An EMAP Publication

SINCLAIR'S SUPERMICRO

NOW MONTHLY

QL USER

April '85 95p
Germany DM5.50

ARCADE ATTACK
HIGH SPEED SPRITES

PSION 2.00 UPGRADES

MICROPERIPHERALS' DRIVES

TALENT'S NEW GRAPHICS

COMPUTER ONE ASSEMBLER

GAMES WORKSHOP'S D-DAY

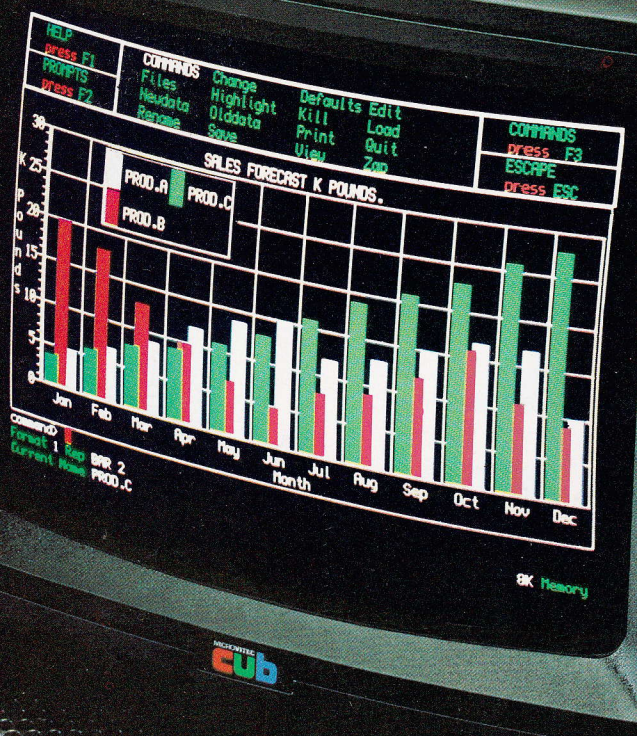
WIN
A COMPUTAMATE
Q-DISC SYSTEM
WORTH OVER £500

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

April 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, Giles Todd, Gareth Jefferson, Martin Croft, Roger Vernon, John Lang, Andy Carmichael, Richard Cross, Ian Williams and Jane Robins

Editorial 01-251 6222

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

Advertising 01-251 6222

Competitions

The winner of our February registration card draw is Justin Mason from the Isle of Wight. He receives a Centronics interface courtesy of Miracle Systems.

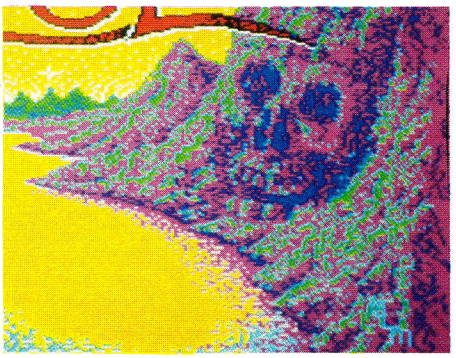
The 'Questions and Answers' competition in the same issue brought forth some amazing answers, but the correct solutions were: D RAM stands for Dynamic Random Access Memory, 'ns' represents nanoseconds (10^{-9} seconds), CP/M is Control Program for Microcomputers and the formattted storage capacity of the Firefly disk, 7.5 Mbytes.

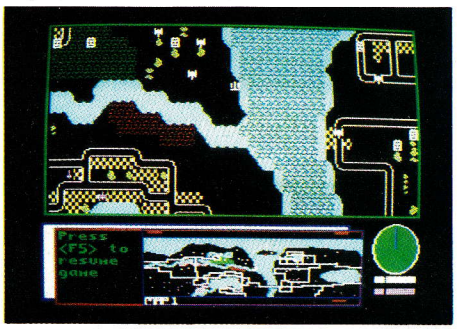
Quite a lot of readers answered all these questions correctly, so the tiebreaker came into play. The final choice was a difficult one, but we decided that the winner should be Chris Abbess from Woodford Green, Essex. His reason was: "Perhaps Lashley's trace will be found in Quest's memory enhancements".

Second prize goes to Mr V Holland from Cambridge, who came up with a different idea: Quest '**E**xecutive' **N**ow **G**ives **R**eliable **A**lternative to **M**icrodrive.

Third prizes go to J Stevens from Romsey, Hants, D E Nixon from Southampton, and B T Szocik from Clapham, who proposed: 'Letters not common to MERLIN and GRAIL are ENGRAM'. Originality was the key to our selection here, but perhaps BTS would care to elucidate?

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.

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

76 Grainger Close, Basingstoke, Hants. RG22 4EA
Telephone 0256 460748

☐ 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

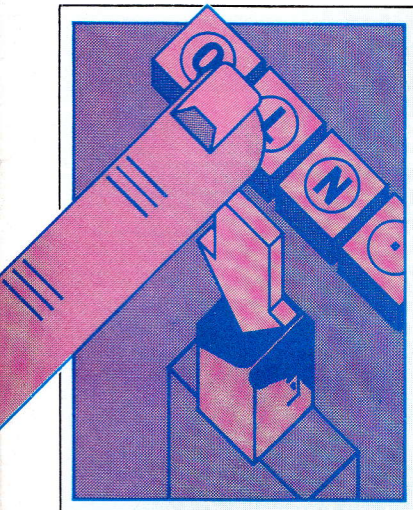
To order products please make cheques/P.O.'s payable to MEDIC DATASYSTEMS LTD. Allow 28 days for delivery. All prices include VAT. Please add £5 p&p.

NAME_____

ADDRESS_____

_____

Signed_____



NEWS

Sid Smith reports on the latest forward and backward steps on the QL front.

Who Gives A TOS?

The QL walks the borderline between the top end of the home computer market and the low end of business machines. To generate a large software base, computers in the home market need huge sales (which the QL hasn't yet got), while business computers need to run an industry standard operating system (which the QL hasn't got, but which three different companies reckon they can sell you).

The two most successful operating system standards in the business world have been CP/M 80, supplied by Digital Research for 8-bit computers, and MS-DOS (or PC-DOS) supplied by Microsoft and implemented on the 16-bit IBM PC and its imitators.

QL users have the choice of sticking to the computer's native QDOS, or of supplementing it with CP/M 80, CP/M 68K or 68K/OS – rival systems which claim an established software base, the possibility of plugging your QL into an up-and-coming industry standard, or both.

CP/M 80 offers the QL user thousands of tried and tested programs, but was written for 8-bit processors; buying a 68008 and running CP/M 80 is like buying a buzz-saw – and cutting wood with the engine turned off.

CP/M 68K makes more use of the QL's central processor, but has far fewer application programs.

68K/OS, say its adherents, more thoroughly exploits the 68000 than any of the alternatives, but at the time of writing it hasn't a single applications package.

Which brings us to QDOS. For the QL owner, QDOS has the great advantage of requiring no additional outlay, and according to this magazine's Technical Consultant – is at least the equal of any of its rivals. Software support continues to be disappointing, but with the price cut in microdrive cartridges and a growing awareness among programmers that sooner or later they'll have to learn 68000 code, there's reason for remaining hopeful.

However, there's no chance of QDOS being adopted by other manufacturers and becoming an industry standard for business software. QDOS will attract software support only in proportion to sales of the QL. By using its own particular OS for the QL, Sinclair has ensured that – like the ZX computers before it – the machine has to make its own way in the world.

All of which is in direct contrast to Atari. Jack Tramiel's new company has likewise priced a machine at the borderline between the home and business markets but has equipped its new ST range with an OS closely related to a potential industry standard.

"But," you may cry, "the Atari OS is some obscurity called TOS – which isn't a potential anything."

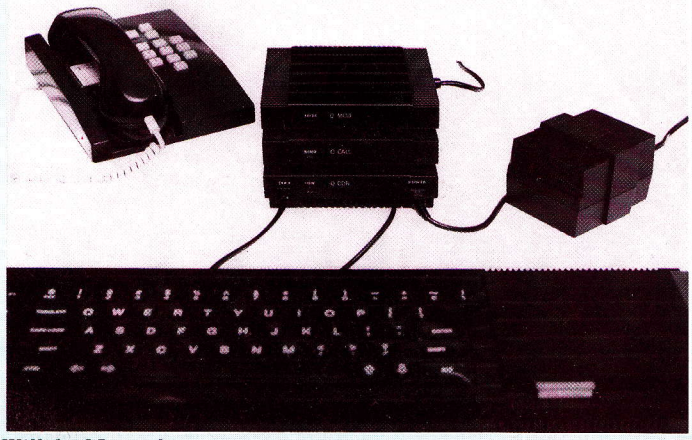
Well, although Atari and Digital Research are curiously shy about it, conceding only that the latter company was responsible for writing part of TOS, a senior DR official has confirmed to us that TOS was "essentially based on CP/M 68K".

In other words, software written under CP/M should be relatively easy to port across to the ST range. Whatever problems the Atari machines face, their support from large business packages should grow much more quickly than the QL's.

Sinclair, meanwhile, claim not to be worried by Tramiel's new range. "We'll believe it when we see it," is what they say.

The Atari ST machines, unveiled at Las Vegas in January, are built around the Motorola 68000 chip (a faster version of the QL's processor), have a minimum of 128K RAM, boast the Macintosh-style GEM user interface (a mouse, windows and icons front end, also employed in the £3000 Acorn ABC range), have better sound, keyboard and interfaces than the Sinclair machine, and compete head-on in price. They're due to hit the UK in Spring.

Apart from insisting that they'll be able to sell the QL in America (they're wrong), Sinclair's attitude is that manufacturers are always announcing products they don't have ready . . . which reminds us, the Atari unveiling comes exactly a year after the QL was launched – coincidence or calculation?



Will the QL ever become a real communicator?

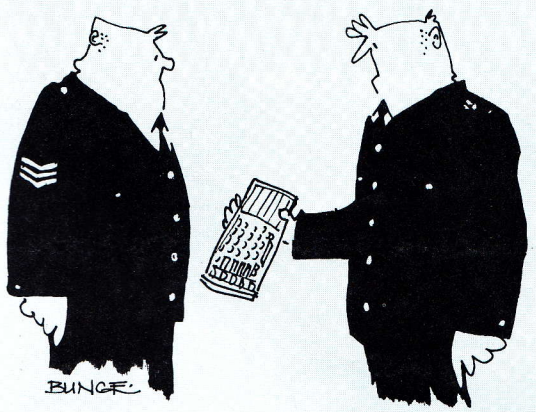
Modem Times

OEL, makers of the QL comms package, seem to be doing a Sinclair on us. OEL have been advertising QCOM – and cashing cheques for it – since late November, and the three-module package was officially launched at the Birmingham Which Computer? Show in mid-January.

But throughout all this time QCOM wasn't ready, and early buyers heard nothing from the company till the end of January, when letters arrived postponing delivery till mid-February – we've still had nothing to date.

OEL attribute the delay to a four-week slippage in the arrival of the customised 'computer on a chip' which sits at the heart of the comms package.

'Jack the Lad?'



Not the usual sort of computer crime sarge, there's a woman outside who claims it indecently assaulted her.

NEWS

Winning Wafer

Work on Sinclair's 0.5Mb wafer scale memory chip for the QL is progressing according to schedule, says a chief designer involved in the project.

As reported in these pages some months ago, Sinclair plans to build its own factory for wafer scale integrated circuits – circuits, that is, which occupy the whole of the several inch diameter surface of a silicon wafer, rather than the tiny fraction which is broken off to form the familiar silicon chip.

In the December/January *QL User*, Sir Clive explained that the first of these WSI circuits would be a "silicon Winchester" for the QL, offering 0.5Mb for a price around £200 (though his latest pronouncements put the cost to between £300 and £400), and capable of massive speed.

"The speed is so high," said Sir Clive, "that you will be able to store whole screens of information and then play them back quickly, to produce animation." So attractive is the prospect of this WSI memory circuit that – according to some reports – Sir Clive is even having second thoughts about the promised QL Winchester interface.

Work on the memory circuits of the WSI unit is being carried out at the British custom circuit house of IMS, possibly the only company in Europe with the necessary expertise in specialist memory design.

"We're doing the memory and the basic wafer," explained IMS's Graham Davies. "The control logic is being done by Sinclair under a design team at the Metalab."

The memory circuit will plug into the QL's external RAM socket, and draw its power from the QL. However, "If the power goes off, either through power failure or because the memory unit is unplugged, then a battery will switch in to preserve the wafer's contents."

STOP PRESS!

Some interesting developments for QL ROM spotters, from Alan Turnbull

Recently, after receiving my QL back from a repair at Camberley under warranty, I was astonished to find yet another response from the function 'VER$' which returns the two-letter code of the current version of the QL ROM.

The code it came back with was 'JS' and not 'JM' . . . so what do you get in 'JS' which is not in previous releases?

Well, you will be glad to know that the construct 'WHEN' together with 'ERRor' is implemented so that error trapping is at last possible on the QL. This is further supported by 25 extra SuperBasic keywords. An example program showing the use of some of these extra keywords in conjunction with 'WHEN' is listed below.

Also with 'JS', you get a trace facility and the option of re-vectoring the messages output to the console such as error reports and prompts.

New functions return values associated with the error status of the QL. ERNUM gives the error code of the last error that occured, whilst ERLIN tells you at which line in the program it happened. All the other new functions return either 0 or 1. For example, if the last error was 'overflow' then the function ERR_OV would return 1.

The procedure REPORT is intended for use with WHEN ERRor as a means of printing the error message, which would have been sent to channel 0, to any other channel.

In terms of bugs, the 'JS' ROM is pretty clean. The number base conversion utility vectors documented in the QDOS Manual as "not working on QDOS versions 1.03 and before" have been fixed and there seems to be an extra vector because the initialisation routine which was at address 360 on previous machines is now at address 362.

'JS' still complains when you try to use a simple variable with the same name as a DIMensioned array but a new bug is introduced: the formal parameter of a procedure/function may not be used as a SELect variable — it must be copied to a LOCal variable and that must be used in the SELect statement.

Perhaps the main bug-correction worthy of note is that the QL can now recognise 16 plug-in peripheral cards rather than just the one. Someone had put a BGE rather than a BLT instruction in the testing loop!

Further examples will appear in future issues of QL User

```
100 REMark       'WHEN' demonstrator for QL 'JS' ROM
110 REMark    COPYRIGHT (c) February 1985, Alan Turnbull
120 :
130 WHEN ERRor
140   AT RND(0 TO 19),RND(0 TO 39)
150   PAPER RND(0 TO 5)
160   PRINT "*** Error number ";ERNUM;" at line ";ERLIN;" ***"
170   PRINT £0;"Error 'not found' status = ";ERR_NF;"   ";
180   REPORT
190 END WHEN
200 :
210 MODE 4
220 REPeat forever
230   DIR mdv9_
240   LINE 83,0 TO RND(0 TO 166),RND(0 TO 100)
250 END REPeat forever
```

USER GROUP NEWS

The latest news and views from IQLUG chairman Leon Heller.

Besides the Swindon workshop mentioned in the last issue, workshops are also planned in Cambridge, Liverpool, Edinburgh, Leeds, the North East and Swindon again, in the autumn. The usual activities will be repeated.

The Camberley Connection

I recently visited Sinclair's Camberley distribution centre. The Camberley facility does not belong to Sinclair, but is run by GSI, a French-based multinational corporation. The level of customer support provided to Sinclair computer users was most impressive. One department deals with written queries, with standard paragraphs called up on a word processing system (not a QL with Quill!) for the most frequent queries, and an individual reply for the more esoteric ones. These are merged into the same letter, if the customer has several questions. Another department answers telephoned enquiries. Since they are handling several thousand queries a week, some mistakes will obviously occur, but I don't know of any other UK microcomputer manufacturer who has something like 30 people working on customer support! One of our members was unable to get a Brother M1009 printer working properly with Quill. I mentioned that Psion were unable to help him, and was immediately handed a piece of paper containing all the necessary information!

Memory Gap

One of our members, David Nowotnik, has sent in the following tip for users who miss the "PRINT MEM" function available in MBASIC. PRINT PEEK_L(163860) – PEEK_W(163852) does much the same thing on the QL.

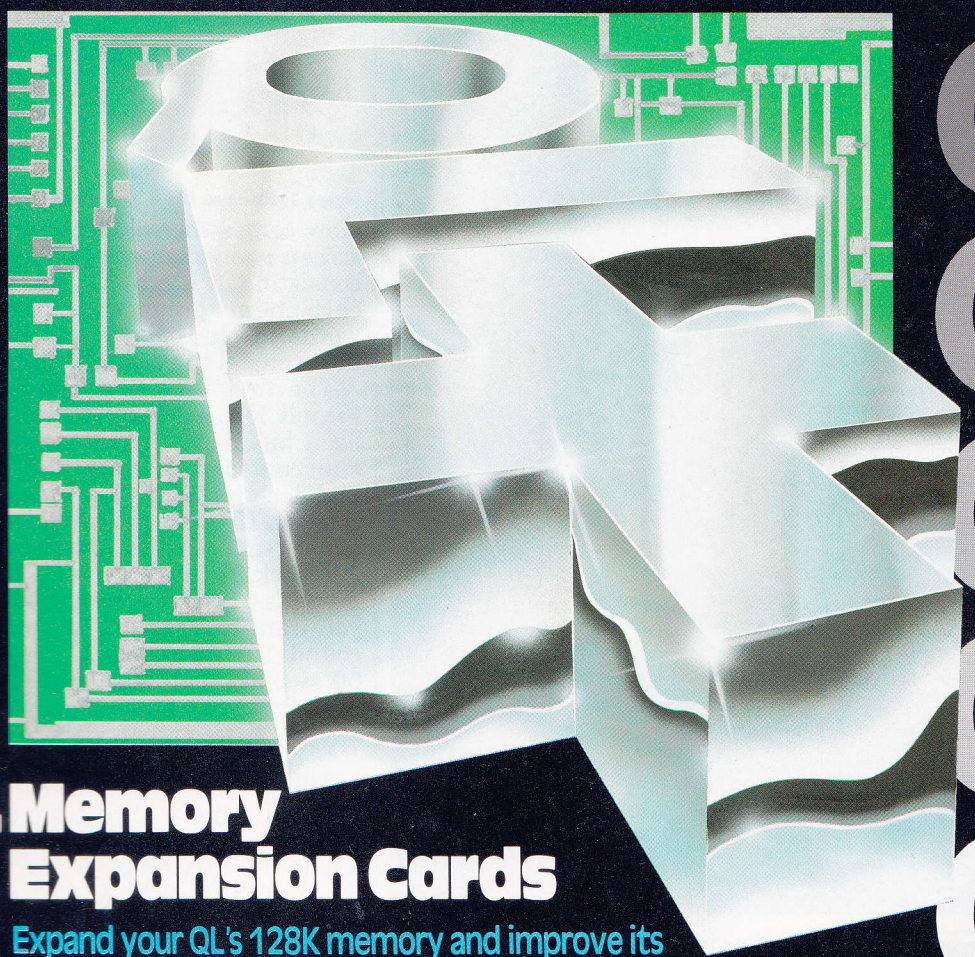
Still on the subject of memory, readers thinking of buying additional memory might be interested to hear that external memory on the QL should run three times faster than the internal memory, if it has been properly designed!

Any IQLUG members who would like to reach a wider audience than Quanta are welcome to use this column. Please send contributions direct to me so they can be passed on to *QL User*.

The Independent QL Users' Group (IQLUG) publishes a monthly newsletter, maintains a software library, supports local groups, and provides members with a free advice service. Workshops are arranged from time to time in different parts of the country. Further details are available from:
Brian Pain, 24 Oxford Street, Stony Stratford, Milton Keynes MK11 1JU. Tel: (0908) 564271

EXPAND YOUR MEMORY

QL+ QL+ QL+ QL+ QL+ QL+ QL+ QL+



QL Memory Expansion Cards

Expand your QL's 128K memory and improve its reliability by adding our 64, 128 or 256K RAM cards priced at a remarkably low £75, £115 and £150 respectively (ex VAT)

QL+ RAM cards give you:

- Optional unique ROM facilities – implementing a RAM disc and extending Superbasic.
- All QL+ RAM cards are powered from the standard QL power supply.
- All cards fit completely within the QL casing.
- All access faster than QL's own memory.
- Extended RAM capabilities, reducing reliance on micro-drives.
- State of the art design, using the latest in D RAM technology.

PCML Ltd., Royal Mills, Esher, Surrey.
TEL: 0372 67282/68631
TLX: 935398.

QL+ gives you more for less.

64K RAM Card-Only £75 + VAT

Order Now from PCML Ltd., please post cheque with order and allow up to 28 days for delivery. Our New CPM 3.0 Z-80 Card will be available in June. Enquiries now welcome.

GAMES PROGRAMMING

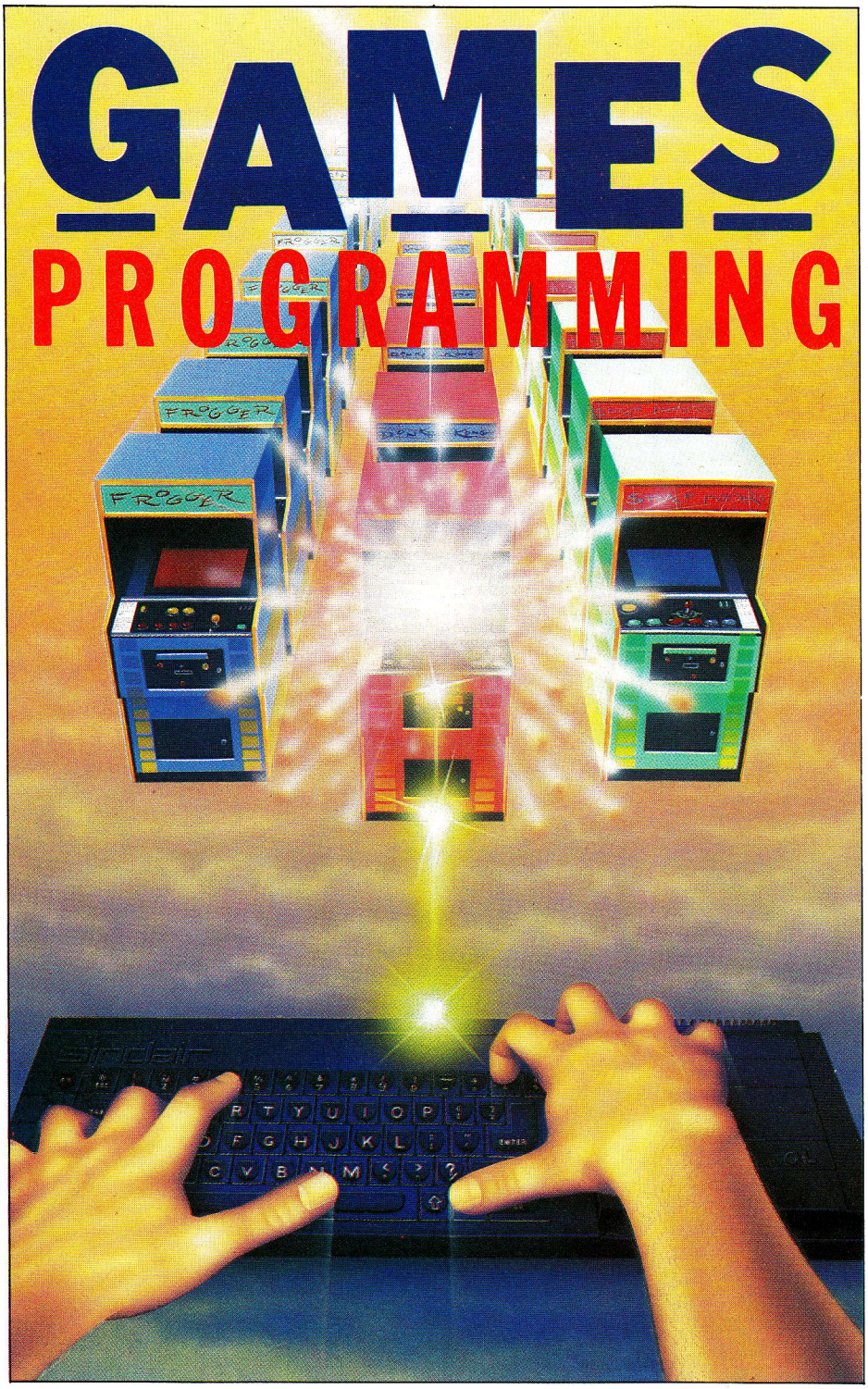


ILLUSTRATION BY TONY RANDELL

Writing machine code games demands flair, imagination and a mastery of all that is QL. In the first of a new series, Ian Williams magics monsters from nowhere.

In this series we provide a number of machine code routines which will ultimately combine to make a Space Invaders type game called *Paladin*. The order in which these routines have been presented means that, right from the start QL users will have something workable on screen with which they may experiment.

Our first chunk of code is the largest as it contains the main control routine for the game. However, most of the calls to subsidiary routines are preceded by a ';' which will cause the assembler (Computer One's in this case) to disregard them. The effect is similar to starting a line with REM in BASIC. In time the relevant section will be explained, subsidiary routine supplied, ';' sign removed and the call activated. Meanwhile, these lines can only serve to whet your appetite for what is to come!

MONSTER MAP

Our first task is to define our space invader. For this we have decided on a block of 8 × 8 pixels, giving a monster 64 pixels square. Designing a sprite is fun. All you need is a piece of squared graph paper and a pencil. Draw out an 8 × 8 grid and then colour in the shape of your invader (fig 1). In mode 512 you have the choice of red, green, black and white. Remember, as each square only occupies a pixel it doesn't pay to be too ambitious and aim for a multicoloured tapestry effect. Chances are, nobody'll notice it, anyhow!

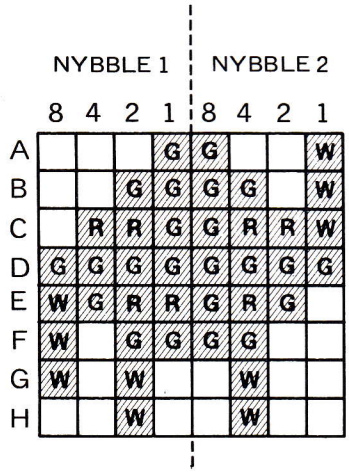


Figure 1. Planning the 8 × 8 grid.

Now comes the intellectual bit. The screen is divided into 256 lines each consisting of 128 bytes. The bytes in each line are grouped into pairs called words (each 16 bits) and it's these that control the colours of the pixels.

The first byte is called high, the second, low. The way it works is this: each bit in the high byte (the first one) controls the green in each pixel and each bit in the low byte controls the red. If you want the pixel white, simply set both bits, and the opposite for black. So, if you want a pixel red then it's 01 (no green and just red). If you want it green, it's the other way round (10). Therefore, 11 = white and 00 = black. Fig 2 should show you exactly what's happening.

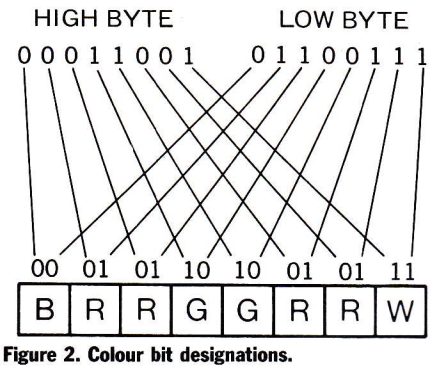


Figure 2. Colour bit designations.

Now work your way through the design calculating the colour of every pixel in binary for each of eight rows. You'll be left with eight 16 digit numbers which can be converted to hex. The method we use saves time and it's a lot easier to use. To design your own sprite follow the instructions exactly.

1. Draw an 8 × 8 grid exactly as shown in fig 1 (don't forget the

numbers on the top).

2. Draw any shape you want and then initial each square with the colour you like at that point.

3. Look at the top row only (that's going to be your first four hex digits).

4. Each square has a number above it (8,4,2 or 1). Count the whites and the greens in the first set of four squares, add them up, and they are your first digit. Do exactly the same for the second set of four squares so you get your second digit.

5. Now count the whites and reds in the first four squares, add them up and they're your third digit.

6. Finally do the same for the second four squares, and you have your fourth digit.

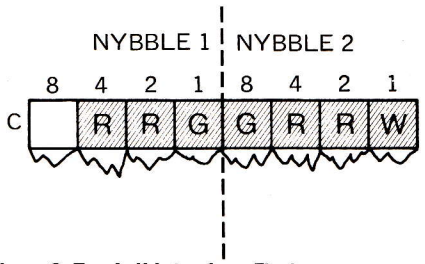


Figure 3. Two half bytes from Fig 1.

If you're still confused, look at fig 3. It represents line 'C' from fig 1. As you can see, it's divided into two sections, each of which is 1 nybble (half a byte). Following the plan above, we first count the whites and greens in nybble 1, add the numbers on top of them, and get $1. Repeat the operation for the second nybble and we get $9. So far, then we've got $19. Now we add up the whites and reds in the first nybble and get $6. Finally, doing the same to the second nybble we get $7. Full definition of line 'C', therefore is $1967. Easy, isn't it? All these numbers are then entered as data shown in the listing as lines 277 – 291. We use eight different sprite shapes and alternate the 'a' patterns with the 'b'.

STEP IN TIME

Ok, now we have our invaders how do we move them? If you look at line 33 you'll see the label **repeat.** It's this loop that deals with all movement, calling three subroutines: **un_paint, slither** and **paint.** Cropping up all over the place is a variable *num_sp.* That is set to the total number of invaders placed on the screen at the start (strictly speaking, it's the number of elements in the array *xpos,ypos).* The '–1' after it is so that the loop will count down *num_sp* times. If we just used *num_sp* by itself, the computer would count '0' as one and we'd get one loop too many.

Un_paint is the first call and this is well explained in the listing (lines 138 – 147). From here the program branches to **blank_out** (125 – 133) and then to **calc_addr** (259 – 270).

Calc_addr is a very useful little subroutine. Lines 262 – 265 calculate the horizontal position on the screen of the first screen word (two bytes) we want to work on. *Scr_top* is a variable set to $20000 which is the address in memory at the top left hand corner. Adding it to *d6,* our *x_pos*ition register, gives the absolute horizontal location of the word. A similar process is then undertaken to find the vertical location by lines 266 – 269.

Un_paint deletes the invader by setting the bytes occupied by it to black using **blank_out** which is fully explained in the listing. Finally, **paint** pops a new invader on the screen. One useful little variable is *flick,* which alternates the invader types, thus giving a type of animation as well as movement.

Entered as it stands, this program will give a set of invaders that cavort merrily across the screen at a frenetic rate. Next month's issue will give the gun at the bottom of the screen and provide a missile launching routine.

```
0000 00020000    1 scr_top          equ     $20000  ; Location of first screen word
0000 00000035    2 num_sp           equ     53      ; Number of invading things
0000 00000010    3 num_base         equ     16      ;
0000 000000C6    4 ut_con           equ     $c6     ; QDOS vector to open console
0000 000000D0    5 ut_mtext         equ     $d0     ; QDOS vector to print a string
0000 00000032    6 score_poshi      equ     50      ; Position (x co-ord) of high score
0000 000001E0    7 score_pos        equ     480     ; Same for ordinary score
0000 00000005    8 bas_pos          equ     5       ;
0000 0000000F    9 nemalt           equ     15      ; Height of mother ship
0000            10 ;
0000            11 ;
0000            12 ;
0000            13 coldstart
0000            14 ;         moveq   #0,d0          ; These three lines initialise the
0000            15 ;         lea     hiscore,a0     ; high score to zero
0000            16 ;         move.l  d0,(a0)        ;
0000            17 ;         bsr     save_message   ; All subroutines will be explained
0000 6100002C   18 warm      bsr     dimbo          ; fully as we print them
0004            19 ;         bsr     clear          ;
0004            20 ;         bsr     setbarriers    ;
0004            21 ;         bsr     savebarriers   ;
0004            22 ;restart lea      bulposx,a0     ; These two lines stops any bullet
0004            23 ;         move.w  #-1,(a0)       ; movement on the screen
0004 6100018A   24           bsr     clear          ;
0008            25 ;         bsr     loadbarriers   ;
0008            26 ;         bsr     new_gun        ;
0008 61000160   27           bsr     paint          ;
000C            28 ;         bsr     print_score    ;
000C            29 ;         bsr     print_hiscore  ;
000C            30 ;         bsr     print_bases    ;
000C            31 ;         bsr     print_flags    ;
000C            32 ;         bsr     bombs          ;
000C            33 repeat                           ;
000C            34 ;         bsr     move_gun       ;
000C            35 ;         move.b  seq_num,d0     ; This is complicated and we'll explain
000C            36 ;         beq.s   no_bang        ; it fully in a future issue
000C            37 ;         bsr     draws_bang     ;
000C            38 ;no_bang bsr     drop_bomb      ;
000C            39 ;         bsr     bulmov         ;
000C            40 ;         move.w  baddyx,d0      ; Checks x position of mother ship and
000C            41 ;         bmi.s   no_mum         ; draws it if it isn't -1
000C            42 ;         bsr     baddybus       ;
000C 41FA031E   43 no_mum    lea     in_wait,a0     ; These lines count down between
0010 5310       44           subq.b  #1,(a0)        ; invader movements
0012 6618       45           bne.s   no_inv         ; If counter not 0 then no move
0014 10BA032C   46           move.b  inv_speed,(a0) ; Reset counter to inv_speed
0018            47 ;         bsr     rev_bul        ;
0018 610000C6   48           bsr     un_paint       ;
001C 610000DA   49           bsr     slither        ;
0020 61000148   50           bsr     paint          ;
0024            51 ;         bsr     rev_bul        ;
0024 45FA0314   52           lea     flick,a2       ; These two lines switch between
0028 0A120001   53           eor.b   #1,(a2)        ; alternative invader designs
002C            54 ;         move.w  baddyx,d0      ; These lines check to see if a mother
002C            55 ;         bpl.s   no_inv         ; ship has appeared
002C            56 ;         lea     badwait,a0     ; If no mum then decrease counter loop
002C            57 ;         subq.b  #1,(a0)        ; (regulates appearances of ship)
002C            58 ;         bne.s   no_inv         ; If counter<>0 then no mother ship
002C            59 ;         lea     baddyx,a0      ; Else intialises mother ship
002C            60 ;         moveq   #0,d6          ;
002C            61 ;         move.w  d6,(a0)        ;
002C            62 ;         moveq   #nemalt,d7     ;
002C            63 ;         bsr     saver          ;
002C 60DE       64 no_inv    bra.s   repeat         ; Does the loop,again
002E            65 ;
002E            66 ; Dimbo sets up certain program variables
002E            67 ;
002E            68 dimbo
002E 41FA032A   69           lea     num_bas,a0     ;
0032 10BC0002   70           move.b  #2,(a0)        ; Because the 68008 can't store data
0036 41FA02F0   71           lea     accel,a0       ; directly into a program counter
003A 30BC0004   72           move.w  #4,(a0)        ; offset address ( that's a memory
003E 7000       73           moveq   #0,d0          ; location which is decided upon when
0040 41FA0317   74           lea     seq_num,a0     ; the assembler calculates the number
0044 1080       75           move.b  d0,(a0)        ; of bytes which need to be added to
0046 41FA02F2   76           lea     flick,a0       ; the Program Counter so that the
004A 1080       77           move.b  d0,(a0)        ; correct address can be found) you
004C 41FA02E0   78           lea     xdir,a0        ; have to lea (load the effective
0050 1080       79           move.b  d0,(a0)        ; address) the label into an address
0052 41FA0326   80           lea     baddyx,a0      ; register which will then point to
0056 30BCFFFF   81           move.w  #-1,(a0)       ; the correct address in  memory.Phew
005A 41FA02D4   82           lea     sd_flag,a0     ;
005E 1080       83           move.b  d0,(a0)        ;
0060 41FA02FB   84           lea     numflag,a0     ;
0064 1080       85           move.b  d0,(a0)        ;
0066 41FA030E   86           lea     score,a0       ;
006A 2080       87           move.l  d0,(a0)        ;
006C 41FA02ED   88           lea     invnext,a0     ;
0070 10BC001E   89           move.b  #30,(a0)       ;
0074            90 ;
0074            91 ; This routine establishes the array which holds the screen positions
0074            92 ; of each of the invaders
0074            93 ;
0074            94 new_screen
0074 41FA02CC   95           lea     inv_speed,a0   ; Sets intital speed of invaders
0078 10BC00D2   96           move.b  #210,(a0)      ; Ditto
007C 41FA0240   97           lea     xpos,a0        ; Sets up A0 and A1 as array pointers
0080 43FA0206   98           lea     ypos,a1        ; for x and y arrays (inv positions)
0084 1E3A02D5   99           move.b  invnext,d7     ; Sets d7 to height of invader gaggle
0088 7003      100           moveq   #3,d0          ; - - - - - - - - - - - - - - - - -
008A 7207      101 frog      moveq   #7,d1          ;  This section spaces out
008C 7C1E      102           moveq   #30,d6         ;  the invader things on the
008E 30C6      103 froglet   move.w  d6,(a0)+       ;  screen
0090 12C7      104           move.b  d7,(a1)+       ;
0092 06460030  105           add.w   #48,d6         ;
0096 51C9FFF6  106           dbf     d1,froglet     ;
009A 06070028  107           add.b   #40,d7         ;
009E 51C8FFEA  108           dbf     d0,frog        ; - - - - - - - - - - - - - - - - -
00A2 1E3A02B7  109           move.b  invnext,d7     ;
00A6 06070014  110           add.b   #20,d7         ;  Same again, but this does
00AA 7002      111           moveq   #2,d0          ;  the inner ones in a nice,
00AC 7206      112 newt      moveq   #6,d1          ;  interlaced pattern!!
00AE 7C36      113           moveq   #54,d6         ;
00B0 30C6      114 newtlet   move.w  d6,(a0)+       ;
00B2 12C7      115           move.b  d7,(a1)+       ;
00B4 06460030  116           add.w   #48,d6         ;
00B8 51C9FFF6  117           dbf     d1,newtlet     ;
00BC 06070028  118           add.b   #40,d7         ;
00C0 51C8FFEA  119           dbf     d0,newt        ;
00C4 4E75      120           rts                    ;
00C6           121 ;
00C6           122 ; This is a lumpy (but quick!) way of removing any colours from the
00C6           123 ; screen at co-ordinates denoted by d6 (x) and d7 (y)
00C6           124 ;
00C6           125 blank_out
00C6 48E72340  126           movem.l d2/d6-d7/a1,-(a7)
00CA 61000114  127           bsr     calc_addr      ;
00CE 7407      128           moveq   #7,d2          ; This clears the screen at position
00D0 4291      129 wipe      clr.l   (a1)           ; (a1) and the 7 long words below to
00D2 D2FC00B0  130           add.w   #128,a1        ; black (i.e. it wipes a sprite and a
00D6 51CAFFF8  131           dbf     d2,wipe        ; big block either side to make sure
00DA 4CDF02C4  132           movem.l (a7)+,d2/d6-d7/a1
00DE 4E75      133           rts                    ; we got it!!)
00E0           134 ;
00E0           135 ; This routine deletes all the invaders from screen prior to them
00E0           136 ; being moved left, right or down
00E0           137 ;
00E0           138 un_paint
00E0 41FA01DC  139           lea     xpos,a0        ; This sets a0 to x array
00E4 43FA01A2  140           lea     ypos,a1        ; Same for a1 to y array
00E8 7434      141           moveq   #num_sp-1,d2   ;
00EA 1E19      142 scruber   move.b  (a1)+,d7       ; Co-ordinates of invaders popped from
00EC 3C18      143           move.w  (a0)+,d6       ; array
00EE 6B02      144           bmi.s   nninv          ; If deceased invader then try next one
```

```
00F0 61D4      145         bsr.s   blank_out       ; Deletes old,tired invader
00F2 51CAFFF6  146 nninv   dbf     d2,scruber      ; Next invader for same treatment
00F6 4E75      147         rts                     ;
00F8           148 ;
00F8           149 ; This routine moves the little invaders horribly across the screen
00F8           150 ;
00F8           151 slither
00F8 3B3A022E  152         move.w  accel,d4        ; Moves descent rate into d4
00FC 45FA0232  153         lea     sd_flag,a2      ; These 3 lines determine whether the
0100 4A12      154         tst.b   (a2)            ; invader's current move is across
0102 663C      155         bne.s   mo_down         ; or down the screen
0104 43FA0228  156         lea     xdir,a1         ; These two lines move the direction
0108 1011      157         move.b  (a1),d0         ; into d0
010A 1600      158         move.b  d0,d3           ;
010C 41FA01B0  159         lea     xpos,a0         ; a0 now points to x pos in inv array
0110 7A00      160         moveq   #0,d5           ;
0112 7234      161         moveq   #num_sp-1,d1    ;
0114 3410      162 shuff1 move.w  (a0),d2         ; x-pos is shifted into d2
0116 6B1C      163         bmi.s   deader          ; if it's negative then try next inv.
0118 4A00      164         tst.b   d0              ;
011A 6B04      165         bmi.s   mo_rite         ; If direction negative then move right
011C 9444      166         sub.w   d4,d2           ; Otherwise move left
011E 6002      167         bra.s   no_rite         ;
0120 D444      168 mo_rite add.w   d4,d2           ; Move right
0122 0C4201F3  169 no_rite cmp.w   #499,d2         ; Test if invader is at the far right
0126 6206      170         bhi.s   dir_ch          ; If so, change direction flag (d3)
0128 0C420003  171         cmp.w   #3,d2           ; Same for far left
012C 6206      172         bhi.s   deader          ;
012E 1600      173 dir_ch  move.b  d0,d3           ; This changes direction flag which
0130 4603      174         not.b   d3              ; changes their direction next time
0132 7AFF      175         moveq   #-1,d5          ; This forces descent next time
0134 30C2      176 deader  move.w  d2,(a0)+        ; Moves new position back to array
0136 51C9FFDC  177         dbf     d1,shuff1       ; Next invader, please
013A 1283      178         move.b  d3,(a1)         ; This sets next horizontal direction
013C 1485      179         move.b  d5,(a2)         ; This stores the side/down flag i.e.
013E 4E75      180         rts                     ; if it's -1 they go down next time
0140 41FA0146  181 mo_down lea     ypos,a0         ; - - - - - - - - - - - - - - - - -
0144           182                                 ; This section does the same
0144 7034      183         moveq   #num_sp-1,d0    ; thing for the vertical movement
0146 1210      184 drop    move.b  (a0),d1         ; as the previous section did for
0148 0C0100FF  185         cmp.b   #-1,d1          ; the horizontal.
014C 6710      186         beq.s   notendg         ;
014E D204      187         add.b   d4,d1           ;
0150 1081      188         move.b  d1,(a0)         ; EXCEPT!!!
0152 0C0100E0  189         cmp.b   #224,d1         ;  ...if an invader descends below a
0156 6306      190         bls.s   notendg         ; given level (set at 224) then
0158 DEFC0004  191         add.w   #4,a7           ; return address is taken off stack
015C           192 ;        bra     endgame        ; and this branches to endgame (more
015C 4E75      193         rts                     ; on this soon!)
015E D0FC0001  194 notendg add.w   #1,a0           ;
0162 51C8FFE2  195         dbf     d0,drop         ;
0166 4212      196         clr.b   (a2)            ;
               197         ...

016A           198 ;
016A           199 ; This puts the invaders on the screen
016A           200 ;
016A           201 paint
016A 41FA0152  202         lea     xpos,a0         ; a0 and a1 become pointers to the x
016E 43FA0118  203         lea     ypos,a1         ; and y arrays (invader positions)
0172 7A00      204         moveq   #0,d5           ;
0174 1A3A01C4  205         move.b  flick,d5        ; Flick stores invader type
0178 7034      206         moveq   #num_sp-1,d0    ;
017A 1E19      207 brush   move.b  (a1)+,d7        ; Pops y position into d7
017C 5405      208         addq.b  #2,d5           ; Selects the type of invader to
017E 02050007  209         and.b   #7,d5           ; display next
0182 3C18      210         move.w  (a0)+,d6        ; Pops x position into d6
0184 6B04      211         bmi.s   dead            ; If x pos is negative (dead) then try
0186 6100001A  212         bsr     plot            ; next one else plot current invader
018A 51C8FFEE  213 dead    dbf     d0,brush        ; Next invader, please
018E 4E75      214         rts
0190           215 ;
0190           216 ; This routine clears the screen to black
0190           217 ;
0190           218 clear
0190 247C0002  219         move.l  #scr_top,a2     ; This clears 8192 long words
0194 0000
0196 323C1FFF  220         move.w  #8191,d1        ; which clears the screen
019A 429A      221 resnlw  clr.l   (a2)+           ;
019C 51C9FFFC  222         dbf     d1,resnlw       ;
01A0 4E75      223         rts
01A2           224 ;
01A2           225 ;  This routine enables sprite appearance or disappearance using EOR
01A2           226 ;  Sprite number (d5) is "exclusively or"ed onto (or off)the screen at
01A2           227 ;  d6 (x co-ord), d7 (y co-ord)
01A2           228 plot
01A2 48E737C0  229         movem.l d2-d3/d5-d7/a0-a1,-(a7)
01A6 6138      230         bsr.s   calc_addr       ; See calc_addr below
01A8 4402      231         neg.b   d2              ; Works out which pixels each
01AA 06020008  232         add.b   #8,d2           ; invader thing will occupy
01AE 41FA005B  233         lea     sprite_defs,a0  ; These lines calculate the position
01B2 E94D      234         lsl.w   #4,d5           ; of the sprite definition in memory
01B4 D1C5      235         add.l   d5,a0           ; and deposit it in A0
01B6 7A07      236         moveq   #7,d5           ; Setting up the count-down
01B8 4246      237 loop2   clr.w   d6              ;
01BA 4247      238         clr.w   d7              ;
01BC 1C18      239         move.b  (a0)+,d6        ; Sprite definitions go into d6 & d7
01BE 1E18      240         move.b  (a0)+,d7        ;
01C0 E56E      241         lsl.w   d2,d6           ; Moves the Green bits into position
01C2 E56F      242         lsl.w   d2,d7           ; Same for Red
01C4 3606      243         move.w  d6,d3           ; These lines assemble the words in
01C6 E18B      244         lsl.l   #8,d3           ; d6 and d7 into a long word and stores
01C8 3607      245         move.w  d7,d3           ; it in d3
01CA 1606      246         move.b  d6,d3           ;
01CC E18B      247         lsl.l   #8,d3           ;
01CE 1607      248         move.b  d7,d3           ;
01D0 B791      249         eor.l   d3,(a1)         ; See text
01D2 D2FC00B0  250         add.w   #128,a1         ; Screen pointer moves down one line
01D6 51CDFFE0  251         dbf     d5,loop2        ; Get next bytes (goto loop2)
01DA 4CDF03EC  252         movem.l (a7)+,d2-d3/d5-d7/a0-a1
01DE 4E75      253         rts
01E0           254 ;
01E0           255 ; This routine calculates the address of a screen word pointed to
01E0           256 ; by d6 (x co-ord) and d7 (y co-ord). The address is returned in a1,
01E0           257 ; d6 and d7 are scrambled  & d2 returns with the bit number (0 - 7).
01E0           258 ;
01E0           259 calc_addr
01E0 02B60000  260         and.l   #$1ff,d6        ; These lines confine everything to
01E4 01FF
01E6 02B70000  261         and.l   #$ff,d7         ; the screen boundaries
01EA 00FF
01EC 3406      262         move.w  d6,d2           ;
01EE E44E      263         lsr.w   #2,d6           ; Divides contents of d6 by 4
01F0 0246FFFE  264         and.w   #$fffe,d6       ; Yields an even number as answer
01F4 06B60002  265         add.l   #scr_top,d6     ; Moves d6 to screen memory (see text)
01FB 0000
01FA EF4F      266         lsl.w   #7,d7           ; Multiplies d7 by 128 (down d7 lines)
01FC DEB6      267         add.l   d6,d7           ; Calculates full on-screen address
01FE 2247      268         move.l  d7,a1           ;
0200 02B20000  269         and.l   #7,d2           ; Calculates the bit number
0204 0007
0206 4E75      270         rts                     ;
0208           271

0208           272 ;
0208           273 ; Sprite definitions
0208           274 ;
0208           275 sprite_defs
0208           276 ; Sprite # 0            invader 1a
0208 19183D3C  277         dc.w    $1918,$3d3c,$5b7e,$ffff,$d66a,$ea56,$a424,$2424
020C 5B7EFFFF
0210 D66AEA56
0214 A4242424
0218           278 ; Sprite # 1            invader 1b
0218 9818BC3C  279         dc.w    $9818,$bc3c,$da7e,$ffff,$576a,$6b56,$2524,$4242
021C DA7EFFFF
0220 576A6B56
0224 25244242
0228           280 ; Sprite # 2            invader 2a
0228 91905258  281         dc.w    $9190,$5258,$1438,$1030,$77ff,$1018,$0008,$0008
022C 14381030
0230 77FF1018
0234 00080008
0238           282 ; Sprite # 3            invader 2b
0238 81885258  283         dc.w    $8188,$5258,$341c,$140c,$00fe,$341c,$5212,$9111
023C 341C140C
0240 00FE341C
0244 52129111
0248           284 ; Sprite # 4            invader 3a
0248 9818BC3C  285         dc.w    $9818,$bc3c,$da7e,$ffff,$576a,$6b56,$2524,$4242
024C DA7EFFFF
0250 576A6B56
0254 25244242
0258           286 ; Sprite # 5            invader 3b
0258 19183D3C  287         dc.w    $1918,$3d3c,$5b7e,$ffff,$d66a,$ea56,$a424,$2424
025C 5B7EFFFF
0260 D66AEA56
0264 A4242424
0268           288 ; Sprite # 6            invader 4a
0268 81885258  289         dc.w    $8188,$5258,$341c,$140c,$00fe,$341c,$5212,$9111
026C 341C140C
0270 00FE341C
0274 52129111
0278           290 ; Sprite # 7            invader 4b
0278 91905258  291         dc.w    $9190,$5258,$1438,$1030,$77ff,$1018,$0008,$0008
027C 14381030
0280 77FF1018
0284 00080008
0288           292 ;
0288           293 ;
0288           294 ;
0288           295         align
0288 00000035  296 ypos    ds.b    num_sp
028E 00000035  297 xpos    ds.w    num_sp
0328           298         align
0328 0000      299 accel   dc.w    0
032A 0000      300 gunpos  dc.w    0
032C 0000      301 in_wait dc.w    0
032E 0001      302 xdir    dc.w    1
0330 0000      303 sd_flag dc.w    0
0332 09010000  304 temp    dc.l    $09010000,$00000002
0336 00000002
033A 0000      305 flick   dc.w    0
033C 0000      306 bulposx dc.w    0
033E 00        307 bulposy dc.b    0
033F 00        308 y_bomb  dc.b    0
0340 0000      309 x_bomb  dc.w    0
0342           310 inv_speed
0342 00        311         dc.b    0
0343 0A        312 order   dc.b    10
0344 0B        313         dc.b    11
0345 0C        314         dc.b    12
0346 0D0E1A0F  315         dc.b    13,14,26,15
034A 1B1C1E1D  316         dc.b    27,28,30,29
034E 1F202221  317         dc.b    31,32,34,33
0352           318         align
0352 00000000  319 seq_reg dc.l    0
0356 0000      320 wumpx   dc.w    0
0358 00        321 wumpy   dc.b    0
0359 00        322 seq_num dc.b    0
035A 00        323 num_bas dc.b    0
035B 00        324 invnext dc.b    0
035C 01        325 badwait dc.b    1
035D 00        326 numflag dc.b    0
035E           327         align
035E           328 mask_tab
035E B0804040  329         dc.w    $8080,$4040,$2020,$1010
0362 20201010
0366 0B0B0404  330         dc.w    $0B0B,$0404,$0202,$0101
036A 02020101
036E 00000000  331 mess    dc.l    0
0372 00000000  332 hiscore dc.l    0
0376 00000000  333 score   dc.l    0
037A 0000      334 baddyx  dc.w    0
037C 0000      335 rndpos  dc.w    0
037E           336  end


SYMBOLS :

rndpos                  R0000037C  hiscore                 R00000372
mess                    R0000036E  mask_tab               R0000035E
badwait                 R0000035C  wumpy                  R00000358
wumpx                   R00000356  seq_reg                R00000352
order                   R00000343  x_bomb                 R00000340
y_bomb                  R0000033F  bulposy                R0000033E
bulposx                 R0000033C  temp                   R00000332
gunpos                  R0000032A  loop2                  R000001B8
sprite_defs             R00000208  resnlw                 R0000019A
plot                    R000001A2  dead                   R0000018A
brush                   R0000017A  notendg                R0000015E
drop                    R00000146  dir_ch                 R0000012E
no_rite                 R00000122  mo_rite                R00000120
deader                  R00000134  shuff1                 R00000114
mo_down                 R00000140  nninv                  R000000F2
scruber                 R000000EA  wipe                   R000000D0
calc_addr               R000001E0  blank_out              R000000C6
newtlet                 R000000B0  newt                   R000000AC
froglet                 R0000008E  frog                   R0000008A
ypos                    R000002B8  xpos                   R0000028E
new_screen              R00000074  invnext                R0000035B
score                   R00000376  numflag                R0000035D
sd_flag                 R00000330  baddyx                 R0000037A
xdir                    R0000032E  seq_num                R00000359
accel                   R0000032B  num_bas                R0000035A
flick                   R0000033A  slither                R000000F8
un_paint                R000000E0  inv_speed              R00000342
no_inv                  R0000002C  in_wait                R0000032C
no_mum                  R0000000C  repeat                 R0000000C
paint                   R0000016A  clear                  R00000190
dimbo                   R0000002E  warm                   R00000000
coldstart               R00000000  nemalt                 C0000000F
bas_pos                 C00000005  score_pos              C000001E0
score_poshi             C00000032  ut_mtext               C000000D0
ut_con                  C000000C6  num_base               C00000010
num_sp                  C00000035  scr_top                C00020000
```

THE SINCLAIR QL SERIES

Series Editor: Robin Bradbeer
Foreword by Nigel Searle,
Managing Director, Sinclair Research

recommended by "QL User" magazine

at last the complete QL library

NEW **Machine Code Programming on the Sinclair QL**
Martin Gandoff

NEW **Database Management on the Sinclair QL**
Mike O'Reilly

NEW **Using Graphics on the Sinclair QL**
Garry Marshall

NEW **Making the Most of the Sinclair QL**
QL SuperBasic and its applications
Dick Meadows

NEW **Profiting from the Sinclair QL**
financial and other spreadsheet applications
Barry Miles

192 pages illustrated £7.95 paperback

fully up-to-date with
Psion Systems version 2.00 software

previously published in the series

Introducing the Sinclair QL
Introduction to SuperBasic on the Sinclair QL
Advanced Programming with the Sinclair QL
Word Processing with the Sinclair QL
Desk Top Computing with the Sinclair QL

192 pages illustrated £6.95 paperback

Order now! Just fill in the order form and return to us with your cheque or money order. Your books will be sent POST FREE in the UK, within 28 days.

ORDER FORM

To: Hutchinson Computer Publishing
17-21 Conway Street, London W1P 6JD

Please send me:

..... copies of **Machine Code Programming on the Sinclair QL**	£7.95 each	09 160531 8
..... copies of **Database Management on the Sinclair QL**	£7.95 each	09 160551 2
..... copies of **Using Graphics on the Sinclair QL**	£7.95 each	09 160521 0
..... copies of **Making the most of the Sinclair QL**	£7.95 each	09 160561 X
..... copies of **Profiting from the Sinclair QL**	£7.95 each	09 160541 5
..... copies of **Introducing the Sinclair QL**	£6.95 each	09 158941 X
..... copies of **Introduction to SuperBasic on the Sinclair QL**	£6.95 each	09 158951 7
..... copies of **Advanced Programming with the Sinclair QL**	£6.95 each	09 158961 4
..... copies of **Word Processing with the Sinclair QL**	£6.95 each	09 158971 1
..... copies of **Desk Top Computing with the Sinclair QL**	£6.95 each	09 158981 9

Overseas orders — add £1 p&p per title.

I enclose my cheque/money order made payable to TBS for

£_____

My name and address is _____

_____

_____

Signed _____

HCP

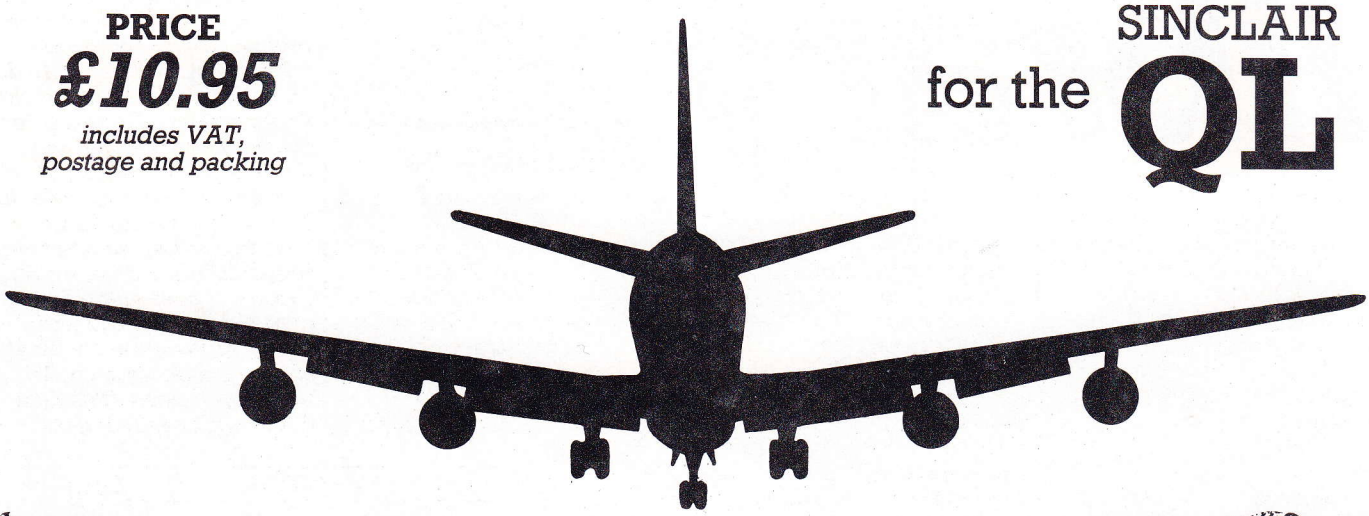
NEW

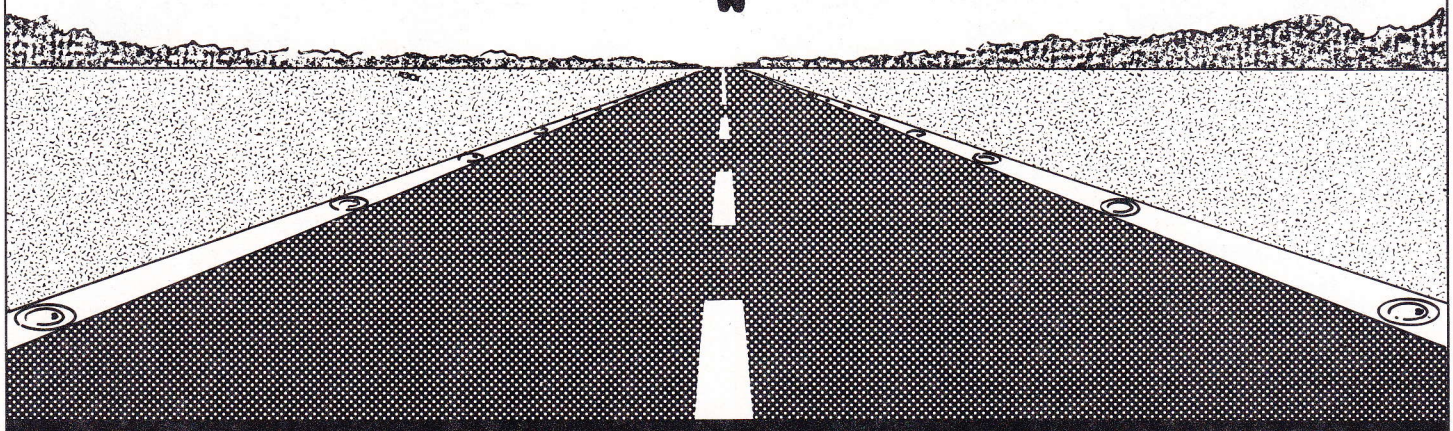
SHADOW SOFT

AREA RADAR CONTROLLER

PRICE
£10.95
includes VAT, postage and packing

for the SINCLAIR **QL**





Please send me AREA RADAR CONTROLLER on microdrive cartridge for the QL

Name .....

Address .....

.....

.....

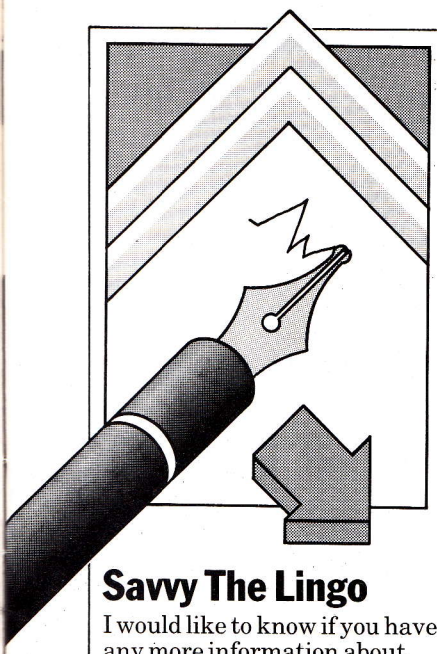
QLU3

Please make cheque or postal order payable to Shadow Soft and send to:
Shadow Soft, 70 Gooseacre, Cheddington, Near Leighton Buzzard, Beds. 0296 668740

A fascinating game and simulation. Control up to 69 aircraft in a busy air traffic zone which includes 2 airfields and 9 airways. Don't run out of fuel and avoid collisions.

VERY ADDICTIVE — HIGHLY FRUSTRATING

QL PROGRAMMERS. *Want someone to market your game? — send it to us for assessment; you will not be ripped off. At worst it will be returned uncopied, at best you will be offered a contract at good royalty rates.*



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.

Savvy The Lingo

I would like to know if you have any more information about Fortran 77 for the QL as I am learning to program in that language at college and it would be a bonus if I could practice programming at home.

Is it possible to order back issues of *QL User*, and if so can you tell me how many issues there have been, where I can get the issues from and how much they will cost?

How about doing an article about the use of electric typewriters with computers?

S Garlick
London E15

The QL currently supports APL, BCPL, Pascal, FORTH and LISP. In the near future you can expect C and PROLOG but, as far as we know, nobody is interested in getting FORTRAN 77 on the QL. Despite new releases FORTRAN along with COBOL would appear to be regarded as 'old hat'.

Next month we plan to review two computer compatible typewriters from Brother: the EP 44 and the TC 600. Finally, for back orders and subscriptions contact Carl Dunne 01-251 6222 ext 2429.

Bug Or Blunder

The Abacus 'Utilities' program in the February 'QL Horizons' article failed to work until "if errnum( ) = 100: etc" was changed to "if edrrnum( ) <> 100: etc". Even now the *ffind* will not work. It is not known where the QL went – it did not return unaided. Sc has been replaced by 'display', as this seems both simpler and adequate for my database.

Since the messages 'No such record, etc' did not remain on screen long enough to read, the prog has been modified to wait until action is taken to continue.

I still await the "up to the minute information" which lead me to return the registration card from your Aug/Sept issue. Although it is obviously slower than the Psion reply service, perhaps your system will come up with something useful – which Psion have so far failed to do.

J A Patrick
Shrops

The utilities program in question has been in constant use for the last three months. The only problem encountered arose when transferring across to Archive Version 2.00. The SC procedure appeared to fail. The remedy was either to run the SCRCON_BAS program supplied by Psion or to replace the call as you have done. The latter has the disadvantage that amendment routines are now less user-friendly.

As regards your query concerning QL User registration we are currently looking into developing a microdrive addendum to the magazine. Its all a little "hush hush" at the moment but provided we can overcome certain logistical problems (ie microdrives) this will be made available, either exclusively or at a discount, to those who have registered.

Fast Copy

One facility missing from Quill is the ability to print multiple copies of a document. The solution is to do it from SuperBasic. Write and format your document in the usual way, and print a copy to check that it is correct. Then, with the correct printer driver still operative, print to the microdrive cartridge in mdv2_ with the command:

<F3> Print, current, whole to mdv2_document . . . Now quit Quill.

If you examine the files on mdv2_ with DIR, you will find one called *document_lis*, and if you execute the command: **COPY_N mdv2_document_lis TO ser1c**, your printer should print it out exactly as it would from Quill. By incorporating the command in a FOR NEXT or REPeat loop, you can have complete control of multiple printing with whatever facilities you want. I have been using a version AH QL and 1.03–ACB Quill.

G J W Cunliffe
Hersham, West Sussex

Sudden Reversal

Having purchased the Psion Chess for my QL I was astounded by the graphics for the 3D representation of the chess board. What I would like to know is, can user definable graphics be controlled from a SuperBasic program, and if so how do I access them? I would be grateful if you could help me because a friend of mine who has a Commodore 64 keeps on bragging how he can define his own graphics.

A QL owner
Leeds

Your in luck! Whilst there are no built-in procedures in SuperBasic for designing and animating sprites, in 'The Progs' this month we include a set of routines which will enable you to do exactly this. If you read the section carefully you'll see that each sprite may be moved by altering parameters within a block. The simple Lunar Lander simulation shows what can be achieved. However, for those not prepared to experiment, the diagram below shows exactly what each parameter in the 16 byte block does. It's just a few simple techniques.

GENERAL DATA		BYTE
num	number of sprites. 0 to 255 (0=routine off)	0
this	internal use only	1

PARAMETER BLOCK		BYTE
status	sprite on/off 255= on : 0= off	0
new	setting this to zero tells the routine not to erase the character as it is not on the screen. (characters are printed and erased by EORing for those in the know)	1
x	x position. set to start value then peek only unless you set 'new' to zero	2
y	y position (see above)	3
xsize	size of character in x direction in groups of four pixels (as used with the sprite definer)	4
ysize	size of character in y direction in pixels	5
xspeed	distance to move in x direction —128 to +127 (2's complement)	6
yspeed	distance to move in y direction (see above)	7
vedge	action to take when a vertical edge is hit:— 0 = stop —1 = wrap round screen 1 = bounce off	8
hedge	action to take when a horizontal edge is hit:— 0 = stop —1 = wrap round screen 1 = bounce off	9
chars	number of frames of animation for this sprite	10
current	current animation frame being used	11
data	address of the data for each frame of animation (the data produced by the sprite definer) The data for each frame should be directly after the data for the frame before	12—15

Out Of Line

As a simple minded mechanical engineer plagued by the unreliability of the microdrives on my QL I decided to have a look at their mechanical aspects. My problem was that the two drives declined to read each others' writing. Moreover microdrive 2 was not interested in the bundled Psion programs. On peering into the black holes at the front of the machine I discovered something interesting. Through all the black dust scraped off the side of the cartridges I could see that the tape heads were at different angles. Daring to remove the cover I found that the head alignment was controlled by moving the circuit board to which the head was soldered, relative to the bottom plate of the microdrive, the two being held together by a solitary self-tapper which screwed not even into a hole but a slot. Furthermore, the cartridge is held down in position by ribs on the underside of the computer's top cover, which means that the case must be produced to very tight tolerances.

As I had no wish to invalidate my guarantee I did not interfere further but on reassembling the machine I discovered that as I tightened the screws nearest microdrive 2 some distortion took place. Leaving these screws undone meant that the cartridge was not forced out of true when inserted and the two microdrives now converse freely. Simple, eh?

F J Lucy
Maldon, Essex

Canon Code

While connecting a Canon PW-1080A printer to my QL, I came across a couple of points which may interest other readers.

I set the printer's dipswitches for the US character set, as I wanted the "#" sign to appear in program listings.

When printing from Abacus or Quill, however, I wanted to use the English character set with its "£" sign. I tried setting the preamble code in the printer driver to <ESCb,@,<ESC>,R,3 (as in the QL manual), but the printer obstinately stayed in US mode. Eventually, I realised that the preamble code should be as follows:

PREAMBLE CODE :
<ESC>,@,<ESC>,R,<ETX>
(Type esc,"@,esc,"R,3)

Although this made the "£" sign print, I was still annoyed that I couldn't use both "£" and the "#" with the word processor.

As a last resort I tried reading the printer manual, and lo! I found that it is possible to extend the printer's character set by using ASCII codes normally reserved as control codes.

Thus, code 6 becomes "£", while 23 remains as "#". To set this up, leave the printer switched for the U.S. set, and amend the printer driver as follows:

PREAMBLE CODE :
<ESC>,@,<ESC>,I,1
(Type esc,"@,esc,"I,"1)

TRANSLATE1: £,<ACK>
(Type "£,6)

John Ennals
London

Jury Rig

I acquired a Brother HR-5 printer to use with my QL. Full of great expectations, I linked the two, using the RS 232 C lead that I received free with my QL. When I tried to print something, nothing happened. Undaunted, I tried using different parities, handshaking, and data types. Still nothing. So I tried reconnecting the leads in the printer end of the lead. I once more got no response. Several times more I did this, but each time the printer did nothing. My parents were by now getting rather disconcerted, and my printer and QL were taken to a company in nearby Loughton who, after three days succeeded.

I hope these findings will be of aid to others.

Allister Chambers
Epping, Essex.

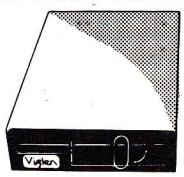
PIN (QL)	WIRE COLOUR	TITLE	PIN (PRT)
1	BLACK	GROUND	7
2	WHITE	T × D	3
3	GREEN	R × D	2
4	BLUE	DTR	20
5	RED	OTS	leave loose
6	ORANGE	DSR	leave loose

Also connect pins 4 & 6, and 8 & 20.

COMPUTER PERIPHERALS
the best service the best value from Viglen

Speed Your Data Access

with a disc drive from

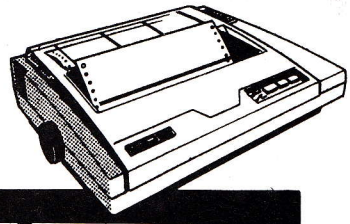


D

Replacing your Microdrive with Disc Drives means instant reliable access **every time.** You need to purchase the disc interface and a drive which includes everything required to plug in and use.

Disc interface	**£129**
200K Single Drive with PSU	**£119**
800K Single Drive with PSU	**£159**
1·5M Dual Drive with PSU	**£299**

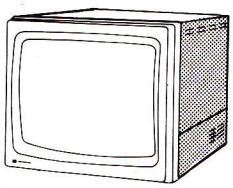
Dot Matrix Printers



Taxan Kaga KP810	**£301**
Canon PW1080A	**£321**
Epson RX80/FT	**£275**
Epson FX80	**£389**
Shinwa CPA80 (100cps)	**£219**
Mannesmann Tally MT80	**£209**

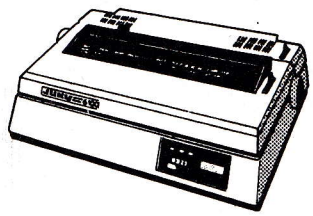
C

Monitors



14" RGB Microvitec
1451 DQ3..... **£275** C

Daisy Wheel Printers

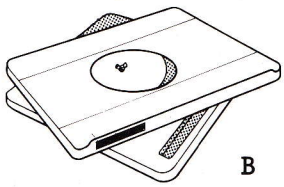


Juki 6100 **£385**

Quen Data **£255** C

QL Serial Centronic I/F Printer Cable £35

Monitor Stand



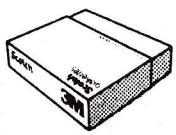
B

Our **Super Low** price **£14.95**

3M Scotch Disc 5¼" diskettes double density

(Lifetime guarantee)

	1-2 boxes	3-9 boxes	10+ boxes
SS 40 TRACK	**£15.00**	**£13.50**	**£13.00**
DS 40 TRACK	**£20.00**	**£18.40**	**£17.70**
SS 80 TRACK	**£23.00**	**£20.70**	**£20.00**
DS 80 TRACK	**£26.70**	**£24.00**	**£23.00**



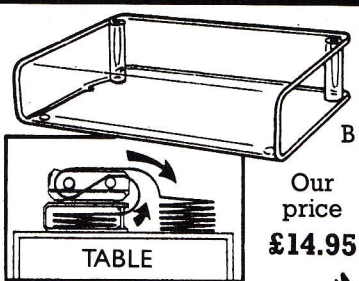
A

10 DISCS PER BOX

Printer Stand

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

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



B

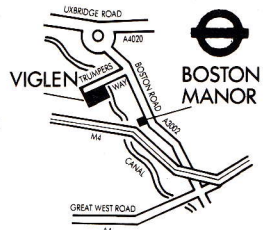
Our price **£14.95**

SHOWROOM OPEN
Weekdays 9.30 - 6pm
Saturdays 9.30 - 4pm

Money-Back Guarantee
If not entirely satisfied with your purchase, simply return to us within 7 days in perfect condition and in original packing including a note stating reason(s) for return.

Please add for carriage:-
A - £2. **B** - £3. **C** - £8. **D** - £12.

ALL PRICES INC. VAT



Post to: Viglen Computer Supplies.
Unit 7, Trumpers Way, Hanwell W7 2QA.
Telephone: 01-843 9903.

Please send me _____

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

Name _____

Address _____

_____

_____

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

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

Card No. _____

Signature _____

ORDERS are usually despatched within 24 hours

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

HARDWARE ON

Ralph Vernon dips into the latest crop of disk and memory systems and finds it's not all fruitful.

Basingstoke based, Microperipherals have made their reputation as major distributors of printers with access to a sizeable dealer network. The QL disk system marks their debut as an independent peripheral manufacturer. The system consists of a disk interface module and two 3.5″ disk drives.

Similar in appearance to CST's interface (reviewed in the March issue) the module consists of a small PCB, with 8K ROM on-board, that slots neatly into the QL's main I/O port adding some three inches to the machine's length.

Conforming with the Sinclair standard, the unit formats 9 bytes per sector with 512 sectors per track and the option of 40 or 80 tracks per disk, single or double sided. Jumpers are used to set the options and are easily accessible on the PCB itself.

In-House Technology

The disk driver was written in-house and owes more to close collaboration with Sinclair Research rather than with QDOS's author Tony Tebby. Nevertheless, a number of extra commands relating to file I/O and job control from the latter's QL Toolkit will be included in ROM.

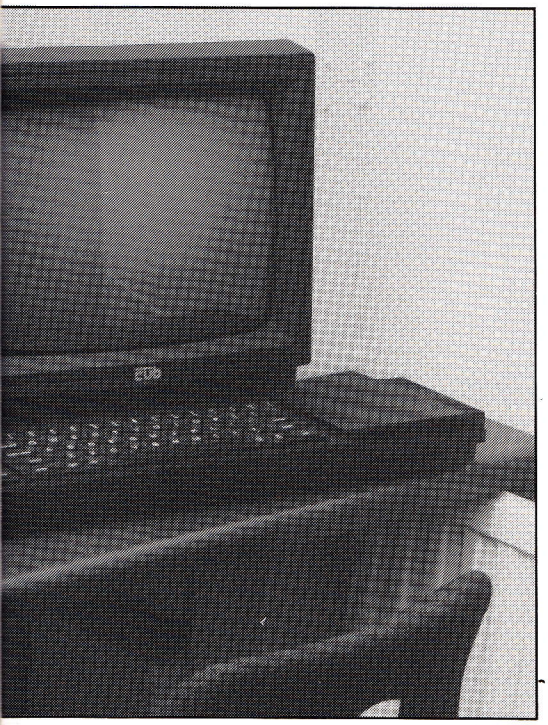
Additionally, Microperipherals have developed a number of utilities themselves, including a microdrive emulator. Here, the commands FSET and MSET determine which device name the disk driver will answer to, 'fdk' or 'mdv' respectively. With FSET, disks operate in the normal fashion. Namely, as separate devices

The MicroPeripherals disk system *in situ* (right) and minus the QL (below). Note the use of 3½″ drives making the set-up very compact.

PHOTOS BY TERRY BEDDIS

THE HORIZON II

working alongside microdrives. With the MSET, QDOS treats floppies as if they are microdrives and ignores the latter altogether. The advantage of this is that all programs that run on microdrive (Psion's packages included) will run on disk without the need for special conversion or configuration programs. Incidentally, this also saves the 1.5K of user RAM that an additional driver would otherwise occupy. Finally, it is possible to have the QL boot up from a 'mdv' named disk by setting a jumper on the interface's PCB.



Bug Hunters

Other EPROM based extensions include extra error messages such as timeout, seek or CRC errors and 'record not found'. On top of these, there is a simple disk editor which will permit tracks and sectors to be examined and corrupted files to be patched up. Also under consideration is a machine code monitor which whilst not a full dissassembler will permit users to manipulate memory and search for strings and bytes.

Microperipherals provide 3.5" Japanese-made drives, specifically designed to go with the QL. Whilst the drives may be purchased singly they are designed to operate in pairs which accounts for the price difference. The first and more expensive drive houses a power supply unit and disk controller, for both itself and a second drive.

As an indication of their committment to this technology, Microperipherals will be launching a 3.5" Winchester in May which will permit up to four of these drives to be plugged in and provide a passable alternative to the horrendously expensive tape streamers. This also means that when the Winchester appears, business users that have purchased the floppies will not be left out in the cold.

The 3.5" floppies are in many respects a distinct improvement upon their 5.25" cousins. Offering the same capacities (80 track – 720K formatted), the disks are almost pocket-sized and, enclosed in a rigid plastic cover, are well protected from the elements. Write-protection simply involves sliding a small panel backwards, as opposed to fooling around with a piece of sticky tape. The spindle about which the disk rotates does not physically clamp onto the floppy but slots in a steel centrepiece, thus minimising contact with the media to the read/write heads alone. In addition, the drives incorporate a warning light that flashes when the disks are worn so that they may be backed-up before it is too late.

In appearance the drive units are less than half the size of the 5.25" drives, housed in black metal casings and conveniently stacked one on top of the other. Seen alongside the QL, they seem less incongruous than the larger version and being lighter, can be more easily transported. The drives will be sold with a utilities disk which will contain Sinclair's official full screen editor and assorted commands from Tebby's QL toolkit.

Finally, for the more technically minded, access times on the disks are 3 milliseconds and flux density is 135 TDI. Also, to counter the smaller disks susceptibility to radio and low frequency interference, the heads are copper shielded. The drives are fully compatible with other QL disk interfaces that are currently available (ie CST).

Judging by the amount of thought that has gone into the design of their disk system, there is little doubt that Microperipherals not only consider the QL add-on market extremely lucrative, but also the user base extraordinarily discerning! Attractively packaged, the system provides a well-rounded set of utilities both in EPROM and on disk which enhance the QL's overall performance, as well as ensure a trouble-free transition from microdrive to disk. Competitively priced, this product will take some beating.

Prices (inc VAT):

Disk Interface	£113.85
3.5" 720K Drive	£194.35
2nd Drive	£171.35

Supplier: Microperipherals Ltd, Intec Unit 3, Hassocks Wood, Wade Road, Hants RG24 0NE (0256 473232)

Further RAMifications

The case for additional RAM for the QL is nowhere near as clearly defined as that for disk drives especially when, in the absence of any expansion board, one excludes the other.

With 128K RAM of which 90K is available for the user, the QL already has more built-in memory than any other similarly priced micro. Furthermore, all the professional software currently available for the QL is geared to run in this environment. However, in some cases extra memory will enhance the software and it is this that provides the only possible justification for RAM extensions.

For example, adding say, 256K to the QL will result in the following increases in the working memory used by Psion's packages:

Archive	12K	129K
Abacus	15K	94K
Easel	8K	125K

In the case of Abacus and Easel, the total number of cells which may be used increases from approximately 1000 to more than 6000 permitting considerably more complex analyses to be undertaken.

With Archive, the extra memory permits much larger files to be stored in memory so that searches and updates can be carried out in seconds, as opposed to tens of seconds.

In all cases, including Quill the extra memory means that the programs themselves are loaded in their entirety so that commands are executed instantaneously, not loaded in on microdrive and overlaid. Also QDOS itself gets in on the act by grabbing more buffer space which ▶

brings about an general improvement in microdrive performance.

Though the advantages of RAM extensions are real it should be borne in mind there is no guarantee that other commercial programs will be written to take advantage of them in the way Psion does. This is especially true now that disk drives are available as an alternative means of enhancing the QL's performance.

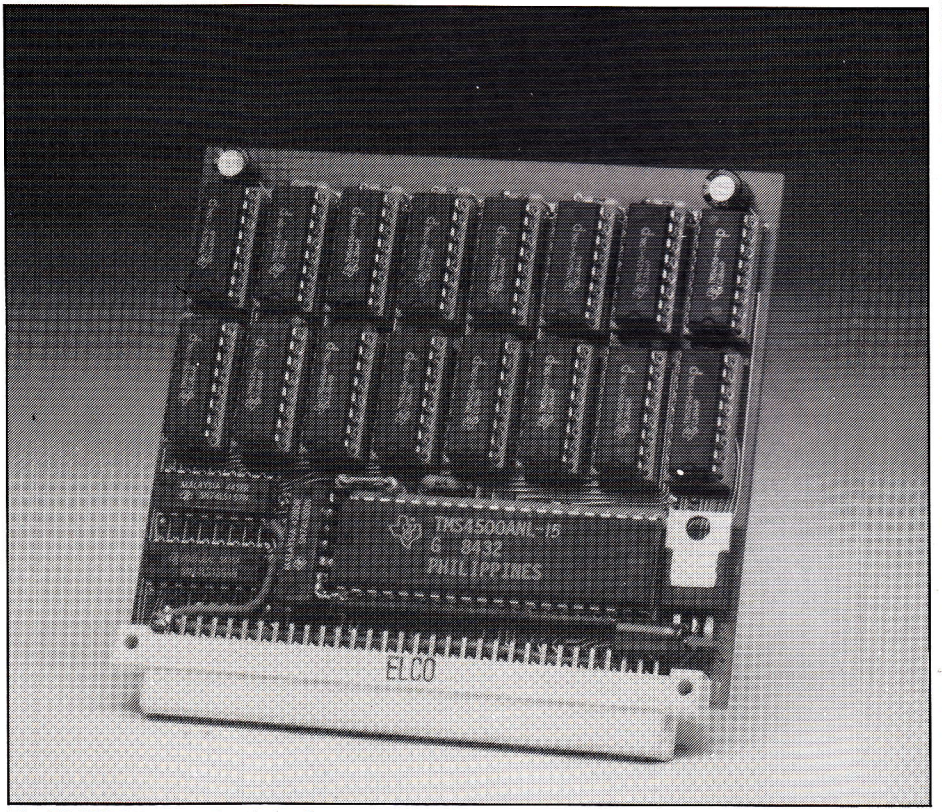
Promising Proposition

Simplex's RAM expansion board slots into the QL's main I/O port. When installed, the board (97mm × 77mm) juts out a centimetre or so, making it impossible to replace the port's plastic cover. This means that whilst the PCB is out of sight, sitting flush with the keyboard's leading edge, it remains exposed. The reason for this, we understand, has more to do with allowing sufficient air to circulate to prevent overheating. However, Simplex have now opted for a fully enclosed ventilated black plastic box that is almost standard with other add-on manufacturers.

Powered off the QL, the PCB incorporates a sophisticated RAM controller and one bank of 8 256K by 1 bit chips (TMS 4256's). A further bank may be added at a cost of £198 to give a total of 512K RAM. The 256K dynamic RAM chips are the latest and account for the unit's remarkably compact size and low power consumption. Unlike conventional static RAM they can operate in two states: "refresh", which uses virtually no power and, "enable". Switching banks between these two states provides, in the case of the 512K unit, the only way of keeping in the 3400 milliwatt limit on the QL. In fact, this unit's consumption is just 3100 milliwatts.

For review purposes, we opted to test the 512K board, as out of the two this would be the more susceptible to spikes. Once installed the unit worked without hitch, even when the QL was left on overnight. The only visible evidence of its presence outside Psion's packages was a slight delay before the TV or Monitor selection display came up. Our only reservation was however, that without a heat sink the unit did get quite hot though this did not interfere with its performance.

Overall, the SDL RAM expansion board lives up to the promises made by its manufacturer, being compact and easy to use. Despite minor defects in its design which we hope will soon be rectified, it represents value for money and should be an attractive proposition for those looking to get more out of Abacus, Archive and Easel. Finally, should some enterprising programmer decide to write a RAM disk driver for this unit, this would not go amiss.



The 512K RAM extension from Simplex Data.

Prices (inc VAT):
256K RAM £198 512K £396

Supplier: Simplex Data Systems, 432 Greenford Road, Greenford, Middx. (01 575 7531)

Paper Heavyweight

Quest's memory expansion board (247mm × 97mm) is quite a large affair, reminiscent of the kind that used to be hidden away *inside* the last generation of desk-top computers. It comes in a black plastic casing and plugs into the QL's main I/O port.

To their credit, Quest give the user a wider choice than most producing 64K, 128K, 256K and 512K units. These use relatively conventional 64 by 1 DRAMs (TMS 4164s) in varying numbers from 4 to 32. This does not affect performance but the higher power consumption has meant more support chips all round, and an extra power supply unit for the 512K unit. The extra PSU, we understand, should be phased out shortly. This being the case, it will be interesting to see how far below the QL power limit the unit falls.

Where Quest would appear to have an edge is in the fact that their units may double up as RAM disks. Each unit comes supplied with a microdrive cartridge, containing a short 2K machine code addendum to QDOS which allows you to use extra RAM as though it were a number of microdrives. Ideally, this means that you can enjoy the benefits of near instantaneous access for all filing operations.

However, whilst attractive on paper, in practice the benefits of RAM disks are nowhere near as pronounced. The need to transfer everything to microdrive at the end of a session and then load it in at the beginning means that any real savings arise only when the QL is used for a prolonged period of time. Furthermore, as Quest's driver comes on a cartridge instead of an EPROM, as one would expect, it also needs to be loaded everytime the facility is to be used.

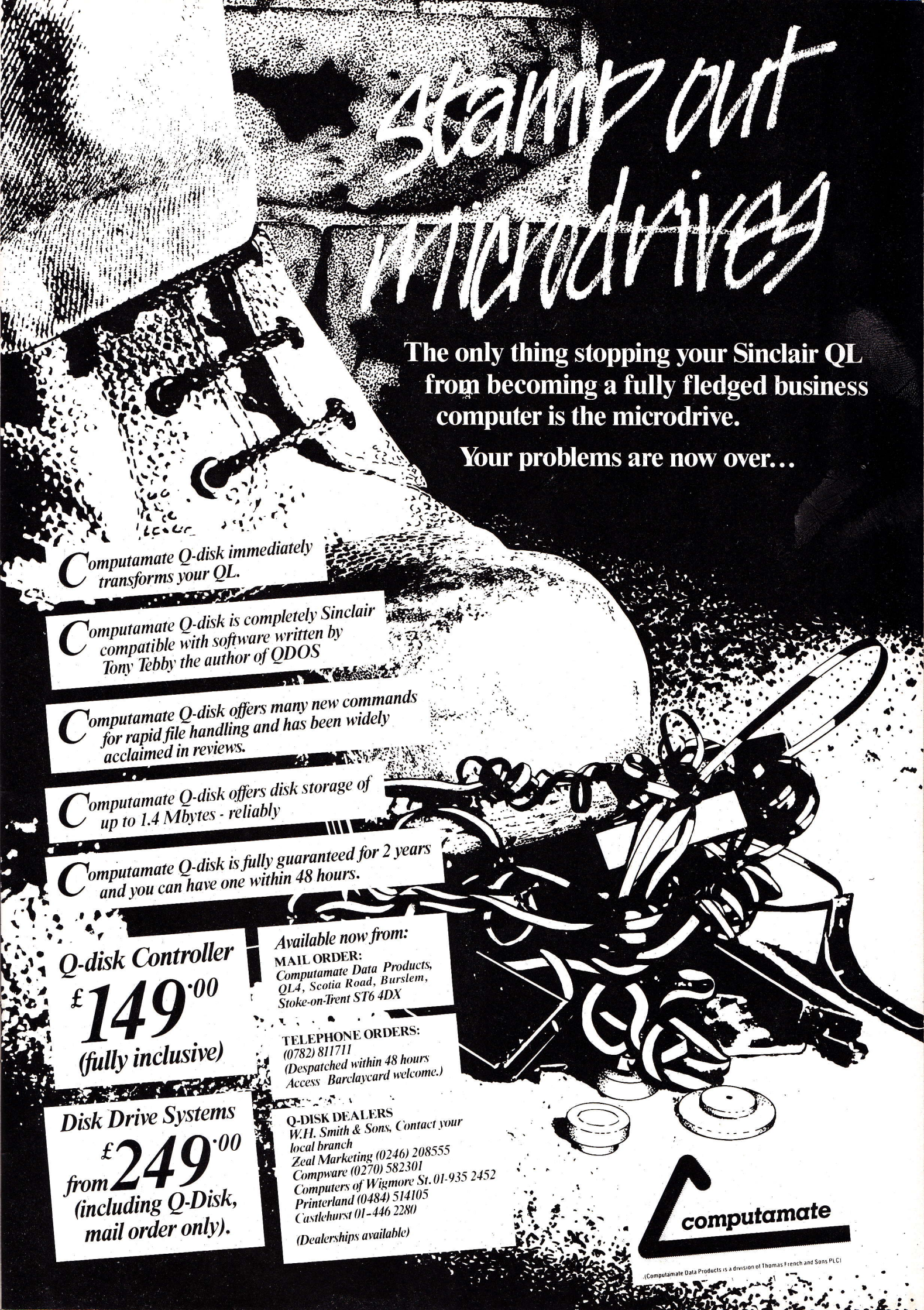
The first to commit themselves to the QL, Quest have been unable to take advantage of improvements in DRAM technology. Whilst this has not affected performance it has meant compromises in both design and cost.

The inclusion of RAM disk software though welcome fails to offset this. However, as with a lot of Quest hardware — the final version may be different.

Prices (inc VAT):

64K	£115	256K	£349
128K	£185	512K	£579

Supplier: Quest International Computers, School Lane, Chandlers Ford, Hants. (04215 66488)



The only thing stopping your Sinclair QL from becoming a fully fledged business computer is the microdrive.

Your problems are now over...

Computamate Q-disk immediately transforms your QL.

Computamate Q-disk is completely Sinclair compatible with software written by Tony Tebby the author of QDOS

Computamate Q-disk offers many new commands for rapid file handling and has been widely acclaimed in reviews.

Computamate Q-disk offers disk storage of up to 1.4 Mbytes - reliably

Computamate Q-disk is fully guaranteed for 2 years and you can have one within 48 hours.

Q-disk Controller
£149.00
(fully inclusive)

Available now from:
MAIL ORDER:
Computamate Data Products, QL4, Scotia Road, Burslem, Stoke-on-Trent ST6 4DX

TELEPHONE ORDERS:
(0782) 811711
(Despatched within 48 hours
Access Barclaycard welcome.)

Disk Drive Systems
from £249.00
(including Q-Disk, mail order only).

Q-DISK DEALERS
W.H. Smith & Sons, Contact your local branch
Zeal Marketing (0246) 208555
Compware (0270) 582301
Computers of Wigmore St. 01-935 2452
Printerland (0484) 514105
Castlehurst 01-446 2280

(Dealerships available)

computamate

(Computamate Data Products is a division of Thomas French and Sons PLC)

MIND-YOUR-O

Has the QL's business potential been underestimated? John Lang thinks so, and his prosperous insurance consultancy testifies to that fact. Here's a QL business success story . . .

Any computer in the business environment, no matter what its cost, is completely useless without relevant software. In the harsh world of microcomputer sales this elementary fact appears to have been forgotten by many firms trying to sell to the small business. The average businessman is simply not interested in ROMs, RAMs, Kbytes etc. All he wants to know is whether the machinery and software will run the various repetitive and time consuming aspects of his business, and at the same time increase efficiency and effectiveness.

So, why purchase a QL for business amid all the adverse publicity (some justifiable and some very definitely not)? Quite simply – for the software potential. As a small firm of insurance consultants, we had been looking for a superior software package to computerise all our motor insurance quotations – a very time consuming affair. Dataller Computer Services of Wigan approached us with a program into which we could feed all the required parameters for a motor quotation and a few seconds later the QL would produce a list of insurers and premiums in ascending order. This was just what we were looking for. To my knowledge this is the only system which comes with four business software programs included in the price of the computer. Provided these worked, we would be in a postition to dip our toes in the water of computerisation without spending thousands of pounds.

The novelty soon wore off . . . the job of entering clients' records is soul-destroying!

The system, consisting of QL, Zenith monochrome monitor, motor quotation software and Psion software was delivered in August. The computer is in operation nine hours a day, five days a week and the insurance system works perfectly. Outside working hours the chance to test Psion's software could be utilised. First impressions were that Archive would be of immediate use in compiling data concerning clients and their policy details. The ease with which files could be created was impressive. However, planning the file on paper first is important as there does not appear to be a way of adding fields to a file or removing them once they have been created and contain records. If there is a way, it is not made clear in the manual. This initial 'creation' took a few hours, sorting out which fields were essential, discarding those useful but not absolutely necessary and deciding how to enter the records so they could be retrieved later. This planning was essential for the data file to be of the most use. On at least three occasions 20 records were entered and then scrapped because the fields had not been thought out carefully enough for retrieval purposes. This may seem elementary to the programmer but to the novice, trial and error is the only way. For example, if you enter all names in the form – Bloggs, J – and then enter a name omitting the comma you will never find Smith, P unless you can remember that on that record you omitted the comma and should be looking for Smith P.

Once the files were created satisfactorily, the job of entering all the clients' records began. The novelty soon wore off and it became apparent why so many businesses with a client or customer base computerise their records over periods which can be as long as a year. The job is mind-blowingly boring! As the record base grew, Archive's power showed with

ILLUSTRATION BY SUE WOOLNOUGH

WN-BUSINESS!

the ease that records and fields could be sorted and located. Initial fears of the microdrives subsided and the carping criticism of these inexpensive storage devices seemed petty. However, the records did get lost once due to not 'closing' the file before removing the cartridge. It is quite true that you only lose your records once – a back-up copy from then on was a must. At this stage Archive's back-up was unreliable so SuperBasic was used for back-up copies, ie **delete mdv1_file_dbf** and then copy **mdv2_file_dbf** to **mdv1_file_dbf.**

As the database grew we wanted to use it for more than just a card index system.

One of the most tedious jobs organisations have is knowing whose payments, renewals, subscriptions etc are due at any one time. As the database grew, we wanted to use it for more than just a card index system, ie a program to list all our renewals as they became due each month. At this point the manual proved a let down. To the uninitiated, the chapter on programming is in places unclear and muddled. But if we could successfully write this procedure it would save a great deal of time. By juggling about with various lists from the manual the required list was produced. Unfortunately it simply kept on scrolling off the screen until only the last twelve names were left. At that time we had no printer and it was essential to stop the program when the screen was full in order to write down the data and then continue until eventually the list was exhausted. This is where the programmer comes into his own. It took a week trying out different statements in different orders before we achieved success. The following program does work and anyone with an organisation that needs this sort of information will find it useful. The fields can be altered to suit each particular application. The 'lprint' statement has been added since we installed a printer. The benefits of Archive were now beginning to show in terms of increasing efficiency. The microdrive cartridges do have their limitations, however, as finding a particular record now takes between two and three minutes. There are approximately 500 records on each cartridge. The time spent writing this program for Archive could not be justified if the QL is to be a business computer. An idea would be for Psion to produce listings of the most common requirements so they can simply be typed in. Businessmen are not interested in how *it* works – simply that *it* does!

Abacus was the next to be tried. Primarily due to lack of knowledge of what a spreadsheet actually was it appeared to be a lot of trouble to design what was needed and make the necessary entries. By persevering with the manual, various formulae were entered into an exact copy of my cash receipts and payments book. Possibly not the most efficient way, but it seemed easier to transfer something which was already familiar. The beauty of this particular spreadsheet soon became apparent. It was no longer necessary to retotal all figures whenever entries were altered. The totals and analysis cross-checks were immediately recalculated and the time saved was enormous.

The ease of this success enthused me to enter formulae for a bank balance and reconciliation. Figures could be transferred automatically from the cashbook part of the spreadsheet into the bank account and balanced in a fraction of the time taken by hand. Our turnover is in excess of £100,000 per annum involving many bankings and cheques. The spreadsheet now enables us to balance all the company's books quite easily and quickly every month instead of every two or three months. And as a result, a more accurate check can be kept on expenditure and cash flow.

It became obvious that sooner or later we would need a printer. Investigation of the market revealed that many printers are sold as basic machines and in order for them to work in a business environment satisfactorily, the purchase of 'add-ons' is necessary. This appeared to be rather messy. We approached the suppliers of our electronic memory typewriter who suggested the Facit 4510 dot matrix printer which is quite widely used in business and needs no 'add-ons'. It is fitted with both centronics parallel and RS232 interface connections at the rear and also has a 2K buffer. The tractor feed attachment supplied with the machine simply clips on top of the paper feed assembly. The Sinclair lead has to have one wire in the 25 way connector altered and then the printer operates. All controls are outside and accessible to the operator. Two DIP switches are situated to the rear enabling connection to different computers quite easily. A mode switch to set 10, 12, 17 cpi, high resolution, pin or block graphics, an on/off line switch and top of form/override switch are all fitted on top. The print was crisp and clear even at 17 cpi. The speed at 10 cpi is 120 characters/second. Setting up the print drivers on Psion software was no problem apart from Easel which appears to need a special driver for all but the Epson RX80. This sturdily built little printer retails around the £400 mark.

From this point Quill could be used as well as printing hard copy from our Archive and Abacus files. Playing with the software prior to installing the printer gave the impression of a sophisticated word processing program which would be extremely useful. But in practise this did not prove to be the case. The program did some very odd things with the edit facilities and the printout was not consistent. The program also crashed with monotonous regularity, a feature we had not come across with Archive and Abacus. This version was clearly not of the same standard as the other two and could not be considered reliable enough for business purposes in its present state. Though the ideas in the program are excellent no business however small, can afford to lose documents.

Early versions crashed with monotonous regularity, a feature we had not yet met.

Easel has not yet been tried but it could be useful in monitoring various financial markets for our clients. With the exception of Quill we have had little trouble with the software packages and none at all with the hardware. Quill is back with Sinclair as we considered it unfinished. The QL and its related software have enabled us to enter the world of computerisation at little cost. It's a powerful machine and the software is not a toy; and by using it we are now more efficient. However, a computer costing £400, including software cannot be expected to do everything and hopefully software houses will produce other business programs. By purchasing our equipment through a specialist software house we have had the day to day back-up we need. After six months in operation we are still happy. Small businesses considering computers but worried about the cost should think about the QL seriously because it's not going to break the bank if it's not right, and may open up future possibilities of using computers in your business.

TERMINAL EMULATION

Moving from machine code to BCPL, Adam Denning brings us one step closer to linking QL to any computer.

In the last issue we developed a number of machine code additions to the standard BCPL library (Metacomco's). This month, by way of introduction, we deal with one further machine code routine. Our 'clock' program *(listing 1)* should be familiar to many readers. It is a simplified version of the program we ran in the December/January issue. It will run alongside our terminal emulator as a job owned by it.

As the emulator always runs in mode 0, our clock routine dispenses with any compensating adjustments to window size. Quite simply, a screen device is first opened using **UT_SCR,** and then a loop starts in which the screen's cursor is homed, the date and time are fetched, converted to strings by **CN_DATE** and **CN_DAY,** and finally displayed by **UT_MTEXT.**

So much for our clock, now we can move on to the terminal emulator proper. Written in BCPL, the program is quite lengthy as it provides an enormous number of facilities. Only part of it is reproduced in this issue, and only that part will be explained now.

Running Commentary

The first few lines – everything between '/*' and '*/' – are 'comment' lines similar to REM statements in BASIC. These contain amongst others things, a reminder of which additional modules will be required when the program is linked. Next we include the BCPL standard header *'libhdr'*, using the **GET** directive (those without floppy disks should replace *flp* with *mdv)*. Following on this, comes a declaration of all the manifest constants used by the program. A manifest constant assigns a value to a particular name. Thereafter that name may be used in place of the value throughout the program. This improves readability and acts as a useful memory aid. For example, we know that each cursor key returns a particular value but, as we don't really want to remember what these values are, we give them names here.

The next section contains the global declarations used within the program. Each identifier preceding a colon in this list is allocated a space in the global vector, BCPL's common area of store. By giving each a number it is particularly easy to reference them. These identifiers may be variable names, procedure addresses or whatever we choose. The manifest constant *ug* is declared in *libhdr,* and equates to the number of the first global which we may use in our programs. We've altered *libhdr* to incorporate our machine code routines, and part of that alteration involved changing the value of *ug*.

The real program begins, as do all BCPL programs, at the declaration of the **START** procedure. This does little aside from set the scene and call subsidiary procedures in a logical and coherent order.

Its first action is to close the default input channel, using the **ENDREAD()** procedure. As this is a CONsole device, closing it shuts down both screen and keyboard. After this a few global variables are set to their initial values and a new CONsole device opened using **FINDINPUT.** The value it returns is assigned to both *SYSIN* and *SYSOUT,* which are global variables declared in *libhdr* and used as the system input and output channels. Once opened, the channels are then **SELECT**ed, the screen cleared, colours set, author and program name printed. Thereafter, **START** is limited to purely executive tasks: the routine **setup()** is called, the serial port is opened (or not), and the clock is loaded and executed using **getclock().** Then the main procedure **do.terminal()** is called. When this returns, the opened channels are closed, the cursor disabled and the program ends.

Computer Crosstalk

When computers communicate they have to decide upon certain 'pro-

LISTING 1 CLOCK

```
STMT            SOURCE STATEMENT

 1 * Executable clock program for the QL
 2 * Especially altered for BCPL programs
 3 *
 4 * Copyright (C) 1984 Adam Denning
 5
 6          SIZE      150
 7
 8 UT_SCR   EQU       $C8
 9 SD_POS   EQU       $10
10 MT_RCLCK EQU       $13
11 CN_DATE  EQU       $EC
12 UT_MTEXT EQU       $D0
13 IO_SBYTE EQU       $5
14 CN_DAY   EQU       $EE
15
16          BRA.S     C_START
17          DC.L      0
18          DC.W      $4AFB
19          DC.W      7
20          DC.B      'B_CLOCK',0
21
22 C_START  MOVE.W    UT_SCR,A2
23          LEA.L     PBLOCK,A1
24          JSR       (A2)
25
26 RE_ENTRY MOVEQ     #SD_POS,D0
27          MOVEQ     #0,D1
28          MOVEQ     #0,D2
29          MOVEQ     #-1,D3
30          TRAP      #3
31
32          MOVE.L    A0,-(A7)
33
34          MOVEQ     #MT_RCLCK,D0
35          TRAP      #1
36          MOVE.L    (A7)+,A0
37          MOVE.L    D1,-(A7)
38
39          LEA.L     50(A4),A1
40          MOVE.W    CN_DATE,A2
41          JSR       (A2)
42
43          ADDA.L    A6,A1
44          MOVE.W    UT_MTEXT,A2
45          JSR       (A2)
46
47          MOVEQ     #IO_SBYTE,D0
48          MOVEQ     #' ',D1
49          MOVEQ     #-1,D3
50          TRAP      #3
51
52          MOVE.L    (A7)+,D1
53          LEA.L     50(A4),A1
54          MOVE.W    CN_DAY,A2
55          JSR       (A2)
56
57          ADDA.L    A6,A1
58          MOVE.W    UT_MTEXT,A2
59          JSR       (A2)
60
61          BRA.S     RE_ENTRY
62
63 PBLOCK   DC.W      $701
64          DC.W      $1207
65          DC.W      150
66          DC.W      12
67          DC.W      12
68          DC.W      0
69
70          END
```

LISTING 2 EMULATOR

```
A terminal program for the QL
By Adam Denning (C) 1984 Adam Denning


Added default setup file

Added second options menu (F4) and CTRL-S pause

This version with special job control routines
And pipes added

Adapted for floppy discs
Command escapes added
Made into 23-line screen and tidied up
Automatic floppy/microdrive select added
NEEDS: flp1_mc1_lib
       flp1_mc2_lib
       flp1_mc3_lib
       flp1_mc4_lib
       flp1_jobinfo
*/

GET "flp2_libhdr"

MANIFEST $(
     no.buffer = 0          // length of directory open buffer
     open.dir = 4           // 'open as directory' key for OPEN
     dir.header.length = 64 // length of each directory entry
     left = 192             // ASCII values of cursor keys
     right = 200
     up = 208
     down = 216
     f1 = 232               // ASCII values of function keys
     f2 = 236
     f3 = 240
     f4 = 244
     f5 = 248
     shift = 2              // effect of SHIFT on function keys
     null = 0               // ASCII control codes
     ctrlc = 3
     bell = 7
     back = 8
     forward = 9
     lf = 10
     upline = 11
     cls = 12
     cr = 13
     ctrlq = 17
     xon = ctrlq
     ctrls = 19
     xoff = ctrls
     ctrlz = 26
     esc = 27
     home = 30
$)

GLOBAL  $(
        serial:ug               // serial port stream identity
        status:ug+1             // status window stream identity
        local:ug+2              // local echo flag
        baudrate:ug+3           // baud rate
        command:ug+4            // command window stream identity
        port:ug+5               // string representing serial port
        online:ug+6             // current online status
        crlf:ug+7               // current CR/LF value
        logoff:ug+8             // character to be transmitted at log off
        jobnua:ug+9             // access rights for job control
        jobinfo:ug+10           // routine for MT_JINF call
        jinfo1:ug+11            // next job ID returned by jobinfo
        jinfo2:ug+12            // priority/status returned by jobinfo
        putpipe:ug+13           // routine to put pipe ID in job's stack
        keypipe:ug+14           // output pipe ID for keyboard input
        serpipe:ug+15           // output pipe ID for terminal input
$)
LET START() BE
  $( ENDREAD()
     jobnua := FALSE
     keypipe,serpipe := FALSE,FALSE
     SYSIN := FINDINPUT("CON_488x232a12x12_128")
     SYSOUT := SYSIN
     SELECTINPUT(SYSIN)
     SELECTOUTPUT(SYSOUT)

     SCREEN(screen.border,white,1)
     SCREEN(screen.clear)
     SCREEN(screen.size,3,1)
     SCREEN(screen.at,5,0)
     WRITES("QL Terminal Emulator")
     SCREEN(screen.size,0,0)
     SCREEN(screen.at,30,4)
     WRITES("By Adam Denning")

     setup()

     serial := FINDINPUT(port)
     IF serial < 0 THEN $(
                        WRITEF("*N*NCannot open %S",port)
                        STOP(serial)
```

tocols' so that each knows what the other means. These protocols involve such things as data parity, the Baud rate and 'handshaking' to be used. The procedure **setup**() prompts the user to enter this information and, as circumstances often lead to the same setups being used, gives the option of letting all the questions be answered automatically by a 'defaults' file.

The procedure begins with a group of local variables declared and initialised, as well as a string containing the name of the defaults file ('flp1_defaults'). As many people do not yet have disk drives on their QLs, we next call a little routine called **chk.dev**() which attempts to open (for input) the filename specified. If it fails, the 'flp' in the name is replaced by 'mdv'. Following that, the cursor is enabled and you are asked if you want to use the defaults file. If you answer with a 'y' or a 'Y', an attempt is made to open the file. If this fails then you will be prompted for the name of another file containing default setups. Pressing ENTER alone at this stage causes the **get.string** procedure to return with a name length of zero, so the file process is aborted and the keyboard is used instead. Otherwise, the named file is opened, and if this too fails, the sequence starts all over again. If a file is being used, then the local variable *usedefs* is set to TRUE and the file is selected for input, otherwise the value of *usedefs* is left at FALSE.

If no defaults file is used a series of prompts will be displayed, each requiring a one character answer. Each character is checked as it is entered. When all of the questions have been answered correctly, status and command windows are opened.

Apart from answering questions, the default files also set the values of certain global variables, such as ONLINE/OFFLINE status and the interpretation codes which will be applied to carriage returns and line feeds. The value of each global is selected using a sequence of conditional expressions, which take the form:

value: = (condition) –> true_value, false_value

If the condition evaluates to TRUE, then the expression returns the value of true_value, while if it is FALSE it is given the value of false_value. Notice that both *local* and *online* are assigned using two conditional expressions, making the code rather hard to read. What we're saying here is that if the value of *usedefs* is TRUE, then the expression will return the

value of the second conditional expression, while if *usedefs* is FALSE then the expression's value is FALSE.

Finally, any non-keyboard stream is closed and the status and command windows are set up by the routines **set.status()** and **set.comm()**. The function **capsin()** is very brief and simply returns the capitalised value of the next character read in.

Clocking In

The last section we're going to look at this month is the **getclock()** routine. This must load the clock file into the transient program area and activate it as a subsidiary job. With the help of the machine code routines we wrote in the last issue this is a very simple matter.

An attempt is made to **OPEN** the file *clock-prog*. If it fails we return straight away without an error. If it succeeds, we read the file's header into a four-word (16 byte) vector. This contains the length of the file in the first four bytes and the data space required by the job in bytes 6 to 9. As the data space value is held in a non-BCPL word aligned way, we can't read it directly. We must build up the value it represents in *datalen* using an eight-bit left shift each time to make each byte 256 times more significant than the next.

With the information at hand we call **CREATEJOB**, which returns with the BCPL address to load the job as its result, and the QDOS job ID of the new job in *RESULT2*, which is a global variable declared in *libhdr*. If the address returned is zero, the call failed, so action is taken. If it succeeded, we load the file into the space created using **READFILE** and then we activate the job with a priority of 1 and a timeout of zero (equivalent to EXEC). In either instance, we then close the file and return.

Next month we will examine a further group of routines, by which time you should be able to run a basic version of the terminal.

```
    getclock()
    do.terminal()
    CLOSE(serial)
    CLOSE(status)
    SCREEN(screen.nocursor)
    ENDREAD() ; ENDWRITE()
AND setup() BE
  $( LET option,usedefs = ?,FALSE
    LET defile = "flp1_defaults"
    port := "sernphz"
    chk.dev(defile)
    SCREEN(screen.cursor)
    WRITES("*N*NDo you want to use previously set defaults? ")
    IF capsin() = 'Y' THEN $( LET tempin = FINDINPUT(defile)
TEST tempin < 0 THEN
    $( LET filename = VEC 10
       WRITES("*NCannot find file - ")
       get.string(filename,42,"Type in the filename")
       TEST filename%0 = 0 THEN $( WRITES("*NUsing keyboard*N")
                                   tempin := 0
                                $)
         OR $( tempin := FINDINPUT(filename)
               TEST tempin > 0 THEN $( SELECTINPUT(tempin)
                                       WRITEF("*NUsing %S*N",filename)
                                       usedefs := TRUE
                                    $)
                 OR WRITEQDOSERR(tempin,SYSOUT)
            $)
    $) REPEATUNTIL tempin >= 0
  OR $( SELECTINPUT(tempin)
        WRITES("*NUsing defaults*N")
        usedefs := TRUE
  $)
  WRITES("*N*NSelect:*N-------*N")
  WRITES("Serial port: (1 or 2) ")
  SCREEN(screen.cursor)
  option := capsin() REPEATUNTIL option = '1' | option = '2'
  WRCH(option)
  port%4 := option
  port%0 := 4

  WRITES("*NParity: (Mark Space Odd Even None) ")
  option := capsin() REPEATUNTIL option = 'M' | option = 'S' |
                                 option = 'O' | option = 'E' |
                                 option = 'N'
  UNLESS option = 'N' DO port%0,port%5 := 5,option
  WRCH(option)

  WRITES("*NRTS/CTS Handshaking: (Yes No) ")
  option := capsin() REPEATUNTIL option = 'Y' | option = 'N'
  WRCH(option)
  option := option = 'Y' -> 'H','I'
  port%0 := port%0 + 1
  port%(port%0) := option
  WRITES("*NData protocol: (Raw  Z:(CTRL-Z = EOF)  C: (LF < > CR)) ")
  option := capsin() REPEATUNTIL option = 'R' | option = 'Z' |
                                 option = 'C'
  port%0 := port%0 + 1
  port%(port%0) := option
  WRCH(option)

  WRITES("*NBaud rate: (1=75 2=300 3=600 4=1200 5=2400 6=4800 7=9600) ")
  option := RDCH() REPEATUNTIL '1' <= option <= '7'
  TEST option = '1' THEN baudrate := 75
    OR baudrate := 75 << (option - '0')
  BAUD(baudrate)

  status := FINDOUTPUT("SCR_338x12a162x0")
  command := FINDINPUT("CON_488x12a12x244_50")

  local := usedefs -> RDCH() = '0' -> FALSE,TRUE,FALSE
  online := usedefs -> RDCH() = '0' -> FALSE,TRUE,FALSE
  crlf := usedefs -> RDCH() - '0',0
  logoff := usedefs -> RDCH(),ctrlz

  IF usedefs THEN $( ENDREAD()
                     SELECTINPUT(SYSIN)
                  $)
  set.status()
  set.comm()
$)

AND capsin() = CAPITALCH(RDCH())

AND chk.dev(filename) BE
  $( LET stream = ?
    stream := OPEN(filename,1,no.buffer)
    TEST stream > 0 THEN CLOSE(stream)
      OR filename%1,filename%2,filename%3 := 'm','d','v'
  $)

AND getclock() BE
  $( LET stream,datalen,addr = ?,0,?
    LET header = VEC 3
    LET cfile = "flp1_clock_prog"
    chk.dev(cfile)
    stream := OPEN(cfile,1,0)
    IF stream < 0 THEN RETURN
    READFILEHEADER(stream,header,16)
    FOR i = 6 TO 9 DO datalen := (datalen << 8) + header%i
    addr := CREATEJOB(header!0,datalen)
    UNLESS addr = 0 DO
      $( READFILE(stream,addr,header!0)
        ACTIVATE(RESULT2,1,0)
      $)
    CLOSE(stream)
  $)
```

DIGITAL PRECISION

DEDICATED TO TOP QUALITY QL SOFTWARE

QL SPRITE GENERATOR
THE COMPLETE SPRITE KIT

THE ULTIMATE QL GAMES DESIGNER

NEW

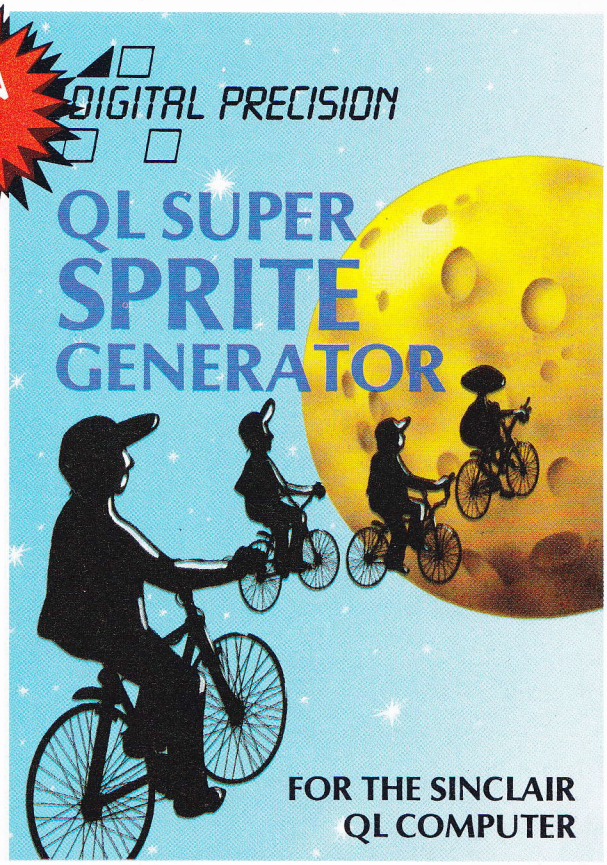
Includes:
- ★ SUPERSPEED SPRITE COMPILER
- ★ SPRITE MANAGEMENT PROCEDURES
- ★ ADVANCED SPRITE CREATION EDITOR
- ★ LIBRARY OF SUPER BASIC EXTENSIONS
- ★ DEMONSTRATION SPRITES AND MUCH, MUCH MORE

HERE IT IS . . . THE PACKAGE YOU HAVE BEEN WAITING FOR! THIS COMPREHENSIVE SYSTEM WILL ALLOW YOU TO CREATE ALL THE SPECIAL EFFECTS OF ARCADE ADVENTURE AND SIMULATION PROGRAMS. REQUIRES ABSOLUTELY NO 68000 MACHINE CODE KNOWLEDGE — THE COMPILER DOES IT ALL FOR YOU, AUTOMATICALLY!

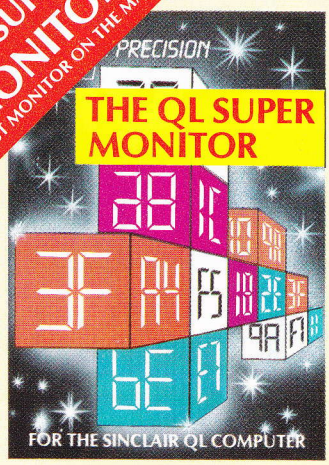
Features include:
★ Easy drawing/amending of sprites ★ No limit to the number of sprites ★ Up to 16 different sprites on screen simultaneously ★ Each sprite can have up to 16 different "shapes" to simulate natural motion ★ Up to 256 different planes to control depth of movement ★ Automatic real-time collision detection ★ Sprite reversal ★ Ultra high speed action — at top speed motion is faster than the eye can see! ★ Variable speed ★ Easy control from within your program, using SUPERBASIC keywords ★ Library of pre programmed sprites ★ Easy to use ★ Full instructions included.

CAN BE USED IN YOUR OWN PROGRAMS, WITHOUT ROYALTY RESTRICTIONS — PRODUCES SELF-CONTAINED 68000 MACHINE CODE!

£19.95 COMPLETE WITH EXTENSIVE MANUAL



THE QL SUPER MONITOR
THE BEST MONITOR ON THE MARKET



NEW

A TOP FLIGHT 68000 MACHINE CODE MONITOR, TO GET THE BEST OUT OF YOUR QL!

This machine code program will allow you to test, debug, check and correct any machine code program — and to understand the operation of anyone else's! This state-of-the-art product is written entirely in machine code and, occupying very little space, can reside in the QL at the same time as your other programs (BASIC machine code).

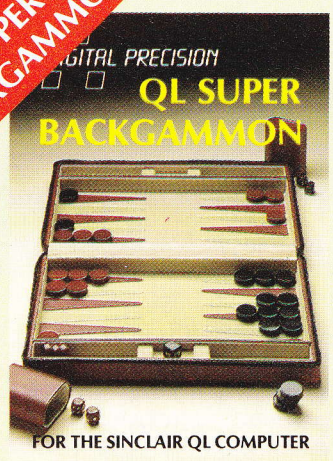
Features include:
Edit/Jump/Trace/Set/ Catalogue/Dump/Load/Save/ Display registers/Set registers/Disassemble/Hex arithmetic etc.

RELEASE 2.0

FREE! A Full 68000 Disassembler Program is supplied free to purchasers of the QL Super Monitor.

£18.95 COMPLETE WITH MANUAL

QL SUPER BACKGAMMON



NEW

An excellent program for beginners and experts at the game. Variable playing strength, full on-screen instructions, obeys all the rules of backgammon. Randomising features ensure a different game each time — trillions of possibilities! Fast response speed — plays an aggresive, challenging and entertaining game. Dual clocks; 3 Playing Modes; Auto-play (Computer vs Computer); Normal (Human vs Computer; Supervisor (Human vs Human). Rules of Backgammon both in booklet and on-screen, in Quill document. 4 playing levels. Computerised dice thrower coupled with override option to allow you to throw real dice. Very easy to operate.

£15.95 COMPLETE WITH RULES OF BACKGAMMON

INTRODUCTORY OFFER!

★ **£1 OFF IF YOU BUY 2 PROGRAMS**

★ **£3 OFF IF YOU BUY 3 PROGRAMS**

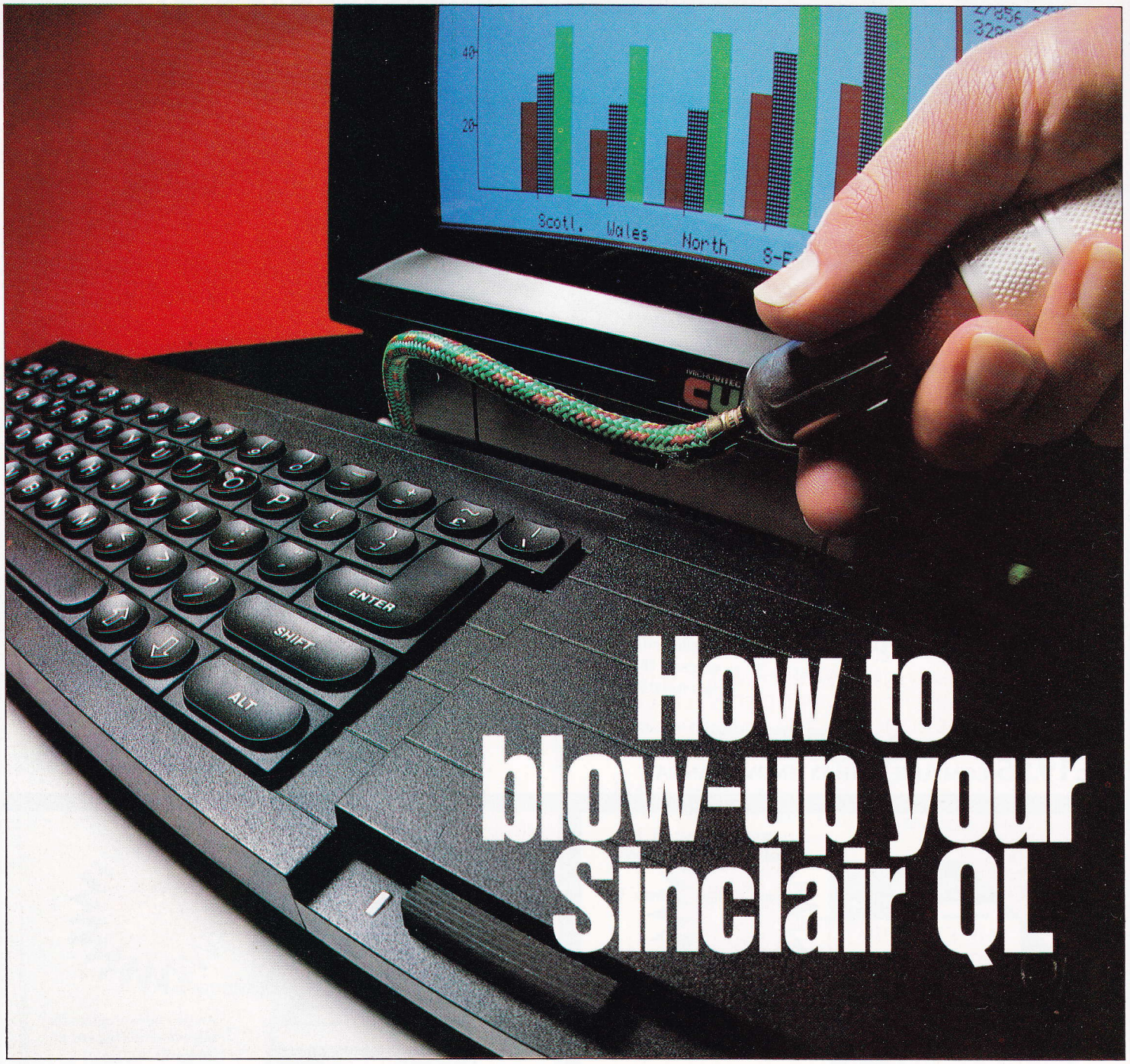
MAIL ORDER: Just send cheque/postal order/cash to:
DIGITAL PRECISION, 91 Manor Road, London E17 5RY

Delivery will be by return of post. For orders from Europe add £1 per program. For airmail orders from all other countries add £1.50 per program. Within the U.K., our deliveries are post free.

DIGITAL PRECISION

IS DEDICATED TO TOP QUALITY SOFTWARE FOR THE QL.

WE WILL PAY TOP ROYALTIES FOR GOOD ORIGINAL QL SOFTWARE — CONTACT US NOW!



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/4/85

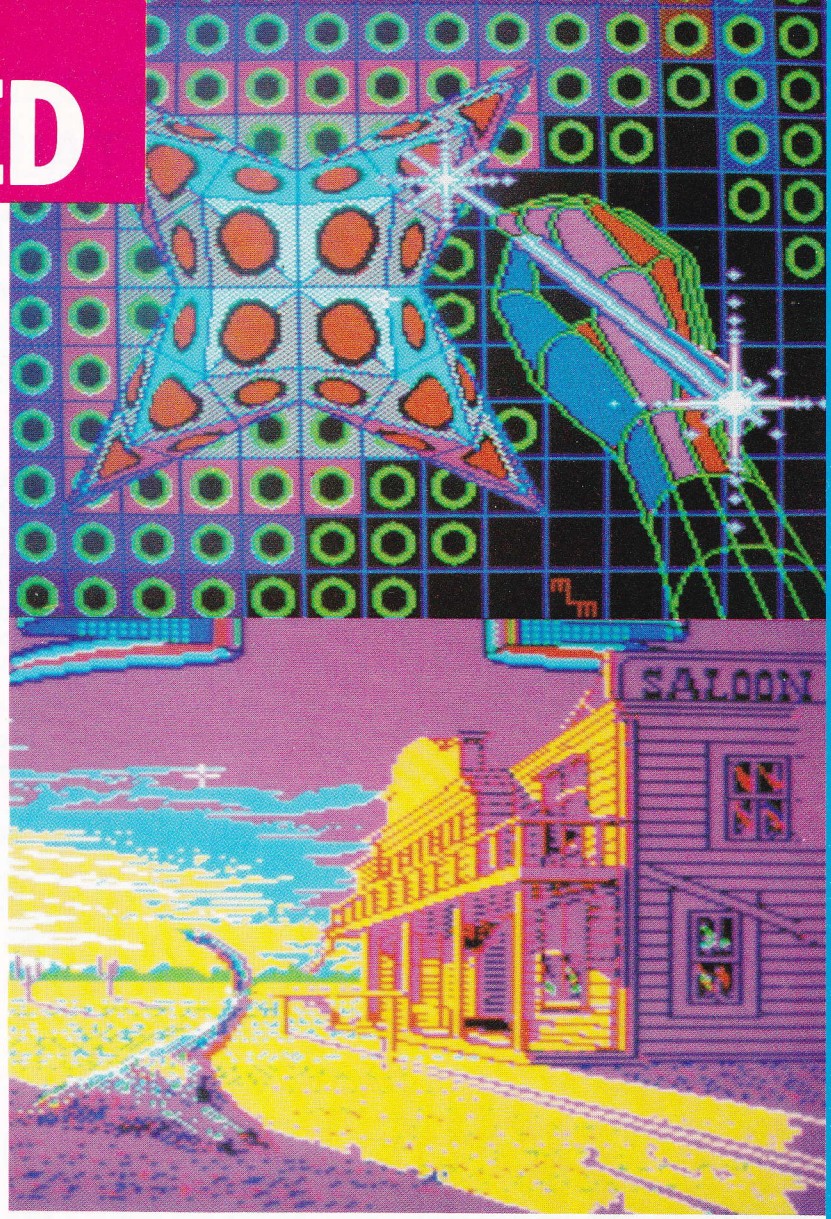
MICRO APL

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).

TALENTED GRAPHICS

Ralph Warren reviews the first in a series of sophisticated graphics packages and assesses the state of the art.



Talent's Mike Masters applies paint to pixel. The result of many hours work is spectacular.

This new graphics package, called GRAPHIQL, consists of two microdrive cartridges and accompanying manual. One cartridge contains some 80K of machine code which constitutes the main program plus a utility for dumping screens to printers. The other houses a set of demonstration screens.

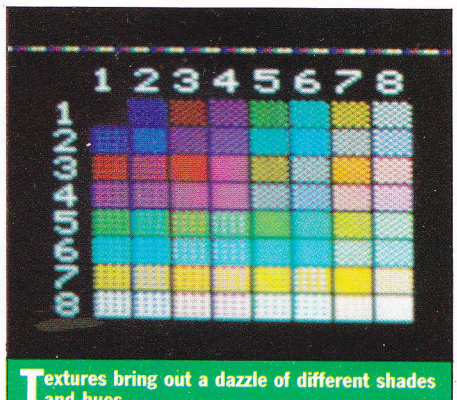
GRAPHIQL permits pictures to be drawn on a screen measuring 256 × 256 pixels. A flashing cursor is used to indicate your position on screen. Initially, this appears as a set of cross wires, however its shape and colour may be altered. The cursor is moved about the screen using either the cursor keys or a joystick. The movement is slow although 'wraparound' at the screen's edges permits movement from one side to the other without having to trek back across the centre.

Each pixel on screen may be individually lit in any one of the QL's eight basic colours. Colour is selected by pressing a single numeric key corresponding to the appropriate colour code (0-7).

Colour may be applied in one of two ways. The new colour simply replacing the old or, blending with one already there to produce a different colour (ie, cyan on magenta produces yellow). Furthermore pixels may be set to flash and, more interestingly, selection of airbrush mode makes it possible to paint narrow swathes of splattered dots across the screen.

Talent have included a facility to magnify (16 times) the area of the screen upon which one is working. One of the highlights, this permits you to touch up displays with pin-point accuracy.

On the design side, Talent have built in a variety of routines for drawing lines, rectangles, circles and ellipses, though not arcs. These are called using a set of simple and easily remembered two character commands. In addition a more sophisticated set of routines permit blocks to be copied, moved across the screen with their contents rotated through 180 degrees, or reflected in the vertical or horizontal. The problem is that only a single block may be defined at any one time and its size may not exceed 4000 pixels which works out at little over 6% of the screen.



Textures bring out a dazzle of different shades and hues.

The true power of the package comes from a facility to define up to 26 'textures' which may then be reproduced on the main screen at any location, in any combination of colours. A 'texture' may be anything from a simple background pattern to an intricate shape or figure like a bird or a tree. It is designed on a separate screen or 'doodle' pad that occupies the bottom half of the screen. Whilst texture manipulation takes some getting used to, especially the colour aspects, it is possible to achieve effects which otherwise would take many hours of free hand sketching.

To make the package as user-friendly as possible a crib line may be switched on which displays status information relating to colour selection, cursor position and current operation. Similar to the Psion packages, there is a help command provided that will page explanations of the various commands on screen. This is stored in memory and is instantly accessible.

Sketches and textures may be saved to cartridge and files stored in a standard form so that pictures may be incorporated into other programs. Files vary in size from 12K for a texture file to 33K for a screen save.

Overall, GRAPHIQL is aimed at the artist and recreational user as opposed to the draughtsman or engineer. Using the QL's low-resolution screen, the emphasis is on colour and composition rather than pin-point accuracy. However, with features such as magnification and texture definition it represents a considerable advance upon electronic sketchpads currently available. Not quite Computer Aided Design but definitely a step in the right direction!

Price: £34.95
Supplier: (Mail order only) Talent Computer Systems, Curran Building, 101 St James Road, Glasgow G4 0N8. 041-552 2128.

NEW — FROM TALENT!

GRAPHIQL

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

Features include:

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

Text can be included in pictures. The characters can be single or double height with flash and underline.
GRAPHIQL pictures can be put into BASIC or assembler programs with the sample routines provided.
Available in March by mail order direct from:

£34.95
+ 50p postage & packing

TALENT
COMPUTER SYSTEMS
Curran Building,
101 St. James Road, Glasgow G4 0NS
Tel: 041-552 2128 (24-hour Credit Card Hot-Line)
SOFTWARE FROM SCOTLAND

CARE ELECTRONICS

UNIT 14, PEERGLOW INDUSTRIAL ESTATE, OLD'S APPROACH TOLPITS LANE, WATFORD, HERTS
☎ Tel: 0923 777155 ☎

PRINTER INTERFACE
QL SERIAL TO PARALLEL CONVERTER

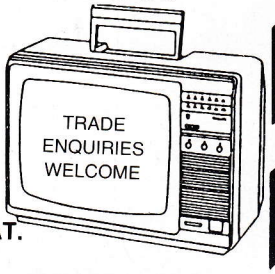
Switchable baud rate. This is necessary because the baud rate on the two serial ports cannot be set independently, and unless you can set the baud rate on serial to parallel converters your printer will be inoperative while using modems and other communications devices. Self-contained, colour matches the QL, full one year guarantee, comes complete with 1.5 metres of cable

★ BAUD RATE SWITCHABLE
★ CRYSTAL CONTROLLED
★ QDOS and SUPERBASIC compatible
★ Drives any CENTRONICS printer
★ Price £49.91 (d) inc. VAT

PRINTER SELECTOR
Capable of taking one QL and up to six printers
Price £65.09 (d) inc. VAT

Attention all software houses, Eprom cartridges for the QL, ROM expansion slot now available, an ideal way of packaging your utility program fast loading/no error and saves expense of microdrive cartridge.
Please consult factory for details.

DON'T BUY A COLOUR MONITOR FOR YOUR QL
Have your Pye/Philips or Ferguson TX TV converted to a TV/Monitor. In house conversion.
£63.25 (a) inc. VAT.
Conversion kit £51.75 (d) inc. VAT



★ Resolution better than 585 × 450 pixels
★ Image clarity comparable to leading monitors
★ Includes RGB lead for connecting with QL
★ Conversions carried out at our workshops within 2/3 days
★ Please telephone for other makes

Philips 14" TV/Monitor at £253 (a) inc. VAT.
Specially converted to run for the QL.

How to order by post, enclose cheque/P.O. made payable to Care Electronics or use Access. 7 days for delivery.
Please add carriage (a) = £8.00
(d) = £2.00

Open
9am-5pm Mon-Fri
9am-4pm Sat

COMPUTAMAGIC!

Win a new Q-Disc disk system from Computamate worth over £500 (complete with dual drives), in this easy-to-enter competition.

One of the first disk drive systems on the market, the Q-Disc consists of a small PCB with on-board ROM and disk driver conforming to the Sinclair-defined standard (40/80 tracks, 9 sectors per track and 512 bytes per sector).

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.

The interface provides an arsenal of additional commands governing random access, file maintenance, job control and numeric conversion. Many of the extra commands work with microdrives as well as disks. Categories include Multitasking, Directory Enquiry, File Maintenance, File Enquiry, Random Access, Numeric Conversion and Development Tools.

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	30 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.

Versions 2.00 of the packages will run unamended on Q-Disc. These versions, 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.

TO ENTER: All that's required is a little verbal dexterity. Simply construct the longest word you can using the letters from the following phrase:

"THE Q-DISC INTERFACE IS COMPUTAMAGIC!"

RULES: Each letter can be used as many times as it appears in the phrase, but no more than that. Proper names (names of towns etc), foreign words, abbreviations and hyphenated words are not allowed.

Words excluding the above found in the OED (complete version) will first be judged on the number of letters and then (in the event of a tie) on the number of different letters they contain.

In the absence of an outright winner, the prize will be awarded to the first entry 'out of the hat'.

All entries must be written on a postcard only and sent to:

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

CLOSING DATE: Entries must be received on or before **20th April, 1985.**

JUDGING: The winner will be chosen by the Editor, whose decision is final, based on the conditions above. No correspondence about the results will be entered into, though the winner's name will be published in a future edition of *QL User*.

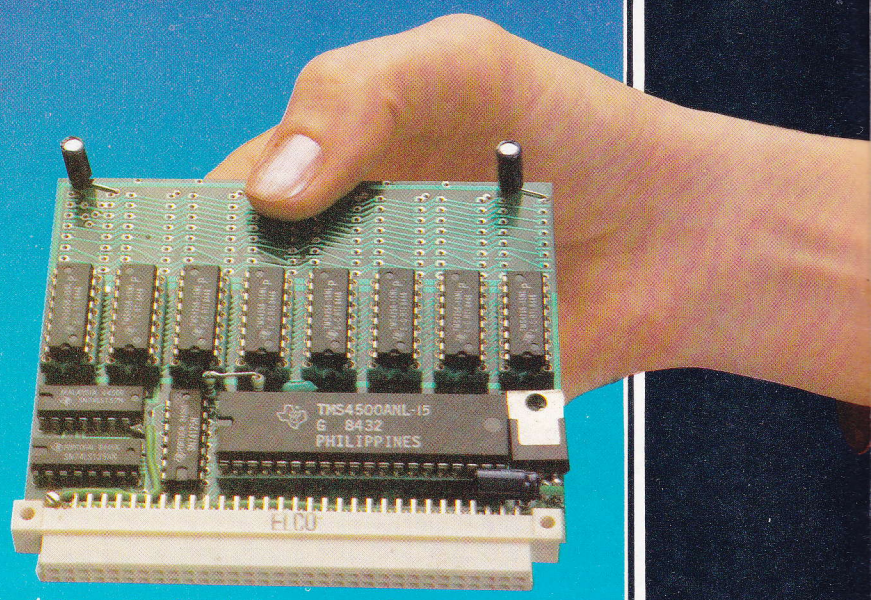
No employee of EMAP or associate companies or anyone connected with Computamate Data Products may enter this competition.

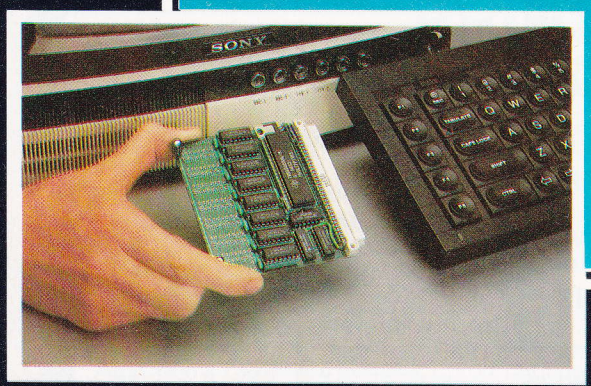
More Memory to your QL

- **¼ MEGABYTE MORE.** This new memory Expansion Board from Simplex Data Ltd. gives your existing QL a further 256k worth of extra memory.
- **EVEN MORE EXPANSION.** In addition to the ever popular ¼ Megabyte board, you now have the unique opportunity to purchase a ½ Megabyte Memory Expansion board–512k! Even more memory for your QL
- **DESIGNED FOR THE SINCLAIR QL.** Most users of the QL will realise the limitations of a 128k system. Simplex Data have not only expanded the memory potential, but have given you, the user, a wider field of operation.
- **EASY TO USE.** Simply remove the port cover located at the end of the QL keyboard and slot in your NEW Simplex Memory Expansion board–It's as simple as that!
- **COMPACT SIZE.** There's nothing like it on the market. Measuring approximately 97mm x 77mm, this little board is a major advance in Micro technology.
- **HOW IS IT DONE?** Utilising the latest 256k x 1 bit Dynamic RAM'S.

Only £198.00*

Inc. VAT, postage and packing. 1 year warranty included. Please allow 28 days for delivery.





*This special offer price is guaranteed for a limited period only and ends the last day of April.

Exclusively available from

SDL Simplex Data Limited

432 Greenford Road, Greenford, Middlesex UB6 8SG. Tel: 01-575 7531

To: SIMPLEX DATA LIMITED, FREE POST, 432 GREENFORD RD, GREENFORD, MIDDLESEX UB6 8SG.

Please send me: _____ (Qty) Memory Expansion Boards at £198.00 (inc. VAT, Post and Packing).

I enclose Cheque/Postal Order for £ _____

Name: _____

Address: _____

_____

Telephone No.: _____ QL/4/85

Payment by ACCESS or BARCLAYCARD phone: 01-575 7531

ACCESS VISA

QL NEWS

News from the world of Sinclair QL computing.

The communications explosion takes shape!

Communications are now the most exciting, essential part of any computer.

In the past six months alone, over 150,000 modems have been sold in the UK.

Now, the QL's own communications explosion is taking shape . . . and it has the potential to make more of communications than any other micro!

Read on and discover exciting new ways to use your QL . . . with the QL modem . . . telephony unit . . . and powerful interface options.

Nº2

DAVID KARLIN

Why Q COM is everything you could wish for in communications.

The QL is now communicating – via Q COM! This exciting three-part peripheral presents QL users with a multitude of ways to exploit the world of communications.

Once connected to the QL, Q COM allows you to access the considerable number of phone-in databases, such as Prestel and QNet.

Q COM enables you to communicate with other computer users. Its facilities include electronic mail, data transfer from Microdrives and bulletin boards.

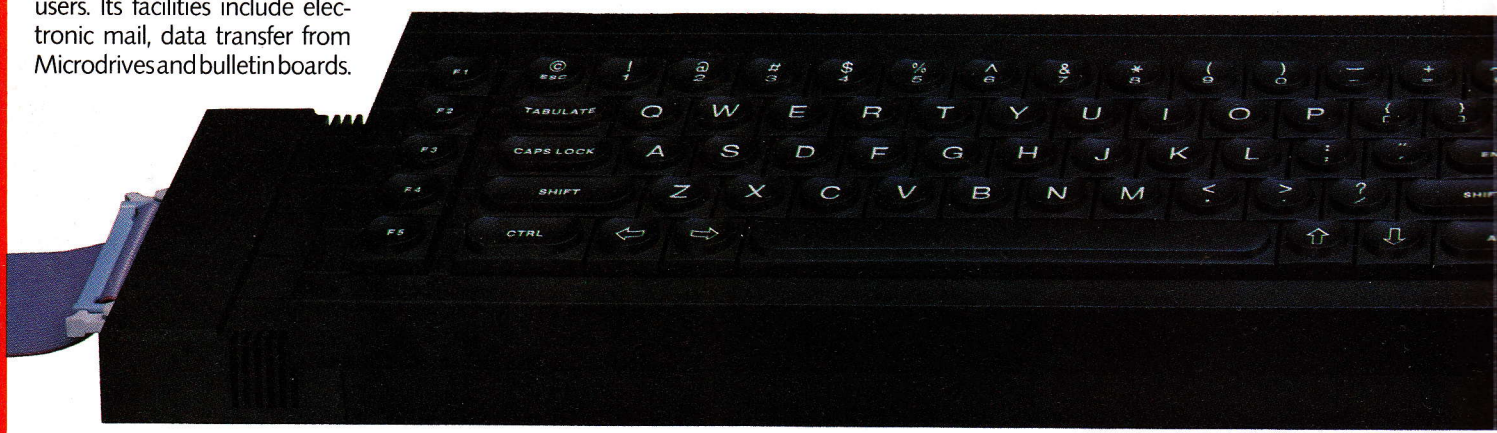
Through it you can link your QL to larger minicomputers. Q COM has full capability in this area, and allows the QL to talk to powerful mainframes.

Q COM's automatic dialling and call acceptance facilities, together with the storage of messages from other modems, will revolutionise the way you use your telephone.

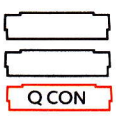
The next few pages of QL News tell you much more.

It's enough for me to say here that with the QL and Q COM, you'll be exploring new openings in communications for some time to come!

David Karlin,
Chief Design Engineer.



The Q COM package
Three special parts to stack!



QL communications interface

This multi-speed interface contains the sophisticated software used to set up QL communications – and to control the Q CALL and Q MOD units.

Q CON also comes complete with Microdrive-based software. This enables the QL to link to larger computers using VT100 and viewdata protocols.

The software will also run *any* standard modem – connected via Q CON's built-in RS-232-C port.

Most importantly, Q CON allows the QL to transmit and receive at rates switchable from 75 to 9600 baud (encompassing the widely-used 75/1200 Prestel rates, and 1200/1200 half duplex rates for user-to-user exchange).

Q CON is specially styled to suit your QL – with similar fluting and ribs – and forms the base module of a vertical-stacking system.

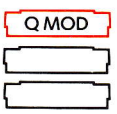
It's supplied with full instructions, software on Microdrive cartridge, and connecting leads.

QL auto dial/answer unit

Q CALL gives every QL user something out of the ordinary.

It's a module which links directly to your telephone, and allows auto-dialling at the push of a single key. In the same way, it will permit incoming calls to be accepted automatically . . . and even trigger pre-programmed activity from the QL!

Q CALL is the central unit of the package. It plugs directly into Q CON – so there are no connecting cables to worry about.



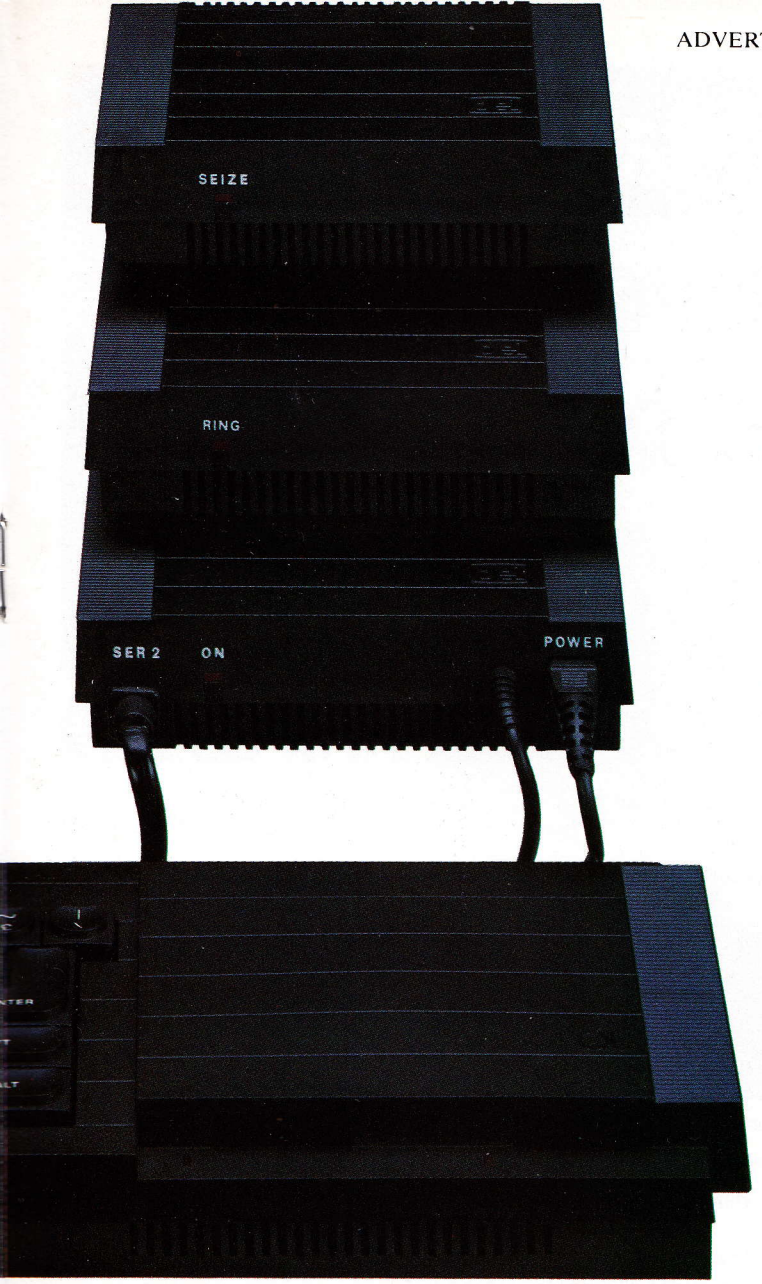
QL modem

Q MOD has all the powerful facilities expected of a modem, in a neat and simple unit.

It uses either V23 75/1200 or 1200/1200 baud rates, for Prestel, Micronet 800 and all the viewdata services described alongside.

It also incorporates a telephone extension socket for manual dialling.

Q MOD is the top unit of



Q COM, and comes with a 9' built-in telephone cable.

All three units are available from Sinclair on (0276) 685311 and from selected Sinclair stockists.

The QL hooked on voice and data

The QL can now act as your personal address book and telephone operator!

Q COM allows you to store hundreds of personal or business numbers.

You can store lengthy passwords and account numbers – and recall them – at the touch of a single key.

And any information that's sent to you from other modem-owners can be gathered and stored on Microdrive cartridge, or incorporated into your QL Quill documents!

Exploring the world of QNet, Prestel, Micronet and more!

Thousands of QL users already enjoy the excitement of linking to a nationwide mainframe.

Q COM turns your QL into an intelligent terminal, allowing you to access many thousands of pages of information, software and communications facilities.

The services brought to you through Prestel can include Micronet 800, Viewfax 258 and QNet, the new QL database.

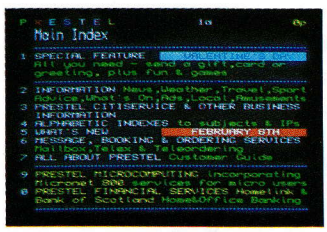
Membership of QNet will bring you free software, QL news and features, and all the wide-ranging services of viewdata!

If armchair shopping is more your style, that's easy too. It's often possible to place a direct order using your QL! For dedicated QL owners, there's a daily selection of software reviews, chart toppers... and all the facts and figures you need to make buying peripherals simple.

With Q COM you can also 'download' software from the system directly into your QL and either use it immediately, or store it on Microdrive cartridge.

In fact the only problem you'll face with a viewdata service is finding enough time to explore its many features!

You can find out how to join QNet by phoning 01-278 3143.



News... information... banking services and QNet. And only a fraction of the QL's new viewdata capability.

QL meets the mainframes!

The Q CON unit of Q COM turns your QL into a VT100 terminal, providing instant access to in-house computing services, both mainframe and mini.

Whether you are using your QL at home or at work, Q COM gives you access to electronic bulletin boards which provide help and advice 24 hours a day. You can leave messages or notices for friends or business contacts and even hold live discussions with them.

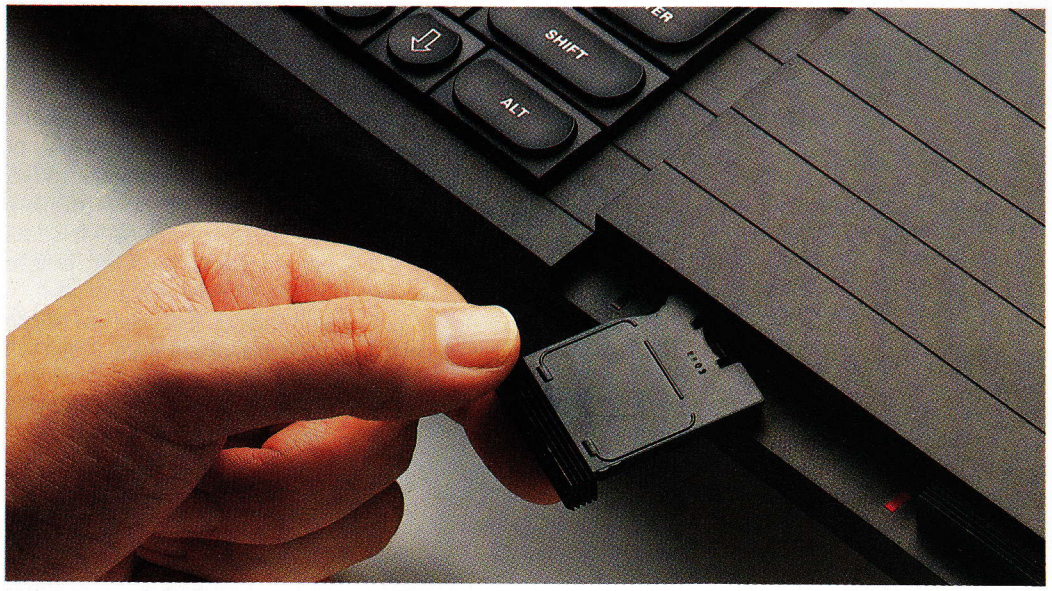
Additional benefits for the QL business user include easy access of in-house company software, and the interrogation of other data bases around the country.

There's also the opportunity of linking to British Telecom Gold – the widely-publicised and popular messaging service.

sinclair

QL Hardware

Microdrive cartridge price cut to only £1.99!



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

On February 1, the cost of Microdrive cartridges came down from £4.95 to £1.99 each.

Microdrive cartridges are the QL's own unique storage medium. Each stores up to 100K of information (that's 40 pages of A4 text), on a cartridge no bigger than a book of matches!

Over 500,000 cartridges are now being used throughout Britain.

You can store up to 50 different data files per cartridge, identified by titles of your own choice.

And QL Microdrives themselves are standard equipment on the new ICL One Per Desk micro, and British Telecom's new Merlin Tonto.

IEEE-488... the instrument connection

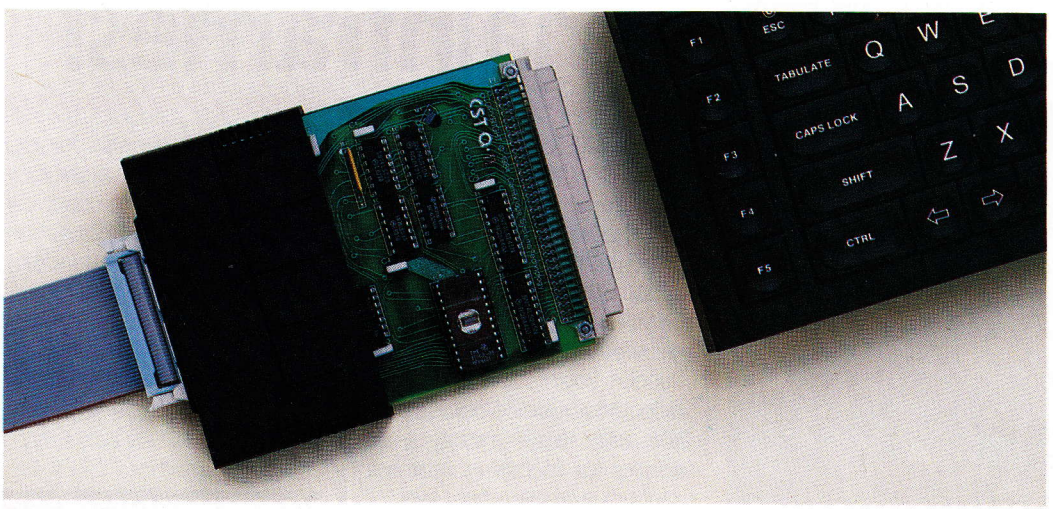
IEEE-488 is the interface standard set by the Institute of Electronic and Electrical Engineers for instrumentation control.

IEEE-488 – or General Purpose Instrumentation Bus – is a parallel interface specifically designed for high speed data transfer between a number of different types of device.

It is commonly used for controlling instrumentation via a computer, allowing the creation of laboratory data acquisition systems, industrial control schemes, etc.

The QL now has a fully-fledged IEEE-488 interface from CST. It plugs neatly into the QL's RAM expansion port, and can control up to 16 instruments simultaneously.

It's available from CST on (0223) 323302.



An IEEE-488 interface slips discreetly into place.

New inter 3½" or 5¼"

With new Q-Disk, you can transform the QL into a powerful small business system – comprising QL, monitor, disk interface, twin disk drives and printer.

Q-Disk upgrades the QL to disk storage. Fitting easily into the QL's left hand RAM expansion port, without the need for a special expansion box, it contains a Western Digital disk controller chip. Software is held in an on-board EPROM (so little of the QL's RAM is used).

Plug in Q-Disk, and the QL accepts one or two disk drives, sized 3 in, 3½ in, 5¼ in, either 40 or 80 track, single or double-sided. Even when two drives are used, they can be different types!

Q-Disk offers up to 1.6 Mbytes of quick, reliable storage with a compatible disk drive.

It's made by Computamate, who also offer a full range of

QL to link students

Strathclyde University, in Glasgow, plans to have a campus network of 7,000 QLs linked to a central VAX minicomputer.

That's one QL for every student . . . a major investment project in a university which is now a leading centre for artificial intelligence work.

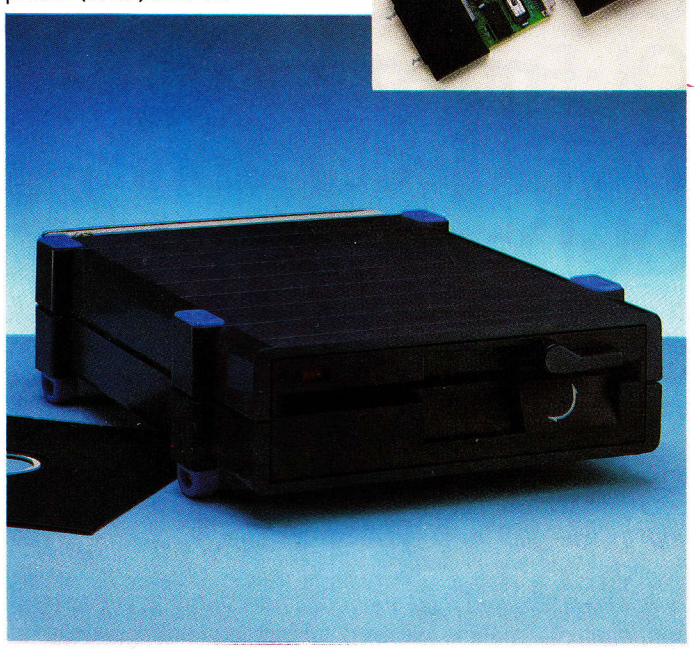
Sinclair is giving support worth £250,000 to the project. And it's likely that QL users everywhere will benefit – the students plan to develop AI programs to run on the QL!

The QL has impressed Prof. James Alty of the University's Computer Science Department, who says *'only the QL could offer the computing power, range of applications, and above all the portability, at a realistic price.'*

face to connect 3½" disk drives

complimentary QL disk drives.

To contact Computamate, phone (0782) 811711.



Single disk unit fitted with 5¼ inch drives and *(inset)* the Q-Disk controller.

The QL's high-tech spec

Dimensions
138 x 46 x 472mm
(5⅜" x 1¾" x 18¾")

Weight
1388 gms (3.055 lbs)

RAM
Massive 128K standard RAM, externally expandable to 640K. Extra RAM is available in 64K, 128K, 256K and 512K units, from third-party suppliers.

ROM
48K, containing Sinclair SuperBASIC and the Sinclair Qdos operating system.

CPU
Motorola 68008 (running at 7.5 MHz) for all principal functions. (Architecturally, the 68008 is a 32-bit processor with an eight-bit data bus. One megabyte of non-segmented address space is available.)

In addition, an Intel 8049 controls the keyboard, generates the sound, and acts as an RS-232-C receiver.

Operating system
Qdos (developed by Sinclair Research) is a single-user multitasking, time-sliced system using Sinclair SuperBASIC as a command language with display handling for multiple screen windows; and device-independent input-output.

Language
Sinclair SuperBASIC, with the advantages of procedure structuring; extendability (including syntax); interpretation speed independent of program size; clean machine code interface; operating system facilities accessible from SuperBASIC; equal capability for strings and arrays; and full error-handling facilities.

Microdrives
The QL incorporates twin QL Microdrives, each with a minimum 100K capacity, 3.5 seconds average access time. Typical loading rate of machine code programs is 2-3K per second.

Video
High resolution graphics capability with colour or monochrome monitor (or TV) in two modes – 512 x 256 pixels (four colours available) and 256 x 256 pixels (eight colours available). Normal character display format of up to 85 x 25 with choice of character sets available (TV format of up to 40 to 60 columns depending on the software).

Keyboard
Full-size, 65-key QWERTY keyboard featuring a space bar, left- and right-hand shift keys, five function keys and four cursor control keys. The keyboard can be angled by means of detachable feet.

Expansion
Excluding RGB monitor, power socket and TV port, eight peripheral/expansion ports are provided – one internal expansion, one Microdrive expansion, one ROM cartridge, two serial and two control channels, and the local area network.

Serial
Two standard RS-232-C communications interfaces for printers, modems, etc. Transmission at rates from 75- 19200 baud or full duplex transmit/receive at seven rates up to 9600 baud.

LAN
For up to 64 QL computers. Data transmission over the net can be achieved at 100K baud.

Power supply
9V DC at 1.8A, 15.6V AC at 0.2A.

Joysticks
Provision for one or two devices for games or cursor control.

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

Price
£399 including VAT, QL programs, full A4 manual, power supply, 4 blank cartridges and free Helpline service.

sinclair

QL Software

Updated versions of Psion software now available!

QL Abacus, Archive, Easel and Quill are the four Psion programs supplied with every QL. They're now converted to 100% machine code, and as a result they load from Microdrive cartridge much faster.

The overlays present in Version One software have been removed, resulting in noticably quicker on-screen performance.

With the compactness of machine code, there's a big saving in QL memory too – all four programs now cope with larger, more professional applications!

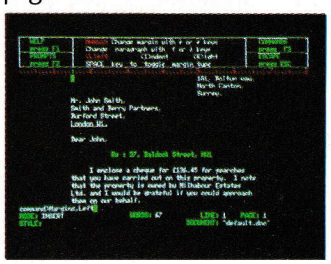
Version Two software is now supplied with every new QL. Existing QLUB members – see back page.

QL·Quill

QL Quill makes it easy to type in, correct and store your letters, memos and reports.

No training is needed – a beginner can be using QL Quill for word-processing within minutes!

QL Quill has the facilities of professional word processing packages: including word wrap, search and replace, justification, page headers and footers.



QL·Easel

QL Easel allows you to create graphs, bar charts and pie charts – at the touch of a key.

The program handles anything from lines and shaded curves to overlapping or stacked bars.

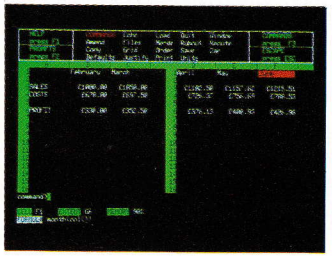
QL Easel designs and scales automatically or under your control. Text can be added and altered as simply as data.

QL·Abacus

QL Abacus is a powerful, yet easy-to-use spreadsheet.

The program allows you to manipulate the contents of whole rows and columns by the names you assign them. There's no need to depend on confusing letters and numbers.

QL Abacus also incorporates a range of functions which let you carry out rapid 'what if' analyses on your data.

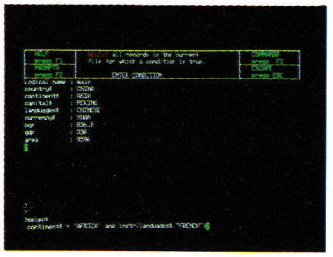


QL·Archive

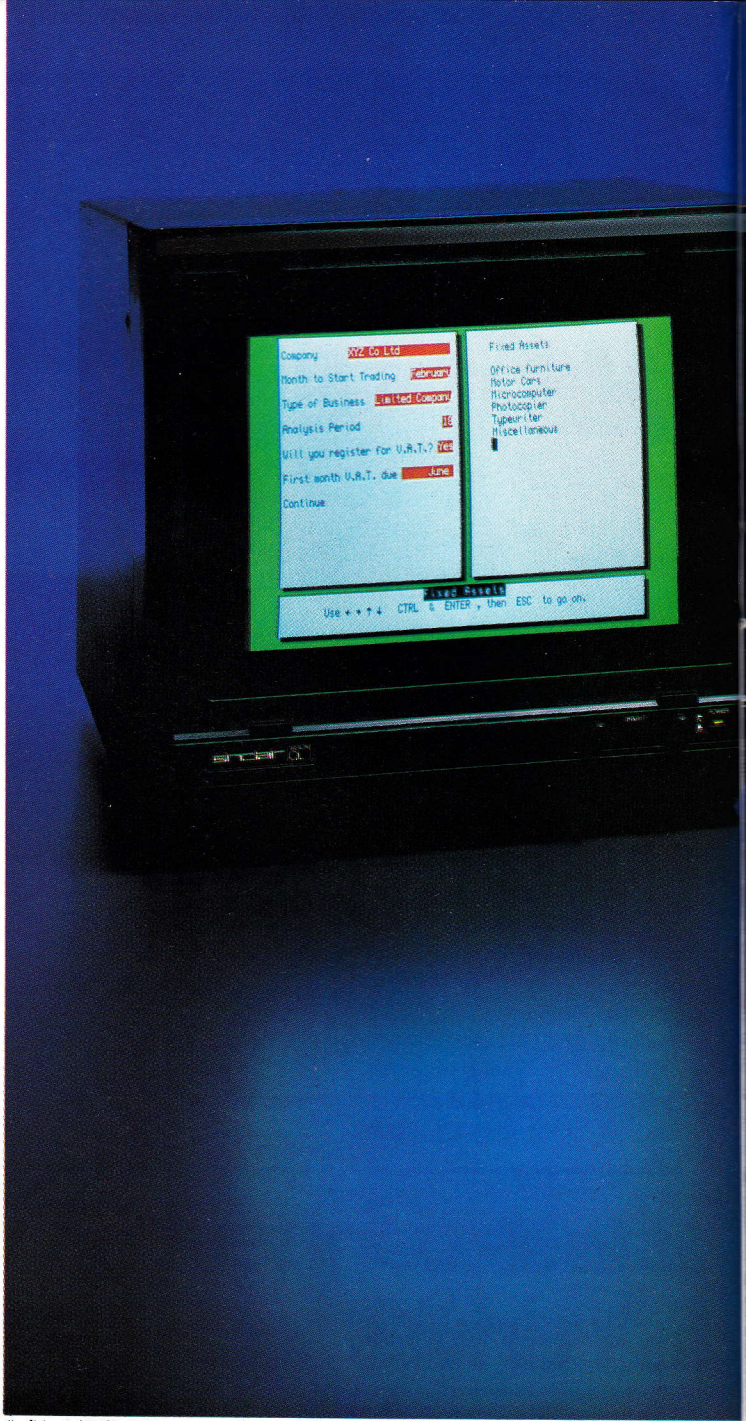
QL Archive is a sophisticated, powerful database program.

It includes a screen editor which allows you to design your own screen and format your reports, and a procedure editor which lets you tailor QL Archive to your own requirements.

QL Archive is ideal for all database uses, yet it's powerful enough to be used by many software houses to generate specific database applications.



Non-members of QLUB can purchase new versions of the above software for £15 per title, or £50 for all four programs. Phone (0276) 686100 for details.



(Left to right) QL Entrepreneur, QL Project Planner and QL Decision Maker from Sinclair.

Coming soon- QL·Entrepreneur, QL·Project Planner, QL·Decision Maker!

Three new QL business programs – with a difference!

QL Entrepreneur, QL Project Planner and QL Decision Maker train you to apply new and exciting management skills – through original and powerful means!

An interactive teaching program gives you a thorough and enjoyable understanding of each subject – backed by a text-book and 'self-test' facilities - and an applications program helps you to use your new expertise for specific problems and projects.

All three titles will increase your understanding and extend your control – making involved subjects easy, stimulating and useful!

QL·Entrepreneur

QL Entrepreneur is an essential program for anyone preparing to start a new business – whatever it may be!

It uses a 'question and answer' format to help you build a workable business plan.

With the input you give, it works out the break-even point of the business; the first 18 months' cash flow, the type of finance needed; the year end Balance Sheet and Profit and Loss accounts . . . and more!

QL Entrepreneur builds your skills and techniques.

It's flexible too, so that you can ask complex 'what if' questions at any stage!

The program comes with a third, blank Microdrive cartridge and a comprehensive A5 manual.

QL·Project Planner

QL Project Planner will produce plans you can understand, monitor and more easily achieve.

First, you break the project down into its individual activities, telling QL Project Planner how long each takes and which are inter-dependent.

When you decide on a starting time/date QL Project Planner will tell you when each activity must start and finish and when the project will be completed.

Each activity is divided into its critically important stages – those which can safely be moved around without altering the time taken by the project and those where movement will affect the completion deadline.

Whether or not you've used project planning systems before, you'll be amazed at the difference QL Project Planner can make.

The program comes with a third, blank Microdrive cartridge and a comprehensive A5 manual.

QL·Decision Maker

Whether you're thinking of buying a house, or taking on a new business contract, QL Decision Maker makes the choices clearer!

It lets you look at the possibilities – and their implications – through a decision tree.

Once you've set out the decisions and their probable costs or results, QL Decision Maker shows the outcomes which would occur from each particular route.

You can see how much money a decision could make for you . . . or cost you. Complex 'what if' questions are dealt with swiftly and graphically.

You can depend on the QL to highlight the best possible route!

QL Decision Maker comes with a third, blank Microdrive cartridge and a comprehensive A5 manual.

All three programs are available from Sinclair stockists, price £39.95 each, or Sinclair Research. Tel: (0276) 686100.

sinclair

Now, buy a QL and you're a member of the QLUB-free!

QLUB is the special Users Bureau for Sinclair QL owners.

Already, there are well over 10,000 QLUB members . . . enjoying a whole range of information and advisory services.

Until now, joining QLUB cost £35 per year. From March 4, every new QL owner can become a member – free of charge!

With your new QL, you'll find a post-paid form. Complete and mail it, and you'll soon be a member of the fastest growing computer club in the country.

And you'll enjoy all the helpful services listed here!

What QLUB membership offers you

Regular newsletters delivered to your door

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.

Special discounts

QLUB members also receive a range of special discounts, with savings of at least 20% on selected software products.

There are also special subscription rates for Personal Computer News and QL User.

Free Helpline service from Psion

All QLUB members are entitled to 12 months special assistance from Psion.

They're at the end of the telephone to answer any questions on using the QL Abacus, Archive, Easel and Quill programs supplied with the computer.

Help is also available on any aspect of using Sinclair SuperBASIC, Qdos, or linking your QL with major peripherals.

Psion will normally answer any queries within 48 hours.

QL program updates are no longer available free to QLUB members. They will be sold separately.

Good news for existing QLUB members too!

As one of the first members of QLUB, you should already have received one *free* update of each of the four QL programs – and a letter with your new membership details.

If for any reason you haven't, you should ring (0276) 686100.

You're a QL owner, but not a QLUB member?

Then joining QLUB is easy and free! Ring (0276) 685311 for full details. You can be a full QLUB member within a few days.

Where to find the QL

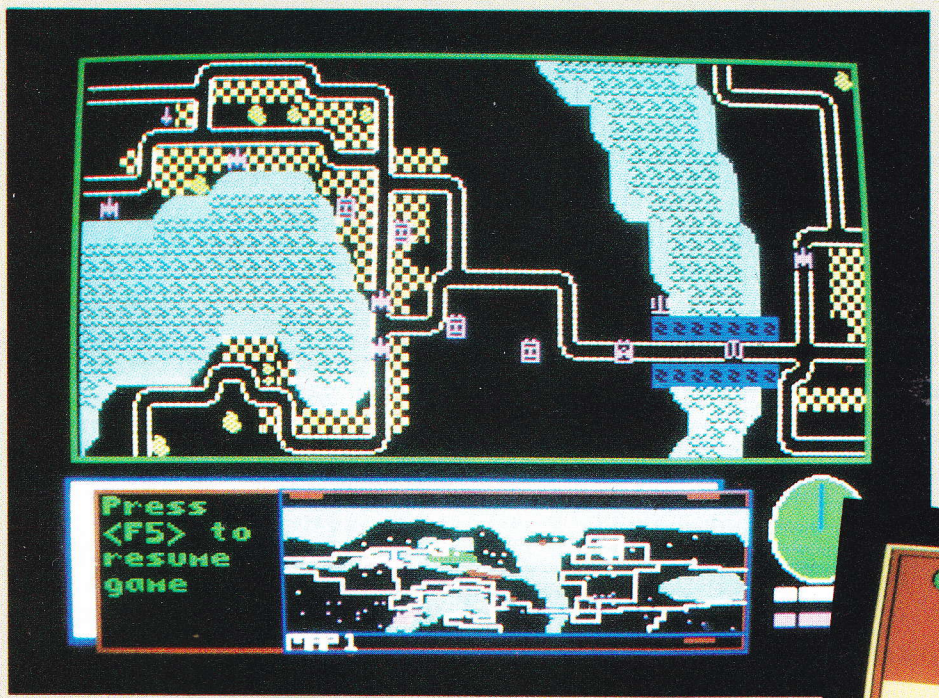
The Sinclair QL is available at selected branches of Dixons, WH Smith, John Lewis Partnership, Currys, Greens in Debenhams and Ultimate, and larger branches of Boots, John Menzies and specialist computer stores nationwide.

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

sinclair

D-DAY

War games move from behind darkened rooms of the MoD to your QL. Martin Croft battles it out.



D-Day has the distinction of being one of the first games produced for the QL. It's not a gamble that any established software house would make lightly – Games Workshop don't quite fit that description.

The Workshop is best known in the fantasy and role playing game field. But this young company's interests now stretch over the lucrative American import market, games publishing, backed up by the house magazine, White Dwarf.

Involvement in software was the next logical step forward. The first three games, Tower of Despair, Battle Cars and D-Day, all for the Spectrum appeared in autumn last year. The latter has now been developed for the QL. It features some 200K of programming, with 20K devoted to computer intelligence in two microdrive cassettes.

On loading, the player is presented with a series of menus. The first is the choice of a one or two player game, then if the one player option is picked, whether the computer should play the Allies or Germans.

Next comes the choice of scenario. Although the game is called D-Day, only two of the four scenarios actually cover the Normandy campaign. 'The Landing' is just that – D-Day itself, while 'The Breakout' is based on the eventual collapse of the German line. The other two scenarios revolve round Arnhem – 'To Arnhem' is the push to relieve the airbourne attempt to take the Arnhem Bridges, while 'Arnhem Invasion' is the actual airbourne drop.

The last choice needed before play can begin is whether the forces should be small or large, ie, 15 units a side, or 50. At this point, you can see the map, whichever size was chosen.

The screen is divided into two areas. Displayed in the upper window is a tactical map. On this are marked terrain features and units such as tanks and mortars. Both the units and individual terrain symbols are a character-sized block.

Below the tactical map is a second window used to list instructions or provide information. The player can call up information on a particular unit by placing it in the centre of the tactical map and hitting F1.

The actual mechanics of play are simple. The player whose turn it is has two choices for each unit – to move or fire it. Movement and firing of units need not take place in any particular order – you can go back and forward as the fancy takes you.

To move a unit, first select and centre it on the tactical map, then hit ENTER. The computer will give you the options of Automatic or Manual movement. In Automatic, a unit will travel North, South, East or West, until it has either run out of movement points, bumped into another unit or the edge of the playing area. Manual movement means that the player moves the units square by square using the cursor keys.

Assuming the firer and target are within range and line of sight, you go into the firing routine. A crooked line is drawn between the two and a siren sounds. After this, one of two messages appears – 'unit damaged' or 'unit destroyed'. About four shots are needed to knock out the target.

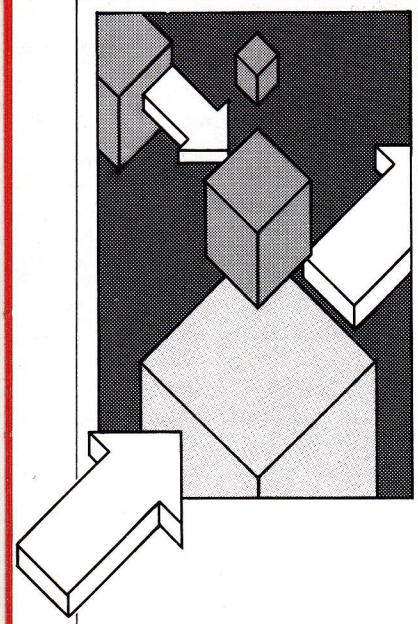
The other permissible action is close combat, whereby you simply move onto an enemy unit. An audible warning sounds, and the battle progress is charted by two bars, shown in the bottom window. These reflect the amount of damage being done by attacker and defender. The only result of close combat seems to be the total destruction of one of the units.

Once one player has finished moving and firing all his units, play moves to his opponent, who performs exactly the same steps. The passage of time is charted by a clock in the bottom left hand corner of the screen, along with losses taken by each side. The main drawback with the game is the size of the map. With 50 units a side, it is far too small. There is no room for tactical thinking – all a commander can do is put down his head and charge, and each game becomes a mindless slugfest, although 15 unit armies allow far more fluid engagements to develop.

Graphically, D-Day is very impressive. The QL's high definition has been married to a board wargame-style map and counter set. The symbols used to denote the various sorts of infantry will be familiar to those used to Avalon Hill or SPI cardboard counters, although they may seem arcane to the uninitiated.

Though the programming is more than adequate the game suffers from planning problems. An increase on the 20K devoted to computer intelligence would be a start.

As it stands, the program is able to provide a fairly strong non-human opponent, but much more could have been done with the available memory. Not many opponents are willing to wait 15 minutes while one fiddles with a keyboard before they can make their turn.



THE PROGS

Sprite design is the theme of this month's section – programs of all descriptions are considered. Send them to 'The Progs', QL User, Priory Court, 30-32 Farringdon Lane, EC1R 3AU.

Simple Synthesizer

David Pearce

Short and sweet, this program converts the QL keyboard into a piano with the keys arranged as follows
('**S**' being the note C):

---E-R--Y-U-I---P-[
A-S-D-F-G-H-J-K-L-;

The length of each note is set to 1 beat but pressing keys 1 to 5 will increase the beat whilst 0 will reduce it. Tunes of up to 255 notes may be composed, saved, loaded and played back.

```
100 REMark **** QL User - Simple Synthesiser ****
110 x=1:DIM g$(255):pitch=0:l=0:j=0:k=0:time=10:me
nu
120 DEFine PROCedure menu
130 x=1:CLS:PRINT "MUSIC SYNTH"\  "-----------"\\\
"1-keyboard ON ( \= OFF) ";\;"2-Play Back";\;"3-Sa
ve Tune";\;"4-Load Tune";\;"5-Choose Instrument"\"
6-New Tune"
140 REPeat chce
150 b$=INKEY$(-1)
160 IF b$= "1" THEN keyboard
170 IF b$= "2" THEN playback
180 IF b$= "3" THEN save1
190 IF b$= "4" THEN file1
200 IF b$= "5" THEN instrument
210 IF b$="6" THEN DIM g$(255):x=1
220 END REPeat chce
230 END DEFine menu
240 DEFine PROCedure keyboard
250 REPeat gett
260 k$=INKEY$(0)
270 IF k$<>"": musak k$: g$(x)=k$:x=x+1:END IF
280 IF k$="\" THEN EXIT gett
290 IF k$=" " THEN x=x-1
300 END REPeat gett
310 END DEFine keyboard
320 DEFine PROCedure musak (a$)
330 IF a$="a" THEN pitch=83
340 IF a$="s" THEN pitch=78
350 IF a$="d" THEN pitch=69
360 IF a$="f" THEN pitch=61
370 IF a$="g" THEN pitch=58
380 IF a$="h" THEN pitch=50
390 IF a$="j" THEN pitch=43
400 IF a$="k" THEN pitch=37
410 IF a$="l" THEN pitch=35
420 IF a$=";" THEN pitch=30
430 IF a$="p" THEN pitch=32
440 IF a$="[" THEN pitch=28
450 IF a$="'" THEN pitch=26
460 IF a$="e" THEN pitch=73
470 IF a$="r" THEN pitch=65
480 IF a$="y" THEN pitch=53
490 IF a$="u" THEN pitch=46
500 IF a$="i" THEN pitch=40
510 IF a$="1" THEN time=10:END DEFine musak
520 IF a$="2" THEN time=20:END DEFine musak
530 IF a$="4" THEN time=40:END DEFine musak
540 IF a$="3" THEN time=30:END DEFine musak
550 IF a$="0" THEN time=5:END DEFine musak
560 BEEP 0,pitch,(pitch+1),1,k,j:PAUSE time:BEEP
570 END DEFine musak
580 DEFine PROCedure playback
590 x =1
600 IF g$(x)<>"\":musak g$(x):x=x+1:GO TO 600:END
IF
610 END DEFine playback
620 DEFine PROCedure save1
630 INPUT "Name of file?";m$:OPEN_NEW #5, m$
640 FOR x=1 TO 255:PRINT #5,g$(x):END FOR x
650 CLOSE #5
660 END DEFine
670 DEFine PROCedure file1
680 INPUT "name of file?";m$:OPEN #5,m$
690 FOR x=1 TO 255:INPUT #5,g$(x):END FOR x
700 CLOSE #5
710 END DEFine
720 DEFine PROCedure instrument
730 PRINT \\\"options:"\"1-Organ"\"2-Eucalalee"\"3
-Oboe"\"4-Clarinet"\"5-Wah"
740 REPeat cho
750 c$=INKEY$(-1):g = c$ INSTR "12345"
760 IF g THEN EXIT cho
770 END REPeat cho
780 IF g=1 THEN l=0:k=0:j=0:menu
790 IF g=2 THEN l=15:k=-8:j=13:menu
800 IF g=3 THEN l=15:k=7:j=0:menu
810 IF g=4 THEN l=7:k=-3:j=15:menu
820 IF g=5 THEN l=2:k=1:j=14:menu
830 AT 12,0: CLS 2:END DEFine instrument
```

Instant Sprites

Richard Cross

Designing multicoloured sprites by hand may shed light on the intricacies of the QL's screen display but can prove time consuming. This set of programs makes design a matter of seconds and provides a simple way of animating them within a SuperBasic program. All three programs should be typed in and saved to microdrive before they are run.

The first program creates a machine code routine used by the remaining programs. The routine will print a number of user-defined sprites to the screen and permits them to be moved from their last position according to a set of parameters. These parameters appear as 2 bytes of general information followed by a number of 16 byte blocks (one for each sprite). Parameters must be stored in memory directly after the machine code routine. As the routine is 552 bytes long, the start of each parameter block will be:

start_address + 552 + 2 + (16*sprite_number (0 to 255)

start_address is the location where the routine has been loaded into memory.

Once all the parameters have been POKED into memory all that remains is to call the routine. Each call will update the sprite's x,y co-ordinates. In a general form the whole process would appear as follows in a SuperBasic program:

a = RESPR (2000)
LBYTES mdv1_sprite_bytes,a
(POKE data in at a + 552 onwards)
CALL a

```
100 REMark **** QL User -Sprite Designer *****
110 REMark **** Machine Code Routine ****
120 MODE 8:PRINT "Creating Machine Code"
130 a=RESPR(1000)
140 FOR x=a TO a+553:READ r:POKE x,r
150 DELETE mdv1_sprite_bytes:SBYTES mdv1_sprite_by
tes,a,554
160 STOP
170 DATA 78,64,78,114,0,0,75,250,2,32,112
180 DATA 0,27,64,0,1,73,237,0,2,16,45
190 DATA 0,1,176,21,108,74,74,20,103,60,96
200 DATA 0,0,148,97,0,0,126,18,44,0,2
210 DATA 20,44,0,3,74,44,0,1,102,8,80
220 DATA 236,0,1,97,46,96,30,72,167,96,0
230 DATA 97,38,76,159,0,6,97,68,210,44,0
240 DATA 6,212,44,0,7,25,65,0,2,25,66
250 DATA 0,3,97,14,82,45,0,1,73,236,0
260 DATA 16,96,174,112,0,78,117,22,44,0,4
270 DATA 24,44,0,5,122,0,124,0,58,4,202
280 DATA 195,227,141,28,44,0,11,202,198,32,108
290 DATA 0,12,209,197,97,0,1,52,78,117,82
300 DATA 44,0,11,16,44,0,11,176,44,0,10
310 DATA 108,2,78,117,81,236,0,11,78,117,114
320 DATA 0,116,0,118,0,120,0,78,117,122,0
330 DATA 124,0,126,0,78,117,97,246,74,44,0
```

```
340 DATA 6,106,20,26,44,0,2,28,44,0,6
350 DATA 72,134,218,70,12,69,0,0,109,86,96
360 DATA 26,97,218,26,44,0,2,28,44,0,4
370 DATA 229,14,218,70,28,44,0,6,218,70,12
380 DATA 69,0,255,108,68,97,192,74,44,0,7
390 DATA 106,22,26,44,0,3,28,44,0,7,72
400 DATA 134,218,70,12,69,0,0,109,52,96,0
410 DATA 255,28,97,162,26,44,0,3,28,44,0
420 DATA 5,218,70,28,44,0,7,218,70,12,69
430 DATA 0,255,108,34,96,0,255,0,74,44,0
440 DATA 8,103,34,107,86,96,46,74,44,0,8
450 DATA 103,24,107,96,96,36,74,44,0,9,103
460 DATA 22,107,96,96,34,74,44,0,9,103,12
470 DATA 107,104,96,24,81,236,0,6,96,0,255
480 DATA 152,81,236,0,7,96,0,254,200,68,44
490 DATA 0,6,96,0,255,136,68,44,0,7,96
500 DATA 0,254,184,97,0,255,52,18,44,0,2
510 DATA 20,44,0,3,97,0,254,238,81,236,0
520 DATA 1,78,117,97,232,26,44,0,4,80,236
530 DATA 0,2,229,13,155,44,0,2,96,0,255
540 DATA 86,97,212,81,236,0,2,96,0,255,76
550 DATA 97,202,26,44,0,5,80,236,0,3,155
560 DATA 44,0,3,96,0,254,114,97,184,81,236
570 DATA 0,3,96,0,254,104,83,131,36,67,97
580 DATA 66,46,4,83,135,114,8,146,128,38,73
590 DATA 44,10,122,0,54,24,24,3,224,75,26
600 DATA 3,224,45,225,77,26,4,224,45,72,69
610 DATA 26,3,227,45,225,77,26,4,227,45,187
620 DATA 145,84,137,81,206,255,220,34,75,67,233
630 DATA 0,128,38,73,81,207,255,206,112,0,78
640 DATA 117,32,1,226,137,239,138,34,65,211,194
650 DATA 211,252,0,2,0,0,38,9,8,131,0
660 DATA 0,34,67,2,128,0,0,0,3,227,136
670 DATA 78,117,0,0
```

Lunar Lander

This short lunar lander simulation shows how the sprite routine may be included into a SuperBasic program. The object of the game is, using the cursor keys, to negotiate a tunnel and land on a moving platform.

```
100 REMark **** QL User - Lunar Lander ****
110 REMark Run in tv mode
120 init
130 REPeat main
140 screen
150 game
160 INPUT #0\\"Another go ? (y/n) > ";an$
170 IF an$=="n" THEN EXIT main
180 END REPeat main
190 :
200 DEFine PROCedure screen
210 WINDOW #2,512,256,0,0
220 PAPER 0:PAPER #2,0:MODE 8
230 BLOCK #2,511,40,0,215,6
240 BLOCK #2,220,120,0,56,2
250 BLOCK #2,220,120,290,56,2
260 LINE 0,80 TO 70,80 TO 70,20 TO 0,20
270 LINE 165,80 TO 95,80 TO 95,20 TO 165,20
280 END DEFine screen
290 :
300 DEFine PROCedure init
310 RESTORE 960
320 a=260144
330 IF PEEK_W(a)<>20032 THEN x=RESPR(2000):LBYTES
flp2_sprite_bytes,a
340 d=a+1000
350 FOR z=d TO d+895 STEP 2:READ x:POKE_W z,x
360 END DEFine init
370 :
380 DEFine PROCedure game
390 ex=0
400 RESTORE 920
410 FOR z=a+552 TO a+565:READ x:POKE z,x
420 POKE_L a+566,a+1000
430 FOR z=a+570 TO a+581:READ x:POKE z,x
440 POKE_L(a+582),a+1384
450 BEEP 0,100,50,1,100,6,15,15
460 grav=0
470 t=0
480 sxi=a+560:syi=a+561:sx=a+556:sy=a+557:px=a+572
490 REPeat m
500 t=t+1
510 grav=grav+.2:IF grav>1 THEN grav=0
520 CALL a
530 k%=KEYROW(1)
540 IF k%&&16 :POKE sxi,(PEEK(sxi))+1
550 IF k%&&2 :POKE sxi,(PEEK(sxi))-1
560 IF k%&&4 :POKE syi,(PEEK(syi))-1
570 POKE syi,(PEEK(syi))+grav
580 IF PEEK(sy)>45 AND PEEK(sy)<173 :tunnel:IF ex=
1:EXIT m
590 IF PEEK(sy)>203:explode:EXIT m
600 IF PEEK(sy)=195:IF PEEK (sx)=PEEK(px):done:EXI
T m
610 IF PEEK(sy)>196:platform:IF ex=1 THEN EXIT m
620 END REPeat m
630 END DEFine game
640 :
650 DEFine PROCedure explode
660 POKE a+555,0:CALL a
670 POKE a+564,2:POKE a+565,0
680 BEEP
690 BEEP 20000,10,20,10,20,8
700 POKE_W (a+560),0:POKE_L (a+566),a+1640
710 FOR 1=1 TO 50:CALL a
720 END DEFine explode
730 :
740 DEFine PROCedure tunnel
750 IF PEEK(a+556)>109 AND PEEK(a+556)<130 THEN RE
Turn
760 explode
770 ex=1
780 END DEFine tunnel
790 :
800 DEFine PROCedure done
810 BEEP
820 AT 0,0:CSIZE 3,1:PRINT "Well Done !!"
830 CSIZE 0,0:PRINT "You landed in ";t;" time unit
s"
840 BEEP 20000,10,20,1000,1
850 END DEFine done
860 :
870 DEFine PROCedure platform
880 x=PEEK(sx):xx=PEEK(px)
890 IF x<xx+16 AND x>xx-16:explode:ex=1
900 END DEFine platform
910 :
920 DATA 2,0,255,0,40,20,4,16,2,2,0,0,3,0
930 DATA 255,0,128,198,4,16,1,0,1,0,2,0
940 DEFine PROCedure sa:DELETE mdv1_module:SAVE md
v1_module
950 END DEFine
960 DATA 0,0,0,0,0,0,0,0,0,0,0,0
970 DATA 0,0,5,80,0,0,5,80,0,0,63
980 DATA 252,0,0,63,252,0,0,43775,43775,0,515
990 DATA 513,32832,32960,2060,2565,41040,8240,1081
5,0,0,43260
1000 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1010 DATA 0,0,0,0,0,0,0,0,0,0
1020 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1030 DATA 0,0,5,80,0,0,5,80,0,0,63
1040 DATA 252,0,0,63,252,0,0,43775,43775,0,515
1050 DATA 513,32832,32960,2060,2565,41040,8240,108
15,3,192,43260
1060 DATA 12,3,192,48,0,3,192,0,0,0,0
1070 DATA 0,0,0,0,0,0,0,0,0,0
1080 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1090 DATA 0,0,5,80,0,0,5,80,0,0,63
1100 DATA 252,0,0,63,252,0,0,43775,43775,0,515
1110 DATA 513,32832,32960,2060,2565,41040,8240,108
15,2,128,43260
1120 DATA 8,2058,8352,32,32,2570,41120,8,0,10794,1
0280
1130 DATA 0,0,32896,2570,0,0,0,0,0
1140 DATA 0,0,0,0,0,0,0,0,0,0,0
1150 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1160 DATA 0,0,0,0,0,0,0,32960,0,0,515,43775
1170 DATA 43775,43775,43775,43712,43520,43520,4352
3,43775,43775,43775,43775
1180 DATA 32963,112,13,707,15,124,61,240,5,84,21
1190 DATA 80,15,124,61,240,3,112,13,192
1200 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1210 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1220 DATA 0,0,0,0,0,0,32960,0,0,515,43775
1230 DATA 43775,43775,43775,43712,43520,43520,4352
3,43775,43775,43775,43775
```

THE PROGS

```
1240 DATA 32963,240,15,707,12,204,51,48,15,60,60
1250 DATA 240,12,204,51,48,3,240,15,192
1260 DATA 0,0,0,0,0,0,0,0,0,0,0
1270 DATA 0,0,0,2056,0,0,2056,41120,0,0,35466
1280 DATA 41120,0,0,10282,32896,0,0,2570,32896,0,0
1290 DATA 2056,8224,0,0,8738,0,0,0,514,0,0
1300 DATA 0,0,0,0,0,0,0,0,0,0,0
1310 DATA 0,0,0,0,0,0,0,0,0,0
1320 DATA 0,0,0,0,0,0,0,0,0,0,0
1330 DATA 0,0,10,8,0,2,10,160,0,0,170
1340 DATA 168,128,0,42,168,0,0,683,33000,32,0
1350 DATA 555,33002,0,0,2094,170,0,0,170,170,0
1360 DATA 2,10,40,0,0,2,138,0,0,32,0
1370 DATA 0,0,0,2,0,0,0,0,0
```

Sprite Designer

The third of our programs is the sprite designer proper. When run it will ask you for the size of the grid you require and re-present pixels on a screen. This will then be displayed and using cursor keys along with ENTER, you will be permitted to fill in the boxes with a variety of colours. Once a sprite has been designed pressing '\' will print it out on screen, along with the values which need to be POKEd in to create it. There is also an option to generate your own BASIC program lines similar to those in the demonstration program (830–1240). Finally, all sprites may be saved to tape.

```
10 REMark **** QL User_Sprite Designer ****
110 OPEN #4,ser1z:BAUD 9600:init
130 WINDOW #1,470,200,25,10:WINDOW #2,470,200,25,1
0
150 MODE 8:INK 7:PAPER 200:STRIP 0:CLS
160 INPUT #0;"x size (groups of four pixels) ?";xs
170 INPUT #0;"y size ( pixels) ? ";ys
190 CLS #0:xs=xs*4:xs=xs-1:ys=ys-1
210 DIM gr(xs,ys):grid xs,ys:x=0:y=0:col=7:cur
250 REPeat main
260 BLOCK 40,20,400,160,col:k$=INKEY$
280 key=CODE(k$):SELect ON key=48 TO 55:col=k$
290 key=KEYROW(1)
310 IF (key&&128)>0 AND y<ys:cur:y=y+1:cur
320 IF (key&&16)>0 AND x<xs:cur:x=x+1:cur
330 IF (key&&4)>0 AND y>0:cur:y=y-1:cur
340 IF (key&&2)>0 AND x>0:cur:x=x-1:cur
350 IF (key&&1)>0:enter
360 IF key=32:out
370 IF key=8:esc
390 END REPeat main
410 DEFine PROCedure grid (along,up)
420 AT 19,0:PRINT "files-press ESC,output-press \"
430 STRIP 0:RESTORE :FOR a=0 TO 7:READ c$:AT (a)*2
,31:PRINT a;"-";c$
440 bw=(INT(180/(along+1)))*2:bh=INT(180/(up+1))
460 BLOCK bw*(along+1),bh*(up+1),10,5,0
470 END DEFine grid
490 DEFine PROCedure cur
510 OVER -1:BLOCK 2,2,(x*bw)+10+(bw/2),(y*bh)+5+(b
h/2),7:OVER 0
530 END DEFine cur
550 DEFine PROCedure enter
560 gr(x,y)=col
570 BLOCK bw,bh,(x*bw)+10,(y*bh)+5,col:cur
590 END DEFine enter
610 DEFine PROCedure out
630 CLS:INPUT "1 -Dump data to printer"\"2 -Create
 BASIC DATA lines"\\"<enter> -neither "\\"? ";r$
635 pri=0
640 IF r$=="1" THEN pri=1:INPUT\\"title? ";t$:PRIN
T #4\t$\
645 IF r$=="2" THEN pri=2:initstore
670 PAPER 0:CLS:addr=udgs
680 FOR d=0 TO  ys
690 FOR a=0 TO xs STEP 4
700 word=calc((gr(a,d)),(gr(a+1,d)),(gr(a+2,d)),(g
r(a+3,d)))
705 IF pri=1 THEN PRINT ;word!;
710 IF pri=1 THEN PRINT #4;word!;
715 IF pri=2 THEN store(word)
720 POKE_W addr,word:addr=addr+2
730 END FOR a
740 IF pri=1 THEN PRINT #4;\
750 END FOR d
755 IF pri=2 THEN PRINT #5:CLOSE #5
760 CALL multi+446,50,50,((xs+1)/4),ys+1,0,0,0,udg
s
770 PRINT #0;"press any key":PAUSE:PAUSE:reprint
790 END DEFine out
810 DEFine PROCedure reprint
820 PAPER 200:CLS:CLS #0:grid xs,ys
840 FOR d=0 TO ys
850 FOR a=0 TO xs
860 BLOCK bw,bh,(a*bw)+10,(d*bh)+5,gr(a,d)
870 END FOR a
880 END FOR d
890 cur
900 END DEFine reprint
920 DEFine FuNction calc (v4,v3,v2,v1)
930 v1=(v1&&3)+((v1&&4)*128)
940 v2=((v2&&3)*4)+((v2&&4)*512)
950 v3=((v3&&3)*16)+((v3&&4)*2048)
960 v4=((v4&&3)*64)+((v4&&4)*8192)
970 RETurn v1+v2+v3+v4
980 END DEFine calc
1000 DEFine PROCedure esc
1030 MODE 8:PAPER 0:PRINT "1 -load character"
1040 PRINT "2 -save character"
1050 PRINT "3 -new charcter"
1060 INPUT \\"choice? ";c$
1070 IF c$="1" THEN lochar
1080 IF c$="2" THEN sachar
1090 IF c$="3" THEN RUN
1100 reprint
1110 END DEFine esc
1130 DEFine PROCedure lochar
1140 INPUT \\"file name (including device)? '";f$
1150 OPEN_IN #5,f$:INPUT #5;xs:INPUT #5;ys
1180 DIM gr(xs,ys)
1190 FOR a=0 TO xs
1200 FOR b=0 TO ys
1210 INPUT #5;gr(a,b)
1220 END FOR b
1230 END FOR a
1240 CLOSE #5
1250 END DEFine lochar
1270 DEFine PROCedure sachar
1280 INPUT \\"filename (including device)? ";f$
1300 DELETE f$:OPEN_NEW #5;f$:PRINT #5;xs\ys
1320 PRINT #5;gr:CLOSE #5
1330 END DEFine sachar
1350 DEFine PROCedure initstore
1351 REPeat gett
1360 INPUT "filename (including MDV?_)? ";file$
1370 IF "mdv" INSTR file$ THEN EXIT gett
1375 IF "flp" INSTR file$ THEN EXIT gett
1390 END REPeat gett
1410 DELETE file$:OPEN_NEW #5,file$
1415 lnum=10:count=0
1430 END DEFine initstore
1450 DEFine PROCedure store(num)
1460 IF count=0 THEN PRINT #5;lnum;" DATA ";num;:l
num=lnum+10:count=count+1:RETurn
1470 IF count<10 THEN PRINT #5;",";num;:count=coun
t+1:RETurn
1480 IF count=10 THEN PRINT #5;",";num:count=0
1490 END DEFine store
3000 DEFine PROCedure init
3020 multi=RESPR(2000):udgs=multi+1000
3030 LBYTES mdv1_sprite_bytes,multi
3040 END DEFine init
3050 DATA "black ","blue  ","red   ","pink  ","gre
en ","cyan  ","yellow","white "
```

DIY Assembler II

Giles Todd

Last month we published the first installment of our type-it-yourself Assembler. Written in SuperBasic and called QSNAIL the program can handle all of the 6800's instruction set. It's only drawback is that it's a little slow. Full info last month.

```
2390 DEFine FuNction read_line$
2400 LOCal line$,get$,outloop,loop
2410 get$=""
2420 line$=""
```

```
2430 REPeat outloop
2440 REPeat loop
2450 IF EOF(#5) THEN
2460    line$="END"
2470    EXIT loop
2480 END IF
2490 get$=INKEY$(#5)
2500 IF CODE(get$)=13 OR CODE(get$)=10 THEN EXIT l
oop
2510 line$=line$&get$
2520 END REPeat loop
2530 IF pass=1 AND CODE(get$)=10 THEN PRINT #chann
el
2540 IF line$<>"" THEN EXIT outloop
2550 END REPeat outloop
2560 RETurn line$
2570 END DEFine read_line$
2580 :
2590 DEFine PROCedure close_file(f$)
2600 CLOSE #5
2610 END DEFine close_file
2620 :
2630 DEFine FuNction next_field$(l$)
2640 LOCal f$,i,loop
2650 f$=""
2660 IF l$="" THEN RETurn f$
2670 i=0
2680 REPeat loop
2690 i=i+1
2700 IF l$(i)<>" " AND CODE(l$(i))<>9 THEN EXIT lo
op
2710 IF i=LEN(l$) THEN EXIT loop
2720 END REPeat loop
2730 IF i=LEN(l$) AND (l$(i)=" " OR CODE(l$(i))=9)
 THEN RETurn ""
2740 REPeat loop
2750 f$=f$&l$(i)
2760 IF i=LEN(l$) THEN EXIT loop
2770 i=i+1
2780 IF l$(i)=" " OR CODE(l$(i))=9 THEN EXIT loop
2790 END REPeat loop
2800 IF i=LEN(l$) THEN l$="":ELSE l$=l$(i TO)
2810 RETurn f$
2820 END DEFine next_field$
2830 :
2840 DEFine PROCedure initialise_arrays
2850 LOCal i
2860 hex$="0123456789ABCDEF"
2870 max_symbols=200
2880 DIM mnemonic$(63,5),symbol$(max_symbols,8),sy
mbol_address(max_symbols),condition$(15,2),cond%(1
5),word(5),shift(15)
2890 FOR i=0 TO 15:shift(i)=2^i
2900 RESTORE 3010
2910 FOR i=0 TO 63
2920 READ mnemonic$(i)
2930 END FOR i
2940 FOR i=0 TO 18
2950 READ symbol$(i)
2960 END FOR i
2970 FOR i=0 TO 15
2980 READ condition$(i),cond%(i)
2990 END FOR i
3000 END DEFine initialise_arrays
3010 DATA "ADD","AND","ASL","ASR","Bcc","BRA","BSR
","CLR","CMP","EOR","JMP","JSR","LSL","LSR","MOVE"
,"OR","ROL","ROR","RTS","SUB"
3020 DATA "ABCD","BTST","DBcc","EXG","MOVEM","MOVE
P","MULS","MULU","NEG","NOP","NOT","ROXL","ROXR","
RTE","RTR","SBCD","STOP","SWAP","TST"
3030 DATA "BCHG","BCLR","BSET","CHK","DIVS","DIVU"
,"EXT","LEA","LINK","NBCD","PEA","RESET","Scc","TA
S","TRAP","TRAPV","UNLK"
3040 DATA "ORG","END","EQU","SET","REG","DC","DCB"
,"DS"
3050 DATA "D0","D1","D2","D3","D4","D5","D6","D7",
"A0","A1","A2","A3","A4","A5","A6","A7","PC","CCR"
,"SR"
3060 DATA "CC",4,"CS",5,"EQ",7,"F",1,"GE",12,"GT",
14,"HI",2,"LE",15,"LS",3,"LT",13,"MI",11,"NE",6,"P
L",10,"T",0,"VC",8,"VS",9
3070 :
3080 DEFine FuNction find_symbol(operand$)
3090 LOCal i,address
3100 i=-1
3110 REPeat loop
3120 i=i+1
3130 IF operand$(1 TO 8)=symbol$(i) THEN
3140    address=symbol_address(i)
3150    EXIT loop
3160 END IF
3170 IF i=max_symbols THEN
3180    address=2^33
3190    EXIT loop
3200 END IF
3210 END REPeat loop
3220 RETurn address
3230 END DEFine find_symbol
3240 :
3250 DEFine PROCedure add_symbol(f$,pc)
3260 LOCal i
3270 i=-1
3280 REPeat loop
3290 i=i+1
3300 IF symbol$(i)="" THEN
3310    symbol$(i)=f$
3320    symbol_address(i)=pc
3330    EXIT loop
3340 END IF
3350 IF i>max_symbols THEN
3360    error_count=ferror(f$&" - too many symbols
",error_count)
3370    EXIT loop
3380 END IF
3390 END REPeat loop
3400 END DEFine add_symbol
3410 :
3420 DEFine FuNction mnemonic(f$)
3430 LOCal i,j,s$,index,temp$
3440 index=-1
3450 temp$=f$
3460 s$=temp$(LEN(temp$)-1 TO)
3470 IF s$=".B" OR s$=".W" OR s$=".L" OR s$=".S" T
HEN
3480    temp$=temp$(1 TO LEN(temp$)-2)
3490 END IF
3500 s$=temp$(LEN(temp$))
3510 IF (s$="I" AND temp$<>"BHI" AND temp$<>"SHI"
AND temp$<>"DBHI") OR (s$="A" AND temp$<>"DBRA" AN
D temp$<>"BRA" AND temp$<>"LEA" AND temp$<>"PEA")
OR s$="X" OR (s$="Q" AND temp$<>"DBEQ" AND temp$<>
"BEQ" AND temp$<>"SEQ") OR (s$="M" AND temp$="CMPM
") THEN
3520    temp$=temp$(1 TO LEN(temp$)-1)
3530 END IF
3540 FOR i=0 TO 63
3550 IF temp$=mnemonic$(i) THEN index=i:EXIT i
3560 IF mnemonic$(i)="Bcc" OR mnemonic$(i)="DBcc"
OR mnemonic$(i)="Scc" THEN
3570    IF f$="DBRA" THEN f$="DBF":temp$="DBF"
3580    s$=mnemonic$(i)
3590    s$=s$(1 TO LEN(s$)-2)
3600    FOR j=0 TO 15
3610    IF f$(1 TO LEN(s$&condition$(j)))=s$&condi
tion$(j) AND LEN(temp$)=LEN(s$&condition$(j)) THEN

3620       index=i
3630       EXIT j
3640    END IF
3650    END FOR j
3660    IF index<>-1 THEN EXIT i
3670 END IF
3680 END FOR i
3690 RETurn index
3700 END DEFine mnemonic
3710 :
3720 DEFine FuNction ferror(ferror$,ec)
3730 LOCal i
3740 BEEP 8132,128
3750 PRINT #channel,ferror$
3760 i=ec+1
3770 RETurn i
3780 END DEFine ferror
3790 :
3800 DEFine PROCedure pseudo_op(l$,f$,pc)
3810 LOCal temp$,temp_address,i,length,mask,giles
3820 IF f$="ORG" THEN
3830    temp$=next_field$(line$)
3840    temp_address=eval(temp$)
3850    IF temp_address=2^33 THEN
```

```
3860        error_count=ferror(whole_line$&" - oper
and not defined",error_count)
3870      END IF
3880      pc=temp_address
3890   END IF
3900   IF f$="END" THEN
3910      pc=pc+0
3920   END IF
3930   IF f$="EQU" THEN
3940      temp$=next_field$(line$)
3950      temp_address=eval(temp$)
3960      IF temp_address=2^33 THEN
3970         error_count=ferror(whole_line$&" - oper
and not defined",error_count)
3980      END IF
3990      change_symbol label$,temp_address
4000      pc=pc+0
4010      object$=cvs$(temp_address,4)
4020   END IF
4030   IF f$="SET" THEN
4035      PRINT "SET not implemented"
4040      pc=pc+0
4050   END IF
4060   IF f$="REG" THEN
4070      PRINT "REG not implemented"
4080      pc=pc+0
4090   END IF
4100   IF f$="DCB.B" OR f$="DCB.W" OR f$="DCB.L" OR
f$="DCB" THEN
4110      PRINT "DCB not implemented"
4120      pc=pc+0
4130   END IF
4140   IF f$="DC.B" OR f$="DC.W" OR f$="DC.L" OR f$=
"DC" THEN
4150      length=2-(f$(3 TO)=".B")+2*(f$(3 TO)=".L")
4160      temp_address=0
4170      temp$=next_field$(line$)
4180      object$=""
4190      REPeat giles
4200      i="," INSTR temp$
4210      IF i=0 THEN
4220         IF pass=1 AND eval((temp$))=2^33 THEN o
perr
4230         object$=object$&cvs$(eval(temp$),length
)
4240         temp_address=temp_address+1:EXIT giles
4250      ELSE
4260         IF pass=1 AND eval((temp$(1 TO i-1)))=2
^33 THEN operr
4270         object$=object$&cvs$(eval(temp$(1 TO i-
1)),length)
4280         temp$=temp$(i+1 TO)
4290         temp_address=temp_address+1
4300      END IF
4310      END REPeat giles
4320      pc=pc+temp_address*length
4330      IF pass THEN PRINT #6,object$;
4340   END IF
4350   IF f$="DS.B" OR f$="DS.W" OR f$="DS.L" OR f$=
"DS" THEN
4360      temp_address=eval(next_field$(line$))
4370      IF f$="DS" THEN
4380         pc=pc+temp_address*2
4390         object$=FILL$(CHR$(0),temp_address*2)
4400      ELSE
4410         pc=pc+temp_address*(f$="DS.B")+temp_add
ress*2*(f$="DS.W")+temp_address*4*(f$="DS.L")
4420         object$=FILL$(CHR$(0),temp_address*(f$=
"DS.B")+temp_address*2*(f$="DS.W")+temp_address*4*
(f$="DS.L"))
4430      END IF
4440      IF pass THEN PRINT #6,object$;
4450   END IF
4460   END DEFine pseudo_op
4470   :
4480   DEFine FuNction dec2hex$(i,flag)
4490   LOCal low,high,result$
4500   IF i<0 THEN i=65536+i
4510   high=i
4520   result$=""
4530   REPeat loop
4540   low=(high/16-INT(high/16))*16
4550   high=INT(high/16)
4560   IF high=0 AND low=0 THEN EXIT loop
4570   result$=hex$(low+1)&result$
4580   END REPeat loop
4590   REPeat loop
4600   IF flag AND result$="" THEN result$="00"
4610   IF flag THEN EXIT loop
4620   IF LEN(result$)>=5 THEN EXIT loop
4630   result$="0"&result$
4640   END REPeat loop
4650   RETurn result$
4660   END DEFine dec2hex$
4670   :
4680   DEFine FuNction eval(t$)
4690   LOCal result
4700   result=2^33
4710   IF t$="" THEN RETurn result
4720   IF t$="*" THEN RETurn program_counter
4730   IF "[" INSTR t$ THEN RETurn expression((t$))
4740   IF t$(1)="#" THEN t$=t$(2 TO)
4750   IF t$(1)="'" THEN result=CODE(t$(2))
4760   IF (CODE(t$(1))>=48 AND CODE(t$(1))<=57) OR t
$(1)="-" THEN
4770      result=t$
4780   END IF
4790   IF t$(1)="$" THEN
4800      result=hex(t$(2 TO))
4810   END IF
4820   IF result=2^33 THEN
4830      IF LEN(t$)>8 THEN t$=t$(1 TO 8)
4840      result=find_symbol(t$)
4850   END IF
4860   RETurn result
4870   END DEFine eval
4880   :
4890   DEFine PROCedure change_symbol(label$,address
)
4900   LOCal i
4910   FOR i=0 TO max_symbols
4920   IF label$(1 TO 8)=symbol$(i) THEN
4930      symbol_address(i)=address
4940      EXIT i
4950   END IF
4960   END FOR i
4970   END DEFine change_symbol
4980   :
4990   DEFine FuNction count_operands(operand$)
5000   LOCal result,i,outloop,loop
5010   IF operand$="" THEN RETurn 0
5020   result=1:i=0
5030   REPeat outloop
5040   i=i+1
5050   IF i>LEN(operand$) THEN EXIT outloop
5060   IF operand$(i)="," THEN
5070      result=2
5080      EXIT outloop
5090   END IF
5100   IF operand$(i)="(" THEN
5110      REPeat loop
5120      i=i+1
5130      IF i>LEN(operand$) THEN
5140         result=0
5150         EXIT loop
5160      END IF
5170      IF operand$(i)=")" THEN EXIT loop
5180      END REPeat loop
5190   END IF
5200   IF result=0 THEN EXIT outloop
5210   END REPeat outloop
5220   RETurn result
5230   END DEFine count_operands
5240   :
5250   DEFine FuNction first_operand$(operand$)
5260   LOCal i,temp$,loop,outloop
5270   i=0:temp$=""
5280   REPeat outloop
5290   i=i+1
5300   IF operand$(i)="," THEN EXIT outloop
5310   temp$=temp$&operand$(i)
5320   IF operand$(i)="(" THEN
5330      REPeat loop
5340         i=i+1
5350         temp$=temp$&operand$(i)
5360         IF operand$(i)=")" THEN EXIT loop
5370      END REPeat loop
5380   END IF
5390   END REPeat outloop
5400   RETurn temp$
```

```
5410 END DEFine first_operand$
5420 :
5430 DEFine FuNction second_operand$(operand$)
5440 LOCal i,temp$,loop
5450 i=0:temp$=""
5460 i=LEN(first_operand$(operand$))
5470 i=i+1:REMark avoid comma
5480 REPeat loop
5490 i=i+1
5500 IF i>LEN(operand$) THEN EXIT loop
5510 temp$=temp$&operand$(i)
5520 END REPeat loop
5530 RETurn temp$
5540 END DEFine second_operand$
5550 :
5560 DEFine FuNction operand_type(operand$,field$,
sd$)
5570 LOCal i,result,temp,t$
5580 result=0
5590 IF operand$="CCR" OR operand$="SR" OR operand
$="USP" THEN RETurn result
5600 IF count_operands(operand$)=0 THEN RETurn res
ult
5610 IF LEN(operand$)>1 AND NOT "(" INSTR operand$
 THEN
5620 temp=eval(operand$(2 TO))
5630 IF operand$(1)="#" THEN
5640    IF temp>=-32768 AND temp<=32767 AND NOT in
side(".L",field$) THEN
5650       result=12
5660    ELSE
5670       result=13
5680    END IF
5690 END IF
5700 END IF
5710 IF result<>0 THEN RETurn result
5720 FOR i=0 TO 7
5730 IF operand$=symbol$(i) THEN
5740    result=1
5750    EXIT i
5760 END IF
5770 END FOR i
5780 IF "MOVEM" INSTR field$ THEN result=0
5790 IF result<>0 THEN RETurn result
5800 FOR i=8 TO 15
5810 IF operand$=symbol$(i) THEN
5820    result=2
5830    EXIT i
5840 END IF
5850 END FOR i
5860 IF "MOVEM" INSTR field$ THEN result=0
5870 IF result<>0 THEN RETurn result
5880 FOR i=8 TO 15
5890 IF operand$="("&symbol$(i)&")" THEN
5900    result=3
5910    EXIT i
5920 END IF
5930 END FOR i
5940 IF result<>0 THEN RETurn result
5950 FOR i=8 TO 15
5960 IF operand$="("&symbol$(i)&")+" THEN
5970    result=4
5980    EXIT i
5990 END IF
6000 END FOR i
6010 IF result<>0 THEN RETurn result
6020 FOR i=8 TO 15
6030 IF operand$="-("&symbol$(i)&")" THEN
6040    result=5
6050    EXIT i
6060 END IF
6070 END FOR i
6080 IF result<>0 THEN RETurn result
6090 IF "(A" INSTR operand$ THEN
6100 FOR i=8 TO 15
6110 IF operand$(LEN(operand$)-3 TO)="("&symbol$(i
)&")" THEN result=6:EXIT i
6120 END FOR i
6130 END IF
6140 IF result<>0 THEN RETurn result
6150 IF "(A" INSTR operand$ AND "," INSTR operand$
 AND ")" INSTR operand$ THEN result=7
6160 IF result<>0 THEN RETurn result
6170 IF "(PC)" INSTR operand$ THEN
6180       result=10
6190 END IF
6200 IF result<>0 THEN RETurn result
6210 IF "(PC," INSTR operand$ THEN
6220    result=11
6230 END IF
6240 IF result<>0 THEN RETurn result
6280 temp=eval(operand$)
6290    IF (temp>=0 AND temp<=hex("7FFF")) OR (tem
p>=hex("80000") AND temp<=hex("FFFFF")) THEN
6300       result=8:REMark absolute short
6310    ELSE
6320       result=9:REMark absolute long
6330    END IF
6340 IF field$(1 TO LEN("MOVEM"))="MOVEM" AND ("/"
 INSTR operand$ OR "-" INSTR operand$ OR reg(opera
nd$,0)<>-1) THEN result=16
6350 IF result=0 THEN result=9
6360 RETurn result
6370 END DEFine operand_type
6380 :
6390 DEFine FuNction overhead(type,f$,s$)
6400 IF f$="MOVEQ" THEN RETurn 0
6410 SELect ON type
6420    =1 TO 5
6430       RETurn 0
6440    =6 TO 8
6450       RETurn 2
6460    =9
6470       RETurn 4
6480    =10 TO 11
6490       RETurn 2
6500    =12
6510       IF f$(1 TO 2)="LS" OR f$(1 TO 2)="AS" O
R f$(1 TO 2)="RO" OR f$="TRAP" OR "Q" INSTR f$ THE
N RETurn 0:ELSE RETurn 2
6520    =13
6522       IF ".W" INSTR f$ OR ".B" INSTR f$ THEN
RETurn 2
6530       IF "Q" INSTR f$ THEN RETurn 0:ELSE RETu
rn 4
6540    =16
6550       RETurn 2
6560    =REMAINDER
6570       RETurn 0
6580 END SELect
6590 END DEFine overhead
6600 :
6610 DEFine FuNction short_branch(f$,link,operand$
)
6620 IF link<>4 AND link<>5 AND link<>6 THEN RETur
n 0
6630 IF f$(LEN(f$)-1 TO)=".S" THEN RETurn 1
6640 IF (eval(operand$)-program_counter)<-126 OR (
eval(operand$)-program_counter)>129 THEN RETurn 0
6650 RETurn 1
6660 END DEFine short_branch
6670 :
6680 DEFine FuNction branch(f$,link)
6690 IF link<>4 AND link<>5 AND link<>6 AND link<>
22 THEN
6700    RETurn 0
6710 ELSE
6720    RETurn 1
6730 END IF
6740 END DEFine branch
6750 :
6760 DEFine PROCedure reset_pointer(f$)
6770 close_file f$
6780 open_file f$
6790 DELETE outfile$
6800 OPEN_NEW #6,outfile$
6810 END DEFine reset_pointer
6820 :
6830 DEFine PROCedure print_object(o$)
6840 LOCal i,j,temp$
6850 IF o$="" THEN
6860    RETurn
6870 ELSE
6880    IF LEN(o$)>10 THEN j=10:ELSE j=LEN(o$)
6890    PRINT #channel," ";
6900    FOR i=1 TO j
6910    temp$=dec2hex$(CODE(o$(i)),1)
6920    IF LEN(temp$)=1 THEN temp$="0"&temp$
6930    PRINT #channel,temp$;
6940    END FOR i
```

"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
800KB + 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

MACHINE CODE TUTORIAL

Continuing our series Adam Denning looks at some elementary 68000 Assembler commands and introduces exception handling.

In the last issue we looked at the general make up of the QL's 68008 microprocessor and surrounding hardware, and discovered that it was more convenient to use binary and hexadecimal number systems as opposed to decimal. This isn't the only difficulty we have to overcome. As a microprocessor's machine code is much less sophisticated than a high level language it is not possible to declare things like string variables, procedures and so on. An entirely different approach to programming is called for.

To illustrate this we shall start by examining a few microprocessor operations.

On The Move

The most fundamental operation a processor can perform is to move data (ie numbers) from one place to another. The mnemonic used in 68000 assemblers to signify data movement is MOVE. It gives us the power to move data just about anywhere we like, in and out of memory and registers, and even to or from outside devices.

In addition to being able to move information we can also perform basic arithmetic upon numbers held in memory or registers. The appropriate mnemonics for addition, subtraction, multiplication and division are respectively ADD, SUB, MUL and DIV. Another operation which falls into this class is the comparison (CMP). This is really no more than a subtraction, but instead of storing the result for later use, the processor discards it, so that only the flags in the status register are affected. Comparisons are used to implement code similar to BASIC's IF . . . THEN statement.

With each item of information held within the computer as a series of binary digits (bits) we also have a few instructions which permit us to manipulate each bit directly, known as logical operators, they are AND, OR, EOR and NOT. NOT is given one operand (number to operate on) and inverts the state of each bit in that number, so that 1001 becomes 0110. We can tell each operation what size of data to work on, but the only options are 8, 16 and 32 bits. AND takes two operands and compares each bit of the first against the corresponding one of the second. If both are set to '1' then the bit in the result is also '1', otherwise it is '0', ie,

11110000 AND
10101010
10100000

The operator OR takes two operands but sets the result bit to '1' if either or both of the operands' bits are '1's. EOR stands for exclusive-OR, and is the same as OR except that the result is set only if one of the operands bits is set, not both. The 'truth tables' below show the results of each operation on one bit.

AND Truth table

Input 1:		Ø	1
Input 2:	Ø	Ø	Ø
	1	Ø	1

Fig 1

OR Truth table

Input 1:		Ø	1
Input 2:	Ø	Ø	1
	1	1	1

Fig 2

EOR Truth table

Input 1:		Ø	1
Input 2:	Ø	Ø	1
	1	1	Ø

Fig 3

These instructions are used for 'masking' bits from an item of data, as AND will remove certain bits, OR will add them, and EOR will 'toggle' them. Toggling switches the state of a bit. If we had a number with bit 0 set and EORed it with 1, the result would have bit 0 reset. If we EORed it with 1 again, bit 0 would be set once more.

Again taking each number as a bit pattern, we can shuffle the bits right or left in a number of ways. If we shuffle them left and fill the empty bits with zeros, this is called a 'left shift'. The mnemonic is LSL. If we shuffle the bits right and fill the

empty bits with zeros, it's called a 'right shift', and the mnemonic is 'LSR'. However, as numbers can be signed, in which the top bit signifies the sign (0 = plus, 1 = minus), we often want to shift right but preserve the sign. This is called 'arithmetic shift right', and the mnemonic is 'ASR'. If we examine the process closely, we see that shifting left is equivalent to multiplying the number by two to the power of the number of shifts, and shifting right is equivalent to dividing by two to the power of the number of shifts.

We can also shuffle the bits in a slightly different way. If, after shuffling, we put the old value of the highest bit in the lowest, we're doing a 'rotate right', and the mnemonic is 'ROR'. Shuffling left and then putting the old value of the lowest bit into the highest is called 'rotating left' and has the mnemonic 'ROL'. The figures below should make these processes a little clearer.

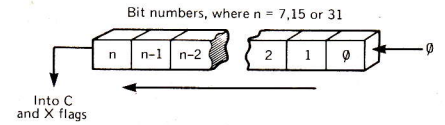


Figure 4 — logical shift left.

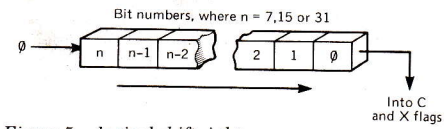


Figure 5 — logical shift right.

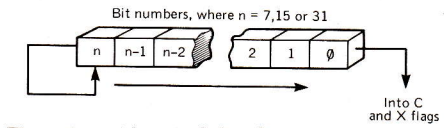


Figure 6 — arithmetic shift right.

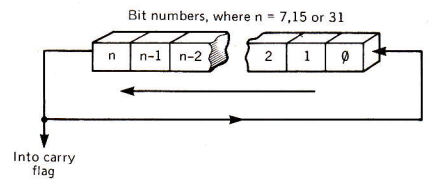


Figure 7 — rotate left.

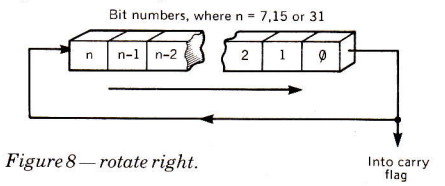


Figure 8 — rotate right.

Shifts and rotates can be performed on data sizes of 8, 16 or 32 bits, and from 0 to 63 operations can be performed in one instruction. Two more rotates exist, ROXL and ROXR, which incorporate the extend flag in the rotation.

Rotating and shifting are useful for their mathematical properties as well as for such tasks as smoothly scrolling graphics across the screen.

BRAnch Line

One of the most unnecessary keywords in SuperBasic is GOTO, as it can be circumvented using the various procedures, functions and select statements. Machine code does not have any of these 'structured' elements though, so the equivalent of a GOTO is very important. This takes two forms. The JMP command permits us to jump to an absolute position in memory and the BRA command allows us to branch to a position relative to our current whereabouts in memory. Invariably the latter command is used as it enables you to write programs that will work no matter where they are loaded into memory.

A number of variations on the BRA command are possible (see tables). These are similar to IF ... THEN GOTO or ON ... GOTO constructions in BASIC, except that the conditional tests are carried out in relation to the contents of the status register.

The ability to go to a subroutine and then come back to where we left off is vital in machine code. The mnemonic for this is JSR, standing for 'jump to subroutine', or 'BSR', standing for 'branch to subroutine'. When one of these instructions is met, the current value of the program counter is 'pushed' onto the A7 stack, so that it can be retrieved when the subroutine ends. The program counter is then loaded with the address of the subroutine, and away it goes. The subroutine usually ends with a return instruction, which has the mnemonic 'RTS'. This 'pulls' the return address off the A7 stack and puts it into the program counter, so the program continues execution at the instruction after the JSR/BSR. Overall, the effect is similar to the now obsolete GOSUB statement in SuperBasic.

BRA	...	branch always	— (tests no flags)
BEQ	...	branch equal	— (if zero flag set (= 1))
BNE	...	branch not equal	— (if zero flag reset (= 0))
BCS	...	branch carry set	— (if carry flag set)
BCC	...	branch carry clear	— (if carry flag clear)
BVS	...	branch overflow	— (if overflow flag set)
BVC	...	branch overflow clear	— (if overflow flag reset)
BMI	...	branch minus	— (if sign flag set)
BPL	...	branch plus	— (if sign flag reset)
BHI	...	branch high	— (if carry and zero flags reset)
BGE	...	branch greater or equal	— (if sign and overflow flags in same state)
BLT	...	branch less than	— (if sign and overflow flags in opposite states)
BGT	...	branch greater than	— (if sign and overflow flags in same state, and zero flag reset)
BLE	...	branch less or equal	— (if sign and overflow flags in opposite states, and zero flag set)

Figure 9 — Table of 68000 mnemonics for conditional branching.

Taking Exception

Learning an elementary set of assembler instructions is only the first step in writing machine code. The next is to get the microprocessor to obey them without interfering with the many other tasks it is involved in. To understand how this is done we must turn our attention to QDOS and its relationship with the 68008 central processor.

Last month, we talked about the status register, which is where the processor keeps all its flags. One we didn't mention was the supervisor mode flag, or 'S' for short. This tells us (and the processor) which of two modes we are in. These are 'user mode' and 'supervisor mode'. We normally run our programs in user mode, and can do almost anything apart from alter the top half of the status register. This half of the register keeps tracks of the system. We can only alter it by switching into supervisor mode and issuing what are known as 'privileged' instructions. If we try to issue these in user mode, something called an 'exception' is generated.

An exception is rather like pressing BREAK when a SuperBasic program is running. However instead of stopping the program and reporting an error, it jumps to a particular location, puts the processor into supervisor mode, and executes whatever code it finds there. The location to which each exception jumps is governed by the 'exception vector', which is an area of memory containing a table of addresses, one for each exception. There are a lot of exceptions, and the one we would have invoked by trying to execute a privileged instruction is called the 'privilege violation' exception.

The important thing to remember about an exception is that in putting us in supervisor mode it lets us reach parts of QDOS. Being able to do this means that we can use a host of ready-made routines which control such things as video display, sound, keyboard and even floating point arithmetic. Without exception we would have to more or less rewrite QDOS everytime we wanted to do something simple. With them, machine code programming is a lot easier and considerably more rewarding. Next month we examine exactly how to get at these routines, as well as look at the QL's addressing modes.

PRINTERLAND

ALL ORDERS DESPATCHED SAME DAY
SELLING PRINTERS SINCE 1980!

FREE with all Printers. QL users Printers Guide. Getting the best from your printer is not easy!

We supply a FREE Booklet with all printers which explains how to obtain all the features available on your printer. Without this you could waste many hours. Buy elsewhere and you'll pay far more and get far less.

DOT MATRIX	EX VAT	INC VAT
BROTHER HR5	£133.00	£152.95
SHINWA CPA 80 (EPSON FX80 COMPATIBLE)	£185.00	£212.75
EPSON RX80	£185.00	£212.75
EPSON RX80 F/T +	£212.00	£244.80
EPSON FX80 F/T	£314.00	£361.10
EPSON FX100 F/T	£425.00	£488.75

DOT MATRIX PLUS NEAR LETTER QUALITY		
KAGA TAXAN 810	£259.00	£297.85
CANON PW 1080A (Special offer)	£249.00	£286.35
KAGA KP910	£349.00	£401.35
CANON 1156	£349.00	£401.35

DAISY WHEEL		
QUENDATA 1120	£225.00	£258.75
JUKI 6100	£325.00	£373.75
EPSON DX100	£356.00	£409.40

QL RS232 INTERFACES	EX VAT	INC VAT
MIRACLE SYSTEMS	£32.50	£37.37
EPSON RS232	£27.00	£31.05

	EX VAT	INC VAT
SINCLAIR QL	£334.00	£384.10

MEMORY EXPANSION BOARDS	EX VAT	INC VAT
64K	£82.00	£94.30
256K	£199.00	£228.85

DISC DRIVES		
MICRO P DISK INTERFACE	£99.00	£113.85
Q DISK DISK INTERFACE	£129.57	£149.00
3 ½" SINGLE DISK DRIVE (.75)	£169.00	£194.35
3 ½" DUAL DISK DRIVE (1.5MB)	£314.00	£361.10
5 ¼" SINGLE DISK DRIVE 200K	£110.88	£150.00
5 ¼" DUAL DISK DRIVE 1.5MB	£347.83	£400.00
FREE — LOTS OF ASSISTANCE + DISKS		

MONITORS	EX VAT	INC VAT
MICROVITEC CUB 653	£220.00	£253.00

FULL PACKAGE OFFER

Sinclair QL. Microvitec Cub 653
Epson RX80 F/T. Price includes all leads and interfaces.
(Other printers also available)

REC RETAIL (INC VAT)	£1070.00
OUR PRICE (INC VAT)	£899.00

SHOWROOM OPEN 9.00 am to 6.00 pm DAILY
9.00 am to 1.00 pm SATURDAY
DELIVERY PRINTERS/MONITORS (SECURICOR) £10.00 Other £1.00

Printerland, Unit 27, Estate Buildings, Railway Street, Huddersfield HD1 1JP

Telephone Huddersfield 514105/687875

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.

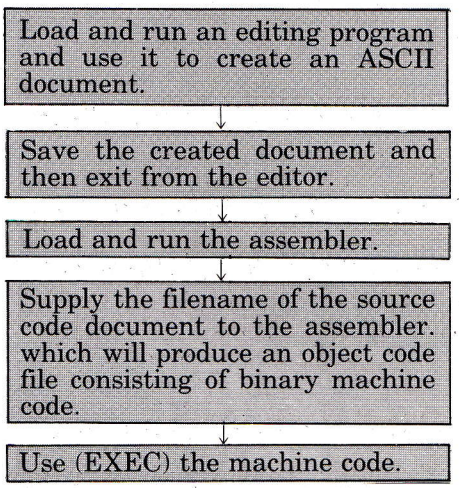
COMPUTER ONE ASSEMBLER

Gareth Jefferson reviews a multitasking assembler/editor from a seasoned software house.

Computer One of Cambridge have created an assembler/editor for the QL that offers all the tools an assembly language programmer is likely to need. It is, however, rather different in use from other assemblers readers may have used.

Traditionally, an assembler is a stand-alone program that simply translates a document (prepared separately) from assembly language source code into machine code. Directives in the source code, or commands available when the assembler is used, allow the machine language object code to be written to memory (and executed from there) or written as files on a mass storage device.

The traditional assembly language programming sequence of activities would therefore look something like this:



This kind of sequential activity can be time-consuming and frustrating. For example, if the source code fails to assemble because of an error, the assembler will report it. The programmer will then have to abort the assembly, invoke the editor, retrieve the document, locate and correct the error, save the document, leave the editor, invoke the assembler and try to assemble the source code again.

The Computer One assembler is in many ways a considerable advance on this old-fashioned approach. Thanks to the power of the QL's 68008 processor, and the multitasking capabilities of the QDOS operating system, the assembler and editor can both be run as tasks at the same time with SuperBasic. The programmer can then use the QDOS CNTL-C command to switch input between jobs.

This is a significant advance in user friendliness as far as the assembly language programmer is concerned, and can be likened to program development in BASIC (where source code can be edited and debugged at any time and run by simply typing RUN). In other words, Computer One's assembler is really a programming environment, rather than just an assembler. Pascal programmers who have tried Borland's Turbo Pascal (available at present only under CP/M or MS-DOS) enjoy a similar programming environment and it is clearly the way of the future.

Looking Glass

Being able to use the editor, assembler and SuperBasic simultaneously does bring problems of its own. Namely, how to display three separate sets of information on a single screen.

When run, or more correctly, EXECd, Computer One's assembler creates a window in the centre of the display where commands may be entered and instruction or error messages appear. This area of the screen is also used by the editor and QDOS, with the result that each window interferes with the other and it is easy to lose track of what job you may be switching to.

Computer One have found a way around this problem which though satisfactory is somewhat inconsistent as it uses different commands for each job. If you are in the editor, for example and the screen happens to be overwritten with rubbish, hitting F4 will refresh the display. In the assembler, instead of F4, the ENTER key is pressed whilst elsewhere normal SuperBasic screen commands apply. Confusing to begin with but after a while you get used to the different commands.

The manual states, reasonably enough that it does not attempt to teach assembly language programming but ironically fails to assume that you can write assembly language source code. It attempts to be a reference manual rather than a tutorial telling you what the program can do rather than how to use it.

A better way would have been to start with a complete, short assembly language source code program. Then, show you how to enter it using the editor, save it as an *asm* file, how to edit it (eg by correcting a deliberate mistake) and assemble it using the assembler. This could then be followed by a concise description of the assembler's syntax, a handy reference listing of the 68000 mnemonics (which are the standard Motorola ones) and finally, a thorough index. This need not take up more than the 38 pages supplied, but would turn purchasers into users much more quickly.

The manual also contains an annoying typographical error. Throughout, 1's and lower case l's use the same character. Attempting to load *C1ass* instead of *Class* will result in the 'not found' error message.

Finally, there is no index – a grave omission!

In operation the assembler is fast, possibly reflecting the fact that it was itself written in 68000 assembly language and easy to use. It accepts standard mnemonics and a syntax very similar to Motorola's. During assembly error codes are produced and the offending lines displayed. Particularly attractive is the fact that the error number will then reappear along with a full description of the error type beneath the offending line in the editor, providing the source file is reloaded from disk or drive. As the assembler does not automatically abort upon encountering an error this means that all errors will be documented in the editor, making it possible to correct all of them in one fell swoop. As for the error detection itself, various source code errors were put through and all were located and accurately diagnosed.

In order to fit both editor and assembler into memory certain sacrifices have had to be made. Only a limited set of directives are supported (listed below). The omission of a GET or INCLUDE command which automatically brings external source files is partially offset by a merge-file facility in the editor. Nevertheless, macros are not supported and whilst provisions have been made for use with a linker, the actual program is not included.

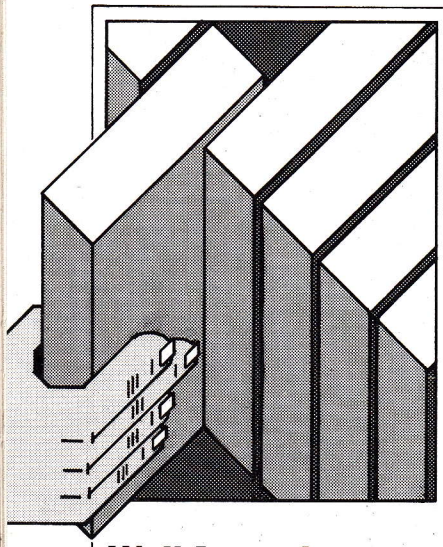
Easy Rider

At £29.95 Computer One's assembler puts the emphasis on ease of use and speed. The absence of macros and a linker are unlikely to affect all but the most proficient programmers. Concurrency, the ability to edit and assemble simultaneously more than makes up for minor shortcomings in the manual and screen handling, and compensates for the limited number of directives available. This gives a professional edge to what is a modestly priced product.

Computer One, Science Park, Milton Rd, Cambs (0223 862616)

Assembler Directives

DC	Define constant
DS	Define storage
DCB	Define constant block
EQU	Equate
ALIGN	Align to word boundary
XREF	External reference
XDEF	External definition
IF . . . [ELSE]	External definition
. . . ENDIF	Conditional assembly



BOOKMARKS

With both serious and recreational programmers in mind Nicky Trevitt investigates the latest in QL literature.

Well Assembled

Anyone interested in programming the QL's advanced Motorola 68008 microprocessor is soon going to find out that this powerful and sophisticated chip is correspondingly complex to program. The hopeful programmer is going to require a good work of reference, and Osborne/McGraw-Hill's *68000 Assembly Language Programming* by Gerry Kane, Doug Hawkins and Lance Leventhal, priced at £11.50 should come top of anyone's list.

This is one in a well-established and respected series which tend to be found everywhere, from electronics departments in universities and polytechnics to the premises of commercial software development houses. They are comprehensive enough for the professionals while remaining easy-to-follow and readable for students and strangers to assembly language programming. They are also 'compatible' with one another in that they share the same structure and layout; work your way through one, and you'll be able to find your way around them all.

Here, you will find 68000 assembly language programming presented in exhaustive detail. The book follows a logical order, starting by looking into the fundamental concepts of assembly language programming, assemblers, 68000 machine structure and assembly language. There is also a useful thumbnail sketch of high-level languages like Cobol and Fortran which summarizes their advantages and disadvantages, and serves as a timely reminder of why you are struggling with assembly in the first place.

The next chapter deals with 'introductory problems', based on the principle of learning by doing. This offers a series of practical programs demonstrating program loops, handling character-coded data, code conversion, arithmetic problems, tables and lists. Section three looks at more advanced topics, like parameter passing techniques, sub-routines, input/output and so on, again with plenty of example programs.

Section four is to be found in all Lance Leventhal books – a guide to better programming and program development, indispensable for all serious programmers. It offers advice on such matters as problem definition, program design, flowcharting, structured programming, top-down design, documentation, debugging, testing and maintenance and re-design – pretty definitive stuff.

The reference part of the book starts with section five, a complete guide to the 68000 instruction set which presents each instruction individually and in detail. This information is summarised in the appendices, which also include 68000 instruction codes and instruction object codes.

I was left with one reservation: the lack of even minimal hardware information. The authors get round this by suggesting that you read McGraw-Hill's *An Introduction to Microcomputers, Volume I – Basic Concepts* before attempting this book, which would provide you with the necessary basic computer knowledge, including addressing methods and the fundamental features of instruction sets. But I could not help feeling that some of this information could have been provided within the book itself – it seems ridiculous that a heavy weight volume on the 68000 chip does not tell you how many address lines there are. It would also have been useful to know about the different versions of the 68000 (the QL's 68008 is, after all, just one member of the family) and their differences.

That said, it is unlikely that the 68000 programmer will be able to find a better treatment of the subject.

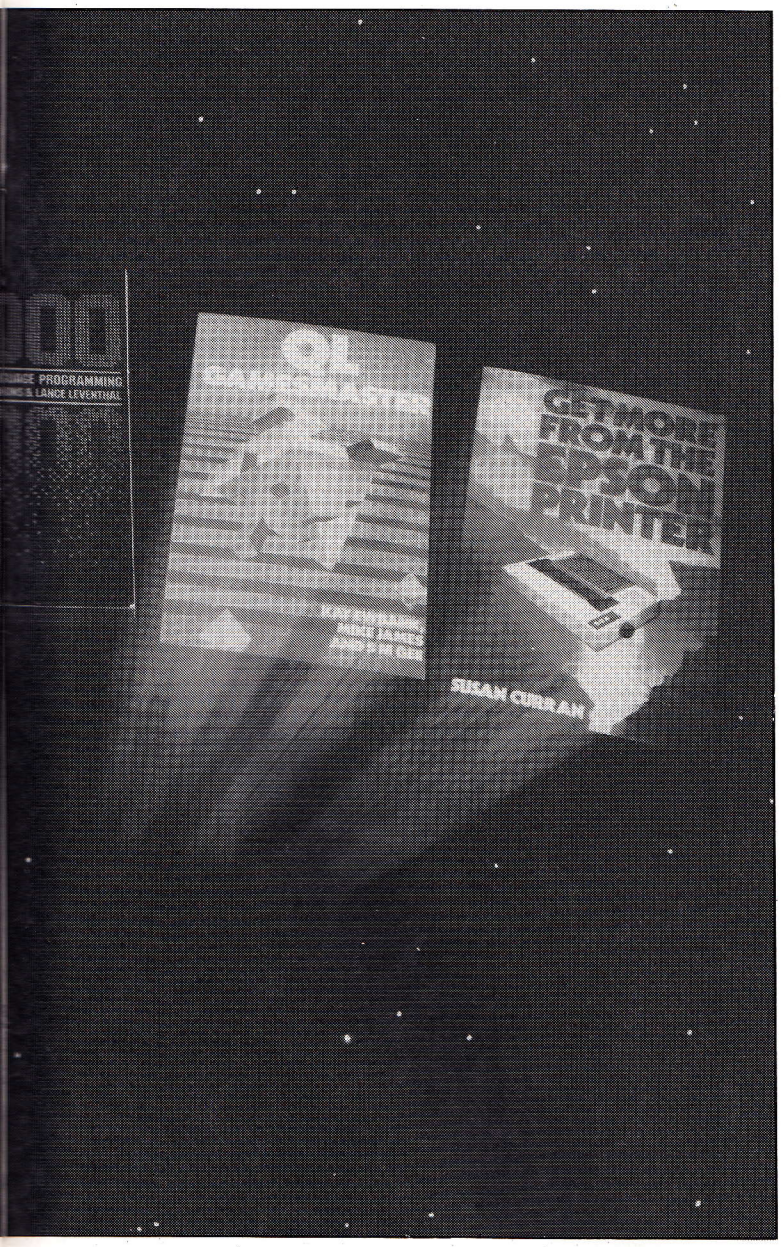
Fair Game

From assembly language to games is quite a leap, but then the QL is all about quantum leaps.

QL Gamesmaster by Kay Ewbank, Mike James and S M Gee, all familiar names, is published by Collins at £7.95. The authors make it clear they wish to be taken seriously. This book, they solemnly state, is not just a collection of games, nor another 'learn to program book'. In fact, while most games books these days try to combine the two, this one manages to place all its emphasis on the 'learn to program' aspect; the games are almost by the way.

There are six games altogether: Ant Hill, Leap Frog, Frogling, Snake (yes, *that* snake), Tadpole, and Snakes and Ladders. Each chapter looks at the game from the point of view of the programmer rather than the player; the techniques are analysed and you are shown how to write your own procedures to perform specific tasks. The book ends with a brief look at 'becoming a master programmer', offering rather repetitive advice which can be found in practically any programming book these days.

If you're looking for adventure, *Sinclair QL*



Adventures could be for you. Written by Tony Bridge and Richard Williams, priced at a reasonable £5.95, this provides you with your very own adventure generator, no less.

The book starts by reviewing the adventure scene, past and present, an exercise that would only be of interest to real dyed-in-the wool adventure hacks. This does, however, include a look at the adventure games available on the machine.

Most of the book is taken up by the adventure generator, QLAG, which provides a framework for an adventure program, and by QLAD, a database program to be used in conjunction with QLAG to provide the details of your adventure. Finally, there is a hurried chapter on how to use the QL's graphics to create monsters, or whatever, to enliven your program.

Well, it certainly beats Snake.

Light Complement

Understanding the 68000 by Leon Heller complements the more heavyweight McGraw-Hill reference book. It's light, not to say skimpy, and feels expensive at £7.95, but fills in some of the holes – there is a great deal of practical information on 68000 hardware, for example, and a useful little chapter on the differences between the parent chip and the QL's 68008.

This is aimed squarely at hobbyists and home users who want to find out more about the mysterious inner workings of their machine. But as the author admits, the book is only an introduction, albeit a practical and readable one, to a highly complex subject, so don't expect too much. To make any sense of it at all, you will need some knowledge of a high-level programming language such as BASIC, although no familiarity with assembly is assumed.

All the chapters are brief. There's a quick look at the 68000 family of processors, chapters on 68000 architecture, signals and bus operation, and a simple 'project' to build – a 'simple 68000 system'. This, we are assured, will take little more than a weekend to build, but be warned – it is assumed you know how to build an electronic circuit.

There's a quick dip into assembly language programming itself, wherein the book explains how to enter a simple program into the text editor to create a source code file, and how to use the assembler to translate the source code into an executable object code file. It is made clear at this point that the book, while quite probably cashing in on the fact that a 68000-derived chip has turned up in the QL, is not aimed at QL users in particular – this chapter assumes you are using a CP/M-based system with a text editor such as Wordstar.

The section dealing with the instruction set is a rather rough and ready introduction to the vast number of instructions available. But the author is quite honest about this, saying he refers only to a subset of the instruction set – that is, he uses just three of the 14 addressing modes.

All this newly acquired knowledge can be put to use by developing a 'monitor', a program that allows you to examine and modify the contents of selected memory locations and execute programs residing in memory.

Finally, there is a summary of the 68000 instruction set and the necessary appendices, but for an exhaustive reference guide, you'll have to look elsewhere.

Print Primer

Everyone needs a printer, and a lot of people have opted for the high-profile, Miss World-promoted, cheap Epson dot matrix range. If you have invested in an Epson, or are thinking of doing so, you may be surprised to find how much more it can do than just print.

Susan Curran spends a lot of time explaining exactly what it can do in *Get More from the Epson Printer*, published at £7.95 by Collins. The sheer size of the book with its 160-plus pages may indeed frighten you off altogether, but stick with it. There's an awful lot of information here which is at worst useful-to-know-in-case, and at best indespensable.

The book starts by introducing the Epson family, which version is best for which tasks, how much they cost (in September 1984) and so on. There's a chapter on starting to use your printer – and if you thought it was simply a matter of plugging in and switching on, you thought wrong – and another, vital, one on interfacing the printer to your computer. And that's not necessarily as easy as it sounds either.

Other chapters cover controlling the printer, using special typestyles, initializing routines and printer drivers, character sets and user-defined characters, bit-image graphics and screen dumps, additional utilities, printer buffers, stationery . . . surely everything you could ever possibly want to know.

Illustrations include sample printouts, and lots of explanatory programs (Epsons are programmable) some at least of which are in QL SuperBasic.

It's a daring project – I haven't come across many books devoted solely to one printer – and I'm not sure that computer users are that interested in the machinations of this particular piece of equipment. But if you find your printer manual less than satisfactory, you could do a lot worse than keep this on a handy shelf.

APPEAL

QL User is currently in possession of a Brother EP44 electronic typewriter, donated by Zeal marketing to be auctioned in aid of the Ethiopian Appeal.

The EP44 retails for around £220, but we will accept the highest bid (over £120) phoned into these offices between 9.45am and 10.00am on Monday 25th March.

All callers should ask for extension 2463, leave their name, number and bid.

At exactly 10.00am we'll stop taking calls and the best offer received will secure the typewriter in exchange for a cheque made out to the Famine in Africa Appeal.

PSION 2.00

Psion's new bundled software – a change for the better?
Dr Andy Carmichael assesses the improvements.

The QL was launched as a package including four applications programs designed to bring professional computing within reach of the masses. The enthusiasm surrounding the launch gave way to bitter disappointment when, after a considerable delay in the machine's delivery, anxious users discovered bugs in the operating system, problems with the microdrives and shortcomings in the the software. Over the last few months, however, circumstances have changed. For their part, Sinclair have ironed out QDOS and improved microdrive reliability. Psion, on the other hand, have set about streamlining their software to overcome problems associated with a shortage of memory and a snail-like performance. The result of all this hard work is to be seen in version 2.00 of all four packages and from a rapid test drive, it would seem they are very much closer to the quality profesional product we were all expecting back in the time before 28 days expired!

Written in the QL's native machine code as opposed to cross-compiled, the programs are considerably more efficient. They load in half the time and run faster. Additionally, they use up less memory time which is all the more surprising when you consider that commands are no longer overlaid from microdrive but are resident in memory and instantly accessible.

'Written in the QL's native machine code, the programs are much more efficient'

Also included with the new set of packages are facilities for tailoring them to your particular system, (eg using floppy disks instead of microdrives, or a different printer interface) and for redefining the sort order of the character set (for Abacus and Archive). The documentation too is spruced up with errors corrected and layout and wording improved.

So on to the test drive itself and first into the slot goes Quill 2.00, the word processor. The first thing that strikes you is the loading time or rather lack of it. No need for a coffee break as each package now loads in under 20 seconds (see table) – remarkable when one considers this only gives time for tape in the cartridge to loop round 3 times.

On loading, Quill 2.00 moves directly from a single copyright screen to the word processing display, secondary screens or messages asking you to put a data cartridge in microdrive 2 no longer appear. In fact, version 2.00 runs quite happily minus data cartridge until you reach 1000–1500 words, at which point you get the following message:

File too big for memory
Transferring to microdrive
*Ensure there is a formatted cartridge in mdv2_*
Press SPACE to continue
Press ESC to remain as a memory file

Error-type messages like this used to portend disasterous situations such as corrupted files or lost documents. Fortunately, this is no longer the case, hitting SPACE sends the microdrive whirring while it creates a temporary file (called *def_tmp* rather than *def_doc*) and once this is done you can continue typing. With large document files a similar message will display when the microdrive is close to full and pressing SPACE will return you to the command display free to save the document. With earlier versions you often had little choice other than to reset your QL and lose many hours work.

The benefit from having all commands resident in memory is soon apparent. Previously it had always seemed self-defeating that commands geared to speeding up operations, for example **Goto,** would themselves take up valuable time being loaded in from microdrive and overlaid. Now with the new version key presses such as <F3 G T> and <F3 G B> zip you from one end of a long document to the other in a matter of seconds. (Try scrolling from one end to the other using the cursor keys and you're likely to be there for some time, even on version 2.00!)

The **Copy** and **Erase** commands have also been improved. These commands were extremely unreliable in version one and could only be used after a document had been backed up for safety's sake. Their worst feature (apart from a tendency to mix up lines on the display!) was the fact that when highlighting the block of text to be copied or erased, the cursor could only move forwards. If you went too far you had little choice other than starting all over again. This has now been remedied and the commands, whilst they remain slow to implement, are at least bug free. Finally, despite improvements, command structure and key presses remain generally the same so that those acquainted with the old versions will have no difficulty changing over to the new.

'The benefit from having all commands resident in memory is soon apparent'

Sadly one bug was discovered hidden away in the system. If Quill cannot open the printer driver file for any reason (we had removed the Quill cartridge from its drive) an error message is given and the program stalls until QL is reset. Moral of the story – always save the file before printing.

Apart from this one blemish, the overall feel of the package is now more professional. The improvements made to Quill bring it up to the business user's level rather than that of a home computer dilettante. In the opinion of our reviewer, this alone makes the QL worth its £400 price tag.

Four auxillary BASIC programs are supplied with Psion's packages. Firstly *INSTALL_BAS*,also available with version one, installs the printer driver with details for bold printing and underlining and options such as continuous or single sheet stationery, etc. There are now 8 makes of printer (previously 2) for which predefined settings are available and an option to choose a parallel or non-standard port rather than the RS232.

Next we have *CONVERT_BAS* and *SCRCON_BAS* for converting version one printer drivers and Archive screens respectively to a format which is compatible with version 2.00.

Finally *CONFIG_BAS* is the program which configures the packages to your system, both in terms of hardware (eg using other devices as default options) and the sorting order of characters. It is this latter facility which adds a new dimension to Aba-

cus and Archive operations.

Why might you want to change the sort order? One obvious reason is to ensure that names will list in alphabetical order irrespective of whether they are printed in upper or lower case. Another reason might be that you wish to use the foreign character set with its accents and double characters. This character set can be obtained by pressing certain keys with CTRL or CTRL and SHIFT together, and is now available for use by Archive, Abacus, Easel and Quill. To achieve the correct ordering of the characters for sorting you have to load and run *CONFIG_BAS* (from the Abacus cartridge) and put all lower case and accented letters next to the upper case letter instead of in their ASCII standard position. It takes time to do, but then you should only have to do it once and for some applications it could prove important.

Lack of memory was the main criticism lodged against Archive. To a lesser degree the same was true of Abacus where business users found the 1170 cells available on the spreadsheet insufficient for complex analyses. In the case of Archive, the lack of memory does not affect the amount of data a single record can store or the number of fields allowed, but it does affect the total number of records that can be accessed. Memory is also required for procedures and user-defined screen formats so if these are used the number of records accommodated is further reduced. The reason for the limitation is that Archive keeps in memory an index of the position of each record. If the database file is unsorted each record requires six bytes of memory. A further 8 bytes per record are required for each sort field (up to a maximum of 4 fields).

'The improvements made to Quill bring it up to the business user's level'

[redacted]

Version 2.00 of Archive gives us another 8K of memory to play with and running the following simple procedure, which adds records to a file one at a time, shows just how many extra records can be fitted in.

```
proc fill
  while 1
    append
    print count ( ), memory ( )
    endwhile
  endproc
```

A file with four sort fields runs out of memory on version one at 303 records, whereas version 2.00 allows 511 records. The number of records possible under the two versions are in proportion to these figures for files with fewer sort fields. If only one field is used for sorting, 1391 records can be accomodated in version 2.00 and for an unsorted file, the total is about 3000 records. The extra available memory will come in handy in many database applications, but if you need more records than this then the only options are to split data into totally separate files or to buy memory expansion.

Filling up memory by using the example procedure above means that Archive's capacity to recover from running out of space can also be tested. Under version one this was almost always fatal and resulted in the corruption of any data files that happened to be open – very frustrating! Happily, version 2.00 does give you the chance to close data files after hitting the end of memory, but our recommendation is to do this at once as a number of other functions (eg edit) do not appear to work properly after this and in our case attempts to delete a number of records from the data file locked the system.

One other bug or undocumented 'feature' concerns the use of multiple data files. The 'current' file is defined as the last file to be opened and in earlier versions this could only be changed by the 'use' command. In version 2.00 however, the current file is changed by any command which references another logical file name. If you have procedures which rely on this feature of version one, they'll have to be modified to work correctly under version 2.00.

There are a number of other minor differences in the workings of some of the commands. For example pressing <ENTER> after the last field when using INSERT or ALTER adds the record to the file, instead of taking you back to the first field and waiting until F5 is pressed. A sensible change this, and one which is documented.

Overall, the most noticeable and important change to Archive and Abacus is the addition of another 8K of memory which in the case of Archive is much needed. The increase in speed of commands is not so noticeable as with Quill but is achieved to some extent.

Easel too, in its new form has more available memory though like Quill several commands, particularly those where a set of options displayed on screen, are now resident in memory once it's all loaded.

Additionally, there are two new printer drivers (JX80 and INTGX) supplied with the upgrade for screen dumps of graphs and figures. It is also possible to choose a different printer driver to the current default while using the print command. This is of limited use in Easel where the printer characteristics change only rarely. However, this would have been welcome in Quill where it is quite inconvenient to install a different printer driver just for a change in print wheels, or cut sheets instead of continuous forms. Perhaps we'll have to wait for version 3.00?

'Version 2.00, unlike its predecessor, is worthy of the title professional software'

[redacted]

A number of the graph formats have changed slightly in Easel as have some of the commands. For example, using line format number 3 the lines used to obscure each other when they overlapped, this is no longer the case. Also some of the command functions have been speeded up though their form remains much as before (ie, the redraw facility). The speed certainly has improved in Easel, for example when it has to redraw the figures.

The four packages supplied with the QL have always been one of the most potent factors in favour of its purchase. With this upgrade Psion have put right all the major bugs which marred the early versions, speeded up and compressed the code to improve performance significantly, and improved the already substantial documentation that accompanies them. Version 2.00, unlike its predecessor, is worthy of the title professional software. What a shame it wasn't this version that arrived with the machines this time last year!

The updated versions are available with new QLs after 4th March or can be bought for £15 each (£50 for all four).

Loading Times (secs)

Version	Quill	Archive	Abacus	Easel
V 1.01	41	39	28	41
V 2.00	19	18	18	19

Available Memory

V 1.01	—	12K	15K	8K
V 2.00	—	20K	23K	12K

CLASSIFIED

INSTA

This new regular feature is designed to provide an instant reference chart for QL information.

Each month the listing will be updated as new products and info are announced.

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

HARDWARE INDEX

Monitors

Citadel Products Ltd	01 951 1848
MBS Data Efficiency	0442 60155
Kaga	
Microvitec PLC	0274 390011
Microworld Computer & Video Centre	0293 545630/
Microvitec, Philips, Sinclair Vision	0273 671863
Opus Supplies Ltd	01 701 8668
JVC	
Printerland	0484 514105/687875
Strong Computer Systems	0267 231246
Microvitec, Philips	
Technomatic Ltd	01 208 1177
Microvitec, Kaga	
Zeal Marketing Ltd	0246 208555
Microvitec, Philips	

Printers

Datasystems	01 482 1711
Star	
MicroPeripherals	0256 473232
Canon	
Microworld	0293 545630/0273 671863
Kaga, Epson, Smith Corona, Shinwa, MicroPeripheral, Quen-data, Juki	
Printerland	0484 514105/687875
Epson, Brother, Kaga, Canon, Juki	
Strong Computer Systems	0267 231246
Brother, Shinwa, Epson, Kaga, Mannesman Tally, Canon, Daisystep, Smith Corona	
Technomatic Ltd	01 208 1177
Epson, Kaga, Juki, Brother	
Twickenham Computer Centre	01 891 4991
Kaga, Ensign, Canon	
Zeal Marketing Ltd	0246 208555
Brother, Epson, Canon, Daisystep	

Interfaces

Cambridge Systems Technology	0223 323302
Care Electronics	0923 777155
Computamate Data Products	0782 811711
Miracle Systems Ltd	0272 603871
Printerland	0484 514105/687875
Sigma Research	231 Coldhams Lane, Cambridge
Technology Research Ltd	0784 63547
Transform Ltd	089 283 4783
Zeal Marketing Ltd	0246 208555

Disk Systems

Computamate Data Products	0782 811711
Compware	0270 582301
CST	0223 323302
Medic Data Systems Ltd	0256 52703
MicroPeripherals	0256 473232
Printerland	0484 514105/687875
Quest	04215 66488
Strong Computer Systems	0267 231246
Zeal Marketing Ltd	0246 208555

NT ACCESS

Modems

Compak Data	13 Beechwood Road, Uplands, Swansea
Medic Data Systems Ltd	0256 52703
OEL	0768 66748
QMOD	
Strong Computer Systems	0267 231246

Memory Expansion

Eprom Services	0532 667183
Medic Data Systems Ltd	0256 52703
Quest	04215 66488
Simplex Data Ltd	01 575 7531

Extras

Action Computer Supplies	01 903 3921
Mains spike eliminator	
Eidersoft	01 478 1291
Quicksoft 11 Joystick	
Management Science Ltd	17 West Hill, London
QL Case	
Sigma Research	231 Coldhams Lane, Cambridge
Joystick Adaptor	
Sinclair Research	0276 686100
Power Supplies	
Transform Ltd	089 283 4783
QL Dust Cover, Microdrive Storage Box, RS232 lead	
Zeal Marketing Ltd	0246 208555
Printer peripherals	

SOFTWARE INDEX

Games

Eidersoft	01 478 1291
QL Art	
Games Workshop	01 965 3713
D-Day	
Printerland	0484 513105/687875
Psion Chess	
Psion	01 723 9408
Psion Chess	
Sinclair Research	0276 686100
Psion Chess	
Talent Computer Systems	041 552 2128
The Lost Kingdom of ZKUL, WEST, GraphiQL	

Utility Programs

Accountancy Software of Torquay	Sinclair Research
QL Cash Trader	
Computer One	0223 86216
Pascal, Forth, Assembler, Typing Tutor, Monitor	
GST Computer Systems	0267 231246
QL Assembler, Q Jump, QL Toolkit	Sinclair Research
Harcourt	Sinclair Research
QL Touch 'n' go	
Metacomo	0272 81392
Assembler, BCPL, Lisp	
MicroAPL	01 622 0395
Printerland	0484 514105/687875
Metacomo Assembler	
Psion	01 723 9408/0553
Quill, Abacus, Easel, Archive	
Quest	04215 66488
Business Accounts	
Strong Computer Systems	0267 231246
TR Computer Systems	093 924 621
QL Payroll	

BOOKS

Publishers

Adder	0223 277050
Century	01 434 4241
Collins	01 940 7070
Granada	01 493 7070
Hutchinson	01 387 2811
Interface	9-11 Kensington High Street, London
McGraw Hill	0628 23431
MicroPress	0892 39606
Melbourne House	01 940 6064
Prentice/Hall	0442 58531
Sunshine	01 437 4343

March '85

A QL Compendium	Gandoff & Kinge
(Addison-Wesley)	
QL Handbook	Hartnell
(Interface)	
Professional & Business Uses of the QL	Lewis
(Collins)	
QL Abacus	Spottiswoode
(Century)	
BASIC Programming on the QL	Cryer & Cryer
(Prentice/Hall)	
Quick QL Machine Language	Giles
(Melbourne House)	

February '85

The Working Sinclair QL	Lawrence
(Sunshine)	
Quantum Theory	San, Katan & Rockman
(Century)	
QL SuperBasic	Wilson
(MicroPress)	
QL Assembly Language Programming	Opie
(McGraw-Hill)	
QL Games Compendium	Hartnell
(Interface)	
The QL book of Games	Hurley & Virgo
(MicroPress)	

Dec/Jan '84

Introduction to SuperBasic on the Sinclair QL	Meadows
(Hutchinson)	
Advanced Programming with the Sinclair QL	Gandoff
(Hutchinson)	
QL Quill	Simon & Spottiswoode
(Century)	
QL Easel	Spottiswoode
(Century)	
QL SuperBasic	Berk
(Granada)	
Assembly Language Programming on the Sinclair QL	Pennell
(Sunshine)	
Developing Applications on the Sinclair QL	Grace
(Sunshine)	
Artificial Intelligence on the Sinclair QL	Brain & Brain
(Sunshine)	

Oct/Nov '84

Desk-Top Computing with the Sinclair QL	Miles
(Hutchinson)	
Introducing the Sinclair QL	Marshall
(Hutchinson)	
Word Processing with the Sinclair QL	O'Reilly
(Hutchinson)	

QL Owners! Join the UCSD Pascal Club

Now you can penetrate the expanding world of the QL while writing applications in the language programmers love:

UCSD Pascal. The original. The fully developed industry standard. Upwards compatible with Apple Pascal, and totally compatible with UCSD Pascal running on computers like the IBM PC, Apricot, Apple Macintosh, TDI Pinnacle, Sage and many many others. Complete with extensions for systems development and commercial applications programming.

UCSD Pascal will run on the standard QL with microdrives, however we do recommend floppy disk drives or plug-in RAM cards.

UCSD P-SYSTEM is fast. You can compile 700 lines of Pascal per minute and achieve a PCW benchmark average of 9.01 with this professional operating system, running on your QL with Sinclair RAM card. And that puts it into 6th place, way above the IBM PC and close to the speed of the IBM AT!

It's made possible by TDI Software Ltd, distributors of the microcomputer development environment praised by novice and professional programmers alike.

Start with the basics: UCSD Pascal or UCSD Fortran-77, a complete set of development utilities, a full-screen full-function editor and a file manager. At the incredibly low entry price of £99.95, inclusive of VAT and three superb manuals: Pascal, Assembler and the Operating System.

Add the second language. Our compilers generate compact code and are truely integrated with the development environment. Edit with ease. When an error is detected, the editor may be invoked and automatically position at the statement in error. You can also write in smaller program segments. This simplifies debugging and testing and lets you use common procedures in multiple applications. Even if they were written in different languages.

For professional developers we also offer a full set of productivity tools such as a symbolic debugger, native code generator, and program analysis tools. This Advanced Development Toolkit comes complete with three useful manuals: Applications Development, Program Development and Internal Architecture Guide, for only £49.95 (inclusive of VAT).

Clip and mail the coupon below. Or call the TDI Sales team on 0272-742796.

You'll find that UCSD Pascal goes QL. And a lot of other places besides.

Please send me (tick as apropriate)
UCSD Pascal @ £99.95, plus £5 p&p, (total £104.95) ☐
UCSD Fortran-77 @ £99.95, plus £5 p&p, (total £104.95) ☐
Advanced Development Toolkit @ £49.95, plus £5 p&p, (total £54.95) ☐
☐ My cheque/postal order for £..... is enclosed made payable to TDI Software Ltd.
☐ Please debit my Barclaycard/Access/Diners Club with £.....
(Delete as necessary).

Card No. | | | | | | | | | | | | | | | | |

Signature .....

Name .....

Title .....

Company .....

Address .....

.....

Post Code .....

Telephone ( )

TDI Software Limited
29 Alma Vale Road, Bristol BS8 2HL. Tel: (0272) 742796

TDI

TDI Software Limited
29 Alma Vale Road
Bristol BS8 2HL.
Tel. (0272) 742796.

*UCSD Pascal is a registered trademark of The Regents of the University of California.

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 _____

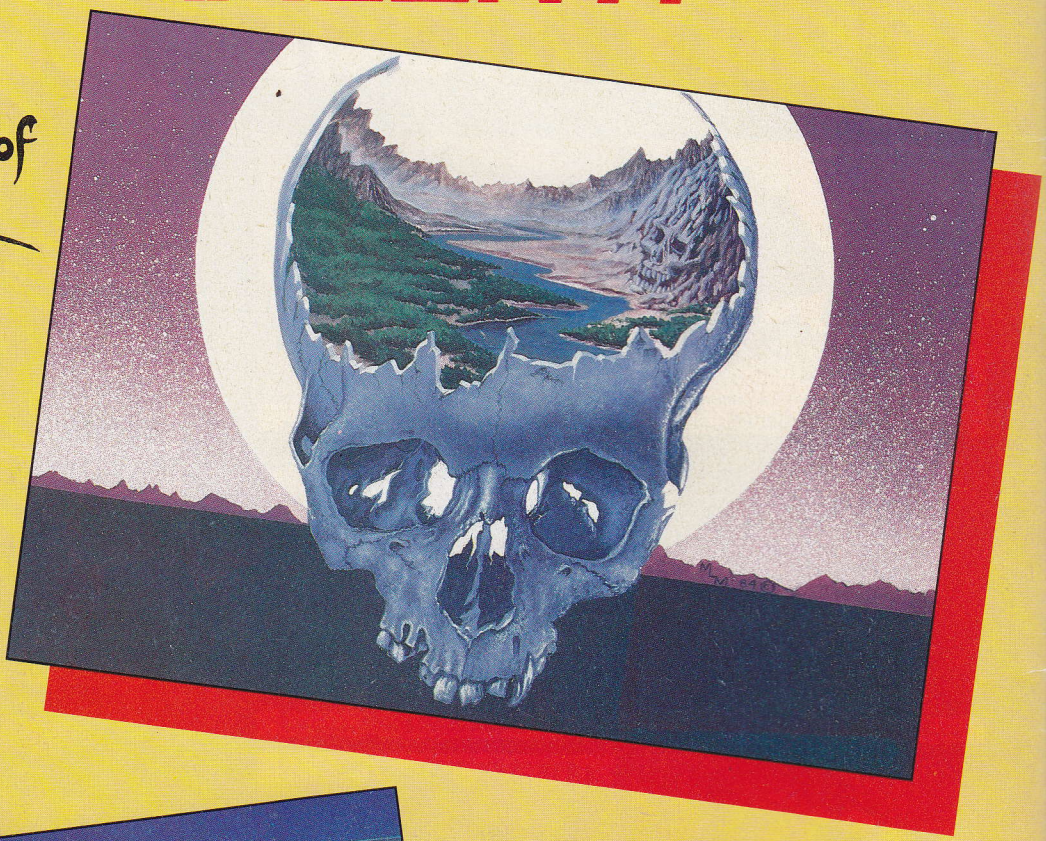
Phone today, or post this coupon to: Metacomco, 26 Portland Square, Bristol BS2 8RZ.
Please send me:
The Assembler Development Kit at £39.95 _____
The BCPL Development Kit at £59.95 _____
The LISP Development Kit at £59.95 _____
I enclose a cheque for £ _____ or debit my ACCESS/VISA Account No.

Card expiry date: _____

Price includes VAT, postage and packing UK mainland only. Delivery allow up to 28 days. QL5

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

Available from selected branches of:
John Menzies
WHSMITH