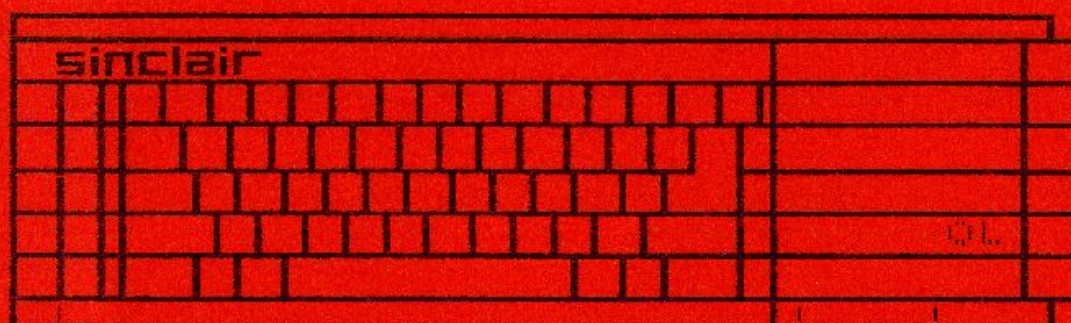


QLONE

QL GROEP NOORD NEDERLAND



december 1988
Jaargang 3 nummer 10

Een uitgave van : QL GROEP NOORD-NEDERLAND

VOORZITTER : J. Doesburg 2e : B. Johnson
Kalkwijk 7 Froukemaheerd 177
9603 BB Hoogezand 9736 RR Groningen
tel. 05980-27633 050-418699

SECRETARIS : vacature

PENNINGMEESTER : B. Scheidema
Rijksstraatweg 313
9752 CE Haren
tel. 050-345123

REDAKTIE QLONE : J.J. Doesburg - A. Stam - J. van Bruggen
Boesweg 27
9414 BD Hooghalen
tel. 05939-501

BANKREKENING : 48.09.90.468 t.n.v. B. Scheidema Haren
GIROREKENING : 3801095 t.n.v. B. Scheidema Haren

Contributie fl. 75.-- per jaar Abonnement QLONE fl. 20.--
Jeugdleden fl. 37,50. per jaar

De volgende clubavond is op 6 december 1988 in het Denksport Centrum, Olieveldersweg 43 te Groningen. Aanvang 19.30 uur.

Copy voor de 15e van de maand inleveren bij, of toezenden aan de redactie. Inleveren tijdens de clubavond kan ook. Listings tekeningen e.d. (brieven als Quill document) uitsluitend op een microdrive cartridge of 3 1/2 inch floppy disk.

Qlone wordt gemaakt op een QL met gebruik van Quill, Archive, SuperBASIC programma's en een QL printer.

INHOUD

Van de redactie	3
EX	4
XINPUT	7
TKII en assembly	10
Tip(s)	11
PF_dump	13
Turbo versus Qliberator	14

Van de redactie,

Beste Sint,

Wat ik graag wil hebben zult u me wel niet kunnen geven, maar ik zou all heel blij zijn als u onderstaand lijstje bij Sir Clive in de schoen zou willen leggen. Om hem zo er toe te bewegen zijn speeltjes in de vorm van schootcomputertjes in de steek te laten en zich weer eens te wijden aan het échte werk.

verlanglijstje:

- een diskdrive van 1,4 meg
- cache voor de diskdrive
- een hogere schermresolutie
- meerdere schermen
- betere geluidsmogelijkheden
- solid state drive met batterij backup
- een Centronics poort
- Toolkit II in rom
- bugs uit de rom
- RISC processor of 68030
- dma (direct memory access)

jvb

EX

De meeste Toolkit 2-bezitters zullen de bovenstaande twee letters herkennen als een commando om een "executable" programma -een job- in te laden en te activeren; genoemde Toolkitbezitters zullen EX dan ook zien als het equivalent van QDOS' EXEC commando.

Maar EX en zijn varianten EW en ET kunnen veel meer. Allereerst een toelichting bij deze vormen. EW zou men kunnen zien als Toolkit's EXEC_W: het laadt de job in het zogenaamde Transient Program Area (TPA) en activeert het maar suspendeert (=op non-actief zetten) de Superbasic-interpreter net zo lang tot de job volledig uitgevoerd is. ET laadt de aangegeven job maar activeert het niet (prioriteit 0). In de volgende uitwijdingen kan overal waar EX staat ook EW of ET gelezen worden.

EX kan gebruikt worden om bepaalde parameters aan de op te starten job mee te geven en wel volgens de volgende regels:

Syntax: [] = optioneel; ... = herhaling
EX program [,datafile]...[,datafile] [commandstring]

Met datafile wordt een input of outputdevice bedoeld waarmee de job (program) zijn data verwerkt (in stopt of uithaalt). Afhankelijk van wat de job nodig is kunnen de in- en/of outputapparaten een mdv_file zijn, een ser_connectie, het beeldscherm of elk ander device. De datafile mag ook aangeduid worden met #n (n=channelnummer), deze moet dan al wel geopend zijn. Program en datafile mogen omschreven worden als alle Toolkit files: met of zonder mdv1_, flp2_ enz.

Commandstring wordt op de job stack geplaatst en kan vervolgens door de job gebruikt worden als bijvoorbeeld een specifieke parameter. Een voorbeeld: we hebben een speciale job, geschikt voor gebruik met EX, en we noemen het UC (dit programmaatje wordt standaard bij Toolkit 2 geleverd). Het neemt een inputfile, bouwt alles om naar hoofdletters en stuurt het naar een outputfile. Het opstarten van dit programma met mdv1_test als input en ser1 gekoppeld aan #3 als outputfile gaat als volgt:

```
EX mdv1_UC,mdv1_test,#3
```

Een ander voorbeeld staat aan het eind van dit artikel.
Nog niet gaan juichen want echt fantastisch wordt EX nu pas.

EX kan een rij van samenwerkende jobs op starten die allemaal aan hetzelfde data-proces werken. De eerste job neemt de data, gebruikt en verbouwt het en stuurt het vervolgens door naar de volgende job die zijn aandeel levert. In deze zogenaamde procesline wordt de data "stream" genoemd, de verbinding tussen de jobs een "pipe" en de jobs zelf een "filter".

In plaats van een in te laden job mag de job ook de Superbasic interpreter zijn (is immers ook een job, job 0). Dit wordt aangeduid met #n (n=channelnummer). Als die channel al open was wordt het gesloten en wordt een nieuwe geopend voor de

procesline.

In EX ziet het er als volgt uit:

Syntax:

EX [#n TO] prog spec TO ... TO prog spec [TO #m]

Elke TO staat voor een pipe. n is een Superbasic output-channel, d.m.v. b.v. PRINT#n levert het de (eerste) data voor het proces. m is een Superbasic input-channel, b.v. met INPUT#m kan vanuit Superbasic de (uiteindelijk) geproduceerde data uit het proces gehaald worden.

Prog spec duidt een job (filter) aan en wel als volgt: program [,datafile]...[,datafile] [;commandstring]

Alle namen (de onderstreepte) mogen ook strings of string-expressions zijn, dit terzijde.

Het aantal en soort datafiles en ook de commandstring zijn dus afhankelijk van wat de job nodig is.

Over het algemeen zullen in een procesline de filters géén datafiles meegegeven krijgen. De filters hebben dan één (primary) input en één (primary) output file en dat zijn de andere filters aan het andere eind van de pipes. Dus: EX filter,<file> TO filter TO filter TO filter,<file>

Alleen de eerste en laatste hebben een datafile <file> voor de eerste en laatste data van de stream de rest gebruiken de voorafgaande filter als bron en de volgende filter als bestemming. Als wel een datafile opgegeven was zouden er dus twee outputs zijn: de datafile en de volgende filter.

De eventuele datafiles die als input door een filter worden gebruikt veranderen zelf niet, alleen de uiteindelijke (laatste) datafile veranderd.

Als de stream aan het eind is verwijderen alle jobs zichzelf en wordt de Superbasic functie EOF gezet. Als dan toch data wordt opgevraagd volgt een End Of File foutmelding.

EEN UITGEBREID VOORBEELD.

Stel we hebben de volgende jobs (filters) op disk die als volgt werken:

UC (is hiervoor al beschreven);

LNO voegt regelnummers toe aan de stream;

PAGE splitst de stream in pagina's en voegt een titel(commandstring) toe en kan het naar meerdere output files sturen;

CCT plakt meerdere inputfiles samen tot één output file.

Verder hebben we een file "mdvl_test" die we aan de volgende procesline willen onderwerpen:

XINPUT

Deze maand het laatste deel van de assembly listing van het XINPUT commando. Alles bij elkaar is het nog een dikke 500 bytes lang geworden. Zo, iedereen al klaar met intikken? Ja? Dan nu nog een tip: men kan het hele programma ook bij mij krijgen.

Albert Stam



```

232 NEXT_4
233 * now check if it was a printable char. if not: it is a ENTER command
234 * check for not printable: D1.B < 32 or D1.B >192
235 * <32: bit 7,6 & 5 unset; >=192: bit 7 & 6 set (=192 is already tested)
236         MOVE.W    D1, -(A7)          first put it somewhere safer
237         MOVE.B     D1, D2             then copy for calculate
238         ANDI.B      #%11100000, D2    is it <32?
239         BEQ.S       TRUE              ..yes, it is <32
240         ANDI.B      #%11000000, D1    is it >=192 then?
241         CMPI.B      #192, D1          ...
242         BEQ.S       TRUE              ..yes, it is >=192
243         MOVE.W      (A7)+, D1
244         BRA.S       NEXT_5            no, it was a printable char.
245 * it was an ENTER command so end this session
246 TRUE     MOVE.W      (A7)+, D5        restore
247         MOVEQ       #0, D0           end input session
248         RTS
249
250 * it was a printable char.; add it.
251 * insert a character(D1) in string at position given by D4
252 NEXT_5   CLR.L       D3
253         MOVE.W      (A1), D3          Get length of inputstring
254         ADDQ.W       #1, D3           update length
255         CMPI.W       #MAXL, D3        too long?
256         BHI         ERR_OV           ...yes
257         MOVE.W      D3, (A1)+
258         ADDA.L       D3, A1           find end of string
259         MOVEA.L      A1, A2           copy
260         SUBQ.L       #1, A1           predecrement
261         SUB.W        D4, D3           (D3-D4) nr. of chars. after insert
262         BEQ.S       ZERO             if len=0: no shift
263         SUBQ.W       #1, D3           DBF checks -1 not 0
264 SHFT_NXT MOVE.B      -(A1), -(A2)    shift
265         DBF         D3, SHFT_NXT     repeat untill all chars shifted
266 ZERO     MOVE.B      D1, (A1)
267         ADDQ.W       #1, D4
268
269 EXIT_KEYTEST
270 * calculate new cursor position:
271 * y=int((x0+c-1)/line_len)  x=x0+(c-1)-(y*line_len)  y=y0+y
272 LEA       WINDOW_DIM, A1

```

```

273      MOVEM.W      (A1)+,D0-D3      D0=line_len; D2=x0; D3=y0
274      MOVE.W       D4,D1            copy; forget old value D1
275      SUBQ.W       #1,D1            (c-1)
276      ADD.W        D1,D2            (x0+c-1)
277      MOVE.W       D2,D1            copy
278      DIVU         D0,D1            y=int((x0+c-1)/line_len)
279      MULU         D1,D0            (y*line_len)
280      SUB.W        D0,D2            x=x0+(c-1)-(y*line_len)
281      * now D1 contains y value; D2 contains x value
282      ADD.W        D3,D1            y=y0+y
283      MOVE.W       D2,(A1)+         store it at X and Y
284      MOVE.W       D1,(A1)+
285
286      * read a key again
287      BSR          KEY_READ
288      TST.L        D0
289      BNE.S        CL_EXIT
290      LEA          INPUT,A1
291      BRA          TEST_KEY
292
293      * wipe old string from screen
294      CLSTR        LEA          X0(PC),A2      set cursor to X0,Y0
295      BSR.S        SET_CURS
296      TST.L        D0
297      BNE.S        FORCE_EXIT2          NC, NO or OR
298      LEA          INPUT(PC),A1
299      MOVE.W       (A1),D5              get length
300      MOVEQ        #-1,D3              timeout
301      CLR_LOOP     MOVEQ        #IO.SBYTE,D0  send a byte
302      MOVEQ        #SPACE,D1          byte to be send
303      TRAP         #3
304      TST.L        D0
305      BNE.S        FORCE_EXIT2          NC, NO, DF or OR
306      DBF          D5,CLR_LOOP
307      CL_EXIT     RTS                  done.
308
309      SET_CURS     MOVE.W       (A2)+,D1      column number
310      MOVE.W       (A2),D2              row number
311      MOVEQ        #SD.POS,D0
312      TRAP         #3
313      FORCE_EXIT2  RTS
314
315      KEY_READ     MOVEQ.L      #-1,D6
316      MOVE.W       D6,D3              infinite time_out
317      LEA          X0(PC),A2
318      BSR.S        SET_CURS            set cursor at Y0,X0
319      TST.L        D0
320      BNE.S        FORCE_EXIT2          NC, NO or OR
321      LEA          INPUT,A1
322      MOVE.W       (A1)+,D2            number of bytes to send
323      MOVEQ        #IO.SSTRG,D0       send the string
324      TRAP         #3
325      TST.L        D0
326      BNE.S        FORCE_EXIT2          NC, NO or DF

```


327		MOVE.W	D6, D3	timeout
328		MOVEQ	#1, D5	do this twice
329	LOOP	MOVEQ	#IO.SBYTE, D0	send a byte
330		MOVEQ	#SPACE, D1	byte to be send
331		TRAP	#3	
332		TST.L	D0	
333		BNE.S	FORCE_EXIT2	NC, NO, DF or OR
334		DBF	D5, LOOP	
335		MOVE.W	#FLASH, D1	
336		MOVEQ	#SD.SETMD, D0	set over mode
337		TRAP	#3	
338		TST.L	D0	
339		BNE.S	FORCE_EXIT2	NC or NO
340		CLR.B	D5	flag: set if cursor visible
341	GET_KEY	LEA	X(PC), A2	
342		BSR.S	SET_CURS	set cursor at Y,X
343		TST.L	D0	
344		BNE.S	FORCE_EXIT2	NC, NO or OR
345		MOVEQ	#CURSOR, D1	
346		MOVEQ	#IO.SBYTE, D0	
347		TRAP	#3	print cursor
348		TST.L	D0	
349		BNE.S	FORCE_EXIT2	NC, NO, DF or OR
350		MOVEQ	#TIME_OUT, D3	
351		MOVEQ	#IO.FBYTE, D0	
352		TRAP	#3	
353		EOR.B	D6, D5	curs.visible: set/unvisible: unset
354		ADDQ.L	#1, D0	
355		BEQ.S	GET_KEY	NC
356		SUBQ.L	#1, D0	
357		BNE.S	FORCE_EXIT3	NO, EF
358		MOVE.W	D6, D3	time_out
359		MOVE.B	D1, D6	save fetched byte
360		TST.B	D5	D5 set: wipe cursor
361		BEQ.S	JUMP1	...no
362		LEA	X(PC), A2	
363		BSR	SET_CURS	
364		TST.L	D0	
365		BNE.S	FORCE_EXIT3	NC, NO or OR
366		MOVEQ	#IO.SBYTE, D0	
367		MOVEQ	#CURSOR, D1	
368		TRAP	#3	
369		TST.L	D0	
370		BNE.S	FORCE_EXIT3	
371	JUMP1	CLR.W	D1	set over mode to 0
372		MOVEQ	#SD.SETMD, D0	
373		TRAP	#3	
374		TST.L	D0	NC or NO
375		BNE.S	FORCE_EXIT3	
376		CLR.L	D1	
377		MOVE.B	D6, D1	fetch saved byte
378	FORCE_EXIT3	RTS		
379				
380	CHANID	DS.L	1	

381	VAR	DS.L	1	
382	WINDOW_DIM	DS.W	1	width of window
383	WINDOW_DIM2	DS.W	1	height of window
384	X0	DS.W	1	string pos. (char. coord.)
385	Y0	DS.W	1	
386	X	DS.W	1	cursor pos. (char. coord.)
387	Y	DS.W	1	
388	INFUT	DS.W	1	length
389		DS.B	MAXL	space for input\$
390		DS.W	1	space for entercommand%
391				

TOOLKIT II EN ASSEMBLY

Voor de assembly-programmeurs de volgende informatie. De Toolkit code verbetert twee TRAP call bugs van QDOS en voegt ook twee nieuwe TRAP routines toe. Het gaat om de volgende:

In de microdriver code, het FS (Filling System) loopt de volgende subroutine call in QDOS versie 1.03 en eerdere vast:

TRAP #2

D0=1; D3=3 := IO.OPEN; open (new), overwrite file

De FS.FLUSH (TRAP #3, D0=\$41) is ook verbeterd: het is nu een complete flush, d.w.z. inclusief de file header info.

De nieuwe traps zijn:

TRAP #3: FS.RENAME

D0=\$4A, A0=channel id, D3.W=timeout

A1=pointer naar de nieuwe naam.

Deze trap renamed een geopende file

TRAP #3: FS.TRUNCATE

D0=\$4B, A0=channel id, D3.W=timeout

Deze trap breekt de geopende file af vanaf de huidige byte positie.

Albert Stam

TIPS & TRUCS

* 1 * K A R A K T E R C O D E S

Voor het testen van b.v. een ingedrukte toets is een vergelijking als `INKEY$='A'` makkelijk te bedenken, maar als het gaat om toetsen als F1, cursor omlaag of CTRL-W is dit moeilijker. Nu zal de karakter-code toegepast moeten worden, maar de karakter-codes kennen de meeste mensen niet van buiten (enkele karakter-codes en mensen daargelaten).

Dan is het mogelijk om de handleiding van de QL uit de kast te pakken (als die nog niet op tafel lag), pagina 5 van het hoofdstuk Concepts op te zoeken en na enig speuren in de lijst per ongeluk de hexadecimale waarde voor een decimale aan de zien. Sneller gaat het echter op de volgende manier:

- 1) typ het commando `'PRINT CODE(INKEY$(-1))'` in
- 2) druk op de betreffende toets

De code wordt zo netjes door de QL zelf gegeven.

* 2 * C T R L - C

Normaal kun je met de toetsencombinatie CTRL-C een cursor in een ander 'window' activeren, zodat die gaat knipperen en de tekst die getikt wordt naar dit 'window' gestuurd wordt. Als je alleen in basic (zonder toolkit II) werkt merk je van dit soort zaken niets, maar zodra je programma's als Assembler Workbench gebruikt is het noodzakelijk veel 'tussen programma's heen en weer te schakelen'.

Echt onhandig is de combinatie CTRL-C niet, maar ikzelf geef toch de voorkeur aan één enkele toets, zoals ESC.

In de systeemvariabelen van de QL zit een variabele waarin de code van de betreffende toets (of toetsencombinatie) opgeslagen is, en wel op adres 163986 (28092 hexadecimaal) en 163987. Deze variabele is in de literatuur bekend als SV_CQCH, wat staat voor System Variable keyboard Change Queue CHaracter code, ofwel systeem-variabele toetsenbord-buffer-wissel-karakter-code. Door hierin een andere waarde te plaatsen is het mogelijk deze toets zelf te kiezen.

Het herdefiniëren gaat als volgt:

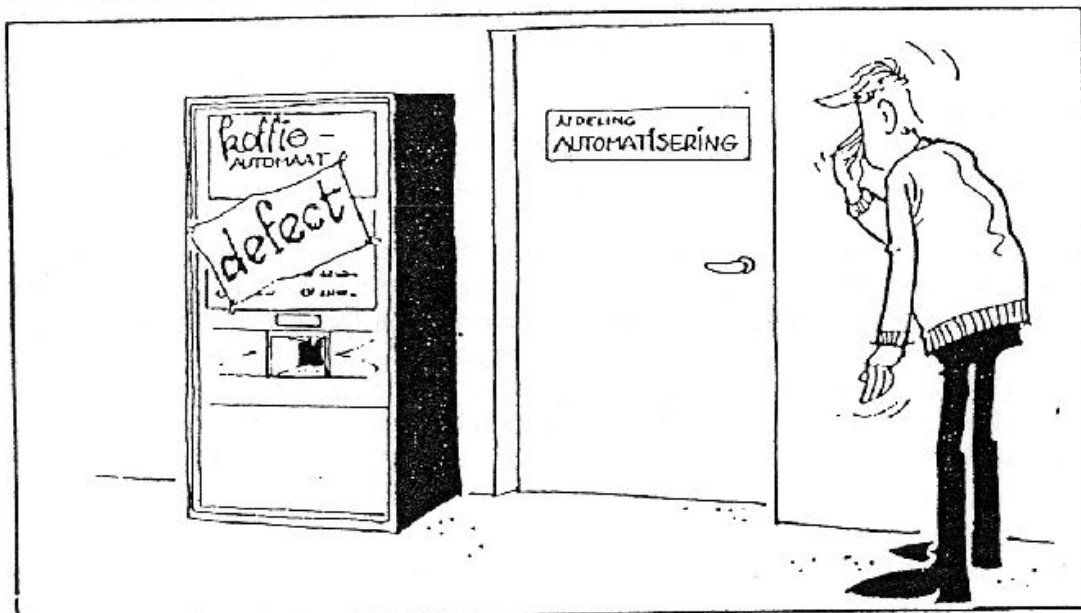
- 1) zoek een toets (of toetsencombinatie) op die niet wordt gebruikt
- 2) zorg dat de cursor in het besturingswindow (window #0) actief is
- 3a) typ 'POKE_W 163986,**'
** is de code van de toets, b.v. 27 voor ESC of
- 3b) typ 'POKE_W 163986, CODE(INKEY\$(-1))' en druk hierna de toets in
Als hierbij per ongeluk de ENTER toets repeteert of twee maal ingedrukt wordt, werkt ENTER niet meer en kunnen we dit dus schijnbaar ook niet meer ongedaan maken, maar ook voor dit probleem is er een oplossing:
Typ opnieuw 'POKE_W 163986,3' en beëindig deze opdracht met cursor omhoog of omlaag, de pijltjestoetsen rechts naast de spatiebalk dus.

Als alles goedgegaan is, is CTRL-C weer beschikbaar voor gewoon gebruik, en wordt de cursor geschakeld door de gekozen toets.

Martijn Plak
tel. 05910-30993

Jaloers op het mooie beepje in de alarmroutine van ToolkitII?
Zo kun je hem in SuperBasic maken!

```
10 BEEP 20480,7,14,1024,5,0,0,0
```



PF_Dump

PF_Dump print een lijst van (in een SuperBASIC programma) gedefiniëerde procedures en functies, inclusief hun parameters en de lijnnummers.

Merge PF_Dump in het SuperBASIC programma en roep het op door intikken van "PF_Dump". Na een tijdje wachten verschijnen de proc/func op het scherm.

Het programma maakt gebruik van de "name table", de "name list" en de "variable area".

De name table bestaat uit blokken van elk 8 bytes. De blokken definiëren het type van de naam, de lokatie in de name list en de lokatie in de variable area.

Het "format" van elk blok is"

word: definitie van het type van de naam.

Er zijn een groot aantal definities voor namen. Van belang zijn:

\$0400 : SuperBASIC procedures.

\$0501 t/m \$0503: SuperBASIC functies.

Het programma sorteert op de hoogste byte van het woord; dus op \$04 en \$05.

word: lokatie in de name list.

long word: Lokatie in de variable area.

De hoogste 2 bytes van het long word is het lijnnummer van de DEF statement.

In de name list staan (goed geraden, maar er is geen prijs te winnen) de namen in ASCII. Voor elke naam staat een byte, die de lengte van de naam aangeeft.

Maar waar in het geheugen staan de name table en de name list? Zoals bekend worden "SuperBASIC variables" relatief t.o.v. het adres in adresregister A6 in het geheugen opgeslagen.

In het programma is "basic" dat bewuste adres. Direct na de start is "entry" het adres van het begin ("base") van de name table en "name_ptr" de base van de name list.

Bij elke doorgang in de loop "REPeat name" worden voor de volgende naam entry en name_ptr opnieuw berekend. Als een SuperBASIC proc/func is gevonden wordt het lijnnummer opgevraagd en het lijnnummer en de tekst van die lijn op het scherm geprint.

```
( 10100 IF typ = 4 OR typ = 5: LIST PEEK_W(entry + 4)
```

Op de volgende manier kan het veel eenvoudiger. Zorg er voor dat b.v. de eerste proc/func begint op lijnnummer 100 en de volgende op 200 enz. "Procedure lp" doet dan hetzelfde als PF_Dump.

```

10000 DEFine PROCedure pf_dump
10010   LOCAL basic, pos%, name_ptr, name, length, entry, typ
10020   CLS#2
10030   basic = PEEK_L(163856) + 104
10040   pos% = 0: name_ptr = basic + PEEK_L(basic + 32)
10050   REPEAT name
10060     length = PEEK(name_ptr)
10070     IF length = 0: EXIT name
10080     entry = basic + PEEK_L(basic + 24) + pos% * 8
10090     typ = PEEK(entry)
10100     IF typ = 4 OR typ = 5: LIST PEEK_W(entry + 4)
10110     pos% = pos% + 1
10120     name_ptr = name_ptr + length + 1
10130   END REPEAT name
10140 END DEFine pf_dump
10150 :
10160 DEFine PROCedure update
10170   SAVE_O ProcFuncDump_bas
10180 END DEFine

```

TURBO VERSUS QLIBERATOR

De meest bekende SuperBASIC compilers voor de QL zijn wel de Qliberator en de Turbo compilers. Begonnen is het allemaal met SUPERCHARGE van Digital Precision, een SuperBASIC compiler voor de QL. Het resultaat hiervan was de SUPERCHARGE compiler, een compiler speciaal gemaakt voor de mensen die geen geheugen uitbreiding voor de QL hadden en was dan ook lang niet zo uitgebreid als de huidige versies van de compilers.



De SUPERCHARGE compiler kon ook niet de gehele SuperBASIC syntax aan. In 1986 kwam Digital Precision dan ook met de eerste versie van de veel betere TURBO compiler. Deze compiler werd vergezeld van een apart verkrijgbare Turbotoolkit die aan SuperBASIC ruim 100 extra commando's toevoegt !! Bovendien corrigeert de Turbotoolkit een aantal 'onvolkomenheden' in SuperBASIC.

De TURBO compiler is een goede compiler die de SuperBASIC vele malen versneld en er EXECutable files van maakt (dus multitasking !!). In de meeste gevallen maakt de TURBO compiler snellere code aan dan zijn grote concurrent, de QLIBERATOR.

De nieuwste versie van deze compiler is helemaal gemaakt om samen te werken met QRAM. Beide compilers geven de mogelijkheid om met getallen met tot 9 cijfers achter de komma te werken! De Qliberator compiler is zeer gebruikersvriendelijk en accepteert gemakkelijker vreemde SuperBASIC constructies dan de TURBO compiler. Van Qliberator is nu ook een budget versie uitgekomen voor de financieel minder gelukkigen onder ons. Deze versie heeft uiteraard niet de vele mogelijkheden van de uitgebreide versie, maar is ook zeker aan te bevelen.

(Bron databank IMPULS)

```

      I
    III
  IIIII
 IIIIIII
IIIIIIIIII
IIIIIIIIIIII
IIIIIIIIIIIIII
IIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIIIIIII

```

```

II
II
II
II
II
II

```

De redactie van

QLONE

wenst u

PRETTIGE KERSTDAGEN

&

GELUKKIG NIEUWJAAR

```

      I
    III
  IIIII
 IIIIIII
IIIIIIIIII
IIIIIIIIIIII
IIIIIIIIIIIIII
IIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIIIII
IIIIIIIIIIIIIIIIIIIIII

```

```

II
II
II
II
II
II

```


DRUKWERK

Afz. Ql. G.N.N.
Redaktieadres
Bosweg 27
9414 BD HOOGHALLEN

