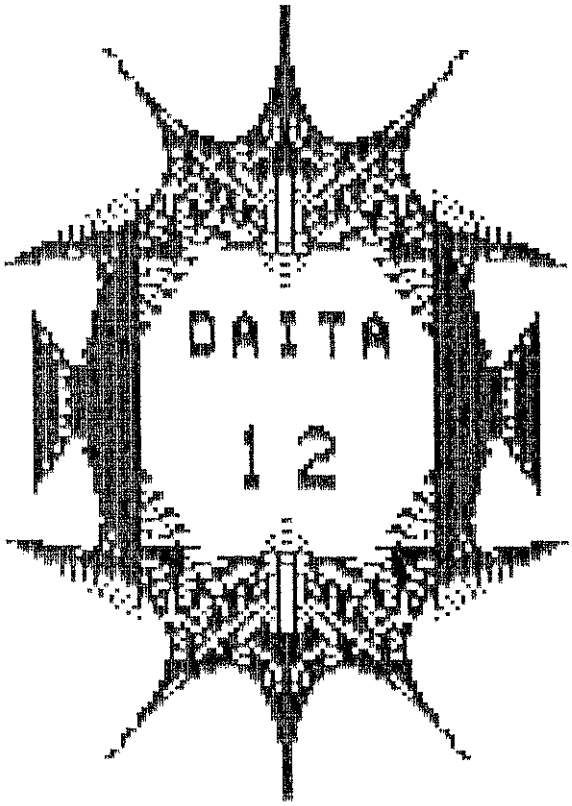
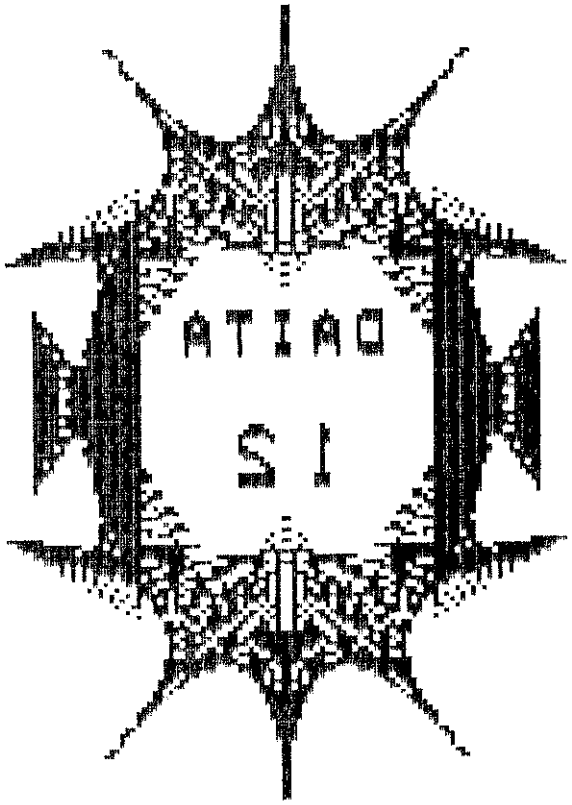
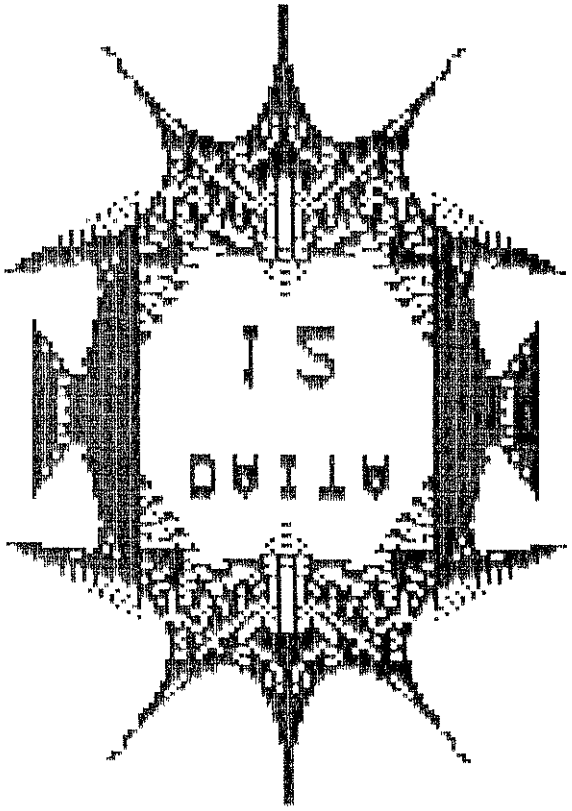
DAITA 12

UITGAVE VAN DE HCC–DAI GEBRUIKERSGROEP







INHOUD DAITA 12

****

ALGEMEEN

HARDWARE

SOFTWARE

PROGRAMMA's

****

ADRESSEN

Voorzitter :Kees Jagerman, Ilperveldstraat 77, 1024 FJ Amsterdam,
Tel. 020-367156

Secretaris
+ redactie DAITA :Wim Bremer, Vleugeltjesbloem 21, 1902 GH Castricum,
Tel. 02518-51878

Coordinator NO :Robert van den Broek, Melde 19, 8265 CP Kampen,
Tel. 05202-17131

Coordinator Zuid
+ Hard- en Soft-
ware-service :Theo Verberkt, Van Buerenstraat 13, 5256 KL Oudheusden,
Tel. 04162-2667

****

VAN DE REDACTIE

Deze keer konden wij ons verheugen in zoveel kopij dat wij een selectie hebben moeten maken en een deel te reserveren voor DAITA 13.
Zo zullen de assembler-programma's behorende bij het COLLECTOR-programma van Ton van Es en Anton Doornenbal niet in deze DAITA worden meegenomen. Wie het echter alvast wil gebruiken krijgt daarvoor in de plaats het machinetaalprogramma.
Inmiddels blijkt ook dat ingestuurde bijdragen weer reacties oproepn bij andere lezers. Wij geloven dat we daar mee op de goede weg zijn.
We hopen dat het verschuiven van plaatsingen niet zal leiden tot vermindering van inzendingen. Overigens, wij vinden dat sommige inzenders onvoldoende gebruik maken van de mogelijkheid om listings en/of teksten met een kleinere letter uit te printen, waardoor de bladspiegel beter benut kan worden. Zie daarvoor het stukje van Jan Blokker in dit nummer.
De gewenste bladspiegelmaten zijn: hoog 27 cm en breed 17,5 cm.

Inmiddels heeft de DAI-gg de beschikking gekregen over een eigen fotocopieer-apparaat waardoor het mogelijk is DAITA geheel in eigen beheer te vermenigvuldigen. Tot dusver waren wij afhankelijk van welwillendheid van grote bedrijven, waar DAITA dan tussendoor voor een schappelijk prijsje kon worden geproduceerd. Het nadeel echter was dat we nooit de zekerheid hadden dat het blad op tijd klaar zou zijn en dat nabestellingen ook altijd vertragingen gaven. Ofschoon er op het zelf maken geen loonkosten drukken is het toch niet voordeliger om het zelf te doen. Grofweg moeten we rekenen met een paginaprijs van vijftien cent. Er van uitgaande dat HCC een deel van de afschrijving op de machine voor zijn rekening neemt, moeten we de prijs voor een DAITA van ca. 30 pagina's voorlopig op f 3,00 stellen. We hopen deze prijs tot einde 1986 te kunnen handhaven. Voor de postabonnee's zal er mogelijk nog een verhoging nodig zijn i.v.m. de verhoging van de PTT-tarieven.

Van het Belgisch front is er helaas geen nieuws te melden. Het bestuur zal dezer dagen trachten weer een afspraak met de fabrikant te arrangeren.

Inmiddels hebben wij "de Bron" weer voor 4 zaterdagen in 1986 gehuurd en wel op resp. 8 maart, 7 juni, 20 september en 13 december.
Onze eerstvolgende bijeenkomst is op 7 december 1985.

Voor de HCC-MICROCOMPUTERDAGEN (22 en 23 november) hebben wij twee kramen gehuurd en wel zo dat wij DAInamic als overbuur krijgen, dus een concentratie van de DAI.

GEVRAAGD: EEN NIEUW TOETSENBORD

Voor een reparatie heeft Theo Verberkt een splinternieuw toetsenbord nodig. Helaas kan de fabriek, noch Microshop Hageland dat leveren. Wie dat wel kan wordt verzocht zich met Theo in verbinding te stellen.

GEVRAAGD: CREATIVE COMPUTING 1984

Kees van Dijk (Tel. 05202-18780) zoekt naar een artikel van David Ahl over de transcriptie naar BASIC van een grafisch programma (PEANO of HILBERT).
Als iemand hem het betreffende nummer zou willen uitlenen, zal hij dat zeer op prijs stellen.

KOPIJ voor DAITA 13

Graag insturen voor 23 november (afgeven bij DAI-gg-Kraam in Utrecht).

Versiering voorpagina: Grafische toepassing van Kees van Dijk.

"HIGH SPEED DATA LOADER"

De High Speed Data Loader werd enige jaren geleden door Hein Kop en mijzelf ontwikkeld om,via de DCE bus, veel gebruikte programma's die we in een Eprom bank geprogrammeerd hadden op een snelle manier in de DAI in te lezen.
Het geheel werd opgebouwd rondom een Epromkaartje dat door Elektuur printservice op de markt werd gebracht en vier Eproms van het type 2716 (4x2kByte)of 2732 (4x4kByte) kon bevatten.
Het printkaartje kon voorzien worden van een 32 pins connector waardoor het kaartje op een basisprint voorzien van de overige elektronika geplaatst kon worden.
Het resultaat was dat we een programma van 10kByte in een seconde konden inlezen.
Dit was vooral voor SPL en dergelijke ideaal.
Ook diverse spelletjes werden op deze wijze opgeslagen en vooral voor kinderen was dit geweldig.
Kaartje er in,een commando en acrobates of packman was geladen.
Helaas door de hoge kosten van Eproms in die tijd is van een verdere bouw dan twee prototypes nooit wat gekomen.
Een ander nadeel is dat het Elektuur kaartje alleen maar 2716 of 2732 Eproms kan bevatten.
Toen wij aan het Commodore 1541 project begonnen werd besloten om de software welke voor de aansturing van de Data Loader nodig was hierin op te nemen vooral omdat gedurende de ontwikkeling hiervan veel gebruik werd gemaakt van SPL.
Gezien de vragen welke iedere keer weer komen over de HSL hebben we deze weer eens doorgespit en alles wat we kwijt konden b.v. Power On initialisatie eruit gesloopt omdat deze nu op onze Floppy Interface kaart zit.
Het resultaat is opgetekend in bijgaand schema.
Het is niet onze bedoeling om een basis printplaat hier voor te maken,tenzij de vraag zo groot wordt dat we wel moeten.

TECHNISCHE BESCHRIJVING.

Voor de adres selectie hebben we PB4 gebruikt en zodanig met behulp van een 74LS138 uitgecodeerd dat de HSL alleen dan actief is als dit bit hoog is.
Het is mogelijk om vanaf deze 74LS138 andere selectie signalen te halen voor andere periferie afhankelijk van de aangeboden adres lijnen.
De adressering van de Eproms geschiedt d.m.v twee 74LS393 welke een klok puls ontvangen van PC0 en een reset signaal van PC1.
Beide DCE bus signalen worden twee maal geinverteerd door een 74LS14.
Dit is gedaan om de klokpuls een redelijke puls vorm te geven die door een te lange "flatkabel" verre van ideaal is.
Dit is ook de reden waarom de twee C's van 1nF in deze twee leidingen zijn opgenomen. In sommige gevallen kan het zelfs wenselijk zijn om in iedere leiding een weerstand van 390 ohm in serie op te nemen.
De data lijnen worden extra gebufferd door een 74LS245

welke alleen aktief is als de HSL geselekteerd is en de DCE bus vrij laat voor de overige periferie.
De selectie van de Eproms worden gerealiseerd door het gebruik van een 74LS157 en een 82S23 Prom.
Door deze combinatie is het mogelijk om via PB2 en PB3 een Eprom selectie door te voeren en de mogelijkheid om een programma over meerdere Eproms te verdelen en deze zonder onderbreking in te lezen.
Voorbeeld:
Eprom 1 (2732) bevat een Spooler programma.
Eprom 2,3 en 4 (2732) bevatten SPL.
Door het commando HSL0 in te geven wordt het Spooler programma ingelezen wat 1 Eprom omvat.
Met het commando HSL1 wordt SPL ingelezen,dit programma omvat 3 Eproms welke nu na elkaar ingelezen worden.
Geeft met het commando HSL2 dan krijgt men de melding "NOT AVAILABLE".
Ruimte welke over is aan het einde van een programma in een daarvoor gebruikte Eprom kan helaas niet gebruikt worden voor een ander programma daar we niet over een programmeerbare adres counter beschikken.
Voorafgaande aan ieder programma dat in Eprom komt moet de volgende informatie staan:

1e byte:	#FD	Dit is een niet gebruikte Opcode. Dit geeft een NIEUW Programma aan.
2e en 3e byte:	Start adres van het programma in het DAI geheugen .Lowbyte-Highbyte.	
4e en 5e byte:	Lengte van het programma.Highbyte-Lowbyte.	

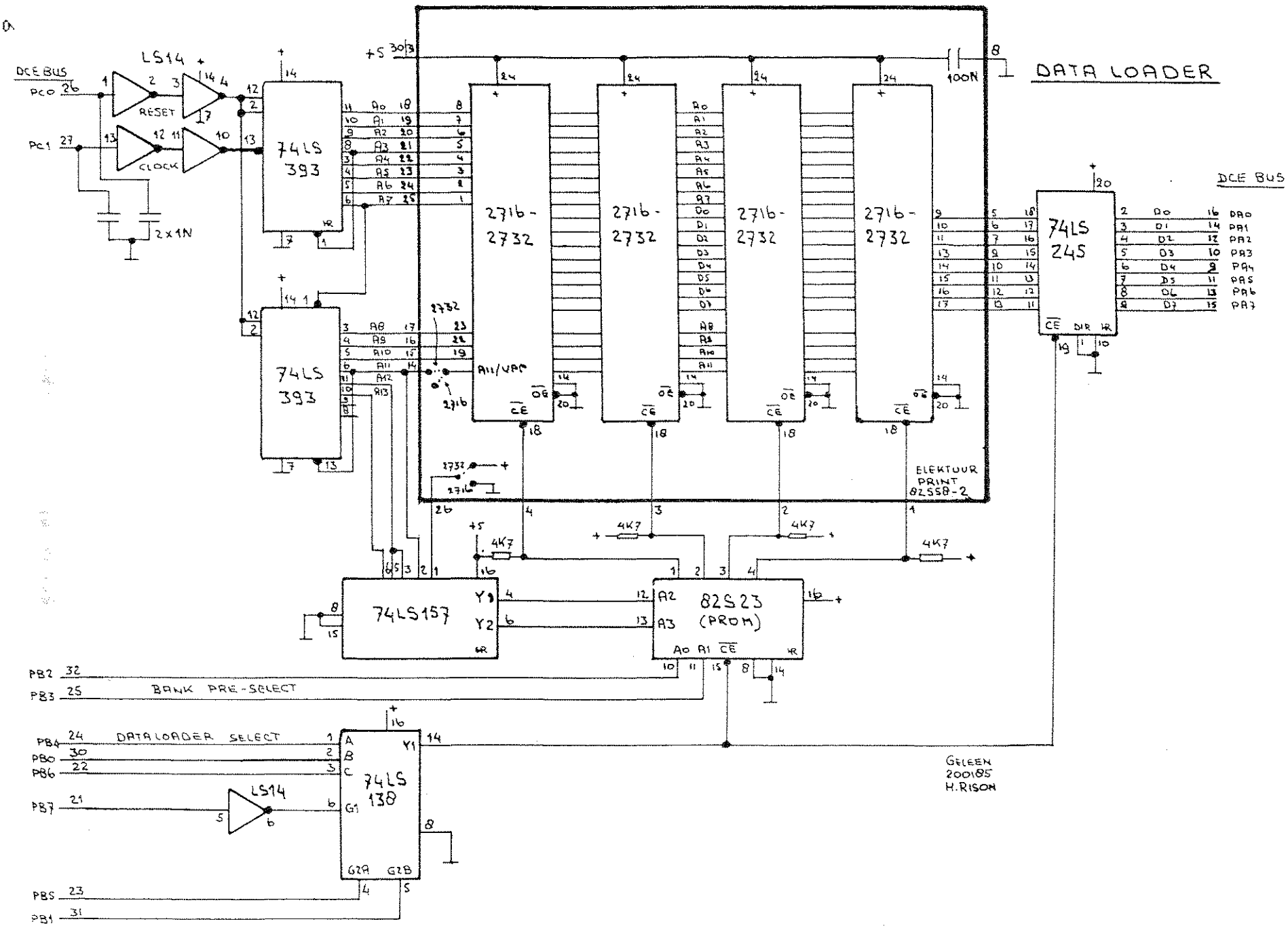
De inhoud van de gebruikte Prom type 82S23 is het volgende:

Adres:	0:	1:	2:	3:	4:	5:	6:	7:	8:	9:	A:	B:	C:	D:	E:	F
Inhoud:#	E:	D:	B:	7:	D:	B:	7:	F:	B:	7:	F:	F:	7:	F:	F:	F

Het Eprom kaartje was verkrijgbaar bij Elektuur printservice,Postbus 75, 6190AB Beek(L) NL onder nr.EPS 82558-2
De connector hiervoor is verkrijgbaar in vrijwel iedere goede Elektronika zaak.

Dit is de manier waarop de High Speed Loader door ons is gebouwd.
Variaties en veranderingen c.q verbeteringen zijn mogelijk daarvan zijn we overtuigd.
Zo hebben we op papier een uitvoering waarin Eproms van de types 2716,2732,2764 en27128 kunnen plaatsen.
Daarvoor moet echter eerst een insteekprint geplakt en gemaakt worden welke met de nu gebruikte connector kan worden aangesloten.
Wie ons daarbij kan helpen wordt vriendelijk verzocht contact met mij op te nemen.
Het systeem in de huidige vorm kan maximaal 4 programma's bevatten.
Voor de uitbouw naar meerdere programmma's is een aanpassing van de DOS software noodzakelijk.

H.Rison
Luxemburgstraat 17
6164 BS Geleen


DATA LOADER
DCE BUS
LS14
RESET
CLOCK
2x1N
74LS 393
74LS 393
2716 - 2732
2716 - 2732
2716 - 2732
2716 - 2732
100N
74LS 245
DCE BUS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
2732
2716
A11/VPP
ELEKTUUR PRINT 82558-2
4K7
74LS157
82S23 (PROM)
PB2
PB3
BANK PRE-SELECT
PB4
PB0
PB6
DATALOADER SELECT
PB7
74LS 138
PB5
PB1
GELEEN 200185 H.RISON

HET REINIGEN VAN DE PRINTERKOP door Gordon Wassermann

Bezit u een Epson FX-80 of een andere Epsonprinter met een uitwisselbare printkop?

En heeft u al meegemaakt, dat een naald niet meer, of slechts zeer zwak drukt? Dat schijnt helaas niet zo weinig voor te komen.

Als het bij uw printer zo is, dan hoeft u waarschijnlijk niet meteen een nieuwe printerkop te gaan kopen. Ik kon in ieder geval na een reiniging mijn printer weer zo aan de gang krijgen, dat hij al nieuw aan het printen is geslagen.

Er is echter een ding zeker: u moet na de reiniging de naalden weer invetten. Doet u dat niet, dan wordt het schriftbeeld eerder slechter, dan beter. Want de schoonmakerij verwijdert niet alleen alle inktresten, die de naalden in hun beweging hinderen kunnen, maar ook de van fabriekswege aangebrachte oliefilm, zonder welke ze nog meer bewegingshinder zouden ondervinden.

Dat kan zelfs zo ver gaan, dat de naalden zich in het inktlint vasthaken en de printkop onder een zeer onaangenaam krijsende herrie tot stilstand gebracht wordt. Dit vormt overigens ook de reden, waarom in de handleiding gewaarschuwd wordt voor reiniging met oplossende middelen.

Wat dat handboek echter verzwijgt -men hoopt nieuwe printerkoppen te verkopen- is de mogelijkheid, de opgeloste olie te vervangen.

Hier volgt de schoonmaakklus in het kort:

1. Verwijder de lichtnetstekker, en het inktlint en de printerkop volgens de gegevens in het handboek. U hoeft de lintkabel echter niet uit de stekkerbehuizing te halen.
2. Verwijder met een wattenstokje met ISOPROPYLALCOHOL grondig de inktresten van de naalden. De naalden worden in de printerkop via via een geleidingswand naar de kop van de printerkop gebracht. De plaats waar ze naar buiten komen, en de plaats die daar direct aan de binnenkant gelegen is, vormen de plekken die het best gereinigd moeten worden, omdat zich daar de meeste inktresten verzameld hebben. Isopropylalcohol is bij apotheken verkrijgbaar en heeft als voordeel dat het (praktisch) geen verontreinigingen bevat en die dus ook na verdamping niet achterlaat.
3. U laat de alcohol verdampen en geeft daarna zowel op de neus van de printerkop als daarbinnen bij die geleidingswand een druppel naaimachine-olie op de naalden.
4. De printerkop kan nu gemonteerd worden, en ook de netstekker kan er weer in. Nu gaan we zonder inktlint maar met papier de zelftest van de printer uitvoeren, om de overbodige olie weg te krijgen. Stop er nu een oud inktlint in, en voer weer een aantal pagina's de zelftest uit. Nu zuigt dat oude inktlint de laatste olieresten op. Deze maken zich bekend, door vettig afgedrukte letters, en als die niet meer voorkomen in de zelftest, dan kunt u deze procedure na een of twee pagina's stoppen.
5. Ze nu het goede inktlint erin. In de komende tijd kan er nog wat olie zichtbaar worden bij de eerste afgedrukte regels, maar na een paar keer is ook dat voorbij.
6. Nu kunt u rustig uw lezenswaardige listing of artikel opsturen aan de redactie van DAITA.

Ik hoop, dat deze tips voor printerbezitters de kosten voor een nieuwe printerkop zullen uitsparen.

(vert.+bew: I.Broekman)

Noot: Isopropylalcohol is licht ontvlambaar. Dus niet roken of gebruiken in de buurt van open vuur!!!

BLOTEKST 11d

Sinds ik eind vorig jaar FWP in handen kreeg, heeft deze uitstekende tekstverwerker van Ger Gruiters mij nauwelijks meer losgelaten. Het programma zit uitstekend in elkaar en de even-eens prima handleiding heb je nauwelijks nodig. Het steeds weer terugkerende probleem voor mij was, dat het programma al-leen geschikt was voor DCR en onbruikbaar voor het werken met mijn floppy, zoals bijna elk programma, dat ik aanschaf. Ik ben gewend, om dan zelf maar aan de slag te gaan en zo ook deze keer. Na verloop van tijd ont-stond zo mijn eigen versie van deze tekstverwerker: DOSFWP.

Ten opzichte van de oorspronkelijke uitvoering waren toen de volgende wijzigingen aangebracht:

1 Universeel voor DOS/DCR.
De tekstverwerker is nu zo opge-zet, dat hij zich instelt op het aanwezige achtergrondgeheugen, hetzij floppy, cassette of MDCR. Er kan tijdens het werken worden omgeschakeld. De gemaakte keuze wordt bij SAVE en LOAD aangege-ven.

2 ORG 2400.
In verband met het DOS heb ik voor een ORG van 2400 gekozen.

3 Alle files worden geladen.
De oorspronkelijke FWP had een vast inlaadadres. Bij deze uit-voering worden ook oude files van FWP met welk ORG-adres dan ook, ingeladen.

4 Onbeperkte header.
Alle waarden van 0...FF kunnen in de header worden gebruikt; waarden boven 80 kunnen alleen via UT worden ingevoerd. Op de-ze manier zijn in combinatie met een daartoe geschikte printer grafische tekens mogelijk.

5 Geen invloed headers.
Wanneer headertekens worden ge-bruikt om naar *cursief* of **dub-bel** schrift over te gaan, worden die niet meegeteld bij de bepa-ling van de tabs en bewerkin-gen, zoals *width equal*.

6 16 tab-posities.
Het aantal mogelijke tabulator-posities heb ik van 10 naar 16 gebracht.

7 Filetype +.
Om tekstverwerkerfiles te onder-scheiden van andere files heb ik voor het filetype + gekozen. Oude files van type 1 kunnen echter ook worden verwerkt.

8 Vermelding sectoren.
Ten behoeve van het werken met de floppy wordt in het menu het aantal door de file ingenomen sectoren vermeld.

Bij deze veranderingen heb ik het op dat moment voorlopig ge-laten, totdat ik het artikel van Wim Bremer in DAITA tegenkwam. Het spoorde mij aan, om nog ver-der te gaan met de wijzigingen. Ik had overigens nog wel wat wensen wat dat betreft. Inmid-dels is het volgende er uit ge-komen:

9 4 conversies mogelijk.
Het aantal conversiemogelijkhe-den heb ik tot 4 teruggebracht.

10 Defaultwaarden in file.
Om bij het wijzigen van een be-staande file niet alle instel-lingen weer opnieuw te hoeven doen, worden de belangrijkste waarden met de file meegeladen en worden bij het inladen de waarden automatisch juist inge-steld (ook de tab-posities). Files zonder deze tabel kunnen ook worden verwerkt. Een RESET is ingebouwd.

11 Echte formfeed.
Er kan nu een echte formfeed worden ingesteld.

12 Linefeed instelbaar.
De regelafstand is instelbaar binnen zekere grenzen.

13 Printerbesturing EPSON.
De printerbesturing is gebaseerd op de standaard EPSON-escape co-des.

14 Formlength instelbaar.
De formlength kan nu worden in-gesteld op elk gewenst aantal regels.

15 Paginanummering bovenaan.
De paginanummering wordt nu ge-plaatst op de kop van het blad op elke gewenste plaats op de regel.

16 Titel op elk blad.
In combinatie met de paginanum-mering kan een titel worden af-gedrukt.

17 Uitgang RS232c.
De printerbesturing is gebaseerd op de standaard serie-interface volgens RS232c. De besturing is in het programma ingebouwd.

18 ☞ Unit form.
Elke tekst kan op formaat A4 worden geprint, met dien ver-stande dat de regelafstand au-tomatisch zo wordt gekozen, dat de tekst het formaat A4 vult. Deze tekst is zó geprint.

19 Tekst in 2 kolommen.
De tekst kan over 2 kolommen worden verdeeld. Het programma verzorgd eventueel automatisch de juiste tab-positie van de beide kolommen.

20 Programma-grootte.
Ondanks deze wijzigingen en aan-vullingen is de grootte van het programma niet gewijzigd. De plaats van de headerbuffer is dus dezelfde gebleven.

Ik heb de lay-out van de menu's enigszins gewijzigd en sommige teksten wat ingekort om ruimte voor mijn toevoegingen te krij-gen. Het programma heeft mijns inziens daardoor aan hanteer-baarheid niets hoeven inboeten. Op het ogenblik wordt deze uit-gebreide tekstverwerker getest en er zullen nog wat aanpassin-gen moeten plaatsvinden, met na-me voor wat betreft de combina-tiemogelijkheden, zoals bijvoor-beeld Unit form / kolommen. Zo-dra het programma geen fouten meer vertoont en alle mogelijke vergrendelingen tegen verkeerd gebruik zijn aangebracht, zal ik nader berichten.

850813 MSX-DAI-gg J.P. Blokker

```
0000                    TITL      'ZOEKEN IN STRING ARRAY'
0000            ; 1-aug-1985
0000            ;
0000            ; Naar aanleiding van een artikel van Kees v Dijk in DAITA 10
0000            ;
0000            ; In DAITA 10 is het artikel nogal verminkt gepresenteerd.
0000            ; Hieronder vind U de listing van het programma
0000            ; met hier en daar enige wijzigingen om het programma
0000            ; wat meer flexabiliteit te geven.
0000            ; Verder was een fout geslopen in de vorige listing n.l.
0000            ; de instructie MOV D,A in de routine VOORT.Dit moest zijn
0000            ; MOV A,D.Daar waar het programma gewijzigd is, is dit
0000            ; aangegeven met de tekens ###.
0000            ; Dit om strings groter dan 128 tekens ook te kunnen
0000            ; supporten.EEN uitzondering is dat de dimensie van
0000            ; de string max. 254 mag bedragen.
0000            ; Dit programma staat ook op de "LANDELIJKE DAI-gg DAG TAPE"
0000            ;
0000            ;                                        Theo Verberkt
0000                    ORG       300H
0300            ;
0300  C31903            JMP       RESET
0303  C35703            JMP       ZOEK
0306  @=0012  TL        EQU       12H             ; type en lengte van TT%
0306  5454    VARNAM    DB        'TT'            ; naam van variable TT%
0308  0000    STRING    DW        0H              ; stringvector
030A  0000    AANT      DW        0H              ; aantal vectoren gehad
030C  0000    NUL       DW        0H
030E  0000    INDEX     DW        0H              ; blijft 0
0310  0000              DW        0H
0312  FFFF    FFFF      DW        0FFFFH
0314  FFFF              DW        0FFFFH
0316  0000    MAX       DW        0H              ; max aantal in array
0318  00      END       DB        0H
0319            ;
0319  C5      RESET     PUSH B                    ; BC bewaren voor basic
031A  0A      CALLM     LDAX B
031B  0B                DCX B
031C  FEB3              CPI       0B3H            ; Token voor CALLM
031E  C21A03            JNZ       CALLM
0321  210700            LXI H     7H              ; 7 verder geeft offset
0324  09                DAD B
0325  E5                PUSH H
0326  C1                POP B                     ; via stack in BC
0327  1600              MVI D     0H              ; MACC niet leegmaken
0329  CD5AE9            CALL      0E95AH          ; maak vector
032C  CDEE03            CALL      VCPTR           ; maak pointer in DE
032F            ;                                  aantal vectoren
032F  1A                LDAX D
0330  4F                MOV C,A                   ; aantal DIMS in A
0331  13                INX D
0332  1A                LDAX D
0333  210000            LXI H     0H
0336  220A03            SHLD      AANT            ; nummer van de laatst
0339            ;                                 ; gevonden string in AANTAL
0339            ;                                 ; HL=0
0339  6F                MOV L,A                   ; eerste DIM in L
033A  2C                INR L                     ; rekenen vanaf 1
```

```
033B  13              INX D
033C  0D      KEER    DCR C
033D  CA4903          JZ        K1
0340  1A              LDAX D
0341  13              INX D
0342  3C              INR A                       ; idem
0343  CD8FDE          CALL      0DE8FH            ; HL=HL*A
0346  C33C03          JMP       KEER
0349          ;
0349  221603 K1        SHLD      MAX               ; aantal vectoren
034C  EB              XCHG
034D  220803          SHLD      STRING            ; eerste vector
0350  211803          LXI H     FND               ; reset ' gevonden'-vlag
0353  3600            MVI M     0H
0355  C1              POP B
0356  C9              RET
0357          ;
0357          ; ENTRY VECTOR 2
0357          ;
0357  C5      ZOEK    PUSH B                      ; bewaren voor BASIC
0358  CDEE03          CALL      VCPTR             ; pointer in DE
035B  D5              PUSH D
035C          ;
035C          ; aantal aanpassen; als AANTAL = MAX dan niet gevonden.
035C          ;
035C  2A1603 VOORT    LHLD      MAX
035F  EB              XCHG
0360  2A0A03          LHLD      AANT
0363  23              INX H
0364  CD14DE          CALL      0DE14H            ; vergelijk DE -HL
0367  CAD803          JZ        KLAAR             ; heel array gehad
036A  220A03          SHLD      AANT
036D  D1              POP D                       ; begin weer voor in ZOEK$
036E  D5              PUSH D                      ; bewaren
036F          ;
036F          ; maak van STRING een stringpointer; pas STRING aan (+2)
036F          ;
036F  2A0803          LHLD      STRING            ; stringvector
0372  D5              PUSH D
0373  CDEE03          CALL      VCPTR             ; maak pointer in DE
0376  7A              MOV A,D
0377  B3              ORA E                       ; lege pointer ?
0378  220803          SHLD      STRING
037B  EB              XCHG
037C  D1              POP D
037D  CA5C03          JZ        VOORT             ; dan volgende
0380  4E              MOV C,M                     ; lengte string in C
0381  0D              DCR C
0382  0C              INR C                       ; flags aanpassen
0383  CA5C03          JZ        VOORT             ; lege string
0386  23              INX H
0387  EB              XCHG                        ; stringpointer in DE
0388  46              MOV B,M                     ; lengte ZOEK$ in B
0389  23              INX H
038A          ;
038A          ; Loop hele string door met het eerste zoek karakter:
038A          ;
038A  1A      ZKKAR   LDAX D                      ; karakter in A
```

ZOEKEN IN STRING ARRAY

```
038B  13              INX D                          ; volgende stringKarakter
038C          ;
038C          ; Ten behoeve van het doorlopen van een string > 128
038C          ; is het uitgesloten gebruik te maken van
038C          ; de instructies  JM en JP
038C          ; Dit is verholpen door de verandering aangeduid met ###
038C          ;
038C  BE              CMP M                          ;                      ###
038D  CA9703          JZ        ZKNXT                ; verder in file       ###
0390  0D              DCR C                          ; aantal -1            ###
0391  CA5C03          JZ        VOORT                ; volgende string      ###
0394  C38A03          JMP       ZKKAR                ; verder in file       ###
0397          ;
0397          ; Er is wel een karakter gevonden - zoek verder.
0397          ;
0397  0D      ZKNXT   DCR C                          ;                      ###
0398  C5              PUSH B                         ; onthoud tellers
0399  D5              PUSH D                         ; en pointers
039A  E5              PUSH H
039B  05      ZKVRDR  DCR B                          ; nog meer zoekKarakters ?
039C  CABA03          JZ        VIND                 ; zoekstring gevonden
039F  0D              DCR C                          ; hele string gehad ?
03A0  C2A903          JNZ       ZK2                  ; nee                  ###
03A3  E1              POP H
03A4  D1              POP D
03A5  C1              POP B
03A6  C35C03          JMP       VOORT                ; volgende string
03A9          ;
03A9  23      ZK2     INX H                          ; verder in zelfde string
03AA  1A              LDAX D
03AB  13              INX D
03AC  BE              CMP M
03AD  CA9B03          JZ        ZKVRDR
03B0  E1              POP H                          ; pointers en tellers herstellen
03B1  D1              POP D
03B2  C1              POP B
03B3  C38A03          JMP       ZKKAR
03B6  C21404          JNZ       V1
03B9  C9              RET
03BA          ;
03BA  E1      VIND    POP H
03BB  D1              POP D                          ; nodig om het stackniveau
03BC  C1              POP B                          ; te herstellen
03BD  210A03          LXI H     AANT
03C0  111103          LXI D     INDEX+3H             ; begin achterin met LSB
03C3  7E              MOV A,M
03C4  12              STAX D
03C5  23              INX H
03C6  1B              DCX D
03C7  7E              MOV A,M
03C8  12              STAX D
03C9  1B              DCX D
03CA  1B              DCX D                          ; DE wijst index aan
03CB  211803          LXI H     FND                  ; al eerder iets gevonden?
03CE  3E00            MVI A     0H
03D0  BE              CMP M
03D1  C2E803          JNZ       UIT                  ; ja
03D4  35              DCR M                          ; nee, zet flag
```

```
3D5  C3E803          JMP       UIT
3D8          ;
3D8  3A1803 KLAAR    LDA       FND              ; wel eens iets gevonden?
3DB  B7              ORA A                      ; Z flag aanpassen
3DC  CAE503          JZ        VINDNT
3DF  110C03          LXI D     NUL
3E2  C3E803          JMP       UIT
3E5  111203 VINDNT   LXI D     FFFF
3E8  E1      UIT     POP H
3E9  CDF303          CALL      VAR
3EC  C1              POP B
3ED  C9              RET
3EE          ;
3EE          ; maak pointer in DE van vector in HL; HL=HL+2
3EE          ;
3EE  5E      VCPTR   MOV E,M
3EF  23              INX H
3F0  56              MOV D,M
3F1  23              INX H
3F2  C9              RET
3F3          ;
3F3  C5      VAR     PUSH B
3F4  213501          LXI H     135H             ; EFSW = input from string
3F7  3601            MVI M     1H
3F9  E5              PUSH H                     ; onthoud EFSW
3FA  210603          LXI H     VARNAM
3FD  223201          SHLD      132H             ; EFEPT wijst naar gezochte naam
400  D5              PUSH D
401  1612            MVI D     12H
403          ;
403          ; TYPE/LENGTE type 1=integer lengte NAAM 2 karakters
403          ;
403  0600            MVI B     0H               ; vooraan beginnen
405  CD57CA          CALL      0CA57H           ; zoek naam
408  D1              POP D
409  DC1104          CC        VUL              ; vul waardeveld
40C  E1              POP H
40D  3600            MVI M     0H               ; zet EFSW terug
40F  C1              POP B
410  C9              RET
411          ;
411          ; de inhoud van TT% wordt met vier bytes gevuld die door DE
411          ; zijn aangewezen.
411  C5      VUL     PUSH B
412  0604            MVI B     4H               ; 4 bytes
414  23      V1      INX H                      ; naar dataveld
415  1A              LDAX D                     ; uit DE
416  77              MOV M,A
417  13              INX D
418  05              DCR B
419  C21404          JNZ       V1
41C  C1              POP B
41D  C9              RET
41E          ;
41E          EINDE   END
```

DAI-DOS 1541: RELATIEVE FILES

Dit artikel beschrijft het principe van relatieve files en de manier waarop ze gebruikt kunnen worden door de DAI-DOS 1541 met de Commodore VC1541 disk drive.

PRINCIPE/VOORDELEN VAN RELATIEVE FILES:

Snelle toegang tot elk record - grote files mogelijk
Relatieve files zijn 'random-access files'. De file blijft op de diskette en wordt niet in zijn geheel in het computer RAM geheugen geladen. Dit gebeurt alleen met die gedeeltes, die voor een bewerking nodig zijn.

Het voordeel van dit systeem van file bewerking is de mogelijkheid tot het hanteren van grote files. De DAI kent van huis uit alleen array-files (zgn. 'sequential' files). T.g.v. de maximum 'heap'-grootte, kan een array nooit groter zijn dan 32KB. Voor string-arrays geldt zelfs een maximale grootte van ca. 20KB vanwege de dubbele geheugen ruimte nodig bij het lezen en schrijven. Relatieve files op de VC1541 kunnen max. 164KB bevatten, d.w.z. de omvang wordt beperkt door de opslag capaciteit van de diskette.

Een relatieve file bestaat uit een aantal 'RECORDS'. Een record is een blok informatie. Laten we als voorbeeld nemen een adressenbestand. Voor elke persoon in dit bestand moeten we de naam, het adres en de woonplaats opslaan. Deze 3 gegevens ('FIELDS') vormen samen een RECORD. De records van alle personen samen vormen dan de 'FILE'.

Een ander voordeel is de manier van opslag van gegevens. Een relatieve file is willekeurig toegankelijk (random access). Dit betekent, dat elk record kan worden gelezen/geschreven zonder de hele file eerst te moeten laden. Dit in tegenstelling tot het hanteren van arrays. Hier moet voor elke wijziging het hele array eerst worden gelezen, dan kunnen er wijzigingen worden aangebracht en daarna moet het array weer worden weggeschreven. Dit vraagt dus veel meer tijd.

Om relative files met de DAI-DOS 1541 te kunnen hanteren, is het hulpprogramma 'FDDMON/O' nodig. Dit staat op de diskette 'FDD TOOLKIT/1' (zie onderaan dit artikel). Dit programma maakt het mogelijk alle opties van de VC1541 disk drive te gebruiken.

FILE/RECORD STRUKTUUR:

Relatieve files gebruiken records van een vaste lengte! Dit is een gegeven, bepaald door de software in de VC1541 drive. Bovendien kunnen alleen karakter strings als records worden gebruikt! De maximum string lengte is 254 karakters.

Het maximum aantal records is 65335 (#FFFF). Dit betekent in de praktijk, dat het aantal records alleen beperkt wordt door de opslag capaciteit van de diskette.

In ons voorbeeld van het adressenbestand nemen we:

naam	20 karakters
adres	20 karakters
plaats	15 karakters

De totale lengte van een record is nu 20+20+15=55 karakters.

De lengte van een record moet van te voren worden vastgelegd. Later kan deze niet meer gewijzigd worden!!! De VC1541 gebruikt dit gegeven voor het opzetten van de file op de diskette.

Om elk afzonderlijk veld in een record te kunnen hanteren, moeten we er voor zorgen, dat elk veld de goede lengte heeft. Is bv. een naam maar 14 karakters lang, dan moeten 6 spaties worden toegevoegd. Later zullen we zien hoe de afzonderlijke velden tot een record worden samengevoegd en hoe ze weer uit een record in afzonderlijke velden te scheiden zijn.

Een file moet altijd een naam hebben. Deze mag max. 16 karakters lang zijn.

COMMUNICATIE DAI - VC1541:

Relatieve files kunnen alleen m.b.v. het programma FDDMON/O gehanteerd worden. Dit programma moet samen met Uw BASIC programma worden geladen. Is de naam van het BASIC deel bv. 'MAILING LIST/B', dan kan het worden geladen met:

```
BOOT"FDDMON/O,MAILING LIST/B"
```

De hele communicatie vindt plaats m.b.v. de object code routines van FDDMON/O. De integratie hiervan in het BASIC programma gaat het eenvoudigst door in het BASIC programma (INT) variabelen te declareren, die de adressen van het machinetaal routines bevatten, zoals:

```
OPEN3 =#400      PRNT3=#406      ERROR=#F2E8
CLOSE3=#421      INP3 =#418      etc.
```

Zie het manual van 'FDD-TOOLKIT/1' (zie onder) voor een complete lijst met adressen.

In afwijking van de gewone lees- en schrijfprocedures bij de DAI-DOS1541, moeten relatieve files door de gebruiker worden geopend en gesloten. Ook het foutkanaal wordt niet automatisch gelezen. Alle communicatie vindt plaats via kanaal 3 van de VC1541, in combinatie met kanaal 15 (commando/fout kanaal) voor de systeem instrukties.

Slechts 1 relatieve file kan tegelijkertijd gebruikt worden. Als gegevens van een tweede file gehanteerd moeten worden, dan moet eerst de eerste file gesloten worden voordat de tweede wordt geopend. Anders verschijnt een foutmelding 'NO CHANNEL'.

OPENEN/OPZETTEN/SLUITEN VAN EEN RELATIVE FILE:

Een relatieve file wordt op de volgende manier GEOPEND:

```
10   FILENAME$="MAILING LIST/R"
20   RECLENGTH=56          (1 meer dan de werkelijke leng-
                           te vanwege een CR aan het eind)
30   OPEN$=FILENAME$+",L,"+CHR$(RECLENGTH)
40   CALLM OPEN3,OPEN$
```

Wanneer een relatieve file de eerste keer wordt geopend, is het raadzaam alvast een aantal records te reserveren. Nodig is dit niet, maar het verhoogd de snelheid bij hetweg-schrijven van records. Een aantal records 'reserveren' betekent dat een aantal 'lege' records op de diskette worden geschreven.
Denkt U bv. in eerste instantie 200 records nodig te hebben, dan is het voldoende om data in het 200e record te schrijven. Alle records 1-199 zijn dan tevens gereserveerd, zodat het schrijven in deze records later veel vlugger gaat. Dit principe kan steeds herhaald worden. Als de 200 records gebruikt zijn, dan is schrijven in record 400 voldoende om ook de records 201-399 te reserveren. Dit reserveren wordt gedaan door #FF - CHR$(255) - in het record te schrijven.

Eerst moet de VC1541 verteld worden welk record gebruikt moet worden. Dit gebeurt als volgt:

```
50   RECNR=200        (als voorbeeld)
60   HB=RECNR SHR 8
70   LB=RECNR IAND 255
80   POS$="P"+CHR$(3)+CHR$(LB)+CHR$(HB)+CHR$(0)
90   CALLM PRNT15,POS$
```

De string POS$ vertelt de VC1541 dat het een 'P'ositie commando is op kanaal 3, het record nummer, en dat begonnen moet worden op het eerste byte (byte 0) van dat record. Deze string wordt gestuurd via commando kanaal 15.

Om [illegible] schrijven, is het gebruik van de

```
100  OP$=CHR$(255)
110  CALLM PRNT3,OP$
```

De print opdracht in regel 110 zal een foutmelding 'RECORD NOT PRESENT' tot gevolg hebben, omdat record 200 nog niet bestaat als #FF erin wordt weggeschreven. Deze 'fout'melding kunt gevoegelijk naast U neer leggen.

Het op deze manier reserveren van een groot aantal records kost de nodige tijd. Voor 500 records van 400 karakters is ongeveer 2.5 minuten nodig.

Een relatieve file wordt GESLOTEN met:

```
120  CALLM ERROR: CALLM CLOSE3
```

Voordat de relatieve file wordt gesloten is het raadzaam eerst het foutkanaal uit te lezen. Een eventuele foutmelding zal dan worden geprint op het scherm.

WEGSCHRIJVEN VAN DATA NAAR EEN RELATIEVE FILE:

Als een relatieve file is opgezet zoals hierboven omschreven dan kunnen gegevens worden weggeschreven. In ons voorbeeld hebben we 3 variabelen: NAM$, ADDR$ en TOWN$, elk met een gedefinieerde lengte. De lengten liggen opgeslagen in de variabelen LN,LA en LT:

```
naam   :  NAM$          LN=20
adres  :  ADDR$         LA=20
plaats :  TOWN$         LT=15
```

Een klein programma zal nu in stappen beschrijven hoe de gegevens naar de file worden gezonden (N.B.: het gebruik van FDDMON/0 is noodzakelijk !).

```
10   INPUT NAM$: PRINT
20   IF LEN(NAM$) >LN THEN 10
30   NAM$=NAM$+SPC(LN-LEN(NAM$))
40   INPUT ADDR$: PRINT
50   IF LEN(ADDR$)>LA THEN 40
60   ADDR$=ADDR$+SPC(LA-LEN(ADDR$))
70   INPUT TOWN$: PRINT
80   IF LEN(TOWN$)>LT THEN 70
90   TOWN$=TOWN$+SPC(LT-LEN(TOWN$))
```

Met dit stukje programma worden de strings gevuld met gegevens voor elk veld van het record. Te lange velden worden niet geaccepteerd. Alle strings worden aangevuld met spaties tot de vereiste lengte.

Nu moet het nummer van het gewenste record worden ingevoerd en omgezet naar een voor de VC1541 begrijpelijke vorm:

```
100  INPUT RECNR: PRINT
110  HB=RECNR SHR 8: LB=RECNR IAND 255
```

De relatieve file kan nu worden geopend:

```
120  FILENAME$="MAILING LIST/R": RECLENGTH=56
130  OPEN$=FILENAME$+",L,"+CHR$(RECLENGTH)
140  CALLM OPEN3,OPEN$
```

De wijzer naar het gewenste record moet aan de VC1541 worden doorgegeven:

```
150  POS$="P"+CHR$(3)+CHR$(LB)+CHR$(HB)+CHR$(0)
160  CALLM PRNT15,POS$
```

Nu kunnen de gegevens worden overgezonden, nadat ze tot een string OP$ zijn samengevoegd:

```
170  OP$=NAM$+ADDR$+TOWN$
180  CALLM PRNT3,OP$
```

Deze invoer procedure kan voor andere records worden herhaald, bv. met een programma-lus. Als de invoer klaar is, moet de relatieve file worden gesloten:

```
190  CALLM ERROR: CALLM CLOSE3
```

LEZEN VAN DATA VAN EEN RELATIEVE FILE:

Gegevens kunnen op 2 manieren van een relatieve file worden gelezen: karakter na karakter (GET) of als een string (INPUT). De GET-methode is bruikbaar als slechts 1 of een paar karakters van een record gelezen moeten worden. Is het hele record nodig, dan kan de INPUT-methode gebruikt worden.

In ons voorbeeld zullen we beide methodes gebruiken. De GET-methode om te zien of een record leeg is en de INPUT-methode om het record te lezen.

De verbinding tussen BASIC en machinetaal is hier wat vreemd doordat vanuit machinetaal karakters direkt in de BASIC variabelen worden ge-poked. Hiervoor moeten 2 variabelen aan het begin van het programma worden gedeclareerd:

```
1     IN$=" "
2     IP$="": FOR I=0 TO 254: IP$=IP$+" ": NEXT
```

De variabele IN$ wordt gebruikt voor tijdelijke opslag van een karakter gelezen met het GET-commando. De 255 spaties in de variabele IP$ dienen voor opslag van een compleet record dat met het INPUT-commando wordt gelezen.

De relatieve file wordt nu op dezelfde manier geopend als bij het schrijven:

```
10    CALLM OPEN3,OPEN$          ) zie 'schrijf data',
20    CALLM PRNT15,POS$          ) regels 100-160
30    CALLM GET3,IN$
40    IF ASC(IN$)=255 THEN 100
```

Als het eerste karakter van het record #FF is, dan is het record leeg.

```
50    CALLM PRNT15,POS$
60    CALLM INP3,IP$
```

Het voor de tweede keer gebruiken van het POS-commando is nodig om de VC1541 weer naar het eerste byte van het record te laten verwijzen. Het hele record is nu opgeslagen in de variabele IP$ en kan nu in de 3 oorspronkelijke delen (naam, adres, plaats) verdeeld worden:

```
70    NAM$ =MID$(IP$,0,LN)
80    ADDR$=MID$(IP$,LN,LA)
90    TOWN$=MID$(IP$,LN+LA,LT)
```

De verschillende velden kunnen nu naar believen worden gewijzigd en daarna weer weggeschreven worden als eerder omschreven.

Vergeet niet de relatieve file te sluiten als al het lezen en schrijven klaar is:

```
100  CALLM ERROR: CALLM CLOSE3
```

VERWIJZINGEN:

Een zeer uitgebreide uitleg van relatieve files, met veel voorbeelden, kunt U vinden in 'Das grosse Floppy Buch', door Englisch/Szczepanowski, uitgegeven door Data Becker GmbH in Duesseldorf. Dit boek is ook in het Nederlands verkrijgbaar.

Het programma FDDMON/O staat op de diskette 'FDD TOOLKIT/1'. Deze kan besteld worden bij Micro Service, Fabritiusstr.15, 6174 RG Sweikhuizen, door overmaking van F 60.- op bankrekening 13.05.78.754 (Rabo-bank, Geleen). Het gironummer van de bank is 1037671. A.u.b. vermelden: 'FDD TOOLKIT/1'.
Deze diskette bevat een aantal nuttige programma's voor een uitbreiding van de mogelijkheden van de DAI-DOS 1541.

Jan Boerrigter, juni 1985

DAI-DOS 1541

Voor bezitters van het DAI-DOS 1541 hebben we een verheugende mededeling.
Een collega, tevens Commodore-bezitter, heeft voor ons een EPROM gemaakt welke in de Commodore 1541 drive komt met een aantal verbeteringen t.o.v. het originele systeem. Deze zijn:
- Floppy-motor begint te draaien wanneer men de schijf er in steekt. Dit heeft een betere centrering van de schijf ten gevolge.
- De tijd van formateren is teruggebracht van 80 sec. naar 24 sec..
- De leestijd is ongeveer 30% korter geworden, m.a.w. de drive is sneller geworden.
- Het bekende ratelen is tot een minimum beperkt en de bewegingen van de leeskop zijn vloeiender geworden.
- De compatibiliteit met andere drives en met de Commodore 64 is gelijk gebleven.

De vervanging is eenvoudig. PROM (24 pins) uit de drive nemen en de EPROM (28 pins) met verloopvoetje op de plaats zetten (meestal zit deze PROM op een voetje).
Idien er belangstelling bestaat voor deze EPROM laat het ons weten. Dan kunnen we bekijken wat de kosten worden en u daarover t.z.t. berichten in een van de DAI-publicaties. Denk wel om de garantie op uw drive!

Hierover het volgende: Commodore heeft sinds enige tijd de reparaties van al zijn apparatuur overgedaan aan een firma in Rotterdam. Deze firma rekent voor iedere reparatie, in of na de garantietijd, ± f 80,00 als minimum bedrag. Bent u binnen de garantietijd dan moet uw leverancier dit bedrag betalen, maar hierover ontstaan nog wel eens problemen. Volgens berichten in tijdschriften kan de reparatie geruime tijd in beslag nemen. Volgens een artikel liep dit soms op tot drie a vier maanden.
Uit ervaring weten we dat de drive erg betrouwbaar is, maar er kan altijd iets stuk gaan.
Mocht u in de toekomst problemen krijgen met bijv. het niet meer kunnen lezen van in de aanvangsperiode op diskettes weggeschreven programma's, bel ons; misschien kunnen wij u helpen.

Het DAI-DOS 1541
Ontwikkelingsteam.

De variable FC in het programma FGT wordt nergens gespecificeerd of gebruikt, maar is wel beschreven als de achtergrondKleur voor het te tekenen karakter.Helaas ,geen kleur te bekennen.Probeer het maar eens met FC=4.De fout is dat er geen kleurnummer ingevuld moet worden,doch het COLORG register.Dus 0,1,2 of 3. Zelfs de DAI-superdemo maakt geen gebruik van FC.Zij vulden de achtergrond door middel van een FILL-statement i.p.v. FC.

TIPS...TIPS...TIPS...TIPS...TIPS...TIPS...TIPS...TIPS...TIPS...

```
1     REM ******* l o t t o **********
2     REM
3     REM programma overgenomen uit HCCN nr.71
4     REM ------------------------------------------------
5     POKE #75,#20:COLORT 7 14 7 14
6     POKE #131,1:PRINT CHR$(12):POKE #BD51,#6A
7     CURSOR 10,10:PRINT "D A I  L O T T O "
8     CURSOR 12,17:PRINT "**************************************"
9     REM ------------------------------------------------------------
10    GOSUB 1000:REM Initieren
20    GOSUB 9000:REM Hoeveel getalreeksen
30    GOSUB 8000:REM Uitvraag voor scherm of printer
40    FOR AK%=1 TO AR%
50    GOSUB 2000:REM Trekking en Controle
60    GOSUB 3000:REM Getallen sorteren
70    GOSUB 4000:REM Bekendmaking
80    GOSUB 7000:REM Nulstellen van getal
90    NEXT AK%
100   GOSUB 5000:REM Tellen van dezelfde getallen
110   GOSUB 6000:REM Afdrukken van de getallen
120   GOTO 9999
130   REM ------------------------------------------------------------
1000  REM Initieren
1010  CLEAR 2000:GOSUB 1500:REM DATUM
1020  DIM GETAL%(7,0)
1030  DIM TOT%(252,0)
1040  DIM SRT%(41,0)
1050  Y%=3:Z%=16:CV%=13
1060  RETURN
1070  REM ------------------------------------------------------------
1500  REM Datum-routine
1510  CURSOR 21,22:PRINT "DATUM : ";:A%=#873
1520  FOR X%=1 TO 2:B%=PEEK(A%):PRINT HEX$(B%-#30);
1530  A%=A%+1:NEXT:IF A%)#878 THEN 1550
1540  PRINT ".";:GOTO 1520
1550  RETURN
1560  REM ------------------------------------------------------------
2000  REM Trekking en Controle
2010  FOR T%=1 TO 7
2020  LET TGETAL%=INT(41,0*RND(1,0)+1,0)
2030  FOR X%=1 TO T%
2040  IF TGETAL%=GETAL%(X%) THEN GOTO 2020
2050  NEXT X%
2060  LET GETAL%(T%)=TGETAL%
2070  TT%=TT%+1
2080  LET TOT%(TT%)=TGETAL%
2090  NEXT T%
2100  RETURN
2110  REM ------------------------------------------------------------


3000  REM Sorteren GETAL 1 t/m GETAL 6
3010  FOR X%=1 TO 5
3020  FOR Y%=1 TO 6-X%
3030  IF GETAL%(Y%))GETAL%(Y%+1,0) THEN GOSUB 3500:REM Wissel
3040  NEXT Y%
3050  NEXT X%
3060  RETURN
```

```
3070  REM ------------------------------------------------
3500  REM Verwissel GETAL(Y) met GETAL(Y+1)
3510  LET H%=GETAL%(Y%)
3520  LET GETAL%(Y%)=GETAL%(Y%+1.0)
3530  LET GETAL%(Y%+1.0)=H%
3540  RETURN
3550  REM ------------------------------------------------
4000  REM Bekendmaking
4010  ON IP% GOTO 4030,4050,4130
4020  GOTO 4130
4030  POKE #131,1:REM Alleen scherm
4040  GOTO 4060
4050  POKE #131,0:REM Beiden
4060  PRINT "De getallen zijn  :";
4070  FOR X%=1 TO 6
4080  PRINT GETAL%(X%);
4090  NEXT X%
4100  PRINT
4110  PRINT "Het reserve getal :";
4120  PRINT GETAL%(7.0):PRINT
4130  POKE #131,0
4140  RETURN
4150  REM ------------------------------------------------
5000  REM Tellen van dezelfde getallen
5005  IF AR%=1 GOTO 5160
5010  WAIT TIME 100:POKE #131,1
5020  PRINT CHR$(12):CURSOR 0,21
5030  PRINT "Even wachten ik tel dezelfde getallen"
5040  CURSOR 17,17:PRINT "--------------------------"
5050  FOR X%=1 TO 41
5060  POKE #BD51,#6A:CURSOR 11,18:PRINT X%
5070  FOR TT%=1 TO AR%*7
5080  IF TT%<100 THEN CURSOR 18,18:GOTO 5100
5090  CURSOR 17,18
5100  PRINT TT%
5110  IF TOT%(TT%)=X% THEN SRT%(X%)=SRT%(X%)+1.0
5120  NEXT TT%
5130  IF X%<41.0 THEN CURSOR 26,18:PRINT SPC(5)
5140  NEXT X%
5150  POKE #131,0
5160  RETURN
5170  REM ------------------------------------------------

6000  REM Afdrukken van de gesorteerde getallen
6010  IF AR%=1 THEN GOTO 6110
6020  FOR X%=1 TO 41
6030  Z%=Z%-1
6040  IF X%<10 THEN CURSOR Y%+1,Z%:GOTO 6060
6050  CURSOR Y%,Z%
6060  PRINT X%;" =";
6070  IF SRT%(X%)<10.0 THEN PRINT " ";
6080  PRINT SRT%(X%);" MAAL"
6090  IF Z%=2.0 THEN Z%=16:Y%=Y%+10
6100  NEXT X%
6110  RETURN
6120  REM ------------------------------------------------
7000  REM Nulstellen van getal
7010  FOR N%=1 TO 7
7020  LET GETAL%(N%)=0
7030  NEXT
7040  RETURN
```

```
7050  REM ------------------------------------------------------------
8000  REM scherm of printer
8010  CURSOR 17,12:PRINT "Wilt u de getalreeksen op :"
8020  CURSOR 17,10:PRINT "1-  het scherm"
8030  CURSOR 17,9:PRINT "2-  printer en scherm"
8040  CURSOR 17,8:PRINT "3-  geen van beide"
8050  CURSOR 17,6:INPUT "Uw keuze :";IP%
8060  IF IP%>0 AND IP%<4 THEN 8080
8070  CURSOR 20,4:PRINT SPC(5):GOTO 8050
8080  IF IP%<>2 THEN 8120
8090  CURSOR 10,2:PRINT "Staat de PRINTER ingesteld"
8100  CURSOR 10,1:PRINT "Druk op spatiebalk voor verder"
8110  CALLM #D6DA:GOTO 8170
8120  IF IP%<>3 THEN 8170
8125  IF AR%=1 THEN GOTO 8170
8130  FOR X%=1 TO 7:CV%=CV%-1.0
8140  CURSOR 0,CV%:PRINT SPC(60)
8150  NEXT X%:CURSOR 10,2
8160  PRINT "Even geduld ik bouw aan de getalreeksen":GOTO 8180
8170  PRINT CHR$(12)
8180  RETURN
8190  REM ------------------------------------------------------------

9000  REM Hoeveel maal de reeksen berekenen
9010  CURSOR 11,10:PRINT "Hoeveel getalreeksen laten berekenen "
9020  CURSOR 11,9:INPUT "Maximaal 36 keer :";AR%
9030  IF AR%>36 THEN AR%=36
9040  CURSOR 0,10:PRINT SPC(60)
9050  CURSOR 0,9:PRINT SPC(60)
9060  RETURN
9070  REM ------------------------------------------------------------
9700  REM TEKST EN UITLEG
9710  REM Na opstarten verschijnt het LOGO,daarna de vraag
9720  REM hoeveel maal 6 uit 41 moet werden aangemaakt.
9730  REM Maximaal is 36 mogelijk meer ingave wordt terug-
9740  REM gebracht naar 36. Dit is ivm het DIM statement die
9750  REM er maar 255 mag hebben en 36 keer 7 is 252.
9760  REM Na ingave verschijnt het menu voor wel of niet op
9770  REM scherm of printer.Na ingave volgt de verwerking
9780  REM Bij de berekening wordt gekeken naar de 0 en de
9790  REM en de dubbelen.
9800  REM
9810  REM De volgende koderingen zijn toegepast:
9820  REM 'GETAL%  ' Hierin de berekende waarde tussen 0 en 42
9830  REM 'TOT%    ' Hierin het totaal aantal getrokken waarden
9840  REM 'SRT%    ' Hierin het aantal gelijken tussen 0 en 42
9850  REM 'TGETAL%' Tussenveld voor vergelijking van dubbelen
9860  REM
9870  REM 'CV%' for/next Cursor Verticaal
9880  REM 'Z% ' cursor positie en teller
9890  REM 'Y% ' cursor positie en teller
9900  REM 'TT%' index in for/next
9910  REM 'T% ' index in for/next
9920  REM 'X% ' index in for/next
9930  REM 'N% ' index in for/next
9940  REM 'H% ' hulpveld bij de sort
9950  REM 'IP%' input veld voor menu
9960  REM 'AR%' input veld voor hoeveel reeksen
9970  REM 'AK%' index voor for/next
9980  REM ------------------------------------------------------------
9999  CURSOR 0,1:POKE #75,#5F:REM HET EINDE
```

ROUTINE VOOR HET TESTEN VAN DE VRIJE RUIMTE IN DE HEAP.

In HCC nr 72 stond een oproep van de secretaris-penningmeester Wim Bremer of het mogelijk was te zien hoeveel vrije ruimte er is in de heap, zodat een 'OUT OF STRING SPACE'-error voorkomen kan worden. Omdat ik zelf bezig ben met een adresbestandsprogramma (waarover in een volgende DAITA misschien meer) en met ditzelfde probleem te maken kreeg, heb ik me eens wat meer verdiept in de DAI-heap.

Bij de DAI worden in de heap de stringvariabelen en de array's geplaatst. Bij het aanzetten van de DAI en bij een RESET wordt de heapgrootte gezet op #100, hetgeen met behulp van het BASIC-commando CLEAR op elke willekeurige waarde tussen 4 en #7FFF gezet kan worden.
De start van de heap wordt gevonden op #29B-#29C, de grootte op #29D-#29E.

Bij het NEW en het CLEAR commando wordt alle ruimte in de HEAP vrijgegeven voor gebruik. Dit geschiedt door slechts 4 bytes te veranderen, de overige bytes in de heap blijven ongewijzigd. Deze 4 bytes zijn de eerste en de laatste twee bytes van de heap (dit verklaart waarom de waarde bij CLEAR minimaal 4 moet zijn, deze bytes worden gebruikt voor de HEAP opzet zelf, de effectieve ruimte is dan dus 0). In de eerste twee bytes wordt [heap-size - 4] gezet in de volgorde hibyte-lowbyte, dus tegengesteld aan de heapsize in #29D. Om aan te geven dat deze ruimte vrij is wordt het eerste bit van het eerste byte op '1' gezet. Bijvoorbeeld: bij RESET wordt de heapsize gelijk #100, - #29D wordt #00, #29E wordt #01 -, en op de eerste twee bytes van de heap, te vinden uit #29B, (bij reset is dit #2EC), komt te staan #80 en #FC (#0100 - 4 = #00FC, met het eerste bit op '1' wordt dit #80FC).
In de laatste twee bytes van de heap komt als aanduiding 'einde heap' #7FFF te staan, dus bij reset is dit op #3EA en #3EB.

Zoals eerder opgemerkt heeft het zetten van het eerste bit op '1' in de lengte bytes de functie van het onderscheiden van een vrije of bezette ruimte ter lengte van de waarde in de lengtebytes volgend op die lengtebytes. Dit is de reden waarom de heap maximaal #7FFF groot kan zijn. Om aan te geven dat dit gebied vrij is wordt #FFFF in de lengtebytes gezet. (M.b.v een machinetaalprogramma is het wel mogelijk een grotere heap op te bouwen, daarover meer in een volgende DAITA).

Als in een programma ruimte benodigd is voor een stringvariabele of een array-element wordt een 'heap-request' gedaan. Deze routine bevindt zich in bank 3 op #E99C-#E9FF. In het DE-register bevindt zich de lengte van de benodigde ruimte. Vanaf het begin van de heap wordt getest of het eerste bit van twee lengtebytes '1' is, ofwel of het heapdeel vrij is. Is dit zo, dan wordt met behulp van deze lengtebytes gekeken of een volgend deel van de heap vrij is. Is dit het geval dan worden de lengtes bij elkaar opgeteld, en wordt het volgende gedeelte onderzocht, totdat een bezet

gebied gevonden wordt. Dan wordt getest of de inmiddels opgebouwde vrije ruimte groot genoeg is. Zo ja, wordt de benodigde lengte op 'bezet' gezet en de restruimte op vrij en wordt de routine afgebroken. Is de ruimte niet voldoende, dan wordt gekeken of er na het bezettte gebied nog een vrij gebied is. Dit wordt herhaald tot een voldoende groot vrij gebied gevonden is of totdat het einde van de heap (de lengtebytes zijn #7FFF) bereikt is. Bij onvoldoende ruimte volgt de 'OUT OF STRING SPACE' error.

De hierna volgende routine telt de vrije gebieden in de heap (in #F4 en #F5) en geeft tevens het grootste aaneengesloten vrije gebied (in #F6 en #F7) aan. Dit laatste is de belangrijkste grootheid.

Deze routine kan in een BASIC programma als volgt gebruikt worden:

```
CALLM#200:MAXFRE=PEEK(#F6)+PEEK(#F7)*256
```

Door vervolgens MAXFRE te vergelijken met de geschatte benodigde maximale ruimte voor een stringvariabele of array-element VOOR het geven van een waarde, kan een 'OUT OF STRING ERROR' voorkomen worden.

F. Moeijes
Raai 4
1422 SX Uithoorn

PS Met dank aan het 'DAI firmware manual' van B.J.Boerrigter

```
1       ;
2       ;                       COUNT FREE      by F. MOEIJES  850811
3       ;
4       ;                       This routine calculates the total free
5       ;                       area in the heap in TOTFRE (#F4+#F5) and
6       ;                       the maximum continuous free area in
7       ;                       MAXFRE (#F6+#F7)
8       ;
9               ORG     200H            ;start at #200
10      TOTFRE  EQU     0F4H            ;contains total free area
11      MAXFRE  EQU     0F6H            ;contains maximum continuous
12      ;                                free area
13      CNTFRE  PUSH PSW
14              PUSH B
15              PUSH D
16              PUSH H
17              LXI H   0H
18              SHLD    TOTFRE          ;set total free area = 0
19              SHLD    MAXFRE          ;set maximum free area = 0
20              LHLD    29BH            ;get start heap
21      LBFRE1  MOV D,M
22              INX H
23              MOV E,M                 ;length bytes in DE
24              INX H
25              MOV A,D
26              ANI     7FH             ;mask 'free/used' bit
27              CMP D                   ;test area free or used
28              MOV D,A
```

```
29                JNZ       LBFRE3          ;if area free (bit7 = 1)
30    ;                                       check next area(s)
31                CPI       7FH             ;else test if end heap
32    ;                                       (= #7FFF) reached
33                JNZ       LBFRE2
34                MOV A,E
35                INR A
36                JZ        0C14DH          ;at end, pop all, return
37    LBFRE2      DAD D                     ;HL points to next area
38                JMP       LBFRE1          ;check next area
39    LBFRE3      PUSH H                    ;save start 1st free area
40    ;                                       (after used area)
41                DAD D                     ;begin next area in HL
42                MOV A,M
43                ORA A                     ;check if this area is free
44                JP        LBFRE4          ;if used, continue
45                INX H
46                MOV A,E
47                ADD M                     ;add lowbyte length next area
48    ;                                       to length previous area
49                MOV E,A
50                MOV A,D
51                DCX H
52                ADC M                     ;add hibyte length next area
53    ;                                       to length previous area
54                ANI       7FH             ;skip bit 'free/used'
55                MOV D,A
56                INX D
57                INX D                     ;add 2 (ex-length) bytes
58                POP H                     ;restore start 1st free area
59                JMP       LBFRE3          ;check next area
60    LBFRE4      LHLD      TOTFRE          ;get old total free area
61                DAD D                     ;add extra free area
62                SHLD      TOTFRE          ;restore total free area
63                LHLD      MAXFRE          ;get old maximum free area
64                CALL      0DE14H          ;compare old/new max free area
65                JNC       LBFRE5          ;if old area greater, continue
66                XCHG
67                SHLD      MAXFRE          ;restore new max free area
68                XCHG
69    LBFRE5      POP H                     ;restore start 1st free area
70                DAD D                     ;HL points to next area
71                JMP       LBFRE1          ;check next area
72                END
```

```
CNTFRE:0200  LBFRE1:0210  LBFRE2:0226  LBFRE3:022A  LBFRE4:0241  LBFRE5:0256
MAXFRE:00F6  TOTFRE:00F4

>D200 25A

0200 F5 C5 D5 E5 21 00 00 22 F4 00 22 F6 00 2A 9B 02
0210 56 23 5E 23 7A E6 7F BA 57 C2 2A 02 FE 7F C2 26
0220 02 7B 3C CA 4D C1 19 C3 10 02 E5 19 7E B7 F2 41
0230 02 23 7B 86 5F 7A 2B 8E E6 7F 57 13 13 E1 C3 2A
0240 02 2A F4 00 19 22 F4 00 2A F6 00 CD 14 DE D2 56
0250 02 EB 22 F6 00 EB E1 19 C3 10 02
>>
```

INDEXMAKER voor MINI-DIGITALE CASSETTES door Inno Broekman

Een DCR-tape bestaat meestal uit het USER programma dat zelfstartend het eerstvolgende BASIC-programma binnenhaalt. Dat eerstvolgende is TAPECONTROLE en bevat de bandinhoud zonder de twee bovengenoemde.
U kunt met INDEXMAKER eenvoudig een inhoudsopgave van uw DCR-tapes maken. De plaats, die het gedeelte TAPECONTROLE op de band inneemt blijft steeds gelijk, zodat u bij aanvulling zonder bezwaar de nieuwe TAPECONTROLE over de oude kunt schrijven. Volg de aanwijzingen in het programma op.

```
10    REM index voor tapecontrole
20    MODE 0:CLEAR 10000:POKE #131,1:CALLM #F000:REM REW
30    POKE #75,#20:PRINT CHR$(12):AD$=""
40    FOR A%=1 TO 21:READ B%:AD$=AD$+CHR$(B%):NEXT A%
50    AD%=PEEK(VARPTR(AD$)) IOR (PEEK(VARPTR(AD$)+1) SHL 8)+1
60    R$="":T%=-1
70    X$="":POKE #131,1
80    IF T%>23 THEN PRINT :PRINT "TEVEEL PROGRAMMA'S VOOR INDEX !!!":GOTO 240
90    PRINT CHR$(12);:CALLM AD%
100   FOR A%=#BFE7 TO #BF85 STEP -2:X$=X$+CHR$(PEEK(A%)):NEXT A%
110   GOSUB 270
120   IF T%=(-1) THEN R$=X$
130   IF X$=R$ AND T%<>(-1) THEN GOTO 230
140   XX$=X$
149   X$=STR$(T%):X$=LEFT$(X$,LEN(X$)-2):IF T%<10 THEN X$=" "+X$
170   X$=X$+XX$
180   XX$="":X%=50
190   IF LEN(X$)>X% THEN X$=LEFT$(X$,X%)
195   READ I$
200   GOSUB 29056
210   T%=T%+1:GOTO 70
220   IF R$="-object-USER" THEN CALLM #F000:REM REW:SKIP1:LOOK
221   IF R$<>"-object-USER" THEN CALLM #F000:REM REW:LOOK
230   PRINT CHR$(12)
240   PRINT :PRINT " Index is gereed."
241   PRINT :PRINT " Type < CLEAR 20000:EDIT1000-4000 >"
242   PRINT :PRINT "Type <BREAK>,<BREAK>,NEW,POKE#135,2, en SAVE'TAPECONTROLE'"
243   PRINT :PRINT "Haal de regels tussen 2900 en 2999 weg."
244   END
250   GOTO 220
260   DATA #F5,#C5,#D5,#E5,1,#40,0,#11,#B1,#80,#21,#9E,#E6,#CD,#CE,2,#E1,#D1,#C1
,#F1,#C9
270   IF LEFT$(X$,1)="0" THEN X$="-basic -"+RIGHT$(X$,LEN(X$)-1):GOTO 340
280   IF LEFT$(X$,1)="1" THEN X$="-object-"+RIGHT$(X$,LEN(X$)-1):GOTO 340
290   IF LEFT$(X$,1)="2" THEN X$="-array -"+RIGHT$(X$,LEN(X$)-1):GOTO 340
300   IF LEFT$(X$,1)="#" THEN X$="-assem -"+RIGHT$(X$,LEN(X$)-1):GOTO 340
310   IF LEFT$(X$,1)="P" THEN X$="-pascal-"+RIGHT$(X$,LEN(X$)-1):GOTO 340
320   IF LEFT$(X$,1)="%" THEN X$="-tekst -"+RIGHT$(X$,LEN(X$)-1):GOTO 340
330   IF LEFT$(X$,1)="&" THEN X$="-s.p.l.-"+RIGHT$(X$,LEN(X$)-1):GOTO 340
340   RETURN
1000  REM dcr tapecontrole
1010  CLEAR 1000:MODE 0:COLORT 8 0 8 0:N%=0
1020  PRINT CHR$(12):CURSOR 10,16:PRINT CHR$(1);" D C R  T A P E C O N T R O L E
 ";CHR$(1)
1030  WAIT TIME 40:PRINT CHR$(12)
1040  PRINT CHR$(12):RESTORE
1050  FOR I%=1 TO 23:READ P$
1060  PRINT P$:NEXT I%
1070  POKE #74,3:POKE #75,#20
1071  II%=24:FOR I%=#B3C9 TO #BFD3 STEP 134
1072  000%=PEEK(I%):IF 000%=#2D THEN II%=II%-1
1073  NEXT I%
1080  FOR X%=23 TO II% STEP -1
1090  CURSOR 10,X%:PRINT CHR$(1)
1100  FOR Z%=1 TO 200
```

```
1110  IF GETC<>0 GOTO 1160
1120  NEXT Z%
1130  CURSOR 10,X%:PRINT CHR$(32)
1140  NEXT X%
1150  GOTO 1080
1160  CN%=23-X%:RESTORE
1170  IF CN%=0 THEN READ P$:N%=1:GOTO 1230
1180  FOR I%=0 TO CN%:READ P$
1190  IF LEFT$(P$,3)="    " THEN CURSOR 0,1:PRINT CHR$(1);"   MEMOCOM kent dit pr
ogramma niet !  ";CHR$(1):WAIT TIME 200:GOTO 1070
1200  IF CN%=X% THEN GOTO 1070
1210  IF CN%>22 THEN GOTO 1240
1220  NEXT I%
1230  PRINT CHR$(12):CURSOR 0,2:PRINT CHR$(1);" MEMOCOM is nu aan het zoeken naa
r : ";CHR$(13);"      ";CHR$(1);P$;".";CHR$(1):GOTO 1250
1240  PRINT CHR$(12):CURSOR 0,2:PRINT CHR$(1);" MEMOCOM draait nu naar het einde
 van de band. ";CHR$(1):GOTO 1340
1250  IF N%=1 THEN GOTO 1300
1260  IF CN%>22 THEN GOTO 1340
1270  FOR LOOP%=0 TO CN%-1
1280  CALLM #F000:REM DCR0:SKIP1
1290  NEXT LOOP%
1300  IF MID$(P$,4,1)="b" THEN LOAD
1310  IF MID$(P$,4,1)<>"b" THEN CURSOR 0,23:CALLM #F000:REM LOOK
1320  PRINT :PRINT "U staat nu vlak voor het gekozen programma."
1330  PRINT :PRINT "Laad het overeenkomstig het FILE-type,":PRINT :PRINT "eventu
eel via hoofdprogramma voor tekstverwerking o.i.d.":END
1340  CALLM #F000:REM DCR0:SKIP9:SKIP9:SKIP9
1350  END
2900  REM de dataregels zijn 50 spaties breed
2970  DATA "                                                  "
2980  DATA "                                                  "
2990  DATA "                                                  "
3000  DATA "                                                  "
3010  DATA "                                                  "
3020  DATA "                                                  "
3030  DATA "                                                  "
3040  DATA "                                                  "
3050  DATA "                                                  "
3060  DATA "                                                  "
3070  DATA "                                                  "
3080  DATA "                                                  "
3090  DATA "                                                  "
3100  DATA "                                                  "
3110  DATA "                                                  "
3120  DATA "                                                  "
3130  DATA "                                                  "
3140  DATA "                                                  "
3150  DATA "                                                  "
3160  DATA "                                                  "
3170  DATA "                                                  "
3180  DATA "                                                  "
3190  DATA "                                                  "
3200  DATA "                                                  "
3210  DATA "                                                  "
3220  DATA "                                                  "
3230  DATA "                                                  "
4000  DATA "                                                  "
29054 REM ##### didacom datastatementgenerator  #####
29056 XA%=PEEK(#124)+PEEK(#125)*256+5:XS$="":XS$=CHR$(#22)
29058 FOR XI%=0 TO X%-LEN(X$):XS$=XS$+" ":NEXT XI%
29061 X$=CHR$(#22)+X$+XS$
29062 FOR XI%=0 TO X%+1
29063 POKE XA%,ASC(MID$(X$,XI%,1))
29064 XA%=XA%+1:NEXT XI%
29067 RETURN
```

DCR/CASS FILE COLLECTOR VOOR DE FWP-BUFFER (24)

Het collector-programma maakt een FWP-file van alle DCR- en cassettetapes. Dit gaat als volgt. Met een speciale CHECK-routine in machine-taal, geschreven door Anton Doornenbal, worden de bestanden op tape ge"chekt" door #300,TX$ op te roepen. De titel TX$ wordt overgebracht naar de EDIT-BUFFER (die is opgebouwd #100 achter het eind van de SYMBOL TABLE, zie de regels 1480-1530) door regel 1090: POKE #131,2:PRINT TX$:POKE #131,0. Het eind van de tape wordt ondekt door een TIME OUT-routine, die zorgt voor een terugkeer naar BASIC na een zekere tijd. Deze is 1 sec. voor DCR-tapes en een 1/2 minuut voor audiocassettes. Deze tijden kunnen worden gewijzigd in regel 2000. Voor DCR-tapes heeft dit geen zin omdat de CHECK-routine, ook zonder TIME OUT-routine, terugwindt na ± 1 sec.. De TIME OUT-routine stopt dit.

Slechte en beschermde files worden overgeslagen onmiddelijk nadat het programma een TIME OUT waarneemt. Om er zeker van te zijn dat de laatste file geen slechte of beschermde is voert het programma twee maal een SKIP uit als er een DCR is geselecteerd (dit is echter twee maal het zelfde stuk tape) en zoekt voor een file twee maal de TIME OUT-tijd als een cassetterecorder is geselecteerd (regels 1010-1140).

De menu's 1 en 2 spreken voor zich. U kunt alles wat u wilt naar de EDIT BUFFER zenden: titels, kant A/B, tape-nummer, datum, zelfs speciale tekens voor de printer (regels 1050-1060, speciale tekens niet inbegrepen).

Het programma zorgt ook voor het SAVE'n van de EDIT-buffer (= FWP-buffer) door #3B8,TIT$ (regel 1230) op te roepen en verifieert het automatisch. Bij een audio-cassette moet u eerst terugspoelen. Geeft u een spatie, dan wordt deze file eveneens geverifieerd en komt het programma terug in menu 1.

U kunt de gemaakte FWP-file onmiddelijk in buffer 1 van FWP laden, hem wijzigen, trimmen, afdrukken, enz.. Tussen zijden van de tapes worden in regel 1085 drie RETURN's gegeven. Slechte of beschermde files worden aangegeven met een "X" achter de naam (regel 1085). Het programma let er ook op dat u een cassette niet nogmaals omdraait, doordat u was vergeten dat u de andere zijde al had verwerkt. Dit wordt gedaan d.m.v. de "flag" genaamd "VLG" in de regels 1265, 1270 en 1370 en een boodschap in de regels 1824-1828. Als het programma alle files van een zijde van een DCR-tape heeft afgewerkt, kunt u de nog vrije ruimte op de tape laten bepalen. Dit wordt gedaan met de subroutine op de regels 2885-3030. Voor dat doel moet de DCR-tape wel "WRITE ENABLED" zijn, want, als de plug er niet in zit is het antwoord 0. Het getal (vrije ruimte) wordt ook naar de EDIT-buffer gestuurd (regel 2990).

U kunt het collector-programma gebruiken voor ieder tekstverwerkingsprogramma dat zijn tekst opslaat als een UT-file. File type 1 is niet noodzakelijk. Voor dat doel moet u in het BASIC-gedeelte van de collector (24) invoegen: 965 POKE #3D2,FILE-TYPE:POKE #3F2,START MOD 256:POKE #3F3,START SHR 8. U kunt dit ook direct in de object code invoeren. START = begin-adres van de tekst-buffer van uw WP.

In het machinetaal-programma zijn verwerkt:

1. Een READ LINE- (= RESTORE LINE NUMBER) routine om de data pointer te setten. (Met dank aan J.Blokker, DAITA 4, pag. 8)
2. Een ON ERROR GOTO-routine, die een boodschap geeft en naar menu 1 gaat (regels 2050 en 1000). (Met dank aan Nico Looije, DAInamic 16, pag. 172.

Als u om welke reden dan ook het programma moet BREAK-en, geef dan RUN 1000. U krijgt dan menu 1 en alles wat u tot dusver hebt verzameld ligt dan vast.


```
**********************
* ANTON DOORNENBAL   *
* Oud Aa 39a         *
* 3621 LA  Breukelen*
* Tel. 03462-63237   *
**********************
```

```
**********************
* TOM VAN ES         *
* Evertsenstraat 1   *
* 3601 JH  Maarssen  *
* Tel. 03465-68097   *
**********************
```


```
930   REM DCR/CAS FILE COLLECTOR (24) (DD 27-8-1985)
940   REM INIT ON ERROR GOTO 2110
950   ERR=2110:POKE #6C,#E:POKE #6D,4
960   POKE 6,ERR IAND #FF:POKE 7,ERR SHR 8
970   REM ENTRY PROGRAM : COLLECTOR (24)
980   CLEAR 2000:DIM P$(3,0):COLORT 8 0 8 8:MODE 0:GOSUB 1480:GOTO 1390
990   REM ENTRY >> MENU 1
1000  GOTO 1390
1010  REM COLLECTOR MAIN LOOP
1020  SL$=" ":IF DEV$="CAS" THEN GOSUB 1990:GOSUB 1730:GOSUB 1670:GOTO 1050
1030  GOSUB 2020:GOSUB 1740:GOSUB 1670
1040  CALLM #F000:REM REW
1050  B$=B1$+P$(0,0)+P$(1,0)+"    "+P$(3,0):T$="TITEL "+P$(2,0)
1060  POKE #131,2:PRINT B$:PRINT T$:PRINT :POKE #131,0:T=0
1070  TX$="":CALLM #300,TX$:IF LEN(TX$)=1 THEN 1090:IF TX$="" THEN 1190
1080  IF MID$(TX$,1,1)<>" " THEN TX$=LEFT$(TX$,1)+" "+RIGHT$(TX$,LEN(TX$)-1)
```

Relocatable code op de DAI: waarom en hoe ?

Rudy Muller

In HCC 73 van juli/augustus stond op bladzij 49 t/m 52 het artikel 'Relocatable assemblers: waarom en hoe ?'. Een misschien ietwat technisch, maar alleszins leesbaar verhaal. Het verhaal gaat vooral in op het 'hoe', het waarom komt maar summier aan de orde; daarnaast is het toegespitst op een 6809, een CPU waarbij relatieve adressering mogelijk is. Zoiets is bij de 8080 onbekend, en dus leek het mij nuttig om aan mijn ervaringen met het maken van reloceerbare machine-taal routines op de DAI een artikeltje in DAITA te wijden. Als voorbeeld heb ik aan het artikel het programma CUBES toegevoegd, waar in het tweede deel van het artikel naar verwezen wordt om een en ander te kunnen verduidelijken.

Het artikel in HCC-73 geeft de indruk dat de relocatie-techniek pas echt noodzakelijk is als het om grote of complexe stukken assembler gaat. In de praktijk van de DAI ben ik op een andere gebruiks-noodzaak gestoten; ik ben in het bezit van een floppy-unit waarop DAI-DOS draait, en dat heeft de eigenschap om zijn besturings-systeem op #0300 te laden, zodat de 'normale' plaats voor machine-taal routines al bezet is. Het gevolg is dat een BASIC-programma dat uitstapjes maakt naar een routine in machine-taal niet zonder meer uitgewisseld kan worden met een ander DAI-systeem (en zelfs niet met een andere versie DAI-DOS). Door ervoor te zorgen dat het machine-code deel op elke willekeurige plaats in het geheugen mag komen te liggen kan die hindernis worden overwonnen. Trouwens ook de DAI-bezitters, die niet behept zijn met DAI-DOS, kunnen toch geconfronteerd worden met het feit dat #0300 al bezet is (door bijvoorbeeld FGT). Het is dan plezierig als de machine-taal routine die U daarnaast wilde gebruiken reloceerbaar was, en tevreden met elk plekje in het geheugen.

Om mijn BASIC-programma's op vitale punten te versnellen maak ik graag gebruik van routine(s) in machine-taal. Daarom - en omdat ik zoiets leuk vind - heb ik een BASIC-programma gemaakt dat een stuk assembler-source omzet in object-code (in hexadecimal-ASCII notatie, inclusief relocatie-informatie). Daarna zet een tweede programma die object-code om in een aantal DATA-statements. Tenslotte worden die DATA-statements ingevoegd in het BASIC-programma dat de machine-code routine nodig heeft. Bij uitvoering van dat programma wordt de machine-code routine 'geladen' en gereloceerd in een INT-array. De DATA-statements bevatten de volgende informatie:

- de grootte van de INT-array.
- de waarde van de INT-elementen.
- de relocatie-punten

Met een kleine subroutine wordt de inhoud van de DATA-statements verwerkt tot de machine-taal routine.
Hoe het geheel er in de praktijk uitzit is te zien op regel 6000 t/m 6900 van het programma CUBES, waarin de machine-code routine uit DAITA-10 van Cees van Dijk wordt gebruikt.
Iedere DAI-gebruiker kan op die manier reloceerbare machine-taal routines aan zijn BASIC-programma hangen, ook zonder de hulp van de programma's die vanuit een source-text via een object-code de beoogde DATA-statements aanmaken.

Voor degenen voor wie het bovenstaande verhaal geen dagelijkse kost is, lijkt het mij nuttig om aan de hand van het programma CUBES de materie nog eens wat verder in detail te gaan bekijken.
Op regel 6000-6999 wordt het 'laden' van de machine-taal routine geregeld.
Op regel 6100 wordt de grootte van de INT-array gelezen (DATA op regel 6900), en gebruikt om de array PAINTER te DIMensioneren.

Op regel 6110 wordt de eigenlijke routine gelezen, (DATA op regel 6901-6908); als die array op adress #0000 lag, dan zou een (met commentaar aangevulde) disassembly het volgende te zien geven:

```
0000 C5            PUSH  B
0001 D5            PUSH  D
0002 CD0D00        CALL  MAIN
0005 D1            POP   D
0006 C1            POP   B
0007 C9            RET
0008 0000    X:    WORD  0       ; X-coordinaat
000A 00      Y:    BYTE  0       ; Y-coordinaat
000B 00      G:    BYTE  0       ; grens-kleur
000C 00      V:    BYTE  0       ; vul-kleur
000D 3A0A00 MAIN:  LDA   Y       ; haal video-
0010 4F            MOV   C,A     ;  coordinaten
0011 2A0800        LHLD  X       ;
                   etc.
```

Echter een INT-array wordt op de heap gealloceerd die op een standaard DAI op #0300 begint, maar bij DAI-DOS op een hoger adres (bij versie 2.0 op #1BF4). De dis-assembly ziet er dus in feite aldus uit:

```
1BF4 C5            PUSH  B
1BF5 D5            PUSH  D
1BF6 CD0D00        CALL  MAIN
1BF9 D1            POP   D
1BFA C1            POP   B
1BFB C9            RET
1BFC 0000    X:    WORD  0       ; X-coordinaat
1BFE 00      Y:    BYTE  0       ; Y-coordinaat
1BFF 00      G:    BYTE  0       ; grens-kleur
1C00 00      V:    BYTE  0       ; vul-kleur
1C01 3A0A00 MAIN:  LDA   Y       ; haal video-
1C04 4F            MOV   C,A     ;  coordinaten
1C05 2A0800        LHLD  X       ;
                   etc.
```

We kunnen nu meteen zien waar de schoen wringt. De instructies op #1BF6, #1C01 en #1C05 verwijzen naar de verkeerde adressen. Overal waar in de routine zo'n interne verwijzing staat, zal het adres daarin moeten worden aangepast aan de definitieve plaats. Dit is te realiseren door bij elke interne verwijzing het start-adres op te tellen.
Op regel 6120 wordt het start-adres bepaald, op regel 6200-6250 worden al de interne verwijzingen aangepast aan de actuele situatie. De (relatieve) plaats van die verwijzingen worden gehaald uit de DATA op regel 6909-6911.
De uiteindelijke vorm wordt aldus:

```
1BF4 C5            PUSH  B
1BF5 D5            PUSH  D
1BF6 CD011C        CALL  MAIN
1BF9 D1            POP   D
1BFA C1            POP   B
1BFB C9            RET
1BFC 0000    X:    WORD  0       ; X-coordinaat
1BFE 00      Y:    BYTE  0       ; Y-coordinaat
1BFF 00      G:    BYTE  0       ; grens-kleur
1C00 00      V:    BYTE  0       ; vul-kleur
1C01 3AFE1B MAIN:  LDA   Y       ; haal video-
1C04 4F            MOV   C,A     ;  coordinaten
1C05 2AFC1B        LHLD  X       ;
                   etc.
```

De CALL (op #1BF6) gaat nu keurig naar het label MAIN (op #1C01).

Omdat dankzij de relocatie-techniek het programma uitwisselbaar is geworden, is het nuttig enige aandacht te gaan besteden aan de veiligheid. Als het programma wordt overgetypt is het niet ondenkbaar dat er een type-fout insluipt. In het

BASIC-gedeelte kan dat meestal geen kwaad, maar in het machine-taal gedeelte is dat hinderlijk door de rare fratsen die het programma bij uitvoering kan gaan vertonen. Gelukkig geeft de opslag-methode in DATA de mogelijkheid om een bijna waterdichte controle uit te voeren.
Regel 6010-6040 bevat een stukje programma dat napluist of de DATA correct is (en in de goede volgorde staat). Als U de boodschap 'CODE OK ...' krijgt hebt U een zekerheid van meer dan 99.99 % dat er geen type-fout in de DATA zit. U kunt dan de controle verwijderen. Krijgt U een fout-boodschap dan zit er een type-fout in de DATA, of - minder waarschijnlijk - in het controle-deel zelf.
Een hint voor hen die zo'n test in hun eigen programma's willen inbouwen: de controle-waardes op regel 6015 en 6022 kunt U simpel verkrijgen door tijdelijk die statements te veranderen in PRINT HEX$(V):STOP.

Het is overigens niet noodzakelijk om het laden van het machine-taal gedeelte en uitvoeren ervan in een en hetzelfde BASIC-programma te doen. Als de grootte van het totale programma problemen geeft is het zelfs noodzakelijk om een op-splitsing te maken. Het programma-deel dat het laden van de machine-taal voor zijn rekening neemt, moet dan worden afgesplitst van het totale programma. Aangezien de array voor de machine-code routine de enige is, zal die netjes op de bodem van de heap worden neergevlijd. Na het laden wordt vervolgens de heap-pointer (op #29B) opgehoogd tot na de INT-array, en moet het start-adres van de machine-taal routine op een veilige plaats (bv #00C8) worden opgeborgen. Daarna kan het laad-programma ermee nokken.
Het resterende deel van het originele BASIC-programma kan vervolgens worden ge-LOAD. Om de machine-taal routine te kunnen aanspreken zal dat het opgeborgen start-adres weer tevoorschijn moeten halen, om daarna zijn weg vervolgen.

Een ander gebruik van de INT-array is om er parameters mee te kunnen doorgeven, en om terug te kunnen ontvangen. Zoals trouwens de routine van Cees van Dijk uit DAITA-10 ook al laat zien, kan dat het beste gebeuren, door ze een vaste plaats te geven. Ik begin daarom al mijn routines met dezelfde 6 instructies (die samen 8 bytes beslaan). De eerste parameter begint dus altijd op het 9de byte van de array, wat dus weer toevallig (!?) samenvalt met de derde integer van de array. Zo is er zelfs af te komen van het POKEn om parameters te zetten, (cq het PEEKen om ze terug te halen), door de machine-taal routine parameters met het formaat van een DAI-integer te laten gebruiken.
Het parameters doorgeven in CUBES zou er dan als volgt uit zien:

```
4010 PAINTER(2)=X0
4020 PAINTER(3)=Y0
4030 PAINTER(4)=B
4040 PAINTER(5)=V
4050 CALLM PAINTER
```

Daarmee zou het hinderlijke uit elkaar rafelen van de X-coordinaat zijn door-geschoven naar de machine-taal routine.

Over het programma CUBES zelf wil ik maar weinig kwijt.
Voor iemand die de naam Reutersvard iets zegt, is er geen verdere aansporing meer nodig om zich onverwijld aan het overtypen te zetten.
Voor de anderen moet het een verrassing blijven, wel wil ik nog een oplossing van een probleem kwijt, dat misschien ook interressant is voor andere vlakvul-programma's.

Zoals het commentaar al suggerreert, de statements in de 4000-serie houden zich bezig met het vullen van vlakjes op het beeldscherm. Bij een nadere beschouwing lijkt er een nutteloos statement te staan (op 4120, 4220 en 4320). Immers wordt door het volgende statement exact hetzelfde vakje gevuld met exact dezelfde kleur. Toch zijn beide statements onmisbaar, samen vormen ze de oplossing voor het probleem dat optreed als het vlak oorspronkelijk is opgebouwd uit vlakjes van willekeurige kleuren. Welke kleur er dan ook gekozen wordt om de rand af te

bakenen, het is altijd mogelijk dat er zich een vlak van juist die kleur binnen de 'omheining' bevindt, en dus door de vlak-vuller voor omheining wordt aangezien. Door eerst een omheining van de ene kleur te kiezen, en vervolgens een andere kan toch het hele vlak gevuld worden.
Althans in het programma CUBES is deze methode effectief, en zal dat in vele andere gevallen ook blijken te zijn.

Het tekenen begint met de kubus aangegeven op regel 2100, dit begin kan niet straffeloos veranderd worden (alleen 4 voldoet ook), elke andere waarde levert uiteindelijk een foute - maar wat heet fout - eindresultaat op, hoewel onderweg juist heel verrassende effecten optreden (zoals bij de waarde 6).
Daarnaast kunnen prachtige effecten worden bereikt, door als de tekening klaar is met behulp van het COLORG-statement ritmische kleur-veranderingen plaats te laten vinden.

```
*RUN
100   REM * * * * * * * * * * * * * * * * * * * * *
101   REM *                         *  25 may 85   *
102   REM *   C U B E S             * - - - - - - - *
103   REM *                         *  03 aug 85   *
104   REM * * * * * * * * * * * * * * * * * * * * *
105   REM *
106   REM *   Een tekening van 9 kubussen in
107   REM *   een onmogelijk patroon naar een
108   REM *   ontwerp uit 1934 van de Zweed
109   REM *   Oscar Reutervard.
110   REM *
111   REM * * * * * * * * * * * * * * * * * * * * *
112   REM *
113   REM * (c) Rudy Muller
114   REM *     Jan Steenstraat 4
115   REM *     7412 TC   Deventer
116   REM *     tel: 05700 - 18667
117   REM *
1000  CLEAR 1000
1010  GOSUB 6000
1100  DIM C(3):C(0)=0:C(1)=1:C(2)=8:C(3)=9
1110  COLORG C(0) C(1) C(2) C(3)
1120  COLORT C(0) C(1) C(2) C(3)
1130  MODE 6A
1140  PRINT CHR$(12);
1200  REM
1201  REM constantes:
1202  REM - A = kubus-grootte
1203  REM - B = kubus-afstand
1204  REM
1210  A=32:P=A/2:Q=A*SQR(3)/2
1220  B=40:YB=B/2:XB=B*SQR(3)/2
1300  REM
1301  REM kubus-posities
1302  REM (0,0): links-onder
1303  REM
1310  DIM X(9),Y(9)
1320  X(0)=120:Y(0)=A+10
1330  FOR I=1 TO 3:X(I)=X(I-1):Y(I)=Y(I-1)+B:NEXT I
1340  FOR I=4 TO 6:X(I)=X(I-1)+XB:Y(I)=Y(I-1)-YB:NEXT I
1350  FOR I=7 TO 9:X(I)=X(I-1)-XB:Y(I)=Y(I-1)-YB:NEXT I
2000  REM
```

```
2001  REM hoofd-programma: TEKEN KUBUSSEN
2002  REM door "gelijktijdig" de bovenkant van kubus (i),
2003  REM de rechterkant van kubus (i+3), en de linkerkant
2004  REM van kubus (i+6) te tekenen wordt het gewenste
2005  REM effect bereikt
2006  REM
2100  N=3:REM start links-boven
2110  FOR M=1 TO 9
2120  X=X(N):Y=Y(N):GOSUB 4100:N=N-3:IF N<1 THEN N=N+9
2130  X=X(N):Y=Y(N):GOSUB 4200:N=N-3:IF N<1 THEN N=N+9
2140  X=X(N):Y=Y(N):GOSUB 4300:N=N-4:IF N<1 THEN N=N+9
2150  NEXT M
2900  PRINT "(c)1934 - Oscar Reutervard"
2910  C=GETC:IF C=0 GOTO 2910
2990  END
4000  REM
4001  REM subroutine: VUL^VLAK
4002  REM in: X0,Y0: start-punt
4003  REM -    B    : grenskleur
4004  REM -    F    : vulkleur
4009  REM
4010  POKE PAINTER+8,X0 MOD 256
4020  POKE PAINTER+9,X0 SHR 8
4030  POKE PAINTER+10,Y0
4040  POKE PAINTER+11,B
4050  POKE PAINTER+12,F
4060  CALLM PAINTER
4070  RETURN
4100  REM
4101  REM subroutine VUL BOVEN-VLAK
4102  REM in: X,Y: kubus-positie
4109  REM
4110  X0=X:Y0=Y+2:F=C(1)
4120  B=C(2):GOSUB 4150:GOSUB 4000
4130  B=C(3):GOSUB 4150:GOSUB 4000
4140  B=F
4150  DRAW X,Y X-Q,Y+P B
4160  DRAW X,Y X+Q,Y+P B
4170  DRAW X,Y+A X-Q,Y+P B
4180  DRAW X,Y+A X+Q,Y+P B
4190  RETURN
4200  REM
4201  REM subroutine VUL LINKER-VLAK
4202  REM in: X,Y: kubus-positie
4209  REM
4210  X0=X-1:Y0=Y-1:F=C(2)
4220  B=C(3):GOSUB 4250:GOSUB 4000
4230  B=C(1):GOSUB 4250:GOSUB 4000
4240  B=F
4250  DRAW X,Y X-Q,Y+P B
4260  DRAW X-Q,Y+P X-Q,Y-P B
4270  DRAW X-Q,Y-P X,Y-A B
4280  DRAW X,Y-A X,Y B
4290  RETURN
4300  REM
4301  REM subroutine VUL RECHTER-VLAK
4302  REM in: X,Y: kubus-positie
4309  REM
```

```
4310  XO=X+1:YO=Y-1:F=C(3)
4320  B=C(2):GOSUB 4350:GOSUB 4000
4330  B=C(1):GOSUB 4350:GOSUB 4000
4340  B=F
4350  DRAW X,Y X+Q,Y+P B
4360  DRAW X+Q,Y+P X+Q,Y-P B
4370  DRAW X+Q,Y-P X,Y-A B
4380  DRAW X,Y-A X,Y B
4390  RETURN
6000  REM
6001  REM relocate PAINTER
6002  REM
6010  V=0:READ N
6011  FOR X=0 TO N
6012  X1=X IAND #1F:X2=32-X1
6013  READ V1:V=V IXOR ((V1 SHR X1)+(V1 SHL X2))
6014  NEXT
6015  V=V IXOR #A7D4FF37
6016  IF V<>0 THEN PRINT "!? ASSEMBLY-DATA ERROR":STOP
6020  READ V1:V=V1 IXOR ((V SHR 7)+(V SHL 25))
6021  IF V1>=0 GOTO 6020
6022  V=V IXOR #52D767AF
6023  IF V<>0 THEN PRINT "!? DISPLACEMENT ERROR":STOP
6030  PRINT "CODE OK (?) - VERWIJDER 6010 t/m 6040":STOP
6040  RESTORE
6100  READ N:DIM PAINTER(N)
6110  FOR X=0 TO N:READ PAINTER(X):NEXT X
6120  PAINTER=VARPTR(PAINTER(0))
6130  V=0:V1=VARPTR(V)+2:V2=V1+1
6200  READ X:IF X<0 THEN RETURN
6210  X1=PAINTER+X:X2=X1+1
6220  V=PAINTER+PEEK(X1)+256*PEEK(X2)
6230  POKE X1,PEEK(V2):POKE X2,PEEK(V1)
6240  GOTO 6200
6900  DATA #1F
6901  DATA #C5D5CD0D,#00D1C1C9,#00000000,#003A0A00
6902  DATA #4F2A0800,#2BEF27CD,#6F00C214,#00232208
6903  DATA #00E523EF,#27CD6F00,#C222002B,#EBE1413A
6904  DATA #0C00EF21,#0CEF27CD,#6F00C252,#000D23EF
6905  DATA #27CD6F00,#C234002B,#0DEF27CD,#6F00C23E
6906  DATA #000C2BEF,#27CD6F00,#C2480023,#3A0A00B9
6907  DATA #C22100EB,#2A0800EB,#CD14DEC2,#2100C9DA
6908  DATA #7900E521,#0B00BEE1,#C9BFC5E3,#C1C90000
6909  DATA #3,#E,#12,#18,#1B,#1F,#26,#29,#30,#38,#3B
6910  DATA #42,#45,#4C,#4F,#56,#59,#5D,#61,#65,#6C
6911  DATA #70,#74,-1
```

WAT TE DOEN NA EEN "OUT OF STRING SPACE" ERROR

door Jan Boerrigter

Naar aanleiding van mijn oproep in de HCC-Nieuwsbrief nr. 72 reageerden zowel Jan Boerrigter als Fred Moeijes op het probleem van het kwijtraken van data bij het invoeren van (string-)array's.
Zie voor de bijdrage van Fred Moeyes elders in dit nummer. Hieronder de bijdrage van Jan Boerrigter.

Wim Bremer

In geval van OUT OF STRING SPACE is de informatie nog wel aanwezig. Maar hoe kan er nog mee worden gewerkt? Er zijn in ieder geval twee methoden denkbaar:

Methode 1
Voor machines met BASIC V1.1 kan het programma weer gestart worden met RUN (linenumber). Als hier een correct regelnummer wordt gekozen - in ieder geval na de "CLEAR" !!, bijv. het regelnummer van het begin van het hoofdmenu - dan is er geen kou aan de lucht. Het eerste wat gedaan moet worden is het data-array wegschrijven naar tape/disk. Dan het programma aanpassen met een grotere "CLEAR", en het data-array weer inlezen. Nu kan met de gegevens-invoer verder worden gegaan.
In diverse artikelen, in o.a. DAInamic, is beschreven hoe voor machines met BASIC V1.0 dit RUN (linenumber) ook mogelijk gemaakt kan worden.

Methode 2
Als de machine met een "OUT OF STRING SPACE"-melding de afloop van het programma onderbreekt, zijn alle gegevens nog steeds correct in het geheugen aanwezig. Daarom kan dan vanuit de "direct"-mode het array nog worden weggeschreven naar tape/disk. Dus voor een array A$(N):

```
OUT OF STRING SPACE IN LINE ....
¥
¥SAVEA A$ "DATA"
```

Ook nu het programma aanpassen, de data weer inlezen en verder werken.

Onderstaand BASIC-programma geeft een methode om het aantal vrije bytes in de "HEAP" te berekenen. Het is geschreven in BASIC met een toelichting, omdat dat beter te begrijpen is. U kunt het zelfde principe ook in een kleine machinetaal routine aan een programma toevoegen.

(Ten behoeve van ruimte iets ingekort. W.B.)

```
1     REM ----------BEREKENEN VAN VRIJE HEAP-RUIMTE
2     REM ----------Jan Boerrigter, 1.6.1985
10    START=PEEK(#29B)+PEEK(#29C)¥256
15    REM --- BEGIN VAN DE HEAP
20    FINISH=PEEK(#29F)+PEEK(#2A0)¥256-2
25    REM --- EINDE VAN DE HEAP
30    I=START:VRIJ=0
35    REM --- "VRIJ" TELT DE NOG TE GEBRUIKEN BYTES.
40    IF I>=FINISH THEN 90
50    LENGTE=(PEEK(I) IAND #7F)¥256+PEEK(I+1)
53    REM --- ALLES IN DE HEAP BEGINT MET 2 LENGTE-BYTES
55    REM --- WAARVAN HET MSB VAN HET 1e BYTE AANGEEFT
57    REM --- OF DE RUIMTE VRIJ OF BEZET IS.
60    IF PEEK(I) IAND #80=#80 THEN VRIJ=VRIJ+LENGTE
65    REM --- IS HET MSB=1, DAN IS DE RUIMTE VRIJ.
70    I=I+2+LENGTE
75    REM --- DE LENGTE-BYTES ZIJN NIET IN "LENGTE" INBEGREPEN
80    GOTO 40
90    PRINT VRIJ
95    REM --- DIT PRINT STATEMENT IS ALLEEN VOOR DEMONSTRATIE.
100   END
```