An EMAP Publication

SINCLAIR'S SUPERMICRO

NOW MONTHLY

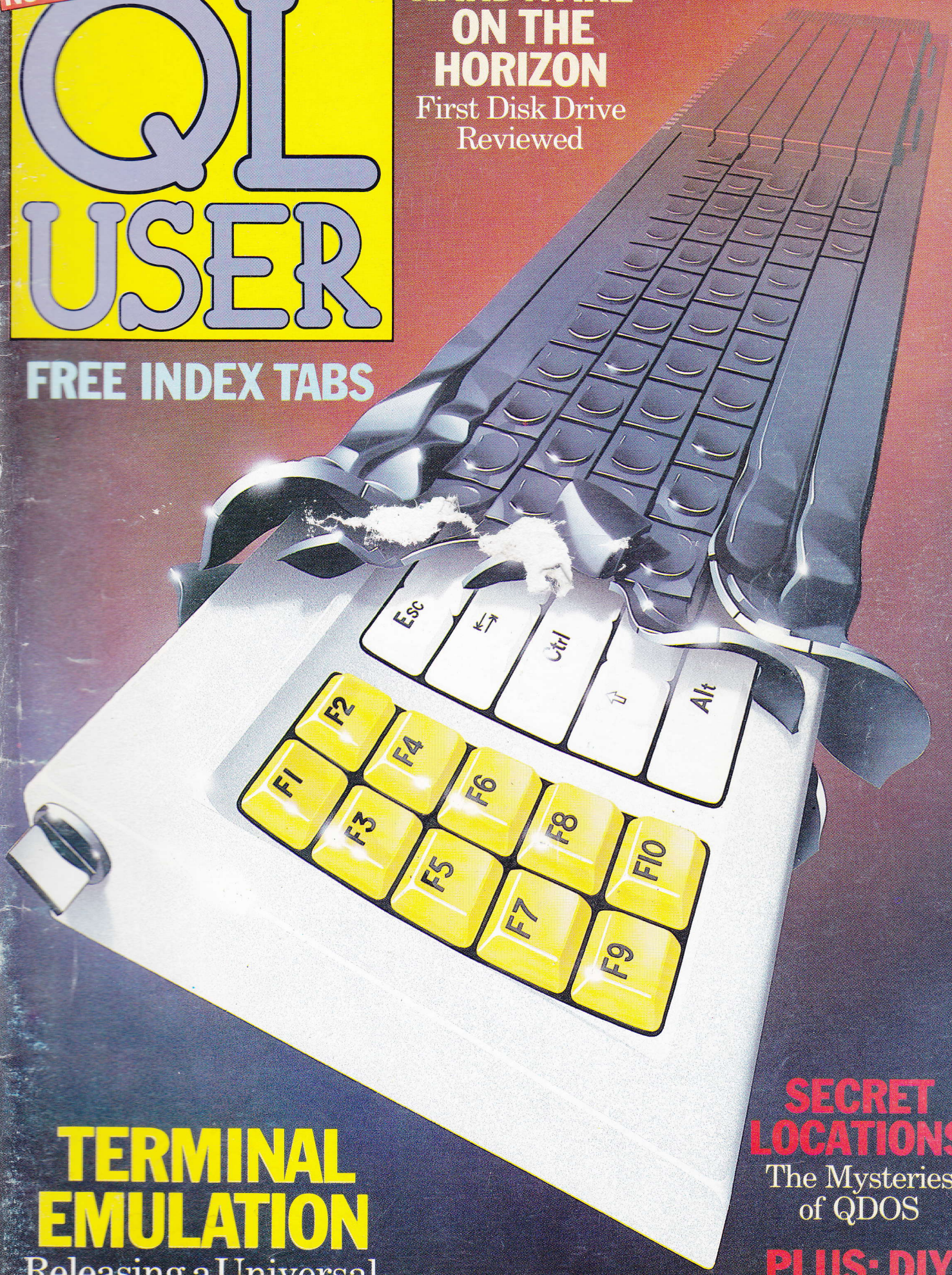
QL USER

March 1985 95p
Germany DM5.50

HARDWARE ON THE HORIZON

First Disk Drive Reviewed

FREE INDEX TABS



TERMINAL EMULATION

Releasing a Universal Communicator

SECRET LOCATIONS

The Mysteries of QDOS

PLUS: DIY ASSEMBLER

AVAILABLE FROM
High Street Computer Retailers and branches of W.H. Smith, Boots, John Lewis Partnership, Laskeys, Currys.

EVEN THE PRICE WILL KEEP YOU IN THE BLACK

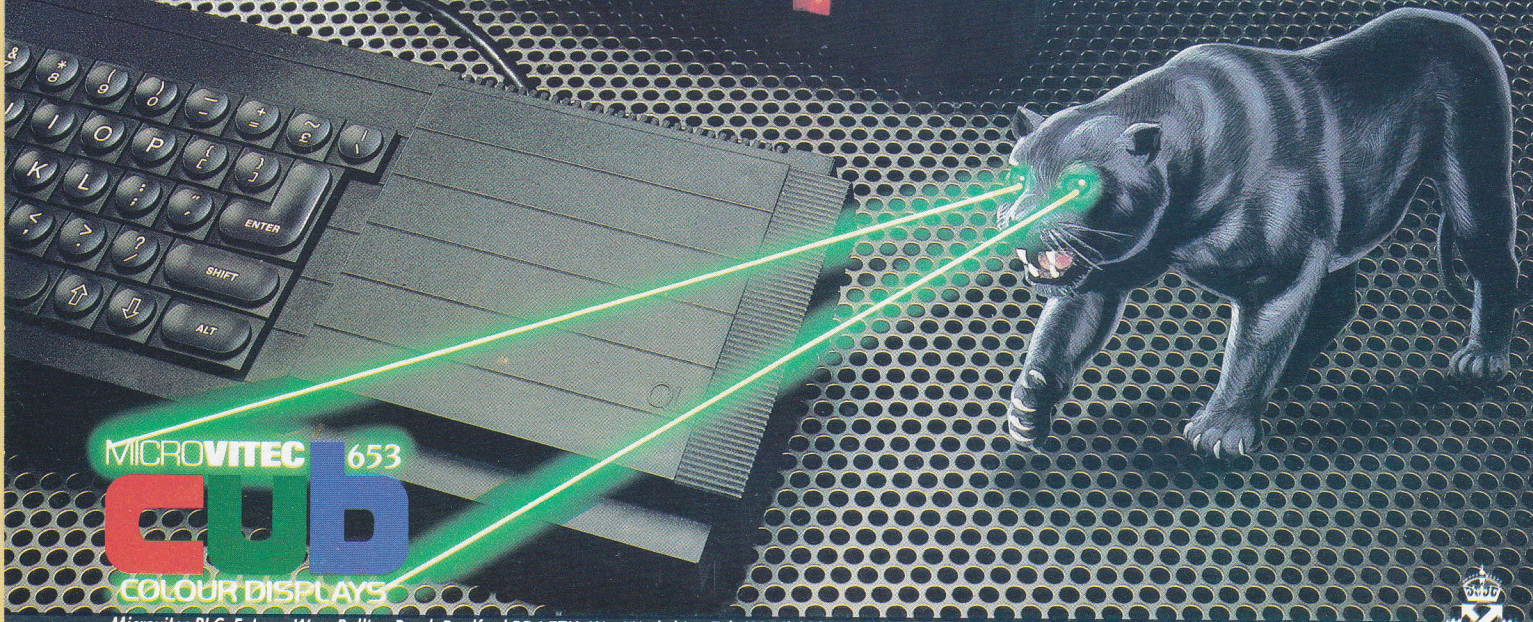
This sleek, black Microvitec CUB medium resolution colour monitor has been designed to be totally compatible with the Sinclair Q.L. An ability to display 85 column text is combined with outstanding graphic capabilities – at a price that won't break the bank.

£275
(inc. VAT)

Specification

Model:
CUB 1451/DQ3
14" Q.L. Monitor
RGB TTL input
Tube Resolution (pixels)
653(H) x 585(V)
Dot Pitch 0.43mm
Bandwidth 18 MHz

Also available in Standard Resolution version for only £225 inc. VAT.



MICROVITEC 653
CUB
COLOUR DISPLAYS

Microvitec PLC, Futures Way, Bolling Road, Bradford BD4 7TU, West Yorkshire. Tel: (0274) 390011. Telex: 517717

QL USER

March 1985

Editor	Paul Coster BSc
Assistant Editor	Paolo Baccanello
Editorial Assistant	Shirley Eborn
Art Editor	Mike Spiller
Technical Consultant	Adam Denning
Associate Editor	Peter Rodwell
Advertising Manager	Phil Baker
Advert Production	Yvonne Moyser
Sales Executive	Tracey Keighley
Publisher	Terry Pratt

Contributors

Sid Smith, Leon Heller, Nicky Trevett, Adam Denning, Peter Rodwell, Alan Turnbull, A Nally, Giles Todd, Mary Sargent and Steve Deary

Editorial 01-251 6222

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

Advertising 01-251 6222

Cover illustration by Vincent Wakerley.

Competitions

Due to problems associated with our recent move, we have been unable to locate the addresses of certain competition prizewinners. No doubt these readers are wondering about their prizes and we would ask winners who have not yet received prizes to write or phone our editorial offices so that delivery can be arranged.

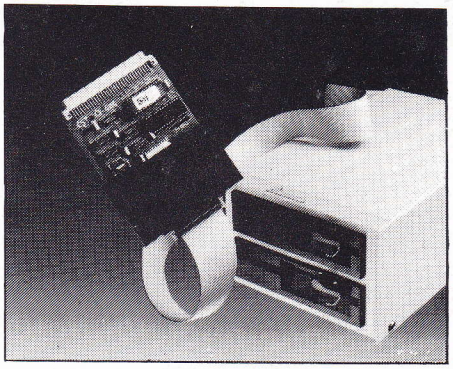
Information

It seems that some readers were confused about the function of last month's free utility program. Those who typed it in will have discovered that it is completely self documenting.

However, if you're still wondering, it links up with the template (also free with the February issue) to provide single key file operations such as copying, deleting, running and printing a directory.

Finally, a note to all budding authors, whether experienced journalists or keen, first-time writers. We are currently paying top rates (up to £120 per text page) for articles which meet the required standard. All submissions are welcomed and will be carefully considered.

CONTENTS



Published and Distributed by EMAP Business and Computer Publications Ltd.
Typesetting by Contemporary Graphics Ltd., 20 Bowling Green Lane, London EC1.
Printing Riverside Press.

QL Mk II

A second version of the QL, incorporating the Psion software on internal ROM chip, is due for imminent release. Rumours are also circulating of a QL built around a disk drive instead of microdrives, and we can reveal that Sinclair is contemplating a rival to the ICL One Per Desk.

First news of the ROM software device was leaked to your reporter by Psion chief David Potter.

"Two versions of the QL will be sold together," he revealed, "one with the software on ROM, and one with it on microdrive. The ROM version will offer qualities of speed and integration similar to those on the ICL One Per Desk – though, of course, neither machine is capable of implementing the programs in ways comparable with larger computers."

ROM versions of the QL software were first mooted by Sir Clive last summer during the ill-tempered Press lunch called to smooth his company's ragged relations with the computer mags. Indeed, it was a lunching hack's assertion that any such product was an admission of inadequacy for the microdrive version which produced Clive's fondly-remembered "For God's sake!" outburst.

In those distant days, however, the ROM was envisaged as an external cartridge to be sold as an upgrade to existing owners – far-removed from Potter's talk of a second QL with re-designed circuit board and bumped up price.

Psion Chairman, David Potter.

Sinclair Research has since confirmed that the Potter revelations are "not far off the mark".

No such official support has yet been given to rumours of a QL incorporating a floppy disk instead of microdrives, though such an option (built, presumably, around a 3" or 3.5" disk) would extend the machine's appeal to people who can afford to indulge their belief that microdrive technology – which works well in its 85K Spectrum manifestation – has been pushed too far for the QL.

We have no doubts, however, about Sinclair's ambition to sponsor a rival to the ICL One Per Desk.

The OPD is an excellent concept, but the announcement of forthcoming alternatives from manufacturers such as Acorn Computers has exposed the widespread belief that it may be vulnerable – particularly on price. It's certainly hard to see how the addition of a modem, monitor, real keyboard, telephone handset, ROM software and new operating system can make the OPD worth £1100 more than the straight QL (Acorn, for instance, intend to come in at around half the OPD's price).

Sinclair's sponsored OPD emulator is in the early stages of development by a well-known software-orientated company, whose name we cannot reveal. The product is planned as a microdrive cartridge and manual, costing £79.95 or less and issued under the Sinclair logo.

Intended for existing owners of the QL and its semi-official OE modem, the software will be an extension of Archive and will offer – particularly when the ROM software QL appears – many of the facilities of the OPD.

Instead of the Scicon comms software found in the OE modem, the company behind the OPD emulator has been examining the possibility of C-based material from a third party. My latest information, however, is that the Scicon material will after all be used.

Though not as flash-looking, Sinclair's product will undercut the OPD so heavily that the latter's sales are certain to be threatened. Indeed, the prospect is so vivid that it's hard to imagine no agreement exists restraining Sinclair from such an exploit.

Sinclair Research is refusing to make any official comment on the plan.

All three Sinclair permutations around the QL's central processor board (a component universally admired, and so ripe for exploitation inside other hardware that one industrialist told me, "The QL will be a great success – and not merely as the QL,") are in the mainstream of the company's Searle-inspired resolve to adapt existing hardware for a wide range of market-niches and pocket-depths.

The imminent arrival of 68000-based machines from Atari and Commodore sharpens Sinclair's need to rapidly exploit its one year hardware lead.

Resolution Resolved

Sinclair would appear to have turned over a new leaf in the New Year. The promise of a microdrive cartridge price cut "by the end of 1984" has been fulfilled, give or take a month.

The 60% reduction will be welcomed by all, except possibly the lines of ladies that used to hand-glue the devices together, whilst an expensive automated assembler stood by unused.

QL software houses are likely to be the first to benefit. Previously they were faced with production costs of £4 to £5 compared to 65p or so on cassettes. Since the mark-up demanded by retailers and distributors is calculated on a percentage basis, and currently runs at 50% of the pre-VAT retail price their costs were doubled by the time programs reached the shelves of local W H Smiths.

'West', the QL adventure from Talent Computer Systems, was a case in point – being supplied for the BBC Micro at £7.95, the Commodore 64 at £9.95 and the QL at a hefty £19.95. Talent's Sales Director, John Tweedy spelt out the problem in no uncertain terms. "Whatever program you put on a microdrive cartridge, you're going to have to charge £19.95 minimum; until Sinclair reduce the price of microdrive cartridges, I see no other way of arriving back at our cassette tape profit of £1.50 per program. And that's assuming wholesalers are prepared to reduce their mark-up."

As regards timing, the price cut would seem to have come at a most propitious moment. The appearance of disk interfaces for the QL mean that the cartridge is no longer the only mass storage medium available. Furthermore, the imminent arrival of a cheap cassette interface looks set to further undermine that position.

Whether the price cut will entice more software houses to produce for the QL remains to be seen. Doubtless, concern over the reliability of the medium will also figure in their calculations. "Just recently we've improved QL microdrives immensely," says Sir Clive. "You'll be surprised." We hope so, for if this is the case then Sinclair's strategy of QL storage on microdrive, hard disc and wafer scale RAM suddenly makes sense.

Spirited Rivalry

GST (that's G for George, not C for Charlie), are about to produce the first applications program for 68K/OS – the operating system which Sinclair commissioned them to write way back at the beginning of 1983, replacing it with QDOS (amongst much mutual recrimination) just after the QL's January '84 launch.

Although they got paid for their work on K/OS, and have been gratified by endorsements from 68000 manufacturers Motorola, GST were naturally disappointed to find their baby with neither software nor hardware support.

However, K/OS has since been issued on a plug-in board for the QL, and a range of software products are now imminent.

"The QL with 68K/OS is an extremely versatile system," said the company's Joe Webster – who wrote part of K/OS, "and we are almost ready to announce a word processor, either on tape or in ROM, which would allow the QL to be run as a powerful multi-tasking business machine. We're talking to other companies about Pascal and Fortran compilers, and C is on its way."

Webster conceded that, until it attracted serious support, the position of K/OS would be distinctly uncomfortable. However, after hints about its possible adoption by some of the big American manufacturers about to launch

6800-based machines, and after re-iterating his hope that other software suppliers would approach GST with plans for K/OS material, Webster showed himself to be sufficiently bullish for a quick knifing of rival QL operating systems.

Quest's CP/M 68K implementation was unworkable on microdrive, he insisted, and their disk systems didn't work (Quest deny this, of course).

He hadn't previously heard of the CP/M 80 board from QL-Plus, but didn't see how it could work with QDOS – even when informed that QDOS author Tony Tebby was (naturally) involved in its implementation.

QDOS itself he dismissed as "not even a real operating system", citing its inability to handle multi-tasking windowing operating adequately, its programmer unfriendliness, and its lack of portability to other systems. (All of this is rejected by Tebby – who, it's felt, often defends his brain-child by defining QDOS as only the very core of the QL's firmware, whereas outsiders tend to regard all the QL's firmware as part of QDOS, including – sometimes – SuperBasic itself. QDOS can't be accused of bad screen handling, thinks Tebby, since the screen handling is performed by outer shells of the operating system, not by its flexible, portable, programmer-friendly heart).

Webster insists we shouldn't get too worried about GST. "We are a reasonably substantial, and reasonably successful company. We regard K/OS and associated products as a speculative venture which might bring us a great deal of money. If it does we'll be thrilled to bits – but if the whole thing falls flat on its face, we will survive and still be profitable.

For Sale

"ICL has used some of the custom chips and other components from the QL. There's also been a degree of collaboration with Sinclair Research – in the sense that it's their design but they've had free access to us for information about microdrives and other shared items."

This was how Managing Director Nigel Searle explained the complex story of Sinclair's joint work with ICL on the latter's One Per Desk micro. Equally convoluted is the arrangement Sinclair has reached with GST. This Cambridge neighbour of Sinclair is acting as a sort of technology retailer, selling the QL board to companies wanting to incorporate it in their own machines.

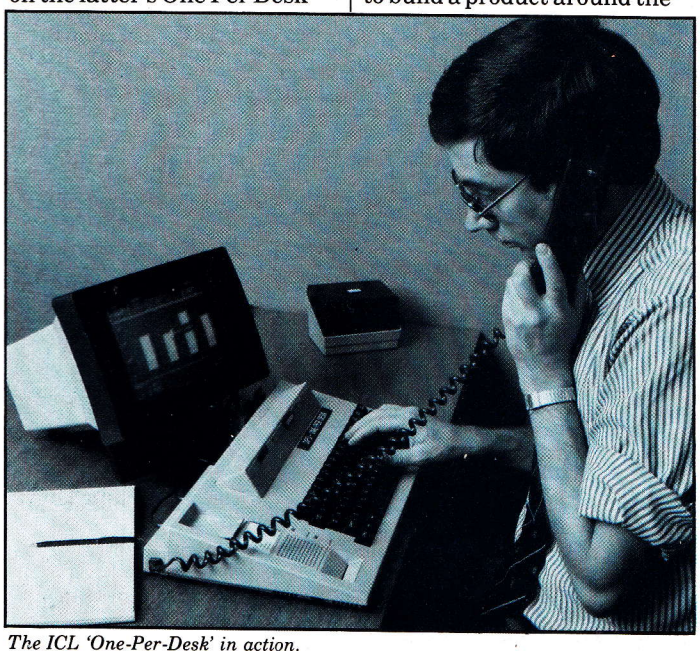
"You know that GST developed an alternative operating system for the QL called 68K/OS," Searle reminded me. "We have a deal whereby we supply GST with boards, and anyone who wants to build a product around the QL board – with or without microdrives and 68K/OS – should talk to them.

"We've got a power of veto over the end product, since we don't want anyone buying the QL board to produce a direct competitor for the QL or the OPD. And, unlike the state of affairs with ICL, we won't be offering GST clients much in the way of technical advice.

"The OPD is a lot more than just a QL," continued Searle. "Although it has the same sort of computing power, it's got a great deal of communications capability and ICL have written their own operating system to take advantage of that."

Interestingly, both Searle and Psion chief David Potter thought that the OPD could read data cartridges from the QL. They're wrong, says ICL; the OPD allocates microdrive data in 1K portions, claimed to be more reliable than the QL's 0.5K.

And what of the OPD itself? Well, the device undoubedly addresses an important new market, but to these eyes it's disappointingly expensive, and too big for the managerial desks it aims to fill – like the QL, it has suffered from Sinclair's inability to make enough flat TV tubes to keep up with a computer assembly line (I didn't like the colour scheme either!).



The ICL 'One-Per-Desk' in action.

Intelligent Terminal

The supply of several hundred QLs to students at a Scottish uiversity seems likely to involve the machine in some of the UK's most advanced computer research.

One hundred QLs from Sinclair Research, and several hundred more from the University itself, will be provided on extended loan to students at Strathclyde University. Strathclyde's importance as a centre of research into machine intelligence, as well as its strong links with the Turing Institute, will mean the development of artificial intelligence (AI) programs on the QL.

The University is keen to use the computer's implementation of Lisp (a language much used in AI work) for tuition and research on such advanced computer capabilities as expert systems – databases capable of responding to enquiries in a way comparable to human experts. On a more mundane level, the QL will be used as a terminal for the University's existing computer network.

Sinclair Research also supports the nearby Turing Institute (named after the British computer pioneer), which was opened late last year as a centre where scientists can come to acquire experience of AI work.

Judith Richards, Assistant Director of Advanced Studies at the Institute, explained that experience in porting a small expert system shell to the Sinclair Spectrum convinced her that the QL could be even more useful in AI research.

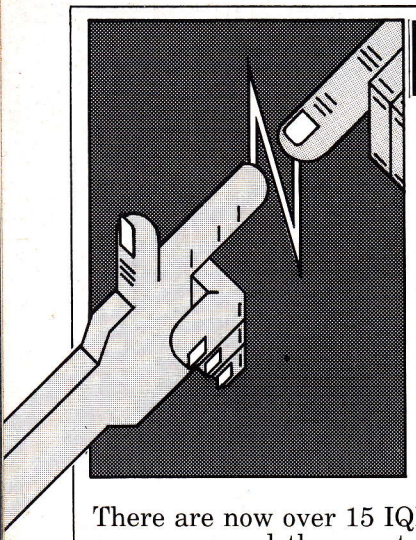
"The QL will be capable of handling much more than just this small demo shell, and because it's inexpensive would form a good basis for people who want experience in expert systems without making a large investment."

Even the Spectrum had been capable of useful work, she explained, the Expertease program from ITL being used – for example – to build up an expert system which could decide on the maintenance schedules of helicopter engines.

"Provided you don't need a really large database, and provided the shell will port over – and there's no reason to believe it won't – the applications possible on the QL will be fairly varied."

Commercially available software might also develop out of the Strathclyde work. "The QL has the capability of producing a great number of business and expert system packages, and if the University works on the machine it's likely that there will be at least some pre-competitive packages around for other people to take into commercial exploitation."

The QL is unlikely to find much use at the Institute itself – where the fashionable hardware is all VAXs and such – but Sinclair expect the University to serve as a test-bed for the machine's usefulness in higher education, apparently hoping that every interested student at Strathclyde will gain experience of the computer, and that, as Judith Richards puts it, "students will become committed to programming on the QL and will want to go on using it when they leave".



USER GROUP

The latest in news and Information from Leon Heller, Chairman of the Independent QL Users' Group.

User Group News

There are now over 15 IQLUG local groups around the country, and on the continent, including London, Birmingham, Manchester and in the west, Bristol.

The Swindon weekend workshop last November is to be repeated three times during 1985, commencing with one in March. The programme will include sessions on assembly language, BCPL, Archive applications, DIY hardware interfacing, disk drives and memory expansion, and communications. Many suppliers of hardware and software will be there to demonstrate their products and answer questions about them.

Lost Memories

Some members who purchased memory expansion modules (such as Simplex Data's 256K unit) have been surprised when using Archive, for example, that they only appear to have about 115K of additional memory. In fact, they have what they paid for, a full 256K of additional RAM, but QDOS filches quite a lot in order to increase the size of its buffers, enhancing microdrive performance. This is borne out by the experience of one member who found that Quill now only "hangs up" for fractions of a second when using overlays, instead of several seconds.

Length Limitation

Using disks on QL (with CST's excellent interface) it appears that Quill has one serious limitation that nobody seems to have discovered (probably because it is so impossibly slow with documents greater than a couple of pages when using microdrives), in that one cannot work with documents longer than about 12 pages. Merging in another document, or copying a block, results in an "out of memory" message and the whole system "locking up", necessitating a reset, and the loss of the text in memory. Archive runs very well on disk, which makes the limitation in the number of records that I mentioned in the December/January issue much more aggravating.

Quill Again

An IQLUG member rang me to ask how to remove a forced page break, since SHIFT –{, as recommended in the documentation, does not work. With a little bit of experimentation, I found that CTRL –{ is the correct way to do it. Whatever you do, DON'T type CTRL as the system will "lock up", and have to be reset, probably resulting in the loss of your document. Psion know all about this bug but it's a pity they seem to keep it t- themselves.

A recent letter from one member complains that if he opens and closes channels repeatedly, incrementing the channel number each time, he eventually gets an "out of memory" message, and the system has to be reset to operate properly. Why he should want to do this escapes me, but the explanation is quite simple. If you open channel #1000, for instance, buffer space for channels 0 to 999 is also allocated by QDOS, so it is wise to keep your channel numbers sequential, starting from #3, as a rule.

Rewrite On Cartridges

If you didn't enclose your Psion software cartridges when returning your QL to be upgraded, and now have two sets, the write-protection can be defeated by sticking a small piece of Sellotape 15mm square over the missing lug on the right-hand side, enabling the old cartridges to be re-used. At nearly £5 a time, this makes a lot of sense. My thanks to Bill Cowhig for this tip.

Disk Save

Now that I have double-sided disk drives on my QL, but still possess plenty of the single-sided variety (used formerly on my TRS-80), I find the latter are quite satisfactory. QDOS should lock-out faulty sectors on disk anyway, when formatting, just as it does with microdrives, so it is probably not too risky. Anyway, it saves a few quid!

Plug Lugging

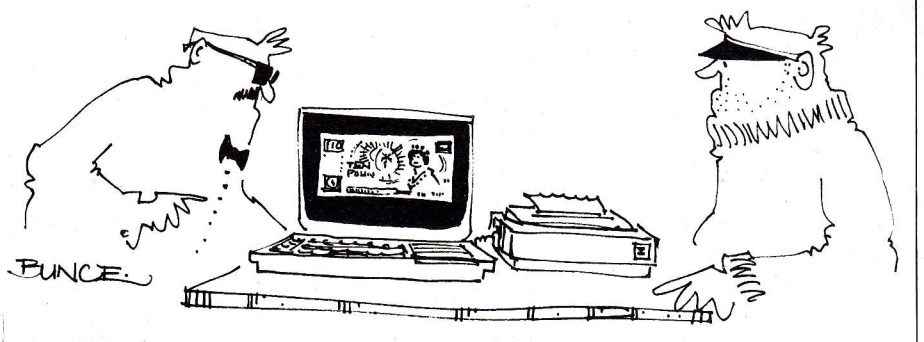
Don Gehring has informed me that QL power supplies are available separately from Sinclair at Camberley for £9.95 each. One of these might be very useful if, like Don, you take your QL to work with you, and don't want to lug the power supply around, as well.

IQLUG is a non-profit making group, with officers democratically elected by members at an Annual General Meeting. Accounts are independently audited and made available to members. The group is affiliated to the Association of Computer Clubs.

Membership is by subscription to Quanta, the group's monthly newsletter, containing 40 pages of members' letters, hints and tips, news on the QL scene, program listings, reviews and so on.

The group maintains a (mostly) free software library, which currently contains about 100 programs. All library software is written by our members, and only non-commercial items will be held there. In addition, a free advice service is provided: members can phone in with their problems, and be put in touch with someone who can help them. A register of members with expertise in various areas is kept.

Membership details are available from: Brian Pain, 24 Oxford Street, Stony Stratford, Milton Keynes MK11 1JU. Telephone: 09087 564271



O.K. Wilf, *print.*

CASE SOLVED

£69.00
+P&P.

- ***antler*** manufactured specially for the QL System.
- Angled Keyboard for practical in-case use.
- Space for 24 Cartridges, QL - accessories & Manual.
- Size conforms to Airline hand luggage specifications.
- Fully Guaranteed for 2 years.

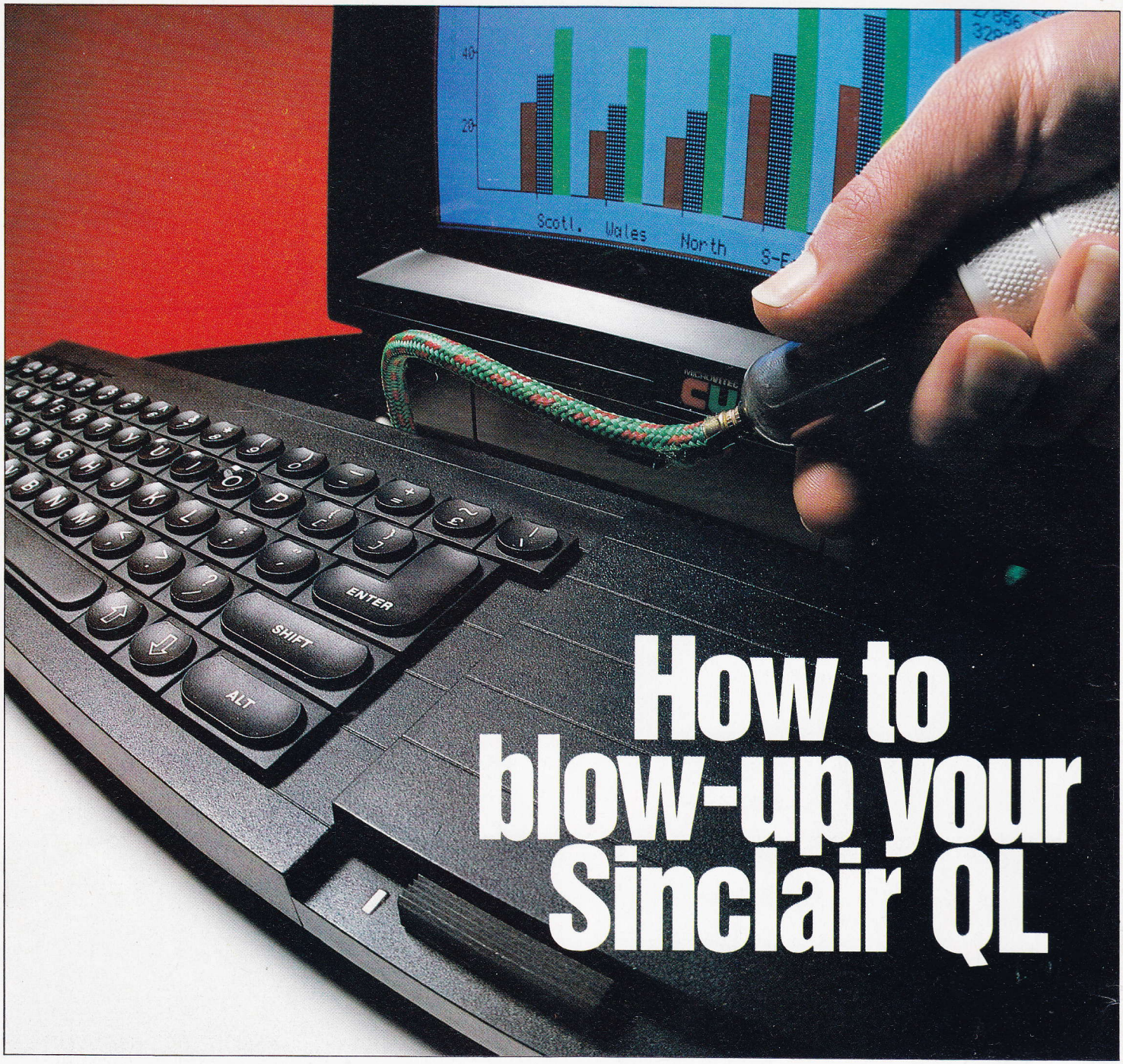
Offer is for the case only.
'QL' is a registered trademark of Sinclair Research Ltd.

To: Management Science Ltd. QL Case offer, 17 West Hill, London, SW18 1RB.
Please supply ..... Case(s) for my QL(s) at £69 & £4 p&p. (inc. VAT)
I enclose a cheque/postal order for £.....
Made payable to: Management Science Ltd.

Name: .....
Street: .....
Town: .....
County: .....
Postcode: .....

Please allow 14 days for delivery.

QL/3/85



How to blow-up your Sinclair QL

First, fuse together your Sinclair QL with **QL/APL**. Immediately you've expanded your QL's potential to the maximum.

The unique **QL/APL** programming language frees you from the constraints of loops and endlessly long programs. **QL/APL** lets the computer cope effortlessly with data types and numeric precision and all that technical stuff. Leaving you free to get your QL to work for you, not against you.

QL/APL is easy to work with, so it's simple to write your own applications.

QL/APL is modular, so you can build up systems step-by-step.

QL/APL is interpreted, so changes can be made and tested immediately.

QL/APL is incredibly powerful, so you never come up against annoying limitations. It's the key to taking advantage of the full 32-bit power of the QL's 68008 processor. It runs on the standard QL keyboard, and allows full access to QL colour graphics and windows. And it's built around MicroAPL's highly- respected IBM-compatible implementation of the APL language – in fact APL was originally developed by IBM for its mainframes.

A software pedigree you can trust implicitly.

Remember, for QL, read QL/APL. And blow the rest

To: MicroAPL Ltd, Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London SW8 5BP.

- ☐ Please send me more details about the power of **QL/APL**.
- ☐ Please send me ____ copies of **QL/APL** at £99.95 plus £1.95 p&p
- ☐ I enclose cheque/P.O. made payable to MicroAPL Ltd.
- ☐ Please charge my Visa/Access card number _____

Card expiry date _____

Name: _____

Address: _____

Tel No _____ QL/3/85

MICRO APL

QL and Sinclair are trademarks of Sinclair Research Ltd.

(Prices include VAT. Please allow 28 days for delivery. For credit card telephone orders, phone 01-622 0395).

"UNLEASH YOUR QL"

NEW SYSTEMS PACKAGES – NOW WITH *QDOS* COMPATIBLE INTERFACE & DISK DRIVES

CAN YOU GET THIS KIND OF SERVICE IN THE HIGH STREET?
WE PROVIDE FULL TECHNICAL ADVICE & AFTER SALES SUPPORT.

NO INTERFACE PROBLEMS	JUST PLUG IN AND GO!	FACTORY-TESTED BEFORE DESPATCH TO YOU	ALL CABLES SUPPLIED	PRINTERS SET UP READY TO USE	WE EVEN FIT MAINS PLUGS TO EVERYTHING

PRINTER & MONITOR PACKAGE DEALS + QL OPTION

PRINTER CHOICE	WITH MICROVITEC COLOUR MONITOR	OR PHILIPS GREEN/AMBER MONITOR
Brother HRS + Mains Adaptor	£425	£265
Brother M1009	£459	£295
Epson RX80F/T	£585	£420
Epson FX80	£725	£590
Epson FX100	£869	£735
Canon PW1080A NLQ	£649	£495
Daisystep 2000	£599	£445
Brother HR15	£715	£569
Epson DX100	£745	£590

Printers are all supplied with RS232C interface. Monitors display full 85 columns of text.

SINCLAIR QL SUPPLIED WITH ANY SYSTEM – £389

NOW ADD DISK DRIVES!

QL Disk Interface	£149
5¼in. Computamate (200KB)	£299
Disk Drive + QDI (400KB)	£336
+ Power Supply (1000KB)	£367
Dual 5¼in Disk Drive (400KB)	£459
+ QDI + Power Supply	£515
(2MB)	£574
3½in Computamate (200KB)	£299
Disk Drive + QDI (400KB)	£336
+ Power Supply (1MB)	£367

Prices include cable, manual.

MONITORS FOR QL

COLOUR:

Microvitec 1451DQ3	£275
Prism	£199

MONOCHROME:

Philips BM7502 (Green)	£92
Philips BM7522 (Amber)	£92

ACCESSORIES

Interlocking Storage Box for 10 Cartridges	£5.95
Epson RS232 I/F	£30.00
Epson RS232 I/F 2K	£65.00
Miracle System I/F	£39.95

ALL PRICES INCLUDE VAT.
FOR DELIVERY ADD £10 (Systems/Monitors). ADD £2 (Accys).
Other Systems can be supplied on request.

ZEAL MARKETING LIMITED

VANGUARD TRADING ESTATE, STORFORTH LANE,
CHESTERFIELD, S40 2TZ
Telephone: 0246 208555 Telex: 547697

NEXT MONTH

Games Programming: Machine Code Techniques

A new series designed to teach low level games programming, using a modular approach. Each month a different area is covered, so that by the end you should be well-versed in all aspects of writing machine code games.

Sound Synthesis

Despite the QL's limited sound capabilities, there's still a lot you can do with a few simple commands. We tell you which . . .

Readers' Programs Special

No doubt due to the winter cold spell, we've been inundated with readers' programs. Here's your chance to see the best of them.

Sinclair Software

With the drop in microdrive prices, software is beginning to take off. We take a look at what's on offer from Sinclair and others in the software market.

The Typewriter Interface

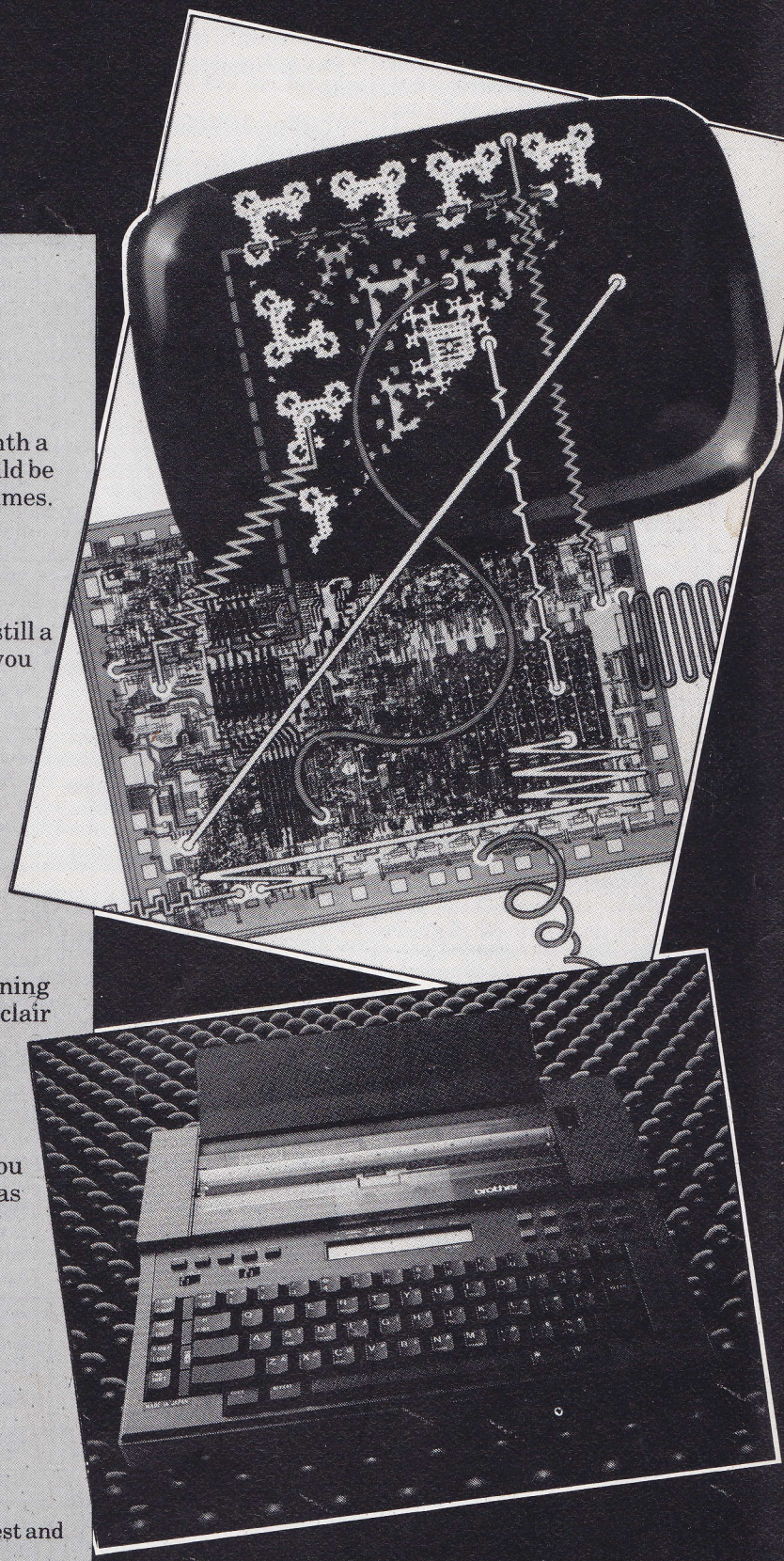
Printers and monitors are not the only peripherals you can connect to the QL. Electronic typwriters can act as second keyboards, printers and communications terminals. We survey the potential candidates.

PLUS:

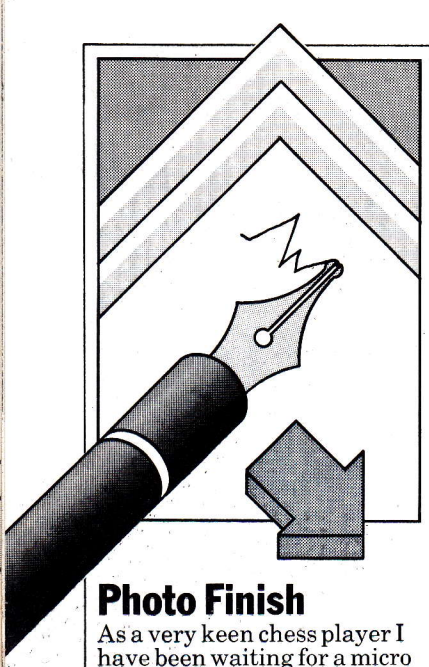
—Turtle Graphics and how to use them.
—Relational databases on Archive.
—Experimenting with Abacus.

AND:

Part two of our disk drive reviews, with products from Quest and MicroPeripherals.



APRIL '85 EDITION ON SALE: 21st MARCH



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, EC1R 3AU.

Photo Finish

As a very keen chess player I have been waiting for a micro to emerge with a strong chess program before plunging into the 'micro revolution', ie I'm one of those rare beasts who doesn't own a computer (gasps from readers). I read with great interest your review of *Psion Chess* but unfortunately you didn't mention one extremely important point . . . I assume *Psion Chess* recently won the 4th World Micro Computer Chess Championships? Can you confirm that this is the same program, as if it is, I for one, will buy a QL on the strength of that alone. I hope the computer didn't really play D7-D5 (Q to Q4) in the position shown on Level 11, or it's back to boring human opponents!
A E Millward
Sheffield

Psion Chess *did share first place with three dedicated computer chess machines though it must be added that the program ran on a full 32-bit Sage computer. However, aside from a reduction in speed its performance should be identical on a QL.*

As for the move D7-D5 we have to admit that the whole set-up was contrived so as to produce an interesting photograph as opposed to good chess and level 11 was selected only after all moves had been made on level 0 (novice).

Memory Drain

Are you under the impression that if you 'NEW' a BASIC program on the QL, then it disappears from memory never to be seen again? Not so! When you load another program, the previous one is still in memory; in fact, if you load a succession of programs, they *all* stay in

This short program will show this:

```
10 FOR i=213500 to PEEK_
L(163856)
20 PRINT CHR$(PEEK (i));
30 END FOR i
```

The address 213500 is approximately where the program starts, it seems to vary slightly. Also held is the microdrive directory, which seems to include some items which have previously been deleted from the cartridge.

I can find no way to access these multiple programs, so this seems to be a fairly useless discovery. However, I think we should note that they are taking up memory space, and should you need a lot of memory for a program, this may cause problems.
N J Lennon
Newquay, Cornwall.

Gone To C

Where has "C" gone? I looked forward to further instructions as I am writing large business programs in BASIC and this seems to be a perfect alternative in both speed and memory usage.

If for whatever reason you do not intend to pursue "C" further can you advise where I may find a book on it.
Paul Thurlow
Dorset

Well 'C' is back and will remain a regular feature. However, the fact that a compiler has yet to come out for the QL does seem to dispell any urgency in the matter. Hopefully, one will materialise from the US (Lattice) in the near future.

Whilst C is fast and efficient it can hardly be considered the perfect alternative to BASIC, rather a unique half-way house between Pascal and Assembler. As a tool for writing large business programs it is as useful as a scalpel is for slicing bread. The incisions may be precise but it will be a while before you sit down to eat.

Don't however be put off, as C is nevertheless a great language. The definitive work on it is Kernighan and Ritchie's "The C programming Language" (Prentice Hall – £21.95).

Tolerant Customer

I have had a QL since July, but have only just come across your very valuable publication. What I read there rather surprises me, but suggests that either I have been very lucky, or more likely that the QL is not a beginners machine. The complaints about the standards of documentation, which is particularly high in the case of the Psion packages, seems to bear out the latter conjecture.

I have experienced the following difficulties:

1) Three of the four applications packages would not read when received. Solution: return to Sinclair. New copies were received within 10 days. I have had no other problems with the microdrives since.

2) Quite bizarre behaviour from Quill when using tabs together with a text width greater than the screen display width. Solution: Use 80 column display width, this is just about acceptable with a 12" monochrome television, very good with a 9" Hitachi monitor.

3) The Psion printer driver cannot be set up to switch off underline on my Brother HR5 (no problem in SuperBasic). Solution: Use the enlarged character set instead while waiting for next edition (who needs underlining anyway).

There are faults. Quill use too much overlay, which slows it down, but the next edition should cure that. The keyboard action is less than perfect, but acceptable at the price. The manual does not give enough information on the operating system. Export is clumsy. But my main complaint is that whoever decided that the microdrives should be referred to as mdv1_ and mdv2_ rather than A: and B: should be shot.
R L Galloway
Herts

Humbug!

In your Dec/Jan issue of QL User a G.L. Riddle had a letter on the Open Channel page about a bug in the SuperBasic DIM statement. I do not think this is a bug. The 68008, being a 16 bit processor, must fetch WORD information from even memory addresses. Therefore the interpreter must ensure that the DIM is always of an even length. Any DIM, with an odd argument will be stepped up to the next even argument (remember that the string starts from element 1;.
Neil Beattie
Kent

Assembler Is The Key

I have an AH version of the QL and I'm not sure whether there is a section missing in my manual, (or if it is possible) but how do you define the five-user defined keys?

I've tried using the method employed on the BBC and others but all I get is error messages.
P Howland
Hereford

The answer is quite simply that it is impossible to program the keys with a simple SuperBasic command. Assembly language is your only hope.

Duplicate Refit

Referring to the early hardware problems, my own QL is D05 ('AH'), but during the ROM refit very recently a sticker with D12 was put on. I noticed that Sinclair had put on a new keyboard and included a copy of the latest User Manual free of charge after I complained about the standard of the earlier one. I hope the above provides useful guidance for other readers with similar machines.

With regard to microdrive problems, the only ones I have experienced have been with Psion's software failing to load. Both my early Archive and later Quill-update tapes had to be replaced. This must be a fault with the Sinclair duplication methods as other software I have purchased and exchanged with other people has always behaved faultlessly. Apart from these Psion tape problems both (or all 3!) of my microdrives have performed exactly 100% from receipt in the middle of June, which is more than I can say for any floppy disk or Winchester units I have used. When people complain of problems they must be referring to the first QLs, all of which have been

upgraded – or are they perhaps on commission from disk interface manufacturers? I'm sure many others have had cause to return the Psion tapes so we're left to wonder why Sinclair still haven't recalibrated their duplication process as this is the only flaw to what I regard as a brilliant machine.
John Lawlor
Aberdeenshire

Soft Spot

There are so many things to say about the QL I hardly know where to start. For some years now I have worked as a freelance programmer on large IBM mainframes. I have always felt discouraged from buying a home computer, because however ingenious the little programs may be, trying to use them to do anything sensible was like being in a straitjacket compared with having the sophistication of monster system software at my disposal. The QL is the first affordable micro with power approaching that of the big machines.

Three important features combine to bring the QL to the frontiers of mainframe computing power: multitasking, the breaking of the 64K memory barrier, and the evident commitment on the part of several companies to producing heavy-duty software like compilers and assemblers which are designed to run in the background while the user can continue interactive work like editing (or playing games, I suppose). Although the big boys often have 24 or even 32-bit addressing (giving a theoretical limit of 4096 meg), most mainframe tasks individually use no more than 512K. Think about it.

Finally, I am very excited about the announcement of APL for the QL. I have a soft spot for APL as it was the first language I ever learned. I would urge any newcomer to programming to look at APL rather than BASIC. Anyone who has used a calculator will feel at home with elementary APL at once: you type '3' and the answer comes back: '3'. Enter '5+7' and you get '12' – no PRINT command is required. For arithmetic, APL uses the same symbols we all learned at school – eg '×' instead of '*' for multiplication.

I would strongly advise against buying the keyword version. No sensible manual would begin by introducing the novice to the wonders of matrix divide, for instance, and I don't remember being at all put out by the more exotic symbols. Your keyword APL examples I thought a bit of a mess! To offer the keyword version for separate sale would be scandalous, anyway, as each of the keywords could be written in ten seconds flat as a one-line function in standard APL, by someone who doesn't even know what the keywords do! As for displaying and printing the special symbols, MicroAPL would do well to provide some kind of plastic overlay for the keyboard, and perhaps also an APL character set for downloading to the most popular printers that allow user-defined characters.
Ian Ray
Cambridge

Now what's this about keyword APL? First, we would point out that standard APL is available for the die-hards anyway so your recommendation would seem a little strong.

Next, references to how easily matrix division may be performed using APL were supposed to illustrate the power of the language not to encourage innumerates to dispense with their times tables.

Finally if APL is to compete with BASIC it should be easy to pick up, simple to use and self-documenting. Standard APL with its exotic symbols, special keyboards or download character set fails to make the grade. Keyword APL with its standard character set and elements of plain English comes much closer. Whereas the former was designed to introduce matheticians to programming, the latter stands a far better chance of introducing laymen to both programming and mathematics.

Serial Hiccup

I have recently purchased a QL and first let me say that I am absolutely delighted but for a few minor hiccups. The reason I am writing is to give you and your readers a piece of information which may be of some benefit.

The first thing I did with my QL was to try to copy the Psion packages. However, I had considerable difficulty. I read in your Oct/Nov issue of problems with microdrives apparently running at different rates. Well after various tests I realised this was possibly what was wrong with mine. The most apparent symptom was the inability sometimes to get a directory on Drive 2 of a cartridge that had been formatted and written to on Drive 1.

I mentioned my suspicions to the shop where I bought the QL and was informed that they had had several similar complaints from owners of machines from a certain batch. The serial number of my machine started -02 and was replaced with no difficulty. Other users may have had similar problems but have not been given a replacement or not even tried it. It may be worth while checking the serial number and pursuing a complaint because this problem could cause severe irritation.

I have not included the shop's name just in case the information they gave me was inaccurate.

Finally, I would like to say that I consider the QL to be by far the best micro on the market and I am particularly pleased with the word processor. This letter was written on Quill, with a Taxan Kaga K910 printer.
Iain Begg
Sheffield

Question Time

You may be interested in the following point. Pressing CTRL and F5 will freeze/unfreeze any action on screen. This is useful in cheating at those rare games and for listing purposes.

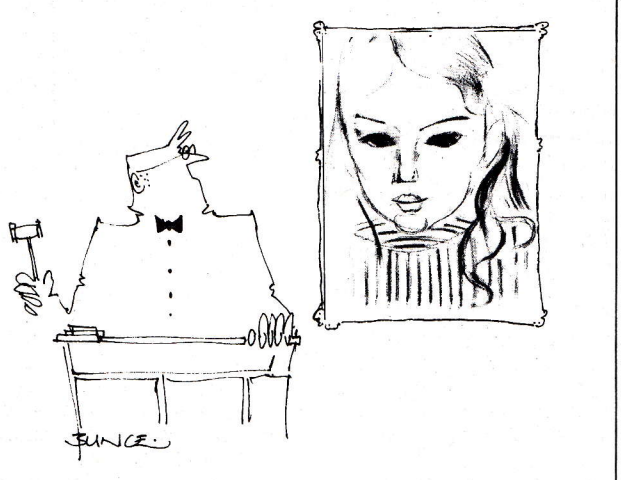
I would be grateful if you could answer the following queries:
1) What are the rates for reader's programs?
2) What is the best RS232 printer under £200?
3) Quill 2. Sinclair says only those users who joined QLUB will receive updates. Perhaps if enough users got together (through a user group) they could pressurise Sinclair to give a free update.
I successfully pressurised Acorn to give me a free update of my BBC O.S. 0.1, to an O.S. 1.2. Perhaps the magazine could help place pressure on Sinclair?
4) Does Sinclair Research ever answer the phone?
A Fullen
Durham

Our rates are calculated in relation to the number of columns per page and apply equally to articles and programs. They are as follows:
£100 3 Column full-text
£120 4 Column full-text

As regards an RS232 printer the Brother HR5 at £160 would be the ideal choice. However, if you shop around you should be able to pick up a Brother M-1009 for £200 which though it requires a printer interface affords a vast improvement in print quality and performance.

In response to your thoughts about pressurising Sinclair we wish you luck but feel that no precedent exists whereby the receiver of freebie software is by rights entitled free updates in perpetuity.

Finally, with BT now privatised we hope that communications will be improved to the point where Sinclair's telephones will be able to cope with the pressure from QL users.



What am I bid for this fine QL Master by Easel after Matisse!

QL

SKETCH PAD £14.90 ★
PARALLEL PRINTER INTERFACE £34.90 ★
JOYSTICK ADAPTOR £ 5.90 ★

SKETCH PAD

A graphics design package allowing the user to create high resolution full colour pictures and diagrams using the cursor keys or a joystick. Features include:-

- ★ Screen dump to any Epson compatible printer.
- ★ Microdrive save and load of screens.
- ★ Use of stipple gives a vast range of colours.
- ★ Graphics and text can be freely mixed.
- ★ Automatic drawing of many shapes.
- ★ Filling of regular or irregular shapes in any colour.
- ★ Drawings saved on microdrive can be redisplayed from within any Super-Basic program using simple procedures supplied on microdrive.

Some applications include:-

★ Labels ★ Signs ★ Flow Charts
★ Graphics for Games
★ Artists Drawing

****

SPECIAL OFFER

SKETCHPAD PLUS JOYSTICK ADAPTOR £19.95

****

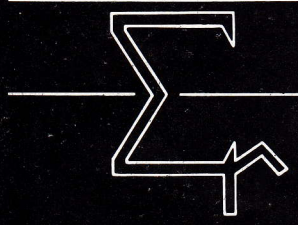
PARALLEL PRINTER INTERFACE

Simply plugs into the 'Ser 1' port of the QL and any centronics compatible printer (eg:- EPSON, CANON, OKI, Kaga, Juki etc.)
works with PSION software and requires no special commands. Full instructions included.

JOYSTICK ADAPTOR

Enables any Atari compatible joystick (eg: Kempston, Quickshot) to be used with the QL joystick ports.

All prices include V.A.T. and p & p within the U.K. Make cheques or postal orders payable to Sigma Research.



Sigma Research

231 Coldhams Lane, Cambridge CB1 3HY.

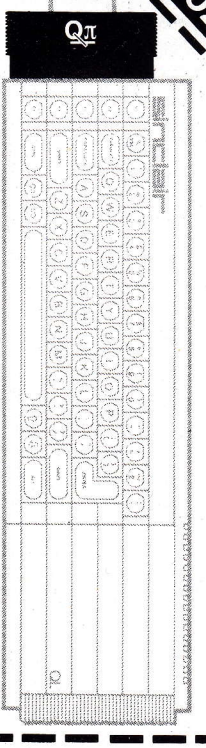
Centronics Parallel Interface for QL*

CST 0223 323302

CST announce the Q-PI Centronics Parallel Interface for Sinclair's QL – available NOW for only

Cambridge Systems Technology can provide the missing link for your QL. The CST Q-PI is a Centronics Parallel Interface offering full Q. DOS device driver software. It will connect your QL to the wide range of printers and plotters which use the standard Centronics interface, and is fully compatible with Sinclair-supplied PSION software.

£50 PLUS VAT



ORDER FORM

Please send_____ Q-PI units @ £86.25 incl VAT £_____
(and)_____ Centronics cables @ £11.50 each incl £_____
(add post & packing @ £2.25 incl per unit) £_____

CHEQUE/P.O. NO REMITTANCE TOTAL £

Name_____ Address_____

_____ Postcode_____

Phone_____ Date of Order_____

CUT AROUND BROKEN LINE AND DESPATCH WITH YOUR REMITTANCE TO:

CAMBRIDGE SYSTEMS TECHNOLOGY 30 Regent Street Cambridge CB2 1DB

QL/3/85

*QL IS A TRADEMARK OF SINCLAIR RESEARCH LTD.

MACHINE CODE TUTORIAL

This month Adam Denning goes backstage to look at number systems and processor architecture.

Why program in machine code? Why indeed, when the QL is already well-supported with excellent languages such as Pascal, BCPL and Lisp? Well, there are a number of reasons. Programs written in high level languages generally have to be compiled or interpreted, which means that a user will have to lay his hands on an appropriate program to do this. Furthermore, a compiled language needs a 'library' of routines to take care of all the machine specific tasks such as Input and Output, screen handling and so on. Additionally, an interpreter is a large program which must be in the machine along with the program which it is intended to interpret. Both these options therefore require a great deal of memory for what may be only simple tasks. Finally, a compiled language only starts to come into its own in terms of memory efficiency when the program being compiled is of a substantial length (ie another compiler) and interpreted language suffers from being extremely slow.

Machine code, then, is not just a compromise. Programs are inherently compact and extraordinarily fast being written, as it were, in the computer's native language. The drawback, however, is that machine code programming is often thoroughly tedious. There are a number of operations which may be performed in a single high level language instruction which will need a whole group of machine code instructions.

Machine code, as we've just said, is the machine's native language. This raises another problem. Different computers use different microprocessors. For example, the IBM PC uses the 8086, the BBC Micro uses the 6502, the Spectrum uses the Z80 and, of course, the QL uses the 68008. So, just as the internal architecture of the processors differs, their machine codes have little in common. Unlike a high level language, which remains approximately the same over many computers, a machine code is specific to a particular processor and only the most basic programming principles are common to more than one machine language.

Bin, Den or Hex

An understanding of number systems underlies all machine code programming and provides us with a useful starting point. *A number system* is quite simply the way in which we choose to count. Most of us tend to count in tens, presumably because (in line with EEC regulations) we have ten fingers. This is known as the denary system, or base ten. Computers, on the other hand, count in twos. This is called binary, or base two. They do this because a microprocessor is a very large collection of electronic switches, which can only ever be in one of two states – "on" or "off". As there are two states the binary system is a natural choice, but for us it's inconvenient to say the least. A binary number consists of a series of ones and noughts, each column being twice as significant as the one to its immediate right. For example, in denary, we talk of units, tens, hundreds, thousands and so on, and here each one is ten times more significant that its predecessor. Each column represents a power of ten, from 10^0 up to whatever we choose. For example,

12345 = (1★10^4) + (2★10^3) + (3★10^2) + (4★10^1) + (5★10^0)
= 10000 + 2000 + 300 + 40 + 5.

Binary is much the same, with ten being replaced by 2. So, decimal 103 in binary is 1100111. 1100111 = (1★2^6) + (1★2^5) + (0★2^4) + (0★2^3) + (1★2^2) + (1★2^1) + (1★2^0)
103 = 64 + 32 + 0 + 0 + 4 + 2 + 1.

The problem with binary is that as there are only ones and noughts in each number, it's fairly difficult to immediately distinguish between 1100111 and 11100111 (the first is 103 in decimal and the second is 231). These numbers are radically different, so it's easy to make mistakes.

An understanding of number systems underlies all machine code programming and provides a useful starting point.

And mistakes are fatal! Most programmers therefore choose a very sensible compromise, base 16. This is called hexadecimal, or more normally hex, and as 16 is a power of 2 things come together rather nicely. As 16 is 2^4, it follows that each group of four binary digits can be replaced by one hex digit to get the same effect. As there are 16 distinct digits in hex, numbers are obviously shorter to write down and mistakes are less likely. As hex needs to be able to represent numbers between 10 and 15 as single digits we use the letters A,B,C,D,E,F.

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Terminology

A machine code . . . a collection of binary numbers which a computer understands and can act upon directly.

Assembly language . . . is a system of mnemonics used by humans as the easiest way to represent machine code.

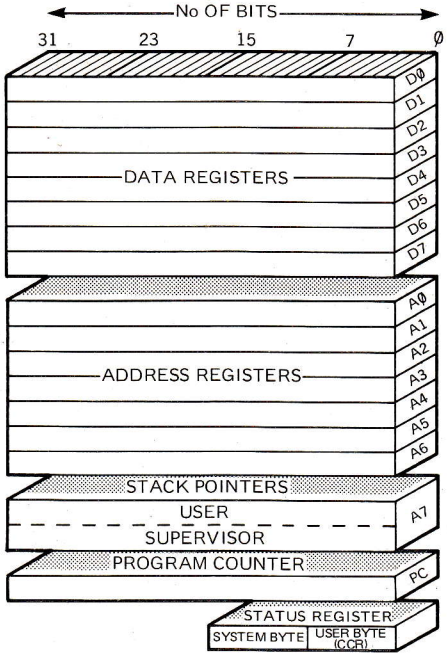
A mnemonic . . . a word or group of letters which form a more easily remembered representation of a machine code instruction.

A Source file . . . a file containing assembly language

An assembler . . . takes a source file and converts it into the true machine code, called the object code.

Object code is the code which the processor actually runs. Its appearance is almost entirely meaningless to us.

So the first principle of machine code programming is: LEARN HEX, LEARN IT WELL AND LEARN IT FAST!



Throughout the rest of the series, a hexadecimal number will be shown by being prefixed with a **'$'** sign, binary numbers will have a **'%'** prefix, and decimal numbers will be shown as normal. The reason for this is that the prefixes above are those recommended by Motorola, the makers of the 68008, for use in their and other assemblers.

68000 Blueprints

As machine code is processor specific, nothing can be done without a rudimentory knowledge of how the processor is laid out. the QL's 68008 is one of the Motorola 68000 family, so it has the same general architecture as the rest of the family. Examining figure 1 we see that it has a lot of things called registers. *A register* is an area of memory inside the processor which can be both examined and altered. Most of the 68000's are 32 bits long, but what does that mean?

We've just discovered that processors talk in binary, and that each digit represents a switch inside the processor which can be either on or off. It's conventional to describe off as binary 0 and on as binary 1, and each particular switch is known as a binary digit. This is generally abbreviated to bit, so a register which is 32 bits long can hold 32 binary digits. As each of these bits can notionally form part of a binary number, we know that a 32 bit register must be able to hold combinations of digits which can be taken to represent numbers from 0 to 32 binary digits long. If all 32 bits are set to ones then the number is %11111111111111111111111111111111. This means that one 32 bit register can hold any number between $0 and $FFFFFFFF, which is a gigantic range (over 4,000,000,000). Likewise, 16 bit registers can hold numbers between 0 and 65535 and 8 bit registers can hold numbers between 0 and 255. There's a relationship here: a register of length x can hold 2^x different numbers, and as the first number is always going to be zero, the range of numbers which can be held in a register is 2^x – 1.

There are instances when we are going to want a register to imagine that it's 32 bit, 16 bit or 8 bit, so these numbers are very useful to remember. Perhaps they're clearer in hex:

8 bit: $00 to $FF
16 bit: $0000 to $FFFF
32 bit: $00000000 to $FFFFFFFF

Moving on we notice that 68000 has eight data registers, D0 to D7, and seven address registers, A0 to A6. There isn't actually much difference except that data registers can be treated as anything from 1 to 32 bits long, depending on the application, while address registers can only be treated as being 16 or 32 bits long. There are also a few miscellaneous operations which can only be carried out on data registers. Address registers get their name because they are often used to hold memory addresses.

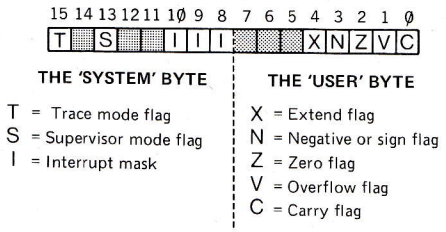
Flying the Flag

We then notice that there are two further address registers, both called A7. These are the stack pointers, one for each of the processor's modes. We'll discuss stacks and modes a little later. We find that the 68000 has a 16 bit status register, which is a very useful part of the processor. The status register contains numerous flags which tell us about the current state of the processor. One or more of these flags is altered automatically by the processor whenever certain

As machine code is processor-specific, it's important to understand how the QL's 68008 processor is laid out.

operations occur, so by examining the flags we can tell what happened.

The flags are shown in figure 2.



The Zero flag is set (turned on) whenever an operation gives a result of zero, and reset (turned off) otherwise.

The carry flag is used to indicate carry in additions and the borrow in subtractions.

The sign flag is tied up with two's complement arithmetic, which we haven't got to yet.

The extend flag is used when we want to perform operations in more than 32 bits. We hardly ever use that one. The other flags are not important just yet.

The last register is the *program counter*. This is used by the processor to point to the next instruction which it has to get from memory.

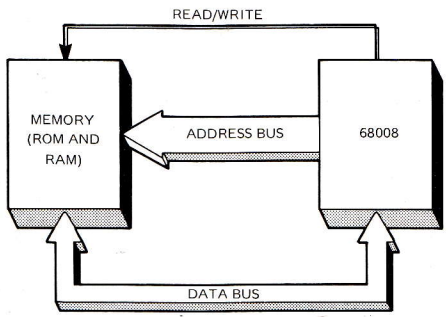
Memory Map

Now we can talk about memory organisation. A processor needs to follow its instructions to do anything at all, and on the QL many of these are held permanently in read only memory (ROM). This ROM contains the SuperBasic interpreter, the QDOS operating system and the device drivers for microdrives, serial lines, the local area network and so on.

ROM is a special form of memory, which as its name suggests can only be read from. (We are at liberty to attempt to write to it – POKE from Basic will do the trick – but it will have no effect.) This sort of memory does not lose its contents when the power is turned off. On the QL, ROM, extends from memory locations with addresses from $0000 to $C000, and there's space for a bit more.

The other sort of memory on the QL is RAM. This stands for Random Access Memory, but it's probably easier to understand if we call it read/write memory, as it can be written to and read from. This is where all our programs and data are stored. When the computer is switched off, all the data in the RAM is lost – forever unless it has been written to microdrive.

On a standard QL, RAM extends from memory locations with addresses from $20000 to $3FFFF, with that between $20000 and $27FFF mapping out the screen, immediately followed by an area known as the system variables. These are just memory locations used by the system to keep track of everything it is up to.



Now, when we execute a machine code program, which starts at address 'x', the processor puts the value of x into the program counter. This address is then put onto the address bus, and a wire from the 68000 called the read/write line is set to a particular value. Things are wired in such a way that when this happens, the memory location at address x puts its contents onto the data bus, which on the 68008 is 8 bits wide. The processor reads this, stores it, and collects the contents of the next memory location as well, as each 68000 instruction is always at least 16 bits long. The instruction is then executed, reading from and/or writing to memory if required. During all this the program counter is updated so that when this instruction has finished it will be pointing to the next.

While we're here, it's useful to know a bit of jargon. We've already talked about single bits, and it (almost!) follows from this that a collection of eight bits is called a byte. Further, two bytes are called a word and two words are called a long word. (Coincidentally, half a byte is called a nybble.)

Stacks

A fundamental concept in computer programming is that of the stack. This is an area of RAM which is used both by the system and by the programmer to temporarily hold items of information. The most common application is during the CALL and RETURN mechanism. Without going into details just yet, a call is equivalent to a Basic GOSUB and a return is equivalent to a Basic RETURN statement. The microprocessor uses call and return a lot. When it calls a subroutine, it needs to keep a record of where it came from before the call so that when it encounters the return (RTS) at the end of the subroutine it can resume where it left off prior to the call. This is where the stack comes in. At any one time, one of the A7 address registers is pointing to an area of memory designated by the programmer as suitable for a stack. When a call instruction is executed, the processor puts the address of the instruction after the call onto this stack and adjusts A7 to point to it, so that when the return comes along it just pops it off the stack, puts it in the program counter and carries on.

This obviously means that if we alter the stack pointer in between the call and the return, things will be well up the creek. It's amazing how many programs go wrong because of an unbalanced stack! It's perhaps odd to discover that the stack pointer is decreased each time an entry is put on it, so that it grows downwards in memory, but it's a very common technique and just about every other microprocessor does the same thing. One odd thing about the 68000 though is that its stack pointer always points to the current stack entry, while most other processors make it point to the next free location. The reason for the 68000's oddity is that it allows the programmer to use any of the address registers as a stack pointer, and various ways of manipulating the stack are available. You soon get used to it, and it's not good practice to play around with return addresses on the stack anyway.

All that we've discussed so far has been fairly easy to understand, but the 68000 is a powerful microprocessor and QDOS is a powerful operating system, so it's inevitable the pace will quicken and matters become more complex. Next month we'll look at the types of operations one can perform in machine code programs, and we'll discuss the 68000's modes and exceptions. Then we can start to program!

TERMINAL EMULATION

The QL as a universal terminal, and host to mini and mainframe computers, explored by Adam Denning using BCPL and machine code.

With built-in serial ports the QL seems an ideal candidate for use as a terminal to larger mainframes. Over the next few issues of *QL User* we will develop a full terminal emulation program which may be used to communicate with most machines provided they support an RS232 connector (or suitable interface).

The emulator is written largely in BCPL, but there are a few extensions to that language's library written in machine code. There is also a subsidiary clock which runs concurrently with the terminal emulator. As the program is compiled, it is as fast as could be desired and will of course happily multi-task. A word of caution, as the program occupies almost the entire QL screen area, any program running with it will have to make minimal use of the screen!

BCPL Plus

The first thing we have to discuss is how to add machine code routines to BCPL. As far as we know, Metacomco is the only company producing a BCPL compiler for the QL, so all the techniques developed will apply exclusively to this product.

Extensions to the library will include two general purpose routines, MOVE and BACKMOVE, along with more specialised functions and procedures to take advantage of facilities unique to the QL. One of these is called GETKEY, is now included as standard in the BCPL package and may be ignored.

The extensions are in four files, each about 130 bytes long. The first file *('mc1_lib')* contains these routines:

BAUD(Baudrate) . . . this sets the Baud rates of the serial ports.

anyinput := PENDING(channel-,timeout) . . . This checks the specified channel for input and returns TRUE if there is. If the channel being checked has reached end of file then the global variable RESULT2 is set to the QDOS error code **err.ef,** meaning 'end of file'. This routine is fundamental to the terminal. One of its uses is to automatically log the terminal on if it detects input on the specified serial port.

keypress := GETKEY(timeout). Very similar to INKEY, except that it returns 0 if no key was pressed in the specified timeout period and RESULT2 then holds the QDOS error code.

The second file *('mc2_lib')* holds routines for moving areas of memory:

MOVE(from,to,words) . . . This procedure moves **'words'** (BCPL words – four bytes) from the BCPL **from** address to the BCPL **to** address. A BCPL address is a long-word aligned quantity representing a valid machine address, shifted right twice (ie divided by 4) to make it a long word address as opposed to a byte address. This is because BCPL is designed around the concept of a *memory cell* which is of an arbitrary size; in the case of the QL this is 32 bits.

BACKMOVE(from,to,words). This is identical to MOVE except that memory is moved from the highest address down, making it useful for moving the contents of a vector up within the vector.

MOVEBYTE(frombyteaddress, tobyteaddress,bytes) . . . This is the same as MOVE except that the from and to addresses are real machine addresses ('byte addresses') and the number of units to move is specified in bytes.

BACKMVBY(frombyteaddress, tobyteaddress,bytes) . . . This is similar to MOVEBYTE except that, as with BACKMOVE, memory is moved from the highest address first. This is very useful for dealing with strings, as we shall see.

The third file *('mc3_lib')* puts sound into the hands of the BCPL programmer.

SOUND(soundvector) . . . Generates sounds as specified by the 8 words of **soundvector** (ie LET soundvector = VEC 7) formed in the following manner:

soundvector!0 := pitch 1 (low 8 bits)
soundvector!1 := pitch 2 (low 8 bits)
soundvector!2 := interval between steps (low 16 bits)
soundvector!3 := duration of sound (low 16 bits)
soundvector!4 := step between pitches (low 4 bits)
soundvector!5 := wrap (low 4 bits)
soundvector!6 := randomness (low 4 bits)
soundvector!7 := fuzziness (low 4 bits)

NOSOUND() . . . Kills any sound being generated.

The last file *('mc4_lib')* si the most complicated and one of the most useful. It contains routines for QDOS job control:

loadaddress := CREATEJOB (code length, datalength) . . . This function creates a job as defined by its two parameters. The BCPL address at the start of the area allocated is returned as the result and the job ID allocated by QDOS is in RESULT2. If the job could not be created then the result is zero and RESULT2 contains the QDOS error code. Note specifically that the result is a **'BCPL address',** making it ideal for READFILE and so on.

result := ACTIVATE(jobID, priority,timeout) . . . This activates the specified job at the specified priority (between 0 and 127), with a result of 0 indicating success and a negative error code being returned otherwise. The timeout is either 0, in which case the job is activated and control is returned to the invoking program, or –1 in which case the activated job takes over until its termination when the BCPL program takes control again. Other values for the timeout should not be used.

result := PRIORITY(jobID, priority) . . . This alters the priority of the specified job to the given value (between 0 and 127) and returns 0 if successful and a negative QDOS error code otherwise.

result :=KILLJOB(jobID) . . . This routine kills the specified job, returning 0 if successful and a negative QDOS error code if not. It uses the QDOS MT_FRJOB trap to ensure success!

result :=SUSPEND(jobID, timeout) . . . Suspends the specified job for the specified number of frames (20ms intervals) between 0 and 32767 (or –1 for infinite suspension). This function returns 0 if all went OK and a negative QDOS error code if not.

result :=RELEASE(jobID) . . . Releases a previously suspended job, returning zero if successful and a negative QDOS error code if not.

Routine Attachment

Moving on, to extend the BCPL language we must come to grips with a fundamental BCPL concept, the *global vector.* This is an area of RAM in which the addresses of procedures and functions and values of variables for use by a program can be held. If values are held in the global vector then the program can be compiled in sections and joined together at the end. The same technique obviously applies to adding extra library routines – if they are declared as global then any program can use them. The QL BCPL compiler includes the linker, which joins program sections together to form the end product – a program which can be EXECd. So to compile a program

which uses some of our library routines, we just type the name of the file containing the library routines into BLINK when required and everything is done for us.

So far so good, but these routines are written in machine code. How do we interface these to BCPL? By following the standard defined in the QL BCPL manual, we can write our routines in such a way that the linker will recognise them as BCPL routines. This requires the length of each module in BCPL words as the first long word of the file, and then the last few words contain the global information. Each individual routine within each module must of course, be aligned to a long word, as this corresponds to a BCPL address. All this is easy to achieve. The manual also tells us which registers we may use, which contain the routine parameters, and how to return a function result.

Let's look at the code for *mc1_lib* first.

Notice that the first bit of code in the file is a long word holding the length of the file divided by 4. This is followed by a CNOP 0,4 directive which ensures that the next instruction is situated on a long word boundary to satisfy BCPL. As with code produced by the BCPL compiler itself, each module must be position independant. Whenever BCPL calls a function or routine it passes the first three arguments in registers D1, D2 and D3. **BAUD** uses only one parameter, held in D1. By calling the QDOS MT_BAUD routine the Baud rate is set to the specified value. Notice that we do not check for errors. To return from a BCPL routine we must jump to the location held in A6 – JMP (A6).

GETKEY is a little more complicated. It is passed by one parameter – a timeout. This is transferred from D1 to D3 and then the QDOS channel ID of the current input stream is found by taking the address in the global variable CIS. A2 is set up to point to the base of the global vector for us, and the global number of CIS is its BCPL offset from (A2). By multiplying this by 4 to convert it into a byte address, we get the address of the stream control block in A1 – MOVE.L (CIS★4) (A2) ,A1. The first long word at (A1) contains the QDOS channel ID, so we load this into register A0. The registers are now set up ready for IO_FBYTE, so we execute that trap. This returns the next character read from the current input stream in D1 or an error code in D0

If D0 is not zero an error has occured, so we store the code in RESULT2 and clear D1 ready for return. Otherwise the character returned is also returned to BCPL after ANDing with $FF to clear the top three bytes of D1. Notice that A0 and A1 are saved on the stack for the duration of the routine, as BCPL needs their original values.

PENDING, again we save A0 and A1 and put the timeout (the second parameter) in D3. The BCPL stream identity (the first parameter) is put into A1 and the QDOS channel ID of that stream is obtained in A0 in much the same way as in GETKEY. The IO_PEND trap is then executed, which returns with no error if there is pending input within the given timeout and an error if there isn't. This error is usually 'Not complete', which we ignore by returning FALSE (0). If the error was 'End of file', though, we put this code into RESULT2 before returning.

The last few bytes of the file is the 'global list'. This consists of the global number of each routine within the

LISTING 1

```
LOC       OBJECT          STMT    SOURCE STATEMENT

                            1 * Three BCPL routines to add to the existing library.
                            2 * By Adam Denning (C) Copyright 1984 Adam Denning
                            3
                            4 * BAUD is passed one parameter and changes the serial line baud rate to the
                            5 * specified value. It is global 100
                            6
                            7 * GETKEY is passed one parameter (the timeout) and returns the character read
                            8 * from the current channel in that timeout. It returns 0 if no key was
                            9 * pressed, and RESULT2 will hold the error code. It is global 101
                           10
                           11 * PENDING is passed two parameters (a stream and a timeout) and returns 0
                           12 * (FALSE) if no input is pending on that channel, 1 (TRUE) if there is
                           13 * pending input. If EOF then err.ef in RESULT2. It is global 102
                           14
    =0064                  15 BAUD      EQU     100
    =0065                  16 GETKEY    EQU     101
    =0066                  17 PENDING   EQU     102
    =0034                  18 CIS       EQU     52
    =000A                  19 RESULT2   EQU     10
                           20
    =0000                  21 IO_PEND   EQU     0
    =0001                  22 IO_FBYTE  EQU     1
    =0012                  23 MT_BAUD   EQU     $12
    =FFFF                  24 ERR_NC    EQU     -1
    =FFF6                  25 ERR_EF    EQU     -10
                           26
0000'  0000 001F           27 FIRST     DC.L    (ENDMOD-FIRST)/4
0004'                      28
                           29           CNOP    0,4
0004'  7012                30
0006'  4E41                31 BAUDHERE  MOVEQ   #MT_BAUD,D0
0008'  4ED6                32           TRAP    #1
                           33           JMP     (A6)
000C'                      34
                           35           CNOP    0,4
000C'  48E7 00C0           36
0010'  2601                37 GKEYHERE  MOVEM.L AO/A1,-(A7)      preserve registers
0012'  226A 00D0           38           MOVE.L  D1,D3            put timeout in D3
0016'  2051                39           MOVE.L  (CIS*4)(A2),A1   get stream control block address
0018'  7001                40           MOVE.L  (A1),A0          ..and the channel ID
001A'  4E43                41           MOVEQ   #IO_FBYTE,D0     Read a byte
001C'  4A80                42           TRAP    #3
001E'  6706                43           TST.L   D0               Error?
0020'  2540 0028           44           BEQ.S   GOTKEY           No
0024'  7200                45           MOVE.L  D0,(RESULT2*4)(A2) Yes; save error code in RESULT2
0026'  0281 0000 00FF      46           MOVEQ   #0,D1            and clear return value
002C'  4CDF 0300           47 GOTKEY    ANDI.L  #$FF,D1          returning byte only
0030'  4ED6                48           MOVEM.L (A7)+,A0/A1      retrieve registers
                           49           JMP     (A6)             return
0034'                      50
                           51           CNOP    0,4
0034'  48E7 00C0           52
0038'  2602                53 PENDHERE  MOVEM.L AO/A1,-(A7)      Preserve registers
003A'  2241                54           MOVE.L  D2,D3            Put timeout into D3
003C'  2051                55           MOVEA.L D1,A1            Put FCB address in A1
003E'  7000                56           MOVEA.L (A1),A0          ...and channel ID in A0
0040'  4E43                57           MOVEQ   #IO_PEND,D0      check input stream
0042'  7201                58           TRAP    #3
0044'  4A80                59           MOVEQ   #1,D1
0046'  670C                60           TST.L   D0               get ready to return TRUE
0048'  7200                61           BEQ.S   YESPEND          ...unless an error occured
004A'  0C40 FFF6           62           MOVEQ   #0,D1            ...which it didn't here
004E'  6604                63           CMPI.W  #ERR_EF,D0       but return FALSE here
0050'  2540 0028           64           BNE.S   YESPEND          EOF error?
0054'  4CDF 0300           65           MOVE.L  D0,(RESULT2*4)(A2)
0058'  4ED6                66 YESPEND   MOVEM.L (A7)+,A0/A1      store error return
                           67           JMP     (A6)             retrieve registers
005C'                      68                                    and return
                           69           CNOP    0,4
005C'  0000 0000           70
0060'  0000 0064 0000 0004 71           DC.L    0
0068'  0000 0065 0000 000C 72           DC.L    BAUD,(BAUDHERE-FIRST)
0070'  0000 0066 0000 0034 73           DC.L    GETKEY,(GKEYHERE-FIRST)
0078'  0000 0034           74           DC.L    PENDING,(PENDHERE-FIRST)
                           75           DC.L    CIS
007C'                      76
                           77 ENDMOD    END
```

module as a long word, followed by the byte offset from the start of the module of each routine. The last entry is the number of the highest global referenced.

Once this has been assembled each routine can be incorporated into other programs by telling them the global numbers of the routines being used. The easiest way of doing this is by adding this to *libhdr*:

```
baud:100
getkey:101
pending:102
```

in the global declaration section, and altering the manifest constant **UG** to 103. As each of the files *mc1_lib* to *mc4_lib* is added, *libhdr* should be updated as appropriate.

Repeat Performance

Moving on to *mc2_lib*:
The routines in this file are short and simple, each has three parameters, the first two of which are addresses (either BCPL or machine) and the last is the number of units to move.

MOVE takes the two BCPL addresses (in D1 and D2) and shifts them left twice to convert them into machine addresses. If D3 (the third parameter) is zero then no memory needs to be moved, so the routine finishes straight away. Otherwise 1 is subtracted from D3 to make it suitable for a DBRA loop (which iterates until the spcified data register equals –1) and the from and to addresses are transferred to addresses registers. A simple loop is then begun in which the specified number of long words are transferred.

BACKMOVE is much the same except that D3 is added to the from and to addresses first and the pre-

LISTING 2

```
LOC       OBJECT          STMT    SOURCE STATEMENT

                            1 * Four BCPL routines to add to the existing library.
                            2 * By Adam Denning (C) Copyright 1984 Adam Denning
                            3
                            4 * MOVE(from,to,words) as in Richards. Global 103
                            5 * BACKMOVE(from,to,words) as above.   Global 104
                            6 * MOVEBYTE(frombyteaddress,tobyteaddress,bytes) as above. Global 105
                            7 * BACKMVBY(frombyteaddress,tobyteaddress,bytes) as above. Global 106
                            8
    =0067                   9 MOVE_WD   EQU     103
    =0068                  10 BACKMOVE  EQU     104
    =0069                  11 MOVEBYTE  EQU     105
    =006A                  12 BACKMVBY  EQU     106
                           13
0000'  0000 0023           14 FIRST    DC.L    (ENDMOD-FIRST)/4
                           15
0004'                      16          CNOP    0,4
                           17
0004'  E589                18 MOVEHERE LSL.L   #2,D1
0006'  E58A                19          LSL.L   #2,D2
0008'  4A83                20          TST.L   D3
000A'  670C                21          BEQ.S   MOVEOUT
000C'  5383                22          SUBQ.L  #1,D3
000E'  2641                23          MOVEA.L D1,A3
0010'  2842                24          MOVEA.L D2,A4
0012'  28DB                25 MOVELOOP MOVE.L  (A3)+,(A4)+
0014'  51CB FFFC           26          DBRA    D3,MOVELOOP
0018'  4ED6                27 MOVEOUT  JMP     (A6)
                           28
001C'                      29          CNOP    0,4
                           30
001C'  E589                31 BACKHERE LSL.L   #2,D1
001E'  E58A                32          LSL.L   #2,D2
0020'  4A83                33          TST.L   D3
0022'  6710                34          BEQ.S   BACKOUT
0024'  D283                35          ADD.L   D3,D1
0026'  D483                36          ADD.L   D3,D2
0028'  5383                37          SUBQ.L  #1,D3
002A'  2641                38          MOVEA.L D1,A3
002C'  2842                39          MOVEA.L D2,A4
002E'  2923                40 BACKLOOP MOVE.L  -(A3),-(A4)
0030'  51CB FFFC           41          DBRA    D3,BACKLOOP
0034'  4ED6                42 BACKOUT  JMP     (A6)
                           43
0038'                      44          CNOP    0,4
                           45
0038'  4A83                46 BYTEHERE TST.L   D3
003A'  670C                47          BEQ.S   BYTEOUT
003C'  5383                48          SUBQ.L  #1,D3
003E'  2641                49          MOVEA.L D1,A3
0040'  2842                50          MOVEA.L D2,A4
0042'  18DB                51 BYTELOOP MOVE.B  (A3)+,(A4)+
0044'  51CB FFFC           52          DBRA    D3,BYTELOOP
0048'  4ED6                53 BYTEOUT  JMP     (A6)
                           54
004C'                      55          CNOP    0,4
                           56
004C'  4A83                57 BMBYHERE TST.L   D3
004E'  6710                58          BEQ.S   OUTBYTE
0050'  D283                59          ADD.L   D3,D1
0052'  D483                60          ADD.L   D3,D2
0054'  5383                61          SUBQ.L  #1,D3
0056'  2641                62          MOVEA.L D1,A3
0058'  2842                63          MOVEA.L D2,A4
005A'  1923                64 LOOPBYTE MOVE.B  -(A3),-(A4)
005C'  51CB FFFC           65          DBRA    D3,LOOPBYTE
0060'  4ED6                66 OUTBYTE  JMP     (A6)
                           67
0064'                      68          CNOP    0,4
                           69
0064'  0000 0000           70          DC.L    0
0068'  0000 0067 0000 0004 71          DC.L    MOVE_WD,(MOVEHERE-FIRST)
0070'  0000 0068 0000 001C 72          DC.L    BACKMOVE,(BACKHERE-FIRST)
0078'  0000 0069 0000 0038 73          DC.L    MOVEBYTE,(BYTEHERE-FIRST)
0080'  0000 006A 0000 004C 74          DC.L    BACKMVBY,(BMBYHERE-FIRST)
0088'  0000 0000           75          DC.L    0
                           76
008C'                      77 ENDMOD   END
```

LISTING 3

```
LOC       OBJECT          STMT    SOURCE STATEMENT

                            1 * Two BCPL routines to add to the existing library.
                            2 * By Adam Denning (C) Copyright 1984 Adam Denning
                            3
                            4 * SOUND(soundvector) generates a specified sound. Global 107
                            5 * NOSOUND() kills any sound currently being made. Global 108
                            6
    =006B                   7 SOUND     EQU     107
    =006C                   8 NOSOUND   EQU     108
                            9
    =0011                  10 MT_IPCOM  EQU     $11
                           11
0000'  0000 001E           12 FIRST    DC.L    (ENDMOD-FIRST)/4
                           13
0004'                      14          CNOP    0,4
                           15
0004'  E589                16 SND_HERE LSL.L   #2,D1           get vector byte address
0006'  2641                17          MOVEA.L D1,A3          into A3
0008'  176B 0003 0006      18          MOVE.B  3(A3),6(A3)    move pitch1 into right place
000E'  376B 000A 0008      19          MOVE.W  10(A3),8(A3)   move interval into right place
0014'  376B 000E 000A      20          MOVE.W  14(A3),10(A3)  move duration into right place
001A'  122B 0013           21          MOVE.B  19(A3),D1      get step
001E'  E909                22          LSL.B   #4,D1          into high nybble of D1.B
0020'  822B 0017           23          OR.B    23(A3),D1      and wrap into low nybble
0024'  1741 000C           24          MOVE.B  D1,12(A3)      store in right place
0028'  122B 001B           25          MOVE.B  27(A3),D1      get randomness
002C'  E909                26          LSL.B   #4,D1          into high nybble of D1.B
002E'  822B 001F           27          OR.B    31(A3),D1      and fuzz into low nybble
0032'  1741 000D           28          MOVE.B  D1,13(A3)      store in right place
0036'  377C 0100 000E      29          MOVE.W  #$100,14(A3)   store 'no reply' command
003C'  36BC 0A08           30          MOVE.W  #$A08,(A3)     store 'sound' and '8 parameter'
0040'  426B 0002           31          CLR.W   2(A3)          commands
0044'  377C AAAA 0004      32          MOVE.W  #$AAAA,4(A3)   all B read as whole bytes
004A'  7011                33          MOVEQ   #MT_IPCOM,D0   send command
004C'  4E41                34          TRAP    #1
004E'  4ED6                35          JMP     (A6)
                           36
0050'                      37          CNOP    0,4
                           38
0050'  47FA 0008           39 KILLHERE LEA.L   K_SOUND,A3     kill sound
0054'  7011                40          MOVEQ   #MT_IPCOM,D0
0056'  4E41                41          TRAP    #1
0058'  4ED6                42          JMP     (A6)
                           43
005A'  0B                  44 K_SOUND  DC.B    $B             'kill sound' command
005B'  00                  45          DC.B    0              'no parameters'
005C'  01                  46          DC.B    1              'no reply'
                           47
0060'                      48          CNOP    0,4
                           49
0060'  0000 0000           50          DC.L    0
0064'  0000 006B 0000 0004 51          DC.L    SOUND,(SND_HERE-FIRST)
006C'  0000 006C 0000 0050 52          DC.L    NOSOUND,(KILLHERE-FIRST)
0074'  0000 0000           53          DC.L    0
                           54
```

LISTING 4

```
LOC     OBJECT          STMT    SOURCE STATEMENT

                        1 * Six BCPL routines to add to the existing library.
                        2 * By Adam Denning (C) Copyright 1984 Adam Denning
                        3 .
                        4 * loadaddress := CREATEJOB(codelength,datalength) returns the job ID in
                        5 * RESULT2. In case of error, the result is zero and the error code is in
                        6 * RESULT2. Global 109
                        7 * issuccessful := ACTIVATE(jobID,priority,timeout) returns zero if OK or
                        8 * the QDOS error code. Global 110
                        9 * issuccessful := PRIORITY(jobID,priority) returns zero if OK or the
                       10 * QDOS error code. Global 111
                       11 * issuccessful := KILLJOB(jobID) returns zero if OK or the QDOS error
                       12 * code. Global 112
                       13 * issuccessful := SUSPEND(jobID,timeout) returns zero if OK or the QDOS
                       14 * error code. Global 113
                       15 * issuccessful := RELEASE(jobID) returns zero if OK or the QDOS error
                       16 * code. Global 114
                       17
  =000A                18 RESULT2   EQU     10
  =006D                19 CREATEJOB EQU     109
  =006E                20 ACTIVATE  EQU     110
  =006F                21 PRIORITY  EQU     111
  =0070                22 KILLJOB   EQU     112
  =0071                23 SUSPEND   EQU     113
  =0072                24 RELEASE   EQU     114
                       25
  =0001                26 MT_CJOB   EQU     1
  =0005                27 MT_FRJOB  EQU     5
  =0008                28 MT_SUSJB  EQU     8
  =0009                29 MT_RELJB  EQU     9
  =000A                30 MT_ACTIV  EQU     10
  =000B                31 MT_PRIOR  EQU     11
                       32
0000' 0000 002C        33 FIRST     DC.L    (ENDMOD-FIRST)/4
                       34
0004'                  35           CNOP    0,4
                       36
0004' 2602             37 CRTJHERE  MOVE.L  D2,D3
0006' 2401             38           MOVE.L  D1,D2
0008' 72FF             39           MOVEQ   #-1,D1
000A' 48E7 00C0        40           MOVEM.L A0/A1,-(A7)
000E' 93C9             41           SUBA.L  A1,A1
0010' 7001             42           MOVEQ   #MT_CJOB,D0
0012' 4E41             43           TRAP    #1
0014' 4A80             44           TST.L   D0
0016' 6708             45           BEQ.S   JOB_OK
0018' 2540 0028        46           MOVE.L  D0,(RESULT2*4)(A2)
001C' 7200             47           MOVEQ   #0,D1
001E' 6008             48           BRA.S   BAD_JOB
0020' 2541 0028        49 JOB_OK    MOVE.L  D1,(RESULT2*4)(A2)
0024' 2208             50           MOVE.L  A0,D1
0026' E489             51           LSR.L   #2,D1
0028' 4CDF 0300        52 BAD_JOB   MOVEM.L (A7)+,A0/A1
002C' 4ED6             53           JMP     (A6)
                       54
0030'                  55           CNOP    0,4
                       56
0030' 700A             57 ACTIHERE  MOVEQ   #MT_ACTIV,D0
0032' 4E41             58           TRAP    #1
0034' 2200             59           MOVE.L  D0,D1
0036' 91C8             60           SUBA.L  A0,A0
0038' 4ED6             61           JMP     (A6)
                       62
003C'                  63           CNOP    0,4
                       64
003C' 700B             65 PRIOHERE  MOVEQ   #MT_PRIOR,D0
003E' 4E41             66           TRAP    #1
0040' 2200             67           MOVE.L  D0,D1
0042' 91C8             68           SUBA.L  A0,A0
0044' 4ED6             69           JMP     (A6)
                       70
0048'                  71           CNOP    0,4
                       72
0048' 48E7 00E0        73 KILLHERE  MOVEM.L A0-A2,-(A7)
004C' 7005             74           MOVEQ   #MT_FRJOB,D0
004E' 4E41             75           TRAP    #1
0050' 2200             76           MOVE.L  D0,D1
0052' 4CDF 0700        77           MOVEM.L (A7)+,A0-A2
0056' 4ED6             78           JMP     (A6)
                       79
0058'                  80           CNOP    0,4
                       81
0058' 2602             82 SUSPHERE  MOVE.L  D2,D3
005A' 48E7 00C0        83           MOVEM.L A0/A1,-(A7)
005E' 93C9             84           SUBA.L  A1,A1
0060' 7008             85           MOVEQ   #MT_SUSJB,D0
0062' 4E41             86           TRAP    #1
0064' 2200             87           MOVE.L  D0,D1
0066' 4CDF 0300        88           MOVEM.L (A7)+,A0/A1
006A' 4ED6             89           JMP     (A6)
                       90
006C'                  91           CNOP    0,4
                       92
006C' 7009             93 RELJHERE  MOVEQ   #MT_RELJB,D0
006E' 4E41             94           TRAP    #1
0070' 2200             95           MOVE.L  D0,D1
0072' 91C8             96           SUBA.L  A0,A0
0074' 4ED6             97           JMP     (A6)
                       98
0078'                  99           CNOP    0,4
                      100
0078' 0000 0000       101           DC.L    0
007C' 0000 006D 0000 0004 102       DC.L    CREATEJOB,(CRTJHERE-FIRST)
0084' 0000 006E 0000 0030 103       DC.L    ACTIVATE,(ACTIHERE-FIRST)
008C' 0000 006F 0000 003C 104       DC.L    PRIORITY,(PRIOHERE-FIRST)
0094' 0000 0070 0000 0048 105       DC.L    KILLJOB,(KILLHERE-FIRST)
009C' 0000 0071 0000 0058 106       DC.L    SUSPEND,(SUSPHERE-FIRST)
00A4' 0000 0072 0000 006C 107       DC.L    RELEASE,(RELJHERE-FIRST)
00AC' 0000 000A       108           DC.L    RESULT2
                      109
00B0'                 110 ENDMOD    END
```

decrement mode is used to transfer the memory from the highest memory address down to the lowest.

Naturally enough, **MOVEBYTE** and **BACKMVBY** follow the same pattern with the difference that the parameters passed need no conversion to machine addresses first, and bytes rather than long words are moved. These four routines are globals 103 to 106, so *libhdr* must be altered to accomodate them.

The third file is also fairly short. The only parameter to **SOUND** is the BCPL address of the vector containing the sound definition. This must be converted to a byte address, by shifting it left twice, and then the information contained in the vector must be moved about to put it into the right form for the MT_IPCOM QDOS trap, which sends all the information to the QL's 8049 processor. All the MOVEs, CLRs and ORs take each individual parameter and move them into the requisite places, along with the 8049 commands:

$0100 tells it to return no reply

$0A is the 'generate sound' command

$08 is the number of parameters being sent

$AAAA tells the 8049 to treat each parameter as two nibbles each.

It sounds complicated but is actually only tedious! **NOSOUND** has no parameters and tells the 8049 (using MT_IPCOM again) to turn all sounds off. This requires the following commands to be sent to the 8049:

$B 'kill sound' command
0 'no parameters'
1 and 'no reply'

These two routines are global 107 and 108, so change *libhdr* as soon as you've assembled the file.

The final file is *mc4_lib*, shown in listing 4:

The six routines here are globals 109 to 114, so the manifest declaration of UG in *libhdr* should now be 115. The routines are relatively self-explanatory as each uses the requisite QDOS trap routine:

CREATEJOB uses MT_CJOB which needs the length of code in the job in D2, the length of its data space in D3 and, unless the job is ROM-resident, A1 should be zero.

ACTIVATE uses MT_ACTIV, which needs the parameters in just the form they're sent by BCPL.

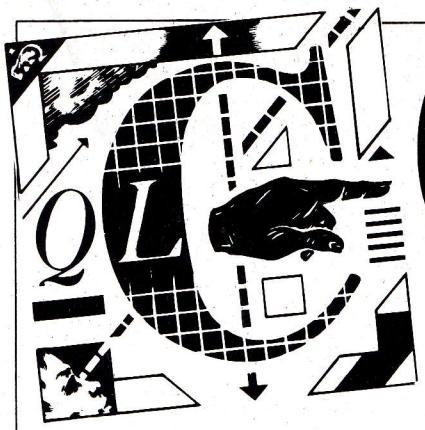
PRIORITY uses MT_PRIOR, which again needs the parameters in the form that they're sent by BCPL.

KILLJOB uses MT_FRJOB which corrupts A0, A1 and A2, so we must save them first.

SUSPEND uses MT_SUSJB, which needs the timeout in D3 and 0 in A1 as there is no 'release flag' for the job we're suspending.

RELEASE uses MT_RELJB

With these added routines the QL BCPL compiler is capable of just about anything, and next month we'll start writing the terminal emulator program proper.



C SERIES

Continuing our look at this elegant language, Peter Rodwell discusses its handling of strings, arrays and simple I/O facilities.

If you're familiar with a high-level language like BASIC, you'll probably be surprised to learn that C contains no provision for handling strings of characters.

At first this may seem a serious limitation. C is, however, not quite a high-level language, as we have seen, and although the standard library of functions does contain some string-handling mechanisms, for many of the applications for which C is best suited, you may need to write your own functions anyway.

Similarly, BASIC hackers are often surprised to learn that C has no I/O facilities whatsoever, in that no commands for I/O exist in the C language itself. Again, though, the standard library of functions which come with every worthwhile C compiler includes a wide range of I/O mechanisms – or you can write your own if you really want or need to.

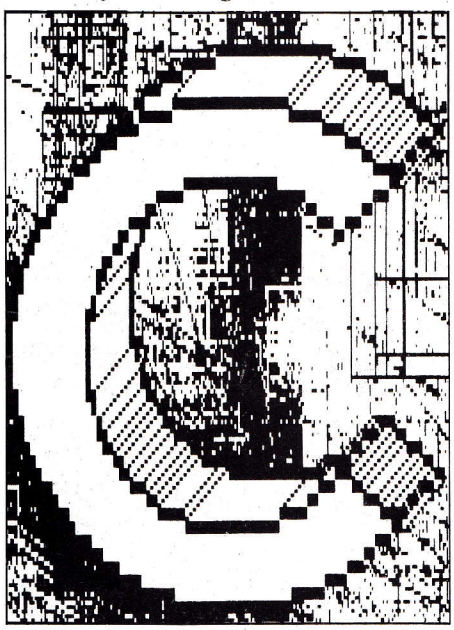
The lack of built-in string handling and I/O can be regarded either as providing increased versatility or requiring extra programming effort, depending on the application you are writing.

In C, strings are in fact arrays of characters, and can be treated just as any other type of array. Consider, for instance, a simple program which allows you to type in a line of text at the keyboard and will then print it on the screen. We can use two of the standard I/O functions for this – **getchar()**, which accepts a single character from the keyboard, and **printf()**, which displays text on the screen. The program could look something like this:

```
main()
{
char      c, buffer[128];
int       x;

x = 0;   c = 0;
while ( x < 128 && c ! = 13)
   {
   c = getchar ();
   if ( c ! = 13 )   buffer[x] = c;
   ++x;
   }
printf ("%s",buffer);
exit (0);
}
```

First, notice the declaration **buffer[128].** This is how we describe an array in C and it's not very different from BASIC – we have to say what type of object is to be stored (done with the char type declaration), we give it a name ('buffer') and we must say how large it is so that the



compiler can reserve the required amount of memory for it; this is done by enclosing the required number of characters in square brackets.

You address the individual elements within an array just as you would in BASIC, by putting the element number within the square brackets. But you must remember that in C, arrays start with element 0, so our array buffer contains 128 elements, numbered from 0 to 127. So, **buffer[0]** is the first element, **buffer[1]** the second and so on. Of course, you can put a variable within the square brackets instead, which is what we have done here – the variable **x** is used, and this is set to zero at the start of the program to guarantee that we start off by addressing the first element in buffer.

As an aside, the compiler is supposed to set a variable to zero when it is declared. Thus, the declaration,

```
int       x;
```

should create the variable **x** and set it to zero automatically. However, one or two compilers don't do this and so I make a habit of always setting variables to a known value just to be safe.

To collect characters from the keyboard and place them in the buffer, we use a **while** loop. Naturally we must impose some method of breaking out of the loop and this is done with two conditions: we carry on accepting characters until **x** equals 128, meaning we have filled the buffer completely, or until the character typed in is a carriage return, ASCII 13. As a precaution, we have also set **c** to zero before entering the loop, thus ensuring that the loop will be executed at least once.

Inside the loop, we use the input function **getchar()** to obtain a single character, and assign that character to the variable **c,** with a statement,

```
c = getchar();
```

The next line checks that **c** isn't a carriage return and if it isn't, places it in the next element of buffer, using the variable **x** as an index. **x** is then incremented and we loop back for the next character.

Once a carriage return has been typed or we have filled the buffer, the program drops out of the loop and we print out the contents of the buffer, using the standard function **printf().**

Fine Print

This is a powerful function which required some explanation. At its simplest, it required just one argument, a string to be printed:

```
printf("Hello");
```

Suppose we used exactly this statement several times throughout a long program and we later realised that "Good morning" would be a more appropriate message; we would have to work our way through the entire program changing "Hello" to "Good morning", and checking carefully that we had found each occurrence of "Hello". An easier method would be to declare "Hello" as a string right at the start of the program:

```
char      greet[] = ("Hello");
```

This sets up a character array called

greet and puts "Hello" into it. We don't need to say how large **greet** must be in this case as the compiler will work it out for us. In fact **greet** will contain *six elements for "Hello"*, as in C strings are terminated with an additional character, set to zero. This allows string-handling functions such as **printf()** to recognise the end of a string.

This method also saves memory, as "Hello" is stored only once in the program. And of course if we want to change "Hello" to "Good morning" we have only one string to alter. All we need now is a way of telling **printf()** what to print each time we want "Hello" to appear.

Spoilt For Choice

We do this by giving **printf()** a *format string* and the name of the string to be printed. In our example here, the format string is **"%s"**, which tells **printf()** that it has to print a string, and this is followed by the name of the string to be printed, **buffer**. There are various symbols which we can put in the format string including

%d to print an integer,
%f to print a floating point number,
%c to print a single character, and
\n to print a carriage return and linefeed sequence.

For instance, we could change the printout statement in our program to move to a new line before printing the contents of **buffer** with:

```
printf("\n%s", buffer);
```

The program ends with the command **exit(0)** which causes it to terminate tidily and return control to the operating system. **exit** in fact allows you to return a code to the operating system; by convention, zero signifies that the program terminated correctly and any other value signifies that some error condition occurred. Not all operating systems do anything with the returned value, however.

Tightening Up

If you have been following this series, you will be aware that one of the glories of C is the way in which code can be tightened up with all sorts of tricks. The example program above is quite loose and we can easily smarten it up; the **while** loop, for instance can be re-written as:

```
while ( x < 128 && (c=getchar())!= 13 ) buffer [x++] = c;
```

What we have done is to move **getchar()** into the condition so that we accept a character and assign it to **c** before testing to see whether it's a carriage return. Note that we have to put brackets around **c=getchar()** to ensure that the test for carriage return is performed on the character typed in at the keyboard and placed in **c**. Then we have done away with the separate statement which increments **x** simply by carrying out the incrementing when we use **x** to place the character in the buffer.

Note, though, that the increment operator (++) comes *after* we refer to **x**; if we used **++x** instead, the effect would be to increment **x** before placing the character in the array. So, on our first trip through the loop, **x** would start off as zero. On reaching **buffer[++x] =c;** it would be incremented to 1, and **c** would be placed in **buffer[1]** instead of **buffer[0]**, which would always remain empty. The effect of this would become apparent with **printf()** tries to display the contents of **buffer;** it considers a zero character to mark the end of a string and that's what it would find in the first element of **buffer** – so it would stop printing and the string you had typed would never appear on the screen!

To get really fancy, we can modify the **while** loop still further to:

```
while ( x < 128 && (buffer[x++]= getchar()) != 13) ;
```

This does away with the variable **c** altogether and puts the incoming character straight into **buffer** before testing it. All the work is done within the **while** statement and the loop itself consists of just a dummy line, the **;** at the end!

But there is one subtle difference here. Because we place the character straight into **buffer** *before* testing it, this means that the carriage return also goes into the buffer. Previously, the test was performed before assigning the character to the array and so the carriage return was not stored along with the other characters. Clearly, the method you choose depends on the application, but you probably won't want to store the carriage return – it's purpose is purely to break out of the input loop and isn't needed in the array. The empty element after the last character typed in serves to mark the end of the string as far as the standard C functions like **printf()** are concerned. This is a convention that is well worth making an effort to keep to in your C programming.

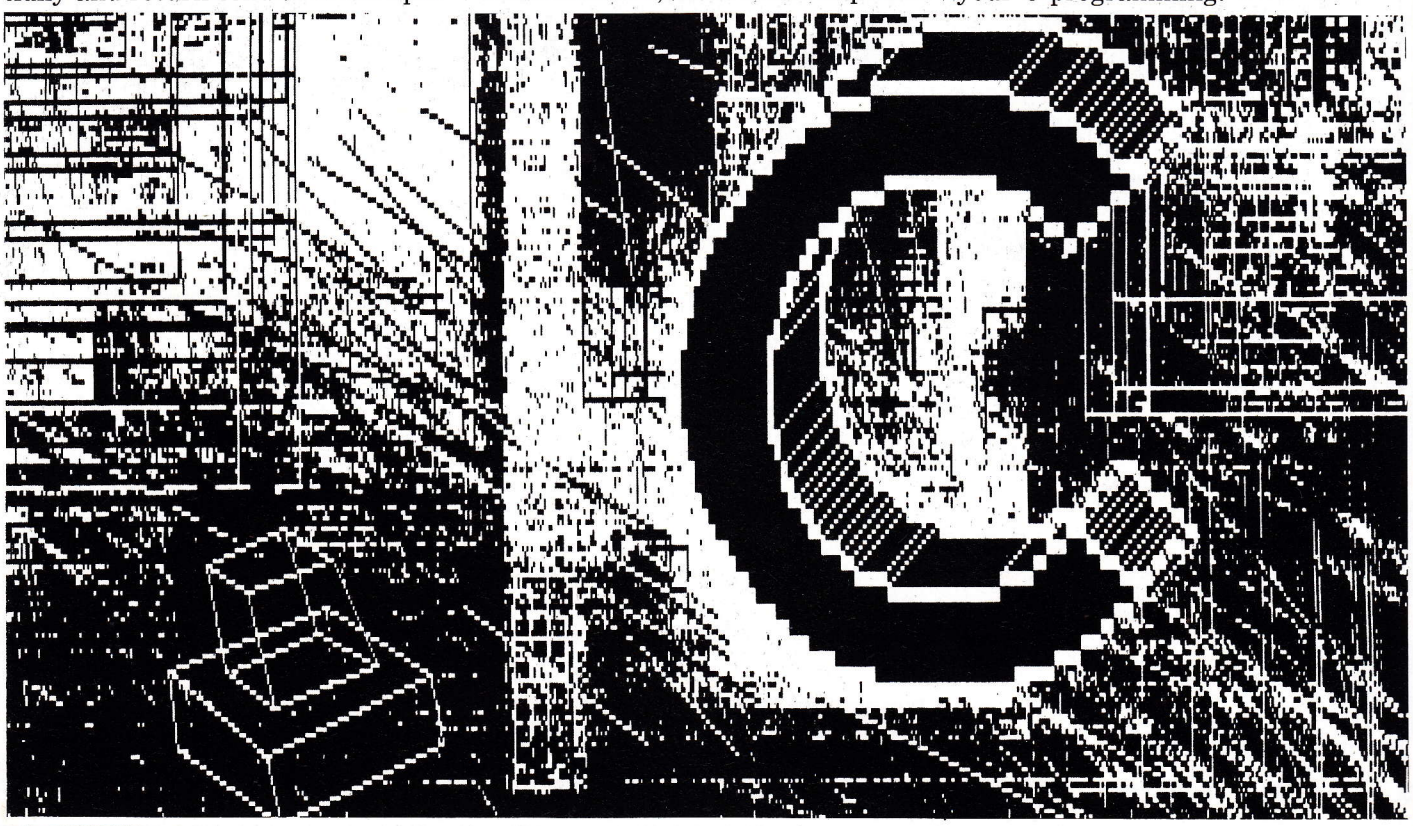


ILLUSTRATION BY GEORGE SNOW

★ ★ SPECIAL OFFER ★ ★

EMAP computer publications have organised a sponsored mass parachute jump in aid of the Ethiopian famine appeal. Weather permitting this should have already taken place by the time this issue reaches the newstand. MICROVITEC have kindly agreed to donate one of their CUB medium resolution colour monitors* to QL User, provided I complete the jump.

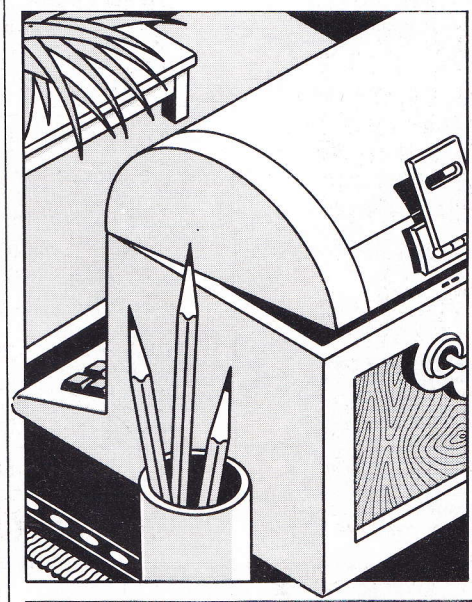
This monitor is now being offered at a greatly reduced price of £200 (as opposed to £275) to the first reader of QL User to phone 01-251 6222 extension 2463 after 9.30am on Monday 25th February.
The sum received for this monitor will be paid directly to the famine relief fund.

Phil Baker

* (See inside front cover)

Secret Locations

Deep down in the QL's memory are places where QDOS hides its secrets. Alan Turnbull investigates whilst our technical team POKE about.



When formatted, a cartridge is split up into a number of sectors each containing 512 bytes. When a file is saved, the information in it is stored in numbered blocks occupying a sector apiece. The sectors in which a file is stored need not necessarily come one after the other but may well be spread out across the length of the tape. How then does the QL know which sector to go to in order to fetch the next block in a file?

The secret lies in a table floating around in the QL's memory known as the microdrive map. The table is arranged in sector number order from 0 to 254. Each sector can correspond to a particular block within a file. Files are numbered in the order found in a normal directory listing. Certain file numbers, however, have a special significance:

File No	
0	Actual cartridge directory
248	Special Sector Map
253	Free Data Sectors
254–255	Faulty Sectors

The relevant System Variable for accessing this information is SV_FSDEF whose address is $28100 (hex) or in decimal 164096. Here QDOS stores a pointer (ie, a further address) to the base of a block of memory in which the cartridge's name is stored over ten bytes starting from $14 (22 decimal) and then for each sector, one byte containing a file number and another the block number.

The following SuperBasic program demonstrates how to get at this information. First a normal directory (line 130) is produced and then a loop is set up (lines 300-490) which extracts a file number (line 310) and block number (line 320) for each sector on the table.

NB. Replace '£' sign with '#' on the listing opposite.

PEEKs & POKEs

Pressing CTRL and F5 simultaneously will freeze a screen until a key is pressed. This effect may be simulated in a program by the following POKEs.

POKE 163891,1 . . . to freeze screen
POKE 163891,0 . . . to activate

Knowing whether the TV or Monitor mode has been selected at the beginning of a session on the QL is particularly useful when it comes to designing screens. To find out which has been selected simply:

PRINT PEEK (163890)

If zero, then Monitor mode has been selected.

The following POKEs allow you to set upper and lower case from within a program.

POKE_W 163976,255 . . . turn on CAPS LOCK.
POKE_W 163976,0 . . . turn off.

If you feel that your QL is a little too slow in repeating a character when you have your finger on its key, then these two POKEs should interest you.

POKE_W 163980,n . . . alters the delay before the character starts repeating
POKE_W 163982,n . . . alters the speed at which the character repeats.

n represents tenths of second. Default values are 30 and 2 respectively. Incidentally, setting both to 0 is a particularly good trick to play if you are feeling mischievious as it makes the entry of any commands near impossible.

To find out which microdrive is currently whirring enter:

PRINT PEEK (164078)

Finally, unlikely though it may be, if you have managed to set up a network of QLs then to discover your station number simply type:

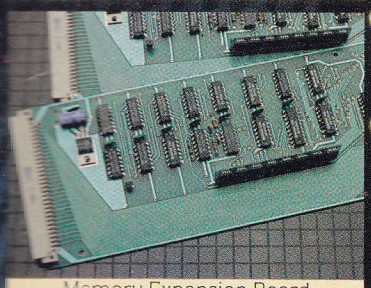
PRINT PEEK (164895)

```
100 REMark QL Microdrive Profile Reporter Program
110 REMark     (c) August 1984, Alan Turnbull
120 :
130 DIR £0;mdv1_
140 MODE 512
150 CLS
160 LET microdrive_physical_layer_address=PEEK_L(164096)
170 LET map_address=microdrive_physical_layer_address+40
180 LET medium_name_address=microdrive_physical_layer_address+22
190 LET medium_name$=""
200 FOR character=1 TO 10
210   LET medium_name$=medium_name$ & CHR$(PEEK(medium_name_address+character-1)
)
220 END FOR character
230 PRINT "QL MICRODRIVE CARTRIDGE PROFILE FOR: ";medium_name$
240 PRINT
250 LET number_of_free_sectors=0
260 LET number_of_bad_sectors=0
270 PRINT "SECTOR  FILE   BLOCK    SECTOR  FILE   BLOCK    SECTOR  FILE   BLOCK"
280 PRINT"  NO.","  NO.","NO.","  NO.","  NO.","NO.","  NO.","  NO.","NO."
290 PRINT
300 FOR sector_number=0 TO 254
310   LET file_number=PEEK(map_address+sector_number*2)
320   LET block_number=PEEK(map_address+sector_number*2+1)
330   PRINT "  ";sector_number,
340   SELect ON file_number
350     =0
360       PRINT "DIR",
370     =248
380       PRINT "MAP",
390     =253
400       PRINT "FREE",
410       LET number_of_free_sectors=number_of_free_sectors+1
420     =254,255
430       PRINT "BAD",
440       LET number_of_bad_sectors=number_of_bad_sectors+1
450     =REMAINDER
460       PRINT file_number,
470   END SELect
480   PRINT block_number,
490 END FOR sector_number
500 PRINT
510 PRINT "FREE SECTORS = ";number_of_free_sectors
520 PRINT "MAXIMUM AVAILABLE SECTORS = ";255-number_of_bad_sectors
530 STOP
```

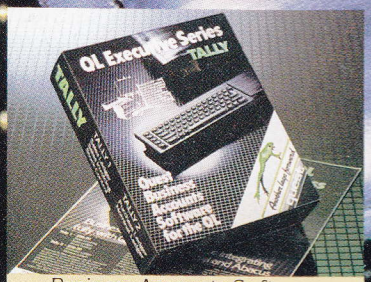
With Quest the QL really means business

Executive Series

QL Expansion Console

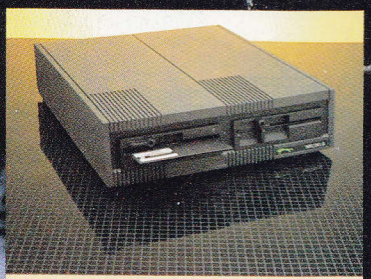


Memory Expansion Board



Business Accounts Software

CP/M 68K



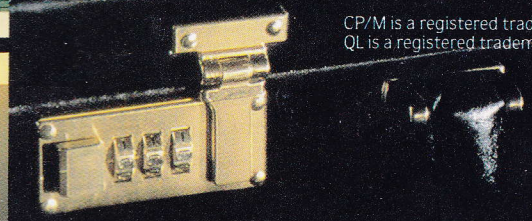
Disk Drive

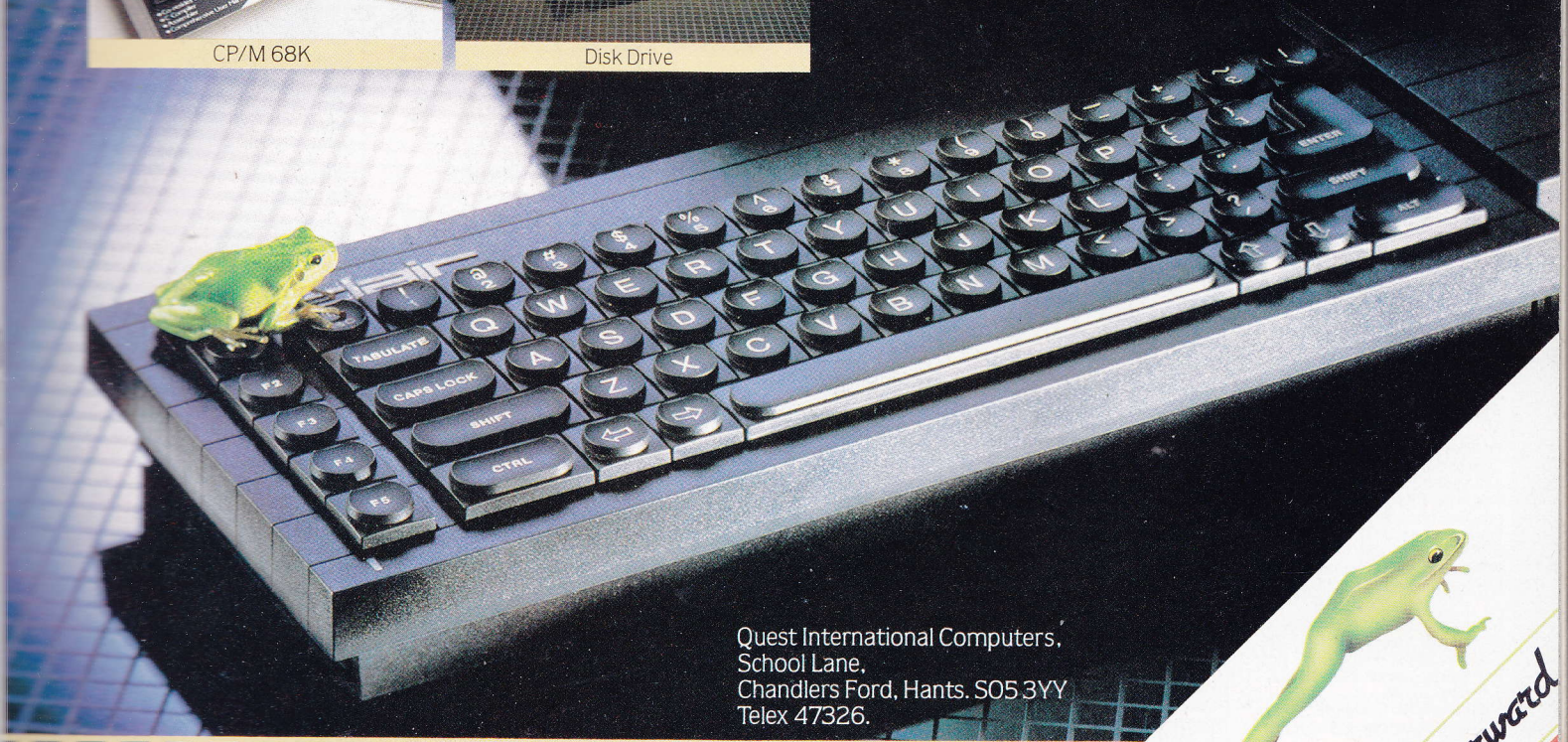
Quest's Executive series of quality peripherals and software for the Sinclair QL allows the business and serious home user to fully realise the full potential of the most exciting computer of the 80's.

Post the coupon below to obtain details of a wide range of Quest Executive products including:

- CP/M 68K
- Business Accounting Software
- Floppy and Winchester Disk Drives
- Memory Expansion Boards
- Monitors
- Printers
- Consumables

Telephone Sales Desk 04215 66488





Quest International Computers,
School Lane,
Chandlers Ford, Hants. SO5 3YY
Telex 47326.

Another leap forward

Please send me details of the Quest Executive Series

Name _____ Tel. No. _____

Address _____ My QL Serial Number is _____

_____

Position _____ QL/3

See Micronet 800 in action!

AVON

Bath. Boots, 1 Marchants Passage, Southgate. Tel: 0225 64402.
Bristol. Avon Computer Rentals, 16 West Street, Old Market. Tel: 0272 550600.
Bristol. Boots, 59 Broadmead. Tel: 0272 293631.
Bristol. Laskys, 16-20 Penn Street. Tel: 0272 20421.
Keynsham Key Computer Systems 42b High Street. Tel: 02756 5575.
Weston-super-Mare. K & K Computers, 32 Alfred Street. Tel: 0934 419324.

BEDFORDSHIRE

Bedford. Boots, The Harpur Centre, Harpur Street. Tel: 0234 56231.
Dunstable. Dormans, 7-11 Broad Walk. Tel: 0582 38302.
Leighton Buzzard. Datasine, 59 North Street. Tel: 0525 374200.
Leighton Buzzard. Milton Keynes Music & Computers, 17 Bridge Street. Tel: 0525 382504.
Luton. Hobbyte, Unit 16, The Arndale Centre. Tel: 0582 457195.
Luton. Laskys, 190-192 Arndale Centre. Tel: 0582 38302.
Luton. Terry More, 49 George Street. Tel: 0582 23391.

BERKSHIRE

Slough. Laskys, 75 Queensmere Centre. Tel: 0753 78269.
Slough. MV Games, 245 High Street. Tel: 75 21594.

BUCKINGHAMSHIRE

Bletchley. RAMS Computer Centre, 117 Queensway. Tel: 0908 647744.
Chesham. Reed Photo & Computers, 113 High Street. Tel: 0494 783373.

CAMBRIDGESHIRE

Cambridge. Boots, 65-67 Sidney Street and 28 Petty Curry Street. Tel: 0223 350213
Cambridge. Cambridge Computer Store, 4 Emmanuel Street. Tel: 0223 358264
Cambridge. Heffers Stationers. 19 Sidney Street. Tel: 0223 358241.
Peterborough. Boots, 40-42 Bridge Street, Queensgate. Tel: 0733 65352.

CHESHIRE

Chester. Boots, 47-55 Foregate Street. Tel: 0244 28421
Chester. Computer Link, 21 St. Werburgh Street. Tel: 0244 316516.
Crewe. Midshires Computer Centre, 68-78 Nantwich Road. Tel: 0270 211086
Ellesmere Port. RFR TV & Audio, 1 Pooltown Road, Whitby. Tel: 051-356 4150.
Hyde. C Tech Computers, 184 Market Street. Tel: 061-366 8223.
Macclesfield. Camera Computer Centre, 118 Mill Street. 0625 27468
Macclesfield. Computer Centre, 68 Chestergate. Tel: 0625 618827.
Marple. Marple Computer Centre, 30-32 Market Street. Tel: 061-427 4328
Stockport. National Micro Centres, 36 St Petersgate. Tel: 061-429 8080
Stockport. Stockport Micro Centre, 4-6 Brown Street. Tel: 061-477 0248.
Widnes. Computer City, 78 Victoria Road. Tel: 051-420 3333.
Wilmslow. Wilmslow Micro Centre. 62 Grove Street. Tel: 0625 530890.

CLEVELAND

Middlesbrough. Boots, 88-90 Linthorpe Road, The Cleveland Centre. Tel: 0642 249616.

CUMBRIA

Kendal. The Kendal Computer Centre, Stramongate. Tel: 0539 22559.
Penrith. Penrith Communications, 14 Castlegate. Tel: 0768 67146.
Whitehaven. P D Hendren, 15 King Street. Tel: 0946 2063.
Workington. Technology Store, 12 Finkle Street. Tel: 0900 66972.

DERBYSHIRE

Alfreton. Gordon Harwood, 67-71 High Street. Tel: 0733 836781.
Chesterfield. Boots, 35-37 Low Pavement, Market Place. Tel: 0246 203591.
Chesterfield. Computer Stores, 14 Stephenson Place. Tel: 0246 208802.
Derby. Boots, 1 Devonshire Walk. Tel: 0322 45886.

DEVON

Exeter. Boots, 251 High Street, Tel: 0392 32244.

DEVON

Exeter. Open Channel, Central Station, Queen Street. Tel: 0392 218187.
Paignton. Computer Systems Ltd, 35 Hyde Road. Tel: 0803 524284.
Plymouth. Syntax, 76 Cornwall Street. Tel: 0752 28705.
Seaton. Curtis Computer Services, Seaton Computer Shop, 51c Harbour Road. Tel: 0297 22347.
Tiverton. Actron Microcomputers, 37 Bampton Street. Tel: 0884 252854.

DORSET

Bournemouth. Brook Computers, 370 Charminster Road. Tel: 0202 533054.
Bournemouth. Lansdowne Computer Centre, 1 Lansdowne Crescent, Lansdowne. Tel: 0202 20165.
Dorchester. The Paper Shop, Kings Road. Tel: 0305 64564.
Poole. Lansdowne Computer Centre, 14 Arndale Centre. Tel: 0202 670901.

ESSEX

Basildon. Basildon Software Centre, 78-80 Liberty Shopping Hall, East Square. Tel: 0268 27922.
Chelmsford. Maxton Hayman, 5 Broomfield Road. Tel: 0245 354595
Chelmsford. Way In Computers, 7 Village Square, Chelmer Village. Tel: 0245 467858.
Colchester. Boots, 5-6 Lion Walk. Tel: 0206 577303.
Grays. H. Reynolds, 28a Southend Road. Tel: 0375 31641.
Harlow. Harlow Computer Centre, 17 Staple Tye. Tel: 0279 22846.
Harlow. Laskys, 19 The Harvey Centre. Tel: 0279 443495.
Hornchurch. Comptel Computer Systems, 112a North Street. Tel: 0402 446741.
Ilford. Boots, 177-185 High Road. Tel: 01-553 2116.
Romford. Software Plus, 72 North Street. Tel: 70 65271.
Southend-on-Sea. Computerama, 88 London Road. Tel: 0702 335443.
Southend-on-Sea. Computer Centre, 336 London Road. Tel: 0702 337161.
Southend-on-Sea. Estuary Personal Computers, 318 Chartwell North, Victoria Circus Shopping Centre. Tel: 0702 614131.

GLOUCESTER

Cheltenham. Laskys, 206 High Street. Tel: 0242 570282.
Cheltenham. Screen Scene, 144 St. Georges Road. Tel: 0242 528979.
Gloucester. Boots, 38-46 Eastgate Street. Tel: 0452 423501.

HAMPSHIRE

Basingstoke. Fishers, 2-3 Market Place. Tel: 0256 22079.
Southampton. Business Electronics, Micromagic At Atkins, 7 Civic Centre Road. Tel: 0703 25903.
Waterlooville. GB Microland, 7 Queens Parade, London Road. Tel: 0705 259911.

HERTFORD

Hitchin. County Computers, 13 Bucklesbury. Tel: 0462 36757.
Potters Bar. the Computer Shop, 107 High Street. Tel: 0707 44417.
Stevenage. DJ Computers, 11 Town Square. Tel: 0438 65501.
Watford. Laskys, 18 Charter Place. Tel: 0923 31905.
Watford. SRS Microsystems, 94 The Parade, High Street. Tel: 0923 26602.
Watford. Watford Electronics, Cardiff Road. Tel: 0923 405588.
Welwyn Garden City. DJ Computers, 40 Fretherne Road. Tel: 96 28444.

HUMBERSIDE

Beverley. Computing World, 10 Swabys Yard, Dyer Lane. Tel: 0482 881831.

KENT

Ashford. DGH, 10 North Street. Tel: 0233 32597.
Ashford. Geerings of Ashford, 80 High Street. Tel: 0233 33366.
Bexleyheath. Laskys, 15-16 Broadway Shopping Centre. Tel: 01-301 3478.
Bromley. Boots, 148-154 High Street. Tel: 01-460 6688.
Bromley. Computers Today. 31 Market Square. Tel: 01-290 5652.
Bromley. Walters Computers, Army & Navy, 64 High Street. Tel: 01-460 9991.
Chatham. Boots, 30-34 Wilmott Square, Pentagon Centre. Tel: 0634 405471.

KENT

Gravesend. Gravesend Home Computers, 39 The Terrace. Tel: 0474 23871.
Gillingham. Regal Software Supplies, 49 High Street. Tel: 0634 579634.
Maidstone. Kent Micros, 51 Union Street. Tel: 0622 52784.
Rainham. Microway Computers, 39 High Street. Tel: 0634 376702.
Sevenoaks. Ernest Fielder Computers, Dorset Street. Tel: 0732 456800.
Shortlands. The Village House of Computers, 87 Beckenham Lane. Tel: 01-460 7122.
Sittingbourne. Computer Plus, 65 High Street. Tel: 0795 25677.
Tunbridge Wells. Modata Computer Centre, 28-30 St. Johns Road. Tel: 0892 41555.

LANCASHIRE

Blackburn. Tempo Computers, 9 Railway Road. Tel: 0254 691333.
Blackpool. Blackpool Computer Store, 179 Church Street. Tel: 0253 20239.
Burnley. IMO Business Systems, 39-43 Standish Street. Tel: 0282 54299.
Preston. 4Mat Computing, 67 Friargate. tel: 0772 561952.
Preston. Laskys, 1-4 Guildhall Arcade. Tel: 0772 24558.
Wigan. Wildings Computer Centre, 11 Mesnes Street. Tel: 0942 22382.

LEICESTERSHIRE

Leicester. Boots, 30-36 Gallowtree Gate. Tel: 0533 21641.
Market Harborough. Harborough Home Computers, 7 Church Street. Tel: 0858 63056.

LONDON

W1. Computers of Wigmore Street, 104 Wigmore Street. Tel: 01 486 0373.
W1. HMV, 363 Oxford Street. Tel: 01-629 1240.
W1. Laskys, 42 Tottenham Court Road. Tel: 01-636 0845.
W1. Lion House, 227 Tottenham Court Road. Tel: 01-637 1601.
W1. Sonic Foto Micro Center, 256 Tottenham Court Road. Tel: 01-580 5826.
W1. Tomorrows World Today, 27 Oxford Street. Tel: 01-439 7799.
W1. Walters Computers, DH Evans, Oxford Street. Tel: 01-629 8800.
WC1. Transam Micro Systems, 59-61 Theobalds Road. Tel: 01-405 5240.
W8. Walters Computers, Barkers, Kensington High Street. Tel: 01-937 5432.
SE7. Vic Oddens Micros, 5 London Bridge Walk. Tel: 01-403 1988.
SE9. Square Deal, 373-375 Footscray Road, New Eltham. Tel: 01-859 1516.
Lewisham. Laskys, 164 High Street. Tel: 01-852 1375.
SE15. Castlehurst Ltd, 152 Rye Lane, Peckham. Tel: 01-639 2205.
EC2. Devron Computer centre, 155 Moorgate. tel: 01-638 3339.
N14. Logic Sales, 19 The Bourne, The Broadway, Southgate. Tel: 01-882 4942.
N22. Boots, 38-40 High Road, Wood Green. Tel: 01-881 0101.
NW3. Maycraft Micros, 58 Rosslyn Hill, Hampstead. Tel: 01-431 1300.
NW4. Davinci Computer Store, 112 Brent Street, Hendon. Tel: 01-202 2272.
NW7. Computers Inc, 86 Golders Green. Tel: 01- 209 0401.
NW10. Technomatic, 17 Burnley Road, Wembley. Tel: 01-208 1177.

MANCHESTER

Bolton. Computer World UK Ltd, 208 Chorley Old Road. Tel: 0204 494304.
Manchester. Boots, 32 Market Street. Tel: 061-832 6533.
Manchester. Laskys, 12-14 St. Marys Gate. Tel: 061-833 0268.
Manchester. Mighty Micro, Sherwood Centre, 268 Wilmslow Road, Fallowfield. Tel: 061-224 8117.
Manchester. NSC Computer Shops, 29 Hanging Ditch. Tel: 061-832 2269.
Manchester. Walters Computers, Kendal Milne, Deansgate. Tel: 061-832 3414.
Oldham. Home & Business Computers, 54 Yorkshire Street. Tel: 061-633 1608.
Swinton. Mr Micro, 69 Partington Lane. Tel: 061-728 2282.

MERSEYSIDE

Heswall. Thornguard Computer Systems, 46 Pensby Road. Tel: 051-342 7516.
Liverpool. Hargreaves, 31-37 Warbreck Moor, Walton. Tel: 051-525 1782.
Liverpool. Laskys, Dale Street. Tel: 051-236 3298.
St. Helens. Microman Computers, Rainford Industrial Estate, Mill Lane Rainford. Tel: 0744 885242.
Southport. Central Studios, 38 Eastbank Street. Tel: 0704 31881.

MIDDLESEX

Enfield. Laskys, 44-48 Palace Garden Shopping Centre. Tel: 01-363 6627.
Harrow. Camera Arts, 42 St. Anns Road. Tel: 01-427 5469.
Harrow. Harrow Micro, 24 Springfield Road. Tel: 01-427 0098.
Hounslow. Boots, 193-199 High Street. Tel: 01-570 0156.
Southall. Twillstar Computers Ltd, 7 Regina Road. Tel: 01-574 5271.
Teddington. Andrews, Broad Street. Tel: 01-997 4716.
Twickenham. Twickenham Computer Centre, 72 Heath Road. Tel: 01-892 7896.
Uxbridge. JKL Computers, 7 Windsor Street. Tel: 0895 51815.

NORFOLK

Norwich. Adams, 125-129 King Street. Tel: 0603 22129.

NOTTINGHAMSHIRE

Sutton in Ashfield. HN & L Fisher, 87 Outram Street. Tel: 0623 54734.

OXFORDSHIRE

Abingdon. Ivor Fields Computers, 21 Stern Street. tel: 0235 21207.
Banbury. Computer Plus, 2 Church lane. Tel: 0295 55890.
Oxford. Absolute Sound & Video, 19 Old High Street, Headington. Tel: 0865 65661.
Oxford. Science Studio, 7 Little Clarendon Street. Tel: 0865 54022.

SCOTLAND

Aberdeen. Boots, 133-141 Union Street. Tel: 0224 585349.
Edinburgh. Boots, 101-103 Princes Street. Tel: 031-225 8331.
Edinburgh. Laskys, 4 St. James Centre. Tel: 031-556 1864.
Glasgow. Boots, 200 Sauchiehall Street. Tel; 041-332 1925.
Glasgow. Boots, Union Street and Argyle Street. Tel; 041-248 7387.
Glasgow. Tom Dixon Cameras, 15-17 Queen Street. Tel: 041-204 0826.

SHROPSHIRE

Shrewsbury. Clairmont Enterprises, Hills Lane. Tel: 3647 52949.
Shrewsbury. Computerama, 13 Castlegate. Tel: 0743 60528.
Telford. Computer Village Ltd, 2/3 Hazeldine House, Central Square. Tel: 0952 506771.
Telford. Telford Electronics, 38 Mall 4. Tel: 0952 504911.

STAFFORDSHIRE

Newcastle-under-Lyme. Computer Cabin, 24 The Parade, Silverdale. Tel: 0782 636911.
Stafford. Computerama, 59 Foregate Street. Tel: 0785 41899.
Stoke-on-Trent. Computerama, 11 Market Square Arcade, Hanley. Tel: 0782 268524.

SUFFOLK

Bury St. Edmunds. Boots, 11-13 Cornhill. Tel: 0284 701516.
Bury St. Edmunds. The Computer Centre, 1-3 Garland Street. Tel: 0284 705503.
Ipswich. Brainwave Micros, 24 Crown Street. Tel: 047 350965.

SURREY

Bagshot. P & H Electronics, 22-24 Guildford Road. Tel: 0276 73078.
Croydon. Laskys, 77-81 North End. Tel: 01-681 8443.
Croydon. The Vision Store, 53-59 High Street. Tel: 01-686 6362.
Croydon. The Vision Store, 96-98 North End. Tel: 01-681 7539.
Epsom. The Micro Workshop, 12 Station Approach. Tel: 0372 721533.
Guildford. Walters Computers, Army & Navy, 105-111 High Street. Tel: 0483 68171.
Wallington. Surrey Micro Systems, 53 Woodcote Road. Tel: 01-647 5636.
Woking. Harpers, 71-73 Commercial Way. Tel: 0486 225657.

SUSSEX

Bexhill-on-Sea. Computerware, 22 St. Leonards Road. Tel: 0424 223340.
Brighton. Boots, 129 North Street. Tel: 0273 27088.
Brighton. Gamer, 71 East Street. Tel: 0273 728681.
Brighton. Laskys, 151-152 Western Road. Tel: 0273 725625.
Crawley. Gatwick Computers, 62 The Boulevard. Tel: 0293 37842.
Crawley. Laskys, 6-8 Queensway. Tel: 0293 544622.
Eastbourne. Boots, 15 Eastbourne Arndale Centre. Tel:

TYNE & WEAR

Gateshead. DP Supplies, St. Andrews House, Westfield Terrace. Tel: 0632 785068.
Newcastle-upon-Tyne. Boots, Eldon Square. Tel: 0632 329844.
Newcastle-upon-Tyne. RE Computing, 12 Jesmond Road. Tel: 0632 815580.

WALES

Aberdare. Inkey Computer Services, 70 Mill Street, The Square, Trecynon. Tel: 0685 881828.
Aberystwyth. Aberdata at Galloways, 23 Pier Street. Tel: 0970 615522.
Cardiff. Boots, 26 Queens Street & 105 Frederick Street. Tel: 0222 31291.
Cardiff. P & P Computers. 41 The Hayes. Tel; 0222 26666.
Swansea. Boots, 17 St. Marys Arcade, The Quadrant Shopping Centre. Tel: 0792 43461.

WARWICKSHIRE

Coventry. Impulse Computer World, 60 Hertford Street Precinct. Tel: 0203 553701.
Coventry. JBC Micro Services, 200 Earlsdon Avenue, North Earlsdon. Tel: 0203 73813.
Coventry. Laskys, Lower Precinct. Tel: 0203 27712.
Leamington Spa. IC Computers, 43 Russell Street. Tel: 0926 36244.
Leamington Spa. Leamington Hobby Centre, 121 Regent Street. Tel: 0926 29211.
Nuneaton. Micro City, 1a Queens Road. Tel: 0203 382049.
Rugby. O.E.M., 9-11 Regent Street. Tel: 0788 70522.

WEST MIDLANDS

Birmingham. Boots, City Centre House, 16-17 New Street. Tel: 021-643 7582.
Birmingham. Laskys, 19-21 Corporation Street. Tel: 021-632 6303.
Dudley. Central Computers, 35 Churchill Precinct. Tel: 0384 238169.
Stourbridge. Walters Computer Systems. 12 Hagley Road. Tel: 0384 370811.
Walsall. New Horizon, 1 Goodall Street. Tel: 0922 24821.
West Bromwich. DS Peakman, 7 Queens Square. Tel: 021-525 7910.
Wolverhampton. Laskys, 2 Wulfrum Square. Tel: 0902 714568.

YORKSHIRE

Bradford. Boots, 11 Darley Street. Tel: 0274 390891.
Leeds. Boots, 19 Albion Arcade, Bond Street Centre. Tel: 0532 33551.
Sheffield. Laskys, 58 Leopold Street. Tel: 0742 750971.
York. York Computer Centre, 7 Stonegate Arcade. Tel: 0904 641862.

See Micronet at John Lewis, Laskys and main Boots branches

Phone 01-278 3143 for your free information pack now!

Micronet 800, 8 Herbal Hill, London EC1

HARDWARE

ON THE HORIZON

At long last, QL disk drives. And first on the scene are CST – we test their interface and preview the competition.

Disk drives are mass storage devices which provide read/write non-volatile memory (ie memory that does not disappear when the computer is switched off). The alternatives to disk drives are cassette tapes, microdrive cartridges, erasable/programmable ROMs and CMOS-backed RAM, that is, battery supported RAM (for which power is never switched off).

Floppy disk drives were invented by IBM in 1973. Originally some 8″ in diameter they have since been reduced in size to 5.25″, and more recently to 3.5″ and 3″ making them more compact and giving faster access.

The floppy disk itself is plastic coated with ferric oxide and is protected by a cardboard or plastic casing. In addition to the 'access' hole about which the disk rotates, the casing has a notch in one corner, a small timing hole slightly off centre and a slot across its radius.

The notch if covered or 'tagged' provides a simple way to prevent further information from being written to the disk.

Holes And Slots

The timing hole provides a means for gauging the speed at which the disk rotates. A beam of light is shone down onto the disk and through a small index hole in it to be picked up by a photosensor on the other side.

The remaining slot permits the drive's read/write head(s) to move freely over the disk's surface and locate the correct track to which information may be written, or from which it may be read.

Like a microdrive cartridge, a disk must be 'formatted' before it may be used. The operation consists of the disk being magnetically divided up into concentric tracks and then further subdivided into sectors. Disks for the QL may be formatted on one or both sides, thus 40 or 80 tracks. However, they will invariably contain nine sectors per track and 512 bytes per sector (double density). Tracks and sectors are numbered and this is stored in a 'header' at the start of every sector along with information pointers to related sectors.

In operation, the disk rotates on a spindle at 300-360 revolutions per minute. An electromagnetic head then moves over the disk's surface guided by an electronic disk controller. The controller also translates electrical signals from the computer, sending them to the head which produces magnetic fields to be recorded on the disk when data is written. The process is reversed when data is being read. Data is synchronised, as with every byte, a timing is sent to keep in step. If a disk drive is double sided, two heads will read and write information to either side of a disk.

Speed: On a microdrive the read/write head is static and the tape alone moves around. This means that to go back to the beginning of a file, the tape must be wound round almost a full circuit. On a disk, a movable head means that little more than a single rotation is required to extract any piece of information. As a result, access times on disk averages 125 milliseconds as opposed to the QL microdrive's 3.5 seconds.

Capacity: Whilst microdrives are limited to a maximum of 80-100K, on disk up to 3-megabytes are available.

Maintenance: Unlike the microdrive tape, the floppy's surface is not in constant contact with the read/write head and consequently is not subject to the same amount of wear and tear. Indeed with some floppies it is not uncommon to find that the integrity of data has been guaranteed by the manufacturer. However, it is highly unlikely that this will ever be the case for microdrives. With the microdrive tape in contact with the read/write head, the medium is subject to considerably more wear and tear than its floppy cousin.

Price: Until quite recently disk drives giving a reasonable amount of storage (1 megabyte) were prohibitively expensive and often cost more than the computer itself. However, this is no longer the case and though in absolute terms they remain more expensive than, say, an additional microdrive or two, large storage capacities mean that the disk user benefits from substantial economies of scale in relative terms.

For example, assuming that the cost of an additional microdrive for the QL matches that for the Spectrum, then 1K's storage works out at 52p (50/95), whereas on a 1 megabyte disk drive (plus interface) this works out at 48p (350/720). Furthermore, the cost of back-ups works out at £4.95 and £.005 (half a penny) respectively.

Drives In Perspective

With the average 8-bit home computer, the benefits of disk drives are often self-evident as the machines depend upon cassette tapes which, as a storage medium, are hopelessly inadequate. However, justifying their acquisition is another matter altogether. Insufficient RAM, slow processors, unsophisticated operating systems and absence of serious software all make it unlikely that the drives will be put to good use.

With the QL, on the other hand, an advanced specification virtually guarantees that if disks are tagged on, users should be able to use them to their limits. Yet, if one brushes aside the sensationalism surrounding microdrives, the benefits of disk drives whilst still apparent are no way near as clear cut.

Exploding The Myth

Seen by many as a substitute for disks, the QL's microdrives have come under a barrage of criticism. This, though often justified – bearing in mind the QL's seemingly exclusive reliance on these devices – has obscured a number of important facts.

Firstly, the drives do work, and when used with the latest versions of Psion's packages, perform adequately. For proof you need look no further than the magazine itself which has been using unexpanded QLs and Quill (V 2.00) to produce much of the material published in the last two issues.

Secondly, in terms of capacity, speed and reliability, even if they cannot compare with disks, the microdrives remain a vast improvement on cassette storage in reducing loading times from minutes to tenths of seconds.

And finally, as they are included in the cost of the machine, the drives come cheap.

Microdrives then, are a viable storage medium in themselves. However, the original intention that QL should rely exclusively upon them (note the absence of disk drives in QDOS), is absurd. More than anything it has contributed to QL's image as an uncertain hybrid between the home and business computer with a doubtful future.

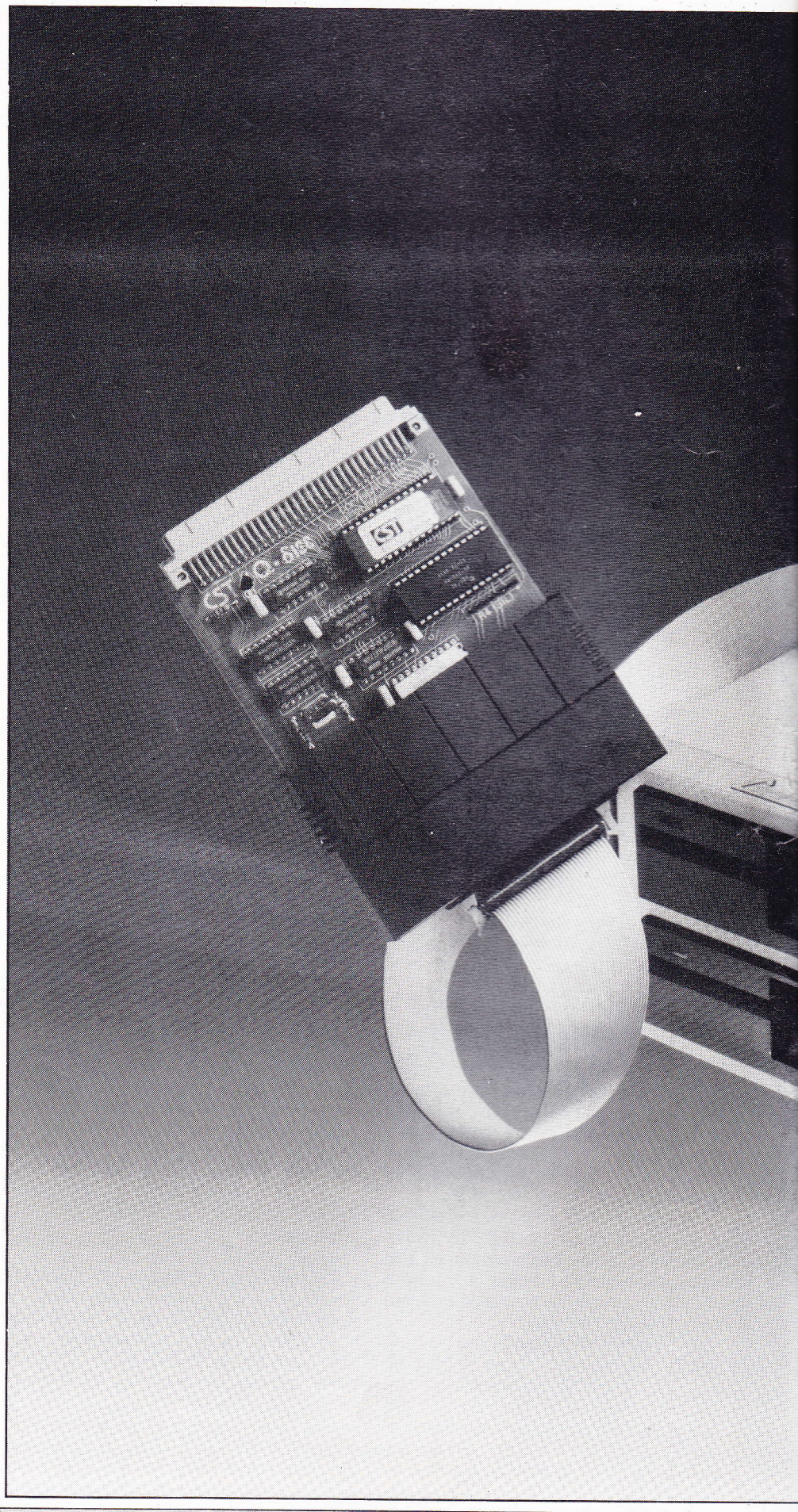
Obviously microdrives have a role to play but that is determined by the uses to which QL is put. If we split potential applications for the QL into four areas – Home, Professional, Business and Software Development, then the need for alternative mediums soon becomes apparent.

The Home

Likely uses here include entertainment (games), education (SuperBasic), household accounts (Abacus) and letter writing (Quill).

None of these applications are likely to demand large amounts of storage. Documents for example will be one-offs, with programs short and accounts simple. In this situation microdrives will suffice, though the high cost of replacement cartridges and the consequent premium paid on software is likely to prove a burden. Within this context a cheap secondary back-up medium such as the facility to load from, or save to cassette could prove to be an attractive proposition.

Here Psion's packages are likely to come into their own as the QL is used



primarily as an analytical tool and wordprocessor. In either respect, microdrives prove adequate though time consuming. Furthermore, the cartridges, as they are compact and easily transportable would seem (at least initially) as attractive as floppies.

Nevertheless, even here a single disk drive would not be amiss. Not only would it provide a speedy way of loading existing software but also give a more reliable back-up for important documents or reports. Finally, as byte for byte cartridges are three to four times more expensive than floppies, the disk user would enjoy quite substantial savings – however not sufficient to offset the initial outlay.

Software Development

Anybody who has tried assembling or compiling a program using microdrive-based software will realise that the processes, involving extensive file manipulation, make heavy weather of an unexpanded QL's I/O capabilities. Little wonder then that almost all commercial QL software has been developed on larger and faster machines where disk drives are standard.

RAM disks which permit virtually instantaneous access, might be considered as an alternative to microdrives. However, as the memory is volatile, users would be compelled to load and save, from and to cartridge at the beginning and end of every session – a tedious duplication of effort and unlikely to appeal to efficiency conscious programmers.

So disk drives would appear to be the obvious choice. With loading times characteristically measured in milliseconds as opposed to seconds, they would greatly enhance the QL as a tool for software development. This in turn would speed up the purely mechanical processes in program production and leave the programmer with more time to apply his creative talents.

Business

This is the domain of the desk top PC, with applications typically covering accounts production, inventory control, sales and budget analysis (Abacus), financial reporting (Quill and Easel) and general correspondence (Quill). Here, without the benefit of dual disk drives or a Winchester the QL simply cannot compete.

Irrespective of their size, businesses require a disproportionately large amount of reliable backing store to archive records for long periods of time. In this environment microdrives are hopelessly inadequate. The limited capacities of cartridges (80K) impose severe constraints upon the size of files, and make comprehensive stock control systems near impossible to construct, let alone maintain.

Additionally, large databases using Archive would be too slow in locating records or making updates. Finally, the need to constantly back-up records would impose considerable wear and tear upon cartridges as, unlike disks, the read/write head is in constant contact with the tape.

Overall, the case for disk drives, as one would expect, varies as the QL moves from the living room to the office. On the home front, disk drives are a luxury that users can afford to ignore. To the professional, microdrives are adequate but disk drives are better. For the programmer, disk drives are a godsend that dramatically improve development times. Finally, for the businessman disk drives are a necessity without which the QL would be useless.

Q-Disc

Computamate's Q-Disc interface is produced by the Cambridge based company CST. Similar to their Q-PI printer interface, Q-Disc consists of a small PCB measuring 105 × 145 × 25mm with an on-board ROM containing their disk driver. The driver conforms with the Sinclair defined standard – disks formatted with 40/80 tracks, 9 sectors per track and 512 bytes per sector. The interface is understood to run with any 5.25″ or 3.5″ Shugart compatible disk drive with an independent power source. The interface was tested using dual double sided, double density Teac 5.25″ drives (2 × 720K) supplied by Computamate themselves.

Installing Q-Disc is a matter of seconds. The unit slides into the QL's main I/O port on the left hand edge of the machine adding a further two inches of ribbed black plastic to the machine's length. A one metre ribbon cable then connects up the interface to the disk drive.

Obviously, once Q-Disc is installed no other device may make use of the I/O port (Q-PI included). Sinclair are understood to be working on an expansion module, though as this is likely to require its own power source it may be some time in the making.

In operation Q-Disc proves to be a joy to use. All existing QL file commands will work with disks, only the device name changes (from "mdv" to "flp").

A real bonus is that the interface

PHOTO BY TERRY BEDDIS

provides an arsenal of additional commands governing random access, file maintenance, job control and numeric conversion. Those acquainted with the QL Toolkit will recognise quite a few of these, as author Tony Tebby played a major role in the development of Q-Disc. Many of the extra commands work with microdrives as well as disks. The commands are as follows:

Multitasking JOBS, RJOBS, AJOBS, SPJOB – will display various jobs and allow you to suspend, activate or realease particular tasks.

Directory Enquiry STAT, WSTAT, WDIR, WDEL, WDEL_F – variations on the 'DIR' theme that permits groups of files to be deleted as well as listed. The 'W' prefix permits the use of 'wild cards'.

File Maintenance FOPEN, FOP_IN, FOP_OVER, FOP_DIR – Similar to the 'OPEN' commands but here any errors will be trapped and the program will not be stalled (ie, ferr = FOPEN filename).

FOP_OVER will be particularly welcome enabling a new file to be created or existing one overwritten. FOP_DIR lets you open a drive's directory file.

File Enquiry FLEN, FTYP, FDAT, FDOS – Will return information concerning a file's length, type, size of data area and your current position within the file.

Random Access GET, PUT, BGET, BPUT – enables you to write or read information to and from a particular location within a file. Information will appear either in its raw form, suitable for assembler programs or formatted so that it may be used by a SuperBasic program.

Numeric Conversion HEX$,BIN$, HEX, BIN – Will convert information on decimal values into binary or hexadecimal strings, or vice versa.

Development Tools FDSEL, FDSIDE, FDTRACK, FDREAD — Sophisticated routines that will permit non-standard disks to be read and further disk drivers to be written.

Miscellaneous RENAME, TRUNCATE, PROG_USE, DATA_USE, (EXEC, EXEC, EXEC_W) – Will permit files to be renamed or whittled down as well as default drives to be specified for program and data files.

With regard to its performance, when linked up to disk drives, the QL is transformed into a fully fledged desk-top PC with file operations carried out in seconds as opposed to minutes. On paper Q-Disc is said to reduce access times from an average of 3.5 seconds (microdrive) to 125 milliseconds. The following two tests illustrate these savings in real terms.

	mdv	flp
1. Loading a screen	3 secs	8 secs
2. Backing up between drives	8 mins	90 secs

Much to the relief of the majority of users and in support of their claim of full Psion compatibility, Computamate include a special conversion program with Q-Disc. Whilst the program cannot miraculously transform old versions into new, it does breathe new life into them. This is achieved by replacing all occurrences of "mdv" by "flp" whilst copying the programs across to disk.

Versions 2.00 of the packages will run unamended on Q-Disc. These versions, we understand will be released shortly and include an additional program called 'config_bas' (written by Psion) which permits the user to set default devices for system, data and help files.

New or old, all the packages benefit from the transfer to disk. On Quill, for example, lengthy documents may be manipulated as easily as memos; loading and saving times are halved; scrolling fluid and block moves near instantaneous. The most marked improvement, however, comes with Archive where faster file access has meant a vast improvement in search and update times.

On the new versions, where Psion's own improvements in speed and memory usage have eliminated the packages' much criticised snail-like qualities, the transfer to disk has raised them to a position where they can compete openly with the likes of WordStar, dBase 2 and VisiCalc.

All in all, being easily installed, neat and compact, Q-Disc leaves little ground for complaint. A third of the cost of a QL, Q-Disc is expensive. Extra commands and full Psion compatibility, however, mean that CST have not only produced a working interface, but given the user everything he might need to get the most out of disks with the least effort. As the device is the first on the scene, it means that the competition have their work cut out for them.

[redacted]

Computamate Data Products,
Scotia Road, Burslem, Stoke-on-Trent. Tel: 0782 811711.

Q-Disc Interface, £149
Teac dual double density (720K formatted) disk drives plus Q-Disc Interface
a) 5.25" or 3.5" single sided (2 × 360K formatted), **£367**
b) 5.25" double sided (2 × 720K formatted), **£574**
All prices include VAT, cables and manuals. Q-Disc will be available from W H Smiths and mail order from Computamate themselves.

Medic QL-Expansion Cartridge System
Medic, 76 Grainger Close, Basingstoke, Hants RG22 4EA.
When seen, this device was still in prototype form with a few bugs yet to be ironed out. The manufacturer, however, is promising the earth in the form of a box which will ultimately house not just a disk interface but modem, memory expansion and parallel printer port as well. Like many QL peripherals, the interfacing software is being written by QDOS author, Tony Tebby, which bodes well for the future.

The disk interface is understood to include a Disk Doctor for recovering corrupted data and is authored by Leon Heller. The modem will parallel Unicom's specification.

The product is scheduled (hopefully) for volume production in February. Prices (inc VAT) will be in the following range:
Disk Interface **£80**
Modem **£80**
Memory 64K to 512K **£100 to £400**

Microperipherals Disk Package
Intec Unit 3, Hassocks Woods, Wade, Basingstoke, Hants
The package will include a disk interface, 720K (formatted) 3.5" disk drive plus utilities disk. The interface, we understand, will be Psion compatible (old and new versions). A small subset of QL-Toolkit disk handling commands will be resident on ROM. The remainder will come on floppy supplied with the drives. These will include disk and screen editors. Prices have yet to be fixed.

Sure Shot Supreme QL Joystick
Cookridge Computer Supplies, PO Box IW9, Leeds LS16 6RE
There is little call for a joystick on the QL at present, though with two sophisticated graphics packages from Talent and Psion on the horizon the situation looks set to change. In which case CCS will benefit from being the first on the scene.

Priced at £19.95, the Sure Shot would seem rather expensive for what is ultimately a relatively unsophisticated device. However, supplied with a Sure Shot standard model (not available for the QL) we found the components of a high quality and the product responsive and durable.

NOW an expansion system as versatile as the QL

All in one - one system, one cartridge

MEDIC takes up where others leave off. Why narrow down your choices? The Q.L. was intended as a versatile computer capable of fulfilling many roles. With our system cartridge, the choice is yours. As well as a disc interface including parallel port you can have any one of the following: memory upto 512K, modem, Eprom card (192K) experimental card including front interface. You can upgrade to any level in line with your needs. Our disc interface affords full Psion, versions 1 and 2, compatibility.

Ask your dealer for details or order direct:

medic datasystems limited

International House, Cliddesden, Basingstoke, Hants. RG25 2JL Telephone: 0256 52703

☐ Please send me further details on your products.

I wish to order the following: (please tick required items)

Individual Products

- ☐ 64K Memory £99.95
- ☐ 128K Memory £129.95
- ☐ 256K Memory £169.95
- ☐ 512K Memory† £259.95
- ☐ Eprom card socketed (192K – excluding Eproms) experimentation card with front interface £24.95
- ☐ Dust Cover that doubles up as a stand £14.95
- ☐ Modem* £79.95

†*Can only be powered if Medic disc system connected*

Packages* *Delivery beginning April*

- ☐ 1 Megabyte disc drive (cased, inc. PSU) disc interface and parallel interface, with cables £249.96
- ☐ 2 1 Megabyte disc drives, (cased, inc. PSU) disc interface parallel interface, with cable £399.95
- ☐ 1 megabyte disc drive, memory inc., disc interface and parallel interface.

	1 disc drive	2 disc drives
☐ 64K	£299.95	☐ £449.95
☐ 128K	£329.95	☐ £479.95
☐ 256K	£359.95	☐ £509.95
☐ 512K	£449.95	☐ £595.95

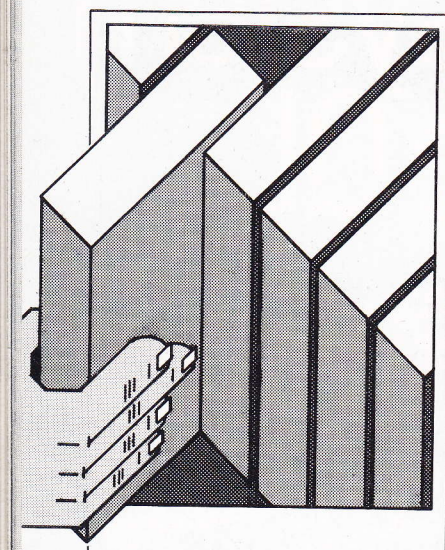
To order products please make cheques/P.O.'s payable to MEDIC DATASYSTEMS LTD. Allow 28 days for delivery. All prices include VAT. Please add £5 p&p.

NAME_____

ADDRESS_____

_____

Signed_____



BOOKMARKS

Nicky Trevett reviews the latest selection of books available for the QL.

The QL handbooks, compendiums and guides continue to pour into the bookshops. There's nothing particularly original among this month's selection, which are all more or less variations of well-worn themes, but some could be well worth your attention.

Down-To-Earth

A QL Compendium by Martin Gandoff and Robin Kinge, published by Addison-Wesley, offers 30 programs for your QL, most of which are games, plus the obligatory introduction to the computer itself, keyboard, microdrives, starting up, the QL editor, and so on. This has become a familiar approach, but I found this particular offering more down-to-earth and entertaining than most. When the authors come across a feature of the QL or the manual they do not much admire, they tend to say so and offer advice on improving matters. Throughout the book, there are numerous practical tips which should prove of value to the hobbyist, all obviously based on the first-hand experiences of the authors.

Chapter two comprises a handy guide to effective SuperBasic programming; not a 'how to program' guide for the beginner, but a lot of advice for users who already have some programming knowledge.

The programs themselves cover computer 'art' and ways of creating 'pretty' displays and backgrounds for games; games of skill (Fruit Machine, etc.); weapon games (Depth Charge, Streafe, Mortar, all mostly highly murderous); 'modern games' (Formula 1, and so on); and traditional games (Horse Race, Noughts and Crosses . . .). I was not exactly knocked out by their originality, but if you enjoy

PHOTO BY TERRY BEDDIS

this sort of thing to play on your QL, and practice your programming technique at the same time, you could do worse, though £7.95 seems a bit high.

One word of warning, as the authors point out, all the programs were developed on a QL version JM. If you use another, be prepared to make occasional fine adjustments to a program before you are able to run it.

Easy As ABC

Tim Hartnell's QL Handbook (£7.95, Interface) is not, for once, a handbook of games. A more suitable title would be Tim Hartnell's QL SuperBasic Handbook, since, as he admits right at the start, it is the built-in SuperBasic language that really interests him about the QL.

His aim is to make SuperBasic easy to learn, appreciate and apply. However, although he says "I've made few, if any, assumptions about the level of programming proficiency you now possess", anyone who is not already reasonably familiar with some version of BASIC is likely to find themselves floundering, since the emphasis is on ways in which SuperBasic transcends BASIC.

After a few pages telling you how to get started on QL, the next five chapters deal with the fundamental components of SuperBasic. There is a brief look at structured programming techniques, a much lengthier digression into graphics and sound, and two interesting chapters introducing two languages that can be 'emulated' by SuperBasic — Logo and Forth.

Finally, by way of light relief, there is a simulation program to enter called Bankruptcy, and a chapter on creating and playing adventures which includes two complete programs.

The Professional Approach

Anyone planning to use their QL strictly for business only might be interested in Colin Lewis' Professional and Business Uses of the QL, published by Collins at £7.95. Colin Lewis sums up the purpose of his book admirably, declaring that it is "For users of the Sinclair QL who wish to use the four packages supplied (with the QL) . . . to solve practical business problems with the minimum of fuss." In other words, people who just want to stick in a ready-written program and make it work for their business without any of this programming nonsense.

And this is exactly what the book helps them to do. There is minimal treatment of such matters as the hardware and the operating system, offering only such information as is absolutely necessary to run the software, and maximum coverage of Quill, Abacus, Archive and Easel. There are of course books already treating these packages at length, not to mention the manual itself, but here the programs are placed strictly in a business context.

The chapter on Quill is a little disappointing, concentrating on simply exploring the features of Quill and how to use it to do your word processing without much on possible business applications. But the sections on Abacus, Archive and Easel are more stimulating, suggesting ways in which Abacus, for example, can be used for investment analysis and a variety of business functions. The Archive chapter includes the whole process of setting up an actual database and then manipulating it, and chapter six offers a great deal of advice on using Easel.

The final part of the book is a little different and breaks the 'no programming' rule by examining the way in which Archive can be used as a programming language to create new programs to perform repetitive tasks. These could be printing address labels, individually addressing letters to clients and producing financial reports incorporating subtotals triggered by the change of a sort key. All these applications are covered, and should prove useful to many business users that they might even develop a reluctant interest in programming!

Keeping It In The Family

Abacus alone is the subject of QL Abacus by Clare Spottiswoode, the third in the series published by Century Communications dealing with each of the QL's companion software packages (QL Quill and QL Easel were reviewed in QL User, December/January).

Here are the further adventures of the Blake family as they make use of Abacus to calculate mortgages, prepare sales forecasts, plan their redecorating budgets and so on. Like QL Quill, the book covers a great deal of ground in a highly entertaining manner, drawing its example applications from both home and business. There is the usual 'first steps' section dealing with the QL itself and starting up Abacus, then two chapters on creating and refining spreadsheets.

Functions and formulae are introduced painlessly in chapter four, and later sections deal with financial functions, standard forms, invoicing and stock control, financial planning and mathematical functions. There is also a useful little chapter on using Abacus with a printer, and another explaining the type of information which can be exchanged between the Psion packages, including the way in which Abacus can receive information from Easel and Archive and save data for use by Easel, Archive and Quill.

At £8.95, I would strongly recommend the book to anyone who wishes to get to grips with Abacus but finds the prospect of wrestling with formulae daunting.

Déja Vu

Basic Programming on the QL by Neil Cryer and Pat Cryer is exactly what it says — another introduction to SuperBasic on the QL. The competition here is rapidly hotting up, and this attempt is a bit pricey at £7.95, especially as I can't help feeling that the subject has already been adequately treated elsewhere.

However, this version has much to recommend it. It is well-ordered and readable and makes every effort to avoid jargon and technical language. It also adopts a 'teaching by doing' approach which means you should have a QL to hand in order to try out the 'activities' as you go along. The activities are exercises to practice and consolidate the techniques just learned. The book is also very lively, with lots of illustrations (screen shots, diagrams, listings) to help make its points.

It should be good for newcomers, starting off with an introduction to programming in general before it moves on to the features of SuperBasic. It covers sound, graphics, colour, tables, function strings, files, and, of course, structured programming, ending with a games program of some originality for you to key in. There is also a useful glossary of BASIC terms.

Brisk Trip

Quick QL Machine Language would appear to be a contradiction in terms, but that is the title of Alan Giles' book, published by Melbourne House. It is a brisk, workmanlike look at the QL's 68000 instruction set, including listings for an assembler and a disassembler, and if it's quick, that's because it assumes the reader already knows about machine code and understands such things as addressing modes, registers, data buses and so on.

The first half of the book deals with the 68000 instruction set, including the addressing modes, commands, initialization and control routines for a disassembler, logical and arithmetic operations, shifts, rotates and so on. The second half is devoted to appendices covering assembler mnemonics, QL ROM version names, function and procedure names, and the two mammoth listings.

If you have programmed in machine code before and want to be able to do so on the QL, this could be for you. Otherwise you will need to learn how to use machine code before attempting such advanced material.

TALENT SPOT

Here's a chance for all you budding programmers to design a short program, AND get it printed in QL User magazine AND (possibly) win some exciting Talent Computer Systems' software.

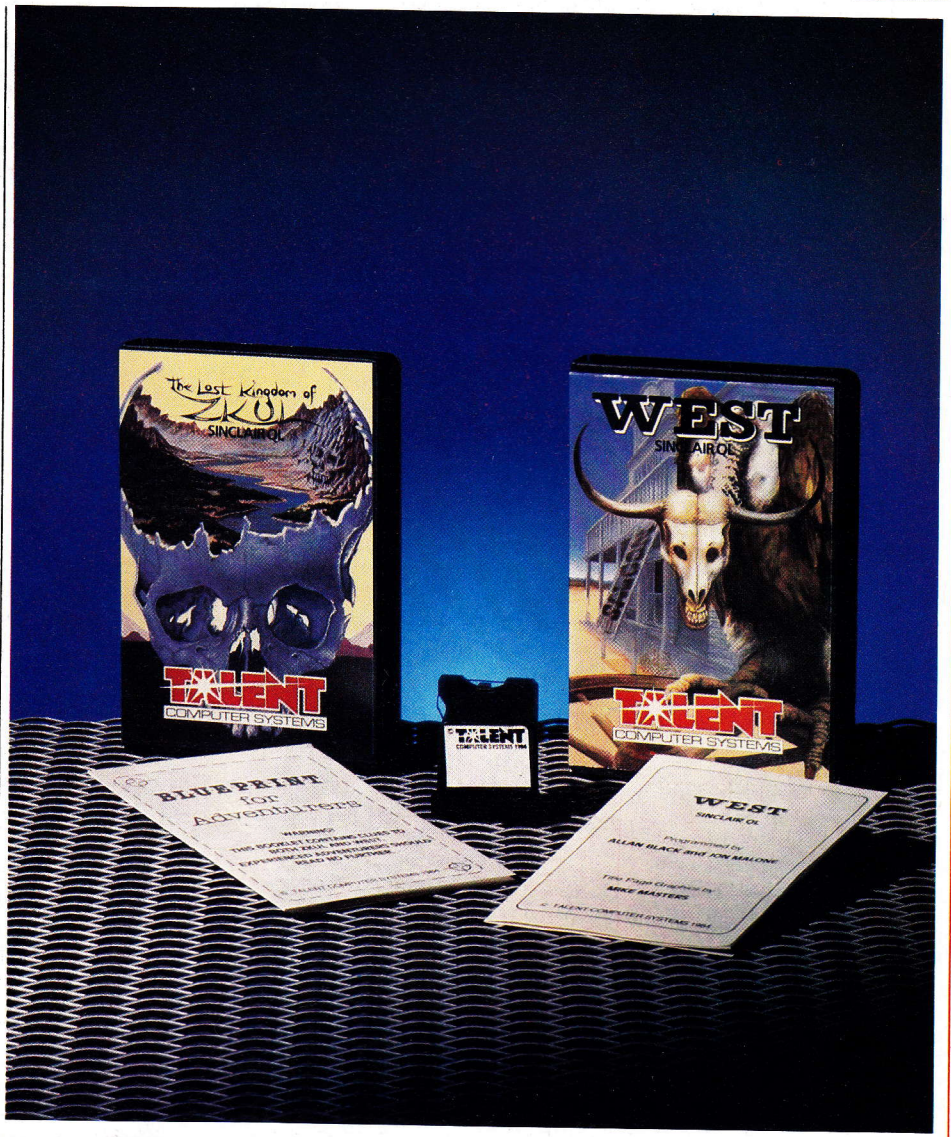


PHOTO BY TERRY BEDDIS

HOW TO ENTER: Make sure you read this section carefully so that your entry is not disqualified.

Entries must be submitted on microdrive *only* (these will be returned same day, as soon as we've made a copy of the program).

Any accompanying information *must* appear as a set of REMark statements within the program – we will not read any covering note, so *SEND THE MICRODRIVE ONLY!*

It may sound obvious, but don't forget to include your name, address and telephone number as one of the REMark lines.

THE PROGRAM: Imagine you have just written an amazing graphic adventure program. All that's left is to come up with an interesting initial loading screen; which is all you have to do to enter this competition.

Things are never that simple, however, and this is no exception. Programs must contain no more than 25 lines (excluding REMark lines), each line containing a maximum of 50 characters (including spaces) – we will be checking these points carefully, so make sure your program is within the limits.

SEND THE DRIVE TO:
Talent Spot Competition,
QL User,
Priory Court,
30-32 Farringdon Lane,
London EC1R 3AU.

CLOSING DATE: All entries must be received on or before 31st March, 1985.

JUDGING: All programs will be checked against the above rules and those that qualify will then be assessed solely on the basis of the screen produced when run.

Programming technique will not be part of the judging, though in the event of more than ten entries being considered potential winners, the ten using the fewest lines will win the prizes.

THE PRIZES: Talent Computer Systems have kindly donated ten of each of their two superb QL adventures ('The lost kingdon of ZKUL' and 'WEST'), for the ten winners.

In addition, each winning program will be printed in a subsequent edition of QL User.

Here's what we said about these two games:

West places you exactly where you'd expect, surrounded by vultures, tumbleweed and 'injuns'. Such a locale has great advantages in a text-only adventure, since we all carry in our heads a much more comprehensive library of images for cowboy country than for any sword and sorcery setting.

The basics of the adventure are well-regulated. The objects are scattered before every new game, but – if you're killed – sensibly left by the body where your reincarnated self can find them again. The locations are numerous and subject to random visitations from tumbleweed, rattlers, bad-tempered bank robbers and your trusty but fickle steed.

If reviewer addiction is anything to go by (and it should be) then *West* will please, not just adventure addicts, but anyone who ever frowned into a puzzle of any sort.

Zkul sends you out into a landscape of wizards, dungeons and axe-wielding dwarves in pursuit of hidden treasure.

There are a bewildering number of settings, some vexing logic tests, a full cast, and an entirely satisfactory store of treasures.

NOTE: Judging will be performed by the Editor of QL User, whose decision is final. No correspondence about the results will be entered into.

No employees of EMAP or associate companies may enter this competition.

Write programs to unlock the multitasking power of your QL!

Use Metacomco's much acclaimed ***Development Kits*** *to create the software for your QL: Use* ***Assembler*** *for its speed, and for access to the QL's many features; use* ***BCPL*** *for systems programming; writing games, utilities, and applications packages; use* ***LISP*** *for manipulating data structures, and for exploring the world of artificial intelligence.*

RELEASE THE POWER OF THE QL

ASSEMBLER £39.95 INC. VAT + P&P

> Standard Motorola 68000 mnemonics.
> English error messages.
> Produces code which can be EXECed, and run as a concurrent job.
> External references allow linkage to high level languages.
> Integral linker allows assembler programs to be linked together.
> Macro expansions.
> Position independent, absolute or relocatable code can be produced.
> Conditional assembly.
> Large range of directives.
> Fully formatted listings.
> 30 character variable names and 32-bit values.

ENHANCED!
NOW INCLUDES ASSEMBLER LINKER*
AND NEW LOW PRICE £39.95

BCPL £59.95 INC. VAT + P&P

> True compiler.
> Multiple BCPL programs can run concurrently.
> Full runtime library includes interfaces to QDOS graphics, file operations, window handling, etc.
> Exception handling provides diagnostics for debugging.
> Link loader allows linkage of separately compiled segments.
> Assembler and BCPL modules can be linked together.
> 32-bit variables use the full QL address space.

LISP £59.95 INC. VAT + P&P

> Full support of QL features including window graphics, and screen handling.
> Compatible with Acornsoft Lisp for the BBC micro.
> Interpreter.
> Turtle graphics.
> Structure editor allows alteration of data structures.
> Prettyprinter displays programs in structured format.
> Tracer to aid in debugging.
> Garbage collector automatically recovers spare memory space.
> 28-bit integers and 250-character names.

Each language is supplied as a complete Development Kit which includes the software (on a Microdrive Cartridge), Metacomco's popular full screen editor, and user manual.

The Metacomco Development Kits are available from HMV, Menzies and many other leading retailers, or direct from Metacomco.

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

*Existing registered users can obtain an upgrade. Contact Metacomco for details.
QL & QDOS are the trade names of Sinclair Research Ltd.

NAME _____
ADDRESS _____
_____
POSTCODE _____
TEL. NO. _____
SIGNATURE _____

Phone today, or post this coupon to: Metacomco, 26 Portland Square, Bristol BS2 8RZ.
Please send me:
The Assembler Development Kit at £39.95 _____
The BCPL Development Kit at £59.95 _____
The LISP Development Kit at £59.95 _____
I enclose a cheque for £ _____ or debit my ACCESS/VISA Account No.

Card expiry date: _____

Price includes VAT, postage and packing UK mainland only. Delivery allow up to 28 days. QL4

QL PLUS IN PROFILE

Mary Sargent sizes up a small company with a product that spans several applications.

"1985 is going to be a fascinating year. The QL is going to sell in large numbers, because it will be a very complete, very portable machine. It represents a milestone in computer hardware design, and although nobody really knows how big this market is going to be, it is certainly measured in millions." So says the managing director of PCML Ltd, a leasing and brokerage firm operating in the IBM mainframe market, and not, you might think, the most immediately obvious person to expound on the virtues of Sir Clive's problem child. In fact, so convinced is John Fuller that the QL is the small-business machine of the future that he and his partner, Derek Batey, have set up a new company, called, appropriately, QL Plus, in order to design add-ons exclusively for the machine.

It's an interesting venture for a company with a successful business which has little to do with home micros, and involves a demonstration of faith almost worthy of the master himself. QL Plus was set up in October 1984, and by November had a prototype add-on in sufficiently good working order to be demonstrated at the Compec Show at Olympia. It aroused a great deal of interest, and led to discussions on marketing and distribution with first Prism, and then, more importantly, Sinclair Research. With this sort of encouragement, it would seem that QL Plus has something rather more interesting than fine phrases to offer.

Wild Card

The first product is a CPM card for the QL which will give access to the huge range of software which the machine currently lacks and desperately needs. This was one of the factors in the decision to concentrate on the QL. Identifying a specific need in the micro-market is vital to a small business intent on becoming leaders in the peripherals field, and since it is likely to be many months before the software backup is available in anything like the quantity or quality needed, access to CPM seemed a logical starting point.

In its simplest form, the card becomes a 64K parallel printer spooler for the QL, dispensing with the need for a dedicated printer buffer and ensuring a continued useful life for the add-on once specific QL software becomes available in quantity. In its more sophisticated application, the peripheral has the capability to become an intelligent floppy disk controller, courtesy of a piggy-back board on the card, again demonstrating a talent for life after CPM. This kind of ingenuity is likely to be the hallmark of QL Plus products. The designer responsible is Robert Harvey, who joined the company in July 1984 as "product development executive", a title which covers a multitude of sins, from dreaming up an add-on in the first place, to soldering the prototype together. He was brought up with the Z80 machines, which explains why the CPM card is in fact a Z80 computer plugged in to the QL expansion socket, but he is unperturbed about working with the 68008 processor. The levels of design difficulty vary with the type of add-on under development, but there have been no insuperable problems. So does designing add-ons for the QL involve a quantum leap in technique?

"Not really," says Robert. "There are a number of things to watch for. One of them is that the power supply can hardly cope with some add-ons, so you have to be careful to design the peripheral so that it uses very little power. The other problem is that it runs much faster than some other micros (about 7MHZ, as opposed to the Z80's 4MHZ) so your design can't be as ad-hoc as, say, something for the Spectrum. You have to take into account that certain things have got to be close to the bus. Design is more critical than it would be with other machines, but the CPM card was very easy to interface to the QL because everything you could possibly need was there. Sinclair have in fact thought about expansion in terms of lots of add-ons."

Bad Rapport

What Sinclair have not thought about, however, is providing sufficient ports for the potential add-ons to plug into. "There is only one expansion slot and nobody seems to be in the business of producing multiple expansion connectors, apart from Quest, who have produced one for their own products which doesn't take standard Euro-cards. It's no use to us." The CPM card is, in Robert's opinion, the most obvious extension to the QL. "People will want floppy disks and probably serial ports. The CPM card gives us a certain amount of play as to what we add on, but it's essentially a stop-gap until expansion boxes come along." This may mean until QL Plus finds time to produce one, although since John Fuller describes his company as "Under capitalised, understaffed and over-loaded" it may be some time before that is possible.

Another thing Sinclair did not seem to have considered was the possibility of co-operating with small companies working to realise the potential of the QL. Initially, there were difficulties in contacting the right people in the Sinclair organisation. For example, an attempt to get a replacement QL for the company's one and only machine which had developed problems within days of the Compec Exhibition resulted in several frustrating hours being blocked by one receptionist after another until a helpful journalist

suggested that John Fuller try Sinclair's Press Office. A working computer was then quickly forthcoming, and Cinderella did get to the Ball, or in this case, the Show, after all, but it was a nerve-wracking incident. An earlier request for information on the QL's memory map elicited nothing from Sinclair Research but a name and address. That, as it turned out, was the most helpful information QL Plus could have, but at the time, it was discouraging to meet so little interest from the manufacturers of the machine about which John Fuller was so enthusiastic.

The name was Tony Tebby, designer of Q-DOS and great white guru of matters QL-ish, and it was he who supplied information and advice on the memory map and operating subroutines without which the CPM card's hardware would have been difficult, and the software impossible, to design. In fact, says Robert Harvey, "it was very well laid out; the addresses you could use, information was given on how to make the interface work, and what signals you needed which were different from the standard 68000 signals." Tony Tebby is now associated with QL Plus in a consultant capacity, and has been involved in the planning stages of future products.

Further RAMifications

These include a RAM card, designed to increase the QL's memory banks. But hasn't the beast already got more memory than the average computer can boast in a life-time?" "The problem," says Robert Harvey, "with any machine with any amount of memory is that software producers always produce software which fills it up. They say, all right, we've 90K inside the QL, let's use it to make a better program. And then you lose out on data storage for those programs. With Abacus, for example, you only have about 8K left." He intends that the RAM expansion will allocate memory for disk drives, both the floppy and micro-drive variety, but there will still be a substantial increase in the memory available to the user. "Everyone always needs more memory!"

The design of the RAM card was not quite so straightforward. "It's a slightly different kettle of fish from the CPM card, as you need to understand the 68008 chip in much more detail. It's not just a simple interface, because you're adding Random Access Memory and you have to talk about refreshing and that side of things. So that's a bit harder, but again, not that hard." Whereas the CPM card communicates with the QL and is a micro-processor in its own right, the RAM card is conceived as being much more an integral part of Sinclair's machine. One of its virtues will be the support it gives the micro-drives in terms of short-period storage, reducing wear and tear on those rather fragile cartridges, and speeding up the operation of the whole computer.

Muddy Critique

Surprisingly Robert Harvey has few complaints about the microdrives, which have been responsible for a good proportion of criticism of the QL. A lot of it, he considers, may have been ill-informed sensationalism, obscuring the fact that few people who regularly use the microdrives complain. "They serve their purpose. In terms of the machine, they provide cheap, mass storage and almost everybody would prefer them over cassettes, even if they have to back-up their cartridges, or rotate through a number of cartridges to even out the wear." John Fuller agrees, "Sinclair will solve the microdrives' problems. Good drives and good cartridges give no trouble. It's a matter of quality control, not a basic design flaw, which has caused the bad publicity."

There are, however, rather more serious repercussions of the bad publicity than the microdrives' reputation, as QL Plus has discovered. When the CPM card was ready for demonstration, the software division of W. H. Smith was extremely interested, not to say excited, about its possibilities. A meeting was set up, the necessary executives were duly impressed and asking whether it could be available by Christmas (1984), but in the event, nothing was finalised, because Smiths remain cautious in their long-term commitment to the QL. Clearly, some of the mud slung with such abandon when the QL emerged was sticky.

The detailed talks with Prism, the major distributor of Sinclair products in the UK, which were encouraging the company in November, are in abeyance due to Prism's current financial difficulties. That may well not matter, since Sinclair Research, having established that QL Plus were indeed worth talking to, is now taking an active interest in letting Sinclair dealers know of the CPM card's existence. So it's happy endings all round, and another innovative British product helping the balance of payments and the unemployment statistics, OK? Well, not quite. There is one giant snag, and it has nothing to do with British shortcomings at all.

Transatlantic Sting

CPM software was invented back in the dark ages by an American company, Digital Research. And Digital Research require a license fee for every unit which uses CPM software. Furthermore, it is necessary to purchase these licences in advance of sales made, and at a price which does not become cost effective until the order is for a minimum of 20,000 licences. At that level, the cost is about 7 dollars per licence, and before that, each licence costs anything up to 15 dollars. Fuller estimates that QL Plus might have to commit something in the region of 150,000 unrecoverable dollars, without any certain information on what the market for the CPM card is likely to be. Clearly, this kind of up-front money is not sitting in his petty-cash box, and if he is to ask for access to the bank's resources, he has to fan his bank-manager's fevered brow with some full order books. If he is to fill order books, he must have a large number of working units available. If he is to produce the working units, he needs the CPM licences and if he is to buy the CPM licences, he needs the bank's backing, which brings the whole thing neatly back to full order books again.

Over The Hill

Attempts to negotiate on the problem have hit a language barrier — Digital Research have been slow to understand the word *compromise*. John Fuller is frustrated. "I've pointed out to them that we're opening up a whole new market for their product they could never have expected. I've also told them that we need the price concessions on small numbers of licences, in order to establish the product. It's no use. They have their structure, and that's the way they've always operated. They can't see anyone else's point of view."

It is a bad case of everyone wanting their slice of the cake before it's baked, and it's particularly galling that it should involve a foreign company which has not been noted for its innovative contribution to microtechnology in recent years. However, all is by no means lost.

Digital Research is now showing signs of granting a price concession for a realistic number of licences and Sinclair's chats with their dealers are already bringing in the first orders. Europe in particular is enthusiastic, and John Fuller is sufficiently optimistic about the CPM card's future to have set up a production unit in Northern Ireland. This is partially funded by the Local Enterprise Development Unit, a government-backed body whose brief is to encourage industry in the area and involves a link-up with a firm already operating in Northern Ireland, Circuit and Systems Design (CSD). It is hoped to start production of the CPM cards early in 1985. As John Fuller said, it's likely to be a fascinating year.

CLASSIFIED

GARDENING PLANS?

Let our QL programs help you.

Shrub bank and planner with 5 programmes £11.95 inc. cartridge and p&p.

House plant bank and plant planner with 5 programs and 180 types of plants £11.95 inc. cartridge and p&p.

Shrub bank and planner £11.95 ☐ House plant bank and planner £11.95 ☐

I enclose cheque for £.....

Name .....

Address .....

Postcode.....For details or payment by credit card phone: 097 423 223

To: Toby Hodd, SuperPlantSoftware, Llangeitho, Tregaron, Wales SY25 6QG

RODENT (QL) SOFTWARE

ADVENTURE WRITER – Design adventures easily! Up to 5 artificially intelligent characters, 24 objects and 32 rooms: chain data files to create multi-level adventures. Advanced language interpretation. Help routines plus full documentation – **£19.95.**

QL ARTIST – Create marvellous pictures: design picture elements, save them on microdrive. Enlargements, stretches, mirror images and rotations can be done with ease. Combine them on screen with each other and with intricate line drawings to create stunning effects. Help option and full documentation. – **£12.95.**

GAMES 1 – Basic Chess, Fruit Machine, Orbiter, Mars Lander, Laserace, Mushroomaze and Eliza. – **All for £8.50.**

GAMES 2 – Snooker Manager (plus save option), Glidepath, Bombshell, Mazeman, El-Dorado, Expert and Nim. – **All for £9.00.**

£5 reduction if microdrive supplied.

Rodent Software, 3 Brook End Drive, Henley-in-Arden, Solihull, West Midlands, B95 5JA.

JOYSTICKS FOR QL

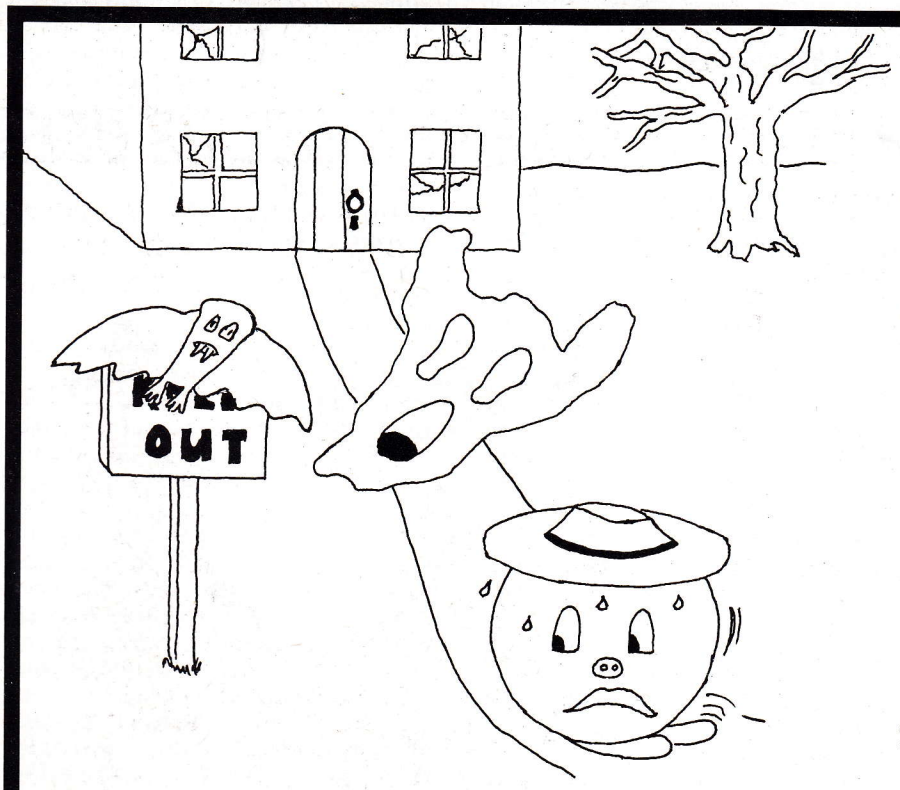
Plugs straight into control port 1 or 2.

NEEDS NO INTERFACE OR ADAPTER – £7.99 each or £15 for two. Add 50p p&p.

Two joysticks plus CONCENTRATION (an addictive 1-4 player game) on microdrive cartridge for only £23.95 inclusive. *Game only: £10.95 inclusive.*

Send to: COMPUTER SUPPLIES
146 Church Road, Boston, Lincs. PE21 0JX.

AD INDEX



"Hungry Harry in the Haunted House"

Copyright * Snowsoft * 1985

Duck from swooping bats, avoid the ghosts, eat the garlic, then kill Dracular before the coffin closes.

* Multilevel Pacman type arcade game.
* Fast machine code graphics.
* Large multi-coloured sprites.

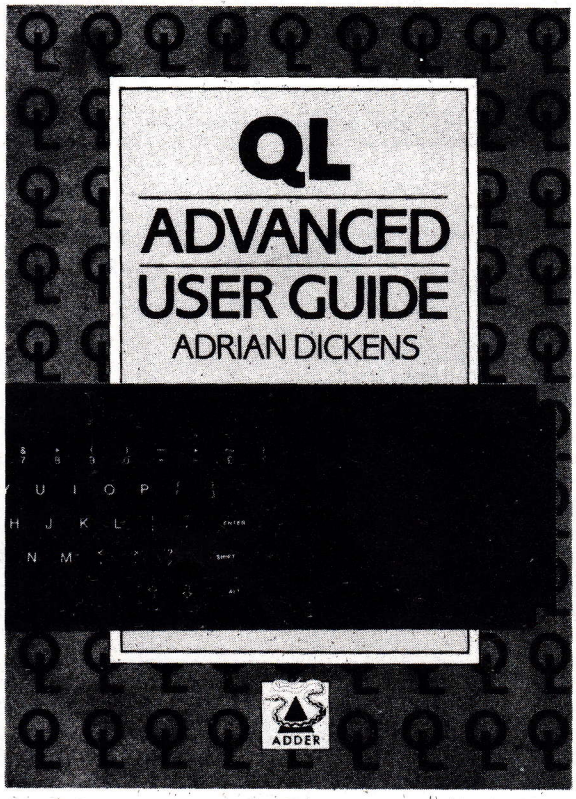
£12.00 (inc. p&p)

Cheques/postal orders to
* Snowsoft * and send to:
6 Bousfield Crescent
Newton Aycliffe
Co. Durham
Tel: (0325) 312417

THE ASSEMBLER

The ADDER 68K Assembler is a professional, fully integrated editor/MACRO assembler development package for the QL. Providing a quick turn-around on the *editing, assembling, debugging* cycle, it assembles Motorola format source files to produce multi-tasking programs, additional Super BASIC facilities plus much more. Features include:

- ★ FULL 68000 MACRO Assembler
- ★ Integrated full screen editor
- ★ Conditional assembly
- ★ External library file inclusion
- ★ Complete error checking diagnostics
- ★ Comprehensive manual
- ★ MACRO ASSEMBLER only £29.95*



THE BOOK

The QL Advanced User Guide (£12.95*) has been written by Adrian Dickens in collaboration with Tony Tebby (QDOS System designer). It is the complete guide to QDOS and the Sinclair QL, covering multi-tasking, transient programs, resident procedures, heaps and stacks, traps and utilities, 68008 assembler programming plus much more. All of these features are illustrated by practical examples, and the powerful QDOS experimentor program allows many facilities to be tried out from BASIC. All of the programs from the book are available on a microdrive cartridge which can be purchased with the book (£9.95*).

ORDER FORM

☐ Assembler development package	**£34.95** incl. VAT and p&p
☐ QL Advanced User Guide Book	**£14.45** incl. p&p (no VAT)
☐ QL A.U.G. programs from book	**£11.95** incl. VAT and p&p
☐ **OFFER** — ABOVE 3 items	£55.95 ALL INCLUSIVE
☐ Blank Microdrive Cartridges	£4.95 incl. VAT

I enclose cheque/PO for £.....

Name .....

Address.....

.....

City..... Postcode .....

Send to: **ADDER PUBLISHING LIMITED, P.O. Box 148, Cambridge CB1 2EQ**

Dealer enquiries welcome

QL and QDOS are trademarks of Sinclair Research Limited

** Price excluding VAT and p&p*

ADDER

QL/3/85

QL BARGAINS

Prices include VAT and carriage

QL COMPUTER — ONLY £379

Psion Chess	£16	4 Microdrive carts	£19
Transform box for 20 cartridges			£6
10 Microdrive carts + Transform box			£49
Serial cable (S)	£11	Parallel interface (P)	£35

PRINTERS

Brother HR5 (S)	£149	Brother M1009 (S&P)	£178
Shinwa CPA80P (P)	£219	Mannesman Tally MT80	219
Epson RX80FT (P)	£254		
Taxan KP810 (P)	£299	Cannon PW1080A (P)	£319
Smith Corona TP-1 (S)	£209	Daisy Step 2000 (P)	£264

MONITORS

Microvitic CUB 1451/DQ3	£259	Prism QL14	£199
Philips Hi-res monochrome			£87

MODEMS

QCON Interface with microdrive software	£76
Q MOD Modem with full duplex operation	£76
Q CALL Auto dial answer unit	£48
Q CON + Q MOD	ONLY £149

DISK DRIVES

Single 3.5in. 720K DS 80T-DRIVE with interface to QL	£217
Single 3.5in. 720K DS 80T-DRIVE (second drive)	£182

SOFTWARE

Please call for latest prices

Metacomco ASSEMBLER — BCPL — LISP — CASH TRADER

STRONG COMPUTER SYSTEMS

Bryn Cottage, Peniel, Carmarthen, Dyfed, SA32 7DJ.
Tel: (0267) 231246 for assistance anytime!!!

QL COMMPAK

MULTI STANDARD ACOUSTIC MODEM AND PRINTER CONTROLLER ONLY £140 INC.

Plugs into SER2 port of QL using QL Serial Cable (not supplied). Accepts phone handset for quick and simple connection.

Modem Speeds	— 300/300, 1200/75 (PRESTEL), 1200/1200
Standards	— Bell 103/113, CCITT V21, CCITT V23
300 Baud Modes	— Originate or Answer
1200 Baud Modes	— Standard or Equalized (for noisy lines)

On board Microcomuter controls speed conversion, buffering and Modem operation. EPROM firmware allows for future developments. In 1200/1200 Mode, Micro uses 'Packet' techniques to avoid collisions and allows fast data transfer between COMMPAK Users.

Printer mode provides Serial to Centronics conversion. uses standard printer cable (available at £15 inc.). Modem 'Log' function copies 300 Baud line data to printer.

QL terminal emulation software package, available free, if micro drive cartridge supplied with order. Otherwise £5.00 inc.

Make cheques/POs payable to COMMPAK DATA. Prices include VAT and p&p. Allow 28 days for delivery.

COMMPAK DATA

13 Beechwood Road, Uplands, Swansea

• DIY Assembler •

The first of three articles describing a practical assembler, written entirely in Sup-erBasic by Giles Todd.

This type-it-yourself assembler is called QSNAIL (as a reflection on its rather less than lightning speed of execution) and will correctly assemble all 68008 instructions as well as support the assembler directives ORG, END, EQU, DC and DS. If you have the patience, it can be used for serious work and has already been used to develop amongst other things a machine code debugger.

Free And Easy

So much for QSNAIL's credentials – how do you use it? The first thing that any assembler needs is a source program and this is created using the Quill. The source layout can be free format – I use tabs to separate the label, mnemonic, operand and comment fields. The assembler expects you to use Motorola standard notation (including the requirement that mnemonics should be in upper case letters) with some minor changes:

1) Labels and symbols must be followed by a colon (:) character when first defined and only the first eight characters of a label or symbol are significant.
   eg LABEL:
2) Comment fields must be preceded by an asterisk (*) character.
3) The expression evaluator is very limited – arithmetic expressions must be surrounded by square brackets and only the binary '+' and '–' operations are supported. The use of '*' to signify the current value of the program counter is also supported.
   eg HERE: EQU [*–LABLE]
4) When using the DC.B directive, ASCII strings must be specified with each character surrounded by single quotes and separated by commas.
   eg STRING: DC.B 'A', 'S', 'C', 'I', 'I'
   To include spaces, use EQU to set a symbol to 32 and then use the symbol wherever you want a space in the string.

When complete, the source program should be PRINTed (not SAVEd although this can be done as well if desired) to a microdrive file. There should be no preamble of printer control codes and any page headers or footers should be preceded by a '*' character so that the assembler can treat them as comments. The default PRINTER_DAT file supplied with the Quill works fine. NB – if you want to re-edit your source file at a later stage, you will have to IMPORT it to the Quill – it will not load unless you SAVEd it as well as PRINTing it to a file.

Having prepared the source program, the next step is to run the assembler. When it has loaded, it will first prompt for an input filename and you should enter the name of the PRINTed Quill file you have just prepared. Then it will prompt for an output filename – this file will be the destination for your assembled machine code. Finally it will ask you where you want the listing file to be sent. Your input should be a valid QDOS device name (eg 'scr_' or 'ser1') or a microdrive filename (handy if you haven't got a printer). The assembler will now get on with its business.

The first pass is mainly concerned with setting up the symbol table (currently limited to a maximum of 200 entries – this can be changed by modifying the value of 'max_symbols' which is set at the beginning of the procedure 'initialise_arrays') so that forward references can be resolved during the second pass. During the first pass, each line is read from the source file, examined to see how many bytes are required for the instruction and, if there is a label, a symbol table entry is created to locate it.

Deleting Errors

The number of errors are reported as they occur and at the end of the first pass, if any have been detected, and the program stops. If no errors are detected, the program proceeds onto the second pass where the completed symbol table is used to create the assembled object code.

Unfortunately, no address information is stored in the object file. If, as you should be, you are writing position independent code, this does not cause great problems but the object file needs to be loaded manually. First, reserve some memory with the 'respr' statement and then use 'lbytes' to load the object file. If your routine is intended to be a resident procedure, simply use 'sbytes' to save it back to file with the correct load address. If it is intended to be a multi-tasking job then use 'sexec'.

Although the assembler as it stands at the moment works reliably and correctly (albeit slowly), there are a number of enhancements which could (and probably should) be made. Most obviously, in the procedure 'pseudo_op' you will see that three assembler directives (SET, REG and DCB) are included but only return the message 'not implemented'. In practice, I have found that they are unnecessary for most program development.

The prime area for enhancement is the second pass main loop. At the moment, much of what is done in the first pass in terms of identifying the instruction type and addressing modes is duplicated in the second pass. If instead, the first pass loop wrote this information to a temporary file to be used in the second pass, the second pass would probably run more quickly.

Yet another area for enhancement is the table searching functions and procedures 'find_symbol', 'add_symbol' and 'mnemonic'. At the moment, they simply do a sequential search through the symbol table and the mnemonic list. I have made a small concession towards optimising for speed by ordering the array 'mnemonic$' in such a way that the most frequently used instructions appear first. However, if the program were altered to use a binary search or (probably even better) hash coding, there would be an overall improvement in speed.

If anyone makes any improvements in these or other areas (anyone for macros?), we would be very interested to hear from them.

QSNAIL: Procedures and Functions

FuNction hex(h$)...
Converts the hex number in h$ to decimal.
PROCedure screen...
Sets up the initial screen parameters.
FuNction file_prompt$...
Gets the input, output and listing filenames.
PROCedure open_file (f$).
Opens the file with name f$ as channel 5.
FuNction read_line$...
Returns the next non-blank line from the source file.
PROCedure close_file...
Closes the file attached to channel 5.
FuNction next_field$ (1$).
Returns the next field (lable, mnemonic, operand or comment) from the line in 1$.
PROCedure initialise_arrays...
Sets up the constants and arrays used by QSNAIL. The arrays are:

Mnemonic$
mnemonic list

symbol$
symbol table

symbol_address
symbol or label addresses – same index as symbol$

condition$
condition code mnemonics

cond%
condition codes – same index as condition$

word
used for bringing up the object code

shift
table of powers of 2 – precalculated to save time building the object code.

FuNction find_symbol(operand$)...
Searches the symbol table for operand$. If found, returns the symbol's address else

returns 2^33 (used a lot as a 'not found' flag).

PROCedure add_symbol(f$,pc)...
Adds the symbol in f$ with address pc to the symbol table.

FuNction mnemonic(f$)...
Searches the mnemonic list for a match with f$. If found, returns instruction type number (0 to 63 – includes pseudo-ops). If not found returns –1.

FuNction ferror (ferror$,ec)
Prints the error message in ferror$ and returns updated error counter ec.

PROCedure pseudo_op(1$,f$,pc)...
Handles assembler directive processing.

FuNction dec2hex$ (i,flag).
Converts the decimal number into a hex string. If flag=0 then it ensures that the returned hex number is five digits long – handy for 68008 addresses.

FuNction eval(t$)...
Attempts to evaluate t$ whether it be an expression, symbol, label or whatever. Returns the value of the string if successful, otherwise returns 2^33.

PROCedure change_symbol(label$, address)...
Changes the value of a symbol already in the symbol table to address.

FuNction count_operands (operand$)...
Returns the number of operands in operand$.

FuNction first_operand$ (operand$)...
Extracts the first operand from operand$. This and the next function are used to separate the source and destination operands of a 68008 instruction.

FuNction second_operand$ (operand$)...
Extracts the second operand from operand$.

FuNction operand_type(operand$, field$,sd$).
A big one. Returns the addressing mode number (1 to 16) of the supplied operand.

FuNction overhead(type,f$,s$)...
68008 instructions can be from two to ten bytes long. This function returns the overhead in bytes of the addressing mode number in type.

FuNction short_branch(f$, link,operand$).
Tests to see if f$ is a short branch instruction. Returns 1 if it is else returns 0.

FuNction branch (f$,link).
Tests to see if f$ is a long branch instruction. Returns 1 if it is else returns 0.

PROCedure reset_pointer(f$)...
Does a 'rewind' of the source file f$ by closing and reopening the file.

PROCedure print_object(o$)...
Prints the assembled object code to the listing file.

FuNction cvs$(number,length)...
Converts number to a binary number of length bytes in a format suitable for writing to the object file.

PROCedure evaluate(mn$,src$,dest$, link,stype,dtype)...
Actually produces the machine code. Uses the procedures link0 to link55 to generate the binary object code for each instruction type.

FuNction reg(r$,type)...
Returns the three bit register field of a 68008 instruction for the addressing mode in type.

FuNction amode(type)...
Returns the three bit mode field for the addressing mode in type.

PROCedure opcode...
Generates the extension words required for the addressing modes in stype and dtype.

PROCedure pcrel(op$)...
Calculates displacements for program counter relative addressing modes.

PROCedure sizetemp...
Sorts out the two bit size field of a 68008 instruction depending on whether the source instruction has a '.B', '.W', '.L' or no extension.

FuNction inside(s$,f$)...
'TRUE' or 'FALSE' version of 'instr'.

PROCEDURE STATUS...
Generates the object code for instructions involving the status register or the user stack pointer.

PROCedure reglist(temp$,dtype)...
Used in setting up a register list extension word for the MOVEM instruction.

FuNction expression (t$)...
Attempts to evaluate arithmetic expressions. This is the one to look at if you want to improve QSNAIL's expression evaluation capability. NB – this procedure is called from eval and, in its turn, calls eval itself, making it recursive. It hasn't hung up on me yet – touch wood!

PROCedure operr...
General purpose error message for second pass operand errors. If you get this error message, check that both your source and destination addressing modes are legal.

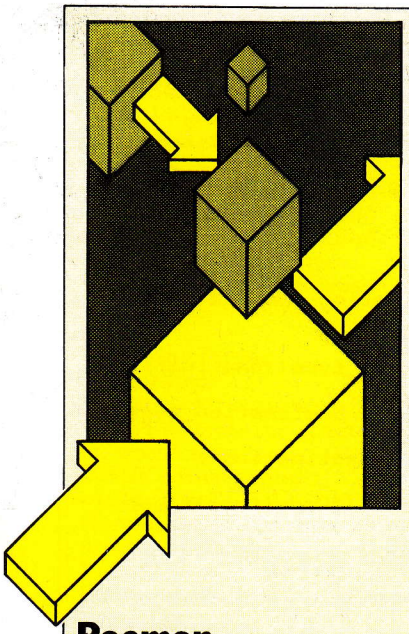
PROCedure tape...
Handy for saving the program.

```
            1 REMark ****        QSNAIL        ****
2 REMark ****       DIY    68008    ****
3 REMark **** TWO PASS ASSEMBLER ****
4 REMark ****    BY GILES TODD    ****
9 REMark ****     QL USER 1985    ****
20 :
30 REMark first pass
40 :
50 initialise_arrays
60 screen
70 filename$=file_prompt$
80 open_file filename$
90 program_counter=0:REMark default pc
100 error_count=0
110 :
120 REMark first pass main execution loop
130 :
140 pass=0
150 PRINT:PRINT "First pass":PRINT
160 REPeat first_pass
170 label$=""
180 REPeat loop
190 line$=read_line$
200 whole_line$=line$
210 field$=next_field$(line$)
220 IF field$<>"" THEN EXIT loop
230 END REPeat loop
250 IF field$="END" THEN EXIT first_pass
260 IF field$(1 TO 1)="*" THEN NEXT first_pass
270 IF field$(LEN(field$) TO)=":" THEN
280    label$=field$(1 TO LEN(field$)-1)
290    IF LEN(label$)>8 THEN
300       label$=label$(1 TO 8)
310    END IF
320    IF find_symbol(label$)<>2^33 THEN
330       error_count=ferror(field$&" multiply def
ined",error_count)
340       NEXT first_pass
350    ELSE
360    add_symbol label$,program_counter
370    IF LEN(line$)>0 THEN field$=next_field$(lin
e$)
380 END IF
390 END IF
400 link=mnemonic(field$)
410 IF link=-1 THEN
420    error_count=ferror(field$&" not found in mn
emonic list",error_count)
430    NEXT first_pass
440 END IF
450 :
460 REMark is it a pseudo-op?
470 :
480 IF link>=56 THEN
490    pseudo_op label$,field$,program_counter
500    NEXT first_pass
510 END IF
520 program_counter=program_counter+2
530 IF LEN(line$)=0 THEN NEXT first_pass
540 IF field$="MOVEQ" THEN NEXT first_pass
550 operand$=next_field$(line$)
560 IF operand$="" OR operand$(1)="*" THEN NEXT fi
rst_pass
570 IF count_operands(operand$)=1 THEN
580    source$=operand$
590    destination$=""
600 END IF
610 IF count_operands(operand$)=0 THEN
620    error_count=ferror(whole_line$&" - illegal
operand",error_count)
630    NEXT first_pass
640 END IF
650 IF count_operands(operand$)=2 THEN
660    source$=first_operand$(operand$)
670    destination$=second_operand$(operand$)
680 END IF
690 :
700 REMark determine source operand type
710 :
720 source_type=operand_type(source$,field$,"s")
730 :
740 IF source_type=0 OR source_type=8 OR source_ty
pe=9 THEN
750    IF short_branch(field$,link,source$)=1 THEN

760       NEXT first_pass
770    END IF
780    IF branch(field$,link)=1 THEN
```

```
790         program_counter=program_counter+2
800         NEXT first_pass
810      END IF
820 END IF
830 program_counter=program_counter+overhead(sourc
e_type,field$,source$)
840 IF destination$="" THEN NEXT first_pass
850 :
860 REMark evaluate destination operand
870 :
880 destination_type=operand_type(destination$,fie
ld$,"d")
890 IF branch(field$,link) THEN program_counter=pr
ogram_counter+2:NEXT first_pass
900 program_counter=program_counter+overhead(desti
nation_type,field$,destination$)
910 END REPeat first_pass
915 old_pc=program_counter
920 IF error_count>0 THEN
930    display_errors error_count
940    close_file filename$
945 CLOSE #channel
950    STOP
960 END IF
970 :
980 REMark second pass
990 :
1000 reset_pointer filename$
1010 program_counter=0:REMark default pc again
1020 length=0
1030 error_count=0
1040 :
1050 REMark second pass main execution loop
1060 :
1070 PRINT \"Second pass"\
1080 pass=1
1090 REPeat second_pass
1100 object$=""
1110 label$=""
1120 REPeat loop
1130 line$=read_line$
1140 whole_line$=line$
1150 field$=next_field$(line$)
1160 IF field$<>"" THEN EXIT loop
1170 END REPeat loop
1180 PRINT #channel,dec2hex$(program_counter,0);
1190 IF field$="END" OR field$="end" THEN EXIT sec
ond_pass
1200 IF field$(1)="*" THEN PRINT #channel,TO 27;wh
ole_line$;:NEXT second_pass
1210 IF field$(LEN(field$) TO)=":" THEN
1220    label$=field$(1 TO LEN(field$)-1)
1230    mnem$=next_field$(line$)
1240    operand$=next_field$(line$)
1250 ELSE
1260    mnem$=field$
1270    operand$=next_field$(line$)
1280 END IF
1290 IF operand$="" THEN operand$="*"
1300 IF operand$(1)="*" THEN
1310    source$=""
1320    destination$=""
1330 ELSE
1340    IF count_operands(operand$)=1 THEN
1350       source$=operand$
1360       destination$=""
1370    ELSE
1380       source$=first_operand$(operand$)
1390       destination$=second_operand$(operand$)
1400    END IF
1410 END IF
1420 link=mnemonic(mnem$)
1430 IF link>=56 THEN
1440    line$=operand$
1450    pseudo_op label$,mnem$,program_counter
1460    print_object(object$)
1470    PRINT #channel,TO 27;whole_line$;
1480    NEXT second_pass
1490 END IF
1500 :
1510 IF link=4 THEN
1520    evaluate mnem$,source$,"",link,source_type
,0
1530    print_object(object$):PRINT #channel,TO 27
;whole_line$;:PRINT #6,object$;
1540    NEXT second_pass
1550 END IF
1560 :
1570 REMark evaluate source operand
1580 :
1590 source_type=operand_type((source$),mnem$,"s")
1600 IF source_type=8 OR source_type=9 THEN
1610    address=find_symbol(source$)
1620    IF address=2^33 THEN
1630       error_count=ferror("Undefined label - "
&whole_line$,error_count)
1640       NEXT second_pass
1650    END IF
1660    source$=address
1670 END IF
1680 :
1690 REMark evaluate destination operand
1700 :
1710 IF destination$<>"" THEN
1720 destination_type=operand_type((destination$),
mnem$,"d")
1730 IF destination_type=8 OR destination_type=9 T
HEN
1740    address=find_symbol(destination$)
1750    IF address=2^33 THEN
1760       error_count=ferror("Undefined label - "
&whole_line$,error_count)
1770       NEXT second_pass
1780    END IF
1790    destination$=address
1800 END IF
1810 ELSE
1820 destination_type=0
1830 END IF
1840 :
1850 REMark evaluate instruction & POKE into memor
y
1860 :
1870 evaluate mnem$,source$,destination$,link,sour
ce_type,destination_type
1880 print_object(object$):PRINT #channel,TO 27;wh
ole_line$;:PRINT #6,object$;
1890 END REPeat second_pass
1892 PRINT #channel,\"First pass pc = ";dec2hex$(o
ld_pc,0)
1895 PRINT #channel,CHR$(12):CLOSE #3
1900 CLOSE #6
1920 IF error_count>0 THEN
1930    display_errors error_count
1940    close_file filename$
1950    STOP
1960 END IF
1970 :
2000 close_file filename$
2010 STOP
2020 :
2030 DEFine FuNction hex(h$)
2040 LOCal i,j,decimal
2050 decimal=0
2060 FOR i=1 TO LEN(h$)
2070 j=-1
2080 REPeat find_hex
2090 j=j+1
2100 IF h$(i)=hex$(j+1) THEN EXIT find_hex
2110 END REPeat find_hex
2120 decimal=decimal+j*16^(LEN(h$)-i)
2130 END FOR i
2140 RETurn decimal
2150 END DEFine hex
2160 :
2170 DEFine PROCedure screen
2180 MODE 4
2185 CLS:CSIZE #1,2,1:PRINT TO 14;"QSNAIL"
2190 CSIZE #1,1,0:PRINT TO 17; 1984 Giles Todd"\\
2200 END DEFine screen
2210 :
2220 DEFine FuNction file_prompt$
2230 LOCal filename$,loop,a$
2250 INPUT "Input filename?  ";filename$
2260 INPUT "Output filename? ";outfile$
2280 INPUT "Send listing to? ";channel$
2310 IF channel$(1 TO 3)=="mdv" THEN DELETE channe
l$:OPEN_NEW #3,channel$:ELSE OPEN #3,channel$
2315 channel=3
2320 RETurn filename$
2330 END DEFine file_prompt$
2340 :
2350 DEFine PROCedure open_file(f$)
2360 OPEN_IN #5,f$
2370 END DEFine open_file
```

Next month: the main routines' source code.



THE PROGS

This is the place to look for readers' QL programs. So, if you've got a computational masterpiece, why not send it in for evaluation. The address is 'The Progs', QL User, Priory Court, 30-32 Farringdon Lane, EC1R 3AU. We pay for everything published.

Pacman

Steve Deary

Despite the absence of REM statements the following Pacman type game, well structured with meaningful data-names, should be reasonably self-explanatory.

The object of the game is to guide a "man" about a maze, consuming everything as you go but at the same time avoiding being eaten by 'ghosts'. The man may be controlled by joystick or cursor control keys. The level of difficulty increases with each screen successfully negotiated. By the fourth screen the maze itself is invisible.

The 'ghosts' speed of movement is controlled by the variable 'handicap' which is set up in line 160. For an easier (or more difficult) game you can try altering the handicap by one or two. Fine tuning of the speed of play can be achieved by adjusting the random number value in line 190 (currently 30).

User defined characters are set up using the procedure at line 1370. Each window can have two character sets (leave the first alone as it contains the standard ASCII characters). Setting up your own characters cannot be done directly in SuperBasic but needs a small machine code routine (just six bytes). When this routine is CALLed three variable parameters must be supplied:

start address = the address where the machine code has been POKEd into memory.

register a0 = the 'channel Id' for the window you are using (65537 for the standard output channel).

register a2 = the address where the table defining your characters has been POKEd into memory.

In this program line 1440 contains the six bytes of machine code, and the character definition table starts at line 1450 and contains two bytes. These tell QDOS that the first character in the set is going to be ASCII character number 128 and that there are six other characters (ie 128 to 134).

Each line 1460 to 1520 represents one character in the new set. Each number in the DATA statement represents the bit pattern for one pixel row of the new character (note: usually only bits two to six inclusive are used).

Therefore, if you wished, you could add more characters to the set simply by:

—inserting extra DATA statements after line 1520

—increasing the number of bytes being POKEd into memory in line 1420.

—tell QDOS about the extra characters by altering line 1450.

```
10 REMark **** QL USER 1985 ****
20 REMark **** PACMAN :Author SN Deary BSc ****
100 initialise
110 REPeat forever
120  score=0:lives=3:screen=1
130  REPeat screens
140   ghosts=screen-1:IF ghosts>2:ghosts=2
150   drawmaze
160   handicap=screen+4
170   REPeat moves
180    move_man
190    IF RND(30)<handicap:move_ghost(RND(ghosts))
200    IF end_of_screen:EXIT moves
210   END REPeat moves
220   IF max=0
230    CLS:AT 5,12:FLASH 1
240    PRINT "Bonus ";screen;"000"
250    score=score+(screen&"000")
260    FLASH 0:FOR i=1  TO 300:i=i:REMark delay
270    screen=screen+1
280   END IF
290   IF lives<1:EXIT screens
300   IF screen>19:EXIT screens
310  END REPeat screens
320  CLS
330  IF screen>19
340   PRINT "You have won the title of Grand Maste
r"
350   PRINT "Too good for this game. I concede!!"
360   STOP
370  END IF
380  AT 5,10:FLASH 1
390  PRINT "G A M E  O V E R"
400  FLASH 0:FOR i=1 TO 1000:i=i:REMark delay
410  IF score>top_score:top_score=score
420 END REPeat forever
430 :
440 DEFine PROCedure move_man
450  INK 2
460  key=KEYROW(1)
470  row=man_row:col=man_col
480  IF key&&2:col=col-1:IF col<0:IF row=5:col=18:
ELSE col=0
490  IF key&&16:col=col+1:IF col>18:IF row=5:col=0
:ELSE col=18
500  IF key&&4:row=row-1:IF row<0:IF col=9:row=10:
ELSE row=0
510  IF key&&128:row=row+1:IF row>10:IF col=9:row=
0:ELSE row=10
520  prize=maze(row,col)
530  IF (prize&&1) OR key=0:RETurn
540  IF prize&&16:gotcha:RETurn
550  AT man_row,man_col*2:PRINT "  ";
560  INK 6:AT row,col*2:PRINT man$;
570  BEEP 200,max
580  maze(row,col)=0
590  IF prize
600   score=score+prize
610   max=max-1:IF max<1:end_of_screen=true
620   AT#4,0,6:PRINT#4,score;
630  END IF
640  man_row=row:man_col=col
650 END DEFine
660 :
670 DEFine PROCedure move_ghost(x)
680  row=ghost_row(x):col=ghost_col(x)
690  not_moved=true
700  IF ABS(man_row-row)<ABS(man_col-col)
710   horizontal:IF not_moved:col=ghost_col(x):ver
tical
720   ELSE vertical:IF not_moved:row=ghost_row(x):
horizontal
730  END IF
740 END DEFine
750 :
760 DEFine PROCedure vertical
770  IF row=man_row:RETurn
780  IF man_row<row
790   row=row-1:IF row>=0:moveit
800  ELSE
810   row=row+1:IF row<=10:moveit
820  END IF
830 END DEFine
840 :
850 DEFine PROCedure horizontal
860  IF col=man_col:RETurn
870  IF man_col<col
880   col=col-1:IF col>=0:moveit
890  ELSE
900   col=col+1:IF col<=18:moveit
910  END IF
920 END DEFine
930 :
940 DEFine PROCedure moveit
950  IF (maze(row,col)&&17):RETurn
960  INK 4:OVER -1
965  not_moved=false
970  AT ghost_row(x),2*ghost_col(x):PRINT ghost$
980  maze(ghost_row(x),ghost_col(x))=maze(ghost_ro
w(x),ghost_col(x))-16
990  AT row,2*col:PRINT ghost$
1000  maze(row,col)=maze(row,col)+16
```

THE PROGS

```
1010  ghost_row(x)=row:ghost_col(x)=col
1020  OVER 0
1030  IF row=man_row AND col=man_col:gotcha
1040 END DEFine
1050 :
1060 DEFine PROCedure gotcha
1070  end_of_screen=true
1080  lives=lives-1
1090  BEEP 30000,1,255,200,4,2
1100 END DEFine
1110 :
1120 DEFine PROCedure initialise
1130  set_up_user_defined_chars
1140  MODE 8
1150  true=1:false=0
1160  top_score=0
1170  CSIZE 2,1
1180  OPEN#3,scr_512x256a0x0
1190  PAPER#3,0:CLS#3
1200  SCALE#3,256,0,0
1210  OPEN#4,scr_456x10a32x14
1220  WINDOW 476,230,23,26
1230  BORDER 5,2
1240  BLOCK#3,24,5,248,26,1
1250  BLOCK#3,24,5,248,251,1
1260  BLOCK#3,10,20,23,131,1
1270  BLOCK#3,10,20,489,131,1
1280  DIM maze(10,18),man$(2),maze$(2,2)
1290  DIM ghost$(2),ghost_row(2),ghost_col(2)
1300  man$=CHR$(131)&CHR$(132)
1310  ghost$=CHR$(128)&CHR$(129)
1320  maze$(0)=CHR$(133)&" "
1330  maze$(1)=CHR$(134)&CHR$(134)
1340  maze$(2)=CHR$(130)&" "
1350 END DEFine
1360 :
1370 DEFine PROCedure set_up_user_defined_chars
1380  start_address=RESPR(100)
1390  register_a0=65537
1400  register_a2=start_address+6
1410  RESTORE 1440
1420  FOR i=0 TO 70:READ byte:POKE start_address+i
,byte
1430  CALL start_address,0,0,255,37,0,0,0,register
_a0,0,register_a2
1440  DATA 32,4,78,67,78,117
1450  DATA 128,6
1460  DATA 4,8,16,44,76,64,84,84,0
1470  DATA 64,32,16,104,100,4,84,84,0
1480  DATA 0,0,0,0,4,0,0,0,0
1490  DATA 124,60,36,60,124,24,24,56,0
1500  DATA 124,120,72,120,192,48,48,56,0
1510  DATA 0,40,16,56,124,124,124,56,0
1520  DATA 40,56,56,124,124,124,124,40,108
1530 END DEFine
1540 :
1550 DEFine PROCedure drawmaze
1560  max=139:end_of_screen=false
1570  RESTORE 1600
1580  READ man_row,man_col
1590  FOR i=0 TO 2:READ ghost_row(i),ghost_col(i)
1600  DATA 5,0,5,8,5,9,5,10
1610  IF screen MOD 4:PAPER 2:ELSE PAPER 1
1620  CLS:CLS#4
1630  RESTORE 1950
1640  PAPER 1:INK 4
1650  FOR row=0 TO 5
1660   opposite_row=10-row
1670   FOR col=0 TO 9
1680    opposite_col=18-col
1690    READ prize:maze(row,col)=prize
1700    maze(row,opposite_col)=prize
1710    maze(opposite_row,col)=prize
1720    maze(opposite_row,opposite_col)=prize
1730    IF prize>1
1740     IF prize=2
1750      chars$=maze$(2)
1760      ELSE IF prize=6:chars$=maze$(0):ELSE cha
rs$=maze$(1)
1770     END IF
1780     AT row,col*2:PRINT chars$;
1790     AT opposite_row,col*2:PRINT chars$;
1800     AT opposite_row,opposite_col*2:PRINT char
s$;
1810     AT row,opposite_col*2:PRINT chars$;
1815    END IF
1820   END FOR col
1830  END FOR row
1840  AT 5,16:OVER -1:INK 4
1850  FOR i=0 TO ghosts:PRINT ghost$;:maze(5,8+i)=
17
1860  AT man_row,2*man_col:OVER 0
1870  maze(man_row,man_col)=0
1880  INK 6:PRINT man$;
1890  key=KEYROW(1)
1900  PRINT #4,"Press any key to start";
1910  a$=INKEY$(-1):CLS#4
1920  AT#4,0,0:PRINT#4,"SCORE:";score;
1930  AT#4,0,13:PRINT#4,"TOP SCORE:";top_score;
1940  AT#4,0,31:PRINT#4,"LIVES:";lives
1950  DATA 2,2,2,2,2,2,2,2,1,2
1960  DATA 6,1,2,1,1,2,1,2,2,2
1970  DATA 2,1,2,2,2,2,2,2,1,1
1980  DATA 1,1,1,2,1,2,1,2,1,8
1990  DATA 1,2,2,2,1,2,2,2,2,2
2000  DATA 2,2,1,2,2,2,1,2,1,1
2010 END DEFine
```

File Probe

Adam Denning

This short program is a very useful utility which dumps out the contents of a microdrive file (or indeed any QL device capable of input) to a specified output device, which may be the screen, a printer or a microdrive file. The contents of the file are printed out in hexadecimal and ASCII, with eight bytes per line, which means that if the output is being directed to the screen it's only really effective in mode 0. The program is written so that it can be used as a multi-tasking job, by EXECing it, or as a piece of machine code to be CALLed from BASIC.

Owners of the BBC Micro will notice that it is identical to that micro's ★DUMP command, which was so useful that a QL equivalent was needed. It uses standard QDOS calls and a little bit of devious programming.

Finally, it should be noted that as the dump is in hex the utility has one very distinct advantage over a straightforward "copy file to device" command – hidden control characters are there for all to see.

NB: 'header_asm can be found in last month's issue.

```
  1                    * A program to dump a named file to a named device as hex and ascii
  2                    * By Adam Denning 17th December 1984
  3                    * Copyright (C) 1984 Adam Denning
  4
  5                              INCLUDE   mdv1_header_asm
156          00000064          DATA      100
157
158 0 00000000 600E             BRA.S     START_P          Ignore standard format code
159 0 00000002 00000000         DC.L      0
160 0 00000006 4AFB             DC.W      $4AFB            Standard format identification
161 0 00000008 0006             DC.W      6                Program name
162 0 0000000A 44554D504552     DC.B      'DUMPER'
163
164 0 00000010 43FA0142 START_P LEA.L     PBLOCK,A1
165 0 00000014 347800C6         MOVE.W    UT_CON,A2
166 0 00000018 4E92             JSR       (A2)
167 0 0000001A 43FA0144 GET_FILE LEA.L    MESSAGE1,A1      Print 1st message
168 0 0000001E 347800D0         MOVE.W    UT_MTEXT,A2
169 0 00000022 4E92             JSR       (A2)
170 0 00000024 7450             MOVEQ     #80,D2           Fetch filename from channel
171 0 00000026 76FF             MOVEQ     #-1,D3
172 0 00000028 43FA014C         LEA.L     BUFFER+2,A1
173 0 0000002C 7002             MOVEQ     #IO_FLINE,D0
174 0 0000002E 4E43             TRAP      #3
175 0 00000030 2F08             MOVE.L    A0,-(A7)         Save console channel ID
176 0 00000032 41FA0140         LEA.L     BUFFER,A0        Get ready for IO_OPEN call by
177 0 00000036 5381             SUBQ.L    #1,D1            converting line fetched to a
178 0 00000038 3081             MOVE.W    D1,(A0)          string, removing LF from count
179 0 0000003A 7601             MOVEQ     #OPEN_INS,D3     Open this file for input
180 0 0000003C 72FF             MOVEQ     #-1,D1
181 0 0000003E 7001             MOVEQ     #IO_OPEN,D0
182 0 00000040 4E42             TRAP      #2
183 0 00000042 4A80             TST.L     D0               Error?
184 0 00000044 670A             BEQ.S     GOT_FILE         No - so continue
185 0 00000046 205F             MOVE.L    (A7)+,A0         Else retreive console channel ID
```

THE PROGS

```
186 0 00000048 347800CC            MOVE.W   UT_ERR,A2          and write requisite error message
187 0 0000004C 4E92                JSR      (A2)               to it. Then try again.
188 0 0000004E 60CA                BRA.S    GET_FILE
189 0 00000050 225F      GOT_FILE  MOVE.L   (A7)+,A1           Put console channel ID in A1
190 0 00000052 2F08                MOVE.L   A0,-(A7)           save file channel ID on stack
191 0 00000054 2049                MOVE.L   A1,A0              make console current channel
192 0 00000056 43FA0116  GET_OUTP  LEA.L    MESSAGE2,A1        Print 2nd message to console
193 0 0000005A 347800D0            MOVE.W   UT_MTEXT,A2
194 0 0000005E 4E92                JSR      (A2)
195 0 00000060 7450                MOVEQ    #80,D2             Get output device specification
196 0 00000062 76FF                MOVEQ    #-1,D3             from console and open it as file
197 0 00000064 43FA0110            LEA.L    BUFFER+2,A1
198 0 00000068 7002                MOVEQ    #IO_FLINE,D0
199 0 0000006A 4E43                TRAP     #3
200 0 0000006C 2F08                MOVE.L   A0,-(A7)           Save console channel ID
201 0 0000006E 41FA0104            LEA.L    BUFFER,A0          Point to start of string
202 0 00000072 5381                SUBQ.L   #1,D1              'Remove' trailing L/F
203 0 00000074 3081                MOVE.W   D1,(A0)            Save count at start of string
204 0 00000076 7602                MOVEQ    #OPEN_NEW,D3
205 0 00000078 72FF                MOVEQ    #-1,D1
206 0 0000007A 7001                MOVEQ    #IO_OPEN,D0
207 0 0000007C 4E42                TRAP     #2
208 0 0000007E 4A80                TST.L    D0                 Error?
209 0 00000080 670A                BEQ.S    GOT_OUTP           No - so continue
210 0 00000082 205F                MOVE.L   (A7)+,A0           Retrieve console channel ID
211 0 00000084 347800CC            MOVE.W   UT_ERR,A2          print requisite error message
212 0 00000088 4E92                JSR      (A2)
213 0 0000008A 60CA                BRA.S    GET_OUTP           and try again
214 0 0000008C 225F      GOT_OUTP  MOVE.L   (A7)+,A1           Swap console and output channel
215 0 0000008E 2F08                MOVE.L   A0,-(A7)           IDs on stack and close the
216 0 00000090 2049                MOVE.L   A1,A0              console device.
217 0 00000092 7002                MOVEQ    #IO_CLOSE,D0
218 0 00000094 4E42                TRAP     #2
219 0 00000096 7800                MOVEQ    #0,D4              D4 is byte counter
220 0 00000098 206F0004  DUMPLOOP  MOVE.L   4(A7),A0          Get input file channel ID
221 0 0000009C 7003                MOVEQ    #IO_FSTRG,D0       and read <8 bytes into buffer
222 0 0000009E 43FA00D4            LEA.L    BUFFER,A1
223 0 000000A2 7408                MOVEQ    #8,D2
224 0 000000A4 76FF                MOVEQ    #-1,D3
225 0 000000A6 4E43                TRAP     #3
226 0 000000A8 2E00                MOVE.L   D0,D7              Save error return in D7
227 0 000000AA 4A41                TST.W    D1                 If bytes got = 0 then leave (eof)
228 0 000C00AC 6750                BEQ.S    JOB_END
229 0 000000AE 5341                SUBQ.W   #1,D1              Get D1 ready for DBRA loop
230 0 000000B0 7C00                MOVEQ    #0,D6
231 0 000000B2 1C01                MOVE.B   D1,D6              and put into D6.L
232 0 000000B4 2057                MOVE.L   (A7),A0            Get output channel ID
233 0 000000B6 3A04                MOVE.W   D4,D5              Print byte counter as 4 hex digits
234 0 000000B8 616C                BSR.S    FOURHEX
235 0 000000BA 6158                BSR.S    SPACES             ...followed by two spaces
236 0 000000BC 6156                BSR.S    SPACES
237 0 000000BE 45FA00B4            LEA.L    BUFFER,A2
238 0 000000C2 2F06                MOVE.L   D6,-(A7)           Save inline counter
239 0 000000C4 1A1A      HEXLOOP   MOVE.B   (A2)+,D5           and print each byte as 2 hex digits
240 0 000000C6 6164                BSR.S    TWOHEX
241 0 000000C8 614A                BSR.S    SPACES             followed by a space
242 0 000000CA 51CEFFF8            DBRA     D6,HEXLOOP
243 0 000000CE 0C6F00070002        CMPI.W   #7,2(A7)           If D6<7 then fill with asterisks
244 0 000000D4 670C                BEQ.S    NOT_END
245 0 000000D6 7C06                MOVEQ    #6,D6              number of asterisks = 6 - old D6
246 0 000000D8 9C97                SUB.L    (A7),D6
247 0 000000DA 6170      STARLOOP  BSR.S    TWOSTARS
248 0 000000DC 6136                BSR.S    SPACES
249 0 000000DE 51CEFFFA            DBRA     D6,STARLOOP
250 0 000000E2 6130      NOT_END   BSR.S    SPACES
251 0 000000E4 2C1F                MOVE.L   (A7)+,D6           Now print bytes as ASCII
252 0 000000E6 45FA008C            LEA.L    BUFFER,A2
253 0 000000EA 121A      ASCLOOP   MOVE.B   (A2)+,D1
254 0 000000EC 612A                BSR.S    ASCOUT
255 0 000000EE 6124                BSR.S    SPACES
256 0 000000F0 51CEFFF8            DBRA     D6,ASCLOOP
257 0 000000F4 720A                MOVEQ    #10,D1             followed by a line feed
258 0 000000F6 614C                BSR.S    OUTCHAR
259 0 000000F8 5044                ADDQ.W   #8,D4              Next line of 8 bytes
260 0 000000FA 4A87                TST.L    D7                 Did IO_FSTRG have an error?
261 0 000000FC 679A                BEQ.S    DUMPLOOP           No; so repeat
262
263 0 000000FE 7002      JOB_END   MOVEQ    #IO_CLOSE,D0       Else close channels and kill job
264 0 00000100 205F                MOVE.L   (A7)+,A0
265 0 00000102 4E42                TRAP     #2
266 0 00000104 205F                MOVE.L   (A7)+,A0
267 0 00000106 7002                MOVEQ    #IO_CLOSE,D0
268 0 00000108 4E42                TRAP     #2
269 0 0000010A 7005                MOVEQ    #MT_FRJOB,D0
270 0 0000010C 72FF                MOVEQ    #-1,D1
271 0 0000010E 4E41                TRAP     #1
272 0 00000110 7000                MOVEQ    #0,D0              In case the code is CALLed
273 0 00000112 4E75                RTS
274
275 0 00000114 7220      SPACES    MOVEQ    #' ',D1            Prints a space
276 0 00000116 602C                BRA.S    OUTCHAR
277
278 0 00000118 0201007F  ASCOUT    ANDI.B   #$7F,D1            Print byte as ASCII, with control
279 0 0000011C 0C010020            CMPI.B   #' ',D1            codes being shown as '.'
280 0 00000120 6C22                BGE.S    OUTCHAR
281 0 00000122 722E                MOVEQ    #'.',D1
282 0 00000124 601E                BRA.S    OUTCHAR
283
284 0 00000126 E04D      FOURHEX   LSR.W    #8,D5              Print D5 as four hex digits
285 0 00000128 6102                BSR.S    TWOHEX
286 0 0000012A 1A04                MOVE.B   D4,D5
287
288 0 0000012C 1205      TWOHEX    MOVE.B   D5,D1              Print D5 as 2 hex digits
289 0 0000012E E809                LSR.B    #4,D1
290 0 00000130 6102                BSR.S    OUTHEX
291 0 00000132 1205                MOVE.B   D5,D1
292
293 0 00000134 0201000F  OUTHEX    ANDI.B   #$F,D1
294 0 00000138 06010030            ADDI.B   #'0',D1
295 0 0000013C 0C01003A            CMPI.B   #':',D1
296 0 00000140 6D02                BLT.S    OUTCHAR
297 0 00000142 5E01                ADDQ.B   #7,D1
298 0 00000144 7005      OUTCHAR   MOVEQ    #IO_SBYTE,D0       Output character in D1
299 0 00000146 76FF                MOVEQ    #-1,D3
300 0 00000148 4E43                TRAP     #3
301 0 0000014A 4E75                RTS
302
303 0 0000014C 722A      TWOSTARS  MOVEQ    #'*',D1            Print two stars
304 0 0000014E 61F4                BSR.S    OUTCHAR
305 0 00000150 722A                MOVEQ    #'*',D1
306 0 00000152 60F0                BRA.S    OUTCHAR
307
308                          * Console device specification
309
310 0 00000154 0000      PBLOCK    DC.W     0                  No border
311 0 00000156 0004                DC.W     4                  black paper green ink
312 0 00000158 01B8                DC.W     440                width
313 0 0000015A 001E                DC.W     30                 height
314 0 0000015C 0024                DC.W     36                 X position
315 0 0000015E 0012                DC.W     18                 Y position
316
317 0 00000160 000B      MESSAGE1  DC.W     11
318 0 00000162 44756D702066        DC.B     'Dump file: ',0
319
320 0 0000016E 0004      MESSAGE2  DC.W     4
321 0 00000170 546F3A20            DC.B     'To:
322
323          00000174    BUFFER    EQU      *
324
325                                END
****** TOTAL ERRORS    0 (line    0)
**** TOTAL WARNINGS    0 (line    0)
     memory usage   12 kbytes
```

All In Order

Here is a short procedure that permits figures of varying magnitude to be printed in a neat tabular form. It should be used to replace "PRINT" statements on a listing.

When the procedure is called, a 'mask' followed by a number or numeric variable should be passed across. The mask will be a string in the form "00000.000". For example:

Command	Output
"0000.000",13.34	0013.340
"£##0.000",–13.34	£ 13.340–

Note: The hash character the mask produces a space, other characters will print.

```
100 a$= "`###00000.###"
110 FOR loop = 1 TO 5
120 PRINT_MASK a$,-13.34^loop:PRINT 13.34^loop
130 END FOR loop
140 STOP
150 DEFine PROCedure PRINT_MASK (mask$ , number$)
160 LOCal loop,l_m,l_n,dec_pos,dec_place,dec_displ
,sign,result$
170 sign = "-" INSTR number$
180 IF sign THEN  number$=number$*(-1)
190 l_m = LEN(mask$)
200 l_n = LEN (number$)
210 dec_place = "." INSTR mask$
220 dec_pos = "." INSTR number$
230 IF NOT dec_place  THEN dec_place = l_m + 1
240 IF NOT dec_pos  THEN dec_pos = l_n+1
250 dec_displ = dec_place - dec_pos
260 result$ = mask$
270 result$(dec_displ+1 TO (dec_displ+l_n)) = numb
er$
280 FOR loop = 1 TO l_m
290 IF result$(loop)="#" THEN result$(loop)=" "
300 END FOR loop
310 IF  sign THEN  result$ = result$&"-"
320 PRINT result$
330 END DEFine
```

QL NEWS

News from the world of Sinclair QL computing.

One year old... and look how we've grown!

When we launched the QL last year, we knew we were starting a revolution.

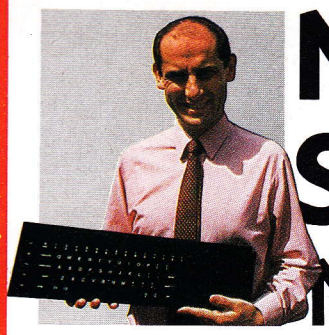
For the first time, the serious computer hobbyist could afford the same power and performance as the professional computer user.

A year later, and the QL is more than a unique computer, it's the heart of a unique system.

And the next 12 months promise even more for QL owners... new software options, extra storage devices, printers, monitors...

Read on, and see how far we've come, and how much further we're going!

Nº1



NIGEL SEARLE

Now it's the quantum leap for QL software and peripherals

Without doubt, the QL was the computer innovation of 1984. Launched to outstanding reviews, it soon gathered thousands of happy owners, and recognition from people like ICL, who have incorporated QL technology and its Microdrives into the new One Per Desk.

The quickest glance at the QL's specification shows what the fuss was all about... 128K RAM, 32-bit processor architecture, 200K built-in mass storage, bundled software, They're features that would normally cost you three or four times as much!

But that's only half the story, because the QL is now the heart of a computer system, with a growing library of software...

As you'll see from these pages, 1985 is the year of the quantum leap for software and peripherals. Already there are no less than five QL languages together with special programs for software developers, a world-beating chess game... and much more on the way!

On the hardware side, there's a special QL monitor to make the most of that high-resolution 512 x 256 pixel display. There are memory expansion boards, Winchester disk drives, printers, and low-cost Microdrive cartridges.

In fact, there's so much going on, we'll be running these regular Newsletters just to keep you in touch!

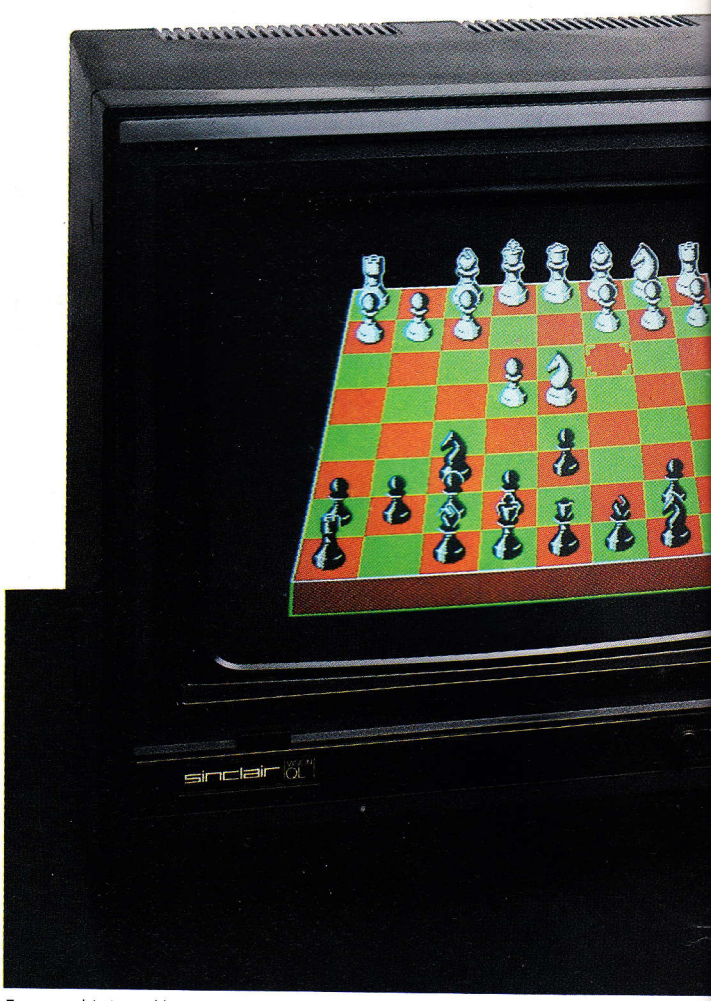
If you already own a QL, the next few pages will give you a taste of the exciting year ahead.

And if you don't . . . take a look at what you're missing. It should be all the persuasion you need!

Now read on...the quantum leap into serious computing starts here.

Nigel Searle

Nigel Searle, Managing Director, Sinclair Research Limited.



From sophisticated business packages to superb animated games... QL software makes the most of the computer's extraordinary specification.

New QL Software

Utilities, languages, games and business packages...with more on the way!

Two things are now certain about QL software. First, there's going to be plenty of it. And second, it's going to set completely new standards for microcomputers...

At the moment, there are well over 100 software programs in development. And the first software releases, shown here, demonstrate how exceptional the best QL software will be.

The QL already has five languages, superb programs for software developers, a top quality accounting package and in QL Chess it has its first game.

QLUB: 10,000 members and growing!

QLUB is the special Users Bureau for Sinclair QL owners. There are now well over 10,000 QLUB members, and membership is growing all the time.

For their annual subscription of £35, QLUB members are enjoying a whole range of information and advisory services, exclusive offers and special discounts.

One of the most important QLUB benefits is the special news magazine, appearing six times a year. The magazine provides a forum for QL owners to exchange views and keep in touch with all the latest developments.

Each issue is packed with updates on QL hardware and software, tips on applying the four QL Programs, and news of how other people are using the QL. QLUB members also receive a range of special discounts, with savings of at least 20% on selected software products.

Current special offers include:
QL Chess for £14.95
QL Toolkit for £19.95
QL Assembler for £31.95
QL Cash Trader for £54.95
Special subscription rates for Personal Computer News and QL User.

The multilingual Sinclair QL

BCPL – a forerunner of C, BCPL has been described as a systems programmer's delight. In the words of QL User, this compiler is a 'brilliant compromise between a high-level language and a low-level systems language'. Whilst not for beginners, this is an essential buy for anyone with a good knowledge of systems programming. Complete with manual.
Available from Metacomco – £59.95. Tel: 0272 428781.

LISP – already well-known for its artificial intelligence applications, LISP is a powerful and versatile language. This is a sophisticated implementation of LISP, by one of its leading exponents, Dr Arthur Norman. This package features full QL graphics, and a full manual is supplied.
Available from Metacomco – £59.95. Tel: 0272 428781.

Pascal – probably the most popular high-level language of all. Pascal is particularly well-suited to structured programming, sophisticated data manipulation and algorithmic problems. Pascal interpreter complete with 87-page manual.
Available from Computer One – £39.95. Tel: 0223 862616.

Forth – this 'new generation' language is proving both popular and easy to learn. The program provides a full implementation of the latest Forth 83 standard with graphics and sound extension.
Available from Computer One – £29.95. Tel: 0223 862616.

APL – the compact mathematics-based interpreted language designed for scientists and mathematicians.

APL keyword interpreter complete with manual.
Available from MicroAPL – £99.95. Tel: 01-622 0395.

Programmer's packs

QL Assembler – two programs operating in tandem. The first is a full-screen editor for creating and altering program files. The second, a Motorola-format compatible 68000 assembler which converts source files written in M68000 assembly language into machine code files which can run on the QL.

Both assembler and editor are written in machine code and can be multi-tasked with SuperBASIC, so you can switch between editor, assembler and SuperBASIC instantly.
Written by GST Computer Systems – £39.95.*

QL Toolkit – a programmer's toolkit with over 70 programs. and extensions to SuperBASIC. Most are linked to SuperBASIC initially and can then be used from commands or from within a program. Enhancements include printer spooling (print a file while running a SuperBASIC program); improved file access (with full random input/output command); job control (allows management of multi-tasking programs including the ability to display, alter priorities, and delete jobs from the QL); and SuperBASIC screen editor.
Written by Q Jump – £24.95.*

World-beating chess!

QL Chess – fresh from its victory at the World Microcomputer Chess Championship. This program sets a completely new standard for games software.

There's a high resolution display, animated 3-D graphics, and 28 levels of play from novice to champion. Features include an openings book of nearly 4000 moves, HINT and TAKEBACK functions that help you learn from your mistakes, and the option to play a human opponent or the computer.
Written by Psion – £19.95.*

Software at work

QL Touch 'n' Go – a unique approach to learning touch-typing skills. The program is designed to give you mastery of the standard QWERTY keyboard in just 24 hours. With practice, you should soon reach 40 words per minute, with over 95% accuracy.
Written by Harcourt – £24.95.*

QL Cash Trader – a unique computerised book-keeping system for small businesses. The program provides a complete course in the principles of accountancy, and goes on to become an essential aid in the day-to-day running of a business. Complete with comprehensive manual.
Written by Accountancy Software of Torquay – £69.95.*

*This title is available from Sinclair Research on 0276 686100, and selected Sinclair stockists nationwide.

Psion trouble-shooting service

All QLUB members can obtain special assistance from Psion on using the QL Quill, Abacus, Archive and Easel programs supplied with the computer. Psion will normally answer any queries within 48 hours.

New QL Hardware
An industry is born

From the moment of its launch, the revolutionary QL attracted massive interest from all quarters.

In one area, the interest quickly turned to action, as high-tech hardware manufacturers realised the immense potential of the QL for vast expansion, for system development and for widespread networking. Already the list of peripherals for the QL is very exciting – and lengthening by the day!

Here, we've covered many of the latest, most important developments.

As more appear, be sure to keep in touch with QL News!

The dedicated Sinclair Vision QL monitor

Once you see the incredible graphics capabilities of the QL you may decide an ordinary TV just can't do them justice.

If that's the case, a high-resolution monitor is needed. (And if you're creating presentation-quality charts, for example, it's quite essential.)

The new Vision QL monitor is specially designed for the computer by Kaga Electronics, with full support from Sinclair Research.

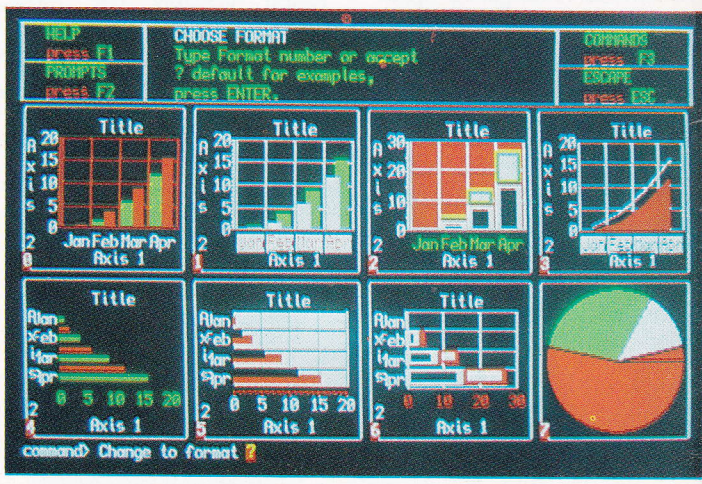
So it exploits the QL's maximum 512 x 256 pixel resolution to the full, with a pin-sharp 85 column display.

It's also specially styled to suit the QL – in looks, and in use. There's a 12" non-glare tube, and etched screen to diffuse reflections.

So the display is bright, sharp, *much* easier to look at . . . and invaluable for those late-night programming sessions!

And like the QL, the Vision monitor is designed with space in mind: it has a compact footprint of just 12½" by 15" – no more than a typical portable typewriter.

It's available from MBS Data Efficiency on 0442 60155 and selected Sinclair stockists.



The QL's superb graphics capabilities – as demonstrated by the Sinclair Vision QL monitor.

Microdrive cartridges. Now only £1.99!

Microdrive cartridges are the QL's own unique storage media. Each stores up to 100K of information, on a cartridge no bigger than a matchbox!

Over 500,000 cartridges are now being used throughout Britain. And QL Microdrives themselves are standard equipment on the new ICL One Per Desk micro.

Now there's more good news for QL enthusiasts: from February 1, the cost of QL Microdrive cartridges are down from £4.95 to £1.99 each!

Sinclair Microdrive cartridges – up to 100K of programs and data on a medium so compact you can pop it into your pocket.

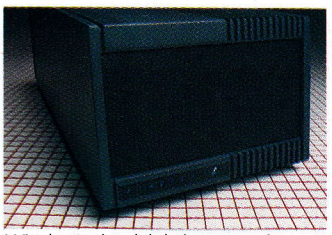
Powerful hard-disk system

For the QL business user, the new Firefly QL Winchester disk will boost the QL's power in one huge leap.

Designed by Quest, it uses CP/M and offers all the benefits of Winchester technology: fast access, reliability, compact size and quiet operation.

With 7.5 Mb storage, the Quest Firefly is ideal for large databases such as stock or customer lists. And at under £1,200, it represents exceptional value for money.

The Firefly will be available very shortly from Quest on 04215 66488.



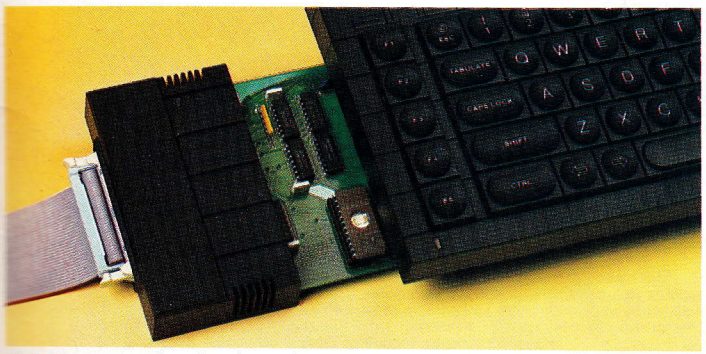
Winchester hard disk drives supplement your QL's built-in mass storage.

Interface options

The QL comes complete with two built-in RS-232C interfaces.

In addition, interfaces for Centronics printers are widely available from manufacturers such as CST, Miracle Systems and Sigma Research . . . with prices from only £35.

And that's just the beginning. For attaching scientific and laboratory instruments to the QL, CST even offer an IEEE-488 interface, which can handle up to 16 connected devices simultaneously!



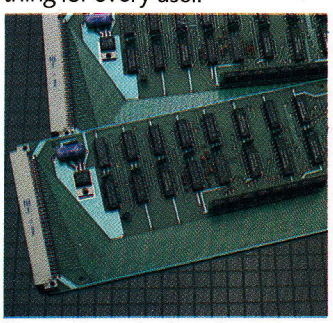
A Centronics interface slips discreetly into place.

Expansion boards for up to 4 times more memory!

Also from Quest, a simple and inexpensive way to expand the QL's RAM: with memory expansion boards.

These compact units connect to the standard QL expansion port, using the QL's internal power source or, for larger boards, an external power source.

The units range from 64K and 128K RAM boards to massively powerful 256K and 512K RAM boards, so there's something for every user.



Compact expansion boards.

Prices start at £117, and the 512K board is a very cost-effective investment at just £587.

With affordable memory like this, the QL is more than a match for any other micro under £2,000!

The spec behind the spectacle

CPU – Central Processing Unit
Fast, powerful Motorola 68008 chip. A second processor, an Intel 8049, controls the keyboard, generates the sound, and acts as an RS-232C receiver.

RAM
128K. Now expandable to 640K.

ROM
48K.

Operating system
Qdos – revolutionary single-user, multi-tasking, windowing operating system.

Storage
Twin built-in QL Microdrives. Up to 100K storage each – transfer rate, up to 15K per second.

Keyboard
Full moving 65-key QWERTY, five function keys, four cursor keys.

Language
Sinclair structured SuperBASIC.

Application software
QL Quill – word processor
QL Abacus – spreadsheet
QL Easel – graphics
QL Archive – database
All four packages supplied with the QL.

Interfaces
Two serial RS-232C interfaces, Microdrive expansion port (up to 6 may be added), ROM cartridge port, local area network, 2 joystick ports, RGB monitor and TV output.

Text screen
Various modes – up to 85 columns by 25 rows on monitor. On TV, up to 60 columns.

Graphics resolution
512 x 256 pixels (four colour), 256 x 256 pixels (eight colour).

Where to find the QL. The Sinclair QL is available at selected branches of Dixons, W H Smith, John Lewis Partnership, Currys, Greens in Debenhams and Ultimate, and larger branches of Boots, John Menzies and specialist computer stores nationwide.

Sinclair Research Ltd
Camberley, Surrey, GU15 3BR.
Tel: Camberley (0276) 686100.

sinclair

CLASSIFIED

DISC DRIVES

QL DISC INTERFACE Double sided/double density, 5¼" or 3½", up to 720K formatted file space/disc. On board s/w, random access files, extensive typeset manual.
QL MICROVITEC CUB MONITOR with cable for QL — **£265** **QL TOOLKIT** adds tabulate, memory modify in hex, decimal, octal, bytes/words + much more — **£14.95**
QL MICRODRIVE CARTRIDGES — 4 in wallet — **£19** **QL CENTRONICS PRINTER INTERFACE — £38**

Prices include VAT and delivery

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

QL SCREEN EDITOR PLUS

Edit BASIC, assembler or other programs the easy way. Bi-directional scrolling, full cursor movement, insert, delete, find etc. RAM based, so no waiting for microdrive access; uses 85 character screen.

Microdrive also contains machine coded BASIC command extensions to allow your programs to use true windows.

£12.50 from S Gaymer, 16/18 Princes St, Ipswich, IP1 1RQ

★ QL COMPENDIUM ★

SOLAR INVADERS. Arrived at last . . . Out of this world! 1 to 4 players.
WALL-BREAKER. Keep the ball in play to break the wall. Fast and smooth animation. 1 to 4 players, 5 difficulty levels.
DRAUGHTS. The classic board game, can you beat it? Full board display, 1 or 2 players.
MINEd YOUR PATH. Help the soldier cross a minefield of ever increasing hardness. Furious fun for 1 to 4 players with 10 difficulty levels.
STATISTICAL AVERAGES. Standard deviation, geometric mean, median, etc. Fast processing and straight forward to use.
CALENDER. Generate a calender for any month of any year, also discovers the date of Easter.
All of the above programs for £10.00 or £5.00 if you send your own micro-drive cart. Menu loading, instructions and p&p included.
Fast delivery from: **EQUATE, 2 FFORDD DERWYN, PENYFFORDD, CHESTER CH4 0JJ**

NEW FROM BEDSOFT

"FULL SCREEN EDITOR" with two modes of operation — full screen cursor editing and full screen line editing featuring: multiple line delete/add, renumber, global string change and many other commands together with full MDV control and **FULL SCREEN DIRECTORY.**
In all a package to save hours..... £11.95

Also the popular **"GAMBLER"** game combining skill and chance.
Become addicted for £5.95

And **"BEAT THE CLOCK"** requiring quick thinking and judgement.
For all ages £5.95

"AUTO-DRAW" Draw on the screen quickly and easily. Converts your drawing into **"BASIC"** commands for adding to your own program. Simplifies **"WINDOW"** definitions for your drawing to be scaled and multiple displayed for games programming. Full colour and "FILL" control, saving on MDV, modifying and *MUCH* more.
For fun and serious use £10.95

★ Please add £5 per order for cartridge *or* send your own *or* send your files on cartridge for printing at £2 per file and have your order returned on the same (formated) cartridge.

Ask for details of our RESIDENTIAL COURSES ranging from introductory to advanced and business users.

BEDSOFT, 30 Lansdowne Road, Bedford

WDSoftware

For the QL: WD UTILITIES (3rd ed) **(base £5.50)**
View 60-file DIRectory on one screen, one-key LOAD, COPY or print 60 files with one key (allows for namesakes). Multiple FORMATting to prevent corruption by stretching of tape. TOOLkit to give dated, numbered modules in program development. PRUNE old files to release space (one key DELETEs a file). Full instructions in QUILL file. Use up to 6 EXTRA MICRODRIVES (add on your Spectrum ones)!

WD MORSE TUTOR **(base £4)**
From absolute beginner to beyond RYA and Amateur Radio receiving. Adjustable pitch. Set speed to your target level (4-18 wpm). Learn code with single characters, progress from groups with lengthened spacing to random sentences; decrease spacing to normal. Write what you hear, then CHECK against SCREEN or PRINTER. Also own message, random figures, letters or mixed.

Ref QL (1st ed) **(base £1)**
100+ useful QL references in an ARCHIVE file.

Prices: £4.50/cartridge plus base price
Two or more programs on one cartridge — pay cartridge + bases. E.g. WD utilities and Ref QL for £11. Send YOUR FORMATted cartridge and pay base price only (but OURs have been FORMATted AT LEAST six times)! Send OUR cartridge and 25p to update earlier editions.
WDSoftware, Hilltop, St Mary, Jersey. Tel. (0534) 81392

Q.L. PROGRAMMER REQUIRED

to help North-West businessman formulate a program to use within his consumer credit business.

Phone, day — (051) 260 5544
night — Burscough (0704) 893922
Ask for Mr Dunn

DataManagement QL software — leading the way

SBUTIL — (£9.00) — Possibly the last word in Basic programming utilities. Constant readout of: Line No & Statement/Data Line & Statement/Dynamic Basic Size. Full SuperBasic speed control via the Basic Editor. 4 formats for all Screens and tastes. No more wondering where things go wrong. This tells you exactly where you are at all times.

MBACKUP — (£8.00) — An independent, fast & reliable backup to/from Mdv or Disk. File additions, over-writes and file sizes fully displayed.

TERMINAL — (£12.50) — A superb full featured terminal package. Full protocol control and ability to recall previous setups. Job control, clock and alarm modules, together with Piping for mdv and printer. Full mdv and disk control without exiting: copy, delete or save to files. Monitor version only.

CHARGEN — (£10.00) — A very comprehensive character generating package. Includes extra functions for SuperBasic to control your creations. Founts can be saved to disk or mdv for recall and use. No copyright restriction on the subsequent publication using these features.

SBEXTRAS — (£6.00) — Many extra features and procedures to add to SuperBasic. Includes full job control, dynamic file access with byte or variable GET/PUT. LENGTH, REMAINDER, QUOTIENT, HEX, DEC and many more.

FM — (£12.50) — A full feature file management package. If you need to do it to a file, this will do it for you. Supports disks/mdvs, wildcard and formatted printing to help your library control.

All the above are 100% m/language, multi-tasking and will run independently under QDOS. Prices shown include VAT & P/P, but add £4.95 per order for a mdv cartridge.

Cheques or P/Os to DataManagement: 12 Larch Way, Haxby, YORK, YO3 8RU or telephone (0904) 760351 (anytime) for our latest list, which is growing daily.

Have **you** written any good **QL** software. We are looking for the **best** software and are prepared to pay the **best** royalties.

Coming Soon: FED — a full feature file editor: **DisAssem** (if you don't know what it is — you won't need it!); **NeatList** — to keep those printouts beautiful; and many, many more. Write or Phone for your list — **Now.**

EXECUTIVE ADVENTURE

Can YOU rise from lowly tramp to high flying executive, in this unusual modern day adventure? Pit your wits against puzzles, mazes; test your skills in money management and devious dealings. No dragons in this highly original and demanding adventure (except the secretary)!

only £15

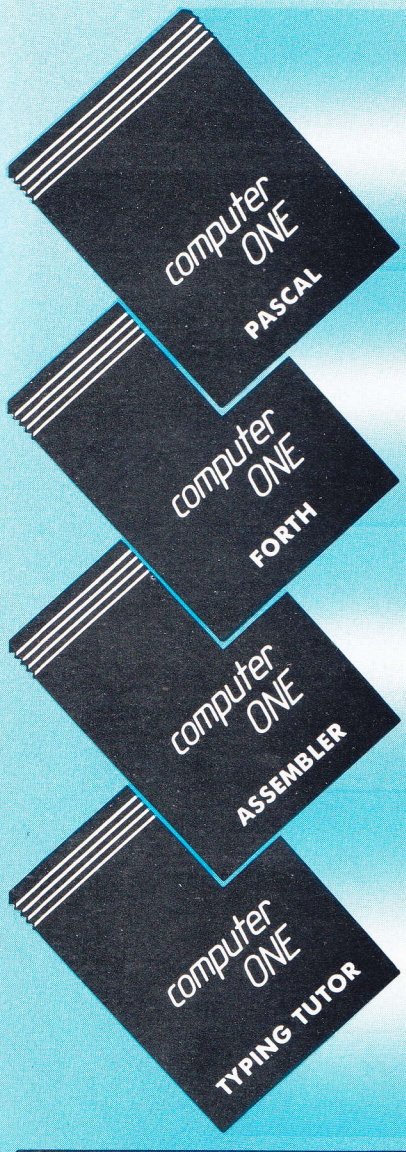
INTERSOFT
7 Richmond Road, Exeter

INVESTORS!
Track your shares with "STOCKMARKET MANAGER" from *Portfolio Software*
Facilities include: • Simple entry of purchases, sales, prices etc. • Portfolio valuation – analysed by investment type comparison of performance against market. • Calculation of % return on each investment. • Assessment of Capital Gains Tax liability. • Handles all investment types, including Traded Options.

Orders/Cheques to: **PORTFOLIO SOFTWARE, PO Box No. 15, London SW11 5RP** No. of copies
Please send: "STOCKMARKET MANAGER" @ £39.95 incl. p&p. ☐
Information sheet (please enclose SAE) ☐
Name .....
Address .....
..... QLU/Apr

QL POWER

AT YOUR FINGERTIPS FROM COMPUTER ONE



PASCAL £39.95 Inc. VAT & p&p.

Computer One PASCAL is a powerful implementation of this classic programming language. Produced specifically for the QL, this package provides a professional and highly educational programming system. Complete with comprehensive 80 page manual, this is the ultimate language package that no QL enthusiast can do without.

INCLUDES: ★ Full Screen Editor ★ QL graphics and sound extensions
★ Example programmes ★ Complete 87 page User Guide
★ Full implementation close to ISO-standard ★ Typically 15x faster than BASIC

FORTH £39.95 Inc. VAT & p&p.

FORTH is easy to learn, highly efficient, and allows you to explore the full capabilities of your QL. Computer One FORTH is a full implementation of the latest FORTH-83 standard with graphics and sound extensions, and may be used to produce machine code applications for your QL. Ideal for writing real-time or games programs.

INCLUDES: ★ FORTH-83 system ★ Forth Screen Editor ★ 48-page manual
★ QL-graphics and sound extensions ★ Example FORTH programs

ASSEMBLER £29.95 Inc. VAT & p&p.

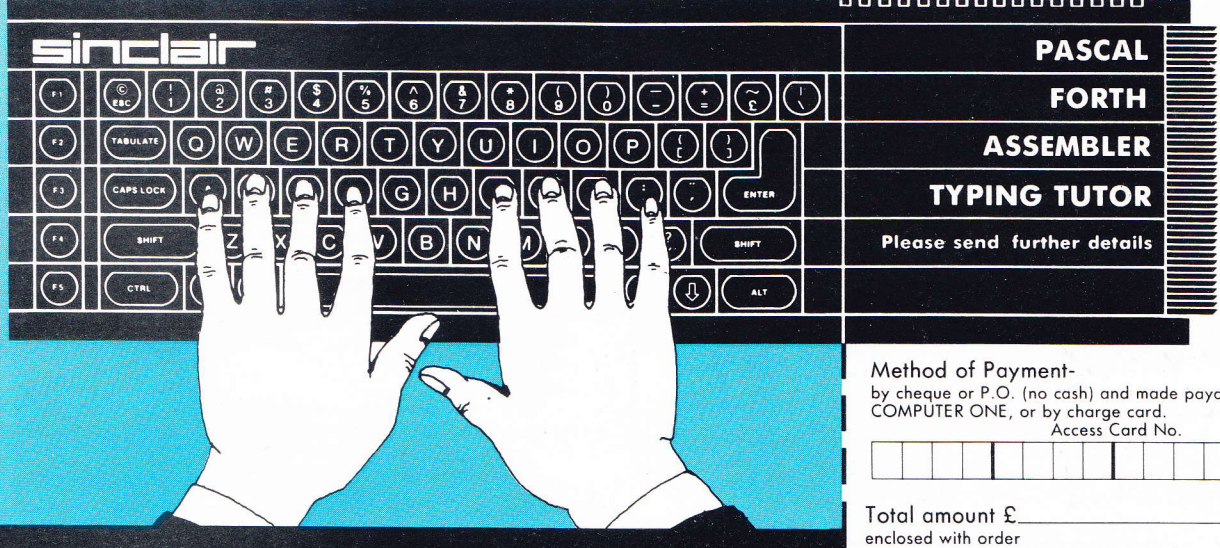
COMPUTER ONE Assembler is a comprehensive assembler program toolkit providing a professional multitasking assembler written in machine code for speed and compactness. An ideal package for the machine code programmer, allowing you easy access to the QDOS operating system, and integration to your Superbasic and Pascal programs.

INCLUDES: ★ 68008 Assembler ★ Full Syntax Checking ★ Full screen editor
★ Comprehensive User Manual ★ Integrates to Pascal and Superbasic

TYPING TUTOR £19.95 Inc. VAT & p&p.

Let your Sinclair QL teach you to type! Computer One's Typing Tutor is a fast, easy and enjoyable way of learning true touch-typing on your QL.

INCLUDES: ★ Self-paced tutorials
★ Speed and accuracy diagnosis
★ Interactive colour graphics
★ User Guide



computer ONE

SCIENCE PARK, MILTON ROAD, CAMBRIDGE

Name_____

Address_____

_____

Signed_____

	Tick Order
PASCAL	☐
FORTH	☐
ASSEMBLER	☐
TYPING TUTOR	☐
Please send further details	☐

Access

Method of Payment-
by cheque or P.O. (no cash) and made payable to COMPUTER ONE, or by charge card.
Access Card No.

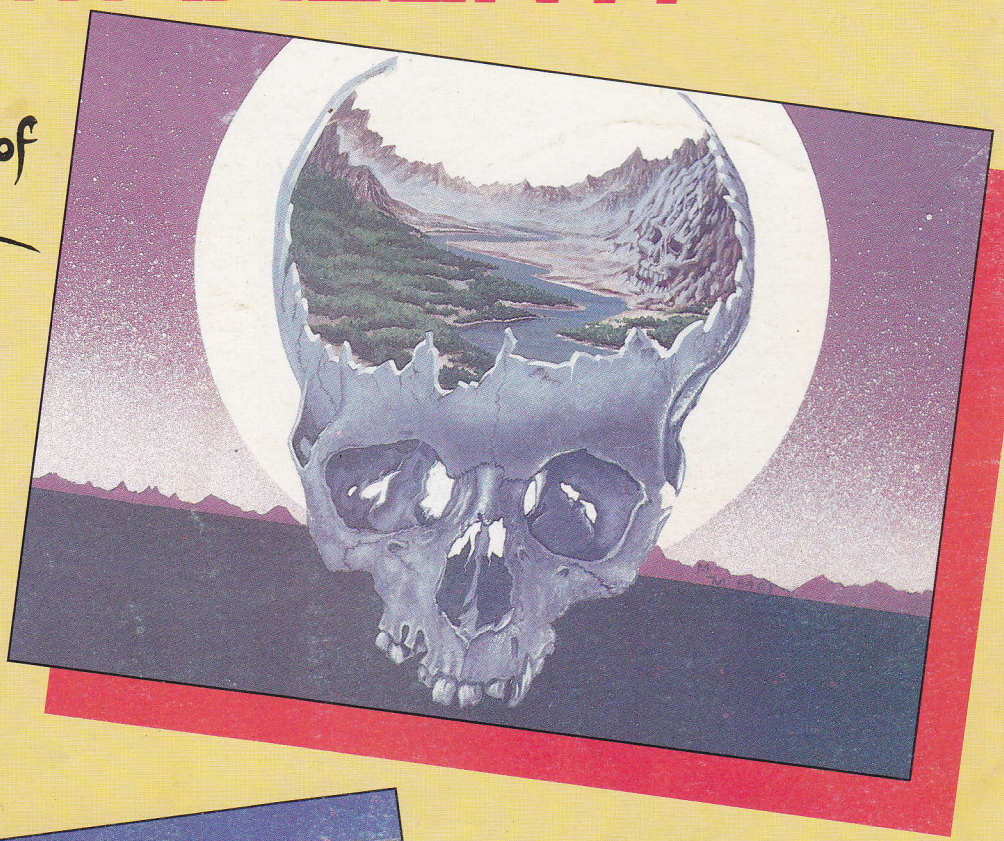
QL/3/85

Total amount £_____
enclosed with order

Credit card sales: UK buyers only. Tel: (0223) 862616

The first QL adventures from TALENT!

The Lost Kingdom of ZKUL



Explore the ancient dwelling-place of the Dwarves – where the Wizard guards the last precious secret.

A classic and complex text adventure with hundreds of locations and a huge vocabulary. Set in real-time with traps, tasks, puzzles and mazes – and a special note-pad feature to aid you.

"the quality of the game is superb"
Micro Adventurer

PRICE
£19.95
+ 50p postage and packing

WEST

You are on the track of a notorious gang of bank robbers who have gone to ground near an abandoned mining town deep in Indian territory. Your task – to outwit and outgun the robbers, collect as much loot as possible and piece together clues on how to escape.

A demanding and exciting text adventure for the experienced games-player, using over 200 words and as many phrases. There are over 130 locations to search – not all of them easy to find. Events happen in real-time outside your control – Indians charge, rattle-snakes slither past and robbers appear and shoot at you. How long can you survive?

PRICE
£19.95
+ 50p postage and packing
(Also on CBM 64, BBC & Electron)

TALENT
COMPUTER SYSTEMS

CURRAN BUILDING, 101 ST JAMES ROAD, GLASGOW G4 0NS
24 HOUR CREDIT CARD HOT-LINE 041-552 2128