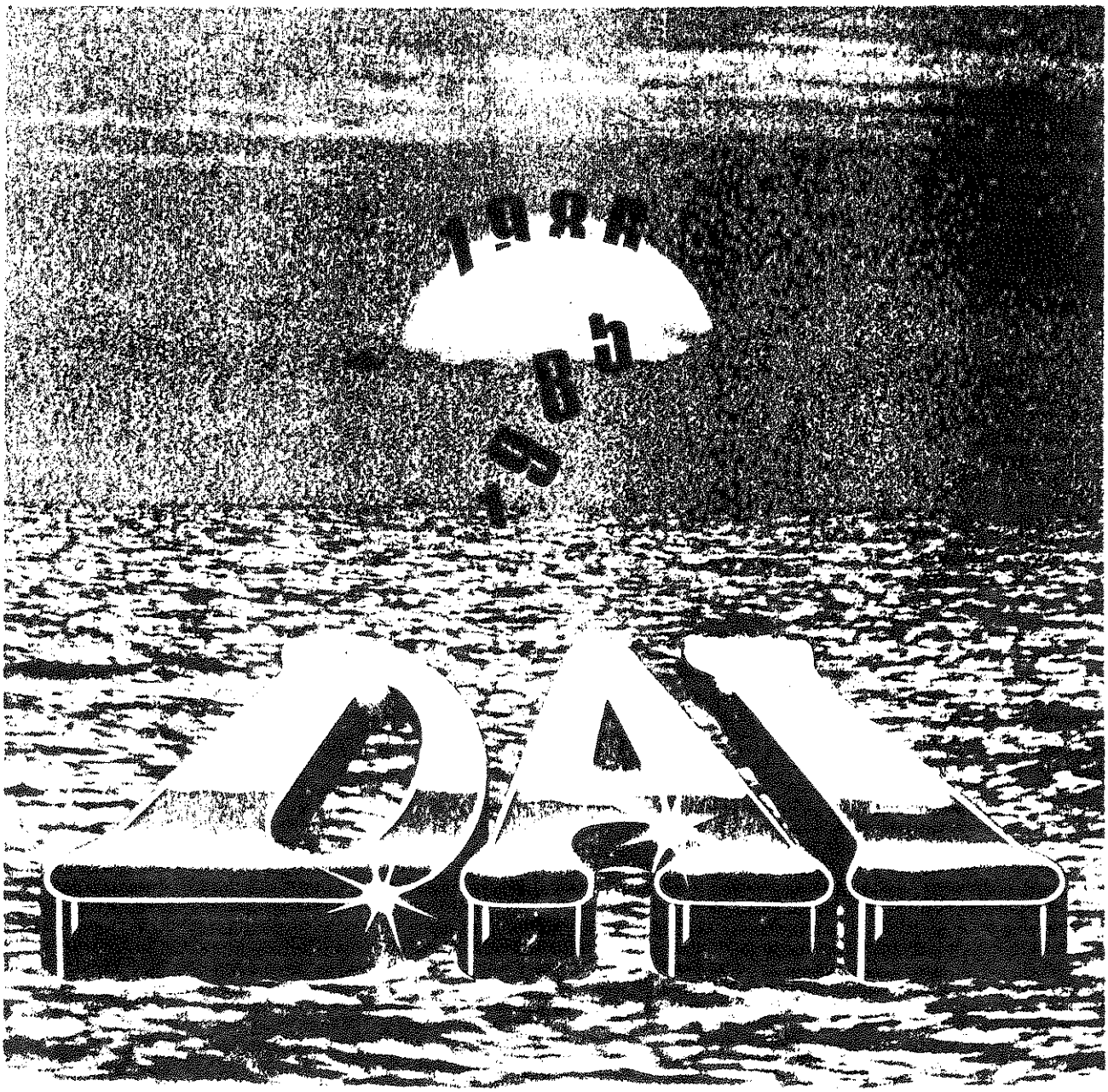
DAITA 13



UITGAVE VAN DE HCC_DAI GEBRUIKERSGROEP

IN MEMORIAM

WIM BREMER

Op 8 november j.l. overleed onze actieve secretaris van de DAI-gg. Nog de avond tevoren vergaderde het bestuur met hem, in heel ontspannen sfeer, ter voorbereiding van de HCC-dagen eind november. Niets wees er toen op, dat voor Wim het einde zo nabij was.

Vanaf de oprichting heeft Wim Bremer steeds enn zeer actieve rol gespeeld in het bestuur van de DAI-gg. Bij alle voorbereidingen voor de bijeenkomsten van de gebruikersclub en de tentoonstellingsdagen, bij de verzorging van ons clubblad DAITA en de contacten met het HCC hoofdbestuur vormde Wim steeds de spil, de regelende en coordinerende figuur. Wij mogen wel zeggen dat dankzij hem de DAI gebruikersgroep een eenheid is gebleven.

De DAI-gg zal verder moeten zonder Wim. Het zal moeilijk zijn, de leemte, die ontstaan is door zijn overleden, op te vullen.

Ons medeleven gaat uit naar zijn achtergebleven echtgenote en de kinderen, voor wie het verlies veel groter is.

Ons blijft de goede herinnering aan een man, die bij vele van zijn clubgenoten, zowel bij de DAI-gg als bij de HCC vele vrienden heeft gemaakt.

Wij zullen Wim heel erg missen.

INHOUD DAITA 13

****

****

ADRESSEN:

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

Secretaris a.i. : Theo Verberkt, Van Buerenstraat 13
+ Hardware en Software 5256 KL Oudheusden
Tel. 04162-2667

Penningmeester : Robert van den Broek, Melde 19
+Redaktie DAITA a.i. 8265 CP Kampen
Tel. 05202-17131

VAN DE VOORLOPIGE REDAKTIE.

Het viel niet mee maar DAITA 13 zit toch in elkaar.
Als zo plotseling ineens alles verandert en je er niet op bent voorbereid, dan gaan de zweetdruppels zich snel vertonen. Van angst of van hard werken laat ik in het midden.

Dat we DAITA 13 onder deze omstandigheden uitgeven zal wel aan het mysterie van het getal te wijten zijn.
Gelukkig was er voldoende materiaal beschikbaar en kunnen we weer tegen een redelijk clubblad aan- en inkijken.
Het ziet er dan wel niet zo uit als we gewent waren met prachtige koppen en goede redactionele opmaak. Maar we zullen ons best doen om het weer op het nivo terug te brengen. Ik hoop dat jullie er toch nog veel plezier aan mogen beleven, want het is de inzenders wel waart dat we hun bijdrage aandachtig zullen bestuderen.

Als nieuwe service in ons pakket is er nu de mogelijkheid om de geplubiceerde programma's op tape te bestellen tegen materiaal en verzendkosten. Wij willen hiervoor de bandjes van de PTT gaan gebruiken en dat komt op ongeveer ŧvijf gulden. Deze uitgave helaas nog niet, door alle drukten is het er bij ingeschoten. De volgend bijeenkomst wel en belooft is belooft.
Je kunt ze nu vast bestellen, ook voor de volgende DAITA. Lever dan gelijk het nieuwe materiaal in voor DAITA 14.
Er zijn vast wel leuke kleine programmaatjes die we nog niet eerder geplubiceerd hebben.

Als ik nog even terug kijk op de HCC-dagen dan mag ik constateren en met een beetje trots natuurlijk dat we geenszins in het vergeet hoekje zitten. We hadden in vergelijking met andere gg's best veel aanloop en ik heb ook nog veel vorige nummers van DAITA verkocht.

Dit is alweer de laatste DAITA van 1985. Dat betekent dat het jaar al weer bijna voorbij is, jammer eigenlijk. Er is genoeg om ons heen gebeurt, niet altijd even prettig.
Maar onze DAI laat ons gelukkig niet altijd in de steek. Laten we hopen dat het nieuwe jaar ons weer menig prettig DAI-uurtje zal bezorgen. En met een blik naar beneden hopen wij dat de fabriek ons in het komende jaar eens niet in de steek zal laten, zodat de vruchtbare bijeenkomsten tussen bestuur en de direktie een vervolg zal hebben met goede resultaten.

Van hieruit wens ik jullie mede namens het bestuur een ieder, prettige feestdagen en een voorspoedig 1986 met vele hobby uurtjes.

Veel plezier, *Robert van den Broek*

ONS BESTUUR.

Een belangrijke zaak binnen een club is het bestuur.
We hebben nieuwe aanvullingen nodig.

Kees Jagerman heeft ons te kennen gegeven zijn functie als voorzitter wegens drukke andere werkzaamheden te willen neerleggen. Dit vinden we jammer maar respecteren deze keuze. We moeten nu dringend aanvullen. Wie wil er zijn schouders onder zetten, om Kees en Wim te vervangen.

We hebben door het wegvallen van Wim zijn dubbele taak enigzins verdeelt. Ik was al bezig met het penningmeesterschap van Wim over te nemen, en neem voorlopig ook de redaktie van DAITA. Theo doet het secretariaat.

We hebben dus behoefte aan twee of drie nieuwe leden.
Wie doet mee? We zijn niet meer zo geografisch afhankelijk.
Daarna zullen we de taken opnieuw bezien en verdelen.
Geef Theo of mij een seintje als je mee wil doen.

vast bedankt, Theo en Robert.

****

ABONNEMENTSPRIJS DAITA.

Voor degene die zijn geabonneerd op de DAITA hebben we een minder leuk bericht. In de calculatie van de uitgekomen nummers hebben we moeten konstateren dat we niet met het bedrag van f15 per jaar rondkomen. Nu tante post ons ook nog een tarief verhoging in het vooruitzicht steld moeten we helaas beslissen de abonnementsprijs op f17,50 te brengen.(per 4 nummers)

Wilt u zo vriendelijk zijn dit bedrag voor 1986 op het voorlopige gironummer 3934123 tnv. R. van den Broek te Kampen met vermelding DAITA-POST, over te maken. U bent dan verzekerd van toezending.

vast hartelijk dank, Robert.

****

(sorry voor de spelling, het moet nog kunnen in 85)

DAI-gg ** S O F T W A R E B I B L I O T H E E K ** DAI-gg

De afgelopen periode zijn er weer een aantal pakketten software door een aantal mensen uitgebracht.Van Fred Moeyes kregen wij het programma "DAI-dres" binnen.Dit programma is een adressen-bestand dat gekenmerkt wordt door zijn compacte gegevensopslag van 509 adressen ,waarbij elk adres 10 rubrieken kan bevatten. Tevens heeft dit programma faciliteiten voor uitgebreide en snelle zoek-,wijzigings-,sorteer- en print akties.

Als voorbeeld: 100 adressen sorteren kost 5 SECONDEN !!!
Werkelijk een mooi stukje programmeertechniek.

****

Van Ruud Muller uit deventer kregen wij weer enige staaltjes op het gebied van grafics binnen.Allereerst het programma "VLAGGEN" Dit programma tekent de vlaggen van zowat elk land wat er bestaat.Zeer fraai.....

Tevens een programma a la FGT "GRATEX" genaamd.Bij dit programma is een demo toegevoegd "ANAGRAM" genaamd.Dit programma is het aloude bekende "WOORDRAADSEL",maar op voortreffelijke wijze in een nieuw jasje gestoken door Robert v/d Broek en Ruud Muller

****

Het Bestuur heeft de resultaten van de enquete gedeeltelijk geevalueerd en is onder andere tot de conclusie gekomen dat er bij veel mensen bezwaar was tegen de prijzen van de software die op dit moment bij de DAI-gg te koop is.Om deze en mischien ook andere mensen hierin tegemoet te komen zijn de prijzen van de software dan ook met ingang van heden gewijzigd.Op de volgende pagina treft U weer onze index van de bibliotheek aan.

Achter in het blad treft U een bestelformulier aan voor software Gelieve U bestelling op dit formulier te plaatsen en dit te sturen naar T.Verberkt op het bekende adres(zie voorin).

Als extra geeft de club in de maand december een KORTING van 25 % bij aankoop van software ter waarde van FL 100,00 of meer.

Tot de volgende DAITA.

DAI-gg ** S O F T W A R E B I B L I O T H E E K ** DAI-gg

NAAM	PRIJS CASS.	PRIJS DCR
* Viditel-pakket monitor-mlp/monitor-basic telesoftware tranciever	F 40,--	F 45,--
* DAI-dres	F 30,--	F 35,--
* Perfect Editor II	F 45,--	F 50,--
op KEN-DOS	F 55,--	
* Boekhouding (huishoud) boodschappenlijst	F 40,--	F 45,--
* Logic simulator 3D mathematic graphics	F 40,--	F 45,--
* DBMS (data base management)	F 30,--	F 35,--
* 6800 Cross-assembler	F 30,--	F 35,--
* RS232 communication programma Terminal-emulator	F 30,--	F 35,--
* Games tape 1: The vally (uitleg + spel) Advancer 1 en 2	F 30,--	F 35,--
* Postzegel-verzameling	F 30,--	F 35,--
* Programmotheek tapecontrol/demo's	F 30,--	F 35,--
* RAMBLE DAI-DOS KEN-DOS floppy		F 45,--
* SHIFT-OTHELLO-DRAWING	F 20,--	F 25,--
* DCR-DIRECTORY LISTING Gotische maandKalender FGT	F 30,--	F 35,--
* VLAGGEN-GRATEX-ANAGRAM	F 20,--	F 25,--

Voor bestelling en van software kunt U zich wenden tot :

Theo Verberkt,
Van Buerenstraat 13
5256 KL Oud-Heusden
Tel:04162-2667.

B O O T

Vooral voor gebruikers van een disk-drive kan het ers nuttig zijn om in een programma een machine-taal routine aan te roepen.

bv. u schrijft met FGT en hebt een ander lettertype nodig.
u bent aan het tekenen en wil MODE 8 aanroepen
u bent klaar met tekenen en hebt SCREENCOPY nodig
u hebt een bestand en wil een trefwoord zoeken
u wilt sorteren in dat bestand.

In al die gevallen kun je IN HET BASIC PROGRAMMA een benodigde ml-routine van cassette of schijf oproepen. Bij gebruik van cassette moeten die ml-routines wel in de goede volgorde op de cassette staan.

Hoe je dat in machinetaal doet kun je in DAInamic 1983 blz 184 lezen: 'READ/WRITE in UTILITY' van Jan Boerrigter

BOOT doet hetzelfde vanuit BASIC. Het is 'relocatable', dat betekent adres-onafhankelijk.
Het werkt op iedere plaats in het geheugen.Je kunt het daarom als een integer-array laden:

```
x10   DIM BOOT%(8)
x20   RESTORE:FOR I%=0 TO 8:READ BOOT%(I):NEXT
x30   BOOT%=VARPTR(BOOT%(0))
x40   INPUT"naam: ";NAAM$:PRINT
x50   CALLM BOOT%,NAAM$
x60   DATA #C55E2356,#2BEB0100,#31CDCE02,#11FFFAD5
x70   DATA #213E0111,#4101CDD1,#02D12A3E,#01CDD102
x80   DATA #C1C3D402
```

of zo

```
10   CLEAR #1100:POKE #29C,#12:POKE #29E,#1
20   DIM A%(8):LOADA A% "BOOT":A%=VARPTR(A%(0))
30   INPUT "NAAM";NAAM$:PRINT
40   CALLM A%,NAAM$
```

'Het laatste voorbeeld laat in regel 10 zien hoe je eerst ruimte maakt om programma's die vOOr in het geheugen horen (zoals MODE 8, SPU, BASICODE of FGT) daar te kunnen plaatsen.
Uitgangspunt is de RESET-toestand.

```
10 CLEAR #100+#X00:POKE #29C,#X+2:POKE #29E,#1
```

In regel 20 wordt het integer-array BOOT van de schijf of cassette geladen.
Met VARPTR(BOOT(0)) wordt het adres van het eerste element gevonden en daarmee het start van 'BOOT'

In regel 40 vindt de aanroep plaats. Na het laden kun je gewoon in BASIC verder, MITS DE ROUTINE NIET OVER HET BASIC-PROGRAMMA IS GESCHREVEN!! Vandaar dus regel 10 in het tweede voorbeeld.

```
300  C5 5E 23 56 EB 01 00 31 CD CE 02 11 FF FA D5 21
310  3E 01 11 41 01 CD D1 02 D1 2A 3E 01 CD D1 02 C1
320  C3 D4 02
```

```
;vanuit BASIC laden van ML-routine

             TITL     'BOOT'
ROPEN        EQU      2CEH
RBLK         EQU      2D1H
RCLOSE       EQU      2D4H
ADRES1       EQU      200H
             ORG      1000H
BASUT        PUSH B                ;bewaar BASIC pointer
             MOV E,M               ;vector van naam in HL
             INX H                 ; maak pointer
             MOV D,M
             XCHG                  ;in HL
             LXI B    3100H        ;type '1'
             CALL     ROPEN        ;lees naam
             LXI D    0FAFFH       ;of 0B350 - bovengrens
             PUSH D
             LXI H    13EH         ;hier komt beginadres
             LXI D    141H         ;tot hier
             CALL     RBLK         ;lees beginadres
             POP D                 ;bovengrens
             LHLD     13EH         ;start ml
             CALL     RBLK         ;lees routine
UIT          POP B                 ;voor BASIC terugkeer
             JMP      RCLOSE       ;sluit af, stop motor
             END
```

Veel plezier

Kees van Dijk

****

```
10    REM WATERMERK (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------
50    COLORG 15 12 0 8:MODE 6
60    FOR I=2.0*PI TO 4.0*PI STEP 3E-2
70    X%=I*30-94:Y%=COS(I)*20.0:Z%=Y%+17
80    FOR K%=40 TO 176 STEP 34
90    DRAW X%,Y%+K% X%,Z%+K% 21:NEXT K%:NEXT I
100   FILL 94,65 111,105 21:FILL 94,133 111,173 21
110   FILL 265,99 282,139 21:FILL 265,167 282,207 21
120   T%=T%+1:IF GETC()32 AND T%<1500 GOTO 120:MODE 0:LOAD
```

```
10    FOR I%=#275 TO #28F:POKE I%,#10:NEXT:REM IMP INT
20    CLEAR 10000:DIM KM%(100,0),BR%(100,0),GE%(100,0)
30    NK%=3:NKD%=2:FF$=CHR$(12):N%=0
40    PRINT FF$:CURSOR 2,22:PRINT "BRANDSTOFVERBRUIK"
50    PRINT :PRINT "KENMERK B.V. MERK  AUTO: ";:INPUT NM$
60    PRINT :PRINT "1e KM STAND MET VOLLE TANK";:INPUT Z$
70    GOSUB 10010:IF FLAG%=1 THEN PRINT :GOTO 60
80    BG%=Z%
90    KMT%=0:BRT%=0:GET%=0
100   N%=N%+1:PRINT FF$:CURSOR 0,12:PRINT "OM TE STOPPEN MET INVOEREN TYPE ***"
110   PRINT :PRINT "KM STAND BIJ TANKEN:";:INPUT Z$:IF Z$="***" THEN 210
120   GOSUB 10010:IF FLAG%=1 THEN PRINT :GOTO 100
130   KM%(N%)=Z%:KMT%=KM%(N%)-BG%
140   PRINT :PRINT "AANTAL LITERS BRANDSTOF";:INPUT Z$:IF Z$="***" THEN 210
150   GOSUB 10010:IF FLAG%=1 THEN PRINT :GOTO 140
160   BR%(N%)=Z%:BRT%=BRT%+BR%(N%)
170   PRINT :PRINT "BEDRAG";:INPUT Z$:IF Z$="***" THEN 210
180   GOSUB 10010:IF FLAG%=1 THEN PRINT :GOTO 170
190   GE%(N%)=Z%:GET%=GET%+GE%(N%)
200   GOTO 100
210   PRINT FF$
220   CURSOR 10,13:PRINT "ZET DE PRINTER KLAAR EN TYPE SPACE"
230   CALLM #DEDA
240   PRINT FF$
250   GOSUB 10270
260   PRINT CHR$(24);CHR$(27);CHR$(66);CHR$(13);
270   PRINT NM$
280   PRINT :FOR I%=1 TO 59:PRINT "*";:NEXT:PRINT "*"
290   PRINT :PRINT "KM STAND    KM GEREDEN  LITERS      BEDRAG      VERBRUIK"
300   FOR I%=1 TO N%-1
310   Z%=KM%(I%):GOSUB 10160:PRINT LEFT$(Z$,LEN(Z$)-3);
320   IF I%=1 THEN Z%=KM%(I%)-BG%:GOTO 340
330   Z%=KM%(I%)-KM%(I%-1)
340   KR%=Z%:GOSUB 10160:PRINT LEFT$(Z$,LEN(Z$)-3);
350   Z%=BR%(I%):GOSUB 10160:PRINT Z$;
360   Z%=GE%(I%):GOSUB 10160:PRINT Z$;
370   Z%=KR%*1000/BR%(I%):GOSUB 10160:PRINT Z$;
380   PRINT :NEXT
390   Z%=KMT%:GOSUB 10160:PRINT :PRINT "TOTAAL GEREDEN ";TAB(32);LEFT$(Z$,LEN(Z$)-3);"    KM"
400   Z%=BRT%:GOSUB 10160:PRINT "TOTAAL BRANDSTOFVERBRUIK ";TAB(32);Z$;" LITER"
410   Z%=GET%:GOSUB 10160:PRINT "TOTAAL AAN BRANDSTOF UITGEGEVEN";TAB(32);Z$;" GULDEN"
420   Z%=KMT%/(BRT%/1000):GOSUB 10160:PRINT "GEM. VERBRUIK 1 OP ";TAB(32);Z$;" KM"
430   Z%=GET%*1000/KMT%:GOSUB 10160:PRINT "GEM. BRANDSTOFPRIJS PER KM ";TAB(32);Z$;" GULDEN"
440   PRINT
450   GOSUB 10290:END
10000 REM CONVERSIE STRING NAAR INTEGER
10010 Z1$=Z$:FLAG%=0:VORZ%=1:Z%=0
10020 IF ASC(Z1$)=ASC("-") THEN VORZ%=-1:Z1$=RIGHT$(Z1$,LEN(Z1$)-1)
10030 L%=LEN(Z1$):IF L%=0 THEN 10140
10040 B%=L%:FOR I%=0 TO L%-1
10050 H$=MID$(Z1$,I%,1)
10060 IF H$="," THEN B%=I%:GOTO 10080
10070 IF H$("0" OR H$)"9" THEN 10140
10080 NEXT I%
10090 Z1$="0"+Z1$+"00000000000"
10100 Z1$=LEFT$(Z1$,B%+1)+MID$(Z1$,B%+2,NK%)
10110 L%=LEN(Z1$):IF L%)11 OR (L%=11 AND Z1$)"02147483647") THEN 10140
10120 FOR I%=0 TO L%-1:Z%=Z%*10:Z%=Z%+(ASC(MID$(Z1$,I%,1))-#30):NEXT
10130 Z%=Z%*VORZ%:RETURN
10140 FLAG%=1:RETURN
10150 REM CONVERSIE INTEGER NAAR STRING
10160 Z1%=Z%:Z$="":VORZ$="":RD%=0
10170 IF Z1%(0 THEN VORZ$="-":Z1%=-1*Z1%
10180 KAPP%=NK%-NKD%-1:IF KAPP%)0 THEN Z1%=Z1%/10^KAPP%
10190 IF Z1% MOD 10)=5 THEN RD%=1
10200 IF NK%)NKD% THEN Z1%=Z1%/10+RD%
10210 Z2%=Z1% MOD 10:Z1%=Z1%/10
10220 Z$=CHR$(Z2%+#30)+Z$
10230 IF Z1%)0 GOTO 10210
10240 Z3%=LEN(Z$):IF Z3%(NKD%+1 THEN Z$="0"+Z$:GOTO 10240
10250 Z$=VORZ$+LEFT$(Z$,Z3%-NKD%)+","+RIGHT$(Z$,NKD%)
10260 Z$=SPC(12-LEN(Z$))+Z$:RETURN
10270 REM PRINTER AAN
10280 POKE #FFF5,#88:POKE #131,0:RETURN
10290 REM PRINTER UIT
10300 POKE #131,1:POKE #FFF5,#C0:RETURN
10310 REM BRANDSTOFVERBRUIK
10320 REM GESCHREVEN IN DAI BASIC V1.1
10330 REM DOOR THEO BOS
10340 REM REF. NP 61 EXAKTE FIXPUNKT RECHNUNG
10350 REM DAInamic DUITSLAND
65400 PRINT :PRINT " ************ E I N D E  L I S T I N G ************"
```

DAI RESET

Reeds eerder is er geschreven over de stoorgevoeligheid van het reset circuit van de DAI.
De ene DAI reageert sneller op storingen van buitenaf dan de andere.
Mij is dan ook al verschillende malen de vraag gesteld: "als mijn koelkast aankomt slaat mijn DAI op tilt, wat moet ik daar aan doen".

Bij gebruikers van de Commodore 1541 Floppy (inmiddels een aardig aantal) komt deze klacht wat vaker voor.
Dit zit echter niet in het door ons ontwikkelde systeem maar vermoedelijk in de totale lengte van de gebruikte kabels.
Het reset signaal gaat n.l.vanaf de DCE Bus via de flatkabel naar de interface en vandaar via de afgeschermde floppy kabel naar de drive.
Hoewel de 1541 voorzien is van een netfilter en kan worden aangesloten op een kontaktdoos voorzien van een randaarde is een verhoogde stoorgevoeligheid met alle nare gevolgen vandien niet uit te sluiten.

Een oplossing is misschien kortere kabels,gebruik maken van een andere geaarde kontaktdoos en als dat niet helpt misschien een kleine verandering in Uw DAI.

Deze verandering bestaat uit het aanbrengen van een condensator van 1 uF/35V over de "RESET" naar aarde.
De condensator is zo'n kleine rode of blauwe tantaalcondensator zoals er zovelen in de DAI zitten.

Waar moet hij komen,nu dat is eenvoudig.
Open Uw DAI (alle kabels eraf natuurlijk en het kabeltje van het metaal rondom het toetsenbord los gemaakt).
U ziet dan recht van het toetsenbord vlak bij de BREAK toest een printbaantje van onder het toetsenbord met een boog omhoog gaan en stoppen bij een boorgat.
Hierin komt de + zijde van de condensator. Links van dit boorgat komt een dikkere printbaan van een blokje (bij mij is het blauw) dit is een weerstandsblokje boven het toetsenbord en stopt ook in een boorgat links van de eerste. Hierin komt de - zijde van de condensator.
U moet NIET het boorgat hebben wat erboven zit of de printbaan aan de rand van de DAI want dat is +5volt.
De tekening helpt U verder.
Deze wijziging is ook uitgevoerd vanaf rev. 7.1 door INDATA.
Ik hoop dat hiermede het probleem is opgelost.

H.Rison

RGB MONITOR

Werkt U ook al jaren met een TV aan Uw DAI en bent U dan ook wel eens jaloers op die mensen die zo'n mooi plaatje op hun RGB monitor hebben.
Nu de RGB monitoren met een medium resolutie goedkoper worden is de aanschaf het overwegen waard.
Maar het is wel oppassen met het geen U koopt.
Wat meestal niet bekend is is dat er twee soorten RGB monitors bestaan n.l. met een analoge en een digitale RGB ingang. De laatste worden vaak IBM compatible genoemd.
Het verschil tussen beide is het volgende:

1. Een analoge RGB monitor ontvangt een drietal analoge signalen welke varieren in spanning.
Hiermede zijn een groot aantal kleuren variaties mogelijk afhankelijk van de amplitude van de aangeboden signalen.

2. Een digitale RGB monitor ontvangt een drietal digitale signalen van TTL niveau wat wil zeggen of +5 volt of 0 volt.
Hierdoor is het aantal kleuren beperkt tot het aantal data bits waarover de processor beschikt bij de DAI maximaal 8.
De DAI kan normaal 16 kleuren leveren en een digitale monitor is dan ook niet direkt geschikt voor de DAI.

Tevens moeten de meeste digitale RGB monitoren 2 sync. signalen ontvangen n.l. horizontale en verticale sync.
De analoge monitor heeft genoeg aan composite sync.
dit is een combinatie van beiden.
De DAI RGB interface kaart is van het analoge type en levert behalve analoge RGB een composite sync. signaal en een audio signaal.
Wil men dus een RGB monitor aansluiten dan moet men dus een DAI RGB interface kaart kopen en deze op de plaats van de normale Video kaart monteren tenzij men een soldeerbout pakt en zelf een RGB kaart bouwt.

Voor de zelfbouwers volgt dan ook een schema met ROM inhoud voor een RGB kaart.
De gegevens in de ROM zijn niet gelijk aan de originale ROM maar geven een betere kleuren weergave.

De werking is als volgt:

De signalen K0,K1,K2,en K3 worden gebruikt voor de kleuren generatie. Met deze 4 bits zijn 16 adres combinaties mogelijk.
Deze 16 combinaties zijn geprogrammeerd in een ROM van het type N82S123 en staan als 8 bits aan de uitgang ter beschikking als het signaal VLB welke als een enable werkt laag gaat.
Deze 8 bits worden dan in z.g.'D'Flipflops geklokt met een het signaal CKL.
De Q uitgangen worden in groepen van 3 via verschillende weerstanden aan elkaar geknoopt in een z.g. summing network.

Deze wordt dan aan de basis van een transistor toegevoerd die dan een spanning afgeeft tussen +5 volt en -5 volt afhankelijk van de spanning aan de basis.
Deze drie groepen vormen dan de Rood,Groen en Blauw uitgang naar de monitor.
Bezit men echter een digitale RGB monitor dan is deze oplossing niet mogelijk.

Inhoud Prom type 82S123

	D7	D6	D5	D4	D3	D2	D1	D0	HEX	ADR
Black	0	0	0	0	0	0	0	0	00	00
Dark Blue	1	1	0	0	0	0	0	0	C1	01
Purple Red	1	0	0	0	0	1	1	1	87	02
Red	0	0	0	0	0	1	1	1	07	03
Purple Red	1	0	1	0	0	1	1	1	A7	04
Emerald Green	0	0	1	1	1	0	0	0	38	05
Khaki Brown	0	1	1	0	1	1	1	1	6F	06
Mustard Brown	0	0	1	0	0	1	0	1	25	07
Gray	1	0	1	0	1	1	0	1	AD	08
Middle Blue	1	1	1	0	0	1	0	0	E4	09
Orange	0	0	1	0	0	1	1	1	27	0A
Pink	1	1	1	0	1	1	1	1	EF	0B
Light Blue	1	1	1	1	0	0	0	0	F0	0C
Light Green	1	0	1	1	1	0	0	0	B8	0D
Light Yellow	0	0	1	1	1	1	1	1	3F	0E
White	1	1	1	1	1	1	1	1	FF	0F

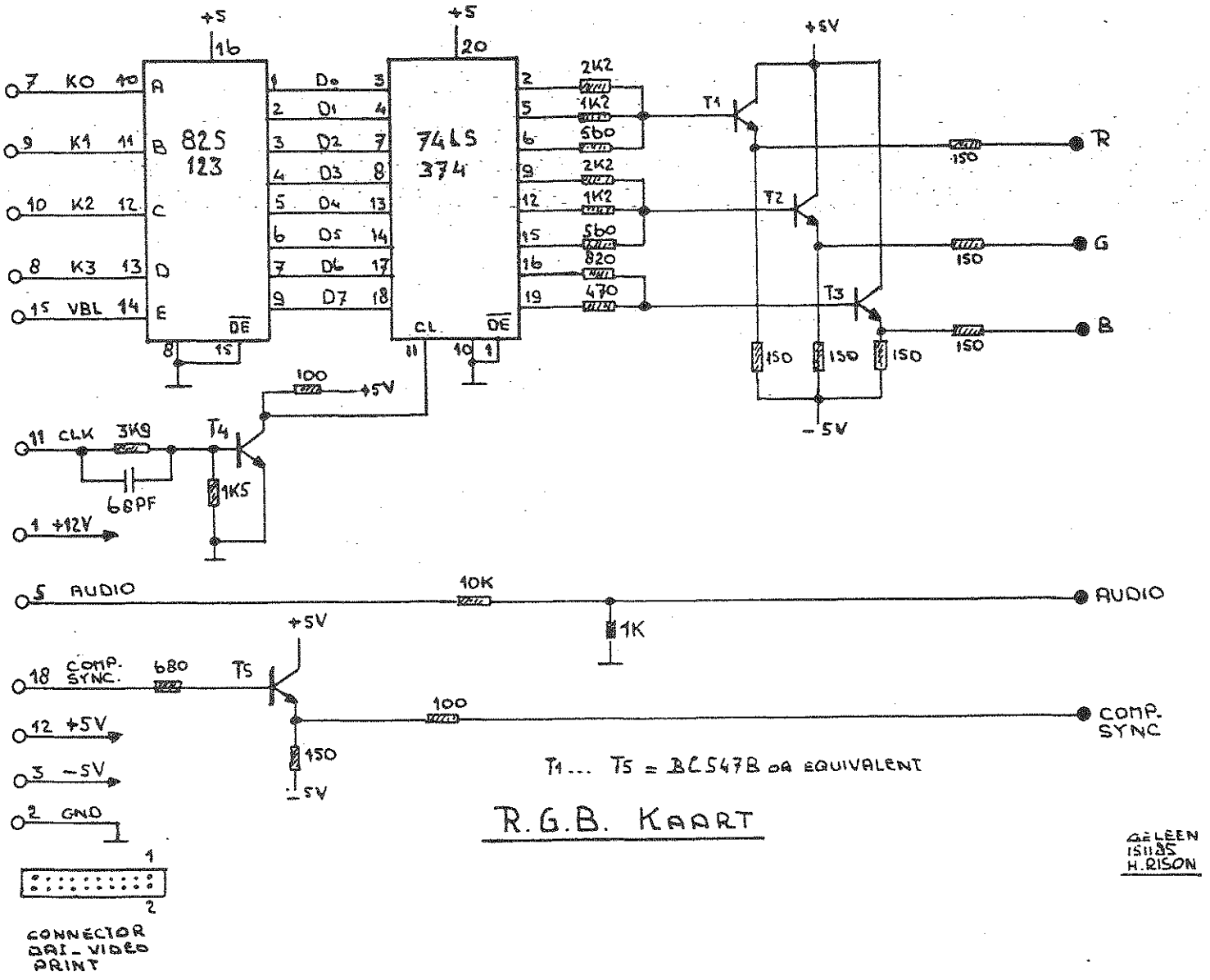
Als een 82S123 eenmaal geprogrammeerd is en de kleuren staan U niet aan dan kan hij helaas niet meer veranderd worden en moet U een nieuwe nemen.
Een alternatief is om een 2716 EPROM te monteren en hiermee te experimenteren totdat U de kleuren die U wenst heeft gevonden.Deze kan dan inplaats van de 82S123 gebruikt worden.
Het audio sigaal kunt U direkt afnemen van de connector.
Het composite sync. signaal wordt via een ermitter-volger aan de uitgang toegevoerd.
De signalen zijn terug te vinden op de volgende pin nrs. van de connector naar de video print:

Pin 2 = Gnd.
Pin 3 = -5V.
Pin 5 = Audio.
Pin 7 = K0.
Pin 8 = K3.
Pin 9 = K1.
Pin 10 = K2.
Pin 11 = CLK.
Pin 12 = +5V.
Pin 15 = VLB.
Pin 18 = Comp.Sync.

De pin nummering begint rechts boven (gezien vanaf het keyboard) met de oneven nrs. aan de bovenzijde en de even nrs. aan de ondezijde.

De mogelijkheid om een digitale monitor aan te sluiten op de DAI wil ik in een volgende uitgave behandelen.
Tevens zal ik dan een vervangingsschema voor een 2716 EPROM hierin op nemen.

H.Rison



R.G.B. KAART

GELEEN
ISIIBS
H.RISON

LEUKE GRAFISCHE TEKENINGEN

```
10    REM PAKHOED (c) DAI
20    REM By Jeroen Overvoorde
30    REM Helmbloem 5
40    REM 3068 AC Rotterdam tel:010-4210426 Nederland
50    REM ------------------------------------------------
60    COLORG 8 3 15 0:MODE 6:FILL 35,50 300,60 21
70    FOR I%=1 TO 50
80    DRAW 120-I%/2,110-I% 205+I%/2,110-I% 21+(I%+53)/100
90    NEXT I%
100   T%=T%+1:IF GETC()32 AND T%<1500 GOTO 100:MODE 0:LOAD
```

****

```
10    REM OMNIVERSUM (c)DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------------
50    COLORG 15 3 8 9:MODE 6:P=7.5
60    FOR I%=42 TO 164 STEP 8
70    FOR Y=I% TO I%+P
80    DRAW 30,Y 300,90+Y 23
90    NEXT Y:P=P-0.5:NEXT I%
100   DRAW 30,170 285,255 23
110   FOR I%=-90 TO 130:Q%=SQR(16900.0-I%*I%)
120   DRAW 0,90+I% 160-Q%,90+I% 20
130   DRAW XMAX,90+I% 160+Q%,90+I% 20:NEXT I%
140   FILL 0,220 XMAX,YMAX 20
150   FOR I%=-50 TO 50:Q%=SQR(2500.0-I%*I%)
160   DRAW 160-Q%,90+I% 160+Q%,90+I% 20:NEXT I%
170   FOR I%=-7 TO 47 STEP 2:Q%=SQR(2209.0-I%*I%)
180   DRAW 160-Q%,90+I% 160+Q%,90+I% 21:NEXT I%
190   T%=T%+1:IF GETC()32 AND T%<1500 GOTO 190:MODE 0:LOAD
```

****

```
10    REM RABOBANK  (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------------
50    COLORG 8 1 15 0:MODE 6
60    FILL 30,30 305,175 21
70    FOR I%=0 TO 10:DRAW 30+I%,30-I% 305-I%,30-I% 21
80    DRAW 30+I%,175+I% 305-I%,175+I% 21:NEXT I%
90    FOR I%=1 TO 37:DRAW 45+I%,25 70+I%,125 22
100   DRAW 290-I%,180 265-I%,90 22:NEXT I%
110   FILL 80,90 255,125 22
120   T%=T%+1:IF GETC()32 AND T%<1500 GOTO 120:MODE 0:LOAD
```

Het mysterie van de verdwenen DIM.

Eind augustus van dit jaar begon mijn DAI kuren te vertonen, een programma dat het "altijd" goed gedaan had deed het niet meer. Na een reset en opnieuw laden was de fout echter verdwenen, dus vooruit met de geit maar weer.
Een paar dagen later trad er weer een fout op een geheel andere plaats op, en weer werd het euvel verholpen door een reset-reload. Maar gaandeweg traden er meer en meer fouten op, ik hoefde soms geen eens meer een reset te doen, omdat het systeem al zo aardig was om dat zelf spontaan te doen. In het systeem hield zich een misdadig element op, dat nergens voor terugdeinsde.
Op een gegeven moment spoorde ik - bij toeval - een fout-situatie op, voordat er rampzaliger gevolgen waren geweest. Bij het LISTen van een programma-deel zag ik dat een BASIC-statement was gemolesteerd: het oude statement "A=127" was veranderd in "A=-2147483521". Door die twee getallen te "behexen" komt het verschil wat beter uit: decimaal 127 is gelijk aan hexadecimaal #7F, terwijl decimaal -2147483521 de hexidecimale waarde #8000007F heeft; om een of andere reden was er dus stiekem één bitje geflipt in het programma.
Omdat ik mijn DAI vlak daarvoor een paar keer met zijn ingewanden had bloot gehad (oa om een weerstand op de RS232-uitgang te overbruggen), en omdat ik zelf uit de software-hoek kom, was voor mij duidelijk waar de verdachte gezocht moest worden: in de harde kern van het systeem, de hardware dus (voor een hardware-collega zou alles echter onmiskenbaar op een fout in de software wijzen). Het simpelste is om alle stekkers en contacten nog maar eens na te lopen en nog eens wat steviger aan te drukken. Dat scheen te helpen, althans voor een kwartiertje. Dezelfde fout sloeg weer onverbiddelijk toe met een spontane reset na het intoetsen van een EDIT-command.

Gaandeweg kwam er een patroon in de opgetreden symptomen, slechts in een beperkt aantal programma's - die nog gekloond waren ook - trad de fout op, en dan meestal na een EDIT-slag. Als in een echte "Krimi" werd het net om de schuldige steeds strakker. Zo bleek dat een nieuwe variabele "PX" na een edit-slag plotseling "%" of "$" te heten, zodat ik de symbol-table eens aan een nader onderzoek heb onderworpen. En ja hoor, in deze achterbuurt trof ik een verdachte snuiter aan: er hield zich tussen de leegstaande krotten een foutieve INT-array schuil. Dit was het gevolg van een ongebruikelijke constructie in het BASIC-programma: in een subroutine stond een statement om de array te laden (6320 LOADA REC RFILE$), echter de array zelf was niet gedimensioneerd met een DIM-statement. Dit werd echter tijdens het draaien niet ontdekt omdat de betreffende subroutine nooit aangeroepen werd.
De fout leek simpel te herstellen door de betreffende subroutine met het gewraakte LOAD-statement uit het programma te verwijderen. Dat was echter niet voldoende; bij het verwijderen van een stuk programma, vindt er geen enkele wijziging in de symbol-table plaats (en dus liep de misdadiger nog steeds op vrije voeten rond). Natuurlijk kan de schuldige er uitgePOKEd worden, maar er bestaat een veel rigoreuzere methode om al het kwaad met wortel en al uit te roeien, en wel door de hele symbol-table schoon te vegen:

```
*CLEAR 20000
*EDIT
<BREAK><BREAK>
*NEW
*POKE #135,2
```

De CLEAR voor het schoonvegen moet een voldoende waarde hebben om het totale programma (in tekst) te kunnen bevatten (op één scherm staan zo'n 500 tot 1000 tekens), anders zal het programma z'n staart verliezen.

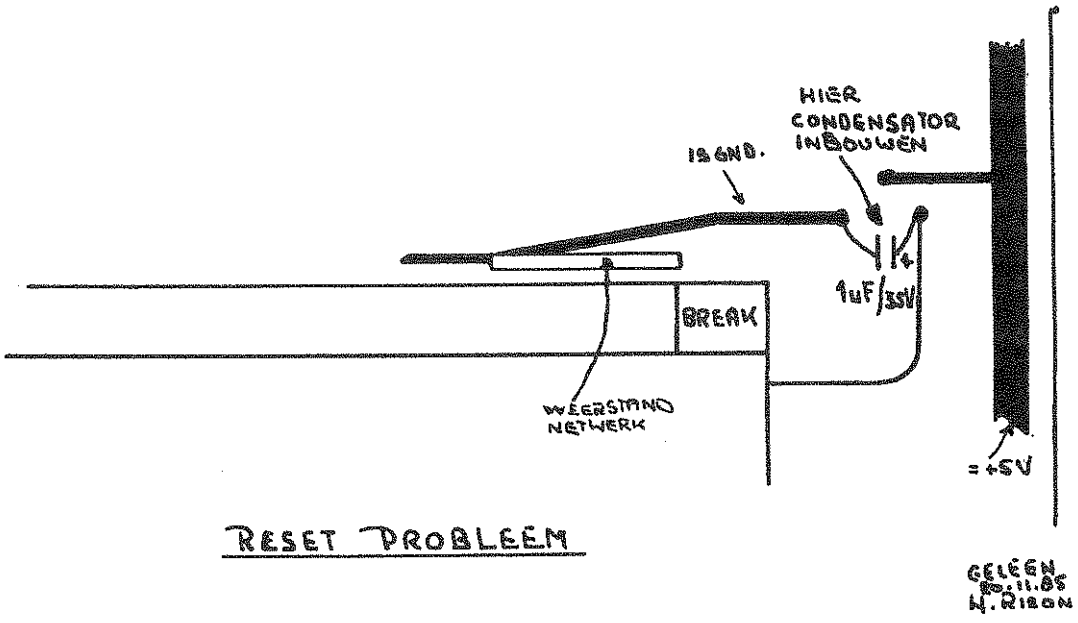
De zinsnede "leegstaande krotten" in de symbol-table is niet zomaar een loze kreet; er schuilt een diepere waarheid in. De symbol-table bevat na verloop van tijd een heleboel vuilnis; omdat bij het SAVEn behalve het programma ook de symbol-table wordt weggeschreven (en bij het LOADen weer opgehaald) blijft die

rommel bestaan. Alleen door een schoonveeg-actie is de zaak op orde te brengen (in het bovengenoemde geval leverde dat een winst van 611 (!!!) bytes op). Het vuilnis ontstaat doordat "oude" variabelen niet meer gebruikt worden. Dat kan bewust zijn gedaan (omdat de variabele inderdaad niet meer gebruikt wordt) of onbewust (omdat de variabele van naam gewijzigd is). Dat laatste kan opzettelijk of onbedoeld gebeurd zijn (bijvoorbeeld als tijdens het intoetsen van een statement de variabele-naam "VARIABLE" is ingetoets als "VARIBALE"; zodra men die fout ontdekt is zal het statement worden geEDIT om er "VARIABLE" van te maken). In de symbol-table echter blijft de oude naam ("VARIBALE") gewoon staan, hetgeen een aantal bytes vuilnis oplevert (bij "VARIBALE" is dat 14 bytes).

Rudy Muller

****

ONDERSTAANDE SCHEMA BEHOORT BIJ HET ARTIKEL DAI RESET VAN HENK RISON OP BLZ. 10.



RESET PROBLEEM

GELEGEN BD. 11.85 H. RISON

Laat ze maar schuiven

Rudy Muller

Een aantal hobby-computers heeft de mogelijkheid om animatie te doen via een zogenaamde "sprite". Zo'n sprite is een grafisch figuurtje dat over het beeldscherm kan worden bewogen zonder de achtergrond-tekening te verstoren. In de meeste gevallen wordt een en ander verzorgd door een aparte video-chip om een schappelijke snelheid te waarborgen.
De hardware van de DAI biedt die mogelijkheid helaas niet. Indien we toch een sprite willen gebruiken in onze programma's zullen we het met behulp van software moeten doen. Aangezien we van BASIC (zelfs DAI-BASIC) geen acceptabele snelheid mogen verwachten, zal het klusje in machine-taal moeten worden opgeknapt.
Een tweede restrictie is dat de sprite maar een beperkte grootte kan hebben. Ik heb gekozen voor 8 * 8 beeldpunten, echter wel met de mogelijkheid dat een aantal beeldpunten doorzichtig mag zijn.
Ten derde kunnen er weliswaar meerdere sprites op het beeldscherm vertoeven, echter als ze met elkaar in botsing komen, vliegen de stukken er letterlijk af. Tenminste in die zin, dat er stukjes verloren kunnen gaan.

In het programma SHIFT komen drie aspecten van het gebruik van sprite's naar voren:
1. de machine-taal routine.
2. de sprite-definitie.
3. de toepassing.

De machine-taal routine MOBJ

De machine-taal routine MOBJ wordt "geladen" door het programma SHIFT (regel 6000 tm 6999) in een INT-array (genaamd MOBJ). Aangezien de plaats in het geheugen van zo'n array geen vast gegeven is, wordt een relocatie-techniek gebruikt (zie daarvoor het betreffende artikel in DAITA 12). Er is eveneens weer gezorgd voor een beveiliging tegen overtype-fouten (regel 6010 tm 6040, zie ook DAITA 12).

De routine MOBJ werkt in alle 4-kleuren text-modes (dus 2[A], 4[A] en 6[A]), en werkt op het beeldscherm via het video-geheugen (....-#BFFF). In feite worden er twee functies geboden: één om een sprite van het beeldscherm te verwijderen, en één om een sprite op het beeldscherm te tekenen. Beide functies kunnen apart worden aangeroepen; beide functies kunnen echter binnen een aanroep ook worden gecombineerd, zodat de sprite over het beeldscherm wordt verschoven.

De machine-taal routine kan worden aangeroepen dmv:

```
CALLM  MOBJ,<parameter>
```

De INT-variabele MOBJ bevat het start-address van de machine-taal routine (gezet op regel 6120); de parameter is het eerste element van een INT-array van negen elementen. De INT-array bevat informatie over scherm-locatie (zowel de oude als een nieuwe), en de sprite zelf:
array(0) : nieuwe positie van sprite (X-coordinaat)
array(1) : nieuwe positie van sprite (Y-coordinaat)
array(2)..(3) : sprite-masker
array(4)..(7) : sprite-kleuren / oude ondergrond
array(8) : huidige positie van sprite (=0: sprite niet op scherm aanwezig).
De positie van de sprite wordt bepaald door de beeldscherm-coordinaten van het beeldpunt waarover het links-onder beeldpunt van de sprite moet komen te liggen.
De aanroeper dient ervoor zorgen dat de sprite (althans het zichtbare gedeelte) binnen het door de mode bepaalde oppervlak van het beeldscherm ligt. Let er op dat in een A-mode YMAX geen betrekking heeft op het zichtbare gedeelte van het

beeldscherm !

Het tekenen van de sprite gebeurd door het adres binnen het video-geheugen te berekenen aan de hand van de opgegeven beelescherm-coordinaten, en op te slaan in array(8), vervolgens wordt de informatie in array(4) tm array(7) verwisseld met de informatie op het beeldscherm (op de positie aangegeven in array(8)). De verwisseling vindt echter alleen plaats voor die beeldscherm-punten waar het sprite-masker "ondoorzichtig" is.
Bij terugkeer bevat array(4) tm array(7) dus de oude ondergrond, en array(8) de huidige positie van de sprite.
(zie regel 2941 tm 2943)

Het verwijderen van de sprite kan geforceerd worden door de waarde 255 op te geven als nieuwe positie van X- en Y-coordinaat. Het wordt eigenlijk op identieke wijze gerealiseerd als het tekenen zelf, dus door het verwisselen van de informatie in array(4) tm array(7) met de informatie op het beeldscherm (op de positie aangegeven in array(8), dus de oude positie van de sprite).
Bij terugkeer bevat array(4) tm array(7) dan weer de oorspronkelijke sprite, en array(8) krijgt de waarde 0 (= geen sprite aanwezig).
(zie regel 3911 en 3912)

Het verschuiven van de sprite gebeurd door de sprite van zijn huidige positie op het beeldscherm te verwijderen, en vervolgens te tekenen op de nieuwe positie.
(zie regel 4031 tm 4033)

De sprite-definitie

Om de vorm van een sprite te definieren is het nodig om een masker te maken. Dit bestaat uit een maximaal 8 * 8 beeldpunten, voor elk beeldpunt moet worden bepaald of de ondergrond zichtbaar is of niet. Hiervoor zijn 8 * 8 bits, ofwel 8 bytes, ofwel 2 INT-array-elementen nodig.
Om de kleuren van de sprite te definieren is het nodig om voor elk van de "ondoorzichtige" punten van het masker het kleur-register (0..3) aan te geven. Hiervoor zijn 8 * 8 bit-paren, ofwel 16 bytes, ofwel 4 INT-array-elementen nodig.

Men kan natuurlijk een sprite ontwerpen, en vervolgens met de hand het masker en de kleurcodes gaan uitrekenen. Het is echter wel zo makkelijk om dit de computer te laten doen.
In het programma SHIFT zit een stukje programmatuur dat het een en ander verwezenlijkt. De sprites worden gedefinieerd in DATA-statements (regel 7200 tm 7299), voor elk punt van de sprite wordt aangegeven of die doorzichtig is (mbv " "), ofwel een kleur heeft (mbv "." ":" "X" en "#").

Het is nuttig om de tekens die een kleur-register weergeven zo te kiezen, dat de DATA-statements al min of meer een indruk geven van de sprite. Voor het programma SHIFT werden de ":" (voor oranje) en de "#" (voor wit) gebruikt (op een zwarte achtergrond). Zo zijn in de programma-listing van SHIFT de plaatjes al duidelijk te herkennen.

Het vastleggen van de sprite-definities in het programma SHIFT wordt verzorgd door regel 1500 tm 1590 en de subroutine op regel 7000 tm 7190. Het geheel is gecompliceerder dan strikt nodig om een sprite-definitie te maken, omdat er enige frutsels bijgebouwd zijn, namelijk:
- het behandelen van vier DATA-sprites tegelijk.
- het tonen van de sprite op de ondergrond in "schaduw"-kleuren.

De volgende subroutine is ontdaan van die frutsels, en het programma-gedeelte

is transparent voor de gekozen tekens voor de kleuren-registers:

```
7000 REM
7001 REM definieer sprite
7002 REM
7010  DIM SPRITE(8)
7020  OP=VARPTR(SPRITE(0)):MP=OP+16:VP=OP+32
7030  READ CC$
7040  READ C0$:READ C1$:READ C2$:READ C3$
7100  FOR I=7 TO 0 STEP -1
7110   MASK=#1:M=0:V1=0:V2=0:READ L$
7120   FOR J=7 TO 0 STEP -1
7130    C$=MID$(L$,J,1)
7140    IF C$<>CC$ THEN M=M+MASK
7141    IF C$=C2$ OR C$=C3$ THEN V1=V1+MASK
7142    IF C$=C1$ OR C$=C3$ THEN V2=V2+MASK
7150    MASK=MASK+MASK
7160    NEXT
7170   MP=MP-1:POKE MP,M
7171   VP=VP-1:POKE VP,V1
7172   VP=VP-1:POKE VP,V2
7180   NEXT
7190  RETURN
7200 REM
7201 REM sprite-data
7202 DATA " ":REM = doorzichtig
7203 DATA ".":REM = kleur 0
7204 DATA ":":REM = kleur 1
7205 DATA "X":REM = kleur 2
7206 DATA "#":REM = kleur 3
7207 REM
7210 DATA "          "
7211 DATA "   ....   "
7212 DATA "  .:XX:.  "
7213 DATA "  .X  X.  "
7214 DATA "  .#  #.  "
7215 DATA "  .:##:.  "
7216 DATA "   ....   "
7217 DATA "          "
```

Een toepassing

Het programma SHIFT is een grafische versie op de DAI van een welbekend schuif-puzzeltje. Het speelveld bestaat uit zestien vierkanten, met daarop vijftien vierkante blokjes (en dus één leeg vakje). Het lege vakje kan worden benut om de vijftien vierkante blokjes heen en weer te kunnen verschuiven. De speler moet door het verschuiven van de blokjes een patroon zien samen te stellen. Bij SHIFT moeten de zijfers (zijfers staat voor zestientallige cijfers) 1 tm F in de goede volgorde worden geplaatst.
De machine-taal routine MOBJ wordt door SHIFT dankbaar gebruikt om de vijftien sprites die de blokjes vertegenwoordigen over het beeldscherm te kunnen verschuiven. Het verschuiven wordt in negen stappen (zie regel 4030 tm 4034) gedaan, zodat de illusie van schuiven wordt gegeven.

Het programma zorgt voor een willekeurige beginstand, waarna de speler door middel van de cursor-toetsen een blokje kan laten verschuiven (in de richting die het pijltje op de toets aangeeft). Behalve het verschuiven zorgt het programma voor het tellen van het aantal verschuivingen en houdt het bij hoeveel blokjes al op de goede plaats liggen. Zodra alle blokjes goed liggen

krijgt de speler een seintje.
Verder kan de speler één (met DELETE CHAR-toets) of meerdere (met TAB-toets) verschuiving(en) ongedaan maken. Helaas ontstaat er daardoor een limiet voor het aantal verschuivingen per speelbeurt, namelijk 1024.
Als een vraagteken wordt ingetoetst, krijgt de speler een tekst gepresenteerd, die precies aangeeft wat het programma van hem/haar verwacht.

Enig commentaar

Om het overtype-werk nog enigszins te beperken zijn in het programma SHIFT maar een gering aantal REM's toegevoegd. Een toegift is echter geen overbodige luxe, vandaar,...

2100 tm 2199: schudden van de blokjes. <==== TIP ... TIP ... TIP
Hiervoor is een elegant algoritme gebruikt dat algemeen toepasbaar is om een aantal eenheden (bv blokjes, kaarten, lotto-balletjes) in een willekeurige volgorde te zetten. (Het probleem schuilt daarbij in het feit dat een eenheid maar één keer getrokken kan worden. In het LOTTO-programma in DAITA 12 wordt dit opgelost door na het trekken van een balletje te kijken of dat balletje al getrokken was).
De gewenste <n> mogelijke waardes worden in een INT-array geplaatst (bij een volgende trekking is dit overigens niet absoluut noodzakelijk), waarna de willekeurige volgorde wordt bewerkstelligd. Als eerste wordt er een willekeurig getal <i> tussen 0 en <n> getrokken, en wordt element <i> verwisseld met element <n>. Vervolgens wordt er een willekeurig getal <ii> tussen 0 en <n-1> getrokken, en wordt element <ii> verwisseld met element <n-1>. Dat proces wordt herhaald tot het gewenste aantal trekkingen <m> is gedaan. Het gewenste aantal trekkingen kan overigens best kleiner zijn dan <n> (bv 7 balletjes van de 41 bij lotto). De laatste <m> elementen van de array bevatten dan de eenheden in een willekeurige volgorde.

2300 tm 2599: onderzoeken en eventueel corrigeren van de beginstand.
In de helft van de gevallen is het onmogelijk om het spel tot een goed einde te brengen. Om dat te onderzoeken wordt eerst het gat naar rechts-onder "verschoven" (2300-2399), vervolgens wordt gekeken hoeveel blokjes er nog niet goed liggen (2400-2499), is dit aantal oneven dan kan het spel niet tot een goed einde worden gebracht. Dat is echter simpel te verhelpen door willekeurig twee vakjes te verwisselen (2500-2599).

3100 tm 3299: terugschuiven.
Alle verschuivingen worden in de INT-array M opgeslagen (byte voor byte om zo de capaciteit van 256 uit te breiden naar 1024), door de cursor-aanslag te POKEn (3440). De verschuiving wordt gePEEKt (3120/3240) en daarna geIXORed met 1, zodat "toevallig" de beweging terug wordt verkregen. De code voor "↑" (#11) wordt omgezet in de code voor "↓" (#10), en vice-versa; hetzelfde gebeurt voor "→" (#12) en "←" (#13). Vervolgens wordt simpelweg de subroutine voor het verschuiven van een blokje (4000-4099) aangeroepen.

5000 tm 5099: tonen van de huidige score. <==== TIP ... TIP ... TIP
Achter de aantallen (5020/5030) wordt een spatie extra geprint. Dit is gedaan om te verkomen dat als het getal beneden een 10-macht daalt, het laatste cijfer achterblijft, zodat er 90 ipv 9 komt te staan.

```
*LIST
100   REM * * * * * * * * * * * * * * * * * * * *
101   REM *                          *  21 okt 84
102   REM *      S H I F T           * - - - - - - - *
103   REM *                          *  08 jun 85  *
104   REM * * * * * * * * * * * * * * * * * * * * *
105   REM *
106   REM *   Ouderwets spel in een DAI-jasje,
107   REM *   waarbij een mozaiek van 15 blokjes
108   REM *   zo verschoven moet worden dat het
109   REM *   gewenste patroon ontstaat.
110   REM *
111   REM * * * * * * * * * * * * * * * * * * * * *
112   REM *
113   REM * (c) Rudy Muller
114   REM *      Jan Steenstr 4
115   REM *      7412 TC   Deventer
116   REM *      tel: 05700 - 18667
117   REM *
1000  CLEAR 5000
1010  POKE #131,1
1020  LIST -999
1100  GOSUB 6000
1200  C0=0:C1=3:C2=9:C3=15:C4=14
1210  COLORG C0 C1 C2 C3
1220  COLORT 0 11 0 0
1230  MODE 2A
1240  PRINT CHR$(12)
1300  DIM M(255):REM gedane verschuivingen
1310  DIM P(3,3):REM P(i,j) = vakje op rij (i) en kolom (j)
1320  DIM Q(15):REM Q(v) = plaats van vakje (v)
1330  DIM R(15):REM R(p) = vakje op plaats (p)
1400  X0=16:Y0=4
1410  FILL X0-3,Y0-3 X0+37,Y0+37 C2
1420  FOR I=0 TO 3
1430  FOR J=0 TO 3
1440  X=X0+9*I:Y=Y0+9*J
1450  FILL X,Y X+7,Y+7 C0
1451  DRAW X+1,Y+1 X+6,Y+6 C3
1452  DRAW X+6,Y+1 X+1,Y+6 C3
1460  NEXT
1470  NEXT
1500  DIM OBJECT(15,8)
1510  DIM L$(15,7)
1520  FOR I=0 TO 7:FOR O=1 TO 4:READ L$(O,7-I):NEXT:NEXT
1530  FOR I=0 TO 7:FOR O=5 TO 8:READ L$(O,7-I):NEXT:NEXT
1540  FOR I=0 TO 7:FOR O=9 TO 12:READ L$(O,7-I):NEXT:NEXT
1550  FOR I=0 TO 7:FOR O=13 TO 15:READ L$(O,7-I):NEXT:NEXT
1590  FOR O=1 TO 15:GOSUB 7000:NEXT
2000  REM
2001  REM start (volgende) spel
2002  REM
2010  PRINT CHR$(12)
2020  NMOV=0:REM number of moves
2030  NCOR=0:REM number of correct placed objects
2040  MPNT=VARPTR(M(0))
2100  REM
2101  REM schud de hele handel
2102  REM
2110  FOR M=0 TO 15:R(M)=M:NEXT
```

```
2120  FOR I=0 TO 3
2130  FOR J=0 TO 3
2140  R=RND(M):P(I,J)=R(R)
2150  IF M>0 THEN M=M-1:R(R)=R(M)
2160  NEXT
2170  NEXT
2200  REM
2201  REM onderzoek beginstand
2202  REM
2210  FOR I=0 TO 3
2220  FOR J=0 TO 3
2230  K=(13-4*J+I) IAND 15
2240  O=P(I,J):R(O)=K:Q(K)=O
2250  NEXT
2260  NEXT
2300  REM
2301  REM schuif gat naar de hoek
2302  REM
2310  P=R(O):IF P=0 GOTO 2400
2320  DP=4:IF P>12 THEN DP=1
2330  N=(P+DP) IAND 15:M=Q(N)
2340  R(O)=N:Q(N)=O
2350  R(M)=P:Q(P)=M
2360  GOTO 2310
2400  REM
2401  REM leg vakjes goed
2402  REM
2410  FLAG=0
2420  FOR P=1 TO 15
2430  M=Q(P):IF M=P GOTO 2480
2440  FLAG=FLAG IXOR 1
2450  S=R(P)
2460  R(M)=S:Q(S)=M
2470  R(P)=P:Q(P)=P
2480  NEXT
2500  REM
2501  REM zorg voor correcte beginstand
2502  REM
2510  IF FLAG=0 GOTO 2900
2520  I0=0:J0=0:IF P(I0,J0)=0 THEN I0=I0+1
2530  I1=3:J1=3:IF P(I1,J1)=0 THEN I1=I1-1
2540  P=P(I0,J0):P(I0,J0)=P(I1,J1):P(I1,J1)=P
2900  REM
2901  REM laat beginstand zien
2902  REM
2910  FOR I=0 TO 3
2920  FOR J=0 TO 3
2930  O=P(I,J):K=13-4*J+I
2940  IF O=0 THEN PX=I:PY=J:GOTO 2950
2941  OBJECT(0,0)=9*I+X0
2942  OBJECT(0,1)=9*J+Y0
2943  CALLM MOBJ,OBJECT(0,0)
2944  IF O=K THEN NCOR=NCOR+1
2950  NEXT
2960  NEXT
3000  REM
3001  REM volgende "move"
3002  REM
3010  P(PX,PY)=0
3020  GOSUB 5000:IF NCOR=15 GOTO 3040
```

```
3030  C=GETC:IF C=0 GOTO 3030
3040  IF C=13 GOTO 3900
3100  REM
3101  REM behandel <CHAR DELETE>
3102  REM
3110  IF C<>8 GOTO 3200
3120  IF NMOV<>0 THEN C=PEEK(MPNT+NMOV) IXOR 1:NMOV=NMOV-1:GOSUB 4000
3130  GOTO 3000
3200  REM
3201  REM behandel <TAB>
3202  REM
3210  IF C<>9 GOTO 3300
3220  PRINT CHR$(12)+"Toets spatie indien U niet verder terug wilt";
3230  IF NMOV=0 GOTO 3000
3240  C=PEEK(MPNT+NMOV) IXOR 1:NMOV=NMOV-1:GOSUB 4000
3250  GOSUB 5000
3260  C=GETC:IF C=0 GOTO 3230
3270  PRINT CHR$(12):GOTO 3000
3300  REM
3301  REM behandel <?>
3302  REM
3310  IF C<>ASC("?") GOTO 3400
3320  PRINT CHR$(12)+"U heeft de volgende mogelijkheden:"
3330  PRINT CHR$(#5E)+" "+CHR$(#8C)+" "+CHR$(#89)+" "+CHR$(#88);"    : schui
 blokje"                                                                    een
3340  PRINT "<char del>: schuif terug       <tab>: schuif 1 voor 1 terug"
3350  PRINT "<return>  : beeindig het spel";
3360  C=GETC:IF C=0 GOTO 3360
3370  PRINT CHR$(12):GOTO 3040
3400  REM
3401  REM behandel <cursor>
3402  REM
3410  IF C<16 OR C>19 GOTO 3500
3420  GOSUB 4000:IF C=0 GOTO 3000
3430  IF C=PEEK(MPNT+NMOV) IXOR 1 THEN NMOV=NMOV-1:GOTO 3000
3440  NMOV=NMOV+1:POKE MPNT+NMOV,C
3450  IF NMOV<1024 GOTO 3000
3460  GOTO 3900:REM too many moves
3500  REM
3501  REM behandel <andere>
3502  REM
3510  IF NCOR<15 GOTO 3020
3900  REM
3901  REM einde spel
3902  REM
3910  FOR I=1 TO 15
3911  OBJECT(I,0)=255:OBJECT(I,1)=255
3912  CALLM MOBJ,OBJECT(I,0)
3913  NEXT
3920  PRINT CHR$(12):PRINT "Nogmaals ?   (Ja/Nee)";
3930  C=GETC:IF C=0 GOTO 3930
3940  IF C=ASC("J") OR C=ASC("j") GOTO 2000
3990  MODE 0:END
4000  REM
4001  REM move object
4002  REM
4010  DX=0:DY=0
4011  ON (C-15) GOTO 4016,4017,4018,4019
4012  C=0:RETURN
```

```
4016  DY=-1:IF PY=0 GOTO 4012:GOTO 4020
4017  DY=1:IF PY=3 GOTO 4012:GOTO 4020
4018  DX=1:IF PX=3 GOTO 4012:GOTO 4020
4019  DX=-1:IF PX=0 GOTO 4012:GOTO 4020
4020  OX=PX:PX=PX+DX:OY=PY:PY=PY+DY
4021  O=P(PX,PY):P(OX,OY)=O
4030  FOR I=1 TO 9
4031  OBJECT(O,0)=XO+9*PX-DX*I
4032  OBJECT(O,1)=YO+9*PY-DY*I
4033  CALLM MOBJ,OBJECT(O,O)
4034  NEXT
4040  IF O=(13-4*OY+OX) THEN NCOR=NCOR+1
4041  IF O=(13-4*PY+PX) THEN NCOR=NCOR-1
4090  RETURN
5000  REM
5001  REM toon score
5002  REM
5010  CURSOR 0,2
5020  PRINT "Aantal verschuivingen : ";NMOV;" "
5030  PRINT "Aantal goed geplaatst : ";NCOR;" "
5040  PRINT "<--- Toets '?' voor hulp --->";
5050  IF NCOR<15 THEN RETURN
5200  REM
5201  REM alles goed
5202  REM
5210  DC=C4:COLORG CO C1 C2 C4
5220  X=XO-2:Y=YO-2
5230  FOR I=0 TO 18:DOT X,Y DC:Y=Y+2:NEXT I
5231  FOR I=0 TO 18:DOT X,Y DC:X=X+2:NEXT I
5232  FOR I=0 TO 18:DOT X,Y DC:Y=Y-2:NEXT I
5233  FOR I=0 TO 18:DOT X,Y DC:X=X-2:NEXT I
5234  IF DC=C4 THEN DC=C2:GOTO 5230
5235  C=GETC:IF C=0 THEN DC=C4:GOTO 5230
5240  COLORG CO C1 C2 C3
5250  RETURN
6000  REM
6001  REM relocate MOBJ
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 #8FFDB6C
6016  IF V<>0 THEN PRINT "!? ASSEMBLY-DATA ERROR":STOP
6020  READ V1:V=V1 IXOR ((V1 SHR 7)+(V SHL 25))
6021  IF V1>=0 GOTO 6020
6022  V=V IXOR #C0000000
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 MOBJ(N)
6110  FOR X=0 TO N:READ MOBJ(X):NEXT X
6120  MOBJ=VARPTR(MOBJ(O))
6130  V=0:V1=VARPTR(V)+2:V2=V1+1
6200  READ X:IF X<0 THEN RETURN
6210  X1=MOBJ+X:X2=X1+1
6220  V=MOBJ+PEEK(X1)+256*PEEK(X2)
6230  POKE X1,PEEK(V2):POKE X2,PEEK(V1)
6240  GOTO 6200
```

```
6900  DATA #53
6901  DATA #C5D5E511,#2001CD15,#013A9800,#26006F22
6902  DATA #3E002B2B,#2B22E600,#CDB10021,#00002242
6903  DATA #013A2201,#1F3A2301,#1F1FE67E,#C60606FF
6904  DATA #2F4F3A27,#01FEFFCA,#52002A88,#00110000
6905  DATA #191FD247,#003F19EB,#29EBB7C2,#41000922
6906  DATA #42013A41,#01473A23,#01324101,#90C604E6
6907  DATA #07EE0711,#83000144,#01814FD2,#6F00040A
6908  DATA #1203130A,#1203130A,#1203130A,#12112801
6909  DATA #26181A00,#00000012,#1325C282,#003A2301
6910  DATA #E607EE07,#3C210100,#293DC298,#002B7D32
6911  DATA #4001CDB1,#00D1D521,#2001CD15,#01E1D1C1
6912  DATA #C93A4001,#32D6002F,#32CD002A,#42017CB5
6913  DATA #C83E0801,#28011130,#01F5C50A,#E60032F4
6914  DATA #0032FA00,#0AE60032,#0601320C,#01CDF200
6915  DATA #2B13CDF2,#00010000,#0913C103,#F13DC2C9
6916  DATA #00C97EE6,#0047AE77,#1AE6004F,#86771A91
6917  DATA #80122323,#7EE60047,#AE771AE6,#004F8677
6918  DATA #1A918012,#C906247E,#12132305,#C21701C9
6919  DATA #00000000,#00000000,#00000000,#00000000
6920  DATA #00000000,#00000000,#00000000,#00000000
6921  DATA #00000000,#0F0F0F00,#00000007,#07070700
6950  DATA #04,#07,#10,#16,#19,#1F,#22,#26
6951  DATA #33,#38,#43,#4C,#50,#53,#57,#5A
6952  DATA #64,#67,#6C,#7E,#8B,#8E,#9B,#A0
6953  DATA #A3,#A8,#AB,#B2,#B5,#B9,#BC,#C4
6954  DATA #C7,#CF,#D2,#D8,#DB,#DE,#E3,#EF
6955  DATA #11D,-1
7000  REM
7001  REM construeer object
7002  REM
7010  OP=VARPTR(OBJECT(0,0)):MP=OP+16:VP=OP+32
7020  X=X0+9*((0-1) IAND 3):Y=Y0+9*((16-0)/4)
7030  FILL X,Y X+7,Y+7 C0
7040  FOR I=7 TO 0 STEP -1
7050  MASK=#1:M=0:V1=0:V2=0
7060  L$=L$(0,I)
7100  FOR J=7 TO 0 STEP -1
7110  C$=MID$(L$,J,1)
7120  IF C$<>" " THEN M=M+MASK
7121  IF C$="X" OR C$="#" THEN V1=V1+MASK
7122  IF C$=":" OR C$="#" THEN V2=V2+MASK
7130  IF C$="#" THEN DOT X+J,Y+I C2
7140  MASK=MASK+MASK
7150  NEXT
7160  MP=MP-1:POKE MP,M
7161  VP=VP-1:POKE VP,V1
7162  VP=VP-1:POKE VP,V2
7170  NEXT I
7190  RETURN
7200  REM objects 1-15:
7201  REM " " = none
7202  REM "." = color-0
7203  REM ":" = color-1
7204  REM "X" = color-2
7205  REM "#" = color-3
7206  REM
7210  DATA ":::::::::",":::::::::",":::::::::",":::::::::"
7211  DATA ":::###::",":::####::",":######:",":::::##::"
```

```
BASIC V1.1

*LIST 7200-
7200  REM objects 1-15:
7201  REM " " = none
7202  REM "." = color-0
7203  REM ":" = color-1
7204  REM "X" = color-2
7205  REM "#" = color-3
7206  REM
7210  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7211  DATA "::::###::::","::::####::::",":::######:",":::::##::::"
7212  DATA "::::###::::",":::##::##:::",":::::::##:",":::::##::::"
7213  DATA ":::::##::::",":::::##::::",":::::::##::",":::##::::::"
7214  DATA ":::::##::::",":::::##::::",":::::::##::",":::##::##:::"
7215  DATA ":::::##::::",":::::##::::",":::::::##::",":::#######::"
7216  DATA "::::####::::",":::######:::",":::#####::::",":::::::##::"
7217  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7220  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7221  DATA ":::######::",":::####::::",":::######:",":::####::::"
7222  DATA ":::##::::::",":::##::::::",":::::::##::",":::##::##:::"
7223  DATA ":::#####::::",":::#####::::",":::::##::::",":::####::::"
7224  DATA ":::::::##::",":::##::##:::",":::::##::::",":::##::##:::"
7225  DATA ":::::::##::",":::##::##:::",":::::##::::",":::##::##:::"
7226  DATA ":::#####::::",":::####::::",":::::##::::",":::####::::"
7227  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7230  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7231  DATA ":::####::::",":::####::::",":::#####::::",":::######::"
7232  DATA ":::##:::##::",":::##:::##::",":::#::::##::",":::##::::::"
7233  DATA ":::##:::##::",":::##:::##::",":::::####::::",":::##::::::"
7234  DATA ":::######::",":::######::",":::#::::##::",":::##::::::"
7235  DATA ":::::::##::",":::##:::##::",":::#::::##::",":::##::::::"
7236  DATA ":::####::::",":::##:::##::",":::#####::::",":::######::"
7237  DATA "::::::::::","::::::::::","::::::::::","::::::::::"
7240  DATA "::::::::::","::::::::::","::::::::::"
7241  DATA ":::#####::::",":::#######:",":::#######:"
7242  DATA ":::#::::##::",":::##::::::",":::##::::::"
7243  DATA ":::#::::##::",":::####::::",":::####::::"
7244  DATA ":::#::::##::",":::##::::::",":::##::::::"
7245  DATA ":::#::::##::",":::##::::::",":::##::::::"
7246  DATA ":::#####::::",":::#######:",":::##::::::"
7247  DATA "::::::::::","::::::::::","::::::::::"
```

****

KREET,

WIE HELPT DE REDAKTIE AAN PLAATJES VOOR DE VOORPLAAT.
U HEEFT VAST WEL EEN OF ANDER LEUK GRAFISCHE PROGRAMMAATJE
IN ELKAAR GEFLANST WAARUIT IETS MOOIS IS ONTSTAAN.
STUUR DAT OP DAN KUNNEN WE DE VOORKANT VAN DE DAITA
ER WEER MEE VERSIEREN.

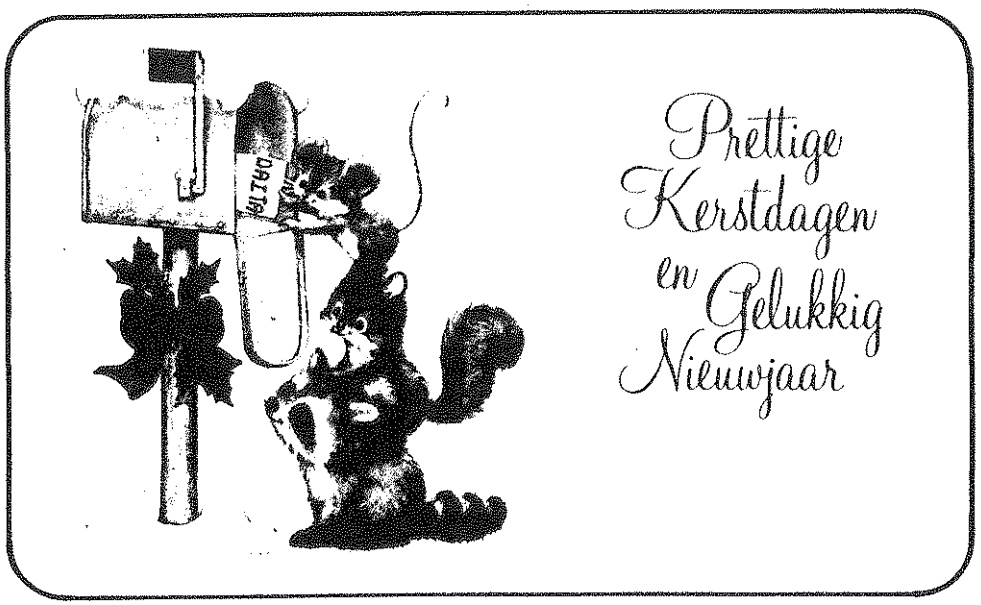
```
SPL V1.1  PAGE 1

0000                    TITL        'Check file V1.0 Anton Doornenbal 841226'
0000            ;
0000            ;
0000  @=02EC SCRPD    EQU         2ECH            ;Start scractch pad
0000  @=0300 ENTRY    EQU         300H            ;Start program part
0000            ;
0000            ;
0000  @=0072 CURSOR   EQU         72H             ;Cursor position address
0000  @=01BE TIMER    EQU         1BEH            ;Timer location
0000            ;                                 ;Writes leader + type on tape
0000  @=02CE ROPEN    EQU         2CEH            ;Detect leader + check on typ
0000            ;                                 ;display it + display file na
0000  @=02D4 RCLOSE   EQU         2D4H            ;Stop motors
0000  @=02D7 MBLK     EQU         2D7H            ;Checksum test data block of
0000  @=02A3 STMEM    EQU         2A3H            ;First free memory location
0000  @=DD55 COL0     EQU         0DD55H          ;If curx<>0 then CRLF,AF corr
0000  @=0070 I7USA    EQU         70H             ;Int 7 vector addres
0000  @=D9A9 VECT7    EQU         0D9A9H          ;Int 7 routine
0000  @=DE1A SUBHD    EQU         0DE1AH          ;HL:=HL-DE , F corrupted
0000  @=005F DTICIM   EQU         5FH             ;Duplicate interrupt mask
0000  @=0040 DPORTM   EQU         40H             ;Duplicate of PORT.(FD06)
0000  @=FD06 PORT     EQU         0FD06H          ;Output port.(bank switching)
0000  @=FFF8 INTMAS   EQU         0FFF8H          ;Interrupt mask port
0000            ;
0000            ;
0000                    ORG         SCRPD           ;Start scratch pad
02EC            RDERR   DS          1H              ;Read error flag, 0 no error
02ED            TIMERR  DS          1H              ;Time out flag, 0 no time out
02EE            TIMMAX  DS          2H              ;Time out setting.( 20mSec.units)
02F0            CURSM   DS          2H              ;Cursor addres start file nam
02F2            STCKPN  DS          2H              ;Stackpointer during CKFILE
02F4            INTM    DS          1H              ;Interrupt mask during CKFILE
02F5            PRTM    DS          1H              ;Port duplicate during CKFILE
02F6            ;
02F6            ;
02F6                    ORG         ENTRY
0300            ;*****************************************************
0300            ;* Checking any file from cassette or from DCR and   *
0300            ;* store it's name into memory starting at the end   *
0300            ;* of basic symbol table.                            *
0300            ;* No check is performed about the filename.         *
0300            ;* If this routine is entered with a string then     *
0300            ;* the filename with it's indentifier will be        *
0300            ;* returned to basic within the 'called' string.     *
0300            ;* A time-out routine is also available to de-       *
0300            ;* tect end off data files.                          *
0300            ;*****************************************************
0300  C5        CKFILE  PUSH B                      ;Save current basic pointer
0301  EB                XCHG                        ;Stringpointer into DE
0302  210000            LXI H       0H
0305  39                DAD SP                      ;Get stackpointer
0306  22F202            SHLD        STCKPN          ;Store stackpointer
0309  D5                PUSH D                      ;Save stringpointer on stack
030A  F5                PUSH PSW                    ;Save variable type on stack
030B  3A5F00            LDA         DTICIM          ;Get current int. mask
030E  32F402            STA         INTM            ;Store it
0311  3A4000            LDA         DPORTM          ;Get port duplicate
0314  32F502            STA         PRTM            ;Store it
```

```
03FB  23              INX H
03FC  5E              MOV E,M
03FD  EB              XCHG
03FE  CDEAE6          CALL      0E6EAH          ;LOOK FOR LINENUMBER IN TEXTBUF
0401  222401          SHLD      124H            ;PNTR TO ADDRES CURRENT DATA LINE
0404  212301          LXI H     123H            ;OFFSET TO NEXT CHAR TO ENCODE
0407  36FF            MVI M     0FFH
0409  F1              POP PSW
040A  E1              POP H
040B  D1              POP D
040C  C1              POP B
040D  C9              RET
040E          ;*****************************************************
040E          ;* ON ERROR GOTO :DAInamic 16 page 172 : N.P.LOOIJE *
040E          ;*****************************************************
040E  0=0006 ERRLIN   EQU       6H              ;FREE BYTES AFTER RST 0
040E  F5              PUSH PSW                  ;SAVE PSW
040F  2A0600          LHLD      ERRLIN          ;CHECK IF LINE TO GOTO
0412  7C              MOV A,H                   ;ON ERROR
0413  B5              ORA L
0414  CA2F04          JZ        OUT             ;QUIT IF 0
0417  210A00          LXI H     0AH             ;OFFSET TO STACKBASE
041A  39              DAD SP                    ;ADD STCKPNTR TO FIND ORIG CALLER
041B  7E              MOV A,M                   ;ORIGINAL CALLER
041C  FE53            CPI       53H             ;CHECK IF THIS IS 0DA53
041E  C22F04          JNZ       OUT             ;IF NOT CONTINUE RST 5
0421  23              INX H
0422  7E              MOV A,M
0423  FEDA            CPI       0DAH
0425  C22F04          JNZ       OUT             ;IF CALLER IS NOT 0DA50
0428  23              INX H                     ;CHECK IF RUNTIME ERROR
0429  7E              MOV A,M
042A  FE40            CPI       40H
042C  CA3304          JZ        ONERR           ;IF NO RUNNING PROGRAM
042F  F1      OUT     POP PSW                   ;RESTORE STACK CONTINU
0430  C3FDC6          JMP       0C6FDH          ;WITH NORMAL RST 5
0433  2A0600  ONERR   LHLD      ERRLIN          ;GET LINENUMBER TO GOTO IN HL
0436  CDF6CA          CALL      0CAF6H          ;FIND THIS LINE IN BASIC
0439  D25B04          JNC       UNDEFL          ;LINE NUMBER NOT FOND
043C  44              MOV B,H                   ;BC START LINE IN BASIC
043D  4D              MOV C,L                   ;TEXT BUFFER
043E  210001          LXI H     100H            ;START OF BASIC POINTERS
0441  3E15            MVI A     15H             ;IN HL,CLEAR EXT.GOSUB AND
0443  3600    LOOP    MVI M     0H              ;FOR NEXT LEVELS
0445  23              INX H
0446  3D              DCR A
0447  C24304          JNZ       LOOP            ;LOOP UNTIL READY
044A  F3              DI
044B  323101          STA       131H            ;OUTPUT TO SCREEN
044E  324000          STA       40H             ;FORCE ROMBANK 0
0451  3206FD          STA       0FD06H
0454  FB              EI
0455  3100F9          LXI SP    0F900H          ;RESET STCKPOINTER CONTINUE
0458  C392C8          JMP       0C892H          ;WITH BASIC EXECUTION
045B  210000 UNDEFL   LXI H     0H              ;IF LINENUMBER IN ERRLIN DOES
045E  220600          SHLD      ERRLIN          ;NOT EXIST ERRORS ARE ENABLED
0461  3E04            MVI A     4H              ;CODE FOR UNDEFINED LINENUMBER
0463  C3F5D9          JMP       0D9F5H          ;PRINT THIS MESSAGE AND BACK
```

```
0000            ;*************************************************
0000            ;*   WRITE FILE ON TAPE, FILE TYPE IS "1"        *
0000            ;*                  BANK 0                       *
0000            ;*************************************************
0000  @=00A4 BLKEND  EQU     0A4H            ;END EDITBUFFER=END BLOCK
0000  @=02C5 WOPEN   EQU     2C5H            ;START MOTOR,WRITE TYP+NAME
0000  @=02C8 WBLK    EQU     2C8H            ;WRITE BLOCK ON TAPE
0000  @=02CB WCLOSE  EQU     2CBH            ;WRITE TRAILER,STOP MOTOR
0000  @=DE1A SUBHD   EQU     0DE1AH          ;HL=HL-DE
0000                 ORG     3B8H
03B8  C5             PUSH B                  ;SAVE BASIC POINTER
03B9  22EE03         SHLD    ADDRES          ;SAVE NAME POINTER
03BC  2AF003         LHLD    BLOK2           ;GET BEGIN BLOK 2
03BF  EB             XCHG                    ;EXCHANGE HL >> << DE
03C0  2AA400         LHLD    BLKEND          ;GET END BLOK 2
03C3  23             INX H                   ;ADD ONE
03C4  CD1ADE         CALL    SUBHD           ;CALC LENGTH BLOK 2
03C7  D5             PUSH D                  ;SAVE BLOK 2 BEGIN
03C8  E5             PUSH H                  ;SAVE BLOK 2 LENGTH
03C9  21F203         LXI H   BLOK1           ;GET BEGIN BLOK 1
03CC  E5             PUSH H                  ;SAVE IT
03CD  110200         LXI D   2H              ;LENGTH BLOK 1
03D0  D5             PUSH D                  ;SAVE IT
03D1  3E31           MVI A   31H             ;FILE TYP = '1'
03D3  2AEE03         LHLD    ADDRES          ;GET ADDRES NAME POINTER
03D6  5E             MOV E,M                 ;LOW BYTE >> REG E
03D7  23             INX H                   ;HL=HL+1
03D8  56             MOV D,M                 ;HIGH BYTE >> REG D
03D9  EB             XCHG                    ;EXCHANGE HL >> << DE
03DA  F3             DI                      ;DISABLE INTERRUPT
03DB  CDC502         CALL    WOPEN           ;START MOTOR,WRITE NAME
03DE  D1             POP D                   ;GET BLOK 1 LENGTH
03DF  E1             POP H                   ;GET BLOK 1 BEGIN
03E0  CDC802         CALL    WBLK            ;WRITE BLOK 1
03E3  D1             POP D                   ;GET BLOK 2 LENGTH
03E4  E1             POP H                   ;GET BLOK 2 BEGIN
03E5  CDC802         CALL    WBLK            ;WRITE BLOK 2
03E8  CDCB02         CALL    WCLOSE          ;WRITE TRAILER,STOP MOTOR
03EB  FB             EI                      ;ENABLE INTERRUPT
03EC  C1             POP B                   ;POINTER TO BASIC
03ED  C9             RET                     ;RETURN TO BASIC
03EE  00       ADDRES  DB      0H              ;NAME POINTER
03EF  00               DB      0H
03F0  00       BLOK2   DB      0H              ;BEGIN BLOK 2
03F1  00               DB      0H
03F2  00       BLOK1   DB      0H              ;BEGIN BLOK 1 IS START-..
03F3  30               DB      30H             ;ADRES BUFFER
03F4            ;*************************************************
03F4            ;* READ LINE : DAITA 4 PAGE 8 : J.P.BLOKKER     *
03F4            ;* SET DATA-POINTER ON A BASIC LINE-NUMBER       *
03F4            ;*************************************************
03F4  C5       SETPTR  PUSH B                  ;ALL
03F5  D5               PUSH D                  ;REGISTERS
03F6  E5               PUSH H                  ;ON
03F7  F5               PUSH PSW                ;STACK
03F8  23               INX H                   ;HL=HL+1
03F9  23               INX H                   ;HL=HL+1
03FA  56               MOV D,M
```

```
0317  CD55DD          CALL      COL0          ;cursor to first column
031A  2A7200          LHLD      CURSOR        ;Get this cursor position
031D  22F002          SHLD      CURSM         ;Store it
0320  2AEE02          LHLD      TIMMAX        ;Get max time
0323  22BE01          SHLD      TIMER         ;Load max time
0326  217F03          LXI H     TIMOUT        ;Get addres TIMOUT
0329  227000          SHLD      I7USA         ;Load it into vect 7
032C  AF              XRA A
032D  32ED02          STA       TIMERR        ;Clear TIMERR
0330  210000          LXI H     0H            ;No requested filename
0333  01FF00          LXI B     0FFH          ;Any file type,display name
0336  CDCE02          CALL      ROPEN         ;Read header
0339  AF              XRA A
033A  4F              MOV C,A                 ;No "name" printing
033B  CDD702          CALL      MBLK          ;Check 1st data block
033E  F5              PUSH PSW                ;Save error 1st block
033F  AF              XRA A                   ;Clear previous error
0340  CDD702          CALL      MBLK          ;Check 2nd data block
0343  21A9D9          LXI H     VECT7         ;Get orginal V7 routine
0346  227000          SHLD      I7USA         ;Disable TIMOUT
0349  47              MOV B,A                 ;Error 2nd block into B
034A  F1              POP PSW                 ;Get error 1st block
034B  B0              ORA B                   ;OR both errors together
034C  32EC02          STA       RDERR         ;Store OR'ed error
034F  CDD402          CALL      RCLOSE        ;Stop motors
0352  FB              EI                      ;Enable interrupts
0353  2A7200          LHLD      CURSOR        ;Get cursor addres after name
0356  EB              XCHG                    ;DE:=after-,HL:=begin of file
0357  2AF002          LHLD      CURSM         ;Get cursor addres file type
035A  E5              PUSH H                  ;Save it for later use
035B  CD1ADE          CALL      SUBHD         ;HL:=filename lenght * 2
035E  7D              MOV A,L                 ;Into accu
035F  1F              RAR                     ;divide it by 2
0360  E63F            ANI       3FH           ;A:=A MOD 64 (Protection only)
0362  47              MOV B,A                 ;B:=length file name
0363  2AA302          LHLD      STMEM         ;Get address filename buffer
0366  D1              POP D                   ;Get cursor addres file type
0367  E5              PUSH H                  ;Save it for later use
0368  77              MOV M,A                 ;Store filename lenght
0369  23    NXTCHR    INX H                   ;Increment buffer pointer
036A  1A              LDAX D                  ;Get character from screen
036B  1B              DCX D
036C  1B              DCX D                   ;Increment screen pointer
036D  77              MOV M,A                 ;Store character into buffer
036E  05              DCR B                   ;Whole name ?
036F  C26903          JNZ       NXTCHR        ;Next char. if not
0372  D1              POP D                   ;Get start file name buffer
0373  F1              POP PSW                 ;Get variable type
0374  E1              POP H                   ;Get string pointer
0375  FE22            CPI       22H           ;String ?
0377  C27D03          JNZ       NOTSTR        ;Skip if not
037A  73              MOV M,E
037B  23              INX H
037C  72              MOV M,D                 ;Stringpointer to filename
037D  C1    NOTSTR    POP B                   ;Remember current basic point
037E  C9              RET                     ;Return to caller
037F          ;*************************************************
037F          ;* This routine enables the user to time which is *
```

```
037F          ;* needed to get a file from cassette or from DCR *
037F          ;* If this time will be to long then the checking *
037F          ;* file routine should be aborted and the timout  *
037F          ;* flag will be set. The time (TIMMAX) units are  *
037F          ;* 20mS. Typicaly times: for DCR 1 Second (#32)   *
037F          ;*                       for CAS 1 Minut  (#BB8)  *
037F          ;*****************************************************
037F  E5      TIMOUT  PUSH H
0380  F5              PUSH PSW                ;Push all
0381  2ABE01          LHLD    TIMER           ;Get time
0384  7D              MOV A,L
0385  B4              ORA H                   ;Zero ?
0386  CA8E03          JZ      ERRORT          ;Goto error routine if true
0389  F1              POP PSW                 ;Restore A + F
038A  E1              POP H                   ;Restore HL
038B  C3A9D9          JMP     VECT7           ;Goto clock int. routine
038E  3D      ERRORT  DCR A
038F  32ED02          STA     TIMERR          ;Set TIMERR flag to FF
0392  21A9D9          LXI H   VECT7           ;Get orginal VECT7 addres
0395  227000          SHLD    I7USA           ;disable TIMOUT
0398  CDD402          CALL    RCLOSE          ;Stop motors
039B  F3              DI                      ;No interrupts
039C  2AF202          LHLD    STCKFN          ;Get old stackpointer
039F  F9              SPHL                    ;restore it
03A0  3AF502          LDA     PRTM            ;Get stored port value
03A3  F630            ORI     30H             ;Disable motors relays
03A5  3206FD          STA     PORT            ;To output port
03A8  324000          STA     DPORTM          ;Update port duplicate
03AB  3AF402          LDA     INTM            ;Get stored int. mask
03AE  32F8FF          STA     INTMAS          ;To int. mask reg.
03B1  325F00          STA     DTICIM          ;Update mask duplicate
03B4  F7              RST 6                   ;Service keyboard + EI
03B5  FF              RST 7                   ;Continue timer routine
03B6  C1              POP B                   ;Restore current basic pointe
03B7  C9              RET                     ;Return to basic
03B8          ;
03B8          ;
03B8          ENDCOD  END                     ;Possible start basic heap
```



NOG ENIGE LEUKE GRAFISCHE PROGRAMMA'S :

```
10    REM TELEAC    (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------
50    COLORG 8 15 0 14:MODE 6
60    FOR P%=65 TO 67:FOR T%=25 TO 27
70    RESTORE:READ A%,B%:FOR I%=0 TO 10
80    READ C%,D%:DRAW A%+P%,B%+T% C%+P%,D%+T% 21:A%=C%:B%=D%
90    NEXT I%:NEXT T%:NEXT P%
100   FOR I%=-100 TO 100:Q%=SQR(10000.0-I%*I%)
110   IF ABS(I%)<97.0 THEN GOTO 130
120   DRAW 166-Q%,126+I% 166+Q%,126+I% 21:GOTO 160
130   Z%=SQR(9409.0-I%*I%)
140   DRAW 166-Q%,126+I% 166-Z%,126+I% 21
150   DRAW 166+Z%,126+I% 166+Q%,126+I% 21
160   NEXT I%
170   T%=T%+1:IF GETC<>32 AND T%<1500 GOTO 170:MODE 0:LOAD
180   DATA 0,0,0,200,200,200,200,0,0,0,200,200
190   DATA 100,200,100,0,200,0,0,200,0,100,200,100
```

****

```
10    REM NEDLOYD   (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------
50    COLORG 8 15 10 9:MODE 6
60    REM WIT
70    FILL 100,5 140,45 21:FILL 59,47 141,87 21
80    FILL 141,89 183,129 21
90    FILL 183,131 265,171 21:FILL 184,173 224,213 21
100   FOR I%=0 TO 40:Q%=SQR(1600.0-I%*I%)
110   DRAW 141+Q%,87-I% 141,87-I% 21
120   DRAW 183+Q%,129-I% 183,129-I% 21
130   DRAW 141-Q%,89+I% 141,89+I% 21
140   DRAW 183-Q%,131+I% 183,131+I% 21:NEXT I%
150   REM ORANJE
160   FILL 101,6 139,44 22:FILL 60,48 141,86 22
170   FILL 141,90 183,128 22
180   FILL 183,132 264,170 22:FILL 185,174 223,212 22
190   FOR I%=1 TO 39:Q%=SQR(1521.0-I%*I%)
200   DRAW 141+Q%,87-I% 141,87-I% 22
210   DRAW 183+Q%,129-I% 183,129-I% 22
220   DRAW 141-Q%,89+I% 141,89+I% 22
230   DRAW 183-Q%,131+I% 183,131+I% 22:NEXT I%
240   T%=T%+1:IF GETC<>32 AND T%<1500 GOTO 240
250   MODE 0:LOAD
```

NOG ENIGE LEUKE GRAFISCHE PROGRAMMA'S :

```
10    REM TELEAC    (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------------------------
50    COLORG 0 15 0 14:MODE 6
60    FOR P%=65 TO 67:FOR T%=25 TO 27
70    RESTORE:READ A%,B%:FOR I%=0 TO 10
80    READ C%,D%:DRAW A%+P%,B%+T% C%+P%,D%+T% 21:A%=C%:B%=D%
90    NEXT I%:NEXT T%:NEXT P%
100   FOR I%=-100 TO 100:Q%=SQR(10000.0-I%*I%)
110   IF ABS(I%)<97.0 THEN GOTO 130
120   DRAW 166-Q%,126+I% 166+Q%,126+I% 21:GOTO 160
130   Z%=SQR(9409.0-I%*I%)
140   DRAW 166-Q%,126+I% 166-Z%,126+I% 21
150   DRAW 166+Z%,126+I% 166+Q%,126+I% 21
160   NEXT I%
170   T%=T%+1:IF GET<>32 AND T%<1500 GOTO 170:MODE 0:LOAD
180   DATA 0,0,0,200,200,200,200,0,0,0,200,200
190   DATA 100,200,100,0,200,0,0,200,0,100,200,100
```

```
10    REM RABOBANK  (c) DAI
20    REM By Jeroen Overvoorde Helmbloem 5
30    REM 3068 AC Rotterdam tel:010-4210426 Nederland
40    REM ------------------------------------------------------------
50    COLORG 0 1 15 0:MODE 6
60    FILL 30,30 305,175 21
70    FOR I%=0 TO 10:DRAW 30+I%,30-I% 305-I%,30-I% 21
80    DRAW 30+I%,175+I% 305-I%,175+I% 21:NEXT I%
90    FOR I%=1 TO 37:DRAW 45+I%,25 70+I%,125 22
100   DRAW 290-I%,180 265-I%,90 22:NEXT I%
110   FILL 80,90 255,125 22
120   T%=T%+1:IF GET<>32 AND T%<1500 GOTO 120:MODE 0:LOAD
```

BESTELLING DAI-GG-SOFTWARE

Te zenden aan Th. Verberkt, Van Buerenstraat 13, 5256 KL Oud-Heusden.

Ondergetekende : _____

Naam : _____

Adres : _____

Postcode/woonplaats: _____

bestelt hierbij de volgende software:

_____ prijs f _____

_____ ,, ,, _____

_____ ,, ,, _____

_____ ,, ,, _____

totaal: f _____

Het totaal-bedrag heeft hij overgeschreven op postrekening 5314900, t.n.v. Th. Verberkt te Oud-heusden.

Handtekening:

BESTELLING DAI-GG-SOFTWARE

Te zenden aan Th. Verberkt, Van Buerenstraat 13, 5256 KL Oud-Heusden.

Ondergetekende : _____

Naam : _____

Adres : _____

Postcode/woonplaats: _____

bestelt hierbij de volgende software:

_____ prijs f _____

_____ ,, ,, _____

_____ ,, ,, _____

_____ ,, ,, _____

totaal: f _____

Het totaal-bedrag heeft hij overgeschreven op postrekening 5314900, t.n.v. Th. Verberkt te Oud-heusden.

Handtekening: