clair impuls

JAARGANG : 8
KWARTAAL : 2

Verschijnt 4 maal per jaar.
Losse verkoopprijs *f* 7,-.

Je hebt onze ZX-microfair op 23 juni a.s. toch al in je agenda genoteerd:

23 juni 10.30/16.00 uur
Sinclair mini-beurs
Houten !!!

zie kaartje achterin!

sinclair impuls

EEN UITGAVE VAN DE

SINCLAIR GEBRUIKERS GROEP

VERSCHIJNT EENMAAL PER KWARTAAL

"Sinclair Impuls", HET blad voor en door de gebruikers van ALLE Sinclaircomputers - ZX80, ZX81, ZXSpectrum, QL en aanverwanten - wordt uitgegeven door de "HCC Sinclair Gebruikers Groep " (SGG).

IMPULSREDACTIE:

Ed Weijgers
Wilhelminalaan 42
2625 KH Delft

Kees Versluis
Copernicuslaan 25
2561 VA Den Haag

Marco Holmer
J P Coenstraat 61 bis
3531 EN Utrecht

Jack Raats (eindred)
Noorddonk 107
4651 ZD Steenbergen

IMPULSKOPIJ:

Voor OD naar Kees Versluis (OD-hulp: 070-3604185), alle andere kopij naar Ed Weijgers of Jack Raats

IMPULSABONNEMENTEN:

Alleen voor HCC-SGG-leden f 25.00 per kalenderjaar (voor anderen f 30.00), over te maken op onze SGG-postrekening, ovv 'Abonnement Impuls' en het jaar.

Adreswijzigingen schriftelijk aan het SGG-adres.

SGG-BESTUUR:

voorzitter:	Piet van Wees
secretaris:	Theo Molenaar
penningmeester:	George Burghgraef
epibratie:	Rob van Staalduinen
publiciteit:	Ed Weijgers
publiciteit en info:	Jack Raats
software en verkoop:	Robert Notenboom

SGG-ADRES EN REKENING:

HCC Sinclair GG
Postbus 76
2260 AB Leidschendam

Postrekening 5374525
tnv HCC Sinclair GG
te Bunnik

SGG-TELEFOONNUMMER:

Infotelefoon 01670-66845 ma en do, 20-22 uur

Gebruik dit telefoonnummer uitsluitend voor problemen bij hard- en software of algemene informatie over Sinclaircomputers en IMPULS-artikelen.

Dit redactionele stukje heeft onder meer de bedoeling om u op de hoogte te stellen van dingen die de redactie onbekend zijn.

Om te beginnen of onze "ZX-Microfair" van 23 juni in Houten een succes is geworden. Die moet namelijk nog gehouden worden op het moment dat dit geschreven wordt.

Verder of het de redactie nu gelukt is om iets af te halen van de opgelopen achterstand bij het verschijnen van IMPULS. De geprinte kopij kan op diezelfde "ZX-Microfair" aan onze epibrator overhandigd, of vlak daarna gepost worden. Maar of dat gebeurde? We hopen dat het daarna niet weer een volle maand duurt voordat IMPULS in de bussen valt, zoals bij de twee voorgaande nummers.

Dit waren dingen die u wel weet en wij niet. Wij weten echter de antwoorden op vragen die u nog gaat stellen. Waarom deze IMPULS wat dunner is uitgevallen dan de vorige bijvoorbeeld. Als u hem doorneemt zult u maar heel weinig verschillende namen tegenkomen onder de artikelen. Het moge u duidelijk zijn dat we nog verder achter raken als we op meer kopij gaan zitten wachten. We zullen toch alle nieuwe bijdragen met gepaste blijdschap ontvangen. Wat er in de volgende IMPULS komt vernemen we graag van u.
We zullen ook de hand in eigen boezem steken bij opmerkingen als "wel OD-artikelen, maar geen voor mijn ...". Moeten we ook maar niet met twee OD-ers in de redactie gaan zitten. Wie durft daar op te merken dat het nog best meevalt in dit nummer?

Dat er overigens namen onder artikelen staan is niet voor niets. Schroom niet om schrijvers te benaderen als u ergens problemen mee hebt of meer wilt weten. In Houten zijn ze er doorgaans ook. Zo kunt u hun bijvoorbeeld gerust om een "source" van MC vragen.

De door ons geintroduceerde manier om files voor de Spectrum te versturen als P(akket)-files lijkt een succes te worden in .... Engeland. Daarvandaan kregen we zelfs een programma om files in en uit te pakken bij de Spectrum+3. Zodra het helemaal gereed is voor publicatie verschijnt het in IMPULS.
Van de serie P-omzetters staat wel "md><p" in deze IMPULS, maar het laatste programma, "bd><p", houdt u nog even van ons tegoed.

U kunt verderop ook lezen dat alle XCOM-programma's gereed zijn. De IMPULSOFT-cassette en de DUCDISK komen eraan. Als u daar niet op wilt wachten, zoals hierboven gezegd: neem gerust contact op.

Wij vinden eigenlijk dat naast alle XCOM-versies ook programma's voor FIDO op die cassette en disk moeten staan. Dus ook voor de diverse opslagmedia en RS232-poorten. Rik Koevoets heeft ons al toegestaan om al zijn FIDOTERM-versies daarvoor te gebruiken.

Tenslotte moeten wij u ook bekennen dat de door ons aangekondigde universele tekstomzetter "dstuvw" nog niet helemaal klaar is.

- rEd -

De SP slaat zijn variabelen op in een geheugengedeelte dat loopt van VARS (sysvar op 23627&28) t/m E_LINE-1 (sysvar op 23641&42), waar zich de eindmarkeerder met de waarde 128 (80 hex) bevindt. ZX-BASIC kent 6 verschillende soorten variabelen: typen 2 t/m 7. De opbouw bij deze soorten vindt u hieronder. Hierbij geldt voor iedere soort dat het type blijkt uit de eerste byte (L), en wel uit de hoogste 3 bits daarvan die het typenummer (T) vormen. De resterende 5 bits vormen de met 96 verlaagde CODE van de naam (N$) van de variabele (kleine letter, dus bereik van 1 t/m 26). Bij type 5 volgen er nog meer naamletters (klein) of -cijfers, met ongewijzigde CODE (48 t/m 57, 97 t/m 122), op de laatste na, waarbij 128 opgeteld is (bit 7=1) om zo het einde aan te geven.

```
 128 64 32 16  8  4  2  1   LET L=PEEK A                [byte]
 --------- --------------   LET T=INT (L/32)            [type]
  b7 b6 b5 b4 b3 b2 b1 b0   LET N$=CHR$ (L-T*32+96)     [letter]
```

2 STRING — LEN: aantal van de volgende bytes

```
+-+-+-+-+-+- - - -+-+-+
|L|LEN|      VAL$     |   VAL$: stringwaarde (tekens)
+-+---+---- - - - ----+
```

3 NUMERIEK MET NAAM VAN 1 LETTER (TOTAAL 6 BYTES LANG)

```
+-+-+-+-+-+-+
|L|  VAL    |              VAL: numerieke waarde (FPR / INT)
+-+---------+
```

4 NUMERIEK MET N DIMENSIES — LEN=1+N*2+(#D1*#D2*...*#DN)*5

```
+-+-+-+-+-+-+-+-+- - -+-+-+-+-+-+-+-+- - - -+-+-+-+-+-+
|L|LEN|N|#D1|#D2|      |#DN|   VAL    |      |   VAL   |
+-+---+-+---+---+- - -+---+----------+- - - -+---------+
```

5 NUMERIEK MET NAAM VAN MEERDERE LETTERS (LAATSTE: +128)

```
+-+-+-+- - -+-+-+-+-+-+-+-+
|L      NAAM      |   VAL   |
+------- - - -----+---------+
```

6 STRING MET N DIMENSIES — LEN=1+N*2+(#D1*#D2*...*#DN)

```
+-+-+-+-+-+-+-+-+-+- - -+-+-+-+- -+-+-+- - - -+-+-+- - -+-+
|L|LEN|N|#D1|#D2|      |#DN|  VAL$    |      |  VAL$     |
+-+---+-+---+---+- - -+---+----------+- - - -+-----------+
```

7 NUMERIEK MET LUSGEGEVENS (TOTAAL 19 BYTES LANG)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|   VAL    |    TO    |   STEP   |RNR|O|     O: opdrachtnummer
+-+----------+----------+----------+---+-+
```

DE LENGTE VAN EEN VARIABELE IN HET GEHEUGEN

- De types 2, 4 en 6 bevatten in LEN op de tweede (L) en derde (H) byte: het aantal van de bytes die nog achter LEN volgen. De totale lengte bij zo'n variabele bedraagt derhalve LEN+3.

- De types 3 en 7 bezitten een vaste lengte: 6 resp 19 bytes.

- Bij type 5 is de lengte gelijk aan het aantal der tekens van de naam vermeerderd met 5 voor VAL (de waarde in FPR / INT).

Tijdens de executie moet de SP elke voorkomende variabele vanaf het begin van het variabelengeheugen opzoeken. Als er veel variabelen zijn gaat dit dus traag, vooral bij variabelen achteraan.

Extra vertraging levert type 5 zonder vaste lengte of LEN-bytes. Bij die soort moeten elke keer alle naamletters gelezen en zonodig met een naam vergeleken worden, tot de laatste herkend wordt en over de 5 bytes van VAL naar een volgende variabele gesprongen kan worden (als die naam niet van een gezochte variabele is). Hoe langer zo'n naam, des te langer dit duurt, weet dat dus wel!

HET CREEREN VAN VARIABELEN - HET TOEKENNEN VAN WAARDEN

Dit geschiedt door opdrachten met: LET, FOR, INPUT, READ en DIM. Hierbij wordt eerst gekeken of de betreffende variabele bestaat.

Een reeds bestaande numerieke of geDIMde variabele krijgt meteen een nieuwe waarde. Opgelet: dit gebeurt tevens bij een ongeDIMde stringvariabele met een substringvormer ('slicer'), bijvoorbeeld LET A$()=.. (of met (n), (n TO), (TO m) of (n TO m) erachter).

Een voorkomende ongeDIMde stringvariabele zonder substringvormer wordt eerst weggehaald, waarbij alle volgende variabelen moeten worden teruggeschoven. Dit gebeurt ook wanneer een variabele van type 2 of 6 geDIMd wordt, of van type 3 door FOR type 7 wordt.

Een niet bestaande of een verwijderde variabele wordt achteraan (opnieuw) gecreeerd met een nieuwe waarde (evt waarden bij DIM).

GeDIMde en ongeDIMde numerieke variabelen met dezelfde naam kunnen tegelijkertijd voorkomen, bij stringvariabelen kan dat niet. Een lusvariabele van type 7 kan nooit meer van type 3 worden. In BASIC kunnen variabelen slechts verwijderd worden met CLEAR, RUN, NEW en USR 0, tenzij u beschikt over MC voor een opdracht als CLEAR LINE r bij OD in ROM 2.2(2) of IC6116 geinstalleerd.

Verderop in deze IMPULS staan nog meer artikeltjes over variabelen, waarin gebruik gemaakt wordt van de hier beschreven opbouw.

- EdW -

-- WIE DOET "ORIGINATE" EN WIE "ANSWER"? --

Deze vraag kan gesteld worden bij de "user to user"-communicatie aangezien daarover geen algemene afspraak schijnt te bestaan. Dit probleem deed zich niet voor bij degenen die zich tot nu toe slechts van 1200/75 en 75/1200 baud bedienden. Nu er steeds meer gecommuniceerd wordt met 1200/1200 en zelfs 2400/2400 baud - de daarvoor geschikte modems worden goedkoper - zou het handig zijn als er wel zo'n afspraak was. Die mogen wij niet zomaar eventjes voor iedereen maken door een munt op te gooien, dat kan slechts als die ergens logisch bij aansluit. Dat is gelukkig mogelijk. DATABANKEN moeten we doorgaans zelf opbellen als we er software uit willen betrekken. Daarbij geldt dus:

- ORIGINATE (MAIN, DIAL) <-> DOWNLOADEN -
- ANSWER (BACK) <-> UPLOADEN -

Houden we dit ook aan bij de communicatie tussen de gebruikers onderling, dan is de vraag "answer ik of answer jij?" overbodig. Nog een voordeel is dat deze afspraak geprogrammeerd kan worden in de communicatiesoftware. Hiervan geven we u een voorbeeld. Bij al onze XCOM-programma's krijgen bezitters van hayes-modems voor en na het up- en downloaden de gelegenheid om hun modem te commanderen (mits RETURN uit regel 175 verwijderd is; andere gebruikers behoeven geen overbodige hayes-vragen te beantwoorden). Hierbij moeten alle hayes-commando's telkens volledig ingetoetst worden (onze programma's moeten immers algemeen bruikbaar zijn). Het is echter mogelijk om de voor uw modem vaste hayes-tekens in een gewijzigde regel 50 op te nemen, zodat u alleen nog maar het teken dat variabel is voor de communicatiestand hoeft te geven.

VOORBEELD (modem: Best 2400 plus)

```
50 INPUT "ATB";N: LPRINT "ATB";STR$ N;"DA"(VAL N$):
   PAUSE PI AND N$="2": RANDOMIZE USR C:
   LPRINT "+++";: PAUSE NOT PI": LPRINT "ATH": RUN
```

"ATB"+STR$ N stelt de "Bell/CCITT mode" van de modem in.

N=0: V21 (300/300), V22 (1200/1200) of V22bis (2400/2400)
(welke hiervan is afhankelijk van de ingestelde baudrate)
N=1: Bell (nvt) N=2: V23 (75/1200) N=3: V23 (1200/75)

Hierna volgt automatisch een van de "on line"-tekens:

"D" bij "1 LOAD DOWN" (N$="1"), of
"A" bij "2 LOAD UP " (N$="2"), altijd afgesloten met CR.

Alleen bij downloaden wordt nu, voordat de MC aangeroepen wordt, gewacht op een toetsaanslag. Die mag pas gegeven worden zodra de "carrier detect"-lamp brandt (anders gaat de modem "off line").

"+++" terug naar de commandostand van de modem, mits er binnen een seconde niets anders ontvangen wordt (dus ook geen CHR$ 13). "ATH"+CHR$ 13 ontkoppelt de modem daarna ("Hook off"). - EdW -

Alle versies van ons "user to user"-communicatieprogramma "XCOM" kregen .05 als toevoeging wegens deze twee nieuwe mogelijkheden:

- DIRECT FILE-PAKKETTEN IN- EN UITPAKKEN

Met optie 7 kunt u nu P-files opbouwen uit N-files op CR. Vooraf met optie 5 geLOADe P-files kunt u op deze manier ook verlengen. Met optie 8 kunt u alle N-files uit een P-file naar CR SAVEn. Bij de speciale OD-versies werken deze opties ook via ODrives.

- HET XMODEM-PROTOCOL IS UITGEBREID MET CRC-CHECK

Communicatie onder XMODEM-protocol geschiedt in datablokjes van 128 bytes. Voor elk goed ontvangen blokje verstuurt de download-kant een ack(nowledge)-teken, waarvoor beide kanten een ">" op hun scherm zien. Anders is dat een nak-teken, zichtbaar als "<", waarop de uploadkant het betreffende blokje opnieuw verstuurt. Goed of fout wordt hierbij onder andere gedetecteerd aan de hand van een met elk datablokje meegezonden testbyte (checksum). CRC is een verbeterd systeem om fouten op te sporen met 2 testbytes. Het is zo geavanceerd dat de kans op fouten vrijwel nihil is.

De downloadkant bepaalt welk systeem er gehanteerd zal worden. Kent die geen CRC, dan verstuurt die met tussenpozen een aantal "naks" tot de andere kant daarop reageert door te gaan uploaden. Als die wel CRC kent, dan verstuurt die eerst drie maal een "C" en gaat pas wanneer daarop niet werd gereageerd over op "naks".

De uploadkant start het uploaden zodra er een "C" of een "nak" ontvangen wordt: bij een "C" met CRC, bij een "nak" zonder CRC. Kent deze echter geen CRC, dan reageert deze alleen op een "nak" (dat is pas het vierde teken als de downloadkant XCOM gebruikt).

—

TWEE CONSEQUENTIES VAN DE UITBREIDING MET CRC

Een hayes-modem geeft bij "carrier detect" de melding "CONNECT". Bij het uploaden zal een "C" hieruit ten onrechte worden opgevat als een vraag van de downloadkant om te gaan uploaden met CRC.

REMEDIE: Stel de Bell/CCITT-stand met "ATV0B" ipv "ATB" in, waardoor de meldingen in cijfers ipv woorden gegeven worden.

Het programma FIDOTERM controleert voor het uploaden niet of een ontvangen byte een "nak" of wat anders is (dit is dus een fout). Als XCOM een "C" verstuurt zal FIDOTERM dus gaan uploaden, maar zonder CRC. Omdat XCOM dan wel CRC verwacht gaat dit dus fout.

REMEDIE: Bij FIDOTERM mag SYMBOL SHIFT U om met uploaden te beginnen pas gegeven worden wanneer er "CCC" is verschenen.

—

SGG-infotel 01670-66845

Een probleem bij het werken met de QL/ST-emulator is dat sommige programma's direct in de QL ingrijpen, buiten de erkende TRAPs (ingangen) van het 'operating system'.

Met name de compiler QLIBERATOR is een boosdoener. Alle versies uitgebracht voor nummer 3.22A veroorzaken steevast een systeem-crash op de QL/ST. Deze fout zit in de extensie module. Gelukkig is in de nieuwe versies het probleem opgelost. Met de vernieuwde extensie module werkt alles weer prima!

Bij de oude versies van QLIBERATOR wordt bij het compileren die extensie in het programma opgenomen. Omdat QLIBERATOR altijd eerst in het programma laat zoeken naar die extensies en pas daarna in de rest van het QL-geheugen, blijven deze programma's de oude, meegecompileerde extensie gebruiken.

Wat nu te doen met alle programma's die met een oude QLIBERATOR-versie zijn gecompileerd?
De enige oplossing leek tot voor kort opnieuw compileren met de goede versie. En omdat dat alleen kan als je de beschikking hebt over het bronprogramma in SUPER-BASIC, was dat in de praktijk alleen bij zelfgemaakte programma's mogelijk.

HOOP.
Echter er is nog hoop. Het blijkt mogelijk te zijn om het gecom-pileerde programma zo te veranderen, dat het progamma van mening is dat er helemaal geen meegecompileerde extensies aanwezig zijn.
Het programma gaat dan op zoek naar de extensies in het geheugen van de QL en wanneer daar de juiste versie van de runtime module resident is geladen werkt de zaak gewoon verder.
Het enige nadeel is dat het programma steeds die extra kilobytes van de oude runtime module onbenut met zich meedraagt de QL in.

PROGRAMMA.
Het onderstaande programma in SUPER-BASIC klaart de klus. Alle met QLIBERATOR gecompileerde programma's kunnen hiermee worden aangepast.
De QL/ST schijnt ook moeite te hebben met de codes die door SUPERCHARGE en TURBO geproduceerd worden, maar die worden door mijn programma helaas niet aangepast.

Het is een recht-toe-recht-aan-programma zonder vriendelijke bediening, die mag U er gerust zelf bijprogrammeren.
Het programma maakt gebruik van de file routines van TOOLKIT II, maar de meeste QL-gebruikers zullen die wel hebben.

Voor het runnen van het aangepaste programma moet u natuurlijk een goede runtime module van QLIBERATOR laden!

Tot slot nog een kleine waarschuwing.
WERK ALTIJD MET EEN KOPIE VAN HET ORIGINELE PROGRAMMA!!
Je weet nooit wat er kan gebeuren.

```
10  REMark QLIBERATOR AANPASSINGEN!
100 REMark TK2_EXT : REMark toolkit II
110 REPeat loop
120 INPUT "Geef Programmanaam (incl devicenaam) in."\
    "'ENTER'=einde: ";file$
130 IF file$='' :STOP
140 lengte=FLEN(\file$)
150 OPEN #3;file$
160 gevonden=zoek ('Runtimes',0,lengte,1)
170 terugtellen=zoek(CHR$(0)&'~Q'&CHR$(12),gevonden,0,-1)
180 IF NOT terugtellen
190    PRINT 'Geen runtime module ingebouwd'
200    ELSE :
210    BPUT#3\terugtellen-1,255
220    PRINT file$&' aangepast!'
230 END IF
240 einde
250 END REPeat loop
260 DEFine PROCedure einde
270   CLOSE #3
280 END DEFine einde
290 DEFine FuNction zoek (x$,begin,eind,stap)
300    LOCal i,j,vlag
310    FOR i=begin TO eind STEP stap
320      BGET#3\i,a
330      IF a <> CODE(x$(1)): NEXT i
340      vlag=i
350      FOR j=2 TO LEN(x$)
360         i=i+stap
370         BGET#3\i,a
380         IF a <> CODE(x$(j)):vlag=0: EXIT j
390      END FOR j
400      IF vlag : RETurn vlag
410    END FOR i
420 RETurn 0
430 END DEFine
```

Ik hoop dat de QL-gebruikers iets aan dit programma hebben. Als er andere oplossingen zijn voor dit probleem hoor ik het graag.

Marco Holmer - J P Coenstraat 61 bis - 3531 EN Utrecht

Relocatable MC-routines kunnen op elk adres geexecuteerd worden, dat is bekend. MC-programmeurs weten ook wel, dat het de nodige moeite kost om routines relocatable te maken. Met mijn programma "reloc$" hieronder kunt u zich die moeite voortaan besparen.

Reloc$ produceert een aangepast CODEblok. Dit wordt later in het beeldschermgeheugen geLOAD en kan daarna verplaatst worden naar ieder gewenst adres. Wegens de beperkte lengte van dit geheugengebied (6912 bytes) kan de originele CODE niet te lang zijn. De meeste programmeurs zullen er toch wel mee uit de voeten kunnen.

Mensen die mijn pijlroutine hebben gekocht, hebben al kennis gemaakt met reloc$. Zij kunnen namelijk de pijlroutine en een speciale schermprintroutine op elk adres in het geheugen gebruiken.

—

DE BASIC "reloc$" LINE 80

```
  10 INPUT "CODE in drive",d: CAT d:
     INPUT "Naam CODE 1",a$'"Naam CODE 2",b$: CLS
  20 INPUT "ORG CODE 1",O'"LEN ",L'
           "LOWBYTE / """" CONTINUE ";L$:
  30 LOAD *"m";d;a$CODE 32768,L AND L<=3072:
     LOAD *"m";d;b$CODE 36096,L
  40 INPUT " SAVE naar drive",d: CAT d:
     INPUT " SAVE naam",a$: CLS
  50 RANDOMIZE L: POKE 16384,PEEK 23670: POKE 16385,PEEK 23671:
     RANDOMIZE O: POKE 32672,PEEK 23670: POKE 32673,PEEK 23671
  60 LET la=USR 32622:
     IF LEN L$ THEN POKE 16390,14: POKE 16391,VAL L$
  70 SAVE *"m";d;a$CODE 16384,la-16384: CLS : STOP : RUN
> 80 CLEAR 32621: LOAD *"m";1;"Creloc$"CODE : RUN
```

DE CODE "Creloc$"CODE 32622,146 in HEXDATAregels:

```
   1 "21 CD 7F 11.02 40 01 33.00 ED B0 ED.4B 00 40 EB"   1524
   2 "09 EB 7E E5.F5 7C C6 0D.67 F1 BE E1.28 25 C5 01"   3737
   3 "00 80 A7 ED.42 EB 73 23.72 23 EB 09.D5 5F 23 56"   5542
   4 "EB 01 50 C3.A7 ED 42 EB.72 2B 73 23.D1 C1 0B 78"   7598
   5 "B1 28 06 0B.23 78 B1 20.C9 EB 70 23.70 23 E5 21"   9188
   6 "00 80 11 35.40 ED 4B 00.40 ED B0 C1.C9 00 00 ED"  10870
   7 "4B 76 5C 00.00 00 00 00.00 00 00 00.00 21 35 40"  11305
   8 "C5 50 59 ED.4B 00 40 ED.B0 C1 5E 23.56 23 7A B3"  13204
   9 "C8 EB 09 D5.E5 5E 23 56.EB 09 EB E1.73 23 72 E1"  15498
  10 "18 E8"                                            15754
```

Reloc$ heeft twee CODE-blokken nodig van dezelfde sourcecode:

- CODE 1, geassembleerd op het orginele ORG-adres,
- CODE 2, geassembleerd op het ORG-adres + 257.

Ik ga ervan uit dat beide CODE-blokken op dezelfde disk staan. U kunt dat natuurlijk ook wijzigen in de BASIC-regels hierboven.

In regel 20 worden drie belangrijke vragen gesteld:

1. Het ORG-adres van CODE 1. Kies voor het assembleren een mooi adres, bijvoorbeeld 50000.

2. De LENgte van het CODE-blok. Het spreekt vanzelf dat beide CODE-blokken dezelfde lengte hebben. De maximale lengte is in regel 30 vastgelegd op 3072 bytes. Dit is om te voorkomen dat de aangepaste CODE niet meer op het scherm past.

3. De waarde van de LOWBYTE van het keuzeadres. Deze optie is van belang voor routines die werken met interruptmode 2. Executie van deze routines is slechts mogelijk op 1 adres binnen een 256 bytemap. Voert u hier een waarde in voor de lowbyte, dan wordt naderhand de lowbyte van het keuzeadres aan de ingevoerde waarde gelijk gemaakt.
   Gebruik (ivm met de eenvoudige werking van reloc$) voor het LoaDen van het I-register niet LD A,x en LD I,A maar:

```
LD HL,x*256
LD A,H
LD I,A
```

   Daarbij is x*256 het adres waar het startadres van uw 'interrupt serving routine' staat.

—

In regel 40 wordt nog naar de naam van het aangepaste CODE-blok gevraagd. Ik geef zo'n blok altijd een naam die eindigt op 'RL'.

Op het scherm verschijnt nu troep. Dit zijn de relocate-routine, een tabel en uw (aangepaste) CODE-blok. Dit geheel wordt meteen geSAVEd (regel 70). Uw routine is nu 'relocatable' geworden. De genoemde tabel bevat adressen die bij de verplaatsing aangepast moeten worden.

—

Het gebruik is simpel. U geeft achtereenvolgens:

```
LOAD ... "naam"CODE:
RANDOMIZE xxx:
RANDOMIZE USR 16386
```

Daarbij is xxx het adres waar de CODE terecht moet komen (let op de kwestie van de lowbyte). Zoals u zult bemerken, wordt de aangepaste CODE weer in het beeldschermgeheugen geLOAD.
Uw routine wordt dan naar adres xxx verplaatst en (mbv de tabel) geschikt gemaakt voor dat adres. Dat is eigenlijk alles. Ik hoop dat het gebruik van reloc$ duidelijk is. Problemen? Bel gerust: 053-339541. Veel plezier ermee!

—

Victor Vogelpoel - Hengelosestraat 104-21 - 7514 AK Enschede

Hier is dan het in TRANS 22 in IMPULS 81 beloofde conversieprogramma voor P-files bij MD. Op de volgende bladzijde treft u het BASIC-gedeelte "md><p" aan en hieronder vindt u de HEXDATAREGELS voor de bijbehorende MC "Cmd><p", die u daaruit kunt verkrijgen door gebruik te maken van het programma "hexloader" in TRANS 22. Daar staat trouwens ook de gebruiksaanwijzing, die voor al deze conversieprogramma's eensluidend is.

SAVE de beide delen op een en dezelfde cartridge. Wanneer u dan later de BASIC-file LOADt, dan wordt de CODE-file er automatisch bijgeLOAD, in welke drive de cartridge zich ook bevindt.

—

DE MC "Cmd><p"CODE 65E3,345 IN HEXDATAREGELS - KeesV -

```
  1  "C3 F4 FD C3.6E FE C3 D3.FE C3 31 FF.CD 81 1C E7"   3003
  2  "CD 8C 1C CD.F1 2B ED 43.DA 5C ED 53.DC 5C CD 99"   5469
  3  "1E ED 43 D6.5C D9 E5 CF.22 22 1C 5C.AF CF 21 E1"   7590
  4  "D9 DD CB 18.46 28 04 3E.11 18 08 DD.CB 43 56 20"   9089
  5  "11 3E 16 32.39 FE CF 2C.21 38 FE 22.ED 5C CF 32"  10765
  6  "E7 00 DD CB.04 BE DD 22.51 5C ED 5B.76 5C 13 DD"  12820
  7  "E5 E1 01 47.00 09 01 0A.00 ED B0 D5.DD 6E 53 DD"  14627
  8  "66 54 11 09.00 19 E5 C1.E1 CD D4 15.77 23 0B 78"  16234
  9  "B1 20 F6 CF.23 C9 CD 81.1C E7 CD 8C.1C CD F1 2B"  18587
 10  "D5 CD 99 1E.AF B1 E1 28.4C ED 43 D6.5C 11 92 5C"  20746
 11  "01 09 00 ED.B0 21 0A 00.22 DA 5C 2A.76 5C 23 22"  21877
 12  "DC 5C D9 E5.CF 22 22 1C.5C DD 36 43.04 E1 D9 DD"  24039
 13  "22 51 5C 21.92 5C 06 09.7E D7 23 10.FB FD 4E 59"  25595
 14  "FD 46 5A 2A.76 5C 11 14.00 19 7E D7.23 0B 78 B1"  27006
 15  "20 F8 CF 23.C9 FD CB 47.86 18 10 FD.CB 47 C6 DD"  29376
 16  "2A 76 5C AF.37 11 11 00.CD 56 05 3E.02 CD 01 16"  30480
 17  "11 C0 09 2A.76 5C E5 E5.7E CB BF CD.0A 0C 06 0A"  32171
 18  "23 7E D7 10.FB E1 11 14.00 19 DD E1.FD CB 47 46"  34144
 19  "20 14 DD E5.DD 7E 00 F5.DD CB 00 BE.CD 70 09 F1"  36419
 20  "E1 77 DD 2B.18 0F DD 5E.0B DD 56 0C.3E FF 37 E5"  38312
 21  "DD E1 CD 56.05 DD E5 C1.C9 21 4C 6D.E5 11 4D 6D"  40548
 22  "01 9B 90 36.00 ED B0 C1.C9"                       41709
```

—

NOG ENKELE OPMERKINGEN

- De opties 1 en 2 werken alleen via MD of CR. De opties 3, 4 en 5 gebruiken MD-syntaxis en werken zonodig ook bij OD, DD en +D.
- Krijgt u een foutmelding, herstart dan met RUN en kies daarna eerst optie 0 om de buffer te legen voor u andere opties kiest.
- Bij optie 1 wordt de cartridge telkens geheel doorgespoeld bij het LOADen der blokken van 512 bytes. De MC gebruikt namelijk de routines uit de SP, niet uit IF1. Door de verschillen tussen alle IF1-versies gelukte het ons niet om dit LOADen te versnellen. Wie daar toch een oplossing voor weet gelieve dit ons te melden.

—

Kees Versluis - SGG-infotel 01670-66845 - Ed Weijgers

HET BASICPROGRAMMA "md><p" LINE 88 - EdW -

```
  1 DEF FN L(A)=PEEK (A+11)+PEEK (A+12)*256+20:
    LET B=28E3: LET T=FN L(23659)-20: LET A=T+FN L(T):
    LET D=PEEK 65535: LET L=A-B: LET F=65E3-A
  2 INPUT ("0 CLEAR (IN ";L;" OVER ";F;")",
           "1 LOAD N>P","2 SAVE P>N",
           "3 LOAD P"  ,"4 SAVE P",
           "5 CAT ";D  ,"6 <> DRIVE"),K
  3 IF K<1 OR K>6 THEN GO TO 99
  4 IF K<>2 AND K<5 THEN INPUT " LOAD 2 LOAD SAVE "(K);"NAAM",N$
  5 GO SUB K*11+D: POKE T,PEEK T+128
  6 IF PEEK A>=128 THEN LET A=A+FN L(A): GO TO 6
  7 RANDOMIZE A: RUN

  8 FOR F=1 TO 10: PRINT CHR$ PEEK (B+F);: NEXT F:
    PRINT " ";PEEK B,FN L(B): LET B=B+FN L(B):
    IF B<A THEN GO TO 8
  9 PAUSE T: RUN

 11 RANDOMIZE A: INPUT USR 65006: RETURN
 19 RANDOMIZE A: INPUT USR 65E3,D,N$:
    LET B=PEEK (A+11): POKE A,B:
    IF B THEN POKE A+19,PEEK (A+16)
 20 FOR F=A+12 TO A+17: POKE F-1,PEEK F: NEXT F:
    IF B<3 THEN POKE F-5,PEEK F: POKE F-4,PEEK (F+1)
 21 RETURN

 29 LET T=128 AND PEEK B>=128: LET N$=CHR$ (PEEK B-T)
 30 FOR F=B+11 TO B+16: LET N$=N$+CHR$ PEEK F: NEXT F:
    LET N$=N$+N$(4 TO 5): IF CODE N$ THEN LET N$(6)=N$(5)
 31 RANDOMIZE B: INPUT USR 65003,D,N$:
    IF T THEN LET B=B+FN L(B): GO TO 22
 32 RUN

 33 LOAD ""CODE A,F: RETURN
 41 LOAD *"M";D;N$CODE A,F: RETURN

 44 SAVE N$CODE B,L: RUN
 52 SAVE *"M";D;N$CODE B,L: RUN

 55 GO TO 8+NOT L
 63 CAT D: PAUSE T: RUN

 75 INPUT "0 CR / 1-6 MD",D: POKE 65535,D OR D<0 OR D>6: RUN

 88>CLEAR 27979: LET D=PEEK 23766: LOAD *"M";D;"Cmd><p"CODE :
    CLEAR #: POKE 65535,D

 99 RANDOMIZE USR 65009: RUN
```

—

NB: Gebruik waar mogelijk KEYWORDS. Dit moet beslist in regel 4.

Van de verschillende OpusROM-versies komen 2.1, 2.2 en 2.22 het meest voor. De twee laatste zijn speciaal bedoeld voor de SP128, maar werken ook op de SP48. In de versies 2.2 en 2.22 zijn twee extra BASIC-opdrachten opgenomen: CLEAR LINE en CLEAR DATA, voor het verwijderen van geselecteerde BASIC-regels of variabelen.

Martin van Drie geeft op DUCDISK-06 een programma om deze beide opdrachten aan ROM 2.1 toe te voegen. Een bezwaar is dat hij onvoldoende rekening houdt met willekeurige MC-combinaties in het IC 6116. (M'n petje af overigens voor zijn vele en goede werk.)

In IMPULS 74-13 vindt u "mc>odram", waarmee elke gebruiker zijn eigen selectie van routines kan maken. Het heeft Ed Weijgers (en niet alleen hem) veel tijd gekost om aan alle wensen tegemoet te komen. U begrijpt dat nieuwe MC op een speciale manier gepresenteerd wordt. Ook bij het programmeren in GENS moet soms rekening gehouden worden met de eisen die het installatieprogramma stelt. Mochten er vragen zijn, schroom dan niet, maar bel of schrijf.

Enfin, twee extra mogelijkheden dus met CLEAR voor ROM 2.1. Deze opdrachten zijn eenvoudig overgenomen uit ROM 2.2 en hun werking kunt u naslaan in de OD-ROM-Disassembly. Het onderstaande programma bevat DATA-regels, die door "mc>odram" geMERGEt worden.

—

HET MERGE-PROGRAMMA "Xclear2.1" (voeg DATA toe in elke regel):

```
100  151                                  : REM DATABLOK tbv MC
101  "FE CA 28 07.FE E4 28 41.18 5B 00 D7.79 1C CD 6F",1885
102  "04 CD DD EA.20 04 D7 B8.19 EB E5 CD.DD EA C1 ED",4563
103  "42 D0 09 EB.2A 5D 5C CD.EE EA 38 09.62 6B 22 55",6374
104  "5C 2B 22 5D.5C 2A 57 5C.CD EE EA 38.06 1B ED 53",8035
105  "57 5C 13 60.69 D7 E5 19.C9 E7 D7 B2.28 E5 79 17",10141
106  "E6 C1 EE 40.CB 3F F5 28.10 D7 18 00.FE 28 20 09",12007
107  "E7 FE 29 28.03 D7 8A 1C.E7 CD 6F 04.F1 E1 D8 20",14222
108  "05 2B CB 7E.28 FB D7 B8.19 D7 E8 19.C9 D7 99 1E",16385
109  "78 FE 40 38.03 D7 9F 1E.60 69 D7 6E.19 C9 A7 ED",18442
110  "52 D8 19 ED.42 3F C9"                                ,19332
111  " CLEAR ",8                          : REM DATABLOK tbv opdracht
112  18,125,28,125,40,142,57,142,0        : REM DATABLOK tbv plaats
113  15,14,16,5,106,14,107,5,0            : REM DATABLOK tbv versie
```

Deze routine bevat twee subroutines die elk tweemaal aangeroepen worden. Ik gebruik hiervoor niet de CALL RELATIVE-optie, maar de mogelijkheid van "mc>odram" om de MC aan te passen aan de plaats in het geheugen. Lees even mee op regel 112: op 'start+18' en op 'start+28' komt het adres 'start+125'; hetzelfde gebeurt voor de subroutine op 'start+142'. Hoewel de routine uitsluitend bestemd is voor ROM 2.1 zijn in regel 113 de aanpassingen opgenomen voor de versies 2.2 en 2.22. Baat het niet, dan schaadt het ook niet.

—

Kees Versluis - Copernicuslaan 25 - 2561 VA Den Haag

Naar aanleiding van vragen over mijn in het artikeltje "POKEN IN PROGRAMMAREGELS" (IMPULS 81-42) aangehaalde programma "toolkit7" (IMPULS 10-40) heb ik mij weer eens in dat programma verdiept.

Het werd destijds min of meer geschreven en gepubliceerd als een grapje, om te laten zien dat er ook in BASIC veel dingen kunnen waarvan je zou denken dat ze alleen in MC mogelijk zijn.
Natuurlijk zouden dergelijke regelmanipulaties in MC wel sneller werken, maar die snelheid valt in BASIC ook best mee.
Bovendien bleek het programma bruikbaarder te zijn dan verwacht. Ik ken ook geen MC die alle mogelijkheden van "toolkit7" biedt: regels verlengen of splitsen; regelblokken vooruit of achteruit over regels heen verplaatsen; regelblokken verwijderen; regelblokken hernummeren met een stapgrootte tov de beginregel ervan, of de regelnummers ervan met een constante verhogen of verlagen.

Tijdens het opnieuw bestuderen van dit programma realiseerde ik mij dat het al zo lang geleden is dat het in IMPULS verscheen en ook dat ik zelf in die tijd zoveel bijgeleerd heb dat ik het nu toch weer anders zou doen (hoewel het helemaal foutloos werkt). Kortom, ik ben het gaan herschrijven. Het is nog korter en zelfs sneller geworden. Het resultaat, dat ik nu maar gewoon "toolkit" heb genoemd, krijgt u op de volgende naast elkaar gelegen bladzijden te zien, compleet met een beschrijving van alle mogelijkheden, waardoor degenen die in BASIC geinteresseerd zijn de werking gemakkelijker kunnen volgen. D'r zitten trucjes genoeg in!

—

HET GEBRUIKEN VAN DIT PROGRAMMA

LOAD het BASIC-programma dat u wilt bewerken, geef CLEAR en dan MERGE .. "toolkit". RUN geeft altijd het menu met zeven opties, met daarboven het adres en het regelnummer van de cursorregel. Met LIST of met PIJLtoetsen kan de cursor in een bestaande regel als > achter het regelnummer gezet worden. Dat is nodig omdat de opties alleen werken met of ten opzichte van die cursorregel.

—

HET SAMENSTELLEN VAN DIT PROGRAMMA

U krijgt "toolkit" in regel 0 door om te beginnen de navolgende regels 10 t/m 17 over te nemen. Laat alle overbodige spaties weg en gebruik overal waar dat in strings mogelijk is BASIC-woorden. In plaats van de vraagtekens in regel 10 moet ingetoetst worden: EXTEND MODE 6 DELETE (de PRINT-komma CHR$ 6, zie IMPULS 72-19). Op analoge wijze krijgt u CHR$ 4 en CHR$ 5 in regel 11, resp 13. Kopieer de regels 10 en 11 naar regels 1 en 2 door ze te EDITten en de regelnummers te veranderen. Zet de cursor dan in regel 10. Verleng regel 10 met de regels 11 t/m 17 door 7 keer optie 1 te kiezen (na RUN; telkens EDIT ENTER erna; duurt telkens langer!). Verwijder dan de regels 1 en 2. Geef RUN en STOP, en vervolgens POKE A+1,0: CLEAR . Nu kan "toolkit" geSAVEd worden (geen LINE).

—

HET BASICPROGRAMMA "toolkit" - EdW - 905 bytes -

```
 10 LET I=SGN PI: LET L=VAL "23670": LET H=L+I:
    LET A$="A+4+PEEK (A+2)+PEEK (A+3)*256":
    LET R$="PEEK A*256+PEEK (A+1)":
    LET R=VAL "PEEK 23625+PEEK 23626*256": RESTORE R:
    LET A=VAL "PEEK 23639+PEEK 23640*256+1": LET B=A+INT PI:
    INPUT "AT ";(A ," LINE ";R)''
          "1 NEXT ]  ?2 TO ]?                          "?"=CHR$ 6
           3 :[ NEXT ?4 ][?
           5 STEP TO ?6 +- TO ?
           7 ERASE 0",k

 11 FOR k=k TO I: LET A=VAL A$: POKE A-I,CODE ":":
    FOR C=A+CODE "?" TO VAL A$-I:                    "?"=CHR$ 4
      POKE C-4,PEEK C: NEXT C:
    RANDOMIZE C-B-I: POKE B-I,PEEK L: POKE B,PEEK H:
    PRINT "EDIT LINE ": STOP : NEXT k

 12 FOR k=k TO I+I: LET A=VAL A$:
    LET B$=CHR$ PEEK (B-I)+CHR$ PEEK B: INPUT " TO ";R:
    FOR C=I TO R:
      LET B$=B$+CHR$ PEEK A+CHR$ PEEK (A+I)+
               CHR$ PEEK (A+k)+CHR$ PEEK (A+PI):
      LET C=VAL R$: LET K=VAL A$: POKE A-I,58:
      FOR R=A TO A+PI: POKE R,32: NEXT R: LET A=K: NEXT C:
    RANDOMIZE A-B-I: POKE B-I,PEEK L: POKE B,PEEK H:
    STOP : NEXT k
```

>>

EEN BESCHRIJVING VAN DE OPTIES

1 NEXT]

EDIT ENTER — Verlengt de cursorregel met de volgende regel. Geef na afloop (om vier bytes achteraan kwijt te raken).

2 TO]

Verlengt de cursorregel t/m een bij TO ? gegeven regelnummer. Nummer- en lengtegegevens worden in B$ opgeslagen tbv van splitsoptie 4 en vervangen door 4 spaties met : ervoor. Na afloop kan het dan ontstane blok regels in zijn geheel worden VERWIJDERD door cursorregelnummer ENTER in te toetsen, dan wel VERPLAATST (gekopieerd) door het te EDITten, het regelnummer te veranderen, GO TO 0 (geen RUN) te geven en optie 4 te kiezen.

3 :[NEXT

Splitst de cursorregel achter een gegeven opdracht. Zet vooraf REM opdrachtnummer aan het begin van de cursorregel. Dit nummer moet uit drie cijfers bestaan (evt nullen voorop) en wordt gebruikt om de :-tekens te tellen (dus ook in strings!). De afgesplitste regel krijgt het nummer van de cursorregel + 1. Na afloop EDIT ENTER PIJL-NEER EDIT ENTER geven tbv FPR-herstel.

>>

```
13 FOR k=k TO PI: LET K=B+CODE "?":                    "?"=CHR$ 5
   FOR C=I TO VAL (CHR$ PEEK (K-PI)+
                   CHR$ PEEK (B+PI)+CHR$ PEEK (K-I)):
     POKE K-4,PEEK K: LET K=K+I: LET C=C-(PEEK K<>58): NEXT C:
   POKE K-PI-I,VAL "13":
   RANDOMIZE R+I: POKE K-PI,PEEK H: POKE K-SQR PI,PEEK L:
   RANDOMIZE VAL A$-K-I: POKE K-I,PEEK L: POKE K,PEEK H:
   RANDOMIZE K-B-PI-I: POKE B-I,PEEK L: POKE B,PEEK H:
   PRINT "EDIT LINE AND NEXT LINE ": STOP : NEXT k

14 FOR k=k TO PI+I: POKE B-I,CODE B$: POKE B,CODE B$(SQR PI):
   FOR C=PI TO LEN B$ STEP k:
     LET A=VAL A$: POKE A-I,13:
     FOR R=NOT I TO PI: POKE A+R,CODE B$(C+R): NEXT R:
     NEXT C: STOP : NEXT k

15 FOR k=k TO LN PEEK PI: INPUT " STEP ";B," TO ";K:
   FOR C=I TO I:
     LET A=VAL A$: LET C=VAL R$=K: LET R=R+B: RANDOMIZE R:
     POKE A,PEEK H: POKE A+I,PEEK L: NEXT C: STOP : NEXT k

16 FOR k=k TO PI+PI: INPUT "+- ";B," TO ";K:
   FOR C=I TO I:
     LET A=VAL A$: LET R=VAL R$: RANDOMIZE R+B:
     POKE A,PEEK H: POKE A+I,PEEK L: LET C=R=K: NEXT C:
   STOP : NEXT k

17 POKE A+I,I: PRINT " ERASE 1": STOP
```

—

4][

Splitst een met optie 2 verlengde cursorregel. Deze optie werkt alleen indien er na optie 2 nog geen RUN gegeven is. Op de cursorregel na herkrijgen alle regels hun oorspronkelijke regelnummers uit B$. Na afloop valt er echter nog meer te doen. Regels met nummers die niet hoger zijn dan voorgaande regelnummers kunnen gewist noch geEDIT worden. Met LIST of PIJL-toetsen komt de cursor er niet in of voorbij. Kies eerst optie 5 of 6 om regelnummers te wijzigen (na GO TO 0 als optie 4 nog nodig is!).

5 STEP TO

Hernummert de regels na de cursorregel (tov het regelnummer ervan) met stappen van een bij STEP ? gegeven grootte, t/m een bij TO ? gegeven (oud) regelnummer. (Niet na GO TO/SUB).

6 +- TO

Verhoogt/verlaagt de regelnummers na de cursorregel met een bij +- ? gegeven waarde t/m een bij TO ? gegeven nummer.

7 ERASE 0

Verwijdert toolkitregel 0, mits die cursorregel is. Deze wordt daardoor regel 1 en kan met 1 ENTER worden gewist.

—

Nooit heb ik goed begrepen wat nou eigenlijk het nut zou kunnen zijn van de functie VAL$. De SP-handleiding geeft wel een aantal voorbeelden om te proberen (veel gehannes met aanhalingstekens). Het klopt ook allemaal wel, maar toepassingen waar je iets aan hebt vond ik er niet bij, en die kon ik niet bedenken ook.
De functie VAL gebruik ik echter geregeld, en niet alleen om de FPR's te vermijden om geheugenruimte te besparen. Toch ligt bij VAL de sleutel om VAL$ beter te begrijpen, zoals we zullen zien.

Stel dat N$ de naam BEVAT van een ongeDIMde numerieke variabele. Dan stelt VAL N$ de getalwaarde van die variabele voor.

```
LET Jaartal=1900+90:          hierdoor verschijnt
LET N$=" jaar tal ":          op het scherm:
PRINT n$;"=";VAL n$            jaar tal =1990
```

Stel dat N$ de naam bevat van een ongeDIMde stringvariabele. Dan stelt VAL$ N$ de stringwaarde van die variabele voor.

```
LET T$="titel "+STR$ 6:       hierdoor verschijnt
LET N$=" t$":                 op het scherm:
PRINT n$;"=";VAL$ n$           t$=titel 6
```

(Hieruit blijkt dat er bij NAMEN van variabelen geen onderscheid bestaat tussen hoofdletters en kleine letters. Ook blijven alle spaties voor, in en achter die namen altijd buiten beschouwing.)

TOEPASSING

We kunnen een programma schrijven waarbij het variabelengeheugen (VARS t/m E_LINE-1) doorlopen wordt door een adresvariabele A. De naam van een variabele kunnen we telkens in N$ zetten mbv oa:

```
LET T=INT (PEEK A/32): LET N$=CHR$ (PEEK A-T*32+96)
```

Hierna dient N$ nog aangevuld te worden met "$" bij T=2 en T=6, of met de resterende tekens van een langere naam wanneer T=5.

Voor de waarde behoeven we echter niet te PEEKen, dat gaat daarna eenvoudig mbv VAL N$ of VAL$ NS (afhankelijk van het type T). We kunnen de naam van een variabele ook een index I meegeven:

```
VAL$ (N$+"("+STR$ I+")")          [Verwar dit niet met N$(I)]
```

Natuurlijk kunnen er ook allerlei andere combinaties van stringwaardige functies en operaties gebruikt worden (ook binnen N$).

—

Eindelijk eens een echte toepassing van VAL$ ontdekt! Toegegeven dat deze niet erg belangrijk is, maar misschien is er iemand die er meer kent, of hierdoor op een idee komt. Laat het ons weten. Voor de functie SGN heb ik overigens ook nooit iets gevonden wat niet op een andere manier handiger ging. U wel? - EdW -

Het gemis van een RS232-interface op de Opus Discovery werd door Arthur Hoornweg verholpen. Dat was al in 1988. Hij ontwierp een kant-en-klaar pakket (hardware, software, handleiding), dat nog steeds bij de SGG verkrijgbaar is. Zie ook IMPULS 63-14 & 72-61.

Arthur heeft een gebruikersvriendelijk pakket gemaakt, waarin oa baudrate, aantal databits, parity en aantal stopbits zijn in te stellen. Er zijn toch een paar nadelen: transmissie sneller dan 4800 baud is niet steeds betrouwbaar en de CODE staat altijd in Spectrum-RAM. Vooral dat laatste is (bij BETA BASIC!) ongewenst.

Hierna staat het MERGE-programma "Xpar>ser". U snapt het al: dat hoort bij het installatieprogramma "mc>odram" uit IMPULS 74-13. U begrijpt misschien ook al wat er gaat gebeuren: in het IC 6116 komt EEN routine, die alles in seriele vorm naar de printerpoort stuurt. Er zijn dan geen speciale RS232-versies meer nodig voor BETA BASIC, TW2, TW3, SW of Devpac. Alles werkt, in BASIC en MC.

Dankzij het tabellensysteem in de OD-ROM is het mogelijk om een seriele routine vast te knopen aan het 'operating system' van de Opus. Dit idee komt van Wim Beekman. Deze routine is geschreven door Jack Raats. Het MERGE-programma bestaat uit twee delen:

- regelblok 60 - 69: wijziging van enkele tabellen,
- regelblok 100-123: seriele routine

Als deze routine geinstalleerd is kan de inhoud van het IC 6116 naar disk geSAVEd worden. Naderhand LOADen van dit CODEblok lukt echter niet op de normale manier. Dat gaat alleen met een omweg:

```
1 LOAD *1;"naam"SCREEN$ : OPEN #3;"CODE ":POINT #3;8185:
  SAVE *#3CODE 16384,2048: CLOSE #3
```

—

Deze routine werkt goed tot 19200 baud. Een nadeel is het vaste protocol: 8 databits, geen parity en 2 stopbits. In de praktijk komt echter nauwelijks iets anders voor. Hoe u de baudrate kunt veranderen ziet u in regel 69 van het MERGE-programma. In plaats van POKEn moet u PRINTen naar het "CODE "-kanaal. Ik geef straks nog een konkreet voorbeeld. 'Waar' kunt u 'wat' veranderen?

```
adres 10095/10096  : zendtimer
adres 10097/10098  : ontvangsttimer
adres 10099        : I/O-borderkleur
```

Het knipperen van de BORDER heeft een controlefunktie en behoort een regelmatig verloop te hebben. De ingesteld kleur is 2, rood.

De baudrate wordt bepaald door baudrate-timers, maar de klokfrequentie van de computer speelt ook een rol. Daarom gelden voor SP48 en SP128 verschillende timers. In de tabel hierna staan van enkele baudrates de L- en H-byte van die timer. Voor meer informatie over deze baudrate-timers verwijs ik u naar IMPULS 74-37.

HET MERGE-PROGRAMMA "Xpar>ser"

```
  60 POINT #3;8192: LET p=FN g()+8:
     POINT #3;p: LET t=FN g():
     POINT #3;t: LET t$=""
  61 LET t$=t$+INKEY$#3+INKEY$#3+INKEY$#3:
     IF CODE t$(LEN t$-2) THEN GO TO 61
  62 FOR f=1 TO LEN t$ STEP 3:
       IF t$(f)<>CHR$ 129 THEN NEXT f
  63 POINT #3;t+f: LET s=FN g():
     RANDOMIZE m: LET s$=FN g$():
     POINT #3;s+4: RANDOMIZE m+104:
     LET s$=s$+FN g$()+INKEY$#3+INKEY$#3+INKEY$#3+INKEY$#3
  64 IF s<8192 THEN LET s=m+L: LET L=L+LEN s$
  65 IF t<8192 THEN LET t=m+L: LET L=L+LEN t$
  66 IF m+L>c AND m=b OR m+L>e THEN
       PRINT "MC te lang": CLEAR #: GO TO 3
  67 POINT #3;s: LPRINT s$;:
     RANDOMIZE s: LET t$(f+1 TO f+2)=FN g$()
  68 POINT #3;t: LPRINT t$;:
     POINT #3;p: RANDOMIZE t: LPRINT FN g$();
  69 POINT #3;10095:
     LPRINT CHR$ 12+CHR$ 0;CHR$ 12+CHR$ 0;CHR$ 2;

 100 326 : REM DATABLOK tbv MC - DATA achter elk regelnummer
 101 "7C 06 0B 2F.4F 3A 73 27.D3 FE DD 21.02 30 DD 36",1523
 102 "01 00 DD 36.00 0F DD 36.01 04 DD 36.00 08 2A 6F",2530
 103 "27 2B 54 5D.1B 7A B3 20.FB CD 5B 14.DD CB 00 76",4258
 104 "28 F7 37 F3.DA 9E EA DD.CB 00 96 C3.A5 EA DD CB",7045
 105 "00 D6 C3 A5.EA 54 5D 1B.7A B3 20 FB.3E 00 AF CB",9081
 106 "39 10 E1 FB.DD 36 00 08.2B 7C B5 20.FB 3A 48 5C",10766
 107 "E6 38 0F 0F.0F D3 FE C9.21 A4 EB 7E.A7 28 06 36",12588
 108 "00 23 7E 37.C9 F3 ED 5B.71 27 21 20.03 42 4B CB",14140
 109 "38 CB 19 CD.5B 14 DD 21.02 30 DD 36.01 00 DD 36",15595
 110 "00 0F DD 36.01 04 DD 36.00 0A DD CB.00 6E 28 12",16767
 111 "DD CB 00 6E.28 0C DD CB.00 6E 28 06.DD CB 00 6E",18467
 112 "20 0C 2B 7C.B5 20 E3 F5.DD 36 00 08.18 31 60 69",19920
 113 "DD 36 00 08.3A 73 27 D3.FE 06 80 2B.2B 2B 2B 19",21211
 114 "2B 2B 7C B5.20 FB DD CB.00 6E CA 40.EB 37 18 03",23002
 115 "B7 18 00 CB.18 30 E8 DD.36 00 08 78.2F 37 F5 19",24491
 116 "2B 7C B5 20.FB 19 19 19.2B 7C B5 28.41 DD CB 00",26074
 117 "6E 28 F5 DD.CB 00 6E 28.EF DD CB 00.6E 28 E9 DD",28310
 118 "CB 00 6E 28.E3 60 69 06.80 2B 2B 2B.2B 19 2B 2B",29508
 119 "7C B5 20 FB.DD CB 00 6E.CA 8E EB 37.18 03 B7 18",31498
 120 "00 CB 18 30.E8 21 A4 EB.36 01 23 78.2F 77 CD BD",33207
 121 "EA F1 FB C9.00 00"                               ,34134
 122 "",0                              : REM DATABLOK tbv opdracht
 123 53,62,60,69,67,69 105,324,219,224,297,302
 124 310,324,319,93,0                  : REM DATABLOK tbv plaats
 125 42,64,43,21,132,64,133,21,0       : REM DATABLOK tbv versie
```

BAUD	SP48 L	H	SP128 L	H
75	1	7	25	7
150	127	3	139	3
300	191	1	197	1
600	222	0	225	0
1200	110	0	112	0
2400	54	0	55	0
4800	26	0	26	0
9600	12	0	12	0
19200	5	0	5	0

In regel 69 ziet u dat 9600 baud wordt ingesteld voor zowel zenden als ontvangen voor SP48 en SP128 (en rood als knipperkleur). Wilt u een andere baudrate (bijv 1200) ga dan als volgt te werk:

```
OPEN #3;"CODE ": POINT #3;10095:
  LPRINT CHR$ 110+CHR$ 0;CHR$ 110+CHR$ 0;      [voor SP48 ]
  LPRINT CHR$ 112+CHR$ 0;CHR$ 112+CHR$ 0;      [voor SP128]
CLOSE #3
```

Let goed op de ';' . Merk op dat de knipperkleur niet verandert.

—

Maar nu de praktijk. Deze RS232-poort wordt met dezelfde BASIC-opdrachten bestuurd als de OD-parallelpoort. In assembler blijft CALPHY dezelfde krachtige routine. Feitelijk verandert er niets! In gebruik vanuit BASIC dient altijd een "b"- of een "t"-kanaal geopend te worden. Denk ook aan de 'status', die u bij het "t"-kanaal mag opgeven. Lees het na: DUC #8-38 en de OD-handleiding.

—

Het werken met RS232 is niet eenvoudig, maar toch belangrijk genoeg om er iets van te weten. Communicatie met andere computers lukt vrijwel altijd via de RS232-standaard. Bestanden en teksten kunnen zo uitgewisseld worden, soms pas na moeizaam proberen.

Deze routine gebruikt de 'handshakelijnen' op de pennen 4 en 5. Op connectors van andere computers zult u dus moeten zoeken naar de juiste pennen. Ik heb de routine getest met een tweede Spectrum (met SGG-IF1 & "C><sggif1" uit IMPULS 81-46) via een nulmodemkabel.

OD		SGG-IF1
2	<------	3
3	------>	2
4	<------	5
5	------>	4
7	-------	7

Arthur Hoornweg geeft in zijn handleiding een aantal interessante toepassingen. Die blijven natuurlijk bruikbaar. In de voormalige SINCLAIR GIDS (nrs 8, 9 en 10) zijn artikeltjes verschenen over RS232-communicatie van SP met QL en PC. Veel succes ermee.

—

Kees Versluis - Copernicuslaan 25 - 2561 VA Den Haag

Het SAVEn van een enkele geDIMde variabele N of S$ is mogelijk met de toevoeging DATA N() of DATA S$() achter de SAVE-naam. Alle variabelen tegelijk, zonder het BASIC-programma, kan aldus:

```
POKE 23635,PEEK 23627: POKE 23636,PEEK 23628: SAVE .. "naam"
```

Hierdoor krijgt de systeemvariabele PROG de waarde van de sysvar VARS, waardoor de SP tijdens het SAVEn geen BASIC-regels "ziet". Die ziet u daarna ook niet meer, waardoor u genoodzaakt bent om RANDOMIZE USR 0 in te toetsen of om op de resetknop te drukken. Staat u dat niet aan, dan kunt u vooraf PROG in SEED opslaan met

```
RANDOMIZE PEEK 23635+PEEK 23636*256
```

hetgeen u in staat stelt om achteraf PROG weer te herstellen met

```
POKE 23635,PEEK 23670: POKE 23636,PEEK 23671
```

Het moge duidelijk zijn dat deze wijze van SAVEn niet vanuit een programma lukt! TerugLOADEn wel. Dat gaat met MERGE .. "naam".

—

Andrew Pennell geeft in "Master Your ZX Microdrive" SAVE-/ LOAD-routines welke wel in BASIC-programma's opgenomen kunnen worden. Maar deze gebruiken zelf een variabele L, waardoor het variabelengeheugen wordt veranderd. Dat is echter helemaal niet nodig. Hier volgt een handigere en kortere uitwerking van zijn ideeen. Het is wel nodig dat het programma deze functiedefinitie bevat:

```
DEF FN P(A)=PEEK A+PEEK (A+1)*256
```

Voor het SAVEn van het variabelengeheugen inclusief de lengte:

```
RANDOMIZE FN P(23641)-FN P(23627):
SAVE .. "Cvar1"CODE 23670,2:
SAVE .. "Cvars"CODE FN P(23627),FN P(23670)
```

Deze regel (of de opdrachten eruit) kunt u overal waar nodig in uw programma opnemen. Dat kan natuurlijk ook in een SUB-routine. De in SEED opgeslagen lengte E_LINE-VARS wordt eveneens geSAVEd.

Voor het terugLOADen en herstellen van het variabelengeheugen:

```
CLEAR : LOAD .. "Cvar1"CODE : DIM V$(FN P(23670)-7):
LOAD .. "Cvars"CODE FN P(23627)
```

Als u hiervan een SUB-routine wilt maken moet CLEAR daar uiteraard niet in opgenomen worden, dat moet voor GO SUB geschieden! De in SEED herLOADe lengte wordt gebruikt om V$ te DIMmen. Daardoor krijgt het geCLEARde variabelengeheugen de juiste grootte. V$ wordt vervolgens overschreven door alle herLOADe variabelen.

- EdW -

Mike Lloyd vroeg zich af of een calculatorprogramma voor de QL nog zin had nu er al vele computercalculators zijn en de rekenapparaatjes voor f 3,50 in de winkel liggen. Gelukkig begon hij er toch aan. Het verscheen in QL-World, in afleveringen van februari t/m juli 1989. Hier volgt een korte beschrijving daarvan.

Mikes motieven lagen in deze nadelen van de meeste calculators: te kleine displays, te weinig geheugenplaatsen (vaak maar een), alleen decimaal rekenen. Zijn QL-CALCULATOR beschikt daarom over

- een display dat ook de laatste acht bewerkingen nog toont;
- vijf geheugenplaatsen;
- de mogelijkheden decimaal, hexadecimaal, octaal en binair;
- vijf constanten voor invoer;
- vijf 'settings' (bijvoorbeeld wel of niet printen);
- de mogelijkheid om bewerkingen gelijktijdig af te drukken.

Hij moest zich ook beperkingen opleggen. Doordat hij alleen met functietoetsen wilde werken zijn de keuzemogelijkheden beperkt. Die functietoetsen zijn echter wel snel te bedienen. Het gebruik is erg eenvoudig geworden, ondanks de uitgebreide mogelijkheden. De calculator beslaat op het scherm 10 regels met 20 posities. Het BASIC-programma start door 'calc' in te toetsen (procedure).

Functietoetsen

F1	Store	op de 5 geheugenplaatsen F1, F2, F3, F4, F5
F2	Fetch	terughalen uit deze vijf naar de display
F3	Constants	Pi, scr-start(131072), 64K, km->Mile, Lb->kg
F4	Settings	F1 en F2 om van talstelsel te wisselen, F3 printer aan/uit, F4 mode REAL, F5 Return
F5	Clear	

De plaats waar de getallen afgedrukt moeten worden kan voor het rekenen met een TAB-opdracht aan een printer doorgegeven worden. Zo kan men de berekeningen meteen in een brief zetten en dus in multitasking werken.
De 'constants' zijn uitsluitend bij de BASIC-versie te wijzigen.

—

CONCLUSIE

Een handige calculator (beter en veelzijdiger dan die van Qpac bijvoorbeeld), die zeker voor elke programmeur van nut zal zijn. Ook elke andere Ql-bezitter zal er graag over willen beschikken. Het enige probleem is nog de compilering. Daarvoor moet het programma iets gewijzigd worden.
QL-WORLD's 'Microdrive-exchange' komt met een iets uitgebreidere versie in gecompileerde vorm.
Deze kan bijvoorbeeld in de plaats komen van de Qpac-calculator.

—

Joop van der Maas - Egberinkseweg 101 - 7548 RS Boekelo

De Spectrum geeft 32 tekens op een schermregel. Dat is weliswaar goed leesbaar, maar voor een aantal toepassingen toch te weinig. Er zijn daarom MC-routines ontwikkeld om met 42, 51 of 64 tekens op een regel te werken. De tekens zijn dan resp 6, 5 of 4 pixels breed, incl de ruimte tussen de tekens. Dit artikel gaat over de tekensets van 64-kolomsroutines in (on)bekende Spectrumsoftware.

Het nadeel van deze tekens is de geringere leesbaarheid. Al vaak is geschreven over het verbeteren van de tekenset in TW2 (al in IMPULS 07, 1985!). Ook zijn er diverse 'font editors', dus het is niet zo moeilijk om een tekenset te ontwerpen of te wijzigen. (Ook de editors in Art Studio en The Artist zijn heel geschikt.)

Aan het eind van dit artikel staat overigens een prima tekenset. Hebt u eenmaal uw ideale tekenset, dan wilt u die in de diverse programma's inbouwen. Ik wil uitleggen hoe u dat kunt aanpakken.

—

Alle tekens zijn opgeslagen als bitpatroon, ook de standaard 32-kolomsset. Hieronder geef ik het bitpatroon van de letter "A" in twee uitvoeringen: standaard en smal (- en # staan voor 0 en 1).

```
-------- 0     -------- 0     De getallen ernaast zijn de
--####-- 60    -----### 7     decimale equivalenten.
-#----#- 66    -----#-# 5     Voor de standaard "A" staan
-#----#- 66    -----#-# 5     die getallen in de SP-ROM op
-######- 126   -----### 7     de adressen 15880 t/m 15887.
-#----#- 66    -----#-# 5
-#----#- 66    -----#-# 5     Deze smalle tekenset noem ik
-------- 0     -------- 0     LOW NIBBLE.
76543210       76543210
```

De meeste 64-kolomsroutines gebruiken het rechter bitpatroon. De bits 0-3 vormen dan de letter, de bits 4-7 worden niet gebruikt. Elk teken wordt dus opgeslagen in 8 bytes. Alle 96 ASCII-tekens vormen zo een geheugenblok van (96 * 8 =) 768 bytes. Het eerste teken in zo'n blok is altijd de spatie en bestaat uit 8 nullen.

—

De grote vraag is altijd: Op welk adres staat de tekenset? Sommige handleidingen vermelden wel een adres. Maak daarbij het volgende onderscheid:

basisadres - de eerste byte van CHR$ 0 (vaak niet PRINTbaar)
beginadres - de eerste byte van CHR$ 32 (spatie)

Het basisadres is meestal alleen van belang voor MC-deskundigen. Het beginadres is tevens het SAVE- en LOAD-adres:

```
SAVE ... "naam"CODE adres,768      LOAD ... "naam"CODE adres
```

Met deze opdrachten zijn de tekensets van verschillende programma's onderling te verwisselen. De adressen staan altijd in RAM.

Hieronder staat een lijstje van programma's en beginadressen van de tekenset. Op dat adres begint dus de spatie: 8 nullen.

TASWORD 2	-	61184	GRAPH
TASWORD 3	-	38400	zie verder
TASWORD 128 / TASWORD +2	-	54528	zie verder
TASCALC	-	25401	???
TASWIDE	-	63232	GRAPH + UDG
SUPERCAT (DUCDISK-08)	-	zie TASWIDE	
FIDOTERM (alle versies)	-	64750	
SKIP-64	-	start + 192	GRAPH + UDG
SUPERFILE, KASBOEK, GROOTBOEK (DATA SKIP)	-	63992	GRAPH + UDG
ADRES 4.0	-	64495	
SYS 64	-	63631	

Al deze programma's hebben dus een 'LOW NIBBLE'-set (bits 0-3). Sommige tekensets bevatten naast de ASCII-set (32-127) nog meer. Dan kunnen ook de graphische bloktekens (128-143) en/of de UDG's (144-164) opgenomen zijn. Dat staat hierboven ook aangegeven. Over TW3, TW128 en TW+2 zal ik het verderop nog hebben.

—

-------- 0	Ik ken EEN programma dat het pixelpatroon
-###---- 112	opslaat in de bits 4-7. Dat is de tekst-
-#-#---- 80	verwerker SPECTRAL WRITER. Hiernaast staat
-#-#---- 80	weer de "A".
-###---- 112	
-#-#---- 80	Ik noem zo'n tekenset HIGH NIBBLE.
-#-#---- 80	
-------- 0	Hieronder staat weer het beginadres:

SPECTRAL WRITER	-	64639	GRAPH

Er is een eenvoudige manier om een 'HIGH NIBBLE'-set te veranderen in een 'LOW NIBBLE'-set en andersom. Ik neem voor het gemak het adres uit Spectral Writer.

HIGH NIBBLE >> LOW NIBBLE:

```
FOR f=64639 TO 64639+96*8-1: POKE f,PEEK f*16: NEXT f
```

LOW NIBBLE >> HIGH NIBBLE:

```
FOR f=64639 TO 64639+96*8-1: POKE f,PEEK f/16: NEXT f
```

Met deze BASIC-regeltjes verplaatsen we dus het bitpatroon naar resp de rechter- en de linkerkant van de byte. Simpel nietwaar?

De meeste 64-koloms routines gaan ervan uit, dat de ongebruikte bits gereset zijn. Het bitpatroon wordt namelijk 'gemerged' met een letter die (links of rechts daarvan) al op het scherm staat.

Er zijn 64-koloms routines die hun tekenset heel compact in het geheugen hebben. Daarbij worden alle bits efficient gebruikt. Ik geef weer het voorbeeld van de "A".

```
-----### 7
-#-#-#-# 85
-###-#-# 117
-#-#---- 80
```

Het zal wel wat van uw voorstellingsvermogen vragen om hierin dezelfde "A" te herkennen.

Ik noem deze set COMPACT_1.

Het zal duidelijk zijn dat de ASCII-tekens hier maar (96 * 4 =) 384 bytes bezetten. Dat is toch een behoorlijke ruimtebesparing, vergeleken met de hiervoor besproken vormen. Nu volgt het lijstje van software met de beginadressen van een COMPACT_1 tekenset:

CHAIN (DUCDISK-15)	-	25886
LINK-ED (DUCDISK-15)	-	25884
LINK-ED (oud)	-	30132
LINKER (oud)	-	30132
PRINT_64 (Sinclair Gids 7)	-	start + 256
DE KLERK	-	64711 zie verder
C64t/r (DUCDISK-20)	-	start + 600

(De programma's LINK-ED en LINKER heb ik 'oud' genoemd, omdat ze alleen werken met standaard Opusdisks. Ze zijn ook niet meer bij de DUC-bank verkrijgbaar. CHAIN werkt wel met alle diskFORMATs.)

De omzetting van COMPACT_1 naar LOW NIBBLE (en terug) is niet zo eenvoudig in BASIC. Vandaar deze MC-omzetter:

DE BASIC "ln><cpt1" LINE 4

```
 1 INPUT "0 LOW NIBBLE > COMPACT_1"'
         "1 COMPACT_1 > LOW NIBBLE"'k:
   IF k<>0 AND k<>1 THEN RUN
 2 INPUT " LOAD ";"LN" AND NOT k;"CPT1" AND k;
         "-set van drive ";d'"naam ";n$:
   LOAD *"m";d;n$CODE 4e3+k*1e4: RANDOMIZE USR (6e4+k+k)
 3 INPUT " SAVE ";"LN" AND k;"CPT1" AND NOT k;
         "-set naar drive ";d'"naam ";n$:
   SAVE *"m";d;n$CODE 5e4-k*1e4,384*(k+1): STOP : RUN
 4 CLEAR 3e4: LOAD *"m";1;"C1n><cpt1"CODE 6e3: RUN
```

DE MC "C1n><cpt1"CODE 60000,62

```
 1 "18 1F 21 50.C3 11 40 9C.01 80 01 7E.F5 0F 0F 0F"   1146
 2 "0F E6 0F 12.13 F1 E6 0F.12 13 23 0B.78 B1 20 EB"   2576
 3 "C9 21 40 9C.11 50 C3 01.80 01 7E 07.07 07 07 D5"   3819
 4 "57 23 7E B2.D1 12 23 13.0B 78 B1 20.ED C9"         5304
```

De vorm COMPACT_1 komt het meeste voor. Ik zal straks nog ingaan op enkele andere vormen van een compact opgeslagen 64-kolomsset.

—

De tekenset van Tasman (TW2-TW3-TW128-TW+2-Taswide-Tascalc) laat veel te wensen over. Een beter leesbare set is simpel te LOADen op de adressen uit het lijstje. Bij TW3-TW128-TW+2 kan nog veel meer aangepast worden. Die programma's hebben niet alleen de 96 ASCII-tekens (32-127), maar ook de tekens 0-31 en 128-255 hebben een pixelpatroon. CHR$ 0 heeft een dubbele funktie:

- inverse "=" in de hulpschermen
- regelscheiding in de tekstbuffer

Alle andere tekens (CHR$ 1 t/m CHR$ 255) kunt u als volgt in een t- od d-file zetten (een demo-tekst dus):

```
10 LET a=4e4:
   FOR f=1 TO 255:
     POKE a,f: POKE a+1,0: LET a=a+2: NEXT f
   SAVE *1;"t TW3set"CODE 39999,510                    [t-file]

10 OPEN #3;"m";1;"d TW3set":
   FOR f=1 TO 255:
     LPRINT CHR$ f;CHR$ f AND f=13'CHR$ 10;: NEXT f:
   CLOSE #3                                            [d-file]
```

Zorg dat na het LOADen regelnummer en ASCII-CODE overeenkomen. U ziet dan ook CHR$ 1 t/m CHR$ 31 en de '2nd character set'. Merk op, dat na EM + T alle tekens ook een 32-koloms vorm hebben.

—

Enfin, u LOADt dus een andere tekenset op het beginadres. Daarna staan de koppen boven de hulpschermen (1-28) nog in de oude set. De volgende regels lossen dat op:

```
100 LET a=38400  [TW3]     of    LET a=54528  [TW128 en TW+2]
101 FOR f=a-208 TO a-17:
      POKE f,15-PEEK (f+472): NEXT f
```

U wilt misschien ook nog wat regenachtige vakantiedagen besteden aan het aanpassen van de '2nd character set', zodat die overeenkomt met de mogelijkheden van uw printer. Een fonteditor is dan heel handig, hoewel de meeste editors alleen met de CHR$s 32-127 werken. Met wat vernuftig rekenwerk moet dat toch kunnen lukken.

Voor TW3 en TW128/+2 staat een handige fonteditor op DUCDISK-20, gemaakt door de heer Grunefeld. Daarmee kunt u ook de 32-koloms tekens van de '2nd character set' veranderen. Na aanpassing van de LOAD- en SAVE-opdrachten werkt het op elk ander disksysteem.

—

Over DE KLERK kan ik niets zeggen. Ik heb een 'gebruikerskopie', geen handleiding. Die handleiding schijnt wel goed te zijn. Misschien staan er aanwijzingen in om het programma naar wens in te richten. Wie wil over dit onderwerp eens een stukje schrijven?

COMPACT_1 is niet de enige vorm om een 64-kolomsset geheugenbesparend op te slaan. Hieronder staan nog twee vormen.

```
-##-###- 100
#--##-#- 154
#######- 254
##-##-#- 218
#-###-#- 186
-#------ 64
```

In Sinclair User no.57 (dec '86) vond ik een 64-kolomsroutine met deze opslagvorm voor de tekens. Hiernaast staan CHR$ 64 en CHR$ 65: 'apestaartje' en "A". Ook valt op dat de tekens maar 6 pixels hoog zijn. Op deze manier passen wel 32 regels op het scherm. Dat zijn dus 2048 tekens op het gehele scherm!

Deze routine is dus handig om veel informatie in een keer op het scherm te zetten, maar wat onhandig in het gebruik. En langzaam.

```
-------- 0
###----- 224
#-#----- 160
#-#----- 160
###---## 227
#-#---## 163
#-#---## 163
------## 3
```

Op ongeveer dezelfde manier is de tekenset opgeslagen in het PD-programma "64cpl", door de Zwitser Hans Jorg Rothenberger. Hier staan ook twee tekens naast elkaar. Nu niet opeenvolgend, maar met een 'sprong' van 67: CHR$ 65 EN CHR$ 132 ("A" en GRAPHIC 4). De reden hiervoor is ongetwijfeld snelheidswinst.

Deze 64-kolomsroutine ondersteunt niet de normale BASIC-opdrachten (PRINT, AT, TAB), maar wordt aangeroepen met de USR-functie. Het voordeel is de snelheid (sneller dan de ROM!). Het programma (incl handleiding en fonteditor) is verspreid onder BetaDisk-gebruikers. Wie zou de BASIC eens willen aanpassen? Via DUCDISK of IMPULSOFTcassette kunnen ook anderen de routine gaan gebruiken.

—

```
-------- 0
-###-### 119
-#-#-#-# 85
-#-#-#-# 85
-###-### 119
-#-#-#-# 85
-#-#-#-# 85
-------- 0
```

En wat dacht u hiervan? Tweemaal dezelfde "A". De reden is snelheid. Deze tekenset vond ik in 'Firescroll', een Engels terminalprogramma voor de VTX5000. Deze tekenset is 96 * 8 = 768 bytes lang en is een apart CODE-blok: "64chrset" op adres 29000. Dit blijkt het basisadres te zijn, het beginadres ligt dus 256 bytes hoger. De tekenset is gemakkelijk te veranderen.

LOAD eerst een LOW NIBBLE tekenset op adres 29256 en geef dan:

```
FOR f=29256 TO 30024: POKE f,PEEK f+PEEK f*16: NEXT f
```

—

```
-------- 0
###-###- 238
#-#-#-#- 170
###-###- 238
#-#-#-#- 170
#-#-#-#- 170
-------- 0
-------- 0
```

In een ander terminalprogramma voor VTX5000, "Dr Scroll", wordt eenzelfde opslagvorm voor de tekenset toegepast, maar bij het printen worden het eerste en laatste byte genegeerd. Het resultaat is dus dat de tekens 6 pixels hoog zijn en dat er 32 schermregels zijn. Ik moet soms wel een beetje turen, maar in dit programma is het een goede keuze geweest.

Het beginadres van de set in "Dr Scroll" is 64509, de lengte is 768 bytes. De set is dus met een gewone fonteditor te wijzigen.

—

Hieronder staan HEXDATA-regels voor een uitstekend leesbare 64-koloms tekenset. De opslagvorm is natuurlijk COMPACT_1, want dat neemt ook op papier minder ruimte in. Deze set kunt u direkt in bv CHAIN opnemen, waarna u het volledige CODE-blok kunt SAVEn.

Als u deze tekenset in bv TASWORD 2 wilt opnemen, dan moet u er eerst een LOW NIBBLE set van gemaakt worden. Dat kunt u doen met het programmaatje "ln><cptl" van een paar pagina's terug.

—

De 64-koloms tekenset "char64cptl"CODE 50000,384 - EdW - (type COMPACT_1)

```
  1  "00 00 00 00.02 22 20 20.05 50 00 00.05 75 57 50"   474
  2  "02 74 71 72.05 12 24 50.02 52 54 30.02 40 00 00"  1240
  3  "02 44 44 20.02 11 11 20.00 52 72 50.00 22 72 20"  1934
  4  "00 00 02 24.00 00 70 00.00 00 02 20.01 12 24 40"  2237
  5  "02 55 55 20.02 62 22 70.02 51 24 70.06 16 11 60"  3059
  6  "01 35 57 10.07 46 11 60.03 46 55 20.07 12 24 40"  3721
  7  "02 52 55 20.02 55 31 60.00 22 02 20.00 22 02 24"  4294
  8  "00 12 42 10.00 07 07 00.00 42 12 40.02 51 20 20"  4703
  9  "00 69 35 70.07 55 75 50.06 56 55 70.07 54 45 70"  5823
 10  "06 55 55 60.07 46 45 70.07 46 44 40.07 54 75 70"  6882
 11  "05 57 55 50.07 22 22 70.03 11 15 70.05 56 55 50"  7735
 12  "04 44 45 70.05 77 55 50.07 55 55 50.07 55 55 70"  8823
 13  "07 55 74 40.07 55 55 61.07 55 65 50.07 47 15 70"  9853
 14  "07 22 22 20.05 55 55 70.05 55 55 20.05 55 77 50" 10743
 15  "05 52 25 50.05 55 22 20.07 12 24 70.06 44 44 60" 11514
 16  "04 42 21 10.03 11 11 30.02 72 22 20.00 00 00 0F" 11915
 17  "02 54 F4 F0.00 71 75 70.04 47 55 70.00 75 45 70" 13397
 18  "01 17 55 70.00 75 74 70.03 27 22 20.00 75 57 17" 14298
 19  "04 47 55 50.02 06 22 70.01 03 11 57.04 55 65 50" 15070
 20  "06 22 22 30.00 57 55 50.00 75 55 50.00 75 55 70" 16040
 21  "00 75 57 44.00 75 57 11.00 75 44 40.00 74 71 70" 17123
 22  "02 27 22 30.00 55 55 70.00 55 55 20.00 55 57 50" 17982
 23  "00 55 25 50.00 55 57 17.00 71 24 70.03 24 22 30" 18761
 24  "02 22 22 20.06 21 22 60.05 A0 00 00.06 9B B9 60" 19639
```

Mocht u het intoetsen van deze DATA-regels te veel werk vinden: de tekenset is ook te downloaden uit SINCLAIR BOX (01670-66845), maar dan als LOW NIBBLE set. (Die is immers het meest gangbaar.)

Het zou me niet verbazen als er nog meer opslagvormen zijn voor 64-kolomssets. Voor op- en aanmerkingen houd ik mij aanbevolen.

—

Kees Versluis - Copernicuslaan 25 - 2561 VA Den Haag

Eerder kreeg u het programma "odpeekpoke". Toen daar een versie van gemaakt was die naar keuze ook hexadecimaal werkt, bleek die in een behoefte te voorzien. Daarom wil ik u die niet onthouden.

—

HET BASICPROGRAMMA "odpphex" LINE 1 - EdW

```
  1>DEF FN H$(B)="0123456789ABCDEF"(B/16+.5)+
                 "0123456789ABCDEF"(B-INT (B/16)*16+1)
  2 CLEAR #: POKE 23658,8:
    INPUT "1 PEEK A" ,"4 POKE A,G",
          "2 DPEEK A","5 DPOKE A,G",
          "3 PEEK A>","6 POKE A,G$"''
             LINE K$ ,"A: "; LINE G$
  3 GO SUB 60: LET A=G:
    OPEN #3;"CODE ":POINT #3;A:
    GO TO CODE K$ OR K$<"1" OR K$>"6"

 49 PRINT #0;"PEEK ";A;"=";CODE INKEY$#3: PAUSE 0: RUN
 50 PRINT #0;"DPEEK ";A;"=";CODE INKEY$#3+CODE INKEY$#3*256:
    PAUSE 0: RUN
 51 LET G=CODE INKEY$#3: RANDOMIZE A:
    PRINT A;",";G;TAB 11;FN H$(PEEK 23671);FN H$(PEEK 23670);
          ",";FN H$(G);"  ";CHR$ G AND G>32:
    LET A=A+1: GO TO 51 OR INKEY$="0"
 52 INPUT "G(0-255): "; LINE G$: GO SUB 60: LPRINT CHR$ G;:
    POINT #3;A: GO TO 51
 53 INPUT "G(0-65535): "; LINE G$: GO SUB 60: RANDOMIZE G:
    LPRINT CHR$ (PEEK 23670 AND G);CHR$ (PEEK 23671 AND G);:
    POINT #3;A: GO TO 51
 54 INPUT "G$: ";G$: LPRINT G$;:POINT #3;A: GO TO 51

 60 IF G$>"/" THEN LET G=VAL G$: RETURN: REM decimaal
 61 LET G=CODE G$(2)*16+CODE G$(3)+(112 AND G$(2)<"A")+
         (7 AND G$(3)<"A")-935: IF LEN G$<5 THEN RETURN
 62 LET G=G*256+CODE G$(4)*16+CODE G$(5)+(112 AND G$(4)<"A")+
         (7 AND G$(5)<"A")-935: RETURN
```

—

Extra's ten opzichte van "odpeekpoke" uit IMPULS 81-33:

Het adres A kunt u hexadecimaal intoetsen door er een #, punt of spatie voor te zetten (ieder teken onder / , zie SUBroutine 60). U kunt daarna of 2 of 4 hexadecimale cijfers intoetsen, waarbij voor de cijfers A t/m F wel hoofdletters gebruikt moeten worden. Hetzelfde geldt bij invoer van het getal G bij de opties 4 en 5.

Optie 3 toont de paren die bestaan uit adres en inhoud nu niet slechts decimaal, maar daarachter ook nog een keer hexadecimaal. De opties 4 en 5 worden nu ook automatisch gevolgd door optie 3.

—

E H F Weijgers - Wilhelminalaan 42 - 2625 KH Delft

HET MERGE-PROGRAMMA "listvars0"

```
  0>SAVE *"m";1;"Cvars"CODE PEEK 23627+PEEK 23628*256,
      PEEK 23641-PEEK 23627+(PEEK 23642-PEEK 23628)*256:
    LOAD *"m";1;"listvars1"
```

HET BASIC-PROGRAMMA "listvars1" LINE 88

```
  1 DEF FN P(A)=PEEK A+PEEK (A+1)*256: IF PEEK A=128 THEN STOP
  2 LET T=INT (PEEK A/32): PRINT T;" ";CHR$ (PEEK A-T*32+96);:
    LET A=A+1: GO SUB T*10: PRINT : GO TO 1

 20 PRINT "$=""";
 21 FOR A=A+2 TO A+1+FN P(A): PRINT CHR$ PEEK A;: NEXT A:
    PRINT """";: RETURN

 30 PRINT "=";
 31 LET T=FN P(23637)+33-A
 32 FOR A=A TO A+4: POKE A+T,PEEK A: NEXT A: PRINT 0;: RETURN

 40 PRINT "(";FN P(A+3);: LET T=A+2+FN P(A)
 41 FOR A=A+5 TO A+1+PEEK (A+2)*2 STEP 2:
      PRINT ",";FN P(A);: NEXT A: PRINT ")";: LET A=T: RETURN

 50 IF PEEK A<128 THEN PRINT CHR$ PEEK A;: LET A=A+1: GO TO 50
 51 PRINT CHR$ (PEEK A-128);: LET A=A+1: GO TO 30

 60 PRINT "$";: GO TO 40

 70 GO SUB 30: PRINT ," TO ";: GO SUB 31: PRINT '"? STEP ";:
    GO SUB 31: PRINT ," NEXT ";FN P(A);":";PEEK (A+2);:
    LET A=A+3: RETURN

 88>CLEAR 26999: LET A=27E3: LOAD *1;"Cvars"CODE A: GO TO 1

 99 FOR A=FN P(23627) TO 9E9: CLS : GO TO 1
```

—

Zet beide programma's op dezelfde schijf en stop die in drive 1 wanneer het programma waarvan u de variabelen wilt zien in uw SP zit. Geef dan MERGE *"m";1;"listvars0": GO TO 0 en de rest gaat vanzelf. Vergeet niet achteraf ERASE "m";1;"Cvars" te geven.

Het variabelengeheugen kan ook rechtstreeks bekeken worden. Geef daartoe MERGE *"m";1;"listvars1": GO TO 99. U ziet de variabelen A en T van dit programma dan ook; ze overschrijven gelijknamige variabelen zelfs (vervang ze zonodig door minder gebruikelijke). In regel 99 wordt A meteen van het FOR-type gemaakt. Als dat pas tijdens de executie gebeurde zou dat de goede werking verstoren!

Let eens op de bijzondere truc van regel 31/32 om FPR's te tonen (zie IMPULS 81-42) en op het gebruik van SUB-routines met GO SUB T*10, dan GO TO / GO SUB 30/31/40 en tenslotte RETURN. - EdW -

-- ALGEMEEN	SP ZXSPECTRUM	DD DISCiPLE
80 ZX80	MD MICRODRIVE	QL QUANTUM LEAP
81 ZX81	OD OPUS DISCOVERY	88 Z88
CR CASSETTERECORDER	BD BETADISK	

—

Nieuw artikel in de HCC bestelservice:

5454702 SINCLAIR RS-232 INTERFACE MET SOFTWARE (CR) 42.50

Voor de volledige lijst verwijzen wij u naar IMPULS 74 pagina 45

—

DE SINCLAIRDAGEN IN HOUTEN

ZATERDAG 10-16 UUR 25 AUGUSTUS 27 OKTOBER

—

DE HCC-MICROCOMPUTERDAGEN

VRIJDAG 23 EN ZATERDAG 24 NOVEMBER 1990 IN DE JAARBEURS, UTRECHT

—

ONDER VOORBEHOUD - BEKIJK STEEDS DE AGENDA IN UW HCC-NIEUWSBRIEF

DE WEG NAAR ONZE SGG-BIJEENKOMSTEN IN HOUTEN

Onze Sinclairgebruikersdagen worden in het vervolg gehouden in:

het HCC-kantoor
Standerdmolen 8 te Houten
telefoonnummer: 03403-78788.

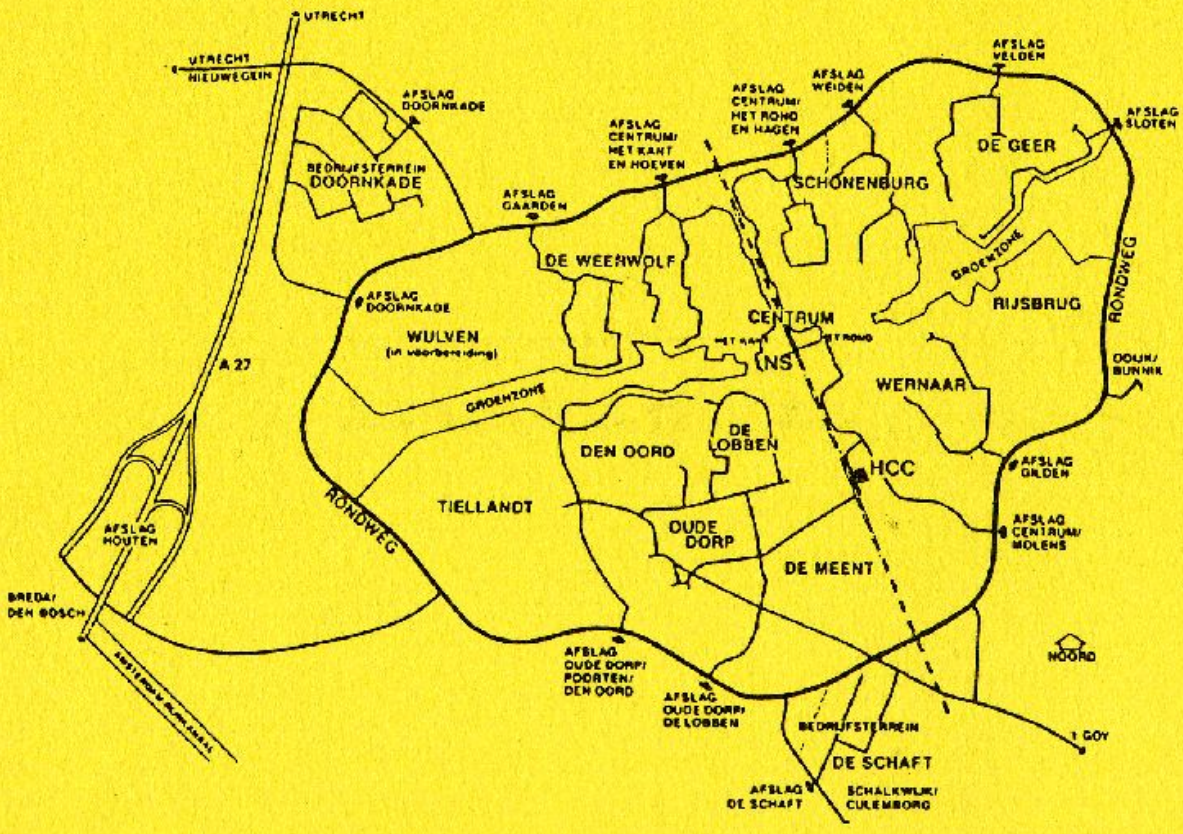
Aanwijzingen voor de automobilisten die komen uit de richtingen

DEN BOSCH	Neem bij Everdingen de afslag Utrecht (A27) en daarna de afslag Houten.
DEN HAAG/ AMSTERDAM	Volg bij Utrecht de richting Arnhem, dan eerst de richting Houten, maar dan de richting Breda en neem de afslag Houten.
ARNHEM	Neem bij Bunnik de afslag, volg even de richting Wijk bij Duurstede en bij Odijk de richting Houten.
HILVERSUM	Volg de richting Den Bosch en neem dan de afslag Houten.

Volg in Houten de rondweg (linksom of rechtsom maakt niets uit). Neem de afslag Centrum/Molens, het HCC-kantoor is 300 m verder. Daar is voldoende parkeergelegenheid, vooral aan de achterzijde.

Aanwijzingen voor de treinreizigers:

Volg bij het verlaten van het station de spoorbaan, niet in de richting van het winkelcentrum, maar de andere kant op. Dan ziet U al direct het grote kantoorpand waarin de HCC is gehuisvest.



SINCLAIR IMPULS

Postbus 76
2260 AB Leidschendam

PORT BETAALD
PORT PAYE
DEN HAAG