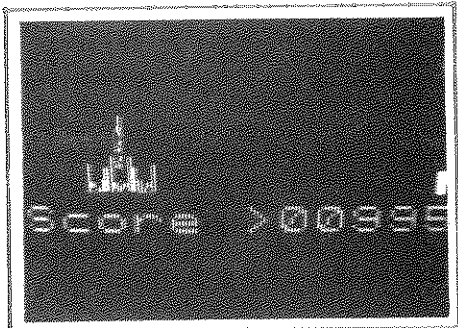


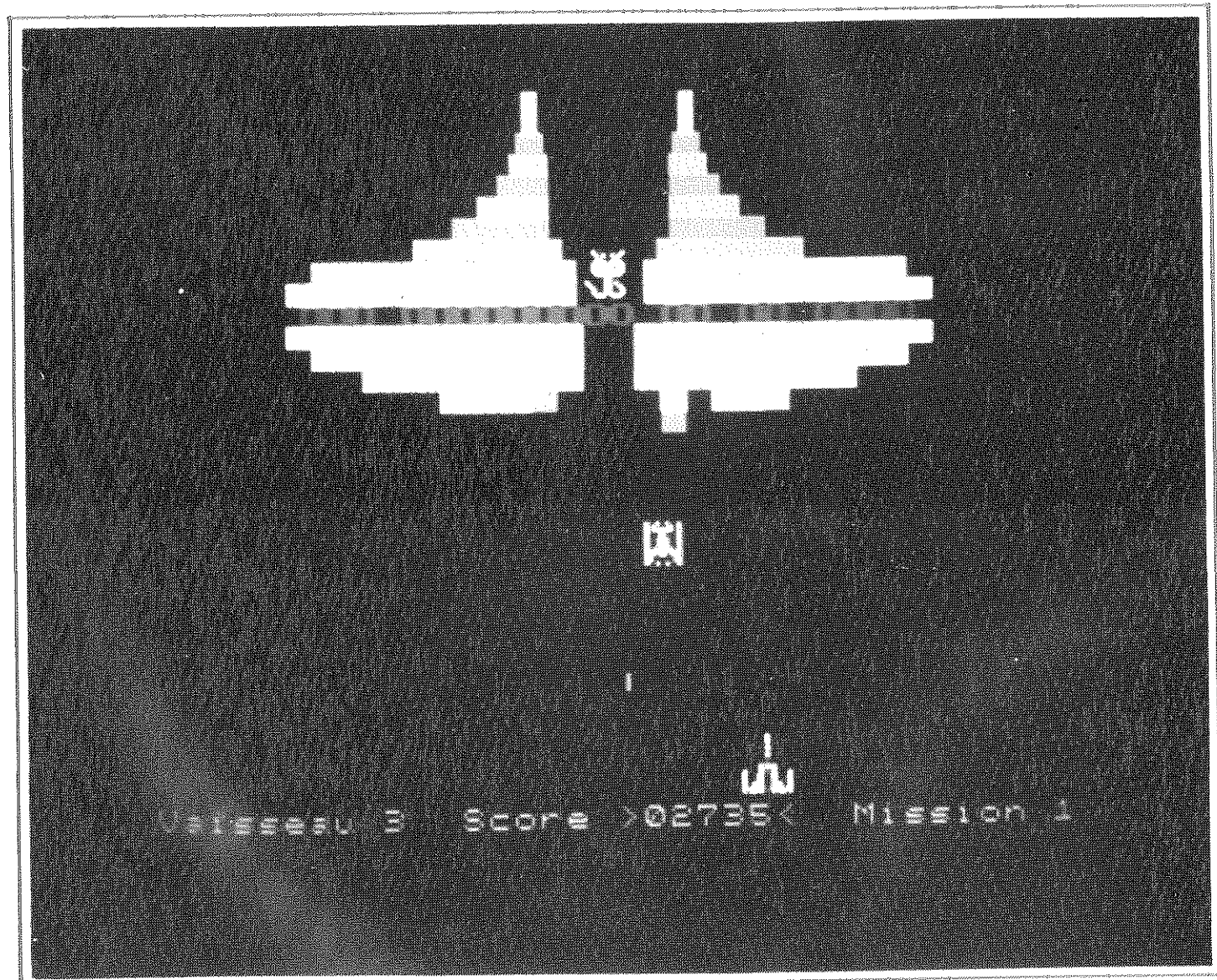
phoenix : mission 4



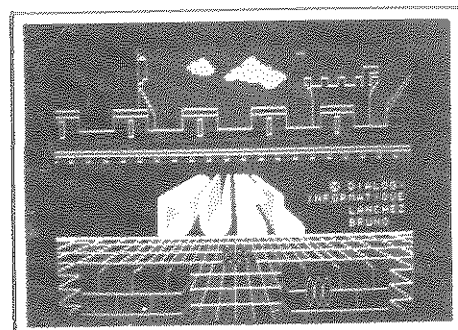
phoenix : close-up



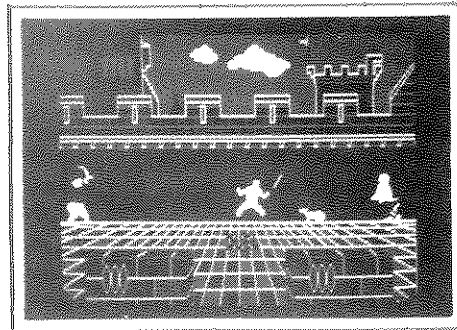
phoenix : score



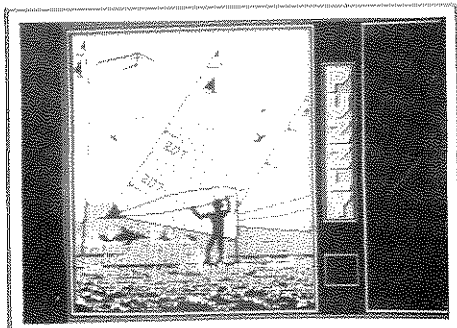
phoenix : final battle



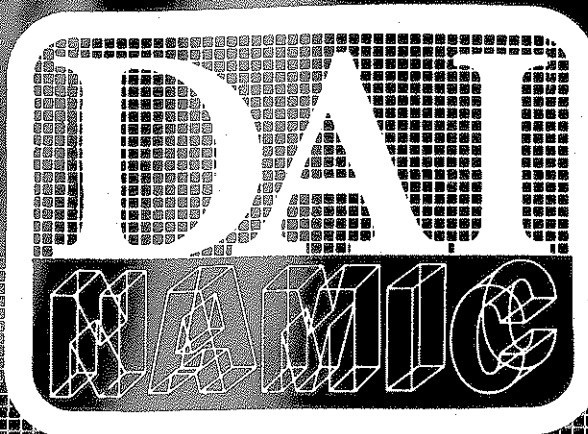
duel



duel



puzzly



L.U.C. CYBER 170/825

CONTROL DATA CORPORATION

GD CONTROL DATA CORPORATION

Communications software for DAI-pc by  
LMPG software group © 1984

personal computer users club

een uitgave van dainamic v.z.w.  
verantw. uitgever w. hermans, mottaart 20 - 3170 herselt

*International*



COLOFON

DAInamic verschijnt tweemaandelijks.

Abonnementsprijs is inbegrepen in de jaarlijkse contributie .

Bij toetreding worden de verschenen nummers van de jaargang toegezonden.

DAInamic redactie :

|                   |                  |
|-------------------|------------------|
| Dirk Bonné        | wdw              |
| Freddy De Raedt   | Herman Bellekens |
| Wilfried Hermans  | Frans Couwberghs |
| René Rens         | Guido Govaerts   |
| Bruno Van Rompaey | Daniël Govaerts  |
| Jef Verwimp       | Frank Druiff     |
|                   | Willy Coremans   |

Vormgeving : Ludo Van Mechelen.

U wordt lid door storting van de contributie op het rekeningnr. **230-0045353-74** van de **Generale Bankmaatschappij, Leuven**, via bankinstelling of postgiro

Het abonnement loopt van januari tot december.

DAInamic verschijnt de pare maanden.  
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans  
Mottaart 20  
3170 Herselt  
Tel. 014/54 59 74

Kredietbank Herselt  
nr. 401-1009701-46  
BTW : 420.840.834

Lidgelden / Subscriptions

Voor Nederland :

Bruno Van Rompaey  
Bovenbosstraat 4  
B 3044 Haasrode  
België  
tel. : 016/46.10.85

GIRO : 4083817  
t.n.v. J.F. van Dunne'  
Hoflaan 70  
3062 JJ ROTTERDAM  
Tel. : (010) 144802

Generale Bankmaatschappij Leuven  
nr. 230-0045353-74

Inzendingen : Games & Strategy

Frank Druiff  
's Gravendijkwal 5A  
NL 3021 EA Rotterdam  
Nederland  
tel. : 010/25.42.75

# DAInamic

PERSONAL COMPUTER USERS CLUB

Herselt , augustus '84

| 4   |       | 3   |      | 2   |     | 1   |     |
|-----|-------|-----|------|-----|-----|-----|-----|
| HEX | DEC   | HEX | DEC  | HEX | DEC | HEX | DEC |
| 1   | 4096  | 1   | 256  | 1   | 16  | 1   | 1   |
| 2   | 8192  | 2   | 512  | 2   | 32  | 2   | 2   |
| 3   | 12288 | 3   | 768  | 3   | 48  | 3   | 3   |
| 4   | 16384 | 4   | 1024 | 4   | 64  | 4   | 4   |
| 5   | 20480 | 5   | 1280 | 5   | 80  | 5   | 5   |
| 6   | 24576 | 6   | 1536 | 6   | 96  | 6   | 6   |
| 7   | 28672 | 7   | 1792 | 7   | 112 | 7   | 7   |
| 8   | 32768 | 8   | 2048 | 8   | 128 | 8   | 8   |
| 9   | 36864 | 9   | 2304 | 9   | 144 | 9   | 9   |
| A   | 40960 | A   | 2560 | A   | 160 | A   | 10  |
| B   | 45056 | B   | 2816 | B   | 176 | B   | 11  |
| C   | 49152 | C   | 3072 | C   | 192 | C   | 12  |
| D   | 53248 | D   | 3328 | D   | 208 | D   | 13  |
| E   | 57344 | E   | 3584 | E   | 224 | E   | 14  |
| F   | 61440 | F   | 3840 | F   | 240 | F   | 15  |

belangrijke ASCII-waarden in DAInp

| functie/symbool  | HEX | DEC |
|------------------|-----|-----|
| back-space       | 8   | 8   |
| TAB              | 9   | 9   |
| linefeed         | A   | 10  |
| clear screen     | C   | 12  |
| CURSOR UP        | 10  | 16  |
| CURSOR DOWN      | 11  | 17  |
| CURSOR LEFT      | 12  | 18  |
| CURSOR RIGHT     | 13  | 19  |
| space-bar        | 20  | 32  |
| Ø                | 30  | 48  |
| A                | 41  | 65  |
| a                | 61  | 97  |
| pijltje rechts   | 89  | 137 |
| pijltje links    | 88  | 136 |
| pijltje boven    | 5E  | 94  |
| pijltje onder    | 8C  | 140 |
| volle blok       | FF  | 255 |
| verticale lijn   | A   | 10  |
| horizontale lijn | B   | 11  |
| 6 hor. lijnen    | 1D  | 29  |

ASCII - HEX - ASCII CONVERSION TABLE

| MSD | 0    | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|-----|------|-----|-----|-----|-----|-----|-----|-----|
| LSD | 000  | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0   | 0000 | NUL | DLE | SP  | 0   | @   | P   | ^   |
| 1   | 0001 | SOH | DC1 | !   | 1   | A   | Q   | a   |
| 2   | 0010 | STX | DC2 | "   | 2   | B   | R   | b   |
| 3   | 0011 | ETX | DC3 | #   | 3   | C   | S   | c   |
| 4   | 0100 | EOT | DC4 | \$  | 4   | D   | T   | d   |
| 5   | 0101 | ENG | NAK | %   | 5   | E   | U   | e   |
| 6   | 0110 | ACK | SYN | &   | 6   | F   | V   | f   |
| 7   | 0111 | BEL | ETB | '   | 7   | G   | W   | g   |
| 8   | 1000 | BS  | CAN | (   | 8   | H   | X   | h   |
| 9   | 1001 | HT  | EM  | )   | 9   | I   | Y   | i   |
| A   | 1010 | LF  | SUB | *   | :   | J   | Z   | j   |
| B   | 1011 | VT  | ESC | +   | ;   | K   | [   | k   |
| C   | 1100 | FF  | FS  | ,   | <   | L   | \   | l   |
| D   | 1101 | CR  | GS  | -   | =   | M   | ]   | m   |
| E   | 1110 | SO  | RS  | .   | >   | N   | ^   | n   |
| F   | 1111 | SI  | VS  | /   | ?   | O   | ~   | o   |

Beste leden,

Mogelijk heb je de laatste weken niet zoveel tijd gespendeerd achter het toetsenbord en zat je ergens in het zuiden van een heerlijk zonnetje te genieten. Dit nummer bevat weer heel wat materiaal dat U kan uitproberen : Frank spit in zijn programmeertechnieken MODE 0 overhoop, Nico geeft een voorbeeldprogramma waarmee je op een vlotte manier uw eigen DATABASE kan creëren : van ARRAY naar EDIT en weer terug zonder problemen. Wij hebben dit programma alvast aangepast om onze artikelen-voorraad in te voeren : het programma berekende voor ons dat er nog 330 pagina's materiaal kant-en-klaar voorradig zijn. Volgende keer willen we deze lijst publiceren, uw suggesties kunnen dan mee de inhoud van ons tijdschrift bepalen. Luc vertelt in zijn brief en zijn programma's hoe hij zijn DAI laat communiceren met mini's zoals CDC en PDP. Armand gaat verder met zijn cursus over microprocessors, als deze reeks afgesloten is gaat hij verder met een speciale reeks over het gebruik van de DCE-bus . Tegen die tijd willen we terug een serie DCE-interface kaarten laten produceren, als U de software-bestelkaart instuurt kan U melden of U later een DCE-interface kaart zou willen bestellen, we zullen dan hiermee rekening houden bij onze bestellingen, de vorige reeks is al lang uitgeput ! Ger geeft nog enige toelichtingen en tips i.v.m. FWP ,voor suggesties en vragen kan U Ger persoonlijk contacteren. Willy brengt in zijn verhaal het vervolg van de DBASIC-specificaties , er zijn ondertussen nog een paar verbeteringen en uitbreidingen aangebracht , DBASIC gebruikers krijgen binnenkort een nieuwe versie toegezonden. Les DAI-istes francophones (et les autres) se rencontrent le 20 octobre a Nivelles : un initiative magnifique de F.Duluins et C.Poels . On peut aussi constater que Cedric a repris les activites DAInamiques en France : la il y a encore beaucoup d'utilisateurs qui ne connaissent pas notre club : il ne doivent plus rester isoles : DAInamic France est la pour les aider ! Aussi pour les francophones , Marc explique les commandes KENDOS (il y a deja plusieurs systemes KENDOS en France !). Wegens plaatsgebek op deze eerste pagina brengen we de rest van ons verhaal in het engels.

tot ziens in Nivelles of Utrecht ...

Dear members,

In this issue you can find a lot of articles and many new splendid software packages. If last year we were 3 years old then we have to celebrate our 4th anniversary this issue. The 3th anniversary software offer was a big success, so why not repeat the action this year ? The formule is slightly different : the reduction you get depends on the total amount of your order :

if your order exceeds 1000 Bfr : reduction is 10 %  
 if your order exceeds 2000 Bfr : reduction is 20 %  
 if your order exceeds 4000 Bfr : reduction is 30 %  
 if your order exceeds 8000 Bfr : reduction is 40 %  
 So : indicate the titles you want, make the total sum and then deduct 10,20,30 or 40 % depending on your total amount. The extra 150 Bfr for DCR-cassettes has to be add afterwards. This is the list with prices of the new packages (audio):

EDUCATION 6 : 1000 Bfr EDUCATION 7 : 1000 Bfr EDUCATION 8 : 1000 Bfr  
 FROBGER : 1250 Bfr EAGLES : 1000 Bfr PHOENIX : 1250 Bfr  
 TANGRAM : 750 Bfr TURTLE-BASIC: 1250 Bfr TOOLKIT 6 : 1000 Bfr

Please allow 2 or 3 weeks for delivery, the offer is valid till 20 october 1984. Good news for graphics freaks : Nico has almost finished the interface and software for digitising pictures from camera or video recorder, first results in next issue ..

DAI dai ...

Wilfried Hermans

|     |                                   |                    |
|-----|-----------------------------------|--------------------|
| 207 | Remark                            | Redactie           |
| 208 | Bladwijzer - contents             |                    |
| 209 | Programmeertechnieken             | Druijff F.         |
| 213 | Array - Edit - Array              | Looije N.          |
| 214 | Solution of mathpuzzle            | Hermans W.         |
| 215 | Brief Luc Maes                    | Maes Luc           |
| 217 | CDC - DAI communication           | Maes Luc           |
| 221 | PDP - DAI communication           | Maes Luc           |
| 225 | Programmering van microprocessors | Beuckelaers A.     |
| 231 | DAInamic France                   | Dufour C.          |
|     | t Programmeertechnieken N16       |                    |
| 236 | t Simulation de GOTO X RUN X      |                    |
| 237 | Logiciels - Forteresse de ZLARG   |                    |
| 238 | Championnat OTHELLO-REVERSI       | OI                 |
| 239 | FWP - tips                        | Gruiters G.        |
| 242 | DBASIC part 2                     | Coremans W.        |
| 250 | 20 oct meeting                    | Duluins - Poels    |
| 251 | Commandes KENDOS                  | Vandermeersch Marc |
| 256 | Tune your flute / guitar          | Zahner E.          |
| 257 | Arrays > 254 (255 ?)              | Boerrigter J.      |
|     | Corrections firmware manual 3     |                    |
| 258 | PROM DATA                         | Meystre A.         |
| 260 | SAVEA & LOADA                     | Vandebergh J.      |
| 263 | Telephon                          | Schall Rolf        |
| 265 | Education 6 - 7 - 8               | diDAIsoft          |
| 266 | Frogger                           | T. Roger           |
| 267 | Toolkit 6                         | different authors  |
| 268 | Phoenix                           | Janin P.           |
| 269 | Tangram                           | Bellekens H.       |
| 270 | Turtle-BASIC                      | Costa W.           |
|     | Eagles                            | Gortz B.           |
| 271 | Software gallery                  |                    |
| 272 | Software gallery                  |                    |

**DAInamic subscription rates :**

Benelux : 1000 Bfr  
 Europe : 1100 Bfr  
 Outside Europe 1500 Bfr  
 (Air Mail)

pay to : Dainamic SUBSCRIPTIONS

B. Van rompaey  
 Bovenbosstraat 4  
 3044 HAASRODE-BELGIUM

\* by check or  
 \* on Bancaccount nr 230-0045353-74  
 of Generale Bank Leuven c/o DAINamic

## PROGRAMMEERTECHNIEKEN

In het tweede deel van de MODE 0 bespreking wil ik beginnen met de scroll te behandelen. Dit begin voornamelijk omdat ik U vorig maal liet zitten met een verwijzing naar dit artikel. Ik POKE'te iets in #8C en #8D waarvan U de reden misschien niet duidelijk was. Als U die betreffende POKE's heeft weggelaten, heeft U gezien dat zij wel degelijk nodig waren.

\*\*\* In het programma PAGE SWAP uit de vorige DAINamic kunt U de regels \*\*\*  
 TIP 100,110 en 120 respectievelijk 200,210 en 220 wel combineren maar TIP  
 \*\*\* het effect is minder fraai. Een voor de hand liggende oplossing is \*\*\*  
 een COLORT 8 8 8 8 van te voren en een COLORT 8 0 0 8 achteraf.  
 Dit werkt echter NIET in een niet standaard geformatteerde MODE 0.

We kunnen de scrollarea, dat is het gebied dat scrollt, zeer eenvoudig tot slechts enkele regels van het beeld beperken. Dit kan handig zijn bij programma's, die een doorlopende 'conversatie' hebben en daarnaast bv een keuze-menu. Ik denk hierbij bv aan een avonturenspeel waarbij steeds verschillende teksten worden gegeven maar waar een inventarislijst steeds bijgewerkt boven in beeld staat en blijft staan. Ook bij administratieve toepassingen kan dit erg praktisch blijken te zijn. De adressen van het gebied, dat wordt gescrolld vinden we in de bytes #8A t/m #8D. Het beginadres staat in #8A en #8B in omgekeerde volgorde, zoals gebruikelijk bij adressen bij een 8080-processor en die heeft de DAI nu eenmaal. Het eindadres analoog in #8C en #8D.

Als we de bovenste regel niet willen laten meescrollen moeten we het adres in #8A,#8B dus vervangen door een adres, dat #86 (= de regellengte) kleiner is. Willen we een of meer regels aan de onderkant unscrolled laten zullen we het adres in #8C,#8D dus met een of meer keren #86 moeten vermeerderen. Het een en ander wordt nu verduidelijkt in een demonstratieprogramma.

```

10 REM SCROLLING AREA / F.H. DRUIJFF / 6/84
20 REM THIS PROGRAM WILL SHOW YOU WHAT YOU
30 REM HAVE TO DO IN ORDER TO GET ONLY A PART
40 REM OF THE SCREEN SCROLLS.
50 TOPSCREEN=#BFEB:LINELENGTH=#86
60 INPUT "LINES UNSCROLLED AT TOP ";TOP:PRINT
70 INPUT "LINES UNSCROLLED AT BOTTOM ";BOTTOM:PRINT
80 IF TOP+BOTTOM>22 THEN PRINT "OOOHHH NO !":GOTO 60
90 LIST
100 T=TOPSCREEN-TOP*LINELENGTH
110 POKE #8A,T MOD 256
120 POKE #8B,T/256
130 B=TOPSCREEN-(24-BOTTOM)*LINELENGTH
140 POKE #8C,B MOD 256
150 POKE #8D,B/256
160 REM WARNING !!!!!
170 REM DO NOT FILL #8A & #8B AND #8C & #8D
180 REM WITH THE SAME DATA.
190 REM ?CHR$(12), MODE 0, LIST, EDIT
200 REM (AS WELL AS [TAB] FOR DCR OWNERS)
210 REM WILL RESTORE THE 'NORMAL' SCROLL.
220 CURSOR 0,23-BOTTOM
230 LIST -:GOTO 230
    
```



Ik hoop dat na de bovenstaande uitleg gelezen te hebben, iedereen nu duidelijk is dat als we het aantal regels in MODE 0 willen beperken en / of uitbreiden, het nodig is om de scrollarea in #8A t/m #8D aan te passen.

No part of this book may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the publisher. Niet uit deze uitgave mag worden vervoerdrukt of openbaar gemaakt door middel van druk, fotocoopie, microfilm of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

De scroll gaat erg vreemde dingen doen als U de adressen niet juist ingeeft. Zoals echter in het laatste programma ook staat wordt de 'normale' MODE 0 weer hersteld door een MODE 0 instructie wat erg voor de hand ligt. Maar ook LIST (niets erachter !) en ?CHR\$(12) en EDIT verzorgen dit. DCR-bezitters kunnen met de [TAB]-toets ?CHR\$(12); (inderdaad met ; ) ingeven en kunnen dat dus gebruiken. Een opmerking, die eigenlijk bij het artikel over PRINT thuishoort, wil ik nu al maken: De CURSOR-instructie werkt relatief ten opzichte van de onderste regel die gescrolled wordt. Zet U de bovenste 5 regels vast, dan is de eerste positie, die tot de scrollarea behoort, te bereiken met CURSOR 0,18. Dit is net zo als we normaal hebben. Zet U echter de onderste 5 regels vast dan is diezelfde beeldpositie ineens te vinden met CURSOR 0,13 !!!

Er blijft voor MODE 0 nog een groot onderwerp over . Het zijn de verschillende lettergroottes. We hebben daar in deze artikelen al eens eerder over gesproken maar toen liet ik het bij wat aanwijzingen voor een aankondiging.

\*\*\* : LIST kan zowel in direct mode als in een programma gebruikt worden. \*\*\*  
 TIP LIST geeft automatisch een schoon MODE 0 beeld. Wilt U dit juist TIP  
 \*\*\* niet dan geeft U een LIST- . U krijgt dan wel de gehele listing \*\*\*  
 maar geen MODE 0:PRINT CHR\$(12); . Ook handig in splitmodes

Nu wil ik laten zien dat U ook een geheel beeld met grotere, grote en reuze letters kunt maken. Eventueel zelfs per regel verschillend. De theorie moet na de vorige onderwerpen niet al te ingewikkeld meer zijn. Elke regel heeft zijn eigen regelcontrolebytes waarvan de eerste steeds in bit 4 & 5 de resolutie dus de grootte aangeeft. Tot op dit moment zijn we uitgegaan van de hoogste resolutie en gebruikten daarvoor 11. Normaal zijn de bytes #BFEF - .. x #86 dan ook #7A. (4 kleuren letters, hoogste resolutie en regelhoogte 11) We kunnen echter de resolutie veranderen door #6A, #5A of #4A in #BFEF te POKE'n. Maar met deze nieuwe lettergroottes hebben we vanzelfsprekend ook andere regellengtes. De regellengtes zijn vreemd genoeg niet identiek aan die van de grafische modes. We zullen het beeld in zijn geheel zelf moeten opbouwen.

Nico Looije maakte reeds een programma, dat alle lettergroottes, inclusief de scroll en EDIT, toestaat. Dit is vooral voor demonstratie doeleinden (school) erg handig. Het zal op een komende toolkit uitgebracht worden.

In de twee voorbeeld programma's heb ik bewust de gehele opbouw verzorgd . Het is namelijk mogelijk om gebruik te maken van MODE 0 ook al gebruiken we hem niet in de standaardvorm. Bij de opbouw moeth we de karakterbytes vullen met spaties (#20 = 32) of het gewenste karakter en de kleurcontrolebyte met 0. Tevens moeten we de regelcontrolebytes op de juiste afstand van elkaar zetten. We kunnen nu snelheidswinst boeken door na MODE 0 eerst de oude regelcontrolebytes te veranderen in #20 en 0 en dan alleen de nieuwe regelcontrolebytes te plaatsen.

```

10 REM CHARACTERS IN SIZES / F.H. DRUIJFF - 5/84
20 MODE 0:PRINT CHR$(12);:POKE #74,0:POKE #75,255:COLORT 8 0 8 15
30 INPUT "TYPE SIZE ";T:ON T GOTO 31,32,33,34:GOTO 30
31 LM=3:NC=60:L=134:P=#7A:GOTO 40
32 LM=2:NC=40:L=90:P=#6A:GOTO 40
33 LM=2:NC=20:L=48:P=#5A:GOTO 40
34 LM=1:NC=10:L=26:P=#4A
40 S=#BFEF:CURSOR 0,0:COLORT 8 8 8 8
50 FOR I=S TO S-23*L STEP -L:POKE I,P:POKE I-1,#40
60 FOR J=I-2 TO I-L STEP -2:POKE J,#20:POKE J-1,0:NEXT:GOTO 50
70 COLORT 8 0 8 15:I=0
100 I=I+1:READ T$:IF T$="" GOTO 100:IF T$="ZZZ" GOTO 200
110 LT=LEN(T$):IF LT>NC THEN T$=LEFT$(T$,NC):LT=NC
120 B=S-I*L-LM*2
130 FOR J=0 TO LT-1:POKE B-J,J,ASC(RIGHT$(T$,LT-J)):NEXT:GOTO 100
200 IF GETC=0 GOTO 200:RESTORE:MODE 0:GOTO 30

```

```

900 DATA 123456789012345678901234567890123456789012345678901234567890
910 DATA THE FIRST LINE IS HERE,THE SECOND LINE
920 DATA TO DEMONSTRATE,VERY CLEARLY
930 DATA IT'S WORKING.,SKIPPING LINES
940 DATA SURELY IS POSSIBLE !!,,,,,CONVINCED ?
950 DATA ONE,TWO,THREE
960 DATA 123456789012345678901234567890123456789012345678901234567890
970 DATA ZZZ

```

Verklaring de gebruikte variabelen : T = Type character  
 LM = Left Margin ; NC = Number of Characters ; L = line Length ; P = Poke info  
 S = Start ; LT = Length of T\$ ; B = Byte to poke ; I & J counters

En nu door elkaar heen in afwisselende groottes.  
 Deze programma's zijn eenvoudig aan te passen aan invoer vanaf het toetsenbord.  
 Probeer U dat ook eens te doen.

```

10 REM DIFFERENT SIZES CHARACTERS / F.H. DRUIJFF - 5/84
20 MODE 0:PRINT CHR$(12);:POKE #74,0:POKE #75,255:COLORT 8 0 8 15
30 S=#BFEF:LC=0
40 S$="123456789012345678901234567890123456789012345678901234567890"
50 LC=LC+1:T=RND(4)+1:ON T GOTO 60,70,80,90
60 LM=6:NC=60:L=134:CB=#7A:GOTO 100
70 LM=4:NC=40:L=90:CB=#6A:GOTO 100
80 LM=2:NC=20:L=48:CB=#5A:GOTO 100
90 LM=0:NC=10:L=26:CB=#4A:GOTO 100
100 POKE S,CB:POKE S-1,#40
110 T$=LEFT$(S$,NC):LT=LEN(T$)
120 FOR J=0 TO LT+LT-2 STEP 2:T=S-J-LM-2:POKE T,ASC(RIGHT$(T$,LT-J/2))
:POKE T-1,0:NEXT
130 S=S-L:IF LC<24 GOTO 50
140 GOTO 140

```

Bij normaal intikken kunt U intikken t/m de zestigste positie.De cursor staat hierna op de een en zestigste, maar als U intikt komt het karakter op de volgende regel. Valse voorlichting van de cursor. Blijft U doortikken kunt U niet zoals veel wordt verondersteld 256 karakters intikken maar slechts 219 (plus [BREAK] of [RETURN]). De BREAK geeft dan trouwens GEEN !. Vervelender is dat in EDIT er wel 256 karakters ingetikt kunnen worden. Sterker nog, er kunnen er veel meer ingetikt worden, die zijn niet op het scherm zichtbaar, maar komen wel in de EDIT-buffer terecht.

\*\*\* Ook als U vergrote karakters heeft gemaakt kunt U soms de tekst \*\*\*  
 TIP op de juiste plaats krijgen door hem domweg te PRINT'en na een TIP  
 \*\*\* CURSOR-instructie. Het bepalen van de juiste cursorpositie is \*\*\*  
 meestal erg ingewikkeld. Er kan beter op goed geluk wat geprobeerd worden. Door een voor de DAI foutief geregeleindadres respectievelijk regellengte zal het vaak onmogelijk blijken.

Als we bepaalde delen van de tekst willen onderstrepen, kan dit op de volgende manier. We creëren een MODE 0 met extra regels zoals het programma PAGE SWAP. Willen we nu een woord onderstrepen, dan zetten we in de minimale-regel-hoogte regel onder het betrokken woord allemaal CHR\$(#FF)'n, die bij minimale regel-hoogte alleen bestaan uit een klein liggend streepje. Pas hier echter mee op, daar de scroll deze regel als normale regel meeneemt. En uw zo extra fraaie tekst veranderd in een grote rotzooi, zeker als U ook andere lettergroottes hebt gebruikt. Aan de andere kant kunt U het geheel nog verfraaien door de streepjes van een andere kleur te maken dan de tekst. Een waarschuwing nog bij het gebruik van een zestienkleuren mode : de kleur kan gecontinueerd worden in het volgende veld. Dus niet klakkeloos voor- en achtergrondkleur combineren.

# ARRAY-EDIT

\*\*\*  
TIP Na UT begint U op een schoon beeld, maar de scroll is niet aangepast. TIP  
\*\*\*

Tot slot wilde ik eigenlijk zonder veel commentaar een aandig 'teken' programma geven waarmee U in MODE 0 tekenen kunt. Het programma is verre van af. Er zitten nog meerdere onvolkomenheden aan. U kunt nauwelijks 'editten' en een tekening die af is kunt U niet bewaren of later reproduceren. Maar al deze zaken kan de geïnteresseerde DAI-er zelf inbouwen. Maakt U een versie die echt goed te gebruiken is (gebruikersvriendelijk) zendt hem mij dan toe. Als hij inderdaad goed is komt hij in een volgende collectie van DAInamic.

```
10      GOTO 200:REM DRAWING IN MODE 0 / F.H. DRUIJFF - 2/84

20      H=GETC:IF H=0 GOTO 20:IF H>47 AND H<57 GOTO 100
30      IF H=8 THEN PRINT " ";
40      IF H=16 THEN Y=Y+1:IF Y>21 THEN Y=21
50      IF H=17 THEN Y=Y-1:IF Y<2 THEN Y=2
60      IF H=18 THEN X=X-1:IF X<0 THEN X=0
70      IF H=19 THEN X=X+1:IF X>59 THEN X=59
80      CURSOR X,Y:GOTO 20
100     I=GETC:IF I=0 GOTO 100
110     PRINT CHR$(H-48)*10+(I-48):GOTO 20

      Initialisatie van het programma.
      De te gebruiken tekens worden met hun ASCII-code in beeld gebracht.

200     MODE 0:PRINT CHR$(12):COLORT 8 0 0 0
210     CURSOR I*3+1,23:PRINT CHR$(I);
220     CURSOR I*3,22:PRINT I;
230     I=I+1:IF I<12 GOTO 210
240     I=13
250     CURSOR I*3+1,23:PRINT CHR$(I);
260     CURSOR I*3,22:PRINT I;
270     I=I+1:IF I<19 GOTO 250

280     CURSOR I*3-56,1:PRINT CHR$(I);
290     CURSOR I*3-57,0:PRINT I;
300     I=I+1:IF I<38 GOTO 280
310     X=30:Y=11:CURSOR X,Y:GOTO 20
```

\*\*\*  
TIP Alle besproken technieken kunnen ook in de splitmodes gebruikt worden. TIP  
\*\*\*

En onder dankzegging voor de tips, die ik kreeg van Nico Looije, Wilfried Hermans en Alain Mariatte, eindig ik met de mededeling dat ik de volgende keer het zal hebben over PRINT. Sommige zaken overlappen elkaar enigzins en ik zal dan misschien bepaalde zaken niet behandeld hebben, die er naar uw idee wel in thuishoren. Ik hoop voor U dat die dan de volgende keer aan bod komen

Frank H. Druijff

Did you ever want to have a userfriendly INPUT routine in your databaseprogram? The program underneath shows you how to use the EDIT facilities of the DAI by using 10 bytes machinelanguage and some POKE's together with PRINT and READ statements. An array will be sent to the editbuffer and then you can manipulate the array. After pressing ~ you will leave the editbuffer leaving you three choices:

1. ~ abort routine no changes will be made
2. RETURN back to the editbuffer
3. other key read the editbuffer into the array

Here is the program together with ! comments!:

```
10      REM EDIT ARRAY N.P.LOOIJE 11/82
20      MODE 0:CLEAR 1000:DIM V$(10,2)
30      PRINT CHR$(12):GOSUB 280
      ! initialise screen!
40      PRINT :INPUT "EDIT ARRAY FROM .. TO .. [0-10]";BOTTOM,
      C TOP
50      REM ----- PRINT ARRAY INTO EDITBUFFER -----
60      POKE #131,2:FOR ROW=BOTTOM TO TOP:FOR COLUMN=0 TO 2
      ! output to editbuffer!
70      PRINT V$(ROW,COLUMN):CHR$(9+COLUMN/2*4):NEXT:NEXT
      ! print arrayelements followed by TAB separators ending
      ! with carriagereturn!
80      PRINT CHR$(0):POKE #131,1
      ! print endmarker chr$(0):output to screen again!
90      REM ----- PERFORM EDIT -----
100     CALLM #F4
      ! initialise editor!
110     KEY=GETC:IF KEY=0 THEN 110:IF KEY<>126 THEN CALLM #F7,
      C KEY:GOTO 110
      ! wait until a key is pressed if it is ~ then abort!
120     PRINT CHR$(12);
130     KEY=GETC:IF KEY=126 THEN 250:IF KEY=13 THEN 100:IF KEY
      C =0 THEN 130
      ! ~ no changes/RETURN restart/OTHER KEY read!
140     REM -----READ EDITBUFFER INTO ARRAY-----
150     PRINT "READING":EDBGN=EDBGN-1
      ! decrement start of editbuffer!
160     FOR ROW=BOTTOM TO 10:COLUMN=0
      ! start at first row and first column of edited area!
170     POKE #291,EDBGN IAND #FF:POKE #292,EDBGN SHR 8
      ! set the READ pointer to the start editbuffer!
180     FOR EDPTR=EDBGN+1 TO #B340:IF PEEK(EDPTR)>#D THEN NEXT
      ! find a separator 0(end), 9(tab) or 13(carret)!
190     IF PEEK(EDPTR)=0 THEN 250
      ! abort if at end of editbuffer!
200     POKE EDBGN,EDPTR-EDBGN-1:POKE #123,#0:EDBGN=EDPTR
      ! store the length before the string to READ (at
      ! the separator):set the inputcount to the first
      ! character:update the start for the next READ!
210     READ V$(ROW,COLUMN):COLUMN=(COLUMN+1) MOD 3
      ! READ:increment column if too much columns reREAD from
      ! column 0!
220     IF PEEK(EDPTR)<>#D THEN 170
```

```

! if separator was not a carret then next column!
230 NEXT ROW
! next row!
240 REM ----- DISPLAY EDITED ARRAY -----
250 FOR ROW=0 TO 10:FOR COLUMN=0 TO 2:PRINT V$(ROW,COLUMN)
C ,:NEXT:PRINT :NEXT
260 END
270 REM ----- INITIALISATION -----
280 FOR MLP=#F4 TO #FD:READ BYTE:POKE MLP,BYTE:NEXT
! mlp in unused area!
290 DATA #EF,#2A,#C9, #23,#23,#23,#7E,#EF,#2D,#C9
! RST 5 DATA 2A,get chr from variable into editbuffer!
300 EDBGN=PEEK(#2A3)+PEEK(#2A4)*256+2
! start editbuffer after BASIC!
310 POSTAB=#6210000:VPTAB=VARPTR(POSTAB)
! tabpositions in variable 6,33 always end with 0!
320 POKE #B4,VPTAB IAND #FF:POKE #B5,VPTAB SHR 8
! store begin tabtable!
330 POKE #A2,EDBGN IAND #FF:POKE #A3,EDBGN SHR 8
! store begin buffer!
340 POKE #A4,EDBGN IAND #FF:POKE #A5,EDBGN SHR 8
! inputpointer editbuffer to start buffer!
350 POKE #A6,#40:POKE #A7,#B3
! end of editarea just below screen!
360 REM -----DATA FOR DEMO -----
370 FOR COLUMN=0 TO 5:FOR ROW=0 TO 2
380 READ V$(COLUMN,ROW):PRINT V$(COLUMN,ROW):NEXT:NEXT:RET
C URN
390 DATA B,TC MATIC,DUE PASA
400 DATA D,ALPHAVILLE,BIG IN JAPAN
410 DATA GB,NEW ORDER,THIEVES LIKE US
420 DATA F,JULIEN CLERC,LILI VOULAIT AIMER DANSER
430 DATA I,AL BANO & ROMINA POWER,CI SARA
440 DATA NL,STAR SISTERS,HOORAY FOR HOLLYWOOD
! international data!

```

N.B. If You want to use a READ statement again you must RESTORE. The program can easily be modified for numeric arrays and different dimensions.

N.P.Looije

**SOLUTION OF MATH'PUZZLE (newsletter 84-21 p. 135)**

```

6 = (1 + 1 + 1)! (faculty)
6 = 2 + 2 + 2
6 = (3 x 3) - 3
6 = SQR(4) + SQR(4) + SQR(4)
6 = 5 + (5/5)
6 = (6 x 6 )/6
6 = 7 - (7/7)
6 =  $\sqrt[3]{8} + \sqrt[3]{8} + \sqrt[3]{8}$ 
6 = (SQR(9) x SQR(9))-SQR(9)
6 = (LOGT(10)+LOGT(10)+LOGT(10))! (faculty)

```

LMPG Software  
p.a. Luc Maes  
Collegestraat 60 E  
B 2300 Turnhout

DAInamic  
Mottaart 20  
Herselt

Geachte heer

naar aanleiding van het programma "de DAI als intelligente terminal" en de mogelijkheden die ons werden geboden op het Limburgs Universitair Centrum, hebben we onze DAI kunnen aansluiten op het mainframe van de universiteit. Deze computer is een Cyber 170 series 825.

Op de universiteit werd ons al snel een poort op 1200 baud toegewezen. Deze snelheid was ideaal voor het programma uit newsletter 10.

Dit programma bleek overigens uitstekend te werken. De operator en de systeem-analist waren zeer verbaasd over de eenvoud van de hard- en software. Men was namelijk al twee weken bezig om hetzelfde klaar te spelen voor een apple, maar men bleek steeds weer moeilijkheden te hebben met software die niet kon gebruikt worden op de gekochte interfacekaarten.

Na wat uitzoeken en proberen hebben we het "terminal"-programma wat kunnen versnellen en uitbreiden, zodat het nu mogelijk is om op 9600 baud te werken en men kan nu ook files van het systeem in de edit-buffer opladen.

Ook heb ik ervoor kunnen zorgen dat er geen terminaldefines meer nodig zijn om gewoon interactief te werken. Om een mooiere output te krijgen is het wel handig om de paginabreedte op 60 te zetten, zodat men geen continuation lines krijgt. Op het CDC-NOS systeem kan dit met het commando TRMDEF,PW=60 (PW staat voor page width)

Verder zijn er geen andere terminaldefines nodig. De DAI verliest geen karakters meer, zelfs niet bij 9600 baud. Dit is ons gelukt door interrupt 4 te activeren en een andere interrupt routine te schrijven, zodat nu telkens de rs-232 poort een karakter heeft ontvangen, dit dadelijk in een circular buffer wordt geschreven. (De buffer loopt van #300 tot #3FF.) Het hoofdprogramma heeft hierdoor meer tijd om de karakters te verwerken.

Het toetsenbord van de DAI is niet erg geschikt om op een mainframe te werken. Daarom hebben we enkele aanpassingen aangebracht. De DAI mist een ctrl-toets (de toets staat er wel, maar is geen echte control-toets maar een shift-lock.). In ons programma hebben we de +-toets als ctrl-toets gedefinieerd. Om bv. ctrl-G te tikken moet men +-G tikken zoals men shift-G zou gebruiken. Dus de +-toets ingedrukt houden. De +-toets hebben we gedefinieerd als escape.

Ook de +-toets heeft een speciale functie: hiermee kunnen files in de edit-buffer opgeladen worden.

Wanneer men die toets indrukt, stuurt de DAI een carriagereturn-linefeed door naar het mainframe en stopt van dan af alle ontvangen karakters in de editbuffer.

De editbuffer start vlak na het programma en kan gaan tot de heap. Om deze mogelijkheid te gebruiken zet men dus de heap met behulp van de pointer op #29B en op #29C zover mogelijk naar achter in de RAM.



Dan maakt men alles klaar om de gewenste file op het scherm uit te listen, en in plaats van return drukt men cursor-+. De DAI laadt dan karakters in het geheugen tot hij een ctrl-G tegenkomt. Op de CDC duidt dit het einde van de file aan. Misschien moet men voor andere systemen een ander controle karakter nemen, control-G laat de terminal even piepen. Het is wel best om voor dat men de file inlaadt, dat men de paginabreedte op oneindig zet, omdat anders om de 60 of 80 karakters een CR mee opgeladen wordt. Dit geeft dat lijnen die langer als 60 of 80 letters zijn in stukken gekapt worden, wat bij BASIC-programma's voor de nodige errors zorgt. Op het CDC-NOS systeem kan dit gedaan worden met het commando TRMDEF,PW=0. Wanneer men vrij veel files inleest en wegschrijft kan men de paginabreedte best op oneindig laten staan. Na het opladen kan men naar BASIC terugkeren (BREAK) en dan het ingeladen BASIC-programma in de textbuffer van de DAI plaatsen met POKE #135,2. Om een BASIC-programma van de DAI in een file op het mainframe te stoppen, gaat men in de editor van het mainframe, men keert terug naar BASIC (BREAK) en tikt dan : POKE #131,0:LIST1-:CALLM #400 Na deze handelingen heeft de DAI het BASIC programma in de file "ingetikt". Men kan de editor dus terug verlaten en de file save op hard-disk of 9track tape. Het is wel interessant te bedenken dat deze manier van save van BASIC-programmas veel sneller is dan met een cassette-recorder. Zoals uit deze uitleg al gebleken is, is deze routine nogal specifiek voor de CDC. Toch denken we ze bruikbaar is op andere mainframes zonder grote veranderingen. Indien andere leden van de club problemen met dit programma hebben, kan men ons steeds proberen te contacteren. Hier volgt nog een listing van het programma met de nodige uitleg. Het programma is misschien niet helemaal 'elegant' geschreven, maar het werkt prima.

veel success ermee

LMPG-soft

*Maes*  
Luc Maes  
Collegestraat 60E  
B 2300 Turnhout  
014 / 41 32 54

*Gobert*  
Paul Gobert  
De Bergen 49  
B 2241 Zoersel  
03 / 383 29 72

# PDP/CDC

SPL V1.1 PAGE 1

CDC-communicatie

```

0000          TITL      'CDC-communicatie'
0000 @=FFF0 SERIN EQU  0FFF0H      ;serial input buffer
0000 @=0131 OUTPSW EQU  131H
0000 @=FFF8 INTMSK EQU  0FFF8H      ;interrupt mask
0000 @=D6BE GETC  EQU  0D6BEH
0000 @=DD94 OUTPUT EQU  0DD94H      ;serial output routine
0000 @=C818 BASIC EQU  0C818H
0000 @=00A0 BDRATE EQU  0A0H        ;4800baud
0000 @=FD06 BANK  EQU  0FD06H
0000 @=0040 DUBANK EQU  40H
0000 @=E4BC SOUND EQU  0E4BCH      ;sound routine
0000 @=01BE WAIT  EQU  1BEH        ;WAIT TIME timer
0000 @=DB32 PRINT EQU  0DB32H      ;print string
0000 @=DE14 COMPAR EQU  0DE14H      ;vergelijk HL en DE
0000          ORG      400H
0400          ;
0400          ;*****
0400          ; Start en initialisatie
0400          ;
0400 3E01 START MVI A 1H
0402 323101 STA OUTPSW      ;output naar scherm alleen
0405 AF XRA A
0406 32B902 STA 2B9H        ;alle toetsen scannen
0409 3EA0 MVI A BDRATE
040B 32F5FF STA 0FFF5H      ;zet baudrate
040E 3EF0 MVI A 0F0H
0410 3206FD STA BANK        ;ROM-bank 4
0413 324000 STA DUBANK
0416 F3 DI
0417 3A5F00 LDA 5FH        ;Duplicaat van interrupt mask
041A 327805 STA MSKSAV
041D 3ED5 MVI A 0D5H
041F 325F00 STA 5FH        ;interrupt 4 toelaten
0422 32F8FF STA INTMSK      ;(4=serial buffer loaded)
0425 21C504 LXI H INTR4
0428 226A00 SHLD 6AH        ;nieuwe interrupt routine
042B FB EI
042C 3E0C MVI A 0CH
042E EF RST 5              ;scherm leegmaken
042F 03 DB 3H
0430 3EFF MVI A 0FFH
0432 32C302 STA 2C3H        ;zet SHIFT-LOCK
0435          ;
0435          ;*****
0435          ; Mainloop : ingedrukte toets verzenden
0435          ;
0435 3ABE01 KEYS LDA WAIT      ;beeptijd voorbij ?
0438 A7 ANA A
0439 C24204 JNZ KEYS2
043C CDBFD9 CALL 0D98FH      ;Disable sound interrupts
043F CDA6D8 CALL 0D8A6H      ;Stop oscillators
0442 CDBED6 KEYS2 CALL GETC
0445 DAA904 JC BREAK        ;indien BREAK, naar BASIC
0448 A7 ANA A
0449 CA7504 JZ PORT        ;geen toets, dan ontvangen bytes verwerken
044C FE13 CPI 13H
044E CA0005 JZ LOAD        ;cursor rechts, file laden
0451 FE11 CPI 11H

```

```

0453 CA7504      JZ      PORT      ;cursor down is nu ctrl
0456 FE12       CPI      12H      ;cursor links is nu escape
0458 C25D04     JNZ      NESC
045B 3E1B       MVI A    1BH      ;escape
045D 47         NESC     MOV B,A
045E F3         DI
045F 3E02       MVI A    2H      ;Check of control
0461 3207FF     STA      0FF07H   ;2e rij
0464 3A01FF     LDA      0FF01H
0467 FB         EI
0468 17         RAL
0469 17         RAL      ;kijk of cursor down ingedrukt is
046A 3E7F       MVI A    7FH     ;        6e bit
046C D27104     JNC      NCTRL   ;strip parity bit als geen ctrl
046F 3E1F       MVI A    1FH     ;strip hoogste 3 bits als wel ctrl
0471 A0         NCTRL   ANA B
0472 CD94DD     CALL     OUTPUT  ;serieel verzenden
0475           ;
0475           ;*****
0475           ; Mainloop : ontvangen bytes verwerken
0475           ;
0475 3A7605     PORT    LDA      WBUF      ;laatst ontvangen byte
0478 47         MOV B,A
0479 3A7705     LDA      RBUF      ;laatst verwerkte byte
047C BB         CMP B
047D CA3504     JZ      KEYS      ;zelfde?
0480 3C         INR A
0481 327705     STA      RBUF      ;ja, dan terug naar toetsen kijken
0484 6F         MOV L,A
0485 2603       MVI H    3H
0487 7E         MOV A,M   ;haal 3**H (**=RBUF)
0488 FE1F       CPI      1FH
048A DA9204     JC      CTRL   ;ctrl-chars niet op scherm zetten
048D EF         TEMP    RST 5
048E 03         DB      3H      ;andere wel op scherm zetten
048F C37504     JMP      PORT      ;volgende byte verwerken
0492 FE0D     CTRL    CPI      0DH   ;carriage return?
0494 CA8D04     JZ      TEMP    ;wel op scherm zetten
0497 FE07       CPI      7H      ;control-G?
0499 CAE204     JZ      BEEP    ;dan piepen
049C FE0C       CPI      0CH   ;HOME?
049E CA8D04     JZ      TEMP    ;wel op scherm zetten
04A1 FE08       CPI      8H      ;CHAR-DEL?
04A3 CA8D04     JZ      TEMP    ;wel op scherm zetten
04A6 C33504     JMP      KEYS      ;verder werken
04A9           ;
04A9           ;*****
04A9           ; Terug naar BASIC
04A9           ;
04A9 F3         BREAK   DI
04AA 3A7805     LDA      MSKSAV
04AD 325F00     STA      5FH      ;restore interrupt mask
04B0 32F8FF     STA      INTMSK
04B3 21C0C6     LXI H    0C6C0H
04B6 226A00     SHLD   6AH      ;restore interrupt vector
04B9 FB         EI
04BA 3E30       MVI A    30H
04BC 3206FD     STA      BANK    ;restore ROM-bank

```

```

04BF 324000     STA      DUBANK
04C2 C318C8     JMP      BASIC    ;naar BASIC-monitor
04C5           ;
04C5           ;*****
04C5           ; Interrupt routine voor intr. 4 RECEIVE BUFFER LOADED
04C5           ;
04C5 F5         INTR4   PUSH PSW
04C6 C5         PUSH B
04C7 3AF0FF     LDA      SERIN   ;haal ontvangen byte boven
04CA A7         ANA A
04CB CADD04     JZ      NONE    ;? laat dan maar
04CE 47         MOV B,A
04CF 3A7605     LDA      WBUF   ;Laatste in buffer (1 byte)
04D2 3C         INR A    ;Volgende plaats
04D3 327605     STA      WBUF
04D6 6F         MOV L,A
04D7 2603       MVI H    3H      ;Buffer (h byte) 3**H **=WBUF
04D9 78         MOV A,B
04DA E67F       ANI      7FH   ;strip parity bit
04DC 77         MOV M,A   ;zet byte in buffer
04DD C1         NONE    POP B
04DE F1         POP PSW
04DF E1         POP H
04E0 FB         EI
04E1 C9         RET
04E2           ;
04E2           ;*****
04E2           ; Verwerking van control-G : BEEP
04E2           ;
04E2 015005     BEEP    LXI B    BBEEP   ;pointer naar beep commando
04E5 3E30       MVI A    30H
04E7 324000     STA      DUBANK   ;BASIC ROM-bank
04EA 3206FD     STA      BANK
04ED CDBCE4     CALL   SOUND    ;voer SOUND commando uit
04F0 3EF0       MVI A    0F0H
04F2 324000     STA      DUBANK   ;terug naar ROM-bank 4
04F5 3206FD     STA      BANK
04F8 3E14       MVI A    14H   ;tijdsduur van beep
04FA 32BE01     STA      WAIT    ;in de timer stoppen
04FD C37504     JMP      PORT    ;verder werken
0500           ;
0500           ;*****
0500           ; Laad een file in de EDIT-buffer
0500           ;
0500 3E0D     LOAD   MVI A    0DH   ;carriage return
0502 CD94DD     CALL   OUTPUT  ;serieel doorsturen
0505 EF         RST 5
0506 03         DB      3H      ;en op scherm zetten
0507 117A05     LXI D    EBUF   ;start van de EDIT-buffer
050A 3A7605     LOAD2  LDA      WBUF
050D 47         MOV B,A
050E 3A7705     LDA      RBUF
0511 BB         CMP B
0512 CA0A05     JZ      LOAD2   ;char ontvangen?
0515 3C         INR A    ;nee, wachten
0516 327705     STA      RBUF   ;ja
0519 6F         MOV L,A   ;ja
051A 2603       MVI H    3H      ;leespointer 1 verder

```



```

051C 7E      MOV A,M      ;haal 3**H (**=RBUF)
051D FE07   CPI        7H      ;ctrl-G?
051F CA3F05 JZ        LODEND   ;ja, gedaan
0522 FE0A   CPI        0AH     ;linefeed?
0524 CA0A05 JZ        LOAD2    ;ja, overbodig karakter
0527 12     STAX D      ;stop char in de EDIT-buffer
0528 13     INX D      ;pointer in EDIT-buffer 1 verder
0529 2A9B02 LHLD      29BH
052C 2B     DCX H
052D CD14DE CALL      COMPAR   ;vergelijk met start van de HEAP
0530 CA3605 JZ        FULL     ;zelfde? dan vol
0533 C30A05 JMP        LOAD2    ;anders verder laden
0536 216905 FULL LXI H    FULMES
0539 CD32DB CALL      PRINT    ;print "Buffer full"
053C C37504 JMP        PORT
053F AF     LODEND XRA A
0540 12     STAX D      ;0 als laatste byte
0541 13     INX D
0542 217A05 LXI H    EBUF
0545 23     INX H
0546 22A200 SHLD     0A2H     ;Start v.d. EDIT-buffer
0549 EB     XCHG
054A 22A400 SHLD     0A4H     ;Einde v.d. EDIT-buffer
054D C37504 JMP        PORT    ;terug interactief werken
0550      ;
0550      ;*****
0550      ; DATA
0550      ;
0550 140000 BBEEP  DB      14H,0H,0H,0H,0H,14H,0H,0H,0H,0H,14H,0H,0H
055D 000F14 DB      0H,0FH,14H,0H,0H,0H,0H,14H,0H,0H,7H,0D0H
0569 0D     FULMES DB      0DH
056A C2F5E6 DB      "Buffer full"
0575 0D     DB      0DH
0576 00     WBUF  DB      0H      ;pointer naar laatst ontvangen byte
0577 00     RBUF  DB      0H      ;pointer naar laatst verwerkte byte
0578 0000   MSKSAV DW      0H
057A 00     EBUF  DB      0H
057B      END      ;begin van de EDIT-buffer
    
```

```

Start 78500=09000000      TITL      'PDP11-communicatie'
0000 @=FFF0 SERIN EQU      0FFF0H    ;serial input buffer
0000 @=0131 OUTPSW EQU     131H
0000 @=FFF8 INTMSK EQU     0FFF8H    ;interrupt mask
0000 @=D6BE GETC EQU      0D6BEH
0000 @=C818 BASIC EQU     0C818H
0000 @=00A0 BDRATE EQU     0A0H      ;4800baud
0000 @=FD06 BANK EQU      0FD06H
0000 @=0040 DUBANK EQU     40H
0000 @=E4BC SOUND EQU     0E4BCH    ;sound routine
0000 @=01BE WAIT EQU      1BEH      ;WAIT TIME timer
0000 @=DB32 PRINT EQU     0DB32H    ;print string
0000 @=DE14 COMPAR EQU     0DE14H    ;vergelijk HL en DE
0000      ORG      400H
0400      ;
0400      ;*****
0400      ; Start en initialisatie
0400      ;
0400 3E01   START MVI A      1H
0402 323101 STA      OUTPSW    ;output naar scherm alleen
0405 AF     XRA A
0406 32B902 STA      2B9H      ;alle toetsen scannen
0409 3EA0   MVI A    BDRATE
040B 32F5FF STA      0FFF5H    ;zet baudrate
040E 3EF0   MVI A    0F0H
0410 3206FD STA      BANK      ;ROM-bank 4
0413 324000 STA      DUBANK
0416 F3     DI
0417 3A5F00 LDA      5FH      ;Duplicaat van interrupt mask
041A 329205 STA      MSKSAV
041D 3ED5   MVI A    0D5H
041F 325F00 STA      5FH      ;interrupt 4 toelaten
0422 32F8FF STA      INTMSK    ;(4=serial buffer loaded)
0425 21DF04 LXI H    INTR4
0428 226A00 SHLD     6AH      ;nieuwe interrupt routine
042B FB     EI
042C 3E0C   MVI A    0CH
042E EF     RST 5      ;scherm leegmaken
042F 03     DB      3H
0430      ;
0430      ;*****
0430      ; Mainloop : ingedrukte toets verzenden
0430      ;
0430 3ABE01 KEYS LDA      WAIT    ;beeptijd voorbij ?
0433 A7     ANA A
0434 C23D04 JNZ      KEYS2
0437 CDBFD9 CALL     0D98FH    ;Disable sound interrputs
043A CDA6D8 CALL     0D8A6H    ;Stop oscillators
043D CDBED6 KEYS2 CALL     GETC
0440 DAC304 JC      BREAK    ;indien BREAK, naar BASIC
0443 A7     ANA A
0444 CA7704 JZ        PORT    ;geen toets, dan ontvangen bytes verwerken
0447 FE13   CPI      13H
0449 CA1A05 JZ        LOAD    ;cursor rechts, file laden
044C FE11   CPI      11H
044E CA7704 JZ        PORT    ;cursor down is nu ctrl
0451 FE12   CPI      12H    ;cursor links is nu escape
0453 C25804 JNZ      NESC
    
```

```

0456 3E1B      MVI A      1BH      ;escape
0458 FE08     NESC     CPI      8H      ;backspace?
045A C25F04   JNZ      NODEL    ;nee, ga verder
045D 3E7F     MVI A      7FH      ;vervang door DEL
045F 47       NODEL   MOV B,A
0460 F3       DI
0461 3E02     MVI A      2H      ;Check of control
0463 3207FF   STA      0FF07H   ;2e rij
0466 3A01FF   LDA      0FF01H
0469 FB       EI
046A 17       RAL
046B 17       RAL      ;kijk of cursor down ingedrukt is
046C 3E7F     MVI A      7FH      ;strip parity bit als geen ctrl
046E D27304   JNC      NCTRL
0471 3E1F     MVI A      1FH      ;strip hoogste 3 bits als wel ctrl
0473 A0       NCTRL   ANA B
0474 CDAD04   CALL     OUTPUT   ;serieel verzenden
0477 ;
0477 ;*****
0477 ; Mainloop : ontvangen bytes verwerken
0477 ;
0477 3A9005     PORT    LDA      WBUF      ;laatst ontvangen byte
047A 47       MOV B,A
047B 3A9105   LDA      RBUF      ;laatst verwerkte byte
047E B8       CMP B      ;zelfde?
047F CA3004   JZ       KEYS      ;ja, dan terug naar toetsen kijken
0482 3C       INR A
0483 329105   STA      RBUF      ;pointer naar laatst verwerkte byte 1 verhogen
0486 6F       MOV L,A
0487 2603     MVI H      3H
0489 7E       MOV A,M      ;haal 3**H (**=RBUF)
048A FE1F     CPI      1FH
048C DA9404   JC       CTRL     ;ctrl-chars niet op scherm zetten
048F EF       TEMP    RST 5
0490 03       DB      3H      ;andere wel op scherm zetten
0491 C37704   JMP      PORT     ;volgende byte verwerken
0494 FE0D     CTRL    CPI      0DH     ;carriage return?
0496 CABF04   JZ       TEMP     ;wel op scherm zetten
0499 FE07     CPI      7H      ;control-G?
049B CAF004   JZ       BEEP     ;dan piepen
049E FE0C     CPI      0CH     ;HOME?
04A0 CABF04   JZ       TEMP     ;wel op scherm zetten
04A3 FE7F     CPI      7FH     ;DEL?
04A5 C23004   JNZ      KEYS     ;nee, terug naar toetsen kijken
04A8 3E08     MVI A      8H      ;maak er BACKSPACE van
04AA C3BF04   JMP      TEMP     ;op scherm zetten
04AD ;
04AD ;*****
04AD ; Seriele output routine
04AD ;
04AD F5       OUTPUT  PUSH PSW   ;onthoud char
04AE 3A00FD   WAIT1   LDA      0FD00H
04B1 E608     ANI      8H      ;serial output ready?
04B3 CAAE04   JZ       WAIT1    ;nee, wacht
04B6 3AF3FF   WAIT2   LDA      0FFF3H
04B9 E610     ANI      10H     ;TICC-buffer leeg?
04BB CAB604   JZ       WAIT2    ;nee, wacht
04BE F1       POP PSW   ;haal char terug

```

```

04BF 32F6FF   STA      0FFF6H   ;laad seriele output buffer
04C2 C9       RET
04C3 ;
04C3 ;*****
04C3 ; Terug naar BASIC
04C3 ;
04C3 F3       BREAK   DI
04C4 3A9205   LDA      MSKSAV
04C7 325F00   STA      5FH      ;restore interrupt mask
04CA 32F8FF   STA      INTMSK
04CD 21C0C6   LXI H     0C6C0H
04D0 226A00   SHLD     6AH      ;restore interrupt vector
04D3 FB       EI
04D4 3E30     MVI A      30H
04D6 3206FD   STA      BANK     ;restore ROM-bank
04D9 324000   STA      DUBANK
04DC C319C8   JMP      BASIC    ;naar BASIC-monitor
04DF ;
04DF ;*****
04DF ; Interrupt routine voor intr. 4 RECEIVE BUFFER LOADED
04DF ;
04DF F5       INTR4   PUSH PSW
04E0 C5       PUSH B
04E1 3AF0FF   LDA      SERIN    ;haal ontvangen byte boven
04E4 A7       ANA A
04E5 CAF704   JZ       NONE     ;? laat dan maar
04E8 47       MOV B,A
04E9 3A9005   LDA      WBUF     ;Laatste in buffer (1 byte)
04EC 3C       INR A      ;Volgende plaats
04ED 329005   STA      WBUF
04F0 6F       MOV L,A
04F1 2603     MVI H      3H      ;Buffer (h byte) 3**H **=WBUF
04F3 78       MOV A,B
04F4 E67F     ANI      7FH      ;strip parity bit
04F6 77       MOV M,A      ;zet byte in buffer
04F7 C1       NONE     POP B
04F8 F1       POP PSW
04F9 E1       POP H
04FA FB       EI
04FB C9       RET
04FC ;
04FC ;*****
04FC ; Verwerking van control-G : BEEP
04FC ;
04FC 016A05   BEEP     LXI B      BBEEP   ;pointer naar beep commando
04FF 3E30     MVI A      30H
0501 324000   STA      DUBANK   ;BASIC ROM-bank
0504 3206FD   STA      BANK
0507 CDBCE4   CALL     SOUND    ;voer SOUND commando uit
050A 3EF0     MVI A      0F0H
050C 324000   STA      DUBANK   ;terug naar ROM-bank 4
050F 3206FD   STA      BANK
0512 3E14     MVI A      14H     ;tijdsduur van beep
0514 32BE01   STA      WAIT     ;in de timer stoppen
0517 C37704   JMP      PORT     ;verder werken
051A ;
051A ;*****
051A ; Laad een file in de EDIT-buffer

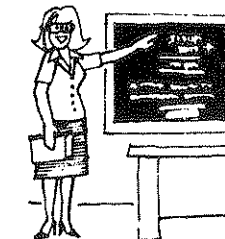
```

```

051A ;
051A 3E0D LOAD MVI A 0DH ;carriage return
051C CDAD04 CALL OUTPUT ;serieel doorsturen
051F EF RST 5
0520 03 DB 3H ;en op scherm zetten
0521 119405 LXI D EBUF ;start van de EDIT-buffer
0524 3A9005 LOAD2 LDA WBUF
0527 47 MOV B,A
0528 3A9105 LDA RBUF
052B B8 CMP B ;char ontvangen?
052C CA2405 JZ LOAD2 ;nee, wachten
052F 3C INR A ;ja
0530 329105 STA RBUF ;leespointer 1 verder
0533 6F MOV L,A
0534 2603 MVI H 3H
0536 7E MOV A,M ;haal 3**H (**=RBUF)
0537 FE7E CPI ;~?
0539 CA5905 JZ LODEND ;ja, gedaan
053C FE0A CPI 0AH ;linefeed?
053E CA2405 JZ LOAD2 ;ja, overbodig karakter
0541 12 STAX D ;stop char in de EDIT-buffer
0542 13 INX D ;pointer in EDIT-buffer 1 verder
0543 2A9B02 LHLD 29BH
0546 2B DCX H
0547 CD14DE CALL COMPAR ;vergelijk met start van de HEAP
054A CA5005 JZ FULL ;zelfde? dan vol
054D C32405 JMP LOAD2 ;anders verder laden
0550 218305 FULL LXI H FULMES
0553 CD32DB CALL PRINT ;print "Buffer full"
0556 C37704 JMP PORT
0559 AF LODEND XRA A
055A 12 STAX D ;0 als laatste byte
055B 13 INX D
055C 219405 LXI H EBUF
055F 23 INX H
0560 22A200 SHLD 0A2H ;Start v.d. EDIT-buffer
0563 EB XCHG
0564 22A400 SHLD 0A4H ;Einde v.d. EDIT-buffer
0567 C37704 JMP PORT ;terug interactief werken
056A ;
056A ;*****
056A ; DATA
056A ;
056A 140000 BBEEP DB 14H,0H,0H,0H,0H,14H,0H,0H,0H,0H,14H,0H,0H
0577 000F14 DB 0H,0FH,14H,0H,0H,0H,0H,14H,0H,0H,7H,0D0H
0583 0D FULMES DB 0DH
0584 C2F5E6 DB "Buffer full"
058F 0D DB 0DH
0590 00 WBUF DB 0H ;pointer naar laatst ontvangen byte
0591 00 RBUF DB 0H ;pointer naar laatst verwerkte byte
0592 0000 MSKSAV DW 0H
0594 00 EBUF DB 0H ;begin van de EDIT-buffer
0595 END

```

## HOOFDSTUK IV : PROGRAMMERING VAN MICROPROCESSOREN



Een programma is een geheel van instructies die door de microprocessor stuk voor stuk dienen uitgevoerd te worden om tot een bepaalde toepassing te komen.

Vermits elk van de microprocessortypes een eigen instructieset heeft, zou een uitgebreide behandeling van de verschillende insstructiesets ons veel te ver leiden. Wij beperken ons derhalve tot een gedetailleerde studie van de instructieset van de microprocessor 8080/8085 van INTEL. In annex 10 is een resumé gegeven van de 8080/8085 evenals een alfabetische lijst en een lijst in hexadecimale volgorde, en in rubrieken.

## 4.1. Samenstelling van een instructie

Een volledige instructie bestaat uit 1, 2 of 3 bytes (4 bytes bij Z80) afhankelijk van het type van instructie en van het gebruikte adresserings-type. Een programma wordt slechts door de microprocessor aanvaard onder de vorm van machinetaal, d.w.z. een opeenvolging van 2 digits lange hexadecimale codes. Vermits zulk een programma voor de gebruiker haast onleesbaar is, worden de instructies voorgesteld in afkortingen die men mnemonics noemt. Een instructie bestaat uit twee gedeelten of velden. Een eerste veld geeft de operatiecode OP CODE van de instructie aan. Het tweede veld geeft de operand(en) aan waarmee deze opcode moet werken. Als operand kunnen worden aangegeven :

- 8 bits data
- 8 bits adres van in- of uitgangskanaal
- 16 bits data
- 16 bits geheugenadres
- een register
- een registerpaar

Het is mogelijk een volledige instructie te schrijven die slechts één byte lang is, wanneer de OP CODE maar enkele bits beslaat. Dienen data of adressen aangegeven te worden in de instructie, dan wordt deze 2, respectievelijk 3 bytes lang.

Bij de bespreking van de verschillende instructies zullen we telkens, naast de mnemonics, de hexadecimale code aangeven, vermits we deze nodig hebben om een programma te schrijven in de vorm die door een microprocessor kan aanvaard en verwerkt worden.

## 4.2. Instructieset van de microprocessor 8080/8085 van INTEL

De instructies voor de 8080/8085 kunnen worden onderverdeeld in verschillende groepen. Voor we de instructies van elke groep in detail behandelen, geven we een algemeen overzicht.

We onderscheiden :

- Transfertinstructies : zij maken het mogelijk :
  - a) gegevens, adressen of constanten te verplaatsen, hetzij :
    - . tussen registers
    - . van geheugen of ingangspoort naar een register
    - . van register naar een geheugenplaats of naar een uitgangspoort
  - b) een constante te plaatsen in een register of een geheugenplaats
- Rekenkundige instructies
- Logische instructies



- Instructies betreffende de behandeling van onderprogramma's (subroutines)
- Instructies betreffende programma onderbreking
- Besturingsopdrachten voor het assembler vertaalprogramma (pseudo-instructies).

Bij de definitie van de instructies gaan we gebruik maken van bepaalde afkortingen en symbolische voorstellingen die we eerst even toelichten.

| Verwijzing naar :                       | gebeurt met : |
|-----------------------------------------|---------------|
| register                                | r             |
| registerpaar                            | rp            |
| adres                                   | adr           |
| 8 bits data of constante                | data 8 (d8)   |
| 16 bits data                            | data 16 (d16) |
| geheugencel                             | M             |
| het nummer van een in- of uitgangspoort | nr (n°)       |

Indien we niet verwijzen naar een register, registerplaats of geheugencel maar wel naar de respectievelijke inhoud, dan wordt dit aangegeven door het tussen haakjes plaatsen van het betreffend element.

- (r) geeft de inhoud van een register
- (rp) geeft de inhoud van een registerpaar
- (adr) geeft de inhoud aan, van wat zich op een adres bevindt

Onderstaande tabel geeft de namen van de verschillende registers of registerparen die voor programmatie toegankelijk zijn. Tenslotte een zo volledig mogelijke informatie te kunnen geven hebben we de binaire naam (in machinetaal) van de verschillende registers of registerparen toegevoegd in een derde kolom van deze tabel.

| Naam                         | Symbool | Binaire equivalent |
|------------------------------|---------|--------------------|
| accumulator                  | A       | 111                |
| register B                   | B       | 000                |
| register C                   | C       | 001                |
| register D                   | D       | 010                |
| register E                   | E       | 011                |
| register H                   | H       | 100                |
| register L                   | L       | 101                |
| geheugencel M                | M       | 110                |
| registerpaar B,C             | B       | 00                 |
| registerpaar D,E             | D       | 01                 |
| registerpaar H,L             | H       | 10                 |
| stack pointer (stapelwijzer) | SP      | 11                 |
| program status word          | PSW     |                    |
| programmateller              | PC      |                    |
| stack (stapelgeheugen)       | STCK    |                    |

Om de verklaring van de instructies zeer bondig te kunnen beschrijven, maken we gebruik van de symbolen:

→ betekent : gaat naar

Voorbeeld :

(B) → C betekent : de inhoud van register B gaat naar register C

↔ betekent : wordt omgewisseld

Voorbeeld :

(HL) ↔ (SP) betekent : de inhoud van het registerpaar wordt omgewisseld met de inhoud van de stack pointer.

De opcode van de instructies wordt zowel in mnemonic als in machinetaal (binair) gegeven. In machinetaal worden de volgende symbolen gebruikt :

- ddd staat voor het register van bestemming (*destination*) zoals bij een transfert opdracht bijvoorbeeld.
- sss staat voor het bronregister (*source*) vanwaar de informatie vertrekt
- rp staat voor een registerpaar waarmee de instructie werkt

Het binair equivalent van de respectievelijke registers wordt in bovenstaande tabel teruggevonden.

De door de instructie beïnvloede toestandenbits worden afgekort door :

- C *carry* overdrachtbit
- Z *zero* nulbit
- S *sign* tekenbit
- P *parity* pariteitsbit
- AC *auxiliary carry* hulpoverdracht

In de instructieset worden vaste symbolen gebruikt :

- I voor het onmiddellijk (*immediate*) manipuleren van constanten
- X voor het aanduiden van instructies met betrekking tot registerparen. Deze vaste symbolen zijn slechts geldig als I of X niet de eerste plaats van de mnemonic innemen.

Rekening houdend met al deze gemaakte afspraken zullen we alle instructies bespreken onder dezelfde vorm.

De eerste lijn van de bespreking geeft in volgorde : de mnemonic, de machinetaal, de toestandenbits die door de instructie beïnvloed worden, het aantal bytes en tenslotte de instructieduur gegeven door het aantal periodes van de besturingsklok.

De aanduiding tussen haakjes duidt op een aantal periodes bij de 8085 indien deze niet dezelfde is dan bij de 8080.

Een tweede lijn geeft de betekenis van de instructie in de engelse taal. Een verklaring van wat de instructie doet en een grafische voorstelling ronden het geheel af.

#### 4.2.1. Transfert instructies

##### a) Transfert tussen registers

MOV r<sub>1</sub>,r<sub>2</sub>    01dddsss    - - - - -    1    5(4)

*Move register to register*

De registerinhoud van r<sub>2</sub> wordt naar register r<sub>1</sub> gebracht (r<sub>2</sub>) → r<sub>1</sub>

Overeenkomstig de afspraken is dit een 1 byte instructie die 5 klokperiodes duurt en de toestandenbits niet beïnvloedt.

Let wel, r<sub>1</sub> en r<sub>2</sub> kunnen ook hetzelfde register zijn

XCHG    11101011    - - - - -    1    4

*Exchange DE with HL*

De inhoud van de registerparen DE en HL wordt omgewisseld (HL) ↔ (DE)

XTHL      11100011    - - - - -      1            18(16)  
*Exchange HL to top of stack*  
 De inhoud van het registerpaar HL wordt omgewisseld met de inhoud van het stapelregister geadresseerd door SP en SP+1. De inhoud aangeduid door SP wordt omgewisseld met deze van L en de inhoud aangeduid door SP+1 wordt omgewisseld met deze van H.  
 (HL) ↔ (TOP OF STACK)

SPHL      11111001    - - - - -      1            5(6)  
*HL to stack pointer*  
 Het stapelregister wordt geladen met de inhoud van het HL register  
 (HL) → SP

b) Transfert van een geheugencel of een ingangspoort naar een register

MOV r,M    01ddd110    - - - - -      1            7  
*Move memory to register*  
 De inhoud van de geheugencel geadresseerd door HL wordt overgebracht naar een register  
 (M) → r met M = (HL)  
 mov M,M is niet toegelaten

LDA adr    00111010    - - - - -      3            13  
*Load accumulator direct*  
 De accumulator wordt geladen met de inhoud van het adres adr  
 (adr) → A  
 Let wel dat in binaire en ook in hexadecimale notatie byte 2 de laagste adresbits en byte 3 de hoogste adresbits aanduiden, dit in tegenstelling tot de assemblertaal.

LDAX rp    00rp1010    - - - - -      1            7  
*Load accumulator indirect*  
 De accumulator wordt geladen met de inhoud van de geheugencel waarvan het adres in het registerpaar rp staat.  
 (adr) → A

LHLD adr    00101010    - - - - -      3            16  
*Load HL direct*  
 Het HL registerpaar wordt direct geladen met de inhoud van adr en adr+1  
 (adr) → (adr + 1) → H

POP rp     11rr0001    - - - - -      1            10  
*Pop register pair off stack*  
 Een registerpaar of het paar accumulator + toestandenbits wordt geladen met de inhoud van de cellen aangegeven door SP en SP + 1.  
 Als het PSW als registerpaar gebruikt wordt, worden alle toestandenbits gewijzigd.  
 De inhoud aangeduid door SP gaat naar Low of naar de Flags deze aangeduid door SP + 1 naar High of de Accumulator.

IN n°      11011011    - - - - -      2            10  
*Input*  
 De accumulator wordt geladen met het woord aangeboden op de poort met het nummer n°. Het nummer bestaat uit 8 bits zodat 256 combinaties mogelijk zijn.  
 (Kanaal) → A

c) Transfert van een register naar een geheugenplaats of naar een uitgangspoort

MOV M,r    01110sss    - - - - -      1            7  
*Move register to memory*  
 De inhoud van een register wordt overgebracht naar een geheugenplaats M geadresseerd door het HL registerpaar.  
 (r) → M met M = (HL)

STA adr    00110010    - - - - -      3            13  
*Store accumulator direct*  
 De inhoud van de accumulator wordt overgebracht naar het adres adr.  
 (A) → adr

STAX rp    00rp0010    - - - - -      1            7  
*Store accumulator indirect*  
 De inhoud van de accumulator wordt overgebracht naar de geheugenplaats waarvan het adres staat in rp.  
 (A) → adr met adr = (rp)

SHLD adr    00100010    - - - - -      3            16  
*Store HL direct*  
 De inhoud van het registerpaar HL wordt overgebracht naar de adressen adr en adr + 1  
 (HL) → adr    (L) → adr  
                   (H) → adr + 1

PUSH rp    11rp0101    - - - - -      1            11  
*Push registerpair into stack*  
 De inhoud van het registerpaar rp of PSW wordt overgebracht in het stapelgeheugen op de adressen aangewezen door SP - 1 en SP - 2.  
 (rp) → STCK  
 (PSW) → STCK  
 HIGH of Accumulator → SP - 1  
 LOW of Flags → SP - 2

OUT nr     11010011    - - - - -      2            10  
 De inhoud van de accumulator wordt naar de uitgangspoort met nummer nr gebracht  
 (A) → Kanaal nr

d) Een constante naar een register of een geheugenplaats brengen

Chers membres,

Vous qui êtes Francophone, vous avez pu remarquer la présence du DAI namic INFO dans quelques unes des revues précédentes. Le DAI namic INFO est une création du club DAI namic France, qui contient des traductions d'articles publiés précédemment en langue Anglaise ou Flammande, mais aussi des articles originaux de nos membres en France.

Tout d'abord il a pour but d'élargir le cercle des utilisateurs du DAI afin d'avoir encore plus de nouveautés et de logiciels à vous présenter. Ainsi dans ce numéro vous est présenté un premier programme (La forteresse de ZLARG) qui ne contient que des textes en Français.

Ensuite les programmes courants du club, en accord avec la Belgique, seront désormais disponibles en France à un prix sensiblement identique. Ces programmes seront accompagnés (Quand nécessaire) d'une notice en français. Par exemple : la notice de FWP sera disponible à partir du 1er Juillet.

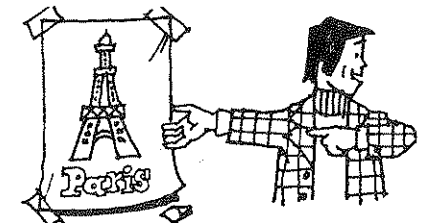
Enfin, nous pensons, et si le nombre de demande est assez important, regrouper les achats de cassettes DCR. En effet vous avez pu noter qu'il est très difficile de se procurer ces cassettes, et que quand on y arrive elles sont vendus à des prix hautement prohibitifs.

Alors, et si vous avez des questions à nous poser, des articles divers concernant le DAI ou des échanges de logiciels qui vous intéressent, n'hésitez plus ! Ecrivez nous à :

DAInamic  
9 Rue LAvoisier  
59140 Dunkerque

Au plaisir de vous lire,

C. DUFOUR



|              |                                                                                                                                    |           |   |    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------|-----------|---|----|
| MVI r,const  | 00ddd110                                                                                                                           | - - - - - | 2 | 7  |
|              | <i>Move immediate to register</i>                                                                                                  |           |   |    |
|              | Een constante (dg) wordt in een register geschreven<br>const → r                                                                   |           |   |    |
| MVIM M,const | 00110110                                                                                                                           | - - - - - | 2 | 10 |
|              | <i>Move immediate to memory</i>                                                                                                    |           |   |    |
|              | Een constante wordt in een geheugenplaats geschreven die<br>door het HL registerpaar wordt geadresseerd.<br>const → M met M = (HL) |           |   |    |
| LXI rp,adr   | 00rp0001                                                                                                                           | - - - - - | 3 | 10 |
|              | <i>Load registerpair immediate</i>                                                                                                 |           |   |    |
|              | Een registerpaar rp wordt geladen met een 16 bits constante<br>die meestal een adres zal zijn.<br>constante (d16) → rp             |           |   |    |

#### 4.2.2. Rekenkundige instructies

|        |                                                                                                                              |            |   |      |
|--------|------------------------------------------------------------------------------------------------------------------------------|------------|---|------|
| INR r  | 00ddd100                                                                                                                     | Z,S,P,-,AC | 1 | 5(4) |
|        | <i>Increment register</i>                                                                                                    |            |   |      |
|        | De inhoud van een register wordt met 1 verhoogd<br>(r) + 1 → r                                                               |            |   |      |
| INRM   | 00110100                                                                                                                     | Z,S,P,-,AC | 1 | 10   |
|        | <i>Increment memory</i>                                                                                                      |            |   |      |
|        | De inhoud van een geheugencel, door het HL register gead-<br>resseerd, wordt met 1 verhoogd.<br>(M) + 1 → M met M = (HL)     |            |   |      |
| DCR r1 | 00ddd101                                                                                                                     | Z,S,P,-,AC | 1 | 5    |
|        | <i>Decrement register</i>                                                                                                    |            |   |      |
|        | De inhoud van een register wordt met 1 verlaagd.<br>(r) - 1 → r                                                              |            |   |      |
| DCR M  | 00110101                                                                                                                     | Z,S,P,-,AC | 1 | 10   |
|        | <i>Decrement memory</i>                                                                                                      |            |   |      |
|        | De inhoud van een geheugencel geadresseerd door het HL<br>registerpaar wordt met één verminderd.<br>(M) - 1 → M met M = (HL) |            |   |      |
| INX rp | 00rp0011                                                                                                                     | - - - - -  | 1 | 5    |
|        | <i>Increment register pair</i>                                                                                               |            |   |      |
|        | De inhoud van een registerpaar (B, D, H of SP) wordt met<br>1 verhoogd.<br>(rp) + 1 → rp                                     |            |   |      |
| DCX rp | 00rp1011                                                                                                                     | - - - - -  | 1 | 5    |
|        | <i>Decrement register pair</i>                                                                                               |            |   |      |
|        | De inhoud van een registerpaar rp wordt met 1 verminderd<br>(rp) - 1 → rp                                                    |            |   |      |

TRADUCTIONS

TRADUCTION DE L'ARTICLE : PROGRAMMEERTECHNIKEN  
DE F.H.DRUIJFF (Nº 16 P154)

Examinons aujourd'hui, le problème de la rapidité d'un programme. Le premier problème est bien sur: comment tester la rapidité d'une instruction ou d'une partie de programme donnée. Le DAI peut nous être d'un grand secours. Le mieux est de faire un programme-timer séparé qui, si souhaité, peut être une subdivision du programme principal aussi longtemps que celui-ci est encore au stade de projet. Les lignes 5 et 95 présentent l'ossature de ce programme :



```

5   WAIT TIME 2:POKE #1BF,#FF:POKE #1BE,#FF
95  T1BE=PEEK(#1BE):T1BF=PEEK(1BF):
    PRINT(#FFFF-T1BE-T1BF*256)/50.0

```

Les variables T1BE et T1BF sont des entiers naturels. Nous utilisons l'horloge de 20 msec. pour diminuer la valeur de 1BE et 1BF jusqu'à ce qu'elle soit nul. Les pokes et le WAIT-TIME de la ligne 5 servent à avoir un départ correct. Puisque l'instruction WAITTIME utilise ces adresses, elle ne peut figurer dans le programme à tester. L'ordre de lecture dans la ligne 95 a son importance: tel quel, il se pourrait qu'une erreur de  $256 \times 20$  msec. = 5 sec. se produise, mais cela se verrait immédiatement, alors que si les deux PEEK'S avaient été lu en sens inverse, une erreur de 20 msec. aurait pu venir fausser le résultat. Les deux exemples de la page 154 vous donnerons une idée de la variation de vitesse de programmes semblant pourtant être identiques. Si nous ajoutons A=3 ou A=B dans la boucle, il faut limiter la boucle à 1000 iterations car le temps augmente trop fort. Si nous ajoutons des fonctions, ce temps va encore augmenter plus fortement. La période chronométrée peut donc être de  $\#FFFF \times 20$  msec. = plus de 20 min. Attention les calculs avec les entiers ne sont pas toujours les plus rapides: voir ainsi  $A\% = \text{SQR}(P\%)$  et  $A! = \text{SQR}(P!)$  !  $A = 2 \times B + 4$  et les lignes suivantes sont assez édifiantes. Que la boucle se trouve sur une seule ligne ou sur deux lignes peut avoir aussi une influence. En principe essayez toujours d'avoir tout sur une seule ligne. Examinons maintenant les sauts conditionnels: -Ces instructions si importantes doivent se voir clairement dans un programme, par exemple: le IF \*\*\* GOTO \*\*\* semble plus clair que le IF \*\*\* THEN \*\*\*. Ci-après un exemple de programme déjà reçu:

```

120 IF A=3 GOTO 140
130 GOTO 160
140 P=2
150 GOTO 160
160 END:REM Un programme desolant !

```

Naturellement cela fonctionne, mais il peut être nettement simplifié

```

120 IF A<>3 GOTO 160
140 P=2
160 END:REM Un programme meilleur !

```

Une plus courte version donnerait :

```

120 IF A<>3 GOTO 160:P=2
160 END

```

Une version sans l'instruction GOTO

```

120 IF A=3 THEN P=2

```

Enfin sans utiliser le IF :

```

120 P=P+P*(SGN(ABS(A-3))-1)-2*(SGN(ABS(A-3))-1)

```

Savez-vous que le ON W GOTO \*\* travaille plus rapidement que le IF \*\* GOTO \*\*. Mais attention car un calcul supplémentaire pourrait annuler ce gain de temps. Faites aussi attention aux programmes où l'on retrouve en 50 :GOTO 170 en 170 GOTO 220 et en 220 GOTO 30 !. Bien souvent ces erreurs proviennent de programmes qui ont été corrigés durant leur élaboration

Les GOSUB et RETURN sont en général très clairs dans un programme, dans la mesure du possible n'utilisez qu'une seule entrée et une seule sortie, à moins que vous ne soyez très sûr de vous et que la clarté n'y perde pas de trop. Essayez aussi le programme qui suit le texte et voyez combien il est rapide : (Problem Program)

TRADUCTION DE L'ARTICLE : PROGRAMMERTECHNIKEN  
DE F.H.DRUIJFF (N° 18 P285)

Voici un petit programme qui m'a été envoyé par Koert Van Espen. Ce programme appelé "STAR - BUILDER" est relativement court et simple, mais mis à part quelques petites fautes, les différents problèmes ont bien été analysés et des solutions leurs ont été trouvées. D'un autre côté, les fautes et la façon d'aborder les problèmes sont illustratifs du raisonnement de beaucoup de programmeurs et pas nécessairement des débutants.

C'est pourquoi, je vais utiliser ce programme comme base de cet article avec bien sûr l'accord de Koert qui savait pourtant que tout ce qu'il avait fait dans ce programme allait être "critiqué".

J'espère que vous prendrez la peine de lire le programme de Koert, de noter toutes les fautes que vous croyez rencontrer et de lire l'article ensuite. Cependant, il y a bien sûr des points qui sont une affaire de goût et il est même possible qu'il y en ait que j'ai négligé. Si c'est le cas, faites-le moi savoir je pourrai en tenir compte dans le futur.

Vous trouverez le programme initial de Koert aux pages 285 et 286 du n.18

Comme vous pouvez le voir c'est un programme assez court, mais qui fonctionne relativement bien. Lorsque l'on étudie le listing, on remarque directement que les numéros de ligne sont seulement des multiples de 10, cela me dit que le programme a été pensé mais encore revu avant d'être envoyé. Ceci manque souvent dans les envois car beaucoup sont contents de voir le programme fonctionner. Le programmeur de métier devra encore faire un manuel d'utilisation après la dernière ligne de programme si celui-ci indique automatiquement les explications. Ainsi par exemple lorsque vous utilisez le DAINATEXT, il est indiqué sur la première page comment vous pouvez retourner au menu avec un SHIFT- RETURN, mais cela n'est plus indiqué lorsque vous entrez du texte!, de plus il n'est pas indiqué clairement la façon ou l'ordre dans lequel vous devez appuyer sur ces touches! Mais attaquons maintenant la critique.

Comme je l'ai déjà mentionné auparavant, je suis un adversaire de la question qui est faite à l'utilisateur pour savoir s'il veut connaître les instructions quand celles-ci sont aussi courtes. Cette routine se trouve bien en fin de programme, avec des numéros de ligne logiques, mais elle contient deux RETURN ce qui est déjà moins bien réussi. L'ensemble aurait pu être repris avec des GOTO, nous allons seulement de la ligne 20 en 10000 et ensuite retour en ligne 100. Si nous voulions plus tard incorporer un

# DAInamic INFO

```
HULP=SCRN(XX,YY):IF HULP =15 OR HULP =3 THEN ...
```

Encore meilleur :

```
IF SCR(XX,YY)<=0 THEN ...
```

mais cela ne change pas la faute, comment la résoudre? Il y a plusieurs méthodes disponibles en voici deux :

1) Nous calculons le nombre d'étoiles que nous plaçons, s'il y en a par exemple 28, alors nous devons attraper 28 étoiles pour obtenir un nouveau terrain. Au lieu de tester littéralement si SCR(XX,YY)=0, nous pouvons mettre le compteur à jour de la manière suivante :

```
TELLER=TELLER+1-SGN(SCR(XX,YY))
```

2) Nous choisissons un autre point et pour gagner du temps nous changeons seulement la valeur YY.

```
200 FOR O=1 TO 30:XX=RND(X)
210 YY =RND(Y):IF SCR(XX,YY)<>0 GOTO 210
220 NEXT
```

Le double emploi de la boucle pour placer les points blancs et rouges me conviendrait des capacités de programmeur de Koert. Sans être plus original, mais suivant mon goût pour la lisibilité je placerais volontiers deux boucles FOR-NEXT séparées. C'est à peine plus de travail et bien plus clair pour les autres. Mais sachez que la solution de Koert est exacte. Un autre argument contre la solution de Koert est qu'après la première boucle, on teste inutilement si la couleur 3 est encore présente. En ligne 800 G, GG et T sont remis à zéro avant le retour au début du programme. Il aurait été préférable que cette initialisation se fasse au début. La mise à zéro des lignes 510 et 900 peut alors ne pas avoir lieu. Après ces changements, la ligne 900 ne contient plus que l'augmentation de 5000. En fait nous pouvons seulement arriver dans la ligne 900 qu'à partir de la ligne 320, aussi on pourrait le faire à cette ligne et supprimer la ligne 900. La ligne 800 peut aussi être supprimée après un petit changement. L'ENVELOPE ne change jamais et peut donc être mise dans l'initialisation. La construction du terrain peut se trouver en fin de programme on peut faire commencer ce bloc par quelques GETC pour contrer le rebond des touches, cela éviterait de commencer avec la dernière commande du jeu précédent. Le GOTO en ligne 200 ne me plaît pas non plus, La détermination de la direction est essentielle et donc je la placerais en début de programme. Elle doit en fait être la première excepté l'initialisation.

La méthode utilisée par Koert pour indiquer combien de tours il existent encore est bonne et originale (voir lignes 120 et 130) bien qu'elle soit étrange. Il aurait été plus clair de l'écrire comme ceci :

CLEAR cela se ferait alors sans problème. Avec un CLEAR le programme est d'ailleurs meilleur, essayez avec un CLEAR 256 par exemple. Les lignes 30,40 et 50 pourraient se trouver avec les explications, de façon à avoir un seul choix dans le programme. Au sujet de la ligne 50 :

```
50 G=GETC : IF G=49 OR G=50 GOTO 100:GOTO 50
```

Cette ligne réagit de la même manière que celle du programme, mais elle n'offre pas la possibilité d'insérer une ligne entre 50 et 100. Les deux méthodes utilisent AND et OR, instructions qui demandent un temps supplémentaire, voici une autre solution :

```
50 G=GETC: IF G<49 GOTO 50:IF G>50 GOTO 50
```

La même remarque peut être faite pour la ligne 530. Koert ne savait apparemment pas, et en cela il n'est certainement pas le seul, que lors d'une combinaison de IF AND ou de IF OR, le DAI examine toujours les deux possibilités et les combine avant de prendre action. Il est préférable de stopper dès qu'elle est fautive. Le AND et le OR sont plus faciles à comprendre comme structure, mais ne sont pas les plus rapides sur le DAI. Si la vitesse est importante, on peut utiliser plus de IF's, et arranger l'ordre des tests pour en avoir le moins possible. Le score est tenu à jour à la ligne 320; après 5 deux espaces sont imprimés afin que les derniers chiffres de la fois précédentes soient effacés. Considérez pourquoi deux espaces sont nécessaires et dans quel cas un seul n'est pas suffisant. Après ce " ", je verrais volontiers un ';' pour éliminer le comportement énervant du curseur. On pourrait aussi rendre celui-ci invisible par un POKE #75,32. En ce qui concerne la ligne 180, X+X est plus rapide que 2.0\*X. L'utilisation des nombres 1.0 et 2.0 est correcte car le RND fonctionne plus vite qu'avec des entiers. Le INT est complètement inutile parce qu'il y a quand même une adjudication à une variable entière.

Et maintenant la faute : Koert prévoit qu'il est possible qu'une 'étoile' vienne se placer à un endroit où il y en a déjà une, il n'y en aurait donc que 29 (ou moins) et il ne serait jamais possible d'avaler 30 étoiles pour obtenir un nouveau champ. Koert résout cela apparemment bien en testant si la place est vide, si ce n'est pas le cas, il diminue le compteur de boucle de 1 unité. Ce serait très bon si le DAI permettait ce genre d'opération, car au début de chaque boucle FOR-NEXT le nombre d'itérations nécessaire est calculé et il n'est plus possible de le changer. Les modifications sur la variable O n'ont donc aucun effet. De plus j'ai des objections contre un FOR avec deux NEXT's. Le test même pourrait être meilleur, non seulement le OR dont nous avons déjà parlé mais aussi la double comparaison de SCR(XX,YY) prend du temps. Une meilleure solution serait :

```

40 IF B=1 THEN PRINT CHR$(255)+" "+CHR$(255)
50 IF B=2 THEN PRINT CHR$(255)+" "
60 IF B=3 THEN PRINT " "

```

La ligne 60 peut même être supprimée par un CHR\$(12) J'ai cherché une autre solution :

```

40 PRINT MID$(CHR$(255)+" "+CHR$(255)+" ",B+B,3)

```

avec un B qui varie de 0 a 2. Je préférerais en fait éviter le CHR\$(255). Je mets alors la ligne concernée en ligne 1, et remplace le string qu'utilise MID\$ par "AAAAAAA". Ensuite je passe en Utility pour tester ou commence le basic (normalement) #3EC s'il n'y a pas de CLEAR. On remplace alors les 41's qui se trouve juste apres #3EC par FF 20 FF 20 20 20. Pour ceux qui ne sont pas familiarisés par ces codes FF est un bloc noir et 20 un espace. Retour au basic et par l'intermediaire de EDIT modifier le numero de ligne. La ligne 330 ne me plait pas non plus. Il est logique de mettre GETC dans une autre variable pour conserver la direction originale, mais pourquoi n'y a-t-il pas de 'point de punition' lorsqu'un changement de direction est donné ? Donc remplacer le DN GG-15 GOTO ... par un GOTO 1100. Bien que le jeu soit rapide, examinons la logique suivie; Dans la ligne 320 se trouve un test insensé à moins que le dernier point ne soit un blanc (!). On peut aussi changer la ligne 330 le IF GG=0 devient alors IF GG<16, de facon a rendre le TAB et RETURN sans influence. J'ai aussi amélioré la beauté du programme en utilisant des entiers aux lignes 50 et 10050. Le resultat final paraîtra dans Games Collection 12 sous le titre de STAR-HUNTER.

(Trad. J.p. Mallien)

**SIMULATION DE GOTO X / RUN X EN SEULEMENT 7 OCTETS  
(N0 16 P208)**

Avec le programme suivant vous pouvez simuler GOTO X et il est également possible de faire RUN suivi d'un numero de ligne en BASIC V1.0 sans vider la table des symboles.

Pour inclure le programme dans l'ordinateur, tapez la ligne suivante en basic:

```

10 DIM JMP(1):JMP(0)=#232344:JMP(1)=#4DC363DF:JUMP=VARPTR(JMP(0))+1

```

Maintenant JUMP contient l'adresse et JMP(0), JMP(1) le programme. Faites attention, si vous avez un CLEAR avant l'execution de la ligne en basic, JMP(0), JMP(1) et JUMP prennent la valeur 0 !!!

Vous pouvez utiliser mon programme selon les modes ci-dessous :

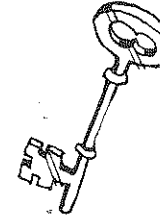
- 1) Pour RUN X tapez par exemple \*X=100:CALLM JUMP,X
- 2) Pour GOTO X tapez par exemple \*10 X=100  
\*20 CALLM JUMP,X:REM GOTO X

NOTE : X, JUMP, JMP(0) et JMP(1) doivent être entiers.

(Voir programme et exemple d'utilisation en basic p208)

# DAInamic INFO

LOGICIELS



La Forteresse de ZLARG (P. Pedelaborde)



Un super programme d'aventure, tout en français, dans l'idée des 'Donjons et dragons'.

Arriverez-vous à détruire la forteresse de ZLARG tel est le défi que vous lance le maître des lieux ! Mais attention, vous et vos hommes aurez à subir, avant de pouvoir arriver à la forteresse, les assauts des monstres les plus fantastiques. (N'oubliez pas aussi de nourrir vos hommes, car un estomac vide mène à tout !!!). Vous pourrez accumuler les trésors, visiter des villages, y acheter tout le nécessaire...

Extrait de la liste des logiciels

|                           | Audio   | DCR     |
|---------------------------|---------|---------|
| Acrobates                 | 90 Frs  | 115 Frs |
| Centipède                 | 90 Frs  | 115 Frs |
| Driver                    | 90 Frs  | 115 Frs |
| Super-Invaders            | 90 Frs  | 115 Frs |
| DAI-Panic                 | 120 Frs | 145 Frs |
| Pac-man                   | 165 Frs | 190 Frs |
| Math-fun                  | 150 Frs | 175 Frs |
| SPL (Macro-assembleur)    | 165 Frs | 190 Frs |
| FWP (Traitement de texte) | 300 Frs | 325 Frs |
| Toolkit 3                 | 150 Frs | 175 Frs |
| Toolkit 4                 | 150 Frs | 175 Frs |
| Toolkit 5                 | 150 Frs | 175 Frs |
| Games collection 6        | 115 Frs | 140 Frs |
| Games collection 7        | 115 Frs | 140 Frs |
| Games collection 8        | 115 Frs | 140 Frs |
| Games collection 9        | 115 Frs | 140 Frs |
| Games collection 10       | 115 Frs | 140 Frs |
| Games collection 11       | 115 Frs | 140 Frs |
| Games collection 12       | 115 Frs | 140 Frs |
| La forteresse de ZLARG    | 145 Frs | 170 Frs |



**6<sup>e</sup> Championnat International de  
programmes d'Othello-Reversi**  
**6th Othello-Reversi Programs World Championship**

organisé par  
**L'ORDINATEUR  
INDIVIDUEL**

Cher Monsieur, Chère Madame,

Vous êtes intéressé(e) par le prochain Championnat  
d'OTHELLO-REVERSI. Je vous rappelle qu'il se déroulera  
en une seule journée, le 22 SEPTEMBRE 1984, en catégorie  
ordinateurs de table, et sur deux jours, les 22 et 23  
SEPTEMBRE 1984, en catégorie ordinateurs de poche.

Ce 6ème Championnat se tiendra au SICOB, au CNIT-LA DEFENSE.  
Le lieu vous sera précisé ultérieurement.

Vous trouverez ci-joint :

- le texte du règlement pour les catégories
  - . ordinateurs de table
  - et
  - . ordinateurs de poche
- un bulletin d'inscription à compléter et à nous retourner  
TRES VITE !

Bien cordialement,

*Brigitte Millé*  
Brigitte MILLÉ  
Promotion

L'ORDINATEUR INDIVIDUEL (Othello) 5 place du Colonel-Fabien 75491 Paris Cedex 10 ☎ (1) 240 22 01

# TIPS

\*\*\*\*\*

several questions with respect to one of the buffers. For small programs the article of Frank Druijff in

first. If already other programs that the heap-pointer 29B-29C can be accomplished by simply pushing

carried out:

size e.g 20000 ).

A4/A5 the end of

n buffer 1 of FWP.  
ed as 3320 (A2/A3)

start buffer 1).  
!!!).

this is done to by-  
duction in FWP.

and will find the

and's like TRIM, MOVE etc. be given  
answer is YES, it can as long as the  
(a parallel printer routine) are not  
se addresses are free.

ses in FWP V1.4 as listed below, by

paragraph 4.0 of the FWP manual.

ote: All addresses as listed in the FWP manual remain the same.

fter having done this, it is possible to execute the main-menu commands  
M", "K", "T", "E" and "W", while working in the buffer. Type SHIFT/RETURN  
ollowed by the relevant command key e.g "T". To improve the speed, the  
sual query message's have been omitted, so be warned !

WARNING !!

\*\*\*\*\*  
Do never use the "T" command if you are modify-ing a BASIC program. \*  
In most cases you will get a garbage. \*  
\*\*\*\*\*

The transferred BASIC program has always 6 spaces between the line-number and start of the instruction. If you don't like this, use the "R" command to replace the 6 spaces by for instance 3. Be careful if there are PRINT statements containing 6 or more spaces. This could be overcome if the line-numbers end with 0. Just replace 0, followed by 6 spaces.

The patches below incorporate also a routine which disables the RS232 (printer) output if you are in BASIC. In the standard version of FWP this worked until the EDIT command was given.

To enable the RS232 output you have to change the value of V5 (vector 5) in C6FD.

The ERROR message "MARKERS NOT CORRECT" can be displayed a much shorter time by changing the content of addresses 1A5E-1A63 in 00.

Finally I would recommend to test your new version first, to check if all the patches have been typed-in correctly.

If you have found an interesting application, or have some specific questions pls. do not hesitate to contact me.

Ger Gruiters                      Laurahof 12  
                                                      5801 JE Venray  
                                                      Tel. 04780-84180

FWP V1.4 PATCHES object code  
 \*\*\*\*\*

```

3BA E5 21 00 00 22 3F
3C0 03 E1 CA D5 OF FE 45 CA 98 18 FE 4B CA D1 2A FE
3D0 4D CA DF 2A FE 54 CA 6D 19 FE 57 CA 80 1A C3 91
3E0 OF

5DA 21 E6 2A

FBE C3 BA 03

2AD1 3E 01 32 1B 03 CD 32 09 DA 6B 09 C3 A6 09 AF 32
2AE1 1B 03 C3 9F 09 F5 21 31 01 7E FE 02 F2 F2 2A 36
2AF1 01 F1 C3 FD C6

```

```

*****
*
* FWP V1.4 PATCHES 24-5-1984 *
*
*****

UNL ; PART OF LISTING
PUSH H ; SAVE CURSOR POS.
LXI H OH ; CLEAR COUNTER USED IN
SHLD STORE3 ; "E" AND "W" ROUTINES
POP H ; RESTORE CURSOR POS
JZ NWL ; MOVED INSTR. TO ALLOW PATCH
CPI 'E' ; ELIMINATE SPACES
JZ REMSP1
CPI 'K' ; KILL TEXT BETWEEN 06-07
JZ KSHRT ; MOVE TEXT BETWEEN 06-07
CPI 'M'
JZ MSHRT ; TRIM LINES
CPI 'T' ; WIDTH EQUAL (RIGHT MARGIN)
JZ TSHRT ; END PATCH, RETURN TO ROUTINE
CPI 'W' ; PART OF LISTING
JMP FILLSP
JMP CURO ; LOAD VECTOR 5
UNL ; PART OF LISTING

LXI H PRCTRL ; EXECUTE PATCHED INSTRUCTIONS
UNL ; PART OF LISTING

JMP PATCH1 ; EXECUTE PATCHED INSTRUCTIONS
UNL ; PART OF LISTING

KSHRT MVI A 1H ; SET KILL FLAG "ON"
STA FLAG2 ; CHECK IF MARKERS ARE OK
CALL CHK167 ; IF NOT, DISPLAY ERROR MSG.
JC FMARK ; EXECUTE "K" (KILL) COMMAND
JMP MOVT1

MSHRT XRA A ; KILL FLAG "OFF"
STA FLAG2 ; EXECUTE "M" (MOVE) COMMAND
JMP MSHRT1 ; SAVE CHARACTER
PRCTRL PUSH PSW ; GET STATUS OUTPUT SWITCH
LXI H 131H ; IN ACCU
MOV A,M ; NO CHANGE IF OUTPUT TO EDIT
CPI 2H ; OR VIA DOUTC
JP NOCHNG ; SET OUTPUT TO SCREEN ONLY
JP NOCHNG ; RESTORE CHARACTER
MVI M 1H ; TO SCREEN RESTART (RST 5)
POP PSW ; PART OF LISTING
JMP OC6FDH
UNL

```



# DBASIC/2

DBASIC V2.1 page 6

This article is a continuation of the article on DBASIC in newsletter 22 page 141.

## 3. DESCRIPTION OF DBASIC 'INTRINSIC' FUNCTIONS

NOTE : -Only functions added by DBASIC are described here.  
-For every function a very small description and eventually an example are given.

+ DEEK(<integer expression>)  
-Returns the two-byte value at address location <integer expression>.

```
ex. 10 HEAP=#29B
    20 PRINT "THE HEAP STARTS AT ADDRESS #";
    30 PRINT HEX$(DEEK(HEAP))
    ...
```

+ DIM(<array name>,<integer expression>)  
-When the value of <integer expression> equals n, then DIM will return the value of the n-th dimension of <array name>.

```
ex. 10 DIM A(10,4)
    20 PRINT DIM(A,1),DIM(A,2)
    ...
```

+ ERL  
-Returns the linenumber in which an error occurred.  
-An error-linenumber equal to zero means that a direct command mode error has occurred.

```
ex. 10 ON ERROR GOTO "TRAP"
    20 ERROR 10
    ...
    1000 "TRAP PRINT ERL:RESUME NEXT
    ...
```

+ERR  
-Returns the error number.

```
ex. 10 ON ERROR GOTO "TRAP"
    20 ERROR 50
    ...
    1000 "TRAP PRINT ERR:RESUME NEXT
    ...
```

+ERR\$  
-Returns the identification of the DBASIC-extension wich caused the error.  
-If the error was generated by a DBASIC statement or command then ERR\$ is empty.

DBASIC V2.1 page 7

```
ex. 10 ON ERROR GOTO "TRAP"
    20 DCR 4:REW 1000
    ...
    1000 "TRAP PRINT ERR$,ERR:REUME NEXT
    ...
```

note : DBASIC V2.1 DCR version only.

+INTEGER(<floating point expression>)  
-Converts a floating point value to an integer value.  
-Note the difference with the INT function : the INT function returns a truncated floating point value.  
-The INTEGER function can be usefull to convert expressions to the integer type before passing them as function parameters to a procedure or user-defined function.

```
ex. 10 PROCEDURE TEST FN FUN
    20 PRINT FUN
    30 END PROC
    ...
    100 FOR I!=0.1 TO 1.0 STEP 0.1
    110 TEST INTEGER(SIN(I!)*10.0)
    120 NEXT I!
```

+ISTR\$(<integer expression>)  
-Returns a string representing the value of <integer expression>.  
-The ISTR\$ function is familiar to the STR\$ function, the ISTR\$ function however returns a string without added '.0'

```
ex. 10 A=1000000000
    20 PRINT ISTR$(A),STR$(A)
    ...
```

+IVAR(<integer expression>)  
-Returns the integer value stored at memory location <integer expression>.  
-The IVAR function can be usefull to point into an integer type array.

```
ex. 10 CLEAR 10000
    20 DIM A(10,10)
    30 FOR I=0 TO 10:FOR J=0 TO 10
    40 A(I,J)=I*J:NEXT:NEXT
    50 FOR I=VARPTR(A(0,0)) TO VARPTR(A(10,10)) STEP 4
    60 PRINT IVAR(I)
    70 NEXT
```

+NDIM(<array name>)  
-Returns the number of dimensions reserved for <array name>.  
-Can be used in procedures and functions to test array-parameters

```

ex. 10 FUNCTION SUM(ARR A)
    20 REM---one dimensional integer array sum ---
    30 LOCAL SUM
    40 IF NDIM(A)<>1 THEN ERROR 100
    50 ELSE FOR I=0 TO DIM(A,1)
    60 SUM=SUM+A(I):NEXT
    70 END IF
    80 FN=SUM
    90 END FN
    ...

```

**+REAL(<integer expression>)**  
 -Converts an integer value to a floating point value.  
 -The REAL function can be useful to convert expressions to the floating point type before passing them as function parameters to a procedure or user-defined function.

```

ex. 10 PROCEDURE TEST FN Z!
    20 PRINT Z!
    30 END PROC
    ...
    100 TEST REAL(XMAX)

```

**+VAR(<integer expression>)**  
 -Returns the floating point value stored at memory location <integer expression>  
 -The VAR function can be useful to point into a floating point type array.

```

ex. 10 CLEAR 10000
    20 DIM A!(10,10)
    30 FOR I=0 TO 10:FOR J=0 TO 10
    40 A!(I,J)=I*J:NEXT:NEXT
    50 FOR I=VARPTR(A!(0,0)) TO VARPTR(A!(10,10)) STEP 4
    60 PRINT VAR(I)
    70 NEXT

```

**+VAR\$(<integer expression>)**  
 -Returns the string to which memory location <integer expression> points to.  
 -The VAR\$ function can be useful to point into a string type array. It is possible to sort a string type array by just swapping the pointers to the strings.

```

ex. 10 CLEAR 1000
    20 DIM A$(100)
    30 FOR I=0 TO 100
    40 A$(I)=CHR$(I)
    50 NEXT
    60 FOR I=VARPTR(A$(0)) TO VARPTR(A$(100)) STEP 2
    70 PRINT VAR$(I)
    80 NEXT

```

## 4. PROCEDURES AND FUNCTIONS

**+Declaration**

-Procedures and functions always have to be declared explicitly.  
 -Within a procedure or function declaration, other procedures or functions can be declared.

```

ex. 10 PROCEDURE PROMPT
    20 PROCEDURE HOME
    30 PRINT CHR$(12);
    40 END PROC
    50 HOME
    60 PRINT "DBASIC V2.1"
    70 END PROC

```

-Procedure PROMPT, called as PROMPT, will clear the screen and print the DBASIC V2.1 prompt in the upper left corner. Note that line 20 to 40 in fact declare a 'sub-procedure' of the procedure PROMPT.

**+Value parameters**

-A value parameter will be assigned a value equal to the evaluation of the matching expression in the caller.

```

ex. 10 PROCEDURE FORWARD R
    20 XNEW=X!+R*COS(ALPHA!);YNEW=Y!+R*SIN(ALPHA!)
    30 IF PEN=DOWN THEN DRAW X!,Y! XNEW!,YNEW! PENCOL
    40 END IF
    50 X!=XNEW!:Y!=YNEW!
    60 END PROC
    ...
    100 FORWARD XMAX/2
    ...

```

-Executing line 100 will draw a line with length equal to XMAX/2.

```

ex. 10 FUNCTION FAC!(N)
    20 IF N=0 THEN FN=1.0
    30 ELSE FN=N*FAC!(N-1)
    40 END IF
    50 END FN
    60 FOR I=0 TO 10:PRINT FAC!(I):NEXT

```

-Executing line 60 will print 0! to 10! (faculty).  
 -Note that FAC! is defined recursively.

```

ex. 10 FUNCTION SEC!(X!)=1/COS(X!)
    20 FUNCTION CSC!(X!)=1/SIN(X!)
    30 FUNCTION COSH!(X)=(EXP(X!)+EXP(-X!))/2.0
    ...

```

**+Variable parameters**

-Variable parameters will refer to the corresponding variables in the procedure- or function-caller.

```
ex. 10 PROCEDURE SWAP VAR A$,B$
    20 LOCAL HELP$
    30 HELP$=A$:A$=B$:B$=HELP$
    40 END PROC
    ...
    100 HELLO$="HELLO":BYE$="BYE"
    110 SWAP HELLO$,BYE$
    120 PRINT HELLO$,BYE$
```

-The effect of the procedure call SWAP HELLO\$,BYE\$ will be that the contents of variables HELLO\$ and BYE\$ will have been swapped.

**+Array parameters**

-Besides unsubscripted variables you can also transfer complete arrays to a procedure or function.

```
ex. 10 DEF FN MAX(ARR A)
    20 LOCAL I,MX
    30 FOR I=0 TO DIM(A,1)
    40 IF A(I)>MX THEN MX=A(I)
    50 END IF
    60 NEXT
    70 FN=MX
    80 END FN
    ...
    100 PRINT "THE MAXIMUM RATE IS";MAX(RATE())
    ...
```

-In this example RATE is a one-dimensional integer type array.  
-Note that the arrays to pass to a procedure or function have to be noted as <array name>().

**+Function parameters**

-Function parameters will refer to the corresponding expressions in the procedure or function caller.

```
ex. 10 FUNCTION INTEG!(LOW!,HIGH!,STP! VAR X! FN Z!)
    20 LOCAL TOT!
    30 FOR X!=LOW! TO HIGH! STEP STP!
    40 TOT!=TOT!+STP!*Z!
    50 NEXT
    60 FN=TOT!
    70 END FN
    ...
    100 PRINT INTEG!(0.0,PI,0.1,X!,SIN(X!))
    110 PRINT INTEG!(0.0,PI/2,0.1,Y!,COS(Y!))
    ...
```

-The function INTEG! calculates approximately the integral of a mathematical function in the interval LOW! to HIGH! (the surface between the function and the X-axis).

-Note that the argument of the function is transferred as a variable parameter.

**+Local and global variables**

-The variables whose names are used in a procedure or function heading to specific value-,variable-,array- or function-parameters will be local variables: Their value will not have been changed after the procedure or function call

```
ex. 10 PROCEDURE DUMMY I
    20 I=0
    30 END PROC
    ...
    100 I=10
    110 DUMMY I
    120 PRINT I
    ...
```

-However when you call a procedure or function with variables, exactly the same as used in the procedure or function heading to specific variable- and/or array-parameters, these variables will be global.

```
ex. 10 PROCEDURE DUMMY VAR I
    20 I=0
    30 END PROC
    ...
    100 I=10
    110 DUMMY I
    120 PRINT I
    ...
```

-The LOCAL statement allows you to define extra local variables.  
-All other variables will be global to the procedure or function.

**+Easy programming**

-The following example shows how easy programming could be done using procedures (and functions).

```
ex. 1 REM -----
    2 REM --- DEFINITION TURTLE COMMANDS ---
    3 REM -----
    10 PROCEDURE PENUP: PENFLAG=0: END PROC
    20 PROCEDURE PENDOWN: PENFLAG=1: END PROC
    30 PROCEDURE FORWARD R: XNEW!=X!+R*COS(ALPHA!): YNEW!=Y!+R*SIN(ALPHA!)
    40 IF PENFLAG=1 THEN DRAW X!,Y! XNEW!,YNEW! PENCOL: END IF
    50 X!=XNEW!: Y!=YNEW!: END PROC
    60 PROCEDURE BACK R: XNEW!=X!-R*COS(ALPHA!): YNEW!=Y!-R*SIN(ALPHA!)
    70 IF PENFLAG=1 THEN DRAW X!,Y! XNEW!,YNEW! PENCOL: END IF
```

```

80  X!=XNEW!:Y!=YNEW!:END PROC
90  PROCEDURE LEFT TETA:ALPHA!=ALPHA!+TETA*PI/180.0:END PROC
100 PROCEDURE RIGTH TETA:ALPHA!=ALPHA!-TETA*PI/180.0:END PROC

110 PROCEDURE CLEAN:FILL 0,0 XMAX,YMAX 0:END PROC
200 PROCEDURE SQUARE L:PENDOWN:FOR INDEX=1 TO 4
210 FORWARD L:LEFT 90:NEXT:RIGTH 360:PENUP:END PROC
300 PROCEDURE PENTA L:PENDOWN:FOR INDEX=1 TO 5
310 FORWARD L:LEFT 72:NEXT:RIGTH 360:PENUP:END PROC
1000 REM -----
1010 REM --- START ---
1020 REM -----
1030 MODE 6:COLORG 0 5 8 14:PENCOL=5
1040 X!=XMAX/2:Y!=YMAX/2
1050 FOR I=50 TO 10 STEP -1
1060 PENTA I:FORWARD 5:LEFT 5:NEXT
...

```

-In this example a few 'turtle' commands are defined as procedures (PENUP,PENDOWN,FORWARD,BACK,LEFT,RIGTH). Combining these 'turtle' commands in procedures as SQUARE and PENTA allows you to draw more or less complex shapes with a single command.

## 5. DBASIC EXTENSIONS

### +Why extensions

-DBASIC statements and commands are principally added for structured programming. Special graphical commands (ex. Turtle graphics) or commands to operate I/O-devices (ex. Memocom-MDCR) are not included on purpose: not everyone will use the graphical possibilities of the DAI-pc ore will have a DCR-drive connected. However, if neccessary, these special graphical commands or I/O-driving packages can be integrated in DBASIC. They can be grouped in so-called DBASIC-EXTENSIONS, loaded and relocated or deleted when needed.

### +\$SYSTEM extension

-\$SYSTEM is a DBASIC extension, designed to extend DBASIC with other extensions and/or delete from DBASIC the extensions which are not needed anymore.  
-\$SYSTEM contains 2 commands: \$EXTEND and \$DELETE

### +\$EXTEND <string expression>

-This command looks for the extension file (\$-type) with name equal to <string expression> and load it into memory starting at the Heap. The command table will be linked to DBASIC and the runtime code will be relocated.

### +\$DELETE <string expression>

-This command will search in memory for the extension with name equal to <string expression>. If the extension is in memory, the extension command-table will be masked out and the memory occupied by the extension is released if the extension is located just below the Heap.

## 6. ERROR REPORTING

### +DBASIC error reporting

-DBASIC has extended error reporting capabilities. A total of 56 error codes are known by DBASIC. Of these 56 error codes 21 are detected during compilation. Besides DBASIC error codes you can define your own error codes by using an error-number greater than any error-number used by DBASIC (>55 for DBASIC V2.1). These user-defined error codes will be reported as:

'ERROR nnn [in line nnnnn]

### +DBASIC extension error reporting

-Besides DBASIC error-reporting, there is also a DBASIC-EXTENSION error-reporting.  
-Thus extensions, like the Memocom-MDCR driving package, can have their own error-codes and error-messages.

ex. -The command 'DCR 10' will generate the error-message:

'DCR DRIVE NUMBER OUT OF RANGE'

-The command 'REW 1000+1' will generate:

'DCR FILE NUMBER OUT OF RANGE'

You can expect more DBASIC news in the next issue's. In the mean time I wish you much succes with DBASIC-programming.

Willy Coremans





# 20 OCT MEETING

# COMMANDES KENDOS

= 1er MEETING INTERNATIONAL DU DAI P.C. (M.I.D.) =

SAMEDI 20 OCTOBRE 1984

Le samedi 20 octobre 1984 (de 9 à 18h) à Nivelles (Belgique), aura lieu le 1er meeting international ayant pour objet, le DAI P.C.

Tous les passionnés de cet ordinateur, isolés ou regroupés aux seins de clubs, sont invités à cette manifestation exceptionnelle. Les principaux clubs, DAI, CAROLODAI, DAIC, DAINAMIC et MICRODAI seront représentés et pourront proposer aux visiteurs leurs services spécifiques.

Le programme de cette journée sera fonction de chacune des associations représentées. La mission principale du meeting est avant tout de favoriser les contacts entre les différents utilisateurs du DAI (parfois un peu isolés) et les clubs. C'est l'occasion pour eux d'échanger leurs vues et expériences, mais aussi, d'envisager des projets communs valorisant le DAI. Les néophytes pourront s'adresser à des personnes un peu plus expérimentées afin de résoudre des problèmes particuliers. Le maximum de nouveautés tant au niveau hard (un nouveau DAI?... ) que soft, sera présenté au public. Bref, tout ce qui se fait de mieux pour le DAI sera, nous l'espérons, rassemblé dans ce meeting international qui constitue une première dans le genre!

De plus, une tombola sera organisée et un prix sera offert par la société INDATA!

## COMMENT SE RENDRE AU MEETING?:

Autoroute PARIS-BRUXELLES: entre Charleroi et Bruxelles, sortie Nivelles-sud (19 bis). Le meeting a lieu dans les batiments du MOTEL-RESTAURANT NIVELLES-SUD. Un parking 400 places est à la disposition des visiteurs. Pour les étrangers qui souhaitent loger sur place, le prix des chambres est d'environ 1000 FB (140 FF). Un centre commercial se trouve à proximité du meeting.

## ADRESSE:

1er MEETING INTERNATIONAL DU DAI P.C.  
MOTEL "NIVELLES-SUD"  
Chaussée de Mons  
NIVELLES BELGIUM

## ENTREE:

50 FB OU 8 FF.

## RENSEIGNEMENTS:

Fabrice DULUINS: (0)67 / 22.82.10  
allée Tour Renard 4, B-1400 NIVELLES

Christian POELS: (0)41 / 37.16.06  
rue des Bas-Sarts 10, B-4100 SERAING

NB: Le meeting reste ouvert à d'éventuelles nouvelles participations de particuliers ou de clubs. Contacter si possible les responsables (2 numéros ci-dessus) avant le 1er octobre, afin de réserver un stand.

Is this a meeting of CIA?



No, they export computers for KGB!

Test de matériel : Ken Dos.

Dans ce test, je ne vous parlerai plus de la vitesse ni de la qualité du Ken Dos qui vous ont été décrites suffisamment dans les précédents numéros. Je me pencherai surtout sur les instructions du système et plus particulièrement sur leur syntaxe. Ce test ne se veut donc pas trop critique.

Voici donc, une description de ces fameuses instructions (certaines sont mieux décrites que d'autres, mon excuse : je n'ai pas mon système depuis longtemps.)

Lexique : (S) = syntaxe  
(D) = drive numéro  
(N) = nom de fichier  
(d) = adresse de départ  
(f) = adresse de fin

de plus, les indications entres paranthèses () ne sont pas obligatoires.

- BAS : (S) : BAS  
Donne les pointeurs BASIC et permet de les changer. Très utile pour travailler avec des programmes comme, par exemple, FGT.

- BUF : (S) : BUF(n)"(x)"  
(n) = numéro de buffer (1-5)  
(x) = EDT : édite le buffer (n)  
= PRT : imprime le buffer (n)  
= CLR : efface le buffer (n)  
= SET : initialise le buffer (n)  
Cette commande vous permet de créer des disk-buffer, ce qui vous permet d'utiliser n'importe quel programme machine avec votre Ken Dos.

- CAS : (S) : CAS  
Branche tout le système sur cassette audio pour la lecture et l'écriture des programmes.

- CLR : (S) : CLR(D)  
Déprotège la disquette contre le formattage. (voir plus loin, commande 'PRT')

- COM : (S) : COM(D) ou COM(D),"(x)(N)"  
x = (+) le fichier peut tourner en auto-run  
= (++) fichier auto-run  
= (&) fichier de commande  
= () le statut du fichier est effacé.

Permet de créer des fichier de commandes (nouvelles) en auto-start. Cela permet d'étendre la table des commandes.

-1-

- CPM (S) : CPM  
Branche le système sur le CPM (optionnel).  
(le CPM n'est pas encore terminé.)
- DCR (S) : DCR  
Branche tout le système sur cassette digitale  
pour la lecture et l'écriture des programmes.
- DIR (S) : DIR(D) ou DIR(D)"A" ou DIR(D)"I"  
A = donne toutes les informations sur les  
fichiers  
I = sous-directory (utilisation future) (???)  
Donne le directory de la disquette (liste des  
fichiers).  
Fonctions supplémentaires :  
-barre d'espace : fichier suivant  
-curseur gauche : LOAD fichier  
-curseur droit : LOAD+RUN fichier  
-RETURN : retour au mode commande
- DNA (S) : DNA  
Branche le système sur le DNA, CALLM2000  
n'est plus nécessaire. Le DNA doit être  
évidemment en mémoire !
- FWP (S) : FWP  
Branche le système sur le FWP, UT/Z3/G400  
ne sont plus nécessaires. Le FWP doit être  
évidemment en mémoire !
- GET (S) : GET(D)"(N),x,xx"  
x = numéro d'enregistrement (1-256)  
xx = buffer numéro (1-5)  
Les buffers doivent d'abord avoir été  
créés à l'aide de la commande 'BUF'  
(voir plus haut).  
Lit l'enregistrement (x) du fichier (N).  
Place les données dans le buffer (xx).
- KEY (S) : KEY ou KEY(?) ou KEY"(D),(N)"  
Permet de former des 'touches fonctions' pour  
obtenir sur la simple pression d'une touche  
la commande 'SAVE' (voir plus loin) ou la  
commande 'HELP' (voir plus loin).  
C'est très utile pour la mise au point des  
programmes.
- LIB (S) : LIB ou LIB1  
LIB donne une liste des commandes du Ken Dos  
(library) et LIB1 donne une liste des  
commandes d'extensions. (créée avec  
l'instruction 'COM')(voir plus haut).
- PRT (S) : PRT(D)

-2-

\*COMMANDES KENDOS\*

Protège la disquette contre le formatage et  
donc, l'effacement des programmes.

- PUT (S) : PUT(D)"(N),x,xx"  
Enregistre le buffer (xx) sur le track  
(le secteur) (n) sous le nom (N). (commande  
GET)(voir plus haut).
- SPL (S) : SPL  
Branche le système sur le SPL. (voir plus  
haut, commandes 'FWP' et 'DNA').
- TEST (S) : TEST(D)  
Teste le drive (D) et branche les opérations  
de lecture et d'écriture sur ce drive.
- TIME (S) : TIME(D)  
Donne l'heure à l'écran ou l'écrit sur le  
disque, la carte optionnelle TIME doit être  
présente.
- WCAS (S) : WCAS  
Branche les routines de lecture sur le disque  
et les routines d'écriture sur les cassettes.
- CLOSE (S) : CLOSE(D)"(N)"  
Ferme le fichier spécifié pour l'écriture.
- VOICE (S) : VOICE"(\$)"  
Envoie les données alphanumériques au  
synthétiseur de parole (optionnel).
- BACKUP (S) : BACKUP(D)"(X)"  
Copie intégralement la disquette du drive (D)  
sur la disquette du drive (X).
- COMPAC (S) : COMPAC(D)"(X)"  
Copie les fichiers non-deletés de la  
disquette se trouvant dans le drive (D) sur  
la disquette du drive (X).
- CREATE (S) : CREATE(D)"(N),(x),xx"  
x = type de fichier (UTY,BAS,RND,...)  
xx = nombre de blocs de 1K bytes.  
Cette commande crée (ou agrandit) des  
fichiers sur la disquette.
- DELETE (S) : DELETE(D)"(N)"  
Efface le programme (N) de la disquette se  
trouvant dans le drive (D).
- FORMAT (S) : FORMAT(D)  
Formate la disquette dans le drive (D).
- LPRINT (S) : LPRINT(n) ou LPRINT"(d),(f)"  
(n)= numero de buffer (1-5)

-3-

\*COMMANDES KENDOS\*

Envoie les données vers la RS-232.  
Si (f) est suivi de '!', le données sont  
alors prises de la RS-232 et mises en mémoire.

- MANUAL (S) : MANUAL(D)  
Lecture ou écriture manuelle de secteur ou  
tracks de la disquette se trouvant dans le  
drive (D).
- RENAME (S) : RENAME(D)"(N),(n)"  
(n) = le nouveau nom de fichier.  
Renomme un fichier d'une disquette.
- RESTOR (S) : RESTOR(D),"(N)"  
Redonne l'accet à un fichier deleté  
(voir plus haut : com. DELETE).
- UNLOCK (S) : UNLOCK(D) ou UNLOCK(D)"(N)"  
Déprotège un fichier ou un disque.  
(voir plus bas commandes LOCK et CODE.)
- VERIFY (S) : VERIFY(D) ou VERIFY(D)"(N)"  
Vérifie tous les secteurs de la disquette se  
trouvant dans le drive (D).
- LOAD (S) : LOAD"(D)(N)"  
Charge un programme BASIC.
- SAVE (S) : SAVE"(D)(N)"  
Sauve un programme BASIC.  
REM : LOAD & SAVE fonctionnent comme en BASIC
- DLOAD (S) : DLOAD(D)"(N),(d)"  
Charge un programme machine à partir de  
l'adresse (d) (cette adresse est optionnelle)  
Si (d) est suivi de '%', la commande 'OUT OF  
MEMORY' ne s'exécutera pas. C'est utile pour  
le chargement des copies d'écran).
- DSAVE (S) : DSAVE(D)"(N),(d),(f)"  
Sauve un programme machine sur la disquette.
- LOADA (S) : LOADA"(D)(N)"  
Lit un fichier de données.
- SAVEA (S) : SAVEA"(D)(N)"  
Ecrit un fichier de données.  
REM : LOADA & SAVEA fonctionnent comme sur  
cassettes.
- R (S) : R (D)(N)  
Charge un fichier machine.  
'%' : comme la commande DLOAD.
- W (S) : W(d) (f) (D)(N)  
Sauve un fichier machine.

REM : R & W fonctionnent comme en BASIC.

- BANK (S) : BANK(x):(xx)  
(x) = numéro de la banque  
(xx) = nom du programme à lancer  
Charge des données de la banque d'EPROM.  
Cette méthode est très rapide. Par exemple,  
FWP (avec lequel j'écris cet article) existe  
sur EPROM, pour le charger, il suffit de  
faire : BANK1:FWP ...1/4 de seconde de  
patience et le programme est lancé.
- CODE (S) : CODE  
Permet d'entrer un code pour la lecture et  
l'écriture de programmes (ou de disque)  
protégés par la commande 'LOCK' (voir plus  
haut). Le code est numérique et est de 5  
nombres.
- COPY (S) : COPY(D)"(N),(x)"  
(x) = numéro du drive d'écriture  
Copie un programme d'un disque à l'autre.
- DATE (S) : DATE  
Entre la date en mémoire.
- DISK (S) : DISK  
Branche les commandes de lecture et  
d'écriture sur le disque. Par exemple après  
la commande 'DCR' (voir plus haut).
- FIND (S) FIND(x)"(\$)" ou FIND"(\$),(d),(f)"  
(\$) = une variable alphanumérique  
  
(x) = un numéro de buffer (1-5)  
Cherche une suite de bytes en mémoire ou dans  
un buffer.
- HELP (S) HELP  
Donne à l'écran un menu d'ou vous pourrez  
tirez des explications condensées (mais  
claires) du fonctionnement de certaines  
commandes.
- INIT (S) INIT(D)  
Initialise un disque : date et nombre de tracks  
par face (40 ou 80 selon le système).
- KILL (S) KILL(D)"(N)"  
Efface définitivement un fichier deleté  
(voir plus haut commande 'DELETE').
- LOCK (S) LOCK(D) ou LOCK(D)"(N)"  
Protège un disque (ou un programme) à l'abris  
des regards indiscrets ou des maladroits.  
Quand la commande LOCK est tapée, c'est le

# ARRAY ... 254

code actuellement en mémoire qui est enregistré et requis pour une prochaine lecture.

- NAME (S) NAME(D)"(X)"  
(X) = nom de la disquette  
Cette fonction nomme la disquette (le nom apparait lors du directory).
- OPEN (S) OPEN(D)"(N)"  
Ouvre un fichier pour l'écriture.
- RCAS (S) RCAS  
Branche la lecture sur les cassettes et l'écriture sur les disquettes.
- SWAP (S) SWAP(X)  
(X) numéro de buffer  
Inverse les données du buffer (X) avec celles du buffer1.

### Remarques générales :

Si l'on ne précise pas le numéro du drive à employer dans une commande, le Ken Dos choisira automatiquement le drive 0.

Dans la nouvelle version du DOS, il suffit de presser la touche curseur gauche pour obtenir un directory de la disquette se trouvant dans le drive 0.

Voilà, c'est à peu près tout, j'espère que cette description vous aura intéressé (ou peut-être décidé)...

Marc VANDERMEERSCH

```
3 REM TUNE YOUR FLUTE / GUITAR, by Emil Zahner CH 8910 Affoltern 26.12.83
4 TON=440:TON!=TON
5 PRINT CHR$(12):PRINT SPC(23);"TASTEN / KEYS":PRINT "Ton aus/off ";
6 PRINT CHR$(136);" ein/on ";CHR$(137);
7 PRINT " hoeh/er/up ";CHR$(#7F+95);" tiefer/down ";CHR$(#7F+13)
8 PRINT "A = 440 Hertz (Schwingungen / cycles)";
9 PRINT " % Abweichung / Off"
10 G=GETC:G=GETC:G=GETC
20 G=GETC:IF G<1 THEN 20
40 IF G=16 THEN TON=TON+1:SOUND 0 0 0 0 FREQ(TON)
50 IF G=17 THEN TON=TON-1:SOUND 0 0 0 0 FREQ(TON)
60 IF G=18 THEN SOUND OFF
70 IF G=19 THEN SOUND 0 0 0 0 FREQ(TON)
80 PRINT " ", " ", TON,:GOSUB 100:IF CURY<3 GOTO 5
90 GOTO 20
100 TONP!=100*TON/TON!
110 PRINT " ";TONP!-100:RETURN
```

The DAI Basic doesnot allow arrays with subscripts which are higher than 254. In programs which handle a lot of data, it is often very usefull if bigger arrays could be used, like in other type of personal computers. To overcome this problem, more than one array could be dimensioned. But this doesnot simplify finding data in the arrays, or sorting the contents.

Another method is the use of multi-dimensional arrays. An example will demonstrate it:

```
---- IMP INT ----
100 DIM N$(A,254)
```

'A' must be a number >0; its maximum value depends on the reserved heap space (CLEAR). For 600 array-elements, 'A' must be at least 2, giving 3 times 255 elements.

To fill this array with 600 data elements, the following routine must be used:

```
1000 FOR X=1 TO N (*N* is 600)
1010 N$(X/255,X-(X/255)*255)="....."
1020 NEXT
```

The first 254 elements are stored in N\$(0,1) thru N\$(0,254), the second 254 elements in N\$(1,-) and the rest in N\$(2,-).

Using the subscripts 'X/255,X-(X/255)\*255' in all other routines which have to handle the array data allow easy access to a number of array elements which is only limited by the maximum dimension of the heap (32K).

(c) - Jan Boerrigter, Febr. 1984

## CORRECTIONS FIRMWARE MANUAL - 3

The following updates can be made in your copy of the DAI pC firmware manual:

#C1E9: Sign extend routine, must read:

RLC - RLC - RRC - RAR - RET

#C311: Label LC71 must read: JP :C316.

Micro Service has no copies of the 'DAI pC Firmware manual' available anymore. Printing of a new edition is not planned, unless at least 100 copies are ordered.

Jan Boerrigter - August 1984.

```

10 REM DAI 48K 74S288-PROM-TABLES / MEYSTRE OCT.83
20 CLEAR 1000
30 DIM MASK(7),T$(4),R(127):RESTORE
40 FOR I=0 TO 7:READ MASK(I):NEXT
50 FOR I=1 TO 4:READ T$(I):NEXT
60 ADR=VARPTR(R(0)):A=ADR
70 FOR I=0 TO 7:FOR J=0 TO 15:READ B:POKE A,B:A=A+1:NEXT:NEXT
100 RAM=ADR:FOR T=1 TO 4
120 PRINT CHR$(12):CURSOR 10,22:PRINT "74S288-PROM ";T$(T)
130 CURSOR 4,20:PRINT "PROM- 00000000000000001111111111111111"
140 CURSOR 4,19:PRINT "ADDR. 0123456789ABCDEF0123456789ABCDEF"
150 POS=17:FOR I=0 TO 7:CURSOR 5,POS-I:PRINT 7-I:NEXT
160 FOR PROMADDR=0 TO 31:PROMDATA=PEEK(RAM):RAM=RAM+1
170 FOR I=0 TO 7:CURSOR 10+PROMADDR,POS-I
180 IF PROMDATA IAND MASK(I)=0 THEN 200
190 PRINT "1":GOTO 210
200 PRINT "0"
210 NEXT
220 CURSOR 10+PROMADDR,8:PRINT HEX$(PROMDATA SHR 4)
230 CURSOR 10+PROMADDR,7:PRINT HEX$(PROMDATA IAND #F)
240 NEXT
270 CURSOR 4,8:PRINT "PROM-"
280 CURSOR 4,7:PRINT "DATA "
290 CALLM #D6DA:NEXT:GOTO 100
600 DATA 128,64,32,16,8,4,2,1
610 DATA "BLUE-DOT","GREEN-DOT","RED-DOT","YELLOW-DOT"
620 DATA #D3,#D3,#F3,#F3,#D7,#D7,#F7,#F7,#5B,#5B,#53,#53,#7B,#7B,#73,#73
630 DATA #5F,#5F,#57,#57,#7F,#7F,#77,#77,#11,#11,#11,#11,#12,#12,#93,#03
640 DATA #FE,#FF,#5E,#BE,#FE,#FF,#7E,#FE,#FE,#FF,#6E,#3E,#FE,#FF,#7E,#7E
650 DATA #FE,#FF,#7E,#FE,#FE,#FF,#7A,#FE,#FE,#FF,#7E,#7E,#FE,#FF,#7C,#76
660 DATA #48,#48,#48,#48,#58,#68,#C8,#48,#08,#40,#48,#48,#48,#48,#48,#48
670 DATA #48,#48,#48,#48,#48,#48,#48,#48,#48,#48,#48,#48,#48,#48,#48
680 DATA #8B,#8B,#27,#20,#07,#00,#4F,#4A,#0A,#0A,#06,#06,#06,#06,#1F,#1E
690 DATA #8B,#8B,#27,#20,#07,#00,#4F,#4A,#0A,#0A,#06,#06,#06,#06,#0F,#0E

```

74S288-PROM BLUE-DOT

```

PROM- 00000000000000001111111111111111
ADDR. 0123456789ABCDEF0123456789ABCDEF

7 1111111100000000000000000000000010
6 1111111111111111111111111100000000
5 00110011000011110000111100000000
4 11111111111111111111111111111110
3 00000000110011001100110000000000
2 00001111000000001111111100000000
1 11111111111111111111111100001111
0 11111111111111111111111111110011

PROM- DDFDFF555577775555777711111190
DATA 33337777BB33BB33FF77FF7711112233

```

\*PROM DATA\*

74S288 PROM GREEN-DOT

```

PROM- 00000000000000001111111111111111
ADDR. 0123456789ABCDEF0123456789ABCDEF

7 11011101110011001101110111001100
6 11101111111011111111111111111111
5 11011111111111111111111111111111
4 11111111101111111111111111111111
3 11111111111111111111111111111110
2 11111111111111111111111111111111
1 11111111111111111111111111111101
0 01000100010001000100010001000100

PROM- FF5BFF7FFF63FF77FF7FFF7FFF77FF77
DATA EFEEEEEEEEEEEEEEEEFA6EFFFFFC6

```

74S288-PROM RED-DOT

```

PROM- 00000000000000001111111111111111
ADDR. 0123456789ABCDEF0123456789ABCDEF

7 00000010000000000000000000000000
6 11111111011111111111111111111111
5 00000100000000000000000000000000
4 00001000000000000000000000000000
3 11111111101111111111111111111111
2 00000000000000000000000000000000
1 00000000000000000000000000000000
0 00000000000000000000000000000000

PROM- 444456C4044444444444444444444444
DATA 88888888808888888888888888888888

```

74S288-PROM YELLOW-DOT

```

PROM- 00000000000000001111111111111111
ADDR. 0123456789ABCDEF0123456789ABCDEF

7 11000000000000001100000000000000
6 00000011000000000000001100000000
5 00110000000000000011000000000000
4 00000000000000110000000000000000
3 11000011110000111100001111000011
2 00101010001111110010101000111111
1 10101011111111111010101111111111
0 10101010000000101010101000000010

PROM- 88220044000000118822004400000000
DATA B87070FAAA6666FEB87070FAAA6666FE

```

\*PROM DATA\*



# SAVEA & LOADA

Deze instructies laten toe op eenvoudige wijze array's (rijen getallen of tekst) op audio cassette of DCR te bewaren.

SAVEA Q% :zet array Q% op cassette  
LOADA TEKST\$ :laat de cassette lopen tot de DAI een strook tegenkomt die hij (zij?) herkent als array (aangeduid door een cijfer 2 vóór de titel). Deze array wordt gelezen en in array TEKST\$ geplaatst.

Omdat de (Engelstalige) handleiding op dit punt nogal vaag, onvolledig, verwarrend, en soms zelfs fout is vatten we de spelregels nog eens samen:

1. Elk array dat je wil bewaren of inlezen moet op voorhand gedimensioneerd worden, zelfs als je slechts één getal of één tekst wil bewaren.

b.v. 10 DIM TEKST\$(0)  
20 INPUT TEKST\$(0)  
30 SAVEA TEKST\$

2. Je kan een array een bepaalde titel meegeven op cassette. Deze titel kan ook een variabele zijn!

b.v. 10 DIM GETALLEN(10)

...  
100 SAVEA GETALLEN " Meetresultaten"

Op de cassette vind je deze 11 getallen terug (met b.v. CHECK) onder de titel:

2 Meetresultaten  
↑ duidt op array

Een ander voorbeeld:

10 DIM KIP(100)

...

100 INPUT "Welke maand"; MAAND\$: PRINT

110 FOR I%=1 TO 100

120 PRINT "Hoeveel eieren heeft kip nummer "; I%; " deze maand gelegd ";

130 INPUT KIP(I%): PRINT

140 NEXT

200 SAVEA KIP " Legresultaten van de maand "+MAAND\$

↑  
naam

↑  
titel

3. De naam van de array wordt niet op cassette bewaard. Het is dus niet nodig dezelfde naam te gebruiken bij het opnieuw inlezen.

b.v. 10 INPUT "Geef een getal, kleiner dan 256"; AANTAL%: PRINT  
20 DIM A%(AANTAL%)  
30 FOR I% = 0 TO AANTAL%  
40 A%(I%) = RND(100)  
50 NEXT  
60 TITEL\$ = "Een reeks van "+STR\$(AANTAL%)+ " willekeurige getallen"  
70 SAVEA A% TITEL\$  
...  
...  
200 DIM B%(255)  
210 CALLM#F000: REM REW1: LOOK (enkel voor DCR)  
220 LOADA B%  
230 FOR I%=0 TO 255: PRINT B%(I%): NEXT

Uit dit voorbeeld blijkt tevens dat bij het inlezen het array (B) groter mag gedimensioneerd worden dan het array (A) dat op cassette staat.

Wat kan er allemaal fout gaan?

1. Programma's of data die al op cassette stonden kunnen overschreven worden. DCR-gebruikers kunnen dit voorkomen door aan het begin van elk programma volgende regel in te lassen:

CALLM#F000: REM SKIP

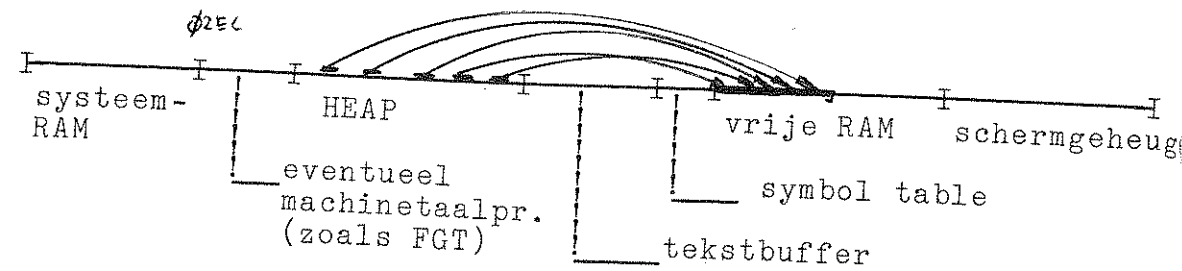
2. Het array dat je wil inlezen is groter dan het array dat je gedimensioneerd hebt. Dit resulteert in een "LOADING ERROR 1"

3. Het array dat je wil inlezen is van een ander type dan het array dat je gedimensioneerd hebt (FPT, integer of string): op het scherm verschijnt: "TYPE MISMATCH IN LINE xx"

# TELEPHON

PAGE 01 -- TELEPHON

4. Een "OUT OF MEMORY"-fout is mogelijk bij het bewaren of inlezen van tekstarrays. Omdat de inhoud van zo een tekstarray verspreid kan zitten over de heap wordt hij eerst gecopieerd in het vrije gedeelte van de RAM, zodat hij een aaneengesloten geheel vormt, en pas dan geSAVED.



Als deze vrije RAM te klein is krijg je een foutmelding. Wat valt daar tegen te doen?

1. De arrays opsplitsen in kleinere blokken
2. De heap verkleinen met een aangepaste CLEAR
3. Overgaan naar een MODE die minder schermgeheugen vereist (b.v. tekstmode)

5. Je wil schrijven op een cassette die niet "write-enabled" ("toegankelijk gemaakt") is. Dat is nogal vervelend, want het programma loopt dan vast zonder dat je een foutmelding krijgt. Druk je op de BREAK-toets, dan verschijnt \*\*\*BREAK op het scherm, i.p.v. het bekende \*\*\*BREAK IN LINE xx, zodat je niet kan verder gaan met CONT



Microprocessor interfacing

NEW ADDRESS :

MIPI P.O.B. 160 1610 AD BOVENKARSPÉL the Netherlands

```

10 REM ..... @ BY ROLF SCHALL AM 23.3.83 .....
20 REM .....
30 REM ..... TELEPHON-KURZ-KRIMI .....
40 REM .....
100 REM ..... VORSPANN
110 MODE 0:PRINT CHR$(12):COLORT 6 1 6 6:POKE #BA2D,#5A
120 POKE #BA27,ASC("T"):CURSOR 0,12:PRINT "ELEPHONE TERROR"
130 CURSOR 10,10:PRINT "A Mini-Super-Thriller by Rolf Schall"
140 PRINT :PRINT TAB(10);"Turn up volume and wait ..."
150 ENVELOPE 0 0,255:15,255;
160 ENVELOPE 1 15,255:0,255;
170 SOUND 0 0 10 0 FREQ(110.0):SOUND 1 1 10 0 FREQ(55.0)
180 WAIT TIME 500

200 REM ... TELEPHONHOERER
210 COLORG 6 1 6 1:MODE 6
250 R1:=300.0:R2:=280.0:R3:=277.0:R01:=R1!*R1!:R02:=R2!*R2!:R03:=R3!*R3!
255 XM:=120.0:YM:=-150.0:XMS:=0.0:YMS:=126.0
260 FOR X!=-100.0 TO 0.0:Y:=SQR(R01!-X!*X!)
270 DOT XM!+X!,YM!+Y! 17:DOT XM!-X!,YM!+Y! 17:DOT XMS!+Y!,YMS!+X! 19:DOT
XMS!+Y!,YMS!-X! 19:NEXT
280 FOR X!=-55.0 TO 0.0:Y:=SQR(R02!-X!*X!):DOT XM!+X!,YM!+Y! 17:DOT XM!-X!,
YM!+Y! 17
285 DOT XMS!+Y!,YMS!+X! 19:DOT XMS!+Y!,YMS!-X! 19:NEXT
290 FOR X!=-53.0 TO 0.0:Y:=SQR(R03!-X!*X!):DOT XM!+X!,YM!+Y! 23:DOT XM!-X!,
YM!+Y! 23:NEXT

295 REM ... APPARAT
300 FOR I:=1.0 TO 8.0:READ X1!,Y1!,X2!,Y2!
310 1 DRAW XM!+X1!,Y1! XM!+X2!,Y2! 17:DRAW XM!-X1!,Y1! XM!-X2!,Y2! 17
320 1 DRAW Y1!-YM!,YMS!+X1! Y2!-YM!,YMS!+X2! 19:DRAW Y1!-YM!,YMS!-X1!
1 Y2!-YM!,YMS!-X2! 19
330 NEXT
350 FOR I:=1.0 TO 11.0:READ X1!,Y1!,X2!,Y2!
360 1 DRAW XM!+X1!,Y1! XM!+X2!,Y2! 23:DRAW XM!-X1!,Y1! XM!-X2!,Y2! 23
370 NEXT

380 REM ... WAEHLSCHIEBE
400 MS:=120.0:MC:=80.0:R1:=40.0:R2:=32.0:R3:=28.0:R4:=20.0
410 FOR I:=0.0 TO PI/2.0 STEP 1.0/R1!
420 1 X!=R1!*SIN(I!):Y!=R2!*COS(I!):XW!=X!/1.8:YW!=Y!/1.8
430 1 DOT MS!+X!,MC!+Y! 23:DOT MS!-X!,MC!+Y! 23:DOT MS!+X!,MC!-Y! 23:DOT
1 MS!-X!,MC!-Y! 23
440 DOT MS!+XW!,MC!+YW! 23:DOT MS!-XW!,MC!+YW! 23:DOT MS!+XW!,MC!-YW! 23:
DOT MS!-XW!,MC!-YW! 23:NEXT
450 P!=PI/7.0:R1:=32.0:R2:=24.0:R3:=5.0:R4:=4.0:FOR I:=PI-P! TO 2.0*PI+P!
STEP P!
460 1 S0:=R1!*SIN(I!)+MS!:C0:=R2!*COS(I!)+MC!:FOR J:=0.0 TO PI/2.0 STEP
1 1.0/5.0
470 2 X!=R3!*SIN(J!):Y!=R4!*COS(J!)
475 1 DOT S0!+X!,C0!+Y! 23:DOT S0!-X!,C0!+Y! 23:DOT S0!+X!,C0!-Y! 23:DOT
1 S0!-X!,C0!-Y! 23:NEXT
480 1 READ N!,A!:S0:=S0!-2.0:C0:=C0!-3.0:FOR J:=1.0 TO A!
490 READ X1!,Y1!,X2!,Y2!:DRAW X1!+S0!,Y1!+C0! X2!+S0!,Y2!+C0! 23:NEXT:NEXT
500 FILL 104,76 136,84 23:FILL 105,77 135,83 20
510 DRAW XM!+30,55 XM!+17,67 23:DRAW XM!+29,54 XM!+16,66 23
520 DRAW XM!+31,56 XM!+16,68 23:DRAW XM!+31,57 XM!+16,68 23

1000 REM ... KLINGELZEICHEN
1010 REM

```

```

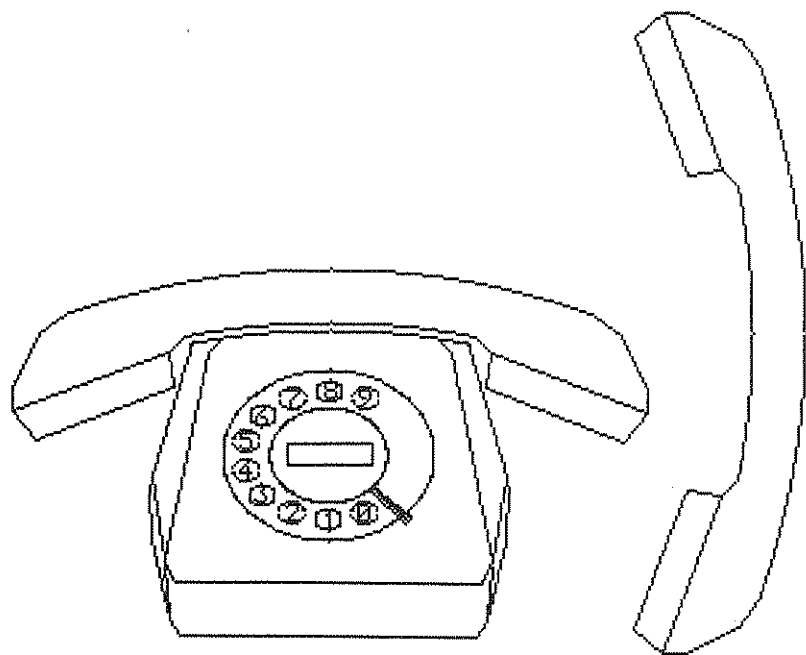
1020 FOR J=1 TO 20:IF GETC<>0 THEN 1500:GOSUB 2000:NEXT
1030 WAIT TIME 10:SOUND OFF
1040 FOR J=1 TO 30:IF GETC<>0 THEN 1500:GOSUB 2000:NEXT
1050 WAIT TIME 10:SOUND OFF
1060 FOR J=1 TO 900:IF GETC<>0 THEN 1500:NEXT
1070 GOTO 1010

1500 REM ... FREITON
1505 COLORG 6 6 1 1:ENVELOPE 0 15
1510 WAIT TIME 10:SOUND OFF :WAIT TIME 60:SOUND 0 0 15 0 FREQ(200.0)
1520 FOR I=1.0 TO 5.0:NEXT:SOUND OFF :WAIT TIME 5
1530 SOUND 0 0 5 0 FREQ(440.0)
1540 FOR I=1.0 TO 2.0:NEXT:SOUND 1 0 5 0 FREQ(440.0)
1550 WAIT TIME 300:SOUND OFF :COLORG 6 1 6 1
1560 IF INT(RND(300.0))<>1.0 THEN 1560:GOTO 1010

2000 REM ..... KLINGEL - U.P. ....
2010 WAIT TIME 1:OS=(OS+1) MOD 3:ENVELOPE 1 15,8;5,8;0:SOUND OS 1 10 0
    FREQ(2000.0)
2020 OS=(OS+1) MOD 3:SOUND OS 1 5 0 FREQ(529):RETURN
7000 DATA 55,124,61,118,100,132,110,125,110,125,120,104
7010 DATA 120,104,120,96,120,96,111,85,61,118,59,105
7020 DATA 120,96,61,118,111,85,59,105
7030 DATA 40,124,46,118,46,118,62,40,62,40,58,32,58,32,0,32
7040 DATA 62,40,67,72,67,72,52,122,56,12,0,12,56,12,60,18
7050 DATA 60,18,67,48,67,48,68,68,62,38,60,20

8998 REM ..... NUMMERN-GRAFIK-DATEI .....
8999 REM .... NUMMERSTRICHZAHLX1Y1X2Y2X1 usw....
9000 DATA 0,5,0,1,0,5,1,0,3,0,1,6,3,6,4,1,4,5,1,2,3,4
9010 DATA 1,3,1,0,3,0,2,0,2,6,1,5,1,5
9020 DATA 2,5,0,0,4,0,1,1,3,3,4,4,4,5,1,6,3,6,0,5,0,5
9030 DATA 3,6,0,1,0,1,1,0,3,0,4,1,4,2,3,3,2,4,3,5,3,5,0,6,4,6
9040 DATA 4,3,0,3,3,6,0,2,4,2,3,0,3,4
9050 DATA 5,6,0,1,0,1,1,0,3,0,4,1,4,3,0,4,3,4,0,5,0,6,1,6,4,6
9060 DATA 6,5,1,0,3,0,0,1,0,3,4,1,4,2,2,3,3,3,1,4,3,6
9070 DATA 7,3,1,0,1,2,2,3,4,5,0,6,4,6
9080 DATA 8,7,1,0,3,0,1,3,3,3,1,6,3,6,0,1,0,2,0,4,0,5,4,1,4,2,4,4,4,5
9090 DATA 9,5,1,0,3,2,1,3,2,3,4,3,4,5,0,4,0,5,1,6,3,6

```



# EDUCATION 6-7-8

**NEW**

SECONDARY EDUCATION TAPE 6

De Natuurlijke getallen (deel 1 : verbale factoren)  
 De Natuurlijke getallen (deel 2 : kennis)  
 De Natuurlijke getallen (deel 3 : rekenregels)  
 Het probleem van Fagnano  
 Berekenen van determinanten : informatie  
 Berekenen van determinanten  
 verschuivingen

Deze nieuwe onderwijstape van het diDAIsoftlabel bevat drie programma's met multiple-choice vragen over de verzameling der natuurlijke getallen.

Deze tape bevat verder een programma dat het probleem van Fagnano illustreert en oplost.

Enkele programma's in verband met de eigenschappen van determinanten en het samenstellen van verschuivingen sluiten deze didactische programmareeks af.

SECONDARY EDUCATION TAPE 7

Het ontstaan van goniometrische functies  
 De cyclometrische functies  
 Het ontstaan van kegelsneden  
 Veeltermbenadering voor sinus en cosinus

Een eerste programma van deze diDAIsofttape illustreert het ontstaan van de goniometrische functies SINUS en TANGENS. Terwijl een punt over een goniometrische cirkel loopt, genereert een ander punt de grafiek.

Deze tape stelt vervolgens elke leraar wiskunde in staat om zijn didactisch handelen bij het aanleren van de cyclometrische functies aan te vullen met een vlotte constructie van deze functies op zijn DAI-microcomputer.

Visueel voorstellen hoe een ellips, hyperbool of parabool ontstaat kan met de volgende programma's op deze tape 7.

Deze verzameltape wordt afgesloten met een programma dat het benaderen van de sinus- en cosinusfunctie door middel van veeltermen demonstreert.

SECONDARY EDUCATION TAPE 8

Functies van de tweede graad  
 Definitie continuïteit  
 Definitie afgeleide  
 Rationale functies  
 Methode van Newton

Een systematische opbouw van de functie  $y=x^2$  naar de functie  $y=ax^2+bx+c$  wordt behandeld in het eerste programma van deze derde nieuwe diDAIsofttape. Vervolgens bevat deze tape programma's die vrij moeilijke definities zoals continuïteit en afgeleide grafisch zeer knap toelichten en ondersteunen.

De rationale functies worden algemeen bestudeerd door middel van het volgende programma. Alle elementen komen aan bod : asymptoten, grafiek, nulpunten, oppervlakte onder de kromme, extreme waarden.

Het laatste programma op deze tape demonstreert de methode van Newton bij het bepalen van nulpunten van functies.

Bruno Van Rompaey wenst alle auteurs van deze programma's van harte proficiat, rekent op een blijvende inzet en belooft hen deze programma's zo snel mogelijk te zullen toezenden.

**NEW**

# Frogger

by T. ROGER

Frog is a game for one player where you must lead your family of frogs safely across the multi-lane motorway dodging the traffic. You must also cross a treacherous and fast moving river with many currents to catch your family unawares. Move your frogs across one at a time avoiding the ever increasing traffic and river populated by many crocodiles and turtles.

Cross the road by jumping your frog inbetween the fast moving vehicles as any contact is deadly and you lose one of your five lives. Cross the river by jumping log to log or use the crocodiles backs if you are brave. Step onto the turtles if you dare, but beware of the ones that dive because your frog cannot swim. Once you have crossed the river, jump your frog into one of vacant holes.

Avoid the holes occupied by the hungry crocodiles as you will be invited to dinner. Collect points for getting a frog safely to its hole, and bonus points for getting the whole family home. Collect some more points by jumping on any flies that appear, but make sure you are not caught by the snake. As the game progresses, more traffic takes to the roads, and more crocodiles populate the river.

Don't think this game is easy, as your family has one final enemy. Your family is on the run from the dreaded French Chef, who will cut your legs off if you take too long. Therefore, keep an eye on your time.

GOOD LUCK ...

**ARCADE CLASSIC**

# TOOLKIT 6

**NEW**

1. One Key commands.  
This little program ables you to enter all BASIC keywords with only one keystroke in command-mode as well in EDIT-mode. Extra flexibel due to automatic start-up.
2. Mirror, Zoom, Shrink.  
A three in one program.  
Mirror performs a reflection of a mode 6 picture on the other half of the screen.  
Zoom enlarges the picture, Shrink reduces the picture.
3. Wipes.  
Fourteen magnificent ways to mix video signals using blue-keying. Speed is adjustable in four steps.
4. Font 5X7.  
A new font for CHARGEN 1.0. Three-color characters from A-Z and 0-9 in a matrix of 5X7. CHARGEN 1.0 included.
5. Mona Lisa.  
This is the picture you have been waiting for. Leonardo's Madonna carressing a DAI Personal Computer.
6. Mona Lisa animation.  
This program makes your Mona Lisa smile, grinn, raise her shoulders and blow up her cheeks. Adapt it to other pictures by changing the data.
7. CHARGEN 1.0 Table creator.  
Create your own font or pictures with this program.
8. Sundown.  
Shiver-free simulation of a sunset.
9. Ribani animation.  
This program moves an object across a mode 2 screen. Adapt it to other objects by changing the data.
10. New Format listing.  
The perfect listing for BASIC programs. Leaves a free line before REM's, indicates level of FOR-NEXT, cuts no BASIC keywords, prints page numbers and titles etc... Most of the extra features can be switched off to obtain a conventional listing. Linelength and pagelength are adjustable. (see listing of TELEPHON-program p. 263-264 in this issue)
11. DCR-Directory.  
No more searching on your DCR tapes. This program does it for you. It brings the tape in position for every file you want after showing a menu of the tape contents. BASIC programs can be LOAded and RUN automaticcally if desired.

with build-in directory printout routine !

**NEW**

# PHOENIX

Your mission is to destroy a most original airbase of the mysterious PHOENIX-colony but first you have to succeed in 5 other missions :

- mission 1 : A first fleet of 14 phoenix-birds moving around in the sky , their eggs are dangerous bombs. You get 100 points for each hit. When you succeed in killing them you can pass to:
- mission 2 : A second fleet of 12 phoenix-birds ,same size , other color ,faster and much more dangerous. If you are still alive (you have 5 bases) you try the next battle :
- mission 3 : 9 eagles fly around in a very realistic way. They can dive and attack your base. They can change into eggs, change to an indestructible shape and become bird again... In the bird-state, you have to hit them 3 times before they die.
- mission 4 : 8 red eagles do attack your base in an even more strategic way ! They are faster and more aggressive than the previous fleet ! still alive ? ... you can try
- mission 5 : If your fight is successfull you can pass to: 16 fleet of rockets , all of them indestructible, all you can do is move your base around and avoid collision ! If your reflexes were good enough, you are ready for the final big battle : the PHOENIX-air-base.
- mission 6 : There is no time for admiration of the beautiful picture, the fight must go on: In the middle of the ship sits a little green monster : this is the one you have to hit. Don't forget the last PHOENIX-bird , he is also indestructible and attacking all the time ... Bombs are also falling from the centre of the ennemy-base, exactly where you have to hit him. The airbase has to be hit in a break-out style but be fast, the ship does regenerate after some time!

To complete your mission, you have 5 bases, a laser-canon and a protection-shield with time-limit.

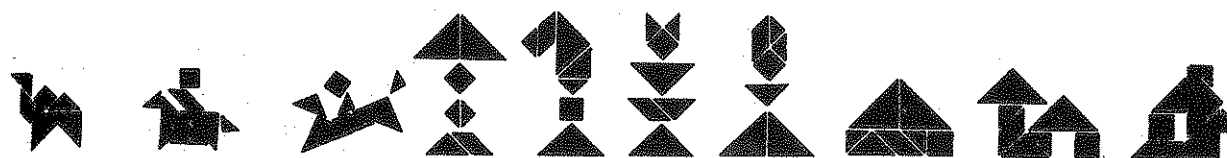
You get an extra base on 5000 points.

PHOENIX is more than one game : in fact it is a super collection of spacegames with an exiting satisfaction if you ever reach the final mission 6 !

PHOENIX is totally in machinelanguage and was written by Pascal Janin, 16 years old .

# TANGRAM

**NEW**



Tangram, the centuries old chinese puzzle, now available on your DAI-computer.

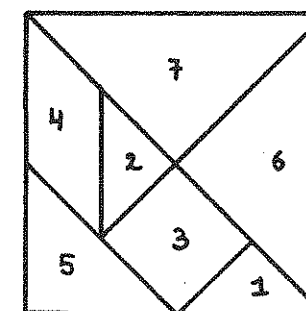
The comparison between Tangram and an ordinary puzzle holds only good in such an extent that by both puzzles the intention is to make one recognizable drawing with a number of several pieces.

An ordinary puzzle is made by a lot of different complicated pieces put together in only one way to make a result that is already known from the beginning.

On the other hand Tangram is made by only seven geometric figures who can be put together in a hundred ways.

The fascination lies in the variety of ways in which these pieces can be put together.

The seven figures of Tangram can be cut out of one square.



Putting together those seven figures to one square would be rather difficult without the solution as given up here.

The rules are very simple. All 7 pieces must be used for each drawing and are laid next to one other, never on top of one other.

After loading the program (load : run) you can create your own drawings by taking each one of the seven figures.

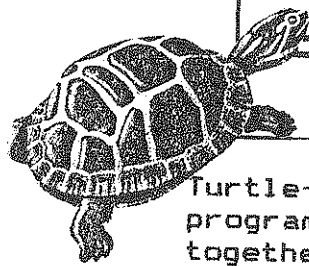
With the "cursor keys" you can move the chosen figure up and around the "working-screen" (mode 6A) and with the keys "<" and ">" you can rotate each figure.

When you have created a drawing with the seven figures you can save it on DCR or cassette.

On tape you will find 15 examples made by Tangram.

SUCCEES BY PUZZLING AND A LOT OF FANTASY AND  
DRAWING PLEASURE BY CREATING NEW DRAWINGS WITH  
TANGRAM





# TURTLE-BASIC

NEW

Turtle-BASIC features a subset of LOGO and structured programming facilities such as REPEAT-UNTIL and WHILE-WEND, together with all the DAI-BASIC statements. The subset of LOGO comes in three different languages, English, Dutch and French to meet your children's wishes.

English :

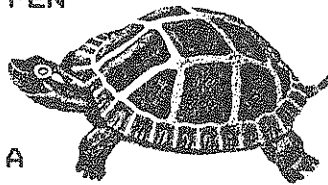
TURTLE  
FORWARD  
BACK  
PEN UP  
PEN DOWN  
PENCOLOR  
HIDE  
SHOW  
RIGHT  
LEFT  
MOVE TO  
TURN  
ERASE  
CLEAR SCREEN  
CLEAN

Dutch :

SCHILDPAD  
VOORUIT  
TERUG  
PEN OP  
PEN NEER  
PENKLEUR  
VERBERG  
TOON  
RECHTS  
LINKS  
GA NAAR  
DRAAI  
WIS  
OPNIEUW  
UITVEGEN

French :

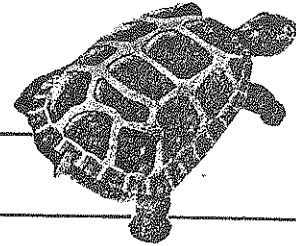
TORTUE  
AVANCE  
RECULE  
LEVE PEN  
BAISSE PEN  
COULEUR PEN  
CACHE  
MONTRE  
DROITE  
GAUCHE  
DEPLACE A  
TOURNE  
EFFACE  
RECOMMENCE  
ESSUYE



The other commands and statements are :

LLIST, LPRINT, HOME, REPEAT-UNTIL, WHILE-WEND, DOKE, DEEK, RESTORELINE, SWAP, IF-THEN-ELSE, INVERSE, NORMAL, PUSH, POP.

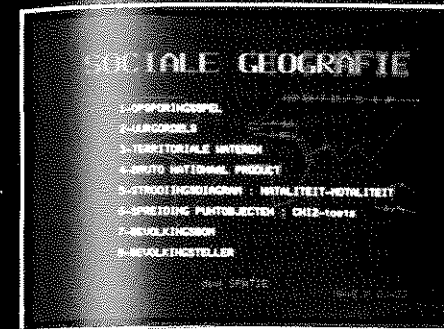
Turtle-BASIC is written as a semi-compiler and linked into DAI-BASIC, therefore no CALLM's are needed. Program's are exchangeable between the different languages. A demonstration program is included.



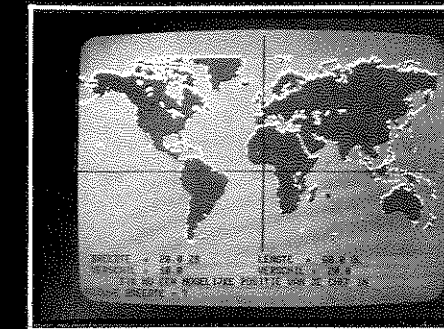
# EAGLES

NEW

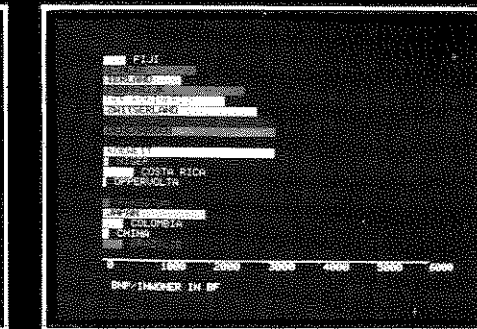
Eagles is a 100% machine language space game with many levels and beautiful features. You have to shoot many different space-monsters before they hit your base... Splendid graphics and sound, amusement for hours ... Eagles is written by Benoit Gortz.



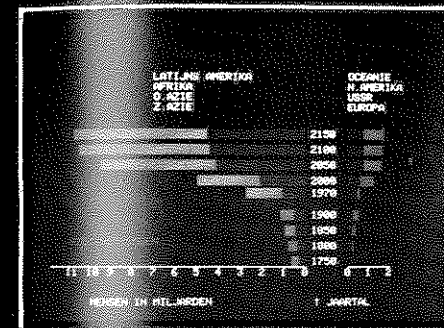
sociale geografie



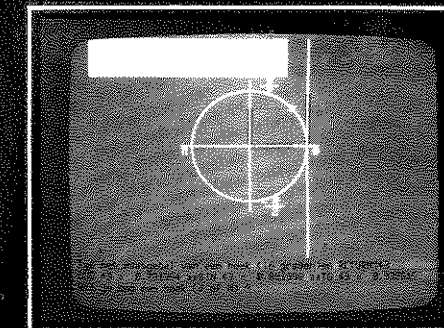
sociale geografie



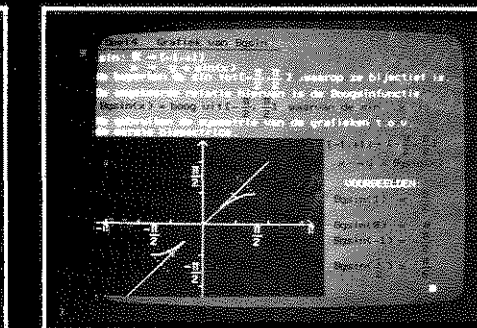
sociale geografie



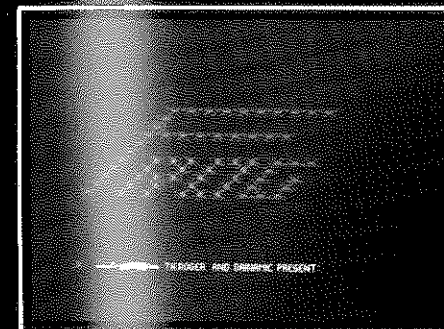
sociale geografie



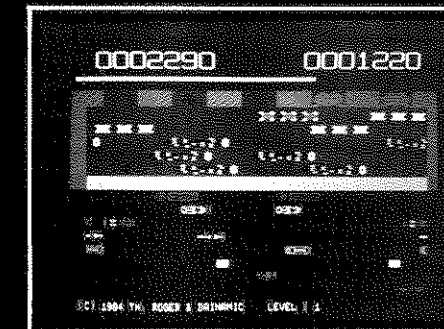
education



education



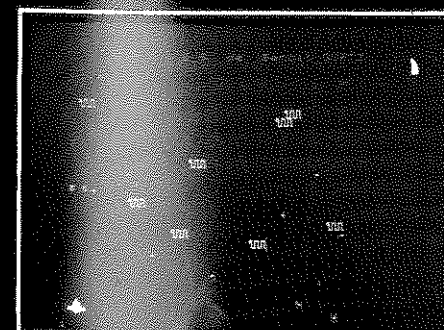
frogger



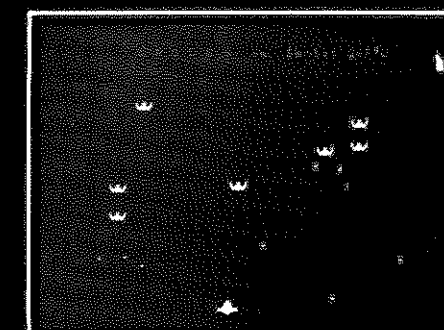
frogger



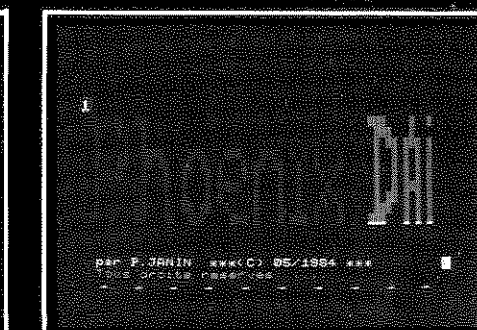
eagles



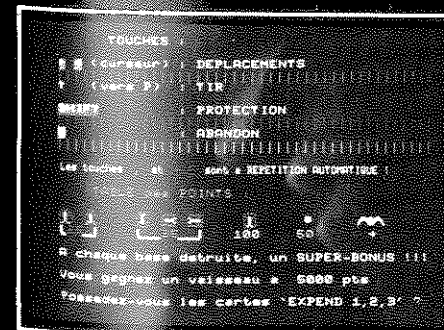
eagles



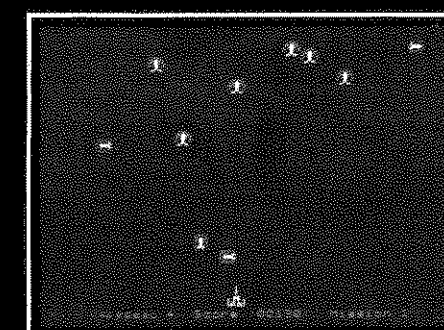
eagles



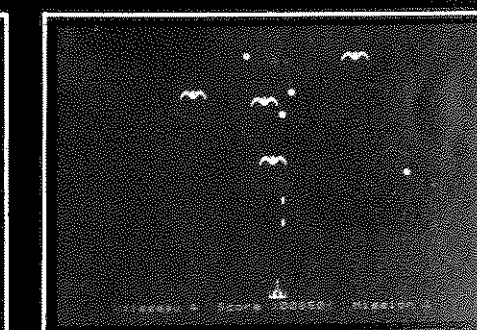
phoenix



phoenix



phoenix : mission 1



phoenix : mission 3