QL Today

Volume 5
Issue 6
March/April
2001

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

Years QL Today

Contents

Advertisers

in alphabetical order

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Im stillen Winkel 12
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
Box1 +49 203 502013
Box2 +49 203 502014
Mobile +49 1701 222277
email: JMerz@j-m-s.com
email: QLToday@j-m-s.com

English office:

Q Branch
20 Locks Hill
Portslade
BN41 2LB
United Kingdom

Tel. +44 1273 386030
Mobile +44 7836 745501
Fax +44 1273 381577
email: qbranch@qbranch.demon.co.uk
email: QLToday@j-m-s.com

Editor:

Dilwyn Jones
41 Bro Emrys
Tal-Y-Bont, Bangor
Gwynedd
United Kingdom LL57 3YT

Tel. +44 1248 354023
Fax +44 1248 354023
email: dilwyn.jones@dj.softnet.co.uk
email: QLToday@J-M-S.com

Co-Editor:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 1708 510764
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL Today is published bi-monthly, our volume begins on beginning of June. Subscriptions begin with the current issue at the time of sign up. Please contact the German or English office for current subscription rates.

We welcome your comments, suggestions and articles. YOU make **QL Today** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email or into one of the JMS-BBS's. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

Article and Advertising deadlines are as follows:

Issue 1: 30 April
Issue 2: 30 June
Issue 3: 30 August
Issue 4: 30 October
Issue 5: 30 December
Issue 6: 28 February

Editorial

Dilwyn Jones

With this issue we have reached 5 years of QL Today. News, cover disks, in depth articles, a lot of change in the QL scene, we have covered them all.

We have entered the 21st century with still a lot of interest in the QL. Back in 1984 (and later when Amstrad bought out Sinclair computers) who could have predicted the QL would still be going strong in 2001 and there would still be a large, active user base - the membership of Quanta and readership of QL Today is testimony to this.

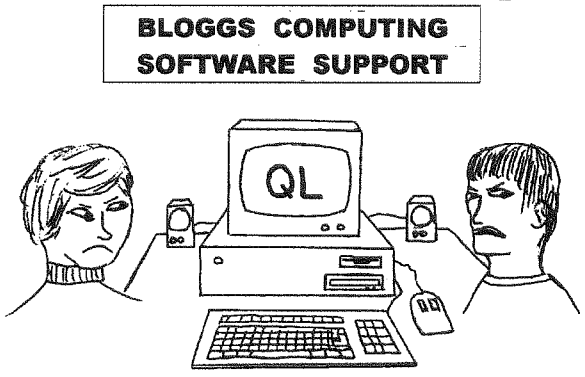
Why stay with the QL? Why throw away nearly 20 years' worth of familiar and reliable technology! Some of us have taken the emulators route (I use QPC2 on a Windows 95 PC) and many take the hardware routes away from the original QL - plenty of Auroras and Q40s about, hopefully there will be a Q60 too! Advancements in QL technology have meant we are bandying terminology like Megabyte memories, Gigabyte hard disks CD-ROM and TCP/IP like there was no tomorrow (that inspired the cartoon!). It may seem like bordering on the excesses of certain other computers in the eyes of some, but the technology is there and affordable so let's use it!

My QL emulators CD-ROM should be out by the time you read this, and it's free (apart from a copying fee, depending where you get it from) and freely copyable. If you have a CD-writer, burn some copies and give them to any ex-QLers you know, or to anyone planning to move to other computers, or even to those who might be into Retro-Computing or who just like tinkering with "something different". It has a large amount of free QL software on it to keep you busy, and has emulators for just about all types of computers (PCs, Amigas, STs, Macs, Linux boxes...) - so do the QL a favour, get a copy and copy it for everyone out there. It won't take the place of that shiny new Q40/Q60 or Goldfire when that arrives, but may just do it's bit for those who wish to move to other hardware platforms.

To quote from the emails of Thierry Godefroy (author of many QL programs):

QDOS/SMS Forever!

Cartoon



"32MB RAM, 16-bit colour, SMSQ/E, IDE 1GB hard disk, Serial mouse, high speed processors, FPU support, CD-ROMs, QL Sampled Sound System, TCP/IP, SuperHermes, High speed serial ports, fast modems, Compswitch, Native File Access.....Wasn't life simple in the days of a good old 128K QL!!!!!"

Time for improvements and adjustments

Jochen Merz

I have probably already mentioned this in one of the previous issues: we tried to keep the price for QL Today stable for as long as possible, but it is absolutely necessary to adjust the price now. Since we started, printing costs have gone up by 26%. Postage up to 44%, depending on where we ship the magazine. UPS shipping to QBranch has got more expensive too, can't tell how much as UPS gets more expensive every year.

So, with the new volume, the price will be slightly higher. It does not compensate for the price increase, but Roy and myself discussed ways to cut costs and time - and, believe it or not, everybody is going to profit from these changes.

But we have to do something. QL Today is not earning any profit, not for the publisher, not for the editors, not for the distributor and not for the authors, but neither the publisher nor the distributor can afford to run it at a loss.

As you know, your subscription can start and end with every issue of every volume. This means, we have to maintain lists which define who's subscription ends when, tick an issue off every time, add renewal forms to those envelopes and so on. This costs a lot of additional time six times a year, when we put the magazines into envelopes. Also, we have to keep track of who renewed with which issue and when people renew later where to start and so on, send reminders (up to three) and spend a lot of unnecessary time and money doing it this way. We thought it would be so much easier if every subscription would end with issue 6 of a volume. We would need to add renewal forms to every envelope of issue 6 only, and we would not need to count, tick issues off lists and so on. And you, the reader, would know that the subscription ends with issue 6 of a volume ... we really hope to cut postage on reminders and save time, and we do ask for your help.

So, you can subscribe at any time, but if you start with issue 3, for example, you pay for issue 3, 4, 5 and 6 to make sure your subscription ends with issue 6, and then you can carry on subscribing for the next full volume.

I hope this makes sense, and I do hope that this optimisation will pay for the fact that we did not adjust the price as much as we had to.

As a special thanks to all our loyal readers we have decided that renewals for the next volume can be made at the old price if the renewal form is returned to us within two weeks of receipt.

The same applies to German customers with automatic renewal - no reminder was ever necessary, so they will benefit from the old price.

NEWS

RWAP Software

from Rich Mellor

The main item of news is that I have moved address - see our new advert for details and the new phone number.

I am still suffering with illness, although it is slowly improving. Therefore, I am now able to do some work on the QL once again.

First up is the Proforma ESC/P2 drivers that have been updated to v1.03 - this fixes a problem in the 360dpi driver which meant that under certain circumstances, it could fail. I hope to finally work out how to program the 1440dpi driver shortly. The price will increase to £10 when this is working, so now is the time to get a copy!

Q-Route v2.00 should also be released shortly, which uses the enhanced colours available under QPC, QXL and Q40 - upgrades will cost £5 each to this new version. Contact us for availability.

Please note that my email address has now changed to:

rwapsoftware2@activemail.co.uk

Please feel free to use this to send any queries or comments on any of our software.

I am also looking into a new print run for the SBASIC/SuperBASIC Reference Manual - if any readers would be interested in purchasing a copy, please send me your name and address.

Website address changes

Jean-Yves Rouffiac writes:

Just to let you know that my page has moved to:

http://website.lineone.net/~jeanyves.rouffiac

U-net wanted me to pay more than £100 for the pleasure of staying with them. LineOne is free. The decision making progress was rather quick!

There is a relocator message at the U-net site, just in case.

Daniel Baum writes:
I have moved my website to another hosting company, which means that it loads **much** faster. This solves the problems people had entering the site.
Tim Swenson has contributed several software reviews, which can be found - obviously - on the reviews page.
The site now has a funky purple 3D logo. I have also cleaned up some of the other graphics to make them more aesthetic.
The site is at **http://www.qldesign.com**

JUST WORDS!

QL-THESAURUS is now version 4.02. This version no longer crashes in high colour mode. Also displays the correct house colours in this mode.
QL-2-PC TRANSFER is now version 3.02. This version removes a bug that prevented the loading of a format file in the older Sinclair ROMs. It also corrects Line Feed problems when transferring ASCII text, and displays the correct house colours in high colour mode.
NEW PRODUCT: Just Words! will be releasing a rhyming dictionary later this year. The program is currently at the beta-test stage, but is not yet ready for release as the data-base, which will eventually grow to about 50,000 words, is still being written.
ESPARANTO: Occasionally Just Words! receives enquiries about Esparanto. We would be interested to hear from any QL-users who are interested in Esparanto to assess the need for a **Q-TYP dictionary.** In particular we would be interested to hear from any user who can supply word lists or dictionaries in QL or PC format.
UK GENERAL ELECTION: My general election analysis program can be downloaded from:
http://members.tripod.co.uk/geoffwicks/election.htm
It is also available from the QUANTA library or directly from me. Please enclose 4 first class stamps to cover costs.
Geoff Wicks, 28 Ravensdale, Basildon, Essex SS16 5HU, UK. Tel: +44 (0)1268 281826
geoffwicks@hotmail.com

News from Tim Swenson

Better BASIC and XREF
I've taken BetterBasic and CrossRef sources from the web page where Chas Dillon has made them PD, compiled them, found a manual for CrossRef, and then made them into a package. This will make them available for the QL community. If there are any bugs in the compiling, I'll fix them. As for bugs in the actual program, I'll have to think about fixing them (the programs are not trivial).
Also, issues #2 and #3 of the Q40/Linux Journal are out and available on my web page.
http://www.geocities.com/SiliconValley/ Pines/5865/

Turbo News

Turbo Compiler and Turbo Toolkit have now been updated by George Gwilt.
Turbo version 4.7 and Turbo Toolkit version 3.28 both now available from the Other Software Page on my website (and presumably from any PD libraries who may have downloaded the packages):
http://www.soft.net.uk/dj/software/other/ other.html

PCBCAD Software V6.03

Version 6.03 of Lear PCBCad is now on my website. This program now contains experimental Q40 hi-colour line drawing routines, but as the author Malcolm Lear does not have a Q40 to test it on, he'd appreciate feedback from anyone who tries it on a Q40 or any hi-colour platform. Gerber RS274X import AND export facility has been added since a recent version.
It may be downloaded from the Other Software Page on my website:
http://www.soft.net.uk/dj/software/other/html

Dilwyn Jones Website Mirror

Phoebus Dokos has set up a mirror of my website in the USA. This should give faster access for north American QLers, especially at peak periods. I have tried to ensure that all the links from page to page and to graphics on the website are relative URLs, so you shouldn't find yourself skipping from site to site when you follow links from page to page.
A preliminary speed test from Pennsylvania (by Phoebus Dokos) showed a speed improvement of about 25-30% over the regular DJ site (UK). For broadband connections the improvement is minimal but for modem users it's substantial.
The pages will be updated twice a week.
The url of the mirror site is:
http://ql2k.cjb.net/ (Cloaked) -or-
http://www.ql2k.50megs.com (Uncloaked)
Phoebus has also indulged in some graphic design (I think that's his occupation?) and contributed a new suggestion for a QL logo, and I must admit I like it. Maintains the red, white and black windows of a typical QL monitor screen,

but in a rather nice layout which should not infringe on a certain clothing manufacturer's logo discussed recently on the QL Users mailing list. Some examples of proposed QL logos can be found on my website.
http://www.soft.net.uk/dj/index.html

QL Sampled Sound System

The QLSSS (Sampled Sound System) version 2d from Simon Goodwin and Mark Swift for QDOS Classic (Amiga and Q40) and SMSQ/E (Q40) featured in a recent QL Today article is now available from my web site.
This system is based on a sound replay device driver, capable of handling mono or stereo sampled sounds, at sample rates of up to 20,000 samples per second. Being a device driver, sounds can be replayed by simply COPYing a file to the device, for example. The system uses Q40-style _ub (unsigned byte) sound sample files.
Disk 1 contains the device driver, instructions, sources, some SuperBASIC programs and converters plus some sound samples used in the example programs.
Disks 2 to 7 (yes, the entire system comes as 6 DD disks and 1 HD disk's worth of files!) contains various sampled sound file examples. Beware, disk 7 contains a zipped 1.5MB _ub file, this is too large for DD or HD disks - it will need to be unzipped to an ED disk or to hard disk.
CAUTION: Allow plenty of cheap rate download time to download the whole lot - a massive download if you have a fairly slow modem like mine.
The files may be found on my Other Software Page.
http://www.soft.net.uk/dj/software/other/other.html

GWASL V1.6

I have now added version 1.6 of the Gwasl (Gwass Assembler Lite) assembler program from George Gwilt to my website. This version includes the symbols file and instructions missing from earlier versions. This 68008 assembler is the preferred assembler for use with Norman Dunbar's Assembler Programming series in QL Today. Versions 1.5 and 1.6 have corrected minor bugs in the ‹ea› instruction modes.
The package is on the Other Software Page, or my be downloaded directly from
http://www.soft.net.uk/dj/software/other/gwasl16.zip

Sprite Editor

As requested in the last QLToday, it is now possible to save binary from sprted.
So, now, there is no excuse to not provide a lot of PE program with a lot of nice sprites... especially not, as it can be found on the cover disk
Jerome Grimbert

New QL Browser

from Tarquin Mills

HyperBrowser is a new web browser for the QL. Until the final release of soql it can currently only view and convert local html files, having the ability to convert xhtml files to plain text. The first version only displayed text and tables, but this release has added the facility to support images such as JPEG too, using Dave Westbury's graphics viewer, Photon.
The program may be found on the web at:
http://www.planet14.sonow.com/comp/hb/
This version can handle much larger XHTML files (files larger than memory), the Easter Egg (a hidden fun feature) now has word wrap (has anybody found it).
The time from asking for a XHTML file, to the start it being rendered has more than halved. Also there have been several smaller changes.

As usual, we received more news way after the deadline. As we started on the layout early, you will find them at the end of the magazine!

Do-It-Yourself Lexicography

Geoff Wicks

Last year the Germans caught up with the Dutch when they revised their spelling. English speakers find the idea of committees of learned men and women spending hours deciding how their language should be spelt rather cute. This is probably just as well. Can you imagine a committee of British, Americans and Australians trying to reach agreement on the spelling of every English word?

I know nothing about how Germany went about its task or the public reaction, but I know what happened when the Netherlands and Belgium did it in 1995. The spelling revision was meant to end interminable discussions about whether electricity was "elektriciteit" or "electriciteit", or whether we eat "chicken soup"

(kippesoep) or "chickens soup" (kippensoep). In practice, after the revision everyone carried on as before, and the new spelling was only compulsory for civil servants and schoolteachers. In the liberal Netherlands people in these professions are strictly forbidden to have sex, although they may indulge in something called seks. If, however, they need a little extra stimulation, they can still pop down to the local sexshop, because that is an English word.

When the new spelling came into force the QTYP and Spellchecker dictionaries that were circulating in the Netherlands became outdated overnight. Our colleagues in the PC world did not have this problem, because there was a market for commercial dictionaries giving the new spelling. But we QLers are well used to doing it ourselves, and in practice it was easier to revise our dictionaries than you may think. Indeed, if you have a reasonable knowledge of a language, it is possible to quickly write a useable QTYP dictionary. And if you are prepared to cheat, OCR technology has made the task much easier.

At the QL2000 show I had an interesting conversation with Dietrich Buder, who is compiling a list of German words. He estimates a complete list would contain about 600,000 words. Fortunately for a Spell checker you need much less than this. I believe the first Spell Checker for the QL had only 15,000 words, although later there was a Jumbo dictionary of 30,000. In practice 50,000 is a good size for an English dictionary, although some languages, such as French and German need more.

If you obtained your copy of QTYP with a word processor or text editor, you will probably not have a QTYP manual. The Text87 manual gives some helpful information on using QTYP, but it is easy to miss this when there are so many other things to learn about the program. To use the spell checker in Text87, you press F6 and are asked for the name of the QTYP dictionary and those of the word lists. These are the clues about how to make your own dictionary.

Spell checkers are "off the peg" products and not "tailor-made". You cannot expect them to contain your name and those of your family and friends; addresses you use regularly; or specialist words you need for your profession or study. Most spell checkers allow you to make a separate list of these. Every time you come across a word that is not in the dictionary, you can save it to a temporary file in memory, and later save this file to disk. In Text87 the key presses are F3, Config, Spelling, Save. (On the command line you will see there is also a Load command.) These facilities enable you to adapt the spell checker to your own needs.

One snag is that text lists can take up a lot of memory. If you have a big list, it is better to merge it with the main dictionary. This is easy with QTYP. On the Text87 disk or QTYP disk you will find a file called "QTYP_ded". You can execute this by a simple line:

```
EX flp1_qtyp_ded
```

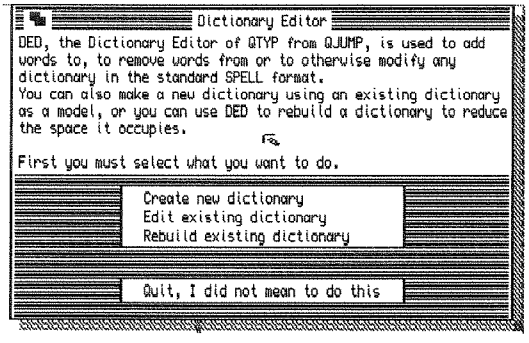
or equivalent.

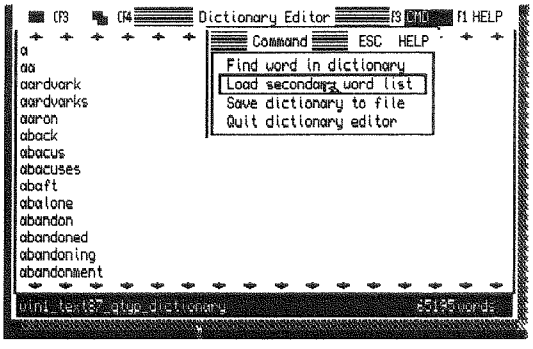
When QTYP-Ded has loaded, you have a choice of three things:

Create new dictionary
Edit existing dictionary
Rebuild existing dictionary

Let us start by editing an existing dictionary. All we do is enter the file name of the dictionary. After loading it appears in a window with an active cursor. We can now scroll through the dictionary and edit it as required. If we press F3 a small command window opens which allows us to find any word in the dictionary, load a secondary word list, save the dictionary or quit the program.

If we choose the second option and load the word list of our specialist words, we see that is does not load instantly, but is loaded in small sections. QTYP is not just loading the word list. It is looking through it word by word, checking for superfluous characters such as punctuation marks, making sure the word is not already in the dictionary and putting it in the correct alphabetical place.





If we press F3 we see the command window now has different options. One of these is to merge the word list into the dictionary. Do this and then press F3 again and the original commands are back. Choose save dictionary to file, enter the file name for the saved dictionary and press enter. You are now asked whether you wish to compress the dictionary. Use "yes" because the "no" option does not produce, as you might think, a plain text list. The dictionary is compressed by letter of the alphabet and then saved.

Adding your word list to the existing dictionary is a simple and painless process.

We can now go a stage further and create a completely new dictionary. When we choose this option, something rather puzzling happens. You are first asked to load an existing dictionary. This is to give the program a few rules about how the dictionary is to be compressed, and any QTYP dictionary will do. This time when you have loaded the dictionary the screen remains blank. Press F3 for the command window and load the word list you have created for the new dictionary. Again this will load in sections. Now press F3 and choose the merge option. Press F3 again, and save the dictionary. Once again you can choose between a compressed and an uncompressed dictionary. On this occasion if you enter "no" the dictionary will be saved as a text file, but usually it is better to compress it.

We now have a new dictionary, but it will not yet be in its optimal form. The compression techniques were optimised for another dictionary and usually another language. QTYP has a routine to "tailor-make" the compression for each dictionary and language. This can take some time. The manuals suggest some hours, but do not let this put you off. The manuals pre-date the Gold Card and QL's are now a lot faster.

This time we choose the dictionary rebuild option, load our dictionary into it, give QTYP a file name for a temporary log file (e.g. ram1_temp) and then go and make the coffee.

I used this routine on a 52,000 word Dutch dictionary with QTYP working in the background while I was typing this article. The original dictionary was 170,478 bytes long, and QTYP rebuilt this to just 143,885 bytes. On my QXL2 system an English dictionary of 25,000 words took 10 minutes to rebuild and a Dutch dictionary of 70,000 words 40 minutes.

Now the snags. Making the dictionary in QTYP is the easy bit. The real work is in the compilation and checking of the word list from which the dictionary is made. This is not too difficult if you are just adding words to an existing dictionary, but if you are compiling a completely new dictionary, especially if it is not in your native language, you have a big task on your hands. This is where I advise you to cheat.

If you have access to a PC, check your word list using the PC's spell checker. When PC word processors were sold on disk, you had to buy language modules separately. Now they are sold on CD's and you will usually find several modules on the disk, from which you can install the spell checker, thesaurus and style checker for several languages on your system.

Just Words! has word lists of several languages that could form the basis of a spell checker, although you should check the accuracy of these before use. They are available for the cost of disk and postage or free of charge if you have email. If you have to build up your own word list, it is a good idea to start by typing in the words from a basic course in the language and make a small dictionary from them. Use this to spell check about 10,000 to 20,000 words of text, and then save and check the extracted words to increase the size of the dictionary. Another possibility would be to make a list of words by using a short basic routine to convert all spaces in an ascii text to line feeds (Chr$(10)). If you have access to an OCR reader you could scan texts for new words.

Using these techniques you may get some bogus words, so always control for accuracy, preferably using a PC spell checker, before adding to your dictionary. Do not worry if your dictionary does not grow quickly. Most of us use only about 20,000 words regularly, and this number of words will be adequate at first. Remember you can always write, phone or email Just Words! for help and advice. I like to keep in touch with what QL-users are doing.

Recently I have had some interesting contacts with users of my QL-2-PC Transfer program. This is optimised to convert to the Windows International character set, which covers most languages spoken by QL users. I was pleasantly surprised to hear from one purchaser that he is successfully

using it to transfer Czech language texts. Czech has some characters that are not in the QL character set and he uses a "hacked" version of Quill. For those of you interested in Bible texts I am also in contact with George Morris over the possibility of producing versions of QL-2-PC Transfer to transfer Greek and Hebrew texts. It should also be possible to produce a version to transfer Cyrillic texts.

George kindly sent me a CD-ROM of public domain language programs, and I have been looking at some PC translation software. In the QL world we sometimes get jealous of PC software, but frankly I was disappointed with the standard of many of the programs on the disk. At most they are only useful for beginners learning the language.

I have said this many times before, but, in a sense, our smallness is our strength. It is often possible to respond to the individual needs of QL-users, and a program does not have to have huge sales to be a viable proposition provided advertising and show costs can be subsidised by other programs.

One program on the disk that did fascinate me was a Japanese word processor. Japanese has two alphabets, Hiragana and Katakana, and also makes regular use of about 2,000 Chinese characters (Kanji). Just think of the problem of typing Japanese using a UK keyboard and then displaying the result in Chinese characters! The program solved this in an ingenious way. You type in romanised Japanese and this appears on the screen as Hiragana. If a word is recognised it is also shown as Katakana if it is a "foreign" word, or Kanji if it is a Japanese word.

Why do I like this program? Because it is different. It shows the individuality and creative thinking I associate with the QL, but miss in most PC software.

Gee Graphics! (on the QL ?) - part 21

H L Schaaf

Gaussian Elimination Now we come to a third direct way of solving simultaneous linear algebraic equations. I've enjoyed many distractions as I was looking into this and will share them with you.

In a classic 1977 reference (cited in the listing) there are algorithms in Fortran called DECOMP and SOLVE. I've tried to rewrite them for the QL, leaving in most of the comments and the GOTO's. See the Listing "GaussianElimination_bas"

You will notice a concept called 'condition number'. This was of concern to A.M. Turing and John von Neumann in the early days (1947-48) of using computers to solve matrices. To arrive at a condition number they investigated the so-called 'norm' of a matrix. I find there can be any number of 'norms'. Fortunately the algorithm we use employs a rather simple 1-norm. And the algorithm claims only to give a 'conservative estimate' of the condition number. The estimate is a guide as to the stability of the solution, so we can use it to get a feel as to how much we might rely on the answers given. It is a nice touch that makes you feel comfortable or else warns you to be suspicious. In addition there is a neat way to find the determinant of the matrix, another nice touch. Also, once the initial matrix is 'decomposed', we can 'solve' for many other vectors representing the right hand side of the equations.

The 'decomposition' is a way of factoring the original matrix into two "triangular" matrices. (Ahah! that answers the question from GG#18). There is a Lower Left triangle referred to as L, and an Upper right triangle referred to as U. So this gets called a LU decomposition. The trick is that the combination of L, U, and a Pivoting Permutation P is equivalent to the original matrix.

Gauss probably did something of this sort in 1801, when he astounded everyone by accurately predicting the orbit of Ceres, a feat which involved over 80 variables in three different coordinate systems. In 1809 Gauss sketched out his method, and in 1810 published a detailed account.

In 1878 Myrick Doolittle of the US Coast & Geodetic Survey made a combination of improvements to Gauss' method and described a way of handling geodetic observations for precise triangulations, such as a baseline from Kent Island, Maryland to Atlanta, Georgia. In effect he gave a method that produced the LU decomposition. This was thenceforth known as the 'Doolittle method'. In 1924 and 1926 R. Cholesky (France) and T. Rubin (Sweden) made modifications to the Doolittle method and that method is identified as the Cholesky-Rubin method.

Doolittle is known for this quotation:
M.H. Doolittle (1887)
"Having given the number of instances respectively in which things are thus and so, in which they are thus and not so, in which they are so and not thus, and in which they are neither thus nor so, it is required to eliminate the general quantitative relativity inhering to the mere thingness of the things, and to determine the special quantitative relativity subsisting between the thusness and the soness of the things."

GaussianElimination_bas will first check the example given by Forsythe, Malcolm and Moler where the answers are 0, -1, and 1. Then we repeat the examples from GG#19 and GG#20. Finally we do a series of Hilbert matrices. Larger Hilbert matrices are known to be ill-conditioned and I've included enough so you can see what happens.

As for speed, I find it takes only a few minutes to do a 100 by 100 matrix with Gaussian elimination, and I get the condition and determinant thrown in as well. The Inverse approach would take several hours, and the Cramer approach would take several weeks! (Based on extrapolations from smaller matrices.)

I find there have been a great many algorithms developed for working with matrices including one known as the "QL" algorithm in 1958. I believe the Q refers to a rotational matrix and the L to a Lower Left triangular matrix. It is similar to a more widely known QR algorithm. More things for me to explore!

Another resource for matrices and the QL can be found in Rich Mellor's "SBASIC/SuperBASIC Reference Manual". There you will find many commands and functions for dealing with matrices. Look for names that begin with MAT and try out the Math Package.

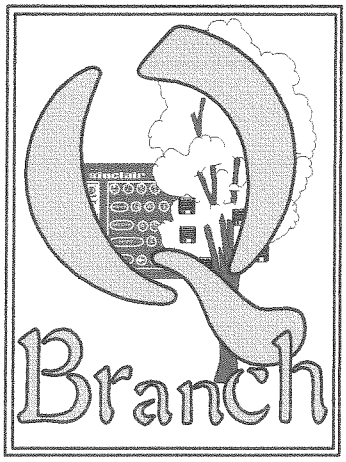
```
100 REMark Listing GaussianElimination_bas
110 REMark to go with GG#21
120 REMark HL Schaaf Feb 12, 2001
130 REMark Gaussian elimination with condition number estimate
140 REMark Ref: "Computer Methods for Mathematical Computations"
150 REMark by George A Forsythe, Michael A Malcolm & Cleve B Moler
160 REMark Prentice-Hall 1977  ISBN-0-13-165332-6
170 WTV : CLEAR
180 :
190 REMark using example from Forsythe
200 DATA 3    : REMark number of simultaneous equations
210 DATA 10, -7, 0, -3, 2, 6,  5, -1, 5
220 DATA 7, 4, 6
230 RESTORE 200
240 READ N
250 DIM A(N,N)
260 FOR I = 1 TO N
270   FOR J = 1 TO N
280      READ A(I,J)
290   END FOR J
300 END FOR I
310 :
320 DECOMP  A
330 PRINT \"Matrix A from Forsythe's example"
340 advise_condition
350 PRINT "Determinant of A is ";Determinant (A)
360 :
370 DIM B(N)
380 FOR I = 1 TO N
390   READ B(I)
400 END FOR I
410 :
420 PRINT \"The vector of constants is",
430 PRINT B(1 TO),
440 SOLVE A,B
450 :
460 PRINT\"The solution is ",,B(1 TO),
470 PAUSE
480 :
490 REMark examples from Cramer_bas and Sadler_bas (in GG#19 and 20)
500 DATA 3
510 DATA 4, 7, -2,  3, -5, 4,  1, 4, 3
520 DATA 19, 11, 29
530 RESTORE 500
540 READ N
550 DIM U(N,N)
560 FOR I = 1 TO N
570   FOR J = 1 TO N
```

```
580     READ U(I,J)
590   END FOR J
600 END FOR I
610 :
620 DECOMP U
630 PRINT \\"Matrix U"
640 advise_condition
650 PRINT "Determinant of U is ";Determinant (U)
660 :
670 DIM C(N)
680 FOR I = 1 TO N
690   READ C(I)
700 END FOR I
710 PRINT \"if the vector of constants is",C(1 TO),
720 SOLVE U,C
730 PRINT \"then the solution is",,C(1 TO),
740 PAUSE
750 :
760 REMark now do another 2 vectors of constants
770 REMark using the same matrix of coefficients
780 DATA 12, 5, 18,  24, 3, 14
790 FOR Vector = 1,2
800   FOR I = 1 TO N
810     READ C(I)
820   END FOR I
830   PRINT \\"if the vector of constants is",C(1 TO),
840   SOLVE U, C
850   PRINT \"then the solution is",,C(1 TO),
860   PAUSE
870 END FOR Vector
880 :
890 REMark now a series of Hilbert Matrices ?
900 CLS
910 PRINT \"A series of Hilbert matrices follow"
920 PRINT "The solutions should all be 1"
930 FOR N = 2 TO 8
940 DIM Hil(N,N) : DIM Hc(N)
950 FOR I = 1 TO N
960   FOR J = 1 TO N
970     Hil(I,J)=1/(I+J-1)
980     Hc(I) = Hc(I) + Hil(I,J)
990   END FOR J
1000 END FOR I
1010 DECOMP Hil
1020 PRINT \\"Hilbert matrix of order ";N
1030 advise_condition
1040 PRINT "Determinant is ";Determinant(Hil)
1050 PRINT \"The vector of constants is"\Hc(1 TO),
1060 SOLVE Hil, Hc
1070 PRINT \"the solution vector is"\Hc(1 TO),
1080 PAUSE
1090 END FOR N
1100 :
1110 ::::::::::::::::::::::::::::::::::::
1120 DEFine PROCedure DECOMP (Matrix)
1130 LOCal I, J, K, T, M, EK, KB, N
1140 N = DIMN(Matrix)
1150 DIM WORK(N) : DIM IPVT(N)
1160 REMark decomposes matrix by Gaussian elimination
1170 REMark and estimates condition of the matrix
1180 REMark use SOLVE to compute solutions to linear systems
1190 :
1200 REMark INPUT matrix A will be triangularized into L and U
1210 REMark order of matrix is N
1220 :
1230 REMark OUTPUT matrix A is changed so as to contain
1240 REMark an upper triangular matrix U and
1250 REMark a permuted version of a lower triangular matrix 1-L
1260 REMark so that Permutation matrix * A = L * U
1270 :
1280 REMark COND is an estimate of the condition of A for the
1290 REMark linear system  A * X = B, where changes in A and B may
1300 REMark cause changes COND times as large in X
1310 :
1320 REMark IPVT is the pivot vector
1330 REMark IPVT(K) = the index of the K-th pivot row
1340 REMark IPVT(N) =  (-1)^number of interchanges
1350 :
```

```
1360 REMark determinant, |A| = IPVT(n)*a(i,i)* . . . for i = 1 to n
1370 :
1380 IPVT(N) = 1
1390 IF (N=1) :GO TO 2200
1400 REMark compute 1-norm of A = Maximum Absolute Column Sum
1410 ANORM = 0
1420 FOR J = 1 TO N
1430   T = 0
1440   FOR I = 1 TO N
1450     T = T + ABS(Matrix(I,J))
1460   END FOR I
1470   IF (T › ANORM) : ANORM = T
1480 END FOR J
1490 :
1500 REMark Gaussian elimination with partial pivoting
1510 FOR K = 1 TO N - 1
1520 REMark find pivot row
1530   M = K
1540   FOR I = K + 1 TO N
1550     IF (ABS(Matrix(I,K)) › ABS(Matrix(M,K))) : M = I
1560   END FOR I
1570   IPVT(K) = M
1580   IF (M ‹› K) : IPVT(N) = -IPVT(N)
1590   T = Matrix(M,K) : Matrix(M,K) = Matrix(K,K) :  Matrix(K,K) = T
1600 REMark skip step if pivot is zero
1610   IF (T = 0) : NEXT K
1620 REMark compute multipliers
1630   FOR I = K + 1 TO N
1640     Matrix(I,K) = -Matrix(I,K) / T
1650   END FOR I
1660 REMark interchange and eliminate by columns
1670   FOR J = K + 1 TO N
1680     T = Matrix(M,J) : Matrix(M,J) = Matrix(K,J) : Matrix(K,J) = T
1690     IF (T = 0) : END FOR J
1700     FOR I = K + 1 TO N
1710       Matrix(I,J) = Matrix(I,J) + Matrix(I,K) * T
1720     END FOR I
1730   END FOR J
1740 END FOR K
1750 :
1760 REMark Condition estimate obtained by solving
1770 REMark two systems of equations
1780 REMark A-transpose*Y = E and A*Z = Y
1790 REMark E either +1 or -1 chosen to cause growth in Y
1800 :
1810 FOR K = 1 TO N
1820   T = 0
1830   IF (K = 1) : GO TO 1870
1840   FOR I = 1 TO K - 1
1850     T  = T + Matrix(I,K) * WORK(I)
1860   END FOR I
1870   EK  = 1
1880   IF (T ‹ 0) : EK = -1
1890   IF (Matrix(K,K) = 0) : GO TO 2240
1900   WORK(K) = -(EK+T) / Matrix(K,K)
1910 END FOR K
1920 FOR KB = 1 TO N - 1
1930   K = N - KB
1940   T = 0
1950   FOR I = K + 1 TO N
1960     T = T + Matrix(I,K) * WORK(K)
1970   END FOR I
1980   WORK(K) = T
1990   M = IPVT(K)
2000   IF (M = K) : END FOR KB
2010   T = WORK(M) : WORK(M) = WORK(K) : WORK(K) = T
2020 END FOR KB
2030 :
2040 YNORM = 0
2050 FOR I = 1 TO N
2060   YNORM = YNORM + ABS(WORK(I))
2070 END FOR I
2080 :
2090 SOLVE  Matrix, WORK
2100 :
2110 ZNORM = 0
2120 FOR I = 1 TO N
2130   ZNORM = ZNORM + ABS(WORK(I))
```

```
2140 END FOR I
2150 :
2160 COND = (ANORM * ZNORM) / YNORM
2170 IF (COND < 1) : COND = 1
2180 GO TO 2250
2190 :
2200 COND  = 1
2210 IF (Matrix(1,1)) : GO TO 2250
2220 :
2230 REMark exact singularity ?
2240 COND = 1E32
2250 END DEFine DECOMP
2260 :
2270 ::::::::::::::::::::::::::::::::::::::
2280 DEFine PROCedure SOLVE (Co_eff, Cn_st)
2290 LOCal I, K, M, T, KB, N
2300 N = DIMN(Co_eff)
2310 REMark INPUT is matrix A, triangularized by DECOMP, and
2320 REMark the right hand side vector of constants, B
2330 :
2340 REMark OUTPUT (overwrites B) is the solution vector X
2350 :
2360 REMark solution of linear system  A * X = B
2370 REMark should not use if DECOMP detected singularity
2380 REMark finds Solution X given Coefficients of A and Constants B
2390 REMark uses IPVT, L and U built during DECOMP,
2400 REMark where L and U have replaced  A,
2410 REMark B will be overwritten by X
2420 REMark SOLVE can be used to solve vectors other than B
2430 :
2440 REMark forward elimination
2450 IF (N = 1) : GO TO 2630
2460 FOR K = 1 TO N -1
2470   M = IPVT(K)
2480   T = Cn_st(M) : Cn_st(M) = Cn_st(K) : Cn_st(K) = T
2490   FOR I = K + 1 TO N
2500     Cn_st(I) = Cn_st(I) + Co_eff(I,K) * T
2510   END FOR I
2520 END FOR K
2530 :
2540 REMark back substitution
2550 FOR KB = 1 TO N - 1
2560   K = N - KB + 1
2570   Cn_st(K) = Cn_st(K) / Co_eff(K,K)
2580   T = -Cn_st(K)
2590   FOR I = 1 TO N - KB
2600     Cn_st(I) = Cn_st(I) + Co_eff(I,K) * T
2610   END FOR I
2620 END FOR KB
2630 Cn_st(1) = Cn_st(1) / Co_eff(1,1)
2640 END DEFine SOLVE
2650 :
2660 ::::::::::::::::::::::::::::::::
2670 DEFine PROCedure advise_condition
2680 PRINT \"An estimated condition number for this matrix is ";COND
2690 sig_digits = 9 - INT(LOG10(COND))
2700 PRINT "about ";sig_digits;" digits are probably OK "
2710 CONDP1 = COND + 1
2720 IF (CONDP1 = COND) THEN
2730   PRINT "Matrix is singular to working precision"
2740   PRINT "results of calculations will be dubious"
2750 END IF
2760 END DEFine
2770 :
2780 ::::::::::::::::::::::::::::::::::
2790 DEFine FuNction Determinant (Matrix)
2800 LOCal N
2810 N = DIMN(Matrix)
2820 REMark finding the determinant is convenient
2830 DETERMN = IPVT(N)
2840 FOR I = 1 TO N
2850   DETERMN = DETERMN * Matrix(I,I)
2860 END FOR I
2870 IF NOT(DETERMN) : PRINT "Matrix is singular" : STOP
2880 RETurn DETERMN
2890 END DEFine
2900 :
2910 REMark end of listing GaussianElimination_bas
```

Q Branch



PROGRAMMING

QD 98	£ 45.00
QD + QBasic	£ 59.00
QD + Qliberator + QBasic	£ 100.00
Qliberator	£ 50.00
Master Spy v 3.3	£ 30.00
QPTR	£ 30.00
Easyptr pt 1 & 2 (together)	£ 30.00
Easyptr pt 3 (C library)	£ 14.00
QMake	£ 15.00
QMon / JMon	£ 22.00
Basic Linker	£ 19.00
DISA 3	£ 31.00
QMenu	£ 14.00

Text 87

£ 79.00

Typset 94 £ 29.00
Fountext 94 £ 39.00
2488 drivers £ 29.00
Epson ESC/P2 drivers £ 26.00

Text 87 is the only QDOS / SMSQ wordprocessor capable of handliing the full screen on the Aurora / QXL / QPC systems. New drivers are currently being written.

Q s Spring Sale

Q Branch still have some second user hardware for sale at low prices so now is the time to snap it up before this year s shows.

We have :

3 full superHermes at £ 65.00 each
1 2Mb romDisq £ 25.00
1 8Mb romDisq £ 70.00
1 QXL II 8Mb £ 110.00
4 Qplane powered backplanes £ 4.00

There may be a couple of Gold Cards available too so call or email for details.

We hope to see you all at a show during the coming months.

'Just Words' by Geoff Wicks

THESAURUS, STYLE CHECK

£ 10.00 ea / ANY 2 PROGRAMS £ 18.00 / ALL 3 PROGRAMS £ 25.00
(Includes Pointer and non-pointer driven versions)
(P.E. versions need Hot_rext, WMAN and PTR_GEN or SMSQ/E to run)
Upgrades from previous versions £ 2.50 + S.A.E. New Manuals £ 1.50

QL2PC

Convert text files from QL to PC formats and much more !

Only £ 10.00 Now with HTML support !

Spelling Crib : PD program £ 1.50 +SAE
or Free if you buy all three programs

UTILITIES

FiFi 2	£ 18.00
QSup	£ 28.00
QSpread v2.04	£ 48.00
Cueshell 2	£ 15.00
Qload / Qref	£ 15.00
Disk Mate 5	£ 16.50
QPAC 1	£ 20.00
QPAC 2	£ 40.00
QTYP 2	£ 30.00
QLQ	£ 28.00

We are currently out of stock of the SuperBasic Reference Manual Place your order now to get one as soon as it is reprinted

The SBASIC / SuperBASIC Reference Manual

The complete definitive guide to BASIC programming in QDOS / SMSQ including three disks of PD toolkits, example procedures and an electronic index.
compiled by Rich Mellor, Franz Hermann and Peter Jaeger

Over 500 pages !
£ 40.00
+ postage

Tel +44 (0) 1273-386030 fax +44 (0) 1273-381577
Mobile +44 (0) 7836-745501
email qbranch@qbranch.demon.co.uk web : http://www.qbranch.demon.co.uk

Q Branch

Feeling out on a limb ?
Reach out for Q Branch.
Suppliers of Quality QDOS/SMSQ products
Hardware and Software.

20 LOCKS HILL, PORTSLADE,
E. SUSSEX. BN41 2LB. UK.

Please Note our new address and phone numbers

ProWesS

ProWesS (now free !)	£ 1.60
DATAdesign	£ 20.00
Fontutils	£ 28.00
File Search	£ 11.00
PFlist	£ 11.00
Dilwyn's Fontpack	£ Call
LINEdesign v 2.16	£ 22.00
PWfile	£ 17.50

Paragraph

The ProWesS word processor

¶ Demo version £ 1.50 + postage
Full Registered version £ 18.00

Version 2.03 available now !

Hardware

We have a small stock of second user items. Auroras / Qubides / Gold Cards / Qplanes / superHermes etc. call us to get details of the items available. These are going fast so call soon.

QXL II	**£ 100.00**
Recycled superHermes	£ 65.00 *
Recycled Gold Card	£ 50.00 *
Recycled Aurora	£ 75.00 *
Qubide	£ 55.00
Qplane	£ 25.00
Aurora cables	£ 3.00
Aurora rom adaptor	£ 3.00
'Arfa Braquet'	£ 8.00
'Son of Braquet'	£ 18.00
The 'Braquet'	£ 16.00
MC plate	£ 6.50

* when available.

14" and 15" monitors for the Aurora - Call.

SMSQ/E

Gold Card / Atari / QXL Version

£ 69.00

Various Atari versions : call for details

QPC 2 full colour version!

£ 82.00 (£ 67.00 SMSQ/E Owners)

Upgrades from QPC 1 £ 35.00 (return master disk)
Upgrades from QPC 2 v1 £ 30.00
Special offer !

Get Cueshell for only £15.00 with any copy of SMSQ/E or QPC 2 upgrade !

Q Branch Programs

The Knight Safe 3	£ 35.00
upgrades from previous versions	£ 5.00
Q - Route v1.08C Route finding programme	£ 25.00
Q - Count Pointer driven home accounting	£ 25.00

The Fractal Collection !

This is a brand new program which will produce stunning animated fractal patterns. It will run on anything from a Gold Card to the Q40 and will be capable of using the power of the colour drivers when they are released. Complete with many example files and routines to design your own screens.

Only £ 35.00

QPC 2 v2 FULL VERSION AVAILABLE NOW

Qubide upgrades to version 2.01 £ 8.00

We can accept payment by VISA, Mastercard and Switch. You can also pay by Eurocheques made out in Sterling or a Sterling cheque drawn on a UK Bank. Prices include Post and Packing in Europe.

Adding To Your QL - 2

Dilwyn Jones

Operating Systems (OS)

Way back in 1984, it was simple. You had QDOS in a few slightly different ROM versions and that was it. Oh, sure, you could have added a board containing less common systems such as GST68k O/S, a system designed for the QL, but eventually abandoned by Sinclair in favour of QDOS and SuperBASIC, but the real option was one version of QDOS or another.

QDOS was originally supplied in a little plug in eprom board, which became known as the 'kludge'. This version of QDOS had many problems and was quickly superceded and replaced. The first common QL ROM versions which are still in widespread use today were the versions called AH and JM. Rumours abound about the original meaning of the letters used to name these ROMs, from names of taxi drivers to names of Sinclair engineers, but no matter. The version letters are what is returned when you enter the command PRINT VER$ on a QL. It is important to note that the letters denoted the version of SuperBASIC and not the ROM version as such - this would have been QDOS v1.01 or whatever, although it became traditional to refer to the SuperBASIC version rather than QDOS version as such.

Version AH and JM were surprisingly stable given that they were fairly early versions of the operating system. They lacked error trapping and serial port character translate facilities introduced with ROM versions such as the JS (probably the most widely used of all Sinclair ROM versions). The error trapping in version JS of SuperBASIC worked, but with a few problems, so JS became a standard version. There were some later versions such as MG as well.

After Sinclair released these later ROM versions, users were still aware of various "features" or "bugs" in the Sinclair ROMs. Some programmers such as John Alexander and Laurence Reeves worked at producing independent ROM versions. The legality of some of the very early attempts at an improved and bug fixed QDOS was sometimes questionable - SOME were still clearly not only QDOS, but based on copyrighted Sinclair ROMs. Had Sinclair chosen to pursue this through the courts they may well have succeeded.

MGUK ROM

John Alexander produced the MGUK ROM, a derivative of the Sinclair MG ROM with bug fixes and a few new facilities. This was briefly available through Sector Software.

MINERVA

A small team at QView, consisting of Jonathan Oakley, Stuart MacKnight and Laurence Reeves, produced a QL ROM called Minerva. QView as a team left the QL scene, but in co-operation with TF Services, Laurence Reeves continued to work on Minerva. By now it had become a distinct version of QDOS, sufficiently different to original QDOS to stand on its own feet. Minerva fixed bugs in QDOS and SuperBASIC, and added all sorts of facilities such as support for the second screen, which the QL hardware had offered all along, but Sinclair QDOS did not fully support. Minerva could auto-boot after a power failure, useful when running software such as a bulletin board service which needed to be constantly on and available. Multiple BASICs allowed you to have more than one SuperBASIC program in memory at a time. The scheduler was improved, graphics were speeded up, error trapping improved, trace facilities added, non-English keyboard support added, "warm" fast reset and support added for split output baudrates when used in conjunction with the Hermes second processor replacement. Several versions of Minerva were released in fairly quick succession.

Later, the Mk 2 version of Minerva was produced. This was a hardware upgrade as well as an operating system upgrade. It added a Philips I2C expansion bus, real time clock support with battery backup, along with 256 bytes of non-volatile memory, allowing the QL to auto-boot from information held in that memory.

Both Minervas (or is that Minervae?) are still available from TF Services for £40 and £65 for UK customers, slightly greater for overseas customers.

A later controversy over the use of original QL ROMs in parts of the world where rights were still held by others caused TF Services to release an older version of Minerva (version 1.89) for use with QL emulators and so on. This provides a good example of the improvements Minerva can offer, but the more recent Minerva ROMs add even more improvements than v1.89

CP/M

An old business orientated operating system (Control Program/Management) for Z80 based computers (and later ported to other processor architectures) became available for the QL in at least 2 guises that I know of - one from Sandy UK PCP and the other from Digital Precision Ltd. Neither is still available, but you may come across the odd copy from second hand dealers. Be aware that like much software from that period, they may not work correctly with modern QL-compatible hardware, operating systems or emulators. While an interesting diversion if you wish to experiment with running CP/M compatible software or just wish to tinker with something different, I don't really suggest this as a viable alternative OS for your QL.

68K O/S

GST produced this operating system as an add on board at about the time of the launch of the QL. It was fairly short lived, as it did not have an on-board version of BASIC, unlike QDOS. Quest also produced some form of 68k OS for the QL (I never saw the Quest OS), although I think this was based more on a version of CP/M for 68000 based computers rather than a new QL OS as such.

DOS

Digital Precision (and others) produced some PC emulators for the QL, which allowed you to run some DOS programs on your QL. DOS was the original operating system for the PC. Strictly speaking, DP's Conqueror operating system used a compatible DOS called DR-DOS and was an emulator rather than a new operating system as such for the QL.

ZX81/Spectrum

Again, software based emulators allowed you to run programs for both the ZX81 and Spectrum operating system on your QL. Several were written over the years by people such as Dr Carlo Delhez, William James and Ergon Development.

SMS2

This was Tony Tebby's original progression from QDOS to a new single user multitasking system. SMS2 became available as a plug in cartridge for Atari ST computers. Although a good operating system, it lacked a BASIC interpreter. Tony Tebby was one of the original QL design team, and largely responsible for the original QDOS.

SMSQ

When Miracle Systems decided to produce a QL on a card which could be plugged into a PC's expansion slot (they were not the first - Sandy UK PCP had proposed such a system years before, but despite a lot fo publicity it never came to market) it was to come with an operating system called SMSQ. There were similarities between SMS2 and SMS for the QXL, but SMSQ was to have a new version of SuperBASIC called SBASIC, which featured many enhancements to SuperBASIC and of course quite a lot of bug fixes. While development suffered from delays, SMSQ did manage to establish itself as an entirely new QL-compatible operating system. It did not include any pointer environment within the operating system itself - that had to be added from disk (the now familiar ptr_gen, wman and hot_rext).

SMSQ/E

After a while, a development of SMSQ was produced for other QL compatible platforms, including Atari ST, Aurora, Q40 and of course the QL itself. The most obvious single difference between SMSQ/E and earlier QL operating systems was that it came with pointer environment built in. Now you had no excuse for not using pointer environment, mouse etc. It was finally built into the operating system. With the support of the likes of Jochen Merz, SMSQ/E slowly but surely won over QL purists. With ST, QXL, Q40, Aurora and QL users all using the same new operating system, and with that operating system being the only one still in active development (apart from Minerva) it became clear this was the OS of the future QL. A German QL software genius, Marcel Kilgus, was to prove many "experts" wrong and port SMSQ/E to run on PC hardware - the emulator we now know as QPC, proving that the QL operating system could if required be ported to other computers without necessarily involving add-on hardware. More recently, SMSQ/E has added the so-called GD2 (Graphic Driver 2) which supports 8-bit and 16-bit colour on various platforms.

Like the earlier SMSQ, SMSQ/E comes as a disk based operating system. While this means it takes slightly longer to start up than a ROM based system, and on some platforms has to be started by another operating system, this does make it easier to replace and upgrade - it has been upgraded many times in its history!

STELLA

This is a Tony Tebby proposal for a future operating system linked to SMSQ/E. You can't actually buy Stella yet, although various proposals for new machines using Stella are known to be in circulation and seeking finance. A French QL user, Arnould Nazarian, who is close to Tony Tebby, is an active proponent of Stella and has written many articles about it for QL magazines and mailing lists.

LINUX

Users of the Q40 and Q60 compatible computers have an alternative operating system available, called Linux 68K. This is supplied on a CD and turns the Q40/Q60 into a good little Linux platform.

Languages

Many of the well known computer languages have been ported to the QL, although some are no longer in production. A company called Metacomco released BCPL and Lisp for the QL in the early days. Computer One released a Pascal compiler. Several Forth compilers have been released, along with APL, Rexx, C, Prolog and J languages. Some of these are still available on the PD software scene, while some of the former commercial products still turn up second hand from time to time through dealers or at shows. In the early days of the QL, new language interpreters and compilers seemed to become available every week!

Hardware

I'll start my look at other hardware devices available by describing the various QL hard disk systems available.

The original QL hard disk systems were based on older small capacity hard drives - systems such as the Rebel Hard Disk system (rare) and the Miracle Hard Disk (not in production now, but plenty of them out there).

Due to the limited expansion capability of QLs, most of these early hard disk systems could not be added via the QL's expansion port since it was already loaded with cards such as memory expanders and floppy disk cards. Some of the expansion systems at the time made use of all the conventional addressable expansion space (e.g. the TrumpCard), so the suppliers of these systems had to resort to ingenious solutions such as hooking them up to the QL via the back EPROM slot. While this was an ingenious solution at the time, it did mean that these hard disk systems were specific to conventionally expanded QLs. If you had an Aurora system for example, you may have difficulty adding these systems to the equivalent expansion slot on the Aurora.

Another example of an early hard system can be found in the public domain. Dirk Steinkopf released a disk containing details of how to build a system based on MFM or RLL drive types. For those who might wish to attempt such a project (the diagrams were supplied) the information was available on the Specials 22 disk from Qubbesoft P/D for example. While Qubbesoft as a company is no longer trading, it may still be possible to obtain copies of this disk as it is public domain.

Qubbesoft were to market a newer interface for the QL and Aurora, using PC-style IDE hard disk drives. The expander was called Qubide and featured a through connector so that memory and disk expansion cards could be added, although in some cases this made QLs wider than the desktops holding them! The Qubide had a useful feature in that its working addresses could be altered via little jumper links on the board, so that if you found yourself using it with an incompatible expander unit, you could select alternative addresses. For example, if using it with a Trump Card or similar units, it could be made to work via the 16KB available for the back EPROM slot of a QL. The Qubide was, and still is, widely used, and probably the single most popular QL hard disk add-on. The unit went out of production with the demise of Qubbesoft P/D, but its popularity means that at some point another company may restart production. There are two main versions of the Qubide. The original version 1 was a straightforward hard disk expansion system. Version 2 was upgraded to provide facilities to support removable media Atapi IDE devices such as Iomega Zip, along with system enhancements such as providing a 'Trash Can' style facility for file deletion (allows later restoration of accidentally deleted disk files). The Qubide uses a file format unique on the QL scene in that it does not use the QXL.WIN format favoured by more recent systems such as emulators. Although the Qubide is capable of handling IDE CD-ROM drives, it does not provide much by way of software support for this as the author of its software ran out of room for the necessary code in the ROM. Third party software such as Dave Walker's Discover went some way towards fixing this by giving a limited file transfer capability, so you could at least copy files from a PC CD, for example.

Some emulators such as QDOS Classic can use a hard disk format similar to the Qubide format.

QXL.WIN Hard Disks

When Tony Tebby produced SMSQ for the QXL, he introduced the QXL.WIN file system container convention for hard disk storage. This is a single large file on the host computer's hard disk, set up so that the QL-compatible system sees the single large file as a single hard disk type of storage area. To the host computer, this is a single, large data file of some kind. To the QL-compatible, it is the complete hard drive. The QXL.WIN container system is used by all systems which use SMSQ or SMSQ/E, plus two QDOS based emulators (uQLx QL emulator for Linux/Unix and QemuLator for Windows platforms), so has become something of a standard for hard drive filing systems for QL-compatibles. Sadly, Qubides are not able to read media formatted in this way. QXL.WIN files need to be created even on removable media such as Iomega Zip, LS120 and so on, but not on floppy disks, which continue to use the same QL disk format.

The convention used was that WIN1_ would be the computer's main hard disk (drive C: on a DOS or Windows system) while WIN2_ would be the next drive (drive D:), the CD-ROM could be a third drive (WIN3_) and removable media such as ZIP drives could be WIN4_ (drive F:) and so on, up to 8 drives. All QXL.WIN files had to be in the root directory of these drives.

The Windows based QPC emulator took the QXL.WIN system a step further by allowing the user to define up to 8 'virtual hard disks' (WIN1_ to WIN8_). Each virtual disk could be in a separate directory on the same host hard drive, or on physically separate hard drives. Thus, your main hard disk (WIN1_) may be C:\QXL.WIN on a DOS based system, while WIN2_ can be a second QXL.WIN on the main hard drive (C:\QL\QXL.WIN on a PC), WIN3_ might be a QXL.WIN on a second hard drive (D:\QXL.WIN), a CD-ROM with a CD containing a QXL.WIN file might be WIN4_ (E:\QXL.WIN on a PC) and so on, up to 8 drives.

All of these systems capable of using QXL.WIN files can address much higher capacities than the older QL hard disk systems. These older systems were typically 20 to 40MB hard disks, while these days many times that capacity can be used. QL files are typically quite small and have much less storage requirements than the 'bloatware' files found on many systems. But it is encouraging that SMSQ/E can access huge hard disk files if required. One example I can quote is a CD-ROM I made containing all of my Line Design clipart for the QL, which ran to hundreds of megabytes (I never thought I'd need anywhere near that much for a QL-compatible system!).

I hope this series is providing useful information. If you have material to add or corrections to offer, please contact me at the usual QL Today editorial address. The next part of this series will hopefully look at other peripherals available for the QL and possible future peripherals too.

QSPREAD - a very special Export Printer Filter

Jochen Merz

Just to prove that I am not just saying I use SMSQ/E for most of my work except for communication, I have (after one year) written a nice export printer filter which makes my life so much easier. I do all my accounting on QSpread. Every month, I have to do about 30 to 50 sheets and export the results so that they can be transferred in an easily readable format (ASCII) to all sorts of other systems. What did I do in the past?

Highlight the block in QSpread, put into Scrap, exec QD with the scrap contents and save it from there (after a final check). Now, why should I do all this manually, fifty times every month? First I had a look at the Export feature of QSpread. The output is useful, a second filter program could turn this easily into what I wanted (e.g. by padding the fields) but this is exactly what Print does, so why not use Print instead, with a small and nice Print filter - BASIC, of course!

Here is the listing:

```
100 REMark Special filter which prints QSpread output
110 REMark to ram8_, construting the filename
automatically
120 REMark All bytes come in via #0
130 REMark nothing goes to #1
140 :
150 REPeat find_filename
160  IF EOF(#0):STOP
170  BGET#0,c
```

I will make the filter NOT print to a printer, but to a file which will be constructed out of the current QSpread filename, and be put into RAM8_. The QD/QSpread printer filter specification was designed so that it provides the filter with the file name which it currently processes. So the first action the filter does is: extract the filename out of the data stream. Every control code in QD/QSpread filters starts with CHR$(1) and ends with CHR$(2). So we loop until we find a CHR$(1) in the incoming data stream #0 (this is what QSpread sends to our BASIC filter). The next character has to be an "f", otherwise it is not what we want - and we want the filename, of course! Next, we add all incoming characters until we receive the terminating CHR$(2) ... and what we collected so far is our filename, honestly! Note: if you created a new sheet and you have not saved it, there will be no filename at all! It is not, that "NO NAME" will be the filename, there simply is no filename, so the filename will be an empty string, in which case we beep and give up!

Next, we create a "destination" filename out of the given filename. We replace the device by "ram8_" because that is where we want to store it, and we replace the extension "tab" by "txt" ... because this is the output file format we want.

Finally, we copy all the incoming data but strip all the format codes (like bold print, italics, character width, Form Feed and so on). The SELect clause does this for us. If we find a CHR$(1), we throw everything away until we receive the terminating CHR$(2). Form Feed is CHR$(12), which we don't want either. And all the other characters are data which we want to go into out file (which is opened with channel #3). You may have noticed that we do not print anything to #1 (the output channel, which is opened automatically for you by QSpread). No need to fill anything in "Print to:" in QSpread's Printer menu in, it will be ignored anyway.

```
180  SELect ON c
190   =1
200   BGET#0,o
210   IF o=CODE("f")
220    f$=""
230    REPeat
240     BGET#0,f
250     IF f=2:EXIT find_filename
260     f$=f$&CHR$(f)
270    END REPeat
280   END IF
290  END SELect
300 END REPeat find_filename
305 IF f$="":BEEP 10000,10000:STOP
310 file$="ram8_"&f$(6 TO LEN(f$)-3)&"txt"
320 OPEN_OVER#3,file$
330 :
340 REPeat loop
350  IF EOF(#0):STOP     : REMark That's it
360  BGET#0,c             : REMark Fetch char
370  SELect ON c
380   =1 :REPeat :BGET#0,c:IF c=2:EXIT :ELSE END
REPeat
390   =12                 :REMark FormFeed - just ignore
it
400   =REMAINDER :BPUT#3,c:REMark Write char
410  END SELect
420 END REPeat loop
```

So, this is another short example of how flexible the QD/QSpread print filter system is, and what it allows you to do. You can let it do much more for you if you want, e.g. convert special characters to DOS, TOS or Windows character sets, save it to DOS disks so that it can be transferred to other systems and so on and so on.

Channels are closed automatically when the filter is terminated, and QD/QSpread will not return from the filter printing as long as the filter does not hit a STOP statement or reaches the end of the filter program.

Do you have any interesting QD or QSpread filters or printer drivers? Please send them in so that we can publish them. There are so many useful things you can do, and maybe some of you did them already. So why not let others benefit - and maybe you can benefit from others!

QLTdis - part 5

Norman Dunbar

This is the fifth edition of the ongoing saga that is known as QLTdis. You might be puzzled as to the part numbers - I'm getting that way myself. Normally I write an article and sometimes Jochen asks me to split it (or where I can split it). This means that, for example, my part 3 ends up spread over 2 or 3 different magazines. Where I use a reference to a previous article, I refer to it using my own numbering - because that is what I am used to when I write it !

Another thing, you are getting the articles, descriptions etc at the same time (nearly) as I write them. This is the reason for the errors that occur from time to time. If I describe a bit of code in one article, I haven't yet written it at that point. When I subsequently come to code the algorithm, I use the printed version - so far so good. At the testing stage, I may find a problem (bad design I hear you cry!) and have to amend the algorithm - hence I have to print a correction. Its as much a pain for me as it is for you. (Honest.)

On with the coding. We start in the file UTILS_ASM first, and add the following sub-routines that I have found necessary as I coded up the disassembly routines.

The following routines act as a wrapper around the hex_l, hex_w and hex_b conversion routines. They add the contents of D4.L (.W or .B) to the current instruction buffer. Note that D4_HEX_L is actually called D4_HEX_4 because a lowercase 'L' and a digit one look the same. (I should have done this with hex_l and str_add_l as well - too late now!)

If you fancy a laugh, and just to show how tricky this assembler stuff can be, I originally got my code all wrong with the following routines. Where I now have 'lea buffer,a1' I had 'lea input,a1'. This meant that whenever I called these routines, they proceeded to modify the input routine's code because they used the code area as a buffer to convert to hex into. Suffice to say, when I then went round the loop asking for addresses, I got random hang ups (some worked, some crashed, some just hung!) depending upon the last address that was converted :o(

I tracked it down over a month (!) of wailing and gnashing and grinding of teeth. Even using QMON it was a task and a half - in the end, a flash of inspiration solved it.

```
*-------------------------------------------------------------------------------
* Convert the long word in D4 to 8 bytes in the input buffer. Once done, add all
* 8 bytes from the buffer to the end of our current decoded instruction.
*-------------------------------------------------------------------------------
d4_hex_4    lea     buffer,a1       ; Start of input buffer
            bsr     hex_l           ; Convert D4.L to hex
            lea     buffer,a1       ; Buffer again
            move.l  (a1)+,d4        ; Get first long word
            bsr     str_add_l       ; Add it to the buffer
d4_last_4   move.l  (a1),d4         ; Get the second long word
            bsr     str_add_l       ; And add it too
            rts

*-------------------------------------------------------------------------------
* Convert the word in D4 to 4 bytes in the input buffer. Once done, add all 4
* bytes from the buffer to the end of the current decoded instruction.
*-------------------------------------------------------------------------------
d4_hex_w    lea     buffer,a1       ; Start of input buffer
            bsr     hex_w           ; Convert D4.W to hex
            lea     buffer,a1       ; Buffer again
            bra.s   d4_last_4       ; Add it to the buffer

*-------------------------------------------------------------------------------
* Convert the byte in D4 to 2 bytes in the input buffer. Once done, add both
* bytes from the buffer to the end of the current decoded instruction.
*-------------------------------------------------------------------------------
d4_hex_b    lea     buffer,a1       ; Start of input buffer
            bsr     hex_b           ; Convert D4.B to hex
            lea     buffer,a1       ; Start of buffer again
            move.w  (a1),d4         ; Get both hex bytes
            bsr     str_add_w       ; Add it to the buffer
            rts
```

```
*------------------------------------------------------------------------------
* A routine to print the character in D1.B to the output file.
*------------------------------------------------------------------------------
one_byte    moveq   #io_sbyte,d0    ; Send one byte to channel
trap_3      moveq   #infinite,d3    ; Timeout
            trap    #3              ; Do it
            rts

*------------------------------------------------------------------------------
* A routine to tab to column D1.W on the current output line
*------------------------------------------------------------------------------
tab_to      moveq   #sd_tab,d0      ; Tab to column
            bra.s   trap_3          ; Do it
```

OK, that should have whetted your appetite a bit, now I need to tell you of some changes I have had to make to the pseudo code in the last exciting (?) episode of QLTdis. If you remember, I explained how I proposed to decode each instruction family - well, since then I have written & tested code to disassemble types 0 to 16 (almost all the easy ones !) and I have come across a couple of small problems. Here then is how the pseudo code should look for those erroneous descriptions.

Type 2

This type also has a word of data and this time both bytes are used. All type 2 operations are acting upon the Status Register except the STOP #N instruction.

```
Example 'ANDI #$data,SR'
Add the word at (A6)+ to the op-code buffer in
hex.
If the word held in d7 is not $4e72
  Add the string ',SR' to the op-code buffer.
Done.
```

Type 9

Type nine is quite simple as the following description shows. Note that when we mask out D0.W leaving only the size bits in bits 6, 7 and 8 we don't bother to shift them we just test if D0.W is zero and if so the size is word otherwise it has to be long. Byte sized EXT instructions are not valid and we cannot use our 'size' sub-routine because this family has a non-standard value in the size bits.

```
Example 'EXT.size Dn'
Mask out all but bit 6 - the size part.
If D0.W = 0
    Add '.W' to the op-code buffer
else
    Add '.L' to the op-code buffer
end if
Add ' D' to the op-code buffer.
Copy D7.W to D0.W.
Call the source register routine.
Done.
```

Type 14

Back to more complex instructions for a while. The EXG instruction can be between two address registers, or two data registers or one of each. The value held in bits 3 to 7 of the op-code define the correct mode.

```
Example 'EXG Dx,Dy'
     or 'EXG Ax,Ay'
     or 'EXG Dx,Ay'
Mask out all but bits 3 to 7 of D0.W.
Shift D0.W right 3 bits.
Save D0.W on the stack - we need it again later.
If D0.W = 9
    add 'A' to the op-code buffer
else
    add 'D' to the op-code buffer
end if
Copy D7.W to D0.W.
call dest register routine (preserves D0)
Add ',' to op-code buffer.
If the word on the stack (our old D0) = 8
    add 'D' to the op-code buffer
else
    add 'A' to the op-code buffer
end if
call source register routine (preserves D0)
Restore stack by adding 2 to A7.
Done.
```

Type 15

MOVEP is the only type fifteen instruction. It is actually quite simple to decode.

```
Example 'MOVEP.size $data(An),Dn)'
     or 'MOVEP.size Dn,$data(An)'
If bit 6 of D7 is set
    add 'L' to the op-code buffer NOT '.L'
else
    add 'W' to the op-code buffer NOT '.W'
end if
Add ' ' to the op-code buffer.
If bit 7 of D7 is set
    add 'D' to op-code buffer
    call dest register routine (preserves D0)
    add ',' to op-code buffer
end if
Add '$' to the op-code buffer.
Convert the word at (A6)+ to hex and add to the
op-code buffer.
Add '(A' to the op-code buffer.
Call the source register routine (preserves D0).
Add ')' to the op-code buffer.
If bit 7 of D7 is clear
```

```
    add ',D' to op-code buffer
    call dest register routine (preserves D0)
end if
Done.
```

Now then, when I wrote the type 16 stuff originally, I noticed that my way of decoding and Andy Pennell's way were very nearly the same. In testing my code, I have found that we both got it wrong. We both looked for the instruction type in bits 3 & 4 first, then decided if it was a memory shift or a register shift later on. This doesn't work because the instruction ASL (A0) decodes as ROXL which is totally wrong. Here then is my new improved version of decoding the shift/rotate instructions. (It works!)

Type 16

Back to a dificult one again. The shifts and rotates are the type sixteen family and there is much wailing and gnashing of teeth required to decode this little lot.

```
Example 'ASL.size Dx,Dy' or 'ASL.size ‹ea›'

Mask out D0.W keeping only bits 6 & 7 which define the
size of the op-code.
Shift these two bits right by 6 places.
If we have D0 = 3 (memory shift/rotate)
    set bit 16 of D7 to show memory
    set D0.W = $0600 (bits 9 & 10)
    set D1.W = 8 (shift count)
else (register shift/rotate)
    clear bit 16 of D7 to show register
    set D0.W = $0018 (bits 3 & 4)
    set D1.W = 2 (shift count)
end if

And D7,D0 (word sized) to extract the 2 bits that tell
us what opcode is being decoded.
Shift D0.W right by D1.W bits (3 or 8 only)

At this point we have the following:

Bit 16 of D7 shows the memory or register shift/rotate
flag.
D0 = 0, 2, 4 or 6 for the shift/rotate instruction
being decoded.

Point A3 at SH_TABLE.
```

The values in D0 are 0,2,4 or 6 and represent 'AS', 'LS', 'ROX' and 'RO'. Rather than have a table with size words in it as well, we simply have a table of 2 character strings and process D0 to see if an 'X' should be added afterwards. This makes life a lot easier and I am all in favour of that! See SH_TABLE below.

```
Get the word at A3 + D0.W into D4.
D4 = 'AS' or 'LS' or 'RO' or 'RO'
Add D4.W to the op-code buffer.
If D0 = 4
    Add 'X' to the op-code buffer (We are doing ROXR
or ROXL)
end if
```

Now we have the correct instruction, we need the direction part of it.

```
If bit 8 of D7 is set
    add 'L' to op-code buffer
else
    add 'R' to op-code buffer
end if

Copy D7.W to D0.W again.
if bit 16 of D6 is set (memory shift/rotate) then
    This is a word sized shift in memory
    Add ' ' to the op-code buffer
    Move 2 (word) to D5 for size. Required by the
effective address routine.
    Call effective address routine (preserves D0)
    Done.
end if
```

We must be doing a register shift/rotate then! Call the 'size into D0' routine which sets D0 to the size bits.

```
if bit 5 of D7 is clear
    This must be a 'count in data' shift/rotate
    add '#$' to the op-code buffer
    mask out all but bits 9 to 11 of D0.W
    if D0.W = 0
        add '8' to the op-code buffer
    else
        call the dest register routine
    end if
else
    This is a 'count in register' shift/rotate
    add 'D' to the op-code buffer
    call the dest register routine
end if
add ',D' to the op-code buffer
call the source register routine

Done.

SH_TABLE    DC.W    "ASLSRORO"
```

In the following sub-routine descriptions, change references to D0.W to read D7.W as D0.W is usually corrupted by the time we get to calling these routines:

```
Source register routine.
Dest register routine.

In the Size in D0 routine the description of OUTPUT
should read:

OUTPUT :
D0.W = old bits 6 & 7 shifted right by 5 bits.

and in the description, we only shift by 5 bits and
not 6.
```

It just gets worse!

And here are a couple of corrections I need to make to the actual code in part 3. I have found an entry in the mask table to be incorrect for the type_12 instructions. The result word for the type 12 instructions is currently $4e50 when it should be $4e60 - I cannot convert from binary to hex it would appear.
At present, the line in question looks like this:

```
type_12 dc.w $fff0,$4e50,12 .....
```

when it should read :

```
type_12 dc.w $fff0,$4e60,12 .....
```

ALSO:
When we come to print the ascii characters for the decoded instruction, we need to start at column 55 not 53 as before - some instrutions are too long (and some of the forthcoming type 17 to 31 may also require a change here too. This doesn't affect any code, because you haven't yet written the code - but it does contradict an earlier description of how the output will appear.

Enough of this, lets get some more typing done! In the file DISS_ASM, we need to remove the current 'stub' routine as shown below:

```
diss        moveq    #0,d0          ; No errors
            rts
```

This needs to be changed so that it controls the whole disassembly process. As outlined in the previous edition (QLTdis part 4), we made a few assumptions about what would be done for us by the time we enter into a 'type' decoding sub-routine. This is where we change all those assumptions into code. The two lines above should be replaced with the following code:

```
*-------------------------------------------------------------------------
* The start of the main disassembly loop.
*
* Assumption 1 - the address of the instruction will be printed with a preceding
* 'L' as an 8 character hex number. This will be followed by 2 spaces.
*
* Assumption 5 - pc_addr(A4) holds the (next) address to be decoded.
*-------------------------------------------------------------------------
diss        move.l  con_id2(a4),a0  ; Get channel id (Screen only)
            moveq   #'L',d1         ; Byte to send = 'L'
            bsr     one_byte        ; Print it
            move.l  a6,d4           ; Get the (next) address
            bsr     print_hex       ; Prints it out
            moveq   #10,d1          ; Column number to tab to = 10
            bsr     tab_to          ; Tab to column 10
```

```
*-------------------------------------------------------------------------------
* Assumptions 6,7 & 8 - A5 holds the buffer where we decode the instruction and
* D5.W will hold the length of the decoded instruction and D5.W will be set to
* zero.
*-------------------------------------------------------------------------------
            moveq   #0,d5           ; D5 is indeed set to zero
            moveq   #0,d6           ; As is the current decoded length
            lea     output+2,a5     ; And A5 is the start of our buffer

*-------------------------------------------------------------------------------
* Now we need to take a trip through our masks table checking which instruction
* needs to be decoded. Of course, we need to load D0 and D7 with the op-code
* first ! This covers assumption 3.
*-------------------------------------------------------------------------------
            move.w  (a6)+,d7        ; Fetch the op-code and increment
            lea     masks,a3        ; Masks table start

mask_loop   move.w  d7,d0           ; Copy the op-code
            and.w   (a3)+,d0        ; Mask it
            cmp.w   (a3)+,d0        ; Is it the same as the table ?
            beq.s   got_type        ; Yes, skip
            addq.l  #4,a3           ; No, skip type and string offset
            bra.s   mask_loop       ; Try the next one

got_type    move.w  (a3)+,d0        ; Fetch the type
            move.w  (a3),d1         ; Fetch the string offset

*-------------------------------------------------------------------------------
* D0.W = the type
* D1.W = the offset from the start of OP_TABLE to the required string.
*
* Assumption 2 - copy the string from the table to the output buffer.
*-------------------------------------------------------------------------------
            lea     op_table,a3     ; Start of string table
            lea     (a3,d1.w),a1    ; Source address
            move.w  (a1),d6         ; Get the size of the string
            lea     output,a2       ; Destination address
            bsr     str_copy        ; Copy string to the output buffer
            adda.w  d6,a5           ; Set A5 to next free byte in buffer

*-------------------------------------------------------------------------------
* Now we calculate and jump to the decoding routine for this type (in D0.W)
*-------------------------------------------------------------------------------
            move.w  d0,d1           ; Put the type word in D1
            move.w  d7,d0           ; Get the op-code where we want it !
            lea     j_table,a3      ; Start of jump table
            lsl.w   #1,d1           ; Convert from type to j_table offset
            move.w  0(a3,d1.w),d1   ; Fetch the offset to the type
            jmp     0(a3,d1.w)      ; Jump to the routine
*                                   ; NOTE - we never return here !

j_table     dc.w    dtype_0-j_table  ; Offset to type routines
            dc.w    dtype_1-j_table
            dc.w    dtype_2-j_table
            dc.w    dtype_3-j_table
            dc.w    dtype_4-j_table
            dc.w    dtype_5-j_table
            dc.w    dtype_6-j_table
            dc.w    dtype_7-j_table
            dc.w    dtype_8-j_table
            dc.w    dtype_9-j_table
            dc.w    dtype_10-j_table
            dc.w    dtype_11-j_table
            dc.w    dtype_12-j_table
            dc.w    dtype_13-j_table
            dc.w    dtype_14-j_table
            dc.w    dtype_15-j_table
            dc.w    dtype_16-j_table
            dc.w    dtype_17-j_table
            dc.w    dtype_18-j_table
            dc.w    dtype_19-j_table
            dc.w    dtype_20-j_table
            dc.w    dtype_21-j_table
            dc.w    dtype_22-j_table
            dc.w    dtype_23-j_table
            dc.w    dtype_24-j_table
```

```
            dc.w    dtype_25-j_table
            dc.w    dtype_26-j_table
            dc.w    dtype_27-j_table
            dc.w    dtype_28-j_table
            dc.w    dtype_29-j_table
            dc.w    dtype_30-j_table
            dc.w    dtype_31-j_table

*------------------------------------------------------------------------------
* ASSUMPTION 3 - This is the common return address from ALL the type routines.
* At this point all we know is that :
*
* A0 which used to hold the channel id for the screen may or may not still do so
* A5 is pointing to the first character after the end if the decoded instruction
* D6.W is the size of the decoded instruction's string
* PC_ADDR(A4) is the original start of this instruction
* A6 is the start of the next instruction to be decoded, and
* we are sitting at column 10 ready to print some hex codes.
*------------------------------------------------------------------------------
* NOTE : We requested a file device (or printer) to be printed to as well. This
* will be added later on, for now, the screen is the output only.
*------------------------------------------------------------------------------
p_hex       move.l  con_id2(a4),a0  ; Fetch output channel id (Screen only)
            lea     output,a5       ; A5 is back at the start of the output area
            move.w  d6,(a5)         ; Now it is a QDOS string with a word count
            move.l  a6,d6           ; Get next instruction start address
            move.l  pc_addr(a4),a3  ; Get original address of this instruction
            sub.l   a3,d6           ; Calculate length of this instruction
            move.w  d6,-(a7)        ; Save for later
            lsr.w   #1,d6           ; Size must be even - so do words
            bra.s   ph_next         ; Skip dbra stuff

ph_loop     move.w  (a3)+,d4        ; Fetch one word
            lea      buffer,a1      ; Output buffer for address
            move.w   #4,(a1)+       ; We know the result is 4 bytes
            bsr      hex_w          ; Convert 4 bytes to text
            lea      buffer,a1      ; Text to print
            bsr      prompt         ; Print it

ph_next     dbra    d6,ph_loop      ; And the rest

*------------------------------------------------------------------------------
* Now we have printed the hex codes, tab to column 33 ready for the decoded
* instruction text from the output buffer.
*------------------------------------------------------------------------------
            moveq   #33,d1          ; Column 33
            bsr     tab_to          ; Tab to column 33
            move.l  a5,a1           ; Output buffer
            bsr     prompt          ; Print it
            moveq   #55,d1          ; Column 55
            bsr     tab_to          ; Tab to column 55
            move.w  (a7)+,d6        ; Restore length of instruction
            move.l  pc_addr(a4),a3  ; Restore original address
            bra.s   asc_next        ; Skip dbra stuff

asc_loop    move.b  (a3)+,d1        ; Fetch a single byte
            cmpi.b  #' ',d1         ; Test less than 'space'
            bge.s   asc_print       ; We can print ascii higher than 'space'

asc_dot     moveq   #'.',d1         ; Force a dot

asc_print   bsr     one_byte        ; Print it

asc_next    dbra    d6,asc_loop     ; Do the rest
            bsr     line_feed       ; And the final linefeed

*------------------------------------------------------------------------------
* Save the next address to be decoded in the appropriate place.
*------------------------------------------------------------------------------
            move.l  a6,pc_addr(a4)  ; Store the next address to decode
            rts                     ; All done

*------------------------------------------------------------------------------
* Buffer to hold the decoded instructions
*------------------------------------------------------------------------------
output      ds.w    31              ; Assume 60 bytes + size word
```

On now to the exciting stuff, the instruction type decoding. Here are types 0 through 16 and type 31 which is required in case of an unknown type being found. The comments that follow will give brief details of what instructions are being dealt with or decoded. For full details, see the article QLTdis part 4.

Type 0 is the simplest instruction type. By the time we get to this point, all the work has been done (see assumptions) and we are ready to print the instructions hex codes etc.

```
*------------------------------------------------------------------------------
* TYPE_0 has nothing to do. Simply exit back to print the hex codes for the
* instruction and then the rest of the line.
*------------------------------------------------------------------------------
dtype_0     bra     p_hex           ; Nothing to do for type 0
```

Type 1 has a single word of data following the op-code but only the lowest byte is required. All we have to do is extract this byte and add it to the instruction (in hex) then finish off the instruction with ',CCR'. See 'QLTdis part 2' for details of each instruction type and the instructions in its 'family'.

```
*------------------------------------------------------------------------------
* TYPE_1 uses the low byte of the following word as data.
*------------------------------------------------------------------------------
dtype_1     move.w  (a6)+,d4        ; Get the word of data
            bsr     d4_hex_b        ; Add the low byte to the buffer
            bsr     comma_ccr       ; Add ',CCR' to the buffer
            bra     p_hex           ; Done
```

Type 2 also has a word of data and this time both bytes are used. All type 2 operations are acting upon the Status Register.

```
*------------------------------------------------------------------------------
* TYPE_2 uses the following word as data.
* Except for the STOP #N instruction, everything acts on the Status Register.
*------------------------------------------------------------------------------
dtype_2     move.w  (a6)+,d4        ; Get the word of data
            bsr     d4_hex_w        ; Add the low byte to the buffer
            cmpi.w  #$4e72,d7       ; STOP #N ?
            beq.s   d2_done         ; Yes, don't add ',SR' to the buffer
            bsr     comma_sr        ; Add ',SR' to the buffer
d2_done     bra     p_hex           ; Done
```

Type 3 family instructions require a register number to be added to the string that we already have in the op-code buffer. As numerous instructions require this, we have extracted the code to a sub-routine which will be explained later. This routine adds to the op-code buffer so no further work is required here.

```
*------------------------------------------------------------------------------
* TYPE_3 requires a source register to be added to the output buffer.
*------------------------------------------------------------------------------
dtype_3     bsr     src_reg         ; Add in the source_register
            bra     p_hex           ; Done
```

Type four is the TRAP #n instruction where 'n' can be any value between 1 and 15. All we do is test to see if 'n' is greater than 9 and if so, put a one in the buffer, subtract 10 from 'n' and then we can add the remaining digit.

```
*------------------------------------------------------------------------------
* TYPE 4 have data in bits 0 to 3 of the instruction word.
*------------------------------------------------------------------------------
dtype_4     andi.b  #$0f,d0         ; Keep bits 0 to 3 only
            cmpi.b  #10,d0          ; Is D0 > 9 ?
            bcs.s   under_9         ; No, skip the next bit
            moveq   #'1',d4         ; Needs a one in the buffer
            bsr     str_add_b       ; Add one byte
            sub.b   #10,d0          ; Subtract ten, d0 is now < 6
under_9     move.b  d0,d4           ; Needed in D4
            add.b   #'0',d4         ; Convert to a digit
            bsr     str_add_b       ; Add it to buffer
            bra     p_hex           ; Done
```

Type five is the LINK instruction. This is quite easily decoded as follows.

```
*------------------------------------------------------------------------------
* TYPE 5 have source registers and data words.
*------------------------------------------------------------------------------
dtype_5     bsr     src_reg         ; Add the source register
            move.l  #',#$',d4       ; Some text to add
            bsr     str_add_3       ; Add the 3 characters to the buffer
            move.w  (a6)+,d4        ; Get the data word
            bsr     d4_hex_w        ; And add to the buffer
            bra     p_hex           ; Done
```

The decrement and branch instructions - type 6 - are covered next. These have a condition code in bits 8 to 11 and are decoded in the following way. These are decoded as 'DBcc Lxxxxxxxx' where 'xxxxxxxx' is the hex address where the branch is to.

```
*------------------------------------------------------------------------------
* TYPE 6 DBcc instructions.
*------------------------------------------------------------------------------
dtype_6     andi.w  #$0f00,d0       ; Mask out the condition code bits
            lsr.w   #8,d0           ; Condition code now in bits 0 to 3
            bsr     cond_code       ; Convert it to text
            bsr     space_d         ; Add ' D' to the buffer
            bsr     src_reg         ; Extract source register
            bsr     comma           ; Followed by a comma
            movea.l a6,a3           ; Copy current PC address
            adda.w  (a6)+,a3        ; Add in the branch offset (Sign extends)
            bsr     ell             ; Add 'L' to the buffer
            move.l  a3,d4           ; Get the destination address
            bsr     d4_hex_4        ; Convert to hex in the buffer
            bra     p_hex           ; Done
```

Type 7 is another simple instruction. BSR is decoded thus:

```
*------------------------------------------------------------------------------
* TYPE 7 instructions.
*------------------------------------------------------------------------------
dtype_7     movea.l a6,a3           ; Current PC address
            cmpi.b  #0,d0           ; If D0 is not zero -> BSR.S
            beq.s   t7_long         ; It's not short !
            bsr     dot_s_space     ; Add '.S ' to the buffer
            andi.w  #$00ff,d0       ; Can't ADDA.B so mask out extra data
            adda.w  d0,a3           ; Add the offset
            bra.s   t7_label        ; Go print the label

t7_long     bsr     space           ; Add a space to the buffer
            adda.w  (a6)+,a3        ; Add the offset (sign extended)

t7_label    bsr     ell             ; Add an 'L' to the buffer
            move.l  a3,d4           ; Get the address
            bsr     d4_hex_4        ; Convert to hex and add to buffer
            bra     p_hex
```

Type 8 is another instruction with condition codes. As above with the DBcc instructions, we process it as follows noting the special case where the condition code is 0 which means BRA rather than BF which doesn't make any sense.

```
*------------------------------------------------------------------------------
* TYPE 8 instructions have condition codes.
*------------------------------------------------------------------------------
dtype_8     andi.w  #$0f00,d0       ; Mask out all but bits 8 to 11
            lsr.w   #8,d0           ; Move bit 8 to bit 0
            cmpi.b  #0,d0           ; Condition code = zero = 'RA' and not 'F'
            bne.s   t8_rest         ; Not zero, do the rest
            move.w  #'RA',d4        ; Zero is BRA
            bsr     str_add_w       ; Add it to the buffer
            bra.s   t8_size         ; Skip over the next bit

t8_rest     bsr     cond_code       ; Decode the condition code

t8_size     move.w  d7,d0           ; Restore D0 (the op-code)
            movea.l a6,a3           ; Get current PC address
            cmpi.b  #0,d0           ; Short branch ?
```

```
            beq.s   t8_long         ; No
            bsr     dot_s_space     ; Add '.S ' to the buffer
            andi.w  #$00ff,d0       ; Mask out spare data - can't ADDA.B
            adda.w  d0,a3           ; Add in the offset
            bra.s   t8_label        ; And skip the next bit

t8_long     bsr     space           ; Add a space
            adda.w  (a6)+,a3        ; Add in the offset (sign extended)

t8_label    bsr     ell             ; Add 'L' to the buffer
            move.l  a3,d4           ; Get the offset to the destination
            bsr     d4_hex_4        ; Convert and add to the buffer
            bra     p_hex           ; Done
```

Type nine is quite simple as the following description shows. Note that when we mask out D0.W leaving only the size bits in bits 6, 7 and 8 we don't bother to shift them we just test if D0.W is zero and if so the size is word otherwise it has to be long. Byte sized EXT instructions are not valid and we cannot use our 'size' sub-routine because this family has a non-standard value in the size bits.

```
*-------------------------------------------------------------------------------
* TYPE 9 - the EXT instructions
*-------------------------------------------------------------------------------
dtype_9     andi.w  #$0040,d0       ; Mask out all but bits 6 to 8
            bne.s   t9_long         ; Not a word sized instruction
            bsr     uu              ; Add 'W' to the buffer
            bra.s   t9_cont         ; Skip the next bit

t9_long     bsr     ell             ; Add 'L' to the buffer

t9_cont     bsr     space_d         ; Add ' D' to the bufffer
            move.w  d7,d0           ; Restore D0
            bsr     src_reg         ; Add in the source register
            bra     p_hex           ; Done
```

Type ten is the MOVEQ instruction and is so simple to decode. The data is held in bits 0 to 7 of D0.

```
*-------------------------------------------------------------------------------
* TYPE 10 - the MOVEQ instructions
*-------------------------------------------------------------------------------
dtype_10    move.b  d0,d4           ; Get the data part
            bsr     d4_hex_b        ; Convert to hex and add to the buffer
            bsr     comma_d         ; Add ',D' to the buffer
            bsr     dest_reg        ; Add the destination register
            bra     p_hex           ; Done
```

Type eleven is the Binary Coded Decimal instructions plus ADDX and SUBX. In the description in part 2, I mentioned that it was quite a tricky instruction to decode. I was wrong ! By testing bit 3 of D7 we know whether the '-(Ax),-(Ay)' version or the 'Dx,Dy' version of the instruction is being used.

```
*-------------------------------------------------------------------------------
* TYPE 11 - The Binary coded decimal instructions plus ADDX and SUBX.
*-------------------------------------------------------------------------------
dtype_11    btst    #3,d7           ; Is this -(Ax),-(Ay) ?
            beq.s   t11_data        ; No
            bsr     mba             ; Add '-(A' to the buffer
            bra.s   t11_src         ; Skip next bit

t11_data    bsr     dddd            ; Add 'D' to the buffer

t11_src     bsr     src_reg         ; Extract the source register
            btst    #3,d7           ; Address reg with pre decrement ?
            beq.s   t11_data2       ; No, data registers
            bsr     r_bracket       ; Add ')' to the buffer
            bsr     comma_mba       ; Add ',-(A' to the buffer
            bra.s   t11_dest        ; Skip the next bit

t11_data2   bsr     comma_d         ; Add ',D' to the buffer

t11_dest    bsr     dest_reg        ; Append the destination register
            btst    #3,d7           ; Need a close bracket ?
            beq     p_hex           ; No, done for Data registers
            bsr     r_bracket       ; Add a ')' to the buffer
            bra     p_hex           ; Done
```

Type 12 is another simple instruction to decode once we know that bit 3 defines the direction of the MOVE to or from the USP. Testing for this instruction has thrown up a possible bug in GWASL where MOVE USP,An is being coded. George Gwilt has been informed and will be looking at the problem when he can.

```
*------------------------------------------------------------------------------
* TYPE 12 - The move to and from USP instructions
*------------------------------------------------------------------------------
dtype_12    btst    #3,d7           ; From USP ?
            beq.s   t12_A           ; No
            move.l  #'USP,',d4
            bsr     str_add_1       ; Add to the buffer

t12_A       bsr     aaaa            ; Add 'A' to the buffer
            bsr     src_reg         ; Add the source register
            btst    #3,d7           ; To USP ?
            bne     p_hex           ; No, done
            move.l  #',USP',d4       ; Yes
            bsr     str_add_1       ; Add to the buffer
            bra     p_hex           ; Done
```

Type 13 is Yet another simple instruction.

```
*------------------------------------------------------------------------------
* TYPE 13 - The CMPM instruction.
*------------------------------------------------------------------------------
dtype_13    bsr     size_decode     ; Add the size specifier and a space
            bsr     l_bracket_a     ; Add '(A'
            bsr     src_reg         ; Insert the source register number
            move.l  #')+,(',d4      ; More text
            bsr     str_add_1
            bsr     aaaa            ; Add 'A' to the buffer
            bsr     dest_reg        ; Insert the destination register
            move.w  #')+',d4        ; More text
            bsr     str_add_w
            bra     p_hex           ; Done
```

Type 14 is a slightly more complex instruction. The EXG instruction can be between two address registers, or two data registers or one of each. The value held in bits 3 to 7 of the op-code define the correct mode.

```
*------------------------------------------------------------------------------
* TYPE 14 - The EXG instructions
*------------------------------------------------------------------------------
dtype_14    andi.w  #$00f8,d0               ; Mask out all but bits 3 to 7
            lsr.w   #3,d0                   ; Bit 3 -> bit 0
            move.w  d0,-(a7)                ; Save it on the stack for now
            cmpi.w  #9,d0                   ; Is d0 = 9
            bne.s   t14_dreg                ; No, its a data register
            bsr     aaaa                    ; Yes, do Address register
            bra.s   t14_next                ; Skip over next bit

t14_dreg    bsr     dddd                    ; It's a data register

t14_next    move.w  d7,d0                   ; restore D0
            bsr     dest_reg                ; Yes - do destination register
            bsr     comma                   ; Add a comma to the buffer
            cmpi.w  #8,(a7)                 ; Was D0 = 8 ?
            bne.s   t14_areg                ; No, do an A register
            bsr     dddd                    ; Do a data register
            bra.s   t14_add2                ; Skip

t14_areg    bsr     aaaa                    ; Do an address register

t14_add2    addq.l  #2,a7                   ; Tidy stack
            bsr     src_reg                 ; Add in the source regsiter
            bra     p_hex                   ; Done
```

Type 15 is MOVEP and is the only type fifteen instruction. It is actually quite simple to decode. Unfortunately, it too doesn't assemble correctly with GWASL. As above, George has been informed.

```
*------------------------------------------------------------------------------
* TYPE 15 - the MOVEP instruction
*------------------------------------------------------------------------------
dtype_15    btst    #6,d7           ; Test bit 6 = the size indicator
            beq.s   t15_word        ; Not set = '.W'
            bsr     ell             ; Must be '.L'
            bra.s   t15_size        ; Skip

t15_word    bsr     uu              ; Word sized

t15_size    bsr     space           ; Add a space to the buffer
            btst    #7,d7           ; If bit 7 is set
            beq.s   t15_not7        ; Skip if not set
            bsr     dddd            ; Adds a 'D'
            bsr     dest_reg        ; Add the dest register
            bsr     comma           ; Add a comma next

t15_not7    bsr     dollar          ; Everything needs a dollar
            move.w  (a6)+,d4
            bsr     d4_hex_w        ; Add the offset to the buffer
            bsr     l_bracket_a     ; Add '(A' to the buffer
            bsr     src_reg         ; Add the source register
            bsr     r_bracket       ; Close bracket
            btst    #7,d7           ; Another check
            bne     p_hex           ; Set = done
            bsr     comma_d         ; Add ',D'
            bsr     dest_reg        ; And the destination register
            bra     p_hex           ; Done
```

Type 16 gave me a lot of grief in testing and I ended up having to rewrite the routine. Here is the working version.

```
*------------------------------------------------------------------------------
* TYPE 16 - the shifts and rotates
*------------------------------------------------------------------------------
dtype_16    andi.w  #$c0,d0         ; Extract bits 6 & 7 only
            lsr.w   #6,d0           ; Shift into bits 0 & 1
            cmpi.w  #3,d0           ; 3 = Memory shift/rotate
            bne.s   t16_reg         ; Not 3 = register

t16_memory  bset    #16,d7          ; Flag memory operation
            move.w  #$0600,d0       ; We only want bits 9 & 10
            moveq   #8,d1           ; We need to shift 8 bits later on
            bra.s   t16_both        ; Skip over register stuff

t16_reg     bclr    #16,d7          ; Flag register operation
            move.w  #$0018,d0       ; We only want bits 3 & 4
            moveq   #2,d1           ; We will shift by 2 bits later on

t16_both    and.w   d7,d0           ; Keep whichever bits we need
            lsr.w   d1,d0           ; Shift right appropriately
            lea     sh_table,a3     ; Our table of instructions
            move.w  0(a3,d0.w),d4   ; Get the instruction we want
            bsr     str_add_w       ; Add it to the buffer
            cmpi.b  #4,d0           ; ROXL or ROXR if D0 = 4
            bne.s   t16_dir         ; Not ROX
            moveq   #'X',d4         ; Must be ROX
            bsr     str_add_b       ; And add it to the buffer

t16_dir     btst    #8,d7           ; 0 = Right, 1 = Left
            beq.s   t16_right       ; Must be zero
            moveq   #'L',d4         ; Going left
            bra.s   t16_add         ; Skip

t16_right   moveq   #'R',d4         ; Going right

t16_add     bsr     str_add_b       ; Add direction to the buffer
            move.w  d7,d0           ; Restore the op-code
            bsr     size_d0         ; Get the size bits in D0
            btst    #16,d7          ; Set = word sized shift in memory
            beq.s   t16_reg2        ; Not memory, must be register

t16_mem     bsr     space           ; Add a space
            moveq   #2,d5           ; Word sized operation
            move.w  d7,d0           ; Need the op-code again
```

```
            bsr     eff_addr        ; work out the effective address
            bra     p_hex           ; And all done for memory operationa

t16_reg2    move.w  d7,d0           ; Fetch the op-code word again
            bsr     size_decode     ; Add the size to the buffer (and a space)
            move.w  d7,d0           ; Again we need D0 !!!
            btst    #5,d7           ; Counter in reg or data ?
            beq.s   t16_data        ; Clear = data

t16_creg    bsr     dddd            ; Counter is in a data register
            bsr     dest_reg        ; Extract the counter register
            bra.s   t16_all         ; Skip

t16_data    bsr     hash_dollar     ; Add '#$' to the buffer
            andi.w  #$0e00,d0       ; Keep bits 9 to 11
            bne.s   t16_not8        ; Data between 1 and 7 if not zero
            bsr     eight           ; Add to the buffer
            bra.s   t16_all

t16_not8    bsr     dest_reg        ; Dest reg holds shift counter

t16_all     bsr     comma_d         ; Add ',D'
            bsr     src_reg         ; And the register to be shifted
            bra     p_hex           ; Done

sh_table    dc.w    "ASLSRORO"      ; First 2 characters of each instruction
```

And that is all there is to it. Notice how much of the decoding is very simple. Who said it was hard doing all this disassembly stuff then?

In the next exciting article, I shall be doing more tutorial work on QDOSMSQ again. After that, we'll be back with more instruction type decoding and, if there is enough space, some or all of the support sub-routines as well. Until such time as we get those typed in, this file will probably not assemble - too many 'invalid references to labels that don't exist yet. See you then.

QL Serial Ports

Following an article by Jonathan Dent in the last issue of QL Today, a brief exchange of views (polite!) occurred on the QL Users Email Mailing List between Tony Firshman and Jonathan Dent concerning QL serial ports. Here is a summary of Tony's responses ('TF:') to Jonathan's articles and some subsequent feedback from Jonathan ('JD:').

TF: Jonathan Dent's article in QL Toady Vol 5 Issue 5 needs comment, over and above saying that it is potentially the most important software since QPAC2 and pointer environment. He says the QL cannot control data flow from the modem to QL. It always could, right from day 1.

JD: Only if the modem will let it. It maybe that certain modems have a limited buffer now but that was certainly unusual at day 1. The point is that a proper V.24/V.28 or RS232 data terminal equipment (DTE) interface should not require the data flow from the modem to be controlled. Data flow from the remote DTE can be controlled at a higher protocol level (e.g. by x-off, x-on), when buffers are getting full. In my experience super-Hermes fulfils this requirement.

TF: If SER2 DTR is connected to the modem RTS, then it will control flow from the modem to QL. The QLs 'DTR' is in fact more correctly described as 'RTS'. SER2 CTS lines on both QL and modem connected together gives the output control from QL to modem. The modem though can only store a limited amount of incoming data and when this buffer is full, then incoming data _will_ be lost if it cannot be sent to the computer. The point of RTS is to temporarily stop input if the computer is busy. The 8049 does this job on the QL. The original 8049 on the QL often did not do this correctly. For instance if it was generating sound, it would 'forget' to set RTS. If data arrived in the 8049, and it was unable to send them to the 8302, then the pointers to next data got out of step. Only a full power down could correct this fault, as QL 'reset' does not reset the 8049. Hermes corrected this problem, and allowed an incoming baud rate of 19200. Actual incoming data rate depends on processor speed and current tasks, but can be up to about 12500bps.

JD: RTS (request to send) is "normally" (V.24) used to turn on the transmit function; mainly useful for half duplex communication. I don't exclude the possibility that certain modems can use it

differently, in fact I would be very pleased to find that this is the case. Can you tell me what modem can work like you describe (and perhaps the AT command required). If a modem can be persuaded to work like that I think soql could be used with a pretty basic black-box QL

TF: He also says superHermes gives 9600 from ser2. What he should have said was super-Hermes (including sH LITE) gives up to a full 19200bps incoming throughput on both SER1 and SER2. The original QL has always had 19200 output via the 8302.

JD: I DO recommend swapping to sH LITE in a black-box or a full sH for a re-housed QL

dbf2htm

Per Witte

This little routine takes a standard Archive database and outputs it to file as a neatly arranged html table, viewable in most browsers. Simply load the program into Archive, arrange the table you wish to output (select, order, etc) and type

```
dbf2htm;'outfilename','title'
```

at the Archive command prompt. You can very simply embellish your table by inserting the appropriate markup directly in the code.

This version of the routine does not perform any translation of character sets, nor does it protect against illegal html characters (which could seriously mess up the display) so the results may not always be as expected. A slightly more advanced version which does have those abilities is available from me by email at **pjwitte@knoware.nl**, or by post at **P. Witte, A. v. Ostadestr 84, 7944 XV Meppel, Netherlands**.
In the latter case, please enclose a 3.5" disk + the equivalent of £1, or no disk and £2.

Comments, suggestions and improvements welcome!

```
proc dbf2htm;fnm$,tit$
  rem Output database as html table
  rem Operates on current db
  rem  pwitte march 3rd 2001
  rem V0.00
  spoolon fnm$ dump
  cls : print "Htlm-ising: ";fnm$;", title: ";tit$
  lprint "<html><header><title>";tit$;"</title></header>"
  lprint "<body><center><h1>";tit$;"</h1></center><br>"
  lprint "<table align=center border=1>"
  let i=0: lprint "<tr>"
  while i<numfld()
    lprint "<th align=center><h3>";fieldn(i);"</h3></th>";
    let i=i+1
    endwhile
  lprint "</tr><tr>"
  first : let c=0
  while not eof()
    let i=0: let c=c+1: print at 4,4;c;
    while i<numfld()
      if fieldt(i)=1
        let v$=fieldv(i)
        if len(v$)=0: let v$=" ": endif
        lprint "<td>";v$;"</td>";
        else
        lprint "<td>";fieldv(i);"</td>";
        endif
      let i=i+1
      endwhile
    lprint "</tr>": next
    endwhile
  lprint "</table></body></html>"
  spooloff : print : print "Done!"
  endproc
```

London Quanta Group meetings

Malcolm Cadman

Malcolm Cadman describes the activities of the London Quanta sub-group

Just one of my occasional mails to encourage you all to come along !
We just had our February meeting today, (11th February), the address is always in the Quanta magazine. Yet if you are not a Quanta member here are the details.

2.00pm to 6.00pm, second Sunday of the month. In basement of the Welsh Congregational Chapel, 90 Southwark Bridge Road, London SE.

Nearest tube is Borough. Plenty of parking space for cars/motor bikes. Cyclists can bring the bike down into the basement!

Cost - £2.50 per attendee. A really good quality hot drink of coffee or tea available at modest cost - with free biscuits and good company!

This meeting was an interesting one, as on the hardware front we had 2 portable PC's running QPC1 and QPCv2. As well as an Apple IMAC portable, which hopes to get QPC running soon under the Real PC emulator. Portables are just that, so convenient to bring to meetings.
The IMAC owner was also experimenting with setting up a WAP enabled mobile phone, to give internet access at the meetings.
I am hoping to have a Pandora QL system available at meetings soon. This is still being worked on by one of the members since its donation to the group. As well as a basic black box QL system.
That is the mixture of QL equipment available at the venue, as well as the equipment that members bring themselves.
As you will perceive we are involved with a variety of ways and means of keeping up an interest in the QL. Your interest and support will be most welcome if you would care to come along. No need to be a Quanta member if you do not wish. No need to attend every meeting, although regular attendance keeps the hire of the venue paid for.
If you have any 'languishing' QL equipment that you would like to donate to the group, then this would be welcome.
Also any software, books, magazines, etc, which can all be used to raise some income at future London Quanta Group Workshops. Remember what you no longer want, someone else may be looking for !
Just contact me, or simply call in at the venue with the 'gear'.

Colours on QPC or Not!

Peter Fox

The arrival of colours on QPC2 has been greatly anticipated and very warmly received but there are still various operations which work much better under QPC1, for example, Text87 and disk operations.

In an attempt to backup QXL.WIN's through QPC2, I came to the conclusion that using QPC1 would be better and therefore installed it. This meant that the boot programme which is shared by both emulators needed modifying since there is no 'DISP_COLOUR' instruction in QPC1. It therefore occurred to me that it would be very helpful if there was some way that 4 colours could be set up by QPC1 and 65,536 colours set up by QPC2 within the boot process.

Marcel confirmed that this was possible using a function 'QPC_HOSTOS'. It returns:

```
0 under DOS
1 under Windows 95,98 and ME
2 under Windows NT and 2000
```

Clearly it is more portable if the condition is placed on DOS since either 1 or 2 returned will permit 65,536 colours and so the line in your boot file could be as follows:

```
IF QPC_HOSTOS=0 THEN DISP_SIZE 800,600:
ELSE DISP_COLOUR 3,800,600
```

This will set up a screen 800 pixels across by 600 pixels down in 4 colours under QPC1 and 65536 colours under QPC2. If you use a different screen size, simply change the values to match the screen configuration in your boot file.
It is remarkably simple and well done Marcel for making it so!

For anyone who now thinks that they will install QPC1, this is not quite as straightforward as it looks. 'WININST.exe' tells you that you can allocate memory for QPC1. It does not work, this can only be done using 'Config.exe' which allows you to configure SMSQ/E for memory allocation and a number of other things and is absolutely necessary if only to be able to match your WIN numbers to the QXL.WIN files on the C:\ or any other partition.

Anyone who has purchased QPC2 recently, and for the first time, will find QPC1 as a folder on the QPC2 disk. The files in that folder need copying to C:\QPC and once 'WININST.exe' and 'CONFIG.exe' have been configured, it should run.

Comment from JMS: if you are happy with QPC2 (and there is no reason why you should not) better forget about installing QPC1. It is lacking many features of QPC2. There does not seem to be any interest in QPC1 anyway left, since QPC2 works so well. We consider removing it from the disk in the future, and we have stopped advertising QPC1 anyway. Removing it from the manual will make the manual more readable either, as it is a mix of QPC1 and 2.

Programming ProWesS in SBASIC - Part 6

Wolfgang Lenerz

Last time we covered the tags for loose items. Now we'll have a look at how Prowess sets up your window with these items (and the other objects). Indeed, if you know the Pointer Environment and the way windows are set up there, you'll notice that under Prowess you have no need to position your items in the window. This is done automatically by Prowess. The automatic positioning is mostly a boon, since you don't have to worry about the window itself, but can sometimes also be quite irritating, when you are trying to set up a window just so, and Prowess (seemingly) doesn't let you.

A little example

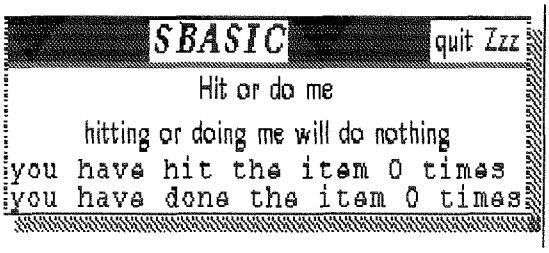
First a little example - it is an example we've already seen more or less like that and which is a copy of one of the example procedures in the example1 program supplied with the Prowess S-Basic Interface (to save you from having to type it in), so I won't comment on it much:

```
100 set_windows
110 test1
115 :
120 DEFine PROCedure test1
130 LOCal object, hit%,hits,dos,times$,mhit$,mdo$
140 LOCal loop%,add_info
150 :
160 : REMark first initialise some variables
170   mhit$="you have hit the item ":times$=" times":hits=0
180   mem=0:object=0:hit%=0
190   mdo$="you have done the item ":dos=0
200 :
210 : REMark now make some strings, note the chr$(0) at the end!
220 :
230   my_hit$=mhit$& hits&times$&CHR$(0)
240   my_do$=mdo$&dos&times$&CHR$(0)
250 :
260 REMark now create the outline object
270 :
280   outl=PWcreate(0,PW('TYPE_OUTLINE'),PW('OUTLINE_QUIT'),
                    PW('OUTLINE_SLEEP'))
290 :
300 REMark now create the item objects
310 :
320   item1=PWcreate(outl,PW('TYPE_LOOSE_ITEM'),
            PW('LOOSE_TEXT_COPY'),"Hit or do me",
            PW('LOOSE_ACTION_DO'),DO_ROUTINE,
            PW('LOOSE_ACTION_HIT'),HIT_ROUTINE)
330   item2=PWcreate(outl,PW('TYPE_LOOSE_ITEM'),
            PW('LOOSE_TEXT_COPY'),"hitting or doing me will do
            nothing")
340 :
350 REMark now we create two infostring objects
360 :
370   info1=PWcreate(outl,PW('TYPE_INFOSTRING'),
            PW('INFOSTRING_TEXT'),my_hit$,PW('INFOSTRING_AUTOSIZE'),0)
380   info2=PWcreate(outl,PW('TYPE_INFOSTRING'),
            PW('INFOSTRING_TEXT'),my_do$)
390 :
400 REMark the main loop
410 :
420   REPeat loop%
430     mem=PWactivate(outl,mem,object,add_info,hit%)
440     IF NOT mem:EXIT loop%    : REMark if mem is
                                   returned as 0, we quit the window
```

```
450     SELect ON object
460       =item1
470         SELect ON hit%
480           =0:hits=hits+1
490              my_hit$=mhit$&hits&times$&CHR$(0)
500              PWchange info1,PW('INFOSTRING_TEXT'),my_hit$
510           =1:dos=dos+1
520              my_do$=mdo$&dos&times$&CHR$(0)
530              PWchange info2,PW('INFOSTRING_TEXT'),my_do$
540          END SELect
550     END SELect
560   END REPeat loop%
570 :
580   PWremove out1
590 END DEFine test1
600 :
610 DEFine PROCedure set_windows
620 LOCal not_compiled,upper%,xo%,yo%,xs%,ys%,ysize_0%
630   not_compiled=IS_OPEN(#0)      : REMark window#0 open?
640   IF not_compiled
650     xs%=0:ys%=0:xo%=0:yo%=0
660     PWscrsize#0,xs%,ys%,xo%,yo%
670     upper%=28:ysize_0%=50
680     PWoutln#0,xs%,ys%-upper%,xo%,upper%+yo%
690     WINDOW#0,xs%,ysize_0%,xo%,ys%-ysize_0%
700        WINDOW#1,xs%    DIV    2,ys%-upper%-ysize_0%,xo%+(xs%    DIV
2),yo%+upper%
710     WINDOW#2,xs% DIV 2,ys%-upper%-ysize_0%,xo%,yo%+upper%
720     BORDER#1,1,255:BORDER#2,1,255
730     PAPER#1,2:PAPER#2,7
740     INK#1,7:INK#2,2
750     CLS#0:CLS#1:CLS#2
760   END IF
770 END DEFine set_windows
```

As usual, I've cut some lines up for more clarity when reading this, but you should, of course, use one line only.

If you run this program, you will get a window with the title of the program (SBASIC), a quit item and a move item in the title bar. Underneath that, you see the first loose menu item, beneath that the second loose menu item, followed, always underneath, by the info items. You can see the result in figure 1.



We know now that, by default, Prowess seems to put objects underneath each other, in the order of creation. Whilst this is not always strictly true, it is a good starting point. Now let's fix ourselves another goal. Suppose we want to have both items next to each other and not one beneath the other, the rest remaining unchanged. Can I achieve this? Of course I can, but I'll have to work a bit harder and tell Prowess to do so, by using a "system tag". A system tag is one you can use on the entire system or only parts of it, or only on one object. But these tags are the same for all objects.

Positioning tags

What we want is a tag that lets me put an object to the left or the right of another one. There are two tags for this. These tags should be used **when you create a new object**, and determine something about the way this object is seen by the system.

PW('POSITION_RIGHT_OF')

This tag takes one parameter: the object which should be to the left of the newly created object. With this tag, we indicate to the system, that we want the newly created object to be to the right

of the object that we have given as parameter to the tag.

PW('POSITION_LEFT_OF')
There again, this tag takes one parameter, the object which should, this time, be to the right of the newly created object. With this tag, we indicate to the system, that we want the newly created object to be to the right of the object that we have given as parameter to the tag.

In our example, we want to have the two loose menu item objects next to each other, i.e. item 1 to the left of item 2. To achieve this, we'll use the *PW('POSITION_RIGHT_OF')* tag on the second item. We can't use the *PW('POSITION_LEFT_OF')* tag on the first item because, when we create the first item, the second one hasn't been created yet, and we need that to pass as parameter for the tag. So, we'll use the *PW('POSITION_RIGHT_OF')* tag on the second loose menu item object.

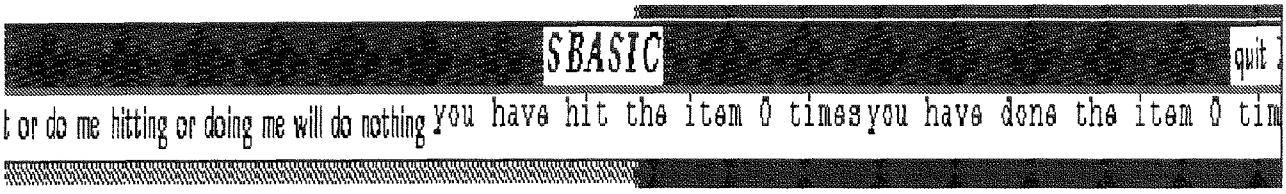
To achieve this, add the following

```
,PW('POSITION_RIGHT_OF'),
item1
```

to line 330, just before the final parenthesis so that this line now becomes:


```
330 item2=PWcreate(outl, PW('TYPE_LOOSE_ITEM'), PW('LOOSE_TEXT_COPY'), "hitting or
doing me will do nothing",  PW('POSITION_RIGHT_OF'), item1)
```

Run the Program. Now you get something approaching figure 2. That wasn't really what we had in mind, was it?



Let's try to figure out what happened. Prowess DID put item2 to the right of item1. That's what we wanted. But then, it put info object info1 to the right of loose item object item2. Why did it do that, when, before, it put the info object underneath the items?

Directions

Prowess works with what I call "directions". It will put objects in the same direction (horizontal: left to right / vertical: downwards) unless told to do otherwise. It is as if Prowess thought in rows and columns. If you look at the window, there is the title bar, and next to it (HORIZONTAL direction) the quit and move items - we're on the same row. Then, underneath, is a "separator" (more of which later, it is just a line to separate objects). Since the separator was put underneath the title bar, we've simply changed direction - vertical. That way, the new item objects item1 and item2 were originally underneath the separator (and the info items again underneath). When we used the PW('POSITION_RIGHT_OF') tag, we simply told the system that we **wanted to change direction.** Thus, Prowess obediently put the second item to the right of the first, and interpreted this request as a change of direction. So now the direction is vertical, and the new objects (the info items) will also be positioned with that direction, i.e. to the right of item1.

How do we get around this? If you look in the manual, you will see that there are also two other direction tags:

PW('POSITION_ABOVE')

This tag takes one parameter, the object which should be below the newly created object. With this tag, we indicate to the system, that we want the newly created object to be above the object that we have given as parameter to the tag.

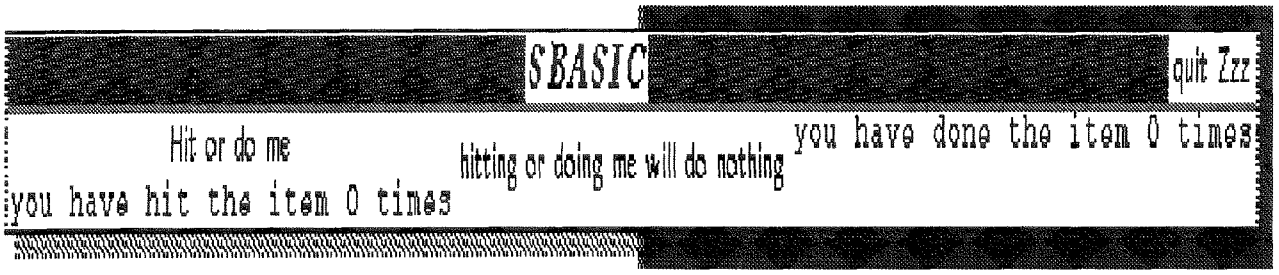
PW('POSITION_BELOW')

This tag takes one parameter, the object which should be above the newly created object. With this tag, we indicate to the system, that we want the newly created object to be below the object that we have given as parameter to the tag.

Perhaps we can use these tags, and try to put the first info item object under loose item object item1. Then, having changed direction, the next info object should be beneath info1. Add "PW('POSITION_BELOW'),item1" to line 370, so that this line now reads:

```
370 info1=PWcreate(out1,PW('TYPE_INFOSTRING'), PW('INFOSTRING_TEXT'), my_hit$,
PW('INFOSTRING_AUTOSIZE'), 0, PW('POSITION_BELOW'),item1)
```

and run the program. You get figure 3. Again, this is not exactly what we expected:



Info object info1 is indeed beneath item1, but the others are still to the right. Apparently, the POSITION_BELOW tag does not change direction as expected. So let's try something else. Here are two new system tags which, again, can be used when creating almost any kind of object:

PW('POSITION_NEXT_ROW')

This tag takes no parameter. The newly created object (for which the tag is used when creating it) will be the first object in a new row, which is positioned at the bottom inside the parent object.

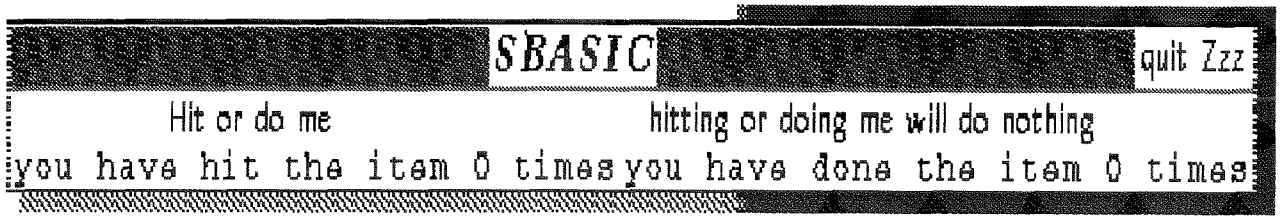
PW('POSITION_NEXT_COLUMN')

This tag also takes no parameter. The newly created object will be the first object in a new column, which is positioned at the right inside the parent object.

The first of these seems promising. Indeed, we want a new "row" to start after the two loose menu items objects. We'll re-amend line 370, taking out the PW('POSITION_BELOW') tag we've inserted earlier, and use the PW('POSITION_NEXT_ROW') tag instead, so that line 370 becomes:

```
370 info1=PWcreate(out1,PW('TYPE_INFOSTRING'), PW('INFOSTRING_TEXT'), my_hit$,
PW('INFOSTRING_AUTOSIZE'), 0, PW('POSITION_NEXT_ROW'))
```

Run the program again. That's more like it, now we get a window as shown in figure 4: the two loose menu items next to each other, followed, underneath, by the two info items next to each other.



Ok, so now we have achieved our primary goal : we have the two loose menu item objects next to each other as we set out to do. Now, we might want the two into items again underneath each other. I'll let you to try and find out how we go about this...

There is actually still more to know about the positioning of items, notably the use of the direction and glue types. We'll come back to these in a future instalment of this series.

QL forveer! W. Lenerz

Indenting BASIC Programs

David Denham

Indenting is a technique used by programs to highlight the structures used. When using structured programming techniques, you need to make sure that all IFs and END IFs, all FOR and END FOR, all REPEATs and END REPEATs, all procedure definitions and end of definitions are matched.
Programmers used to languages like C will be familiar with indenting, where lines of code within given structures are shown to the right of the structure definitions. SuperBASIC programmers can use similar techniques. Here is a trivial example:

```
100 FOR a=1 TO 100
110 PRINT a
120 END FOR a
```

Consider the FOR a loop. If we push anything inside this to the right, we can instantly see and identify at a glance the lines of code within this structure. My personal preference is to add 2 spaces per indent - this seems to give a good, clear degree of indenting without becoming unmanageable with deeply nested structures, e.g. loads of multi line IF.THEN definitions.

```
100 FOR a=1 TO 100
110   PRINT a
120 END FOR a
```

If, when typing in the program, you type in the 2 spaces before the PRINT statement in line 110, SuperBASIC remembers how many spaces you typed, so that when listing the program, it reproduces the spacing, so you can see the indenting.

That was obviously a trivial example. Let us consider a more complex example:

```
100 DEFine PROCedure NUMBER
110 FOR a=1 TO 3
120 IF a=1 THEN
130 PRINT'One'
140 ELSE
150 IF a=2 THEN
160 PRINT'Two'
170 ELSE
180 PRINT'Three'
190 END IF
200 END IF
210 END FOR a
220 END DEFine NUMBER
```

Written in that way, it becomes increasingly hard as the nesting becomes deeper to make sure that you have matched all IFs, ELSEs and END IFs in that kind of routine. If indented in this way it becomes easier to see what's happening. If the END DEFine ends up in the same indent position on the line, chances are you've got it right provided the indenting itself was correct:

```
100 DEFine PROCedure NUMBER
110   FOR a=1 TO 3
120     IF a=1 THEN
130       PRINT'One'
140     ELSE
150       IF a=2 THEN
160         PRINT'Two'
170       ELSE
180         PRINT'Three'
190       END IF
200     END IF
210   END FOR a
220 END DEFine NUMBER
```

All down this routine, you can see that all the ENDs match in terms of position with the corresponding start of their structures, and you

can to some extent read down it and by visual clues (position of the code across the line) match up what belongs where and follow the structures more easily than you can with the original non-indented listing.
I find that the single greatest use of indenting for me is to identify nested IFs, ELSEs and END IFs. It can also be very useful with SELect statements.

```
100 DEFine FuNction NUMBER(n)
110 LOCAL text$,num
120 num=n
130 SELect ON num
140 =1:text$="One"
150 =2:text$="Two"
160 =3:text$="Three"
170 END SELect
180 RETurn text$
190 END DEFine NUMBER
```

We would indent this in this way:

```
100 DEFine FuNction NUMBER(n)
110   LOCAL text$,num
120   num=n
130   SELect ON num
140     =1:text$="One"
150     =2:text$="Two"
160     =3:text$="Three"
170   END SELect
180   RETurn text$
190 END DEFine NUMBER
```

Within the SELect statement, I tend to push the = clauses 2 spaces to the right, and then code in between the = statements 2 spaces to the right again, but I doubt this is significant. My working rule is that whenever I have started a new structure, I push the code contained two spaces to the right, and then the line defining the end of the structure (be it a PROC, FN, FOR loop, REPEAT loop, SELECT clause, IF clause etc) back to the same degree of indent as the defining line, so that I can see neatly where all my SuperBASIC structures start and end.
There is nothing worse than going back to a program a year or two after I wrote it and can't figure out what I was doing when I wrote it! I now indent my programs when I write them, and I even wrote a very simple BASIC program which did the indenting for me automatically, by reading a BASIC program line by line from file, which worked rather like this:

Fetch a line of BASIC
use INSTR to check for DEFs, loop definitions etc
if found, increase indent of next line by 2
check for END DEFs, end of loops etc
if found, decrease indent of this line by 2
fetch next line of BASIC etc etc

This worked, but got a little confused with multi line statements. It was a slow BASIC program after all.
Shortly after I wrote it, I found a few other similar indenting programs written by programmers far more experienced than myself, which worked rather better than mine. These are still not perfect - the example listed below can still be confused by poorly written BASIC or complex multi-statement lines, but works well enough for most simple purposes. It's included with the permission of the author, someone who may be known to readers! Oh, I tried running it on itself and it works! It can be Turbo compiled too, to speed it up a bit. The program tries to remove the original indenting, so you don't have to worry too much about double indenting. And the listing has been processed by the program itself, so you can see what it does (as long as QL Today manages to reproduce it as I intended).

```
100 REMark INDENTING BASIC PROGRAMS by Dilwyn Jones
110 MODE 4
120 WINDOW 448,202,32,15:WINDOW#2,448,200,32,16
130 WINDOW#0,448,40,32,216:CLS#0:BORDER 1,255:CLS
140 CSIZE 0,0:PRINT FILL$('-',74)
150 CSIZE 2,1:PRINT'   INDENTED BASIC PROGRAM LISTINGS'
160 CSIZE 0,0:PRINT FILL$('-',74)
170 PRINT\TO 21;'Copyright (C) Dilwyn Jones, 1990'
180 REPeat get_filenames
190   INPUT\'Indent which program?';inp$
200   IF inp$='' THEN EXIT get_filenames
210   INPUT'Save to where ? ';out$
220   IF out$='' THEN EXIT get_filenames
230   IF NOT(inp$==out$) THEN EXIT get_filenames
240   BEEP 5000,70:PRINT'Names must be different!'
250 END REPeat get_filenames
260 IF inp$='' OR out$='' THEN AT#0,1,0:PRINT#0,'PROGRAM FINISHED.':WINDOW 448,200,32,16:STOP
270 REPeat get_filenames
```

TF Services

Compswitch

A UK 4-way trailing socket designed to switch off computer peripherals automatically when the computer is switched off, or (in the case of an ATX computer) when it auto-powers down. *Compswitch* has one control socket, and three switched sockets.

Cost..... £24

**********NEW**********

superHermes

A major hardware upgrade for the QL

All Hermes features (working ser1/2 at 19200, independent baud rates/de-bounced keyboard/ keyclick) IBM AT kbd I/f // HIGH SPEED RS232 at 57600// serial mouse port and 2 other RS232 inputs// 3 I/O lines // EEPROM

Cost (including manual/software)**£90** (£92/£93)
IBM AT UK layout Keyboard.....**£11** (£13/£15)
Serial mouse .....**£8** (£8.50/£9)
Capslock/scrollock LED .....**£1** (£1.50/£1.50)
Keyboard or mouse lead .....**£3** (£3.50/£3.50)
High speed serial (ser3) lead .....**£4** (£4.50/£4.50)

Hermes available for £25 (£26/£27) **Working ser1/2 and independent input, debounced keyboard.**

SuperHermes LITE: All Hermes features (see above) + an IBM AT keyboard interface only.
Cost (incl keyboard lead) .....**£53** (£54/£55)

QL REPAIRS (UK only)

Fixed price for unmodified QLs, excl microdrives. QLs tested with Thorn-EMI rig and ROM software.

£27 incl 6 month guarantee

Minerva

The ORIGINAL system operating system upgrade

OTHER FEATURES COMMON TO ALL VERSIONS

DEBUGGED operating system/ autoboot on reset of power failure/ Multiple Basic/ faster scheduler- graphics (within 10% of lightning) - string handling/ WHEN ERROR/ 2nd screen/ TRACE/ non-English keyboard drivers/ "warm" fast reset. V1.97 with split OUTPUT baud rates (+ Hermes) & built in Multibasic.

First upgrade free. Otherwise send £3 (+£5 for manual if requd). Send disk plus SAE or two IRCs

MKI...£40 (£41/£43) **MKII...£65** (£66/£67)

MINERVA RTC (MKII) + battery for 256 bytes ram. CRASHPROOF clock & I^2C bus for interfacing. Can autoboot from battery backed ram. Quick start-up.

QL RomDisq

Up to 8 mbyte of flash memory for the QL

A small plug in circuit for the QL's ROM port (or Aurora) giving 2, 4 or 8 mbytes of permanent storage - it can be thought of as a portable hard disk on a card, and reads at some 2 mbytes per second.

Think of it - you could fully boot an expanded QL, including all drivers/SMSQ etc off **RomDisq** at hard disk speed with only a memory expansion needed.

2 mbytes RomDisq.....**£39** (£40/£41)
4mbytes RomDisq.....**£65**(£66/£67)
8 mbytes RomDisq.....**£98** (£99/£100)
Aurora adaptor.....**£3** (£3.50/£4)

MPLANE

A low profile powered backplane with ROM port

A three expansion backplane with ROM port included for RomDisq etc. Aurora can be fitted in notebook case and powered off single 5V rail - contact QBranch for details. Two boards (eg Aurora and Gold Card/Super Gold Card/ Goldfire fixed to base. Suitable for Aurora (ROM accessible from outside) & QL motherboard in tower case. Specify ROM facing IN towards boards, or OUT towards back of case.

Cost ..... £34 (£35/£36)

I2C INTERFACES

Connects to Minerva MKII and any Philips I^2C bus

Power Driver Interface 16 I/O lines with 12 of these used to control 8 current carrying outputs (source and sink capable)
2 amp (for 8 relays, small motors) ..... **£40** (£43/£44)
4 amp total (for motors etc) ..... **£45** (£48/£50)

Relays (8 3a 12v 2-way mains relays (needs 2a power driver) .....**£25** (£28/£29)

Parallel Interface Gives 16 input/output lines. Can be used wherever logic signals are required.....**£25** (£27/£28)

Analogue Interface Gives eight 8 bit analogue to digital inputs (ADC) and two 8 bit digital to analogue outputs (DAC). Used for temp measurements, sound sampling (to 5 KHz), x/y plotting.....**£30** (£31/£32)

Temp probe (-40°C to +125°C).....**£10** (£10.50/£11)
Connector for four temp probes.....**£10** (£10.50/£11)
Data sheets.....**£2** (£2.50/£3)
Control software & manual (for all I/F).....**£2** (£2.50/£3)

QL SPARES

Keyboard membrane ..... **£24** (£25/£26)
1377 PAL ..... **£3** (£3.50/£4)
Circuit diagrams ..... **£3** (£3.50/£4)
68008 cpu or 8049 IPC..... **£8** (£8.50/£9)
8301/8302 or JM ROM or serial lead.....**£10** (£10.50/£11)
Power supply (sea mail overseas)..... **£12** (£17/£21)

Prices include postage and packing (Airmail where applicable). Prices are: UK (Europe /Rest of world). Payment by cheque drawn on bank with UK address/ postal order, Giro transfer (58 267 3909) or CASH! I can no longer accept card payments as UK only does PDQ transaction. SAE or IRC for full list

13 Nov 00

29 Longfield Road, TRING, Herts, HP23 4DG
Tel: 01442-828254 Fax/BBS: 01442-828255
tony@firshman.demon.co.uk http://www.firshman.demon.co.uk

```
280   INPUT\'Indent by how many spaces per step (0 to quit)?';spaces
290   IF spaces›=0 THEN EXIT get_filenames
300   BEEP 5000,70:PRINT'Invalid entry!'
310 END REPeat get_filenames
320 IF spaces=0 THEN AT#0,1,0:PRINT#0,'PROGRAM FINISHED.':WINDOW 448,200,32,16:STOP
330 PRINT\'Indenting, please wait...'\\
340 OPEN_IN#3,inp$
350 OPEN_NEW#4,out$
360 indent=0
370 REPeat get_lines
380   IF EOF(#3) THEN EXIT get_lines
390   IF INKEY$=CHR$(27) THEN PRINT'‹ESC›':EXIT get_lines
400   INPUT#3,in_lne$
410   backwards=0
420   space=' ' INSTR in_lne$
430   out_start=space
440   REPeat unspace
450     IF in_lne$(out_start)‹›' ' THEN EXIT unspace
460     out_start=out_start+1
470   END REPeat unspace
480   out_lne$=in_lne$(out_start TO )
490   IF ('DEFine ' INSTR out_lne$)=1 THEN indent=indent+spaces:backwards=spaces
500   REMark checking for the colons in the following lines allows suppression of indents for short forms
510   IF ('REPeat ' INSTR out_lne$)=1 AND (':' INSTR out_lne$)=0 THEN
      indent=indent+spaces:backwards=spaces
520   IF ('FOR ' INSTR out_lne$)=1 AND (':' INSTR out_lne$)=0 THEN
      indent=indent+spaces:backwards=spaces
530   IF ('SELect ' INSTR out_lne$)=1 AND (':' INSTR out_lne$)=0 THEN
      indent=indent+spaces:backwards=spaces
540   IF ('WHEN ' INSTR out_lne$)=1 AND (':' INSTR out_lne$)=0 THEN
      indent=indent+spaces:backwards=spaces
550   REMark I use the next line for the Turbo pseudo-keyword WHEN_ERROR
560   IF ('WHEN_ERROR ' INSTR out_lne$)=1 AND (':' INSTR out_lne$)=0 THEN
      indent=indent+spaces:backwards=spaces
570   REMark in the next line, an unconditional NEXT is most probably the end of a FOR loop rather than
      part of a REPeat loop
580   IF ('END ' INSTR out_lne$)=1 OR ('NEXT ' INSTR out_lne$)=1 THEN indent=indent-spaces
590   IF ('ELSE ' INSTR out_lne$)=1 THEN backwards=spaces:REMark multi line IF statement
600   IF LEN(out_lne$)›5 THEN
610     REMark look for multi line IF - will fail if THEN omitted
620     IF ('IF ' INSTR out_lne$)=1 AND out_lne$(LEN(out_lne$)-4 TO)=='THEN ' THEN
630       indent=indent+spaces:backwards=spaces:REMark simple single line IF...THEN
640     ELSE
650       REMark try for single line IF... without THEN - it normally won't have a colon
660       IF ('IF ' INSTR out_lne$)=1 AND ('THEN ' INSTR out_lne$)=0 AND (':' INSTR out_lne$)=0 THEN
670         indent=indent+spaces:backwards=spaces
680       END IF
690     END IF
700   ELSE
710     IF ('IF ' INSTR out_lne$)=1 THEN indent=indent+spaces:backwards=spaces:REMark VERY short form of IF!
720   END IF
730   total=indent-backwards:IF total‹0 THEN total=0
740   done$=FILL$(' ',6-space)&in_lne$(1 TO space)&FILL$(' ',total)&out_lne$
750   PRINT done$:PRINT#4,done$
760 END REPeat get_lines
770 CLOSE#3:REMark input file
780 CLOSE#4:REMark output file
790 AT#0,1,0:PRINT#0,'PROGRAM FINISHED.'
800 WINDOW 448,200,32,16:REMark restore default window
810 STOP
```

LAST MINUTE NEWS

Q-CELT

First of all, apologies to anyone expecting to see us at the Hove QL Show at the start of March. Due to a freak snowfall, we couldn't make it, and were snowed in for over a week in 16 inches of the stuff! That coupled with the Foot and Mouth outbreak made for us sadly missing the show.

However, we have been busy. Phoebus Dokos has finished a fantasic new CD-ROM for QLs containing virtually every decent INFOCOM text adventure game ever made: Called the Interactive Fantasy CD-ROM. These have been converted from the PC Public Domain, and uses the QL Version of the Infocom ZIP interpreter (which is included on the CD) to run on the QL. They have been tested on virtually all screen modes and seem to run without any problems. The CD contains hundreds of games, in a large 150Mb QXL.WIN file, plenty to keep you busy this summer! The CD will make its show debut at the QUANTA AGM in April, including some fabulous computer artwork on the CD Cover by Phoebus. Cost will be £5.00 Sterling (plus P&P), with FREE enhancements and upgrades provided by Phoebus himself to all purchasers! These will include additional games as Phoebus converts them, Maps of the game layouts etc. These enhancements will ONLY be available by email - simply give me your email address when you purchase a copy of the CD and I will forward it to Phoebus for inclusion in his list, and all future updates will be emailed directly to you.

As well as this, the ZeXcel Spectrum Emulator CD for QL's will be available from the end of April also. This has been a long time coming, owing to a problem on my part converting files correctly - this has now been rectified. This CD will contain the full version of the ZeXcel Sinclair ZX Spectrum Emulator program for the QL By Ergon (now freeware) and several thousand Spectrum Games, Demos, Utilities and other Programs - enough to keep you interested for the rest of your life. This is a must have for anybody who moved from a Spectrum to a QL all those years ago, and misses those classic games - This will cost £10.

Also, the much awaited QL Emulator CD by Dilwyn Jones is now ready. This CD is stuffed to the gills with quality QDOS Software, as well as QL Emulators for virtually all systems - including Linux, Windows, DOS, Amiga, Mac and ST. There are over 1,000 Programs on the CD now, and it is hoped this CD will help renew some interest in the QL, and maybe bring back some of those former QL users who moved to other systems before the Emulators were widely available. The CD is completely freeware - please copy it at will and give copies to as may people as possible. The main reason for its creation is to promote awareness in the QL.

The "Z88 Heaven" CD we sell (which contains several hundred Z88 programs, as well as documentation and Pictures, adverts, and Z88 Memorabilia, as well as Z88 Emulators for the PC) has also been updated. It now contains a few bug fixes and some new additional programs.

Q-CELT are changing email addresses

The new email address is

qceltcomputing@hotmail.com

The reason for the change is simple - there are, believe it or not, a couple of other companies here in Ireland with similar names, and recently I have got a few stray emails intended for them. After calling Microsoft, I found out there is a shop called QCELT Gifts which uses hotmail, and also a place called QELTIC which does Irish gifts on-line. Therefore, I think changing the email address to something more specific is a good idea. I'll keep checking the old email address for a while yet - maybe a month or so, but when the flow dries up, I will cancel it.

Jochen Merz Software

QSpread 2001 will be ready for release at the Quanta AGM. We have started improving it some time ago, but we held it back because we were hoping to add hi-colour features. However, as we can't tell yet when this is going to happen, and as the other features have been tested well, we decided to release it. The new features make the use of QSpread so much easier in various aspects and contains several bug fixes too. Please see J-M-S ad on page 35 for more details.

We have also improved QMenu again, but due to the work on QSpread it has not been finished. You will get it anyway with most free updates of our JMS software. However, as changes are not obvious, and they will only be used by the new QSpread, there is no urgent need to update QMenu on its own.

Getting into QPAC2 - Part 1

Roy Wood

Five years ago, when QL Today started, Jochen Merz explained his BOOT file. This was quite interesting for many readers, and we were thinking about explaining the BOOT file (which has changed considerably!) again. We may do it in the next issue, if there's enough interest. Discussing it with Roy Wood brought up the idea of publishing his QPAC2 add-on manual in QL Today, which explains BOOT files on a more general level. It assumes, of course, that you own QPAC2. As we still have readers who do not use QPAC2 (which letters prove), we feel it is a good idea.

QPAC 2 is a very much maligned suite of programs and extensions and has gained the reputation in the QL community of being difficult to set up and use. This has occurred mainly because the manual gives little insight to the raw beginner and, like many other manuals written by programmers, assumes an understanding of jargon that makes it inaccessible to the average computer user. I would like to try to overcome this reluctance to wade into the wide world of the Pointer Environment with a short guide to what the program actually does and what the new keywords allow you to do. I apologise if the explanations in this booklet seem a bit simple but I am trying to introduce the concepts behind QPAC 2 to all types of user. I would suggest that you run the Boot file provided and then go through the explanation in the section entitled 'The Boot File Itself.' After this try to experiment with a backup copy of the Boot file by changing some of the commands and adding others. You will soon see that it is not as difficult as it seems at first. One thing that I do need to point out at the start is the difference between a HIT and a DO. A HIT is a selection stroke and can be done with the SPACE bar or the left hand mouse button. A DO is an execution stroke and can be done with the ENTER key or with the right hand mouse button.

So what do I get with QPAC 2

QPAC 2 is, in reality, not a program but a collection of programs and extensions to the standard QDOS. When it is loaded into the system several completely separate procedures and functions become available to the user and a whole batch of new keywords are added to SuperBasic. I will deal with these separately in a later section but, for the moment, I will give you an overview of the things that QPAC 2 can do.

The Button Frame

The button frame is the first thing that many people think of when they see a system running under QPAC 2. This is typically a row of buttons across the top or down the side of the screen which contain the names for programs or devices all of which can be instantly called up or brought into use. There is no necessity to actually have this button frame visible at all. QPAC 2 just makes it available for you to use.

Procedures

QPAC 2 loads a collection of short procedures into the resident program area of the QL's memory and these programs can be called up directly from the command line, by giving them a 'Hotkey' or by creating a button and placing that in the button frame. These procedures mostly perform functions that are already available in the QL's operating system but do them in a much easier way. They also provide more information to the user than the in-built functions do. Other functions, such as the 'Stuffer Buffer' and 'Sydef' are not provided by the original system.

Files

At some point in your use of the computer you will need to copy, delete, rename or update a file. You can, of course, do this by going to the command line and typing 'COPY flp1_myfile TO flp2_myfile' but, if you want to copy more than one file from a disk that is a tedious process. QPAC 2 allows you to set up files menus from the various devices attached to your QL and then to perform operations on selected files. In addition to this you can display the size, type and date of the files.

Extensions

QPAC 2 links a number of new keywords into the QL's operating system. Many of these have to do with the button frame or setting up the 'HOTKEYS' that allow you to call or load the various programs that you want to run but others relate to the 'Stuffer Buffer' or control of the QL's heap.

The Pointer Environment Extensions

There are three extension files that have to be loaded before QPAC 2 can be used and these are the files which control the Pointer Environment itself. There is a common misbelief that, in

order to use the Pointer Environment, you have to have a mouse. Any pointer driven program can be operated quite easily with a mouse although some of them do work better if one is present. All of the Pointer driven product sold by Q Branch have menus that are activated by use of the function keys (F1 to F10) or by combinations of the CONTROL key and a selected letter the function of the Pointer Environment files is as follows:

Ptr_gen
This is the extension that puts the little arrow into the active window on the screen and then allows it to be the active cursor. All this means is that you can move the little arrow to whenever you want by using a mouse, trackball or the cursor keys and then press either 'ENTER' or 'SPACE' to activate that item. (One interesting point that many users of PCs may have noticed is that the pointer generated by Windows cannot be moved by the cursor keys - score another one for the QL)

Hot_rext
This extension provides all of the keywords that create the HOTKEYS to allow the computer to perform set tasks at the press of the 'ALT' key and a selected letter. For instance, if you had set up a line in your Boot file that said:

```
ERT HOT_KEY ('?',CHR$(177))
```

pressing 'ALT' and the '?' key would result in õ being printed. There are many other uses of the HOTKEY system but these are adequately explained in the manual. One particular hotkey I find extremely useful is:

```
ERT HOT_KEY ('Q','CHR$(240)&"Q"&CHR$(10))
```

which simulates pressing the F3 key (used by many programs to call up the commands menu) followed by the letter 'Q' (used by many programs for 'Quit') followed by the 'ENTER' key to activate the command. This is a quick way to exit many programs.

Wman
This extension is the one which controls the windows that the programs use. It saves the current window when another is drawn over it and then restores that window when it is picked to the top. If anything has been changed since it was buried then that is written in. This is, quite literally a Window Manager.

Writing a Boot file

The first thing that makes many inexperienced users quake is the thought of writing a boot file. This is the file that the computer looks for when it is started or reset and it tells the computer exactly what it has to do, how it has to behave and which programs to load.

All commercial and most Public Domain programs have some kind of Boot file attached to them. QPAC 2 caused a stir because, although there is a Boot file on the disk it actually tells you not to use it and to write your own. The QL itself gives you all the tools that you need to write this Boot file and the Toolkit 2 'ed' command will even check the syntax as you go to allow you to ensure that you get it right. This is assuming, of course, that you have already loaded the QPAC 2 file with a line such as: LRESPR flp1_QPAC2. Once this is done the extensions, procedures and keywords are linked into the operating system and the editor can recognise them as you write the Boot file. A Boot file can be very short, consisting of as little as 20 lines and loading the bare minimum of programs and extensions or it can be a huge beast (mine is over 200 lines). The first file I wrote when I started using QPAC 2 was of the short variety but, as you will find out yourself, you are never quite satisfied with the result and the file tends to evolve over time. As soon as you acquire a hard disk and a decent sized memory expansion the number of programs available in one go increases and the Boot file does too. I tend to use QD for my editing because of its enhanced features but the are many editors available and you can even use Quill provided you remember too export the finished article in ASCII format.

So How do I get Started?

The first thing that you need to do is to think about which programs that you want to load and how you want them to be present in the system. In the early days of the QL, when memory expansion was rare, most people would place a microdrive into mdv1_ and reset the computer every time they wanted to start a program. This would activate the Boot file on that disk and load the required program. When the user had finished using that program the process was repeated for the next program and so people never ever looked at the contents of these Boot files. As memory expansion became more available it became easier for more programs to be loaded at the same time and full multitasking ability of the QL came into play. Despite this a large number of QL users still followed the previous procedure and were unaware that they could switch between tasks very easily. There were a few programs available for the early QL which made multitasking easier. These included Choice (for ICE users), Taskmaster, and QRAM a precursor of QPAC 2. What give QPAC 2 such an advantage over these earlier systems is the ability to specify how you want the system to use the chosen programs. There are three ways that you can introduce an executable program to your system in the Boot file:

1. HOT_RES or HOT_RES1

This command will load a program into the resident program area. This means that the program is loaded but not started and that memory is permanently locked up until the computer is reset. If you add the '1' to the end of the HOT_RES command there will only be one version of the program started at a time. The choice of which version of the command to use is dependent upon the use to which you put the program. If you want several copies of this program to be available to you so that you can use them with different data and cycle between them then the first version is the one you should choose. If you only want to use one version then you should choose the second version. eg: You can load more than one file into Text 87 and move between the windows displaying these files so I load it with the following line:

```
ERT HOT_RES1 ('t','Win1_WP_Text87plus4')
```

QSpread, on the other hand, can only handle one file at a time so if I used it frequently I would use the version of the program that made more than one copy available in this way:

```
ERT HOT_RES ('s','Win1_progs_Qspread')
```

The main difference here is that pressing 'ALT/t' will always give me the same copy of Text 87 whereas pressing 'ALT/s' will give me new versions of the program each time.

2. HOT_LOAD and HOT_LOAD1

These two commands will load and execute the program chosen in the HOT_LOAD line. As with the previous command the program will be loaded again each time the HOT_KEY is pressed unless the '1' is placed after the command. If a program that has been HOT_LOAD'ed is buried or put to sleep then

keying the HOT_KEY or DO'ing the button will start a new version of that program. If, however, the program has been HOT_LOAD1'ed and then buried or put to sleep the call to the HOT_KEY will call the original version of that program to the screen.

3. EX, EXEC, EXEP etc.

These commands are the ones most familiar to uses or QDOS and are not part of the QPAC 2 code. I have included them here, however, to complete the list of the ways you can start a program from your Boot file. for a fuller explanation of the various ways to use these programs and their variants (EXEC_W, EW etc.) see either the QL User Guide or Jan Jones Superbasic Handbook. The programs that are part of QPAC 2, such as SYSDEF JOBS, CHANNELS, RJOB, EXEC etc can all be started with an EXEP command such as:

```
EXEP 'RJOB'
```

but QPAC 2 has a more elegant way of dealing with this if you want to use the button frame:

```
BT_SLEEP 'RJOB'
```

will produce a button with the word 'Rjob' in it which will start the program whenever it is DOne.

Next issue, we will have a look at the Boot file itself.

Cover Disk Volume 5

Since the last cover disk, George Gwilt has upgraded the Turbo Compiler to version 4.7 and also upgraded the Turbo Toolkit to version 3.28, so this cover disk includes copies of the new versions. The Turbo Toolkit files are included within the Turbo compiler package.

We promised to deliver the "missing" part of Turbo with this issue. This is the long awaited TurboPtr package, a package which finally allows you to compile pointer driven programs with Turbo.

Readers of Norman Dunbar's assembler programming series may be using the Gwasl assembler which we featured on a previous cover disk. George Gwilt has now upgraded this program to version 1.6, to address a small problem with the ‹ea› mode instructions. So, as there was space available on this cover disk, we decided to include a copy of Gwasl v1.6. Gwasl's name comes from Gwass-Lite, the "lite" version of George Gwilt's Gwass assembler for 68020 and higher processors. Gwasl is a slightly cut down version which works on 68008 QLs.

We have also added J. Grimberts sprite editor, reviewed in the previous issues of QL Today.

All of the programs have been compressed with the Info-zip program, and a copy of Unzip is provided to decompress the packages for you, along with a short BASIC boot program.

Further information about Info-zip and Info-unzip, along with the latest full versions of these packages for the QDOS/SMSQ platforms may be obtained from Jonathan Hudson's website:

http://www.bigfoot.com/~jrhudson/

The packages supplied and their filenames are:

turbo47_zip	Turbo compiler v4.7 and Turbo toolkit v3.28
tptr24_zip	Turbo Pointer package v24
gwasl16_zip	Version 1.6 of the Gwasl assembler
sprted_zip	Sprite Editor
unzip	Unzip version 5.32
boot	A short BASIC program to help you decompress the zip files

To decompress the packages, you will need to put the cover disk (or a copy of it) in drive FLP1_ on your system, and if decompressing to floppy disk you will need one blank, formatted disk per package you intend to decompress.

If you only have a single floppy disk drive, you should be able to unzip the packages to a ramdisk and then re-copy the files onto another floppy disk.

Start the BASIC program with

```
LRUN FLP1_BOOT
```

You will be asked to enter a number from 1 to 4 to identify the package you wish to decompress.

Then you will be asked to specify the name of the drive holding both the unzip program and the package to be decompressed (i.e. the cover disk).

Finally, you will be asked where to decompress the files to - enter the name of the destination drive/directory. There will be a pause while unzip loads, then decompression starts. When it tells you it has finished, press any key to close the unzip program.

In case of a faulty or unreadable disk, please return the disk together with one International Reply coupon to either distributor.

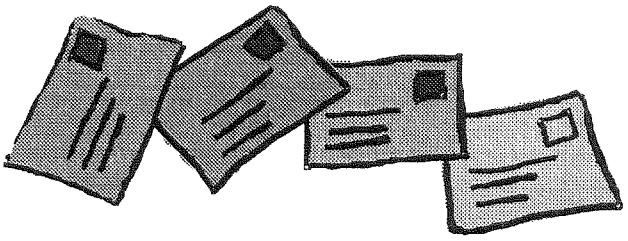
In the last issue Wolfgang Uhlig expressed his opinions on the state of QDOS/SMSQ/E hardware and software development. In this issue Peter Graf would like to express a contrary view. As always, the views put forward are those of the authors and not of the magazines' owners and editors.

Opposite View

Peter Graf

In his article "Q40 or QPC" Wolfgang Uhlig aswers my contribution, which was only published in the German issue.
The author claims that I would scream and complain about faster chips, bigger harddisks and new peripherals. My supposed opinion is called "ridiculous" and decorated in a way that makes it look really silly. Nonsensical things like "yes, only faster, bigger and so on!" are quoted as if I said them. My name is directly mentioned.
These and other things have been arbitrarily invented by Wolfgang Uhlig. They have absolutely nothing in common with my opinion. It's the opposite: I am very interested in faster CPU's, like to use big harddisks and modern peripherals. Indeed the Q40 was a step to make QL style hardware fast and more modern.
Furthermore the QL Today author calls supporters of QL Hardware "arrogant" and "unbelievable stupid". As the only person who defended QL hardware in this QL Today discussion, I feel personally insulted by these words.
The author has confirmed by email that my person was meant with "ridiculous" and the rest of the according paragraph. He is not willing to take back any of his wrong statements or insulting phrases. Under such circumstances a proper discussion isn't possible, so I won't answer on the technical matter of Q40 or QPC.
I find it unfair that QL Today published this attack against my article, without giving the reader the opportunity to read the original before. (It was only published in the German issue.)
I would find it good if the QL Today author and the editors could feel some regret. Such punches against those who work hard to bring progress to QL hardware are sad and de-motivating.

Letter-Box



Ian Pizer writes about printers and colour:

After much hesitation, and after many fruitful years with an EPSON Stylus 800 (B/W) printer, I finally bought an EPSON Stylus Colour 880 which has full ESC/P2 emulation. It works correctly both with Aurora and QPC. One must be careful with EPSON's documentation. For the 680, even the rep. at www.EPSON.de got it wrong. It does not do full ESC/P2 though the 980 does. www.EPSON.de is better than www.EPSON.co.uk and can also be read in English. They answer queries in English if you ask by FAX.
The new StylusColour2_pfd (from RWAP) printer driver for Prowess also works and prints the LineDesign COLREP_ldp file successfully from Aurora and also from QPC. This page file provide the key numbers for 18 colours and 9 grays, plus black and white. You can use these numbers in LineDesign to indicate the "fill colour" (see Attributes) so you can use colours for designs and text (you do not see colours on the LineDesign screen but the chosen colours are printed).
Paragraph2.03 also prints colour from QPC and all the colours I tried printed the correct colour.

When you hit PRINT in either LineDesign or Paragraph you need to ensure that you have set the correct default driver. This means you have already loaded the new EpsonColour2_pfd driver to pws_pf_driv_ and that in pws_mine_proforma_cfg you have included the new StylusColour2_pfd driver.
On the whole I am very pleased with the possibility of using colour for text and/or images. One can live without colour but it does add a pleasant new dimension when appropriate. I offer thanks to Rich Mellor and Roy Wood for their chromatic help.

RWAP SOFTWARE

QL Cash Trader v3.7 £5

A well established accounts package for the small to medium sized business, including automatic generation of profit & loss account, balance sheet, VAT returns, reports and analysis for audit trails and management decisions. Previously sold for over £100.*

QL Payroll v3.5 £5

Manage a payroll for a small to medium sized business. Handles up to 99 employees easily, producing P45s and P60s as well as the payslips on a monthly or weekly basis. Calculates tax and national insurance and is easy to update to take account of the current tax year rules.

Q-Help v1.05 £10
Q-Index v1.04 £5

Q-Help: on-screen help for SuperBASIC commands, including TK2, Turbo Toolkit, SMSQ/E and PD toolkits. Can be used to add help to your own programs - simply produce ASCII text for each help page, add an index and Q-Help automatically cross-references and displays the links.
The PD toolkits referred to are available for £2.
Q-Index: The SuperBASIC index supplied with the Reference Manual - enter a topic such as 'screen resolution' and find out the commands which relate. Launch Q-Help for further info on the chosen command.

Sidewriter v1.08 £10

Produce landscape printouts of Easel/Qspread spreadsheets and output from QL Genealogist, as well as any other standard text file. You can specify the fonts to be used on the page. Works with all EPSON compatible printers, from 9 pin dot matrix to laser printers. A most useful utility by Dilwyn Jones - you know it must be easy to use.

ProForma FSC/P2 Drivers v1.03 £8

New improved colour and monochrome printer drivers, providing up to 720dpi for all programs written for use with ProWesS, such as LineDesign and Paragraph. Works on all Epson inkjet printers which support binary mode compression (740, 850 and 900 models at least). 1440 dpi to follow.

QL Genealogist v3.26 £20
Genealogy For Windows £50

Store your family tree for posterity. Add individuals with details of their parents and children, watch all of those links build up into a formal family tree layout. Text files and pictures may also be linked to individuals as well as notes and events, making this the perfect way to preserve the history of your family.
QL version now supports FileInfo II and QMenu as well as allowing you to link both male and female trees. Sample tree of the Royal family since 1066 included.
PC version is event driven - enter the details as they appear in documents and it generates the tree from these. QL data and GEDCOM can be transferred to the PC version.
Both programs easy to use and complete with a step by step tutorial.

Image D v1.03 £10

Produce graphical representations of 3D objects - view them as wireframe, hidden line and shaded. Perspective and magnification can be controlled and views can be saved to file for subsequent printing. Multiple objects can be defined and positioned relative to each other. Simple to use yet produces excellent results.

SBASIC SuperBASIC Reference Manual £40
Updates £6 each, £10 for 2 (Current Version Rel 3)

Have you ever tried to write a program, but been lost as to the means of performing a certain action? This Reference Manual provides you with a full description and examples of how to use all of the keywords found on each of the different QLs, plus SMSQ/e, Toolkit II and many different public domain toolkits. Details of any possible problems are provided, together with descriptions of how to use the device drivers and how to ensure that your programs are compatible across the range of QL platforms.
This book is ideal for all QL users and is kept up to date with regular updates.
Orders are currently being taken for the next print run of this popular tome.
(Note: Price for the book does not include post & packing).

QL Cosmos v2.04 £5

Ever wondered what the stars in the sky looked like 100 years ago? Or, maybe you want to learn the constellations and names of what you see in the sky. This is the program for you - generates pictures of the stars and planets for any given place or time and provides details on these objects. Includes Halley's Comet, the Moon and the Solar System planets.

Q-Route v1.08C £25

The latest version of this popular route finding program. Find the quickest route or the shortest route between any two places, using roads. A wide range of maps is available for this program (see elsewhere in this advert). The program is easy and quick to use. You can even add your own places and roads to the maps to include local detail.

Flashback SE v2.03 (Upgrade only) £5

The ultimate database program - extremely fast and flexible, easy to use, updated to cope with the latest versions of the QL operating system and still maintained. A report module is included to allow you to format output in any way, including mail-merge. Unfortunately only available as an upgrade from the original version (original still available from Sector Software).

Return To Eden v3.08 £10
Nemesis MKII v2.03 £8
The Prawn v2.01 £8
Horrorday v3.1 £8
West v2.00 £5
The Lost Kingdom of Zkul v2.01 £5

A wealth of QL adventures - mainly text only.
Save the Galaxy from the ambitions of the evil dictator Nemesis.
Battle against werewolves and dracula look-alikes on a Hammer Horror set in the comical Horrorday.
Take the part of a prawn with a hangover, lost in a strange land in the hilarious Prawn.
Solve a bank-robbery by fighting the bad guys and collecting the loot in real-time old West.
Battle countless dwarves in the atmospheric Lost Kingdom of Zkul.
Return to Eden is a massive adventure over 3 disks with colourful graphics - control 3 characters in their quest to find the missing Prince.
All six adventures are available together for only £25.

D-Day MKII v3.04 £10
Grey Wolf v1.8 £8
War In The East MKII v1.24 (Upgrade Only) £5

For the gaming enthusiast - D-Day is a classic table top wargame for one or two players - you control either the Allies or the Axis forces during WWII. With the ability to define your own army set ups and a choice of 4 different scenarios, this should keep you entertained for a while.
Grey Wolf is a graphical simulation of a submarine - can you sink the enemy shipping whilst avoiding their planes and destroyers??

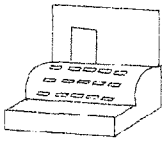
Open Golf v5.20 £8
QuizMaster II v2.07 £5
Stone Raider II v2.00 £5
Hoverzone v1.2 £5
Deathstrike v1.5 £5
Flightdeck v1.05 £10

A range of games to keep both the young and the young at heart amused. Some are old favourites, like Golf and a pub quiz program (500+ questions). Others are fast, colourful arcade games. Flight simulator also now available. Plenty of variation and skill required - what more can you ask for?
All 6 programs only £28.

Britain.map v1.11 £2
BIG Britain Map (needs 2MB) v2.02 £5
Various Area Maps of Britain (ask for details) £2 ea
Ireland Map v1.00 £5
Belgium Map v1.01 £2
Catalonia Map v1.02 £2

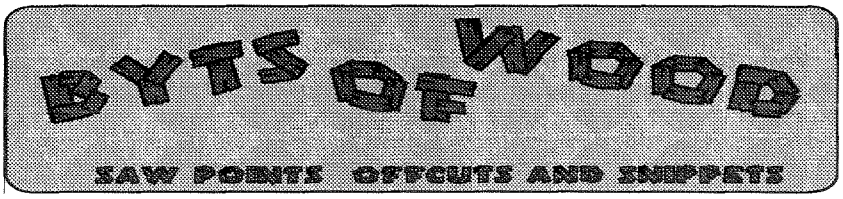
The latest maps for Q-Route. Maps of various areas of Britain have been created by cutting them out of Big Britain Map - they will use less memory and can contain more detail. Areas covered: Scotland, NE England, NW England, S&W Yorkshire, Wales & Derbyshire, London area and South England.
Latest version of Q-Route is recommended.

RWAP Software, 26 Oak Road,
Shelfield, Walsall, West Midlands
WS4 1RQ
TEL: 01922 691607



Cheques in £sterling payable to 'R.Mellor'

* Also known as Trading Accounts



It is a strange thing this serendipity. First Jochen and I were discussing the difference between emulators and native hardware platforms and deciding to instigate a reader survey to find out what the general user thinks and then the ql-users internet newsgroup catches fire with Marcel Kilgus and Peter Graf each defending his corner for their particular product. To be fair to Marcel he was not making great claims as to the authenticity of QPC 2 as a QL platform but both Peter Graf and Pheobus Dokus (a greek QL user now resident in the US) were saying that QL software should be run on 'native hardware' and that QPC2 was not a platform because you cannot run other O/S on it.

By this criteria the QL itself would not be a 'platform' although you could run PC Conqueror and the Spectrum emulator on it so maybe it was. But then you can do the same on QPC 2 so....

All of this was beside the point and a mere semantic argument. For me, and for many other users I suspect, the whole debate hinges upon how useful it is to them and not which particular piece of silicon is doing the number crunching. In the last column I mentioned the issue of benchmarks and how little faith I had in them and, low and behold, there in the same issue an article appeared with the comparative benchmarks for the different systems. Serendipity again. Unless Dilwyn is ill, or too busy to do the proof reading of the magazine's contents, I never get to see the other articles until I get the copies for distribution. So I never know what is in them while I write these articles.

This Platform Boots

Now I really enjoy using my Q 40 and it does all of the hard work for Q Branch these days. The only exception is the updating of my Q Branch Office invoicing program because, for some reason, yet to be ascertained, the Qliberator Compiler will not work with it. I used to think this was a problem which was confined to the Q 40 and had something to do with the hardware the program was running on but QPC 2 exhibits some of the same problems so I suspect it is part and parcel of the new version of SMSQ/E and I hope it can be resolved.

I also use a 700MHz Athlon slot A PC running Windoze 98SE to do the website work, provide internet access and perform a number of other tasks. According to the published benchmarks this should be far slower than the Q 40 but it is not so. QPC 2 running on this machine is appreciably faster than the Q 40 if they are both already booted up. The true proof of the speed and efficiency of a machine is the way that programs perform when running on it and, out of all of the QDOS/SMSQ programs we have none take as much power as ProWesS. I used to do a comparison between my Q 40 and Aurora machines at shows by redrawing a screen containing my exceptionally complex mixing desk in LINEdesign. If I do the same thing between QPC 2 and the Q 40, QPC 2 wins.

Benchmarks are those left by people who sit around too long.

Unplanned Obsolecence

During the many trips to QL shows around England Steve Hall and I often discussed the problems of the problems of designing new hardware for the QL. Steve's approach was that we should use standard commercial PC plug in boards to extend a new QL based motherboard. His argument for this was that the various PC hardware manufacturers could buy chips in the thousands rather than in tens and could mass produce stuff in other countries at rates which would make them very cheap. If you make enough of these boards you can afford to throw some of them in the bin if they don't fire up - something Tony and I appreciate given the problems with the Q 40 boards. The very cost of the components made it essential that Tony burned the midnight oil trying to trace the faults instead of just throwing the failures away.

I had a different take on this because I believed - and still do to an extent - that mass manufacturers of anything tend to lean towards the 'it worked last time so lets tweak it and do it again' concept whereas QL hardware designers have always gone for a 'lets try to do something new that could not be done before' idea. The Q 40, the RomDisq, The QXL and the Super Gold Card were all examples of this because they broke the boundaries and achieved a lot with a small number of components.

One thing they also have in common is a reliance on the continued supply of those

components and the Super Gold Card and QXL have already fallen foul of the changing face of electronics. The Q 40 also has some problems in this area, not as drastic yet but it does need some attention. 72 pin EDO SIMM modules have defied the gravity exhibited by the other RAM chips and become gradually more expensive and harder to find over the last year. The other problem is the ISA slot vital for its I/O. This is now seen as a 'legacy' device by the PC manufacturers and no longer supported. As soon as it becomes an 'unsupported device' the manufacturers stop making them and they will become unavailable.

To The Future and Beyond

The above comments have great relevance to the upcoming designs for the Q 60 if it is to succeed. It is vital that the Q 60 is developed from rather than based on the Q 40. Any use of ISA slots and 72 pin memory would make this a 'dead' board in a very short time. True there are probably enough second hand I/O cards floating about to service a few Q 40/60s for a while but the use of second hand components only adds to the possibilities of a system failure and to the amount of time spent sourcing components. The ISA bus is an attractive prospect because it does not need so many of fiddly interrupt requests that the PCI bus suffers from. In return, its output is slugged. The use of 72 pin memory would probably be the worst factor. As I said above the old SIMMs (Single Inline Memory Module) get more expensive each month because they are no longer manufactured. The higher memory ranges become very hard to find as second hand components and, like its predecessor, the 30 pin SIMM, all that will remain will be the low end chips. People tend to hoard the higher memory ranges.

If the Q 60 is developed from the Q 40 I hope that the parallel port will become a bi-directional one and that there will be a facility for sound input instead of just stereo sound output. Both things are lacking in the Q 40. I have not yet seen a Q 60 so I have no idea if these things have been implemented. There are other factors which will also cause more problems for the Q60 if is is built on the same board design as the Q40 .These are the configuration of the board mountings and keyboard output, and power supply.

The Q40 is based on a PC AT layout and can be installed in a standard PC AT case with the minimum of effort. The AT case has now been superceded by the ATX format and these cases have a completely different layout for the keyboard and mouse ports. I have not tried an AT board in an ATX case to see what would fit where so I cannot be sure of this but, at the very least, you will need to enlarge the keyboard hole to get it to fit the AT keyboard connector on the Q 40. I suspect that the slots will line up with the expansion slots on the case but will the mounting holes do the same?

The Power Supply is another potential problem because the ATX standard is to use the motherboard to switch the power on and off. If the power sockets are not connected to the motherboard then there is no power switch and therefore no way apply the signal to power the rest of the system and nothing will work.

It is fairly simple to find out which pins on the power connector do the switching but it does take away the simplicity of putting the board in a case and switching it on. There are quite a few AT cases around at the moment but, again, they are no longer being made and will, soon, become rare.

These things are all issues that the hardware designer has to deal with and need some careful consideration.

And the Fire Still Burns

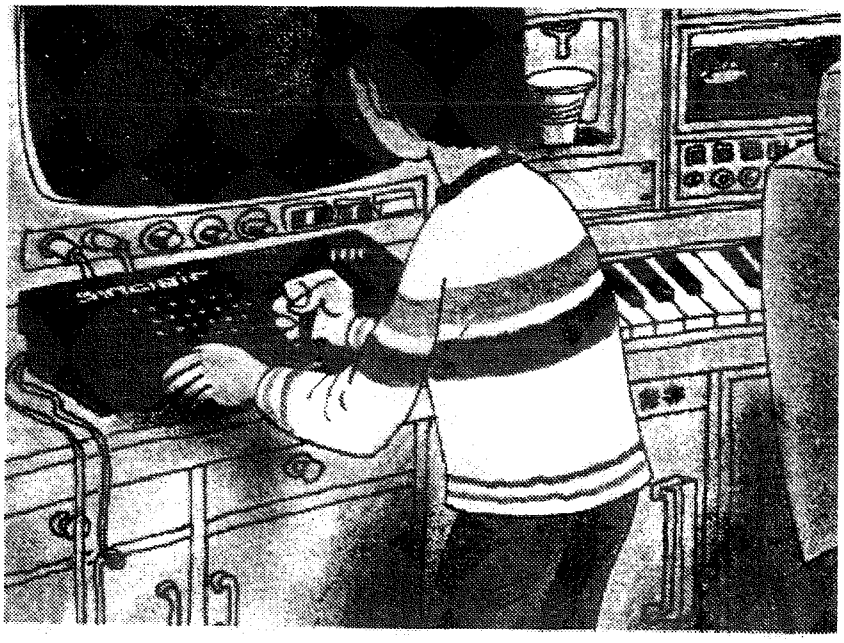
I prodded Nasta into making a comment on the Goldfire project during this discussion and he also had some interesting things to say about hardware design. The first thing to emphasise here is that the hardware is just the starting point. Projects like the GoldFire, the Q 40 and most of the other QL 'mould breakers' of the past would be just pieces of silicon and copper without the code which is needed for the logic chips and the eventual O/S implementation. For most hardware designers this is either an added burden or something they have very little aptitude for.

He also sketches out some of the ground I covered above about chips being obsolete before he even has the design pinned down. He mentions the perrenial problem of users demanding features for new projects which are, frankly, unsupportable given the kind of software support we currently have. One thing we must realise and take into account here is that the system we use has charms which are intrinsically QL. The system, by its very simplicity, is very portable

and very flexible. When I got my Q 40 I was able to copy my Aurora backup files to the hard disk and start a system with very few modifications. On my PC, on the other hand, a fairly simple upgrade took a whole weekend and left me with no internet connection for three days. We should all be wary of asking too much from people who already have given us a lot. You can be sure that the people crying the loudest about the lack of support for scanners and other such stuff are the last ones to write any code for themselves to do these things.

Time For a Challenge

All of the above is very relevant to the future of the QL community because, if we don't go forward, we stagnate but I am also very much in agreement with Wolfgang Uhlig's comments in the last issue. Where is the point of a super fast high capacity machine if the programs stay the same as those we have today. Are you going to buy a Q 40 or a Q 60 to run Quill ?
Peter Graf commented that the Q 40 had inspired the development of new software but most of that software, and here I again agree with Wolfgang, is fragmentary. Yes we do have a high colour operating system as a direct result of the Q 40 but what else do we have from this? There are a couple of good graphics viewers such as Dave Westbury's 'Photon' and an improved ProWesS but no great leap for all of the capabilities of the Q 40.
This is no criticism of the Q 40 itself maybe it could be seen as a criticism of the narrow view that some software writers seem to have but even that is not it really. Programs



written for the PC are often the products of teams of programmers each working one one aspect of the finished product. This is often the reason that the end result is a hodge podge of conflicting code. We have a few good and inventive programmers working on their own in the QL field and many of these give their time, and the results of their labours, for free but I also agree that we need to get closer to the more professional looking and better performing programs that can be found on the PC if we are to survive. QPC 2v2 with its new colour drivers will be more of a drive towards this because more people can afford to get it so there will be more colour driver users and more demand for enhanced software. At present there are no programs which stretch my 32Mb Q40 and it easily copes with as many programs as I can throw at it. I would like that to change. I want to see it being pushed to its limits by programmers so the next generation of hardware will be justified.
So there's a challenge for you all. Get writing.

Of Icons and Men

Anyone who seriously doubts that 'Uncle Clive' as some would dub the QL's creator is an icon for our times would be interested in two things which struck me in the last two months. The first was a snippet in one of the trade papers that we have at work. It mentioned a new printer that is being offered for users of the Palm Pilot and the Compaq iPAQ. This tiny device uses thermal paper on a roll. Does that remind you of anything? The article finishes with a mention of a Sinclair website at **www.sinclairmuseum.co.uk** references to the QL on this site are pretty sparse so lets start bombarding the guy with emails eh guys?
Just the next week I was reading a story to my little daughter and, low and behold there in the main picture was a black box computer clearly marked 'Sinclair'. Certainly in the UK he is the man who is most linked with the development of home computing and Gates are the things you have at the end of the garden.

Boot and Mouse Disease

Another Hove show (well this one was in Portslade but close enough) has come and gone. This time I opted for the local Town Hall and everyone who attended was very happy with the venue and the show. One thing I learned this time was the pitfalls of not using a hotel for the venue. In previous years I had only to give the hotel's phone number and the visitors from a far had somewhere to stay. This time I forgot all about that until a few weeks before the show and had to do a lot of phoning and running around to find places for people to stay. I am amazed that I did not think of using this venue before since it is only moments from my house but I had not even been inside it until I went there to give blood early last year. Even then I did not think of using it as a venue until I found out that it was actually cheaper than hiring a hotel for the day.

I think I will use the same venue for next year's show but I will find a friendly hotel a long way in advance. I was very surprised at the number of hotels in Hove who were fully booked. Maybe the Brighton 'Dirty Weekend' is not dead after all.

The number of people who were able to attend the show was a bit disappointing although there were a number of factors which made it difficult. Many people felt constrained by the need not to spread the foot and mouth disease any further and stayed at home. The weather forecast was bad (although the weather itself was not) and our train service (sic), CONNEX, decided to close services in part of the area. All this and a bomb in London made it a sparse show. Nonetheless I enjoyed it and I will do it again next year.

Right at the end I spoke with Geoff Wicks and asked how he was doing. I knew that he had been trying new jobs since he returned to the UK from Holland but I was a bit taken aback when he told me, 'I'm training as a prophet these days'

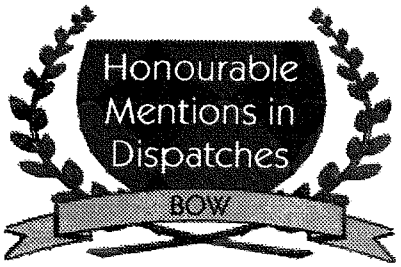
A mental image of Geoff with a long white beard holding up immaculately spelled and style checked stone tablets and expounding on the deaths of kings and second comings sprang to mind until I corrected my mis-hearing of,

'I'm trading at a profit these days'

For a QL trader which is more unlikely I ask myself.

Marcel keeps it Zipped

The close proximity to my house proved a boon for Marcel Kilgus who managed to delete the source files for QPC 2 from his laptop at the start of the show. Good lad that he is he keeps backups on a Zip disk but had not brought a Zip drive with him. I was able to nip home in a break in the show and get him my portable so he could re-install the files. I was also able to collect a new floppy drive for one of User Group to replace one which had died in his machine so it was definitely a useful venue.



Honourable Mentions In Dispatches

A departure this issue because I want to give the honourable mention to my wife, Saskia, who did the catering for the show. She prepared sandwiches and provided cakes, tea and coffee to all who came. Helped by a couple of friends, Judy and Irina not to mention my irrepressible six year old daughter who went around to all and sundry saying 'I think you need some tea/coffee/food/sweets' and generally drumming up trade. This was their first QL show and they were all impressed by how nice QLers were. The following night we received a number of complimentary emails about the catering from those who attended so I think she deserves this issue's accolade.

If this issue is the last issue of your subscription, benefit from our renewal offer!

The QL Show Agenda

Sunday, 29th of April - UK / Portishead

Quanta Workshop & AGM
Somerset Hall, Portishead, Somerset.
Expect all major QL dealers and other interesting people from the QL scene to be there! Of course, every QLer is invited - whether being a Quanta member or not!

Sat., 2nd of June - CANADA / Montreal

North American Show

Quanta and NESQLUG are pleased to sponsor The North American QL 2001 Show. The show site is Loyola High School, Montreal, Canada on Saturday, June 2nd from 9:00 AM to 5:00 PM. At 10:00 AM, the ladies will meet with Dorothy Boehm to plan touring.

So far, four European vendors have indicated plans to attend: J-M-S, Q-Branch, T F Services, and Q-Celt. Marcel Kilgus (author of QPC) will come as well.

Loyola High School is located at 7272 Sherbrooke West at the intersection with West Broadway. It is across the street from Concordia University and is about 15 minutes drive south from Dorval Montreal International Airport.

Arrangements have been made to lodge at Concordia University for $15 ($10 US) per person per night. These are nice student rooms with singles and doubles available plus the use of on-floor kitchen facilities. Call Jeff Peter 514 848-4756 after March 1st to reserve a room. Mention that you are attending the computer show at Loyola High School to get this great rate.

Bus 105 runs past Loyola and connects with the Montreal Metro (Subway) which can take you to a great many of the numerous Montreal tourist attractions. It's only a 6 km drive on Sherbrooke to get downtown but parking is expensive.

Do you plan to attend? Admission is free but please contact Al Boehm so your name tag will be ready. Plus Al will send you additional information and directions. Mail to: Al Boehm, 2501 Ermine Dr., Huntsville, AL 35810; or telephone: 256 859-8051; or email **albertboehm@juno.com**

If you need a ride from the airport or are willing to carry someone, contact Bill Cable Tel: 603 675-2218, email cable·cyberportal.net. The local show host is Francois Lanciault, email: **lancif@videotron.ca**

Saturday, 23rd of June - NL / Eindhoven

Same venue as always-St. Joris College.
Show starts at 10am and usually ends between 4pm and 5pm.
Admission is free, of course! Everybody is welcome!