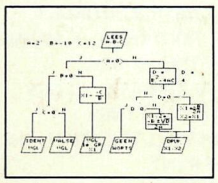
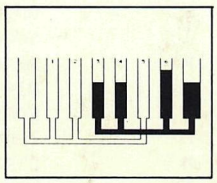
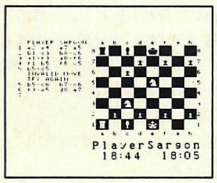
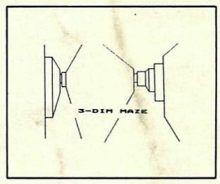
DAI NAMIC

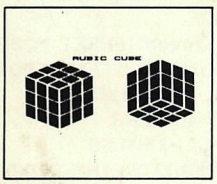
tweemaandelijks tijdschrift januari-februari 1982

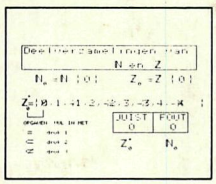


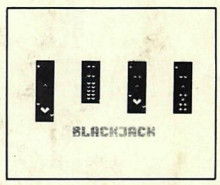












personal computer users club

een uitgave van dainamic v.z.w., heide 4 - 3171 westmeerbeek belgie

heide 4 3171 westmeerbeek belgium

INHOUD — CONTENTS

B = basic M = machine language H = hardware I = info

GETALCONVERSIES 8K M0 J.VERDONK

```
1     REM ***********************************
2     REM ** OMZETTEN VAN GETALSTELSELS  **
3     REM **          J.P.VERDONK        **
4     REM **                  APRIL 1980  **
9     REM ***********************************
10    DIM A$(9.0),G$(10.0),A(10.0),B(10.0),G(10.0)
11    PRINT CHR$(12)
20    PRINT "VAN WELK (X) NAAR WELK (Y) TALSTELSEL (MAX 16)?"
21    PRINT "VOOR VERANDEREN VAN TALSTELSEL TYP GETAL =0"
22    PRINT "VOOR EINDE PROGRAMMA TYP X EN Y=0"
30    INPUT "X= ";X:PRINT "  ";:INPUT "Y= ";Y:PRINT :GOTO 300
31    REM VERANDEREN VAN EERDER GEKOZEN TALSTELSEL
40    IF X=0.0 AND Y=0.0 THEN END
50    INPUT "GETAL ";Z$:C=0.0:D=0.0:E=-1.0:H=0.0:IF Z$="0" THEN PRINT
      :PRINT :GOTO 30
51    REM OMZETTEN NAAR DECIMAAL (TOT REGEL 150)
60    IF X=10.0 THEN C=VAL(Z$):GOTO 160
70    FOR I=LEN(Z$)-1.0 TO 0.0 STEP -1.0
80    A$(I)=MID$(Z$,I,1)
81    REM OMZETTEN VAN STRING NAAR ASCII WAARDE
90    A(I)=ASC(A$(I))
91    REM WAARDE BEPALEN PER CIJFER
100   IF A(I)>47.0 AND A(I)<58.0 THEN B(I)=A(I)-48.0:GOTO 350
101   REM OOK VOOR CIJFERS INDIEN NODIG
```

GETALCONVERSIES part 2

```
120   PRINT :PRINT "HET GETAL ";A$(I);" KEN IK NIET!":GOTO 50
130   C=C+(B(I)*INT((X^D)+0.5)):D=D+1.0
140   NEXT I
150   IF Y=10.0 THEN PRINT "  -";X;"  =  ";C;"  -10":GOTO 50
151   REM TERUGREKENEN IN ANDER STELSEL
160   E=E+1.0
170   F=INT(C/INT((Y^E)+0.5))
171   REM GROOTSTE EXPONENT BEPALEN
180   IF F>=1.0 THEN 160
181   REM NU GAAT HET NIET MEER DUS 1 ERAF
190   E=E-1.0:H=H+1.0
191   REM VAN LINKS > RECHTS
200   G(E)=INT(C/INT((Y^E)+0.5))
210   C=C-(G(E)*INT((Y^E)+0.5))
211   REM OMZETTEN NAAR ALFANUMERIEKE ASCII WAARDE
220   IF G(E)>=0.0 AND G(E)<=9.0 THEN G(E)=G(E)+48.0
230   IF G(E)>=10.0 AND G(E)<=16.0 THEN G(E)=G(E)+55.0
231   REM HETZELFDE MET DE REST
240   IF E>0.0 THEN 190
241   REM NU GETALLEN UITPRINTEN
250   PRINT "  -";X;"  =  ";
255   IF G(H)<47.0 THEN G(H)=32.0:H=H-1.0
260   IF G(H)>47.0 THEN PRINT CHR$(G(H));:G(H+1.0)=0.0:G(H)=0.0:IF H>1.0
      THEN H=H-1.0
270   IF H<1.0 THEN PRINT "  -";Y:GOTO 50
```

GETALCONVERSIES part 3

```
280   GOTO 255
290   STOP
300   IF X<0.0 OR X>16.0 THEN 20
310   IF Y<0.0 OR Y>16.0 THEN 20
320   GOTO 31
330   STOP
350   IF B(I)>(X-1.0) THEN PRINT :PRINT "HET CIJFER ";A$(I);" HOORT NIET
      THUIS I N HET ";X;"-TALLIG STELSEL !":GOTO 50
360   GOTO 130
370   STOP
```

```
VAN WELK (X) NAAR WELK (Y) TALSTELSEL (MAX 16)?
VOOR VERANDEREN VAN TALSTELSEL TYP GETAL =0
VOOR EINDE PROGRAMMA TYP X EN Y=0
X= ?10  Y= ?2
GETAL ?137  - 10.0  =  10001001  - 2.0
GETAL ?255  - 10.0  =  11111111  - 2.0
GETAL ?60  - 10.0  =  111100  - 2.0
GETAL ?0

X= ?10  Y= ?16
GETAL ?254  - 10.0  =  FE  - 16.0
GETAL ?33  - 10.0  =  21  - 16.0
GETAL ?100  - 10.0  =  64  - 16.0
```

D E D C E - B U S

DE DCE-BUS IS DE WEG OM DE COMPUTER TE VERBINDEN MET VERDERE APPARATEN. VOORBEELDEN : FLOPPY-DISKS, PRINTER MET PARALLEL INPUT, DOOR DAI GEFABRICEERDE REAL-WORLD-CARDS ,(EENVOUDIGE ?) ZELF ONTWORPEN SCHAKELINGEN,...
ER ZIJN 2 METHODES OM DE DCE-BUS TE GEBRUIKEN :

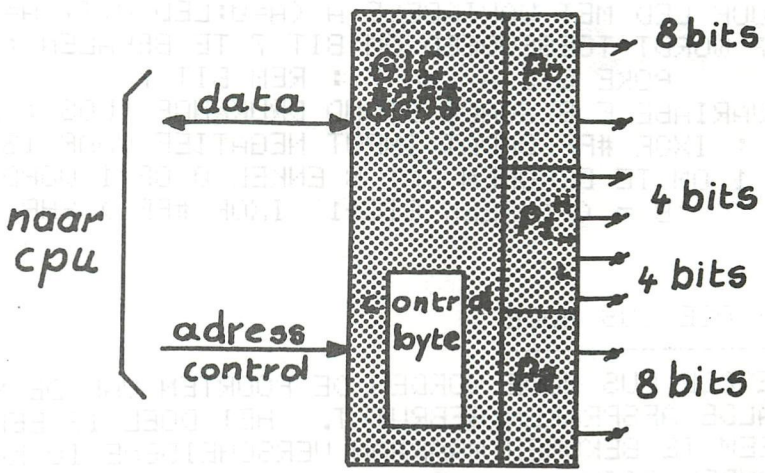
1) GEBRUIKEN ALS EEN EENVOUDIGE 3 X 8 BITS PARALLEL I/O. WE ZIJN DAN VRIJ IN DE OPZET VAN DE TE VERBINDEN SCHAKELING EN DE BESTURINGSSOFTWARE. WEL OPGELET VOOR DE JUISTE SPANNING (ZIE TTL NIVEAU'S) EN OVERBELASTING (MAX SINK = 1.6 MA/.4 V).
2) GEBRUIKEN VOLGENS DE DCE-BUS STANDAARD DOOR OFWEL ENKEL DAI REAL-WORLD-CARDS TE GEBRUIKEN OF ZELF COMPATIBELE SCHAKELINGEN ONTWERPEN

A) BESCHRIJVING VAN DE DCE-BUS:

INTERN IN DE PC BESTAAT DE DCE-BUS VOORNAMELIJK UIT 1 IC : INTELS 8255 (PROGRAMMABLE PERIPHERAL INTERFACE). VANAF NU ZULLEN WE DIT IC AANDUIDEN MET DE NAAM GIC (GENERAL INTERFACE CONTROL).
DIT IC BEVAT 2 8-BITS POORTEN (P0 EN P1) EN 2 4-BITS POORTEN (P2H EN P2L)
DE GIC IS ZO ONTWORPEN DAT WE DEZE 4 POORTEN AFZONDERLIJK ALS IN- OF UITGANG KUNNEN INSTELLEN. VOOR DEZE INSTELLING STUREN WE EEN KONTROLEBYTE NAAR EEN INTERN REGISTER IN DE GIC.
VIA DE DATA, ADRES EN CONTROL BUS IS DE 8255 VERBONDEN MET DE 8080.
ADRESSEN VAN DE GIC : #FE00 : POORT 0
#FE01 : POORT 1
#FE02 : POORT 2H + 2L
#FE03 : KONTROLEBYTE (NIET UITLEZEN !)
EEN POORT DIE ALS UITGANG INGESTELD IS GEDRAAGT ZICH ALS EEN LATCH. DWZ ALS WE ER EEN BEPAALDE WAARDE INSCHRIJVEN, DAN BLIJFT DEZE WAARDE BEHOUDEN TOT WE ER EEN NIEUWE WAARDE INSCHRIJVEN. DEZE WAARDE KUNNEN WE OOK UITLEZEN MET BV PEEK.
TOT HIERTOE HEBBEN WE ENKEL MODE0 VAN DE 8255 BESCHREVEN.
DE 8255 HEEFT OOK NOG MODES 1 EN 2 DIE EEN ZEER SNELLE IO TOELATEN (WERKEN MET HANDSHAKING EN INTERRUPT SIGNALEN).
IN DIT ARTIKEL BEPERKEN WE ONS TOT MODE0, OMDAT MODES 1 EN 2 VEEL COMPLEXER ZIJN.
BIJ WIJZIGING VAN DE MODE WORDEN ALLE OUTPUTLATCHES GERESET.
BIJ HARD RESET WORDEN ALLE POORTEN ALS INGANG GEZET.

OP DE VOLGENDE PAGINA GEVEN WE EEN LIJST VAN DE HEX KONTROLEBYTES VOOR EEN BEPAALDE INSTELLING VAN DE GIC (ENKEL MODE 0), MET DAARNAAST EEN TEKENING VAN DE GIC .

P0	P2H	P2L	P1	KONTR
OUT	OUT	OUT	OUT	80
OUT	OUT	INP	OUT	81
OUT	OUT	OUT	INP	82
OUT	OUT	INP	INP	83
OUT	INP	OUT	OUT	88
OUT	INP	INP	OUT	89
OUT	INP	OUT	INP	8A
OUT	INP	INP	INP	8B
INP	OUT	OUT	OUT	90
INP	OUT	INP	OUT	91
INP	OUT	OUT	INP	92
INP	OUT	INP	INP	93
INP	INP	OUT	OUT	98
INP	INP	INP	OUT	99
INP	INP	OUT	INP	9A
INP	INP	INP	INP	9B

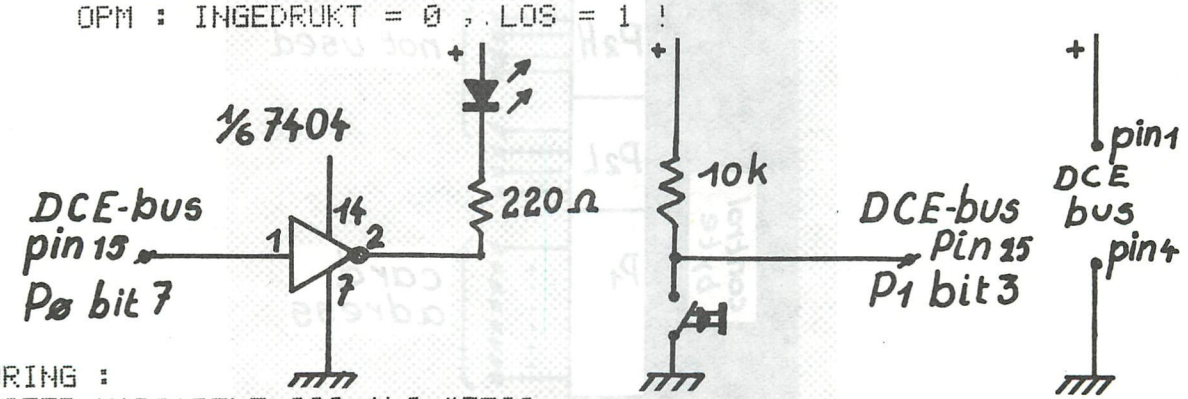


VOOR DE NUMMERS OP DE CONNECTOR VERWIJZEN WE NAAR DE PC MANUEL PAGINA 37 (PIN ON PC CARD). WIJZIG VOLGENDE FOUTEN :

```
P0B2 :  ? WORDT 12
P0B3 : 12 WORDT 10
+5V  :  3 WORDT  1
GND  :  ? WORDT  4
-5V  :  4 WORDT  3
```

B) VOORBEELD AANSLUITING EN BASIC BESTURING :

EENVOUDIG VOORBEELD MET 1 BIT OUTPUT EN 1 BIT INPUT.
OUTPUT : BIT 7 VAN POORT 0 DIE LED ON/OFF STUURT
BUFFERING MET 7404 TTL IC
INPUT : BIT 3 VAN POORT 1 DIE BESTUURD WORDT DOOR EEN DRUKKNOP
OPM : INGEDRUKT = 0 , LOS = 1 !



BESTURING :
DEFINIEER VARIABELE GIC ALS #FE00

```
GIC = #FE00
```

SET 8255 KONTROLEBYTE (P0 = OUT, P1 EN P2 = INP)

```
POKE GIC+3,#8B
```

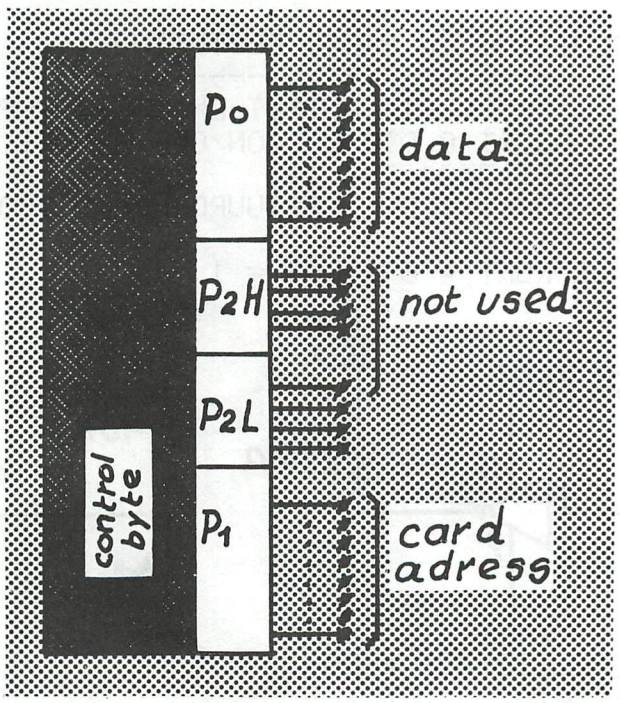
```
BESTUUR LED MET VARIABELE A (A=0:LED UIT, A=1:LED AAN) :
SHL 7 WORDT TOEGEVOEGD OM BIT 7 TE BEPALEN :
        POKE GIC,A SHL 7 : REM BIT 7
ZET VARIABELE B ZOALS STAND DRUKKNOP (LOS : B=0, INGEDRUKT :
B=1) : IXOR #FF OMDAT INPUT NEGATIEF WAAR IS  : SHR 3 EN
IAND 1 OM TE BEKOMEN DAT B ENKEL 0 OF 1 WORDT  :
        B = ( ( PEEK(GIC+1) IXOR #FF ) SHR 3 ) IAND 1
```

C) DE DCE BUS MODE

IN DE DCE BUS MODE WORDEN DE POORTEN VAN DE 8255 VOLGENS BEPAALDE AFSPRAKEN GEBRUIKT. HET DOEL IS EEN UNIVERSEEL SYSTEEM TE BEKOMEN WAARAAN VERSCHEIDENE IO KAARTEN TEGELIJK VERBONDEN ZIJN.

SIGNALEN :

DATA	: BIDIRECTIONEEL KANAAL VOOR DATA NAAR EN VAN KAART	
RD'	: PULS VOOR LEZEN DATA VAN KAART OP DATA	P2 BIT2
WR'	: PULS VOOR SCHRIJVEN VAN DATA NAAR KAART	P2 BIT1
BUS EXP	: BUS EXPAND : UITBREIDIING TOT 31 KAARTEN (TOT 15 KAARTEN : GEBRUIKT ALS BUS ON/OFF 1 = OFF, 0 = ON)	P2 BIT0
CARD ADRES	: SELEKTEERT 1 KAART + VERDERE REGISTER SELEKTIE OP KAART	



DE BASIC INSTRUKTIES OUT EN INP WORDEN GEBRUIKT OM DATA UIT TE WISSELEN MET DE KAARTEN. OP VOLGENDE PAGINA GEVEN WE EEN ASSEMBLER LISTING VAN DE ROUTINES DIE AANGEROEPEN WORDEN DOOR DEZE BASIC INSTRUKTIES. BESTUDERING VAN DEZE LISTING ZAL DUIDELIJK MAKEN HOE HET DCE BUS PROTOCOL VERLOOPT.

```
PAGE 01    BASIC OUT AND INP ROUTINES  30/08/80

002                    *THIS ASSEMBLER PROGRAM SHOWS THE ROM RESIDENT ROU-
003                    *TINES OUT AND INP TO DRIVE THE DCE-BUS
004                    *
005                    *STORE BUS-ADRES IN REGISTER D
006                    *FOR OUT : STORE DATA IN REGISTER E
007                    *FOR INP : ROUTINE RETURN WITH DATA IN REGISTER E
008                    *
009                    GIC     EQU   :FE00     ADRES OF GIC DATABUS
010                    RWMOP   EQU   :80       GIC MODE ALL PORTS OUTPUT
011                    RWMIP   EQU   :90       GIC MODE P0=INP,OTHERS=OUT
012                            ORG   :D8C8
013 D8C8 F5            OUT     PUSH  PSW
014 D8C9 E5                    PUSH  H
015 D8CA 2103FE                LXI   H,GIC+3   ADRES GIC CONTROL IN H,L
016 D8CD 3680                  MVI   M,RWMOP   SET GIC MODE
017 D8CF 2B                    DCX   H         ADRES GIC P2 IN H,L
018 D8D0 36FE                  MVI   M,:FE     CLEAR BUS EXPAND SIGNAL
019 D8D2 EB                    XCHG            IN L: DATA ; IN H: BUSADRES
020 D8D3 2200FE                SHLD  GIC       DATA IN P0 ; BUSADRES IN P1
021 D8D6 EB                    XCHG            ADRES GIC P2 IN H,L
022 D8D7 34                    INR   M         SET BUS EXPAND SIGNAL
023 D8D8 36FD                  MVI   M,:FD     SET WRITE STROBE TRUE   (#)
024 D8DA 36FF                  MVI   M,:FF     RESET STROBE
025 D8DC 35                    DCR   M         CLEAR BUS EXPAND SIGNAL
026 D8DD E1                    POP   H
027 D8DE F1                    POP   PSW
028 D8DF C9                    RET
029 D8E0 F5            BINP    PUSH  PSW
030 D8E1 E5                    PUSH  H
031 D8E2 2103FE                LXI   H,GIC+3   ADRES GIC CONTROL IN H,L
032 D8E5 3690                  MVI   M,RWMIP   SET GIC MODE
033 D8E7 2B                    DCX   H         ADRES GIC P2 IN H,L
034 D8E8 36FE                  MVI   M,:FE     CLEAR BUS EXPAND SIGNAL
035 D8EA 7A                    MOV   A,D       BUSADRES IN A
036 D8EB 3201FE                STA   GIC+1     STORE BUSADRES IN GIC P1
037 D8EE 34                    INR   M         SET BUS EXPAND SIGNAL
038 D8EF 36FB                  MVI   M,:FB     SET READ STROBE TRUE    (#)
039 D8F1 3A00FE                LDA   GIC       DATA TO A
040 D8F4 5F                    MOV   E,A       DATA IN E
041 D8F5 36FF                  MVI   M,:FF     RESET STROBE
042 D8F7 35                    DCR   M         CLEAR BUS EXPAND SIGNAL
043 D8F8 E1                    POP   H
044 D8F9 F1                    POP   PSW
045 D8FA C9                    RET
046                    * (#) = ON THIS MOMENT THE DATA EXCHANGE ON THE
047                    *       DCE-BUS TAKE PLACE

PAGE 02    BASIC OUT AND INP ROUTINES  30/08/80

048 D8FB                       END

****************************
* S Y M B O L   T A B L E *
****************************

BINP   D8E0   GIC    FE00   OUT    D8C8   RWMIP  0090
RWMOP  0080
```

V I E R O P E E N R I J

Een uitdaging voor strategen en tactici ...

Hierbij geven we U de listing van "vier op een rij": onze DAIversie van het bekende gezelschapsspel. Het programma is klaar om twee deelnemers tegen elkaar te laten wedijveren : wie het eerst vier op een rij (horizontaal of vertikaal) van zijn kleur kan plaatsen is de winnaar. We zouden geen computerfanaten zijn indien we U het volgende niet zouden voorstellen :

Wie schrijft het sterkste programma om de zetten te bepalen ? Wij verwachten van U een subroutine die de situatie kan overzien en die met goede zetten voor de dag komt ! We gaan alle ingezonden subroutines(op tape, geschreven of geprint) tegen mekaar laten optreden en we zien dan wel wie de sterkste is.

Plaats uw subroutine ergens op hogere lijnnummers, zodat we de subroutine kunnen mergen met het hoofdprogramma. Dus : in plaats van de input op lijn 11Ø doen we GOSUB ... de subroutine van speler A en op lijn 21Ø GOSUB ... routine van speler B. De winnaar mag de wisselbeker op zijn monitor plaatsen ... tot een sterker programma die komt afsnoepen.

444444444444444444444 OP EEN RIJ 444444444444444444444444444444444444444444

```
 5    CLEAR 1000
 6     MODE 0
 9     ENVELOPE 0 15,5;5,5;
10     REM vier op een rij
11     PRINT CHR$(12)
12     CURSOR 10,10:PRINT "V I E R  O P  E E N  R I J "
13     CURSOR 10,8:FOR X=1.0 TO 28.0:PRINT CHR$(29);:NEXT
14     CURSOR 35,0:PRINT "a program by DAInamic"
15     IF GETC=0.0 GOTO 15
20     DIM V(7.0,7.0)
25     FOR X=0.0 TO 7.0:FOR Y=0.0 TO 7.0:V(X,Y)=8.0:NEXT:NEXT
26     COLORG 0 8 3 14
27     MODE 4A
28     FILL 0,0 XMAX/2,YMAX 3:FILL XMAX/2,0 XMAX,YMAX 14
29     FOR X=XMAX/2.0 TO 10.0 STEP -1.0
30     DRAW X,0 X,YMAX 0:DRAW XMAX-X,0 XMAX-X,YMAX 0
31     NEXT
50     GOSUB 2000
```

```
 60      PRINT CHR$(12)
100      REM SPELER A
110      INPUT "SPELER A....WELKE RIJ ";R
111      IF R>7 GOTO 110
115      PRINT
116      FOR X=10.0 TO (R+1.0)*12.0+20.0:NOISE 0 15:DRAW X,100 X,105 3:NEXT
117      SOUND OFF
120      C=3.0:GOSUB 1000
130      GOSUB 2000
140      FILL 10,100 (R+1)*12+30,110 0
200      REM SPELER B
210      INPUT "SPELER B.....WELKE RIJ";R
211      IF R>7 GOTO 210
215      PRINT
217      FOR X=XMAX-10.0 TO (R+1.0)*12.0+25.0 STEP -1.0:NOISE 0 15:DRAW X,10
       0 X,105 14:NEXT
218      SOUND OFF
220      C=14.0:GOSUB 1000
230      GOSUB 2000
240      FILL XMAX-11,100 (R+1)*12+25,105 0
250      GOTO 100
1000     FOR X=1.0 TO 7.0:IF V(R,X)=8.0 THEN GOSUB 1500:V(R,X)=C:RETURN
1010     NEXT:PRINT "DE RIJ IS AL VOL !!":RETURN
1500     SOUND 1 0 15 0 FREQ(1000.0):SOUND 1 0 15 2 FREQ(100.0)
1505     FOR M=7.0 TO X+1.0 STEP -1.0
1510     FILL R*12+30,M*12 R*12+38,M*12+8 C
1520     FILL R*12+30,M*12 R*12+38,M*12+8 8
1525     NEXT
1530     FILL R*12+30,X*12 R*12+38,X*12+8 C
1535     SOUND OFF
1540     RETURN
2000     FOR X=1.0 TO 7.0:FOR Y=1.0 TO 7.0

 2010      FILL X*12+30,Y*12 X*12+8+30,Y*12+8 V(X,Y)
2020      NEXT:NEXT
2030      FOR X=1.0 TO 7.0:FOR Y=1.0 TO 4.0
2040      IF V(X,Y)<>8.0 THEN IF V(X,Y)=V(X,Y+1.0) AND V(X,Y)=V(X,Y+2.0) AND
        V(X,Y)=V(X,Y+3.0) THEN GAME=V(X,Y):GOTO 3000
2045      NEXT:NEXT
2049      FOR Y=1.0 TO 7.0:FOR X=1.0 TO 4.0
2050      IF V(X,Y)<>8.0 THEN IF V(X,Y)=V(X+1.0,Y) AND V(X,Y)=V(X+2.0,Y) AND
        V(X,Y)=V(X+3.0,Y) THEN GAME=V(X,Y):GOTO 3000
2055      NEXT:NEXT
2100      RETURN
3000      IF GAME=14.0 GOTO 3500
3010      FOR X=10.0 TO XMAX:DRAW X,0 X,YMAX 3:NEXT:GOTO 20
3500      FOR X=XMAX-10.0 TO 0.0 STEP -1.0:DRAW X,0 X,YMAX 14:NEXT:GOTO 20
```

test-soundmonitor-val/str

EEN TEST-TAPE VOOR DE CASSETTE-INTERFACE

ALS ER NOG PROBLEMEN ZIJN MET VOLUME EN TOON VOOR DE CASSETTE-INTERFACE KAN EEN TEST-TAPE ERG NUTTIG ZIJN.
WE MAKEN DEZE TAPE DOOR EEN 20-TAL HERKENNINGSSIGNALEN NA ELKAAR OP TE NEMEN. WE CLEAREN ONZE DAIpc DOOR UIT EN AAN TE ZETTEN OF DOOR "NEW" OF DOOR DE RESETKNOP IN TE DRUKKEN.
NU GAAN WE DIT BLANCO-PROGRAMMA 20 KEER SAVEN:

```
FOR X=1 TO 20:SAVE "TEST" :NEXT
```

DE RECORDER IN OPNAME ,TELKENS SPACE-BAR DRUKKEN EN ONZE TEST-TAPE IS KLAAR.
MET "CHECK" OF "LOAD" KUNNEN WE NU EENVOUDIG DE JUISTE INSTELLING VAN VOLUME EN TOON VASTSTELLEN.
WE KUNNEN OOK 20 KEER CALLM hexD2B8 DOEN,DAAR ZIT NAMELIJK DE ROUTINE DIE HET HERKENNINGSSIGNAAL PRODUCEERT.
(ZIE HANDBOEK P.136 02C5 WOPEN)

SOUND-MONITOR

BIJ DE MEESTE TV'S IS HET MOEILIJK OM GELUID EN BEELD VAN DAIpc GELIJKTIJDIG OPTIMAAL TE TUNEN. WE KRIJGEN EEN PRACHTIG GELUID ALS WE ONZE DAIpc AANSLUITEN AAN DE STEREOSET.
TWEE KLEINE VERSTERKERTJES GEBRUIKEN IS NATUURLIJK OOK EEN GOEDE OPLOSSING. IN HET JULI/AUG 80 NUMMER VAN ELEKTUUR WORDT EEN EENVOUDIGE CONSTRUCTIE BESCHREVEN. HET ONTWERP IS GEBASEERD OP IC LM386 ,EEN PRINTONTWERP (5cm X 5cm) IS BIJGEVOEGD.
ELEKTUUR JULI/AUG 80 P.8-32 Nr 100:UNIVERSELE LUIDSPREKER-UNIT.

VAL(xxxx)---STR$(xxxx)

VAL( ) EN STR$( ) ZIJN TWEE COMPLEMENTAIRE FUNCTIES.
TOCH KUNNEN ER RARE DINGEN GEBEUREN ALS WE ZE SAMEN GEBRUIKEN.
PROBEERT U MAAR:

```
A=1234 :A$=STR$(A) : PRINT VAL(A$)
```

WE KRIJGEN NIET DE OORSPRONKELIJKE WAARDE VAN A MAAR INVALID NUMBER !!
OORZAAK VAN DIT MISVERSTAND IS DE SPATIE VOOR HET GETAL ,
DEZE SPATIE IS MEE OPGENOMEN IN A$ EN DOET BASIC BESLUITEN DAT DE WAARDE VAN A$ ONMOGELIJK TE VINDEN IS.
HET HANDBOEK GEEFT OP P.101 EEN (ONLEESBARE)OPLOSSING VOOR DIT PROBLEEM:

```
10 INPUT A:PRINT
20 A$ = STR$(A)
30 A = VAL(RIGHT$(A$,LEN(A$)-1))
40 IF LEFT$(A$,1)="-" THEN A=A*(-1)
50 PRINT A: GOTO 10
```

DOOR DE FORMULERING OP LIJNNUMMER 30 KUNNEN WE DUS DE CONVERSIE-PROBLEMEN OMZEILEN.

RAKETSPEL 8K M1A H. BAKKER

```
1     MODE 0
2     COLORT 10 0 0 0
3     DIM A!(6.0)
10    PRINT CHR$(12):PRINT
20    PRINT "              ***** R A K E T S P E L *****":PRINT
30    PRINT "           === H. Bakker / DAI-Computer ===":PRINT :PRINT
40    PRINT " Probeer met 6 raketten zoveel mogelijk vliegtuigen te"
45    PRINT " raken. Lanceer de raket door het erbij behorende cijfer"
50    PRINT " in te typen, als er geen raket in de lucht is.":PRINT
51    PRINT " Het resultaat wordt onder in het beeld bijgehouden."
52    PRINT :PRINT " LET OP....De raket vliegt niet altijd even hard....!"
53    PRINT :PRINT " Let daarom op de te verwachten MACH-snelheid..!"
54    PRINT " Deze snelheid verschijnt rechtsonder in het beeld."
55    PRINT :PRINT "    (snelheid vliegtuig = MACH 0,5)"
57    PRINT CHR$(13):PRINT TAB(10);"Zullen we dan maar ?"
58    C!=GETC:IF C!=0.0 THEN 58
59    PRINT :PRINT TAB(10);"***** SUCCES *****!":WAIT TIME 40
60    E!=INT(RND(1.0)*3.0)+1.0
65    X!=0.0:FOR Q!=0.0 TO 6.0:A!(Q!)=0.0:NEXT
70    COLORT 0 14 0 0
80    PRINT CHR$(12):MODE 1A
90    D!=0.0:X!=0.0:F!=0.0
100   FOR A!=10.0 TO 60.0 STEP 10.0
110   DRAW A!,2 A!,5 15:DRAW A!-1,1 A!+1,1 10
120   PRINT TAB(A!-A!/4+1);A!/10.0;:NEXT A!
130   DRAW 1,0 70,0 6:PRINT
140   PRINT "  Aantal afgeschoten raketten:";F!
```

RAKETSPEL part 2

```
150   PRINT "  Aantal geraakte vliegtuigen:";D!
155   CURSOR 40,2:PRINT "Raket snelheid:"
160   CURSOR 40,1:PRINT "MACH.";E!
170   GOSUB 1000
200   C!=GETC:IF C!=0.0 THEN WAIT TIME 14:GOSUB 1000
210   IF C!=0.0 THEN 200
211   IF C!>54.0 OR C!<49.0 THEN 200
220   C1!=10.0*(C!-48.0):Q!=C1!/10.0
222   IF A!(Q!)<>0.0 THEN 200
224   A!(Q!)=1.0
230   F!=F!+1.0
240   B!=2.0
300   FOR I!=1.0 TO E!*2.0:B!=B!+1.0
310   DRAW C1!,B!-2 C1!,B!+3 15:DRAW C1!-1,B!-1 C1!+1,B!-1 15
320   DRAW C1!-1,B!-2 C1!+1,B!-2 0:IF B!<40 THEN 342
330   IF B!>40.0 AND B!<50.0 AND X!<C1! AND X!+6>C1! THEN 500
342   DOT C1!,B!-1 0
350   NEXT I!:GOSUB 1000
360   IF B!>54 THEN 700
370   GOTO 300
500   P!=5.0:FOR T!=0.0 TO 2.0:FILL X!+P!,55 X!,40 15
510   FILL X!+P!,55 X!,35 0:P!=P!+3.0:NEXT T!
520   X!=0.0:D!=D!+1.0:GOTO 700
```

RAKETSPEL part 3

```
700   E!=(INT(RND(1.0)*3.0))+1.0:CURSOR 30,2:PRINT F!
720   CURSOR 30,1:PRINT D!
725   CURSOR 45,1:PRINT E!
730   IF F!=6.0 THEN INPUT "*** NOG EEN KEER *** (J/N)";D$:GOTO 770
740   GOSUB 1000
760   GOTO 200
770   IF D$="J" THEN 60:PRINT "    ";:END
1000  X!=X!+1.0
1010  DRAW X!+2,45 X!+6,45 15
1020  DRAW X!+2,46 X!+5,46 15:DRAW X!+2,44 X!+5,44 15
1040  DRAW X!+3,47 X!+4,47 15:DRAW X!+3,43 X!+4,43 15
1060  DOT X!+3,44 0:DOT X!+3,46 0:DOT X!+3,47 0:DOT X!+3,43 0
1080  DRAW X!+1,43 X!+1,47 0
1090  IF X!=64.0 THEN X!=0.0:E!=INT(RND(1.0)*3.0)+1.0:FILL 70,48 62,43 0:CURSO
45,1:PRINT E!
1100  RETURN
```

GRAFIEK 8K M. VERMEULEN

```
10    MODE 3A
20    FILL 0,0 XMAX,YMAX 10
30    FOR A=0.0 TO 80.0 STEP 3.0
40    DRAW 80,0 A,100 0
50    DRAW 0,50 XMAX,100-A 0
60    DRAW 80,100 A,0 0
70    DRAW XMAX,50 0,A 0
80    NEXT
90    END
```

videoram mode o

EEN OVERZICHT VAN DE VIDEORAM IN MODE Ø (CHARACTER MODE)

Deze tabel geeft respectievelijk lijnnummer, color code byte van het eerste character van een lijn, informatiebyte van het eerste character en ook deze locaties voor het laatste character van een lijn. U kan met peek en pook deze organisatie verkennen, zolang de controlebytes van de lijnen onveranderd blijven, is alles o.k. (De lijn-controle bytes zitten nog voor de informatie voor het eerste character : voor een 48 K is de eerste controlebyte BFEF, voor een 32 K 7FEF. De color byte van de eerste lijn zit respectievelijk op BFEE en 7FEE).

Volgend programma drukt voor U deze control en colorbytes :

```
1Ø FOR A=Ø TO 23:PRINT A+1,:PRINT HEX$(#FEF -#86*A),
2Ø PRINT HEX$(#FEE - #86*A):NEXT
```

#BFEF voor 48 K ,#7FEF voor 32 K ,#1FEF voor 8 K

LINE NUMBER	# LOCATION COLOR CODE BEGIN LINE	# LOCATION BEGIN LINE	# LOCATION COLOR CODE END LINE	# LOCATION END LINE
23.0	# BFEA	# BFED	# BF6A	# BF6D
22.0	# BF64	# BF67	# BEE4	# BEE7
21.0	# BEDE	# BEE1	# BE5E	# BE61
20.0	# BE58	# BE5B	# BDD8	# BDDB
19.0	# BDD2	# BDD5	# BD52	# BD55
18.0	# BD4C	# BD4F	# BCCC	# BCCF
17.0	# BCC6	# BCC9	# BC46	# BC49
16.0	# BC40	# BC43	# BBC0	# BBC3
15.0	# BBBA	# BBBD	# BB3A	# BB3D
14.0	# BB34	# BB37	# BAB4	# BAB7
13.0	# BAAE	# BAB1	# BA2E	# BA31
12.0	# BA28	# BA2B	# B9A8	# B9AB

LINE NUMBER	# LOCATION COLOR CODE BEGIN LINE	# LOCATION BEGIN LINE	# LOCATION COLOR CODE END LINE	# LOCATION END LINE
11.0	# B9A2	# B9A5	# B922	# B925
10.0	# B91C	# B91F	# B89C	# B89F
9.0	# B896	# B899	# B816	# B819
8.0	# B810	# B813	# B790	# B793
7.0	# B78A	# B78D	# B70A	# B70D
6.0	# B704	# B707	# B684	# B687
5.0	# B67E	# B681	# B5FE	# B601
4.0	# B5F8	# B5FB	# B578	# B57B
3.0	# B572	# B575	# B4F2	# B4F5
2.0	# B4EC	# B4EF	# B46C	# B46F
1.0	# B466	# B469	# B3E6	# B3E9
0.0	# B3E0	# B3E3	# B360	# B363

PROPELLER 32K MODE 6A DAI

```
10    CLEAR 1800: COLORG 0 0 0 0:MODE 6A
15    PRINT "BUSY CALCULATING"
20    HX!=XMAX/2.0:HY!=YMAX/2.0
30    S!=SQR(0.4*YMAX)
31    GOSUB 1000
35    QQ=0
43    FOR R=0 TO 251 STEP 6
45    Z!=ZZ!(R):QQ=QQ+2
48    FOR AW=0 TO 8:W=AW*112:COLR=(AW MOD 3)+21:PP=QQ+W:GOSUB 2000:PP=1176-QQ+W
GOSUB 2000:NEXT:NEXT
49    MODE 6
50    FOR TT=30 TO 1 STEP -1
70    COLORG 0 0 0 15:WAIT TIME TT:COLORG 0 0 15 0:WAIT TIME TT:COLORG 0 15 0 0
WAIT TIME TT
75    IF TT=6 GOTO 70
80    NEXT:STOP
1000  DIM ZZ!(253.0)
1010  ST!=PI/504.0:STP!=-ST!
1020  FOR QQ=0 TO 252:STP!=STP!+ST!
1030  ZZ!(QQ)=S!*SIN(STP!)
1040  NEXT
1050  RETURN
2000  SS=(PP+1008) MOD 252
2005  XX=(PP/252) MOD 4+1
2010  ON XX GOTO 2020,2030,2040,2050
2020  AA!=ZZ!(252.0-SS):BB!=ZZ!(SS):GOTO 2060
2030  AA!=-ZZ!(SS):BB!=ZZ!(252.0-SS):GOTO 2060
2040  AA!=-ZZ!(252.0-SS):BB!=-ZZ!(SS):GOTO 2060
2050  AA!=ZZ!(SS):BB!=-ZZ!(252.0-SS)
2060  DOT Z!*AA!+HX!,Z!*BB!+HY! COLR
2070  RETURN
```

4 color demo

4 COLOR DEMO (LISSAJOUS)

Dit programma illustreert duidelijk de mogelijkheden in 4-kleuren modes (2,4,6).Op lijnnummers 16-50 doen we een aardigheidje om het tekenen vlugger te laten verlopen : we berekenen eerst de coördinaten en stoppen ze in arrays A en B.
Op lijnnummers 100-120 voeren we de tekening uit.
Op lijnnummers 300-345 goochelen we een minuutje met de kleurenregisters van COLORG en van 400-430 laten we het lot(rnd) de effecten bepalen.
8K en 12K toestellen kunnen dit programma lopen in MODE 4 mits volgende aanpassingen:

```
5  CLEAR 2100
10 MODE 4
40 A(N)=XMAX/2 + 75*COS(X):B(N)=YMAX/2 + 50*SIN(X)
110 DRAW XMAX/22,YMAX A(X),B(X) 0
```

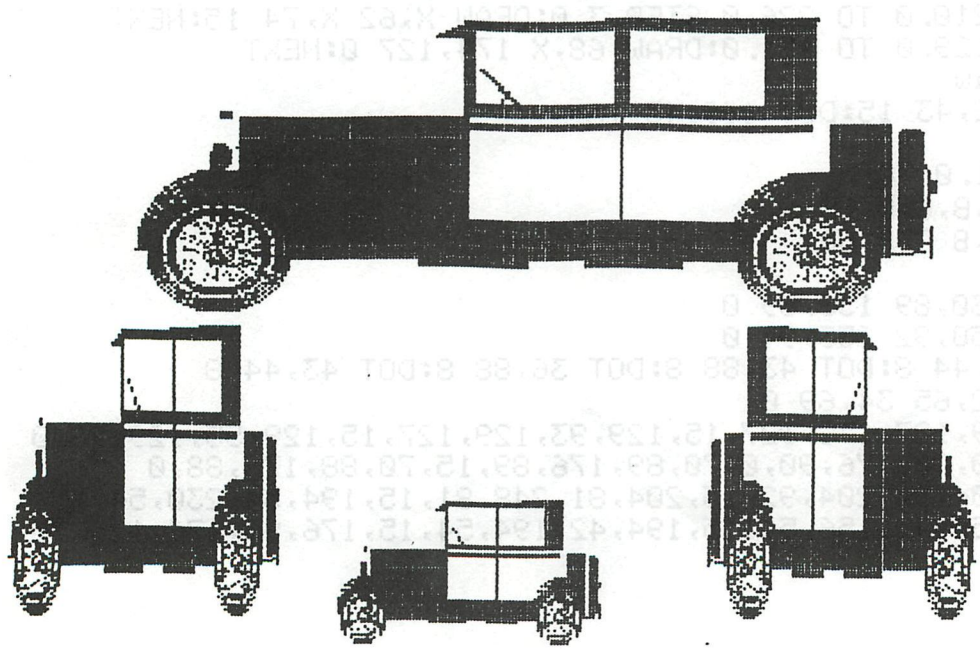
```
5       CLEAR 5000
10      MODE 6
16      DIM A(250.0),B(250.0)
20      COLORG 8 0 15 3
30      FOR X=0.0 TO 2.0*PI STEP 3E-2
40      A(N)=XMAX/2.0+100.0*COS(X):B(N)=YMAX/2.0+100.0*SIN(X*2.0)
45      N=N+1.0
50      NEXT
100     FOR X=0.0 TO 209.0
110     DRAW 150,125 A(X),B(X) 0
115     DRAW 0,0 A(X),B(X) 3
116     DRAW A(X),B(X) XMAX,0 15
120     NEXT
300     FOR X=0.0 TO 50.0
320     COLORG 0 A 0 0
330     WAIT TIME 15
335     COLORG 0 0 A 0
337     WAIT TIME 15
338     COLORG 0 0 0 A
339     WAIT TIME 15
340     A=A+1.0:IF A=16.0 THEN A=1.0
345     NEXT X
400     FOR X=0.0 TO 50.0
410     COLORG RND(15.0) RND(15.0) RND(15.0) RND(15.0)
420     WAIT TIME 20
430     NEXT X
```

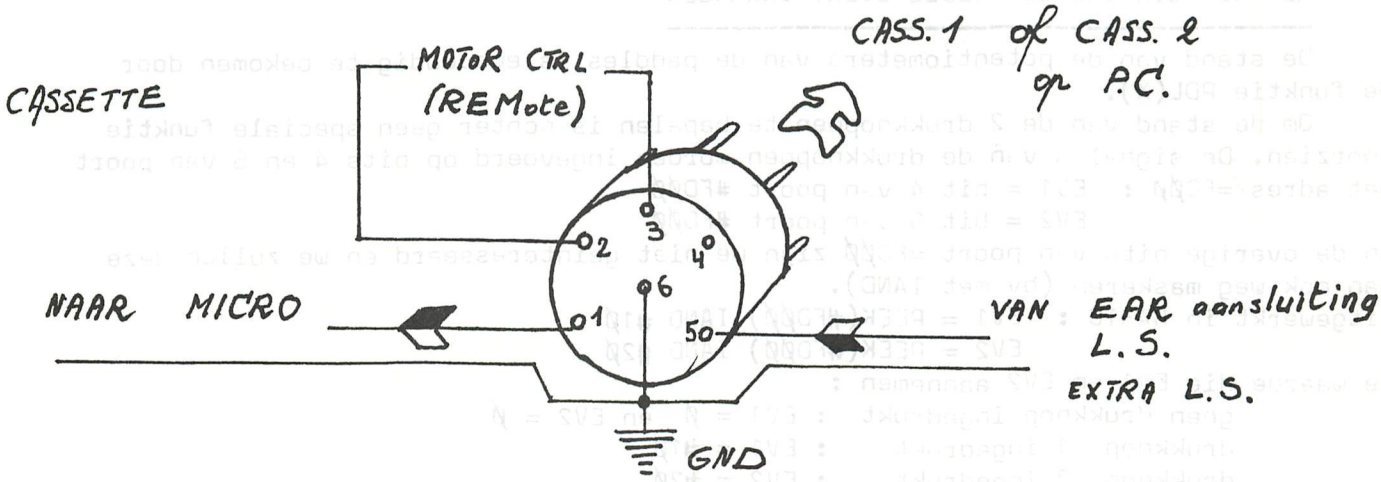
```
  5  CLEAR 4000:DIM A(255.0),B(255.0)
 10      COLORG 8 0 14 15
 15      FOR X=0.0 TO 2.0*PI STEP 2.5E-2
 16      A(N)=COS(X):B(N)=SIN(X)
 17      N=N+1.0:NEXT
 18      PRINT N
 20      MODE 6
100      REM FILL
110      FOR X=1.0 TO 8.0
120      READ A,B,C,D,CO
130      FILL A,B C,D CO
140      NEXT
150      DATA 69,92,178,128,0,85,98,126,124,8,132,98,174,124,8,69,40,248,92,
    0
160      DATA 69,55,176,92,14,248,70,254,92,14,46,48,64,90,0,36,44,43,88,0
200      REM wielen
210      FOR X=0.0 TO 128.0
220      DRAW 71,43 71+25*A(X),43+30*B(X) 15
225      DRAW 254,43 254+25*A(X),43+30*B(X) 15
230      NEXT
250      FOR X=0.0 TO 255.0
260      DRAW 71,43 71+22*A(X),43+22*B(X) 0
270      DRAW 254,43 254+22*A(X),43+22*B(X) 0
272      DOT 71+17*A(X),43+17*B(X) 8:DOT 254+17*A(X),43+17*B(X) 8
273      DRAW 71,43 71+13*A(X),43+13*B(X) 14:DRAW 254,43 254+13*A(X),43+13*B
    (X) 14
274      DRAW 71,43 71+5*A(X),43+5*B(X) 15:DRAW 254,43 254+5*A(X),43+5*B(X)
    15
280      NEXT
281      FOR X=64.0 TO 192.0:DRAW 256,78 256+6*A(X),78+5*B(X) 0:NEXT
285      FILL 110,37 140,39 0:FILL 158,37 192,39 0
290      FOR X=122.0 TO 127.0:DRAW 178,X 187,122 0:NEXT
292      FOR X=210.0 TO 226.0 STEP 3.0:DRAW X,62 X,74 15:NEXT
293      FOR X=129.0 TO 131.0:DRAW 68,X 174,127 0:NEXT
300      REM draw
305      DOT 162,43 15:DOT 189,43 15
310      N=24.0
320      FOR X=1.0 TO N
330      READ A,B,C,D,CO
340      DRAW A,B C,D CO
350      NEXT
360      DRAW 130,89 138,89 0
365      FILL 250,92 253,94 0
370      DOT 36,44 8:DOT 43,88 8:DOT 36,88 8:DOT 43,44 8
375      FILL 32,65 34,69 0
400      DATA 69,127,178,127,15,129,93,129,127,15,129,55,129,93,0
410      DATA 70,90,176,90,0,70,89,176,89,15,70,88,176,88,0
420      DATA 204,54,204,92,15,204,81,248,81,15,194,54,230,54,15
430      DATA 156,42,156,54,15,194,42,194,54,15,176,96,176,124,15
```

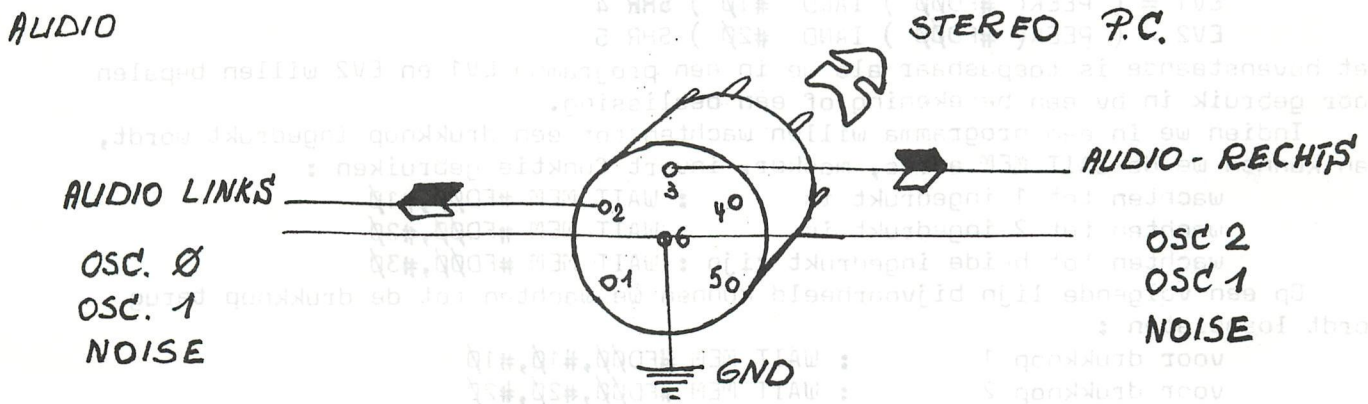
Citroën B 14 Coach - 1927

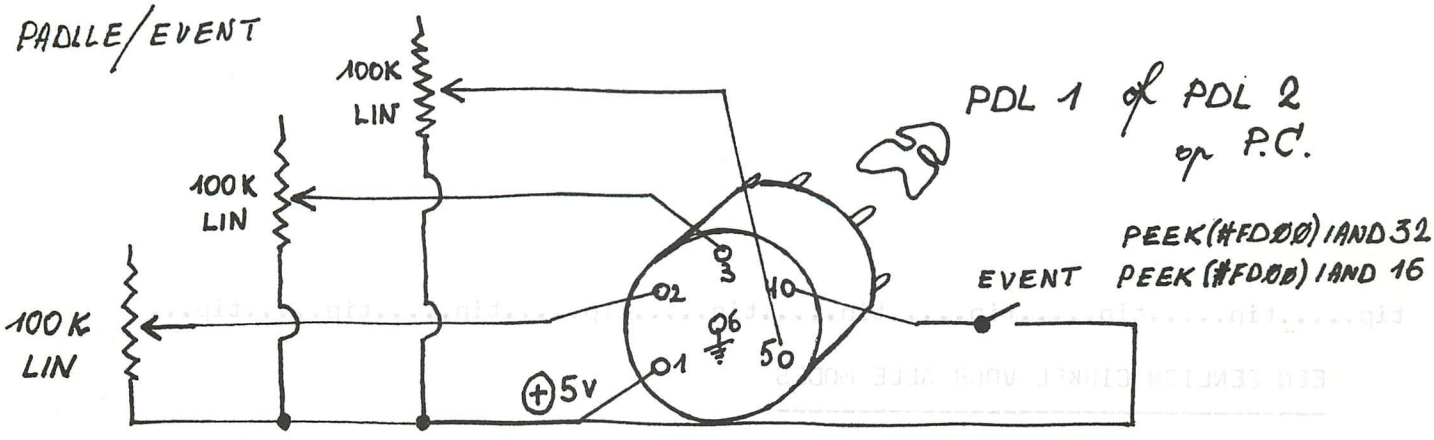
citroen

```
 440      DATA 36,77,34,72,0,36,54,34,61,0,34,72,34,61,0
 450      DATA 160,97,172,110,0,161,97,173,110,0,164,102,167,97,0
 460      DATA 35,43,49,41,0,46,87,49,90,15,61,90,64,87,15
 470      DATA 33,46,36,46,0,33,41,33,52,0,252,94,252,97,14
 472      GOTO 500:REM DELETE THIS LINE FOR HARDCOPY
 475      DIM CC(15.0):CC(8.0)=32.0:CC(0.0)=127.0:CC(14.0)=48.0:CC(15.0)=46.0

 479      MODE 6A
 480      FOR Y=15.0 TO 140.0:FOR X=25.0 TO 104.0:PRINT CHR$(CC(SCRN(X,Y)));:
NEXT:PRINT :NEXT
 481      FOR X=0.0 TO 10.0:PRINT :NEXT
 482      FOR Y=15.0 TO 140.0:FOR X=105.0 TO 184.0:PRINT CHR$(CC(SCRN(X,Y)));
:NEXT:PRINT :NEXT
 483      FOR X=0.0 TO 10.0:PRINT :NEXT
 484      FOR Y=15.0 TO 140.0:FOR X=185.0 TO 264.0:PRINT CHR$(CC(SCRN(X,Y)));
:NEXT:PRINT :NEXT
 485      FOR X=0.0 TO 10.0:PRINT :NEXT
 486      FOR Y=15.0 TO 140.0:FOR X=265.0 TO 335.0:PRINT CHR$(CC(SCRN(X,Y)));
:NEXT:PRINT :NEXT
 500      FOR Y=20.0 TO 130.0 STEP 2.0:FOR X=30.0 TO 280.0 STEP 3.0:A=SCRN(X,
Y):DOT XMAX-X,115+Y A:NEXT:NEXT
 600      FILL 0,0 280,133 8
 610      FOR Y=135.0 TO 245.0 STEP 2.0:FOR X=30.0 TO 280.0 STEP 3.0:A=SCRN(X
MAX-X,Y):DOT (XMAX-X)/3,Y-115 A:NEXT:NEXT
 611      FOR Y=135.0 TO 245.0 STEP 2.0:FOR X=30.0 TO 280.0 STEP 3.0:A=SCRN(X
MAX-X,Y):DOT (XMAX-X)/3+100,(Y-115)/2 A:NEXT:NEXT
 615      FOR Y=135.0 TO 245.0 STEP 2.0:FOR X=30.0 TO 280.0 STEP 3.0:A=SCRN(X
MAX-X,Y):DOT XMAX-(XMAX-X)/3,Y-115 A:NEXT:NEXT
 620      FOR Y=135.0 TO 245.0 STEP 2.0:FOR X=30.0 TO 280.0 STEP 3.0:A=SCRN(X
MAX-X,Y):DOT XMAX-((XMAX-X)/3+100),160-((Y-115)/2) A:NEXT:NEXT
1000      GOTO 1000
```




CASS. 1 of CASS. 2
of P.C.
CASSETTE
MOTOR CTRL
(REMote)
NAAR MICRO
VAN EAR aansluiting
L.S.
EXTRA L.S.
GND


AUDIO
STEREO P.C.
AUDIO LINKS
AUDIO - RECHTS
OSC. Ø
OSC. 1
NOISE
OSC 2
OSC 1
NOISE
GND


PADLE/EVENT
100K
LIN
100K
LIN
100K
LIN
PDL 1 of PDL 2
of P.C.
PEEK(#FDØØ) AND 32
EVENT
PEEK(#FDØØ) AND 16
+5V

HET GEBRUIK VAN DE PADDLE-EVENT INGANGEN

De stand van de potentiometers van de paddles is eenvoudig te bekomen door de funktie PDL(X).

Om de stand van de 2 drukknoppen te bepalen is echter geen speciale funktie voorzien. De signalen van de drukknoppen worden ingevoerd op bits 4 en 5 van poort met adres =FDØØ : EV1 = bit 4 van poort #FDØØ
EV2 = bit 5 van poort #FDØØ
In de overige bits van poort =FDØØ zijn we niet geinteresseerd en we zullen deze dan ook weg maskeren (bv met IAND).
Uitgewerkt in BASIC : EV1 = PEEK(#FDØØ) IAND #1Ø
EV2 = PEEK(#FDØØ) IAND #2Ø
De waarde die EV1 en EV2 aannemen :
geen drukknop ingedrukt : EV1 = Ø en EV2 = Ø
drukknop 1 ingedrukt : EV1 = #1Ø
drukknop 2 ingedrukt : EV2 = #2Ø
Vermits =1Ø en =2Ø niet zo praktisch is voor verdere berekeningen kunnen we een shift right operator toevoegen, zodat EV1 en EV2 enkel Ø of 1 kunnen worden.

```
EV1 = ( PEEK( #FDØØ ) IAND  #1Ø ) SHR 4
EV2 = ( PEEK( #FDØØ ) IAND  #2Ø ) SHR 5
```

Het bovenstaande is toepasbaar als we in een programma EV1 en EV2 willen bepalen voor gebruik in bv een berekening of een beslissing.

Indien we in een programma willen wachten tot een drukknop ingedrukt wordt, dan kunnen we de WAIT MEM adres, masker, invert funktie gebruiken :

```
wachten tot 1 ingedrukt is       : WAIT MEM #FDØØ,#1Ø
wachten tot 2 ingedrukt is       : WAIT MEM #FDØØ,#2Ø
wachten tot beide ingedrukt zijn : WAIT MEM #FDØØ,#3Ø
```

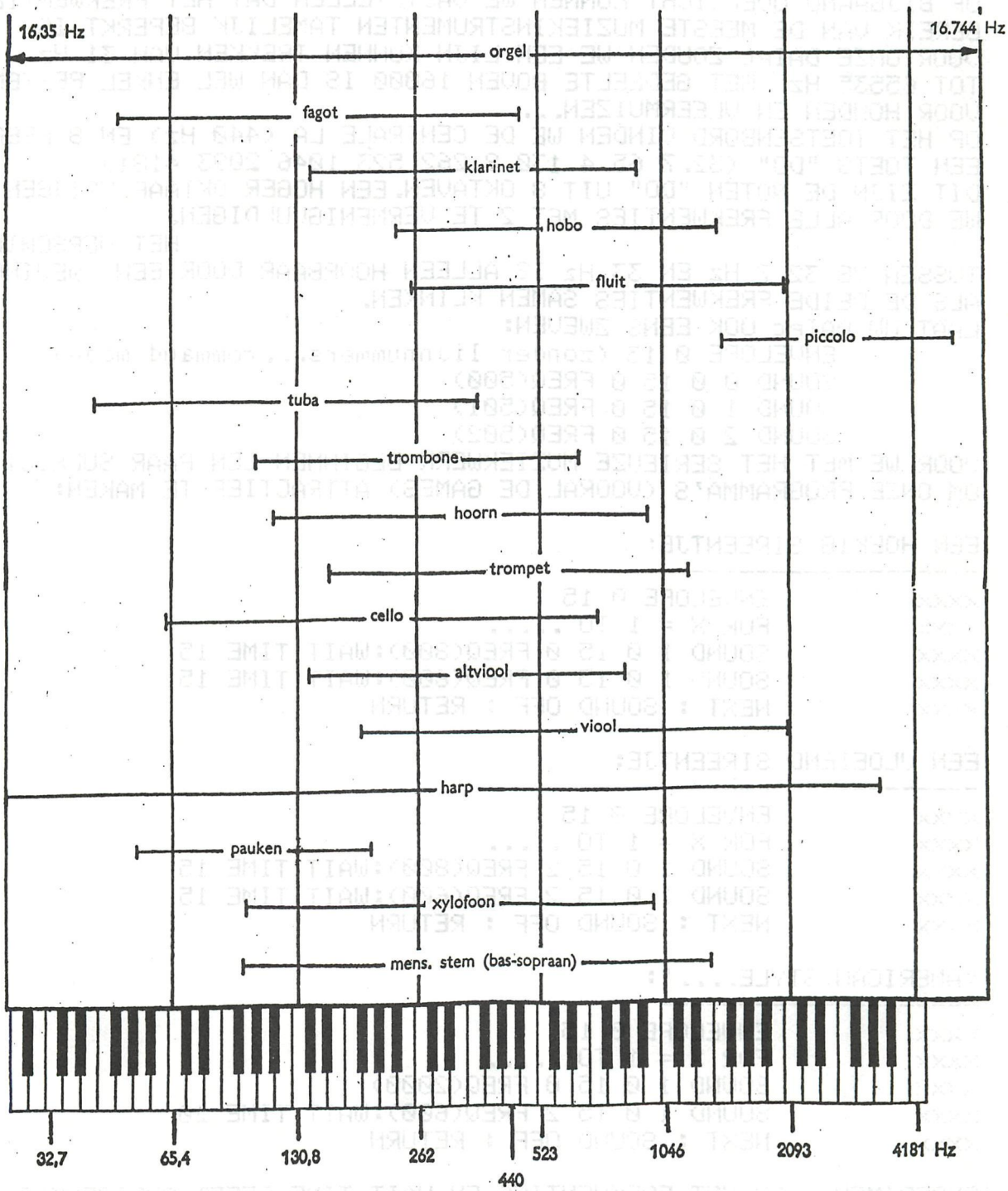
Op een volgende lijn bijvoorbeeld kunnen we wachten tot de drukknop terug wordt losgelaten :

```
voor drukknop 1         : WAIT MEM #FDØØ,#1Ø,#1Ø
voor drukknop 2         : WAIT MEM #FDØØ,#2Ø,#2Ø
voor beide drukknoppen : WAIT MEM #FDØØ,#3Ø,#3Ø
```

tip.....tip.....tip.....tip.....tip.....tip.....tip.....tip.....tip.....tip...

EEN EENLIJN CIRKEL VOOR ALLE MODES

```
FOR X=Ø TO 2*PI STEP Ø.Ø1 : DOT XMAX/2 +YMAX/2 * SIN(X) , YMAX/2 + YMAX/2 *
COS(X) COLOR : NEXT X
```



FREKWENTIEBEREIK VAN VERSCHILLENDE MUZIEKINSTRUMENTEN
uit ELECTRONISCHE ORGELS door WIM VAN BUSSEL Prisma 1540

OP BIJGAAND OVERZICHT KUNNEN WE VASTSTELLEN DAT HET FREKWENTIE-
BEREIK VAN DE MEESTE MUZIEKINSTRUMENTEN TAMELIJK BEPERKT IS.
VOOR ONZE DAIpc ZOUDEN WE EEN LIJN KUNNEN TREKKEN VAN 31 Hz
TOT 65535 Hz. HET GEDEELTE BOVEN 16000 IS DAN WEL ENKEL BESTEMD
VOOR HONDEN EN VLEERMUIZEN....
OP HET TOETSENBORD VINDEN WE DE CENTRALE LA (440 Hz) EN 8 KEER
EEN TOETS "DO" (32.7 65.4 130.8 262 523 1046 2093 4181)
DIT ZIJN DE NOTEN "DO" UIT 8 OKTAVEN. EEN HOGER OKTAAF KRIJGEN
WE DOOR ALLE FREKWENTIES MET 2 TE VERMENIGVULDIGEN.
HET VERSCHIL
TUSSEN VB 32.7 Hz EN 33 Hz IS ALLEEN HOORBAAR DOOR EEN ZWEVING
ALS DE BEIDE FREKWENTIES SAMEN KLINKEN.
LAAT UW DAIpc OOK EENS ZWEVEN:

```
ENVELOPE 0 15 (zonder lijnnummers... command mode)
SOUND 0 0 15 0 FREQ(500)
SOUND 1 0 15 0 FREQ(501)
SOUND 2 0 15 0 FREQ(502)
```

VOOR WE MET HET SERIEUZE MUZIEKWERK BEGINNEN EEN PAAR SUBROUTINES
OM ONZE PROGRAMMA'S (VOORAL DE GAMES) ATTRACTIEF TE MAKEN:

EEN HOEKIG SIREENTJE:

```
xxxx        ENVELOPE 0 15
xxxx        FOR X = 1 TO .....
xxxx        SOUND 1 0 15 0 FREQ(800):WAIT TIME 15
xxxx        SOUND 1 0 15 0 FREQ(600):WAIT TIME 15
xxxx        NEXT : SOUND OFF : RETURN
```

EEN VLOEIEND SIREENTJE:

```
xxxx        ENVELOPE 0 15
xxxx        FOR X = 1 TO .....
xxxx        SOUND 1 0 15 2 FREQ(800):WAIT TIME 15
xxxx        SOUND 1 0 15 2 FREQ(600):WAIT TIME 15
xxxx        NEXT : SOUND OFF : RETURN
```

"AMERICAN STYLE....":

```
xxxx        ENVELOPE 0 15
xxxx        FOR X = 1 TO .....
xxxx        SOUND 1 0 15 0 FREQ(2000)
xxxx        SOUND 1 0 15 2 FREQ(600):WAIT TIME 20
xxxx        NEXT : SOUND OFF : RETURN
```

EXPERIMENTEREN MET FREKWENTIES EN WAIT TIME GEEFT ONBEPERKTE
MOGELJKHEDEN, BEZORG ONS UW BESTE RESULTATEN VOOR PUBLICATIE..
VOLGENDE KEER BESTUDEREN WE EEN EENVOUDIGE MUZIEKPARTITUUR EN

GAAN WE HET MUZIEKSTUKJE PROGRAMMEREN.....

BLOXY COLORG-DEMO 8...32K MODE 2...6

```
100   MODE 4:COLORG 0 0 0 0:FOR X=5 TO XMAX-10 STEP 12:FOR Y=5 TO YMAX-10 STEP
2:FOR Z=0 TO 2

230   C=INT(RND(4.0)+20.0:DRAW X+Z*2,Y+Z*2 X+10-Z*2,Y+Z*2 C

240   DRAW X+Z*2,Y+Z*2 X+Z*2,Y+10-Z*2 C

260   DRAW X+Z*2,Y+10-Z*2 X+10-Z*2,Y+10-Z*2 C

265   DRAW X+10-Z*2,Y+Z*2 X+10-Z*2,Y+10-Z*2 C

270   NEXT:NEXT:NEXT

275   FOR X!=1.0 TO 50.0

280   COLORG 0 RND(16.0) RND(16.0) RND(16.0):WAIT TIME 25:NEXT

300   FOR Y!=1.0 TO 50.0

310   C!=RND(10.0)+6.0

320   COLORG 0 C! 0 0:WAIT TIME 10

330   COLORG 0 0 C! 0:WAIT TIME 10

340   COLORG 0 0 0 C!:WAIT TIME 10:NEXT

900   FOR X!=1.0 TO 30.0

1000  COLORG 0 1 9 12:WAIT TIME 10

1010  COLORG 0 5 13 15:WAIT TIME 10

1020  COLORG 0 3 4 11:WAIT TIME 10

1030  NEXT:GOTO 275
```

DE TEKENOPDRACHTEN WORDEN BEREKEND IN FUNCTIE VAN XMAX EN YMAX,
ZODAT HET PROGRAMMA GESCHIKT IS VOOR ALLE 4 KLEUREN-MODES.
VOOR 8K TOESTELLEN VERVANGEN WE OP LIJNUMMER 100"MODE 4" DOOR "MODE 2".
32K TOESTELLEN KUNNEN HET PROGRAMMA GEBRUIKEN IN HOOGSTE RESOLUTIE
DOOR "MODE 4" TE VERVANGEN DOOR "MODE 6".
DIT PROGRAMMA MAAKT GEBRUIK VAN DE KLEUREN 20-23.
OP LIJNNUMMER 230 WORDT RANDOM EEN KLEUR 20...23 GEKOZEN.
20 KIEST DE KLEUR VAN REGISTER 1,21 VAN REGISTER 2,22 VAN REGISTER 3
EN 23 VAN REGISTER 4.DAAR DE KLEURREGISTERS OORSPRONKELIJK OP
0 STAAN ZIEN WE DE OPBOUW VAN DE TEKENING NIET.

8080 instructionset

INSTRUCTION	FUNCTION		HEX
MOVE GROUP		reg	A B C D E H L M
MOV A, reg	(A)←(reg)		7F 78 79 7A 7B 7C 7D 7E
MOV B, reg	(B)←(reg)		47 40 41 42 43 44 45 46
MOV C, reg	(C)←(reg)		4F 48 49 4A 4B 4C 4D 4E
MOV D, reg	(D)←(reg)		57 50 51 52 53 54 55 56
MOV E, reg	(E)←(reg)		5F 58 59 5A 5B 5C 5D 5E
MOV H, reg	(H)←(reg)		67 60 61 62 63 64 65 66
MOV L, reg	(L)←(reg)		6F 68 69 6A 6B 6C 6D 6E
MOV M, reg	(M)←(reg)		77 70 71 72 73 74 75 --
ACCUMULATOR GROUP			
ADD reg	(A)←(A)+(reg)	*	87 80 81 82 83 84 85 86
ADC reg	(A)←(A)+(reg)+(CY)	*	8F 88 89 8A 8B 8C 8D 8E
SUB reg	(A)←(A)-(reg)	*	97 90 91 92 93 94 95 96
SBB reg	(A)←(A)-(reg)-(CY)	*	9F 98 98 9A 9B 9C 9D 9E
ANA reg	(A)←(A)∧(reg)	*	A7 A0 A1 A2 A3 A4 A5 A6
XRA reg	(A)←(A)∀(reg)	*	AF A8 A9 AA AB AC AD AE
ORA reg	(A)←(A)∨(reg)	*	B7 B0 B1 B2 B3 B4 B5 B6
CMP reg	(A)-(reg)	*	BF B8 B9 BA BB BC BD BE
INCREMENT/DECREMENT REGISTER			
INR reg	(reg)←(reg)+1	**	3C 04 0C 14 1C 24 2C 34
DCR reg	(reg)←(reg)-1	**	3D 05 0D 15 1D 25 2D 35
REGISTER PAIR GROUP		rp	B D H SP PSW
INX rp	(rp)←(rp)+1		03 13 23 33 --
DCX rp	(rp)←(rp)-1		0B 1B 2B 3B --
LDAX rp	(A)←((rp))		0A 1A -- -- --
STAX rp	((rp))←(A)		02 12 -- -- --
DAD rp	(H,L)←(H,L)+(rp) ***		09 19 29 39 --
PUSH rp	((SP)-1)←(rh),((SP)-2)←(rl), (SP)←(SP)-2		C5 D5 E5 -- F5
POP rp	(rl)←((SP)),(rh)←((SP)+1), (SP)←(SP)+2		C1 D1 E1 -- F1 *
DIRECT ADDRESS GROUP			
LDA addr	(A)←(addr)		3A al ah
STA addr	(addr)←(A)		32 al ah
LHLD addr	(L)←(addr),(H)←(addr+1)		2A al ah
SHLD addr	(addr)←(L),(addr+1)←(H)		22 al ah
IMMEDIATE GROUP			
MVI A, data	(A)←data		3E dd
MVI B, data	(B)←data		06 dd
MVI C, data	(C)←data		0E dd
MVI D, data	(D)←data		16 dd
MVI E, data	(E)←data		1E dd
MVI H, data	(H)←data		26 dd
MVI L, data	(L)←data		2E dd
MVI M, data	(M)←data		36 dd
ADI data	(A)←(A)+data	*	C6 dd
ACI data	(A)←(A)+data+(CY)	*	CE dd
SUI data	(A)←(A)-data	*	D6 dd
SBI data	(A)←(A)-data-(CY)	*	DE dd
ANI data	(A)←(A)∧data	*	E6 dd
XRI data	(A)←(A)∀data	*	EE dd
ORI data	(A)←(A)∨data	*	F6 dd
CPI data	(A)-data	*	FE dd
LXI B, addr	(B)←ah,(C)←al		01 al ah
LXI D, addr	(D)←ah,(E)←al		11 al ah
LXI H, addr	(H)←ah,(L)←al		21 al ah
LXI SP,addr	(SP_h)←ah,(SP_l)←al		31 al ah

INSTRUCTION	FUNCTION		HEX
JUMP GROUP			
JMP addr	(PC)←addr		C3 al ah
JNZ addr	If Z=0, (PC)←addr		C2 al ah
JZ addr	If Z=1, (PC)←addr		CA al ah
JNC addr	If CY=0,(PC)←addr		D2 al ah
JC addr	If CY=1,(PC)←addr		DA al ah
JPO addr	If P=0, (PC)←addr		E2 al ah
JPE addr	If P=1, (PC)←addr		EA al ah
JP addr	If S=0, (PC)←addr		F2 al ah
JM addr	If S=1, (PC)←addr		FA al ah
PCHL	(PC_h)←(H), (PC_l)←(L)		E9
CALL GROUP			
CALL addr	(TOS)←(PC),(PC)←addr		CD al ah
CNZ addr	If Z=0, (TOS)←(PC),(PC)←addr		C4 al ah
CZ addr	If Z=1, (TOS)←(PC),(PC)←addr		CC al ah
CNC addr	If CY=0,(TOS)←(PC),(PC)←addr		D4 al ah
CC addr	If CY=1,(TOS)←(PC),(PC)←addr		DC al ah
CPO addr	If P=0, (TOS)←(PC),(PC)←addr		E4 al ah
CPE addr	If P=1, (TOS)←(PC),(PC)←addr		EC al ah
CP addr	If S=0, (TOS)←(PC),(PC)←addr		F4 al ah
CM addr	If S=1, (TOS)←(PC),(PC)←addr		FC al ah
	N.B. (TOS)←(PC) designates the following:- ((SP)-1)←(PC_h),((SP)-2)←(PC_l),(SP)←(SP)-2		
RETURN GROUP			
RET	(PC)←(TOS)		C9
RNZ	If Z=0, (PC)←(TOS)		C0
RZ	If Z=1, (PC)←(TOS)		C8
RNC	If CY=0,(PC)←(TOS)		D0
RC	If CY=1,(PC)←(TOS)		D8
RPO	If P=0, (PC)←(TOS)		E0
RPE	If P=1, (PC)←(TOS)		E8
RP	If S=0, (PC)←(TOS)		F0
RM	If S=1, (PC)←(TOS)		F8
	N.B. (PC)←(TOS) designates the following:- (PC_l)←((SP)),(PC_h)←((SP)+1),(SP)←(SP)+2		
RESTART GROUP			
RST 0	(TOS)←(PC),(PC)←00_{16}		C7
RST 1	(TOS)←(PC),(PC)←08_{16}		CF
RST 2	(TOS)←(PC),(PC)←10_{16}		D7
RST 3	(TOS)←(PC),(PC)←18_{16}		DF
RST 4	(TOS)←(PC),(PC)←20_{16}		E7
RST 5	(TOS)←(PC),(PC)←28_{16}		EF
RST 6	(TOS)←(PC),(PC)←30_{16}		F7
RST 7	(TOS)←(PC),(PC)←38_{16}		FF
ROTATE/CONTROL/SPECIAL GROUP			
RLC	(A_{n+1})←(A_n),(A_0)←(A_7),(CY)←(A_7)	***	07
RRC	(A_n)←(A_{n+1}),(A_7)←(A_0),(CY)←(A_0)	***	0F
RAL	(A_{n+1})←(A_n),(A_0)←(CY),(CY)←(A_7)	***	17
RAR	(A_n)←(A_{n+1}),(A_7)←(CY),(CY)←(A_0)	***	1F
NOP	No operation		00
HLT	Processor stopped until interrupt or reset		76
DI	Interrupts disabled		F3
EI	Interrupts enabled after next instruction		FB
XTHL	(L)↔((SP)),(H)↔((SP)+1)		E3
SPHL	(SP_h)←(H),(SP_l)←(L)		F9
XCHG	(H)↔(D),(L)↔(E)		EB
DAA	Decimal adjust accumulator	*	27
CMA	(A)←$(\overline{A})$		2F
STC	(CY)←1	***	37
CMC	(CY)←$(\overline{CY})$	***	3F
OUT port IN port	Not used in DCE Systems		D3 port DB port

NOG EEN PAAR INTERESSANTE COMBINATIES VAN BASIC EN MACHINETAAL

Deze routine geeft de geheugenplaats in de VIDEO-RAM (waar eventueel een dot terecht komt).

De basic routine :

```
1Ø  MODE 4A
2Ø  INPUT "SCRN";X,Y:POKE #2Ø11,Y:POKE #2Ø13,(X IAND #FF):POKE
    #2Ø14,X SHR 8
3Ø  CALLM #2ØØØ
4Ø  PRINT:PRINT HEX$(PEEK(#2Ø51));HEX$(PEEK(#2Ø5Ø)):GOTO 2Ø
```

De machinetaal routine(in te voeren in utility met het S (substitute) command:

```
2ØØØ F3 F5 3A 4Ø ØØ F5 E6 3F F6 8Ø 32 Ø6 FD E5 D5 C5
2Ø1Ø 3E 3C Ø1 3C ØØ CD B9 EB 22 5Ø 2Ø C1 D1 E1 F1 32
2Ø2Ø Ø6 FD F1 FB C9
```

Twee manieren om het toetsenbord te gebruiken als schrijfmachine :

1.
```
1ØØØ  A=GETC:IF A=Ø THEN 1ØØØ
1Ø1Ø  PRINT CHR$(A);:GOTO 1ØØØ
```

U zal merken dat nu ook de cursortoetsen en de tabtoets een character herbergen.

2.
```
1Ø  CALLM #3ØØØ
15  IF PEEK(#2Ø1Ø >=Ø THEN 1Ø
2Ø  PRINT CHR$(PEEK(#2Ø1Ø));
3Ø  GOTO 1Ø
```

De machinetaalroutine voor dit programma :

```
3ØØØ F5 E5 CD BB D6 32 1Ø 2Ø E1 F1 C9
```

GETC-routine in BASIC-ROM (#D6BB)
plaatst ASCI waarde van toets in register A.

```
1000  PRINT CHR$(12)
1010  FOR A=0 TO 10
1020  POKE #B9E4+2*A,#FF
1025  POKE #B9E4+2*A+#86,#FF
1030  NEXT
1035  CURSOR 23,12:PRINT "WARNING"
1040  FOR B=20 TO 1 STEP -1
1045  COLORT 0 9 9 0
1046  GOSUB 1100
1050  WAIT TIME B
1055  COLORT 0 9 0 9
1056  GOSUB 1100
1060  WAIT TIME B
1065  NEXT
1070  GOTO 1040
1100  RJ=GETC:IF RJ<>32 THEN RETURN
1130  PRINT :INPUT "LIST PROGRAM < Y/N > ";RJ$
1140  IF RJ$="Y" THEN PRINT CHR$(12):PRINT :LIST 1000-1070:GOSUB 2150:GOTO 20
1141  IF RJ$="N" THEN PRINT CHR$(12):PRINT :GOTO 20
1145  CURSOR 0,10:PRINT SPC(30):CURSOR 0,11
1150  RETURN
```

```
    60000 CLEAR 300
60006   POKE #29C,3:POKE #29E,0:POKE #3EC,#80:POKE #3ED,#28
60010   FOR T%=0.0 TO 10.0:READ D%
60020   FOR T1%=0.0 TO 15.0:READ D1%
60026   POKE D%,D1%:D%=D%+1.0:NEXT:NEXT
60030   POKE #71,3:POKE #70,0:REM interrupt 7 aanpassen
60040   REM eenheden seconden op   #3BA
60041   REM tientallen seconden op #3BB
60042   REM eenheden minuten op    #3BC
60043   REM tientallen minuten op  #3BD
60044   REM eenheden uren op       #3BE
60045   REM tientallen uren op     #3BF
60046   REM NA "RUN" KAN U ZONDER MEER
60047   REM EEN NIEUW PROGRAMMA LADEN OF INVOEREN
60048   REM DE KLOK LOOPT NIET VERDER BIJ
60049   REM LOAD,SAVE EN CHECK.
60050   REM NA EEN RESET KAN U DE KLOK OPNIEUW STARTEN
60051   REM DOOR : POKE#71,3:POKE#70,0
60052   REM U KAN HET DISPLAYEN STOPPEN DOOR :
60053   REM POKE #363,#C3 (vb INDIEN U NAAR EDIT-MODE GAAT)
60054   REM DISPLAY HERVATTEN DOOR :
60055   REM POKE #363,#C2
60056   REM VOOR 32K TOESTELLEN :
60057   REM IEDER #BF.. VERVANGEN DOOR #7F..
60058   REM VOOR 12K TOESTELLEN :
60059   REM IEDER #BF.. VERVANGEN DOOR #2F..
60060   REM VOOR 8K  TOESTELLEN :
60061   REM IEDER #BF.. VERVANGEN DOOR #1F..
60100   DATA #300,#C5,#D5,#E5,#F5,#21,#B9,#03,#06,#0A,#0E,#06,#16,#00,#1E
      ,#32,#34
60110   DATA #310,#7B,#BE,#C2,#57,#03,#72,#23,#34,#78,#BE,#C2,#5E,#03,#72
      ,#23,#34
60120   DATA #320,#79,#BE,#C2,#5E,#03,#72,#23,#34,#78,#BE,#C2,#5E,#03,#72
      ,#23,#34
60130   DATA #330,#79,#BE,#C2,#5E,#03,#72,#23,#34,#78,#BE,#C2,#41,#03,#72
      ,#23,#34
60140   DATA #340,#2B,#23,#3E,#02,#BE,#C2,#5E,#03,#2B,#3E,#04,#BE,#C2,#5E
      ,#03,#36
60150   DATA #350,#00,#23,#36,#00,#C3,#5E,#03,#F1,#E1,#D1,#C1,#C3,#A9,#D9
      ,#3A,#EF
60160   DATA #360,#BF,#FE,#7A,#C2,#57,#03,#21,#BA,#03,#7E,#C6,#30,#32,#77
      ,#BF,#23
60170   DATA #370,#7E,#C6,#30,#32,#79,#BF,#23,#7E,#C6,#30,#32,#7D,#BF,#23
      ,#7E,#C6
60180   DATA #380,#30,#32,#7F,#BF,#23,#7E,#C6,#30,#32,#83,#BF,#23,#7E,#C6
      ,#30,#32
60190   DATA #390,#85,#BF,#3E,#20,#32,#75,#BF,#32,#7B,#BF,#32,#81,#BF,#32
      ,#87,#BF
60200   DATA #3A0,#C3,#57,#03,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00
      ,#00,#00
60500   INPUT "INPUT THE TIME < HHMMSS >";T$:PRINT :A%=#3BF
60510   FOR D=0.0 TO LEN(T$)-1.0:T1$=MID$(T$,D,1)
60520   IF ASC(T1$)>47 AND ASC(T1$)<58 THEN POKE A%,VAL(T1$):A%=A%-1.0:IF
      A%=#3B9 THEN PRINT CHR$(12);:LIST 60040-60070
60530   NEXT
60550   END
```

DIGITAAL/ANALOOG CLOCK

H. VAN COOTEN 8K

```
/     REM CLOCK PROGRAM
 15     REM 1 MINUUT DIGITAAL
 20     REM 1 MINUUT ANALOOG
 25     REM AFWISSELEND  !!
 30     REM MET SECONDENWIJZER BIJ ANALOGE AANWIJZING.
 35     REM
 37     REM HARRY VAN COOTEN/JUNI 80/8K DAI
 40     PRINT CHR$(12);
 50     COLORT 12 4 12 12
 60     PRINT "CLOCK PROGRAM."
103     POKE #BFEF,#5C:REM OF #1FEF,#2FEF,#7FEF
104     PRINT                    (8K)  (12K) (32K)
105     PRINT
106     PRINT
107     PRINT
110     PRINT                          CLOCK PROGRAM."
111     REM
120     PRINT                          HARRY VAN COOTEN."
130     PRINT                          ";D$
140     PRINT
150     PRINT
160     PRINT
170     INPUT "INVOER S.V.P.  UREN TIENTALLEN";A%
180     PRINT
185     IF A%>2.0 THEN 170
190     INPUT "INVOER S.V.P.  UREN EENHEDEN  ";U%
195     IF U%>9.0 THEN 190
200     PRINT
210     INPUT "INVOER S.V.P.  MINUTEN TIENTALLEN";B%
211     PRINT
212     INPUT "INVOER S.V.P.  MINUTEN EENHEDEN  ";M%
213     PRINT
214     INPUT "INVOER S.V.P.  SECONDEN TIENTALLEN  ";C%
215     PRINT
216     INPUT "INVOER S.V.P.  SECONDEN EENHEDEN    ";S%
217     PRINT
218     IF B%>5.0 THEN 210
219     IF M%>9.0 THEN 212
220     IF C%>5.0 THEN 214
221     IF S%>9.0 THEN 216
225     REM
260     PRINT CHR$(12);
261     PRINT "  DIGITAL CLOCK."
262     POKE #BFEF,#5C:REM OF #1FEF,#2FEF,#7FEF
280     CURSOR 10,17
289     PRINT A%;U%;".";B%;M%;"    ";C%;S%
290     A=6.0
291     AA=#6B
292     POKE #BFEF-(A*#86),AA
295     S%=S%+1
```

```
 296 IF S%>9 THEN 400
 300 IF S%>6.0 AND C%>4.0 THEN 2040
 320 GOTO 1000
 400 S%=0
 410 C%=C%+1
 420 IF C%>5 THEN 450
 430 GOTO 1000
 450 C%=0
 460 M%=M%+1
 470 IF M%>9 THEN 500
 480 GOTO 1000
 500 REM
 502 M%=0
 510 B%=B%+1
 520 IF B%>5 THEN 550
 530 GOTO 1000
 550 B%=0
 600 U%=U%+1
 610 IF U%>3 THEN 700
 620 GOTO 1000
 700 U%=0
 710 A%=A%+1
 720 IF A%>2 THEN 800
 730 GOTO 1000
 800 A%=0
1000 FOR V=1.0 TO 1325.0
1010 NEXT V
1015 GOTO 280
2040 REM
2050 S9=PI/30.0
2060 M9=PI/1800.0
2070 U9=PI/21600.0
2071 M%=M%+1
2072 IF M%>9 THEN 2074
2073 GOTO 2087
2074 M%=0
2076 B%=B%+1
2077 IF B%<6 THEN 2087
2078 B%=0
2079 U%=U%+1
2080 IF U%>3.0 AND A%=2.0 THEN 2084
2081 IF U%>9.0 THEN 2083
2082 GOTO 2087
2083 U%=0:A%=A%+1:IF A%<3 THEN 2087
2084 A%=0:GOTO 2087
2087 M=(B%*10.0)+M%
2088 U=(A%*10.0)+U%
2090 MODE 2:REM OF MODE 4 OF MODE 6
2110 FOR X=0.0 TO 2.0*PI STEP PI/6.0
2120 DOT 30+30*SIN(X),30+30*COS(X) 15
 2130 NEXT X
2150 M3=M*PI/30.0
2160 U3=U*PI/6.0+M3/12.0
2170 S1=30.0
2171 S2=30.0
2172 M1=30.0
2173 M2=30.0
2174 U1=30.0
2175 U2=30.0
2190 FOR S3=0.0 TO 2.0*PI STEP S9
2200 DRAW 30,30 S5,S6 0
2210 DRAW 30,30 S1,S2 10
2220 DRAW 30,30 M5,M6 0
2230 DRAW 30,30 M1,M2 15
2240 DRAW 30,30 U5,U6 0
2250 DRAW 30,30 U1,U2 15
2270 WAIT TIME 27
2290 S5=S1
2292 S6=S2
2300 M5=M1
2302 M6=M2
2310 U5=U1
2312 U6=U2
2330 S1=30.0+28.0*SIN(S3)
2340 S2=30.0+28.0*COS(S3)
2350 M3=M3+M9
2360 M1=30.0+28.0*SIN(M3)
2370 M2=30.0+28.0*COS(M3)
2380 U3=U3+U9
2390 U1=30.0+20.0*SIN(U3)
2400 U2=30.0+20.0*COS(U3)
2403 NEXT S3
2404 MODE 0
2405 C%=0
2407 S%=0
2410 M%=M%+1
2420 IF M%<10 THEN 260
2430 M%=0
2440 B%=B%+1
2450 IF B%<6 THEN 260
2460 B%=0
2470 U%=U%+1
2480 IF U%<4 THEN 260
2490 IF U%<10 THEN 260
2500 U%=0
2510 A%=A%+1
2520 IF A%<3 THEN 260
2530 A%=0
2540 GOTO 260
2971 M%=M%+1
```

```
      REM ********FORMAAT-LISTING**********************************
    5       REM
   10       REM NEEM ZELF HET FORMAAT VAN DE LISTING IN DE HAND
  120       REM DEZE PROCEDURE BESTAAT ERIN DAT WE EERST HET PROGRAMMA
  130       REM NAAR DE EDIT-BUFFER STUREN,
  140       REM DOOR TWEEMAAL "BREAK" HET PROGRAMMA DAAR LATEN ZITTEN,
  150       REM EN DAN DE LIJNNUMMERS 60000-60050
  160       REM DE EDIT-BUFFER LATEN UITPRINTEN.
  170       REM IN DE EDIT-BUFFER STAAT NL NIET DE RUNTIME-CODE,
  180       REM MAAR HET GEDECODEERD FORMAAT.
  190       REM VOORAF MOET EEN CLEAR GEMAAKT WORDEN,
 1100       REM AFHANKELIJK VAN DE LENGTE VAN HET PROGRAMMA.
 1110       REM DE EDIT-BUFFER MOET NL HELEMAAL IN DE HEAP-RUIMTE KUNNEN,
 1120       REM ANDERS GAAN WE DE RUNTIME-CODE OOK UITPRINTEN.
 1130       REM DIT KAN GETEST WORDEN DOOR EERST DE LISTING TE LATEN
 1140       REM GEBEUREN MET DE PRINTER UIT.
 1150       REM ER IS GEEN BEPERKING VOOR DE LENGTE VAN DE LIJNEN.
 1160       REM DE SUBROUTINE OP 60030 ZORGT ERVOOR DAT DE LIJNNUMMERS
 1170       REM LOGISCH WORDEN GERANGSCHIKT.
 1180       REM ALLE WIJZIGINGEN IN DE EDIT-BUFFER BLIJVEN INTACT
 1190       REM ZODAT U DE SCHIKKING NAAR SMAAK KAN UITVOEREN.
10030       REM DE PROCEDURE VERLOOPT ALS VOLGT:
10035       REM CLEAR xxxxx
10040       REM EDIT n m / (gewenst deel van het programma)
10050       REM eventuele wijzigingen in edit-buffer(schikking)
10060       REM BREAK BREAK
10070       REM LOAD "programma met lijnnumers 60000-60050"
10080       REM U kan ook het programma intikken.
10090       REM RUN 60000
11000       REM HET EERSTE LIJNNUMMER WORDT VERMINKT,
11010       REM DOORDAT ER WAARSCHIJNLIJK EEN POINTER OPGETELD
11020       REM WORDT,MISSCHIEN VINDT IEMAND HIERVOOR DE OPLOSSING?

60000 FOR X=PEEK(#A2)+PEEK(#A3)*256 TO PEEK(#A4)+PEEK(#A5)*256-2
60002 IF FLAG=0.0 THEN FLAG=1.0:GOSUB 60030
60005 C=PEEK(X):IF CNT>75 THEN CNT=0.0:PAGE=PAGE+1.0:PRINT :PRINT "
60007 IF PAGE>=50.0 THEN PRINT :PRINT :PRINT :PAGE=0.0:GOSUB 60030
60010 PRINT CHR$(C);:CNT=CNT+1.0
60015 IF C=13.0 THEN PAGE=PAGE+1.0:GOSUB 60030:CNT=0.0
60020 NEXT
60025 GOTO 60025
60030 FOR Y=1.0 TO 6.0
60035 IF PEEK(X+Y)=32 GOTO 60040
60037 NEXT
60040 FOR M=1.0 TO 7.0-Y:PRINT " ";:NEXT
60050 RETURN
```

z-commando/program chaining

MACHINE LANGUAGE UTILITY : HET Z COMMANDO

IN DE PERSONAL COMPUTER MANUEL HANDELT EEN HOOFDSTUK OVER MOGELIJKHEDEN VAN DE UTILITY.
WIE ECHTER PROBEERT EEN MACHINETAALPROGRAMMA TE STARTEN MET HET G OF L COMMANDO, ZAL VOLGENS ONZE ERVARING ALS ENIG RESULTAAT EEN BONTE VERZAMELING KLEUREN OP HET SCHERM BEKOMEN. OM DEZE CRASH VAN DE MACHINE TE VERMIJDEN DIENEN WE VOORAF EEN INITIALISERING TE DOEN.
DIT KAN EENVOUDIG DOOR GEBRUIK VAN HET Z COMMANDO. SPIJTIG GENOEG VERMELDT DE MANUEL OVER DIT COMMANDO EN HET GEBRUIK ERVAN VOOR G OF L COMMANDO'S NIETS.
HET Z COMMANDO BESTAAT UIT DE LETTER Z ONMIDDELLIJK GEVOLGD DOOR EEN CIJFER VAN 1 TOT 7
HIERONDER GEVEN WE EEN (ONVOLLEDIG) OVERZICHT :

COMMANDO	GEVOLG
Z1	RESET ALLE CPU REGISTERS STACKPOINTER OP F900 GEZET
Z2	HERSTELLEN VAN DE INTERRUPTVECTOREN (0 TOT 7) MASK, TICC EN GIC BYTES WORDEN OP BEPAALDE STAND GEZET (TE BEKIJKEN MET V COMMANDO)
Z3	Z1 + Z2

VOOR GEBRUIK VAN EEN G OF L COMMANDO VOEREN WE EERST EEN Z3 UIT. HIERDOOR WORDT INTERRUPTVECTOR 0 KLAAR GEZET VOOR GEBRUIK DOOR HET L COMMANDO.
DEZE Z3 MOET SLECHTS 1 MAAL GEBEUREN NADAT DE MACHINE OPGEZET IS OF NA EEN RESET.

TIP ... TIP ... TIP ... TIP ... TIP ... TIP ... TIP ... TIP ...

PROGRAMMA-CHAINING

INDIEN UW PROGRAMMA TE GROOT IS VOOR HET BESCHIKBAAR GEHEUGEN EN HET PROGRAMMA KAN GESPLITST WORDEN IN DEELPROGRAMMA'S DAN KAN HET LADEN EN STARTEN VAN DE VERSCHILLENDE DELEN BIJ DAIPC AUTOMATISCH GEBEUREN. U ZORGT ERVOOR DAT DE LAATSTE LIJN VAN IEDER DEEL "LOAD" BEVAT. DEZE "LOAD" ZAL:

1) HET OUDE DEEL WISSEN
2) DE CASSETTE STARTEN
3) HET NIEUWE DEEL LADEN
4) EN ... HET NIEUWE DEEL STARTEN !!!

DEZE MOGELIJKHEID OPENT INTERESSANTE PERSPECTIEVEN, VOORAL VOOR 8K EN 12K TOESTELLEN.

LIST-PRINT

Doel :

Het maken van een listing van een BASIC-programma op de matrix printer, met op het einde van elke pagina de naam van het programma en een paginanummer.

Restricties :

1) Het te listen programma mag de regelnummers 65400 en hoger niet gebruiken.
2) Het aantal characters per regel (eksklusief het regelnummer) mag maksimaal 74 zyn.
3)Bedoeld voor papier waar 66 regels per pagina op kunnen. Aan te passen door de waardes van TT en Y te veranderen in de regels 65460, 65490, 65500 en 65505.

Werkwyze :

Merge het te listen programma en LIST-PRINT (L-P) volgens de methode zoals beschreven in het DAI-handbook, pagina 59, par. 6.2.4.4.

RUN 65400

L-P vraagt naar de naam van het te listen programma en naar het nummer van de eerste pagina van de te maken listing (hetgeen de mogelykheid geeft door te nummeren).
Hierna geeft L-P enige instructies betreffende het instellen van de printer.

Het duurt enige tyd voordat het printen start, tevens treedt er telkens een korte pauze op na het printen van het pagina-nummer.

Als laatste wordt geprint :

65400 REM ***** E I N D E L I S T I N G *****

'NAAM PROGRAMMA' Pagina 'XXX'

Hierna gaat L-P in de loop '65220 GOTO 65220' , waardoor nu de printer afgeschakeld kan worden zonder dat er op de volgende pagina ekstra printout komt.

Jean Dessart.

L I S T - P R I N T

```
65400 REM ************ E I N D E  L I S T I N G ************
65405 PRINT CHR$(12):PRINT :PRINT :PRINT :PRINT
65410 PRINT "LIST-PRINT (max. regelnummer is 65399).":PRINT :PRINT :PRINT
65415 PRINT "Naam van het programma ?"
65420 INPUT N$:PRINT :PRINT :PRINT "Nummer eerste pagina ?":INPUT P%:PRINT
65425 PRINT :PRINT :PRINT "Zet de vouw van het papier gelijk"
65430 PRINT "met de bovenzijde van het PRINT-HEAD."
65435 PRINT :PRINT "Schakel de printer aan en druk "
65440 INPUT "op de RETURN-toets.";NN$:PRINT
65442 PRINT :POKE #FF06,#E
65443 PRINT N$:PRINT :S=0.0
65445 A=PEEK(#2A0):AA=PEEK(#29F):P=A*256.0+AA:Q=P
65450 L=PEEK(Q):A=PEEK(Q+1):AA=PEEK(Q+2):R=A*256.0+AA
65455 IF R=65500.0 THEN 65460:Q=Q+L+1.0:GOTO 65450
65460 TT=54.0:ADRA=Q+4.0:ADRB=Q+6.0:Q=P
65465 L=PEEK(Q):POKE ADRA,PEEK(Q+1):POKE ADRA+1,PEEK(Q+2)
65470 T=T+1.0:Q=Q+L+1.0:L=PEEK(Q)
65475 IF PEEK(Q+1)=#FF AND PEEK(Q+2)=#78 THEN GOTO 65495
65480 IF T=TT THEN 65485:GOTO 65470
65485 POKE ADRB,PEEK(Q+1):POKE ADRB+1,PEEK(Q+2):GOSUB 65500
65490 T=0.0:TT=59.0:Q=Q+L+1.0:GOTO 65465
65495 S=1.0:PRINT :GOTO 65485
65500 LIST 1-65400:IF TT<>54.0 THEN GOTO 65505:PRINT :GOTO 65510
65505 Y=59.0-T:FOR X=1.0 TO Y:PRINT :NEXT X
65510 PRINT TAB(50);N$;"      Pagina";P%:P%=P%+1
65515 PRINT :PRINT :PRINT :PRINT :IF S=1.0 THEN 65520:RETURN
65520 GOTO 65520
65525 END
```

ZAKDOEK LEGGEN

```
5     COLORG 12 3 15 3
7     MODE 6
9     FOR A=YMAX/8.0 TO YMAX/140.0 STEP -2.0
10    FOR I=0.0 TO 2.0*PI STEP PI/150.0
20    R=A*(3.0-COS(6.0*I))
30    X=R*COS(I)
40    Y=R*SIN(I)
50    DOT X+XMAX/2,Y+YMAX/2 3
60    NEXT I
65    NEXT A
67    FOR S=1.0 TO 3.0
70    FOR A=1.0 TO 15.0
80    COLORG 0 A 12 3
95    WAIT TIME 100
96    NEXT A
97    NEXT S
100   GOTO 100
```

VARIABLES IN DAI-BASIC

To name variables in DAI-BASIC we can use from 1 to 14 characters, with the exceptional property that every character is recognised by BASIC. So AAAAAAAAAAAAA and AAAAAAAAAAAAB are distinct variables.

The first character must be alphabetic, and the rest may be either alphabetic or numeric (alphanumeric).

DAI-BASIC recognises several types of variables:

NUMERIC VARIABLES (NUMBERS)

A! = floating point	10 e-18 to 10 e18	ex: 2.98653
B% = integer	-/+ (2 e31)-1	ex: 645364726

At POWER-ON the DAIpc is initialised to IMP FPT: more clearly, every variable named is a floating point variable, unless explicitely shown otherwise.

We show that a variable is INTEGER by terminating it's name by a "%" character.

We can verify this if, after we have input a program, we give the IMP INT command. If we now list the program every variable name terminates by a "!". Every integer variable has lost it's "%".

Integer and floating point variables are stored in memory with a completely different format, those who wish to include machine language in their BASIC programs are advised to take this into consideration!

>>>>>>>>> It is possible, having written a program in FPT,to convert it to INT !!! (and conversely). This is the menu:

```
*LOAD program
*IMP FPT
*CLEAR  xxxx (more than program size)
*BREAK BREAK
*IMP INT
*POKE #135,2
*LIST
```

ALPHANUMERIC VARIABLES (STRINGS)

STRINGS can be from 0 to 255 characters long and their name is terminated by the sign "$". ex: A$

ARRAY VARIABLES

These can have the various following caracteristics:

A!(X,Y,...)	floating point array	A(X,Y,...) if IMP FPT
A%(X,Y,...)	integer array	A(X,Y,...) if IMP INT
A$(X,Y,...)	string array	

The suscripts cannot be greater than 255.

To sum up A can stand for all sorts:

A	A!	A%	A$	A(9,9)	A!(9,9)	A%(9,9)	A$(9,9)

In IMP FPT is A=A! and A(9,9)=A!(9,9).

GRAFIC ARTIST (Grafisch Tablet)

This program makes it possible to create drawings, grafics and texts. Input is done with the paddels, event knob and the keyboard for text.

The program presents a tablet with menu at the top. On this tablet is a (non destructif) cursor that is moved with the paddle control. By moving the cursor into the appropriate compartment of the menu you can select the different functions. EVENT is the general ON/OFF switch for most instructions.

To run the program you may choose either MODE 5 or MODE 6. It is written in a modular way so that instructions can at will be modified or expanded.

The standard instructions

T:TRACE The cursor moved with the paddle leaves a trace of the selected colour.EVENT acts as ON/OFF switch.

A:DRAW By selecting a first then a second point with EVENT the two points are joined by a line.

I:FILL The normal BASIC instruction (opposite corners selected with EVENT).

C:CIRCLE First choose the center,then any point of the circumference.
There are different possibilities:
-MODE 5, COMPUTE flag ON: The circumference is first drawn, then the circle is filled with rays. The drawing can be stopped at any moment with EVENT.

p10

GRAFIC TABLET 2 /

If the drawing is stopped while the circumference is drawn, then after D appears the length of the part drawn. If it is stopped while the rays are drawn then appears after A the surface of the part filled in.
Note: all numerical values are expressed in screen points (250*250).
-MODE 5, COMPUTEflag OFF: A solid circle is drawn with horizontal lines so that colour conflicts are avoided for this 16 colour MODE.
-MODE 6, COMPUTEflag ON: First the circumference is drawn then the rays. While the rays are drawn COLORG can be influenced with the horizontal pot, this affects the existing drawing and presents astounding possibilities.
For numerical results see MODE 5 COMPUTEflag ON.
-MODE 6, COMPUTEflag OFF: The same solid circles as in MODE 5.

S ELLIPSE You determine a point on the LEFT of the perimeter, then the TOPmost point of the ellips. The order is important.
Note: Circles and ellipses that would go out of the tablet will not be drawn, a signal sounds to warn you.

W WIPE Clears a part of the picture. Which part, is determined by selecting opposite corners with EVENT.

O DOT Leaves a dot every time you press EVENT.

? COMPUTE ON/OFF switch for calculations.
At the bottom of the right side is a flag to remind you.

D: DISTANCE With COMPUTEflag ON we get behind D the distance between the two points selected with EVENT.With COMPUTEflag OFF we are in DOT MODE.

L CLEAR Clears the whole tablet ,MODE and PENCOLOUR are unchanged.

D) -With COMPUTEflag ON we get an approximate area or perimeter for any shape.The precision is determined by the array dimention(255).

-With COMPUTEflag ON we get the length of any line.

H HELP Info-page with concise information over different orders. These orders can eventually be changed into other grafic functions (E.G. fill the tablet with vertical and horizontal reference lines.)

N SCANNING With COMPUTEflag ON : vertical lines.
With COMPUTEflag OFF: horizontal lines.
A chosen rectangle (see FILL or FRAME) is scanned with horzontal or vertical lines.
The distance separating the lines is determined by the previous order.
e.g. first A(DRAW) then N gives lines every 2 steps.
first I(FILL) then N gives lines every 3 steps.
n.b. first T(TRACE) then N gives vert.+horiz. lines with a fixed distance between them.

T TEXT Draws where the cursor is positioned, any letter chosen by means of the keyboard.

COLOR The last compartment of the menu is for selecting PENCOLOR.Above this compartment there is a white surface (mode 5), by turning the hor.pot one can see all the 15 available colours (this does not change the background colour otherwise we would end up with an invisible CURSOR !).
By pressing EVENT one selects the shown colour as PENCOLOR.A noise sounds as warning that one must get out of the menu. The chosen colour is shown alongside the tablet.

REM In MODE 6 (4 colours) the colours are not shown but you get the colour code in hexadecimal:

F=15=white	E=14=yellow	D=13=light green
C=12=light blue	B=10=pink	A=10=orange

and so on ...

starting machine language programming

Verkenningstochten in machinetaal

Bijgaand machinetaalprogramma simuleert een kleine grafische toestand in machinetaal.

Toelichting (we verwijzen naar de lijnnummers van de ASSEMBLERlist.)

1 ORIGIN BEPALEN:deze routine zal geplaatst worden vanaf hex 300

2-5 de REGISTERS SAVEN op stack

6 Het scherm initieren voor MODE 2

9-10 registers HL,C laden met de begincoördinaat voor de dot

11 de dot plaatsen

12 GETCroutine van ROM aanroepen
(ASCII waarde van toets bevindt zich na de routine in A)

13 vlaggen aanpassen

14 testen op 0(geen toets)

15 testen op18(cursor links)

17 testen op 19(cursor rechts)

19 testen op 9 (TAB)=terug naar BASIC

LEFT en RIGHT wissen de actuele dot,verhogen of verlagen de Xwaarde en plaatsen de nieuwe dot.

39 OUT:hier komen we terecht bij TAB ...terug naar BASIC(MODE2A)

Dit is de OBJECTCODE BLOC die we met SUBSTUTUTE in UTILITY kunnen invoeren:

```
0300 E5 C5 D5 F5 3E 02 EF 18 21 23 00 01 20 00 CD 3E
0310 03 CD BB D6 B7 CA 11 03 FE 12 CA 2A 03 FE 13 CA
0320 34 03 FE 09 CA 48 03 C3 11 03 CD 43 03 2D CD 3E
0330 03 C3 11 03 CD 43 03 2C CD 3E 03 C3 11 03 3E 0F
0340 EF 1E C9 3E 00 EF 1E C9 F1 D1 C1 E1 C9 00 00 00
```

```
10   MODE 2
20   X=XMAX/2.0:Y=YMAX/2.0
30   DOT X,Y 15
40   G=GETC:IF G=0.0 THEN 40
50   IF G=18.0 THEN 100
60   IF G=19.0 THEN 200
70   IF G=9.0 THEN END
100  DOT X,Y 0
110  X=X-1.0:DOT X,Y 15:GOTO 40
200  DOT X,Y 0
210  X=X+1.0:DOT X,Y 15:GOTO 40
```

Dit is hetzelfde programma in BASIC,vergelijkt U maar de snelheid(met REPEATtoets)

Merk ook op dat de machinetaal-routine geen error geeft bij OFF SCREEN !!

starting machine language programming

```
PAGE 01

001
002 0300 E5                ORG   :300
003 0301 C5                PUSH  H
004 0302 D5                PUSH  B
005 0303 F5                PUSH  D
006 0304 3E02              PUSH  PSW
007 0306 EF                MVI   A,2        **** INIT MODE 2***
008 0307 18                RST   5
009 0308 212300            DATA  :18
010 030B 012000            LXI   H,35       **** INIT REGISTERS
011 030E CD3E03            LXI   B,32       **** FOR DOT
012 0311 CDBBD6      GETC  CALL  :D6BB
013 0314 B7                CALL  DOT
014 0315 CA1103            ORA   A
015 0318 FE12              JZ    GETC       **** NO KEY
016 031A CA2A03            CPI   18
017 031D FE13              JZ    LEFT
018 031F CA3403            CPI   19
019 0322 FE09              JZ    RIGHT
020 0324 CA4903            CPI   9
021 0327 C31103            JZ    OUT
022 032A CD4303      LEFT  JMP   GETC       **** NO VALID KEY
023 032D 2D                CALL  WIS
024 032E CD3E03            DCR   L
025 0331 C31103            CALL  DOT
026 0334 CD4303      RIGHT JMP   GETC
027 0337 2C                CALL  WIS
028 0338 CD3E03            INR   L
029 033B C31103            CALL  DOT
030 033E 3E0F        DOT   JMP   GETC
031 0340 EF                MVI   A,15
032 0341 1E                RST   5
033 0342 C9                DATA  :1E
034 0343 3E00        WIS   RET
035 0345 EF                MVI   A,0
036 0346 1E                RST   5
037 0347 C9                DATA  :1E
038 0348 F1                RET
039 0349 D1          OUT   POP   PSW
040 034A C1                POP   D
041 034B E1                POP   B
042 034C C9                POP   H
043 034D                   RET
                           END

****************************
* S Y M B O L   T A B L E *
****************************

DOT    033E   GETC   0311   LEFT   032A   OUT    0349
RIGHT  0334   WIS    0343
```

PEEK & POKE

VIDEO RAM IN MODE Ø

In MODE Ø is er een delay van 3 characters aan het begin van de lijn. Nu is op de meeste TV's deze ruimte wel te zien en dus te gebruiken. We geven alle adressen voor een 48K machine,voor andere modellen de klassieke aanpassing:8K:1°°°,12K:2°°°,32K:7°°°.
Voor de eerste lijn zijn deze ongebruikte posities:

éBFED éBFEB éBFE9

de corresponderende color bytes zitten 3adressen lager:
éBFEA éBFE8 éBFE6
Proberen in COMMAND MODE:
POKE éBFED,83 : POKE éBFEA,éFF
resultaat : "S" met gekleurde achtergrond.
COLORT 8 0 12 0 en de achtegrond wordt lichtblauw.
Door van deze adressen é86 of 134 (of veelvouden daarvan) af te trekken kan je ook de vrije posities van de overige lijnen gebruiken. Deze eerste 3 posities vormen een non-scrolling gebied,dit geeft extra mogelijkheden voor opmaak in MODE Ø!
De bytes worden wel gecleard door PRINT CHR&(12),LIST,MODE Ø.
IN TABEL:

	CHAR	COL	CHAR	COL	CHAR	COL
LIJN 23	#BFED	#BFEA	#BFEB	#BFE8	#BFE9	#BFE6
LIJN 22	#BF67	#BF64	#BF65	#BF62	#BF63	#BF60
LIJN 21	#BEE1	#BEDE	#BEDF	#BEDC	#BEDD	#BEDA
LIJN 20	#BE5B	#BE58	#BE59	#BE56	#BE57	#BE54
LIJN 19	#BDD5	#BDD2	#BDD3	#BDD0	#BDD1	#BDCE
LIJN 18	#BD4F	#BD4C	#BD4D	#BD4A	#BD4B	#BD48
LIJN 17	#BCC9	#BCC6	#BCC7	#BCC4	#BCC5	#BCC2
LIJN 16	#BC43	#BC40	#BC41	#BC3E	#BC3F	#BC3C
LIJN 15	#BBBD	#BBBA	#BBBB	#BBB8	#BBB9	#BBB6
LIJN 14	#BB37	#BB34	#BB35	#BB32	#BB33	#BB30
LIJN 13	#BAB1	#BAAE	#BAAF	#BAAC	#BAAD	#BAAA
LIJN 12	#BA2B	#BA28	#BA29	#BA26	#BA27	#BA24
LIJN 11	#B9A5	#B9A2	#B9A3	#B9A0	#B9A1	#B99E
LIJN 10	#B91F	#B91C	#B91D	#B91A	#B91B	#B918
LIJN 9	#B899	#B896	#B897	#B894	#B895	#B892
LIJN 8	#B813	#B810	#B811	#B80E	#B80F	#B80C
LIJN 7	#B78D	#B78A	#B78B	#B788	#B789	#B786
LIJN 6	#B707	#B704	#B705	#B702	#B703	#B700
LIJN 5	#B681	#B67E	#B67F	#B67C	#B67D	#B67A
LIJN 4	#B5FB	#B5F8	#B5F9	#B5F6	#B5F7	#B5F4
LIJN 3	#B575	#B572	#B573	#B570	#B571	#B56E
LIJN 2	#B4EF	#B4EC	#B4ED	#B4EA	#B4EB	#B4E8
LIJN 1	#B469	#B466	#B467	#B464	#B465	#B462
LIJN Ø	#B3E3	#B3E0	#B3E1	#B3DE	#B3DF	#B3DC

```
1     PRINT TAB(20);"CHAR";TAB(27);"COL";TAB(33);"CHAR";TAB(39);"COL";TAB(45
      );"CHAR";TAB(51);"COL"
6       P=#BFED
10      FOR X=1.0 TO 60.0:PRINT CHR$(#C0);:NEXT:PRINT
20      FOR X=23.0 TO 0.0 STEP -1.0
30      PRINT "LIJN ";X;TAB(20);"#";HEX$(P);TAB(26);"#";HEX$(P-3);TAB(32);"#
      ";HEX$(P-2);
40      PRINT TAB(38);"#";HEX$(P-5);TAB(44);"#";HEX$(P-4);TAB(50);"#";HEX$(P
      -7)
50      P=P-#86
60      NEXT
70      FOR X=1 TO 60:PRINT CHR$(#C0);:NEXT:PRINT
```



```
***  RADAR-SIMULATIE                                        PAGE 01


1         REM radar-simulatie
5         CLEAR 3000:ENVELOPE 1 15,10;10,10;5,10;0
10        N=2.0*PI/50.0
20        DIM SI(50.0),CO(50.0)
30        FOR X=1.0 TO 50.0
40    1     SI(X)=SIN(H):CO(X)=COS(H)
50    1     H=H+N
60    1     NEXT
100       MODE 4:COLORG 0 8 1 5
105       A=XMAX/2.0:B=YMAX/2.0:GOSUB 200
110       FOR X=1.0 TO 50.0
120   1     DRAW A,B A+40*SI(X),B+40*CO(X) 5
130   1     DRAW A,B A+40*SI(X),B+40*CO(X) 0
135   1     IF X=25 THEN DOT 80,45 5:SOUND 1 0 15 0 FREQ(2000.0)
136   1     IF X=27.0 THEN DOT 80,45 1
137   1     IF X=29 THEN DOT 80,45 0:SOUND OFF
140   1     NEXT
150       GOTO 110
200       FILL 0,0 XMAX,YMAX 8
210       FOR Z=-40.0 TO 40.0
220   1     DRAW A-SQR(1600.0-Z*Z),B+Z A+SQR(1600.0-Z*Z),B+Z 0
230   1     NEXT
240       RETURN
```

*** TUBULAR BELLS *** SIP R. PAGE 01

```
1          REM tubular bells *** sip r.
5          CLEAR 1000
10         DIM NOTE(12.0),MUSIC(16.0),OCTAVE(16.0),DUREE(16.0)
20         FOR A=1.0 TO 12.0:READ NOTE(A):NEXT A
30         FOR A=1.0 TO 16.0:READ MUSIC(A),OCTAVE(A),DUREE(A):NEXT A
40         ENVELOPE 0 2,5;4,5;8,5;12,5;8,5;4,5;2,5;
41         ENVELOPE 1 4,10;8,10;15,10;8,10;4,10;
45         FOR MELODIE=1.0 TO 10.0
50     1     FOR A=1.0 TO 4.0
60     2       FOR B=1.0 TO 16.0
65     3         FREQUE=NOTE(MUSIC(B))
70     3         SOUND 0 0 15 0 FREQ(FREQUE*OCTAVE(B))
75     3         WAIT TIME 10/DUREE(B)
80     3         NEXT B
90     2       NEXT A
100    1     FOR A=1.0 TO 4.0
110    2       FOR B=1.0 TO 16.0
115    3         FREQUE=NOTE(MUSIC(B))
120    3         SOUND 0 0 15 0 FREQ(FREQUE*OCTAVE(B))
130    3         SOUND 1 0 6 0 FREQ((FREQUE/2.0)*OCTAVE(B))
135    3         WAIT TIME 9/DUREE(B)
140    3         NEXT B
150    2       NEXT A
160    1     FOR A=1.0 TO 4.0
170    2       FOR B=1.0 TO 16.0
180    3         FREQUE=NOTE(MUSIC(B))
190    3         SOUND 0 0 15 0 FREQ(FREQUE*OCTAVE(B))
200    3         SOUND 1 0 4 1 FREQ((FREQUE/2.0)*OCTAVE(B))
210    3         SOUND 2 0 8 0 FREQ(FREQUE*OCTAVE(B)*2.0)
220    3         WAIT TIME 8/DUREE(B)
230    3         NEXT B
240    2       NEXT A
250    1     FOR A=1.0 TO 4.0
260    2       FOR B=1.0 TO 16.0
270    3         FREQUE=NOTE(MUSIC(B))
280    3         SOUND 0 1 12 0 FREQ(FREQUE*OCTAVE(B))
290    3         SOUND 1 0 8 0 FREQ((FREQUE/2.0)*OCTAVE(B))
300    3         SOUND 2 0 3 2 FREQ(FREQUE*OCTAVE(B)*2.0)
310    3         WAIT TIME 8/DUREE(B)
315    3         SOUND OFF
320    3         NEXT B
330    2       NEXT A
335    1     SOUND OFF
```

*** TUBULAR BELLS *** SIP R. PAGE 02

```
340    1     NEXT MELODIE
9000       STOP:END
10000      DATA 261,277,293,311,329,349,369,391,415,440,466,493
10001      DATA 5,1,1,10,1,1,5,1,1,12,1,1,5,1,1,8,1,1.5,10,1,1.25,5,1,1,1
C          ,2,1,5,1,1,3,2,1,5,1,1,12,1,1.5,1,2,1.25,5,1,1,12,1,1
```

SIZE : #4D1 DATE : _____ AUTEUR : R.SIP TITLE TUBULAR BELLS

FPT	:	A	loopvariabele
FPT	:	B	loopvariabele
FPT	:	FREQUE	te spelen frekwentie
FPT	:	MELODIE	loopvariabele
FPT	:	X	restant
FPT (ARR)	:	DUREE	lengte v.d. noot
FPT (ARR)	:	MUSIC	de melodie
FPT (ARR)	:	NOTE	frekwentie van de 12 gebruikte noten
FPT (ARR)	:	OCTAVE	oktaaf


```
2      REM FAKKEL IN MODE 0
5      MODE 0:PRINT CHR$(12);
10     POKE #75,32
20     FOR X1=2.0 TO 56.0 STEP 4.0
30     FOR Y1=2.0 TO 21.0 STEP 3.0
35     CURSOR X1,Y1+1:PRINT CHR$(46);CHR$(46);CHR$(46)
40     CURSOR X1,Y1:PRINT CHR$(19);CHR$(16);CHR$(18)
41     CURSOR X1+1,Y1-1:PRINT CHR$(10)
50     NEXT:NEXT
```

variabelen atlas

```
60000   REM VARIABELEN ATLAS
65005   DIM QQQ$(100.0)
65009   PRINT "SIZE : #";HEX$((PEEK(#2A4)*256+PEEK(#2A3))-(PEEK(#2A0)*256
      +PEEK(#29F))-#400);" ";
65010   PRINT TAB(13);" DATE : ";:GOSUB 65180
65012   PRINT " AUTEUR : ";:GOSUB 65180:PRINT " TITLE ";:GOSUB 65180:PRIN
      T
65016   FOR QQ=1.0 TO 65.0:PRINT CHR$(#A4);:NEXT:PRINT :PRINT
65020   QQB%=PEEK(#2A1)+256*PEEK(#2A2):QQE%=PEEK(#2A3)+256*PEEK(#2A4)
65030   QQI%=QQB%
65040   IF QQI%>QQE%-4 THEN 65160
65050   QQ%=PEEK(QQI%)
65060   QQ3$="":FOR QQJ%=QQI%+1 TO QQI%+(QQ% IAND #F):QQ3$=QQ3$+CHR$(PEEK
      (QQJ%)):NEXT
65070   IF LEN(QQ3$)<2 GOTO 65080:IF LEFT$(QQ3$,2)="QQ" GOTO 65140
65080   IF QQ% IAND #30=0 THEN QQ1$="FPT"
65090   IF QQ% IAND #30=#10 THEN QQ1$="INT"
65100   IF QQ% IAND #30=#20 THEN QQ1$="STR"
65110   IF QQ% IAND #40=0 THEN QQ2$="        :  "
65120   IF QQ% IAND #40=#40 THEN QQ2$=" (ARR) :  "
65130   QQQ$(QQCNT)=QQ1$+QQ2$+QQ3$:IF QQ% IAND #30=#20 THEN QQQ$(QQCNT)=Q
      QQ$(QQCNT)+"$"
65135   QQCNT=QQCNT+1.0
65140   QQI%=QQJ%+3:IF QQ% IAND #F0<#20 THEN QQI%=QQI%+2
65150   GOTO 65040
65160   REM ALFABETISCH
65165   FOR QQX=0.0 TO QQCNT-2.0
65166   IF QQQ$(QQX)<QQQ$(QQX+1.0) THEN 65170
65167   QQM$=QQQ$(QQX):QQQ$(QQX)=QQQ$(QQX+1.0):QQQ$(QQX+1.0)=QQM$
65168   QQFLAG=1.0
65170   NEXT
65172   IF QQFLAG=1.0 THEN QQFLAG=0.0:GOTO 65165
65175   FOR QQX=0.0 TO QQCNT-1.0
65177   PRINT QQQ$(QQX);TAB(20);:FOR QQZ=1.0 TO 40.0:PRINT CHR$(#A4);:NEX
      T:PRINT
65178   NEXT
65179   POKE #131,1:STOP
65180   FOR QQ=1.0 TO 9.0:PRINT CHR$(#A4);:NEXT:RETURN
```

VARIABELEN ATLAS ...documenteer uw BASICprogramma's

Een lijst met gebruikte variabelen + omschrijving is een aardige documentatie bij BASICprogramma's.Dit programma bekijkt voor U de SYMBOL TABLE,haalt er alle variabelen uit en drukt een keurig gerangschikt lijstje.Indien U dan nog invult waarvoor de variabelen gebruikt worden,is uw programma heel wat duidelijker voor collega's en voor uzelf.
De routine geeft ook de juiste omvang van uw programma(zonder rekening te houden met grafische mode of HEAP size).
De routine meldt geen variabelen die beginnen met QQ en kon zichzelf dus niet documenteren.

```
65005  array om de variabelennamen tijdelijk op te slaan
65009  berekening geheugenomvang
65010 - 65016 titelopmaak
65020  situering SYMBOL TABLE
65080 - 65120 vaststellen van het type-variabele
65160  alfabetisch rangschikken (bubblesort)
65175  printout
```

Gebruik

MERGING op de gekende manier,dan RUN 65000.

BELANGRIJK

Bij de meeste programma's zal U tot de vaststelling komen dat er in de SYMBOL TABLE heel wat variabelen zitten die er eigenlijk niet meer thuis horen.
Alle variabelen die ooit ingevoerd zijn tijdens de programma-ontwikkeling en ook vele typfouten die als variabele door BASIC herkend zijn blijven namelijk ten eeuwige dage in de symbol table zitten.Dit vertraagt uw programma en maakt de omvang (dus ook de LOAD en SAVE tijd)onnodig groter.
Daarom deze goede gewoonte:
Indien uw programma helemeel klaar is ....grote kuis houden.
(vraag ter referentie eerst PRINT FRE)
Dan het programma integraal naar de EDITBUFFER sturen.(na CLEAR....)
Dan NEW. Het programma zit nu nog in de EDITBUFFER als"source code".
NEW heeft de bestaande SYMBOL TABLE gecleard.
POKE hex 135,2 haalt het programma terug uit de EDIBUFFER en verplicht BASIC de SYMBOL TABLE helemaal terug op te bouwen.
Met lange programma's kan dit even duren!
Als de prompt verschijnt is de SYMBOL TABLE opnieuw opgebouwd, ditmaal zonder overbodige variabelen.
Vraag nu even PRINT FRE om te controleren hoeveel geheugenruimte je gewonnen hebt...
ter referentie:het programma "DIGITAAL/ANALOOG CLOCK" van H.van Cooten bevatte voor de operatie 64 variabelen,daarna nog 31 !!
De geheugenomvang werd gereduceerd met 360 bytes.

nb:de printout is gebaseerd op de grafische characterset van EPSON X80.

UITSLAG VAN "DAInamic-ORIGINEEL" WEDSTRIJD

Er zijn verrassend veel programma's ingestuurd voor onze eerste wedstrijd. De keuze was dan ook niet eenvoudig.
Uiteindelijk is BARRICADE van de heer DRUYFF bekroond.
Het is erg moeilijk om te verwoorden waarom dit programma de voorkeur van de jury kreeg,typ het maar eens in en bekijkt U het maar.
De uitslag is definitief,maar commentaar is erg welkom.
Hierbij geven we U de beschrijving van het programma door de auteur.
Dank en gelukwensen voor alle inzenders,een nieuwe wedstrijd volgt spoedig...

BARRICADE

Bij het spel zelf staat geen uitleg omdat die in vele gevallen overbodig is.
Na RUN krijgt U een kader in beeld.Zodra nu de spatiebalk wordt ingedrukt verschijnt een balletje in beeld dat gelijk horizontaal of vertikaal gaat bewegen.Zodra het balletje tegen iets opbotst kiest het een andere richting.Wanneer de spatiebalk wordt ingedrukt wordt er een barricade achter het balletje gezet.De bedoeling is het balletje te vangen.Is dit gelukt dan krijgt U gelijk een nieuw balletje.Als het vijfde balletje gevangen is,verandert het beeld van kleur,de barricades worden dan geteld.
Komt het beeld weer in de oorspronkelijke kleur terug dan kan met spatiebalk het resultaat aagevraagd worden.Het tellen kan onderbroken worden met toets S.
Standaard opdracht is:probeer de vijf balletjes te vangen met zo weinig mogelijk barricades.
Variant:probeer zo veel mogelijk barricades te zetten.

Het programma is afgedrukt in FPT :alle variabelen zijn integer (voor de snelheid).Indien U bij het begin IMP INT typt,hoeft U de %tekens niet in te voeren.
veel plezier .....

```
  2       REM ************************************************************
  3       REM *                                                          *
  4       REM *        HET WINNENDE PROGRAMMA                            *
  5       REM *                                                          *
  6       REM *        B A R R I C A D E   f.h.Druyff                    *
  7       REM *                                                          *
  8       REM ************************************************************
 10       POKE #75,32
 20       MODE 2:T%=0
 30       COLORG 0 0 0 0
 40       DRAW 0,0 XMAX,0 22
 50       DRAW 0,0 0,YMAX 22
 60       DRAW 0,YMAX XMAX,YMAX 22
 70       DRAW XMAX,0 XMAX,YMAX 22
 80       K%=0:L%=5:M%=9:N%=14:COLORG K% L% M% N%
 90       IF GETC<>32 GOTO 90
100       B%=1
110       I%=RND(XMAX):J%=RND(YMAX)
120       IF SCRN(I%,J%)=M% OR SCRN(I%,J%)=N% GOTO 110
130       DOT I%,J% N%
140       IF SCRN(I%-1,J%)=10 THEN IF SCRN(I%+1,J%)=10 THEN IF SCRN(I%,J%-1)=
      10 THEN IF SCRN(I%,J%+1)=10 GOTO 110
150       ON RND(4.0) GOTO 300,400,500
200       IH%=I%+1:IF SCRN(IH%,J%)=M% GOTO 260
210       DOT IH%,J% N%
220       IF GETC=32 THEN DOT I%,J% M%:GOTO 240
230       DOT I%,J% K%
240       I%=IH%
250       GOTO 200
260       IF SCRN(I%-1,J%)=M% THEN IF SCRN(I%,J%-1)=M% THEN IF SCRN(I%,J%+1)=
      M% GOTO 600
270       ON RND(3.0) GOTO 400,500
300       IH%=I%-1:IF SCRN(IH%,J%)=M% GOTO 360
310       DOT IH%,J% N%
320       IF GETC=32 THEN DOT I%,J% M%:GOTO 340
330       DOT I%,J% K%
340       I%=IH%
350       GOTO 300
360       IF SCRN(I%+1,J%)=M% THEN IF SCRN(I%,J%-1)=M% THEN IF SCRN(I%,J%+1)=
      M% GOTO 600
370       ON RND(3.0) GOTO 500,200
400       JH%=J%+1:IF SCRN(I%,JH%)=M% GOTO 460
410       DOT I%,JH% N%
420       IF GETC=32 THEN DOT I%,J% M%:GOTO 440
430       DOT I%,J% K%
440       J%=JH%
450       GOTO 400
460       IF SCRN(I%-1,J%)=M% THEN IF SCRN(I%+1,J%)=M% THEN IF SCRN(I%,J%-1)=
```

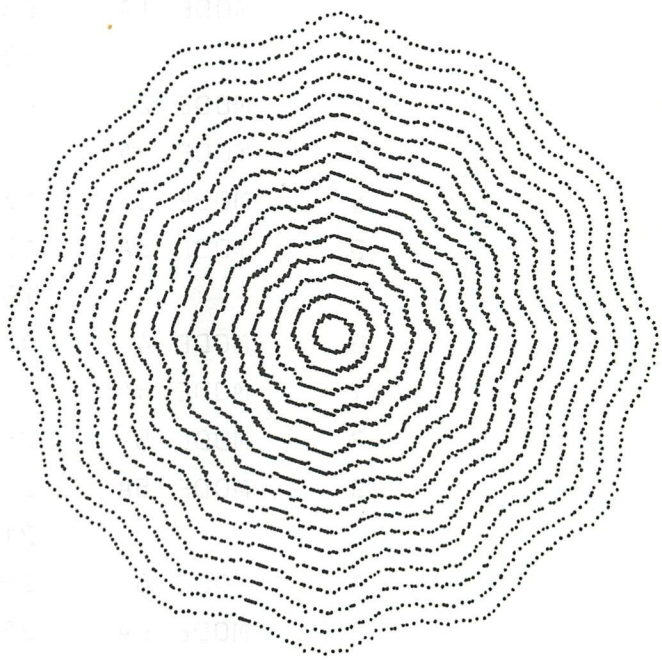
barricade

```
470       ON RND(3.0) GOTO 200,300
500       JH%=J%-1:IF SCRN(I%,JH%)=M% GOTO 560
510       DOT I%,JH% N%
520       IF GETC=32 THEN DOT I%,J% M%:GOTO 540
530       DOT I%,J% K%
540       J%=JH%
550       GOTO 500
560       IF SCRN(I%+1,J%)=M% THEN IF SCRN(I%-1,J%)=M% THEN IF SCRN(I%,J%+1)=
        M% GOTO 600
570       ON RND(3.0) GOTO 200,300
580       GOTO 400
600       B%=B%+1
610       IF B%<6 GOTO 110
620       COLORG 0 0 5 10:J%=1
630       FOR I%=1 TO XMAX-1
640       IF SCRN(I%,J%)=5 THEN T%=T%+1
650       NEXT
660       IF GETC=83 THEN MODE 0:GOTO 710
670       J%=J%+1:IF J%<YMAX-1 GOTO 630
680       COLORG K% L% M% N%
690       IF GETC=0 GOTO 690:MODE 0
700       CURSOR 7,15:PRINT "JE HEBT ER";T%;" BARRICADES VOOR GEBRUIKT."
710       CURSOR 6,7:PRINT "voor nog een spelletje: druk de spatiebalk in."
720       CURSOR 25,5:PRINT "(anders S)"
730       G%=GETC:IF G%=32 GOTO 20:IF G%<>83 GOTO 730
740       POKE #75,95
750       END
```

```
***  GEKLEURDE ACHTERGRONDEN IN MODE 0                                PAGE 01


10        REM gekleurde achtergronden in mode 0

20        REM list & print chr$(12) herstellen het formaat
25        FOR X=1.0 TO 24.0
26    1     PRINT "T E S T   C O L O R   I N   M O D E  0 ************"
27    1     NEXT
30        FOR X=#BF6C TO #BFEA STEP 2.0:POKE X,#FF:NEXT
40        FOR X=#B652 TO #B652+20.0 STEP 2.0
50    1     FOR Z=X TO X+#86*10.0 STEP #86
60    2       POKE Z,#FF:NEXT:NEXT
70        FOR X=#B400 TO #B450 STEP 2.0
80    1     POKE X,#FF:NEXT
100       FOR X=#B3E4 TO #BFEE STEP #86
110   1     POKE X,192+RND(32.0)
115   1     WAIT TIME 5
120   1     NEXT
122       FOR X=1.0 TO 20.0:WAIT TIME 5
125   1     COLORT RND(16.0) RND(16.0) RND(16.0) RND(16.0)
127   1     NEXT
130       GOTO 100
```

*** KANTEN KLEEDJE J.C.DELVOYE

```
1         REM kanten kleedje j.c.delvoye
5         COLORG 8 1 2 3
10        MODE 6
20        FOR A=YMAX/8.0 TO YMAX/1400.0 STEP -2.0
30    1     FOR I=0.0 TO 2.0*PI STEP PI/150.0
40    2       R=A*(3.0-COS(12.0*I)/10.0)
50    2       X=R*COS(I)
60    2       Y=R*SIN(I)
65    2       DOT XMAX/2-X,YMAX/2-Y 1
80    2       NEXT I
90    1     NEXT A
100       FOR B=1.0 TO 10.0
110   1     FOR A=1.0 TO 15.0
120   2       COLORG 0 A 12 3
130   2       WAIT TIME 150
140   2       NEXT A:NEXT B
160       GOTO 160
```



```
1     REM PROGRAMMA OM ALLE TEKENS TE LATEN ZIEN
5     PRINT CHR$(12)
10    FOR I=J*128+0 TO 127+J*128
20    X=I MOD 16
30    Y=(I-I MOD 16)/16
35    IF J=1 THEN Y=Y-8
40    CURSOR 7+3*X,22-2*Y
50    IF I=12 THEN GOTO 90
60    PRINT CHR$(I)
90    NEXT
100   J=1-J
110   IF GETC<>0 GOTO 5:GOTO 110
```

ROMROUTINES & ENTRYPOINTS SSETM , SMODE : CHANGE MODE

PURPOSE : Changes the MODE of the screen

ENTRY : Mode CODE in register A
CALL : RST 5 + DATA : 18
(see example on p.9, line 304-307)
EXIT : All registers preserved except:
CARRY UNSET : OK
CARRY SET : insufficient room for requested MODE.

CODES :

FF	MODE 0	24 lines X 60 characters
0	MODE 1	65 lines X 72 blobs
1	MODE 1A	65 lines X 72 blobs 53 visible + 4 lines char
2	MODE 2	4 COLOR MODE
3	MODE 2A	
4	MODE 3	130 lines X 160 blobs
5	MODE 3A	130 lines X 160 blobs 106 visible + 4 lines char
6	MODE 4	4 COLOR MODE
7	MODE 4A	
8	MODE 5	256 lines X 336 blobs
9	MODE 5A	256 lines X 336 blobs 212 visible + 4 lines char
A	MODE 6	256 lines X 336 blobs
B	MODE 6A	256 lines X 336 blobs 212 visible + 4 lines char

NOTES : Going from any A-MODE to another A-MODE retains the last 4 lines of text.
Going from an ALL-GRAFIC MODE (1,2,3...) to its corresponding SPLIT MODE (1A,2A,3A...), the bottom area of grafics is preserved on screen above the text area.The rest of the graphics is preserved in a buffer off-screen.It is still possible to draw on this in the split mode.

***é9D holds information about the current screen mode,this can be checked in BASIC programs.You will find here FF for MODE 0,0 for MODE 1, 1 for MODE 1A and so on.

```
PAGE 01    FLASH : CHANGE COLOR 2 (4 COLOR MODE)

002                    *TO START : CHANGE VECTOR 7 TO HEX 305
003                    *TO CHANGE FLASHPERIOD SET PERIO (MIN 2 - MAX FF)
004                    *PERIO = 0 : NO FLASHING, ALWAYS COL0
005                    *PERIO = 1 : NO FLASHING, ALWAYS COL1
006                    COLCNB  EQU   :BFF6
007                    COLCH   EQU   :A
008                    VEC7    EQU   :D9A9
009                            ORG   :300
010 0300 19            PERIO   DATA  :19        X 20 MS = 1/2 FLASHPERIOD
011 0301 05            COL0    DATA  :5
012 0302 0F            COL1    DATA  :F
013 0303 00            TIM     DATA  :0         COUNTER INCR EACH 20 MS
014 0304 00            TFL     DATA  :0         FLAG FOR COLOR 0 OR 1
015 0305 F5                    PUSH  PSW
016 0306 E5                    PUSH  H
017 0307 210303                LXI   H,TIM      ADRES TIM IN H,L
018 030A 3A0003                LDA   PERIO
019 030D FE00                  CPI   :0
020 030F CA2203                JEQ   SET0       ALWAYS COL0
021 0312 FE01                  CPI   :01
022 0314 CA2E03                JEQ   SET1       ALWAYS COL1
023 0317 BE                    CMP   M
024 0318 D23F03                JGE   INCR       NO COL CHANGE IF TIM<=PERIO
025 031B 3A0403                LDA   TFL
026 031E B7                    ORA   A
027 031F CA2E03                JZ    SET1       TEST FLAG AND JUMP TO CHANGE
028 0322 AF            SET0    XRA   A
029 0323 320403                STA   TFL
030 0326 3A0103                LDA   COL0
031 0329 E60F                  ANI   :0F
032 032B C33803                JMP   SETC
033 032E 3E01          SET1    MVI   A,:1
034 0330 320403                STA   TFL
035 0333 3A0203                LDA   COL1
036 0336 E60F                  ANI   :0F
037 0338 C6A0          SETC    ADI   :A0        CONVERT COL TO COLCNTR BYTE
038 033A 32F6BF                STA   COLCNB
039 033D 3601                  MVI   M,:01      PRESET TIM
040 033F 34            INCR    INR   M
041 0340 E1                    POP   H
042 0341 F1                    POP   PSW
043 0342 C3A9D9                JMP   VEC7       START ROM PROGRAM RESTART 7
044 0345                       END

****************************
* S Y M B O L   T A B L E *
****************************

COL0    0301    COL1    0302    COLCH   000A    COLCNB BFF6
INCR    033F    PERIO   0300    SET0    0322    SET1    032E
SETC    0338    TFL     0304    TIM     0303    VEC7    D9A9
```

PRIEM
Problemen
Raadsels
Inzichten
En
Methodes

GETALLEN

prime numbers

1. Definities

- Elk natuurlijk getal dat juist twee verschillende delers heeft is een priemgetal.
- Een natuurlijk getal dat geen priemgetal is, wordt een samengesteld getal genoemd.

Gevolgen: a. 1 is geen priemgetal.
b. 2 is het enige even priemgetal.
c. de twee enige delers van een priemgetal p zijn 1 en p.

2. Belang van de priemgetallen

Elk samengesteld getal is precies op één manier te schrijven als een product van priemgetallen. Men noemt dit ontbinden in priemfactoren. De priemgetallen kunnen bijgevolg als bouwstenen van de natuurlijke getallen worden beschouwd. Dit zou als gevolg kunnen hebben dat vele betrekkingen tussen natuurlijke getallen uitsluitend met priemgetallen moeten kunnen worden uitgedrukt. Dit is echter verre van waar. Tot op heden zitten er in de theorie van de priemgetallen nog vele onopgeloste raadsels, zodat het belang van de priemgetallen in de structuur van de natuurlijke getallen misschien toch moet gerelativeerd worden!?
Aan het einde van dit artikel vermelden we enkele belangrijke onopgeloste "geheimen" van de priemgetallen.

3. Algoritmen om priemgetallen te bepalen

Twee belangrijke algoritmen kunnen gebruikt worden om priemgetallen op te sporen:

a. het algoritme gebaseerd op de zeef van Eratosthenes
b. het algoritme gesteund op het systematisch testen op mogelijke priemdelers

We zullen beide algoritmen algemeen toelichten en laten volgen door een aantal BASIC-programma's, gebaseerd op deze algoritmen. Alle afgedrukte programma's werden ingestuurd door leden van DAInamic. De redactie heeft sommige een weinig aangepast en van een aantal gelijkaardige een equivalent gemaakt.
We danken in dit verband volgende leden (alfabetische orde): H Assink, M.J. Berkx, J Beumers, J Davids, Desausois, T de Vries, K Esveld, F Geuther, G Goethals, L Moentack, J Schepens, H van Cooten en R Verboom. Onze bijzondere felicitaties gaan naar Luc Moentack die ons de snelst lopende uitwerking van één van beide algoritmen toestuurde (11 seconden).

3.1 De Zeef van Eratosthenes voor priemgetallen tot 1000

Dit algoritme is gebaseerd op de successieve eliminatie van alle veelvouden van de priemgetallen tot de vierkantswortel uit 1000.Dat dit laatste voldoende is wordt gestaafd door het feit dat elk samengesteld getal niet groter dan 1000 noodzakelijk een priemdeler moet hebben kleiner dan of gelijk aan de vierkantswortel uit 1000.
Deze eliminatie kan gebeuren door in 1000 (500) adressen een bepaalde code (bv 0) te poken en daarna voor de niet-priemgetallen de adresinhoud te wijzigen. Met de peek-instructie kunnen nadien de gezochte priemgetallen gegenereerd worden.
Alle gepubliceerde programma's werden uitgeprint na een IMP INT, zodat alle variabelen integers zijn, tenzij ze expliciet met de floating-point-notatie (!) staan aangegeven.

Programma 1:

```
5     REM 11 SECONDEN
10    CLEAR 1100:DIM P(250.0):PRINT CHR$(12)
30    INPUT "FROM 1 TILL  ";LS:PRINT
60    GOSUB 500:K=1
90    FOR M=0 TO L-1:PRINT TAB(K);P(M);
100   K=K+5:IF K=56 THEN PRINT :K=1
110   NEXT M:END
500   P(0.0)=2:I=3:L=0:GOSUB 600:RS!=SQR(LS)
510   I=I+2:IF I>RS! THEN GOSUB 700:RETURN
520   R!=SQR(I):FOR J=1 TO 12
530   IF P(J)>R! THEN GOSUB 600:GOTO 510
540   IF (I/P(J))*P(J)=I GOTO 510
550   NEXT J
600   L=L+1:P(L)=I:A=3*I+20000:B=20000+LS:C=2*I
610   FOR D=A TO B STEP C:POKE D,1:NEXT D:RETURN
700   L=1:FOR N=20003 TO B STEP 2
720   IF PEEK(N)<>1 THEN P(L)=N-20000:L=L+1
730   NEXT N:RETURN
```

prime numbers

Commentaar: Dit programma laat toe de priemgetallen te berekenen gelegen tussen 1 en LS , waarbij LS waarden kan aannemen gelegen tussen 3 en 1600. We onderstellen verder in deze commentaar dat LS gelijk is aan 1000. Het programma reserveert 1000 geheugenplaatsen om (eventueel) een niet-priemgetalcode in te stockeren. In de lijnnummers 500 tot 550 worden de priemgetallen tot $\sqrt{1000}$ bepaald en in de subroutine in 600 en 610 worden telkens de onpare veelvouden van deze priemgetallen als kandidaat-priemgetal weggepookt. De routine in 700-730 slaat alle gevonden priemgetallen op in een array P, die tenslotte in de statements (60) 90-110 wordt uitgeprint. Belangrijk voordeel: alle berekende priemgetallen blijven in het geheugen beschikbaar.

Programma 2:

*** 14 SECONDEN ***

```
      REM 14 seconden ***
5        PRINT CHR$(12):PRINT " 2",
10       CLEAR 10000:M=1000:DIM A(1):P=VARPTR(A(1)):N=SQR(M)
20       M=M/2:FOR I=1 TO M:POKE P+I,0:NEXT
30       FOR I=1 TO N:IF PEEK(P+I)=#AB THEN 100:K=2*I+1:PRINT K,:J=I
40         J=J+K:IF J>M THEN 100:POKE P+J,#AB:GOTO 40
100        NEXT:FOR I=N+1 TO M:IF PEEK(P+I)<>#AB THEN PRINT 2*I+1,
110        NEXT:END
```

Commentaar: In 500 opeenvolgende adressen wordt code 0 gepookt; deze 500 adressen corresponderen met de 500 oneven getallen die moeten worden onderzocht; in lijn 40 wordt in de geheugenplaatsen die horen bij de veelvouden van een gevonden priemgetal éAB gezet; lijn 100 test op deze niet-priemgetalcode om de priemgetallen vanaf $\sqrt{1000}$ tot 1000 zelf te genereren. Met deze uitwerking van het algoritme wordt 2 niet als priemgetal bekomen, zodat dit vooraf moet gegeven worden.

Een tweede manier om de eliminatie van de priemveelvouden te bekomen, is gebruik te maken van een array. Elk element van die array kan men laten corresponderen met een oneven getal tussen 0 en 1000. Bij de start van het programma staat elk element van die array op 0. Door systematisch de inhoud van die array-elementen die horen bij de veelvouden van gevonden priemgetallen te wijzigen, elimineert men de niet-priemgetallen.

Programma 3:

```
***  15 SECONDEN
```

```
      REM 15 seconden
5        CLEAR 2500
10       PRINT CHR$(12)
20       DIM P(250,1)
30       PRINT "START!!!":PRINT 2,:FOR K=0 TO 1:FOR I=1 TO 249
40           IF P(I,K)=0 THEN II=I*2+1+K*500:PRINT II,:GOSUB 100
50           NEXT I:NEXT K:PRINT "KLAAR!!!":END
100      IF K=0 THEN FOR J=I TO 249 STEP II:P(J,0)=1:NEXT J
110      JJ=J-250:IF JJ<250 THEN FOR J=JJ TO 249 STEP II:P(J,1)=1:NEXT
C        J
120      RETURN
```

Commentaar: Dit programma illustreert de Zeef van Eratosthenes door middel van een array P. Daar de maximale dimensie van een array-index 255 is,moet een 2-dimensionale matrix gebruikt worden om de 500 oneven getallen voor te stellen. Deze gedwongen constructie eist zeker tijd op bij het elimineren van de veelvouden. Lijn 40 test op de priemgetalcode en in de subroutine (100-120) wordt de inhoud van de array-elementen die horen bij de veelvouden van een gevonden priemgetal op 1 gezet (niet-priemgetalcode). Ook in dit programma wordt 2 niet als priemgetal gevonden, zodat de instructie PRINT 2, vereist is.

Programma 4:

Commentaar: De uitwerking van het algoritme volgens de Zeef is in dit programma analoog met deze in programma 3. Alleen correspondeert hier elk natuurlijk getal met een array-element, zodat nu ook het even priemgetal 2 ontstaat vanuit het algoritme. Dit eist echter tijdens de verwerking heel wat supplementaire testen en bewerkingen, zodat een langere duurtijd voor de verwerking noodzakelijk wordt.

prime numbers

*** 51 SECONDEN ***

```
      REM 51 seconden ***
2         REM dit programma berekent de priemgetallen van 2 tot 1000
3         REM in 51 seconden; er wordt gebruik gemaakt van de
4         REM "zeef van eratosthenes" door uit een rij getallen van
5         REM 1 tot 1000 alle veelvouden van elk gevonden priemgetal
6         REM te verwijderen; de rij getallen vormen de indices van
7         REM de 'array' r; omdat op de dai pc de dimensie van een 'array
C         '
8         REM niet de waarde 1000 mag hebben, is de 'array' 2-dimensionaa
C         l
9         REM gemaakt; hierdoor wordt het programma helaas langzamer.
10        CLEAR 5000:DIM R(3,250)
20        A=250:B=1000
30        S=INT(SQR(1000))
40        FOR T=2 TO B:I=INT((T-1)/A):J=T-A*I:IF R(I,J)=1 THEN 70
50          PRINT T,:IF T>S THEN 70
60          FOR K=T TO B STEP T:I=INT((K-1)/A):J=K-A*I:R(I,J)=1:NEXT K
70          NEXT T:END
```

Algemene opmerkingen : bij het Zeefalgoritme van Eratosthenes

- snel algoritme
- het "moeilijke" in dit algoritme is, indien het zeven gebeurd door middel van een array, het verband creëren tussen de indexwaarden van de 2-dimensionale array en het corresponderende priemgetal.

3.2 Algoritme gesteund op het systematisch testen op mogelijke priemdelers

Dit algoritme steunt op de eigenschap dat een getal N een priemgetal is als het niet deelbaar is door alle priemgetallen kleiner dan of gelijk aan $\sqrt{N}$.
De meest efficiënte vorm van dit algoritme bestaat erin elk oneven getal tussen 3 en 1000 te testen op mogelijke priemdelers, kleiner dan of gelijk aan zijn vierkantswortel. Wordt zulke priemdeler gevonden dan is het onderzochte getal niet priem.
Een moeilijkheid bij de opbouw van dit algoritme is het uittesten op priemde-

lers kleiner dan of gelijk aan de vierkantswortel uit het onderzochte getal. Hoe groter dit getal wordt, hoe meer priemdelers aan bod komen. De meest efficiënte manier hier is tijdens de verwerking een array op te bouwen met alle priemgetallen tot $\sqrt{1000}$. Op deze wijze worden geen overbodige testen en bewerkingen uitgevoerd en wordt de snelste tijd bekomen.
Sommige leden trachten dit probleem "benaderend" op te lossen door alle mogelijke priemdelers vooraf in een array te stoppen en deze dan bij elk priemonderzoek te gebruiken.

Programma 5a:

```
***  28 SECONDEN ***
```

```
      REM 28 seconden ***
5        PRINT CHR$(12)
10       CLEAR 3000:DIM A(15.0)
20       PRINT " 2"," 3",
22       A(2.0)=3:E=3:S=2
25       B!=SQR(1000.0)
30       FOR I=3 TO 999 STEP 2
40         FOR J=2 TO S
50           IF I MOD A(J)=0.0 THEN 100
60           NEXT J
70         IF I<B! THEN A(E)=I:E=E+1:S=S+1
80         PRINT I,
100        NEXT I
110      END
```

Commentaar: In dit programma wordt de array met de priemdelers tot $\sqrt{1000}$ systematisch opgebouwd; dit programma verloopt duidelijk trager indien hier niet beperkt wordt in de array tot de priemgetallen kleiner dan of gelijk aan $\sqrt{1000}$. (programma 5b)

Programma 5b:

```
***  110 SECONDEN ***
```

```
      REM 110 seconden ***
5        PRINT CHR$(12)
10       CLEAR 3000:DIM A(200.0)
20       A(1.0)=2:E=2:S=1:PRINT " 2",
30       FOR I=3 TO 999 STEP 2
40         FOR J=1 TO S
50           IF I MOD A(J)=0.0 THEN 100
60           NEXT J
70         A(E)=I:E=E+1:S=S+1
80         PRINT I,
100        NEXT I
110      END
```

Programma 6:

```
***  47 SECONDEN ***
```

```
      REM 47 seconden ***
2        DIM P!(10.0):P!(0.0)=2.0:P!(1.0)=3.0:P!(2.0)=5.0:P!(3.0)=7.0:P
C        !(4.0)=11.0:P!(5.0)=13.0:P!(6.0)=17.0
3        P!(7.0)=19.0:P!(8.0)=23.0:P!(9.0)=29.0:P!(10.0)=31.0:PRINT "ST
C        ART"
4        PRINT "2";:FOR A!=3.0 TO 1000.0 STEP 2.0:FOR B!=0.0 TO 10.0:IF
C             FRAC(A!/P!(B!))=0.0 THEN 6:NEXT B!:IF CURX>55.0 THEN PRINT
C
5          A=A!:PRINT A;:NEXT A!
6        IF A!=P!(B!) THEN 4:NEXT A!:PRINT " END":END
```

Commentaar: In dit programma worden alle priemgetallen tot $\sqrt{1000}$ in een array P gezet. Alle oneven getallen worden daarna op al deze delers getest. Dit geeft noodzakelijk een langere verwerkingstijd. Een tweede element dat een rol speelt in de langere tijdsduur is het gebruik van de FRAC-operator die floating-point variabelen eist. Deze laatste vragen meer verwerkingstijd dan integers.

Programma 7:

```
***  52 SECONDEN ***
```

```
      REM 52 seconden ***
5        POKE #131,1:PRINT CHR$(12):POKE #131,0:PRINT
10       PRINT "      ***** PRIME NUMBERS FROM 1 TILL 1000 *****"
20       PRINT :PRINT :PRINT "                                EXECUTION TIME = 5
C        2 SEC."
30       PRINT :PRINT "                                (without MATH-chip)":PRINT :
C        PRINT
40       CLEAR 1000
50       DIM P!(167.0)
60       P!(1.0)=3.0:I!=3.0:L!=1.0:PRINT " 2"," 3",
70       I!=I!+2.0:IF L!=167.0 GOTO 120
80       R!=SQR(I!):FOR J!=1.0 TO 167.0
90         IF P!(J!)>R! THEN L!=L!+1.0:P!(L!)=I!:P=I!:PRINT P,:GOTO 70
100        M!=I!/P!(J!):IF INT(M!)=M! GOTO 70
110        NEXT
120      PRINT :PRINT :PRINT "                  ***** TERMINATED *****"
130      PRINT :PRINT "       *** ALL PRIME NUMBERS REMain in memory ***
C        "
```

Commentaar: Programma 7 bouwt de array P met alle oneven priemdelers systematisch op. Voor alle oneven getallen I worden de priemdelers tot $\sqrt{I}$ getest. (lus in lijn 80-110). Als het getal I geen priemgetal is wordt de lus verlaten in lijn 100. Het algoritme eindigt door middel van een test op het totale aantal oneven priemgetallen (167) kleiner dan 1000. Het programma eist dus dat dit aantal vooraf bekend is. De looptijd zal zeker ingekort worden indien overal waar mogelijk met integers gewerkt wordt en indien de statements, die nu alle priemgetallen in het geheugen beschikbaar houden, geëlimineerd worden. Dit laatste is uiteraard inherent verbonden met de doelstelling van elk programma waartoe werd het geschreven ?: snelheid of beschikbaarheid van de resultaten!

De volgende programma's wijken ergens af van vorig algemeen gesteld algoritme. Voor de meeste echter blijft de verwerkingstijd zeer gunstig.

Programma 8:

```
***  35 SECONDEN


      REM 35 seconden
10      PRINT CHR$(12)
20      PRINT TAB(12);"*******PRIEMGETALLEN*******"
30      PRINT 2,
40      N=1
50      FOR N=3 TO 999 STEP 2:D=3
60        IF D*D>N THEN 80:IF N MOD D=0 THEN 90
70        D=D+2:GOTO 60
80        PRINT N,
90        NEXT N
100     END
```

Programma 9:

```
***  47 SECONDEN ***IMP INT


      REM 47 seconden ***imp int
10      PRINT CHR$(12)
20      PRINT " 2",
30      FOR X=3 TO 1000 STEP 2
40        T=3
50        A=SQR(X)
60        IF T>A THEN 100
70        IF X=X/T*T THEN 110
80        T=T+2
90        GOTO 60
100       PRINT X,
110       NEXT X
120     END
```

Programma 10:

```
***  57 SECONDEN ***
```

```
      REM 57 seconden ***
10        PRINT CHR$(12):PRINT "STARTING":PRINT
20        FOR N=2 TO 3:K!=2.0:M!=N/K!:L!=INT(M!):IF L!<>M! OR L!=1.0 THE
C          N PRINT N,:NEXT N
30        FOR N=5 TO 1000 STEP 2:FOR K!=2.0+T! TO SQR(N) STEP 2.0:M!=N/K
C            !:L!=INT(M!):IF M!<>L! THEN NEXT K!:PRINT N,:T!=1.0
40          NEXT N
50        PRINT "FINISHED":END
```

Commentaar:

Programma's 8,9 en 10 wijken af van het algemeen voorgestelde algoritme doordat alle oneven getallen X tot aan $\sqrt{X}$ getest worden op alle mogelijke oneven delers, en niet alleen op de mogelijke priemdelers tot $\sqrt{X}$. Het tijdsverschil tussen deze programma's wordt vooral verklaard door het gebruik van de snelle MOD-operator in programma 8 en de floating-point variabelen in programma 10.

Programma 11:

```
***  NOG MEER PRIEMGETALLEN....***
```

```
      REM nog meer priemgetallen....***
2         DIM S(53.0):FOR K=1 TO 53:READ S(K)
3           G=G+S(K):FOR N=9 TO SQR(G) STEP 2:IF G MOD (N)=0 THEN 3:NEXT
C            N:PRINT G,
4           NEXT K:FOR K=6 TO 53:GOTO 3
5           DATA 2,1,2,2,4,2,4,2,4,6,2,6,4,2,4,6,6,2,6,4,2,6,4,6,8,4,2,4
C           ,2,4,8,6,4,6,2,4,6,2,6,6,4,2,4,6,2,6,4,2,4,2,10,2,10
```

Commentaar: Merkwaardig aan dit programma is de DATA-lijn. Door middel van dit DATA-statement wordt een getallentabel gecreëerd, waaruit alle 2-,3-,5- en 7-vouden zijn verdwenen. Elk getal G dat in de tabel voorkomt moet bijgevolg nog getest worden op priemdelers van 9 tot $\sqrt{G}$. Het programma test in feite alle mogelijke oneven delers van 9 tot $\sqrt{G}$ (lijn 3). Door alle data-elementen van deze tabel in een array in te lezen, kunnen steeds grotere getallen van deze tabel op hun priemeigenschap worden onderzocht. Het programma eindigt dan ook slechts met een foutmelding indien het berekende priemgetal te groot wordt om met een integervariabele te worden voorgesteld. De DATA-gegevens die de eigenlijke tabel zonder 2-,3-,5- en 7-vouden opstellen zijn de laatste 48. Telkens deze doorlopen zijn,

werden deze veelvouden uit 210 (= 2.3.5.7) natuurlijke getallen geëlimineerd.

Een aantal programma's zijn nog verder verwijderd van het basisalgoritme omdat alle (ook de even) getallen worden getest en/of omdat oneven delers worden opgespoord tot N/2, daar waar $\sqrt{N}$ voldoende is.
We publiceren ook hiervan enkele programma's om de lezer te tonen dat in dit geval de verwerkingstijd snel toeneemt. Het al dan niet gebruik maken van integer-variabelen heeft ook in dit geval duidelijk invloed op de duurtijd.

Programma 12:

```
***  447 SECONDEN ***
```

```
      REM 447 seconden ***

5       REM ***** inleiding priemgetallen *****
10      MODE 0:PRINT CHR$(12):COLORT 8 8 0 0
20      CURSOR 15,12:PRINT "PRIEMGETALLEN TOT 1000"
30      FOR Z!=1.0 TO 10.0
40        COLORT 8 0 0 0:WAIT TIME 30
50        COLORT 8 8 0 0:WAIT TIME 20
60        NEXT Z!:COLORT 8 0 0 0:PRINT CHR$(12)

100     REM ***** het programma *****
105     PRINT " 2.0",
110     FOR A!=2.0 TO 1000.0
120       FOR B!=2.0 TO INT(A!/2.0)
130         IF FRAC(A!/B!)=0.0 THEN NEXT A!:END
140       NEXT B!:PRINT A!,:NEXT A!
```

Programma 13:

```
***  277 SECONDEN ***
```

```
     REM 277 seconden ***
2       B!=2.0:PRINT "S":PRINT 2.0,:FOR N!=3.0 TO 1001.0 STEP B!:FOR K
C           !=B!+A! TO N!/B! STEP B!:M!=N!/K!:IF M!<>INT(M!) THEN NEXT:
C         PRINT N!,:A!=1.0:NEXT
3       NEXT N!:PRINT "F"
```

4. Aantal priemgetallen en hun verdeling in de natuurlijke getallen

De verzameling van de priemgetallen is oneindig. Deze stelling werd reeds door Euclides bewezen. De spreiding van de priemgetallen in de verzameling van de natuurlijke getallen wordt nochtans groter naarmate de getallen groter worden. Het eerste Priemgetallen-Theorema beschrijft de asymptotische dichtheid van de priemgetallen als volgt:

$$\lim_{x \to \infty} \frac{\pi(x)}{x/\ln x} = 1$$

Hierin stelt $\pi(x)$ het aantal priemgetallen voor niet groter dan x. Dit theorema werd in 1895 bewezen door J Hadamard en C de la Vallée-Poussin. Het stelt dus dat het aantal priemgetallen kleiner dan of gelijk aan een gegeven getal x, voor zeer grote x-waarden gegeven wordt door $x/\ln x$.

Volgens het tweede Priemgetallen-Theorema kan de asymptotische dichtheid van de priemgetallen beschreven worden door:

$$\lim_{x \to \infty} \frac{\pi(x)}{\int_2^x \frac{dt}{\ln t}} = 1$$

Dit betekent dat voor grote waarden van x het aantal priemgetallen kleiner dan of gelijk aan x gegeven wordt door de bepaalde integraal $\int_2^x dt/\ln t$. Voor de onbepaalde intergraal vindt men volgende ontwikkeling:

$$\int \frac{dt}{\ln t} = \ln(\ln x) + \ln x + \frac{(\ln x)^2}{2.2!} + \frac{(\ln x)^3}{3.3!} + \frac{(\ln x)^4}{4.4!} + \ldots$$

Met een eenvoudig programma kan met de DAI pc deze formule geëvalueerd worden tot exponent 19, zodat we in staat zijn om voor verschillende waarden van x de berekeningen uit te voeren.

In onderstaande tabel hebben we voor een aantal x-waarden $\pi(x)$ opgegeven, en hun respectievelijke benaderde waarden voor $x/\ln x$ en $\int_2^x dt/\ln t$ berekend.

x	$\pi(x)$	$x/\ln x$	$\int_2^x dt/\ln t$
2	1	2,885	0
10	4	4,343	5,120
100	25	21,715	29,081
1000	168	144,765	176,563
1500	239	205,108	246,762

x	$\pi(x)$	x/lnx	$\int_2^x dt/\ln t$
2000	303	263,127	313,751
2500	367	319,528	378,541
3000	430	374,702	441,677
3500	489	428,894	503,501
4000	550	482,273	564,242
4500	610	534,961	624,068
5000	669	587,048	683,102
5500	725	638,606	741,441
6000	783	689,694	799,164
6500	842	740,356	856,328
7000	900	790,633	912,991
7500	950	840,557	969,192
8000	1007	890,155	1024,97
8500	1059	939,453	1080,35
9000	1117	988,470	1135,38
9500	1177	1037,230	1190,06
10000	1229	1085,740	1244,43
15000	1754	1559,932	1773,96
20000	2262	2019,491	2280,98 *
25000	2762	2468,74	2776,37
30000	3245	2910,09	3261,25
35000	3732	3345,09	3737,58
40000	4203	3774,78	4206,73
45000	4675	4199,95	4669,72
50000	5133	4621,17	5127,29
55000	5590	5038,9	5580,05
60000	6057	5453,5	6028,5
70000	6935	6274,51	6913,5
80000	7837	7086,05	7785,92
90000	8713	7889,5	8553,56 **
100000	10095	8685,889	9387,17

* vanaf hier moet exponent 18 in de ontwikkeling worden genomen

** vanaf hier geldt exponent 17

5. Enkele onopgeloste problemen in de theorie der priemgetallen

a. Als een getal p priem is en p+2 eveneens, noemt men dat getallenpaar een

priemgetallentweeling. De gemiddelde afstand tussen twee priemgetallen neemt steeds toe. Priemgetallentweelingen komen bijgevolg ook steeds zeldzamer voor. Houdt dit verschijnsel ooit op of is de verzameling van de priemgetallentweelingen oneindeig ? Is(1000000009649 , 1000000009651)de grootste priemgetallentweeling ?

b. Een tweede belangrijk raadsel is het Vermoeden van Goldbach. Dit kan als volgt geformuleerd worden: elk even getal groter dan 2 is de som van twee priemgetallen. Dit vermoeden is nog niet algemeen bewezen. Toch heeft men nog geen even getal gevonden waarvoor het niet waar is.

6. Een toemaatje: ooit al gehoord van de "gelukkige getallen" ?

Net zoals de priemgetallen kunnen de gelukkige getallen eveneens met een zeefproces worden bepaald. Dit algoritme kan als volgt worden omschreven:

1. schrijf alle natuurlijke getallen op tot 100 (we onderstellen hier dat alle gelukkige getallen tot 100 worden bepaald)
2. elimineer elk tweede getal: op deze wijze verdwijnen alle even getallen. De tabel die overblijft is :

1 3 5 7 9 11 13 15 17 19 21 23 25 27 ....

2. In plaats van nu elk 3-voud te elimineren zoals bij de zeef van Eratosthenes, schrappen we nu elk derde getal. De tabel die ontstaat is dan:

1 3 7 9 13 15 19 21 25 27 ....

3. In deze tabel staat na de 3 het cijfer 7, zodat nu elk zevende getal wordt geschrapt. We bekomen:

1 3 7 9 13 15 21 25 27 31 33 37 43 45...

4. Vervolgens wordt elk negende getal geëlimineerd:

1 3 7 9 13 15 21 25 31 33 37 43 45 49 51 55 63..

5. Na eliminatie van respectievelijk elk 13de,15de en tenslotte elk 21ste getal worden de gelukkige getallen tot 100 bekomen:

1 3 7 9 13 15 21 25 31 33 37 43 49 51

63 67 69 73 75 79 87 93 99

Deze tabel bestaat uit 23 gelukkige getallen, waarvan er 9 priem zijn en 14 samengesteld.

In het door de redactie onderzochte gebied (gelukkige getallen tot 100) is het Vermoeden van Goldbach overdraagbaar naar de gelukkige getallen: d.w.z. elk even getal is de som van twee gelukkige getallen. Is dit steeds zo ?

7. Enkele programma-ideetjes

1. Een programma dat tot 1000 alle priemgetallentweelingen afdrukt.

2. Een programma dat voor alle even getallen groter dan 2 en kleiner dan 1000 de twee priemgetallen afdrukt waarvan het de som is. (Vermoeden van Goldbach)

3. Een programma dat tot 1000 alle gelukkige getallen afdrukt.

4. Een programma dat tot 1000 alle getallen afdrukt die priem en gelukkig zijn.

Alles opsturen naar het gekende redactie-adres!!

8. Slot

Voor die enkelen die na het uittesten van alle programma's uit dit artikel de priemgetallen tot 1000 nog niet uit het hoofd kennen, drukken we ze als besluit af: ***PRIEMGETALLEN TOT 1000***

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97
101	103	107	109	113
127	131	137	139	149
151	157	163	167	173
179	181	191	193	197
199	211	223	227	229
233	239	241	251	257
263	269	271	277	281
283	293	307	311	313
317	331	337	347	349
353	359	367	373	379
383	389	397	401	409
419	421	431	433	439
443	449	457	461	463
467	479	487	491	499
503	509	521	523	541
547	557	563	569	571
577	587	593	599	601
607	613	617	619	631
641	643	647	653	659
661	673	677	683	691
701	709	719	727	733
739	743	751	757	761
769	773	787	797	809
811	821	823	827	829
839	853	857	859	863
877	881	883	887	907
911	919	929	937	941
947	953	967	971	977
983	991	997		

D E D A I A A N D E M O N I T O R

Om te kunnen zien wat zich allemaal afspeelt binnen uw DAI computer, dient deze aangesloten te worden op een televisie toestel. Hiertoe bevindt zich aan de achterzijde van uw DAI een uitgang, welke verbonden moet worden met de antenne ingang van een willekeurige TV.
Het beeld wat dan verschijnt op het scherm is prima, maar kan in veel gevallen met een vrij kleine ingreep verbeterd worden.
Het bij de DAI (en bij veel personal computers) toegepaste principe van signaaloverdracht naar een beeldscherm heeft voor- en nadelen. Als voordeel geldt de eenvoudige aansluiting op vrijwel ieder TV toestel.
Het principe gaat als volgt: De informatie bestemd voor het beeldscherm gaat niet rechtstreeks naar dat beeldscherm maar wordt in de DAI eerst gemengd met een hoogfrequente wisselspanning (= 'moduleren'). Hierdoor wordt het signaal geschikt voor de antenne ingang van een TV.
In de TV gebeurt dan eerst het omgekeerde: het hoogfrequente signaal wordt weggefilterd en de beeldscherminformatie blijft over ('de-moduleren'- zie fig. 1).

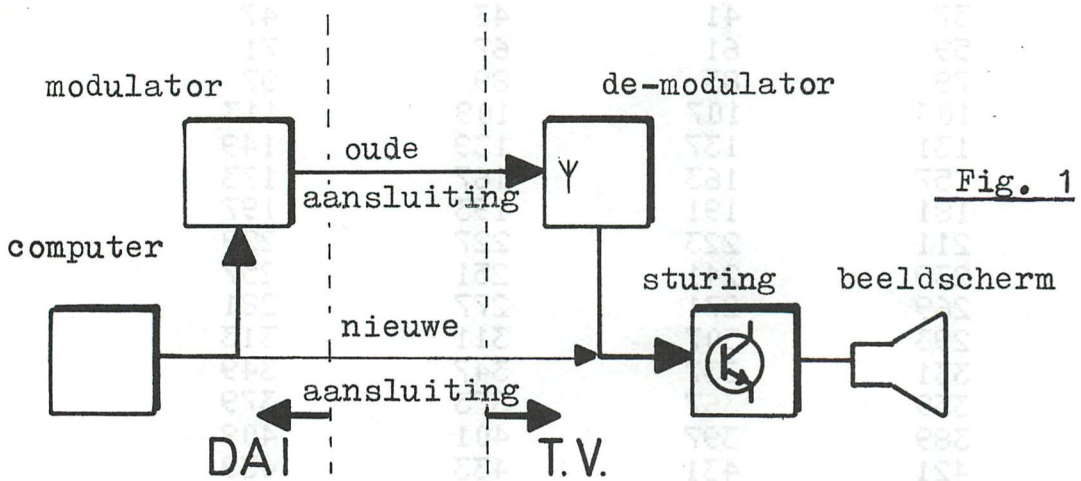


Fig. 1

Als gevolg van deze beide signaalconversies gaat de kwaliteit van het beeldsignaal achteruit, wat uiteindelijk merkbaar is in een iets minder scherp beeld.
Als we nu erin zouden slagen het beeldsignaal rechtstreeks van de DAI naar de beeldbuissturing van een TV te voeren, omzeilen we de beide signaalconversies, en verbetert het

beeld aanzienlijk .
Het probleem nu is: hoe doen we dat. Dit is inderdaad een probleem, want er zijn veel verschillende soorten TV toestellen. Bij b.v. kleuren- en oudere TV's met radiobuizen is het beter om NIET aan ombouw te beginnen.
Het eenvoudigste gaat het ombouwen bij moderne, draagbare zwart/wit toestellen. Belangrijk is het dat de TV een ingebouwde laagspanningsvoeding heeft, welke de 220 Volt voedingsspanning omzet in een lagere spanning en tegelijkertijd de 220 Volt galvanisch scheidt van de rest van de TV.
Verder is het noodzakelijk dat u beschikt over een schema van uw TV ('hoe zit wat waar aangesloten'),dat u dit schema kunt 'lezen' en dat u enige ervaring heeft met elektronica.

Kijk dan eerst op dit schema of de 220 Volt inderdaad omlaag getransformeerd wordt en dat de inwendige voedings(laag)-spanning galvanisch gescheiden is van de netspanning.
Is dit het geval, zoek dan de transistor op, die zorgt voor de intensiteit sturing van het beeldscherm. Meestal is deze als volgt aangesloten (zie pijl in Fig. 2):

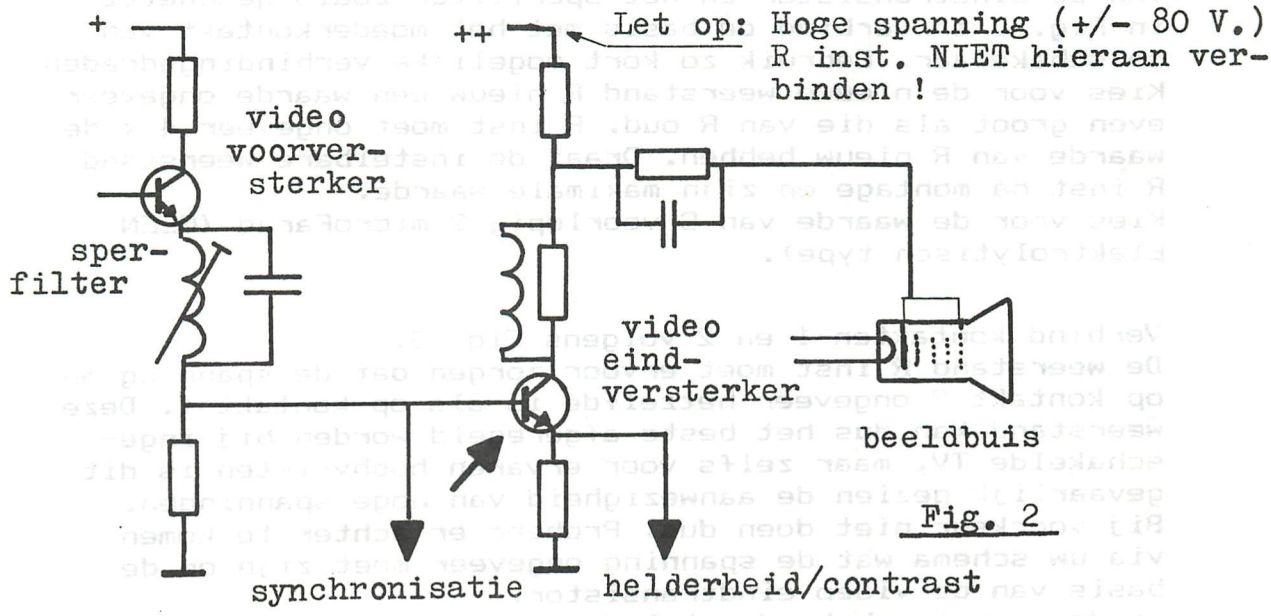
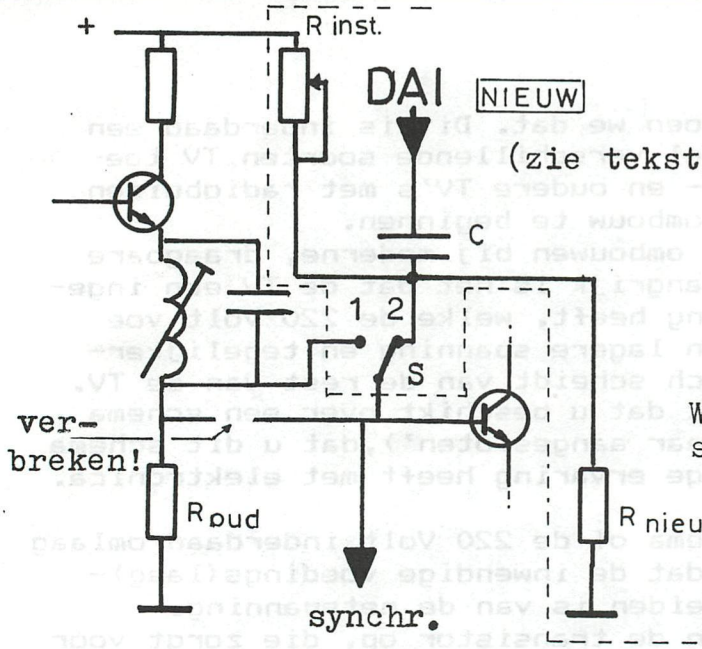


Fig. 2

Heeft u deze gevonden, dan is het karwei al voor een groot deel klaar. Nu volgt het ombouwen van de DAI en de TV.

Eerst de TV. Wijzig de TV schakeling zoals geschetst in fig. 3.

videomonitor-interface



Schakelaar S:
Stand 1: Normaal TV
,, 2: Monitor

Fig. 3

Weerstand Rinst. eerst instellen op maximale waarde!

Monteer aan de achterzijde van de TV een schakelaar met 1 wiselkontakt. Verbreek de verbinding tussen de basis van de eindtransistor en het sperfilter zoals geschetst in Fig. 3 en verbind de basis met het moederkontakt van de schakelaar. Gebruik zo kort mogelijke verbindingsdraden Kies voor de nieuwe weerstand R nieuw een waarde ongeveer even groot als die van R oud. R inst moet ongeveer 4 x de waarde van R nieuw hebben. Draai de instelbare weerstand R inst na montage op zijn maximale waarde.
Kies voor de waarde van C voorlopig 2 microFarad (GEEN Elektrolytisch type).

Verbind kontakten 1 en 2 volgens Fig. 3.
De weerstand R inst moet ervoor zorgen dat de spanning op op kontakt 2 ongeveer hetzelfde is als op kontakt 1. Deze weerstand kan dus het beste afgeregeld worden bij ingeschakelde TV, maar zelfs voor ervaren hobby-isten is dit gevaarlijk gezien de aanwezigheid van hoge spanningen. Bij voorkeur niet doen dus. Probeer er achter te komen via uw schema wat de spanning ongeveer moet zijn op de basis van de video eindtransistor.
Is de spanning bekend, stel dan R inst zo in, dat de deling van de voedingsspanning door R inst en R nieuw de gewenste basisspanning oplevert.
Beter is echter: monteer R inst zo, dat deze met een schroevendraaier of iets dergelijks bereikbaar blijft bij een gesloten TV-kast.
De condensator wordt met een stukje coax kabel verbonden met een extra antenne ingangsbus. Vergeet niet de afscherming van dit stukje coax kabel te verbinden met de voedingsnul. Het beste kunt u de monitor ingang naast de bestaande antenne ingang monteren en de afscherming van de coax kabel verbinden met de afscherming van de oude en nieuwe antenne ingang.

Sluit de TV kast en schakel het apparaat in, nadat u de regelaar voor helderheid op uw toestel zo laag (donker) mogelijk heeft gedraaid. Schakel schakelaar S op 'MONITOR' en draai langzaam de helderheidsregelaar hoger (lichter) totdat de regelaar ongeveer halverwege staat.
Als u R inst al ingesteld had, dan moet nu ongeveer de normale helderheid op uw scherm zichtbaar zijn. Is dit niet het geval dan moet R inst bij te donker beeld iets kleiner gemaakt worden, bij te licht beeld iets groter.
Staat R inst nog steeds maximaal, draai dan R inst LANGZAAM in waarde terug tot de juiste helderheid is bereikt.
Blijft het beeld donker nadat R inst ongeveer tot 1/4 van zijn maximale waarde is teruggedraaid, vervang R inst dan door een andere die 1/3 is van de oude R inst waarde en monteer in serie hiermee een weerstand van 1/10 * R nieuw.
Begin dan weer met R inst maximaal.
Schakel nu vervolgens S over op 'NORMAAL TV' en daarbij mag de intensiteit niet erg veel veranderen.

Vervolgens de DAI. Verwijder de bovenste helft van kast door eerst de vier zwarte dopjes aan de zijkant te verwijderen.
LET OP: Als u dit doet binnen zes maanden na aankoop, vervalt de garantie op uw computer !
Links achter boven het moederbord bevindt zich de TV modulator op een klein printje.
Links achter op dit printje bevindt zich de eigenlijke modulator, die eruit ziet als een blikken doosje van 4x2,5 cm.
In de linkerzijde van dit doosje verdwijnen twee draadjes.
De voorste is de voeding, en de achterste (het dichsts bij de antenne uitgang) vervoert de beeldinformatie.
Dit draadje nu moet verbonden worden met de monitoringang die we net gemaakt hebben aan de TV (Fig. 4).

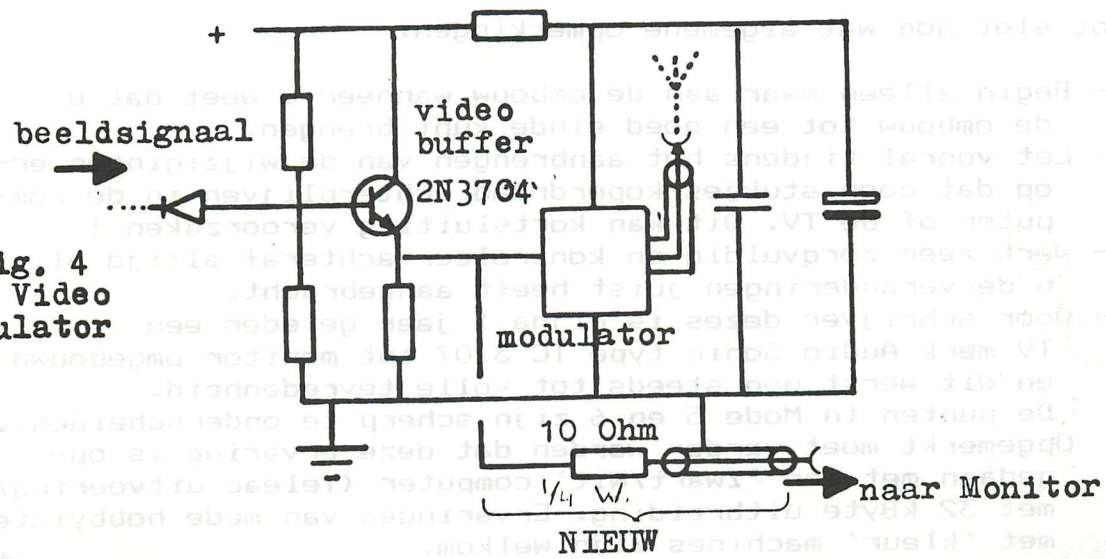


Fig. 4
DAI Video
modulator

Het beste kunnen we een stukje coax kabel solderen aan dit draadje via een weerstand van 10 Ohm en de afscherming van deze kabel verbinden met het blikken huis van de modulator. Voer deze coax kabel via de achterzijde of onderzijde naar buiten en knip de kabel op de gewenste lengte af.
Aan deze zijde monteren we dan een contrastekker (zoals die van de modulator) zodat de meegeleverde verbindingskabel tussen DAI en TV gebruikt kan blijven worden.

Dan nu het grote moment: Verbind de DAI met de monitor en schakel beide apparaten in. Nu moet het u bekende plaatje verschijnen.
Is dit niet het geval, dan zal waarschijnlijk de belasting van de TV op de DAI te groot zijn, of condensator C is te klein gekozen. In het laatste geval is dat te zien aan het niet egaal van 'kleur' (helderheid) zijn van het beeld. Geprobeerd kan worden of een grotere waarde van condensator C verbetering brengt, maar let bij het vervangen van C door een elektrolytisch type op de polariteit.
De gelijkspanning op de DAI uitgang is ongeveer 2,4 Volt en de spanning op de basis van de video eindtransistor in de TV moet u meten of berekenen aan de hand van de waarden R inst, R en de voedingsspanning in de TV.
Mocht een grotere capaciteit van C (tot ongeveer 100 micro Farad) geen oplossing brengen, dan kunt u nog proberen om de modulator in de DAI uit te schakelen door het signaal draadje waarvan we de beeldinformatie aftakken door te knippen. De oorspronkelijke TV uitgang van de DAI werkt dan echter niet meer !

Wanneer ook dit niet helpt en u weet zeker dat alles goed is aangesloten, wijzig dan zowel R als R inst in de TV. Maak beide 2 x zo groot, regel R inst weer af zoals beschreven en probeer het opnieuw.
Als dit tenslotte niet wil werken, dan moet u gaan denken aan een extra buffer (aanpassings schakeling) en het zou te ver voeren om deze oplossing hier te bespreken..

Tot slot nog wat algemene opmerkingen:

- Begin alleen maar aan de ombouw wanneer u weet dat u de ombouw tot een goed einde kunt brengen.
- Let vooral tijdens het aanbrengen van de wijzigingen erop dat geen stukjes koperdraad achterblijven in de computer of de TV. Dit kan kortsluiting veroorzaken !
- Werk zeer zorgvuldig en kontroleer achteraf altijd of u de veranderingen juist heeft aangebracht.
- Door schrijver dezes is bijna 1 jaar geleden een TV merk Audio Sonic type TC 3107 tot monitor omgebouwd en dit werkt nog steeds tot volle tevredenheid. De punten in Mode 5 en 6 zijn scherp te onderscheiden... Opgemerkt moet verder worden dat deze ervaring is opgedaan met een 'Zwart/Wit' computer (Teleac uitvoering) met 32 kByte uitbreiding. Ervaringen van mede hobbyisten met 'kleur' machines zijn welkom.
- Last but not least: Door bovengenoemde wijziging verdwijnt het geluid van de TV in de stand Monitor.

** S U C C E S **

```
20   PRINT :POKE #FF06,#E:PRINT "TIMING OF MATH-FUNCTIONS (WITH AMD 9511)"
22   PRINT :POKE #94C,0:POKE #A32,0
25   FOR N=0.0 TO 79.0:PRINT CHR$(166);:NEXT:PRINT
30   CLEAR 1000:DIM TA(38.0,2.0),M1%(2.0),S1%(2.0),MS1%(2.0),MAX(2.0):I1%=75648:I2%=958473:F1=867.565:F2=0.96857
100  FOR N=0.0 TO 37.0
110  POKE #94C,N:POKE #A32,N
120  W=0.0
123  IF W=1.0 THEN POKE #D4,#7B
124  IF W=0 THEN POKE #D4,0
150  FOR I%=0 TO 4:POKE #3B9+I%,0:NEXT
160  CALLM #340
170  FOR J=1.0 TO 2000.0
175  GOSUB 293
180  NEXT J
190  CALLM #350
200  IF N=0 GOTO 4000
202  M1%(W)=PEEK(#3BD)*10+PEEK(#3BC):S1%(W)=PEEK(#3BB)*10+PEEK(#3BA):MS1%(W)=PEEK(#3B9)*20
203  TA(N,W)=1000.0*((M1%(W)-MZ)*60.0+(S1%(W)-SZ))+MS1%(W)-MSZ
204  W=W+1.0:IF W<2.0 THEN 123
205  NNNZ=INT(N):POKE #FF06,#E:PRINT NNNZ
210  LIST 293
230  PRINT "EXECUTION TIME WITH      :";M1%(1.0);" MIN",S1%(1.0);" SEC",MS1%(1.0);" MSEC"
232  PRINT "EXECUTION TIME WITHOUT   :";M1%(0.0);" MIN",S1%(0.0);" SEC",MS1%(0.0);" MSEC"
235  PRINT "CALCULATION TIME WITH    :";1000.0*((M1%(1.0)-MZ)*60.0+(S1%(1.0)-SZ))+MS1%(1.0)-MSZ;" MSEC"
237  PRINT "CALCULATION TIME WITHOUT :";1000.0*((M1%(0.0)-MZ)*60.0+(S1%(0.0)-SZ))+MS1%(0.0)-MSZ;" MSEC"
245  FOR NN=0.0 TO 79.0:PRINT "-";:NEXT
247  PRINT
250  NEXT N
251  GOTO 5000
252  REM ************************************************
256  RETURN
257  X=J:RETURN
258  X%=J:RETURN
259  X=I1%:RETURN
260  X%=1:RETURN
261  X=1.0:RETURN
262  X%=I1%:RETURN
263  I3%=I1%+I2%:RETURN
264  I3%=I1%-I2%:RETURN
265  I3%=I1%/I2%:RETURN
266  I3%=I1%*1000:RETURN
267  F=J+J:RETURN
268  F=J-J:RETURN
269  F=J/J:RETURN
270  F=J*J:RETURN
271  F=J^2.0:RETURN
272  F=SQR(J):RETURN
273  F=J^0.5:RETURN
274  F=SIN(J):RETURN
275  F=COS(J):RETURN
276  F=TAN(J):RETURN
277  F=ASIN(F2):RETURN
278  F=ACOS(F2):RETURN
279  F=ATN(J):RETURN
280  F=ALOG(F2):RETURN
281  F=EXP(F2):RETURN
282  F=LOG(J):RETURN
283  F=LOGT(J):RETURN
284  F=FRAC(F1):RETURN
285  F=INT(F1):RETURN
286  F=SGN(J):RETURN
287  F=ABS(J):RETURN
288  F=FREQ(I1%):RETURN
289  F=I1% IAND I2%:RETURN
290  F=I1% SHL 3.0:RETURN
291  F=I1% MOD 7.0:RETURN
292  F=PI:RETURN
```

TIMING OF MATH-FUNCTIONS

WITH AMD 9511

uit het programma "FUNCTIE-PLOT" van H.BAKKER

```
3999 REM ##############################################
4000 MZ=PEEK(#3BD)*10+PEEK(#3BC):SZ=PEEK(#3BB)*10+PEEK(#3BA):MSZ=PEEK(#3B9)*20
4010 PRINT ">>>>>> LOOP-TIME :",MZ;" MIN",SZ;"SEC",MSZ;"MSEC":PRINT
4020 GOTO 202
5000 W=0.0
5004 KK=(TA(37.0,W)-TA(0.0,W))/37.0:TAA=TA(0.0,W)
5005 FOR NZ=0 TO 37:TA(NZ,W)=TA(NZ,W)-(KK*NZ+TAA):NEXT
5007 MAX(W)=TA(0.0,W):FOR NZ=0 TO 37:IF TA(NZ,W)>MAX(W) THEN MAX(W)=TA(NZ,W)
5010 NEXT:A$="WITH":IF W=0.0 THEN A$=A$+"OUT"
5015 PRINT :POKE #FF06,#E:PRINT A$:PRINT
5020 K=76.0/MAX(W)
5030 FOR NZ=0 TO 37
5040 IF NZ<10 THEN PRINT " ";
5042 IF NZ<100.0 THEN PRINT " ";
5044 PRINT NZ;
5050 FOR JJ=0.0 TO INT(TA(NZ,W)*K):PRINT CHR$(166);:NEXT:PRINT
5060 NEXT NZ:W=W+1.0:IF W<2.0 THEN 5004
5070 PRINT :POKE #FF06,#E:PRINT "CALCULATION TIMES WITH CORRECTION:":PRINT
5072 PRINT "============================================================"
5075 PRINT "!  N    TIME        %        TIME        %       RATIO !"
5076 PRINT "!       WITH      WITH      WITHOUT     WITHOUT     (%)  !"
5077 PRINT "============================================================"
5080 PRINT :FOR NZ=0 TO 37
5081 PROC1=100.0/MAX(1.0):PROC0=100.0/MAX(0.0)
5082 TW=(INT(TA(NZ,1.0)*1000.0))/1000.0:PRW=(INT(TA(NZ,1.0)*PROC1*1000.0))/1000.0
5083 TN=(INT(TA(NZ,0.0)*1000.0))/1000.0:PRN=(INT(TA(NZ,0.0)*PROC0*1000.0))/1000.0
5087 VERHZ=INT((TA(NZ,0.0)+1E-5)/(TA(NZ,1.0)+1E-5)*100.0+0.5)
5088 IF NZ<10.0 THEN PRINT " ";
5089 IF NZ<100.0 THEN PRINT " ";
5090 PRINT NZ;TAB(7);TW;TAB(18);PRW;TAB(29);TN;TAB(40);PRN;TAB(51);VERHZ
5095 PRINT
5100 NEXT
```

START/STOP VIA INTERRUPT 7

```
PAGE 01

001                    *deze  routine verzorgt START/STOP
002                    *van  BASIC & machinetaal programm
003                    *besturimg met EVENT 1 (ANI 32)
004                    *             of EVENT 2 (ANI 16)
005                    *zet interrupt 7 op #300 door:
006                    *V7D9A9-300  (=D9A9 vervangen door  300)
007                           ORG   :300
008 0300 F3                   DI
009 0301 F5                   PUSH  PSW
010 0302 3A00FD        TEST   LDA   :FD00      input port
011 0305 E620                 ANI   32          maskeer bit 5 (of 4)
012 0307 C20203               JNZ   TEST        indien ingedrukt:loop TEST
013 030A F1                   POP   PSW
014 030B FB                   EI
015 030C C3A9D9               JMP   :D9A9       vevolg interrupt 7 procedure
016 030F                      END

****************************
* S Y M B O L   T A B L E *
****************************

TEST    0302
```

VERGELIJKENDE STUDIE OVER DE MATHCHIP AMD 9511

Door zijn grote verscheidenheid aan mathematische ofte wiskundige functies is DAIpc al een grote uitblinker.
We kunnen ons echter de vraag stellen hoe het met de uitvoeringstijden van de verschillende functies zit. Een ander belangrijk probleem voor vele personal-computers-gebruikers stelt het al dan niet aankopen van de oh zo dure AMD 9511 "MATHCHIP".
Krijgen we waar voor ons geld? Is de relatief grote som voor de aankoop van deze mathchip wel verantwoord voor een of andere toepassing? Over deze en andere gelijkaardige vragen zullen we proberen wat meer duidelijkheid te brengen.
Daarom heb ik een BASIC-programma geschreven dat de looptijden van de verschillende BASIC-functies en sommige BASIC-statements nagaat.
De tijden worden achtereenvolgens gemeten met en zonder MATHCHIP.
nb: Indien men een MATHCHIP in het toestel heeft kan men die disabelen of inactief maken door POKE éD4,0.
actief door POKE éD4,é7B.

Toelichting bij het programma:

30 Initialisatie van de veranderlijken
Ik laat U nu raden wat lijn 22 juist doet.
100 HOOFD-FOR-NEXT lus.Er worden 37 functies getest.
110 wat doen deze POKES hier ?
120 W=1: met MATHCHIP W=0: zonder
124 zet MATHCHIP aan of uit.
160 reset de chronometer en start hem (CALLMé340)
190 voer de functie 2000 maal uit en stop chrono.
203 bereken de tijden en sla deze op in een tabel (TA).
247 druk de resultaten af voor elke functie.
251 indien de 37 functies werden getimed ga dan naar 5000.
293 hier bevinden zich de te testen functies.
4000 deze routine wordt aangeroepen voor de 0-functie ttz een RETURN zonder meer. De bedoeling is dat de tijd benodigd voor het uitvoeren van de lus wordt afgetrokken van de BRUTO-tijd.Doch er volgen complicaties !!
5000 deze routine gaat alle tijden statistisch vergelijken zodat een zinvolle vergelijking mogelijk wordt.
5050 hier wordt een BARGRAPH afgedrukt.
5070 Nu de complicaties...
functie 37 is ook een RETURN en wel om de volgende reden:
De computer heeft wat tijd nodig om een bepaalde subroutine te vinden en hoe verder de subroutine in het geheugen zit, hoe langer deze zoektijd duurt.
Ik heb verondersteld dat de zoektijd lineair toeneemt.
Dit lijkt me een redelijke veronderstelling vermits de verschillende subroutines ongeveer even lang zijn.
Er wordt daarom geinterpoleerd tussen de 2 RETURN-tijden deze tijd wordt dan afgetrokken van de looptijd.
Dit geeft dan TIME WITH en TIME without in de tabel.
Vervolgens worden deze tijden procentueel vergeleken met de traagste functie met en zonder MATHCHIP.
Om te besluiten wordt de onderlinge verhouding berekend.

U vraagt zich misschien nog steeds af wat lijnen 22 en 110 doen?
En hoe komt het dat elke lijn apart gelist wordt?
En waarom geen ON GOSUB maar GOSUB met een lijnnummer?
Het antwoord is zeer eenvoudig: Het programma verandert zichzelf.
Na enig zoekwerk (lees PEEK&POKEwerk) werden de bytes gevonden die de geliste regelnummer en de subroutine waarnaar gesprongen wordt bepalen. Aldus is het mogelijk deze tijdens de loop van programma aan te passen.Misschien mogen we hier wel spreken van "SELF-MODIFYING-CODE".
Het werkt in ieder geval prima en is volgens mij de beste manier om een zo constant mogelijke looptime te creeren.
Een belangrijk gevolg van deze manier van werken is echter dat het een uiterst delicate zaak wordt om het programma te wijzigen.
Tevens wil ik de collega's die het programma wensen in te tikken waarschuwen:In ieder geval SAVEN voor RUN !!
Ook niet vergeten de HEAPPOINTERS te veranderen!
Het machinetaalprogramma start op é300 en loopt tot é3B9.
Op te merken zijn de twee routines op é 340 en é 350 om de chrono te starten en te stoppen.Een dergelijke methode is ook nodig om de real-time clock uit DAInamic N3 -waarvan de chrono is afgeleid-, te starten en te stoppen.

```
5     REM ROTERENDE ELLIPSEN  J.BEUMERS
6     REM Dit programma roteert een ellips over 15 gr,30 gr....
10    MODE 6:COLORG 0 5 10 15
20    CLEAR 3000
30    DIM A(250.0),B(250.0)
40    FOR X=0.0 TO 2.0*PI STEP PI/125.0
50    A(N)=120.0*COS(X):B(N)=30.0*SIN(X)
60    N=N+1.0:NEXT
70    FOR FI=0.0 TO PI*11.0/12.0 STEP PI/12.0
80    P=SIN(FI):Q=COS(FI)
90    FOR N=0.0 TO 249.0
100   DOT Q*A(N)-P*B(N)+167.0,P*A(N)+Q*B(N)+127.0 10:NEXT
110   NEXT:DOT 167,127 15
120   GOTO 120
```

```
PAGE 01     :CLOCK FOR MATH TEST

002                     *339 = UNITS OF 20 MSEC
003                     *33A = UNITS OF SEC
004                     *33B = TENS OF SEC
005                     *33C = UNITS OF MIN
006                     *33D = TENS OF MIN
007                            ORG   :300
008 0300 E5                    PUSH  H
009 0301 D5                    PUSH  D
010 0302 F5                    PUSH  PSW
011 0303 213903                LXI   H,:339    20 MSEC
012 0306 1E32                  MVI   E,:32
013 0308 34                    INR   M
014 0309 7B                    MOV   A,E
015 030A BE                    CMP   M
016 030B C23103                JNZ   EXIT1
017 030E C5                    PUSH  B
018 030F 060A                  MVI   B,:A
019 0311 0E06                  MVI   C,:6
020 0313 1600                  MVI   D,:0
021 0315 72                    MOV   M,D
022 0316 23                    INX   H         SEC
023 0317 34                    INR   M
024 0318 78                    MOV   A,B
025 0319 BE                    CMP   M
026 031A C23003                JNZ   EXIT2
027 031D 72                    MOV   M,D
028 031E 23                    INX   H         TENS OF SEC
029 031F 34                    INR   M
030 0320 79                    MOV   A,C
031 0321 BE                    CMP   M
032 0322 C23003                JNZ   EXIT2
033 0325 72                    MOV   M,D
034 0326 23                    INX   H         MINUTES
035 0327 34                    INR   M
036 0328 78                    MOV   A,B
037 0329 BE                    CMP   M
038 032A C23003                JNZ   EXIT2
039 032D 72                    MOV   M,D
040 032E 23                    INX   H         TENS MIN
041 032F 34                    INR   M
042 0330 C1            EXIT2   POP   B
043 0331 F1            EXIT1   POP   PSW
044 0332 D1                    POP   D
045 0333 E1                    POP   H
046 0334 C3A9D9                JMP   :D9A9     CONT INTERRUPT PROC
047                            ORG   :340      START TIMER
048 0340 F3                    DI
049 0341 F5                    PUSH  PSW
050 0342 3E00                  MVI   A,:0      CHANGE VECTOR
051 0344 327000                STA   :70       TO OUR ROUTINE
052 0347 3E03                  MVI   A,:3
053 0349 327100                STA   :71
054 034C F1                    POP   PSW
```

```
PAGE 02     :CLOCK FOR MATH TEST

055 034D FB                     EI
056 034E C9                     RET
057                             ORG    :350        STOP TIMER
058 0350 F3                     DI
059 0351 F5                     PUSH   PSW
060 0352 3EA9                   MVI    A,:A9       OLD VECTOR
061 0354 327000                 STA    :70
062 0357 3ED9                   MVI    A,:D9
063 0359 327100                 STA    :71
064 035C F1                     POP    PSW
065 035D FB                     EI
066 035E C9                     RET
067 035F              END       END

****************************
* S Y M B O L   T A B L E *
****************************

END     035F    EXIT1   0331    EXIT2  0330


]#P.
0300 E5 D5 F5 21 39 03 1E 32 34 7B BE C2 31 03 C5 06
0310 0A 0E 06 16 00 72 23 34 78 BE C2 30 03 72 23 34
0320 79 BE C2 30 03 72 23 34 78 BE C2 30 03 72 23 34
0330 C1 F1 D1 E1 C3 A9 D9

0340 F3 F5 3E 00 32 70 00 3E 03 32 71 00 F1 FB C9

0350 F3 F5 3E A9 32 70 00 3E D9 32 71 00 F1 FB C9
```

```
5     FOR I=0 TO 4:POKE #339+I,0:NEXT
10    INPUT "START";S$
20    CALLM #340:REM START
30    A=GETC:IF A=0 THEN 30
40    CALLM #350:REM STOP
50    PRINT PEEK(#33D)*10+PEEK(#33C),PEEK(#33B)*10+PEEK(#33A),PEEK(#339)*20
55    REM #3BD-#3BA IN THE TEST PROGRAM
60    GOTO 5
*
```

```
63000 REM SCREEN COPY ON EPSON MX-80  (MODES 1,2,3,4)
63002 POKE #131,1
63005 INPUT " WELKE BACKGROUNDKLEUR ";BG:PRINT :POKE #131,0
63007 PRINT CHR$(27);"E"
63008 FOR Y=YMAX-3 TO 0 STEP -3
63010 FOR X=0.0 TO XMAX-2 STEP 2
63020 V=0
63030 IF SCRN(X,Y+2)<>BG THEN V=V+1
63040 IF SCRN(X+1,Y+2)<>BG THEN V=V+2
63050 IF SCRN(X,Y+1)<>BG THEN V=V+4
63060 IF SCRN(X+1,Y+1)<>BG THEN V=V+8
63070 IF SCRN(X,Y)<>BG THEN V=V+16
63080 IF SCRN(X+1,Y)<>BG THEN V=V+32
63090 PRINT CHR$(160+V);:NEXT:PRINT :NEXT
*
```

PADDELEN MET FGT

```
T-1000
100  MODE 0:PRINT CHR$(12)
105   PRINT "PADDELEN MET FGT ...":FOR X=1 TO 59:PRINT CHR$(255);:NEXT
107   PRINT :PRINT
110   LIST 280-380
120   G=GETC:IF G=0 THEN 120
130   GOTO 500
280   REM "HUIS    a   VLIEGTUIG  l"
290   REM "KERK    b   HELIKOPTER m"
300   REM "HEK     c   U.F.O      n"
310   REM "BOOM    d   VOGEL      o"
320   REM "MAN     e   FIETS      p"
330   REM "VROUW   f   HOND       q"
340   REM "AUTO    g   LADDER     r"
350   REM "TREIN   h   VIS        s"
360   REM "CAMION  i   LANTAARN   t"
370   REM "AUTOBUS j   POPULIER   u"
380   REM "FABRIEK k   HALVE MAAN v"
400   REM ***SUBROUTINE FGT
410   SOUND 1 0 15 0 FREQ(440):POKE #2F2,X MOD 256:POKE #2F3,X/256:POKE #2F4,Y
420   POKE #2F5,SP:POKE #2F6,ID
430   C=FC**40+CC:F=FF**80+PF**40+ZF**20+VF**10+DF:POKE #2F0,C:POKE #2F1,F
440   CALLM #300,A$:SOUND OFF
450   RETURN
500   MODE 4:COLOR6 0 15 15 15
505   X=10:Y=10:CC=15:PF=0:DF=0:VF=0:ZF=0:G=ASC("A")
510   A=PDL(1):B=PDL(2)
520   C=SCRN(A,B):DOT A,B 15:WAIT TIME 3:DOT A,B C
525   REM kies nieuwe tekening met keyboard
530   G=GETC:IF G<>0 THEN IF G<20 THEN 550:VF=0:ZF=0:A$=CHR$(G):FILL 0,0 15,15 0:X=0:Y=0:GOSUB 400
532   REM plaats nieuwe tekening met event
535   IF PEEK(#FD00) IAND 48<>0 THEN X=A:Y=B:GOSUB 400
540   GOTO 510
550   REM aanpassing van ZF en VF met CURSOR^toetsen
560   IF G=16 THEN VF=0
570   IF G=17 THEN VF=1
580   IF G=18 THEN ZF=1
590   IF G=19 THEN ZF=0
595   IF G=9 THEN FILL A,B A+15,B+15 0:REM wis met TAB
600   GOTO 510
1000  REM Bij OFF SCREEN : MODE 4: RUN 510
```

EXACTE TIMER +++ TIME OUT +++ ANTWOORDTIJD +++ REACTIEMETING

Adressen é1BE (LOW) en é1BF(HIGH) bevatten een timerteller.
Deze constructie (eerst LOW BYTE dan HIGH BYTE) is typisch voor 8080 microprocessor.We kunnen duidelijk de splitsing HIGH/LOW vaststellen als we van een getal de HEX& vragen.

```
vb:    PRINT HEX&(65535) :éFFFF   HB= FF   LB=FF
       PRINT HEX&(255)   :éFF     HB= Ø    LB=FF
       PRINT HEX&(1000)  :é3E8    HB= 3    LB=E8
```

De benadering via HEX& is echter niet de eenvoudigste manier om de beide delen van het getal weg te POKEN.
De LOW BYTE kunnen we ook bekomen door volgende constructies:

```
GETAL MOD é100
GETAL MOD 256
GETAL IAND 255
```

De HIGH BYTE komt te voorschijn op de volgende manieren:

```
GETAL SHR 8
INT(GETAL/256)
```

Op interruptbasis wordt elke 20 milliseconden deze timer met 1 verminderd, totdat hij de waarde Ø bereikt.(é1BE en é1BF beiden Ø)
Deze Ø-inhoud blijft behouden totdat een nieuwe waarde in de teller wordt geplaatst.
Doel van deze teller is de generatie van tijdseenheden voor de WAIT TIME instructie van BASIC.
In het programma wordt de waarde eerst vermenigvuldigd met 50 zodat we de vertraging kunnen opgeven in seconden.
Deze waarde wordt dan verdeeld over de twee teller-bytes volgens bovenstaande beschrijving.
Het programma gaat nu in een loop totdat de beide adressen Ø bevatten.
De uitvoeringstijden van het programma zijn niet belangrijk omdat de timing constant gebeurt door de interrupt-werking.
OPMERKING: Tijdens een actieve cassetteroutine (dus als de cursor niet meer flikkert) worden alle interrupts genegeerd en gaat het decrementeren van de teller niet verder.
Meteen is ook duidelijk waarom de maximum WAIT TIME waarde 65535 is: dit is de maximale waarde die we kwijt kunnen over 2 bytes.
Bijgaande programma's maken duidelijk hoe we deze timer-techniek in BASIC-programma's kunnen gebruiken.

```
5     REM TIME OUT BEWAKING MET INTERRUPT-TELLER
10    TIMEMAX=5:REM IN SEK
20    T=TIMEMAX*50:POKE #1BE,T MOD #100:POKE #1BF,T SHR 8:REM T IN x20 MS
30    G=GETC
40    IF PEEK(#1BE)=0 AND PEEK(#1BF)=0 THEN 100:REM TEST
50    IF G=#D GOTO 80:REM car ret,nieuwe string,restart
60    IF G<>0 THEN PRINT CHR$(G);
70    GOTO 30
80    PRINT :GOTO 10
100   SOUND 1 0 15 0 FREQ(1000):WAIT TIME 5:SOUND OFF :PRINT "  TIME OUT"
110   GOTO 10
```

```
5     REM BEPALING VAN RESPONSTIJD
10    T1=#FFFF:REM MAXIMUMTIJD IN EENHEDEN VAN 20 MS (21 MIN)
20    POKE #1BE,#FF:POKE #1BF,#FF
30    INPUT A$
40    T2=PEEK(#1BE) IOR (PEEK(#1BF) SHL 8):REM NIEUWE STAND
50    IF T2=0 THEN PRINT " OUT OF TIME ....":GOTO 20
60    PRINT " U GEBRUIKTE ";(T1-T2)/50.0;" SEC VOOR HET ANTWOORD"
70    GOTO 10
```

```
10    REM REAL TIME CLOCK (maximaal [ 65535/50 ] seconden = 21 minuten)
20    POKE #1BE,#FF:POKE #1BF,#FF:REM start clock.
30    REM De verlopen tijd sinds start clock is steeds :
40    PRINT (#FFFF-(PEEK(#1BF)*256)-PEEK(#1BE))/50
50    GOTO 40
60    REM Na 21 minuten is de tijd niet meer significant !!!
70    REM U kan steeds opnieuw starten door lijn 20 uit te voeren.
80    REM GEEN WAIT TIME uitvoeren ondertussen !!!
90    REM U kan WAIT TIME simuleren door een dummy lus:
100   REM vb: FOR X=1 TO 10000:NEXT
```

x-bus layout

PIN-OUT OF THE 50-PENS CONNECTOR INSIDE DAIpc

1	GROUND	SCREENING LINE
2	D0	DATA BIT 0
3	GROUND	SCREENING LINE
4	D1	DATA BIT 1
5	GROUND	SCREENING LINE
6	D2	DATA BIT 2
7	GROUND	SCREENING LINE
8	D3	DATA BIT 3
9	GROUND	SCREENING LINE
10	D4	DATA BIT 4
11	GROUND	SCREENING LINE
12	D5	DATA BIT 5
13	GROUND	SCREENING LINE
14	D6	DATA BIT 6
15	GROUND	SCREENING LINE
16	D7	DATA BIT 7
17	GROUND	SCREENING LINE
18	-	NO CONNECTION
19	GROUND	SCREENING LINE
20	MEMW	MEMORY WRITE STROBE
21	GROUND	SCREENING LINE
22	-	NO CONNECTION
23	A10	ADDRESS LINE 10
24	MEMR	MEMORY READ STROBE
25	A14	ADDRESS LINE 14
26	A11	11
27	A12	12
28	A13	13
29	A9	9
30	A15	15
31	A7	7
32	A8	8
33	A5	5
34	A6	6
35	A3	3
36	A4	4
37	A1	1
38	A2	2
39	A0	0
40	INTA	INTERRUPT ACKNOWLEDGE
41	CS LOW ROM	CHIP SELECT LOWER ROM
42	CS LB UPP ROM	CHIP SELECT LOWER BANK UPPER ROM
43	PSEUDE A12	A12 AFTER START-LOGIC
44	CS UB UPP ROM	CHIP SELECT UPPER BANK UPPER ROM
45	+5V	5 VOLT
46	+5V	5 VOLT
47	RAMOP	NOT RAM OPERATION
48	CK2	TTL LEVEL CLOCK (2MHz)
49	HOLD	HOLD REQUEST
50	SYNC	CPU SYNC SIGNAL

DAS GESETZ DER GROSSEN ZAHL A. MEYSTRE 3/81

Mit den Zufallszahlengeneratoren R = RND(0) und R = RND(1) laesst sich durch die Abfrage IF R> 0.5 der Muenzwurf (Kopf oder Zahl) simulieren. Bei einer homogenen Muenze (Schwerpunkt in der Mitte) wird beim mehrmaligen Werfen Kopf und Zahl gleich oft vorkommen, auch wenn es am Anfang anders scheint. Die Wahrscheinlichkeit von Kopf (oder von Zahl) ist p = 0.5.

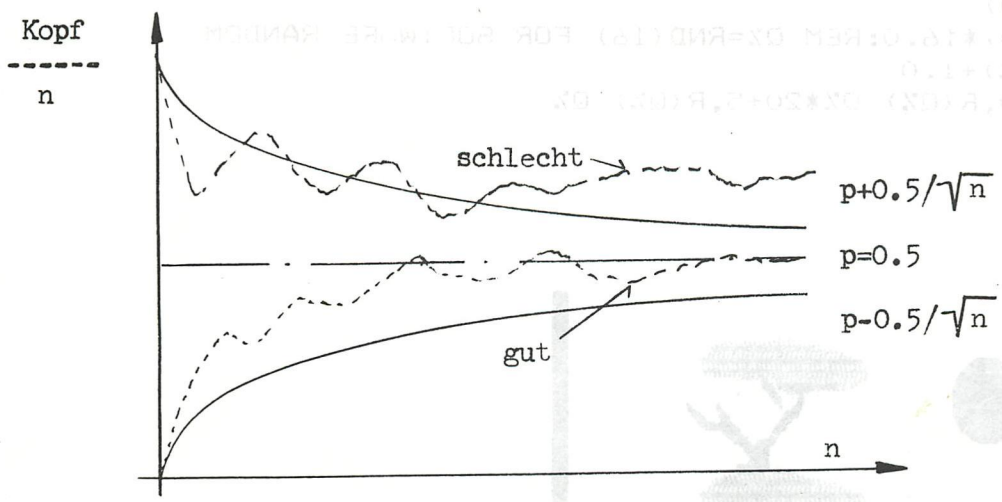
Das Gesetz der grossen Zahl besagt, dass wenn die Muenze unendlich oft geworfen wird, folgender Wert entsteht.

$$p = \frac{\text{Anzahl Kopf}}{\text{Anzahl Wuerfe}} = 0.5$$

Das vorliegende Programm simuliert ein Muenzwurfexperiment mit 320 Wuerfe und zeichnet laufend das erhaltene Ergebnis. Die Streuungsgrenzen um den erwarteten Wert sind als rote Flaeche dargestellt. Diese Grenzfunktion ist gegeben mit

$$S = p \pm \frac{\sqrt{p\,(1-p)}}{\sqrt{n}} \qquad \text{fuer } p=0.5 \qquad S = 0.5 \pm \frac{0.5}{\sqrt{n}}$$

wobei n = Anzahl Wuerfe
p = Wahrscheinlichkeit Kopf zu erhalten

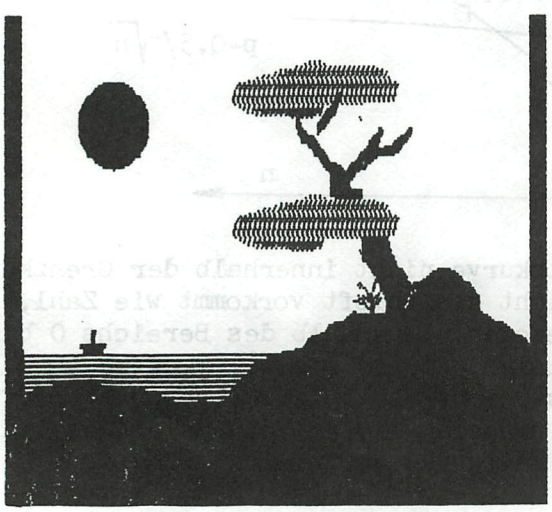


Pendelt sich die Wahrscheinlichkeitskurve nicht innerhalb der Grenzkurve ein, so bedeutet dies, dass Kopf nicht gleich oft vorkommt wie Zahl, oder auf den Zufallsgenerator des DAI bezogen, innerhalb des Bereichs 0 bis 1 kommen Werte unterhalb 0.5 nicht gleich oft vor wie Werte oberhalb 0.5. Mit anderen Worten der Generator ist nicht echt. Bei meinem DAI trifft dies leider fuer RND(0) zu. RND(1) hingegen liefert ein gutes Resultat. Der gewuenschte Generator wird via Zeile 510 definiert.

Das stoerende Flackern des Cursors laesst sich entfernen mittels
POKE#75,#20

```
10    REM BEISPIEL AUS DER STATISTIEK      MEYSTRE 3/81
20    XOFF=7.0:YMIT=107.0:YOFF=7.0
30    ROT=3.0:GRUEN=5.0:GELB=14.0:COLORG GRUEN ROT GELB 0
40    MODE 6A:MODE 6A
50    PRINT CHR$(12):POKE #75,32:REM CURSOR BLANCO
60    PRINT "ZUFALLSEXPERIMENT MIT TREFFERWAHRSCHEINLICHKEIT p=0.5"
70    PRINT "       ANZALH PROBEN   n                          "
80    PRINT "       RELATIVE HAEUFIGKEIT r/n                   "
90    R=0.0:N=1.0:NSIG=1.0
100   FOR I=1.0 TO 10.0:GOSUB 400:NEXT
110   FOR I=1.0 TO 310.0:GOSUB 400:GOSUB 500:NEXT
120   FOR I=1.0 TO 10.0:GOSUB 500:NEXT
130   GOTO 130
400   REM GRENZKURVE
410   DRAW XOFF+NSIG,YMIT-100/SQR(NSIG) XOFF+NSIG,YMIT+100/SQR(NSIG) ROT
420   NSIG=NSIG+1.0:RETURN
500   REM MOMENTANWERT DE HAEUFIGKEIT
510   IF RND(1.0)>0.5 THEN R=R+200.0
520   DOT XOFF+N,YOFF+R/N GELB
530   CURSOR 35,2:PRINT N
540   CURSOR 35,1:PRINT (R/200.0)/N
550   N=N+1.0:RETURN
```

```
5     MODE 5
10    REM RANDOM DISTRIBUTIE
15    REM HET PROGRAMMA EINDIGT MET "OFF SCREEN",
16    REM ONDERTUSSEN KAN U WEDDENSCHAPPEN AFSLUITEN....
20    DIM R(15.0)
30    Q%=RND(1.0)*16.0:REM Q%=RND(16) FOR SOFTWARE RANDOM
40    R(Q%)=R(Q%)+1.0
50    DRAW Q%*20,R(Q%) Q%*20+5,R(Q%) Q%
60    GOTO 30
```



LIST

```
PAGE 01     : FASING KEYBOARD MUSIC

002                     SCTRL   EQU     :FC06     8253 CONTROL
003                     SCH0    EQU     :FC00     CHANNEL 0
004                     SCH1    EQU     SCH0+2
005                     SCH2    EQU     SCH0+4
006                     TABLE   EQU     :9000
007                     VOL0    EQU     :FD04     VOLUME CH 0+1
008                     VOL2    EQU     VOL0+1    VOLUME CH 2(+NOISE)
009                             ORG     :300
010 0300 E5             IN      PUSH    H         SAVE REGISTERS
011 0301 C5                     PUSH    B
012 0302 D5                     PUSH    D
013 0303 F5                     PUSH    PSW
014 0304 010000                 LXI     B,:0
015 0307 3E30                   MVI     A,:30     CH0 MODE0 2BYTES
016 0309 3206FC                 STA     SCTRL
017 030C 3EFF                   MVI     A,:FF     VOL 0+1 MAX
018 030E 3204FD                 STA     VOL0
019 0311 3E0F                   MVI     A,:F      VOL 2
020 0313 3205FD                 STA     VOL2
021 0316 CDBBD6         GETC    CALL    :D6BB     GETC:ASCII IN A
022 0319 CA1603                 JZ      GETC      NO KEY
023 031C FE09                   CPI     9         TAB:OUT
024 031E CA5603                 JZ      OUT
025 0321 07                     RLC               A          A*2
026 0322 4F                     MOV     C,A       OFFSET VALUE IN C
027 0323 2106FC                 LXI     H,SCTRL
028 0326 3636                   MVI     M,:36     CONTROL BYTE FOR 8253
029 0328 210090                 LXI     H,TABLE
030 032B 09                     DAD     B
031 032C 7E                     MOV     A,M
032 032D 3200FC                 STA     SCH0      OSCILLATOR 0
033 0330 23                     INX     H
034 0331 7E                     MOV     A,M
035 0332 3200FC                 STA     SCH0
036 0335 210090                 LXI     H,TABLE
037 0338 09                     DAD     B
038 0339 7E                     MOV     A,M
039 033A 3C                     INR     A
040 033B 3C                     INR     A
041 033C 3202FC                 STA     SCH1      OSCILLATOR 1
042 033F 23                     INX     H
043 0340 7E                     MOV     A,M
044 0341 3202FC                 STA     SCH1
045 0344 210090                 LXI     H,TABLE
046 0347 09                     DAD     B
047 0348 7E                     MOV     A,M
048 0349 3C                     INR     A
049 034A 3C                     INR     A
050 034B 3204FC                 STA     SCH2      OSCILLATOR 2
051 034E 23                     INX     H
052 034F 7E                     MOV     A,M
053 0350 3204FC                 STA     SCH2
054 0353 C31603                 JMP     GETC
```

```
PAGE 02    : FASING KEYBOARD MUSIC

055 0356 F1          OUT     POP   PSW
056 0357 D1                  POP   D
057 0358 C1                  POP   B
058 0359 E1                  POP   H
059 035A C9                  RET
060 035B                     END

****************************
* S Y M B O L   T A B L E *
****************************

GETC   0316   IN     0300   OUT    0356   SCH0   FC00
SCH1   FC02   SCH2   FC04   SCTRL  FC06   TABLE  9000
VOL0   FD04   VOL2   FD05

#P.
0300 E5 C5 D5 F5 01 00 00 3E 30 32 06 FC 3E FF 32 04
0310 FD 3E 0F 32 05 FD CD BB D6 CA 16 03 FE 09 CA 56
0320 03 07 4F 21 06 FC 36 36 21 00 90 09 7E 32 00 FC
0330 23 7E 32 00 FC 21 00 90 09 7E 3C 3C 32 02 FC 23
0340 7E 32 02 FC 21 00 90 09 7E 3C 3C 32 04 FC 23 7E
0350 32 04 FC C3 16 03 F1 D1 C1 E1 C9
```

```
5     REM FASING KEYBOARD MUSIC:TABLE CEATOR
6     REM ----------------------------------
10    READ A$:IF A$="STOP" THEN END
20    A=ASC(A$)
25    PRINT A
30    READ F
40    FR=FREQ(F):REM OF FR=FREQ(F/2)...
50    POKE #9001+2*A,FR/256:POKE #9000+2*A,FR MOD 256
60    GOTO 10
100   DATA Z,528,X,595,C,660,V,704,B,792,N,880,M,990,",",1056
110   DATA S,561,D,627,G,748,H,836,J,935,L,1122,;,1254
115   DATA .,1188,/,1320
120   DATA STOP
```

PAGE 01 **SHORT RANDOM ROUTINE (NOISE)**

```
001                          ORG    :300
002                RANDOM    EQU    :2FE
003                BITS      EQU    :FD00
004 0300 E5                  PUSH   H
005 0301 F5                  PUSH   PSW
006 0302 3A00FD              LDA    BITS
007 0305 2AFE02              LHLD   RANDOM
008 0308 29                  DAD    H
009 0309 87                  ADD    A
010 030A F20E03              JP     NOSET
011 030D 23                  INX    H
012 030E 22FE02    NOSET     SHLD   RANDOM
013 0311 F1                  POP    PSW
014 0312 E1                  POP    H
015 0313 E9                  PCHL
016 0314           END       END

****************************
* S Y M B O L   T A B L E *
****************************

BITS   FD00   END    0314   NOSET  030E   RANDOM 02FE
```

SDOT

PURPOSE : DRAWS A DOT ON THE SCREEN
CALL : RST 5 + DATA 1E (actual call:E710)
ENTRY : C = Y
HL= X
A = COLOUR OF DOT
EXIT : CARRY UNSET:ALL REGISTERS PRESERVED EXCEPT F
CARRY SET :A CONTAINS ERROR CODE (colour not available, off screen)
OTHER REGISTERS (EXCEPT F) PRESERVED

SDRAW

PURPOSE : DRAWS A LINE ON THE SCREEN
CALL : RST 5 + DATA 21 (actual call:E71B)
ENTRY : B = Y1
DE= X1
C = Y2
HL= X2
A = COLOUR FOR LINE
EXIT : AS DOT
note: The package will draw the line starting at the left end, whichever order the parameters are given in.

SFILL

PURPOSE : FILLS A RECTANGULAR AREA ON SCREEN
CALL : RST 5 + DATA 24 (actual call:E818)
ENTRY : AS DRAW
EXIT : AS DRAW
notes: The middle of the rectangle is filled first,then the left and the right vertical strips.
Rectangle is filled in same order,whichever order parameters are given in.

SSCRN

PURPOSE : ASKS COLOUR OF A POINT ON SCREEN AND SIZE OF GRAPHICS SCREEN
CALL : RST 5 + DATA 27 (actual call: E8F4
ENTRY : C = Y, HL = X
EXIT : AS DOT EXCEPT THAT IF CARRY UNSET:
A contains colour, B,DE MAX coordinates of screen

*** WILHELMUS *** PAGE 01

```
1       REM wilhelmus ***
10        CLEAR 1000
20        DIM TOON(8)
30        DIM FR(68),WT(68)
40        FOR I=0 TO 8:READ TOON(I):NEXT
50        DATA 392,440,494,523,587,659,698,784,880
60        FOR I=1 TO 18
70    1     READ FR(I),WT(I)
80    1     FR(I+18)=FR(I):WT(I+18)=WT(I)
90    1     NEXT
100       FOR I=37 TO 68
110   1     READ FR(I),WT(I)
120   1     NEXT
130       T=3
140       FOR I=1 TO 68
150   1     IF FR(I)=9 THEN SOUND OFF :GOTO 180
160   1     SOUND 0 0 15 0 FREQ(TOON(FR(I))*1.5)
170   1     SOUND 1 0 12 0 FREQ(TOON(FR(I))*1.5-9)
180   1     WAIT TIME WT(I)*T
190   1     NEXT
200       SOUND OFF :PRINT
210       INPUT "NOG EEN KEER ";JOFN$
220       IF JOFN$="NEE" GOTO 300
230       IF JOFN$="JA" GOTO 140
240       PRINT :PRINT "BEGRIJP IK NIET ! MOET HET ";
250       GOTO 210
300       PRINT :END
510       DATA 0,10,3,8,9,2,3,10,4,5,5,5,6,5,4,5,5,10,4,5,5,5,6,10,5,10,
C         4,5,3,5,4,10,3,20,9,10
520       DATA 5,5,6,5,7,20,8,10,7,20,6,10,5,10,4,5,5,5,6,10,5,10,4,10,3
C         ,10,4,20,9,10
530       DATA 0,10,3,5,2,5,3,5,4,5,5,10,4,20,3,10,2,10,0,10,1,5,2,5,3,8
C         ,9,2,3,10,2,10,3,40
```

```
5       REM FUNNY SIRENE
10      O=#FC00
20      C=#FC06
30      POKE C,#36
100     FOR X=0.0 TO 50.0:POKE O,X:POKE O,X:NEXT
110     FOR X=50.0 TO 0.0 STEP -1.0:POKE O,X:POKE O,X:NEXT
120     GOTO 100
```

```
5       REM ENDLESS CONTINUATION LINES.....
6       REM WITH ILLUSION OF A RIGHT MOVING BLOCK OF TEXT
7       REM ALSO TRY : POKE #BF68,#C0, FOR RUNNING MESSAGE !
8       REM #7B HOLDS COUNT OF CONTINUATION LINES
10      PRINT "TEST  ";
20      POKE #7B,0
30      GOTO 10
```

T U E R M E V O N H A N O I mit Problemlösu

A. MEYSTRE / MAI 81

Hatten Sie schon die nötige Geduld um die 511 Züge zur richtigen Umschichtung auszuführen?
Falls ja -- herzliche Gratulation, falls nein -- lassen Sie es doch durch den Computer tun!
Das modifizierte HANOI-Programm löst das Problem mit wählbarer Geschwindigkeit.

Erklärung der rekursiven Lösung in Pseudo-BASIC:

Hauptprogramm:
```
10  A=1 : B=2 : C=3 : M=8
20  GOSUB MOVTOWER (M,A,B,C)
30  END
```

Unterprogramm:
```
500 DEFF SUB MOVTOWER (M,A,B,C)
510 IF M=1 THEN PRINT A,C: RETURN
520 GOSUB MOVTOWER (M-1,A,C,B)
530 PRINT A,C
540 GOSUB MOVTOWER (M-1,B,A,C)
550 RETURN
```

Das Unterprogramm MOVETOWER ruft sich selbst immer wieder auf, bis das ganze Problem gelöst ist.
Jeder Aufruf speichert alle 4 Parameter in einen Stack fortlaufend ab. M ist die Turmhöhe,
A, B und C enthalten die momentanen Turmpositionen 1, 2 und 3. Um die 9 Scheiben Umzuschichten,

werden $2^9 - 1 = 511$ Züge benötigt

Für eine Turmhöhe von 3 Scheiben sieht der Vorgang für die 7 Züge wie folgt aus:

CALL MOVETOWER in Zeile	PARAMETERWERTE M	A	B	C	Ausgabe des Zuges Zeile	PRINT A	, C
20	3	1	2	3			
520	2	1	3	2			
520	1	1	2	3			
	1	1	2	3	510	1	3
RETURN	2	1	3	2			
	2	1	3	2	530	1	2
540	1	3	1	2			
	1	3	1	2	510	3	2
RETURN	2	1	3	2			
RETURN	3	1	2	3			
	3	1	2	3	530	1	3
540	2	2	1	3			
520	1	2	3	1			
	1	2	3	1	510	2	1
RETURN	2	2	1	3			
	2	2	1	3	530	2	3
540	1	1	2	3			
	1	1	2	3	510	1	3
RETURN	2	2	1	3			
RETURN	3	1	2	3			
RETURN	3	1	2	3			

Um die obige Prozedur in BASIC zu schreiben, ist der Rekursionsvorgang sowie die Parameterübergabe zu simulieren. Diese Simulation findet in den Zeilen 1000 - 1220 statt.
Im Originalprogramm wurde lediglich eine zusätzliche Abfrage für den automatischen Ablauf eingebaut. Die Zeilen 111 - 114 wurden ersetzt durch

```
111 IF A$<>"JA" THEN GOSUB 350                    Handeingabe wie früher
112 IF A$= "JA" THEN GOSUB 1000: WAIT TIME WT     Automatische Lösung
```

Um die automatischen Züge zu beobachten, kann die Verweilzeit pro Zug WT eingegeben werden.

neue Variablen:

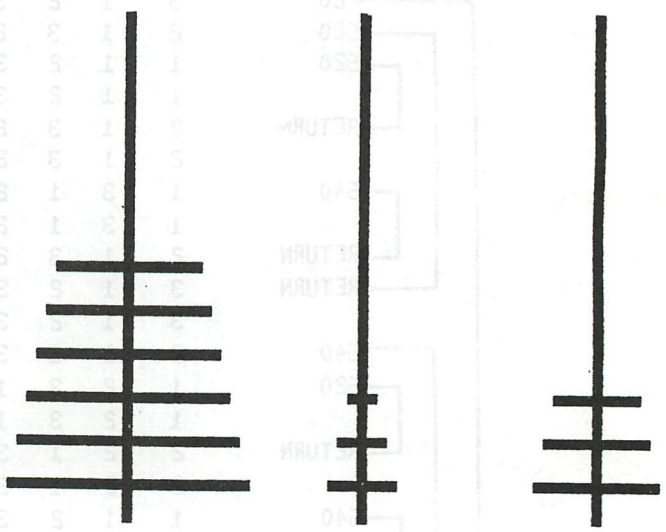
ST(TH,2)	Stack mit 2 mal TH Elemente und Stackpointer SP
TH	Turmhöhe (Anzahl Scheiben + 1)
FA,FB,FC	Turmpositionen 1,2 und 3
RET	Rücksprungposition
RESTART	Wiedereintrittsposition

Das Unterprogramm lässt sich für eine beliebige Turmhöhe anwenden.

Literatur: THE COMPUTER JOURNAL volume 20 number 3 1977
P.J. Hayes, A note on the tower of Hanoi problem

towers of hanoi

```
1    REM HANOI  ERWEITERTE DAI-VERSION   A. MEYSTRE  6/81
5    CLEAR 2000:DIM Z(200.0)
8    MODE 0:PRINT CHR$(12):PRINT :PRINT
9    PRINT "     .............TUERME VON HANOI............."
10   PRINT :PRINT
12   PRINT "     SIE MUESSEN ALLE BALKEN VOM TURM 1 ZU TURM 3"
13   PRINT "     UMSCHICHTEN, SO DASS NIE EIN LAENGERER BALKEN"
14   PRINT "         AUF EINEN KUERZEREN ZU LIEGEN KOMMT."
15   PRINT "   UM DIE BALKEN ZU BEWEGEN, GEBEN SIE DIE NUMMER DES"
16   PRINT " TURMES AN VON WELCHEM DER BALKEN GENOMMEN WERDEN SOLL,"
17   PRINT "        SOWIE DIE NUMMER DES EMPFAENGERS AN"
18   PRINT :PRINT "            GEBEN SIE DIE TURMHOEHE ALS"
19   INPUT "           ZAHL ZWISCHEN 0 UND 12 EIN ";N:PRINT :IF N<1 OR N>12 THEN 19
22   PRINT :PRINT "        WOLLEN SIE DEN AUTOMATISCHEN ABLAUF?"
24   PRINT "        ANTWORT JA ODER NEIN ";:INPUT A$:PRINT
25   IF A$<>"JA" THEN 28
26   NZ=INT(2^N-0.5):PRINT :PRINT "        ES WERDEN ";NZ;" ZUEGE BENOETIGT !!!"
27   PRINT :PRINT "        VERWEILZEIT PRO ZUG IN 1/50-SEKUNDEN = ";:INPUT WT
28   PRINT CHR$(12):COLORT 7 0 0 0:COLORG 7 4 5 1:MODE 2A
30   JC1=0:Y9=60.0:C1=4.0:C2=5.0:C3=1.0:C0=7.0
33   DRAW 0,0 71,0 C1
36   FOR I=1.0 TO 3.0
38   DRAW I*24-12,0 I*24-12,Y9 C2
40   Z(1.0)=0.0:Z(I*20.0)=10.0:NEXT
50   M=1.0:C=C3
60   FOR I=1.0 TO N
70   Z(1.0)=I:Z(20.0+I)=11.0-I
80   GOSUB 900:NEXT
90   GOTO 110
100  PRINT "     UNGUELTIGER ZUG"
110  JC1=JC1+1:PRINT "     IHR ZUG VON  <1,2 OR 3>  ";
111  IF A$<>"JA" THEN GOSUB 350
112  IF A$="JA" THEN GOSUB 1000:WAIT TIME WT
120  IF M1<>INT(M1) OR M1<1.0 OR M1>3.0 GOTO 100
130  IF M2<>INT(M2) OR M2<1.0 OR M2>3.0 GOTO 100
140  IF M1=M2 OR Z(M1)=0.0 GOTO 100
150  P1=Z(M1)+20.0*M1
160  P2=Z(M2)+20.0*M2
170  IF Z(P1)>Z(P2) GOTO 100
200  M=M1:C=C0:GOSUB 900
210  Z(M2)=Z(M2)+1.0:Z(P2+1.0)=Z(P1)
220  Z(M1)=Z(M1)-1.0
230  M=M2:C=C3:GOSUB 900
240  G=G+1.0
250  IF Z(3.0)<N GOTO 110
300  PRINT "     SIE HABEN  ",JC1,"ZUEGE BENOETIGT";
310  INPUT A$:GOTO 1
350  P=GETC:WAIT TIME 5:IF P=0.0 GOTO 350
360  M1=P-48.0:PRINT M1;:PRINT "  NACH  ";
370  P=GETC:WAIT TIME 5:IF P=0.0 GOTO 370
380  M2=P-48.0:PRINT M2;:PRINT "   ";:PRINT JC1;:PRINT " ZUEGE"
390  RETURN
900  X=M*24.0-12.0
910  Y=4.0*Z(M)
920  X1=Z(Z(M)+20.0*M)+2.0
930  DRAW X-X1,Y X-1,Y C
935  XX1=X1+X:IF XX1>XMAX THEN XX1=XMAX
940  DRAW X+1,Y XX1,Y C
950  RETURN
960  END
```



```
1000  REM UNTERPROGRAMM ZUR REKURSIVEN LOESUNG DES PROBLEMS
1010  REM BEIM AUTOMATISCHEN ABLAUF      A. MEYSTRE  MAI/81
1020  ON RESTART GOTO 1100,1080
1030  DIM ST(N,2.0):SP=0:TH=N:FA=1:FB=2:FC=3:RET=1
1040  REM ENTRY IN MOVETOWER
1050  SP=SP+1:ST(SP,1.0)=TH:ST(SP,2.0)=RET
1060  IF TH=1 THEN RESTART=1:M1=FA:M2=FC:GOTO 1200
1070  TH=ST(SP,1.0)-1.0:HELP=FB:FB=FC:FC=HELP:RET=2:GOTO 1050
1080  TH=ST(SP,1.0)-1.0:HELP=FB:FB=FA:FA=HELP:RET=3:GOTO 1050
1100  REM RETURN FROM MOVETOWER
1110  ON RET GOTO 1130,1150,1140
1130  RETURN
1140  SP=SP-1:TH=ST(SP,1.0)+1.0:HELP=FA:FA=FB:FB=HELP:RET=ST(SP,2.0):GOTO 1100
1150  SP=SP-1:TH=ST(SP,1.0)+1.0:HELP=FB:FB=FC:FC=HELP:RET=ST(SP,2.0)
1160  RESTART=2:M1=FA:M2=FC
1200  PRINT M1;:PRINT "  NACH  ";
1210  PRINT M2;:PRINT "   ";:PRINT JC1;:PRINT " ZUEGE"
1220  RETURN
```

---

attractive cursor

```
10     REM A VERY ATTRACTIVE CURSOR
20     POKE #74,0:POKE #75,#FF:COLORT 8 0 15 0
```

```
10    MODE 0:PRINT CHR$(12):COLORT 8 0 0 8
14    REM
15    REM Tekst die met CURSOR op 'n bepaalde plaats
16    REM wordt gezet,moet met 'n spatie beginnen.
17    REM
20    CURSOR 30,20:INPUT " Tekst kleur";TK
30    CURSOR 20,15:INPUT " Achtergrond kleur";AK
40    INV=30.0:CURSOR 13,10:GOSUB 1000:INPUT " INVERSE Achtergrond kleur";INVAK
50    INV=25.0:CURSOR 3,5:GOSUB 1000:INPUT " INVERSE Tekst kleur";INVTK
70    COLORT AK TK INVAK INVTK:PRINT :PRINT :END
998   REM
999   REM * * * * *   De inverse routine   * * * * *
1000  FOR I=2.0*CURX+8.0 TO 2.0*CURX+8.0+2.0*INV STEP 2.0
1010  POKE #B3E2+CURY*#86-I,#FF:REM ADAPT #B3E2 FOR RAM SIZE
1020  NEXT:RETURN
```

DAI 24 x 60 CHARACTER DISPLAY

POS →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
LINE ↓	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==	==
23	BFE7	E5	E3	E1	DF	DD	DB	D9	D7	D5	D3	D1	CF	CD	CB	C9	C7	C5	C3	C1	BF	BD	BB	B9	B7	B5	B3	B1	AF	AD	AB	A9	A7	A5	A3	A1	9F	9D	9B	99	97	95	93	91	8F	8D	8B	89	87	85	83	81	7F	7D	7B	79	77	75	73	71
22	BF61	5D	59	55	51	4D	49	45	41	3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BEFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81				
21	BEDB	D7	D3	CF	CB	C7	C3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB		
20	BE55	51	4D	49	45	41	3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BDFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81				
19	BDCF	CB	C7	C3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF	
18	BD49	45	41	3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BCFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75				
17	BCC3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF				
16	BC3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BBFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69				
15	BBB7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF							
14	BB31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BAFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69							
13	BAAB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF									
12	BA25	21	1D	19	15	11	0D	09	05	01	B9FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69									
11	B99F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF											
10	B919	15	11	1D	09	05	01	B8FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69											
9	B893	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF															
8	B80D	09	05	01	B7FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69															
7	B787	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF																			
6	B701	B6FD	F9	F5	F1	ED	E9	E5	E1	ED	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69																				
5	B67B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF																						
4	B5F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69																							
3	B56F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB	F7	F3	EF																									
2	B4E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69																										
1	B463	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B3FF	FB	F7	F3	EF																											
0	B3DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69																												

↑
x

high address byte x + 8 (MODE BYTE) BIT 7 = 0/1 BIT 6 = 1 BIT 5-4 = 10/11 BIT 3-0 line repeat count (BASIC)

grafiek 5e graadspolynomen

```
1     REM GRAFIEKEN  VAN  VIJFDE GRAADS POLYNOMEN  MET
2     REM GEHELE COeFFICIENTEN;  C.W.A.van Dijk-KAMPEN
3     MODE 0:PRINT CHR$(12):CLEAR 2000
4     PRINT :PRINT :PRINT "                                5   4   3   2   "
5     PRINT "            de grafiek van  ax  +bx  +cx  +dx  +ex  +f"
6     PRINT "            ******************************************"
7     PRINT :INPUT "              a=";A:INPUT "  b=";B:INPUT "  c=";C:INPUT "  d=";D:INPUT "  e=";E:INPUT "  f=";F
8     PRINT :PRINT :PRINT "functie nog  veranderen ..? < J / N >"
9     G!=GETC:WAIT TIME 5:IF G!=0 THEN 9
11    IF G!=74.0 THEN 3
12    PRINT :PRINT :PRINT :PRINT :INPUT "HET DOMEIN GAAT VAN ";XMI!
13    INPUT "  NAAR ";XMA!:PRINT
14    PRINT "domein nog veranderen . . ? < J / N >"
15    G!=GETC:WAIT TIME 5:IF G!=0 THEN 15
16    IF G!=74 THEN 12
18    N!=0.0
19    YMI!=1E6:YMA!=-1E6
20    MODE 6:DIM Y!(YMAX):COLOR6 0 15 8 5
25    D1!=XMA!-XMI!:S!=D1!/250.0
30    FOR X!=XMI! TO XMA! STEP S!
35    N!=N!+1.0
40    Y!(N!)=A!*X!*X!*X!*X!*X!+B!*X!*X!*X!*X!+C!*X!*X!*X!+D!*X!*X!+E!*X!+F
50    IF YMA!<Y!(N!) THEN YMA!=Y!(N!)
52    IF YMI!>Y!(N!) THEN YMI!=Y!(N!)
60    NEXT
65    MODE 6
70    RAN!=YMA!-YMI!+1.0:SCY!=YMAX/RAN!
90    F1!=XMAX/250.0
92    DRAW 0,0 XMAX,0 5:FOR I!=0.0 TO D1!:DOT I!*XMAX/D1!,0 15:NEXT
100   FOR N!=1.0 TO 250.0
120   M!=Y!(N!)-YMI!:P!=M!*SCY!+1.0
140   DOT N!*F1!,P! 15
150   NEXT
200   G!=GETC:IF G!=0.0 GOTO 200
201   IF G!=18.0 THEN 3
202   IF G!<>32.0 THEN 200
203   MODE 6A
204   PRINT "                  5    4    3    2      "
205   PRINT "de functie is ";A;"X +";B;"X +";C;"X +";D;"X +";E;"X+";F
206   PRINT :PRINT "HET DOMEIN IS < ";XMI!;" ";XMA!;" >";
210   G!=GETC:IF G!=0.0 THEN WAIT TIME 5:GOTO 210
211   IF G!=18 THEN 3
212   GOTO 12
```

Binaire representatie in het geheugen van integer- en floating point getallen

1. Inleiding

Programma 1 beeldt op scherm de binaire representatie af van een integer- of een floating point getal. Vooreerst vraagt het programma of een integer (I) of een floating point getal (F) moet worden omgezet. Nadien wordt het om te zetten getal opgevraagd.

Programma 1

```
100   REM BITRIJGENERATOR
110   PRINT CHR$(12):GOSUB 2000
120   INPUT " INTEGER (I) OF FLOATING POINT GETAL (F) ";T$:PRINT
130   IF T$="I" THEN 170
140   IF T$="F" THEN 150
145   GOTO 120
150   INPUT "     FLOATING POINT GETAL";W:PRINT :M%=VARPTR(W):GOTO 20(
170   INPUT "     INTEGER GETAL";W%:PRINT
180   M%=VARPTR(W%)
200   FOR A%=M% TO M%+3
300   P%=PEEK(A%)
400   FOR X%=7 TO 0 STEP -1.0
500   B%=P% IAND (2^(X%+1))
600   C%=B% SHR X%:ST$="0":IF C%=1 THEN ST$="1"
700   PRINT ST$;
1000  NEXT:PRINT " ";
1100  NEXT
1150  PRINT
1200  GOTO 120
2000  CURSOR 1,22
2050  PRINT "BITRIJGENERATOR VOOR INTEGERS EN FLOATING POINT GETALLEN"
2100  PRINT :PRINT :RETURN
```

Indien met dit programma de binaire voorstelling bepaald wordt van:

a. het integer-getal 23 vindt men:

00000000 00000000 00000000 00010111

b. het floating point getal 23.0 vindt men:

00000101 10111000 00000000 00000000

c. het floating point getal 20.23 vindt men:

00000101 10100001 11010111 00001010

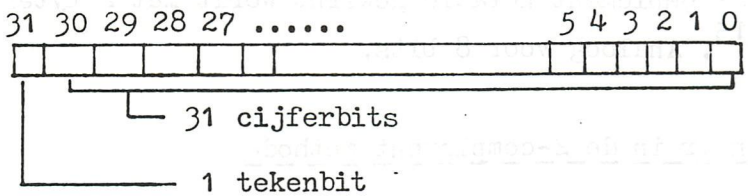
Vastgesteld wordt dat zowel voor de voorstelling van een integer- als voor deze van een floating point getal 4 bytes worden gebruikt.

De doelstelling van dit artikel is de opbouw van deze bitrijen toe te lichten, zodanig dat de lezer zelf met papier en potlood (eenvoudige) omzettingen kan uitvoeren en deze daarna met programma 1 controleren. Een beter inzicht in de opslag van de data in het geheugen wordt hierdoor bekomen.

2. Integer-getallen

2.1 2-complement notatie met 32 bits

De integer-getallen worden in 4 bytes door middel van de 2-complement notatievorm voorgesteld. Dit betekent dat van de 32 beschikbare bits er 31 (bit 0 tot en met bit 30) gebruikt worden voor de representatie van de numerieke waarde en 1 (bit 31) voor het teken van het getal.



2.2 Voorstelling van een positief getal in de 2-complement notatie

Voor een positief getal is de tekenbit steeds 0. De numerieke waarde wordt bekomen door het decimaal equivalent te bepalen van het binaire getal voorgesteld door de 31 cijferbits.

voorbeeld:

00000000 00000000 00000100 10100000

de 31 cijferbits bepalen het getal 1184

de tekenbit is 0 het getal is positief

De gegeven bitrij stelt bijgevolg het getal (+)1184 voor.
Het grootste integer-getal dat in deze notatievorm kan worden gerepresenteerd is bijgevolg:

$$01111111\ 11111111\ 11111111\ 11111111 = 2^{31} - 1$$
$$= 2147483647$$

2.3 Voorstelling van een negatief getal in de 2-complement notatie

Voor een negatief getal is de tekenbit steeds 1. De numerieke waarde wordt bekomen door het decimaal equivalent te bepalen van het binaire getal voorgesteld door de 31 cijferbits en dit af te trekken van $2^{31} = 2147483648$

voorbeeld:

10000000 00010110 11011000 00000111

de 31 cijferbits bepalen het getal 1497095

de numerieke waarde van het voorgestelde getal is dan:

2147483648 - 1497095 = 2145986553

Daar de tekenbit negatief is ,stelt de opgegeven bitrij bijgevolg het getal - 2145986553 voor.

Het kleinste negatief integer-getal dat met 4 bytes in de 2-complement methode kan worden voorgesteld is dus dit waarbij het getal bepaald door de 31 cijferbits gelijk is aan 0. In dit geval wordt slechts 0 afgetrokken van 2147483648, zodat dit kleinste getal gelijk is aan : - 2147483648, met als binaire representatie:

10000000 00000000 00000000 00000000

opmerking: indien in de 2-complement notatie gewerkt wordt met 2 bytes moet worden afgetrokken van 2^{15}. Analoog voor 8 bits.

2.4 Van decimaal naar binair in de 2-complement methode

In vorige twee nummers werden de algoritmen gegeven om, vanuit de binaire representatie het decimaal equivalent te bepalen. Nu wordt het omgekeerde probleem gesteld: welk is de binaire 2-complement notatie van een gegeven decimaal getal ?

voorbeeld 1: binaire representatie van het integer-getal: (+)2543

De corresponderende bitrij kan als volgt bepaald worden: bereken het gehele quotient en de rest van de deling van 2543 door 2. Het quotient is 1271 en de rest 1. Deze rest is de meest rechtse bit van de gevraagde binaire notatie; dit algoritme wordt nu herhaald op het quotient 1271 tot 0 als quotient bekomen wordt; de nieuwe resten worden telkens links van de vorige geschreven.

Schematisch krijgen we volgende notatie:

gehele quotienten :	0	1	2	4	9	19	39	79	158	317	635	1271	2543
resten	1	0	0	1	1	1	1	0	1	1	1	1	

binaire representatie

We besluiten:

2543 = 100111101111

Om tot een representatie met 4 bytes te komen worden deze cijferbits links met nullen aangevuld tot 31 bits worden bekomen. De bijhorende tekenbit is 0, zodat de 2-complement notatie van (+)2543 is:

(+) 2543 = 00000000 00000000 00001001 11101111

voorbeeld 2: binaire representatie van -2543

Theoretisch kan dit als volgt gebeuren: de tekenbit moet 1 zijn, terwijl de cijferbits het binaire getal 2147483648 - 2543 = 2147481105 moeten vormen.

Van dit getal kan de binaire representatie met 31 bits bepaald worden volgens

het algoritme uit het vorige voorbeeld. We vinden:

2147481105 = 1111111 11111111 11110110 00010001

7bits 8 bits 8bits 8bits

31 cijferbits

zodat:

- 2543 = 11111111 11111111 11110110 00010001

De aftrekking van 2147483648 kan vermeden worden met volgend meer practisch algoritme om de 2-complement notatie van -2543 te bepalen:

a. bepaal de 2-complement vorm van het positieve getal (2543)
   volgens de methode uit voorbeeld 1
   Dit geeft:

   2543 = 00000000 00000000 00001001 11101111

b. inverteer alle bits van deze bitrij,dwz vervang een 1 door een 0 en omgekeerd.
   Men bekomt: 11111111 11111111 11110110 00010000

c. tel bij deze nieuwe bitrij 1 op:

```
  11111111 11111111 11110110 00010000
                              +        1
  ---------------------------------------
  11111111 11111111 11110110 00010001
```

   Deze laatste som is de gewenste 2-complement notatie.

3. Floating point getallen

3.1 Definitie en notatie van floating point getallen in het decimaal en binair talstelsel

Een floating point getal bestaat uit een geheel deel en een fractiedeel. Beide delen worden in de gewone notatie naast elkaar geschreven en zijn gescheiden door een punt. Het deel links van de punt is het gehele deel, terwijl de cijfers rechts ervan het fractiedeel vormen.

voorbeelden: in het decimaal talstelsel 2.75 - 0.678 567.0

in het binair talstelsel 1011.010 -101.011 0.011

Een floating point getal kan geschreven worden als de som van de producten van elk van zijn cijfers met een positieve of een negatieve macht van het grondtal

van het talstelsel waarin het floating point getal is uitgedrukt.

voorbeeld 1: in het decimaal stelsel

$$376.14 = 3 \times 10^2 + 7 \times 10^1 + 6 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}$$

voorbeeld 2: in het binair stelsel

$$1011.101 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$(= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125)$$

$$(= 11.625)$$

De plaats van de punt geeft de overgang aan tussen de positieve en de negatieve machten van het grondtal.

3.2 De wetenschappelijke notatievorm voor floating point getallen

Omdat in een computer niet met een onbeperkt aantal cijfers kan worden gewerkt, gebruikt men meestal de wetenschappelijke notatievorm om floating point getallen voor te stellen. In deze representatie van de floating point getallen onderscheidt men 3 delen:

a. een geheel deel
b. een fractiedeel
c. een(expliciet vermelde) macht van het grondtal van het talstelsel

Het voordeel van deze notatievorm is dat zeer grote en zeer kleine getallen met een minimum aantal cijfers kunnen geschreven worden.

voorbeeld: 17.15×10^9 i. p. v. 17150000000 in het decimaal stelsel

11.01×10^{101} i. p. v. 1101000 in het binair stelsel

3.3 De genormaliseerde wetenschappelijke notatievorm: definitie

Door de exponent van de macht van het grondtal op de juiste wijze aan te passen kan er steeds voor gezorgd worden dat het gehele deel in de wetenschappelijke notatievorm nul is.

voorbeeld: $17.15 \times 10^9 = 0.1715 \times 10^{11}$ in het decimaal stelsel

$11.01 \times 10^{101} = 0.1101 \times 10^{111}$ in het binair talstelsel

Indien het fractiedeel uit een vast aantal cijferposities bestaat, zal een floating point getal bijgevolg met de grootste nauwkeurigheid kunnen worden voorgesteld, indien geeist wordt dat het cijfer dat onmiddellijk na de punt volgt verschillend is van nul. Ook aan deze eis kan voldaan worden door de exponent van

het grondtal op een gepaste wijze te veranderen.

voorbeeld: $0.0023 \times 10^{4} = 0.23 \times 10^{2}$ (decimaal)

$0.0011 \times 10^{1010} = 0.11 \times 10^{1000}$ (binair)

Indien vorige twee voorwaarden bij de voorstelling van een floating point getal (in om 't even welk talstelsel) voldaan zijn, noemt men de notatie genormaliseerd. Samengevat kan gesteld worden: een wetenschappelijke notatie van een floating point getal is genormaliseerd dan en slechts dan als:

1. het gehele deel ervan nul is
2. het eerste cijfer van het fractiedeel verschillend is van nul; het enige getal dat op deze eis een inbreuk mag maken is het getal nul zelf.

3.4 Voorstelling in het geheugen van de binaire genormaliseerde wetenschappelijke notatievorm van een floating point getal

Rekening houdend met de definitie van een genormaliseerde wetenschappelijke notatie kan de standaardvorm hiervan in basis twee (binair 10 genoteerd) als volgt worden opgeschreven:

$$0.BBBBBBBBB \times 10^{bbbbb}$$

Hierin stellen de letters B en b bits (0 of 1) voor. Het aantal B's is afhankelijk van het vast aantal bits dat gebruikt wordt om de fractie voor te stellen. Verder in deze tekst is dit aantal 24. Het aantal b's bepaalt de maximale waarde van de exponent van het grondtal 2. Dit aantal is 7.

Daar de fractie uit 24 bits bestaat zal een nauwkeurigheid van $\frac{1}{2^{24}} \approx 0.000000059$ bekomen worden. Dit komt neer op een nauwkeurigheid van ongeveer 7 decimale cijfers.

De exponent van het grondtal wordt met 7 bits weergegeven in 2-complement notatie, d.w.z. hij varieert van:

$$0111111 = 2^{6} - 1 = 63$$

$$\text{tot} \quad 1000000 = -64$$

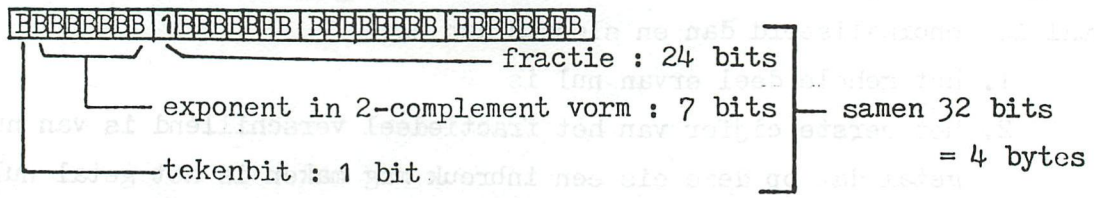
De grootste macht waarmee de fractie bijgevolg kan vermenigvuldigd worden is

$$2^{63} \approx 9.2 \times 10^{18}$$

en de kleinste: $2^{-64} \approx 5.4 \times 10^{-20}$

zodat kan gesteld worden dat de strikt positieve floating point getallen die expliciet kunnen worden voorgesteld gelegen zijn tussen 10^{-19} en 10^{19}.

De 7 exponentbits worden door de tekenbit tot één byte vervolledigd. Deze tekenbit is 0 voor positieve floating point getallen en 1 voor negatieve. De grenzen die hoger werden gegeven voor de strikt positieve floating point getallen (10^{-19} en 10^{19}) kunnen nu aangepast worden voor de negatieve: -10^{19} en -10^{-19}.
Deze teken-exponent-byte tesamen met de 3 fractie-bytes vormen de 4 bytes die gebruikt worden om floating point getallen voor te stellen. Ze zijn als volgt in het geheugen opgeslagen:



Daar deze voorstellingswijze genormaliseerd is, is de meest beduidende bit van de fractie - met uitzondering van het floating point getal 0.0 - steeds 1. Dit werd dan ook zo weergegeven in de schematische voorstelling. Daar het gehele deel nul is, zou het slechts een verlies aan geheugenruimte zijn indien ook dit gehele deel in de voorstellingswijze zou worden opgenomen. De punt (scheiding tussen geheel en fractie deel) is bijgevolg ook overbodig. Impliciet staat ze voor de tweede byte.

3.5 Binaire genormaliseerde wetenschappelijke notatievorm: enkele voorbeelden

Door middel van voorbeelden zullen de papier-en-potlood-algoritmen voor de verschillende gevallen behandeld worden.

Voorbeeld 1 Binaire representatie van het floating point getal 20.23

Deze voorstelling wordt in 3 fasen berekend:

1. binaire voorstelling van het gehele deel 20

   $$20 = 10100$$

2. binaire voorstelling van het fractiedeel: 0.23
   Hiertoe maken we volgende bedenking:

   $$0.23 = 2 \times 10^{-1} + 3 \times 10^{-2}$$

   We onderstellen nu dat de binaire equivalenten van de machten van 10 (tien) met negatieve exponenten gekend zijn met een nauwkeurigheid van 32 cijferbits en in een niet-genormaliseerde vorm. Deze binaire equivalenten staan in tabel 1 en werden berekend met programma 2. Gezien slechts 7 decimale cijfers nauwkeurig kunnen worden weergegeven, volstaat het deze tabel op te bouwen tot 10^{-7}.

Programma 2

```
5       CLEAR 1000
10      D%=10
20      FOR K=1.0 TO 7.0
25      DT%=2:B$=""
30      FOR N%=1 TO 32
40      Q%=DT%/D%
50      IF Q%=1 THEN DT%=DT%-D%
60      DT%=DT%*2
70      C$=MID$(STR$(Q%),1,1)
80      B$=B$+C$
90      NEXT
100     PRINT "BINAIR EQUIVALENT VAN";1.0/D%
110     FOR I=0.0 TO 3.0
120     PRINT MID$(B$,I*8,8);"  ";
130     NEXT
140     PRINT :PRINT :D%=D%*10
150     NEXT
```

Tabel 1 Tabel van de binaire equivalenten van 10^{-1} tot en met 10^{-7}

```
BINAIR EQUIVALENT VAN 0.1
00011001  10011001  10011001  10011001

BINAIR EQUIVALENT VAN 1E-2
00000010  10001111  01011100  00101000

BINAIR EQUIVALENT VAN 1E-3
00000000  01000001  10001001  00110111

BINAIR EQUIVALENT VAN 1E-4
00000000  00000110  10001101  10111000

BINAIR EQUIVALENT VAN 1E-5
00000000  00000000  10100111  11000101

BINAIR EQUIVALENT VAN 1E-6
00000000  00000000  00010000  11000110

BINAIR EQUIVALENT VAN 1E-7
00000000  00000000  00000001  10101101
```

De berekening van de binaire voorstelling van het fractiedeel gebeurt nu als volgt:

```
2 x 0.1 = 2 x 00011001 10011001 10011001 10011001
        =     00110011 00110011 00110011 00110010

3 x 0.01= 3 x 00000010 10001111 01011100 00101000
        =     00000111 10101110 00010100 01111000
```

Opgeteld geeft dit:

```
0.23 = 00110011 00110011 00110011 00110010
     + 00000111 10101110 00010100 01111000
       00111010 11100001 01000111 10101010
```

opmerking: gezien in de binaire representatie de notatie 0. niet gebruikt wordt, staan in het rechterlid uitsluitend cijfers van het fractiedeel.

De berekeningen van fase 1 en fase 2 kunnen nu als volgt schematisch worden samengebracht:

gehele deel	fractiedeel	teken en exponent
00010100	00111010 11100001 01000111 10101010	00000000

3. tijdens fase 3 wordt deze binaire wetenschappelijke notatie genormaliseerd. Dit gebeurt door alle bits van het gehele en het fractie deel zoveel plaatsen naar rechts op te schuiven tot het gehele deel nul wordt. Telkens 1 bit wordt opgeschoven, moet de exponent met 1 worden vermeerderd. Na deze fase bekomt men volgende genormaliseerde vorm:

gehele deel	fractiedeel	teken en exponent
00000000	10100001 11010111 00001010 00111101	00000101

Door de fractie te beperken tot 3 bytes - eventueel afronden naar boven vanuit de vierde byte - en de exponent vooraan te plaatsen, wordt de gewenste binaire notatie van het floating point getal 20.23 bekomen.

exponent	fractie
00000101	10100001 11010111 00001010

Voorbeeld 2: Binaire representatie van het floating point getal -20.23

Het enige verschil met het floating point getal uit het vorige voorbeeld is het minteken. Dit wordt gecodeerd in bit 7 van de teken-exponentbyte, zodat de binaire representatie van -20.23 is:

teken en exponent	fractiedeel
10000101	10100001 11010111 00001010

Voorbeeld 3: Binaire representatie van het floating point getal 0.012

fase_1: Vermits het gehele deel 0 is, is deze fase hier overbodig

fase_2:

$$0.012 = 1 \times 10^{-2} + 2 \times 10^{-3}$$

1×10^{-2}	=	00000010 10001111 01011100 00101000
2×10^{-3}		00000000 10000011 00010010 01101110
0.012	=	00000011 00010010 01101110 10010110

We bekomen alzo de voorstelling:

geheel deel	fractiedeel	teken en exponent
00000000	00000011 00010010 01101110 10010110	00000000

fase_3: om deze vorm te normaliseren moeten de bits van de fractie 6 plaatsen naar links worden verschoven. De exponent vermindert hierbij telkens met 1, zodat na de verschuiving deze de waarde -6 heeft. In 2-complement notatie met 7 bits wordt dit: 1111010.

Na verschuiving krijgen we volgende representatie:

geheel deel	fractiedeel	teken en exponent
00000000	11000100 10011011 10100101 10000000	01111010

Als we het gehele deel in de notatie weglaten, de fractie beperken tot 3 bytes - met afronding vanuit de vierde byte - en de exponent voor de fractie plaatsen, bekomen we:

teken en exponent	fractiedeel
01111010	11000100 10011011 10100110

Voorbeeld 4: Binaire representatie van het floating point getal - 0.012

Door codering van het minteken in de notatie van vorig voorbeeld bekomen we:

11111010 11000100 10011011 10100110 xxxxx

8080 simulator

8 0 8 0 SIMULATOR

This 8080 simulator could be a fine and easy introduction into the world of assembly language.
There is a BASIC part (some kind of disassembler) and a machine language part.
Please save the BASIC program after typing in,before RUN !
After initialisation of the screen and DATA,the programs asks for a string on line 550.
560-570 : check for special command CLEAR or INIT
CLEAR : all registers (except PSW) zero.
INIT : user has to fill in reg. values.
please note the format : 50 = 50 decimal
:50 = 50 HEX
Most mnemonics are used following INTEL standard, except:
, becomes . (MOV A.B)
DATA is preceded by space,slash : MVI A /:FF
ADDRESS is preceded by space,slash : LXI H /:BFEF
1020-1170 : translate the mnemonic into HEX instruction.
1117 : special case : bit shift left
1118 : special case : bit shift right
After the matching instruction has been found, this value(s) is (are) POKED into some reserved locations of the ml program. (hex 32B- hex 32D).
If the user instructions (1,2 or 3 bytes) are on the right locations, there is a call to hex312 (line 540), to execute the ml program, including the users instruction.
The ml program takes the old regs values from locations hex300-hex308,executes the users instructions and updates actual and previous values on the screen.
The bit-contents on the left are AFTER the instruction, the bit-contents on the right are BEFORE the instruction.
We have to reserve some room for the ml program:
POKE HEX 29C,4:CLEAR 1000.
Now we can enter the ml program with substitute.(from hex 300-hex 3A4).

SIZE : #13F2 DATE : 8/1/81 AUTEUR : W.HERMANS TITLE8080 SIMUL

FPT	L1	screen ram location	10000-
FPT	L2	screen ram location	10000-
FPT	X	loop-variabele	
FPT	Y	loop-variabele	
INT	AD	address	30000-
INT	ARROW	x-pos for arrow (bit shift)	
INT	D	compute-variabele in SUB HEX	2500-
INT	EF	endflag : to check if instruction is complete	
INT	EFH	additional endflag	3500-
INT	F	loopvariabele	500
INT	FOUND	matchflag for "/"	1000-
INT	H	compute-variabele in SUB HEX	2500-
INT	HM	" " " " " "	
INT	I	loop-variabele	
INT	INSTR	hex-value of instruction	
INT	L	lenght STRING	1010
INT	LOC	location in user reserved bytes	
INT	OPE	variabele in HEX SUB	2500-
INT	ROUTINE	address of user reserved locations	535
INT	V	value of register	
INT	X	loop-variabele	
INT	XP	loop-variabele	
INT (ARR)	HEXL	number of bytes for instruction	
STR	A$	string	
STR	COM$	command string	
STR	F$	initiate screen	500-
STR	M$	MID$ of S$	
STR	REG$	INPUT IN INIT	3000-
STR	S$	GENERAL INPUT	
STR (ARR)	HEXL$	MNEMONICS	

8080 simulator

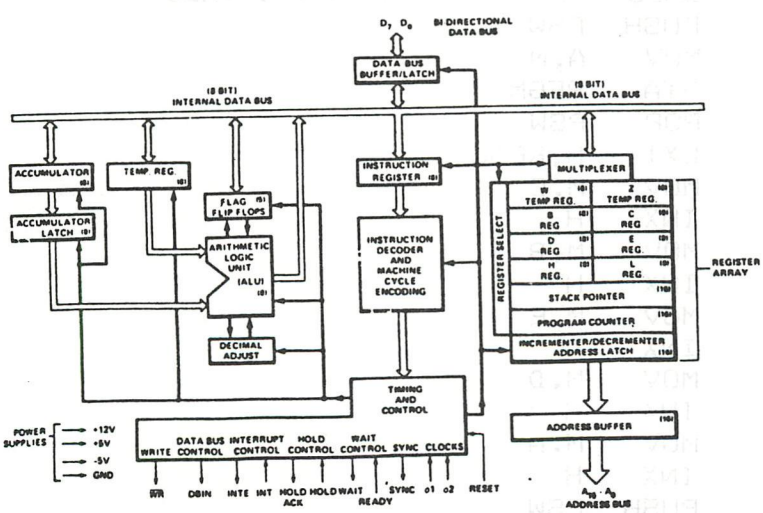
```
PAGE  1  --    ***8080 SIMULATOR***      V4.2                    --  FLST V2.0


1         REM ***8080 SIMULATOR***    V4.2
9         REM DATA LIST WITH MNEMONICS,LENGTH OF OPCODE, PLACE =
C         OPCODE
10        DATA NOP,0,LXI B,2,STAX B,0,INX B,0,INR B,0,DCR B,0,MVI B,1,
C         RLC,0,EMP,0,DAD B,0,LDAX B,0,DCX B,0,INR C,0,DCR C,0,MVI C,1,
C         RRC,0
20        DATA EMP,0,LXI D,2,STAX D,0,INX D,0,INR D,0,DCR D,0,MVI D,1,
C         RAL,0,EMP,0,DAD D,0,LDAX D,0,DCX D,0,INR E,0,DCR E,0,MVI E,1,
C         RAR,0
30        DATA EMP,0,LXI H,2,SHLD,2,INX H,0,INR H,0,DCR H,0,MVI H,1,
C         DAA,0,EMP,0,DAD H,0,LHLD,2,DCX H,0,INR L,0,DCR L,0,MVI L,1,
C         CMA,0
40        DATA EMP,0,LXI SP,2,STA,2,INX SP,0,INR M,0,DCR M,0,MVI M,1,
C         STC,0,EMP,0,DAD SP,0,LDA,2,DCX SP,0,INR A,0,DCR A,0,MVI A,1,
C         CMC,0
50        DATA MOV B.B,0,MOV B.C,0,MOV B.D,0,MOV B.E,0,MOV B.H,0,MOV
C         B.L,0,MOV B.M,0,MOV B.A,0
55        DATA MOV C.B,0,MOV C.C,0,MOV C.D,0,MOV C.E,0,MOV C.H,0,MOV
C         C.L,0,MOV C.M,0,MOV C.A,0
60        DATA MOV D.B,0,MOV D.C,0,MOV D.D,0,MOV D.E,0,MOV D.H,0,MOV
C         D.L,0,MOV D.M,0,MOV D.A,0
65        DATA MOV E.B,0,MOV E.C,0,MOV E.D,0,MOV E.E,0,MOV E.H,0,MOV
C         E.L,0,MOV E.M,0,MOV E.A,0
70        DATA MOV H.B,0,MOV H.C,0,MOV H.D,0,MOV H.E,0,MOV H.H,0,MOV
C         H.L,0,MOV H.M,0,MOV H.A,0
80        DATA MOV L.B,0,MOV L.C,0,MOV L.D,0,MOV L.E,0,MOV L.H,0,MOV
C         L.L,0,MOV L.M,0,MOV L.A,0
85        DATA MOV M.B,0,MOV M.C,0,MOV M.D,0,MOV M.E,0,MOV M.H,0,MOV
C         M.L,0,MOV M.M,0,MOV M.A,0
90        DATA MOV A.B,0,MOV A.C,0,MOV A.D,0,MOV A.E,0,MOV A.H,0,MOV
C         A.L,0,MOV A.M,0,MOV A.A,0
95        DATA ADD B,0,ADD C,0,ADD D,0,ADD E,0,ADD H,0,ADD L,0,ADD M,0,
C         ADD A,0
100       DATA ADC B,0,ADC C,0,ADC D,0,ADC E,0,ADC H,0,ADC L,0,ADC M,0,
C         ADC A,0
105       DATA SUB B,0,SUB C,0,SUB D,0,SUB E,0,SUB H,0,SUB L,0,SUB M,0,
C         SUB A,0
110       DATA SBB B,0,SBB C,0,SBB D,0,SBB E,0,SBB H,0,SBB L,0,SBB M,0,
C         SBB A,0
115       DATA ANA B,0,ANA C,0,ANA D,0,ANA E,0,ANA H,0,ANA L,0,ANA M,0,
C         ANA A,0
120       DATA XRA B,0,XRA C,0,XRA D,0,XRA E,0,XRA H,0,XRA L,0,XRA M,0,
C         XRA A,0
125       DATA ORA B,0,ORA C,0,ORA D,0,ORA E,0,ORA H,0,ORA L,0,ORA M,0,
C         ORA A,0
130       DATA CMP B,0,CMP C,0,CMP D,0,CMP E,0,CMP H,0,CMP L,0,CMP M,0,
C         CMP A,0
135       DATA RNZ,0,POP B,0,JNZ,2,JMP,2,CNZ,2,PUSH B,0,ADI,1,RST 0,0
140       DATA RZ,0, RET,0,JZ,2,EMP,0,CZ,2,CALL,2,ACI,1,RST 1,0
145       DATA RNC,0,POP D,0,JNC,2,OUT,1,CNC,2,PUSH D,0,SUI,1,RST 2,0
150       DATA RC,0,EMP,0,JC,2,IN,1,CC,2,EMP,0,SBI,1,RST 3,0
155       DATA RPO,0,POP H,0,JPO,2,XTHL,0,CPO,2,PUSH H,0,ANI,1,RST 4,0
160       DATA RPE,0,PCHL,0,JPE,2,XCHG,0,CPE,2,EMP,0,XRI,1,RST 5,0
165       DATA RP,0,POP PSW,0,JP,2,DI,0,CP,2,PUSH PSW,0,ORI,1,RST 6,0
170       DATA RM,0,SPHL,0,JM,2,EI,0,CM,2,EMP,0,CPI,1,RST 7,0
```

```
PAGE  2  --    ***8080 SIMULATOR***      V4.2                    --  FLST V2.0


175         DATA 9,S,11,Z,15,AC,20,P,24,CY

180         REM DATA FOR 8080 REG
190         DATA C,B,E,D,A,PSW,L,H,M,------
300         COLORT 8 0 12 0:MODE 0:POKE #75,#20
310         PRINT CHR$(12):CURSOR 17,12:PRINT "8 0 8 0   SIMULATOR"
315         POKE #75,32
320         CURSOR 3,2:PRINT "w.h. V4.1":CURSOR 40,2:PRINT "from
C           DAInamic"
360         CLEAR 4000:DIM HEXL$(255.0),HEXL(255.0)
500         FOR I=0 TO 255
510     1     READ HEXL$(I),HEXL(I)
520         NEXT
525         PRINT CHR$(12):FOR X=#B565 TO #B569 STEP 2:POKE X,#FF:POKE
C           X-3,#FF:NEXT
526         POKE #B9A6,#CD:POKE #B920,#C8:POKE #B5FC,#C7
527         CALLM #312:FOR F=1 TO 5:READ F,F$:CURSOR F,11:PRINT F$
528         CURSOR F+30,11:PRINT F$:NEXT:CURSOR 30,23:PRINT "#"
532         FOR X=0 TO 8:CURSOR 3,22-X*2:READ A$:PRINT A$:NEXT
533         CURSOR 40,3:PRINT "BYTE "
534         GOSUB 10000:REM REGISTER COLORS
535         LOC=#32B:ROUTINE=#312
537         GOSUB 5000:REM # & d
540         IF EF=1 THEN EF=0:CALLM #312:GOTO 535
545         CURSOR 21,3:PRINT SPC(15)
547         SOUND 1 0 15 0 FREQ(1000.0):WAIT TIME 5:SOUND OFF
550         CURSOR 10,3:INPUT "INSTRUCTIE ";S$
555         CURSOR 46,1:PRINT SPC(13):CURSOR 46,1:PRINT S$

560         REM SECIAL CASES
570         IF S$="CLEAR" THEN CURSOR 0,1:PRINT SPC(59):FOR X=#300 TO
C           #308:POKE X,0:NEXT:CALLM #312:GOSUB 5000:GOTO 545
575         IF S$="INIT" THEN GOSUB 3000:GOTO 535

1000        REM DECODE STRING
1005        REM FORMAT : MOV A.B    MVI A /:FF   MVI A /255
1010        L=LEN(S$)
1015        IF L=0 THEN 1115

1020        REM LOOK FOR "/"
1025        FOUND=0
1030        FOR X=0 TO L-1
1040    1     M$=MID$(S$,X,1)
1050    1     IF M$="/" THEN FOUND=X
1060        NEXT
1070        IF FOUND=0 THEN COM$=S$:GOTO 1090
1080        COM$=LEFT$(S$,FOUND-1)
1090        FOR X=0 TO 255
1100    1     IF COM$=HEXL$(X) THEN INSTR=X:GOTO 1117
1110        NEXT
1115        FOR X=1 TO 5:POKE #B576,#F3:WAIT TIME 10:POKE #B576,#F8:WAIT
C           TIME 10:NEXT:GOTO 540
1117        IF INSTR=7 OR INSTR=#17 THEN GOSUB 20000:REM LEFT
1118        IF INSTR=#F OR INSTR=#1F THEN GOSUB 21000:REM RIGHT
1120        CURSOR 46,3:PRINT HEX$(INSTR):CURSOR 37,3:PRINT HEXL(X)+1:
```

```
PAGE  3  --    ***8080 SIMULATOR***      V4.2                    --  FLST V2.0


C           POKE LOC,INSTR:IF HEXL(INSTR)=0 THEN EF=1:GOTO 540
1130        M$=MID$(S$,FOUND+1,1)
1140        S$=RIGHT$(S$,L-1-FOUND)
1142        CURSOR 48,3:PRINT SPC(7)
1145        CURSOR 48,3:PRINT S$:IF M$<>":" THEN OPE=VAL(S$)
1150        IF M$=":" THEN GOSUB 2500:REM HEX
1160        IF HEXL(INSTR)=1 THEN POKE LOC+1,OPE:EF=1:GOTO 540
1170        IF HEXL(INSTR)=2 THEN POKE LOC+1,(OPE IAND 255):POKE LOC+2,
C           (OPE SHR 8):EF=1:GOTO 540

2500        REM SUB INPUT HEX
2510        OPE=0:HM=0
2565        FOR I=LEN(S$)-1 TO 1 STEP -1
2570    1     H=ASC(MID$(S$,I,1))
2575    1     IF H>47 AND H<58 THEN D=H-48:GOTO 2600
2580    1     IF H>64 AND H<71 THEN D=H-55:GOTO 2600
2590    1     PRINT "   ERROR HEX NUMBER":EFH=1:RETURN
2600    1     OPE=OPE+D*(#10^HM)+0.5
2605    1     HM=HM+1
2610        NEXT
2630        RETURN

3000        REM INIT
3010        XP=0:AD=#300:REG$="C":GOSUB 3500
3020        XP=7:AD=AD+1:REG$="B":GOSUB 3500
3030        XP=14:AD=AD+1:REG$="E":GOSUB 3500
3040        XP=21:AD=AD+1:REG$="D":GOSUB 3500
3050        XP=28:AD=AD+1:REG$="A":GOSUB 3500
3060        XP=35:AD=AD+2:REG$="L":GOSUB 3500
3070        XP=42:AD=AD+1:REG$="H":GOSUB 3500
3075        CURSOR 0,1:PRINT SPC(59)
3080        RETURN
3500        CURSOR XP,1:PRINT " ";REG$;:INPUT S$
3510        IF LEFT$(S$,1)=":" THEN GOSUB 2500
3515        IF EFH=1 THEN EFH=0:RETURN
3520        IF LEFT$(S$,1)<>":" THEN OPE=VAL(S$)
3530        IF OPE>255 THEN RETURN
3540        POKE AD,OPE
3545        CALLM #312
3547        GOSUB 5000
3550        RETURN
5000        FOR X=0 TO 8:CURSOR 28,22-X*2:V=PEEK(#300+X):PRINT SPC(10)
5010    1     CURSOR 28,22-X*2:PRINT
C           SPC(3-LEN(HEX$(V)));HEX$(V);SPC(6-LEN(STR$(V)));V:NEXT
5020        RETURN
10000       FOR X!=0.0 TO 8.0
10005   1     L1!=48974.0-268.0*X!
10006   1     L2!=L1!-60.0
10010   1     FOR Y!=0.0 TO 17.0
10020   2       POKE L1!-Y!*2,#FF
10030   2       POKE L2!-Y!*2,#FF
10040       NEXT:NEXT:RETURN
20000       FOR ARROW=24 TO 9 STEP -1
20010   1     CURSOR ARROW,15:PRINT CHR$(136):WAIT TIME 5:CURSOR ARROW,15:
C       1     PRINT CHR$(32)
```

```
PAGE  4  --    ***8080 SIMULATOR***     V4.2                    --  FLST V2.0


20020      NEXT:RETURN
21000      FOR ARROW=9 TO 24
21010   1    CURSOR ARROW,15:PRINT CHR$(137):WAIT TIME 5:CURSOR ARROW,15:
C       1    PRINT CHR$(32)
21020      NEXT:RETURN
```

```
Ö000 00

0310 00 00 E5 C5 D5 F5 21 13 BF CD 5D 03 21 00 03 4E
0320 23 46 23 5E 23 56 23 7E 2A 06 03

032B 00 00 00

032E 22 06
0330 03 F5 7E 32 08 03 F1 21 00 03 71 23 70 23 73 23
0340 72 23 77 23 F5 D1 73 21 4F BF CD 5D 03 21 2B 03
0350 36 00 23 36 00 23 36 00 F1 D1 C1 E1 C9 3E 09 32
0360 00 00 11 00 03 22 10 03 1A CD 82 03 13 2A 10 03
0370 01 F4 FE 09 22 10 03 3A 00 00 3D 32 00 00 C2 68
0380 03 C9 06 08 B7 17 DC 9F 03 D4 A2 03 2B 2B 2B 2B
0390 F5 3E 05 B8 C2 99 03 2B 2B F1 05 C2 84 03 C9 36
03A0 31 C9 36 30 C9
```



8080 simulator

```
PAGE 01     :8080 SIMULATOR

002                     *USER INSTRUCTIONS ON RESERVED LOCATIONS
003                     REGC    EQU   :300       C
004                     REGB    EQU   :301       B
005                     REGE    EQU   :302       E
006                     REGD    EQU   :303       D
007                     REGA    EQU   :304       A
008                     REGP    EQU   :305       PSW
009                     REGL    EQU   :306       L
010                     REGH    EQU   :307       H
011                     REGM    EQU   :308       M (HL)
012                     *
013                     TLEFT   EQU   48975      TOPLEFT OF OUTPUT~BLOCK
014                     TRIGHT  EQU   48915      (ADAPT for RAM-SIZE)
015 0000 00             COUNT   DATA  0
016                     *
017                             ORG   :310
018 0310 0000           CURSOR  DBL   0          INFO FOR OUTPUT TO SCREEN
019 0312 E5                     PUSH  H
020 0313 C5                     PUSH  B
021 0314 D5                     PUSH  D
022 0315 F5                     PUSH  PSW
023 0316 2113BF                 LXI   H,TRIGHT
024 0319 CD5D03                 CALL  DISPL      *DISPL PREVIOUS VALUES
025 031C 210003                 LXI   H,REGC     *GET PREVIOUS VALUES
026 031F 4E                     MOV   C,M
027 0320 23                     INX   H
028 0321 46                     MOV   B,M
029 0322 23                     INX   H
030 0323 5E                     MOV   E,M
031 0324 23                     INX   H
032 0325 56                     MOV   D,M
033 0326 23                     INX   H
034 0327 7E                     MOV   A,M
035 0328 2A0603                 LHLD  REGL
036 032B                USER    RES   3,0        *USER ROUTINE
037 032E 220603                 SHLD  REGL       *UPDATE VALUES
038 0331 F5                     PUSH  PSW
039 0332 7E                     MOV   A,M
040 0333 320803                 STA   REGM
041 0336 F1                     POP   PSW
042 0337 210003                 LXI   H,REGC
043 033A 71                     MOV   M,C
044 033B 23                     INX   H
045 033C 70                     MOV   M,B
046 033D 23                     INX   H
047 033E 73                     MOV   M,E
048 033F 23                     INX   H
049 0340 72                     MOV   M,D
050 0341 23                     INX   H
051 0342 77                     MOV   M,A
052 0343 23                     INX   H
053 0344 F5                     PUSH  PSW
054 0345 D1                     POP   D
```

```
PAGE 02    :8080 SIMULATOR

055 0346 73                    MOV   M,E
056 0347 214FBF                LXI   H,TLEFT    *UPDATED VALUES
057 034A CD5D03                CALL  DISPL      *SEND TO SCREEN
058 034D 212B03                LXI   H,USER     * CLEAR USER INSTRUCTION(S)
059 0350 3600                  MVI   M,0
060 0352 23                    INX   H
061 0353 3600                  MVI   M,0
062 0355 23                    INX   H
063 0356 3600                  MVI   M,0
064 0358 F1                    POP   PSW
065 0359 D1                    POP   D
066 035A C1                    POP   B
067 035B E1                    POP   H
068 035C C9                    RET
069 035D 3E09          DISPL   MVI   A,9
070 035F 320000                STA   COUNT
071 0362 110003                LXI   D,REGC
072 0365 221003                SHLD  CURSOR
073 0368 1A            NEXT    LDAX  D
074 0369 CD8203                CALL  DBYTE
075 036C 13                    INX   D
076 036D 2A1003                LHLD  CURSOR
077 0370 01F4FE                LXI   B,-268     OFFSET 2 LINES
078 0373 09                    DAD   B
079 0374 221003                SHLD  CURSOR
080 0377 3A0000                LDA   COUNT
081 037A 3D                    DCR   A
082 037B 320000                STA   COUNT
083 037E C26803                JNZ   NEXT
084 0381 C9                    RET
085 0382 0608          DBYTE   MVI   B,8
086 0384 B7            BNEXT   ORA   A
087 0385 17                    RAL
088 0386 DC9F03                CC    ONE
089 0389 D4A203                CNC   ZERO
090 038C 2B                    DCX   H
091 038D 2B                    DCX   H
092 038E 2B                    DCX   H
093 038F 2B                    DCX   H
094 0390 F5                    PUSH  PSW
095 0391 3E05                  MVI   A,5        *NIBBLE-SPACE ?
096 0393 B8                    CMP   B
097 0394 C29903                JNZ   NSPACE
098 0397 2B                    DCX   H
099 0398 2B                    DCX   H
100 0399 F1            NSPACE  POP   PSW
101 039A 05                    DCR   B
102 039B C28403                JNZ   BNEXT
103 039E C9                    RET
104 039F 3631          ONE     MVI   M,:31      *"1"
105 03A1 C9                    RET
106 03A2 3630          ZERO    MVI   M,:30      *"0"
107 03A4 C9                    RET
```

```
PAGE 03     :8080 SIMULATOR

108 03A5                        END

****************************
* S Y M B O L   T A B L E *
****************************

BNEXT   0384    COUNT   0000    CURSOR 0310    DBYTE   0382
DISPL   035D    NEXT    0368    NSPACE 0399    ONE     039F
REGA    0304    REGB    0301    REGC   0300    REGD    0303
REGE    0302    REGH    0307    REGL   0306    REGM    0308
REGP    0305    TLEFT   BF4F    TRIGHT BF13    USER    032B
ZERO    03A2
```

TABLE 1

INSTRUCTION	FUNCTION		HEX
MOVE GROUP		reg A B C D E H L M	
MOV A, reg	(A)←(reg)		7F 78 79 7A 7B 7C 7D 7E
MOV B, reg	(B)←(reg)		47 40 41 42 43 44 45 46
MOV C, reg	(C)←(reg)		4F 48 49 4A 4B 4C 4D 4E
MOV D, reg	(D)←(reg)		57 50 51 52 53 54 55 56
MOV E, reg	(E)←(reg)		5F 58 59 5A 5B 5C 5D 5E
MOV H, reg	(H)←(reg)		67 60 61 62 63 64 65 66
MOV L, reg	(L)←(reg)		6F 68 69 6A 6B 6C 6D 6E
MOV M, reg	(M)←(reg)		77 70 71 72 73 74 75 —
ACCUMULATOR GROUP			
ADD reg	(A)←(A)+(reg)	*	87 80 81 82 83 84 85 86
ADC reg	(A)←(A)+(reg)+(CY)	*	8F 88 89 8A 8B 8C 8D 8E
SUB reg	(A)←(A)-(reg)	*	97 90 91 92 93 94 95 96
SBB reg	(A)←(A)-(reg)-(CY)	*	9F 98 98 9A 9B 9C 9D 9E
ANA reg	(A)←(A)∧(reg)	*	A7 A0 A1 A2 A3 A4 A5 A6
XRA reg	(A)←(A)∨(reg)	*	AF A8 A9 AA AB AC AD AE
ORA reg	(A)←(A)∨(reg)	*	B7 B0 B1 B2 B3 B4 B5 B6
CMP reg	(A)-(reg)	*	BF B8 B9 BA BB BC BD BE
INCREMENT/DECREMENT REGISTER			
INR reg	(reg)←(reg)+1	**	3C 04 0C 14 1C 24 2C 34
DCR reg	(reg)←(reg)-1	**	3D 05 0D 15 1D 25 2D 35
REGISTER PAIR GROUP		rp	B D H SP PSW
INX rp	(rp)←(rp)+1		03 13 23 33 —
DCX rp	(rp)←(rp)-1		0B 1B 2B 3B —
LDAX rp	(A)←((rp))		0A 1A — — —
STAX rp	((rp))←(A)		02 12 — — —
DAD rp	(H,L)←(H,L)+(rp) ***		09 19 29 39 —
PUSH rp*	((SP)-1)←(rh),((SP)-2)←(rl), (SP)←(SP)-2		C5 D5 E5 — F5
POP rp*	(rl)←((SP)),(rh)←((SP)+1), (SP)←(SP)+2		C1 D1 E1 — F1 *
DIRECT ADDRESS GROUP			
LDA addr	(A)←(addr)		3A al ah
STA addr	(addr)←(A)		32 al ah
LHLD addr	(L)←(addr),(H)←(addr+1)		2A al ah
SHLD addr	(addr)←(L),(addr+1)←(H)		22 al ah
IMMEDIATE GROUP			
MVI A, data	(A)←data		3E dd
MVI B, data	(B)←data		06 dd
MVI C, data	(C)←data		0E dd
MVI D, data	(D)←data		16 dd
MVI E, data	(E)←data		1E dd
MVI H, data	(H)←data		26 dd
MVI L, data	(L)←data		2E dd
MVI M, data	(M)←data		36 dd
ADI data	(A)←(A)+data	*	C6 dd
ACI data	(A)←(A)+data+(CY)	*	CE dd
SUI data	(A)←(A)-data	*	D6 dd
SBI data	(A)←(A)-data-(CY)	*	DE dd
ANI data	(A)←(A)∧data	*	E6 dd
XRI data	(A)←(A)∨data	*	EE dd
ORI data	(A)←(A)∨data	*	F6 dd
CPI data	(A)-data	*	FE dd
LXI B, addr	(B)←ah,(C)←al		01 al ah
LXI D, addr	(D)←ah,(E)←al		11 al ah
LXI H, addr	(H)←ah,(L)←al		21 al ah
~~LXI SP,addr~~	~~(SP$_h$)←ah,(SP$_l$)←al~~	*	31 al ah

INSTRUCTION	FUNCTION		HEX
JUMP GROUP			
JMP addr*	(PC)←addr		C3 al ah
JNZ addr*	If Z=0, (PC)←addr		C2 al ah
JZ addr*	If Z=1, (PC)←addr		CA al ah
JNC addr*	If CY=0, (PC)←addr		D2 al ah
JC addr*	If CY=1, (PC)←addr		DA al ah
JPO addr*	If P=0, (PC)←addr		E2 al ah
JPE addr*	If P=1, (PC)←addr		EA al ah
JP addr*	If S=0, (PC)←addr		F2 al ah
JM addr*	If S=1, (PC)←addr		FA al ah
PCHL *	(PC$_h$)←(H), (PC$_l$)←(L)		E9
CALL GROUP			
CALL addr*	(TOS)←(PC),(PC)←addr		CD al ah
CNZ addr*	If Z=0, (TOS)←(PC),(PC)←addr		C4 al ah
CZ addr*	If Z=1, (TOS)←(PC),(PC)←addr		CC al ah
CNC addr*	If CY=0, (TOS)←(PC),(PC)←addr		D4 al ah
CC addr*	If CY=1, (TOS)←(PC),(PC)←addr		DC al ah
CPO addr*	If P=0, (TOS)←(PC),(PC)←addr		E4 al ah
CPE addr*	If P=1, (TOS)←(PC),(PC)←addr		EC al ah
CP addr*	If S=0, (TOS)←(PC),(PC)←addr		F4 al ah
CM addr*	If S=1, (TOS)←(PC),(PC)←addr		FC al ah
	N.B. (TOS)←(PC) designates the following:- ((SP)-1)←(PC$_h$),((SP)-2)←(PC$_l$),(SP)←(SP)-2		
RETURN GROUP			
RET *	(PC)←(TOS)		C9
RNZ *	If Z=0, (PC)←(TOS)		C0
RZ *	If Z=1, (PC)←(TOS)		C8
RNC *	If CY=0, (PC)←(TOS)		D0
RC *	If CY=1, (PC)←(TOS)		D8
RPO *	If P=0, (PC)←(TOS)		E0
RPE *	If P=1, (PC)←(TOS)		E8
RP *	If S=0, (PC)←(TOS)		F0
RM *	If S=1, (PC)←(TOS)		F8
	N.B. (PC)←(TOS) designates the following:- (PC$_l$)←((SP)),(PC$_h$)←((SP)+1),(SP)←(SP)+2		
RESTART GROUP			
RST 0 *	(TOS)←(PC),(PC)←0$_{16}$		C7
RST 1 *	(TOS)←(PC),(PC)←8$_{16}$		CF
RST 2 *	(TOS)←(PC),(PC)←10$_{16}$		D7
RST 3 *	(TOS)←(PC),(PC)←18$_{16}$		DF
RST 4 *	(TOS)←(PC),(PC)←20$_{16}$		E7
RST 5 *	(TOS)←(PC),(PC)←28$_{16}$		EF
RST 6 *	(TOS)←(PC),(PC)←30$_{16}$		F7
RST 7 *	(TOS)←(PC),(PC)←38$_{16}$		FF
ROTATE/CONTROL/SPECIAL GROUP			
RLC	(A$_{n+1}$)←(A$_n$),(A$_0$)←(A$_7$),(CY)←(A$_7$)	***	07
RRC	(A$_n$)←(A$_{n+1}$),(A$_7$)←(A$_0$),(CY)←(A$_0$)	***	0F
RAL	(A$_{n+1}$)←(A$_n$),(A$_0$)←(CY),(CY)←(A$_7$)	***	17
RAR	(A$_n$)←(A$_{n+1}$),(A$_7$)←(CY),(CY)←(A$_0$)	***	1F
NOP	No operation		00
HLT *	Processor stopped until interrupt or reset		76
DI	Interrupts disabled		F3
EI *	Interrupts enabled after next instruction		FB
XTHL	(L)↔((SP)),(H)↔((SP)+1)		E3
SPHL*	(SP$_h$)←(H),(SP$_l$)←(L)		F9
XCHG	(H)↔(D),(L)↔(E)		EB
DAA	Decimal adjust accumulator	*	27
CMA	(A)←$\overline{(A)}$		2F
STC	(CY)←1	***	37
CMC	(CY)←$\overline{(CY)}$	***	3F
OUT port	Not used in DCE Systems		D3 port
IN port			DB port

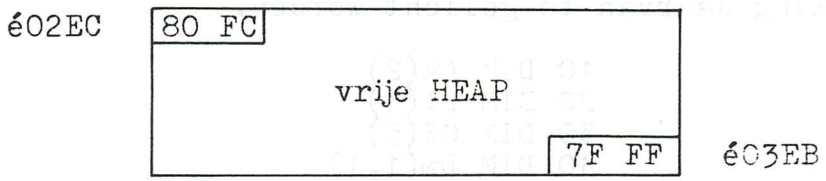
HET GEBRUIK VAN DE "HEAP".

1. De HEAP (in goed nederlands: 'hoop') is dat gedeelte van het RAM dat door een BASIC programma gebruikt wordt om strings en arrays op te slaan.

   De HEAP begint op adres é02EC. Bij een reset (of power-on) worden voor de HEAP 256 bytes gereserveerd. De grootte van de HEAP staat in de HEAPsize pointer é029D/E (é010C).

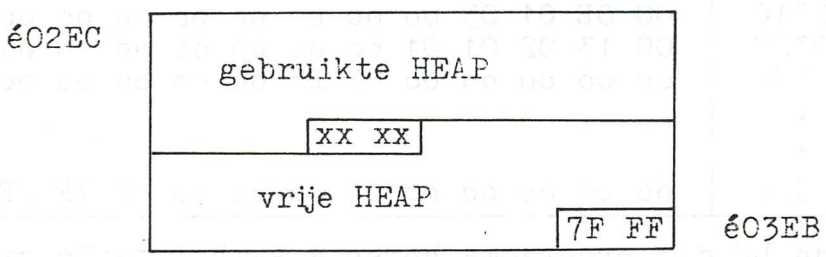
   Van het aantal voor de HEAP gereserveerde bytes gebruikt de HEAP er zelf vier: twee om het einde van de HEAP aan te geven (en tevens de maximale grootte) en twee om de nog vrije HEAP ruimte te berekenen.

   Na het inschakelen van de DAI ziet de HEAP er als volgt uit:



   De vrije HEAP-ruimte wordt berekend uit: 7FFF + 80FC = 00FB. Er zijn dus 252 vrije bytes ter beschikking.

   Als (bv. na RUN van een programma) een gedeelte van de HEAP is gebruikt, is de pointer voor het berekenen van de vrije ruimte aangepast en opgeschoven tot achter het laatste gebruikte byte:



2. CLEAR commando:

   De grootte van de HEAP kan met behulp van het CLEAR commando aangepast worden aan het aantal en de grootte van de arrays en de strings die in een programma gebruikt worden.

   bijv. CLEAR 1000 : 1000 bytes worden voor de HEAP gereserveerd. Er zijn dan dus 996 bytes ter beschikking.

   Het getal 1000 in het CLEAR commando heeft een decimale waarde !

   Na uitvoering van dit CLEAR commando is de HEAPsize pointer é029D/E aangepast (é03E8). De tekstbuffer - waarin het BASIC programma staat - en de symbol table zijn naar een hoger RAM adres opgeschoven.

   De maximale grootte van de HEAP is 32767 bytes (é7FFF).

Het is goed elk programma met CLEAR xxxx te beginnen. Dit hoeft slechts één keer in een programma gedaan te worden. Alleen een reset of power-on brengen de HEAP terug tot 256 bytes. Een NEW commando doet dit niet !

3. ARRAYS:

Wanneer in een programma arrays gebruikt worden, dan moet hiervoor eerst geheugenruimte gereserveerd worden. Deze ruimte wordt gezocht in de HEAP. D.m.v. een CLEAR commando moet gezorgd worden dat de HEAP voldoende ruimte heeft.

Het dimensioneren van de geheugenruimte voor arrays wordt gedaan met het DIM commando.

Aan de hand van onderstaand programma voorbeeld zal de werking daarvan toegelicht worden.

```
10 DIM A%(2)
20 DIM B!(3)
30 DIM C$(5)
40 DIM D%(1,1)
```

Na een RUN van dit programma ziet de HEAP er als volgt uit:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
02E0													00	0E	01	02
02F0	00	00	00	00	00	00	00	00	00	00	00	00	00	12	01	03
0300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0310	00	0E	01	05	00	00	00	00	00	00	00	00	00	00	00	00
0320	00	13	02	01	01	00	00	00	00	00	00	00	00	00	00	00
0330	00	00	00	00	00	80	B3	00	00	00	00	00	00	00	00	00
.																
.																
03E0	00	00	00	00	00	00	00	00	00	00	7F	FF				

In de bij dit programma horende symbol table staat dan:

```
51 41 52 EE 02 41 42 42 FE 02 61 43 62 12 03
51 44 52 22 03
```

(Het adres waar de symbol table begint staat in de pointer op adres é02A1/2).

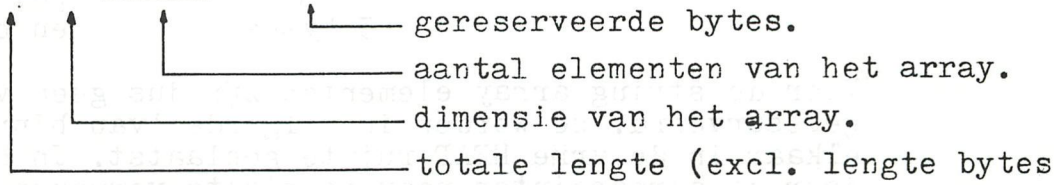
De symbol table geeft aan welke variabelen gebruikt zijn en waar in de HEAP ruimte voor de arrays is gereserveerd:

A% - 51 41 52 EE 02 - A (41), % (51 .. 52) op adres é02EE.
B! - 41 42 42 FE 02 - B (42), ! (41 .. 42) op adres é02FE.
C$ - 61 43 62 12 03 - C (43), $ (61 .. 62) op adres é0312.
D% - 51 44 52 22 03 - D (44), % (51 .. 52) op adres é0322.

In de HEAP is nu de ruimte é02EC tot é0334 gebruikt. Op é335/6 staat nu de pointer é80B3. Door dit op te tellen bij é7FFF wordt de vrije HEAP ruimte gevonden: éB3 (179 bytes).

De reservering in de HEAP voor de verschillende variabelen is als volgt:

```
A%(2)   - 00 0E 01 02      + 12 bytes.
B!(3)   - 00 12 01 03      + 16 bytes.
C$(5)   - 00 0E 01 05      + 12 bytes.
D%(1,1) - 00 13 02 01 01   + 16 bytes.
```



3.1. Voor INTEGER en FLOATING POINT ARRAYS worden per element 4 bytes gereserveerd. Wordt in het programma op een bepaald moment een waarde toegekend, dan wordt die waarde direkt in de gereserveerde bytes geplaatst.

```
A%(2) - 00 0E 01 02 oo oo oo oo oo oo oo oo oo oo oo oo
                      A%(0)       A%(1)       A%(2)
```

één dimensionaal array met 3 elementen.
totale lengte 14 bytes.

Indien nodig, kan het adres van een array element opgevraagd worden met bijv.: PRINT HEX$(VARPTR(A%(1))).

Wordt in het programma aan A%(1) de waarde 10 toegekend, dan staat in het array:

```
A%(2) - 00 0E 01 02 oo oo oo oo 00 00 00 0A oo oo oo oo
```

3.2. STRING ARRAYS worden op een heel andere wijze behandeld. Bij het dimensioneren van een string array worden per element 2 bytes in het array gereserveerd:

```
C$(5) - 00 0E 01 05 oo oo oo oo oo oo oo oo oo oo oo oo
                    C$(0) C$(1) C$(2) C$(3) C$(4) C$(5)
```

één dimensionaal array met 6 elementen.
totale lengte 14 bytes.

De twee bytes die per element gereserveerd zijn, worden gebruikt als adrespointers. Bij het dimensioneren wordt hier nog niets ingevuld. Wordt echter in het programma op een bepaald moment aan één van de elementen een alfa-numerieke waarde toegekend, dan gebeurt het volgende:

C$(1) = "DAI" geeft de volgende veranderingen in de HEAP te zien:

```
C$(5) - 00 0E 01 05 oo oo 36 03 oo oo oo oo oo oo oo oo
```

pointer naar de plaats in de HEAP waar de string is neergezet.

De string "DAI" is nu geplaatst op adres é0336:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0330	oo	oo	oo	oo	oo	00	03	44	41	49	80	AE	oo	oo	oo	oo

D A I

3 bytes

pointer aangepast en opgeschoven.

Voor de string array elementen zijn dus geen vaste plaatsen gereserveerd; ze worden in volgorde 'van binnenkomst' achter elkaar in de vrije HEAP ruimte geplaatst. In het array wordt door de adrespointer naar de plaats verwezen waar de string is neergezet.

Wordt in de loop van het programma de string van lengte veranderd, dan schuift alles wat in de HEAP na deze string komt gewoon op en worden de bij deze strings behorende adrespointers aangepast.

4. STRINGS.

Behalve voor arrays wordt de HEAP ook gebruikt om 'gewone' strings in op te slaan. Dit gebeurt overeenkomstig de behandeling van string-arrays. Het volgende voorbeeld maakt dit duidelijk.

```
10  A$ = "DAI"
```

Na RUN van dit programma staat in de symbol table:

21 41 22 ED 02 - A (41), $ (21 .. 22) op adres é02ED.

In de HEAP staat dan:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
02E0													00	03	44	41
02F0	49	80	F7													

5. Opmerkingen:

- Wanneer meer HEAP ruimte gebruikt wordt dan gedimensioneerd is, wordt de error-melding "OUT OF STRING SPACE" gegeven.
- In principe wordt bij een CLEAR alle array elementen op nul gezet. Toch is het verstandig aan het begin van het programma alle elementen met een FOR ... NEXT loop te initialiseren.
- Wanneer een array niet gedimensioneerd wordt, dan volgt bij RUN van het programma de foutmelding "UNDEFINED ARRAY IN LINE xx".
- Tijdens een programma kan de dimensionering van een array zonder problemen veranderd worden door een nieuw DIM commando.

Jan Boerrigter/juni '81

16 colors in 4-color modes

```
1     REM PROGRAMMA OM TE LATEN ZIEN DAT HET OOK MOGELYK IS OM
2     REM 16 KLEUREN TE KRYGEN IN EEN 4 KLEUREN MODE
3     REM ************************************************
4     REM *                                              *
5     REM *                  JAAP MOL      18-04-1981    *
6     REM *                  VELDWEG 123                 *
7     REM *                  WESTZAAN                    *
8     REM *                                              *
9     REM ************************************************
10    MODE 2A:REM ..........  ANDERE MODE ANDERE WAARDEN !!!
20    COLORG 0 10 9 5:REM ... IN KLEUR 10 KOMT DE POKE
25    RAM=#4000:REM .......... OFFSET VOOR 48K
30    P1=31480.0+RAM:REM .... SCREENBYTE RECHTSONDER
40    P2=32750.0+RAM:REM .... SCREENBYTE LINKSBOVEN
50    P3=31502.0+RAM:REM .... EERSTE LINE CONTROL BYTE
55    REM ..................  NEEMT STEEDS MET 24 BYTES TOE
60    REM ..................  HIERMEE KAN DE ACTERGROND
65    REM ..................  OF HET OBJECT ZELF VAN KLEUR
70    REM ..................  VERANDERD WORDEN // + MEER
75    REM ..................  ACHTERGROND  191 < POKE < 208
80    REM ..................  OBJECT 207 < POKE < 224
90    P4=31584.0+RAM:REM .... ZOMAAR EEN SCREENBYTE
100   POKE P4,255:REM ....... 255 BINAIR OP 'T SCHERM
110   FOR X=6.0 TO XMAX STEP 10.0
112   DRAW X,0 X,YMAX 9
114   DRAW X-2,0 X-2,YMAX 10
116   DRAW X-4,0 X-4,YMAX 5
120   NEXT
130   FILL 10,30 40,40 10
140   FILL 20,0 30,40 10
185   GOTO 400
190   WAIT TIME 100
200   REM OBJECT KLEUR WISSELING
210   PLAATS=P3+10.0*24.0:REM LINE CONTROL BYTE 10
220   FOR X=208.0 TO 223.0
230   POKE PLAATS,X:REM ..... OBJECT RWISSELT VAN KLEUR
240   WAIT TIME 50
250   NEXT
300   REM ACHTERGROND KLEUR WISSELING
310   PLAATS=P3+10.0*24.0:REM LINE CONTROL BYTE 10
320   FOR X=192.0 TO 207.0
330   POKE PLAATS,X:REM ..... ACHTERGROND WISSELT VAN KLEUR
340   WAIT TIME 50
350   NEXT
400   REM ALLES DOOR ELKAAR
405   L=207.0
410   FOR X=P3 TO P3+50.0*24.0 STEP 24.0
420   POKE X,L
430   L=L-1.0
435   IF L=191.0 THEN L=207.0
440   NEXT
450   L=223.0
460   FOR X=P3 TO P3+50.0*24.0 STEP 24.0
470   POKE X,L
475   WAIT TIME 25
480   L=L-1.0
485   IF L=207.0 THEN L=223.0
490   NEXT
999   DRAW 0,0 XMAX,YMAX 3:REM GEEFT COLOR NOT AVAILABLE
```

talk editor

```
PAGE  1  --    TALK EDITOR Sip/Hermans                    --  FLST V2.0


5         REM TALK EDITOR Sip/Hermans
10        REM INIT SCREEN
11        POKE #75,32
15        MODE 0:CLEAR 2000:PRINT CHR$(12);
20        INPUT "ADRESSE D'IMPLANTATION (#6000) STARTADDRESS FOR TABLE
C         ";AD:START=AD:PRINT CHR$(12);
21        PRINT :PRINT :PRINT TAB(27);"TALK":PRINT :PRINT
30        PRINT "COMMANDS:"
40        PRINT "0 1 2 ..... FREQUENCE CHANNELS 0 1 2"
50        PRINT "A B C ..... VOLUME CHANNELS 0 1 2"
60        PRINT "N ......... VOLUME NOISE"
70        PRINT "D  msec.....DELAY"
80        PRINT "E ......... END"
90        PRINT "G ......... GO"
110       PRINT "X ......... NEW"
115       PRINT "L ......... LIST"

120       REM INPUT
122       POKE #B78E,#CC
124       CURSOR 0,2:PRINT "COMMAND:?";
125       CURSOR 50,6:PRINT "#";HEX$(AD);
130       A=GETC:IF A=0 THEN 130

140       REM DECODE
145       IF A=ASC("X") THEN GOTO 10
150       IF A=ASC("G") THEN POKE AD,#FF:TALK #6000:SOUND OFF :GOTO
C         120
160       IF A=ASC("0") THEN 1000
170       IF A=ASC("1") THEN 1100
180       IF A=ASC("2") THEN 1200
190       IF A=ASC("A") THEN 2000
200       IF A=ASC("B") THEN 2100
210       IF A=ASC("C") THEN 2200
220       IF A=ASC("N") THEN 2300
230       IF A=ASC("E") THEN POKE AD,#FF:GOTO 120
240       IF A=ASC("D") THEN 3000
250       IF A=ASC("L") THEN POKE AD,#FF:GOTO 5000:REM DISPLAY
260       GOTO 120

1000      REM CHANNEL 0
1003      POKE AD,0:AD=AD+1
1005      CURSOR 0,3:PRINT "CH 0:";
1010      CURSOR 6,3:INPUT "FREQUENCE:";FR
1020      IF FR<31 OR FR>#FFFF THEN 1010
1030      FR1=FREQ(FR)
1040      A1=FR1 SHR 8:A2=FR1 IAND #FF
1041      GOSUB 3100
1050      POKE AD,A2:AD=AD+1:POKE AD,A1:AD=AD+1:GOTO 120

1100      REM CHANNEL 1
1105      POKE AD,2:AD=AD+1:CURSOR 0,3:PRINT "CH 1:";:GOTO 1010

1200      REM CHANNEL 2
1205      POKE AD,4:AD=AD+1:CURSOR 0,3:PRINT "CH 2:";:GOTO 1010
```

```
PAGE  2  --    TALK EDITOR Sip/Hermans                    --  FLST V2.0


2000        REM VOLUME CHANNEL 0
2003        POKE AD,#8:AD=AD+1
2005        CURSOR 0,4:PRINT "CH 0:";
2010        CURSOR 6,4:INPUT "VOLUME:";VO
2020        IF VO<0 OR VO>#F THEN 2010
2025        GOSUB 3100
2030        POKE AD,VO:AD=AD+1:GOTO 120

2100        REM VOLUME CHANNEL 1
2105        POKE AD,#9:AD=AD+1:CURSOR 0,4:PRINT "CH 1:";:GOTO 2010

2200        REM VOLUME CHANNEL 2
2205        POKE AD,#A:AD=AD+1:CURSOR 0,4:PRINT "CH 2:";:GOTO 2010

2300        REM VOLUME NOISE
2305        POKE AD,#B:AD=AD+1:CURSOR 0,4:PRINT "NOIS:";:GOTO 2010

3000        REM DELAY
3005        POKE AD,#C:AD=AD+1
3010        CURSOR 0,5:INPUT "DELAY:";DL
3020        IF DL<0 OR DL>#FFFF THEN 3010
3030        D1=INT(DL/256.0):D2=DL IAND #FF
3031        GOSUB 3100
3040        POKE AD,D1:AD=AD+1:POKE AD,D2:AD=AD+1:GOTO 120
3100        FOR X=0 TO 6:CURSOR 0,X:PRINT SPC(59);:NEXT:RETURN

5000        REM LIST
5010        PRINT CHR$(12);
5030        DI=START
5040        IF CURY<1 THEN IF GETC<>32 THEN 5040
5045        V=PEEK(DI)
5050        IF V<8 THEN GOSUB 6000:REM CHANNELS
5060        IF V>7 AND V<#B THEN GOSUB 7000:REM VOL CHANNELS
5070        IF V=#B THEN GOSUB 8000:REM VOL NOISE
5080        IF V=#C THEN GOSUB 9000:REM DELAY
5090        IF V=#FF THEN GOSUB 10000:REM END
5095        IF V=#FF THEN IF GETC=0 THEN 5095:PRINT CHR$(12);:GOTO 21
5100        GOTO 5040

6000        REM CHANNELS
6010        PRINT TAB(10);"CHANNEL ";V/2;TAB(25);"FREQ :
C           ";(2000000)/(PEEK(DI+2)*256+PEEK(DI+1))
6020        DI=DI+3:RETURN

7000        REM VOLUMES CHANNELS
7010        PRINT TAB(10);"VOLUME ~ CHANNEL ";V-8;" :
C           ";TAB(30);PEEK(DI+1)
7020        DI=DI+2:RETURN

8000        REM VOL NOISE
8010        PRINT TAB(10);"VOLUME ~ NOISE  : ";PEEK(DI+1)
8020        DI=DI+2:RETURN

9000        REM DELAY
9010        PRINT TAB(10);"DELAY : ";PEEK(DI+1)*256+PEEK(DI+2)
```

dipswitch setting epson

MX80-DAI trough serial interface, dipswitch setting

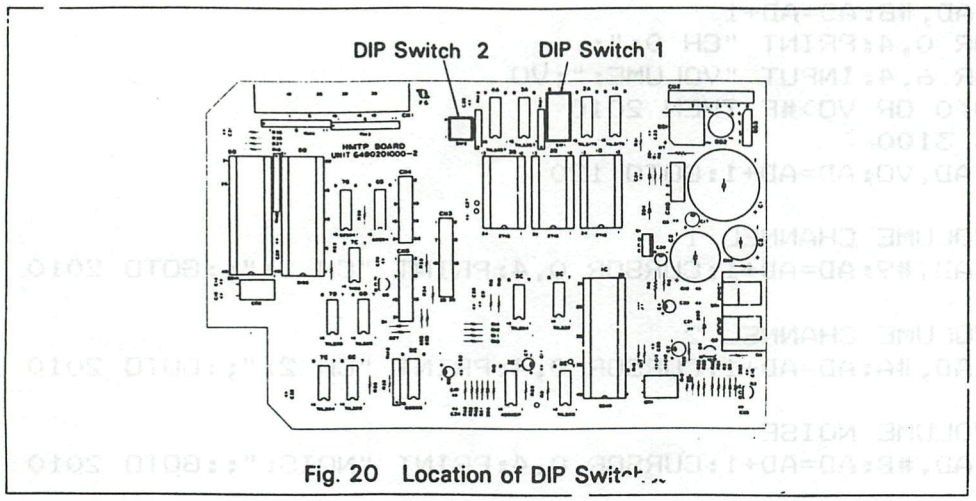


Fig. 20 Location of DIP Switch..

DIP switch 2 : 1 ON
2 ON
3 OFF
4 OFF

DIP switch 1 : 1 ON
2 ON
3 OFF
4 OFF
5 OFF
6 OFF
7 OFF
8 ON

DIP switch on serial card:

1 OFF
2 OFF
3 OFF
4 OFF
5 ON
6 OFF
7 OFF
8 ON

CABLE connections

DAI	MX-80
1	1
2	3
4	20
7	7

PAGE 3 -- TALK EDITOR Sip/Hermans -- FLST V2.0

```
9020        DI=DI+3:RETURN

10000       REM END
10010       PRINT TAB(10);"END ------------------------------------":
C           RETURN
```

Specifications

Print method	Serial impact dot matrix
Print Rate	80 CPS
Print Direction	Bidirectional
Number of Pins in Head	9
Matrix	9x9
Line Spacings	1/8", 1/6", 7/72", plus programmable
Throughput at 10 CPI	Logical seeking function– 105 LPM, 20 character line; 73 LPM, 40 character line; 46 LPM, 80 character line.

PRINTING CHARACTERISTICS

Character Set	Full 96-character ASCII with descenders
Graphics Characters	64 block characters
Printing Modes	Standard. Double (advance paper 1/206th and repeat line). Emphasized (shift right and double strike). Double Emphasized (combination of above).

PRINTING SIZES

	Characters per inch	Max. Characters per line
Normal	10	80
Normal Expanded	5	40
Compressed	16.5	132
Compressed Expanded	8.25	66

FORMS HANDLING

Line Feed	Programmable length 1 to 85/72nds
Form Feed	Programmable length to 66 lines
Horizontal Tab	To 112 positions
Vertical Tab	To 64 positions

MEDIA HANDLING

Paper Feed	Adjustable tractor-type pin feed
Paper Width Range	4" to 10"
Number of Parts	3
Paper Path	Rear

INTERFACES

Standard	Centronics-style 8-bit Parallel
Optional	RS232, IEEE488
Buffer Size	1 line

SWITCHES/LIGHTS/DETECTORS

Indicators	Power Light; Printer Ready; Paper Out; On Line
Switches	Power On/Off; On Line; Form Feed; Line Feed
Detectors	Internal buzzer (bell) responds to Paper Out and error conditions with a periodic 3-second tone for 30 seconds.

RELIABILITY

Print Head Life Expectancy	50 to 100 x 10^6 characters
MCBF (Excluding Print Head)	5 million lines

INKED RIBBON

Color	Black
Type	Cartridge
Life Expectancy	3 Million characters

ENVIRONMENTAL CONDITIONS

Operating Temperature Range	41°F to 95°F
Operating Humidity	10 to 80% non-condensing

POWER REQUIREMENT

Voltage	115V, 60Hz or 220-240V, 50 Hz
Current	<1 amp
Power Consumption	100 VA maximum

SELF TEST MODE

Depressing Line Feed Switch while turning power ON engages self-test which prints all characters in ROM.

PHYSICAL CHARACTERISTICS

Height	4.2"
Width	14.7"
Depth	12.0"
Weight	12 lbs.

Specifications subject to change without notice.

carpenters mystery

```
100   REM THE CARPENTERS MYSTERY w.hermans
110   CLEAR 1000:MODE 0:PRINT CHR$(12):COLORT 8 0 8 8
120   CURSOR 5,20:PRINT "T H E   C A R P E N T E R ' S     M Y S T E R Y"
130   WAIT TIME 20:CURSOR 4,18:D$="Move the red block from upper-left to upper-right"
140   FOR X=0 TO LEN(D$)-1:PRINT MID$(D$,X,1);:SOUND 1 0 15 0 FREQ(1000):WAIT TIME 3:SOUND OFF :
WAIT TIME 3:NEXT
150   POKE #B9A6,#CC:POKE #B814,#C8
160   POKE #75,32:CURSOR 5,10:INPUT "WHO PLAYS : COMPUTER Ä C ü OR HUMAN Ä H ü ";W$
170   IF W$<>"C" AND W$<>"H" THEN 160
180   CURSOR 5,10:PRINT SPC(50)
190   IF W$="C" THEN 500:CURSOR 2,10:PRINT "CURSOR-KEYS to move assistent, SHIFT-CURSOR to move
block"
200   IF GETC=0 THEN 200
210   GOTO 500
220   X1=A*20:Y1=B*20
230   NOISE 0 15
240   FILL X1+3,Y1+6 X1+6,Y1+9 CO
250   FILL X1+4,Y1+10 X1+5,Y1+12 CO
260   DRAW X1+3,Y1+11 X1+6,Y1+11 CO
270   DRAW X1+3,Y1+2 X1+3,Y1+5 CO
280   DRAW X1+6,Y1+2 X1+6,Y1+5 CO
290   DRAW X1+2,Y1+2 X1+3,Y1+2 CO
300   DRAW X1+6,Y1+2 X1+7,Y1+2 CO
310   SOUND OFF
320   RETURN
330   CO=21:GOSUB 220
340   SOUND 1 0 15 0 FREQ(50):SOUND 0 0 15 0 FREQ(5000):SOUND 0 0 15 2 FREQ(51)
350   DRAW X1+2,Y1+9 X1,Y1+11 21
360   DRAW X1+7,Y1+9 X1+9,Y1+11 21
370   WAIT TIME 10
380   CO=SCRN(A*20+7,B*20+7)
390   DRAW X1+2,Y1+9 X1,Y1+11 CO:DRAW X1+7,Y1+9 X1+9,Y1+11 CO
400   SOUND OFF
410   RETURN
500   REM INITIATE
510   MODE 4:COLORG 0 14 3 9
520   FILL 15,15 124,104 21:FILL 16,16 123,103 20
530   FILL 21,61 59,100 22
540   FOR X=1 TO 60:DRAW 60,80 110,80 23:DRAW 110,80 105,75 23:DRAW 110,80 105,85 23
550   DRAW 59+X,61 59+X,100 3:DRAW 20+X,61 20+X,100 0:NEXT
560   WAIT TIME 100:FILL 16,16 123,103 20
570   DIM F(7,6)
580   FOR X=1 TO 7:FOR Y=1 TO 6:F(X,Y)=99:NEXT:NEXT
590   F(1,1)=2:F(2,1)=2:F(4,1)=2:F(5,1)=2
600   F(1,3)=4:F(3,3)=1:F(3,4)=1:F(4,3)=3:F(4,4)=3
610   F(3,1)=0:F(3,2)=0
620   FOR Y=1 TO 4:FOR X=1 TO 5
630   XP=X*20:YP=Y*20
640   IF F(X,Y)=0 OR F(X,Y)>5 THEN 670:ON F(X,Y) GOSUB 3000,3010,3020,3030
650   CO=23:IF F(X,Y)=4 THEN CO=3
660   FILL XP,YP XP+XO,YP+YO CO
670   NEXT:NEXT
1000  IF W$="C" THEN 2000:REM COMPUTER PLAYS
1010  REM MOVE
1020  IF F(4,3)=4 THEN 5000:REM END
1030  IF FLAG=0 THEN FLAG=1:A=3:B=3:GOTO 1050:REM FIRST DOT
1040  G=GETC:IF G=0 THEN 1040
1050  CO=SCRN(A*20+4,B*20+4):GOSUB 220
1060  GOSUB 3040
1070  IF A>5 THEN GOSUB 330:A=5
1080  IF A<1 THEN GOSUB 330:A=1
1090  IF B>4 THEN GOSUB 330:B=4
1100  IF B<1 THEN GOSUB 330:B=1
1110  CO=21:GOSUB 220
1120  GOTO 1010
```

```
2000  A=3:B=3
2010  COMMAND$="515005152271140413603625O5/033626314114342273722737205005"
2020  COMMAND$=COMMAND$+"33626311434142700515036/140411404127005150362 4/22737205336263362614227372053636163114041272270051500515"
2030  COMMAND$=COMMAND$+"271140436250535036263141142737227370525005336263114143636005150227"
2040  FOR DO=0 TO LEN(COMMAND$)-1
2050  IF MID$(COMMAND$,DO,1)="/" THEN NEXT:REM CODE
2060  G=VAL(MID$(COMMAND$,DO,1))+16
2070  CO=SCRN(A*20+4,B*20+4):GOSUB 220
2080  GOSUB 3040
2090  CO=21:GOSUB 220
2100  WAIT TIME 5:NEXT
2110  GOTO 5000:REM END
3000  XO=18:YO=18:RETURN
3010  XO=18:YO=38:RETURN
3020  XO=38:YO=18:RETURN
3030  XO=38:YO=38:RETURN
3040  REM MOVE CURSOR OR MOVE BLOCK
3050  IF G<16 OR G>23 THEN GOSUB 330:RETURN:REM FALSE KEY
3060  IF G<20 THEN ON G-15 GOTO 3080,3090,3100,3110:GOTO 3120
3070  GOTO 3120:REM MOVE BLOCK
3080  B=B+1:RETURN
3090  B=B-1:RETURN
3100  A=A-1:RETURN
3110  A=A+1:RETURN
3120  SIZE=F(A,B)
3130  IF SIZE=0 OR SIZE>4 THEN GOSUB 330:RETURN:REM NOT VALID
3140  CO=23:IF SIZE=4 THEN CO=3
3150  ON G-19 GOSUB 3180,3380,3550,3720
3160  SOUND 0 0 15 0 FREQ(1000):WAIT TIME 10:SOUND OFF
3170  RETURN
3180  REM UP
3190  IF B=4 THEN GOSUB 330:RETURN:REM NOT POSSIBLE
3200  IF SIZE=2 OR SIZE=4 THEN IF B>2 THEN GOSUB 330:RETURN
3210  ON SIZE GOTO 3220,3240,3260,3280
3220  IF F(A,B+1)<>0 THEN GOSUB 330:RETURN
3230  GOTO 3290
3240  IF B>2 OR F(A,B+2)<>0 THEN GOSUB 330:RETURN
3250  GOTO 3290
3260  IF F(A,B+1)<>0 OR F(A+1,B+1)<>0 THEN GOSUB 330:RETURN
3270  GOTO 3290
3280  IF B>2 OR F(A,B+2)<>0 OR F(A+1,B+2)<>0 THEN GOSUB 330:RETURN
3290  GOSUB 3930
3300  FOR Y=0 TO 20:GOSUB 4020
3310  DRAW A*20,B*20+Y-1 A*20+XO,B*20+Y-1 20
3320  DRAW A*20,B*20+YO+Y A*20+XO,B*20+YO+Y CO
3330  NEXT:SOUND OFF
3340  F(A,B+1.0)=F(A,B):ON SIZE GOTO 3350,3350,3360,3360
3350  F(A,B)=0:GOTO 3370
3360  F(A,B)=0:F(A+1.0,B)=0
3370  GOSUB 3950:RETURN
3380  REM DOWN
3390  IF B=1 THEN GOSUB 330:RETURN:REM N.P.
3400  ON SIZE GOTO 3410,3410,3430,3430
3410  IF F(A,B-1.0)<>0.0 THEN GOSUB 330:RETURN
3420  GOTO 3440
3430  IF F(A,B-1.0)<>0.0 OR F(A+1.0,B-1.0)<>0.0 THEN GOSUB 330:RETURN
3440  GOSUB 3930
3450  FOR Y=0 TO 20:GOSUB 4020
3460  DRAW A*20,B*20+YO-Y+1 A*20+XO,B*20+YO-Y+1 20
3470  DRAW A*20,B*20-Y A*20+XO,B*20-Y CO
3480  NEXT:SOUND OFF
```

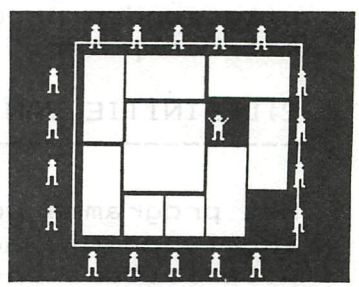
```
3490  F(A,B-1.0)=F(A,B):ON SIZE GOTO 3500,3510,3520,3530
3500  F(A,B)=0:GOTO 3540
3510  F(A,B+1.0)=0:GOTO 3540
3520  F(A,B)=0:F(A+1.0,B)=0:GOTO 3540
3530  F(A,B+1.0)=0:F(A+1.0,B+1.0)=0
3540  GOSUB 3950:RETURN
3550  REM LEFT
3560  IF A=1 THEN GOSUB 330:RETURN
3570  ON SIZE GOTO 3580,3600,3580,3600
3580  IF F(A-1.0,B)<>0.0 THEN GOSUB 330:RETURN
3590  GOTO 3610
3600  IF F(A-1.0,B)<>0.0 OR F(A-1.0,B+1.0)<>0.0 THEN GOSUB 330:RETURN
3610  GOSUB 3930
3620  FOR Y=0 TO 20:GOSUB 4020
3630  DRAW A*20-Y+XO+1,B*20 A*20-Y+XO+1,B*20+YO 20
3640  DRAW A*20-Y,B*20 A*20-Y,B*20+YO CO
3650  NEXT:SOUND OFF
3660  F(A-1.0,B)=F(A,B):ON SIZE GOTO 3670,3680,3690,3700
3670  F(A,B)=0:GOTO 3710
3680  F(A,B)=0:F(A,B+1.0)=0:GOTO 3710
3690  F(A+1.0,B)=0:GOTO 3710
3700  F(A+1.0,B)=0:F(A+1.0,B+1.0)=0
3710  GOSUB 3950:RETURN
3720  REM RIGHT
3730  IF A=5 THEN GOSUB 330:RETURN
3740  ON SIZE GOTO 3750,3770,3810,3790
3750  IF F(A+1.0,B)<>0.0 THEN GOSUB 330:RETURN
3760  GOTO 3820
3770  IF F(A+1.0,B)<>0.0 OR F(A+1.0,B+1.0)<>0.0 THEN GOSUB 330:RETURN
3780  GOTO 3820
3790  IF A>4.0 OR F(A+2.0,B)<>0.0 OR F(A+2.0,B+1.0)<>0.0 THEN GOSUB 330:RETURN
3800  GOTO 3820
3810  IF A>4.0 OR F(A+2.0,B)<>0.0 THEN GOSUB 330:RETURN
3820  GOSUB 3930
3830  FOR Y=0 TO 20:GOSUB 4020
3840  DRAW A*20+Y+XO,B*20 A*20+Y+XO,B*20+YO CO
3850  DRAW A*20+Y-1,B*20 A*20+Y-1,B*20+YO 20
3860  NEXT
3870  F(A+1,B)=F(A,B):ON SIZE GOTO 3880,3890,3900,3910
3880  F(A,B)=0:GOTO 3920
3890  F(A,B)=0:F(A,B+1)=0:GOTO 3920
3900  F(A,B)=0:GOTO 3920
3910  F(A,B)=0:F(A,B+1)=0
3920  GOSUB 3950:RETURN
3930  ON F(A,B) GOSUB 3000,3010,3020,3030
3940  RETURN
3950  FOR Y=1 TO 4
3960  FOR X=1 TO 5
3970  IF F(X,Y)=2 THEN F(X,Y+1)=99
3980  IF F(X,Y)=3 THEN F(X+1,Y)=99
3990  IF F(X,Y)=4 THEN F(X+1,Y)=99:F(X,Y+1)=99:F(X+1,Y+1)=99
4000  NEXT:NEXT
4010  RETURN
4020  SOUND 0 0 15 0 FREQ(300+Y*10-SIZE*50):RETURN
```

```
5000  FOR Y=1 TO 10
5010  COLORG 0 0 3 0:SOUND 1 0 15 2 FREQ(1000):WAIT TIME 10
5020  COLORG 0 5 3 9:SOUND 1 0 15 2 FREQ(600):WAIT TIME 10
5030  NEXT:SOUND OFF
5040  GOTO 100:REM AGAIN
```

playing the game,solution...

The object of the game is to move the RED block from upper-left to upper-right. You can move your little assistant with the CURSOR-keys. You can move a block if the assistant is in the lower-left corner of the block.(this could be modified but will make the program even longer !) If your assistant is in position, you move the block with SHIFT-CURSOR.The SOLUTION is entered on lines 2010-2030, encoded in COMMAND$,following this table: 0 = assistant UP 1 = ass down 2 = ass left 3 = ass right 4 = block up 5 = block down 6 = block left 7 = block right. We needed 223 characters (moves) to solve the mystery ! Please send your solution if you can make it faster !

```
3299  REM CORRECTIONS AS SUGGESTED BY K..VERBEKE
3300  FOR Y=0 TO 19:GOSUB 4020
3320  DRAW A*20,B*20+YO+Y+1 A*20+XO,B*20+YO+Y+1 CO
3450  FOR Y=0 TO 19:GOSUB 4020
3470  DRAW A*20,B*20-Y-1 A*20+XO,B*20-Y-1 CO
3620  FOR Y=1 TO 20:GOSUB 4020
3630  DRAW A*20+Y+1+XO,B*20 A*20-Y+XO+1,B*20+Y CO
3830  FOR Y=1 TO 20:GOSUB 4020
3840  DRAW A*20+Y-1,B*20 A*20+Y-1,B*20+YO CO
```



```
T
3299  REM CORRECTIONS AS SUGGESTED BY M.STREICHER
3310  DRAW A*20,B*20+Y-1 A*20+XO,B*20+Y-1 20
3460  DRAW A*20,B*20+YO-Y+1 A*20+XO,B*20+YO-Y+1 20
3630  DRAW A*20-Y+XO+1,B*20 A*20-Y+XO+1,B*20+YO 20
3850  DRAW A*20+Y-1,B*20 A*20+Y-1,B*20+YO 20
```

N O S B A S I C O D E

1) INLEIDING

Sinds enige tijd zendt de NOS in het programma HOBBYSCOOP elke week een BASIC programma uit. De informatie wordt uitgezonden in de vorm van het normale cassette signaal, dwz zeggen inhoud, seriele standaard en baudrate zoals gebruikt voor SAVE en LOAD. Vermits deze kenmerken specifiek zijn voor een bepaald type computer, was elke NOS uitzending slechts bruikbaar voor een beperkt aantal hobbyisten. Daarom ook werd elke week uitgezonden voor een ander type computer.
Om dit probleem te vermijden werd gezocht naar een signaal-formaat dat eender welke microcomputer kan generen en inlezen.
Nu is het zo dat de meeste microcomputers een programmma saven in het formaat waarin een basic programma intern opgeslagen wordt. Dit formaat is eigen aan elk type computer en het is dan ook niet geschikt als universele codering.
Daarom wordt in de NOS BASICODE het programma in de vorm van een "LISTING" overgedragen. Dus een gewone LIST wordt uitgevoerd, maar ipv het programma te printen op het scherm, wordt elk character door een speciale routine omgezet in een seriele signaalvorm.
Het speciale aan dit signaal is dat we vooraf alle kenmerken afspreken waaraan het moet voldoen. Dus dit signaal blijft identiek, eender met welk type computer het opgewekt is.
Tevens zorgen we ervoor dat dit signaal opneembaar is op een gewone cassetterecorder = uit te zenden langs de radio.
Op deze wijze hebben we een NOS BASICODE signaal verkregen.
Nu maken we voor elke microcomputer ook nog een routine die het omgekeerde doet, dwz BASICODE terug omzetzetten in het specifieke interne basic formaat. Indien we het ontvangen BASICODE signaal inputten in deze routine dan kunnen alle uitgezonden programma's ingelezen worden door elk type microcomputer.

2) DEFINITIE VAN DE NOS BASICODE

Het programma wordt overgedragen in de vorm waarin het werd ingetyped of waarin het door LIST wordt getoond.
Alle letters en cijfers worden in ASCII voorgesteld met het 8ste bit = 1. Spaties in het programma mogen worden onderdrukt behalve in strings en REM's. Elke regel afsluiten met 'CR'.
Het programma wordt voorafgegaan door het ASCII teken 'start of text' (STX = #02) en afgesloten door een 'end of text' (ETX = #03).
Als laatste volgt een checksum die het resultaat is van de bitsgewijze exclusive OR van alle voorgaande bytes.

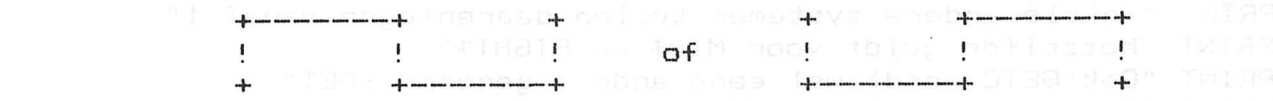
Serieele code :
    baudrate : 1200
    opbouw   : 1 startbit logisch 0
               7 ASCII bits (LSB eerst)
               1 MSB logisch 1
               2 stopbits logisch 1

Voorbeeld :

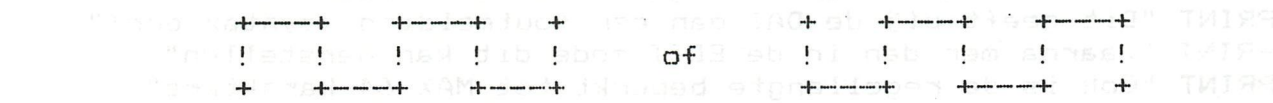
ASCII 'X' = #58 of binair voorgesteld : 0 1 0 1 1 0 0 0
of met least significant bits eerst : 0 0 0 1 1 0 1 0

Toonmodulatie :

een 0 bit = 1 volle periode van 1200 Hz (= 1/1.2 ms)



een 1 bit = 2 volle perioden van 2400 Hz (= 1/1.2 ms)



Begin en einde tape signaal :
leader : 3 seconden 2400 Hz (stopbit)
trailer : 3 seconden 2400 Hz (stopbit)

3)NOS BASICODE EN DAI

Voor de DAI microcomputer kan het genereren en terug omzetten van de BASICODE geheel software gebeuren. Dwz de bestaande input/output voor cassette wordt gebruikt.
Dit is in tegenstelling met de andere microcomputers,die naast de software routines meestal ook nog extra hardware nodig hebben.
Als de NOS een programma uitzendt, dan verbindt u de cassette-recorder met de tuner/versterker en maakt een opname van het programma en bijbehorende uitleg (best zal u deze opname iets overmoduleren). Daarna wordt de cassetterecorder op de normale wijze verbonden met de DAI.
Vooreerst laadt u het machinetaalprogramma Basicode read/write. Dit programma is reeds uitgezonden door de NOS (in DAI formaat). Indien u dit gemist heeft, dan kan u het altijd intypen vanaf de bijgevoegde hex dump.
Als de pointers niet mee gesaved zijn, vergeet dan niet de heappointer aan te passen :
*UT >S29B -00 -06 >B *CLEAR xxxx of *NEW
U kunt nu verder handelen zoals beschreven in bijgevoegd basic uitleg programma.
Opmerking : in tegenstelling met de gewone DAI LOAD wordt door de basicode leesroutine geen NEW gedaan. Dit laat het mergen van basicprogramma's toe. Indien dit niet gewenst is geef dan een NEW voor het laden.

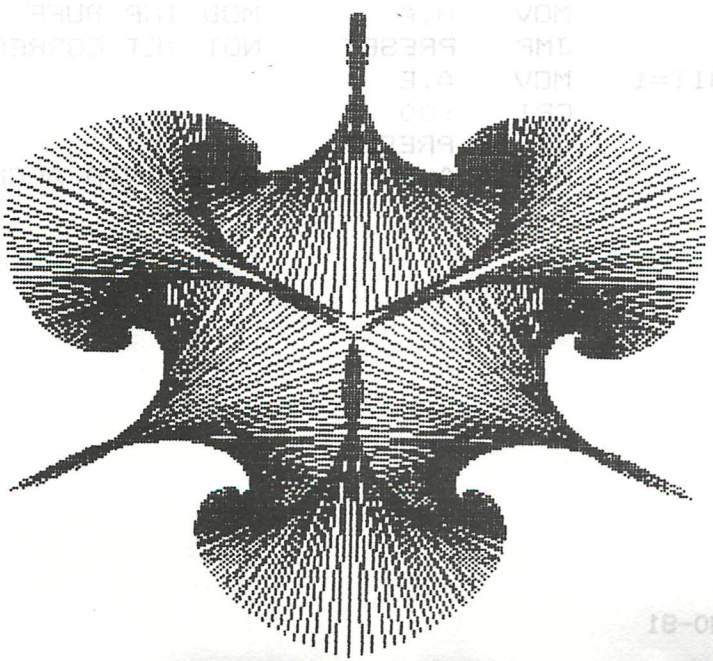
4) BESLUIT

Ter dokumentatie volgen achter dit artikel hex dump en sourcecode van het machine taal programma en een LIST van het uitlegprogramma.
Dit artikel werd gebaseerd op de ingezonden tekst van dhr Th v Lieshout, Postgalei 5, 1687 VP WOGNUM, tel 02297-2648.
Hij maakte deel uit van de werkgroep die de NOS BASICODE ontworpen heeft en hij is ook de originator van de mlp vertaal-programma voor de DAI.
Leden die de NOS BASICODE gebruiken mogen altijd hun ervaringen opsturen naar de redaktie (bv in de vorm van een kort artikel voor de newsletter).

```
1     PRINT CHR$(12)
10    PRINT "Dit machinetaal programma maakt het mogelijk BASIC "
20    PRINT "programma's uit te wisselen met andere uP's."
30    PRINT "In een werkgroep is er een code afgesproken waaraan"
40    PRINT "alle bekende systemen zullen voldoen."
50    PRINT "Voorlopig zijn er een aantal beperkingen betreffende"
60    PRINT "de te gebruiken statements."
70    PRINT "Zo zijn de volgende opdrachten uit den boze :"
80    PRINT "POKE,PEEK,CURX,CURY,CURSOR,SOUND,ENVELOPE,NOISE,"
90    PRINT "FRE,VARPTR,DOT,DRAW,FILL,MODE,IN,OUT,CALLM."
100   PRINT "Er zijn ook een aantal statements die niet direct"
110   PRINT "uitwisselbaar zijn doch na een kleine verandering wel:"
120   PRINT "bv LEFT$(A$,X) waar X bij de DAI telt vanaf 0"
130   PRINT "enkele andere systemen tellen daarentegen vanaf 1"
140   PRINT "hetzelfde geldt voor MID$ en RIGHT$"
150   PRINT "Ook GETC wordt wel eens anders genoemd :GET"
160   PRINT "Als U ergens leest HOME kunt U dit vervangen door CHR$(12)"
170   PRINT "Bij enkele systemen is het bij een PRINT opdracht"
180   PRINT "niet nodig met aanhalingstekens te sluiten :"
190   PRINT "Dit geeft bij de DAI dan een foutmelding (syntax err)"
200   PRINT "waarna men dan in de EDIT mode dit kan herstellen"
201   PRINT "Ook is de regellengte beperkt tot MAX 64 karakters"
205   PRINT
210   PRINT "DRUK DE SPATIEBALK IN VOOR VERDER LEZEN"
220   A=GETC:IF A<>32.0 GOTO 220
230   PRINT CHR$(12)
240   PRINT " Nu dan enkele aanwijzingen voor het "
250   PRINT "NOS BASICODE machinetaal programma"
260   PRINT
270   PRINT "Dit programma is in te lezen in machinetaal :"
280   PRINT "kies UT(ility) ,toets R(ead) en return,start tape"
290   PRINT "indien dit niet automatisch gebeurt."
300   PRINT "Als het programma ingelezen is dan staat dit van"
310   PRINT "#29B tot #527 :de pointers worden dus meegeladen."
311   PRINT "Het is nuttig na het inlezen NEW in te tikken"
315   PRINT
320   PRINT "DE NOS BASICODE IS IN TE LEZEN DMV CALLM 750"
330   PRINT
340   PRINT "DE NOS BASICODE IS WEG TE SCHRIJVEN DMV:"
350   PRINT " CALLM 1000:LIST:CALLM 1250"
355   PRINT
360   PRINT "het verdient aanbeveling de cassette voor de leader"
370   PRINT "(=aanloopstrook )te starten"
380   PRINT "het programma wordt op checksom gecontroleerd :"
390   PRINT "bij een fout verschijnt er een F op de plaats van de cursor"
400   PRINT "dit is te verhelpen door POKE #75,#5F"
410   PRINT "Een listing verschijnt in het Oktobernummer(1981)"
420   PRINT "van het blad DATABUS (uitgeverij KLUWER)"
430   PRINT
435   PRINT " DRUK OP DE SPATIEBALK VOOR VERDER LEZEN"
440   A=GETC:IF A<>32.0 GOTO 440
445   PRINT CHR$(12)
450   PRINT "Voor de fijnproevers zijn er nog enkele features"
460   PRINT "tw: Bij het wegschrijven is het volgende mogelijk"
470   PRINT "bv CALLM 1000:LIST 68-134:LIST 567-785:CALLM1250"
480   PRINT "Ook is het mogelijk slechts een deel in te lezen"
490   PRINT "U kunt het inlezen dan onderbreken door BREAK in te"
495   PRINT "drukken en daarna CALLM 904"
500   PRINT "in dat geval wordt de checksom niet gecontroleerd"
510   PRINT "Programma: Th v Lieshout te Wognum "
```

HEX-DUMP OF BOOTSTRAP BASICODE

```
029B 27 05 00 01 27
02A0 06 28 06 29 06 50 B3 C5 E8 00 00 00 00 00 00 00
02B0 00 00 00 00 00 00 00 00 00 00 35 32 37 0D BA 02
02C0 BA 02 01 00 00 C3 B8 D2 C3 F1 D2 C3 27 D4 C3 25
02D0 D3 C3 40 D3 C3 45 D4 C3 A2 D3 C9 00 00 C9 00 00
02E0 C3 B4 DD C9 00 00 24 24 24 3C 24 18 75 07 E5 D5
02F0 C5 21 40 00 36 28 21 75 00 36 FF 2A A3 02 24 22
0300 EC 02 22 C6 03 0E 00 1E 00 FB 06 FF 16 01 F3 26
0310 FF 2E 03 2C FA 09 03 3A 00 FD E6 80 BA CA 13 03
0320 57 7D FE 08 DA 2A 03 7C 17 67 7C 07 F6 F1 FE FF
0330 CA 3C 03 FE F5 CA 48 03 67 C3 11 03 7B FE 00 CA
0340 11 03 78 F6 80 C3 4E 03 29 29 29 78 E6 7F 1F 47
0350 21 04 F0 1C DA 13 03 FB 7B E6 80 1E 00 C2 79 03
0360 79 A8 4F 78 FE 03 CA 74 03 2A EC 02 77 23 22 EC
0370 02 C3 09 03 1E 80 C3 09 03 79 E6 7F B8 21 75 00
0380 CA 88 03 36 46 C3 8A 03 36 5F 21 40 00 36 30 21
0390 9D 03 22 E1 02 21 96 02 36 01 C3 C2 03 E5 D5 C5
03A0 2A EC 02 EB 2A C6 03 23 22 C6 03 7A BC 7E C2 C2
03B0 03 7B BD 7E C2 C2 03 21 96 02 36 00 21 B4 DD 22
03C0 E1 02 C1 D1 E1 C9 76 07 E5 D5 C5 21 31 01 36 03
03D0 2A A3 02 24 22 EC 02 22 C6 03 3E C3 32 DD 02 21
03E0 F5 03 22 DE 02 C3 EB 03 C3 C8 03 21 E0 04 3E 82
03F0 36 83 C3 FA 03 E5 D5 C5 F6 80 47 21 E0 04 AE 77
0400 2A EC 02 70 23 22 EC 02 78 C3 C2 03 E5 D5 C5 21
0410 31 01 36 00 21 E0 04 46 2A EC 02 36 83 23 70 22
0420 EC 02 21 40 00 36 28 11 00 00 21 06 FD F3 CD 03
0430 05 CD AD 04 7A FE 65 C2 2E 04 CD FF 04 2A C6 03
0440 46 23 22 C6 03 21 06 FD 11 00 00 CD 20 05 CD C3
0450 04 0E 01 CD FB 04 78 0F 47 DC E5 04 D4 D5 04 0C
0460 79 FE 09 FA 56 04 CD 06 05 CD AD 04 CD 0B 05 CD
0470 AD 04 2A EC 02 EB 2A C6 03 13 7A BC C2 ED 04 7B
0480 BD C2 3D 04 21 06 FD 11 00 00 CD FB 04 CD AD 04
0490 7A FE 39 CD 03 05 FA 8D 04 FB 21 40 00 36 30 3E
04A0 C9 32 DD 02 21 00 00 22 DE 02 C3 C2 03 CD CD 04
04B0 CD 12 05 CD CD 04 CD 12 05 CD CD 04 CD 12 05 CD
04C0 CD 04 C9 CD CD 04 CD 16 05 CD CD 04 C9 13 7B E6
04D0 01 F6 20 77 C9 CD 20 05 CD C3 04 00 00 00 00 C9
04E0 FC CD C3 0C 04 CD FF 04 CD AD 04 37 C9 00 7F C3
04F0 3D 04 00 00 00 00 00 00 00 00 C9 CD F6 04 C9 CD
0500 F2 04 C9 CD F9 04 00 CD FF 04 C9 CD FA 04 CD 06
0510 05 C9 CD 0B 05 C9 CD 0B 05 CD 12 05 CD F6 04 C9
0520 CD 19 05 CD F7 04 C9 80
```



```
002                    *       TITL  NOS BASICODE RD  TH V LIESHOUT,WOGNUM
003                PTRIN       EQU   :2EC
004                            ORG   :2EE
005 02EE E5        COLDST      PUSH  H
006 02EF D5                    PUSH  D
007 02F0 C5                    PUSH  B
008 02F1 214000                LXI   H,:0040
009 02F4 3628                  MVI   M,:28     CASS MOTOR 1 ON
010 02F6 21753C                LXI   H,:75
011 02F9 36FF                  MVI   M,:FF     CURSOR BLACK
012 02FB 2AA302                LHLD  :2A3      END SYMB TABLE
013 02FE 24                    INR   H         RESERVE 256BYTS
014 02FF 22EC02                SHLD  PTRIN
015 0302 22C603                SHLD  PTROUT
016 0305 0E00                  MVI   C,:00     CHECKSUM
017 0307 1E00                  MVI   E,:00
018 0309 FB        START       EI              C=CHECKSUM
019 030A 06FF                  MVI   B,:FF     B=BYTE
020 030C 1601                  MVI   D,:01     1ST RND NOT EQU
021 030E F3                    DI              H=SHIFTREG
022 030F 26FF                  MVI   H,:FF     L=UNITS OF 49uS
023 0311 2E03      PRESET      MVI   L,:03     E=BITCOUNTER
024 0313 2C        TCOUNT      INR   L         INCR TIMECNTR
025 0314 FA0903                JM    START     D=MEM CASS INP
026 0317 3A00FD                LDA   :FD00     CASS INTERFACE
027 031A E680                  ANI   :80       MASK
028 031C BA                    CMP   D         EQU?
029 031D CA1303                JZ    TCOUNT    N CHANGE NXT RND
030 0320 57                    MOV   D,A       MODIFY D
031 0321 7D                    MOV   A,L       TIMECNTR IN ACCU
032 0322 FE08                  CPI   :08       t>312 uS?
033 0324 DA2A03                JC    SHORT     t<312uS
034 0327 7C        LONG        MOV   A,H       INPBUFF IN ACCU
035 0328 17                    RAL             SHIFT,SET 0
036 0329 67                    MOV   H,A
037 032A 7C        SHORT       MOV   A,H
038 032B 07                    RLC             SHIFT
039 032C F6F1                  ORI   :F1       SET 1
040 032E FEFF                  CPI   :FF       COMP FF=2400HZ
041 0330 CA3C03                JZ    BIT=1     GOTO BIT=1
042 0333 FEF5                  CPI   :F5       COMP F5=1200HZ
043 0335 CA4803                JZ    BIT=0     GOTO BIT=0
044 0338 67                    MOV   H,A       MOD INP BUFF
045 0339 C31103                JMP   PRESET    NOT YET CORRECT
046 033C 7B        BIT=1       MOV   A,E
047 033D FE00                  CPI   :00
048 033F CA1103                JZ    PRESET
049 0342 78                    MOV   A,B       BYTE INTO ACCU
```

```
050 0343 F680          ORI   :80       SET"1"IN BYTE
051 0345 C34E03        JMP   BYTE
052 0348 29      BIT=0 DAD   H          **IRRELEVANT**
053 0349 29            DAD   H          *INSTRUCTIONS*
054 034A 29            DAD   H          ***DELAY!!****
055 034B 78            MOV   A,B        BYTE INTO ACCU
056 034C E67F          ANI   :7F        SET"0"IN BYTE
057 034E 1F      BYTE  RAR
058 034F 47            MOV   B,A        MODIFY BYTE
059 0350 2104F0        LXI   H,:F004    CLEAR SHIFTREG.
060 0353 1C            INR   E          INCR BITCNTR
061 0354 DA1303        JC    TCOUNT     N STARTBIT ON CY
062 0357 FB      CHKSUM EI
063 0358 7B            MOV   A,E        BITCNTR IN ACCU
064 0359 E680          ANI   :80        CLR EXPT ENDFLG
065 035B 1E00          MVI   E,:00      CLR BITCOUNTER
066 035D C27903        JNZ   STOTEX     ENDFLAG SET
067 0360 79            MOV   A,C        CHECKSUM IN ACCU
068 0361 A8            XRA   B          EXOR W B
069 0362 4F            MOV   C,A        STORE IN C
070 0363 78            MOV   A,B        BYTE IN ACCU
071 0364 FE03          CPI   :03        END OF TEXT?
072 0366 CA7403        JZ    ENDFLG     JUMP ENDFLAG
073 0369 2AEC02  RETURN LHLD  PTRIN      INPOINTER IN HL
074 036C 77            MOV   M,A        STORE A IN BUFF
075 036D 23            INX   H          INCR INPOINTER
076 036E 22EC02        SHLD  PTRIN      STORE INPOINTER
077 0371 C30903        JMP   START      NEXT
078 0374 1E80    ENDFLG MVI   E,:80      SET ENDFLAG
079 0376 C30903        JMP   START      NEXT
080 0379 79      STOTEX MOV   A,C        CHECKSUM IN ACCU
081 037A E67F          ANI   :7F        MASK 8th BIT
082 037C B8            CMP   B          COMPARE W BYTE
083 037D 217500        LXI   H,:75      :75=ADDR CURSOR
084 0380 CA8803        JZ    GOOD
085 0383 3646    FALSE MVI   M,:46      "F"IN CURSOR
086 0385 C38A03        JMP   LAST
087 0388 365F    GOOD  MVI   M,:5F      "-"IN CURSOR
088 038A 214000  LAST  LXI   H,:0040    CASS MOTOR 1 OFF
089 038D 3630          MVI   M,:30
090 038F 219D03        LXI   H,NEXT     DINC(INCOM DATA)
091 0392 22E102        SHLD  :2E1       INP. FROM NEXT
092 0395 219602        LXI   H,:296     INSW IN H&L
093 0398 3601          MVI   M,:01      TURN INSW ON
094 039A C3C203        JMP   RET
095 039D E5      NEXT  PUSH  H
096 039E D5            PUSH  D
097 039F C5            PUSH  B
098 03A0 2AEC02        LHLD  PTRIN      INPOINTER IN H&L
099 03A3 EB            XCHG             EXCHANGE W D&E
100 03A4 2AC603        LHLD  PTROUT     OUTPOINTER IN HL
101 03A7 23            INX   H          INCR OUTPOINTER
102 03A8 22C603        SHLD  PTROUT     MODIFY OUTPOINTR
103 03AB 7A            MOV   A,D
104 03AC BC            CMP   H          COMP H w D (MSB)
105 03AD 7E            MOV   A,M        M INTO ACCU
106 03AE C2C203        JNZ   RET        NOT EQU
```

```
107 03B1 7B              MOV  A,E
108 03B2 BD              CMP  L          COMP L w E (LSB)
109 03B3 7E              MOV  A,M        M INTO ACCU
110 03B4 C2C203          JNZ  RET        NOT EQU
111 03B7 219602          LXI  H,:296     INSW INTO H&L
112 03BA 3600            MVI  M,:00      SET INSW ON KEYB
113 03BC 21B4DD          LXI  H,:DDB4    SUBROUT. KEYB.
114 03BF 22E102          SHLD :2E1
115 03C2 C1        RET   POP  B
116 03C3 D1              POP  D
117 03C4 E1              POP  H
118 03C5 C9              RET
119                PTROUT EQU  :3C6
120                *      END  ****************************************
121                *      TITL NOS BASICODE WR  TH V LIESHOUT,WOGNUM
122                *      ORG  :3C2
123                *RET   POP  B          **ALSO PART OF**
124                *      POP  D          *READOUTROUTINE*
125                *      POP  H
126                *      RET
127                *PTRIN EQU  :2EC       *MEMORY
128                *PTROUT EQU :3C6       *MEMORY
129                       ORG  :3C8
130 03C8 E5        CSTART PUSH H          COLD START
131 03C9 D5              PUSH D
132 03CA C5              PUSH B
133 03CB 213101          LXI  H,:131     OUTP  SWITCH
134 03CE 3603            MVI  M,:03
135 03D0 2AA302          LHLD :2A3       END SYMB TABLE
136 03D3 24              INR  H          RES.256 BYTES
137 03D4 22EC02          SHLD PTRIN      SET POINTERS
138 03D7 22C603          SHLD PTROUT
139 03DA 3EC3            MVI  A,:C3      *POINTER
140 03DC 32DD02          STA  :2DD       *
141 03DF 21F503          LXI  H,START1   *
142 03E2 22DE02          SHLD :2DE       *
143 03E5 C3EB03          JMP  JUMP1
144 03E8 C3C803          JMP  CSTART     HEX 3E8=1000 DEC
145 03EB 21E004    JUMP1 LXI  H,CHECKS
146 03EE 3E82            MVI  A,:82      START OF TEXT
147 03F0 3683            MVI  M,:83      CHKSUM INCL EOT
148 03F2 C3FA03          JMP  START2     1ST RND NOT PUSH
149 03F5 E5        START1 PUSH H
150 03F6 D5              PUSH D
151 03F7 C5              PUSH B
152 03F8 F680            ORI  :80
153 03FA 47        START2 MOV  B,A        SAVE CHR IN B
154 03FB 21E004          LXI  H,CHECKS
155 03FE AE              XRA  M          NEW CHECKSUM
156 03FF 77              MOV  M,A        MODIFY CHKSUM
157 0400 2AEC02          LHLD PTRIN      SAVE CHR
158 0403 70              MOV  M,B
159 0404 23              INX  H          INCR POINTER
160 0405 22EC02          SHLD PTRIN
161 0408 78              MOV  A,B
162 0409 C3C203          JMP  RET        RETURN
163 040C E5        TBLEND PUSH H
```

```
164 040D D5                 PUSH  D
165 040E C5                 PUSH  B
166 040F 213101             LXI   H,:131    OUTP SWITCH
167 0412 3600               MVI   M,:00     BACK
168 0414 21E004             LXI   H,CHECKS  SAVE CHECKSUM
169 0417 46                 MOV   B,M
170 0418 2AEC02             LHLD  PTRIN
171 041B 3683               MVI   M,:83     END OF TEXT
172 041D 23                 INX   H
173 041E 70                 MOV   M,B       LAST CHR=CHKSUM
174 041F 22EC02             SHLD  PTRIN
175 0422 214000             LXI   H,:40     CASSMOTOR 1 ON
176 0425 3628               MVI   M,:28
177 0427 110000             LXI   D,:0000
178 042A 2106FD             LXI   H,:FD06   BIT 0=CASSETTE-
179 042D F3                 DI              ******OUTPUT***
180 042E CD0305     LEADER  CALL  DEL121
181 0431 CDAD04             CALL  BITONE
182 0434 7A                 MOV   A,D
183 0435 FE39               CPI   :39       TIME= 3 SEC
184 0437 C22E04             JNZ   LEADER
185 043A CDFF04             CALL  DEL059
186 043D 2AC603     DATA    LHLD  PTROUT    PULL CHR FROM
187 0440 46                 MOV   B,M       TABLE
188 0441 23                 INX   H
189 0442 22C603             SHLD  PTROUT    NEW POINTER
190 0445 2106FD             LXI   H,:FD06   CASS OUTPUT
191 0448 110000             LXI   D,:0000
192 044B CD2005             CALL  DEL312
193 044E CDC304             CALL  BITZER    STARTBIT
194 0451 0E01               MVI   C,:01     C=BITCOUNTER
195 0453 CDFB04             CALL  DEL043
196 0456 78         ASCII   MOV   A,B
197 0457 0F                 RRC
198 0458 47                 MOV   B,A
199 0459 DCE504             CC    INS1      BITONE
200 045C D4D504             CNC   INS0      BITZER
201 045F 0C                 INR   C
202 0460 79                 MOV   A,C
203 0461 FE09               CPI   :09
204 0463 FA5604             JM    ASCII
205 0466 CD0605             CALL  DEL090
206 0469 CDAD04             CALL  BITONE    1ST STOPBIT
207 046C CD0B05             CALL  DEL144
208 046F CDAD04             CALL  BITONE    2ND STOPBIT
209 0472 2AEC02             LHLD  PTRIN
210 0475 EB                 XCHG
211 0476 2AC603             LHLD  PTROUT    COMPARE PTRIN
212 0479 13                 INX   D
213 047A 7A                 MOV   A,D
214 047B BC                 CMP   H
215 047C C2ED04             JNZ   INSDAT    *DELAY TO MAKE
216 047F 7B                 MOV   A,E       *BOTH JUMPS
217 0480 BD                 CMP   L         *EQUAL
218 0481 C23D04             JNZ   DATA
219 0484 2106FD             LXI   H,:FD06   CASS OUTPUT
220 0487 110000             LXI   D,:0000
```

```
221 048A CDFB04            CALL  DEL043
222 048D CDAD04  TRAILR    CALL  BITONE
223 0490 7A                MOV   A,D
224 0491 FE39              CPI   :39      TIME= 3 SEC
225 0493 CD0305            CALL  DEL121
226 0496 FA8D04            JM    TRAILR
227 0499 FB                EI
228 049A 214000            LXI   H,:0040
229 049D 3630              MVI   M,:30
230 049F 3EC9              MVI   A,:C9    OUTPUT POINTER
231 04A1 32DD02            STA   :2DD     BACK
232 04A4 210000            LXI   H,:0000
233 04A7 22DE02            SHLD  :2DE
234 04AA C3C203            JMP   RET
235 04AD CDCD04  BITONE    CALL  TOGGLE
236 04B0 CD1205            CALL  DEL171
237 04B3 CDCD04            CALL  TOGGLE
238 04B6 CD1205            CALL  DEL171
239 04B9 CDCD04            CALL  TOGGLE
240 04BC CD1205            CALL  DEL171
241 04BF CDCD04            CALL  TOGGLE
242 04C2 C9                RET
243 04C3 CDCD04  BITZER    CALL  TOGGLE
244 04C6 CD1605            CALL  DEL417
245 04C9 CDCD04            CALL  TOGGLE
246 04CC C9                RET
247 04CD 13      TOGGLE    INX   D
248 04CE 7B                MOV   A,E
249 04CF E601              ANI   :01
250 04D1 F620              ORI   :20
251 04D3 77                MOV   M,A
252 04D4 C9                RET
253 04D5 CD2005  INS0      CALL  DEL312
254 04D8 CDC304            CALL  BITZER
255 04DB 00                NOP
256 04DC 00                NOP
257 04DD 00                NOP
258 04DE 00                NOP
259 04DF C9                RET
260              CHECKS    EQU   :4E0
261                        ORG   :4E2
262 04E2 C30C04            JMP   TBLEND
263 04E5 CDFF04  INS1      CALL  DEL059
264 04E8 CDAD04            CALL  BITONE
265 04EB 37                STC
266 04EC C9                RET
267 04ED 00      INSDAT    NOP
268 04EE 7F                MOV   A,A
269 04EF C33D04            JMP   DATA
270 04F2 00      DEL032    NOP            **DELXXX=DELAY**
271 04F3 00      DEL028    NOP            ********XXX uS*****
272 04F4 00                NOP            *CALL+RET=DELAY*
273 04F5 00                NOP            *****27 uS******
274 04F6 00      DEL016    NOP
275 04F7 00      DEL012    NOP
276 04F8 00      DEL008    NOP
277 04F9 00      DEL004    NOP
```

18.00

BRT 2 LIMB. 18.30 Liedjes uit de lage landen : Een nieuwe dag (Johan Verminnen) ; Mestreech bleif toch Mestreech (Johnny Blenco) ; Laat me nog even blijven (Kitty Prins) ; Ik hou van Nederland (Willeke Alberti) ; Machtige mees (Louis Neefs) ; Frederiekje (Martine Bijl) ; Twee vrienden (De Snaar) ; Orgelconcert (Thérèse Steinmetz) ; Meisje (Miek en Roel) ; Als ik met vakantie ga (Jan Blaaser).
BRT 3 18.00 Nws — 18.15 Uitdeinend — 18.30 Het Omroepkoor van de BRT : Fabula Puèrilis, Karel De Branbander ; Dies irae voor koor, trompet, hoorn en trombone, Reznicek.
RTBF 1 18.05 Sportmagazine — 18.25 Katoliek godsdienstige uitzending.
RTBF 2 18.30-19.30 Verzoekplatenprogramma.
RTBF 3 18.05 Autrement dit.
HIL. 1 18.05 Akkoord — 18.30 Hobbyscoop. Programma over stereofonie, elektronika, foto en film, zendamateurs, DX-ers en Lucht- en Ruimtevaart.
HIL. 2 18.00 Nws — 18.10 Brood & Spelen — 18.40-19.10 Liturgie en kerkmuziek met o.a. Wat er niet in het liedboek staat, recente kerkliederen besproken en gezongen, en informatie over liedbewerkingen en motetten voor de adventstijd.
WDR 2 18.07-20.00 Blinklichter.
WDR 3 18.05-19.30 Pianoconc. nr 11, Mozart ; Schicksalslied op. 54, Brahms ; Simfonie nr 6, Dvorsjak.

```
278 04FA C9          DEL000  RET
279 04FB CDF604      DEL043  CALL   DEL016
280 04FE C9                  RET
281 04FF CDF204      DEL059  CALL   DEL032
282 0502 C9                  RET
283 0503 CDF904      DEL121  CALL   DEL004
284 0506 00          DEL090  NOP
285 0507 CDFF04      DEL086  CALL   DEL059
286 050A C9                  RET
287 050B CDFA04      DEL144  CALL   DEL000
288 050E CD0605              CALL   DEL090
289 0511 C9                  RET
290 0512 CD0B05      DEL171  CALL   DEL144
291 0515 C9                  RET
292 0516 CD0B05      DEL417  CALL   DEL144
293 0519 CD1205      DEL246  CALL   DEL171
294 051C CDF604              CALL   DEL016
295 051F C9                  RET
296 0520 CD1905      DEL312  CALL   DEL246
297 0523 CDF704              CALL   DEL012
298 0526 C9                  RET
299 0527                     END
```

```
****************************
* S Y M B O L   T A B L E *
****************************
```

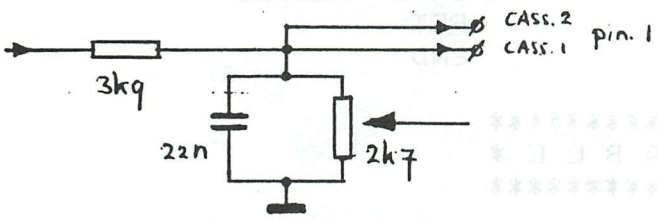
```
ASCII  0456   BIT=0  0348   BIT=1  033C   BITONE 04AD
BITZER 04C3   BYTE   034E   CHECKS 04E0   CHKSUM 0357
COLDST 02EE   CSTART 03C8   DATA   043D   DEL000 04FA
DEL004 04F9   DEL008 04F8   DEL012 04F7   DEL016 04F6
DEL028 04F3   DEL032 04F2   DEL043 04FB   DEL059 04FF
DEL086 0507   DEL090 0506   DEL121 0503   DEL144 050B
DEL171 0512   DEL246 0519   DEL312 0520   DEL417 0516
ENDFLG 0374   FALSE  0383   GOOD   0388   INS0   04D5
INS1   04E5   INSDAT 04ED   JUMP1  03EB   LAST   038A
LEADER 042E   LONG   0327   NEXT   039D   PRESET 0311
PTRIN  02EC   PTROUT 03C6   RET    03C2   RETURN 0369
SHORT  032A   START  0309   START1 03F5   START2 03FA
STOTEX 0379   TBLEND 040C   TCOUNT 0313   TOGGLE 04CD
TRAILR 048D
```

SOME REMARKS ABOUT CASSETTE INTERFACING

1. DAI cass.outputlevel more high

In the DAInamic-newsletter of april 1981 the DAIpc cassette-interface was first published (page 45). The scheme probably was based on DAIpc versions delivered before spring 1981. In the newer versions (REV.4 on main board) the 1k2-resistor of the output circuit is replaced by an 2k7-resistor (fig.1). This brings the output on a higher signal-level, though the output-impedance is rising too.

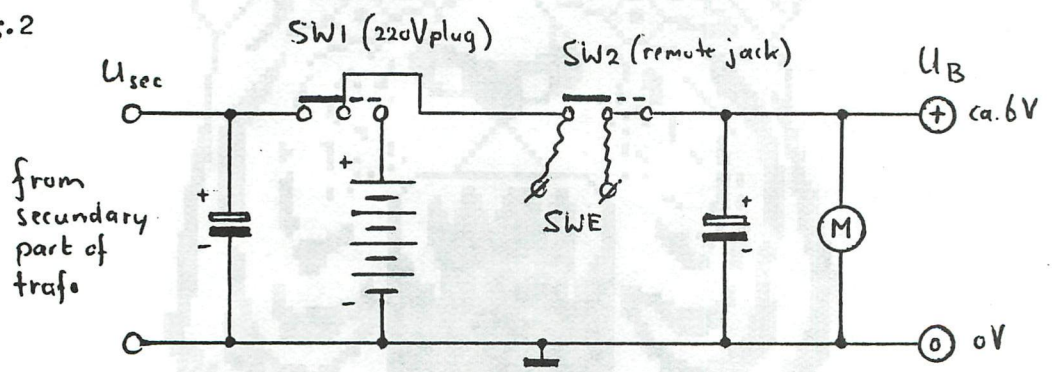
fig.1



2. Deminishing of motor-control current and indicating motor-running

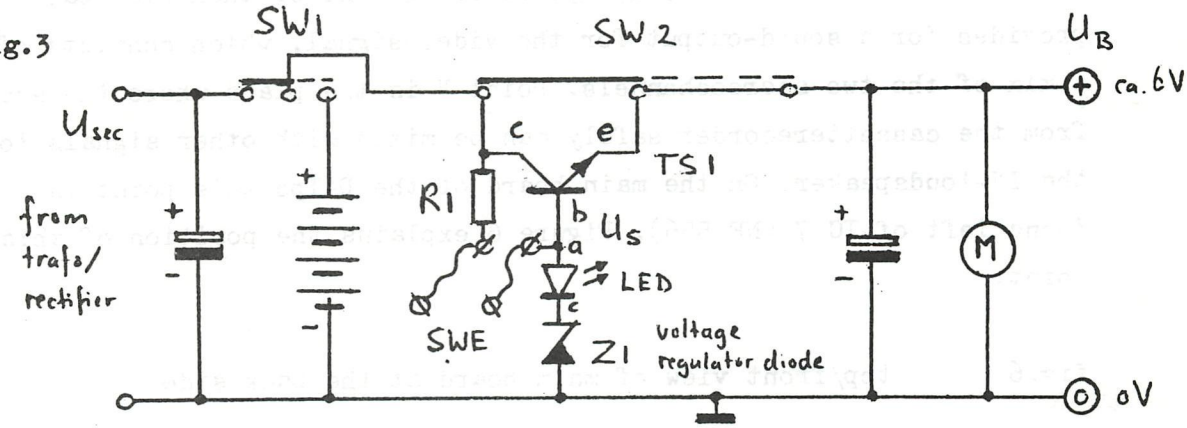
It may happen that the cassette-motor does not stop at the end of a LOAD- or SAVE-procedure or a BREAK despite usage of remote control. This is caused by the motorcurrent which is to be switched by a relais in the DAIpc (RL 1 and RL 2 according to Dessart, DAInamic page 207). DAI uses different types for the relais: CELDUC, HAMLIN and maybe others. In stead of replacing the present relais by types for higher-power-switching a small and lower-cost operation on the low-cost cassetterecorder might be the solution. Figure 2 shows a circuit which will be present, at least with great resemblance, in most low-cost cassetterecorders.

fig.2



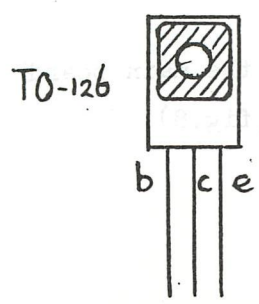
SW 1 is switching the power source between batteries and 220 V and is set by the 220 V-plug. SW2 is set open by plugging in the remote-jack. Power-control is then handed over to the external switch SWE, which is , in case of direct connecting, the DAI-relais. The current to be switched by the relais can be reduced by building in a quasi-power-stabilisation circuit into the recorder according to fig 3. At the same time a LED can be inserted for visual signalling motor-running.

fig.3



Care should be taken in chosing the values of Z1 and R1. U_B is not allowed to sink too much, as in that case the motor-speed will go under the necessary value. In an Audio Sonic CT-226 recorder a choice of 10 Ω for R1 and 4V7 for Z1 with a red LED did suffice. This,however, only if the LS-stage-power-usage, which is dependant on the set output-volume, is not maximal. Notice that the stabilisation was chosen to be not complete,' because of too much loss of voltage at U_B . The reduction in current to be switched by the DAI-relais depends on the h_{FE}-factor of the used transistor and U_S . Some experimenting with values will most likely result in a good solution.

fig.4 component details



TS 1

BD 433/435/437

h_{FE} = 85 ÷ 475
if I_C = 500 mA

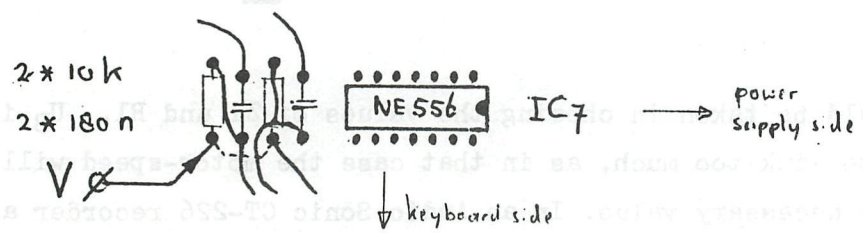
LED

a(node) = long pin
c(athode) = short pin

3. Sound from cassette on TV-loudspeaker

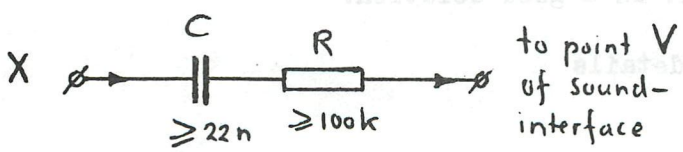
For searching the precise start or end of a certain file on cassette, it would be convenient to be able to hear the sound coming from the cassetterecorder without disconnecting the external-speaker (ear phone) plug. This feature is realised through a simple operation in the DAIpc. Figure 5 shows the sound- and noise-interface of the DAI-pc. The end of the circuit exists of 3 opamps of IC 14. One of them (IC 14a) provides for a sound-output for the video-signal, which consists of a mix of the two stereochannels. Point V is the place where the sound from the cassetterecorder safely can be mixed with other signals for the TV-loudspeaker. On the main board of the DAIpc this point is found left of IC 7 (NE 556). Figure 6 explains the position of this point.

fig.6 top/front view of main board at the back side



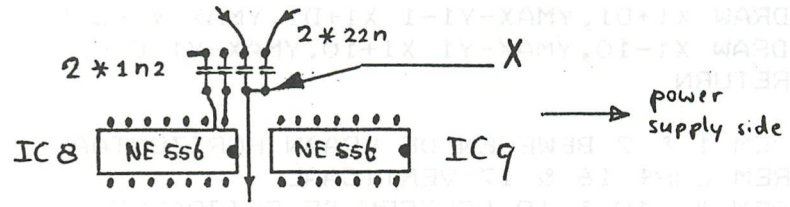
The cassette-sound-signal can be taken from the input-signal in the cassette-interface (DAInamic newsletter page 45), or from a cassette-signalclipper (e.g. the one suggested by DAI-club Eurocontrol Beek page 129). Both methods can be realised by means of a connection according to fig. 7.

fig.7



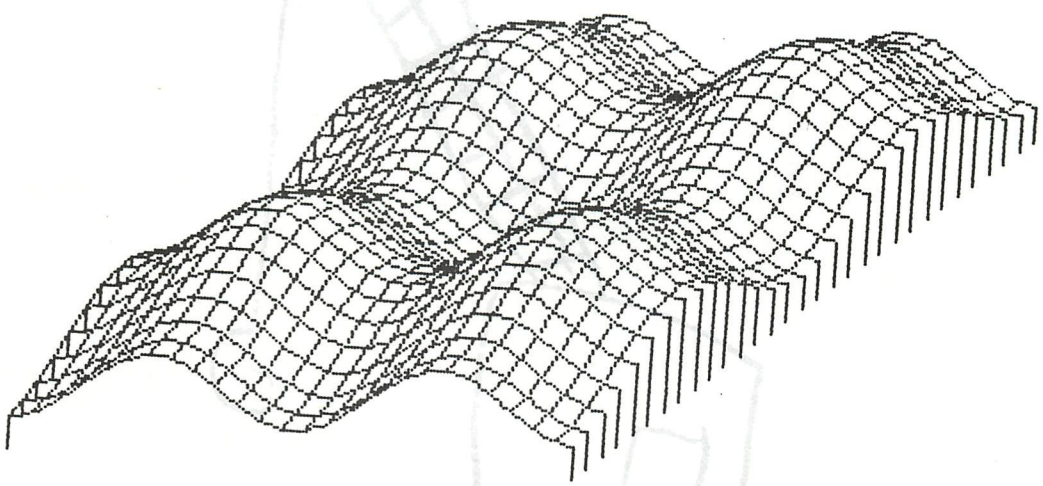
If no signal-clipper is used, point X can be found on the main board somewhat backsided of the middle between IC8 and IC9 (fig.8).

fig.8 top/front view of main board at the back side



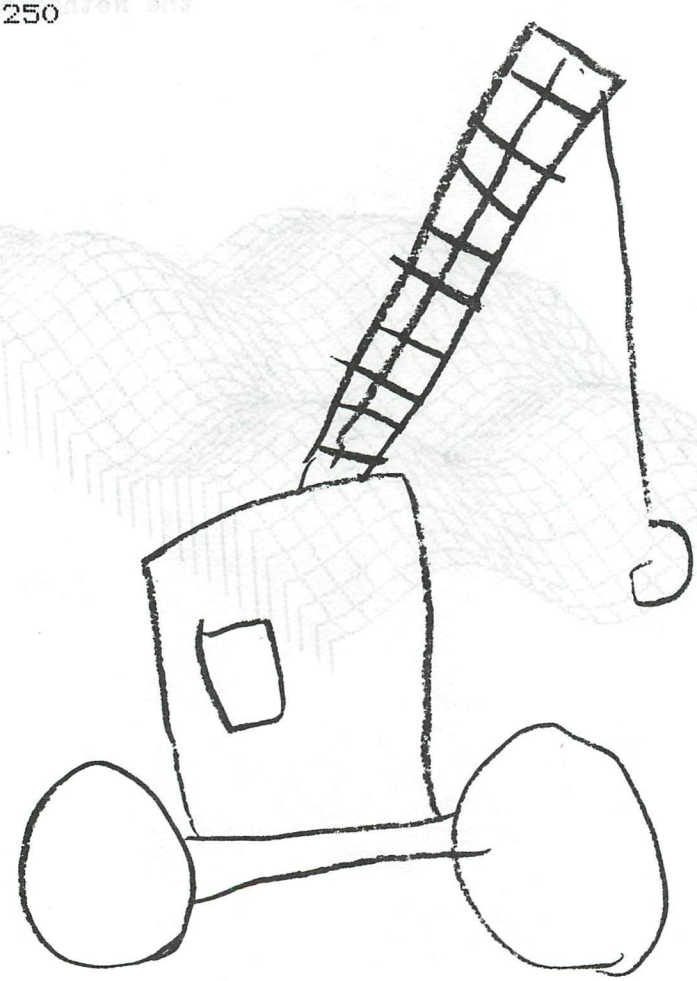
In case of usage of the Eurocontrol-clipper point X can be the collector of the BC 108-transistor. A resistor (e.g. 220 Ω) must be inserted between the + 12 V-line and X. Here too some experimenting with values of the resistors will bring about the objected goal: cassette-sound on the TV-loudspeaker with a reasonable volume.

Fred de Jong
Dormigstraat 23
6371 VX Schaesberg
the Netherlands



crane

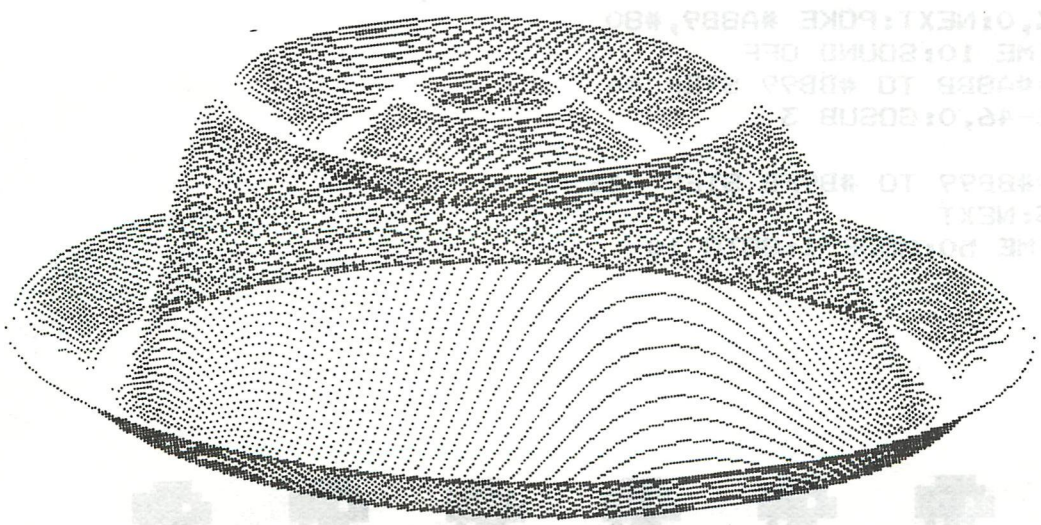
```
10          REM KRAAN
15          REM IMP INT
20          GOTO 140
30          DRAW X1,YMAX-1 X1,YMAX-Y1+1 C
40          DRAW X1-D1,YMAX-Y1-1 X1-D1,YMAX-Y1-5 C
50          DRAW X1+D1,YMAX-Y1-1 X1+D1,YMAX-Y1-5 C
60          DRAW X1-10,YMAX-Y1 X1+10,YMAX-Y1 C
70          RETURN

110         REM 1 & 2 BEWEGEN DE KRAAN HORIZONTAAL
120         REM CHR$ 16 & 17 VERTICAAL
130         REM "  18 & 19 BEWEGEN DE GRIJPARMEN
140         MODE 0:MODE 2:COLORG 0 8 10 13
150         DRAW 10,YMAX XMAX-10,YMAX 21
160         C=21.0:X1=11.0:Y1=5.0:D1=1.0:GOSUB 30
170         A=GETC:IF A=0.0 GOTO 170
180         X2=X1:Y2=Y1:D2=D1
190         IF A=18.0 THEN D2=D1+1.0:IF D2<11.0 GOTO 270
200         IF A=19.0 THEN D2=D1-1.0:IF D2>0.0 GOTO 270
210         IF A=16.0 THEN Y2=Y1-1.0:IF Y2>0.0 GOTO 260
220         IF A=17.0 THEN Y2=Y1+1.0:IF (Y2<40.0) AND (SCRN(X1+D1,
C           YMAX-Y2-5)=0) AND (SCRN(X1-D1,YMAX-Y2-5)=0) GOTO 260
230         IF A=50 THEN X2=X1+1.0:IF X2<XMAX-10.0 GOTO 260
240         IF A=49.0 THEN X2=X1-1.0:IF X2>10.0 GOTO 260
250         SOUND 1 0 15 0 FREQ(100.0):WAIT TIME 10:SOUND OFF :GOTO 170
260         C=20.0:GOSUB 30:C=21.0:X1=X2:Y1=Y2:D1=D2:GOSUB 30:GOTO 170
270         IF (SCRN(D2+X1,YMAX-Y1-5)=0) AND (SCRN(X1-D2,YMAX-Y1-5)=0)
C           GOTO 260
280         GOTO 250
```



```
5     REM  GRAFIC  'THE HAT'
7     REM  r.corswandt  9/81
8     REM
10    MODE 6:COLORG 12 Ø 15 Ø
12    REM  HORIZT. POS. - VERT.POS.
20    P=165.Ø     : Q=14Ø.Ø
30    XP=144.Ø : XR=1.5*3.14159
40    YP=56.Ø : YR=1.Ø : ZP=64.Ø
50    XF=XR/XP : YF=YP/YR : ZF=XR/ZP
60    FOR ZI= -Q TO Q-1.Ø
70    IF ZI < (-ZP) OR ZI>ZP GOTO 15Ø
80    ZT=ZI*XP/ZP : ZZ=ZI
90    XL=INT (Ø.5+SQR(XP*XP-ZT*ZT))
100   FOR XI= -XL TO XL
110   XT=SQR(XI*XI+ZT*ZT)*XF : XX=XI
120   YY=(SIN(XT)+Ø.4*SIN(3.Ø*XT))*YF
130   GOSUB 17Ø
140   NEXT XI
150   NEXT ZI
160   GOTO 16Ø : REM ENDE
170   X1=XX+ZZ+P
180   Y1=YY-ZZ+Q
190   DOT X1,Y1 15
200   IF Y1=Ø GOTO 22Ø
210   DOT X1,Y1-1 Ø
220   RETURN
```

Ein kleines Grafic-Programm " DER HUT "
Um eine anderen Schatteneffekt zu erzeugen können schwarz und weiß in Zeile Nr.190 und Nr. 210 vertauscht werden. Drei zusätzliche Grafik-Figuren kann man erhalten, wenn die beiden SINUS-Funktionen in Zeile Nr.120 durch COS ersetzt werden. Verändert man nur eine SIN-Funktion ist es nötig XP,YP und ZP etwas zu verkleinern 144 auf 124 ,56 auf 46 usw. da die Figur sonst zu groß werden würde.

Rainer Corswandt ,Lüdenscheid,Deutschland/W

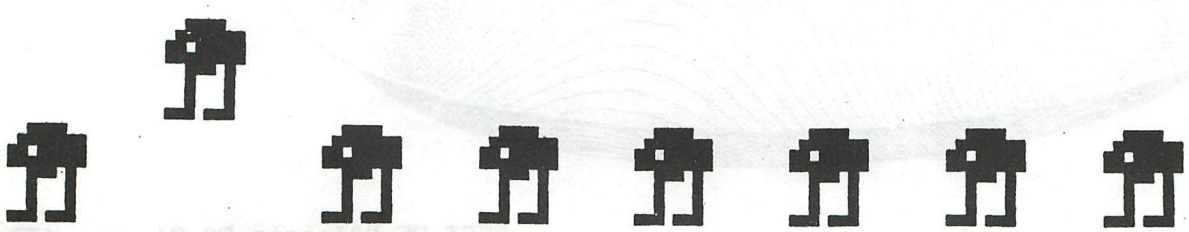


POKE-ACTION

DAI-BASIC offers wonderful DOT,DRAW & FILL commands.
This makes programming graphics very easy, but to illustrate that one can do fine tricks with POKE in the VIDEO-RAM,I wrote the following lines of program. To assist you in this way of graphics-programming,we publish the MEMORY-MAP of MODE 4.
Next time we will work out MODE 2 and/or MODE 6 .You can use these maps for the 16-color modes,but then you have to poke into 2 bytes to illuminate some dots.We POKE in ODD addresses, this has result on COLOR register COLORG X X X X .
The action-subroutine is in front of the program (3-6) to gain speed.Please compare the program in INTEGER & FLOATING POINT !

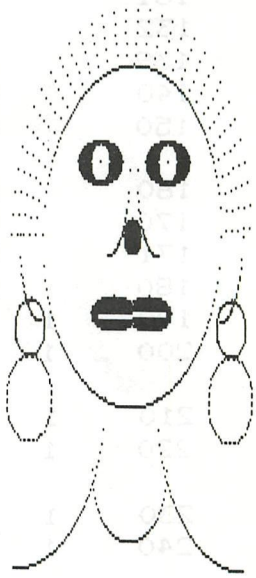
20-40 : creatures in position
50-70 : one creature down
74-78 : sorry,I couldn't stop him...
80-100: back to your position (too high)
110-120: back in the row!
130 : clear screen,again....

```
1     REM POKE-ACTION w.hermans  *** IMP INT ***
2     GOTO 10:REM SKIP ACTION ROUTINE
3     POKE X%,238:POKE X%+46,34:POKE X%+92,34
4     POKE X%+138,34:POKE X%+184,58:POKE X%+230,255
5     POKE X%+276,223:POKE X%+322,127:POKE X%+368,60:POKE X%+414,0
6     RETURN
10    MODE 4:COLORG 8 0 0 0
20    FOR X%=#B9D7 TO #B9FF STEP 4
30    GOSUB 3
40    NEXT
50    FOR X%=#B9FB TO #A8BB STEP -46
60    GOSUB 3
70    NEXT
74    NOISE 0 15
75    FOR X%=#A99F TO #A8B9 STEP -46:POKE X%,#80:WAIT TIME 2
76    POKE X%,0:NEXT:POKE #A8B9,#80
78    WAIT TIME 10:SOUND OFF
80    FOR X%=#A8BB TO #BB99 STEP 46
90    POKE X%-46,0:GOSUB 3
100   NEXT
110   FOR X%=#BB99 TO #B9FB STEP -46
120   GOSUB 3:NEXT
130   WAIT TIME 50:MODE 1:GOTO 10
```



sientje

```
5           REM sientje
10          REM hoofd en neus
12          COLORG 0 15 0 0
15          MODE 6
20          R=60.0:FOR X=-R TO R:Y=SQR(R*R-X*X)
30          DOT X+XMAX/2,Y+YMAX/2 15:DOT X+XMAX/2,-Y+YMAX/2 15:NEXT
40          FOR X=-6.0 TO 6.0:Y=SQR(6.0*6.0-X*X):DRAW X+167,Y+127 X+167,
C           -Y+127 15:NEXT
50          FOR X=0.0 TO 30.0:Y=SQR(30.0*30.0-X*X):DOT X/2+150,-Y+150 15:
C           NEXT
55          FOR X=-30.0 TO 0.0:Y=SQR(30.0*30.0-X*X):DOT X/2+185,-Y+150
C           15:NEXT

60          REM rechter oor
70          FOR X=-30.0 TO 0.0:Y=SQR(30.0*30.0-X*X)
80          DOT X/2+107,-Y+YMAX/2 15:NEXT

90          REM linker oor
100         FOR X=0.0 TO 30.0:Y=SQR(30.0*30.0-X*X)
110         DOT X/2+227,-Y+YMAX/2 15:NEXT

120         REM haar
130         FOR X=3.0*PI/2.0 TO 5.0*PI/2.0 STEP 0.1:S=80.0
140     1     FOR X1=3.2 TO 4.2 STEP 0.2
150     2       DOT XMAX/2+S*SIN(X)*X1/4.0,YMAX/2+S*COS(X)*X1/4.0 15
160         NEXT:NEXT

190         REM ogen
200         FOR X=-15.0 TO 15.0:Y=SQR(15.0*15.0-X*X)
210     1     DRAW X+145,Y/2+153 X+145,-Y/2+153 15
220         DRAW X+190,Y/2+153 X+190,-Y/2+153 15:NEXT
240         FOR X=-6.0 TO 6.0:Y=SQR(6.0*6.0-X*X)
250     1     DRAW X+145,Y+153 X+145,-Y+153 22
260         DRAW X+190,Y+153 X+190,-Y+153 0:NEXT
270         DOT 145,153 15:DOT 190,153 15

280         REM mond
290         FOR X=-15.0 TO 15.0:Y=SQR(15.0*15.0-X*X)
292     1     DRAW X+155,Y/2+100 X+155,-Y/2+100 22
294     1     DRAW X+179,Y/2+100 X+179,-Y/2+100 22
300     1     DRAW X+155,Y/2.5+100 X+155,-Y/2.5+100 15
310         DRAW X+179,Y/2.5+100 X+179,-Y/2.5+100 15:NEXT
320         FOR X=0.0 TO 1.0:DRAW 145,98+X 190,99+X 0:NEXT

330         REM oorringen
350         FOR R=0.0 TO 1.35 STEP 1.35:FOR X=0.0 TO 2.0*PI STEP 0.1:S=
C       2       10.0
360     2       DOT 100+100*R+S*SIN(X),95+S*COS(X) 15
370     2       DOT 100+100*R+S*SIN(X)*3.0/2.0,70+S*COS(X)*3.0/2.0 15
380         NEXT:NEXT

390         REM hals
400         FOR X=0.0 TO 60.0:Y=SQR(60.0*60.0-X*X)
410         DOT X+90,-Y+70 15:NEXT
420         FOR X=-60.0 TO 0.0:Y=SQR(60.0*60.0-X*X)
430         DOT X+245,-Y+70 15:NEXT
450         FOR X=-25.0 TO 25.0:Y=SQR(25.0*25.0-X*X)
460         DOT X+167,-Y+45 15:NEXT
500         PRINT CHR$(12):CURSOR 23,2:PRINT "S I E N T J E "
510         COLORG 0 15 15 0:WAIT TIME 30:COLORG 0 15 0 0:WAIT TIME 200
520         GOTO 510
```



```
10        REM COMPLEMENTEN EN SUPPLEMENTEN
11        REM UIT CREATIVE COMPUTING SEPT 1980
12        REM AUTEUR R.CARLSON, USA
65        PRINT CHR$(12)
69        CLEAR 3000
70        DIM R(100.0),A$(8.0),B$(10.0),C$(12.0),Q$(3.0)
80        PRINT :PRINT "  Dit programma helpt bij het oplossen
C         van "
90        PRINT :PRINT "     wiskundeproblemen met complementen
C         "
91        PRINT :PRINT "     en supplementen . "
100       PRINT :PRINT " Typ JA in als je met de computer wilt
C         werken, "
110       PRINT :PRINT " of NEE als je een aantal opgaven  wilt
C         ."
119       PRINT
120       INPUT Q$
130       IF LEFT$(Q$,1)="J" THEN T=1.0
131       IF LEFT$(Q$,1)<>"J" THEN T=0.0
132       IF T=0.0 GOTO 150
139       PRINT
140       PRINT :INPUT " Hoeveel sommen wil je ? ";N
150       PRINT CHR$(12)

160       REM DEF FNA(A,B,C)=(A+B*C*90)/(B+1)
170       REM DE FUNCTIE A ZAL DE VERGELIJKING OPLOSSEN DIE
171       REM DE WAARDEN VAN A,B,C GEBRUIKT.
180       FOR M=1.0 TO N
190     1   A=INT(60.0*RND(M))+1.0
200     1   IF RND(M)<0.5 THEN A=-A

210     1   REM A GAAT VAN -60 TOT +60
220     1   B=INT(4.0*RND(M))+1.0

230     1   REM B=1,2,3,4
240     1   C=INT(2.0*RND(M))+1.0

250     1   REM C=1 OF 2 VOOR HET COMPLEMENT OF HET SUPPLEMENT
260     1   R(M)=(A+B*C*90.0)/(B+1.0)

270     1   REM R(M) BEVAT HET HUIDIGE ANTWOORD
280     1   IF C=1.0 THEN C$=" complement ":IF C<>1.0 THEN C$="
C       1   supplement "
290     1   IF SGN(A)=1.0 THEN A$="  meer  "
291     1   IF SGN(A)<>1.0 THEN A$=" minder "
300     1   ON B GOTO 310,330,350,370
310     1   B$=" "
320     1   GOTO 380
330     1   B$=" tweemaal "
340     1   GOTO 380
350     1   B$=" driemaal "
360     1   GOTO 380
370     1   B$=" viermaal "
380     1   PRINT :PRINT " Som  ";M;". ";
389     1   PRINT
```

```
390     1     PRINT :PRINT " Een hoek is ";ABS(A);" graden ";A$;
391     1     PRINT :PRINT "   dan ";B$;" zijn ";C$;" .":PRINT
400     1     PRINT :PRINT " Wat  is  de  hoek ? "
410     1     PRINT
420     1     K=0.0
430     1     IF T<>1.0 THEN 680
440     1     PRINT
450     1     PRINT :INPUT " Typ nu de uitkomst in ! ";A1
460     1     PRINT
470     1     IF ABS(A1-R(M))>1E-3 THEN 510
479     1     PRINT
480     1     PRINT " Inderdaad, het antwoord is ";R(M)
490     1     PRINT
500     1     GOTO 680
510     1     IF A1<>C*90.0-R(M) THEN 550
519     1     PRINT
520     1     PRINT " Probeer de ";C$
530     1     GOTO 450

540     1     REM K IS VOOR VOORTGANGSCONTROLE
550     1     K=K+1.0
560     1     ON K GOTO 570,590,620,650
570     1     PRINT :PRINT " Weet je, dat hoeken ";C$;" zijn, ";:
C       1     PRINT CHR$(8);" als ze samen ";C*90.0;" zijn . "
580     1     GOTO 450
590     1     PRINT :PRINT " Probeer deze vergelijking  eens ."
600     1     PRINT :PRINT " X= ";B;"(";C*90.0;"-X) +";A
610     1     GOTO 450
620     1     PRINT :PRINT " Je vergelijking vereenvoudigt dit : "
630     1     PRINT :PRINT B+1.0;"X= ";B*C*90.0;" + ";A
640     1     GOTO 450
650     1     PRINT :PRINT " Het juiste antwoord is ";R(M)
660     1     PRINT :PRINT " Probeer nog een probleem . "
670     1     PRINT
679     1     IF T=0.0 GOTO 150
680         NEXT M
690         PRINT
700         PRINT
710         IF T=1.0 THEN 770
720         PRINT :INPUT " Wil je de antwoorden ?";Q$
730         IF LEFT$(Q$,1)="N" THEN PRINT " Typ dan de uitkomst in
C           ! "
735         INPUT A1
740         FOR M=1.0 TO N
750     1     PRINT :PRINT " Som ";M;".  Het antwoord is ";R(M);"
C       1     graden."
755     1     PRINT :PRINT " Is er veel verschil tussen de twee
C       1     uitkomsten ? "
760         NEXT M
770         END
775         PRINT :PRINT
```

RESTART ROUTINES IN THE DAI pC.

1. The 8080 microprocessor in the DAI knows 8 instruction codes that are one-byte CALL instructions: RST 0 through RST 7. In many computer systems, these instructions are used in combination with interrupts.
This article will describe the way in which the DAI uses these interrupts.

2. RST 1, RST 4, RST 5.

These 3 restart instructions are used for bank switching. In the memory area E000-EFFF the DAI uses 4 banks of each 4K ROM 'in parallel'. Via bits 6 and 7 of output port FD06, one of these banks is selected.
Normally, bank 0 is switched on, but via software instructions one of the other banks can be activated. Therefore, the RST 1, RST 4 and RST 5 instruction codes are used. These instructions are followed by one data byte.

When the program counter encounters one of these RST instructions, it goes to the interrupt vector routines in the area 0000-003F. The interrupt vector address from the area 0062-0071 is loaded, and the program counter is set to this address.

The routines which are found on the vector addresses prepare the selection of the required ROM bank:

RST 1: ROM bank 3 (encode - utility)
RST 4: ROM bank 1 (math. package)
RST 5: ROM bank 2 (screen package)

Via the general ROM bank switching routine on address C6CF the selected ROM bank is activated.

The data byte after the RST instruction indicates which address in the particular ROM bank has to be jumped to. It is an offset to the startaddress E000.

Example: RST 5, data 18: ROM bank 2, address E018.
There a jump to the screen mode changing routine can be found.

When switching to another ROM bank, the previous selection is saved in memory. On return from the switched bank, the old bank select is restored again.

3. The other Restart instructions.

All other RST instructions are used on interrupt base. The interrupts are generated by the timers in the 5501 Timer and Interrupt controller.

4. RST 7, Clock interrupt.

The 20 ms page blanking signal for the TV is used as clock signal. Each time this clock interrupt is present, the program counter is set on 0038. Via the interrupt vector routine, the program counter is set on address D9A9.

The RST 7 routine on this address enables only stack interrupts and checks the contents of timer 01BE/F. Each time when a RST 7 interrupt is present, this counter is decremented. As long as it is not zero, nothing happens.

When this timer is zero, then on each RST 7 interrupt the clock timer 01C0 is decremented. Again, nothing happens when it is not zero.

But when the clock timer is also zero, a RST 5, data 12 routine is activated.

This routine flashes the cursor according the information in the pointers 0074-0077 (see memory map).
After changing the contents of the screen location pointed by the cursor, the old interrupt mask is restored and the program returns from interrupt to its normal sequence.

5. RST 6, Keyboard interrupt service.

Each time an interrupt from timer 4 is present, the program counter is set to D578 via the interrupt vector routine on address 0030.

The RST 6 routine reloads timer 4 and enables only clock and stack interrupts.
The keyboard counter 01C1 is decremented on each RST 6 interrupt. When the result is not zero, the routine is aborted. Else, the keyboard counter is reloaded and a keyboard scan is performed (the GETC routine).
On exit, the original interrupt mask is restored again.

6. RST 3, Sound interrupt.

On an interrupt from timer 3, the interrupt vector routine on address 0018 load D755 into the program counter.

This RST 3 routine enables clock and sound interrupts only. Timer 3 is reloaded and ROM bank 1 is selected.

Now the program continues on address EE6E in bank 1, which is the Sound program.
On exit, the old ROM bank and the old interrupt mask is restored again.

7. RST 2, Stack interrupt.

When stack overflow occurs, an RST 2 interrupt is the result. Via address 0010 in the interrupt vector routine area, the program counter is loaded with D9E2.

The RST 2 routine resets the stackpointer on F900. The running of inputs and the encoding of stored lines is disabled. The input is returned to the keyboard and the timers for sound and keyboard interrupts are reloaded.

Then the error messages 'STACK OVERFLOW' is printed.

8. RST 0, Utility.

The RST 0 interrupt is used only by the Utility program. On this interrupt, the program counter is set on 0000.

The vector address, required in this interrupt vector routine, is only present after a Z2 or a Z3 command in utility. Then location 0062/63 is loaded with EB5D, the startaddress of the RST 0 routine in ROM bank 3.

The RST 0 interrupt is caused by timer 0; it is used in the LOOK routine in utility.

On a RST 0 interrupt, all CPU registers are saved in the RAM area 0053-005E. Then the program continues on a address which is given by the LOOK routine and indicates the next instruction to be performed.

The program checks this instruction. If it is a CALL or a RST instruction, then the next address is saved too.

Then a check is performed to see if the next instruction address is within the frame given by the LOOK window. When the result is positive, the contents of all registers, including stack pointer, flags and program counter, is displayed on the screen.

On exit, the timer 0 is reloaded, the interrupt mask set and - among other instructions - the CPU registers are restored again.

Because the program runs now under RST 0 interrupts, it runs much slower than in normal runtime!

Jan Boerrigter - okt.'81

ADDITIONAL INFORMATION ON MEMORY MAP

After working out the whole Utility program (1½K on EA00-EFFF in ROM bank 3), the following updates on the memory map are available.

0045-0046	Not used.
0047	Used to store EI/DI instructions in RST Ø.
0048/49	High address look window.
004A/4B	Low address look window.
004C-4F	Store current instruction of traced program.
0050	Flag for LOOK: FF First time L addr laddr haddr 00 only L or L laddr haddr
0051/52	Address current instruction (I)
0053	Contents accumulator (A)
0054	Contents flag register (F)
0055	Contents register B
0056	Contents register C
0057	Contents register D
0058	Contents register E
0059	Contents register H
005A	Contents register L
005B/5C	Contents stackpointer (SP)
005D/5E	Contents program counter (P). Points to next instruction to be performed.
0060	Init. value for TICC (FC after Z2 instruction).
0061	Init. value for GIC (1B after Z2 instruction).
0062/63	IØUSA: By Z2 instruction set to EB5D.

... IF YOU ACCIDENTLY HIT 'RESET' .....

In the last Newsletter (page 134) a routine is given to save the pointers of the HEAP, the textbuffer and the symboltable in the RAM-area at the beginning of each BASIC program.
The RAM addresses 0045 - 004E are used for this purpose.

This is a very useful method, although some reservation must be made.
The same RAM-area is also used by the DAI Utility routines to save the contents of CPU registers etc.
Because normally BASIC programs and the Utility routine are not used together, it doesn't give any problem.
Only if you want to look in the particular RAM area (via UT Display e.g.), you will find complete different values!!

So take care.

Jan Boerrigter.

```
 0 zwart
 1 blauw
 2 d.rood
 3 rood
 4 paars
 5 groen
 6 d.bruin
 7 l.bruin
 8 grijs
 9 blauw
10 oranje
11 rose
12 l.blauw
13 l.groen
14 geel
15 wit
```

alle adressen in HEXvorm!

29B-29C	start heap
29D-29E	size heap
29F-2A0	start text buffer
2A1-2A2	start symbol table
2A3-2A4	end of symbol table
2A5-2A6	bottom screen ram

131,0	output scrn+ RS232
131,1	screen only
131,2	edit buffer
135,2	read from edit buffer

75	cursor symbol
74	cursor mode
72-73	cursor position

MODE	XMAX	YMAX
1/2	71	64
3/4	159	129
5/6	335	255

40,28 cass motor 1 ON
40,18 cass motor 2 ON
40,30 1 and 2 OFF

```
COLORG R1 R2 R3 R4
       20 21 22 23
16 :R2*R1 R4*R3
17 :R1*R2 R3*R4
18 :R3*R1 R4*R2
19 :R1*R3 R2*R4
```

32K 7XXX
12K 2XXX
8K 1XXX

MERGE
°CLEAR XXX
°LOAD"A"
°EDIT BREAK/BREAK
°LOAD"B"
°POKE 135,2

IMP INT *** IMP FPT
°IMP FPT
°CLEAR XXXX
°EDIT BREAK/BREAK
°IMP INT
°POKE 135,2

LIJN	CTRL	COLOR	LIJN	CTRL	COLOR
23	BFEF	BFEE	11	B9A7	B9A6
22	BF69	BF68	10	B921	B920
21	BEE3	BEE2	9	B89B	B89A
20	BE5D	BE5C	8	B815	B814
19	BDD7	BDD6	7	B78F	B78E
18	BD51	BD50	6	B709	B708
17	BCCB	BCCA	5	B683	B682
16	BC45	BC44	4	B5FD	B5FC
15	BBBF	BBBE	3	B577	B576
14	BB39	BB38	2	B4F1	B4F0
13	BAB3	BAB2	1	B46B	B46A
12	BA2D	BA2C	0	B3E5	B3E4

CTRL&COLOR BYTES IN A-MODE

MODE	CTRL	COLOR	LIJN
1A/2A	BAE7	BAE6	3
	BA61	BA60	2
	B9DB	B9DA	1
	B955	B954	0
3A/4A	ACD3	ACD2	3
	AC4D	AC4C	2
	ABC7	ABC6	1
	AB41	AB40	0
5A/6A	7557	7556	3
	74D1	74D0	2
	744B	744A	1
	73C5	73C4	0

```
FD00 b2 page signal
     b3 serial out rdy
     b4 right paddle
     b5 left  paddle
     b6 random data
     b7 cass. input
FD01    Trigger paddle
FD04 0-3 volume ch.1(0)
     4-7 volume ch.2(1)
FD05 0-3 volume ch.3(2)
     4-7 volume noise
FD06 b0 cass.out
     b1/2 paddle select
     b3 paddle enable
     b4 cass motor 1
     b5 cass motor 2
     b6/7 ROM BANK SWITCH
```

```
FF00 ser.inp.buff
FF01 b0-6 keyb.inp.
     b7 in7 DCE
FF02 Interr.reg.
FF03 b1 frame error
     b2 overrun error
     b3 rec.buf.loaded
     b4 trans.buf.empty
FF04 COMMAND REGISTER
FF05 BAUD RATE REGISTER
FF06 ser.out buf.
FF07 keyb.output
FF08 interr.mask reg.
```

```
FF09 TIMER 0
FF0A TIMER 1
FF0B TIMER 2
FF0C TIMER 3
FF0D TIMER 4
8253
CH 0 FC00/FC01
CH 1 FC02/FC03
CH 2 FC04/FC05
STATUS FC06/FC07
```

TEST EVENT
PEEK(éFD00) IAND 32
PEEK(éFD00) IAND 16
PEEK(éFD00) IAND 48

x-bus layout– 8080 instructionset
random distribution real time clock
fasing keyboard music–propeller
short random routine music scale
dot-draw-fill-scrn – vier op een rij
wilhelmus – screencopy mx-80
funny sirene – radar-simulation
endless continuation lines–citroen
towers of hanoi roterende ellipsen
attractive cursor videoram mode o
memory map mode o–barricade
grafiek 5e graadspolynomen –
test-soundmonitor-val/str grafiek
8080 simulator – variabelen atlas
heap-story videomonitor-interface
16 colors in 4-color modes sirenes
talk editor–variables in dai basic
dipswitch setting epson raketspel
carpenters mystery timer routines
nos basicode clock for math-test
remarks cassette interfacing
crane –romroutines & entrypoints
the hat (graphics) – tubular bells
poke-action – paddelen met fgt
sientje (graphics) prime numbers
complementen & supplementen

THE BEST OF DAINAMIC 80~81