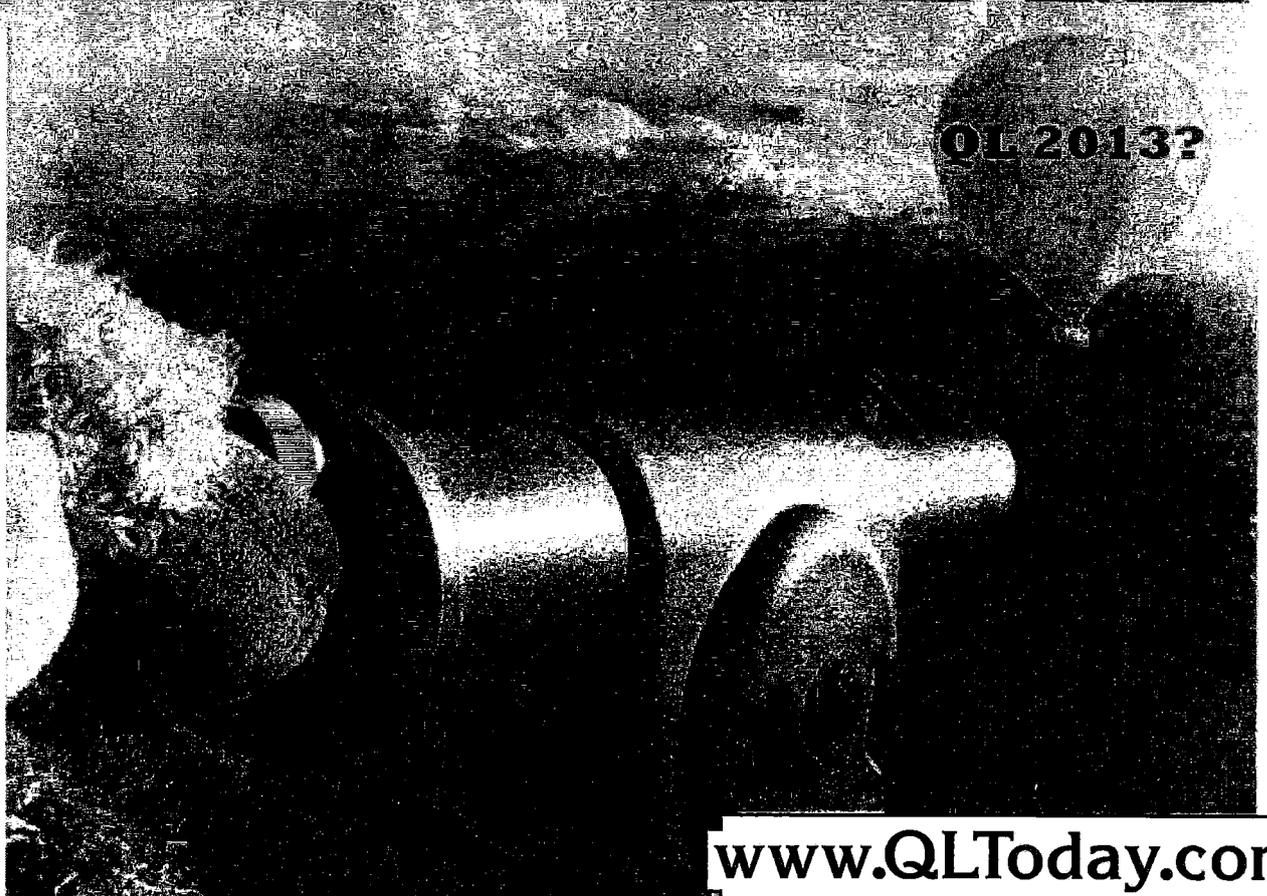


QL Today

Volume 17
Issue 1
Sept. - Nov.
2012

ISSN 1432-5454

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



www.QLToday.com

Contents

- 3 Editorial
- 4 News
- 7 Quite large Integers - Part 3
George Gwilt
- 14 A Serial Nightmare: The Story of the Ser-USB Drivers - Part 3
Adrian Ives of Memory Lane Computing
- 20 More Assembler Discussions
George Gwilt & Norman Dunbar
- 24 Glossary of Abbreviations and Terms Part 5 - J to L
Dilwyn Jones and Lee Privett
- 28 Assembler with a 68020+ *George Gwilt*
- 31 Prottes, Austria - 2012
"To Pig or not to Pig" *Tony Firshman*
- 34 I2C Interface for QL Emulators
Ian Burkinshaw
- 42 Programming in Assembler, Part 31
LibGen - Library Generator - Part 2
Norman Dunbar
- 51 18th Year of QL Today *Jochen Merz*

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software Tel. +49 203 502011
 Kaiser-Wilhelm-Str. 302 Fax +49 203 502012
 47169 Duisburg email: smsq@j-m-s.com
 Germany email: QLToday@j-m-s.com

Editor:

Geoff Wicks Tel. +44 1332 271366
 Flat 5b email: gtwicks@btinternet.com
 Wordsworth Avenue email: QLToday@j-m-s.com
 Derby DE24 9HQ
 United Kingdom

Co-Editor & UK Office:

Bruce Nicholls Tel +44 20 71930539
 38 Derham Gardens Fax +44 870 0568755
 Upminster email: qltoday@q-v-d.demon.co.uk
 Essex RM14 3HA email: QLToday@j-m-s.com
 United Kingdom

QL Today is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

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. 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.

QL Today reserves the right to publish or not publish any material submitted. Under no circumstances will **QL Today** be held liable for any direct, indirect or consequential damage or loss arising out of the use and/or inability to use any of the material published in **QL Today**. The opinions expressed herein are those of the authors and are not necessarily those of the publisher.

This magazine and all material within is Copyright 2012 Jochen Merz Software unless otherwise stated. Written permission is required from the publisher before the reproduction and distribution of any/all material published herein. All copyrights and trademarks are hereby acknowledged.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at <http://www.dilwyn.uk6.net/arch/index.html>

Advertisers

in alphabetical order

- Jochen Merz Software (J-M-S) 23
- QLForum 27
- Quanta 37
- QuoVadis Design 17

**The deadline for the next issue
is the 14th of November 2012!**

Editorial

by Geoff Wicks

Recently there has been a major Sinclair event in the UK about which you have read nothing in either QL Today or the Quanta Magazine. The news came in too late for our last issue and by the time you are reading this, the event will have taken place.

This year it is the 30th Anniversary of the Spectrum and a celebratory show was held, the details of which could make QL-ers jealous. It was planned for two days in four rooms of a university campus next door to the original Sinclair building in Cambridge. The snag was that tickets cost £18 for one day – which puts the Quanta subscription into perspective – or £23 for two days. Nevertheless, the show website warned:

"We believe the demand is greater than the supply, so book your ticket early".

Many years ago a group of us attended a general Sinclair show in Norwich. The event was well attended, but we were hopelessly outnumbered by the Speccies and had our little QL redoubt at one end of one of the two halls. What surprised us was how little had changed in the Spectrum world. It was like stepping into the past with stall after stall selling the games that had made the Spectrum famous. In contrast we QL-ers had continued to develop both the hardware and the operating system to transform the original black box into a serious computer.

You can see some of this in our news section. Our lead story is of Memory Lane Computing leaving the QL scene. Unlike the Speccies our numbers are too few to support hardware developers. It is not a simple matter of finance as many think. There is also a huge demand on the time of a developer and much frustration for little reward as you can read in Adrian Ives' account of the SER-USB drivers. Our second news story is of several innovative programs that have been released over the summer. These programs could not have been written without the continued development our operating system.

Our own 30th Anniversary is in 2014, but so far only two people, Dilwyn Jones and Urs König, have suggested celebrating it. If we are to celebrate we need to start planning now, but we cannot have the lavish celebration of the Speccies. In the UK Quanta is no longer the rich organisation it once was. Its QLis21 celebration cost £3,167 and its QLis25 celebration £2,714 both financed out of the reserves. In 8 years these have fallen from £16,239 to £8,191. Can Quanta justify taking a further £2,000 from the reserves to finance a show that under 40 people will attend?

The challenge for Quanta would be to organise a prestige event for about £1,000. It is possible because we ran QL2004 in Eindhoven as a no frills show for less. Sin(QL)_Air paid the costs. (One of the frills of QLis21 was spending £576 of members' money on AA signposting. It worked out at about £25 per car, and was the reason I resigned suddenly from the Quanta committee.)

The great strength of QL2004 was that for one day we gathered almost all of the QL developers together. Few of us who attended will forget the atmosphere at the unofficial after-show dinner. It gave a boost to the QL community at a time when enthusiasm was beginning to flag. Just a thought, but could we recreate that again, somewhere on the continent in 2014?

QL-SD Development Halted

Adrian Ives has ceased work on the QL-SD project, a mass storage device that would fit into either a microdrive or the ROM slot. In an announcement at the beginning of June he wrote:

"I am sorry to announce that I have taken the decision to withdraw from developing QL hardware. In the current economic climate, it is no longer practical to devote resources to such a small (almost non-existent) market, and I need to concentrate my energies elsewhere.

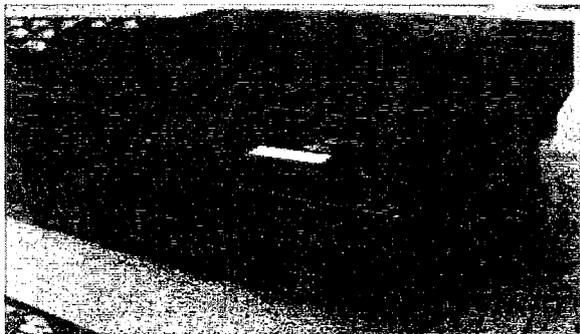
Unfortunately this means that I have withdrawn from the QL-SD project. I understand that Peter (Graf) is talking with other parties who may be more able to bring the QL-SD to market. I wish them all the best and will make available the source code for the already written drivers to allow this to happen as painlessly as possible.

Plans for Q-BUS have also been shelved.

Thank you to the few people who have expressed a genuine interest in new QL hardware."

The project was originally conceived by Peter Graf who had developed it to an advanced stage until time pressures prevented further development. He passed further development over to Memory Lane Computing who continued work on the project.

In reply to questions about the future of his projects Adrian said he would be publishing the schematics and code of Q-Bus online, but only when he had the time to do so. As far as the QL-SD was concerned he had forwarded all his work onto Peter Graf for a decision on how to proceed further. Memory Lane Computing could not offer further support for the drivers as they were moving onto other projects.



In a discussion on the QL-users email group comparisons were drawn between QL hardware projects and those of the Spectrum and the ZX81. Several people pointed out that sales of the QL predecessors had far outstripped sales of the QL.

For comparison Rich Mellor said he had sold over 540 QL keyboard membranes in approximately 4 years and had had 1227 customers. He had sold over 1050 ZX81 keyboard membranes to 722 customers. He was of the opinion that, given the right product, there still was potential for new QL hardware. If only 1 in 20 of his customers purchased a QL-SD that would still be 50 units.

Following the discussion Adrian Ives amplified his reasons for leaving the QL market:

"It's a bit of an oversimplification to lay the reasoning for my decision to withdraw solely on the lack of interest. It has more to do with simple economics. It costs money to buy the stock to build the units. As I have said before about the Ser-USB, unless the stock can be bought in bulk, it is not possible to obtain worthwhile discounts. This makes the product more expensive and thus reduces the likely number of sales. Add to that a severely depressed economy and the continued mismanagement of the Euro crisis, which further depresses any market from continental Europe, and you have a pretty dire situation. Almost a perfect storm, in fact.

I wish I could afford to do this as a hobby, but I can't. I considered a number of ways of moving it forward, including seeking funding from Quanta (banks in the UK don't lend to small businesses any more, so that route is closed and, anyway, what kind of a business case is it to say "Two or three people have said that they will buy one and then, when other people know they are available, a lot more people will buy them"?).

In the end, and to be absolutely blunt about it, it simply wasn't worth the effort required for the small return.

But the root cause of this is that there are significantly less QLs in circulation than ZX81s or Spectrums. It always was a niche machine and, even in today's more retro-friendly environment, it is a minor player. This is a great shame but it is a true and unavoidable fact and it will always influence decisions about resourcing new projects for it."

Summer Software

Some innovative QL programs have been released over the summer months.

QJEWELS

This is a GD2 game with SSS written by Tobias Fröschle.

Dilwyn Jones writes:

"Be prepared to waste a lot of time on this one! QJewels is a free new game for GD2 systems (e.g. QPC2), based on the popular Jewels genre. QJewels is a colourful and (if you have the Sampled Sound System installed on your system)

noisy game. Arrange three or more jewels in a horizontal or vertical line to remove them and earn points - to move a jewel just drag and drop it one square away to form a line of three or more of those jewels.

Standard or compact screen display. "Hint"

mode. Automatic "no more moves" display. Pointer driven program.

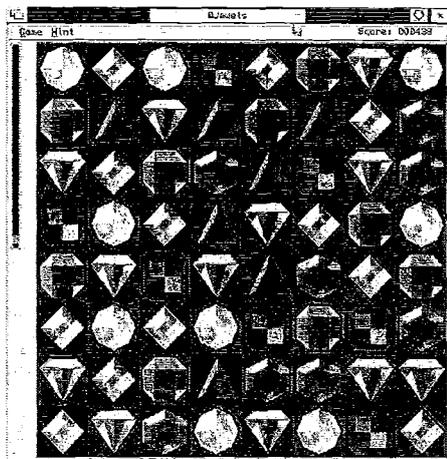
To be able to run this game you'll need a GD2 hi-res display system and SMSQ/E. Sampled Sound System optional - get sound accompaniment if you have that system. Most basic requirement - lots of time! If you're anything like me you'll waste a lot of time on this game, written for QL systems by Tobias Fröschle. It's great to see authors writing software to use the modern QL systems."

The game (173Kb) can be downloaded from:
<http://www.dilwyn.me.uk/games/index.html>

ANALOGUE CLOCK

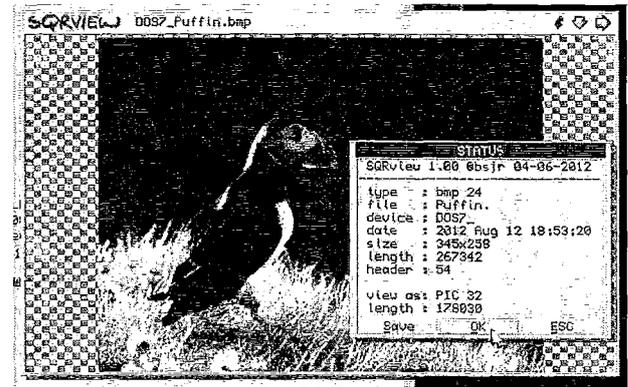
Also from Tobias Fröschle is an Analogue Clock program, that runs on GD2 systems and features a choice of clock faces.

It can be downloaded (60Kb) from:
<http://www.dilwyn.me.uk/utills/index.html>



SQRview

New from Bob Spelten is a screen viewer, SQRview, that can display BMP, PIC, PSA, SCR and SPR modes. The program only runs on High Colour systems. Displayed images can be saved in _pic and sometimes _spr formats.



SuQcess QDOS VERSION

Bob has also now released a QDOS version of the spreadsheet program SuQcess. It is not as powerful as the SMSQ/E version.

Both programs appear on a rewritten website that also contains on its utilities page a new tool for chaining sprites and an update to the Qwatch clock program.

<http://members.upc.nl/b.spelten/ql/>

SETW

George Gwilt has updated his SETW program to version 7.09

<http://gwiltprogs.info/>

He writes that the new version improves the appearance of _asm output on the lines of suggestions made by Norman Dunbar in the last issue of QL Today. Also SETW allows a user to present lists of text items, sprites, blobs and patterns by preset files rather than typing them in while SETW is running. This feature, which did not work on some previous versions, is now again operational.

QSTRIPPER

Norman Dunbar has set up a dedicated website for Qstripper and has published a number of upgrades. Most of the conversion codes for accented characters in the PC version have now been added and he has continued to increase the number of platforms on which the program will run, including the Raspberry Pi. An export option to Open Office format has also been added.

<http://qstripper.sourceforge.net>

The program has received recognition outside the QL World with its inclusion in Softpedia's

database. The site informed Norman Dunbar: "It is featured with a description text, screenshots, download links and technical details." <http://www.softpedia.com/get/Office-tools/Other-Office-Tools/QStripper.shtml>



QL-Today Index

Brian Kemmett has updated his index of QL Today to include volume 16. The complete index of volumes 1 to 16 can be downloaded as a PDF file from Dilwyn Jones' website: <http://www.dilwyn.me.uk/gen/qltoday/qltoday.html> Once again QL Today is grateful for Brian's index, which is extensively used by the editor.

QL Today Index for Volume 1 to 16



CATEGORY	TITLE	AUTHOR	VOL	ISSUE	PAGE
Corrections	ANAGRAMS Error		10	4	7
	Beginners Basic - Correction to Part 2	Stuart Honywood	1	5	7
	Bug found in Pfootmap on LineDesign Demo	PROBS	1	1	14
	Colour Drivers for QL should read DXL	Dilwyn Jones	5	1	6
	Corrected info for Jerome Grimbert's website		6	2	50
	Correction v8 15 re-TURBO	George Gault	9	1	38
	Corrections for v8 13 - Error in "CONVERT" listing	Dave Bunbury	5	5	31
	Corrections for v8 14 - Error in "Calendar" listing	Dilwyn Jones	6	5	31
	Daniel Baum's correct website address		5	2	6
	Directory Deletion in QFAC2	Dave Bonham	7	1	32
	Holborn View 88s Telephone Number correction		6	2	50
RENBORDS	The missing bit	George Gault	9	6	22
LINE DESIGN	Demo Disk fix update	Peter Pital	1	5	46
	Missing Scrn Dumps from v6 15 (TurboPTR & Qmenu)		7	1	59
	MODE32AMP Error		10	4	7

JUST WORDS! first Year Results

In its first year the new Just Words! website has had 4,159 visitors, although many of these were

not QL-ers. The most popular page on the site was QL news followed by freeware downloads:

- 1: QL News (433 hits)
- 2: Freeware Downloads (385 hits)
- 3: Advice and Help (278 hits)
- 4: Maps (264 hits)
- 5: Dictionaries (227 hits)
- 6: QL maps (66 hits - launched half way through the year)

The top five Advice and Help articles were:

- 1: GD2 Colour Tutorial (93 hits)
- 2: User Friendly Programming (90 hits)
- 3: Transferring Spreadsheets (78 hits)
- 4: Transferring LineDesign pages to a PC (64 hits)
- 5: Compiling Dictionaries (59 hits)

Items 2 and 5 were among 4 articles republished on an eBook download site.

The most popular freeware download by far was Roger Godley's GD2 version of Xchange:

- 1: GD2 Xchange (39 downloads)
- 2: Postcodes (19 downloads)
- Solvit Plus 2 (19 downloads)
- 4: Style-Check (15 downloads)
- 5: Spelling Crib (12 downloads)

Where operating systems were known 81% of visitors used Windows, 10% MacOS and 9% Linux. Mozilla (Firefox) was the most popular browser (62%) followed by Internet Explorer (18%).

www.gwicks.net/justwords.html

RASPBERRY PI

As we close the main news pages there are reports that a QL emulator for the Raspberry PI is imminent. Tobias Fröschle has successfully run uqlx on the machine and several people are alpha testing the program. Tobias does not wish to have a full release until he is satisfied it is bug free.



Quite large Integers - Part 3

by George Gwilt

In previous articles I showed how an assembler language program could perform arithmetic on large integers. I show here how an S*BASIC program could make use of this with the CALL procedure.

Before any arithmetic can be done the assembler routines have to be loaded and the space for the numbers has to be set up.

Loading the Assembler routines

In QL Today, Volume 11 Issue 3 I described the use of a program called Set_Hex which produces a function used to load assembler code. The function is called Load_Hex and it returns the address where the code is loaded. In the sample program listed below, the routine Set_Up loads the code and sets the address in a variable called asad (for ASsembly ADdress).

Setting the Numbers Space

The procedure Init, which immediately follows the call to Set_Up, uses the assembler code to set up space for a set of numbers of the required size.

The Arithmetic

Once the first two actions have been taken numbers can be entered into the system and manipulated. This is done by a set of procedures and functions.

The first procedure for entering numbers is Push (number, a%). this will set 'number' in space a%. Thus

```
Push 456,3
```

will put the number 456 in the fourth space.

Since Push cannot enter integers greater than $2^{31}-1$ (2147483647) the procedure Adj has been provided. To use this, the number must be sliced into sections nine digits long, starting at the least significant end.

The most significant portion, which, of course, may be less than nine digits long, must now be entered using Push. The second section is now entered by Push into a second space. Adj is now used on the two spaces. The next nine digits are now entered and the operation Adj is repeated. When there are no more sections to be entered the complete number is set.

For example, to enter

```
14223496784072601046391
```

it is first split into the numbers

```
14223, 496784072, 601046391
```

To set the complete number in position 0 we use the commands:

```
Push 14223,0  
push 496784072,1  
Adj 0,1,0  
Push 601046391,1  
Adj 0,1,0
```

If you want to enter a negative number it is necessary to enter it positively and then use Neg to make it negative.

The full list of procedures and functions is given below. Three of these have been made possible by an addition to the CALLED assembler routine described previously. These are the procedure Sqt and the functions Bits and Numad.

PROCEDURES

Add a%, b%, c%	- adds a% to b% and puts the answer to c%
Adj a%, b%, c%	- sets $a\% \cdot 10^9 + b\%$ to c%
Clra%	- sets a% to zero
Divd a%, b%, c%	- sets a% DIV b% to c%
Dupla a%, b%	- duplicates a% in b%
Init size, set	- produces space for "set" numbers of "size" long words
Modu a%, b%, c%	- sets a% MOD b% to c%
Multa a%, b%, c%	- sets a% * b% to c%
Nega a%	- negates a%
Power a%, n%, b%	- puts $a\%^{n\%}$ to b%
Pushk, a%	- sets the number k to a%
Sqta a%, b%	- puts $\text{INT}(\text{SQRT}(a\%))$ to b%
Subta a%, b%, c%	- subtracts b% from a% and puts the answer to c%

FUNCTIONS

Bits a%, n	- returns for a% the position of bit -n or the number of bits (n=0)
Comp a%, b%	- returns 0 if a% = b% else 1
Count a%	- returns -1, 0 or 1 if a%-1 is negative zero or positive It also sets a% to a%-1
Countba a%, b%	- Same as Count but subtracts b% instead of 1
FAdd, FAdj, FDivd, FModu, FMult, FNeg, FPower and FSqt:	- all perform the same actions as Add, Adj etc but return 1 if the action was successful and 0 otherwise
Numa a%	- returns the address of a%
Range%	- returns the number of integers which can be stored
Size%	- returns the number of longwords holding an integer
Test a%	- returns -1, 0 or +1 for <0, 0 or >0

The code itself is given here.

```
1000 Set_Up
1002 DEFINE PROCEDURE Init(length,number)
1004 REMark Sets up "number" numbers of size "length" long words
1006 IF asad=0:Prerror 1:RETurn
1008 CALL asad,0,length,number
1010 END DEFINE
1012 :
1014 DEFINE PROCEDURE Add(a%,b%,c%)
1016 REMark Adds a% to b% and puts the answer in c%
1018 Do_It 1
1020 END DEFINE
1022 :
1024 DEFINE PROCEDURE Subt(a%,b%,c%)
1026 REMark a% - b% to c%
1028 Do_It 2
1030 END DEFINE
1032 :
1034 DEFINE PROCEDURE Mult(a%,b%,c%)
1036 REMark a%*b% to c%
1038 Do_It 3
1040 END DEFINE
1042 :
1044 DEFINE PROCEDURE Divd(a%,b%,c%)
1046 REMark a%/b% to c%
```

```

1048 Do_It 4
1050 END DEFine
1052 :
1054 DEFine PROCedure Push(number,a%)
1056 REMark put "number" to a%
1058 IF asad=0:Prerror 1:RETurn
1060 CALL asad,5,a%,number
1062 END DEFine
1064 :
1066 DEFine PROCedure Dupl(a%,b%)
1068 REMark copy a% to b%
1070 IF asad=0:Prerror 1:RETurn
1072 CALL asad,6,a%,b%
1074 END DEFine
1076 :
1078 DEFine PROCedure Clr(a%)
1080 REMark a% is set to zero
1082 IF asad=0:Prerror 1:RETurn
1084 CALL asad,7,a%
1086 END DEFine
1088 :
1090 DEFine PROCedure Neg(a%)
1092 REMark a% is negated
1094 IF asad=0:Prerror 1:RETurn
1096 CALL asad,8,a%
1098 END DEFine
1100 :
1102 DEFine FuNction Test(a%)
1104 REMark a% is tested
1106 IF asad=0:Prerror 1:RETurn 0
1108 CALL asad,9,a%
1110 RETurn PEEK_W(asad+2)
1112 END DEFine
1114 :
1116 DEFine FuNction Decimal$(a%)
1118 REMark returns a% as a string of decimal digits
1120 IF asad=0:Prerror 1:RETurn 0
1122 CALL asad,10,a%
1124 RETurn PEEK$(PEEK_L(asad+4),PEEK_W(asad+2))
1126 END DEFine
1128 :
1130 DEFine FuNction Comp(a%,b%)
1132 REMark returns 1 if a%=b% and 0 otherwise
1134 IF asad=0:Prerror 1:RETurn 0
1136 CALL asad,11,a%,b%
1138 RETurn (PEEK_W(asad+2)=0)
1140 END DEFine
1142 :
1144 DEFine PROCedure Adj(a%,b%,c%)
1146 REMark a%*10^9 + b% to c%
1148 Do_It 12
1150 END DEFine
1152 :
1154 DEFine PROCedure Modu(a%,b%,c%)
1156 REMark sets a% MOD b% to c%
1158 Do_It 13
1160 END DEFine
1162 :
1164 DEFine PROCedure Power(a%,b%,c%)
1166 REMark a%^b% to c%

```

```

1168 Do_It 14
1170 END DEFine
1172 :
1174 DEFine FuNction Size%
1176 IF asad=0:Prerror 1:RETurn 0
1178 CALL asad,15
1180 RETurn PEEK_W(asad+2)
1182 END DEFine
1184 :
1186 DEFine FuNction Range%
1188 IF asad=0:Prerror 1:RETurn 0
1190 CALL asad,16
1192 RETurn PEEK_W(asad+2)
1194 END DEFine
1196 :
1198 DEFine FuNction Count(a%)
1200 REMark Subtracts 1 from a% and returns 0 if a% =0
1202 IF asad=0:Prerror 1:RETurn 0
1204 CALL asad,17,a%
1206 RETurn PEEK_W(asad+2)
1208 END DEFine
1210 :
1212 DEFine FuNction FAdd(a%,b%,c%)
1214 RETurn Do_Itf(1)
1216 END DEFine
1218 :
1220 DEFine FuNction FSubt(a%,b%,c%)
1222 RETurn Do_Itf(2)
1224 END DEFine
1226 :
1228 DEFine FuNction FMult(a%,b%,c%)
1230 RETurn Do_Itf(3)
1232 END DEFine
1234 :
1236 DEFine FuNction FDivd(a%,b%,c%)
1238 RETurn Do_Itf(4)
1240 END DEFine
1242 :
1244 DEFine FuNction FNeg(a%)
1246 Neg a%
1248 IF asad=0:RETurn 0:ELSE RETurn (PEEK_W(asad+2)=0)
1250 END DEFine
1252 :
1254 DEFine FuNction FAdj(a%,b%,c%)
1256 RETurn Do_Itf(12)
1258 END DEFine
1260 :
1262 DEFine FuNction FModu(a%,b%,c%)
1264 RETurn Do_Itf(13)
1266 END DEFine
1268 :
1270 DEFine FuNction FPower(a%,b%,c%)
1272 RETurn Do_Itf(14)
1274 END DEFine
1276 :
1278 DEFine PROCedure Finish
1280 IF asad=0:RETurn
1282 Init 0,0 : REMark to return space
1284 RECHP asad:asad=0
1286 END DEFine

```

```

1288 :
1290 DEFine FuNction Countb(a%,b%)
1292 REMark Subtracts b% from a% and sets TEST result in asad+2
1294 IF asad=0:Prerror 1:RETurn 0
1296 CALL asad,18,a%,b%
1298 RETurn PEEK_W(asad+2)
1300 END DEFine
1302 :
1304 DEFine PROCedure Do_It(o%)
1306 IF asad=0:Prerror 1:RETurn
1308 CALL asad,o%,c%,a%,b%
1310 END DEFine
1312 :
1314 DEFine FuNction Do_Itf(o%)
1316 IF asad=0:Prerror 1:RETurn 0
1318 CALL asad,o%,c%,a%,b%
1320 RETurn (PEEK_W(asad+2)=0)
1322 END DEFine
1324 :
1730 DEFine PROCedure Sqt(a%,b%)
1740 IF asad=0:Prerror 1:RETurn
1750 CALL asad,21,b%,a%
1760 END DEFine
1770 :
1780 DEFine FuNction FSqt(a%,b%)
1790 IF asad=0:Prerror 1:RETurn 0
1800 CALL asad,21,b%,a%
1810 RETurn PEEK_W(asad+2)
1820 END DEFine
1830 :
2000 DEFine FuNction Load_Hex
2010 REMark This returns the address of an
2020 REMark ALCHPd area containing the HEX
2030 REMark DATA at line 2160
2040 REMark If a mistake occurs -1 is returned
2050 LOCAL m,asad,adr,top,x,k,wd%
2060 RESTORE 2160:READ top
2070 IF top<=0:RETurn -1
2080 asad=ALCHP(top)
2090 IF asad<0:RETurn -1
2100 k=INT(top/2):adr=asad
2110 m=top-2*k:top=asad+top
2120 FOR x=1 TO k:READ wd%:IF adr+2>top:RECHP asad:RETurn -1:ELSE :
      POKE_W adr,wd%:adr=adr+2
2130 IF m:READ wd%:IF adr+1>top:RECHP asad:RETurn -1:ELSE :POKE adr,wd%
2140 RETurn asad
2150 END DEFine
2160 DATA 2398

```

*** Here follow the DATA lines which contain the assembled code ***

```

3000 DEFine PROCedure Set_Up
3010 asad=Load_Hex
3020 END DEFine
3030 :
3040 DEFine PROCedure Prerror(n%)
3050 IF n%=1:PRINT "NO GOOD"
3060 END DEFine
3070 :

```

Example of Use

The code shown above is not a complete program. If you run it all it does is to load the assembler code and name its address "asad". You can now type commands on the keyboard and experiment with the various arithmetic operations.

As an example of the use of this code I give below the instructions which perform the calculation of PI. The key procedure is Set_Pi. If you call this the result will, eventually, be a listing of PI to the number of places you have requested. Shortish lengths will give a virtually instantaneous result. But if you ask for 20,000 places you will have to be prepared to wait. On my Apple MAC with Windows XP running under VMware this takes three hours which is shorter than on any other of my machines.

Set_Pi asks for the number of places of PI and calls Piple to get the answer.

Piple opens a window filling the whole screen, indicates the number of places of PI requested, calls CalcPI to do the work and finally prints the answer together with a note of the time taken.

My BOOT sets a procedure in the polled list which adds 1 to the long word at \$DC of the system variables every 1/50th of a second. This was described in QL Today Volume 12 Issues 3 and 4 and this what I use for timing events.

CalcPI does the calculation. It first assesses the number of long words needed to hold the integers involved in the calculations and uses Init to set this. It then sets the scaling factor z. This is a power of 10 two more than the number of places requested (in the hope that at least the places of PI up to two before the end will be accurate). It then calculates PI by Machin's formula:

$$4*PI = 4*atan(1/5) - atan(1/239)$$

The calculations of atan are performed by Atn. Atn continues to add terms while they are greater than 1/z. Since this can take some time I have arranged that every few seconds a noise will be made. This is intended to indicate that something is happening rather than the computer lying idle nursing a crashed program.

Here, then, is the code

```
5000 DEFine PROCedure Set_Pi
5010  LOCal lp
5020  OPEN#4,con:WINDOW#4,300,40,20,SCR_YLIM=80:PAPER#4,WHITE#:INK#4,BLACK#:
      BORDER#4,2,2:sz%=3
5030  REPEAT lp
5040    CLS#4:PRINT#4,"Give the size (Zero to EXIT): ";
5050    sz%=EDIT#(4,sz%,6):IF sz%=0:EXIT lp: ELSE :Piple sz%
5060  END REPEAT lp
5070  END DEFine
5080  :
5090  DEFine PROCedure Piple(n%)
5100  LOCal ch%,p$(10)
5110  IF ch%=0
5120  ch%=3:OPEN#ch%,scr:WINDOW#ch%,SCR_XLIM,SCR_YLIM,0,0:INK#ch%,WHITE%
5130  END IF
5140  USE ch%
5150  tp=PEEK_L($280DC):CLS:PRINT "PI to "&n%&" decimal places":CalcPI n%:
      tm=PEEK_L($280DC)-tp
5160  DIM p$(n%+2):p$=Decimal$(10):Pr_P(p$):PRINT:Pr_T:Noise1
5170  USE
5180  SUSPEND_TASK 300
5190  END DEFine
5200  :
5210  DEFine PROCedure CalcPI(m%)
5220  REMark to get pi to m% places in [10]
5230  LOCal n%
5240  IF m% < 3 OR m% > 32000:Do_Er 5240
5250  n%=2+INT((m%+2)/9.63296)
5260  Init n%,11:REMark space for 11 integers each n% long words
5270  Push 10,3:REMark put 10 in [3]
```

```

5280 IF NOT FPower(3,m%+2,3):Do_Er 5280:REMark Set  $10^{(m\%+2)}=z$  in [3]
5290 Atn 5,9:Atn 239,10
5300 Push 4,7
5310 IF NOT FMult(7,9,9):Do_Er 5310
5320 IF NOT FSubt(9,10,10):Do_Er 5320
5330 IF NOT FMult(7,10,10):Do_Er 5330
5340 END DEFine
5350 :
5360 DEFine PROCedure Atn(o%,k%)
5370 LOCAL lp,s,t_base,t_end,bp%
5380 REMark puts atan(1/o%)*z to [k%]
5390 s=0:Clr k%:REMark clear answer space
5400 Push 1,0:REMark set 1st of 2r+1
5410 Push 2,1: REMark [1] = 2
5420 Push o%,2:REMark [2] = 1st of  $o\%^{(2r+1)}$ 
5430 Mult 2,2,5:REMark [5] =  $o\%^2$ 
5440 t_base=PEEK_L($280DC):bp%=1
5450 REPEAT lp
5460 IF NOT FMult(0,2,6):Do_Er 5460:REMark  $(2r+1)*o\%^{(2r+1)}$ 
5470 IF NOT FSubt(6,3,4):Do_Er 5470:REMark  $(2r+1)*o\%^{(2r+1)} - z$ 
5480 t_bend=PEEK_L($280DC)
5490 IF t_bend-t_base>150:Noise:t_base=t_bend
5500 IF Test(4) = 1:EXIT lp:REMark finished
5510 IF NOT FDivd(3,6,6):Do_Er 5510:REMark  $z/[(2r+1)*o\%^{(2r+1)}]$ 
5520 IF s:Neg 6
5530 s=1^^s
5540 IF NOT FAdd(6,k%,k%):Do_Er 5540
5550 Add 0,1,0:REMark 2r+1
5560 IF NOT FMult(2,5,2):Do_Er 5560:REMark  $o\%((2r+1)$ 
5570 END REPEAT lp
5580 END DEFine
5590 :
5600 DEFine PROCedure Do_Er(p%)
5610 USE:CLS:PRINT "Not good pi @ "&p%:STOP
5620 END DEFine
5630 :
5640 DEFine PROCedure Noise
5650 BEEP 9000,167,6,23,bp%
5660 bp%=bp%+1:IF bp%>6:bp%=1
5670 END DEFine
5680 :
5690 DEFine PROCedure Noise1
5700 BEEP 31000,52,97,2,381
5710 END DEFine
5720 :
5730 DEFine PROCedure Pr_P(p$)
5740 PRINT:PRINT p$(1)&". "&p$(2 TO LEN(p$)-2)
5750 END DEFine
5760 :
5770 DEFine PROCedure Pr_T
5780 LOCAL sec,min%,hr%
5790 hr%=INT(tm/180000)
5800 min%=INT(tm/3000)-60*hr%
5810 sec=tm/50-3600*hr%-60*min%
5820 PRINT "Time "&FDEC$(hr%,3,0)&": "&G1$(FDEC$(min%,2,0))&": "&G1$(FDEC$(sec,5,2))
5830 END DEFine
5840 :
5850 DEFine FuNction G1$(d$)
5860 IF d$(1)=" ":RETurn "0"&d$(2 TO)
5870 RETurn d$
5880 END DEFine

```

Finally I should say that the code shown assumes that it is running under SMSQ/E and that Turbo TK code is loaded.

A Serial Nightmare: The Story of the Ser-USB Drivers - Part 3

28-APR-2011 Project Resumed

by Adrian Ives of Memory Lane Computing

Between the 21st and 27th of April I received a number of e-mails from potential customers interested in buying a Ser-USB should they become available. This seems to have been the result of a "letter writing" campaign organised by Tony Firshman. I am grateful that Tony did this, because there is no doubt that without those e-mails the Ser-USB project would have been abandoned.

29-APR-2011 v0.92.014 Release Candidate 7

By now I had replaced QDOS Slave Blocks with "Private Slave Blocks", an area of heap allocated by the driver at startup and managed in exactly the same way as the QDOS slave table. This broke the link with the QDOS slaving system, saved memory and won the Battle of the Slave Blocks.

The tide, it seemed, was turning.

05-MAY-2011 v0.92.023 Release Candidate 12

There followed a flurry of "Release Candidates" until number twelve, at which point I was planning to release 1.0 on the following Monday; May 9th, 2011.

This release removed the bulk of the debug-code and I considered to be as stable as it could be, given the hardware and OS limitations.

07-MAY-2011 Beta testing extended by one week

By now I had received the PCBs for the production prototypes, but there was a problem: the space for the USBWiz module did not line up correctly, making it impossible to mount the boards in the case that I had chosen. To give myself some time to come up with a fix, I extended the beta testing, thus putting back the release date for 1.0.

08-MAY-2011 Production Prototype 001

After redesigning the physical layout and changing the case, Ser-USB Mark-I, serial number 001, was completed. It was to be the first of only three such units to be built. Unit 001 has special sentimental value because it was actually built using the USBWiz module taken from the original (now defunct) prototype.

Unit 001 is still in use at Memory Lane Computing today, where it is regularly used for testing driver updates and for moving software between machines.

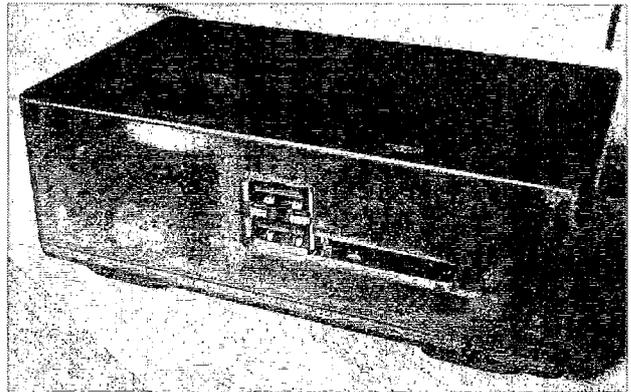


Illustration 4: Ser-USB 001

The Mark-II was intended to be the real production model. It would be in a smaller case and have its sockets mounted directly on the PCB. Instead of a 9 Pin D connector, the Mark-II would use an 8 pin mini-DIN socket for the serial connection to save space.

13-MAY-2011 The first Ser-USB is shipped

The first commercial sale of a Ser-USB is completed as Mark-I unit 002 leaves the workshop, accompanied by a floppy disk containing the first release of the 1.0 driver (which would not be publicly announced until the following Monday).

16-MAY-2011 v1.0 Release

To be precise, this was actually version 1.00.024. The last three digits of the version string represent the EDDE build number. It included the Queue Manager, the Driver Command Manager, Partition Manager and USBWiz Terminal. It had been nothing short of a marathon getting to this point, but finally, it was done.

With 1.0 out of the door, work immediately turned to production of the Mark II units and the inevitable bug fixes that would be needed as new problems were discovered. There was also the little matter of a ROM driver to be dealt with.

21-MAY-2011 ROM Driver enters Alpha Testing

Creating the ROM driver introduced a new challenge: how to fit 18376 bytes of driver code into a 16384 byte EPROM! The answer was not to. Instead, some components from the driver, mostly the S*BASIC extensions were removed. These would be loaded as a separate LRESPR file.

However, this posed another problem. To use the driver on standard QL hardware required the Queue Manager. You couldn't load the Queue Manager from the Ser-USB because you needed the Queue Manager running before you could access the drive.

The answer was to introduce a bootstrap loader facility that allowed a code overlay to be loaded from the Ser-USB before the device driver itself was active. This was the first time that bootstrapping (as opposed to BOOTing) had been implemented for a QDOS device driver. At the time I was very pleased with the solution, but it did introduce a lot of support headaches, especially when issuing software updates. The bootstrap data had to be stored in a reserved area on the SD Card that could only be written by a special utility. This made upgrades difficult because it required the user to perform this operation, and if they made a mistake it could leave them without a functioning system.

23-MAY-2011 v1.02 Release

- Fixed a bug that caused UMount n followed by MOUNT n to report "Bad or Changed Medium".
- The default number of Private Slave Blocks was increased to 64.
- The Debugging Service Interface was brought in line with the other installable module interfaces (in other words a complete replacement could be installed).
- The new QM_GO command in the installable Queue Manager was a shortcut way of doing a QM_START followed by DRIVER_ASYNC_IO 3.
- The USBWiz Terminal could now run if the Ser-USB driver was loaded.
- The ROM version of the driver was released.

27-MAY-2011 Production of Ser-USB Mark II units begin

With the arrival of the redesigned circuit boards, it was now possible to start producing the Ser-USB as originally designed.

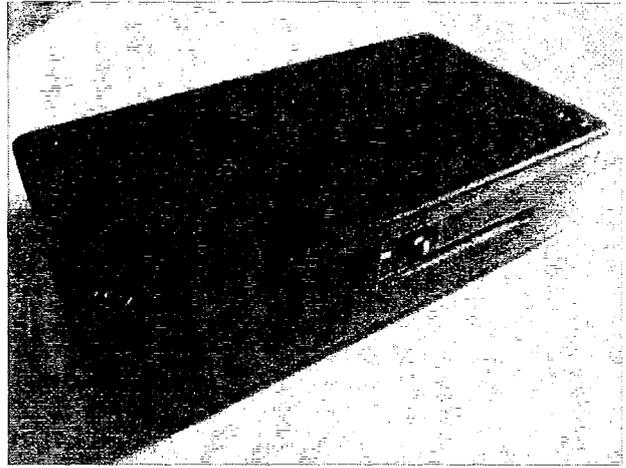


Illustration 5: The Ser-USB Mark II

02-JUN-2011 v1.03 Release

- DRIVER_FLUSH command was moved from the core driver to the extensions.
- DRIVE_CAPACITY command was moved from the extensions into the core driver.
- The ROM driver now also supported using non-superHermes serial ports greater than 2 under SMSQ.
- Fixed the improper display of large block counts as negative numbers in the Partition Manager.
- Fixed a memory corruption issue in the Partition Manager.
- The EDDE ECF Updater utility (needed to update the bootstrap code) now had a Config Block for setting the default location of the files.

07-SEP-2011 v1.04 Release

- Implemented limited integration with the new Ser-USB FAT Driver that allowed both drivers to be loaded at the same time.
- USBWizTerm detected either Native or FAT Ser-USB drivers and worked correctly with either (or both) loaded.
- Trap #2/3 I/O Servicers were now run able to run entirely with private stack space on versions of the OS that supported it (only Minerva).
- The broken MOUNT command was fixed.
- Fixed bug in ROM driver that tried to change the baud rate back to 9600 after setting it to 4800 on standard QLs.
- The ROM driver now did an automatic QM_GO if it was started with a configuration to use a standard QL.
- The equivalent of a QM_GO could be

invoked from machine code with a special long function code argument to `qm_do_async_op`.

- A new `QM_STOP` command in the installable Queue Manager switched to synchronous I/O and forced all Queue Manager services to shut down.
- Equivalent of a `QM_STOP` could be invoked from machine code with a special long function code argument to `qm_do_async_op`.
- New functions in the Driver S*BASIC Extensions were added to read drive and partition information:

`DRIVE_MAP(d)`

Return the address of the drive's map

`DRIVE_NAME$(d)`

Return the name of the drive

`DRIVE_PTCOUNT(d)`

Return count of partitions on drive

`DRIVE_PTSTART(d,n)`

Return start LBA of partition on drive

`DRIVE_PTNAME$(d,n)`

Return name of partition on drive

- The default number of Private Slave Blocks were increased from 64 to 128 in the light of real-world user experience with the driver.
- The Partition Manager no longer required the Driver S*BASIC Extensions or Toolkit II to be loaded.

Version 1.04 was the last of the 1.x series. It was a major release in the sense that it introduced the FAT Driver and a whole new way of using the Ser-USB. Instead of loading the native QDOS driver, users could use the FAT Driver instead. This implemented a suite of S*BASIC procedures and functions for managing files on FAT16 and FAT32 volumes, with all of the file system handling taking place on the Ser-USB. It was a far better solution for resource-constrained standard QLs and is still the way we normally use Ser-USBs here at Memory Lane.

JAN-2012 Ser-USB 2.0

It would be three months before I returned to the Ser-USB driver to make sweeping changes to its driver architecture, abolish the Queue Manager and create a driver that, finally, I was happy with.

I had been collaborating with Peter on the QL-SD project and, as a result, the EDDE core

became EDDE 2, introducing one very fundamental change that would definitely benefit the Ser-USB: Write As You Go (WAYG) map handling. Instead of waiting a set time after the last write to a drive and then flushing the entire map to the disk (as the QUBIDE driver did, and so EDDE had inherited this behaviour), WAYG writes individual map sectors as they become dirty, massively reducing the overhead and removing the need to repeatedly write the entire map each time, most of which never changes. This was also essential to reduce wear and tear on flash memory devices. Not having to send the entire map across the serial port five seconds after the last write was also going to be a huge performance benefit for Ser-USB.

I should thank Peter for WAYG, though. When the very first EDDE driver ran with the QL-SD, it was his comment that the map flushing was like "Taking a Coffee Break" that made me tackle the issue! WAYG was written in just two days and it turned out to be a surprisingly simple algorithm. It's been in EDDE 2 ever since.

Returning to the Ser-USB driver, the first thing to go was the Queue Manager. As a solution I have never liked it. It was messy, complex, and unreliable. It would not be missed. What I chose to do instead was to retain the Save State Engine and allow trap #3 calls down to the serial driver to return all the way to the top calling level if they were incomplete, suspending the driver's state so that it could be re-entered on the next scheduler loop.

This didn't take as much work as I had anticipated as all trap #3 serial I/O calls were already routed through a mechanism known as the Watchdog Timer. Unrelated to the d3 timeout value passed to the traps, this is a timer that limits all serial transactions to a set period of time according to the QL's real time clock, thus allowing the driver to recover from a serial I/O operation that never completes.

In practice this means reading the clock on entry to a "watched" trap #3, calling the trap, testing the return code, and retrying if not complete and the configured maximum elapsed time hadn't been reached according to the clock. Internally, a watchdog timeout returns the new error code `errwx`, but application programs never see this as it is converted to `errnc` by the driver if a "real" timeout has occurred.

As all serial I/O was routed through the watchdog routines (`watch_begin` and

QUO VADIS
DESIGN

Independent Information
Technology Services

www.ql-qvd.com

QL/QDOS/SMSQ/E Software

QUO VADIS
DESIGN Independent Information
Technology Services

QL/QDOS/SMSQ/E Software

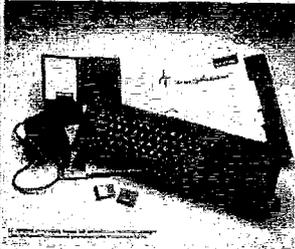
Home Products Support Company Contact

Welcome

Quo Vadis Design sells software for the Sinclair Quantum Leap computer (QL) and variants including a new O/S called SMSQE.

The QL is a computer in its 25th year Anniversary.

The Sinclair QL - a quantum leap in personal computing



Software emulations of the QL now exist which can run on a PC/Mac with Windows/Linux or Mac Operating systems.

News

QVD QL News Blog - keep up to date

[News Blog](#)

24/02/2009

Quo Vadis Design Website Launched

01/02/2009

FEATURED PRODUCT



BUY NOW!

Copyright © 2009 Quo Vadis Design. All Rights Reserved.

[Home](#) | [Products](#) | [Support](#) | [Company](#) | [Contact](#) | [Privacy](#)

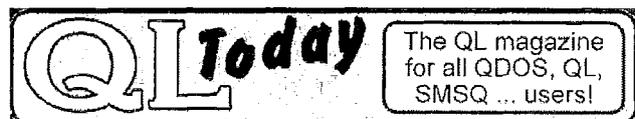
Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

Special Offers available from
Jochen Merz Software for its
25 years in QL Trading

Check the QL News Blog on
our website for the special
offers



Subscriptions taken online

watch_trap3) it was easy to add the code necessary to "put the driver into suspense" by a call to `ss_save_state` in the Save State Engine.

After making these changes I was rewarded by a driver that was finally able to access the Ser-USB on a black box QL without any need for the Queue Manager.

Well, to be more accurate, I had a driver that worked under Minerva 1.98.

When it was tested with JM and JS the whole thing fell over. This led to a further "wobble moment" on January 11th, when I decided that the new driver would have to be limited to Minerva. This was not the first time that something which worked under one version of the ROM would not work under another. There are also differences between SMSQ and QDOS that caused problems in the early days, such as the ability to start a task from the scheduler loop service, which SMSQ and Minerva can do, but JS cannot.

Eventually, though, a workaround was found. The reason why it worked under Minerva and not JS was connected with trap #2 `io.open`.

When Tony Tebby wrote QDOS he implemented the IOSS Retry mechanism (this has been discussed earlier) for all trap #3 calls. It's a powerful feature of QDOS that allows device drivers to be written that would otherwise be far more complex and allows non-blocking I/O to boost performance. Unfortunately, the retry mechanism does not extend to trap #2 and this is a real pain. It means that trap #2 calls must complete atomically; they cannot return "Not Complete" and have the IOSS retry them until all their processing is done. This is not so bad for `io.close`, but `io.open` has to do a lot of work and if it doesn't finish what it is doing, subsequent trap #3 calls will likely fail with disastrous results.

This is no problem at all when the driver is managing the hardware directly, but when it is making calls down to another driver, and that driver cannot complete its operations immediately then you are caught in a deadlock scenario.

That's why the code that handles trap #2 calls into the Ser-USB driver has no option: it has to pass a timeout in `d3` to any trap #3 call it makes to the serial driver, and this means that the scheduler will get re-entered with potentially disastrous consequences. Except that they weren't disastrous under Minerva. Using timeouts worked.

For JM, JS and the multi-lingual MG ROMs, an additional workaround had to be introduced: the Extended Open Handler. This is the most contro-

versial aspect of the 2.0 driver as it is working around QDOS to introduce new behaviour. The principle is not very simple:

- When an `io.open` call is made, calls down to the serial I/O driver are allowed to initiate a deferred driver process that is executed on the frame interrupt.

- A new return address is pushed on the user stack, pointing to the Extended Open Handler; a short piece of code which runs in user mode, retrying the original call at 50 frame intervals until the code returns something other than `err.nc` - at which point all of the processing has been completed.

- On each frame interrupt, the driver restores the saved state established when the deferred driver process was created and continues executing. This process continues on each timer tick, returning `err.nc` until all of the `io.open` code has been executed.

Complex though this arrangement is, it does actually work ... unless an application program tries to make an `io.open` call in supervisor mode.

The Extended Open Handler effectively converts trap #2 `io.open` into a non-atomic call and, in fact, the Ser-USB driver is the only device driver that can return `err.nc` from such a call and not be signalling an error. It's actually possible to set a flag to disable the Extended Open Handler and tell the driver to do exactly this, leaving it up to the application program to handle the retries.

There was one other issue to be solved that was related to trap #2. A call to `io.close` could trigger a similar deferred driver process, the difference being that `io.close` would always return. This means that the situation can arise when the driver is busy still handling the residual processing from a close or delete operation and is thus not ready to perform another transaction. This doesn't matter for trap #3 calls, because the driver just returns `err.nc` and the IOSS will retry until it can handle the request, but a new trap #2 call has to handle this situation. The solution was to invoke the Extended Open Handler for this as well.

Job done. Well, not quite. There's still the issue of supervisor mode calls into the driver that will fail (this is actually true of trap #2 and trap #3 calls) and cannot be fixed due to the need to make calls to the underlying serial driver. And then, there are programs that just don't like the Extended Open Handler. I don't blame them. I am hijacking the user stack pointer, inserting a false return address, and executing a chunk of code that

they don't even know is happening (like trap #1 mt.susjb calls).

For programs written in S*BASIC and compiled with TURBO or Q-Liberator, there are two workarounds for this final issue: the extensions SRU_BUSY and RETRY_OPEN, the latter allowing io.open calls to be made with the Extended Open Handler disabled.

There is, at the time of writing, no fix for supervisor mode calls into the driver.

As the last section has shown, the 2.0 drivers are very different. They represent a huge investment of time and energy and come close ... very close ... to achieving what I had considered impossible. I began to allow myself to think that the nightmare might finally be over.

Numerous other changes were also made in 2.0; I have listed the most significant of them below:

- The Ser-USB native driver was split into two versions: Destiny (for QL emulators running on PCs, Macs and Linux systems) and Legacy (for standard QLs and compatibles).
- The device name was changed from USB to SRU (i.e. USB1_ is now SRU1_).
- The driver core was changed to EDDE 2. This has a more open architecture and is less complex.
- Write As You Go (WAYG) map handling. This removed the irritating delayed-action map flush that brought the system to a halt for long periods of time.
- The Queue Manager was no longer required (or supported).
- Tighter IOSS retry integration meant that QDOS was handling the retry of serial I/O operations and not the driver. This improved overall reliability.
- A new Extended Open Handler emulates IOSS retries on trap #2 io.open calls by hi-jacking the user mode return address.
- The ROM driver did not need to load any overlays.
- The ROM driver allowed the QL to boot from a Ser-USB drive.
- The ROM driver presented an option at the QL startup screen to either automatically start the driver or to manually start it with the SRU_START command.
- Updated S*BASIC procedures and functions.

- Changes were made to the way that FORMAT was implemented. In particular, a new command SRU_FORMAT was introduced for the legacy driver.
- The API system was replaced by the separate EDDE2 Link Layer Driver, making it possible for application programs to call the EDDE API without any knowledge of the underlying device.

I am sure that there are still more bugs to be discovered. The Ser-USB driver is, after all, doing something remarkable. It is abstracting a block device driver across obsolete serial hardware to bridge the 20th and 21st centuries. But QDOS did not admit defeat easily. The serial ports did not submit without a fight. This war may still not have been decisively won ... but the last battle has definitely been fought!

The 2.0 driver has come too late to save the Ser-USB. The future of QL hardware is now with more advanced hardware such as QL-SD or even Q-BUS, but the development of the Ser-USB drivers has, in the final analysis, been a journey that was worth taking.

It's just not one that I would want to take again. Probably.

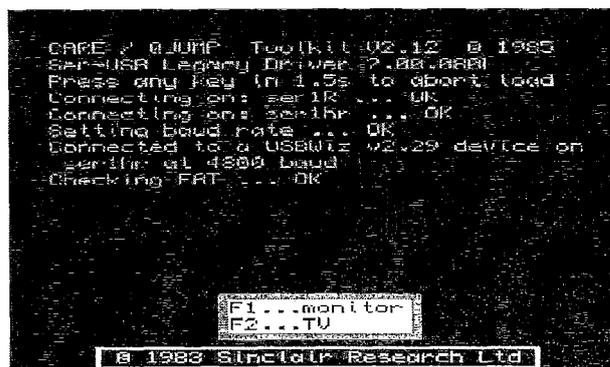


Illustration 6: Ser-USB 2.0 Legacy Driver, ROM version

EPILOGUE

10th February, 2012.

I am reading the report from one of the 2.0 beta testers, Urs König. He had what was going to be the final version, intended for the public release on the 20th of February. I am appalled by what I see. Pages of problems ranging from unreliable WCOPY commands to corrupted partitions to a catastrophic system crash when saving a one line BOOT program. It is nothing short of a disaster, and it is made much worse by the fact that he is reporting things that have either all been fixed or have never been reported before!

I have to make a decision. It is surprisingly easy for me after twelve months of this serial nightmare.

Enough is enough. I end the beta test, freeze Ser-USB development and withdraw the product

from sale. The 2.0 drivers will be made available as a "curiosity"; a glimpse of things that could have been but were not to be.

I hadn't won the war after all. But at least I would win the peace.

More Assembler Discussions

by George Gwilt and Norman Dunbar

Norman's [ND] answers to George's [GG] comments on Assembler - Part 31. Comments inserted by the Editor are marked [ED].

[GG] I must say that I found Norman Dunbar's article Part 31 both useful and interesting.

[ND] Thanks. I worked hard on it, and still managed to spot a few errors when the magazine arrived. Why is it that when I proof read something, I never see any (!) errors, but as soon as it is committed to hard copy, they are blatant? I have detailed the errors at the end of this text.

[GG] Naturally I went through the process he describes to produce a window definition for his Library Generator. Unfortunately when it came time to place the loose items in the window I discovered that I had not entered all the items of text that I should have, so I aborted the program preparatory to a retry. At this point I remembered that SETW allows a user to preset a list of text items, putting them in a file called ram1_text_list. But when I tried this I found that the current version of SETW would not accept such a list. As a result I have produced version 7.09 of SETW correcting this fault and was able to produce the window suggested by Norman.

[ND] I was aware that this could be done, but as I had never had more than a few text items, I never bothered. It seems that I'd have been stuck had I done so! Thanks for the fix though.

[GG] I was surprised, however, at two of his amendments to the resulting _asm file. He set the select key for ESC and MOVE to 3 and 5. Though that is quite correct, I wondered why he did not press these select keys when invited to do so at an earlier stage. Thus pressing ESC would automatically result in 3 being set for ESC. Also pressing ALT/F4, which is the key for MOVE, would result in the select key being set as 5. How does he know that 5 is the correct key? I certainly wouldn't!

[ND] I didn't think to do so is the simple answer. Maybe I'm too used to EasyPointer 3 which requires me to enter the event number rather than the keypress. If I press ESC in EasyPointer 3, I get a key code of 27 and not the event number of 3.

Also, the EasyPointer Manual gives the numbers required to be entered to get an event.

And finally, Some of the ALT+Fn keys combinations, on Linux, do different things. For example, If I press ALT+F1 or ALT+F2 or ALT+F3 or ALT+F4, the key press is intercepted by the GUI system (KDE in my case) and causes a desktop switch to take place. I have 4 virtual desktops and this is the key combination to switch between them.

I only have Windows running in an Emulator on my laptop, so I cannot use those keys at all. This made me unable to try the key combination to see if it would enter the event number, even when running windows.

[GG] Later on Norman sets the justification code xjst to -1 for three loose items. Again I would like to know why this was done. Setting -1 causes the item to be printed so that it stops just one pixel before the right hand margin of the item's hit area. Now the width of the hit area for all three loose items is just two pixels larger than the width of the item itself. This means that setting xjst to any of 1, 0 and -1 will result in the item being placed in the middle of the area with one pixel between both margins.

[ND] I hadn't realised that the object was so large to be honest. I wanted the text to be right justified and when running, it appears to be correct.

[GG] My feeling is that this is in fact a veiled criticism of SETW since it always sets both xjst and yjst to zero and does not allow the user to choose a value for himself.

[ND] Not at all. Most of the time I only ever use default left justification, I just wanted to try out right justification on this utility. No criticism intended at all.

[GG] Norman has altered the item "flag" in the main window section wd0. He has split this into two bytes, "flag" and "shad". In general this is laudable. The flag, which deals with whether a window is cleared or not and whether the arrow keys move the pointer or not, can have either or both of its most and least significant bits set. For a shadow size of 2 and with both flag bits set SETW will set flag to a rather uninformative 33026. Had this been expressed in hexadecimal it would have been set as \$8102 which much more clearly shows what is happening. Indeed this was one of the changes I have been making to SETW and I am glad that Norman thinks this a reasonable alternative.

[ND] This is indeed a good change - thanks.

[GG] However, once again, I have to point out that Norman has been using an old faulty version of SETW which erroneously sets the most significant bit of the flag byte to signal that the window must be cleared. If this flag bit is set it signals that the window must not be cleared. So, it is a relief to see that, in the event, Norman has correctly set his new flag byte to zero.

[ND] Yes, in the code I set it to zero, but I left in a comment in the text saying "feel free to leave it at \$80..." Oops!

It is at this point I shall relate the story of a conversation, by email, between George and myself on the matter of the clear window flag. I found that with this utility, it made no difference at all whether it (the flag) was set to Zero, \$80 or \$81 - everything looked exactly the same when the code was executed.

However, after a chat with George and more experiments, I discovered that this flag affects only the window. If you have Information or Application Windows "on top of" the Window, you never see the window being cleared or not.

What I did was to delete the application window from LibGen and try running it with all possible values of the flag. When set to \$8x it did indeed show the current SuperBasic background (a program listing in my case) and when the window was moved around, the section of the program listing was displaying, moved around with it!

As LibGen has the entire window filled with Infor-

mation Windows 9 (at the top) and an Application Window (at the bottom) then the whole of the Window is covered and so, the background doesn't show through regardless of the flag setting.

Mystery solved!

[GG] Finally, Norman has cleaned up the _asm file by inserting blank lines between individual items such as information windows. This seems a good idea and one that I have incorporated in SETW v7.09.

[ND] And I promise to only use this version, from now on, until a new one comes out of course. Hopefully, we won't need to discuss the flag byte again! And I won't have to remember to change it either!

[ND] And finally, the errors in the previous article are as follows:

Page 48, Item 16.5. "QL Toady" slipped in again. It should of course be "QL Today". Thankfully the code is correct.

Page 51. At the end of the last paragraph of text, above the final code section, I have this "You can leave the word set to \$8002 if you wish" This should of course read "\$0002" - again, thankfully, the code is correct (final two lines on the page show the bytes in question).

[ED] The dialogue continued over several emails.

[GG] Very interesting.

I'm amused about your remarks on the setting of the selection key. The reason why the selection key for ESC is 3, is, of course, that ESC is one of the keystrokes that causes an event. The selection key for any such keystroke is the event code.

[ND] This is true, but I'm almost 100% certain that when I pressed ESC in SETW, I got a key code of 27 - which is what I would expect for the ESC key. I was looking for 3 rather than 27. I might get a chance to test a little program to see if pressing ESC (and thus getting 27) does actually generate a CANCEL event anyway. I need to know for my own sanity!

[GG] I'm quite sure that SETW always gives 3 for the key press ESC for a select key.

[ND] Mind you, from what you say below, I have a

suspicion that ESC will cause WMAN to trigger the CANCEL event anyway.

[GG] Yes, WMAN always takes ESC to be the event "cancel". This can either be the select key for a loose or menu item or can be processed by the program as an event.

You say that Easyptr requires you to key "3" if you want to set the selection key for ESC and that if instead you press ESC Easyptr sets the code to 27.

[ND] In EasyMenu, you select a keypress by pressing the K key, then the keypress desired. To set an event you press the C key, and enter the event number. At least, in EasyMenu 3 you do.

[GG] If so I find this very odd. It means that if you do set 27 as the selection key you will never be able to select that loose item by pressing ESC because the PE software will translate ESC to the value 3.

[ND] This is exactly what I need to test, however, the EasyMenu manual allows you to use C then event and doing this causes the event to be handled internally by EasyMenu's own code, rather than the application having to handle these events.

[GG] If a loose item's select key is set to 27 WMAN will never select the item by keypress.

[ND] It's something I always used when writing QLibertated SuperBasic programs because using the internal event handling saved me having to write code to handle sleep, move, cancel etc. I'm a lazy developer!

[GG] Furthermore, you will not be able to get Easyptr to set keypresses 3 to 8, since it will, presumably, set the selection code to one of 3 to 8 instead of 51 to 56.

[ND] EasyMenu does it differently from SETW, you tell it whether you are giving it a key press (K) or key code/event (C) then enter the details.

[GG] You also mention that with some emulations you do not have access to some key combinations. I find this is true of the MAC. I have to press the function key to get access to F1 to F4 when running QPC2. However, I would have thought that if you can't get SETW to set a particular keypress as the selection keystroke that you wouldn't be able to use that keystroke anyway in the completed program.

[ND] This is correct. The CTRL+Fn key combination, for example, cannot be used in SETW/EasyMenu to generate the menu and nor can it be used in the generated application. Linux (in my case) intercepts it long before it gets to the application.

[GG] With my MAC and with one of my son's window's 7 laptops you have to use the Fn key with the Fx keys. Thus, given Fn/CTRL/F4 SETW sets the select key to 5. Also Fn/SHIFT/F3 sets the select key (correctly) to 243.

[ND] Unless I stick a Loose Item on the application and set it to be the event code/key press required - then if I can't use the keypress shortcut key combination, at least I can click it with the mouse!

[GG] I have now come to the conclusion that it is a waste of time to set selection keystrokes to any of the event loose items.

[ND] Agreed!

[GG] As far as I can see it makes no difference to the actual operation of the final program. I may have comments on those lines when I see what your program is!

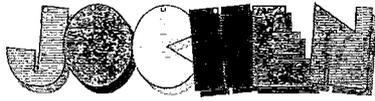
[ND] As I've said before George, your comments are most welcome.

[ED] There was a final test by Norman:

[GG] I'm quite sure that SETW always gives 3 for the key press ESC for a select key.

[ND] I tested this just now, after installing the latest version of SETW - you'll be pleased to hear - and it displays "Event - cancel" when I choose the ESC key as my selection key for a loose item. This is excellent, especially as I was "certain" that I saw it print up a 27 which is why I "had" to edit the source and put in a keycode of 3.

I shall not need this in future, SETW "just does it right".



**Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany**

**Phone +49 203 502013
Fax +49 203 502012
Email: SMSQ@J-M-S.com**

QPC2 Version 3 + SMSQ/E Software QL-Emulator for PC's	EUR 59,90
QPC2 Version 3 - Upgrade from QPC2 Version 2	EUR 19,90
QPC2 Version 3 - Upgrade from QPC2 Version 1	EUR 39,90
QPC Print - printer emulation driver for QPC	EUR 39,90
BUNDLE: QPC2 and QPCPrint	ONLY EUR 79,90
Agenda Agenda program for WMAN and Prowess	[V1.09] EUR 14,90
Success Database front-end for WMAN	[V2.05] EUR 19,90
QD2003 Pointer-Environment-Editor	[VB.01] EUR 29,90
QD2003 Upgrade from Version 9 and older	[VB.01] EUR 14,90
QMAKE Pointer-driven MAKE for GST/Quanta Assembler	[V4.31] EUR 14,90
BASIC Linker	[V1.21] EUR 14,90
WINED Floppy/Harddisk Sector- & File-Editor	[V1.26] EUR 14,90
FiFi II File-Finder - Extremely useful!	[V4.31] EUR 14,90
FiFi II Upgrade from Fifi Version 3 or older	[V4.31] EUR 9,90
EPROM Manager	[V3.02] EUR 14,90
QSpread2003 Spreadsheet Program	[V4.04] EUR 29,90
QSpread2003 Upgrade from Version 3 and older	[V4.04] EUR 14,90
QPAC I Utility programs	[V1.11] EUR 19,90
QPAC II Files, Jobs & other Things	[V1.45] EUR 29,90
QTYP II Spell checker	[V2.17] EUR 19,90
QPTR Pointer Toolkit	[V0.30] EUR 29,90
DISA Interactive Disassembler	[V3.04] EUR 29,90
CueShell	[V2.14] EUR 29,90
CueShell for QPC	[V2.14] EUR 14,90
SER Mouse software mouse driver for serial mice	EUR 10,00
EasyPTR Version 4	[V4] EUR 59,90
EasyPTR Version 4 - Upgrade from earlier versions	[V4] EUR 39,90
QDT - QL Desktop program	EUR 59,90
QMENU Version 8 - with new, printed Manual	[V8.02] EUR 24,90
QMENU Version 8 - Update from earlier Versions, also with printed manual	EUR 17,90
QMENU Version 8 - New/Update for QL Today subscribers, with prt'd manual	ONLY EUR 14,90

Please add EUR 4.90 for postage to all destinations - Germany, Europe, Worldwide!

We accept VISA, MasterCard & Diners Club online and offline!

Details for money transfers:

- Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43
- Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000
- Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000
- The Netherlands: Jochen Merz, Gironummer 3258439, Postbank NL Amsterdam
- and from all other countries in EUR with IBAN and BIC to account
Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360
- UK customers can pay in £ (convert EUR prices above to £ by multiplying with 0.83) to
Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
or send cheques in £ - no fee for UK sterling cheques!
- If you wish to pay via paypal, send money to Paypal@J-M-S.com

Cheques payable to Jochen Merz only!
Price list valid until 30th of Nov. 2012

Glossary of Abbreviations and Terms

Part 5 - J to L

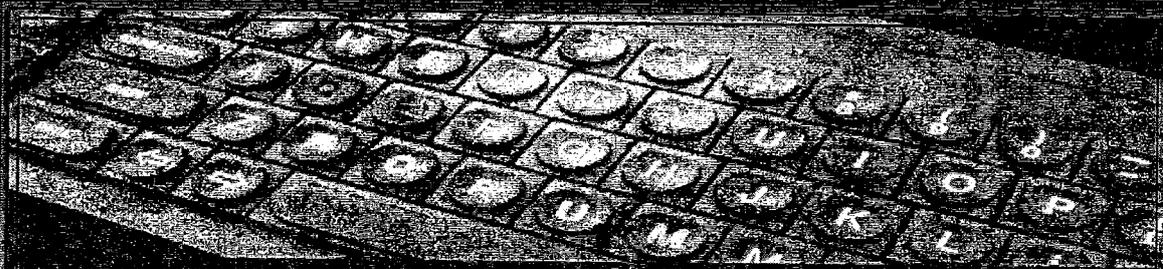
by Dilwyn Jones
and Lee Privett

Again, we continue here from where we ended last issue.

Jumper	A connection on a circuit board that allows different circuits to be linked together by the specific location of a small metal loop. This allows different permutations and configurations to be realised
JPEG	Joint Photographics Expert Group, name of a body to agree on graphics compression standards for still pictures. Used generally to describe a file saved in this format. Not in widespread use on the QL, though there is a QL PD program to convert between JPEG and GIF, and there are several GIF file readers for the QL
KB	Abbreviation for KiloByte, or 1,024 byte. The unit 1,024 is used rather than 1,000 as it is a number which is a power of 2, which makes it easier and more logical to handle in computer terms. 1,024 KB makes 1 MB or 1 MegaByte, see below
Kernel	Special code at the heart of a computer's operating system.
Keyed	Termed used to describe specific orientations of connectors and plugs so that only the correct connection is made
KHz	KiloHertz, a measure of the number of cycles per second
LED	Light Emitting Diode, small fairly low current device used to replace filament bulbs, as their robustness and longevity made them much more reliable and ideal as indicators. Available now in red, orange, yellow, green, blue & white colours
LCD	Liquid Crystal Display, sandwiched between two layers a liquid changes its opacity when an electrical charge is passed though it, used in displays.
Linker	A special program which joins up two or more code files and builds them into a single executable or code file, and works out the "links" between the code so that they are all joined up correctly together.
Linux	Operating system originally created by one Linus Torvalds (a former QL user himself!). Allows us to use a QL emulator called uQLx. There is also a version of the Qlay emulator which can run on Windows. Linux is an example of an open source operating system originally based on Unix.
Logical Operators	Used to determine if a condition being tested is true or false, e.g. IF x=0 AND y=1 THEN PRINT "True" : ELSE PRINT "False" : END IF
LONG WORD	2 Words or 4 bytes of computer memory. Sometimes referred to as a 32 bit value. For those who understand binary numbers, this corresponds to a 32 digit binary number, so a long word of computer memory can store quite large values
Loop	A programming structure which repeats a statement or block of code until a condition for ending the repetition occurs. FOR loops run a set number of times, whereas a REPEAT loop runs until a certain condition occurs then exits from the loop when that condition occurs.
LQ	Letter quality, a term used to describe print quality
LSB	Least Significant Byte, the lowest 8 bits of a numeric value. When you write the number as a binary form, this will be the rightmost 8 bits. Can also stand for Least Significant Bit when specifically referring to one single bit of the data's value

Machine Code	Machine Code is the name of the instructions that a processor can run directly. At its simplest level, machine code is a sequence of numbers in memory. Rather than program directly in machine code, programmers usually write code in what is called Assembly Language, a human readable text form of machine code, which is then converted by a program called an Assembler directly into machine code which the computer can run without having to do any further conversion when the program runs.
Macro	A piece of pre-written code or routine or value which is added to a program where indicated to save having to type it in each time you write a program. You will often come across this term when using assembler programs.
Make	Make reads in a makefile, which is a list which specifies which source code files are needed to build the final program. This sort of approach allows you to write a program in sections, which can later be compiled into a single program. Writing code in sections like this makes it easier to maintain very large programs.
MB	Megabyte, or 1,024 KiloBytes, or 1,024 times 1,024 bytes. Nowadays, computer memory is often so large that it is measured in MB rather than Bytes
Mbps	Mega Bits Per Second, not to be confused with MBps (MegaBytes per second)
MDV	Device name for the microdrive tape loop storage devices on a Sinclair QL. MDV is an abbreviation of microdrive. Two of these tape drives were supplied built into the case of the QL and in theory up to 6 more could be plugged into a slot on the right hand side of the QL. Each microdrive could store about 100 kilobytes of data.
Menu	A list of items on the screen, from which you are invited by the computer to choose one or more of those items
Menu Extension	A handy little toolkit written by Jochen Merz to simplify the writing of programs of your own which can be controlled by a mouse or cursor arrow keys and adds facilities such as allowing you to create menus and lists for file selections, list selections, and so on. The term Menu Extensions refers to the software itself, whilst the term Qmenu refers to the printed programming instruction manual. If, like me, you have difficulty remembering which term refers to what, try to remember the sentence (from the manual) which says: "QMenu - How to program and use The Menu Extension"
MERGE	The act of joining two SuperBASIC programs together with a command called MERGE.
MESS	Multiple Emulator Super System. An emulation engine which can emulate over 250 computer systems, including the QL
MHz	MegaHertz, a measure of the number of cycles per second
Microdrive	The original QL was supplied with two built in tape loop drives, called microdrives. The tape cartridges which plugged into these drives were called Microdrive Cartridges and could each store up to about 100 kilobytes of data. The microdrives were also known as MDVs, since the operating system called the drives MDV1_ and MDV2_. Now largely obsolete.
Minerva	A replacement operating system chip for the QL. The original versions of the QDOS operating system for the QL did have a few problems which were not sorted out before the QL was discontinued. Minerva is produced by TF Services, and fixes these problems and provides a few extra facilities as well. A Minerva ROM has the characters "JSL1" as its version identifier, which were the forename initials of the designers Jonathan (Oakley), Stuart (McKnight) and Laurence (Reeves).
MODEM	MOdulator/DEModulator. A device which plugs between a computer and a telephone line allowing data to be sent over a telephone line.

Monadic Operator	A symbol which can precede a number, such as + - NOT and "" which tells us how to interpret the value of the number, variable or function which follows, e.g. LET num=-(a_value) would ensure that num becomes the negated value of the variable called "a_value"
Mouse	Device for processing hand movements to a pointer system
MP	Multi Processing
MSB	Most Significant Byte, the top 8 bits of a number (the leftmost part when written as a binary string). Can also refer to Most Significant Bit when referring to one particular bit of a data's value
MT	Multi Tasking
Multi Tasking	More than one program running at the same time, a bit like a secretary answering the phone and typing a memo at the same time. Not the same as Task Switching (qv) where more than one program may be in the computer's memory, but only one running at a time, e.g. Quill and Archive in memory, but you type something into Quill and then switch to Archive to type in something else and so on. A good example of multi tasking is when you use Quill to type in a letter, and elsewhere on the screen a little clock is running constantly showing you the current time and date while you are typing.
Multi-Threading	The ability of a processor to run several threads of execution seemingly at the same time. What the processor does is to run one program for a few microseconds, then another for a few microseconds and so on, giving the impression of running at the same time.
Name	Term used to identify a variable, procedure or function. e.g. in the expression LET a=1 the name is "a". The Name Table is a list of these names, including details such as whether the name refers to a numeric variable, string variable, procedure, or function.
Nesting	Term used to describe structures one inside the other. For example, a FOR loop written to run inside another FOR loop.
NET	Device name used for the QL's networking system. The QL network system was also implemented on the QXL and Aurora cards.
NIC	Network Interface Card, allow the computer system to connect, transmit and receive data to and from a network
NLQ	Near Letter Quality, a term used to describe print quality
OEM	Original Equipment Manufacturer
Open Source	Programs supplied with the source code (or the source code is available to everyone). Anyone can study the source code to see how it works and make changes if permitted by the software licence. Generally, the software licence would prevent you being able to sell for profit any development you might make of the package.
OS	Operating System, the program or collection of routines that controls the computer examples include MS-DOS, TOS, AMIGOS and QDOS
OS X	Name given to the current Apple Mac operating system. Its main virtue is that it allows us to run Daniele Terdina's QL emulator, QemuLator for OSX.



The Sinclair QL Forum

A Place for Sinclair QL Users to Meet!

QL FORUM

View [all topics](#)

GENERAL		TOPICS	POSTS	LAST POST
	The Welcome Area Introduce yourself here!	23	163	by igorstellar D Fri Feb 10, 2012 3:00 pm
	Help for New Users A section for new users to ask for help and also for people to post helpful tips.	14	87	by olliraa D Mon Feb 06, 2012 12:59 pm
	General QL Chat A place to discuss general QL issues.	50	336	by ... D Fri Feb 10, 2012 3:31 pm
	Hardware Having hardware related question? Post here!	70	589	by thorsinclair D Fri Feb 10, 2012 12:35 pm
	Software Anything software related.	5	31	by ... D Thu Feb 09, 2012 11:25 pm
	QL Emulation Discussion and advice about emulating the QL on other machines.	22	122	by Brane2 D Sun Feb 05, 2012 10:23 pm
	Compatibles Corner An area to discuss compatible QL or compatible computers as the Thor, Auron, 486 and 650.	2	25	by ... D Wed Jan 11, 2012 11:25 pm
MARKETPLACE		TOPICS	POSTS	LAST POST
	For Sale Sell your QL items here!	19	91	by RWAP D Thu Jan 19, 2012 5:26 pm

The online community for all things ql



www.qlforum.co.uk

Assembler with a 68020+

by George Gwilt

Norman Dunbar's articles on assembler programming all assume that the processor is a simple 68000/8 as in the original QL. But all the modifications to the hardware from the Super Gold Card onward make use of one or other of the Motorola processors which contain the enhanced set of instructions available from the 68020 upwards. This enhanced instruction set is also available in the latest version of QPC2. If it is there, why not use it by means of an appropriate assembler such as GWASS?

What I want to do here is to show how three of these 68020+ instructions can help to solve a common problem. The three I will use are from the set of bit field instructions. Each of these eight instructions operates on a set of contiguous bits – the bit field. The start of the field is determined by the offset, measured in bits, from a specified effective address. The offset can be any number between -2^{31} and $2^{31}-1$ inclusive. The width, or size, of the bit field can be any number from 1 to 32 inclusive. In general these instructions can take a value from, set a value to or simply test, a bit field.

The problem to be solved is the conversion of a set of bytes to an unsigned integer. These bytes can be decimal, octal or hexadecimal. Other conditions are that there may be any number of characters, that the integer must fit into a long word and that the last character must be a separator, or terminator.

Separator

Each character must be tested to see if it is a separator. If it is, the routine is finished. The obvious test for a separator is to compare the character with the separator. But what is a separator? If all the characters are set each one on a new line the separator will simply be LF. However, if the numbers are spread across lines the separators could be SPACEs as well as LF. Or TABs may be allowed as well as SPACE. Then again it may be that the input is part of some arithmetic expression. In this case such characters as asterisk, back slash, brackets, curly brackets, square brackets and so on may be needed. Clearly with such a large number of possible separators the voluminous code needed to test each one explicitly is to be avoided. It was when I was faced with this problem that I turned to the bit field instructions. It occurred to me that what you needed was a simple binary test to be applied to any of the 256 characters which might appear in a byte. All I needed was an array of 256 bits, the content of eight long words. The BFTST instruction would perform the separator test at one blow. If a bit was zero the corresponding character would be a separator but if not, not. Thus if the tenth bit were zero then TAB, having value 9, would be a separator.

From decimal, octal or hexadecimal to number

If the character just read in is not a separator it should be a character forming part of the number. At this stage, therefore, a test must be made to ensure that the character is a proper one. This also can be tested by using BFTST.

Program

Now we come to the routines which convert the input to unsigned integers. All three of the routines follow the same general form. The characters are read successively and processed until a separator is found at which stage the operation is complete. Errors must be signalled if a character is neither a separator nor a proper digit and also if the integer is too large to fit into a long word. Here, then is the code for decimal digits.

```
; dec sets the value of the decimal string (A0) to D1.L
; D0 is set to 0 if OK: -1 if not
```

```
dec_reg    reg        d2
dec        movem.l    dec_reg, -(sp)
           moveq     #0, d0
           moveq     #0, d1
```

```

dec1      move.b      (a0)+,d0      character -> D0.L
          bftst      sep_tab{d0:1} separator? . .
          beq        dec_end      . . yes so finished
          bftst      dec_tab{d0:1} Decimal digit? . .
          bne        dec_err      ----> . . no
          subi.b     #"0",d0      -> 0 - 9
          move.l     d1,d2        ans copied to D2.L
          asl.l      #2,d1        ans * 4
          bcs        dec_err      ----> overflow
          add.l      d2,d1        ans * 5
          bcs        dec_err      ----> overflow
          add.l      d1,d1        ans * 10
          bcs        dec_err      ----> overflow
          add.l      d0,d1        add in new digit
          bcc        dec1        OK so go for next digit
dec_err   moveq      #-1,d0      mark error
dec_end   movem.l   (sp)+,dec_reg
          rts

```

You will see that the second and fourth instructions of the loop starting at dec1 are the BFTST instructions testing for separators and correct digits. This is common to all three routines.

In this routine, the answer so far is multiplied by 10 by a set of instructions and the next digit added. At each instruction in the performance of the multiplication it could be that a binary bit slips out from the top of the register. This is noted and an error signalled. The first possible error occurs when the copy of the answer so far is shifted up two bits. An astute reader will notice that the BCS test will only indicate an error if the second top bit in the answer so far is 1 and that the top bit is ignored. Thus it might appear that if the top nibble of the answer so far is in the range \$8 to \$B inclusive an erroneous integer might slip through the net. This is, in fact, not so since in these cases one or other of the following BCS tests must inevitably detect the overflow.

Now follows the code for hexadecimal digits. Here overflow is more simply detected since all we need do is check that there are no more than eight digits. You will notice that this routine contains the instruction BFEXTU. This extracts bits 5 and 6 from the character just read and places the value in D2. The value is 1 for 0 to 9, 2 for A to F and 3 for a to f. The adjustment of the character to its hexadecimal value is then performed by table lookup from "adj".

```

; hex sets the value of the hex string (A0) to D1.L
; D0 is set to 0 if OK: -1 if not!!

```

```

hex_reg   reg        d2/d3
hex       movem.l    hex_reg,-(sp)
          moveq      #0,d0
          moveq      #0,d1
          moveq      #8,d3      count up to 8 hex digits
hex1      move.b     (a0)+,d0    character to D0.L
          bftst      sep_tab{d0:1} separator? . .
          beq        hex_end    . . yes so finished
          bftst      hex_tab{d0:1} hex character? . .
          bne        hex_err    ----> . . no!
          bfextu     d0{25:2},d2 to see if 0-9, a-f, or A-F
          add.b      adj(d2.w),d0 set D0.L to 0 - 15
          lsl.l      #4,d1      ans * 16
          or.b       d0,d1      add new character
          dbf        d3,hex1     get the next one
hex_err   moveq      #-1,d0      mark error
hex_end   movem.l   (sp)+,hex_reg
          rts

adj       dc.b       0,-'0',10-'A',10-'a'

```

The third routine, which is for octal input, shows three instances of the use of BFTST. As indicated in the remarks following the routine dec the BCC instruction checks only the last bit shifted out from the top of D1.L. In the present case we need to check whether any of the three bits shifted out of D1 are non zero. To do this we replace the shift instruction by ROL which not only performs the required shift but also sets the three top bits in the bottom of the register. These are then tested by BFTST.

```

; oct sets the value of the octal string (A0) to D1.L
; D0 is set to 0 if OK: -1 if not!!

oct      moveq    #0,d0
         moveq    #0,d1
oct1     move.b   (a0)+,d0      character to D0,L
         bftst   sep_tab{d0:1} separator? . .
         beq     oct_end      . . yes so finished
         bftst   oct_tab{d0:1} octal character? . .
         bne     oct_err  ----> . . no!
         subi.b  #'0',d0      D0 -> 0 to 7
         rol.l   #3,d1        ans * 8 and overflow
;                                     to bottom 3 bits of D1
         bftst   d1{29:3}     did overflow occur? . .
         bne     oct_err  ----> . . yes
         or.b    d0,d1        add new character
         bra     oct1         get the next character
oct_err  moveq    #-1,d0       mark error
oct_end  rts

```

A Macro

The above routines require four tables indicating the correct characters in a group. They are for the separators, decimal characters, hexadecimal characters and octal characters. Setting up these tables by hand is tedious and can easily result in errors. The following macro, s_tabm, and the routine set_tab are designed to do the task.

The macro has two parameters. The first is the name to be given to the table and the second is the list of acceptable characters. To indicate the characters TAB and LF, t and l can be used. T and L will do just as well. To clear the bit in the table corresponding to each character the macro calls the subroutine set_stab. This subroutine uses BFCLR which is the third of the bit field instructions.

```

s_tabm  macro    where,what
         bra     w2\@
\1      dcb.l    8,-1      Set all characters to unacceptable
w1\@    dc.b     "\2",0    The zero is appended to indicate "end"
w2\@    lea     \1,a0
         lea     w1\@,a1
         bsr     set_stab
         endm

```

```

; On entry A0 -> where
;           A1 -> what (list ending 0 with t=TAB and l=ENTRY)

```

```

set_stab moveq    #0,d0
lp       move.b   (a1)+,d0      Ended? . .
         bne     alt_D0      . . no
         rts
alt_D0   cmpi.b   #'t',d0
         beq     alt1        -> 9
         cmpi.b   #'T',d0
         beq     alt1
         cmpi.b   #'l',d0
         beq     alt2        -> 10

```

```

        cmpi.b    #"L",d0
        beq      alt2
alt3    bfcldr   (a0){d0:1}      Set to acceptable
        bra     lp
alt1    moveq   #9,d0
        bra     alt3
alt2    moveq   #10,d0
        bra     alt3           Get the next character

```

Here, finally, are the four instructions which can be used to set up the four tables.

```

s_tabm  sep_tab,< t1 ()*+,-./{}[:&<>>!>
s_tabm  dec_tab,< 0123456789>
s_tabm  oct_tab,< 01234567>
s_tabm  hex_tab,< 01234567890aAbBcCdDeEfF>

```

Note the use of the less than and greater than signs, "<" and ">," surrounding the parameters. These characters show that anything between them is to be taken as part of that parameter, including spaces and commas which would otherwise denote the parameter's end. Also, to include ">" as part of the parameter, it is necessary to set it as ">>" inside the parameter to indicate that the ">" is not to be taken as the end marker.

Prottles, Austria - 2012

"To Pig or not to Pig"

by Tony Firshman

Of course I had to go to the QL show at Prottles last June, organised again by Gerhardt Plavec. It is one of the perks of being self-employed - tax-deductible holid ... ummm business trips.

The pre-arranged condition was for Andrea and I to share a massive Pig's leg at Prater - a memorable feast in 2010.

I stayed in the central Holiday Inn along with Jochen, Andrea, Marcel and his guest Sandra. ... so a large twin room all for me (8-(#

Easyjet was mightily expensive to Vienna. Jochen gave me a clue - go to Salzburg and get a train. That was really **not** the answer, as the train journey is five hours. However Bratislava cost £65 via Ryanair, and a bus from the airport to Vienna centre cost £14 return, including free wifi. Easyjet to Vienna costed £230 return, plus train to the centre. No decision. [*Jochen adds: not quite - Salzburg to Vienna is 3 hours, and using a Westbahn Train you get free Wifi, and the rate can be as low as 10 EUR - and you get fresh and excellent coffee, cappuccino or latte macchiato ... not included in the 10 EUR, of course!*]

I parked my motorbike for free (as usual) in short term parking at Stansted, right next to the

terminal. How many people know about this non-advertised service? It applies to all major UK airports.

The only hassle was the small (10kg) cabin bag allowance from Ryanair. That isn't quite true. Everyone knows about the Ryanair tricks. See <http://www.youtube.com/watch?v=ZAg0IUyHHFc> That is all true, except the last one. "If you haven't pre-paid for the steps you can f*****g jump". Not quite though. They do not charge for the 'jacks' (toilets). I am **sure** Michael O'Leary is well familiar with the video, as he announced last October that they would be charging for the toilets. There was a mighty furore, and it even featured on BBC news. They withdrew the proposal, of course, the following day.

The first hurdle on the Ryanair site is the included travel insurance. This was a pre-filled checkbox, which was not alterable. I found out how to cancel the insurance. In the country of origin drop-down further down, there was a country well down the list called 'No thanks' - after Nigeria. 'United Kingdom' of course was right at the top. Secondly there was no mention of free baggage. All they said was "Checked baggage" (what is that?) was £15 each item. On the nth

page of the terms and conditions (who reads those!) they mention that 10kg of cabin baggage is free. Thirdly they charge £45 if you don't check-in online. Online check-in costs £12. Fourthly they charge £12 for using a credit or debit card. Using the Ryanair Mastercard is free, but this has to be pre-filled with cash and expires. Fifthly they try the same travel insurance trick when checking in. This time though the country is called "Don't want!"

When interviewed on BBC Radio 4 about these sorts of shenanigans, O'Leary said "It is my company and I can do what I like. If people don't like it they can go elsewhere". I did until this trip. He won, damn his eyes.

Now 10kg is not a lot, with one's mighty Canon, so my 17" macbook was not on. I took my lpad. I bought this earlier this year for my 'business' trip to Bill Cable again for weight reasons. I am sure you remember him as a QL trader - Wood and Wind Computing. I managed to take my folding Brompton because the iPad is so light. but that is another story. The only hassle with the iPad is no ports to upload photos. I use the brilliant Eyefi SD card. It acts as an access point and send pictures via wifi. My Austrian photos are here:

<http://tinyurl.com/9fwfhju>

On arriving in Bratislava, we exited through a building site and marginal border control. The arrivals looked pretty good (noted for later)! If I hadn't worked out in advance from the web what buses were available, I would have been totally lost. However the number codes on an insignificant bus stop matched. No mention of the bus company or the destination! I got on the very plush Slovak Lines bus, and settled down to what I guessed might be the only opportunity for a while to get my internet fix! 75 minutes later we were in central Vienna.

I had been to the central Holiday Inn before, so the journey was easy. Jochen and Andrea arrived later in the afternoon with a disaster. Andrea's car had developed what seemed to be an automatic transmission fault. so train to Prottes the next day.

Andrea and I thought about Pig, but as Marcel wanted to go to Figlmüller for the "largest Schnitzel in Austria" we did. Pig would wait until Sunday after Marcel had left. We got caught in mighty thunderstorm and the place was full. Fortunately many people abandoned the queue and we got the last table. I suppose we had mainly dried out by the time we sat



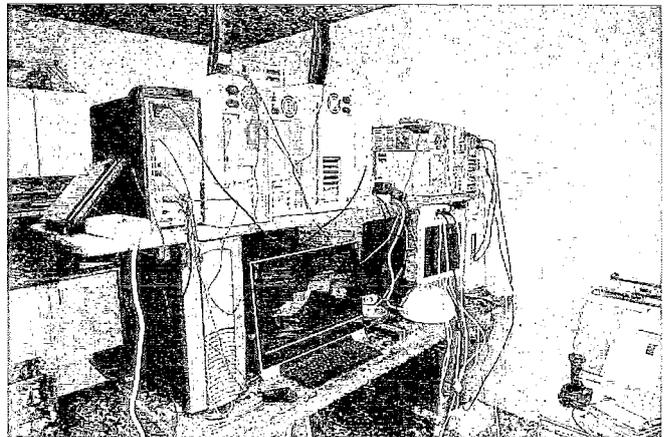
Laaarge Schnitzel - and Marcel is already Speed-Eating



Prottes Central Station



Chat at the venue



Computing power ... or is this controlling Prottes Railway Central Station?



The Grill-Master - well done, as last time!



More chat in the garden of the venue



Deep down in the dungeons, sorry, cellar.



Always the same faces - where have YOU been?

down. No Weißbier for Jochen - actually no beer at all.

We had an entertaining train journey to Prottes, passing again through the largest active inland oil field in Western Europe. Lots of donkeys nodding in the midst of agriculture. Quite bizarre. We arrived at a platform-less staff-less station next to a barrier-less level crossing. I assume Prottes is not a major commuter station.

All the usual cronies were installed at Gerhard's house and we had a splendid day chatting, playing with trains and consuming barbecued food. There were not too many real QLs in evidence. One special visitor was Louis Seidelmann who walked from the Czech Republic. Well not quite - he did plan to come by bicycle, but took a train to Gänserndorf and walked the 10KM to Prottes. Ironically we had passed through the same station a little while before. I supplied some microdrive cartridges to him last year, and was very pleased to see him at the show. In fact he provided the only new QL hardware at the show. He had manufactured new micro drive rollers and brought them for sale.

One diversion was a passing funeral procession complete with Los Angeles style band.

Back to Vienna and thence to what I announced was a really fantastic cellar restaurant - Esterhazy Keller. Well it would have been great but I had forgotten the restaurant (actually closed by our arrival time) was the right hand stairs. It goes down three stories into the mediaeval brick lined caverns. In our second best cavern, we had to make do with a cafeteria. Never mind - it had Weißbier for Jochen! Avoiding all the very good local beer houses, we walked to the Danube and had a noisy drink in a trendy waterside cocktail bar. Andrea and I were still looking forward to our statutory Pig on Sunday.

Sunday was tram day at the Straßenbahnmuseum. We had a pleasant lunch under the trees in view of a mooning gnome. I was saving myself for the evening Pig. The tram museum did not have any of the real drama of Strasshof in 2010, but it was very interesting. Engineering today is not nearly as solid as it was a hundred years ago. We go for lightness, economy, fragility and unreliability.

Come the evening, the weather was debatable. It was hard persuading anyone to venture out of the door, but I assured Andrea and

Jochen there was a good restaurant a millisecond away. Sure enough we had a good meal (pizza!) a few raindrops away. I managed to avoid seeing who won the Grand Prix - I would watch that on my return! so no Pig, and I had come under false pretences.

Jochen and Andrea were going by limping car to Salzburg and made it just over the German border to the next ADAC breakdown services, who could attend to their car. They had no intention of being stranded car-less in Vienna. In fact the problem was a mis-firing engine and the car was returned as good as new for the home journey.

I went to central Vienna for lunch. I was going to try the *real* Esterhazy Keller this time. When I got near, my phone suddenly went wild - emails arrived. It had automatically connected to the free wifi



in the bar opposite. I had used the wifi in 2011 on a church choir trip to Vienna. I admitted defeat and ate in that bar and got my second internet fix.

I took the bus back to the better half of the Bratislava terminal and thence home. I arrived lighter as I left my iPad on the plane! It was replaced under insurance, and the police have been following up the 'theft' for months! Apparently no-one is seriously using it, as it has not connected with Apple. If it did it could probably be traced and given to its present 'owners' - Aviva.



So ended yet another memorable trip, and Pig will have to wait until next time.

[Jochen adds: once again, we would like to say a big THANK YOU to the organisers of the Vienna meeting for their hospitality! A nice event every time! Looking forward to seeing you all again!]

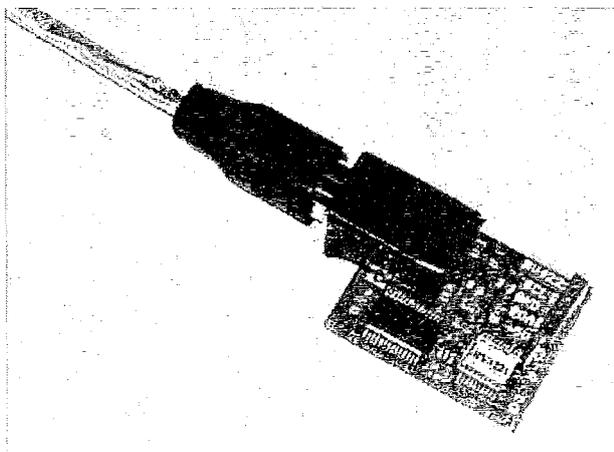
Too many faces on the previous page - now something different...

I2C Interface for QL Emulators

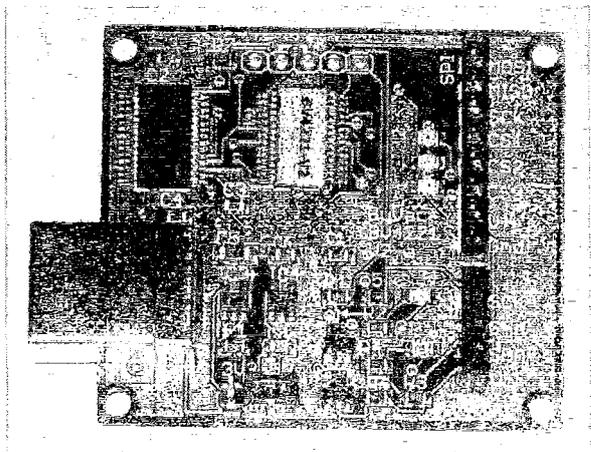
by Ian Burkinshaw

In this part I will deal with some updates and share my experience of using the BV4221-V2.

I will start with the BV4221-V2 from ByVac.



BV4221



BV4221-v2

The original version of the BV4221 was a PCB with the following dimensions 32mm x 25mm. The version 2 PCB is 45mm x 40mm. Version 2 PCB also has screw fixing holes as well. The other major feature differences between the two version is shown below:

- SPI interface added
- I2C address finder added
- Master clock rate selectable
- Inspector mode operates at 100k
- 5V or 3V3 logic switchable
- Two on board voltage regulators

This does offer some interesting opportunities, one of which I have included in my new routines is the I2C address finder. By using this, it is possible to check which I2C devices are connected to the converter. The return from the converter when the 'x' command is used, are the address(es) of all I2C devices connected. From the address range it is possible to determine what types of device are connected. This done from the first digit in the address hex code as follows:

<u>Address in hex</u>	<u>Device</u>
'4x'	PCF8574 Parallel port
'4x'	MCP23017 Parallel port (More on this device later)
'5x'	DS1803 Potentiometer
'7x'	PCF8574A Parallel port
'9x'	PCF8591 AD/DA converter
'Ax'	PCF8570 RAM
'D0'	DS1307 RTC (Real Time Clock)

Note the DS1307 has only one address, unlike the remaining devices, which use the second digit (x) to select multiple devices on the I2C bus, up to 8 devices with the devices we having been using in this series.

It is outside the scope of this series of articles, but the BV4221-V2 does have the other bus system, which can be used with the I2C bus system. This is called SPI (Serial Peripheral Interface bus). By the way if you wondering what the I2C ("I squared Cee") means it is "Inter Integrated Circuit" bus. I will not be going into the SPI now, but may revisit it in a later article. However, the two interfaces are not a million miles apart in that they are both serial interfaces, the I2C uses two wires, where the SPI bus uses four. Having said that, the protocols used are very similar. Typical SPI devices are the same as I2C devices but SPI being much faster is also used on memory devices and displays. The MMC standard (SD Cards) for example, have an SPI interface. It could be an interesting project, but after the problems Adrian Ives had with his SD Card products, not one I will try. So if you understand the workings of one bus system, it is not a big step to the other. Please see the references below for further information. The ByVac I2C Foundation pages are a very good place to start.

In my original articles on using the USB to I2C converter I was using version 1 of the BV4221, which is now no longer available. Recently I have purchased some BV4221-V2's, so I could test all my programs using this version. Now in part one of this series, I did say and I quote, *(Please note, I used the original V1 of the BV4221, ByVac now supply V2 which also has a SPI interface. The commands are the same, so the programs listed in the article should still work.)*. Well this remark was made on the basis of the manual for the BV4221-V2, which I had read, just to make sure there was nothing that would stop things working. However when it came to trying things practically it turned out not to be the case. The protocols are nearly the same, but the BV4221-V2 responses are slightly different. There are subtle differences between the original BV4221 and the version 2. This is stated in the BV4221-V2 user manual, but the manual is not that clear on what these differences are.

This has meant I have had to rewrite the start up routines to reflect these differences. The first difference is the time it takes for the BV4221-V2 to initialise – around 4 seconds. The original converter was virtually instant, or certainly less than 1 second. Or put another way I never caught myself out with the original converter not being ready to go when I started things up.

So you will now see that I have put in a couple of PAUSE statements to slow things down during to first start up stages.

The second difference is that on V1 the end of the received strings was character 32 which is the 'Space' character. This affected my original extract_read_data routine. The third thing I found was the returns to commands such as 'V' which asks for the version number of the converter was not as expected or the same at the version 1 converter. Also the response to the first CR (carriage return) was also not the same at the original converter. Now in my original programs I set the converter into decimal mode. This is where things caused the biggest problems. Since in Hex mode there was no difference between V1 and V2. In decimal mode on V2 the converter does not issue a CR after the version number. I informed ByVac and they agree this is a bug, and will be fixed in the next issue of firmware. However I have no idea when or if this will happen. So it is better to run the BV4221 in Hex mode. I also found the same problem with the new commands on V2 such as "x" which returns the addresses if devices connected to the I2C bus. Useful command this one, as you will see from the updated program below. So it is best to stay in hex mode. Other than the start up times and issuing CR's, there are no differences in using the commands between V1 and V2.

One other bug I discovered and also reported to ByVac is that in decimal mode, returns from the converter are getting truncated to two digits. This causes problems in reading data from devices because any return greater than 99 is now incorrect. So for example 255 gets returned as 55!!

Because I wrote the original routines in decimal mode, I was not aware the command strings are case sensitive. So if the hex numbers are sent as is from the SMSQ, commands such as HEX which returns the letters in upper case, this can cause problems. For example if you send the line "s A0 p", what happens is you end up changing the device address to "00". Since upper case "A" is the change address command for the converter. However if you send "s a0 p" then everything is OK. But note, the converter returns these hex letter digits in upper case, confusing. So you will find in my revised program below a routine to convert upper case hex letters to lower case, to be sent to the converter.

I also took the opportunity to improve the user experience and error trapping as well as running the BV4221-V2 in hex mode only, so as to keep away from the bugs outlined above. Also I added some features which the version 2 converter has which the version 1 does not. As always, my programs are just there to show what can be done, they are not fully developed, just to get you going for guidance and examples of how to use the I2C bus with this converter.

So you will find the new common routines and start up routine below. As always, I have commented them so you can see what is going on.

```
10 REMark I2C test routines
20 I2C_init
30 I2C_Start
40 :
50 PRINT
60 PRINT "LED Flash"
70 ledflash
80 PRINT
90 PRINT "LED Binary Count"
100 ledcount
110 PRINT:PRINT
120 PRINT "Input Test (Press 'Space bar' to exit test"
130 input_test
183 non_print_reply
250 PRINT
260 :
270 PRINT "End          ":CLOSE#3:STOP
280 :
500 DEFine PROCedure monitor
510 c$=""
520 REPeat test
530 a$=INKEY$(#3)
540 IF a$="" THEN GO TO 530
550 c$=c$&a$
560 END REPeat test
570 END DEFine monitor
580 :
```

Q U A N T A



Independent QL Users Group

World-wide Membership is by subscription only,

Offering the following benefits: Bimonthly Magazine - up to 52 pages

Massive Software Library - All Free! : Free Helpline and Workshops

Regional Sub-Groups. One near you?

Advice on Software and Hardware problems

1 year Membership Subscription £18 (includes eMag)

If you want a printed copy of Quanta magazine,

add the 2012 postage rates below

UK & NI £2.50, Europe £10.00, Rest of World £14.00

PayPal Surcharge about 5% - PayPal (see QUANTA Web Site)

Cash, Cheques and Postal Orders Accepted

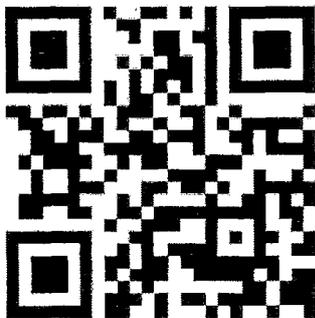
***** Now in our Twenty Ninth Year *****

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane, Stretford

Manchester. M32 9EH(UK).

Tel. 0161 865 2872



mail: membership@quanta.org.uk

<http://www.quanta.org.uk>

Email: membership@quanta.org.uk
and ask about our special
3 Year discount

```

1000 DEFine PROCedure I2C_init
1010 CLS
1020 ram1=174:ram1$="AE":REMark PCF8570 address, all address links open, ie all address pins high
1030 ram2=172:ram2$="AC":REMark PCF8570 address, A2=high, A1=high, A0=low
1040 ram3=170:ram3$="AA":REMark PCF8570 address, A2=high, A1=low, A0=high
1050 ram4=168:ram4$="A8":REMark PCF8570 address, A2=high, A1=low, A0=low
1060 ram5=166:ram5$="A6":REMark PCF8570 address, A2=low, A1=high, A0=high
1070 ram6=164:ram6$="A4":REMark PCF8570 address, A2=low, A1=high, A0=low
1080 ram7=162:ram7$="A2":REMark PCF8570 address, A2=low, A1=low, A0=high
1090 ram8=162:ram8$="A0":REMark PCF8570 address, all address links closed, ie all address pins low.
1100 parallelA1=126:parallelA1$="7E":REMark PCF8574A address, all address links open, ie all address
pins high
1110 parallelA2=124:parallelA2$="7C":REMark PCF8574A address, A2=high, A1=high, A0=low
1120 parallelA3=122:parallelA3$="7A":REMark PCF8574A address, A2=high, A1=low, A0=high
1130 parallelA4=120:parallelA4$="78":REMark PCF8574A address, A2=high, A1=low, A0=low
1140 parallelA5=118:parallelA5$="76":REMark PCF8574A address, A2=low, A1=high, A0=high
1150 parallelA6=116:parallelA5$="74":REMark PCF8574A address, A2=low, A1=high, A0=low
1160 parallelA7=114:parallelA6$="72":REMark PCF8574A address, A2=low, A1=low, A0=high
1170 parallelA8=112:parallelA7$="70":REMark PCF8574A address, all address links closed, ie all
address pins low
1180 parallel1=78:parallel1$="4E":REMark PCF8574 address, all links open, ie all address pins high
1190 parallel2=76:parallel2$="4C":REMark PCF8574 address, A2=high, A1=high, A0=low
1200 parallel3=74:parallel3$="4A":REMark PCF8574 address, A2=High, A1=low, A0=high
1210 parallel4=72:parallel4$="48":REMark PCF8574 address, A2=High, A1=low, A0=low
1220 parallel5=70:parallel5$="46":REMark PCF8574 address, A2=low, A1=high, A0=high
1230 parallel6=68:parallel6$="44":REMark PCF8574 address, A2=low, A1=high, A0=low
1240 parallel7=66:parallel7$="42":REMark PCF8574 address, A2=low, A1=low, A0=high
1250 parallel8=64:parallel8$="40":REMark PCF8574 address, all links closed, ie all address pins low
1260 adda1=158:adda1$="9E":REMark PCF8591 address, all address links open, ie all address pins high
1270 adda2=156:adda2$="9C":REMark PCF8591 address, A2=high, A1=high, A0=low
1280 adda3=154:adda3$="9A":REMark PCF8591 address, A2=high, A1=low, A0=high
1290 adda4=152:adda4$="98":REMark PCF8591 address, A2=high, A1=low, A0=low
1300 adda5=150:adda5$="96":REMark PCF8591 address, A2=low, A1=high, A0=high
1310 adda6=148:adda6$="94":REMark PCF8591 address, A2=low, A1=high, A0=low
1320 adda7=146:adda7$="92":REMark PCF8591 address, a2=low, A1=low, A0=high
1330 adda8=144:adda8$="90":REMark PCF8591 address, all address links closed, ie all address pins low.
1340 rtc=208:rtc$="D0":REMark DS1307 real time clock, one fixed address with this device.
1350 digpot1=94:digpot1$="5E":REMark DS1803 Digital Poteniometer, all links open, IE all address pins
high.
1360 digpot2=92:digpot2$="5C":REMark DS1803 Digital Poteniometer, A2=high, A1=high, A0=low
1370 digpot3=90:digpot3$="5A":REMark DS1803 Digital Poteniometer, A2=high, A1=low, A0=high
1380 digpot4=88:digpot4$="58":REMark DS1803 Digital Poteniometer, A2=high, A1=low, A0=low
1390 digpot5=86:digpot5$="56":REMark DS1803 Digital Poteniometer, A2=low, A1=high, A0=high
1400 digpot6=84:digpot6$="54":REMark DS1803 Digital Poteniometer, A2=low, A1=high, A0=low
1410 digpot7=82:digpot7$="52":REMark DS1803 Digital Poteniometer, A2=low, A1=low, A0=high
1420 digpot8=80:digpot8$="50":REMark DS1803 Digital Poteniometer, all links closed, ie all address
pins low.
1430 DIM tdata(7)
1440 DIM days$(7,3)
1450 RESTORE
1460 FOR a=1 TO 7
1470 READ d$
1480 days$(a)=d$
1490 NEXT a
1500 DATA "Mon","Tue","Wed","Thu","Fri","Sat","Sun"
1510 END DEFine I2C_init
1520 :
1530 DEFine PROCedure I2C_Start
1540 REMark Utilities for exploring the USB to I2C BV products
1550 REMark -----
1560 REMark Start up
1570 REMark -----
1580 REMark Determine Com Port Number
1590 CLS:PRINT
1600 sererror=0
1610 INPUT "Is USB to I2C connected (Y/N) ";i$
1620 IF i$="n" THEN PRINT "Program Aborted":STOP
1630 PRINT "Please wait, USBtoI2C is resetting":PAUSE 200:REMark Wait for power up reset of the
USBtoI2C converter to finish, just making sure the converter is ready.
1640 INPUT "Enter com port number 1,3, etc ";ser$
1650 ser$="Ser"&ser$
1660 PRINT "Opening Com Port ";ser$

```

```

1670 DoCom
1680 PRINT "Please wait, ensuring USB to I2C converter ready after opening Com port"
1690 PAUSE 200:REMark opening serial port, sends USB to I2C converter into reset. Note the SPI LEDS
    flash during this process.If you do not see the flash then the converter may not be fully reset.
1700 PRINT#3;CHR$(13);:REMark Carriage Return to set the baud in the USB to I2C converter, required
    on first pass to inialise USB to I2C converter.
1710 print_reply:print_reply:PRINT " Reply from USB to I2C converter after sending CR. The
    print_reply is called twice to handle the first return from the USB to I2C converter which is an
    echo of the CR sent"
1720 PRINT#3;CHR$(13);:print_reply:print_reply:PRINT " Reply from second CR sent, just ignore this"
1730 PRINT
1740 PRINT#3;"V";CHR$(13);:REMark Command to USB to I2C converter for converter firmware version.
1750 PRINT "Return USB Converter Version Number:-"
1760 non_print_reply:extract_read_data:I2CVer$=d$:PRINT I2CVer$:non_print_reply
1770 PRINT
1780 REMark print_reply:rem returns a device address in decimal.
1790 PRINT#3;"x";CHR$(13);:REMark This command finds I2C device adresses on the I2C bus.
1800 PRINT "I2C devices connected to the I2C bus :-"
1810 extract_read_data:PRINT d$:I2CDev$=d$:non_print_reply:REMark This should return all the device
    addresses that are on the I2C bus.
1820 decode_I2C_device I2CDev$
1830 display_I2C_devices_found
1840 PRINT
1850 REMark PRINT#3;"D";CHR$(13);:REMark send command to put USB to I2C Converter into decimal mode.
    Not recommended with V2 converters.
1860 REMark non_print_reply
1870 REMark PRINT "USBtoI2C Converter now in Decimal mode"
1880 REMark PRINT#3;CHR$(13);
1890 REMark print_reply:rem displays current address in decimal form.
1900 PRINT
1910 END DEFine I2C_Start
1920 :
1930 DEFine PROCedure print_reply
1940 c$=""
1950 REPeat loop
1960 a$=INKEY$(#3)
1970 IF a$="" THEN GO TO 1960
1980 IF a$=CHR$(13) THEN EXIT loop
1990 c$=c$&a$
2000 PRINT a$;
2010 IF a$="," THEN EXIT loop
2020 END REPeat loop
2030 END DEFine print_reply
2040 :
2050 DEFine PROCedure non_print_reply
2060 c$=""
2070 REPeat loop
2080 a$=INKEY$(#3)
2090 IF a$="" THEN GO TO 2080
2100 b$=a$
2110 c$=c$&b$
2120 IF a$="," THEN EXIT loop
2130 END REPeat loop
2140 END DEFine non_print_reply
2150 :
2160 DEFine PROCedure extract_read_data
2170 d$=""
2180 REPeat data_loop
2190 a$=INKEY$(#3)
2200 IF a$="" THEN GO TO 2190
2210 IF a$=CHR$(13) THEN EXIT data_loop
2215 IF a$="," THEN EXIT data_loop
2220 b$=a$
2230 d$=d$&b$
2240 END REPeat data_loop
2250 END DEFine extract_read_data
2260 :
2270 DEFine PROCedure decode_I2C_device (I2CD$)
2280 I2CDev=1:DIM I2CDevices$(10,2):HexC=1:I2CTemp$=""
2290 dlen=LEN(I2CD$)
2300 FOR count=1 TO dlen
2310 IF I2CD$(count)="," THEN I2CDev=I2CDev+1:NEXT count

```

```

2320 I2CTemp$=I2CTemp$&I2CD$(count):HexC=HexC+1
2330 IF HexC=3 THEN HexC=1:I2CDevices$(I2CDev)=I2CDevices$(I2CDev)&I2CTemp$:I2CTemp$=""
2340 NEXT count
2350 arrayt=10-I2CDev:arrayt=(10-arrayt)+1
2360 FOR count=arrayt TO 10
2370 I2CDevices$(count)=" "
2380 NEXT count
2390 END DEFine decode_I2C_device
2400 :
2410 DEFine PROCedure display_I2C_devices_found
2420 PRINT
2430 PRINT "Addresses"
2440 PRINT " Hex Binary Decimal Type of device"
2450 FOR count=1 TO 10
2460 HexDec=0
2470 I2CDev$=I2CDevices$(count)
2480 HexDec=HEX(I2CDev$)
2490 HexBin$=BIN$(HexDec,8)
2500 IF count >=1 AND count<=9 THEN PRINT count;" "&I2CDev$&" "&HexBin$&" ";HexDec;" ";
2510 IF count >=10 THEN PRINT count;" "&I2CDev$&" "&HexBin$&" ";HexDec;" ";
2520 IF I2CDev$(1)=" " THEN PRINT "No Device Found"
2530 IF I2CDev$(1)="4" THEN PRINT "PCF8574 or MCP23017 Parallel I/O"
2540 IF I2CDev$(1)="5" THEN PRINT "DS1803 Digital Potentiometer"
2550 IF I2CDev$(1)="7" THEN PRINT "PCF8574A Parallel I/O"
2560 IF I2CDev$(1)="9" THEN PRINT "PCF8591 A/D & D/A"
2570 IF I2CDev$(1)="A" THEN PRINT "PCF8570 256 Byte RAM"
2580 IF I2CDev$(1)="D" THEN PRINT "DS1307 RTC (Real Time Clock)"
2590 NEXT count
2600 END DEFine display_I2C_devices_found
2610 :
2620 DEFine PROCedure DoCom
2630 WHEN ERRor
2640 IF ERR_NI
2650 sererror=1
2660 PRINT "Error opening com port ";ser$
2670 PRINT "No Port open"
2680 END IF
2690 END WHEN
2700 REMark Set the size of the communications buffer to 16K
2710 Combuff=8192*2
2720 OPEN#3;ser$&"ir":REmark i=ignore hardware handshake, r=raw data
2730 BAUD ser$,115200
2740 SER_BUFF ser$,Combuff,Combuff
2750 IF sererror=0 THEN PRINT "Comport "&ser$&" open"
2760 END DEFine DoCom
2770 :
3000 DEFine PROCedure hex_case_con (uhex$)
3010 hex1$=uhex$(1)
3020 hex2$=uhex$(2)
3030 uh1=CODE(hex1$)
3040 IF hex1$>="A" AND hex1$<="F" THEN uh1=uh1+32
3050 uh2=CODE(hex2$)
3060 IF hex2$>="A" AND hex2$<="F" THEN uh2=uh2+32
3070 lhex$=CHR$(uh1)&CHR$(uh2)
3080 END DEFine hex_case_con
3090 :
4000 DEFine PROCedure ledflash
4010 FOR a=1 TO 10
4020 PRINT#3;"s-";paralle1A1$;" ff p";CHR$(13);:REmark s=start message to USB to I2C converter, p=end
of message to USB to I2C converter.
4030 non_print_reply:REmark Stops printing the USB to I2C reply also ensure serial buffer is cleared.
4040 PAUSE 25
4050 PRINT#3;"s-";paralle1A1$;" 00 p";CHR$(13);
4060 non_print_reply:REmark Stops printing the USB to I2C reply also ensure serial buffer is cleared.
4070 PAUSE 25
4080 NEXT a
4090 END DEFine ledflash
4100 :
5000 DEFine PROCedure ledcount
5010 FOR a=0 TO 255
5020 hhex$=HEX$(a,8)
5030 PRINT a;" ";hhex$;" ";

```

```

5040 hex_case_con hhex$:hhex$=lhex$
5050 PRINT#3;"s-";parallelA1$;" "&hhex$&" p";CHR$(13);
5060 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared.
5070 PAUSE 5
5080 NEXT a
5090 END DEFine ledcount
5100 :
6000 DEFine PROCedure input_test
6010 REMark REPEAT input_loop
6020 PRINT#3;"s-";parallelA1$;" ff p";CHR$(13);:REMark need to ensure any lines used as an input are
set high.
6030 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared.
6040 hexadd=HEX(parallelA1$):hexadd=hexadd+1:parallelin$=HEX$(hexadd,8):REMark adds 1 to the HEX
device address, for reading data from the device.
6050 REPEAT input_loop
6060 PRINT#3;"s-";parallelin$;" g-1 p";CHR$(13);:REMark reads input data.
6070 extract_read_data:AT 59,0:PRINT "Data return from selected device ";d$:non_print_reply
6080 FOR a=1 TO 200:NEXT a
6090 ax=KEYROW(1)
6100 IF ax&&64 THEN EXIT input_loop
6110 END REPEAT input_loop
6120 END DEFine input_test
6130 :

```

Now if you run this program you should get a result like this. It is the return from my test card, which has all the devices with the exception of the MCP23017, which we will come to next.

That is all for this time. Next time I will look at a cheaper alternative, the PCF8574(A) parallel device.

```

Is USB to I2C connected (Y/N) y
Please wait, USBtoI2C is resetting
Enter com port number 1,3, etc 5
Opening Com Port Ser5
Comport Ser5 open
Please wait, ensuring USB to I2C converter ready after opening Com port
(0x7E) Reply from USB to I2C converter after sending CR. The print_reply is called to
lice to handle the first return from the USB to I2C converter which is an echo of the
CR sent
Error not an I2C command Reply from second CR sent, just ignore this

Return USB Converter Version Number:-
2.2

I2C devices connected to the I2C bus :-
5E,7E,9E,AE,00

Addresses
Hex Binary Decimal Type of device
1 5E 01011110 94 DS1803 Digital Potentiometer
2 7E 01111110 126 PCF8574A Parallel I/O
3 9E 10011110 158 PCF8591 A/D & D/A
4 AE 10101110 174 PCF8570 256 Byte RAM
5 00 11010000 208 DS1307 RTC (Real Time Clock)
6 00000000 0 No Device Found
7 00000000 0 No Device Found
8 00000000 0 No Device Found
9 00000000 0 No Device Found
10 00000000 0 No Device Found

LED Flash

LED Binary Count
0 00 1 01 2 02 3 03 4 04 5 05 6 06 7 07 8 08 9 09 10 0A 11 0B 12 0C 13 0D 14 0E 15 0F
F 16 10 17 11 18 12 19 13 20 14 21 15 22 16 23 17 24 18 25 19 26 1A 27 1B 28 1C 29 1D
0 30 1E 31 1F 32 20 33 21 34 22 35 23 36 24 37 25 38 26 39 27 40 28 41 29 42 2A 43 2B
B 44 2C 45 2D 46 2E 47 2F 48 30 49 31 50 32 51 33 52 34 53 35 54 36 55 37 56 38 57 39
9 58 3A 59 3B 60 3C 61 3D 62 3E 63 3F 64 40 65 41 66 42 67 43 68 44 69 45 70 46 71 47
7 72 48 73 49 74 4A 75 4B 76 4C 77 4D 78 4E 79 4F 80 50 81 51 82 52 83 53 84 54 85 55
5 86 56 87 57 88 58 89 59 90 5A 91 5B 92 5C 93 5D 94 5E 95 5F 96 60 97 61 98 62 99 63
3 100 64 101 65 102 66 103 67 104 68 105 69 106 6A 107 6B 108 6C 109 6D 110 6E 111 6F
F 112 70 113 71 114 72 115 73 116 74 117 75 118 76 119 77 120 78 121 79 122 7A 123 7B
B 124 7C 125 7D 126 7E 127 7F 128 80 129 81 130 82 131 83 132 84 133 85 134 86 135 87
7 136 88 137 89 138 8A 139 8B 140 8C 141 8D 142 8E 143 8F 144 90 145 91 146 92 147 93
3 148 94 149 95 150 96 151 97 152 98 153 99 154 9A 155 9B 156 9C 157 9D 158 9E 159 9F
F 160 A0 161 A1 162 A2 163 A3 164 A4 165 A5 166 A6 167 A7 168 A8 169 A9 170 AA 171 AB
B 172 AC 173 AD 174 AE 175 AF 176 B0 177 B1 178 B2 179 B3 180 B4 181 B5 182 B6 183 B7
7 184 B8 185 B9 186 BA 187 BB 188 BC 189 BD 190 BE 191 BF 192 C0 193 C1 194 C2 195 C3
3 196 C4 197 C5 198 C6 199 C7 200 C8 201 C9 202 CA 203 CB 204 CC 205 CD 206 CE 207 CF
F 208 D0 209 D1 210 D2 211 D3 212 D4 213 D5 214 D6 215 D7 216 D8 217 D9 218 DA 219 DB
B 220 DC 221 DD 222 DE 223 DF 224 E0 225 E1 226 E2 227 E3 228 E4 229 E5 230 E6 231 E7
7 232 E8 233 E9 234 EA 235 EB 236 EC 237 ED 238 EE 239 EF 240 F0 241 F1 242 F2 243 F3
3 244 F4 245 F5 246 F6 247 F7 248 F8 249 F9 250 FA 251 FB 252 FC 253 FD 254 FE 255 FF
F

Input Test (Press "Space bar" to exit test
Data return from selected device FF

```

References

http://www.byvac.com/bv3/index.php?route=product/product&product_id=88

(Please note, I used the original V1 of the BV4221, ByVac now supply V2 which also has a SPI interface. The commands are the same, so the programs listed in the article should still work.)

<http://www.byvac.com/bv3/index.php?route=product/category&path=44>

PCF8570 Ram Data Sheet

http://www.nxp.com/documents/data_sheet/PCF8570.pdf

PCF8574(A) Data Sheet

http://www.nxp.com/documents/data_sheet/PCF8574.pdf
<http://focus.ti.com/lit/ds/symlink/pcf8574.pdf>

PCF8591 Data Sheet

http://www.nxp.com/documents/data_sheet/PCF8591.pdf

DS1307 RTC (Real Time Clock)

<http://datasheets.maxim-ic.com/en/ds/DS1307.pdf>

DS1803 Digital Potentiometer Data Sheet
<http://datasheets.maxim-ic.com/en/ds/DS1803.pdf>

MCP23017 Data Sheet

<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en023499>

I2C Tutorials

http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html

http://www.i2c.byvac.com/ar_foundation.php

http://doc.byvac.com/index.php5?title=I2C_Foundation#SPI

TF Services I2C manual

<http://www.dilwyn.me.uk/docs/manuals/index.html>

Advanced I2C information, but still worth a read to understand I2C protocols

http://www.nxp.com/documents/user_manual/UM10204.pdf

Programming in Assembler, Part 31 LibGen - Library Generator - Part 2

by Norman Dunbar

Introduction

In the last issue, I started the creation of the LibGen utility by explaining how to create the initial window using the latest version of SETW. In this article, we shall add code gradually, to enable the various features of the program, starting simple and getting more complicated as we go on.

LibGen Processing

The code for LibGen should work as follows:

1. The program starts with only the "Esc", "Move" and "Sym file" loose items enabled. Everything else is unavailable.
2. The user hits "Sym file". This causes all loose items except "Move" and "Esc" to be set to unavailable - in case the edit is aborted, or an error occurs. The user then types in the name of the sym.lst file created by George's sym_bin utility. The user then terminates or aborts the edit.

On a successful edit, the affected loose items are made available again (except "Save"). On an aborted edit, or error, the loose items are left unavailable, except for "Sym file" which is reset to available. The user will remain at this step until a successful edit completes.

3. The "Sym file" name entered by the user is changed by removing the "_sym.lst" extension and adding "_lib" in it's place to form the "Lib file" default file name, and by having the extension "_bin" added on to form the "Bin file" default value. These defaults are displayed in the appropriate information windows.
4. When a suitable symbol file name has been entered, all the application specific loose items will be enabled with the exception of "Save". The user may use the "Lib file" and "Bin file" loose items to amend the default file names for the two files that will be created by LibGen on hitting "Save".

5. When the user hits the "Load", the "Sym file" is opened and read in two passes. The first counts the number of code offset lines that will be added to the menu. The second pass will add each one to the buffer allocated, dynamically, for this purpose.

At end of file, the file will be closed and the buffer added to the application sub-window as a menu. All items in the menu will be selected by default. If the file loads correctly, the "Save" loose item will be enabled.

6. When the user hits "Save", the currently selected items in the application sub-window menu will be written out to the "Lib file", followed by a command to import the "Bin file". When complete, the file will be closed and all items will be set to the starting position where only "Sym file" is available.

LibGen Code

The first version of the code does nothing more than display the window on the screen and enter the loop to read the pointer and, as usual, this will only return (from WMAN) when an event happens or an error occurs in either a loose item hit routine or the application menu hit routine.

The following code is pretty much a template for any SETW/EasyPEasy built applications. It displays the window on screen and that's about all it does – pressing the ESC key or HITting/DOing the "Esc" loose item will end the program.

```
bra start
dc.w 0
dc.w $4afb

fname dc.w fname_e-fname-2
dc.b "LibGen - Library Generator"
fname_e ds.b 0
ds.w 0

;-----
; We need the various equates files etc.
;-----

in win1_georgegwilt_peass_keys_pe
in win1_georgegwilt_peass_qdos_pt
in win1_georgegwilt_peass_keys_wwork
in win1_georgegwilt_peass_keys_wstatus
in win1_georgegwilt_peass_keys_wman
in win1_georgegwilt_peass_keys_wdef

;-----
; Offsets into the data area for working storage.
;-----
id equ 0 Channel id storage.
wmvec equ 4 WMAN vector storage.
slimit equ 8 IOP_FLIM output buffer.
```

Departing from the template next, I define meaningful names for my loose items and information windows. It's much easier to determine which loose item or information window is being affected when reading the code back in 6 months or so, when you read names as opposed to a list of numbers.

I could have simply used an "IN" directive and a separate file at this point, which may prove useful for larger applications, but for now, I'm simply including the equates directly into my template.

You will note that the three strings I'm defining storage for are initialised to be zero length.

This is important because when we come to allow the user enter the symbol filename, the existing string is presented for editing. If we left the data uninitialised, we could get some interesting results as random strings were presented for editing. It's much better to initialise to an empty string as part of the program initialisation.

```

;-----
; Loose items we may need.
;-----
li_symfile equ 2
li_libfile equ 3
li_load    equ 4
li_save    equ 5
li_binfile equ 6

;-----
; Information windows we may need.
;-----
iw_symfile equ 2
iw_libfile equ 3
iw_binfile equ 5

;-----
; Working buffer for the three file names. 40 characters allowed. The
; three strings are initialised to be of zero length.
;-----
sym_buffer dc.w 0                A zero word count is useful!
           ds.w 20                Space for 40 characters inc N/L.

lib_buffer dc.w 0
           ds.w 20

bin_buffer dc.w 0
           ds.w 20

;-----
; Buffer for the 2 extra filename extensions we desire. These will be
; added to the end of the supplied sym file name from the user.
;-----
lib_extn dc.w lib_extn_e-lib_extn-2
          dc.b '_lib'
lib_extn_e equ *

bin_extn dc.w bin_extn_e-bin_extn-2
          dc.b '_bin'
bin_extn_e equ *

The remainder of the code is back to the template again.

;-----
; Console definition, and code to open it.
;-----
con      dc.w con_e-con-2        Size of channel definition.
          dc.b 'con_'
con_e    equ *

op_con   lea con,a0              We want a console.
          moveq #-1,d1            For this job.
          moveq #0,d3            Timeout.
          moveq #io_open,d0
          trap #2                Do it.
          rts

;-----
; The main code itself.
;-----

```

```

start   lea (a6,a4.1),a6      Make A6 point to the job's dataspace.
        bsr op_con           Open a con channel.
        move.l a0,id(a6)     And store the channel id.
        moveq #iop_pinf,d0   Trap to get Pointer Information.
        moveq #-1,d3         Timeout.
        trap #3              Do it.
        tst.l d0             Is ptr_gen present?
        bne sui             No, bale out via SUI.
        move.l a1,wmvec(a6)  Yes, store the WMAN vector.
        beq sui             Oops! WMAN wasn't actually found.

flim    movea.l a1,a2        The WMAN vector is required in A2.
;                                              The channel id is already in A0.
        lea slimit(a6),a1    Result buffer.
        moveq #iop_flim,d0   Query maximum size of window.
        moveq #0,d2         D2 is required to be zero.
;                                              D3 is the timeout.
        trap #3              Do it.
        tst.l d0             Did it work?
        bne sui             No, exit via SUI.

        subi.l #$C0008,(a1)  Minus 12 (width) & 8 (height).
        lea wd0,a3          Get address of window definition.
        move.l #ww0_0,d1    Get size of the working definition.
        bsr getsp          Easy PEasy - ALCHP memory and set A0.
        movea.l a0,a4       Which we save in A4.
        lea wst0,a1        Status area address.
        movea.l a1,a0       Copy to A0.
        moveq #wst0_e-wst0-1,d1 How many bytes to clear - 1.

```

So far, we have seen most of this before. However, we depart from the normal template in the next few lines, from label `st_clr` onwards.

```

st_clr  clr.b (a0)+          Clear one byte.
        dbf d1,st_clr      Then the remainder.

st_loose lea ws_litem+li_libfile(a1),a0  Status byte for Lib file.
        moveq #3,d1        Four status bytes to reset.

st_unav move.b #wsi_unav,(a0)+  Set loose item to unavailable.
        dbf d1,st_unav      And the rest.

```

First we initialise all of the status area, including the loose items, to a byte of zero, in the normal manner with the small loop at `st_clr`. For the loose items, this happens to be the status code for available.

However, as we don't want every loose item to be available when the program starts, the code at `st_loose` onwards will set the status byte for the 4 loose items in question, to unavailable – there is another way I can set these 4 status bytes, as we shall see later on in the code.

These will be made available by the code in other loose item hit routines as appropriate. I can use a loop at `st_unav` because of the order I created my loose items in SETW. The 4 loose items, "Lib file", "Load", "Save" and "Bin file" are set to unavailable in this loop. If you created your loose items in a different order, you may need to do each one individually.

Because of this status byte being set in the application's initialisation, these four loose items will be unavailable when the application displays its window on screen. Back to the template code again.

```

        movea.l id(a6),a0    Channel ID in A0.
;                               A1 = status area.
;                               A3 = window definition.
;                               A4 = working definition.
        move.l wd_xmin+wd_rbase(a3),d1  Get minimum size.

```

```

    andi.l # $FFFF, d1      Mask off scaling factors.
    jsr wm_setup(a2)        Set up the window.

    moveq #-1, d1           Use the current pointer position.
    jsr wm_prpos(a2)        Position as a primary window, then.
    jsr wm_wdraw(a2)        Draw the contents.
;-----
; The main Read Pointer loop.
;-----
wrpt    jsr wm_rptr(a2)      Enter read pointer loop in WMAN.
        beq.s no_err        Since D0 is zero D4 is non zero.
        bra sui            An error occurred exit via SUI.

no_err  movea.l (a4), a1     Status area address.
        btst #pt_can, wsp_weve(a1) Check for CANCEL event.
        bne sui            Exit.

        bra.s wrpt         No more events, read pointer again.

```

I have only included a check for the CANCEL event in the above code. Normally there would be SLEEP and SIZE event checking, probably as an absolute minimum. However, I need to keep the code size to a minimum for the magazine, so these applications will only have the minimum required.

Next comes the loose items and application window hit routines. In this first version of the code, the loose item hit routines for the following 4 loose items simply do nothing except reset the status from selected back to available when hit.

```

;-----
; Dummy, for now, loose item action routines.
;-----
afun0_6 bra li_reset       Bin file.
afun0_5 bra li_reset       Save.
afun0_4 bra li_reset       Load.
afun0_3 bra li_reset       Lib file.
afun0_2 bra li_reset       Sym file.

```

Before we delve into proper hit routines, the following table is a reminder of what registers are set on entry to a loose item hit routine.

Register Description

```

D1.L    High word = pointer X position, Low Word = pointer Y position.
D2.W    Selection keystroke letter, in its upper cased format, or;
        1 = Hit/SPACE or;
        2 = DO/ENTER.
        D2.W may be an event code if an event triggered this action.
D4.B    An event number - if an event triggered this action routine.
A0.L    Channel id.
A1.L    Pointer to the status area.
A2.L    WMAN vector.
A3.L    Pointer to loose menu item.
A4.L    Pointer to window working definition.

```

Next up is the first of our working loose item hit routines. This one handles the "Move" action. Also showing in the following code is the routine where we reset the appropriate loose item's status back to available from the currently selected status. You can see from the above, that the majority of the loose items simple reset their status and exit back to WMAN.

```

;-----
; MOVE hit. Move the window.
;-----
afun0_1 bsr move

;-----
; Reset current loose item status to available & redraw. Entry point
; li_reset resets the current loose item while entry at li_rest must
; have a loose item number in D1.W.
;-----
li_reset move.w wwl_item(a3),d1 Get the loose item number.

li_rest move.b #wsi_mkav,ws_litem(a1,d1.w) Set status to available.
        moveq #-1,d3 Request selective redraw.
        jsr wm_ldraw(a2) Do it.
        bra.s li_done

```

There are two entry points here, the first at li_reset handles the current loose item. If entry is at li_rest then D1W should be holding the appropriate loose item number. Within a hit routine, as you may remember, A1 holds the pointer to the status area and A3 points at the definition of the loose item within the working definition. By extracting the loose item number from the definition and adding it to A1 plus the offset to the start of the status bytes for the loose items, we can change the status to wsi_mkav which is actually the value available + redraw.

The code then calls wm_ldraw to redraw only those loose items which have the redraw bit set. This avoids flicker and doing unnecessary work redrawing unchanged loose items. When redrawn, the redraw bit is cleared by WMAN, leaving the status at available.

The code finishes by exiting through li_done to clear out the D0 and D4 registers to indicate no errors and no events. After this, it returns back to WMAN and back into the pointer loop.

```

;-----
; ESC pressed, set cancel event and exit.
;-----
afun0_0 bset #pt__can,wsp_weve(a1) Set the CANCEL event bit.
        moveq #pt__can,d4 CANCEL event number in D4.
        bra.s li_exit

li_done moveq #0,d4 No events.

li_exit moveq #0,d0 No errors.
        rts Exit, and exit from wm_rptr too.

;-----
; Application sub-window hit routine
;-----
ahit0 moveq #0,d4 No events.
      moveq #0,d0 No errors.
      rts Exit back to the read pointer loop.

```

The code that handles a hit on the "Esc" loose item shows the alternative manner of handling loose items. It simply sets the CANCEL event bit in the event register in the status area (addressed by A1), sets the event code in D4 and exits back to WMAN with D0 set to show no errors.

Having an event code in D4 causes WMAN to exit from the pointer reading loop and returns back to WMAN with D4 set. WMAN will see that an event has been set in D4 and this will cause a return to our own application code at label no_err (a long way above!). The code there checks for the CANCEL event and, if found, exits the program.

Note:

You may be wondering why we gave the ESC loose item a keystroke appropriate to the CANCEL event number (3) when we did a little editing of the file created by SETW in the last article, and why we have to have a loose item hit routine that sets the CANCEL event? Surely WMAN handles all that?

Well, if you comment out the first two instructions at afun0_0 above, reassemble and execute the program and then HIT or DO the ESC loose item, you will see it become selected, but the program still runs. However, if you press ESC, WMAN handles that and exits from the program.

So, we must have the hit routine cause the program to exit when the loose item is HIT or DOne because WMAN isn't seeing the ESC key being pressed to cause an exit. The hit routine for the lose item sets the CANCEL event which causes the return to WMAN to return to our code and thus, exits from the program. Simple?

Even if there was no loose item to explicitly close the program, as long as the application checked for a CANCEL event, as LibGen does, we can still close the program by pressing the ESC key because WMAN intercepts the ESC key, generates the CANCEL event and exits back to our application code where, hopefully, events are checked for.

Immediately following the "Esc" loose item code we have a dummy "does nothing yet" routine for the application window hit routine. Finally, the last few lines pull in the window definition created by SETW and George's EasyPEasy library.

```
-----  
; Pull in our window definition file.  
-----  
  
    in win1_source_qltoday_libgenWin_asm  
  
-----  
; We need George's Easy PEasy code next.  
-----  
  
    in win1_georgegwilt_peass_peas_sym_lst  
    lib win1_georgegwilt_peass_peas_bin  
  
-----  
; And finally, George's sprites.  
-----  
  
    in win1_georgegwilt_peass_csprc_sym_lst  
    lib win1_georgegwilt_peass_csprc_bin
```

If you save and assemble the above, you should be able to execute the utility and see it in "action".

You will only have the "Sym file" action, other than move or "Esc" available to you on startup and all the filenames will be blank. At the moment though, even if the "Sym file" loose item is enabled, the hit routine does nothing other than set the status back to available.

Now that we have the main part of the code to handle the initialisation, display and so on working, it's time to add some meat to the bones of what we have.

Handling the Sym File Loose Item

Looking back at our LibGen processing description above, we have already completed the first step. Step 2 requires us to let the user type in the symbol file name when the "Sym file" loose item is HIT or DOne.

Additionally, the "Load" loose item becomes available when the action routine completes. Keeping it simple for now, type in the following code at the location of the label afun0_2 which is currently showing a branch to the li_reset routine.

The following code replaces that which currently exists.

```
-----  
; SYM FILE loose item action routine.  
-----  
afun0_2  movem.l d5-d7/a0-a4,-(a7) Preserve important registers.  
        bsr sym_hit          Do it all.  
        movem.l (a7)+,d5-d7/a0-a4 Restore important registers.  
        moveq #li_load,d1     Load loose item.  
        bsr li_rest          Make Load available.  
        bra li_reset         Make Sym file available.  
  
sym_hit  rts                  Temporary code for now.
```

Once again, assemble and execute the code. Now when you HIT or DO the "Sym file" loose item, you should see the "Load" loose item become available.

The code preserves the registers we need to preserve over an action routine, branches out to our temporary hit code and on return, restores the registers before setting the "Load" loose item's status byte to available.

Finally, the code exits via the li_reset routine which causes the current loose item ("Sym file") to be reset to available and redrawn.

That's the easy bit done. The next part gets into the real code for a HIT or DO on the "Sym file" loose item. Replace the current one line at label sym_hit with the following code.

```
-----  
; This code carries out all the nasty work for a hit on the Sym file  
; loose item. It is called from afun0_2 above.  
-----  
sym_hit  moveq #iw_symfile,d1   Info window number in d1.w.  
        lea sym_buffer,a3      Current sym file buffer.  
        moveq #1,d2            Blue Ink colour.  
        bsr iw_input          Get input from desired info window.  
        blt.s sh_exit         Something went wrong, bale out.
```

It's at this point that having some equates defined for the various information windows comes in handy. The code above starts off by loading D1 with the number of the information window that will eventually allow us to type in the file name and which will also display the file name when we have typed it.

A3 holds the address of a buffer, which we set up way back at the start. On the first run of the program, this buffer holds a zero length string.

After it has been run and used, whatever the last symbol file name that you typed in will be there.

D2 needs to hold the ink colour, blue in this case, as we will be clearing the information window shortly.

The code is written so that when any information window is being used to edit data, the ink colour is blue, but when the data has been entered, it is printed with black ink.

We then branch off to a subroutine names iw_input to allow the user the ability to type in a file name directly into the appropriate information window. This routine will be discussed later.

On return, if any errors were detected, we bale out.

The calling code can handle this, as desired. In this example, LibGen does nothing with errors. The program continues to run, in this case, and you can try again, if desired.

```
-----  
; Did we abort the edit?  
-----  
sh_esc  cmpi.w #27,d1          ESC?  
        beq.s sh_sym          Yes.
```

The `iw_input` routine sets the terminating character in `D1`. This can be `ESC`, `ENTER` or the `Up` or `Down` Arrows. We are interested only in the `ESC` key as this implies that the user decided to abort the edit. If we find the `ESC` key terminated the edit, we bale out via the `sh_sym` label, which tidies up the potential garbage that is now showing in the information window.

The code at `sh_sym` also prints the file name in black ink as opposed to the editing colour of blue.

If the user terminated the edit normally, we have completed step 2 in our `LibGen` processing and are ready to carry out step 3. The following code does exactly that.

```

;-----
; Copy sym filename to other buffers and add appropriate extensions.
; The sym file is assumed to have a '_sym_lst' extension present.
;-----
sh_ok    move.l a2,-(a7)          Preserve WMAN vector.

        lea lib_buffer,a2       Destination buffer.
        lea sym_buffer,a3       Source buffer.
        bsr cp_string           Copy to lib file.
        subi.w #8,(a2)          Strip off '_sym_lst'.
        bcs.s sh_err            Negative is bad!
        lea lib_extn,a3         Lib file extension.
        bsr ap_string           Add lib file extension.

        lea bin_buffer,a2       Destination buffer.
        lea sym_buffer,a3       Source buffer.
        bsr cp_string           Copy to bin file.
        subi.w #8,(a2)          Strip off '_sym_lst'.
        bcs.s sh_err            Negative is bad!
        lea bin_extn,a3         Bin file extension.
        bsr ap_string           Add it to the bin file.

        moveq #0,d2             Black ink required for filenames.
        move.l (a7)+,a2         Restore WMAN vector.
        moveq #iw_libfile,d1     Info window required.
        lea lib_buffer,a3       String address.
        bsr iw_print            Print lib file.

        moveq #iw_binfile,d1     Info window.
        lea bin_buffer,a3       String address.
        bsr iw_print            Print bin file.
        bra.s sh_sym           Skip error handling.

;-----
; If the lib or bin filename lengths go negative after subtracting the
; 8 bytes necessary for the assumed '_sym_lst' extension, we bale out
; but need to tidy the stack first.
;-----
sh_err   move.l (a7)+,a2         Get the WMAN vector again.

```

The code above simply copies the file name entered by the user from the input buffer to the buffers set aside for the "Lib file" and "Bin file".

For each of these, the "_sym_lst" extension is removed and a new extension appropriate to the file name being generated is appended.

Note:

You will note that there is not much in the way of error trapping going on here. This is, again, to keep code to a minimum.

A proper application would do various checks to ensure that the symbol file actually existed, that it had the correct extension and so on, before manipulating the file name to create the defaults for the other two file names.

The only error trapping that is happening is a check that when subtracting 8 from the string length – to remove the characters '_sym_lst' – that the string length doesn't go negative. If it does, we bale out via sh_err and sh_sym where we tidy up the display again.

The strings are moved around and appended to using some more useful routines in one of my libraries. These will be discussed later.

Finally, for this action routine, we have the following code which calls yet another of my library routines, iw_print, to clear the information window in question, and print the contents of the correct buffer to it.

```
;  
; Print the sym file name. We do this at the end of a normal edit and  
; when the user aborts with ESC. This keeps the info window tidy.  
;  
sh_sym  move.w d1,-(a7)           Preserve the terminator keypress  
        moveq #iw_symfile,d1     Information window desired.  
        moveq #0,d2              Black ink.  
        lea sym_buffer,a3        Filename to print.  
        bsr iw_print             Print it.  
        move.w (a7)+,d1          Restore the terminator keypress  
  
sh_exit rts
```

Unfortunately, at this point, if you assemble the code, you will see 8 errors. All caused by a lack of my own library routines. The next section holds the code for the routines we are using, but have not yet created.

Unfortunately, we have had to split this article at this point due to space limitations ... more in the next issue.

17th Year of QL Today

by Jochen Merz

First of all, I would like to thank you, the readers, once again for re-subscribing. Many of you subscribed early, which is not only encouraging but also helps keeping the costs down. Sending out reminders costs money ... and I rather invest this money in more pages of the magazine, as you were able to see in the past - and now ... 50 pages instead the "average" 32. And the second "Thanks" goes to our authors - without them we would not be able to fill so many pages - without them, QL Today would not exist.

Every year I wonder whether we shall still have enough readers and authors to continue. Going back some years, when Roy Wood was handling the UK issues, we set ourself "red lines" where we would stop (e.g. when the readership falls under 400 ... and we continued ... then when the readership falls under 300 ... and we continued. I have given up on setting these lines, even though we are under 200 readers now.

I was a bit worried that the DVD last volume could have been interpreted as "we're finishing". It was not interpreted that way, we hope - and we do not finish, as you can see.

Unfortunately, the QL scene does not have as many supporters as the Spectrum scene (referring to Geoff's Editorial). Many people have left the QL scene. Does anybody know what they are doing nowadays? Whenever I think about the past, so many names come into mind - what has happened to them? Are they still connected to the QL somehow? I only know one person, Andreas Budde of former ABC-Ware, who still thinks about the QL and has tried to get Tony Tebby's operating system into various hardware in the past.

But what about all the other names? Freddy Vaccha of Digital Precision, to name an unforgettable person (everybody who saw him live at Microfairs will not forget him). What is Ron Dunnett doing? Nasta? How are the organizers of earlier QL shows doing nowadays? So many people we used to know and used to see at QL shows - does anybody know what they are up to nowadays? If you know anybody who knows about these people, please let us know. If you know these people and get them to write for us, even better!

The QL Show Agenda

Unfortunately, not much to see here. Unfortunately, not many visitors in Vienna this time (see article and pictures in this issue). It's a pity, as the organisers still try to add sightseeing, and not just the event - which on its own has been a nice, social event. Vienna two years ago was a big success - visitors from several countries - friends who have not met for some years!

Will there be a QL meeting somewhere in Europe? Is there any interest in a meeting somewhere? - maybe this is the better question.

Travelling has become so expensive nowadays, that the realization of a meeting, combined with holidays somewhere worth visiting, is probably the only way. Urs König did a great job with the meeting in Luzern, followed by the Austrian meeting a year later. Most visitors have not been to Vienna at this time, so there was a good reason.

Are there any other places worth visiting? Definitely! Many! And probably in areas who would allow people from many countries to visit. But do we have QLers in these areas who are prepared/willing to find a venue, or even have a venue like Gerhard Plavec in Prottes?

We have had Heidenreichstein and Vienna in the Eastern parts of Austria, Salzburg many years ago, Berchtesgaden several times (many sightseeing places!), Luzern ... if we continue this way, then the Schwarzwald/Freiburg/Basel area is still missing. Any QLers up there willing to do something? Any visitors willing to come? Please let us know! If anything is to be planned for next summer/autumn, then we need to know before Christmas for issue 3 at the very latest - better in issue 2 ... the Xmas issue. At the current exchange rate, Switzerland is probably unaffordable for us "EURO-sufferers". Visiting Germany, however, should be fairly cheap for Swiss visitors (and for UK visitors probably as well).

Or how about a meeting in the Hamburg area? This would allow people from the Northern countries to come as well - QLers which we have not seen or met for many years! Anybody prepared?

We need feedback - and we need early planning in case something should happen. Some kind of Holidays - at least an extended weekend - needs to be reserved and booked in advance. I can only speak for myself, but travelling long distances "just" for a day or two is just too stressful - and we do not talk about 100, 200km or so. Well, I guess we all are getting older.

You may be curious about the pictures on the cover. The small sheep is known to those who remember Tony Firshman's reference to "Klein Schniffi" two years ago. The sheep is our mascot and is travelling with us ... and it has seen many places so far. That's why it is looking forward to an unknown, not yet visited place next year.

The Schwarzwald/Freiburg/Basel area would definitely be interesting to me, Andrea and also Klein Schniffi. And - maybe for 2014 - a visit to the UK. I have not been there for several years and would love to come again and meet many of you which I have not seen for quite a while - remembering the "good old times". But as said above: just driving to a meeting and back is out of the question - it really has to be combined with some kind of holiday, sightseeing, touristical stuff.

So, let's plan ahead! Even if there is not much QL news, there are still many supporters (as you can see with the start of this volume of QL Today) and I feel that the QL scene is so unique, with so many nice and friendly people - often way beyond customer/dealer relationship - that it would be a shame to let this dry up. Please send us emails! How do you feel about events? Are you prepared to travel? Are you prepared to do something? Looking forward to your replies,

all the best

