QL Today DEUTSCH
Jahrgang 2
Ausgabe 1
Mai/Juni
1997
ISSN 1432-5446
Das Magazin über QL, QDOS, Sinclair Computer, SMSQ...

Buttons und Listings

QL *Today* DEUTSCH

ISSN 1432-5446

Herausgeber:
Jochen Merz Software
Im stillen Winkel 12
47169 Duisburg
Deutschland

Tel. +49 203 502011
Fax +49 203 502012
Box1 +49 203 502013
Box2 +49 203 502014

QL *Today* erscheint alle zwei Monate, Erscheinungsdatum der ersten Ausgabe ist der 15. Mai. Das Abo beginnt mit der aktuellen Ausgabe zum Zeitpunkt der Bestellung. Das Abo kostet wie folgt:

Deutschland	DM 70,-
England	DM 60,-
Rest der Welt	DM 70,-

Leser in Deutschland erhalten zum englischen Haupt-Teil auch einen deutschen Teil, der im Ausland für zusätzliche DM 10,- bezogen werden kann.
Zusätzlich zu **QL *Today*** ist auch eine Mitgliedschaft im deutschen Sinclair QL User Club zu ermäßigten Bedingungen möglich. Weitere Informationen und Antragsformulare sind bei Jochen Merz Software erhältlich.

Bezahlung kann in DM erfolgen, entweder mit Verrechnungsscheck einer Bank mit Sitz in Deutschland oder Euroscheck. Schecks sollten auf Jochen Merz Software ausgestellt sein. Es besteht auch die bequeme Möglichkeit der Einzugsermächtigung, auch hier nur bei Banken in Deutschland. Zahlung per Kreditkarte ist ebenfalls möglich - hier wird neben Ihrer Kartennummer auch die Gültigkeitsdauer benötigt.

Ihre Kommentare, Vorschläge und Artikel sind herzlich willkommen. SIE machen **QL *Today*** möglich. Wir verbessern das Magazin wo immer möglich, um Ihren Vorstellungen gerecht zu werden. Artikel sollten auf 3,5" Diskette (DD oder HD) eingeschickt werden. Das Format sollte ASCII, Quill oder Text87 (Druckertreiber angeben!) sein. Bilder sollten im _SCR-Format geschickt werden, GIF und TIF ist auch möglich. BITTE senden Sie auch einen Ausdruck der Bilder. Wenn ein Bild an einer bestimmten Stelle platziert werden soll, geben Sie es bitte auch an.

Redaktionsschluß für Artikel und Werbung:

Ausgabe 1:	15. April
Ausgabe 2:	15. Juni
Ausgabe 3:	15. August
Ausgabe 4:	15. Oktober
Ausgabe 5:	15. Dezember
Ausgabe 6:	15. Februar

Inhalt

Kleinanzeigen

Da QL Today eine der wichtigsten Quellen für QL-Neuigkeiten werden wird, werben auch die meisten QL-Händler hierin. Nun stellt sich natürlich die Frage, warum nur QL-Händler inserieren dürfen, warum nicht auch QL-User, die ihre eigenen Programme, Hardware oder Entwicklungen verkaufen möchten.

Anfangs konnte man selbstgeschriebene Software von Clubs vertreiben lassen, beispielsweise dem deutschen QL User Club. Aber warum soll man es denn nicht selbst vermarkten können? Es gibt hier keinen Unterschied zwischen privaten und kommerziellen Anzeigen, nur mit dem QL und Drumherum sollte es schon etwas zu tun haben.

Bis zu 50 Worte im englischen oder deutschen Teil kosten DM 5,- (oder 3 Internationale Antwortscheine), bis zu 100 Worte kosten DM 10,- (oder 6 Internationale Antwortscheine). Soll die Anzeige im deutschen und englischen Teil erscheinen, ist der Preis zu verdoppeln.

QL *Today* behält sich vor, eingeschicktes Material nicht zu veröffentlichen. **QL *Today*** ist unter keinen Umständen für die Richtigkeit der abgedruckten Artikel und Programmen haftbar, ebenso nicht für aus fehlerhaftem Material hervorgerufene Datenverluste, Unbenutzbarkeit oder ähnliche Probleme, die aus Artikeln in **QL *Today*** herrühren könnten. Die Meinung in diesem Magazin entspricht der des jeweiligen Autors und nicht notwendigerweise der des Herausgebers.

Dieses Magazin unterliegt dem Copyright und jegliches hierin veröffentlichte Material darf nicht ohne schriftliche Erlaubnis von **QL *Today*** reproduziert, übersetzt oder sonstwie verbreitet werden. Allen Copyrights und Trademarks wird hiermit Rechnung getragen.

Editorial

Jochen Merz

Liebe Leser,

nun liegt ein weiteres neues QL-Today Jahr vor uns. Schon wieder ist diese Ausgabe dicker geworden als geplant - 22 Seiten! Es kann daher gut sein, daß die nächste Ausgabe zum Ausgleich nur 18 Seiten umfassen wird, außer es treffen hier Unmengen von Artikeln ein (Wink mit dem Zaunpfahl!).

Wie man sieht hat sich nicht viel geändert - warum auch, es war ja gut so wie es war! Die einzige Veränderung liegt in der Verwendung einer neuen Schrift die, wie uns von manchen Test-Lesern bestätigt wurde, deutlich besser lesbar ist und größer wirkt, obschon der gleiche Informationsgehalt auf der gleichen Fläche untergebracht werden kann.

Für all diejenigen, die nur die deutsche Ausgabe erhalten, existiert auch wieder eine Zusammenfassung auf den letzten Seiten - sollte etwas für Sie besonders Interessantes in der englischen Ausgabe sein, das Sie gerne übersetzt haben möchte, so lassen Sie es uns wissen.

Dank an dieser Stelle allen Autoren, Dietrich Buder für's Kontroll-Lesen - und natürlich allen treuen Lesern.

Viel Neues gibt's:

SMSQ/E liegt in Version 2.83 vor - für das (Super)GoldCard Problem, das bei einigen Kunden auftrat, gibt es nun einen Debug-Modus (Erläuterung dazu gibt's beim kostenlosen Update dabei).

PROGS's neues LineDesign kann viel, viel mehr. Auch hier ist ein Update kostenlos, Voraussetzung für die neue Version ist allerdings der Besitz von ProWesS.

Auf der Hardware-Seite tut sich einiges: neue SuperGoldCards werden produziert (allerdings werden es die letzten sein, da es den Floppy-Disk-Controller-Chip nicht mehr gibt) und sowohl die Super-Duper-GoldCard (oder ColdFire) wird bald kommen - wahrscheinlich gibt es auf dem Treffen in Solms schon einen Prototypen zu sehen.

Miracle schlägt auch endlich zu - ein neues Projekt wird entwickelt: die UltraGoldCard.

In England gab es etwas Tolles zu sehen: Spezial-Gehäuse-Anfertigungen für Aurora, mehrere QL-Erweiterungs-Karten, zwei Disklaufwerke und zwei 5,25" Schächte - in schwarzem Metall, tragbar, mit Netzteil und optional auch allen möglichen Kabeln. Sehr beeindruckend! Das Ganze heißt Pandora. Wir hoffen, einen Test und Bilder in der nächsten (englischen) Ausgabe zu bringen.

Viele warten geduldig auf die Mehr-Farben. Nun, Tony arbeitet fleißig daran. Es ist doch wesentlich mehr Arbeit als anfänglich gedacht und zuerst wird es die Farben für die QXL geben. Hier ist es am einfachsten, da jede Grafikkarte 256 Farben kann - d.h. das Testen ist am einfachsten. Dafür muß jedoch das gesamte QXL-Interface neu geschrieben werden, und genau das geschieht momentan. Anschließend kommen QPC und Aurora dran.

Also: Geduld, es lohnt sich!

Zu den Mailboxen: noch einmal, die 502013 ist ideal für Modems bis 14.400 Baud und dem ZyXel 16.800. Die 502014 sollte nur für 28.800 und 33.600 Modems genommen werden. Der Inhalt beider Boxen ist identisch! In der letzten Zeit ist jedes Mal, wenn ich nach England gefahren bin, die Box gecrasht - bin ich zu Hause läuft sie wochenlang ohne Absturz. Zufälle gibt's... Nun ja, wenn die Box mal nicht antwortet wissen Sie: Jochen ist nicht da, also später mal probieren!

Ich hoffe, daß in Jens Wildgrubers Artikel nun alle Exponenten und Indizes ordentlich zu lesen sind - beim ersten Teil hatte ich sie leider unterschlagen. Und wo wir gerade bei kleinen Problemchen sind: die Cover Disk enthält vier defekte ZIP-Dateien, die glücklicherweise nur Zugaben und nicht Inhalt der Magazine war. Sie können die Disk gerne bei einer nächsten Sendung mitschicken oder sie einzeln mit 2,- DM Rückporto zusenden oder diese Dateien aus der Box herunterladen - sie sind einzeln zu finden und auch alle vier zusammen, damit Sie nicht großartig herumsuchen müssen.

Bei Einsendung von Artikeln wäre ein Hinweis auf alte oder neue Rechtschreibung nett, damit entsprechend Korrektur gelesen werden kann.

Also bis zur nächsten Ausgabe,

herzliche Grüße

Jochen Merz

Buttons - die Antwort

Wolfgang Uhlig

In der letzten englischen QL-TODAY-Ausgabe tauchte wieder einmal das Thema auf: Probleme mit QPAC2, dem Hotkey-System usw. und wie schreibe ich als Anfänger ein gutes Boot-Programm. Desweiteren nahm Geoff Wicks in sehr persönlicher Weise Stellung zur "button-mania" einiger Leute. Er fragte Leser zu begründen, warum sie Buttons nützlich finden. Ich will hiermit zu beiden Themen, die meines Erachtens stark zusammengehören, einen kleinen Beitrag liefern. Das Bootprogramm Mein Boot-Programm besteht in mehr oder weniger derselben Form, seit ich eine QXL-Karte und SMSQ/E als Betriebssystem habe. Natürlich finden immer mal wieder kleine Veränderungen statt, eine Software kommt dazu, eine andere fliegt hinaus. Hier mal ein Parameter verändert oder dort ein in dieser Phase oft benötigter HOTKEY dazu, aber das Gerüst bleibt. Da ich nicht auf anderen Konfigurationen arbeite, also kein Minerva, JS oder MGG, auch keine anderen Bildschirmauflösungen berücksichtigen muß, erübrigen sich auch Abfragen hinsichtlich dieser Dinge. Ich arbeite auf einer QXL mit SMSQ/E und that's it.

Das oben beschriebe Boot-Programm ist bis auf wenige Details genau mein Boot-Programm, natürlich liegen die einzelnen Programme in verschiedenen Unterverzeichnissen, die habe ich hier weggelassen. Außerdem benutze ich die deutsche Version von QPAC2 und deswegen sind einige Namen anders, z.B. 'Dateien' statt 'files'. Läßt man alle REMarks weg (von denen einige Leute wie ich hoffe, etwas lernen können, ist ein äußerst kompaktes und, wie ich finde, überhaupt nicht unübersichtliches Programm, das für viele QL-Benutzer völlig ausreichend sein sollte. Und damit sind wir dann auch gleich bei Geoff Wicks, dem "Button-Hasser". Ich finde es doch ein wenig gewagt, von der Nicht- oder doch Benutzung von Buttons charakterliche Qualitäten ableiten zu wollen, wo wie Geoff das tut. Meiner Meinung nach liegt das ganze etwas komplexer und hat mehr mit den ganz persönlichen Vorlieben zu tun, wie jemand mit dem Computer arbeitet:

Da sind die Menschen, die es vorziehen, immer nur ein oder zwei Dinge zugleich zu tun. Für sie ist es ein Greuel, wenn sich mehr als zwei Jobs auf dem Bildschirm befinden. Alles muß immer schön aufgeräumt sein und nichts darf die Augen ablenken. Das andere Extrem sind Menschen, die garnicht genug Jobs auf dem Screen haben können, die zu jederzeit alles verfügbar haben müssen, denen jede beim Suchen und Aufrufen von anderen Dateien oder Tasks verlorene Sekunde ein Riesendorn im Auge ist. Viele Menschen arbeiten sehr gerne mit der Tastatur, darunter vor allem jene, die viel schreiben. Sie können meistens mit 10 Fingern schreiben, haben eine Unmenge von Tastaturbefehlen, Macros und dergleichen mehr im Kopf und finden den Griff zur Maus eher störend. Das Gegenteil trifft häufig für jene zu, die viel zeichnen, malen, und nur mit zwei oder drei Fingern tippen können. Manche Menschen haben eine Lese-Rechtschreibschwäche und sind deswegen sehr dankbar, wenn sie mit der Maus auswählen können aus Items, die richtig geschrieben sind. Die Reihe von Bespielen ließe sich fortsetzen. Was jedoch deutlich wird ist, daß es nicht "die richtige" Methode gibt. Man ist deswegen auch kein Maschinenstürmer oder Häretiker, wenn man sich in der einen Richtung wohler fühlt als in der anderen.

An meinem Bootprogramm kann man sehen, daß ich zu den 'Knopfliebhabern' gehöre, wenn auch nicht in so extremem Maße wie zuvor beschrieben. Ich will an einem Beispiel erklären, warum das so ist. Seit ein paar Monaten schreibe ich an einem sehr umfangreichen Programm in SBASIC. Es wird bald zur Verwaltung eines Fahrradgeschäfts hier in Nijmegen, wo ich wohne, benutzt werden. Während der Arbeit an einem solchen Programm lernt man nicht nur unglaublich viel, sondern hat auch ständig Ideen: "Hier mache ich mir einen neuen Sprite", "hier muß ich mir eine kleine Extension schreiben", "hier könnt ich doch ein Logo in das Fax einbauen", "aus den Druckerbefehlen, die ich nötig habe für meinen HP könnte ich doch gleich einen Druckertreiber machen", usw.. Es hört nie auf, ständig starte ich andere Programme oder wecke bereits bestehende.

Zur Entwicklung brauche ich folgende Programme, nicht immer ALLE gleichzeitig aber die meisten schon:

- QD in einer Ausführung mit SBAS/QD-Thing zum Entwickeln und Ausprobieren
- QD in einer Ausführung mit QBASIC-Thing zum Kompilieren
- QD in einer gepatchten Version mit holländischer Menüführung
- QD um verschiedenste Datenfiles zu betrachten und verändern

- Text87 um eine Benutzeranleitung und eine Dokumentation zu schreiben
- EASYMENU um Menüs zu entwerfen, oft zweimal, um zu vergleichen
- EASYSPRITE um Sprites zu entwerfen
- APPMAN um obige zu verwalten
- DBAS_PTR_EXE, ein Databaseprogramm (PD), oft zwei oder dreimal, um verschiedene Databases zu vergleichen
- QFAX zum Ausprobieren, ob das Faxprogrammteil funktioniert
- QTPI zum Austausch zwischen dem Laden und meinem Zuhause und dabei natürlich auch
- ACP, das Packerprogramm
- SBASIC um Programmteile zu tracen

und last but not least

- die Dateienmenus win_, ram_ und flp_, die mehr oder weniger häufig an der Reihe sind.

Ich muß sagen, daß ich keine Lust hätte, zwischen diesen Programmen immer mit CTRL C hin- und herschalten zu müssen. Das Argument, all die Buttons würden ein heilloses Durcheinander auf dem Bildschirm anrichten, ist falsch, denn wie jeder weiß, überdeckt das Fenster, in dem man arbeitet, alle anderen. In Text87, z.B., sieht man nicht einen einzigen Button!

Gerade, WEIL ich einen halbwegs aufgeräumten Desktop mag, sind Buttons für mich wichtig! Will ich in einem Programm im Moment nicht arbeiten, lege ich es einfach schlafen. Es legt sich artig in den Buttonframe am rechten Bildschirmrand und wartet, bis es wieder an der Reihe ist. So habe ich auf dem Bildschirm ein oder zwei offene Programme, in denen ich arbeite und eine Reihe von Buttons, auf die ich jederzeit sehr schnell mit der rechten Maustaste zugreifen kann. Was ist daran unübersichtlich oder kompliziert?

WO ist der entscheidende Vorteil - um auf Geoffs Beispiel einzugehen - von:

a) mit einer Tastenkombination das 'hotkey'-menu aufrufen, durchscrollen und das Programm wählen, das ich brauche (= mindestens vier Tastendrücke)

gegenüber

b) mit der Maus auf den Button des gewünschten Programms gehen und rechts klicken (= 1 Click) ???

Es ist auch ein Denkfehler zu meinen, nur weil man Buttons hätte, müsse man sie auch immer benutzen. Wie wahrscheinlich viele habe ich ein paar geliebte Tastenkombinationen, mit denen ich schneller zum gewünschten Ergebnis komme als mit der Maus. Ist zum Beispiel meine Uhr (qwatchqxl) unter einem Programmfenster verschwunden, wäre ich natürlich dumm, sie mit Maus-Doppelklick, rechter Mausklick auf 'Pick' und wieder rechter Mausklick auf 'qwatch' holen zu wollen, dafür habe ich schließlich ALT 'u'.

Der große Vorteil des Hotkeysystems ist doch gerade die FLEXIBILITÄT die es bietet! Wie man im Bootprogramm sehen kann, ist es sehr einfach, wenn man sowieso schon alle möglichen HOTKEY-Definitionen gemacht hat, dafür auch noch Buttons anzulegen. Es gibt Leute, die tun es und es gibt Leute, die tun das nicht und beide finden das für sich völlig gut und richtig! Und so sollte es auch sein. Wer daraus charakterliche oder psychische Eigenschaften ableitet, sagt mehr über sich selbst als über den anderen.

ALT F1 = Wolfgang Uhlig, Delistraat 80,
NL - 6524 KS Nijmegen

```
100 REMark ***** Beispiel Bootprogramm für SMSQE
110 REMark ***** Zuerst einige Vorgaben (sog. "defaults")
120 PROG_USE win1_
130 DATA_USE win1_
140 DEV_USE 1,WIN1_mein_liebstes_Unterverzeichnis_
150 DEV_USE 2,WIN1_noch_eins_
160 :
170 REMark ***** jetzt die unmißbaren Dinge
171    REMark ***** Wenn Du kein SMSQ'E' hast, müssen hier
172    REMark ***** ptr_gen, wman und hot_rext geladen werden!
180 LRESPR Qpac2
190 LRESPR Menu_rext
200 LRESPR QLib_run_336mod
210 LRESPR qlib_bin
220 LRESPR History1v26c_cde: REMark ***** genial, muß man haben, holt mit den
230 his_def #0,50          : REMark ***** Pfeiltasten die letzten Befehle zurück
240 LRESPR EASYPTR_ptrmenr_cde
250 LRESPR qbasic_rext
260 LRESPR FileInfo2_bin
270 :
```

```
280 REMark ***** zwei Erweiterungen zum Programmieren mit DBAS, einem PD-Databaseprogramm
290 LRESPR DBAS_DBAS_BIN
300 LRESPR DBAS_DATA_BIN
310 :
320 REMark ***** größere Buchstaben auf einem QXL 800x600 Bildschirm
330 adr1=RESPR(875):adr2=RESPR(875)
340 LBYTES tall_1_qls,adr1
350 LBYTES tall_2_qls,adr2
360 CHAR_DEF adr1,adr2
370 :
380 REMark ***** um ein wenig Geduld fragen
390 CLS #0
400 PRINT #0, "hör auf zu nörgeln, ich bin doch gleich soweit!"
410 :
420 REMark ***** jetzt alle meine HOT_KEYS definieren
430 REMark ***** eine HOTKEY-Definition besteht immer aus:
440 REMark ***** 'ERT HOT_' mit einem Mitglied aus der 'LOAD'-'THING'-'RES'-Familie, und der
450 REMark ***** Klammer, in der man zuerst den 'key', also die Taste, angibt und dann
455 REMark ***** den Job, den der HOT_KEY auslösen oder aufrufen soll
460 REMark ***** Ordnet man einen kleinen Buchstaben zu, funktioniert der HOT_KEY mit kleinen UND
470 REMark ***** mit großen Buchstaben. Nimmt man einen Großbuchstaben, funktioniert der HOT_KEY
480 REMark ***** NUR mit Großbuchstaben.
490 :
500 REMark ***** zuerst mal die QPAC2 'Knöpfe'
510 ERT HOT_THING ('f','files')
520 ERT HOT_WAKE ('p','Pick')
530 ERT HOT_WAKE ('w','Wake')
540 ERT HOT_WAKE ('l','rjob')
550 ERT HOT_WAKE ('h','hotkeys')
560 REMark ***** die nächsten drei sind sehr nützlich und machen das System extrem gut handhabbar
570 REMark ***** der erste bringt immer Job 0 = BASIC nach oben
580 ERT HOT_PICK ('b','')
590 REMark ***** der nächste zeichnet den buttonframe neu und bringt den Mauszeiger (=pointer) auf
600 REMark ***** den ersten Button; so kann man ALLES JEDERZEIT zurückfinden!
610 REMark ***** Man kann QPAC2 so konfigurieren, daß das gleiche geschieht beim Druck auf beide
620 REMark ***** Maustasten. Starte einfach Menuconfig oder Config und gehe durch die
630 REMark ***** Konfigurationspunkte von QPAC2. Du kannst es unmöglich verfehlen.
640 ERT HOT_THING ('.','button_Pick')
650 REMark ***** der dritte bringt jeden Job zum Schlafen (gähn)
660 ERT HOT_THING ('z','button_sleep')
670 :
680 REMark ***** Jetzt einige Programme, die resident sein müssen (QD, wenn man das SBAS/QD THING
690 REMark ***** benutzen will) oder die man resident haben möchte wegen des viel schnelleren
700 REMark ***** Zugriffs. In diesem Beispiel ein paar kleine Programme von mir selbst.
710 ERT HOT_RES ('u','qwatchqxl_obj')
720 ERT HOT_RES ('û','QD')
730 ERT HOT_RES ("{","util_obj")
740 ERT HOT_RES ('d','wolledatum_obj')
750 ERT HOT_RES ('!','wollewecker_obj')
760 :
770 REMark ***** Hier sind sie endlich: meine liebsten Programme
780 ERT HOT_LOAD ('æ','EASYPTR_easysprite_exe')
790 ERT HOT_LOAD ('Æ','EASYPTR_easymenu_exe')
800 ERT HOT_LOAD ('ö','EASYPTR_appman_obj')
810 ERT HOT_LOAD ('x','DBAS_DBPTR_exe')
820 ERT HOT_LOAD ('ô','macro_exe')
830 ERT HOT_LOAD ("ò","LDESboot_obj"):REMark ***** ein Linedesign booter, nicht LD selbst
840 ERT HOT_LOAD ('ù','QD_lib'):REMark ***** QD mit QBASIC als THING
850 ERT HOT_LOAD ('T','text87plus4')
860 ERT HOT_LOAD ("Ö","menuconfig_german")
870 ERT HOT_LOAD ("ì","QLFED_obj")
880 ERT HOT_LOAD (" ","QTPI_qtpi155_exe")
890 ERT HOT_LOAD ("}",'ACP_obj')
900 ERT HOT_LOAD ("à","Qspread;'\mein_liebstes_Tabellenblatt'")
910 ERT HOT_LOAD ("L","qlib_obj")
```

```
920 :
930 REMark ***** Vielleicht fragst Du Dich, warum ich so viele Tasten nehme, die man sich nicht
940 REMark ***** merken kann. Ich BRAUCHE sie mir nicht zu merken, sie kriegen einen Button. Aber
950 REMark ***** der Vorteil ist: die meisten 'normalen' Tasten sind so frei für vorübergehende
960 REMark ***** ALTKEYS oder HOTKEYS während einer Sitzung und: die kann ich mir dann merken!
970 :
980 REMark ***** ALT/F3 startet einen SBASIC und tut ('do'), was in my_prog_bas (ohne Zeilennummern) steht
990 ERT HOT_THING (CHR$(240),'sbasic';'do myprog_bas')
1000 :
1010 REMark ***** einige Beispiele für HOT_KEYS
1020 REMark ***** schreibe meine Adresse mit ALT/F1 in jedem Programm wo Texteingabe möglich ist
1030 ERT HOT_KEY (CHR$(232),'Wolfgang Uhlig','Delistraat 80','NL - 6524 KS Nijmegen')
1040 REMark ***** ALT/F4 öffnet ein 'File_select' Fenster in SBASIC
1050 ERT HOT_KEY (CHR$(244),'outl:load file_select$(,,,,,,,1,)','')
1060 REMark ***** ALT/F5 setzt in QD einen Marker, entfernt alle Zeilennummern und kehrt zu Marker zurück
1070 ERT HOT_KEY (CHR$(248),CHR$(240)&"MM"&CHR$(32)&CHR$(240)&"ZL"&CHR$(240)&"MG")
1080 REMark ***** ALT/F6 das gleiche, aber mit einfügen der Zeilennummern
1090 ERT HOT_KEY (CHR$(234),CHR$(240)&"MM"&CHR$(32)&CHR$(240)&"ZE"&CHR$(240)&"MG")
1100 :
1110 REMark ***** der letzte ist ein Beispiel für HOT_CMD: wo Du auch bist, dieser Hotkey bringt
1120 REMark ***** Job 0, BASIC, nach oben, schreibt den Befehl (=command=cmd) in Kanal 0 und ENTERt ihn
1130 REMark ***** Frag mich nicht, warum QFAX mit ERT HOT_LOAD nicht geht!?
1140 ERT HOT_CMD (CHR$(250),'ex qfax;"-W 2"')
1150 :
1160 REMark ***** Okay, jetzt wird der die Buttonleiste (button-frame) gebaut.
1170 REMark ***** Für die Dateienmenüs nehme ich den EXEP Befehl. Dadurch werden sie zu normalen Jobs und
1180 REMark ***** ein Job kann einfach gelöscht werden, falls was schief geht.
1190 REMark ***** Die Parameter hier sind: '\b' = Farbkombination, '\n' = der Name, den DU dem Button geben
1200 REMark ***** willst und '\I' = Inhalt des Laufwerks
1210 REMark ***** Es gibt noch mehr Parameter, aber so geht's schon prima
1220 EXEP "files";'\b 3 \n WIN1 \I win1_'
1230 EXEP "files";'\b 3 \n WIN2 \I win2_'
1240 EXEP "files";'\b 3 \n RAM1 \I ram1_'
1250 EXEP "files";'\b 3 \n RAM2 \I ram2_'
1260 EXEP "files";'\b 3 \n FLP1 \I flp1_'
1270 EXEP "files";'\b 3 \n CD \I win3_':REMark ***** wenn das CD-Rom auf ›E: liegt auf Deinem PC
1280 :
1290 REMark ***** Jetzt zuerst die QPAC2 buttons. Ich muß zugeben, daß ich bis heute den genauen
1300 REMark ***** Untersschied zwischen BT_SLEEP, BT_WAKE, BT_HOTKEY und BT_EXEC nicht verstanden
1310 REMark ***** habe, aber so wie ich es gemacht habe, funktioniert's gut, also was soll's?
1320 REMark ***** Auch hier kannst Du dem Button einen Namen Deiner Wahl geben, zum Beispiel:
1330 REMark *****  ‹BT_SLEEP 'wake','aaufstehn Vatta, aufe Maloooche!'› oder sowas
1340 REMark ***** Klammern sind hier nicht nötig.
1350 BT_SLEEP 'Wake'
1360 BT_SLEEP 'Pick'
1370 BT_SLEEP 'Ljob'
1380 BT_SLEEP 'Jobs'
1390 BT_SLEEP 'Kanäle'
1400 BT_SLEEP 'hotkeys'
1410 BT_SLEEP 'sysdef','Neues Unterverzeichnis?'
1420 REMark ***** jetzt kommen die buttons für die wichtigsten Programme, die natürlich
1430 REMark ***** ge HOT_LOADed or _CHPed or _RESed sein müssen (siehe oben)
1440 REMark ***** Du nimmst die Taste, die Du oben definiert hast und gibst einen Namen Deiner
1445 REMark ***** Wahl, das isses!
1450 BT_HOTKEY 'ü','QD-BASIC'
1460 BT_HOTKEY 'ù','QD-LIB'
1470 BT_HOTKEY 'à','QSPREAD'
1480 BT_HOTKEY 'T','T91'
1490 BT_HOTKEY 'ò','LDES'
1500 BT_HOTKEY "Ö","CONFIG"
1510 BT_HOTKEY 'æ','SPRITE'
1520 BT_HOTKEY 'Æ','MENU'
1530 BT_HOTKEY 'ö','APPMAN'
1540 BT_HOTKEY 'x','DATABASE'
1550 BT_HOTKEY "}","PACKER"
```

```
1560 BT_HOTKEY "i","QLFED"
1570 BT_HOTKEY " ","QTPI 1.55"
1580 BT_HOTKEY "{","UTIL"
1590 REMark ***** und jetzt noch der Befehl, der alles zum Laufen bringt:
1600 HOT_GO
1610 :
1620 REMark ***** ein paar Sachen müssen wir noch machen
1630 EXEC freemembt :REMark ***** ein kleine Programm, das in die Buttonleiste geht
1640                 REMark ***** und den freine Arbeitsspeicher angibt
1650 HOT_DO "u"     :REMark ***** ruft die Uhr auf
1660 DISP_UPDATE 2,2:REMark ***** wichtig für die QXL
1670 IO_PRIORITY 10 :REMark ***** ebenso
1680 :
1690 REMark ***** zwei Tastatureinstellungen
1700 POKE_W 163980,20:REMark ***** setzt Verzögerung vor Autorepeat auf 20 (ms?)
1710 POKE_W 163982,1 :REMark ***** macht Autorepeat sehr schnell (...,5 ist langsam)
1720 TRA 3 :REMark ***** nur SMSQE! setzt tra table auf IBM
1730 EX win1_util_sysmon_exe :REMark ***** wichtiges Hintergrundprogrämmchen
1740 HOT_DO '.' :REMark ***** siehe oben, bringt Mauszeiger auf ersten Button, die Sitzung kann anfangen.
1750 :
1760 REMark ***** Eine Prozedur, die mich in die normale QL Auflösung bringt und Lonely Joker startet ;)
1770 DEFine PROCedure qql
1780 DISP_SIZE 512,256
1790 WMON:PAUSE 25
1800 EX LJ_exe
1810 END DEFine qql
1820 :
1830 REMark ***** und zurück zur Arbeit :(
1840 DEFine PROCedure svga
1850 DISP_SIZE 800,600
1860 END DEFine
1870 REMark ************************************************************
```

Hast Du Worte

Jochen Merz

Falls nicht, mit diesem Artikel und QTYP kann sich jeder haufenweise Worte aus bestehenden Wörterbüchern holen. Nachfolgend wird beschrieben, wie Sie alle Worte eines QTYP-Wörterbuchs in eine ASCII-Datei extrahieren können.

Verschiedene Kunden fragten mich wie dies denn gehe, und ich versprach, darüber zu schreiben. Es gibt verschiedene Gründe warum jemand die Worte aus einem Wörterbuch extrahieren möchte: wenn Sie das Wörterbuch editieren wollen kann es sein, daß Sie es einfacher finden, dies mit einem Editor wie QD zu machen. Außerdem ist dies der einzige Weg, zwei Wörterbücher ineinander zu mischen (also indem man ein Wörterbuch extrahiert und dann in das andere mit dem Wörterbuch-Editor mischt).

Das Programm zum Extrahieren ist in SBASIC für SMSQ geschrieben – wenn Sie es in SuperBASIC unter QDOS laufen lassen wollen, müssen Sie ein paar kleine Änderungen vornehmen – QDOS ist eben nicht ganz so komfortabel. Es werden auch zwei Funktionen der Menü-Erweiterung benutzt – FILE_SELECT$ zum Einlesen des Wörterbuch-Dateinamens und REPORT_ERROR, um eine Fehlermeldung auszugeben und auf die Reaktion des Anwenders zu warten. Natürlich müssen Sie auch die QTYP_SPELL-Erweiterung geladen haben, sonst haben Sie von BASIC aus ja keinen Zugriff auf QTYP-Wörterbücher.

Zuerst wird ein Fenster eingerichtet. Danach wird ein Dateiname eingelesen. SPELL_CLEAR löscht ein eventuell geladenes Wörterbuch aus dem Speicher und das angewählte Wörterbuch wird eingeladen.

Der Dateiname der Wortliste entsteht aus dem Wörterbuch-Dateinamen, gefolgt von "_words". Wenn dieser Name zu einem Problem führt wird abgebrochen. Bedenken Sie, daß die Länge des Dateinamens auf 36 Zeichen begrenzt ist.

Der String begin$ enthält alle Zeichen mit denen die Wörter des Wörterbuchs beginnen können. Das Beispiel enthält das normale

Alphabet und die deutschen Umlaute. Dies kann für andere Sprachen sehr einfach angepaßt werden.

Danach wird jedes Zeichen aus diesem String in QTYP's Vergleichs-Puffer geschrieben und so lange Worte gesucht bis nichts mehr Passendes gefunden wird, also SPELL_WORD$ einen Leerstring ergibt.

Auf diese Art kann man eine ASCII-Datei mit dem gesamten Inhalt eines Wörterbuches füllen. Sie können diese ASCII-Datei mit dem Wörterbuch-Editor in ein anderes Wörterbuch mischen.

Während ich diesen Beitrag schreibe habe ich festgestellt, daß das Beispielprogramm sich unter QPC nicht wie erwartet verhält - aber daran wird gearbeitet. Wir wissen noch nicht, ob es sich um einen Bug in QPC oder in der SPELL-Erweiterung handelt. Zu dem Zeitpunkt, zu dem Sie dies hier lesen, kann der Fehler bereits behoben sein. Schauen Sie einfach in den Versionsnummern der JMS-Werbung nach oder in Neuigkeiten der englischen Ausgabe.

```
100 IF DEVTYPE(#0)<>1:OPEN#0,con
110 OUTLN#0,512,256,0,0:IF DEVTYPE(#1)<>1:OPEN#1,con
120 WINDOW 512,256,0,0:BORDER 1,7:PAPER 0:CLS
130 CLOSE
140 :
150 REPeat
160  dic$=FILE_SELECT$('Wörterbuch auswählen',,,,,,,3)
170  ERT SPELL_CLEAR
180  dic=SPELL_NEW(dic$)
190  IF dic>0:EXIT
200  REPORT_ERROR dic,,,3
210 END REPeat
220 :
230 words$=dic$&'_words'
240 words=FOP_OVER(words$)
250 IF words<0:REPORT_ERROR words:CLOSE:STOP
260 :
270 ct=0
280 begin$='aäbcdefghijklmnoöpqrstuüvwxyz'
290 FOR c=1 TO LEN(begin$)
300  AT 22,0:PRINT \\\begin$(c)
310  ERT SPELL_CHECK(#dic,begin$(c))
320  REPeat
330   word$=SPELL_WORD$(#dic)
340   IF NOT LEN(word$):EXIT
350   PRINT#words;word$;
360   ct=ct+1:AT 24,8:PRINT ct
370  END REPeat
380 END FOR
390 CLOSE
400 PRINT \\'Done ...'
```

Easy-Tripel Teil II

Fortsetzung aus QL Today Deutsch Heft 3/96

Jens Wildgruber

Die Umwandlung der Zeitangabe in Sekundendarstellung benutzt u. a. bislang wenig oder gar nicht dokumentierte Fähigkeiten unseres Betriebssystems - ich rede hier, wie übrigens immer, nur von MINERVA und SMSQE - bei Aktionen auf dem Arithmetik-Stack. Die Übernahme des kompletten Arithmetik-Paketes von MINERVA nach SMSQE halte ich für eine überaus glückliche Entscheidung, sichert sie doch in einem gewissen Umfang Kompatibilität mit der am weitesten entwickelten QDOS-Version. Seit neuestem gibt es auch ein Update des Blattes Section 16,12 im QDOS/SMS Referenz-Handbuch, das über all die schönen neuen Funktionen informiert (MINERVisten finden sie natürlich im MINERVA Technical Guide unter ASM.3 dokumentiert).

Nun aber zur Sache. Zuvörderst muß die Liste der Systemzahlen auf den neuesten Stand gebracht werden; also am Anfang des Programms hinter 'UTWINT EQU $CE' einfügen:

```
BUFFER      EQU    -28
RI.STACK    EQU    -60
SMS.SPJB    EQU    $0B
CV.FPDEC    EQU    $F0
UT.WTEXT    EQU    $D0
QA.OP       EQU    $11C
QA.FLTLI    EQU    $09
QA.DIV      EQU    $10
IOB.SMUL    EQU    $07.
```

Der Programmabschnitt, an dem jetzt herumgebastelt wird, heißt 'unlink_pol'; nach dem TRAP #1-Aufruf (3. Zeile) geht es so weiter:

```
tripel_
     MOVE.L  (SP),A0          ; Fenster-ID wiederherstellen
     MOVE.W  A5,D2            ; Ort für den Ausdruck
     ADDQ.W  #1,D2            ;   der Tripelanzahl
     MOVE.W  A6,D1            ;      vorbereiten
     MOVE.W  #-1,D3
     MOVEQ   #IOW.SCUR,D0     ; Text-Cursor ist an der
     TRAP    #3               ; gewünschten Position
     MOVE.W  D7,D1            ; Tripelanzahl nach D1
     MOVE.W  UT.WINT,A2       ; Tripelanzahl drucken
     JSR     (A2)
     LEA.L   text,A1          ; ...und das Wort 'Tripel'
     MOVE.W  UT.WTEXT,A2      ; gleich hinterher
     JSR     (A2)
time MOVE.W  A5,D2            ; Ort für den Ausdruck
     ADDQ.W  #2,D2            ;    der Zeit
     MOVE.W  A6,D1            ;       vorbereiten
     MOVE.W  #-1,D3
     MOVEQ   #IOW.SCUR,D0     ; Text-Cursor ist an der
     TRAP    #3               ; gewünschten Position
     LEA.L   clock,A3         ; aufaddierte 50.tel Sekunden
     SUBA.L  A6,A6            ; A1 zeigt auf 32 bytes für
     LEA     RI.STACK(A4),A1  ; den Arithmetik-Stack
     MOVE.L  (A3),(A1)        ; Zeit in 20ms auf den Stack
     MOVEQ   #QA.FLTLI,D0     ; Umwandlung der Zeit von
     MOVE.W  QA.OP,A2         ; LongInteger in FloatingPoint
     JSR     (A2)
     MOVE.L  2(A1),8(A1)      ; Zeit in FP-Format wird auf
     MOVE.W  (A1),6(A1)       ; NOS kopiert
     MOVE.W  #$806,(A1)       ; neuer TOS: '50' im
     MOVE.L  #$64000000,2(A1); QDOS/SMSQ-FP-Format
     MOVEQ   #QA.DIV,D0       ; und nun die Zeit durch 50
     MOVE.W  QA.OP,A2         ; dividieren (NOS : TOS)
     JSR     (A2)
     LEA.L   BUFFER+2(A4),A0  ; Zeiger auf String-Buffer
     MOVE.W  CV.FPDEC,A2      ; wandelt FP zu ASCII dezimal
     JSR     (A2)
     LEA.L   BUFFER(A4),A0    ; String mit leerem 1. word
     ADDQ.L  #1,D1            ; D1 hält Stringlänge, wird um
     MOVE.W  D1,(A0)          ; 1 erhöht, da zum String auch
     MOVE.B  #'s',1(A0,D1.W)  ; die Maßeinheit 's' gehört
     MOVEA.L A0,A1            ; A1: Zeiger für UT.WTEXT
     MOVE.L  (SP)+,A0         ; Kanal-ID wiederherstellen
```

```
    MOVE.W  UT.WTEXT,A2     ; ....und drucken
    JSR     (A2)
    MOVEQ   #-1,D1          ; Priorität des Jobs
    MOVE.B  #8,D2           ; auf 8 zurücksetzen
    MOVEQ   #SMS.SPJB,D0
    TRAP    #1.
```

Danach geht es mit der Zeile MOVEM.L (SP)+,A0-A4, in der die Register wiederhergestellt werden, im ursprünglichen Programm weiter.

Unmittelbar vor 'tripel' ist einzufügen:

```
    MOVE.L  (SP),A0         ; Fenster-ID wiederherstellen
    MOVEQ   #0,D7           ; D7 wird Zähler für die Tripel
```

und ganz am Ende (vor 'end'):

```
text DC.W    7
    DC.B    ' Tripel',0.
```

Das Programm ist damit ein bißchen über die ursprüngliche Zielsetzung hinaus erweitert worden: die ausgedruckten Tripel werden in D7 mitgezählt, und das Ergebnis der Zählung durch die Zeilen hinter 'tripel_' auf den Bildschirm gebracht - von Interesse allenfalls für Zahlentheoretiker.

Im übrigen hoffe ich, daß durch die Kommentierung hinreichend klar wird, was da im einzelnen geschieht. QA.FLTLI ist eine jener Funktionen, die aus dem Arithmetik-Paket von MINERVA stammen. Sie wandelt eine LongInteger in eine Fließkommazahl um (was für ein Gehampel war das doch bisher, und ich möchte wohl wissen, wie häufig dieser unnütze Code in diversen Basic-Erweiterungen im Speicher herumlungert).

Zur eigenwilligen Benutzung von CV.FPDEC aber ist eine Begründung vonnöten. Aufgabe dieser Routine ist die Umwandlung einer Fließkommazahl, die auf dem Arithmetik-Stack liegt, in eine Zeichenkette, die in einen Puffer hineingeschrieben wird. Zur Weiterverarbeitung - wie etwa durch die Ausdruckroutine UT.WTEXT - ist aber häufig ein QDOS-String verlangt, also eine Zeichenkette, der die Längenangabe in einem word vorangestellt ist. Nach der offiziellen Dokumentation wird diese Längenangabe durch die Konvertierungsroutinen nicht bereitgestellt; durch einen Zufall habe ich aber bemerkt, daß die Längenangabe (und nicht nur im Falle von CV.FPDEC, aber systematisch bin ich der Sache nicht nachgegangen) in D1.W enthalten ist. Das steht nicht im Handbuch, auch nicht bei MINERVA, aber weil's funktioniert, verwende ich es eben. Natürlich ist das eigentlich 'illegal', aber ich gehe davon aus, daß an diesen Routinen sowieso nichts mehr 'verbessert' wird - schließlich gibt es wichtigeres an unserem Betriebssystem zu tun: etwa dafür zu sorgen, daß der CIRCLE-Befehl auf allen Bildschirmformaten etwas auf den Monitor bringt, das in einem vernünftigen Verhältnis zur 'Idee' eines Kreises steht (hallo, Tony!)...

Der TRAP #1-Aufruf am Ende dieses Programmteils setzt die Priorität des Jobs auf 8 (zurück); das macht nur Sinn, wenn sie vorher höher war. In der Tat ist es nur fair, die eigentliche Rechnerei auf unserem System mit erhöhter Priorität laufen zu lassen, da das Programm doch, was hiermit noch einmal in Erinnerung gerufen sei, auf Konkurrenz zu MS-DOS-Programmen angelegt ist, denen der Prozessor nahezu ungeteilt zur Verfügung steht. Dazu wären die gleichen vier Zeilen des TRAP #1-Aufrufes, nur mit '127' statt '8' im Register D2.B, unmittelbar vor dem Label 'link_pol', also nach der Sicherung der Anwendungsfenster-ID, einzufügen.

Wird nun das Programm nach dem Assemblieren gestartet und eine der Grenzen angewählt, erscheint auf dem Bildschirm '0 Tripel' und '0s' - wie aufregend! Also soll das Programm mal etwas sinnvoll Zeitraubendes zu tun bekommen - z.B. eine vollkommen leere Schleife 10 Millionen mal durchlaufen; das könnte etwa so vonstattengehn:

```
tripel
    MOVE.W  #199,D6    ; 200 Schleifendurchläufe
schleife0
    MOVE.W  #49999,D5 ; 50000 Schleifendurchläufe
schleife1
    DBF     D5,schleife1
    DBF     D6,schleife0;
```

danach geht's mit 'unlink_pol' weiter..

Als Zeitbedarf (in Klammern stehen die Werte mit ausgeschaltetem Cache), ermittelt nach dem

Löschen aller anderen Jobs, zeigt mir das Programm auf der SuperGoldCard nun '2.64s' (4.34s) an, und auf der mit 25MHz getakteten QXL-Card '3s' (12.38s) - nein, erklären kann ich es nicht, warum die QXL-Card dabei so alt aussieht, aber auf jeden Fall gibt es keinen Grund, letztere einzustampfen: ihre Stunde kommt, wenn die Schleife mit Inhalt gefüllt wird; und warum sollte man auch den Cache ausschalten? Überrascht war ich eher über die Wirksamkeit des Mini-Cache, mit dem der 68020er ausgerüstet ist (ganze 256 byte); bei durchschnittlichen Programmen sind es meistens nur ein paar Geschwindigkeitsprozente zwischen CACHE_ON und CACHE_OFF. Hier ist nun offensichtlich die Schleife so kurz, daß sie komplett in den Cache hineinpaßt, und das bringt einen Faktor über 1.6! Das ändert sich leider wieder, wenn mit dem Ausdrucken auf den Bildschirm begonnen wird - dann schmilzt der Faktor auf etwa 1.3 zusammen.

8. Das Basic-Programm entpädagogisieren

Eine Maßnahme zur Beschleunigung ist ja schon ergriffen worden, nämlich die Markierung der Dreiecksseiten als Integer-Variable. Ebenso läßt sich leicht einsehen, daß B% nur den Zahlenvorrat von 1 bis C%-1 durchlaufen muß, da Katheten anständigerweise immer etwas kürzer (und Integer-Katheten natürlich mindestens um 'eins') als die Hypothenuse zu sein haben. Das bringt nicht sehr viel, aber immerhin erspart es über 5000 Schleifendurchläufe!

Wird die Grenze, bis zu der der Zahlenvorrat auf Pythagoreische Tripel untersucht wird, n% und die Anzahl der Durchläufe der C-Schleife N_C, der B-Schleife N_B und der A-Schleife N_A genannt, gelten folgende Beziehungen:

$N_C = n\%$

$N_B = (n\%-1).n\%:2$

$N_A = (n\%-1).n\%.(n\%+1):6.$

Die folgende Tabelle zeigt die Anzahl der Schleifendurchläufe, die sich daraus für die vorgesehen Grenzen ergibt:

n%	100	150	200	400
N_C	100	150	200	400
N_B	4.950	11.175	19.900	79.800
N_A	166.650	562.475	1.333.300	10.666.600
N_A:N_B	34	50	67	134
N_A:Summe(N)	97%	98%	98.5%	99.3%.

Wie nicht anders zu erwarten, ist das Programm fast ausschließlich (mindestens 97% der Durchläufe) mit der inneren Schleife beschäftigt; jede Operation, die überflüssigerweise in ihr durchgeführt wird, muß mindestens 34mal (bis weit über 100mal) so häufig durchgerechnet werden, als wenn sie vor der Schleife stünde. Ein Beispiel dafür wäre die Berechnung von B^2 in der A-Schleife, und noch unsinniger ist dieselbe Maßnahme mit C^2 an derselben Stelle: bei n% = 100 kann C^2 nur 100 verschiedene Werte annehmen, wird aber 166650 mal ausgerechnet!

Konsequentes Auslagern solcher entbehrlichen Rechnereien spart etwa 40% der Ausführungszeit; das Programm könnte dann etwa so aussehen:

```
100 CLS: n% = 100: anzahl = 0
110 REMark Uhr anstellen, falls vorhanden
120 FOR C% = 1 to n%
130   C_QUADRAT% = C%*C%
140   FOR B% = 1 TO C%
150     DIFFERENZ% = C_QUADRAT% - B%*B%
160     FOR A% = 1 TO B%
170       REMark anzahl = anzahl + 1
180       IF DIFFERENZ% = A%*A%: PRINT A%,B%,C%
190     END FOR A%
200   END FOR B%
210 END FOR C%
220 REMark Uhr anhalten, Zeit auslesen.
230 REMark PRINT anzahl
```

In der inneren Schleife wird nun nur noch a^2 mit dem bereits vor Eintritt in die Schleife bestimmten Wert von $c^2 - b^2$ verglichen (die REMark-Zeile 170, mit der sich die Anzahl der Durchläufe der inneren Schleife mitzählen läßt, kommt gleich zum Zuge). Weitere Zeitersparnis läßt sich jetzt nur noch mit Abänderungen des Such-Algorithmus bewerkstelligen. So ist z.B. nicht einzusehen, zu welchem Zweck die A-Schleife noch weiter durchlaufen werden soll, wenn bereits ein passendes a^2 gefunden wurde - ein weiteres wird es ohnehin nicht geben. Dasselbe gilt, wenn a^2 bereits größer geworden ist als c2 - b2. in beiden Fällen könnte die innere Schleife sofort abgebrochen werden. Die Zeilen 160 bis 190 müßten dann etwa so aussehen:

```
160      FOR A% = 1 TO B%
170        anzahl = anzahl + 1
175        IF DIFFERENZ% <= A%*A%
180          IF DIFFERENZ% = A%*A%: PRINT A%,B%,C%
185          NEXT B%
187        END IF
190      END FOR A%
.......
230 PRINT anzahl
```

Das Mitzählen der Schleifendurchläufe in einer Fließkommavariablen ist natürlich ziemlich zeitraubend, aber es liefert dafür einen Meßwert für die eingesparten Rechenzyklen: 138362 Durchläufe anstelle von 166650, immerhin! Womöglich ist es sogar vorteilhafter, den Zahlenvorrat für a^2 nicht von unten nach oben, sondern von oben nach unten durchzukämmen? Dafür wären nur zwei Zeilen abzuändern:

```
160      FOR A% = B% TO 1 STEP -1
175        IF DIFFERENZ% >= A%*A%
```

Nun sind es nur noch 44715 Durchläufe der inneren Schleife, und wenn in Zeile 170 das REMark wieder eingefügt wird, zeigt sich, daß diese Abbruchbedingung etwa 60% der Ausführungszeit einspart. Alles in allem sind die Tripel mit der SGC nun in rund vier Sekunden auf dem Bildschirm. Falls jemand eine Idee hat, wie sich die Sache in Basic noch weiter beschleunigen ließe: nur her damit, ich lerne gern dazu!

9. Dasselbe in Assembler

Nein, natürlich nicht dasselbe; denn in mancher Hinsicht führt Basic auch in die Irre. So dauert z.B. das Durchlaufen einer Schleife in Basic ein bißchen länger, wenn der Schleifenindex abwärtsgezählt wird - auf der Hardwareebene, die durch Assembler angesprochen wird, ist es genau umgekehrt. Geradezu dramatisch unterscheiden sich beide Sprachen aber im Zeitbedarf für Ganzzahlrechnungen. In den Basic-Zeilen

```
100 a% = 1: b% = 1
110 FOR i = 0 to 100000
120   c% = a%*b%
130 END FOR i
```

kann man in Zeile 120 das '*' gern durch ein '+' ersetzen, es macht sich in der Ausführungsdauer kaum bemerkbar (ca. 4% schneller). Aber in Assembler....

Nun ist unser Programm zum Glück genau soweit gediehen, daß sich entsprechende Messungen damit durchführen lassen; es muß ja nur die 'schleife1' mit entsprechendem Inhalt gefüllt werden:

```
tripel
     MOVE.L    #1,D0      ; a% = 1
     MOVE.L    #1,D1      ; b% = 1
     MOVE.W    #199,D6    ; 200 Schleifendurchläufe
schleife0
     MOVE.W    #49555,D5 ; 5000 Schleifendurchläufe
schleife1
     MULU      D1,D0      ; a% =: a% * b%
```

```
        DBF     D5,schleife1
        DBF     D6,schleife0.
```

Nach Assemblieren und Programmdurchlauf ersetze man die MULU-Zeile durch

```
        ADD.L   D1,D0       ; a% =: a% + b%
```

und teste:

Ergebnis: Auf der SGC in einer relativ 'leeren' Maschine (d.h. alle Jobs außer Job 0 gelöscht und dessen Priorität auf 1 gesetzt) dauert das Multiplizieren 14,96s, das Addieren 3,52s. Zieht man davon die Zeit ab, die für das Durchlaufen der Schleife benötigt wird (2,64s), so verbleiben für

10000000 Multiplikationen 12,38s
10000000 Additionen 0,88s,

d.h. eine Multiplikation vom Typ MULU Dx,Dy dauert auf dem 68020-Prozessor unter vergleichsweise realistischen Bedingungen 1,23µs, eine Addition ADD.L Dx,Dy 0,088µs. Bei einem Takt von 24MHz auf der SGC wird eine Longword-Addition also in 2 Takten durchgeführt. Und weil die Meßeinrichtung nun einmal vorhanden war, habe ich gleich noch ein paar weitere Instruktionen untersucht:

ADD.L	Dx,Dy	0,088µs	2 Takte
ADDQ.L	#n,Dx	0,088µs	2 Takte
ADDI.L	#n,Dx	0,264µs	6 Takte
ADDI.W	#n,Dx	0,178µs	4 Takte
SUB.L	Dx,Dy	0,088µs	2 Takte
SUBQ.L	#n,Dx	0,088µs	2 Takte
SUBI.L	#n,Dx	0,264µs	6 Takte
SUBI.W	#n,Dx	0,178µs	4 Takte
CMP.L	Dx,Dy	0,088µs	2 Takte
CMPI.L	#n,Dx	0,264µs	6 Takte
CMPI.W	#n,Dx	0,178µs	4 Takte
DBF[1]	Dx,label	0,264µs	6 Takte
CMP.L Dx,Dy + BGE.S label[2]		0,264µs	6 Takte
CMP.L Dx,Dy + BGE.S label[3]		0,352µs	8 Takte
MULU	Dx,Dy	1,232µs	ca.30(32?) Takte
DIVU	Dx,Dy	2,198µs	ca.45(48?) Takte

[1]: bei Durchführung des Sprunges
[2]: wenn der Sprung nicht durchgeführt wird; auch BGT.S usw.
[3]: bei Durchführung des Sprunges; auch BGT.S usw.

Natürlich weiß ich, daß man so etwas auch in geeigneten Dokumentationen nachschlagen kann, aber was soll einer machen, wenn ein Handbuch eben nicht zur Hand ist und der Sonnabend sich schon anschickt, zum Sonntag zu werden?

Für die Pythagoreischen Tripel kommt folgendes dabei zum Vorschein:

Aufwärtszählende Schleifen brauchen mit CMP & Bcc & ADDQ 10 Takte für einen Durchlauf, während das DBF-Konstrukt beim Abwärtszählen mit 6 Takten auskommt. Wie ein glücklicher Zufall es will, ist in der entscheidenden inneren Schleife ohnehin das Abwärtszählen angesagt. Und Multiplikationen (Größenordnung 30 Takte) oder gar Divisionen (Größenordnung 45 Takte) wären demnach tunlichst zu vermeiden, und wiederum ganz zufällig ist dies bei der Berechnung der pythagoreischen Tripel auch möglich.

In der C-Schleife geschieht ja nichts anderes als die Berechnung von c^2 für die Zahlen von 1 bis n%, d.h. es werden aufeinanderfolgende Quadratzahlen bestimmt. Der Nachfolger (i%+1)2 einer beliebigen Quadratzahl $i\%^2$ läßt sich aber auch nach der 1. binomischen Formel berechnen: $(i\% + 1)^2 = i\%^2 + 2i\% + 1 = i\%^2 + i\% + (i\%+1)$;

ein Beispiel: $15^2 = 225$; $16^2 = 225 + 15 + 16$.

Die C-Schleife könnte also, angenommen C stehe in D2 und C^2 in D5, so

aussehen:

```
      MOVEQ     #0,D2      ; Initialisierung von C (=0)
      MOVEQ     #0,D5      ; Initialisierung von C² (=0)
c_schleife
      CMP.W     GRENZE(A4),D2 ; GRENZE(A4) hält n%
        BEQ     unlink_pol ; wenn C = n%, dann war's das
      ADD.L     D2,D5      ; D5: C²+C
      ADDQ.L    #1,D2      ; D2: C+1
      ADD.L     D2,D5      ; D5: C²+C+(C+1)=(C+1)²
      BRA.S     c_schleife.
```

Und in der alles entscheidenden A-Schleife mit ihren mordsmäßig vielen Durchgängen geht es ganz ähnlich: dort wird immer der Vorgänger einer Quadratzahl gesucht, und der läßt sich mit der 2. binomische Formel auffinden. Und so ergibt sich der eigentliche Kern des Programms mit überwiegend schnellen Instruktionen:

```
tripel
*******************************************************
* Registerverwendung:                                 *
* D0: A         D1: B      D2: C      D6: DIFFERENZ%    *
* D3: A²        D4: B²     D5: C²     D7: Tripel-Anzahl *
*******************************************************
      MOVEQ     #0,D2      ; Initialisierung von C (=0)
      MOVEQ     #0,D5      ; Initialisierung von C² (=0)
c_schleife
      CMP.W     GRENZE(A4),D2 ; GRENZE(A4) hält n%
        BEQ     unlink_pol ; wenn C = n%, dann war's das
      ADD.L     D2,D5      ; D5: C²+C
      ADDQ.L    #1,D2      ; D2: C+1
      ADD.L     D2,D5      ; D5: C²+C+(C+1)=(C+1)²
      MOVEQ     #1,D1      ; Initialisierung von B (=1)
      MOVEQ     #1,D4      ; Initialisierung von B² (=1)
b_schleife
      CMP.W     D1,D2      ; B = C?
        BEQ.S   c_schleife ; wenn ja, dann nächstes C
      MOVE.L    D5,D6      ; D6: C²
      SUB.L     D4,D6      ; D6: C² - B²
      MOVE.L    D1,D0      ; Initialisierung von A (=B)
      MOVE.L    D4,D3      ; Initialisierung von A² (=B²)
      BRA.S     vergleich
a_schleife
      SUB.L     D0,D3      ; D3: A² - A
      SUBQ.L    #1,D0      ; aus A wird A-1
      SUB.L     D0,D3      ; D3: A²-A-(A-1)=(A-1)²
vergleich
      CMP.L     D3,D6      ; DIFFERENZ% >= A² ?
        BLT.S   a_schleife ; nein, also weitersuchen
* da DIFFERENZ% >= A², muß jetzt auf Gleichheit
      CMP.L     D3,D6      ; geprüft werden
        BGT.S   next_B     ; war nix, also nächstes B
      ADDQ.L    #1,D7      ; Tripel gefunden - zählen!
*******************************************************
* hier wird die Ausdruckroutine aufgerufen            *
*******************************************************
next_B
```

```
ADD.L     D1,D4       ; D4: B²+B
ADDQ.L    #1,D1       ; D1: B+1
ADD.L     D1,D4       ; D4: B²+B+(B+1)=(B+1)²
BRA.S     b_schleife.
```

Werden nun für einen 'normalen' Durchlauf der A-Schleife (also DIFFERENZ% < A^2) die Takte nach der experimentell gewonnenen Tabelle aufaddiert, so dauert ein Durchlauf 14 Takte = 0.62 µs. Liegt die Grenze bei 200, wäre also ein Zeitbedarf von etwas unter 1s zu erwarten - ohne den zeitsparenden Abbruch der Schleife, der die Anzahl der Durchläufe und damit den Zeitbedarf auf ca. ein Viertel reduziert. So sollte die Programmausführung bei einer Grenze von 200 also etwa 0.2s bis 0.25s dauern.

Das kommt auch ganz gut hin; denn wenn das Programm nun assembliert und auf der SGC gestartet wird, zeigt es beim Anklicken von '200' an, daß es in 0.22s (ohne Cache 0.3s) 127 Tripel gefunden hat!

Weil seitenlange Assemblerlistings für Redakteur, Leser und Autor so überaus kurzweilig sind, mache

Neue Bildschirm-schriften für text87

Dietrich Buder

Die nachfolgenden Ausführungen sind speziell für normale QLs interessant und gelten exakt für den 24-Nadeldrucker STAR XB24-10, vermutlich aber auch für weitere Drucker.

Bekanntlich hat bei text87 das Aussehen einer Bildschirmschrift nichts mit dem späteren Druckergebnis zu tun. So ist es z.B. möglich, am Bildschirm in Courier zu schreiben und in Script auszudrucken; das hängt nur von der bestehenden Einstellung unter [F3] [K]onfig [S]chriften ab.

Wenn bei einem Schriftstück verschiedene Schriften oder Normal und Kursiv oder Fett und Dünn o.ä. erforderlich sind, sollten auf jeden Fall auch gut unterscheidbare Bildschirmschriften dafür genommen werden. Bei der kleinsten Unvorsichtigkeit nämlich springt text87 wieder auf die voreingestellte Hauptschrift zurück. Das wird zwar in der Ecke oben links angezeigt, fällt aber beim Schreiben häufig nicht gleich auf, sondern erst nach dem Drucken. In dem Fall muss später ein Teil des Textes umständlich verbessert und nochmals gedruckt werden.

Der wesentliche Vorteil von text87 gegenüber Quill aber besteht in der Option, auch proportionale Schriften verwenden zu können, sofern der Drucker das auch kann. Das trifft für alle neueren Drucker wohl zu.

Und nun bin ich bei meinem eigentlichen Thema. Ich möchte, dass nicht nur das Schriftbild einigermaßen stimmt, sondern auch die Abstände zwischen den Worten ungefähr richtig sind. Um diese Forderung zu erfüllen, müssen die proportionalen Zeichenbreiten der Bildschirmschrift exakt mit denen des Druckers übereinstimmen. Und genau an der Stelle lässt uns text87 bisher im Stich. Kurz gesagt, von den mehr als 40 mir bekannten Bildschirmschriften sind über die Hälfte nicht proportional und die ca. 15 proportionalen Schriften unbrauchbar.

Die für das Anzeigefeld von text87 benutzte Schrift 'DEFAULT11' ist sehr schön, hat aber bei vielen Zeichen zu kleine proportionale Breiten, bei einigen auch zu große. Mit ihr erfolgt der Zeilenumbruch, wenn die Bildschirmzeile erst zu ca. 80% gefüllt ist. Dadurch sind die Abstände zwischen den Worten auffallend groß.

Die anderen mitgelieferten Schriften haben teilweise zu große proportionale Breiten. Die Bildschirmschrift geht deshalb über die eingestellte Randmarkierung hinaus und es entsteht am Bildschirm ein scheußlicher Flattersatz. Natürlich ist der Ausdruck hinterher richtig.

Obendrein fehlen meist die Zeichen ab CHR$(128); also deutsche Umlaute und die weiteren Länder spezifischen Zeichen gibt es nicht.

Weiterhin haben die Ziffern nicht gleiche Breiten, was die korrekte Bildschirm-Darstellung einer Tabelle vereitelt.

Die Schrift Cinema hat viel zu breite Strichstärken und ist dadurch unleserlich.

Deshalb auch mein obiges, vernichtendes Urteil.

Ich habe mir nun die Mühe gemacht, mit Hilfe des Programms 'FOUNTED89' einige Bildschirmschriften bestmöglich meinem Drucker anzupassen. 100% korrekt ist das leider nicht möglich, wenn bei jeder Schrift eine gewisse stilistische Einheitlichkeit der Buchstaben erhalten bleiben soll. Für die interessierten Leser dazu nachfolgend einige Bemerkungen und Zahlen, andere Leser mögen hier abbrechen.

Wenn eine Zeile auf einem DIN A4-Blatt 166 mm breit ausgedruckt werden soll, muss bei text87 die rechte Randbegrenzung auf 166 eingestellt sein. Die linke Randbreite ist getrennt einstellbar und kann bei 30 mm liegen. Für den rechten Rand verbleiben dann 14 mm. Das sieht sehr gut aus und das Blatt lässt sich vernünftig wegheften.

Bei dieser meiner Einstellung passen 65 nicht proportionale Zeichen auf eine Zeile. Bei genauem Hinsehen stellt man fest, dass die nicht proportionalen Zeichen auf dem Bildschirm maximal fünf Pixel breit sind und in dem Fall zum nächsten Zeichen einen Abstand von einem Pixel haben. Eine Zeile hat also 65 * 6 = 390 Pixel und nutzt somit die QL-Schirmbreite von 512 Pixeln nicht voll aus.

Im Handbuch meines Druckers sind die proportionalen Zeichenbreiten (pbr) für IBM- und Standard-Betrieb verzeichnet, andere Handbücher wie vom Epson Stylus schweigen sich diesbezüglich aus. ***[Die genauen Zeichenbreiten und die Formeln zum Umrechnen der skalierbaren Schriften findet man für alle EPSON-Drucker im ESC/P Referenz Handbuch - Editor]***

pbr	Zeichen (Beispiele)
18	i l ! , . : ; ' |
24	f t I ([
30	1 2 3 a c e o s x z J Z " = ? # $ §
36	b d h k p u y A B C D E F G % &
42	m w M U W æ Æ

Die Breiten stehen erfreulicherweise in einfachen Zahlen-Verhältnissen zueinander, nämlich 3:4:5:6:7. Somit gibt es prinzipiell keinerlei Probleme für die Erstellung korrekter Bildschirmschriften. Die Breiten-Verhältnisse geben nämlich gleich die korrekte Zahl der Pixel vor, denn die nicht proportionalen Bildschirmschriften haben ja die Breite sechs für alle Zeichen.

Geringe Probleme bereiten nur einige wenige Buchstaben bei bestimmten Schriften: Größere Breiten würden wohl beim M oder W besser aussehen, könnten aber zu dem schon erwähnten Flattersatz führen.

Bei der Schrift Blippo sind für m und w nur sechs Pixel Breite möglich, wenn die Eigenheit dieser Schrift erhalten bleiben soll. Die dadurch entstehenden zu großen Abstände zwischen den Worten sind das kleinere Übel, denn größere Abstände zwischen den einzelnen Buchstaben eines Wortes sind noch schlimmer. - Das ist ja auch der Grund, warum es proportionale Schriften gibt.

Courier, Prestige, TW-Light u.a. haben Serifen, die kleinen Abschlussstriche an den Schrifttypen. Das i hat bei diesen Schriften unten Serifen nach beiden Seiten, was also zu einer Gesamtbreite von vier Pixeln führt. Da nur drei Pixel zulässig sind, kann hier kein korrekt aussehendes i gebildet werden.

Helvet, LetterGothic, Blippo u.a. sind serifenlose Schriften, bei denen das U mit sieben Pixeln eigentlich zu breit ist. Auch hier muss also zwischen korrekter Breite und gutem Aussehen ein Kompromiss gewählt werden. Noch weitergehende Feinheiten meiner Gestaltung sollen hier unerwähnt bleiben.

Als Höhe sind mindestens 11 Pixel erforderlich, wenn sowohl die Unterlängen von g p q und y als auch die übergroßen Zeichen wie Ä Ö Ü Ã Ñ vernünftig dargestellt werden sollen.

Bei der QL-Schrift mit 10 Pixeln Höhe haben deshalb ü und Ü eine um einen Pixel geringere Höhe als u und U; sieht nicht schön aus, aber anders geht es nicht, wenn die Buchstaben beim Lesen unterscheidbar bleiben sollen.

Mit dem Programm 'FOUNTED89' ist es ohne Schwierigkeiten möglich, einige serifenlosen Schriften zu erstellen, wegen der geringen Pixelzahl aber unmöglich, deutlich unterscheidbare Bildschirmschriften mit Serifen zu gestalten. Für Schriften mit doppelter Breite oder gar doppelter Breite und Höhe ließe sich das wohl machen, nur erscheint mir der Aufwand dafür zu groß.

Kursiv-Versionen der erstellten Schriften sind immer machbar, allerdings ist das Ergebnis nicht sonderlich schön. Es geht ja auch nur darum, dass sich die Schriften deutlich unterscheiden. - Ich hoffe, Jochen Merz gelingt es, mein mitgeliefertes Demo in QL Today einzubauen. Ein Bild sagt mehr als tausend Worte ***[einfach mal umblättern - Editor]***.

Ich bin gern bereit, falls seitens der text87-Nutzer Interesse besteht, diese meine überarbeiteten Bildschirmschriften zu Verfügung zu stellen.

Erwähnenswert erscheint mir noch eine persönliche Feststellung. FOUNTED89 ist neben Quill eins der wenigen Programme, die ich auf Anhieb in den Griff bekommen habe.

■

** Demo für text87-Bildschirmschriften **

Die nicht proportionale Schrift OCR-A ergibt 65 Zeichen pro Zeile

Die proportionalen Zeichenbreiten des Druckers STAR XB24-10 wurden für eine Reihe von text87-Proportional-Bildschirmschriften bestmöglich übernommen:

Courier **A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ**
Optimo A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ
Helvet A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ
LGothic A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ
Blippo A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ
Cinema **A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ**
Courier *A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ*
Optimo *A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ*
LGothic *A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ*
Blippo *A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ*
Cinema ***A quick brown fox jumps over the lazy dog. §äöüßÄÖÜµπ***

Dietrich Buder

01.03.97.

Der obere Teil zeigt, wie es auf dem Bildschirm aussieht, der untere, wie es gedruckt aussieht.

QL Today Englisch Mai 1997

Zum Inhalt der englischen Ausgabe QL-Today
H.P.Recktenwald

Eigene Anmerkungen in *[Klammern]*.

Wer Interesse an Texten jeweils zurückliegender Ausgaben hat, wird gebeten, dies mir mitzuteilen.

Nachricht dann bitte an H.-Peter Recktenwald, e-mail "phpr·berlin.snafu.de" oder Tel. 030 8520413 (jederzeit, evtl. Anrufbeantworter).

NEUES

- **QUBBESOFT** kündigt eine **Ethernet-Karte** für den Aurora-QL an *[bei Ergänzung des ROM-Erweiterungssteckers um eine Leitung zur Schreibfreigabe anscheinend auch für den schwarzen QL]*.

- Die **GOLDFIRE**-Karte wird auf 256M Bytes Ram ausbaubar sein.

- QUBBESOFT ist per Fax und Stimme unter derselben Rufnummer erreichbar: 00441376 347852 *[Vorwahl aus Deutschland]*.

QL Sound Card

Jeremy Reeves, 22 London Street, Kingswood, Bristol, England, ist im Begriff, eine preisgünstige Karte zur Ton-Aufnahme und -Wiedergabe herzustellen, die er für ca. 30 Pfund Sterling anzubieten beabsichtigt. Die Karte wird an den ROM-Anschluß des QL angesteckt - und wird aus Preisgründen NICHT mit Durchführungssteckern versehen sein.

Steve Johnson bietet die neue **Utility-Diskette** SJPD 17 an. Weiter SJS179 und SJS180 mit einer Anzahl neuer Bilder zu den Themen Landwirtschaft und Musik. SJS181 enthält die Version 3.33 von Ghostscript (wie von J.Hudson jüngst beschrieben). Das Angebot Klassischer Literatur wurde um eine große Anzahl neuer Texte erweitert - CB160...CB177.

Weitere "News" in der englischen Ausgabe.

Clock Trimmer

Die versehentlich nicht abgedruckte Programmliste ist nachgetragen, sie soll auch in der nächsten Beipack-Diskette enthalten sein.

Sortier-Routinen, Teil 2

Dilwyn Jones gibt einen kurzen Abriß über Funktionsweise und Anwendungsbereiche der in der vorigen Diskette enthaltenen Sortierprogramme:

Die Verfahren:

1. **Exchange Sort - Sortieren durch Vertauschen**. Nur für sehr kurze Datensammlungen, kurz aber langsam.
2. **Blubber-Sort - Durchhangeln bis zum geeigneten Ort**. Wohl das bekannteste *[aber auch nach "exchange" langsamste]* Sortierverfahren. Wegen des sehr geringen Programmaufwandes für kleine Dateien oft vorgezogen.
3. **Insertion Sort - Sortieren durch Einfügen**. Eine Variante des Bubble-Sort (2). Die Langsamkeit wirkt sich beim Einfügen/Entfernen von Datensätzen in bereits sortierte Dateien kaum mehr aus, das Verfahren wird dort also in vielen Fällen sogar eine gute Wahl sein.
4. **Shell Sort - Sortieren durch Teilung**. Für große Dateien mit Geschwindigkeitsvorteil.
5. **Quicksort - Rekursives Shell-Sort-ähnliches Verfahren**. Die Rekursion selber wird als ordnender Faktor genutzt. Darum ist das Programm (meist) kürzer und oft auch schneller.
6. **Pigeon Sort - Sortieren durch Einordnung**. Das jüngste Verfahren (Paul Birch, 1988). Hier *[wie aber auch bei anderen Verfahren möglich]* werden nicht die Datenposten selbst sortiert, sondern zunächst nur eine Liste von Zeigern, Postennummern o.dgl., was zum großen Teil die Ursache seiner besonders hohen Geschwindigkeit ist. Es zeichnet sich außerdem durch Einfachheit des Programms und Universalität der Sortierkriterien aus *[letzteres trifft auch auf jedes andere "tag-sort"-Verfahren zu, das Zeiger statt der Daten sortiert. Hierzu ein Beispiel in FORTH in der nächsten deutschen Ausgabe QL-Today]*. Gestaffeltes Sortieren ist bei solchen Verfahren besonders leicht. Man sortiert dann lediglich mittels der neuen Kriterien die zuvor erhaltene Liste.

Einfügen neuer Posten: Hier werden einige Überlegungungen vorgestellt, wie sie in der einschlägigen Literatur allenthalben behandelt werden. Dazu gibt es die Beispielprogramme "SORTS_ADD_LINEAR_BAS" und "SORTS_ADD_BINARY_BAS", deren Namen den Zweck beschreiben.

15 Programming Mistakes von Richard Mellor *[verfaßt]*.

Zur Erzeugung Leser-lichen Wohlwollens steht voran, daß wir alle ja Fehler machen, und daß es nur sehr wenige *[immerhin!]* Programme ohne Fehler gebe. Dann folgen Hinweise auf grundlegende Dinge, die auf jeden Fall beachtet werden sollen:

1. Die Annahme, der Anwender würde das Handbuch lesen. **Dem ist nicht so.**

2. Eingabefehler. Bei der Eingabe müssen jegliche auch nur entfernt möglichen (versehentlichen) Fehler zuverlässig abgefangen werden.

3. Beachtung der Schreibweise. Groß- und Kleinschreibung muß geeignet behandelt werden.

4. Unzureichende Dokumentation *[Was zu vermeiden angesichts (1) mitunter schwerfällt]*.

5. Prüfung des Programms. Mehrere mit dem Programm nicht vertraute Personen sollten es testen, da allein aufgrund der Arbeitsweise der Programmierer sonst niemals erreichte Fehlerstellen trotz gewissenhafter Prüfung unentdeckt bleiben müsssen.

6. Eingabegerät. Universalität bezügl. Maus-, Cursor- und Tastatursteuerung ist anzustreben, und es gibt *[außer der Umsatz"förderung" für eigene Produkte]* keinen vernünftigen Grund dagegen.

7. Variablenverwendung. Genaue Chronik über die Variablenverwendung vermeidet schwer zu analysierende Irrläufer. Dazu gehört auch, daß neue Variable nicht blind initialisiert werden, ohne daß ihre Anwendung bereits klar ist. Fehlermeldungen durch undefinierte Variable sind dann ein ohne jeden Zusatzaufwand geliefertes überaus nützliches Hilfsmittel. Dergleichen gehört zur inneren Dokumentation eines Programms, die mindestens so sorgfältig geführt werden sollte, wie die Beschreibung für die Anwender.

8. Programmentwurf. Der Autor favorisiert das "Top-Down"-Vorgehen, d.h. den Entwurf von der Zielvorstellung ausgehend durch die geeigneten Rahmenaufrufe mit noch leerem Inhalt vorzunehmen, und diesen Stück für Stück ins Detail gehend auszufüllen. Egal wie, auf jeden Fall ist neben der klaren Zielvorstellung die sorgfältige Planung aller Stufen erforderlich. Nur so werden dem Programmierer auch die eigenen Grenzen des Könnens auffallen, wo er sich ggf. entsprechend informieren wird, oder die Hilfe anderer in Anspruch nehmen *[edel formuliert, doch das dazu erforderliche Selbstbewußtsein scheint eher extrem rar zu sein...]*.

9. Ausführungsgeschwindigkeit. In SBasic beeinflussen lange Namen die Geschwindigkeit nicht, sie können darum leicht als Dokumentationshilfe dienen. *[Der noch folgende Hinweis auf den Einfluß nur der am häufigsten benutzten Aufrufe ist insofern fragwürdig, als es in einem System mit Multitasking nur sehr wenige Programmteile gibt, die nicht wenigstens insofern zeitkritisch sind, als sie während ihrer Ausführung Rechnerzeit beanspruchen, die darum anderen Programmen nicht mehr zur Verfügung steht]*.

10. Wieder-Erfindung des Rades... Es sollte möglich sein, bewährte Hilfsprogramme in ein neues Programm einzugliedern, so z.B. müssen die Druckertreiber nicht immer wieder neu programmiert werden *[wären sie denn nur etwas weniger "individuell" gestaltet.]*.

11. Die Systemherrschaft vermeiden. Mit Ausnahme weniger hochspezialisierter Programme ist es unsinnig, lästig und überheblich, wenn, wie leider in allzuvielen Fällen, der Verfasser eines Programms sein Werk für so überragend hält, daß daneben kein anderes Programm mehr im QL der Beachtung wert ist, und er darum das ganze System für sich pachtet, nicht einmal z.B. nur durch OPEN_IN ggf. den Dateienzugriff anderen Programmen erlaubt: Er befindet sich im Irrtum *[oder hat derartigen Mist verzapft, daß sein(!) Programm sonst Fehler erzeugt]*.

12. Verwendung undokumentierter Systemaufrufe ist nicht tolerabel, solche Schlaumeierprogramme werden über kurz oder lang nicht nur im QL mit Sicherheit scheitern.

13. Verwendung fremder Programmteile sollte nur bei deren genauer Kenntnis stattfinden, im übrigen s. (10).

14. Voraussetzen von Systemgegebenheiten macht Programme u.U. untauglich für eine Vielzahl interessierter potentieller Anwender. Meist muß das nicht sein. Erst, wenn eine spezialisierte Anwendung nur mit Hilfe bestimmter Eigenheiten bestimmter Systeme durchführbar ist, mag man das dulden *[wobei dann wenigstens die entsprechend detaillierten Hineise nicht fehlen dürfen]*. Das QDOS und die zugehörige Dokumentation *[die es n.b. auch in deutscher Sprache und kostenlos gibt]* erlauben die Programmierung in system-unabhängiger Form so umfassend, daß solche fixen Annahmen im allgemeinen unötig sind. Geht man noch weiter und bedient sich einer Computersprache, die auf völlig unterschiedlichen Rechnern einsetzbar ist, erleichtert das die Übertragung und entsprechende Programme werden einem weit größe-

ren Anwenderkreis zugänglich. Das ist mit der meist deutlich geringeren Ausführungsgeschwindigkeit abzuwägen, hat aber doch oft große Vorteile.

15. Zuviele Programme zu schnell schreiben endet meist im Desaster, oder belästigt den Anwender mit zahllosen "Versionen" in schneller Folge.

Der Aufsatz schließt mit der Ermahnung ab, man möge kein Programm als endgültig beendet ansehen. Es werden sich immer wieder Fehler finden oder Änderungen notwendig sein. Darum sollte ein Programmierer die Anwender seiner Programme nicht irgendwann im Stich lassen, indem er deren Unterstützung außer acht läßt.

True Confessions

Ein neues "Mein Boot-Programm", diesmal mit vielen "Buttons" und von Wolfgang Uhlig. [Lang. Für diesen Abriß zu lang].

Z88 Sourcebook

Eine frei erhältliche umfassende Zusatzdokumentation zum Z88, die eine Unmenge auch weniger Bekannter überaus nützlicher Details dieses Rechners beschreibt. Sie ist u.a. erhältlich bei

Mr. Ian Braby, Z88 Software Library,
1 Butts Cottages, Copse Road, St. Johns
Woking GU21 1SU, England.

Ein Katalog der P.D.-Bilbliothek wird für £1,50 (+ p.p.) versandt, die ganze Sammlung mit über 230 Programmen für £15.

QMAC Reviev Resumee

Es lohnt, sofern einer denn selber Programmiert, den Kauf dieses Assemblers in der jüngsten von QUANTA herausgegebenen Version in Betracht zu ziehen.

Letter Box

Ein Leser beschreibt seine Ratlosigkeit beim Übergang auf ein **System mit Harddisk**. Die Antwort von Dilwyn Jones: Es gibt verschiedene Möglichkeiten, Programme auf Harddisk zu übertragen. Die leichteste davon ist, den Anweisungen aus dem Handbuch zu folgen. Die nächste und wohl am ehesten allgemein anwendbare besteht darin, das **DEV-Device** zu benutzen, welches im Goldcard-Handbuch beschrieben ist. Mit dem DEV-Device kann fast allen Programmen anstelle von WIN ein FLP-Laufwerk vorgetäuscht werden. Dazu legt man zunächst für das betr. Programm ein Directory an, etwa mit

```
MAKE_DIR "WIN1_xyz_"
```

Dorthin kopiert man nun alle zugehörigen Dateien, z.B.

```
WCOPY"flp1_","WIN1_xyz_"
```

Dann wird das DEV-Device entsprechend präpariert

```
DEV_USE 1,WIN1_xyz_
DEV_USE 2,WIN1_xyz_
DEV_USE 3,FDK1_
DEV_USE 4,FDK2_
```

Womit die WIN- und FDK-Laufwerke mit Namen DEV unter der oben jeweils vorangestellten Nummer erreichbar werden. Schließlich der entscheidende Schritt

```
DEV_USE FLP
```

Nun gibt es DEV nicht mehr, dafür wird jeder Zugriff auf FLP entsprechend obiger Liste für DEV umgeleitet. Die eigentliche FLP ist aber nun überdeckt und nicht mehr auffindbar.

```
FLP_USE FDK
```

macht die Floppy-Laufwerke mit Namen FDK wieder zugänglich.

Um die alten Namen später wiederherzustellen, ruft man auf

```
DEV_USE DEV
FLP_USE FLP
```

Neben diesen Maßnahmen für den Betrieb von Programmen fehlt auch nicht der Hinweis, daß Schäden an einem WIN-Laufwerk wegen der ungleich größeren Datenmenge schreckliche Folgen haben können, und daß es darum unbedingt ratsam ist, regulär immer wieder aktuelle Sicherungsdateien anzulegen. *[Hierzu, auch automatisch, dient u.a. mein Programm CDISK mit der SBasic-Erweiterung CBACK, das kostenlos bei mir oder aus verschiedenen Mailboxen rsp bei "ftp.nvg.unit.no" in Internet erhältlich ist].*

Anmerkung des Editors: Die folgenden Artikel wurden erst in letzter Minute in die englische Ausgabe hineingebracht, daher nur ein paar ganz kurze Kommentare von mir:

QTPI-Einstellungen - sehr nützliche Informationen über die möglichen Einstellungen in QTPI.

QL-Shows - Bericht über die QL-Shows in Stafford und dem Quanta Jahrestreffen in London.

FPU Support - Sehr interessanter Artikel über Fließkomma-Coprozessor-Unterstützung für QDOS und SMS von Simon N. Goodwin.

■

Die zukünftigen QL-Treffen

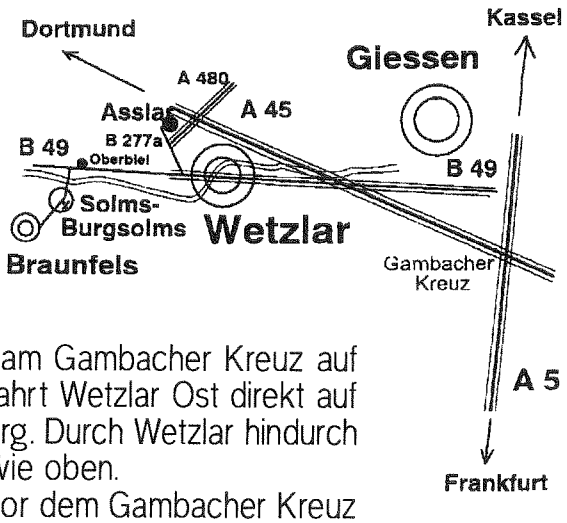
Die folgende Liste sollte Ihnen eine Idee geben wo demnächst QL Treffen stattfinden. Vielleicht ist ja eins in Ihrer Nähe, vielleicht auch etwas weiter entfernt und es lohnt sich trotzdem hinzufahren. Wie wollen hier sowohl große, wichtige Treffen als auch lokale Treffen auflisten - es kann ja sein, daß jemand in Urlaub oder beruflich in der Nähe von anderen Treffen ist und mal hineinschauen möchte. Also, Club-Regionalleiter usw. - bitte gebt mir Bescheid über die Daten. Englische Treffen sind hier wieder nicht gelistet, sie sind auf der Rückseite der englischen Haupt-Ausgabe zu finden.

24. Mai Eindhoven, Niederlande.

Das letzte Eindhoven war sehr gut besucht, Auch dies soll wieder ein größeres Treffen werden. Fahren Sie am besten so, daß Sie von Venlo aus auf der Autobahn nach Eindhoven kommen. Verlassen Sie die Autobahn am "Knoopunkt Leenderheide" (unter der Autobahn ist ein sehr großer Verteilerkreis) und fahren Sie Richtung "Centrum". Am nächsten Kreisverkehr biegen Sie links ab und bleiben Sie bis zur ersten Ampel auf dieser Straße. An der Ampel biegen Sie links ab, es sollte die "Roostenlaan" sein. Sie finden hier auch schon Schilder "St. Joris College" (hier findet das Treffen statt). Weitere Details gibt es bei Sjef van de Moolengraaf, Tel. +31 40-2442309.
Das Treffen startet um 10 und endet um 17 Uhr, aber seien Sie besser vor 15 oder 16 Uhr da!

15. Juni Solms, Deutschland in der Taunushalle.

Und wieder mal steht ein schönes QL-Treffen in Deutschland an. Solms liegt ziemlich in der Mitte Deutschlands, so daß die Nordlichter nicht allzu weit fahren müssen (wie im letzten Jahr) und auch aus dem Süden ist es eine erreichbare Nähe. Von Westen kommend über die Sauerlandlinie A45 aus Dortmund - Gießen am Wetzlarer Kreuz über die A480 und B277a auf die B49 (E44) in Richtung Weilburg/ Limburg. Nach ca. 4km kurz hinter dem OT Oberbiel geht es über die Lahn nach Solms-Burgsolms. In Solms der Hauptstraße in Richtung Braunfels folgen bis zur Taunushalle, die linker Hand liegt.



Von Norden über Kassel (A5) kommend am Gambacher Kreuz auf die A45 Richtung Dortmund bis zur Ausfahrt Wetzlar Ost direkt auf die B49 (E44) in Richtung Weilburg/Limburg. Durch Wetzlar hindurch (Schnellstraße) und dann hinter Oberbiel wie oben.
Von Süden kommend über Frankfurt A5 vor dem Gambacher Kreuz Richtung Wetzlar halten. Direkt über die A45 von Aschaffenburg her ab dem Gambacher Kreuz wie oben.