QDOS manual

Way back in January 2984, when the Sinclair QL was launched, there was much said about 'multi-tasking' and the amazing QL operating system — QDOS. Now, seven months after ordering the QL and two months after receiving the machine and its 'User Guide', I have taken delivery of the 'QDOS Manual' at a cost of £13, including postage and packing.

The document Sinclair Research calls the 'QDOS Manual' is, in fact, the missing pages from the QL Users' Guide detailing such mysteries as 'multi-tasking', system variables and QDOS system calls which the advanced programmer would find invaluable.

I ended up with my QDOS Manual by writing polite letters to Sinclair Research asking for the technical data "... I felt I had a right to be supplied with". This resulted in my getting a letter from the Customer Relations Department suggesting that I forward a cheque for £13. Try it — it might work for you. It arrived within a few days — honest!

Once you get your hands on one of the elusive documents, written by Tony Tebby, you will find it has some 150 pages and is written in a highly technical manner.

The first thing I noticed was a secret revealed which has been kept closely guarded by Sinclair: The QL was initially intended as a 32K/128K RAM configuration with an option to upgrade, similar to the Spectrum 16K/48K arrangement.

A brief introduction and overview of QDOS starts the manual, closely followed by details of changes to previous versions. My manual documents QDOS Version 1.03 — the earlier ones run through the release numbers 0.05, 0.06, 0.07, 0.08, 1.00, 10.01 and 1.02. In fct, you can find out which version of QDOS you have by looking through the Rom for a string in the format *n.nn.* My 'AH' version QL has version 1.02 QDOS.

The first half or so of the manual systematically describes each system call to QDOS (via the MC68008 *Trap* instructions) detailing input parameters, output parameters, registers affected and possible error return situations. The subjects covered by these *Traps* are: job creation and deletion, job management, resident procedure control, display handling, Intel i8049 communication, serial port control, real-time clock control, memory allocation and file handling.

Other chapters are devoted to such subjects as device drivers, Rom drivers, interrupts, arithmetic handling, QL Microdrive format, interfacing to QL SuperBASIC, system variables and example assembly programs.

Details on QL Peripheral expansion are not available until "after the launch of the various devices".

As I said earlier, the manual is not exactly written with the complete beginner in mind — more the advanced proramer, or hobbyist.

With this said, I have to say the QDOS Manual is everything I wished. Let us not get romantic, though. It is quite obvious this extra information should have been supplied in the first place with the QL — not as an optional extra several months later. I hear that 'multi-tasking' is not implemented on QL versions FB, PM and AH. We will all have to wait for Sinclair to plug in the precious JM version, "within ten working days".

Then, your QDOS Manual will become a QL Bible.

Alan Turnbull
Gale Green
Stockport
Cheshire

A new bug

Ian Logan wants to hear about 'new' Spectrum bugs huh? OK, here's one.

```
1 DEF FN e(n,m)=n+m: DEF FN
f()=FN e(29,FN e(17,8)): REM num-
bers chosen especially for
HHGTTG fans.
PRINT FN f()
42
```

is the wrong answer. *FN e(n,m)* just calculates the sum of two numbers, FN *f*() should give the sum of 29 plus the sum of 17 and 8 (29+17+8=54), but it doesn't. In fact, the computer has worked out 17+17+8=42.

This is because, although the Spectrum uses a stack to keep track of the order of functions (ie, it must perform *Fn e*(17,8) before *Fn e*(29 . . .) it holds the numbers it works with, its operands, in fixed locations. This means that the 29 and the 17 will occupy exactly the same areas of memory because they are both the first operand of an *Fn e*. Since the 17 is interpreted after the 29, it just replaces it before the computer has had a chance to do anything with the 29.

So the computer ends up evaluating *Fn e* (17,17+8).

Julian Skidmore
25 Cossall Road
Trowell
Notts

Evil and sinister

Firstly, congratulations on the improvements to *Popular Computing Weekly*. It was always the leading magazine, but is now even better (perhaps with the exception of the unamusing advertisement from Automata).

Anyway I digress from the main point, which is to point out my extreme displeasure at receiving inside PCW this week a leaflet advertising a Dianetics publication. You cannot fail to have read a High Court Judge's recent comments about the Scientologists — the group behind Dianetics — whose activities parallel those of the Unification 'Church' (Moonies).

The Judge described Scientology as "Evil" and "Sinister", and by allowing Scientology a place in your publication, you may have unwittingly lined up some reader or readers for a particularly nasty experience.

H Petfield
6 Cranley Gardens
Wallington
Surrey

The magazine will not be taking any more Scientology ads.

Vic20 modulator

It was interesting to read Phil Roger's reply to P Whalley in *Peek & Poke* in the July 19 issue on the question of faulty Vic20 modulators.

A 'faulty' Vic modulator is often caused by users and not by the Black Box itself. Many people (quite possibly Mr Whalley not included) remove the modulator from the Vic by tugging at the cable between the modulator and the machine, and not by the recommended method of holding the actual connecting plug. Constant tugging wrenches the leads inside the connector out of position causing loss of sound, or picture, or both.

M J Davies
The Waverley
79 Rhosmaen Street
Llandeilo
Dyfed

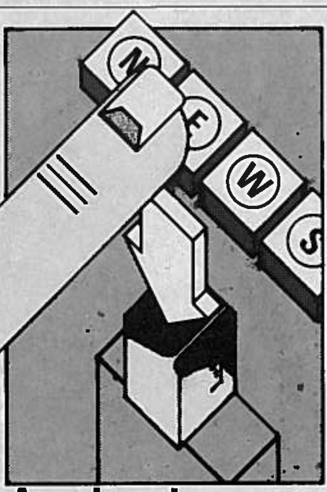
Stop bickering!

I would just like to point out to Mr Bathurst (August 9 issue), that, while the Commodore 64 has only 39K for Basic programming, it has another 19K which can be used either for machine code, or for storing graphic or sound detail.

I also object to letters like that of Mr Haighley (same issue), which, while it has a point to make — that magazines do not print enough for the Electron (I agree, but can't the Electron use BBC software?), the writer also decides to slip in a quick slagging-off of the Commodore 64.

Well, I own a TZQPA 3200042K computer with built-in speech synthesiser, instant retrieval disc drive, TV-resolution graphics, and it's better than all the other computers put together, so stop bickering!

M Valentine
101A Underdale Road
Shrewsbury



Alex Wilkie

NEWS • NEWS •

A run-down of new products, people and events in the QL market by Sid Smith of Micronet.

A resigned attitude . . .

One of the three people who created the QL resigned in disgust the day it was launched.

The big three were Jan Jones (who wrote the Basic compiler), David Karlin (responsible for the hardware) and Tony Tebby – author of QDOS. It was he who, appalled by Sinclair's promise that the QL would be shipped within six weeks handed in his notice immediately after the machine's January unveiling.

In the event Tony hung on a little longer, eventually fixing a date to leave after Sinclair's management decided to ship those first 89 machines without, he says, telling the software division.

He still takes a lively interest in the QL – so lively that many of his opinions are unprintable! It's worth recording however his belief that the delay was exclusively due to problems with the machinery itself, the apparent difficulties with SuperBasic and QDOS being caused by the firmware writers having to conform with frequent hardware updates. Status conscious QL users, in fact, might care to enliven their one-upmanship over operating systems with a little snobbery regarding the hardware issue of their machines: check the underside of your QL for the two digits following the green D; 07 is good, anything higher than 09 is excellent. Particularly interesting meanwhile are Tony's opinions on the shape the QL's promised new operating system – expected next year – ought to take.

'I'm fairly confident that the new version will include improvements to the Basic editor, but the cost in memory of any full screen editor would be disproportionate, especially when you consider what else you could have for the same amount of code,' he says.

Preferable alternative goodies, feels Tony, would include direct access file handling in Basic, the ability to read or write data to anywhere in a microdrive file; formatted print using, parameters on a print command to describe in detail the format of output; a full set of job control facilities for examining what multi-tasking programs are running, and for killing them, suspending them, changing their priorities, etc; and a spooler, for outputting files to a printer while the user gets on with other tasks.

'All that lot would take about half as much ROM space as the 2K you'd need for a full screen editor,' says Tony, though he's dubious about the likelihood of Sinclair coming to sensible decisions about what to offer in the new ROMs.

'They'll probably be more impressed by marketing appeal,' he remarks, 'than by what the facilities actually do, which is unfortunate because marketing appeal only sells the first few machines, it doesn't sell the next million.'

So what'll happen to people with the old operating system, I asked innocently. Will they get a free upgrade?

'There is no old operating system and there never has been. The operating system which is out now is the same as the operating system which was out in January. Anything else is just a story invented by journalists looking for something to write about.'

Gulp.

'But in any case, QDOS contains linkages to facilitate expansion. All the things I've mentioned can be supplied externally – on microdrive or ROM cartridge, for example – to be booted up on power on, and will look just as if they form part of the machine. They could be built into the new ROM, but they certainly don't need to be.'

All smiles from Psion and Potter.

Slowness in the eye of the user

Quill is slow. I know because Psion Managing Director David Potter told me. He even said why.

But first the good news: 'In September we'll be releasing a Quill Version 2 which will overcome all the problems of Version 1 thanks to an intensive effort to specially code-down, and using special techniques to compress the software. We've also put special additional features in the screen driver, writing our own outside of QDOS to speed writing to the screen.'

Phew! But why are all these hairy techniques necessary?

David explained: 'Quill is actually designed as a very powerful word processor to run very large documents. It was designed for substantial machines. You could criticise it in the sense that you're trying to pack a hell of a lot into a modest machine – the QL.

'The effects of this are, firstly that the amount of data memory left – because the program is very substantial – is modest.

'Secondly, we've had to resort to overlaying, which was one of the points made in *QL User*, – when you press for Command the microdrive whirrs to get another overlay in. When the code compression is finished, all the commands will be resident in RAM at all times, and additional memory will be available to the buffer space so that Microdrive 2 will be less used.

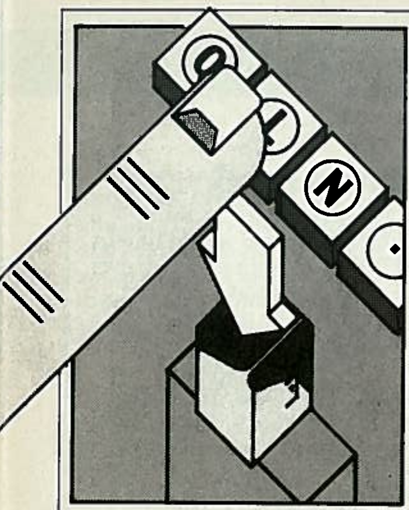
'The third factor is that there is no video chip in the QL, and therefore everything on the screen has to be done by the processor. That is a hardware design feature of the QL which frankly is pretty tricky.

'And the final factor is the microdrives.'

Now none of these observations about Quill will cause particular surprise to readers of this magazine, who will remember similar remarks in the somewhat critical review in our last issue. What is surprising is that they were made when Mr Potter phoned to complain about said review.

NEWS

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



Massive Cuts

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

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

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

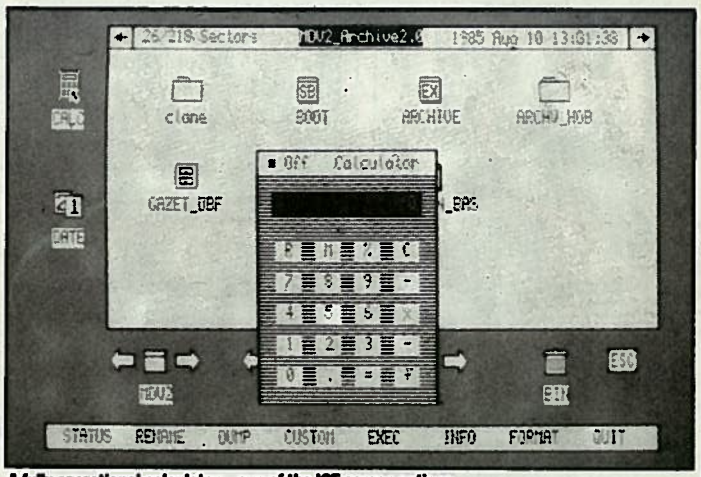
The same cannot be said of games software houses. Whilst the QL is well supported in terms of business and development software, there are few games of note running on it. Despite the obvious signs that 8-bit processors such as the Z80 and 6502 have had their day none of the big names have been prepared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

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

ICEd Over

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

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



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

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

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

MG Extinct

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

Psion Update

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

1. Memory allocation problems have been ironed out so that you may re-use your programs without getting an error message and being compelled to reset the QL.
2. Quill incorporates an option to import a file by line or paragraph.
3. In addition to the current Epson screen dump, Easel will cater for nine other printers and will allow for standard output to Epson and HWP plotters.
4. Archive now include an option to design your own printed forms using SEDIT.
5. A facility to load from and save to the Network device has been implemented.
6. The algorithms controlling use of memory with Quill have been improved so that it is now possible to create 12K documents on Quill before the program overlays it to microdrive.

Further information from Psion – 01-723 9408

RAMbling On

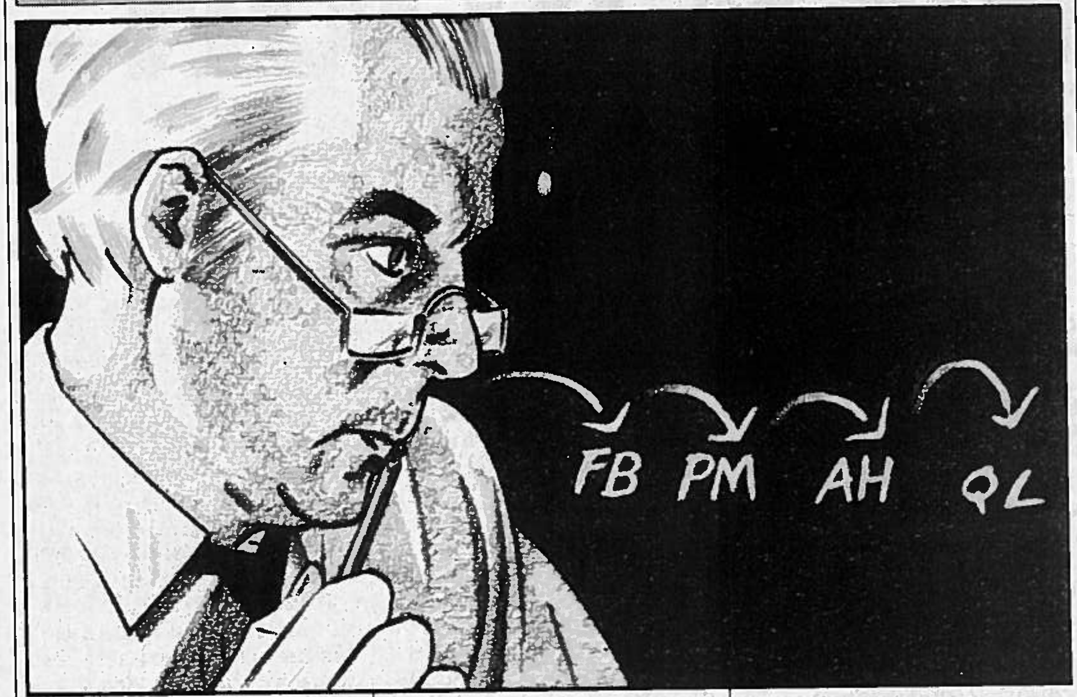
Silicon Express have come up with one of those simple ideas that you wish you'd thought of yourself. Simply prise the lid off your QL, yank out the 16 64K DRAMs installed courtesy of Sinclair and plug in a similar number of the very latest 256K chips.

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

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

ROMBLINGS

A Historical Perspective



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

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

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

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

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

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

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

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

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

Listing 1

```
* SuperBASIC extensions to provide:
* ---------------------------------
*
* 1) A new function QDOS$ to return the QDOS release number
* 2) A command CALL to replace the faulty (pre-'JS') ROM-based version.
*
* COPYRIGHT (c) April 1985, Alan Turnbull, B.Sc.
*
*
               LEA       EXTENSIONS(PC),A1     ; Point to list of extensions
               MOVEA.W   $110,A0               ; Vector for linking-in extensions
               JSR       (A0)                  ; Call routine
               RTS                             ; Return to SuperBASIC
*
* Table of extension definitions
*
EXTENSIONS:    DC.W      1                     ; Number of procedures
               DC.W      CALL-*                ; Offset of start of routine
               DC.B      4                     ; Length of name
               DC.B      'CALL'                ; Name of procedure
               DC.W      0                     ; End of procedures
               DC.W      1                     ; Number of functions
               DC.W      QDOS-*                ; Offset of start of routine
               DC.B      5                     ; Length of name
               DC.B      'QDOS$',0             ; Name of function with padding
               DC.W      0                     ; End of functions
*
* Run-time modules
*
QDOS:          CMPA.L    A3,A5                 ; Were any parameters provided?
               BNE       ERR_BP                ; Yes, report 'Bad parameter' error
               MOVEQ     #6,D1                 ; Reserve 6 bytes on arithmetic stack
               MOVEA.W   $11A,A0               ; Required vector
               JSR       (A0)                  ; Call routine
               MOVE.L    $0058(A6),A1          ; Get arithmetic stack pointer
               SUBQ.W    #6,A1                 ; Make room for string
               MOVE.W    #4,$00(A6,A1.L)       ; Put length of string on stack
               MOVEQ     #0,D0                 ; Get system information
               TRAP      #1                    ; Call routine
               MOVE.L    D2,$02(A6,A1.L)       ; Put QDOS release code on stack
               MOVE.L    A1,$0058(A6)          ; Reset arithmetic stack pointer
               MOVEQ     #1,D4                 ; Signal string result
               RTS                             ; Return to caller: successful
ERR_BP:        MOVEQ     #-15,D0               ; Signal 'Bad parameter'
               RTS                             ; Report error
CALL:          MOVEA.W   $118,A0               ; Vector to get long word integer
               JSR       (A0)                  ; Call routine
               BNE       CALL_RET              ; Report error if unsuccessful
               LSL.L     #2,D3                 ; Get number of long word parameters
               BEQ       ERR_BP                ; Report error if no parameters
               ADD.L     D3,$0058(A6)          ; Reset arithmetic stack
               MOVE.L    $00(A6,A1.L),-(A7)    ; Put start address of routine on stack
               MOVEM.L   $04(A6,A1.L),D1-D7/A0-A5 ; Place parameters in processor registers
               MOVEQ     #-15,D0               ; Preset error flag
CALL_RET:      RTS                             ; Either return with error or jump to CALL
```

Listing 2

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

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

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

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

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

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

CMPI.L #'2.00',D2

is a bit of a giveaway!

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

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

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

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

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

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

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

Listing 3

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

Keyword	'AH' ROM (QDOS v1.02)	'JM' ROM (QDOS v1.03)	'JS' ROM (QDOS v1.10)
PRINT	28586	28662	30376
RUN	30232	30322	32164
STOP	30334	30424	32266
INPUT	28584	28660	30374
WINDOW	30646	30736	32638
BORDER	30684	30774	32676
INK	28364	28440	30154
STRIP	28368	28444	30158
PAPER	28372	28448	30162
BLOCK	30660	30750	32652
PAN	28406	28482	30196
SCROLL	28410	28486	30200
CSIZE	24756	24828	26274
FLASH	26026	26098	27564
UNDER	26020	26092	27558
OVER	26048	26120	27586
CURSOR	24792	24864	26310
AT	24806	24878	26324
SCALE	26100	26172	27638
POINT	26118	26190	27656
LINE	26136	26208	27674
ELLIPSE	26160	26232	27698
CIRCLE	26160	26232	27698
ARC	26240	26312	27778
POINT_R	26122	26194	27660
TURN	30416	30506	32408
TURNTO	30408	30498	32400
PENUP	30474	30564	32466
PENDOWN	30478	30568	32470
MOVE	30492	30582	32484
LIST	28036	28112	29824
OPEN	25926	25998	27464
CLOSE	25892	25964	27430
FORMAT	25714	25786	27252
COPY	25740	25812	27278
COPY_N	25744	25816	27282
DELETE	25570	25642	27110
DIR	25576	25648	27116
EXEC	25246	25318	26764

▶

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

EIDERSOFT MEANS BUSINESS

Eidersoft is pleased to announce a range of hardware and software packages for the professional QL User.

EIDERSOFT DISK SYSTEM
Eidersoft 3.5 inch QL Disk Systems are in a class of their own. Each system comes complete with a package of software that includes **I.C.E.** (see our colour advertising), backup routines, a disk database and much more! What's more the system is fully guaranteed and professional help no more than a phone call away.

£520 **SYSTEM 1** 128K RAM, Ram Disk, Parallel printer port with buffer, I.C.E. + full software, 2 disks, TWIN 3.5 Disk Drives with built in PSU and all leads and accessories.
COLOUR Matching QL Black.

£540 **SYSTEM 2** As above with 256K Extra RAM and many additional toolkit commands.

£630 **SYSTEM 3** As 1 but with 640K RAM *requires return of QL for 7-10 days.
Full inspection of these high powered units available by appointment.

PRICE INCLUDES 24 Hour Delivery

**** MANY SOFTWARE TITLES NOW ON DISK ****
Please telephone

ARCHIVER
Archiver is a collection of business programs for the Psion ARCHIVE™ Database. Archiver programs are open to the user and can be modified to suit individual requirements. Archiver includes programs for **INVOICING, STOCK CONTROL (linked), APPOINTMENTS and MAILING*** as well as many useful Archive routines.
* NOW with **QUILL™ MAIL MERGE!**
Incredible Value @ £18.95 + £1 P&P

QSPELL
QSPELL is the onlyy speling checker availible for Psion QUILL™. It contains a dictionnary of 25000 words and allows you to add 1000 more. The multi-tasking Editor lets you edit your documents in Quill and fullyhighlights mistakes. Fully menu driven with inbuilt help screens.
At only £19.95 + £1 P&P it's at least a ¼ of the price of similar specification checkers".

PROBLEMS WITH POWER SPIKES?
The Power clean Masterplug not only filters the mains, but provides 4 neat fully fused minature sockets to tidy up wiring. It comes complete with a moulded 13 amp plug. (grey) ONLY £24.95 complete.

CREDIT CARDS PHONE 0708 852647 THE OFFICE, HALL FARM, N. OCKENDON, UPMINSTER ESSEX, RM14 3QH

Please supply I.C.E. (£49.95 + £1.50 P&P) / QSPELL (£19.95 + £1 P&P) / ARCHIVER (£18.95 + £1 P&P)
MASTERPLUG POWER CLEAN (£24.95 + £1 P&P) / LABELS KIT (£24.95 + P&P)
DISK SYSTEM 1 (£520 inc.) / DISK SYSTEM 2 (£540 inc.) / DISK SYSTEM 3 (£630 inc.)

Please supply further information on ☐ disk systems ☐ memory expansion ☐ software (state title/s)

NAME .... STREET .... TOWN ....

POSTCODE .... M/D /DISK (State type) .... COUNTY ....

I enclose a cheque/PO made payable to Eidersoft for £ ....

QUILL and ARCHIVE are the Trade Marks of PSION LTD.

Bugging the ROM

Compiler Simon Goodwin summarises the bugs, features and international variations which lurk inside each QL version and explains how you can upgrade your ROM.

Sinclair produced at least seven versions of the QL ROM the collection of built-in routines which look after Basic and machine code programs. Five of the ROM versions are still considered current, yet they vary enormously in functions and reliability. To check the version of your QL, turn it on without a tape in drive 1, press F1 or F2, and type PRINT VER$. A two-or three-letter code will appear at the top of the screen.

This article compares the features or, to be more honest, the bugs in current QL ROMs and explains how all the versions came into existence in the first place. Next month I will list the universal bugs in every QL ROM regardless of vintage and explain how to get around them.

This litany may make the QL seem rather a disaster but that is not really fair. The first QLs were unfinished and bug-ridden as a consequence but later versions are no worse than the average Amiga, ST or PC. All complex systems contain bugs but hardware manufacturers are curiously shy about admitting and correcting. In fact, bugs are rarely a problem if you know about them and can avoid them. This information is based on my experience, the Sinclair bug list and reports from QL users. If you have found any I have missed, please send details.

Team effort

In 1983 the QL ROM was planned as a team effort. The operating system was to be written by the Cambridge programming house GST, while Sinclair staff contributed code to handle devices and the new Basic interpreter. SuperBasic was designed originally in 1982 for the SuperSpectrum, one of the many Sinclair designs which never passed the drawing board stage.

On January 12, 1984, long before the QL hardware and software were finished, Sinclair launched the QL in London. The prototype machines at the launch used the GST 68K/OS operating system but Sinclair never shipped a QL in that form.

The original plan was to squeeze SuperBasic and the QL operating system into 32K of ROM. At first only a fraction of SuperBasic was to be built into the QL – just enough commands to load and run the Psion Packages. Most of the language was to be loaded as required from Microdrive; almost all the standard SuperBasic commands and functions are still implemented as extensions, although they are built-in to current ROMs.

Sinclair abandoned the GST operating system because it was slow and greedy for memory; it did not leave sufficient space for SuperBasic in ROM or the Psion packages in RAM. GST released 68K/OS independently as a 32K plug-in option at the end of 1984.

The first production QLs contained two-thirds of a stop-gap operating system, Qdos, written by Tony Tebby, a Sinclair engineer who had originally been hired to work on satellite TV hardware.

The other one-third of Qdos and SuperBasic was supplied in an ugly plug-in cartridge — 'the kludge'. It had been impossible to squeeze the required code into 32K so Sinclair put an extra 16K outside the computer. All the software was in expensive, individually-programmed EPROM chips, rather than mass-produced ROMs.

The version number of Qdos jumped suddenly from 0.08 to 1.00 when the first production QLs appeared but that was more brave than honest; the code was still being developed in April, 1984 as those machines slipped out. There were amazing bugs. You could not edit a 'bad line'. PRINT -2 - 2 gave zero. Basic programs could not exceed 32K. There were error-trapping keywords but no code to handle them.

Special tokens to allow array initialisation survive from that time but they have never worked; they are left-overs from the initial SuperBasic designs. READ, DATA, GOSUB and RESTORE had been added at a late stage, to make the original, elegant design more standard. The last-minute changes provoked a flood of problems.

Sinclair gave each ROM version a two-letter code. First was Qdos 1.00, the largely-untested "FB" version. The next major version, "PM", was faster and more tolerant of the Microdrives but it was still laced with faults. At that stage new versions, such as "EL" and "TB", were popping up inside Sinclair every week.

In all 13,000 kludged QLs were produced but in June, 1984 the so-called final QL ROM emerged, the "AH" version. By that time Sinclair had stopped naming ROMs after taxi-drivers and started picking on women in the office — "AH" stood for "Angela's Holiday".

The "AH" ROM was really three 16K EPROM chips. The plug-in kludge was avoided by soldering two chips piggy-back in one socket. The chips contained Qdos 1.02, the first usable version of the QL built-in software. It was about 20 percent faster than "FB"; since then, code speed has changed very little. The "AH" ROM and the mass-produced follow-up "JM" were supplied as a free upgrade to those with kludges.

Exceptional bugs

"AH" and "JM" were very similar. Only four bugs were exceptional to "AH". None was serious at the time, although two are worth bearing in mind if you have an expanded QL system.

If two tasks tried to read a file simultaneously, the second would miss the beginning and read the directory header instead. At the time that was fairly academic as there were no multi-tasking programs on the market. Floating point arrays were limited to 384K in size but memory expansion was not available in those days.

The other two bugs fixed between

"AH" and "JM" were trivial X="." Set X to zero under the "AH" ROM, instead of giving an error, and you could type-in integer FOR loops, although they would not work unless you changed the variable name to a floating point type. The correction for this bug was scarcely an improvement; rather than make integer loops work properly, "JM" and later ROMs will not let you type them in at all.

The "JM" software was the first to fit into two chips, using the space allocated originally on the QL circuit board. The first socket was intended for a 16K chip but on "JM" systems it held a 32K component. The second socket held the remaining code in a partly-used 16K chip.

Upgrades from "AH" to "JM" or later ROMs are not a simple matter of swapping components. Unfortunately the control signals required by three EPROMs are not the same as those needed by two ROMs. I have converted my QL from EPROM to ROM; it is fiddly but not too difficult if you are used to messing around inside computers. If not, you should get a QL repair firm to do the job for you – it is easy to write off a QL by damaging the circuit board.

Turn off the power, then unplug the EPROMs, which are behind the CTRL connectors in sockets labelled IC33 and IC34. Disconnect the trailing wire to the top EPROM; it is needed only to provide an extra signal for the third chip. Then plug the 32K ROM — marked '0000' after the version number — into the slot for IC33; the other ROM, marked '8000', goes into the other socket.

There are six positions for wire links to the right of IC34, labelled JU1 to 6. The first two are connected for EPROMs; to use ROMs you must cut link 1 and connect links 3 and 4. Finally, unplug the chip labelled SN74LSOON immediately to the right of the links; it is needed only by EPROMs

The upgrade from "AH" to "JM" is fiddly and does not fix many bugs. The next version of Qdos is a more popular upgrade, although it creates almost as many bugs as it cures. Early in 1985, Sinclair began shipping Qdos version 1.10, the "JS" ROM. At first it was claimed to be a development version, not intended for release. That may well have been the case but tens of thousands of ROMs were made. The "JS" ROM is the last version used in machines made for sale in Britain.

The "JS" ROM killed several annoying bugs of previous versions. It was the first to let you INPUT strings of more than 128 characters from Basic; it also handles CALL correctly in programs of more than 32K which usually crashed a machine running earlier versions.

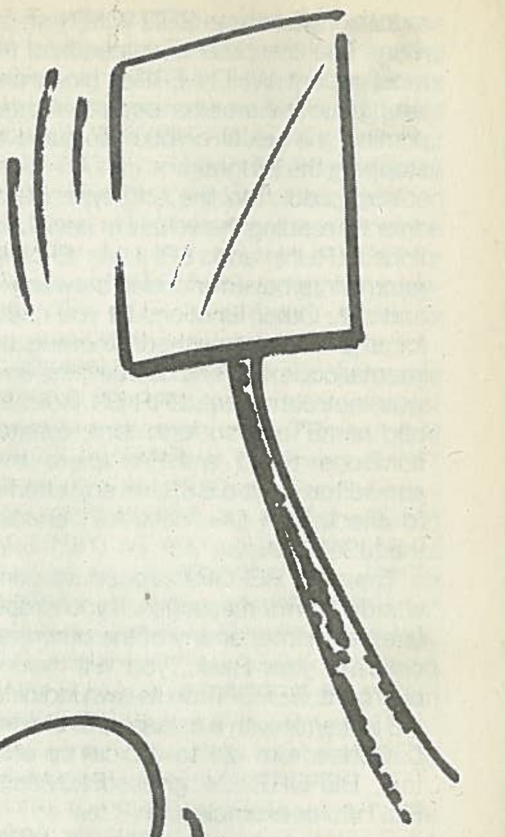
The "JS" ROM can change the display mode without setting the ink and paper in SCR windows to black and lets you define new procedures and functions with names you have used previously in the same program.

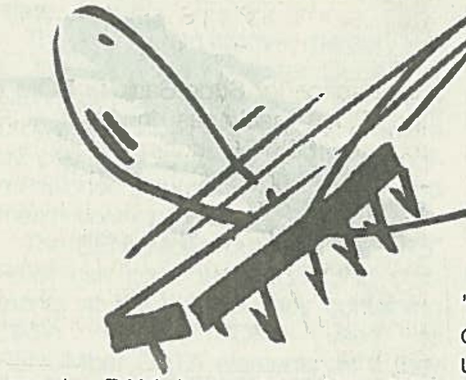
Machine code programmers will be pleased to find that the bugs in the number-base conversion routines vectors 260 to 270 have been fixed. Task handling is more friendly; you do not have to type Control C to retrieve the SuperBasic cursor when a task stops.

The "JS" ROM is the first version which can link more than one plug-in device into the system when you turn it on. QL devices can use a 256K area divided into 16 slots for ROM and port addressing. Those slots are used by disc controllers, sound boards, modems, and so on. Previous ROMs linked only one device into the first slot, however many were connected.

This bug was not serious. Most users have onlv one such device, a disc; expansion RAM does not require a slot. Many peripheral designers avoid the problem by putting a routine to link other devices in their own start-up code, so that the one gadget which is called can look up the others. Such a routine appears in chapter 9 of Andy Pennell's *Sinclair Qdos Companion*.

The other changes in the "JS" ROM are less helpful. The revised Basic prevents you entering integer and string SELect statements, which did not work anyway unless you had a compiler. Even the ROM version function, VER$, was a problem. In the process of changing an "M" into an "S", Sinclair stopped VER$ allocating memory for the value it returns. The machine may crash if you try to test VER$ without copying it to a temporary variable first. T$=VER$: IF "JS"=T$ will work will but IF "JS"=VER$ stops the "JS" ROM in its tracks.



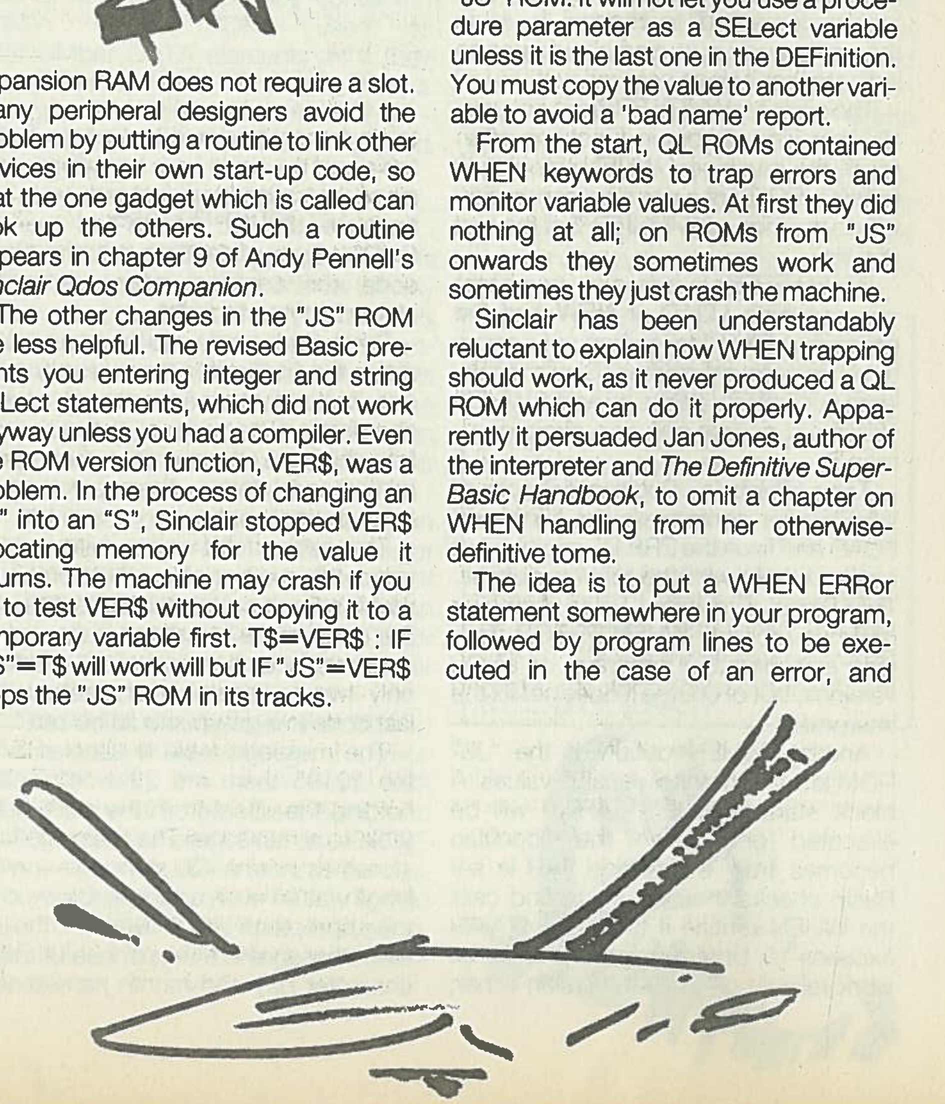


Parameters and SELect fell out in the "JS" ROM. It will not let you use a procedure parameter as a SELect variable unless it is the last one in the DEFinition. You must copy the value to another variable to avoid a 'bad name' report.

From the start, QL ROMs contained WHEN keywords to trap errors and monitor variable values. At first they did nothing at all; on ROMs from "JS" onwards they sometimes work and sometimes they just crash the machine.

Sinclair has been understandably reluctant to explain how WHEN trapping should work, as it never produced a QL ROM which can do it properly. Apparently it persuaded Jan Jones, author of the interpreter and *The Definitive SuperBasic Handbook*, to omit a chapter on WHEN handling from her otherwise-definitive tome.

The idea is to put a WHEN ERRor statement somewhere in your program, followed by program lines to be executed in the case of an error, and



rounded off with an END WHEN statement. The computer keeps track of the most recent WHEN ERRor block and jumps into it if an error occurs, without printing the usual cryptic message or stopping the program.

You check the line and type of the error by reading the values of new functions ERLIN and ERNUM. ERNUM returns internal error codes between -1 and -21. Other functions let you check for a given error, without knowing the internal code; ERR NC is true if the error was 'not complete', ERR BN indicates bad name' and so forth. Unfortunately someone typed a BRA where they should have put a BSR, so any attempt to check ERR DF, 'drive full', crashes the "JS" ROM.

The new REPORT procedure prints standard error messages. If you already use that name, or any of the other new ones, in your Basic, you will have to change it. REPORT on its own indicates the last error with a message to channel 0. Codes from -21 to -27 call up other text; REPORT -24 gives 'F1..Monitor F2..TV', for example.

No check

There is no check on the number you supply but only 27 messages are in the standard format, so beware. REPORT -28 and its brethren print a very long string of gibberish. REPORT 1,-19 prints 'not implemented' to channel 1; -19 is the polite code a routine should use to indicate that it does not work yet.

The code for WHEN ERRor is not usually that kind. Errors in functions often crash the machine if WHEN trapping is in force; SQRT of a negative expression will do the trick, as will INKEY$ at the end of a file.

WHEN ERRor is extremely persistent you can type LOAD or NEW and the computer will still try to trap your errors to a non-existent routine. Similar problems occur if you delete an active WHEN statement or type one as a direct command.

Tony Tebby's Supertoolkit clears WHEN after commands like NEW and LOAD and fixes the ERR DF mistake but it cannot help with the other problems. The Digital Precision Turbo compiler gives you reliable WHEN ERROR trapping anywhere in a program, on all QL versions, but of course it does not fix the interpreter.

Another WHEN option in the "JS" ROM lets you monitor variable values. A block starting WHEN VAR›10 will be executed only when the condition becomes true; every time VAR is set Basic checks the new value and calls the WHEN routine if the value of VAR exceeds 10. Unfortunately this does not work reliably on any QL version either; sometimes it gives a 'bad name' report or calls the routine more than once.

The last new command for the "JS" ROM is TRA. This makes it easier to customise QL software for use in other countries. TRA normally has two parameters. The first points to a table to be used to TRAnslate characters sent through the serial ports and the second is the address of a new error-message table. Both tables must start with the 'nonsense' word value 19195, which crops up all over the QL system as an indication that 'data follows'.

The next two words in the serial table contain the offset to two translation lists, measured from the start of the table. The first list starts with a 256-byte list of substitute codes for each character code from 0 to 255. When a character is

to be transmitted it is looked up in the table and the code from the appropriate place is transmitted instead; you can translate the QL end-of-line marker, CHR$(10), by POKEing 13, the usual code for carriage return, into the eleventh byte of this list.

Put the value zero into the appropriate slot in the first list if you want to translate one code into a sequence of several characters. They appear in the second list, which the QL uses only when transmitting serial data — there is no time to use it during input.

The second list starts with a byte value, the number of four-byte entries in that list. Each entry after that starts with the code to be translated, followed by three replacement codes. If you need only two replacement characters, the last code in a group should be zero.

The message table is simpler. After the 19195 there are 29 words, each holding the offset from the start of the table to a message. The messages are stored as normal QL strings — a word length, at an even address, followed by the appropriate text. Beware — the last two messages are sequences of three-character day and month names, with no length word. That is why REPORT -28 goes haywire.

If either parameter of TRA is zero the corresponding table is left alone. TRA 1 sects the standard message table and allows characters to be transmitted through the serial ports unmolested.

Turbo Toolkit contains an example program which uses TRA to translate Sinclair error reports into plain English; other TRA demonstrations have been printed in the user-group magazines *Quanta* and *Quaser*.

A special version of the "JS" ROM was produced for American QLs, which must be compatible with the rather feeble National Television Standards Committee TV standard. The American ROM had a three-letter name, "JSU"; it contains all the "JS" ROM bugs and features, plus changes which you should bear in mind if you develop programs which may be used in the States.

An American television set can display only 192 lines of pixels. In TV mode the American QL hardware ignores the first and last 32 lines of screen memory; in monitor mode it works with the usual 256 lines. American QL owners can swap between U.K. and U.S. TV lineage by POKEing 1 or 2 to address 163890 and typing NEW.

You still get 20 lines of text into a standard TV mode window, as the character set in the "JSU" ROM has been crushed vertically. Characters are drawn on an 8 by 5 dot matrix, rather than the 10 by 5 used on European systems. In monitor mode the crushed characters are still used but they are spaced by an extra two blank lines; rows of text are the normal height but look like a ransom note.

Another change compensates for the different shape of dots on an American display. The QL graphics co-ordinate routines compensate for the shape of each dot so that circles do not look elliptical and squares do not appear as rectangles. Routines which use pixel co-ordinates, such as WINDOW and BLOCK, do not perform any compensation, which is why vertical and horizontal units are different.

Atlantic ROMs

European and American ROMs use different compensation factors, so that graphics shapes look the same on either side of the Atlantic. Unfortunately there is no way to compensate for the difference in BLOCK and WINDOW shapes. Many programs use a mixture of graphics and pixel co-ordinates; they may look satisfactory on one side of the Atlantic but they will not line-up properly across the water.

Things are still tricky, even if you stick to one co-ordinate system. If you work entirely in graphics co-ordinates your shapes will not be distorted, although they may escape off the edge of the screen. If you use pixel co-ordinates everything will fit on the display but the vertical and horizontal proportions will be different on an American screen.

Sinclair's last fling was Qdos version 1.13, which usually crops up in "MG" ROMs. They have never been supplied in the U.K. although the chips work well in a British machine. The "MG"ROM has only one new bug and kills several important faults in previous versions.

First, the new bug. The "MG" ROM line-drawing routine does not always plot the point at the end of a line or arc, so that one-pixel gaps may appear at the corners of graphic drawings. If that disturbs you, a 'patch' program to correct the bug is available free from Qsoft, PO Box 56, DK 4000, Roskilde, Denmark. Send a disc or cartridge for the program and an international reply coupon for return postage.

Serious bug

The most serious QL filing bug has been fixed in the "MG" ROM. The Microdrive system does not hang up, stalling the computer, if file access is performed when the system is very short of memory. The QL file system uses spare RAM to buffer information en route between Microdrives and your program. The "AH" and "JM" versions could get stuck in a loop if the free memory fell to 1K, because the multi-tasking Microdrive handler would over-write the current block with new information before the application program had time to digest the original data. That is a common cause of failure when RAM-hungry Psion packages are used.

Sinclair tried to cure the problem in the "JS" ROM but managed only to fudge things so that all was well until there were just 512 bytes of free RAM — not much of an improvement.

Unless you have an "MG" ROM, it is worth performing a check for free memory before Microdrive access. This function returns the amount of space free for Microdrive buffering:

```
DEFine FuNction BUFFER
SPACE RETurn PEEK L(163856)
-PEEK L(163852)
END DEFine BUFFER SPACE
```

The "MG" ROM is the first to be able to close the second serial port, SER2. Earlier ROMs used to close SER1 instead whenever you tried to close SER2.

Apparently the "MG" ROM is the first to work correctly on a QL with eight Microdrives. I suspect such systems are rare, even though Spectrum Microdrives will work if plugged into the QL extension drive socket. It is claimed that earlier ROMs used to forget about MDV2 after you had used MDV8.

"MG" SuperBasic has been thoroughly spring-cleaned. You can use any number of parameters and LOCALs in a procedure or function. Previous versions of the Basic allowed only enough space for nine such names. If you used more, the program could lock up or be corrupted by the appearance of spurious PRINT keywords in place of names towards the end of the program. That knowledge may help you to spot program listings which were improved by their authors, untested, before publication.

The "MG" system is less prone than its predecessors to use up RAM as a program runs. Earlier ROMs lost track of some memory every time a slice of a dimensioned string array was passed as a parameter — PRINTed, for instance. If that happened in a loop, as usual, the program ran slowly, constantly grabbing more and more memory, until it failed 'out of memory' discarding all variable values. "MG" does not get into this state.

The only way to recover memory lost in this way was to enter CLEAR or NEW,both difficult to do in a running program. You could avoid the problem by copying slices to temporary, undimensioned strings but that is a slow and messy solution.

The RENUMber routine in the "MG" ROM can cope with RESTORE statments at the start of lines containing DATA. Earlier ROMs used to RENUMber DATA elements as if they were line numbers. This bug was a hangover from the days of the kludge – the original RENUM totally ignored RESTORE.

Any channel

The "MG" CURSOR command lets you use graphics co-ordinates on any channel. Other ROMs let you use only four co-ordinates with the default channel, channel 1. DATA values in brackets no longer cause the other items on the line to be ignored. CLS and PAN can cope with windows narrower than the cursor.

String comparison works properly on characters with ASCII codes greater than 127. Previously you had to check the CODE of the character, rather than compare the string correctly, to check reliably for cursor, function and other special keys.

The WHEN routines are still unfinished in the "MG" ROM, although the trivial ERR DF bug has been fixed. Daft parameters no longer upset READ and INPUT and you can OPEN and CLOSE channels *ad nauseam* without the system complaining; older ROMs limited you to 32,767 OPENs in a session, which used to upset dreadful programmers.

The "MG" ROM was designed for use in continental Europe and is in several versions, with key layout, characters and messages customised for different nations. The 32K ROM is the same for each country and the 16K ROM holds all the information, which varies between versions. VER$ has an extra letter at the end, to show the country — "MGE" for Spain, or "MGS" for Sweden, for example. The dot in the Qdos version is replaced by the country-code letter.

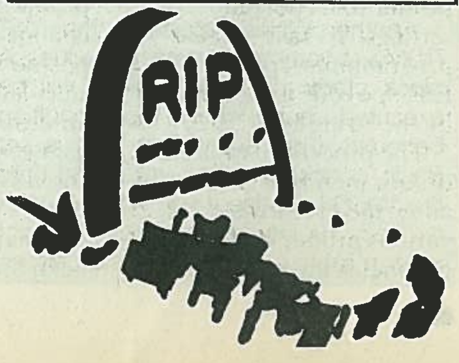
If you want to use a foreign-language "MG" ROM it is easy to create a new set of error messages using TRA but that may not be sufficient. The French "MGF" ROM expects a particularly odd key layout. "A" and "Z" swap places with "Q" and "W", so the layout, like that of French typewriters, is known as AZERTY rather than QWERTY. The "M" key is moved to where "," is normally found, and the "£" key works as an extra shift, putting a circumflex accent or an umlaut over the next vowel typed. Several other characters are shuffled to make space for accented letters.

The last Sinclair effort was the Greek ROM. It works well in the U.K. as soon as the error messages are changed from their heiroglyphic Greek form which, to be fair, is about as clear as the original English. These ROMs use Qdos 1.13 but they are further developments of the "MG" ROM and cannot be mixed with other Qdos 1.13 chips. Greek ROMs display their version as "Sigma FP",although the chips are marked "EFP";presumably the Sinclair Mexican chip makers could not find a sigma stamp.

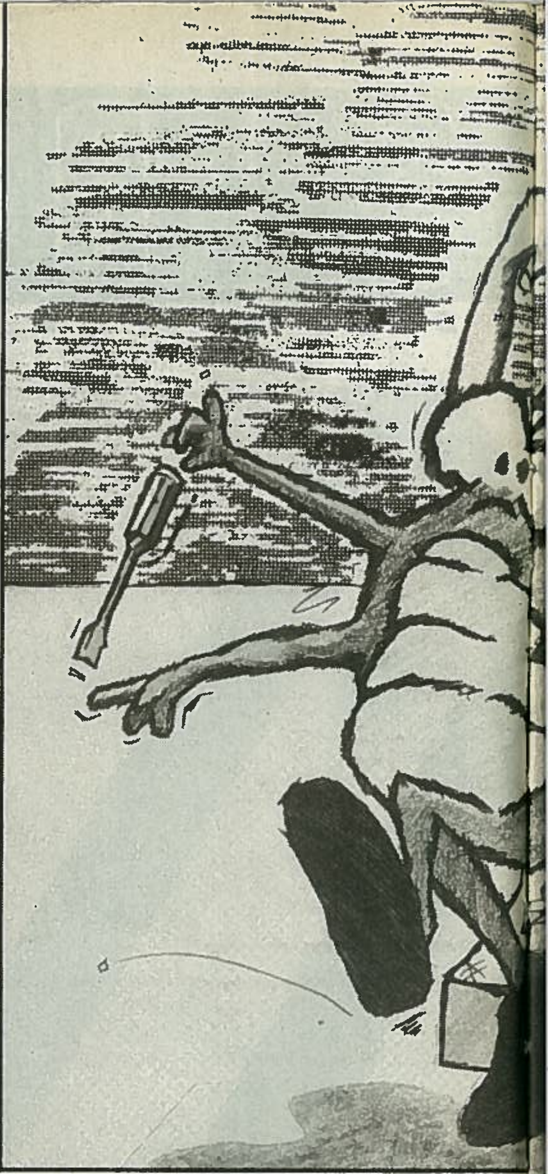
The "MG" ROM is a great improvement on its predecessors but even so it contains 30-odd unfixed bugs. Next month I will list them and explain how to circumvent them.

All the QL ROMs from JM onwards, including JSU, are plug-compatible. All you need to do the change versions to disconnect the power and replace the components in sockets IC33 and IC34 with a different version.

"AH", "JM", "JS" and "MG" ROMs are available in Europe from Adman Services, 53 Gilpin Road, Admaston, Telford TF5 0BG. Remeber that the wiring of the ROM sockets must be changed if you switch to or from version "AH". The American "JSU" ROM is supplied by Curry Computer, PO Box 5607, Glendale, AZ 85312-5607, U.S.A.



Beating the bugs



Simon Goodwin concludes his survey of QL system bugs with a summary of the faults shared by every QL, regardless of vintage or nationality.

Forty-six errors in early versions of the QL operating system were described last month and I explained how they could be fixed by ROM upgrades. Now I list another 31 bugs which crop up in every version of the QL and tell you how to circumvent them. If you have found others, please let *QL World* know.

There is no easy way to define a bug. It is well-known in the computer industry that one person's bug is usually someone else's feature. It is almost as well-known that the person with the bug will be a customer and the person with the feature will be a marketeer.

In the absence of an industry-standard definition of a bug, I have set out to list all the quirks of the QL ROM which cause apparently correct programs to give unexpected results, or no results at all. I have also counted a few undocumented restrictions. Any non-violent action which prevents the entry of further commands is automatically considered a bug, unless Sinclair specifically warns against it in the big black QL User Guide.

That raises the question of how should a bug be fixed? In some cases, it is sufficient to detail the problem so that people can avoid it. If you document a bug, it is vital to tell users how to get the result they want without getting into difficulty.

Another approach is to make it impossible to reach the circumstances which cause the problem. Integer FOR loops did not work on the AH version of the QL, so Sinclair changed later versions so that you could not even type them in. That approach may be justified on grounds of expediency or efficiency but often it is just an excuse for leaving a real flaw uncorrected.

The last kind of bug-fix is a rare and wonderful thing. It lets you do what you originally wanted, in the way you intended. That is the best for customers, unless their original idea was a daft one, but it is the most expensive for all concerned. It often leads to the introduction of new problems, because a technique which used to work is clobbered by the fix, or because the correction passes you to more faulty code.

This list deals only with idiosyncracies of the QL ROM, the SuperBasic language and the underlying collection of operating routines called Qdos. Some of these bugs can cause other programs to fail. Software developers should guard against the most common problems by defensive programming in their own code, which is why I have included plenty of detailed technical information.

The problems are collected under two headings. Input/Output bugs affect the flow of information between the QL and peripherals such as drives or the display. SuperBasic bugs affect the executive of QL Basic programs; many are corrected if you compile your programs.

SuperBasic bugs

PROBLEM 1: When a REPeat or FOR statement is encountered, the value of the corresponding identifier is set to zero. For instance, this line prints five values, 0 to 4, rather than 3 and 4, as you would expect in any other Basic:

X=3: FOR X=X TO 4:PRINT X

The same problem occurs if the value of X is used to compute the end-point of the loop, or the step.

REASON: The SuperBasic interpreter does not use the same format to store loop details and simple variables. Whenever a loop starts the old value is thrown away and a new, zerod storage area is allocated.

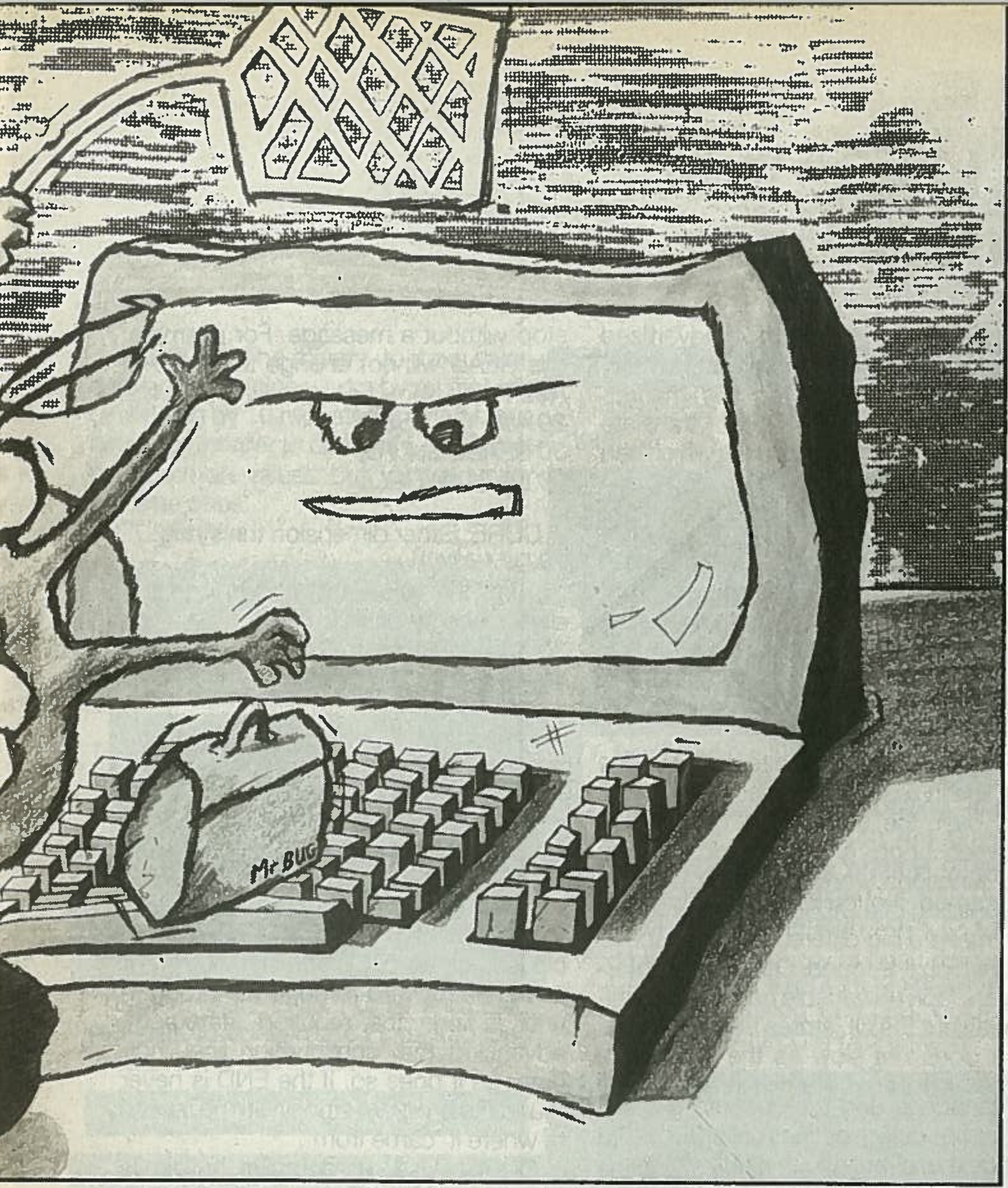
CURE: The best solution is to change the variable name — it is poor programming practice to use the same name for two logically-distinct purposes, in this case as a terminal value and an instance count. Nonetheless, there are times when that may be convenient. SuperBasic compilers do not have this bug, as they analyse the program during compilation and allocate space for the value and loop details from the start.

PROBLEM 2: The QL trigonometric package gives silly results for COS between 16384*PI radians — about three million degrees — and 65535 radians. Greater values give an overflow error. For a weird result, type:

PRINT COS(60000)

CURE: use SIN(X+P1/2) instead of COS(X), or do not do it. It is extremely unlikely that your program will fall foul of the bug unless it has gone haywire, in which case it will probably run into the overflow error. Three million degrees should be enough for anyone.

PROBLEM 3: RESPR, the function to reserve memory, does not work while a task is running. It gives a 'not complete' error.



REASON: The memory map in the Concepts section of the QL User Guide shows that the area of memory used by RESPR fits between the top of RAM and the first task loaded. Tasks may not be moved, so you cannot expand the RESPR area while any task is loaded.
SOLUTION: Use a toolkit function like ALLOCATION or ALCHP to obtain memory from the common heap at the other end of the memory map. Compilers re-direct RESPR calls to the heap automatically, since all compiled programs run as tasks.

PROBLEM 4: Every time you type RUN after a SuperBasic error — or BREAK — the system loses track of another 16 bytes of memory. You do not get this memory back until you type NEW.

REASON: The system does not tidy data allocated on the user A7 stack when an error occurs after RUN.

CURE: Start your program with GO TO instead of RUN. CLEAR and the 'Out of Memory' condition do not release the space — you must type NEW. Remember to SAVE your program first. Luckily the creepage is only small and you have to type RUN many times before you exhaust the capacious QL memory.

PROBLEM 5: You cannot de-allocate space reserved from SuperBasic with RESPR. There is no command to do this in Basic. The required machine code routine exists in the ROM — TRAP 1 with D0=15 — but it does not work; it may do nothing, allocate MORE space, or crash the system.

REASON: Resident procedure space is intended for device drivers and commands which are linked permanently into SuperBasic. If such code was overwritten the machine would crash as soon as the system tried to use it. That does not explain why the ROM routine was partly-written in the first place and it does not help people who use RESPR memory to store data such as character sets, or machine code to be accessed with CALL.

CURE: Use toolkit functions to work with space on the common heap. ALLOCATION or ALCHP will grab space and DEALLOCATE or RECHP will release it. Memory allocated by a compiled task is usually released automatically when the task stops but you can prevent this, if need be, with Turbo Toolkit. Extension commands and devices should always be linked from SuperBasic with RESPR.

PROBLEM 6: A GO SUB in a single-line FOR loop acts like an END FOR. This prints all the values of X from zero to ten, but only displays 'Counting' after the last value:

```
5 FOR x=1 TO 10: PRINT x: GO SUB 20
10 PRINT "Finished": STOP
20 PRINT "Counting": RETurn
```

CURE: Compile the program or use a multi-line FOR loop, with an explicit END FOR, instead of the short form. The problem crops up only if the GO SUB is on the same line as the FOR. Alternatively, replace the GO SUB with a procedure or function call.

PROBLEM 7: It is impossible to use an Integer or string variable as the identifier of a FOR loop. Early QLs let you enter these statements but could not run them; versions from JM onwards reject the loop as a 'bad line'.

CURE: All QL compilers support integer FOR loops, although you may have difficulty entering them. Turbo and *Q-Liberator* have implicit type directions so that you can tell the compiler to make a variable an integer without upsetting the interpreter by putting a percent sign after the name. No-one has yet implemented string FOR loops.

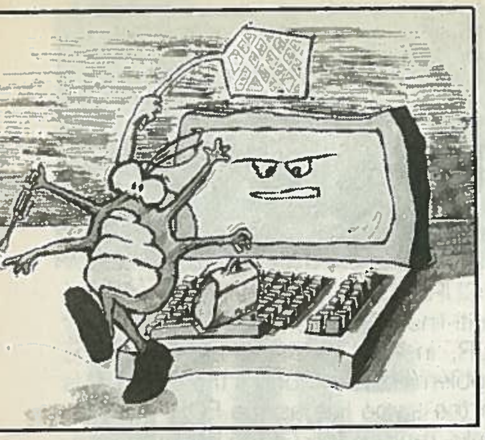
PROBLEM 8: The integer division operator DIV does not check for all cases of binary overflow; it gives spurious results instead. Try:
PRINT -32768 DIV -1

REASON: The QL uses the common "twos complement" format to encode signed integers into 16 bits. In this scheme, the representation of 32768 and −32768 is identical. To avoid ambiguity, the maximum valid integer is 32767 and the minimum is −32768. DIV does not check for the case when the minimum is negated; this should cause an overflow.

CURE: Avoid extreme values with MOD and DIV, or use a compiler to check your code. Turbo and *Supercharge* diagnose the overflow correctly. Q-Liberator repeats the division using floating point maths, postponing diagnosis until the value is assigned to an integer variable.

PROBLEM 9: You cannot use an integer or string variable as the identifier of a SELect statement. Early QLs let you enter these statements but could not run them; versions from JS onwards reject the statement as a 'bad line'.

CURE: Compile your program with Turbo on any QL version, or with Supercharge on AH or JM versions. Integer SELect is extremely fast.



PROBLEM 10: The INT function gives an overflow error if its argument exceeds 2 to the power of 31 minus 2, about 2.14 billion (U.S.).

REASON: The QL floating point maths scheme uses 31 bits to store the digits of a value. That allows any number from about minus 2.14 to plus 2.14 billion to be stored exactly. If a value is outside this range the computer can store it only approximately, as the first nine digits with an exponent to indicate the position of the decimal point. The machine cannot be sure of the value of the last digits. INT fails rather than return an approximate result.

CURE: Avoid such values, or use a range check to filter out values which INT cannot truncate. There is no way to INT larger values accurately without increasing the precision of the floating point format. That would require a major ROM re-write and make floating point maths much slower and more cumbersome.

PROBLEM 11: You cannot BREAK into a single-line recursive procedure.

REASON: Sinclair did not put a check for BREAK in the procedure-call code. This oversight saves time and is unlikely to cause problems.

CURE: Split the procedure over several program lines.

PROBLEM 12: You cannot interpret more than one SuperBasic program at a time.

REASON: The SuperBasic task is handled specially by the ROM — it is the only task which can grow and shrink dynamically as it runs. Most of the interpreter code would multi-task satisfactorily but interpreted programs would be very slow, because of the need to move programs around memory as they collided with one another. Much information, such as resident procedure details, would have to be duplicated between copies. The BREAK mechanism would also have to be altered, as at present it will interrupt only task 0.

CURE: Several firms have advertised routines to make the interpreter multitask; none has reached the market. SuperBasic compilers will translate programs into standard tasks which can run within fixed bounds.

PROBLEM 13: Contrary to Sinclair claims, the speed of SuperBasic interpretation declines steadily as program size increases. A jump to the end of a large program on a standard QL can take 100 times as long as a jump to a routine at the start.

REASON: When the interpreter jumps to a line in a program it has to read and skip over the start of all the lines in between. Loops and calls to procedures and functions work the same way, so the position of a DEFinition in a program can make a big difference to the speed with which it is found. Other Basic interpreters look up variable names in a similar way but the QL stores them by index, so it does not slow as the number of names in your program increases.

CURE: Compile the program. That fixes the address of each line or routine so that it can be found instantly wherever it may be in the program.

PROBLEM 14: If you type EDIT after breaking into a procedure or function, SuperBasic can present you with a 'not implemented' error and the wrong line.

CURE: Press BREAK and type the EDIT command again. Do not be tempted to edit the line you are given, or you could corrupt the whole program.

PROBLEM 15: SuperBasic locks-up if you type CLEAR or edit a line after trying to call a procedure or function which was defined at the end of the program but deleted later. This happens only if there is no line left with a number beyond that of the DEF of the deleted routine.

REASON: The interpreter gets stuck in a loop if it is asked to clear details of a PROC/FN call which does not make sense.

CURE: Do not do it, or keep a STOP or REM on line 32767 at the end of your program.

PROBLEM 16: If you try to READ or INPUT a value into a slice of a string which has not been dimensioned, the value will not be stored and Basic may stop without a message. For example, this READ will not change the value of A$:

```
20 A$="QL USER!"
30 READ A$(4 TO)
50 DATA "WORLD"
```

CURE: Either dimension the string:

```
10 DIM A$(8)
```

or read the value and assign it to the slice in two steps:

```
30 READ T$
40 A$(4 TO)=T$
```

PROBLEM 17: If an END is missing from a program, interpretation may stop with no indication of the problem or its location. Spurious ENDs are ignored, with no message.

REASON: When the interpreter needs to find an END — after a conditional clause, or an EXIT, for instance — it searches forward through the program until it finds the required statement, advancing the 'continuation line' indicator as it does so. If the END is never found the program stops, with no record of where it 'came from'.

CURE: Use a compiler or style checker, such as Better Basic, to make sure that starts and ends are matched properly. Under such circumstances the Turbo Toolkit routines TRACE and HOW COME help you to determine where a program failed.

PROBLEM 18: Memory remains allocated every time you jump out of a procedure or function without performing a proper RETurn or reaching END DEFine. The space is recovered by CLEAR.

CURE: Do not do it. In a well-designed program, every routine should have a single entry and exit point. Avoid GO TOs and arbitrary use of REPeat and END REPeat to perform jumps, e.g.:

```
REMark Very bad style
REPeat loop
   FRED
   DEFine PROCedure FRED
END REPeat loop
END DEFine
```

This will work but it is horrendous style and will consume memory at a rate of about 1K per second.

PROBLEM 19: Very large numbers take a long time to be converted for PRINTing. Try this:

FOR I=1 TO 10:PRINT 123456E610

Beating the bugs

CONTINUED

REASON: The binary to text value conversion routine works by multiplying or dividing by 10 repeatedly until it has a nine-digit integer to deal with. This is fast for common values but very slow for extreme ones."

CURE: Wait.

PROBLEM 20: The QL internal arithmetic routines are accurate to more than nine digits but only a maximum of seven digits are displayed.

CURE: Compile the program with Turbo or Supercharge, both of which show all nine digits and correct small errors in ROM floating point routines.

PROBLEM 21: You can use only one short-form FOR or REPeat statement on a SuperBasic line. If you put more than one, only the last one will work.

CURE: Add END FORs and END REPeats to convert the loops into long forms.

Input/output bugs

PROBLEM 22: FILL sometimes colours the same line twice. This causes problems only if you are using OVER -1, as the effect is to reverse the effect of the FILL on that line.

CURE: Start drawing the pattern from a point at the top or the bottom.

PROBLEM 23: If you MERGE a file of direct commands, only the first line will be read and the file will not be closed. That makes it impossible to use another tape or disc in that drive later.

CURE: Put the Supercharge/Turbo Toolkit command END CMD at the end of the command-file line. That closes the file.

PROBLEM 24: The machine may crash if a syntax error — 'bad line' — on a line without a number is read from a command line.

CURE: Do not do it. Check your command file by putting a number at the start of each line and LOADing it; potential 'bad lines' will be marked with 'MISTake'.

PROBLEM 25: Reports may be lost or the SuperBasic interpreter may be locked out if the standard SuperBasic channels 0, 1 and 2 are CLOSEd.

CURE: Do not do it — Psion take note.

PROBLEM 26: If you load a file with LBYTES after editing and saving it there is a risk that the file loaded will not include all the changes you made.

REASON: When you CLOSE a file, Qdos prepares to replace all the parts which have been changed. They are copied from temporary storage in memory 'slave blocks' into the correct place in the file. In the interests of speed, LBYTES suspends this copying and loads the file directly from the drive, without checking to see whether or not some blocks have changed recently but have not yet been written back to the medium.

CURE: Wait for the drive to stop after CLOSE before using LBYTES, or use the Super Toolkit FLUSH command.

PROBLEM 27: The Microdrive handler gives a delayed and misleading 'bad or changed medium' message if you try to store something on a tape which is write-protected.

REASON: The handler does not check whether or not a drive is write-protected when a file is opened or data is written. Luckily for the existing data, the low-level routines detect that the tape is protected and do not allow writing. The control software assumes a 'bad medium' when several attempts to write have failed.

CURE: Beware, and do not assume disaster if you get a 'bad medium' report on one of your master tapes. Machine code programmmers can tell whether or not the currently-turning tape is write-protected with IPC call 1 but this does not tell you which drive is turning. To find that, enter supervisor mode to prevent asynchronous changes and read the drive number byte at 164078 before interrogating the IPC. A proper fix should be buried deep in the code of the device driver.

PROBLEM 28: If you set a position for binary random access far beyond the capacity of a cartridge you may get a misleading 'bad or changed medium' message, instead of 'out of range'.

REASON: The Microdrive handler uses 24-bit addresses internally, limiting the length of a file to about 16.7 million bytes. Larger values corrupt other information packed into a 32-bit register with the address.

CURE: Do not be misled by the message — your tape is intact. Then lower your sights; 16 MB Microdrives are some way off yet.

PROBLEM 29: You cannot draw a block of width 512. Nothing happens if you try it.

CURE: Use CLS or two horizontally-adjacent BLOCKs.

PROBLEM 30: The priority of SuperBasic, task 0, may be set to zero, preventing further command entry.

CURE: Use Turbo Toolkit or Supercharge extensions which check for this case and exclude it.

PROBLEM 31: Pressing the combination of keys CTRL, ALT and 7 — or 2 or 5 on some machines — usually causes the QL to crash at once.

REASON: These keys trigger a level 7 interrupt, which is intended to call-up a hardware debugger which Sinclair staff used while testing development systems. The facility is still there in production machines, even though the external debugger is not. The interrupt re-sets the 8049 second processor but not the main 68008. If the interrupt occurs while these two chips are communicating, as is highly likely, the 8049 'loses its place' and crashes, preventing keyboard input.

CURE: In general, do not do it. The keys have been chosen to make accidental entry very unlikely. The system call MT.TRAPV (TRAP 1, D0=7) lets you specify a routine to be executed when hardware errors or interrupts occur. Unfortunately this is of limited use, as the keyboard and RS232 die when the 8049 re-sets.

You should not treat this list as an indictment of the QL. Every computer system, of whatever vintage, contains bugs and the QL is no worse in this respect than other ambitious designs. At least, and at last, you are forewarned by this list. Each version of the QL ROM has its own bugs, besides those listed. Read the article entitled Version Therapy, in last month's *QL World*, to find more about specific ROM faults.

Return of the ROMs

When I revealed the results of three years' research into the idiosyncracies of the QL built-in software — the operating system Qdos and the SuperBasic interpreter — I found and explained 77 bugs in the QL ROM.

Since then, with the help of *QL World* readers, I have identified another 11 problems, so now is the time for an update.

This is more than just a list of faults; it explains how to circumvent them. All complex systems contain bugs, though hardware manufacturers curiously are shy about admitting them and sorting them out. Bugs are rarely a problem if you know about them and how to avoid them. All most users want to know is how to get the result they need without getting into difficulty.

There is no sure definition of a bug. One person's bug is usually someone else's feature. I have included quirks of the QL ROM which cause apparently correct programs to give unexpected results, or no results at all, plus a few undocumented features. The list deals only with idiosyncracies inside the QL ROM — the SuperBasic language and the associated collection of operating routines called Qdos.

Some of the bugs may cause other programs to fail, so I have included technical information to help software developers guard against the most common problems by defensive programming in their own code. The bugs are in two groups — problems which afflict all QLs, followed by a list of faults specific to certain ROM versions.

'New' bugs in QLs everywhere

Integer input

Dilwyn Jones reports a sometimes annoying bug in all QL ROMs. Integers — whole number values stored in variables with a percent sign at the end of their names — can have values between -32768 and 32767. The statement X%=-32768 works satisfactorily but X%=-32769 gives an error as you might expect.

You cannot INPUT a value of -32768, If you try to do so you get an 'error in expression' report because the QL works out the value of the digits before its sorts out the sign, plus or minus, and +32768 is not a valid integer. Qdos uses the same code to convert values from all devices, so the bug is present whether your INPUT is from the key-

board or a file.

It is really just sloppy coding on the part of the ROM authors who seem to have difficulty with the value -32768. I pointed out previously the weird results you can get using that value with the integer DIV and MOD operators.

Simon Goodwin follows up last year's look at the QL built-in ROM software with 11 new bugs and more about the QL

Window rules

You can define the position of any window on the screen in terms of co-ordinates in picture elements or pixels. The co-ordinate scheme assumes that there are 512 pixels across the screen and 256 downwards. Window widths and horizontal co-ordinates are always rounded to an even value. This means you cannot put a one-pixel gap between two windows in MODE 4, the highest resolution QL display mode. The minimum gap is two pixels.

You cannot deal with this by setting a BORDER width of one in the window, as horizontal border widths are also rounded up, so that BORDER 1,7 gives a white border one pixel wide in the horizontal lines but two pixels wide vertically. You can easily see this if you use a stippled border pattern:

```
MODE 4 : BORDER 1,7,0
```

This bug is not properly-documented but understandable when you think about the QL display design. The restriction exists because QL windows are designed to be able to cope with a change of MODE at any time. One mode allows four colours, with 512 dots across the screen, while the other allows eight colours with 256 dots on each line. A gap of one pixel in MODE 4 would become a problematic gap of half a pixel as soon as MODE 8 was selected.

Merge bugs

The MERGE and MRUN commands become confused if you use them inside a SuperBasic procedure or function because the act of merging new program lines invalidates stored information about where in the program execution should continue.

SuperToolkit 2 re-defines those commands to detect attempts to use MERGE inside a DEFinition. It stops the program with a 'not implemented' report if it runs into trouble.

Cotangent error

Dr. Helmut Aigner of Austria discovered that the Co-Tangent function, COT, gives a result of 1 when asked to find the co-tangent of zero, whereas COT(0) is undefined and should really give an 'overflow error.'

The error is in the Qdos maths package, rather than in SuperBasic, so it affects other languages. In general, if a language uses the Basic 7-9 digit precision, it is likely it will inherit this bug. It is easy enough to check for the special case of zero explicitly in programs which use co-tangents.

Startup keys

According to published documentation about Qdos it should be possible to tell whether the user started the QL by pressing F1 or F2 by reading the value in address 163890, known as SV.TVMOD. This information would be very useful when programs start as they could work out whether or not the user had a monitor and set windows to suit automatically.

When the QL starts PEEK(163890) is 0 for a monitor display (F1) and 1 or 2 for a TV display (F20); 1 indicates a European TV, capable of displaying 256 horizontal lines of pixels, and 2 means that a 525-line American display was selected, with 192 lines of pixels and characters eight rather than 10 pixels high.

Unfortunately the MODE command, used to switch between four- and eight-colour displays, has a bug which means that the value of SV.TVMOD, the F1/F2 flag, is affected as soon as you issue your first MODE command. The result is that programs have to deduce whether you are using a TV or a monitor indirectly by checking the screen mode — four or eight colours — rather than the initial selection you made after turning on the machine. This is a fault because it does not necessarily follow that you are using a monitor because you are in MODE 4 before you start using Quill. Nor does it follow that you have a TV because you load a Psion program from MODE 8.

Current versions of the Psion package no longer check SV.TVMOD because of the bug. You can circumvent the fault when using programs which test SV.TVMOD by POKEing the required value back into 163890 but this will not help if your program loader issues a MODE command before it tests for TV or monitor selection.

This bug can be cured by re-writing the MODE command to set register D2 to -1 — meaning no change — before calling the operating system. Anyone who owns a copy of Speedscreen will find that it fixes this bug automatically by replacing the standard MODE routine with an enhanced and corrected version. If you want to use it to keep the original F1/F2 value you should load Speedscreen at the start of a session before the first MODE command.

CLS

By far the most interesting QL bug occurs in the CLS command. All known ROM versions accept undocumented CLS parameters and do unexpected things as a result. The CLS command allows a single optional numeric parameter. Officially it is a value between 0 and 4, referring to different sections of the display as documented in the QL User Guide.

Non-standard values cause calls to other display device routines, using whatever parameters happen to be in registers when the call takes place. Some of those routines are not normally accessible from standard SuperBasic. The property appears to be an accident, although it can be useful in practice.

The internal routines to clear different areas of the screen form a sequence of distinct system-calls — SD.CLEAR, SD.CLRTP, SD.CLRBT, SD.CLRLN and SD.CLRRT, using system call numbers 32 to 36 inclusive. CLS converts parameters between 0 and 4 into a call number of 32 to 36. So choose the appropriate ROM routine.

Other system call numbers correspond to different display operations and the code for CLS allows parameters outside the documented range of 0 to 4.

Parameters between 5 and 7 give a 'bad parameter' report but CLS changes the current STRIP colour, the background colour used when printing characters. CLS 8 works like STRIP 0! You can put a channel number before the parameter to select the window affected by the command CLS #0,8.

CLS 9 works like INK 0, which is particularly interesting when you realise that the system call to set the strip colour is number 40 and the call to set the ink is number 41. The sequence continues through the TRAP #3 display routine, so CLS 10 sets FLASH 1, CLS 11 sets UNDER 1 and CLS 12 selects OVER 0.

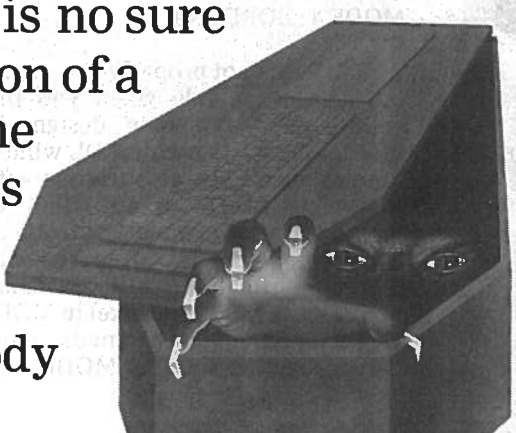
Values between 13 and 15 give a 'bad parameter' again, as do all parameters which give results between 5 and 7 if you make them MOD 8 but then things become really interesting. CLS 16 plots a point at graphics co-ordinate 0,0. The next three have no obvious effect but CLS 17 draws a zero-length line, while 18 and 19 draw zero length arcs and ellipses.

CLS 20 calls SD.SCALE, system call 52, and has the rather annoying effect of setting an enormous graphics scale so that lines, arcs and ellipses all appear in the bottom left pixel of the window. Use SCALE 100,0,0 to set things to rights.

Parameters from 21 to 95 give 'bad parameter'. CLS 96 appears to do nothing but in fact it checks the channel for pending input, using system call zero. The parameter values have 'wrapped around' internally to start again at the lowest call numbers. CLS 97 waits for one character before returning. CLS 98 uses the INPUT routine IO.FLINE to read the character codes between 32 and 191. Characters are displayed but not returned and the buffer size is just 3.

CLS 99 calls IO.FSTRG; it fetches a line of up to three characters of any code without displaying them. ALT keystrokes count as two characters. CLS 100 calls IO.EDLIN, the Basic line editor. A long strip of gibberish appears; you can edit the text but any attempt to insert characters after the second gives a 'buffer full' error.

“There is no sure definition of a bug. One person’s bug is often somebody else’s feature.”



CLS 101 to 104 give 'bad parameter' again. You should not enter CLS 105 as it locks up the machine; it corresponds to SD.EXTOP, the extended operation call, and in this case it hands over control to a non-existent routine.

CLS 106 and 107 have no obvious effect but they read the window size in pixels and characters. You get a three-pixel-wide, horizontally-striped border with CLS 108.

CLS 109 to 111 give 'bad parameter', unfortunately, so you cannot access the routines to turn the cursor on and off. CLS 112 and 113 call the POS and TAB routines, in both cases causing an 'out of range' report. CLS 114 moves to the next line unless the window could need to scroll in which case it gives an 'out of range' error. This call, to SD.NL, could be useful in screen-handling programs if you have some way to trap the error — it prevents having to keep checking the current line when moving. CLS 115 moves the cursor left, giving 'out of range' after the left most column of the window, and CLS 116 moves the cursor right.

CLS 117, 118 and 119 give 'bad parameter' but CLS 120 scrolls down the window by 10 lines; 121 and 122 scroll each side of the window, while CLS 123 pans the window right. Parameters from 124 to 127 are rejected and at CLS 128 we are back to the same effect as CLS 0. The parameter values cycle round in a 128-step sequence.

Special bugs

The remaining bugs affect only the specific versions of the QL noted. The August 1987 edition of *QL World* explained how you can upgrade the ROMs in your QL. You can obtain most QL ROM versions from Adman Services at 53 Gilpin Road, Admaston, Telford TF 5 0BG.

I said initially that JM and later versions were made using mass-produced 'mask-programmed' ROM chips, whereas the AH and earlier versions used individually-programmed EPROMs. The upgrade procedure from EPROM to ROM is significantly more complicated than from one ROM to another, when you can just swap the chips in their sockets.

Since then I have heard from D. A. Masters, who bought a JM QL with EPROMs in it. It appears that the first 100 or so JM QLs were made with EPROMs rather than ROMs because the JM software was ready but had not arrived from the manufacturing sub-contractor. The upgrade procedure from JM EPROMs is the same as that for AH chips.

Second processor

I have found a cure for the CTRL-ALT-7 bug, documented last year. Most QLs lock up i f you type those characters because the software in the second processor, separate from the main ROM, treats that keypress as a request to call up external hardware which only Sinclair owned.

Add-on keyboard manufacturer Schoen recently produced a replacement second processor to cure key-bounce problems for people using its keyboard and this upgrade also prevents CTRL-ALT-7 interrupting the machine. Unfortunately the Sinclair key-bounce fix, the version 1.2 chip from Applied Technology, does not correct the CTRL-ALT-7 bug.

Editing cursor

The first two workable QL versions, AH and JM, have a bug in 10.FLINE and 10.EDLIN, routines used by INPUT and EDIT. If the data entered becomes too large for the available storage buffer the routine gives an error message but leaves the cursor turned on in the input window.

This does not cause problems in the JM ROM versions of those commands because the ROM code turns off the cursor after an error to be on the safe side. It can cause problems if you write your own machine code programs and call 10.EDLIN or 10.FLINE.

You can prove that the error exists by typing CLS 98 to call 10.FLINE directly, then typing three characters to fill the buffer. An error is reported and the command cursor appears at the bottom of the screen but the cursor at the top of the screen is still flashing. Type CTRL C to get back to the command line, then enter INPUT X$. Finally, press ENTER and let INPUT turn off the stray cursor.

The DIY Toolkit function EDLINE$ and the Turbo Toolkit EDIT%, EDIT$, EDITF functions all contain code to turn off the cursor explicity so they are not affected by the bug.

Bad names

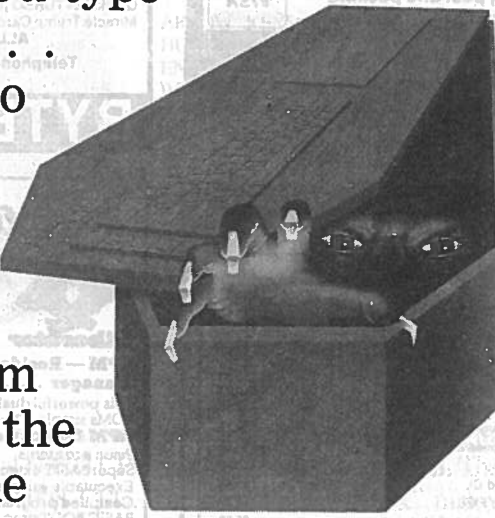
The AH and JM versions of QL SuperBasic have the annoying bug that they will not let you re-define names which have caused the computer to give a BAD NAME report. You might become irritated by the standard QL display speed and type: -SPEED 2 to turn on Speedscreen, only to find that it was not loaded. The system reports a BAD NAME error — unless you have the ROM version — because it does not recognise the command. After that normally you would use RESPR to reserve some space for the code, load it with LBYTES and call the start address.

In this case you are usually safe, because Speedscreen turns itself on automatically when you load it but the -SPEED command is still considered a 'bad name' by an old QL system. So you cannot check the Speedscreen version with -SPEED 1, because -SPEED is still defined as a Basic 'bad name' rather than the name of an extension command.

The same confusion occurs if you try to use any other Toolkit commands before loading them. If you try to use them before they are loaded you confuse the system. Such commands work properly after you type NEW, because that clears out all prior SuperBasic definitions, leaving only the resident commands and functions. Unfortunately this also gets rid of your program and all the variable values.

If you try to run a program compiled with version 2 of Turbo on a system where commands are multiply-defined the compiled code produces a message and a list of re-defined names it needs to use. Type NEW and try again.

"Toolkit commands work after you type NEW. . . this also gets rid of your program and all the variable values."



CLEAR is not sufficient to persuade SuperBasic to release unset names. The interpreter tends to grab memory whenever possible and release it only under extreme circumstances. Resident command definitions over-rule SuperBasic ones in JS and MG versions of the QL, so this bug does not affect later ROMs.

No cursor

Another annoying quirk of the AH and JM ROMs is the way the cursor vanishes after you have finished using a task. Nothing appears on the screen until you type CTRL C to switch to another window.

Later QL ROMs turn on the cursor automatically in the 'next' task window — usually the SuperBasic window zero at the bottom of the screen — when the task which was previous accepting input terminates.

Translation

The JS version of the QL introduced a new command, TRA, which translates or exchanges the codes of characters transmitted through the serial ports automatically. The bad news, according to top Danish software house Dansoft, is that TRA translates values only after it has adjusted the parity of characters, so that character codes greater than 127 may not be translated.

JS ROM key

The JS ROM has another exceptional bug in its handling of the CAPS LOCK key. If you press CTRL and ESC at the same time on most QLs you get character code 128. ESC is not a letter of the alphabet so you would not expect pressing CAPS LOCK to have any effect on the code you get.

A sloppy comparison statement in the JS ROM means that CTRL ESC gives a code 160 if CAP LOCKS is in effect and code 128, as expected, otherwise. This is a very esoteric bug but it is worth noting if you are writing a program and planned to use CTRL ESC as a control keystroke. Code 160 normally is obtained by pressing CTRL SHIFT "2".

● *Simon Goodwin revealed 46 other bugs specific to particular QL ROM versions in the August, 1987 issue of* QL World *and listed 31 bugs in all QL ROM versions in the September issue. This list brings to 88 the number of published ROM bugs. Doubtless there are more, although we must have tracked most of the important ones by now. If you have found others, please let us know.*

BUGS AT LARGE

Tame hacker Simon Goodwin tracks another batch of bugs which afflict all QL systems, plus secret priority levels, and undocumented commands which let you use cursor-control and random access files on any unexpanded QL

In three previous articles I have revealed the results of four years' research into the idiosyncracies of the QL built-in software, the Qdos operating system and the SuperBasic interpreter. To date I have found and explained 88 bugs in the QL system.

This is an update to my original articles, which were published in the August and September, 1987 and June, 1988 issues. Since then I have found 14 new bugs, bringing the total to 102. I have also found more about the Microdrive write-protection fault mentioned in September, 1987.

There is no sure definition of a bug. I have concentrated on quirks of the QL system software which cause apparently correct programs to give unexpected results, or no results at all. In some cases programs which should not work are prone to do bizarre things. I count these as bugs, although arguably they are undocumented features. Some of them are even useful.

All complex systems contain bugs. Hardware manufacturers are curiously shy about admitting them, apparently on the basis that what you do not know will not hurt you. This encourages naive users to place unreasonable reliance on inherently fallible systems. In fact, bugs are rarely a problem if you are forewarned. All most users want to know is how to get the result they need without getting into trouble.

Some system bugs may cause other programs to fail, so I have included technical information to help software developers guard against the most common problems by defensive programming in their own code. Where relevant, I have discussed the implications of the bugs for people intending to compile their programs.

This list deals with idiosyncracies inside the QL. Most of these concern the SuperBasic language or the associated collection of routines called Qdos. Some of the most obscure bugs occur in the second processor software which is programmed on to the 8049 IPC chip rather than the main system ROMs.

All QL programmers write new and original bugs as a matter of course but they are outside the scope of this article. So far as I can tell, all the new bugs occur on every version of the QL. The last five are concerned with the pair of linked QL processors. All the others stem from mistakes in the 48K QL system ROM.

(* Bug 89 — RENUM *)

When you RENUMber a program the QL automatically adjusts references to line-numbers after RESTORE, GO TO and GO SUB commands. Unfortunately it does not alter the parameters of resident procedures which refer to line numbers; they include commands like RUN, SAVE, LIST and EDIT.

If you need to re-number a program with RUN in it, use GO TO [line] instead of RUN [line]. If you write programs which LIST or SAVE parts of themselves, you can not re-number them automatically. You must fix the LIST or SAVE statements. Unfortunately that is just the kind of program it is useful to be able to re-number.

(* Bug 90 — NETWORKING *)

The QL is incapable of writing an empty 'packet' of data over the network. If you try to make it do this, the whole machine locks up. It is easier to run into this bug than you might expect. Imagine you have opened a network file in the usual way, with:

```
OPEN #3, NETO-1
```

Then you realise you do not want to write anything to the file, or have nothing to write yet. Being tidy-minded, you type:

```
CLOSE #3
```

At that point the QL freezes, trying to write a zero-length block. The same thing can happen if you supply the device name NETO-1 to a program which tries to write an empty file.

This bug was found by David Oliver of CST. His fix is not very elegant; the Thor XVI locks up for 30 seconds if you try to write a zero-length block, then normal service resumes, accompanied by an 'Xmit Error' report.

(*Bug 91 — TOP PRIORITIES *)

According to all the published documentation, QL task priorities are numbers between 0 and 127. The higher the priority of a task, compared to that of other tasks running at the time, the larger the share of available processing power used to run that particular task. It transpires that priorities greater than 127 are allowed. The system call MT.PRIOR (TRAP #1, DO=11) accepts higher priority values, in the range 128-255.

Most people set priorities with toolkit commands like the Turbo Toolkit SET PRIORITY or Tony Tebby's SPJOB. SET-PRIORITY follows the book and rejects priorities greater than 127 but SPJOB passes the low byte of any integer value to MT.PRIOR. Thus you can use values above 127.

Negative values work, too. SPJOB 0,0,-1 sets the priority of SuperBasic (task 0,0) to 255 in the JOBS list. Other negative values −2 to −256 correspond to priorities from 254 down to zero.

LIST TASKS in Turbo Toolkit shows non-standard priorities as negative values, whereas the SuperToolkit JOBS command shows them as positive. I tested the new priorities by running three compiled SuperBasic integer print loops at the same time. One ran at a priority of 255 and the others at the default priority, 32. The high-priority task received about five times as much processing time as each of the other two. This matches the results for similar ratios of conventional priorities, like 8, 8 and 63, so it seems that the non-standard values work as an extension of the normal range.

(* Bug 92 — MICRODRIVE FORMATTING *)

Microdrive formatting fails at once with an 'in use' error if any Microdrive is running when you issue the FORMAT command. This can stop SuperBasic unexpectedly if you write to one drive and FORMAT another soon after, while the computer is still verifying files which have just been written.

You also run into trouble if you try to add files to a cartridge and fill it. You cannot re-format the tape to get some more space until the system has finished checking the data it managed to write, even though you have indicated that you want to wipe the cartridge.

To avoid the 'in use' report, check the system variable SV.MDRUN before issuing a format command. PEEK (164078) gives you the number of the currently-running Microdrive, or zero if no drive is in use. The procedure SAFE-

FORMAT, in listing one, will format any device, checking SV.MDRUN if necessary.

(* Bug 93 — SCROLL quirks *)

Explained how the CLS command could recognise undocumented parameter values. Non-standard parameters are accepted and give results which seem bizarre at first but correspond to internal QL system calls.

The SCROLL keyword has a similar but superior feature. SCROLL does much the same thing as CLS but expects a second parameter eight greater than the CLS equivalent. For instance, SCROLL 0,24 has the same effect as CLS 16 which, in turn, works like the documented command POINT 0,0. All three call the system routine SD.POINT.

SCROLL is more useful than CLS because it accepts an extra parameter — the number of pixels to be scrolled. This is passed to Qdos in register D1. Several other calls expect parameters in D1, so we can use SCROLL as an alternative way to pass values to the system. For instance, try:

FOR X=0 TO 255,7 : SCROLL X,17 : PRINT "Hello";

In this case the SCROLL command passes the value of X to SD.SETIN, with results which are interesting but not particularly useful. We can do better.

(* SECRET RANDOM ACCESS *)

Microdrives and discs allow random access to file data. You can wind back and forth through a file, re-reading or re-writing information, with no need to CLOSE and RE-OPEN the file every time you want to move backwards and no need to read intervening information as you move round a file.

This is very useful if you are writing a data-handling program, as it means you can extract data from anywhere in a file without the system having to fetch irrelevant data.

The TRAP #3 keys FS.POSAB and FS.POSRE are recognised by every QL and let machine-coders move the file pointer to any ABsolute or RElative position. There are apparently no SuperBasic commands to move the file pointer; you appear to need a Toolkit command like SET-POSITION.

This is a problem if you want other people to use your SuperBasic. You cannot rely on other users owning a particular toolkit. All toolkits soak some RAM and that is particularly precious on an unexpanded QL.

Packing

Hackers may be amazed to learn that many QL users are still struggling in 128K and they are in particular need of improved file-handling. Would it not be pleasant if we could find a way to use random access, on any QL, without a Toolkit?

It transpires that SCROLL can do the job. SCROLL #3,N%,42 allows random access to a file open on channel 3 — on disc, RAM-disc, Winchester or Microdrive. This calls FS.POSAB, setting the file pointer to the value of N%. SCROLL #3,0,42 re-winds to the start of the file, SCROLL #3,1,42 points after the first character, and so on. Listing two illustrates a simple random access program. It was tested on a JS QL but should work on other models. The first line opens a file and the second fills it with 10 sample lines of data, each seven characters long including the 'enter' code at the end of each line. Then a SCROLL command re-winds the pointer to the start of the file without closing it.

Loops

A REPeat loop is used to read and display each line until the end of the file is reached. The EOF function works well with random access files; you get an 'end of file' error, as you might expect, if you try to set the file pointer to a number greater than the total length of the file.

The final loop lets you select any record by number, using random access to find, read and print the appropriate line from the file. Type 10 to stop the program.

It is feasible to use SCROLL to position the pointer and PRINT new data into the middle of a file. The characters printed over-write the old ones at that position, so it is a good idea to use fixed-length records. The file is extended if you print at the end but you cannot insert characters in the middle of a file without over-writing what was there previously.

Unfortunately, SCROLL expects integer parameters, so you cannot use SCROLL 42 to move more than 32 down a file. SCROLL #3,n,43 might cure this by allowing relative moves with FS.POSRE. In practice it is rejected by the poor checking in the SCROLL keyword code and gives a 'bad parameter' error.

(* Bug 94 — PAN possibilities *)

Once I had investigated the SCROLL and CLS bugs it seemed worth checking PAN in case it allowed access to other system calls. PAN uses the same technique to convert its parameter into a Qdos trapkey but it adds 27 to the parameter value. PAN #c%,0,124 has the same effect as AT #c%,0,0.

All these keywords are meant to handle parameter values between 0 and 4, so they use shared code which checks the value and rejects it if it gives more than 4 when taken modulo 8. SCROLL and CLS add 32 and 40, so they both reject the same values, but PAN adds 27, allowing access to different system routines.

PAN lets you turn cursors on and off without a Toolkit. PAN 0,115 turns the cursor on in the default channel, while PAN 0,116 turns it off again. These instructions are vital when writing compiled multi-tasking programs which use INKEY$ or PAUSE; if a task does not display a cursor it cannot be selected for input with Control C.

To make PAUSE and INKEY$ work correctly, put PAN #0,0,115 at the start of a task. Note that the default channel for PAUSE and INKEY$ is #0; Sinclair did not reveal this.

```
Listing 1 - Safe microdrive formatting.

   DEFine PROCedure SAFE-FORMAT(device$)
   IF LEN(device$) > 4
     IF device$(1 TO 3)=="mdv"
       REPeat poll: IF PEEK(164078)=0 THEN EXIT poll
     END IF
   END IF
   FORMAT device$
   END DEFine SAFE-FORMAT
```

Like SCROLL, PAN passes an extra parameter to Qdos in register D1. We can use this to call FS.POSRE, passing a relative offset for the file pointer. You can access any part of a long file by using SCROLL 42 to get to a known place, then PAN 40 to move fowards or backwards from there. PAN #,N%,40 passes the integer N% to FS.POSRE. N% is the offset from the current position in the file, so:

```
SCROLL #3,30000,42: PAN #3, 20000,40
```

positions the file pointer after the 50,000th byte in a file. Use PAN 40 repeatedly if you need to wind more than 64K down the file.

It would be pleasant to be able to 'truncate' a file, discarding characters after a certain point so that space could be re-used. You may find that PAN #3,0,48 will truncate the file on channel #3 after the current position but this relies on the undocumented system — call FS. TRUNC, TRAP #3, DO=75.

Unfortunately, FS.TRUNC was a real afterthought rather than something which was not documented. Standard QLs do not recognise FS.TRUNC; you need the Sinclair QL Toolkit, SuperToolkit 2 or a disc expansion to make this call work. They include an extra command TRUNCATE, so there is not much point using the PAN version unless you want to be deliberately obscure.

(* Bug 95 — WINDOW parameters *)

WINDOW does not check the number of parameters you pass to it. Quanta members who discovered this bug hoped that the 'undocumented' parameters would allow extra control over windows but this is not the case. The WINDOW code uses only its last four parameters, plus the first one — the optional channel number — if the parameter list starts with a hash. The other values are ignored.

If you put extra parameters accidentally in a WINDOW command it can be difficult to determine what has gone wrong, unless you know about this minor bug. The keyword code could be fixed by adding a check on the value in D3 after calling the parameter-fetching sub-routine.

(* Bug 96 — DLINE channel *)

DLINE allows an optional, undocumented channel parameter. If you put a hash and a channel number between the command and line details you can re-direct the 'automatic listing' which normally appears in channel 2 when the program is changed. It is difficult to imagine how this could be useful.

(* Bug 97 — CHARACTER CODES *)

The CHR$ function is meant to convert a number between 0 and 255 into a character with the corresponding code. In fact, it accepts any integer value from −32768 to 32767 as a valid parameter. The resultant character depends on the value of the bottom eight bits of the integer.

This bug is unlikely to cause problems as it does not affect correct programs; it means that some technically-incorrect programs produce useful results. For instance, consider listing three, a useful snippet of code which compresses an integer value, X%, into a two-character string.

This is useful when packing numbers into fixed-length records in a file. If you did not compress the value and PRINTed it normally, it would occupy between two and seven characters, depending on the value. The packed representation uses a fixed length of two bytes for every value, saving space in most cases and making it easy to skip over values.

Note that the code must handle negative values of X% separately and the second character code must be reduced modulo 256, to ensure that the parameter value never strays from the range 0 to 255. You can manage with faster and simpler code, like this:

```
RETurn CHR$ (X% DIV 256) & CHR$(X%)
```

The first CHR$ expression takes negative values in its stride, because of the bug. The second part does not need the MOD because CHR$ ignores the top eight bits of its parameter.

The TURBO SuperBasic compiler has its own fast code for CHR$, to avoid the need to call the Sinclair slow resident function. This bug is duplicated deliberately in compiled code to preserve compatibility. The parameter of CHR$ is always an integer, so the code is not slowed by the need to handle the bug correctly.

Q-Liberator does not generate its own code if it can use an existing resident routine. In this case it is exactly compatible with the interpreter, because it calls the interpreter routines for every resident command or function it executes.

(* Bug 98 — POKE PARAMETERS *)

The POKE and POKE-W commands have a similar bug to CHR$; again this quirk can sometimes be useful. To be compatible with the ZX Spectrum, the POKE and POKE-W commands let you store signed values as well as unsigned ones. For instance, you can POKE X,-1 or POKE-2 X,65530 even though strictly the parameters of POKE should be in the range 0 to 255, or −32878 to 32767 for POKE-W.

Ignores

All POKE commands accept any 32-bit long integer value as a second parameter without complaint. POKE ignores the top 24 bits of the value and POKE-W ignores the top half. The effect is that, like POKE-L, POKE and POKE-W allow any parameter values in the range plus or minus about two billion. For instance:

```
POKE 131072, 131074
```

stores the value 2 in the first byte of QL display memory. The low byte of the value 131074 is 2 and POKE ignores other bytes.

Once again, Turbo duplicates this bug for compatibility in compiled programs. Turbo code converts both parameters from floating point values, as SuperBasic has no 'long integer' data-type. This

Listing 2 - Random access without a Toolkit.

```
    OPEN-NEW #3,MDV1-TEST
    FOR L=0 TO 9 : PRINT #3;"Line:";L
    SCROLL #3,0,42 : REMark Rewind
    REPeat show
      INPUT #3,A$
      PRINT #3,A$
      IF EOF(#3) : EXIT show
    END REPeat show
    REPeat scan
      INPUT "Enter record No. 0-9:";R
      IF R<>INT(R) OR R<0 OR R>9 : EXIT scan
      SCROLL #3,R*7,42
      INPUT #3,A$
      PRINT #3,A$
    END REPeat scan
    CLOSE #3
```

makes it slower than it would be if POKE and POKE-W worked only with integers but the Turbo code is still much faster than the Sinclair resident POKE routine.

(* Bug 99 — BEEP INTERACTION *)

Very high-pitched notes produced with BEEP interfere with keyboard polling. If you use BEEP 0,0 to generate a continuous tone you will find it difficult to type-in anything else while the tone sounds. BEEP 0,1 is not bad but keystrokes are still lost while the beeping is active. The problem occurs because the keyboard is read by a program in the 8049 second processor.

The same program also generates sounds by sending pulses to the QL squeaker. When the 8049 is making a high-pitched note it does not have sufficient time between clicks to scan the keyboard.

There is no easy fix for this, as the faulty code is buried in the IPC, which includes ROM, RAM and processor all in one chip. NEC makes a user-programmable version of this chip, the 8749HC, but you will need a few specialised tools to disassemble, patch and re-program the IPC. In practice it is much easier to avoid sustained use of pitches 0 and 1.

(* Bug 100 — KEYROW *)

Another IPC bug is concerned with keyboard polling. The QL manual says it is tricky to detect three or more key depressions with the KEYROW function but reassures the reader that SHIFT, ALT and CTRL do not interact misleadingly with other keys. Unfortunately this is not true. Compware programmer Francesco Balena has discovered that the arrow keys can interfere with CTRL and ALT.

If the UP and LEFT arrow keys are pressed at the same time, a common event in games and joystick programs, the CTRL and ALT keys are indistinguishable. Normally KEYROW (7) returns a set bit for each of the keys but if you press UP, LEFT and ALT you get the same KEYROW pattern as for UP, LEFT and CTRL; in either case, both the bits for CTRL and ALT are set, even though only one of them is pressed.

The only way to avoid this problem is not to use diagonal cursor movements in conjunction with CTRL or ALT. This bug is probably a documentation error rather than a real coding mistake, as it would be rather inconsistent if the QL could cope with those key combinations without trouble.

(* Bug 101 — BEEP/KEYROW CRASHES *)

If a task if loaded or unloaded while the SuperBasic interpreter executes a BEEP or a KEYROW instruction, the whole computer is likely to crash. BEEP and KEYROW call MT.IPCOM, passing the address of a parameter table stored on the user A7 stack. This is bad programming because SuperBasic may re-locate the stack in memory at any time if memory is needed for other tasks.

If Basic moves during BEEP (IPC 10/11) or KEYROW (IPC 9) the table address is invalidated and gibberish may be passed to the IPC. In the original Qdos documentation, designer Tony Tebby warned: "IPC communication is completely unprotected. The command must not contain any errors or the entire machine will hang up.

Interpreter

To fix the interpreter you must re-define the BEEP and KEYROW keywords. The machine code could be the same apart from a switch into supervisor mode — freezing multi-tasking — during the IPCOM call. The easiest alternative is to compile the program. This cures the problem because the stack of a compiled program never moves while a task runs.

(* Bug 102 — SERIAL OVER-RUN *)

Chas Dillon and Tony Price report a problem in the handling of serial queues. The second processor can get its pointers in a muddle, when running fast communications programs, if it is asked to do something else at the same time, like generating a sound or recognising a keypress.

Characters are delayed and jumbled inside the IPC, so that they reach the buffers in QL main memory out of order. It is not an easy bug to demonstrate as it depends on precise external timings but it is consistent enough to make fast serial input irritatingly unreliable. The bug occurs even if you use 'handshaking' hardware to regulate the flow of data.

If the IPC is disturbed while reading from a serial port it may lose several characters or introduce a 'lag' so that each new character received causes an earlier character to be passed from the IPC to the main processor.

A proper cure for this bug would involve re-programming the 8049. It might be wiser to circumvent this by building a dedicated QL serial port but that would still take a great deal of hardware and software effort. Alternatively, use a slow data rate like 300 baud or buy a Thor XVI.

Listing 3 - Integer packing with CHR$.

```
IF X%<0
   RETurn CHR$(256+X% DIV 256) & CHR$(X% MOD 256)
ELSE
   RETurn CHR$(X% DIV 256) & CHR$(X% MOD 256)
END IF
```

(* UPDATE — MICRODRIVE WRITE PROTECTION *)

In my September, 1987 bug list I mentioned that write-protecting a Microdrive does not stop the system trying to write to it.

In fact, the QL tries to write the data eight times but each time the low-level code aborts because the tape is write-protected. This means the drive runs for a little more than a minute, then a 'bad or changed medium' message appears.

At the time I suggested that you might cure this problem by using IPC call 1 to test the write-protect status of the currently-turning drive.

I have since tried this and it does not work. IPC call 1 has a bug in it which means it always indicates that the tape can be written-to, even if it is write-protected. This is a really annoying bug, because you cannot re-program Qdos round it.

The only way to avoid spurious bad or changed medium errors is to ensure that you never try to write to a cartridge which has the plastic 'write-protect' tag removed. It is for you to check this, because the computer cannot check for you.

(* FUTURE BUGS *)

This is certainly not a definitive list of QL bugs, although it covers all the problems I have been able to analyse in detail. If you have extra information about these or other QL bugs, please share your discoveries by writing to me, care of *QL World*.

Sinclair QL Preservation Project (SQPP)

On January 12th 1984 Sir Clive Sinclair presented the Sinclair QL Professional Computer in a Hollywood-style launch event at the Intercontinental Hotel, Hyde Park Corner, London. This was exactly 12 days earlier than Steve Jobs presented the Apple Macintosh.

The QL still is a very good example of an innovative, stylish, powerful and underestimated product. On one hand it failed in the market in the long run but on the other it influenced many developments which ended in today's products.

2009 was the year of its 25th anniversary in which month by month new activities were launched.

Sinclair QL 25th anniversary
1984-2009 – 25 years of innovation...

A story of what it takes to innovate...

Urs König (aka cowo)
Director Technologies, PC-WARE Systems (Schweiz) AG

Jan 12th – Congratulation to the QL's 25th birthday. Message spread to VIP, community and media.
http://www.qlvsjaguar.homepage.bluewin.ch/SinclairQL_25th_anniversary_1984_to_2009.html

Check out this 25th anniversary presentation…
http://www.cowo.ch/downloads/SinclairQLis25-compressed.ppt

Try QPC, a virtual QL running under Windows...
http://www.cowo.ch/downloads/QPC_a_virtual_QL.zip

The Mac that never was

Feb 19th – Massive coverage (11 pages) of the QL in the April Issue of Personal Computer World (PCW) magazine.
http://www.pcw.co.uk

New-Sinclair QL £399
There's no comparison chart, because there's no comparison!

Mar 12th – Sinclair QL Preservation Project (SQPP) launched, starting with Documents/Publications from Sinclair Research Ltd and various computer magazines of the years 1984 to 1986.
http://www.qlvsjaguar.homepage.bluewin.ch/SinclairQL_preservation_project.html

QL forever!

Urs König (aka cowo)
http://www.qlvsjaguar.homepage.bluewin.ch