Dr. Wolfgang Werbeck
D – 78 Freiburg
Innsbrucker Straße 2
Telefon 49 45 77

Freiburg, 25.2.82

letter from mr Werbeck

Herrn
W. Hermans
Heide 4
B-3171 Westmeerbeek

Lieber DAInamic-Freund.

Beim Blättern in älteren Heften der Elektronik-Zeitschrift "Elektor" (in den Niederlanden "Elektuur") fand ich in der Dezember-Nummer 1981 einen interessanten Artikel mit dem Titel "IPROM". Ich dachte dabei sofort an den leeren 24-poligen Sockel auf dem Board meines DAI P.C. Denn der Arithmetic Processor AM9511 ist mir viel zu teuer.
Beim IPROM wird ein 2 K x 8 Bit - RAM (HM 6116 LP, Preis DM 40,00) als "Pseudo-ROM" verwendet. Man kann - nach entsprechendem Aufbau mit zusätzlichen Bauteilen - ein Programm wie bei einem RAM einfach laden. Es ist dann durch einen kleinen Schalter vor Überschreiben geschützt und bleibt mit Hilfe von zwei Knopfzellen auch nach Abschalten des Computers erhalten. Man benötigt also weder ein Programmiergerät noch ein Löschgerät und hat dennoch ein ROM, in das man jederzeit ein neues Programm einschreiben kann.
Leider ist die Anschlußbelegung des freien Sockels völlig ungeeignet für die Aufnahme eines IPROMs.
Fragen Sie doch einmal einige Hardware-Spezialisten im DAInamic Club, ob sie eine Lösung für dieses Problem wissen.

Ich habe mir übrigens die vier zusätzlichen Tasten nach der Bauanleitung von Herrn Dessart (DAInamic Nr.7,S.207 ff.) eingebaut und bin sehr zufrieden damit.

Mit besten Grüßen

W. Werbeck

IMP INT ******* G E T A L L E N ******* IMP FPT

We kennen in programma's vele plaatsen waar gebruik gemaakt wordt van getallen. Deze getallen kunnen echter op verschillende manieren voorkomen en niet iedereen is bekend met of zich bewust van deze verschillen.

I) Getallen kunnen voorkomen als constanten of als variabelen
Voorbeeld: DOT X,Y+2 9 hier zijn X en Y variabelen en 2 en 9 constanten.
II) Getallen kunnen voorkomen in drie vormen in een programma als geheel getal (integer) of als breuk (floating point); de derde verschijningsvorm is hier niet interessant namelijk de geschreven vorm (string).

Om alle misverstanden te voorkomen wil ik nu reeds van tevo ren zeggen dat de programmeur in alle gevallen bepaalt welk type een getal zal zijn.

U weet waarschijnlijk al uit de foutmeldingen dat getallen in elk type binnen bepaalde grenzen moet liggen.
Bij gehele getallen (integers) zijn deze grenzen +/- 2^31, bij drijvende komma getallen (brrrr floating point dus) heb ik destijds vastgesteld dat de grenzen hier afhankelijk zijn van de omstandigheden. In een programma kon ik de variabele op de waarde 9.22E18 krijgen maar bij invoer van C=8E18 kreeg ik reeds overflow.
Als we in een programma trouwens de waarde van een variabele steeds laten toenemen krijgen we op den duur OVERFLOW; ook als we een getal steeds kleiner laten worden (zonder de min ervoor steeds groter) krijgen we OVERFLOW.
Als we een variabele echter door delen met een getal groter dan 1 kleiner laten worden (het getal komt steeds dichter bij nul) krijgen we geen waarschuwing van de DAI voor deze UNDER FLOW maar wordt de variabele gewoon nul gemaakt.
Hier dient in bepaalde gevallen wel degelijk rekening mee gehouden te worden:Als we een getal eerst honderd maal halveren en dan honderd maal verdubbelen zouden we het oorspronkelijke getal terug moeten krijgen, we zullen afhankelijk van de startwaarde iets anders krijgen, meestal nul.

```
programma:    IMPFPT
              10 A=123
              20 FOR I=1 TO 100:A=A/2:NEXT
              30 FOR I=1 TO 100:A=A*2:NEXT
              40 ?A
```

Bij het bestuderen van programma's van anderen heb ik vaak moeten vaststellen dat het niet bij iedereen bekend is dat programma's sneller werken als integerwaarden gebruikt worden. Een ander voordeel van het gebruik van integers ligt in de logica: ik vind het vreselijk als iemand (via zijn/haar DAIprogramma beweert dat ik door een bepaalde slag of zet 3.0 pionnen heb veroverd, alsof je er ook 2.74653 zou kunnen veroveren. Ik hoor nu al de protesten: Ja maar dat wilde ik niet dat deed de DAI. Klopt omdat U hem daartoe de opdracht gaf al geef ik toe dat het vaak een impliciete opdracht is geweest en waar U niet aan gedacht hebt. De meest vergeten opdracht van dit type is de opdracht IMP FPT die U (ja U) geeft door de DAI aan te zetten of reset te geven.
Na deze IMP FPT zijn alle variabelen en vele (niet alle) con stanten van het fpt-type.
Eerst gaan we wat in op het snelheidsaspect. Ik hoop dat U mij niet gelooft en ik zal ook niet trachten iemand te overtuigen als de DAI dat voor mij kan doen.

Ga achter de DAI zitten, zet hem aan of reset en tik in:

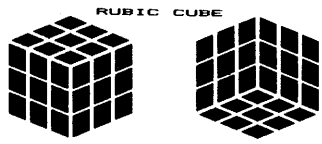
```
10 MODE 1:K=10
20 A=0:B=0
30 DRAW XMAX-1,YMAX-2 A+1,B+1 K
40 B=B+2:IF B+1<YMAX-1 GOTO 30
50 IF K<>0 THEN B=0:K=0:GOTO 30
60 END
```

Geef RUN en tracht de tijd vast te stellen die de DAI hiervoor nodig heeft, eventueel programma uitbreiden met 5 J=0 en 55 J=J+1:IF J<50 GOTO 10.
geef LIST en zie dat het programma niet meer is wat U intikte.
Geef nu reset of machine uit/aan en tik weer precies hetzelfde in, tik RUN en ik denk dat U overtuigd bent.
N.B. het progamma is ter demonstatie het kan heus sneller:

```
10 MODE2:M=XMAX-1:N=YMAX-1:FOR K=10 TO 0 STEP -10
20 FOR B=1 TO N STEP 2:DRAW M,N 1,B K:NEXT:NEXT:END
```

Als U de listing van de fpt-versie nog herinnert zult U opgemerkt hebben dat er 20 A=0.0:B=0.0 stond. De ingevoerde constanten blijken automatisch in fpt-notatie te staan.
Vreemd genoeg blijkt echter in regel 30 dat de daar ingevoerde constanten wel in de door ons ingetikte vorm staan.
We kunnen dit begrijpen als we in het handboek lezen dat getallen voor de bepaalde gebruikstoepassingen vaak in een standaard vorm verwacht worden.
Bij een toewijzing zoals regel 20 kijkt DAI naar IMPINT resp IMPFPT maar bij een DRAW als regel 30 verwacht hij integers. Zijn variabelen of constanten niet in de verwachtte vorm, zet de DAI ze voor ons om naar het juiste type. Dit kost echter wel tijd!!!!
We analyseren de hopeloze regel 30:
Voor elke keer wordt de XMAX en YMAX aangeroepen.
Elke keer wordt deze constante met een constante verminderd het resultaat is dus ook ... constant.
Bij de variabele A ,die overigens een constante waarde heeft wordt volkomen overbodig elke keer 1 opgeteld, we hadden A vanzelfsprekend 1 groter kunnen definieren. Maar het is nog erger: bij A een fpt-variabele wordt 1 een int-constante opgeteld. Dit is onmogelijk en dus wordt eerst de integer omgezet in fpt, daarna de optelling gedaan en dan wordt het geheel weer omgezet in int voor de DRAW-opdracht.
U kunt zelf nagaan dat zeer veel instructies integers verwachten of daar ook mee kunnen werken, als U ze echter fpt voert protesteren ze niet maar straffen U wel af door het programma trager af te werken of zelfs andere dingen te doen dan U bedoelde.
Ter illustratie geen protest bij COLORG 0.0 5.3 9.8 3.4 of FOR I=0 TO XMAX maar wat gaat dat allemaal veel sneller in integers.
U begrijpt dat ik er een voorstander van ben om standaard te beginnen met IMPINT (inderdaad spatie niet nodig) en dan wel ! te zetten (ook .0) indien dit gewenst is.
N.B. er is een verschil tussen IMPINT en IMPFPT gevolgd door IMPINT A-Z en omgekeerd, speelt er wat mee en verbaast U.

*** aanwijzingen gebruik RUBIK'S CUBE ***



De gekozen kleuren zijn de standaard kleuren zoals op de originele kubus te vinden zijn. De gekozen volgorde is die zoals de kubus in de verpakking zit.
Komen de kleuren bij U niet zo goed over (ik gebruik RGB) kunt U de kleuren wijzigen in regel 2010.
Pas op geef eerst IMPINT voordat U EDIT2010 doet.
De linker kubus in beeld geeft voor,rechter en bovenvlak, de rechter kubus is de kubus in de hand "horizontaal" gedraaid zo dat U onder,linker en achtervlak ziet. Speel wat na "I" met de vlakken zodat U weet wat U doet.
EDe Equator zit tussen boven en ondervlak, het Middenvlak tussen linker en rechtervlak en het Staande vlak tussen voor en achtervlak, te besturen met E,e,M,m,S en s.
Andere kleuren voor de vlakken zijn te geven in regel 2013 t/m 2019. zie de voorbeelden in regel 2011 en regel 2012.
Het middelste blokje van elk vlak kan ook een andere kleur krijgen (dan wel alle zijvlakken dezelfde kleur middenblokje) door de REM's in regel 2080 en 2090 te verwijderen.
Het middenblokje is dan zwart; andere kleur door 2080 te beginnen met KM=#70 als kleur 7 gebruikt moet worden.
In de stand W is het mogelijk de teller te resetten door C
Het draaitempo kan veranderd worden door de loop op regel 50 te veranderen. De draaiingen kunnen net zoals bij LIST beinvloed worden. Veel plezier. fhd

---

TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS

Begin programmeren in bijna alle gevallen met IMPINT

Zet de uitleg van een programma wel in het begin van de "RUN" maar niet in het begin van de LIST.
Dwz begin bv met 10 CLEAR 3000:GOSUB 10000: enz
terwijl we de uitleg dan zetten vanaf regel 10000; het programma wordt sneller en overzichtelijker.

Doe niet mee aan de domme amerikaanse gewoonte te beginnen met HOE HEET JE ? om dat gegeven vervolgens nooit meer te gebruiken.

Begin niet met "WILT U UITLEG ?"
Er zijn twee mogelijkheden:
I) uitleg klein geef die dan ongevraagd.
II) uitleg groot maak een apart uitlegprogramma.
    voordelen 1-als U geen uitleg nodig hebt minder inlezen
              2-programma mogelijk iets sneller
              3-hele grote programma's passen door splitsing wel in het geheugen.
              4-inlezen hoofdprogramma tijdens geven uitleg

Begin elk groter programma met een "leader" of "header"
Hier bedoel ik mee een beeld waarin de naam van het programma staat en de naam van de auteur.
Liefst op een enigszins verantwoorde manier.

Laat de aankondiging en de uitleg niet zolang in beeld staan als U goeddunkt maar laat de gebruiker van het programma deze vrijheid, door bv te wachten op indrukken spatiebalk.

Geef bij het vragen om een moeilijkheidsgraad aan wat gemakkelijk en wat moeilijk is.

snoopy

```
200   MODE 6A:PRINT CHR$(12);:POKE #75,32
210   COLORT 12 0 12 12:COLORG 12 0 10 15
220   POKE #7557,#4A:POKE #7556,223
230   PRINT "SNOOPY":POKE #74D0,208
240   DIM C%(4)
250   C%(1)=12:C%(2)=0:C%(3)=10:C%(4)=15:K%=1
260   FOR X%=0 TO 48 STEP 2
270   FILL X%,Y% 330-X%,210-Y% C%(K%)
280   K%=K%+1:IF K%>4 THEN K%=1
290   Y%=Y%+2:NEXT
300   REM *** TEKENING
310   RESTORE
320   FOR Y%=1 TO 44:READ A$:A$=RIGHT$(A$,LEN(A$)-1):KY%=Y%*2
330   FOR X%=1 TO LEN(A$)-1:T$=MID$(A$,X%,1):IF T$=" " THEN 390
340   IF T$="X" THEN C%=0:GOTO 370
350   IF T$=":" THEN C%=10:GOTO 370
360   IF T$="-" THEN C%=15
370   DOT 100+X%,150-KY% C%:DOT 100+X%,149-KY% C%
380   DOT 230-X%,150-KY% C%:DOT 230-X%,149-KY% C%
390   NEXT:NEXT
400   REM *** AUTOMATISCHE KLEURWISSELINGEN
410   CURSOR 2,2:PRINT "OM DE KLEUREN TE DOEN STILHOUDEN,DRUK OP DE SPATIEBALK ";
420   C1%=RND(16):C2%=RND(16):COLORG 12 0 C1% C2%:WAIT TIME 5
430   G%=GETC:IF G%=32 THEN 470
440   WAIT TIME 5:G%=GETC:IF G%=32 THEN 470
450   WAIT TIME 5:G%=GETC:IF G%=32 THEN 470
460   GOTO 420
470   CURSOR 2,2:PRINT "OM DE KLEUREN TE DOEN WISSELEN,DRUK OP DE SPATIEBALK  ";
480   G%=GETC:WAIT TIME 3:IF G%<>32 THEN 480
490   GOTO 410
1000  REM *** SNOOPY
1010  DATA .                XXXX
1020  DATA .               X----XX
1030  DATA .              X-XXXXX-X                XXXXX
1040  DATA .             X-XXXXXXX-X             XXX-----XX
1050  DATA .          XXXX-XXXXXXX-XXX       XXXX----------XX
1060  DATA .        XX:::X-XXXXXXX-XXXXXXXXX---------------XX XXX
1070  DATA .      XX::::::X-XXXXX-X---------------------------XXX-X
1080  DATA .     X:::::::::XX----XX-----XX---------------------XXX-X
1090  DATA .    X::::::::::XXXXXX-------XX--------------------XXXXX
1100  DATA .   X::::::::::XX:::X-----------------------------XX
1110  DATA . X::::::::::XX::::X----------XXXXXXXXXXXXXXXXXX
1120  DATA . X:::::XXXXX::::X----------X
1130  DATA . X::::X---X:::::X---------X
1140  DATA . X::::X-XX-X::::X--------X
1150  DATA . X::::X-XX-X::::X--------X                  XXXX
1160  DATA .  X:::X-XX-X::::X--------X                XX----XX
1170  DATA .  X::::X---X::::X---------X              X--X-X---X
1180  DATA .   X::::XXX::::::X---------X             XX--X--XXXX
1190  DATA .    X:::::::::::::X---------XXXXXXXX      XX---XX--X
1200  DATA .     XX:::::::::::XX-------------X       X----X--XX
1210  DATA .       XX:::::::::::XXXX---XXXXXX      X-----XXX
1220  DATA .         XXX:::::::::::::XXXXX         X-----X
1230  DATA .            XXXXXXXXXXXX X:::X       X-----X
1240  DATA .                         X::::XX     X-----X
1250  DATA .                        X::::X--XXX X-----X
1260  DATA .                        X::::X-----XXX---X
1270  DATA .                       X:::X----------XXX
1280  DATA .                       X:::X--X----------XXX
1290  DATA .                      X:::X----X------------XX
1300  DATA .                      X:::XX----X-------------X
1310  DATA .                     X:::X--X----X-------------X
1320  DATA .                     X::XX---X----X-------------X
1330  DATA .                    X::X X----X---XXXX----------X
1340  DATA .                    X::X X----XX-----X----------X
1350  DATA .                   X:::X X---X-----XX----------X
1360  DATA .                   X:::X  X---XXX---X---------X
1370  DATA .                  X:::X    XX---XXXX-------XXX
1380  DATA .                 X::::X     XXXX-------X-----X
1390  DATA .                X:::::X         X------X-----X
1400  DATA .  XXXXXXXXXXXXXX:::::X          X------X------XXXXXXXXX
1410  DATA .         X:::::::::::X     XXXXXX-------XXXXXXXXX------X
1420  DATA .    XXXXX:::::::::::X    X---------------------X--X--X
1430  DATA .      X:::::::::::::X    X-----------------X--X--XXXXXX
1440  DATA .XXXXXXXXXXXXXXXXXXX      XXXXXXXXXXXXXXXXXXXXXXX
1450  REM ---------------------------- EINDE ----------------------
```

```
REM *** SNOOPY ************************************
REM *** UITGEZONDEN IN NOS-BASICODE 21-3-1982 ****
REM *** GEWIJZIGD EN AANGEPAST DOOR : ************
REM ******************** DE BONT CORNEEL *********
REM ******************** OOSTEINDE 10 ************
REM *************** 2338 BAARLE - HERTOG *********
REM ************************************************
```

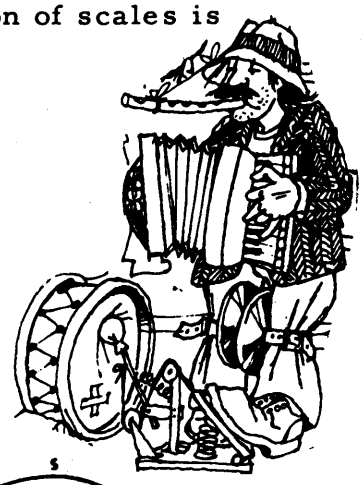
change 16 color mode in 4 color mode

```
1     REM ##################################################
2     REM ##### CHANGE 16 COLOR MODE IN 4 COLOR MODE #####
3     REM ##### PROGRAM  BY  W.De Winter  82-04-27   #####
4     REM ##################################################
5     CLEAR 100:DIM TABLE%(15):GOTO 60
9     REM CONSTRUEER 4 COLOR DATA WOORDEN (SUBROUTINE)
10    ON COLORF% GOTO 30,40,50:RETURN
20    ON COLORB% GOTO 30,40,50:RETURN
30    LOWBYTE%=LOWBYTE% IOR (1 SHL BITPOSITION%):RETURN
40    HIGHBYTE%=HIGHBYTE% IOR (1 SHL BITPOSITION%):RETURN
50    HIGHBYTE%=HIGHBYTE% IOR (1 SHL BITPOSITION%):LOWBYTE%=LOWBYTE% IOR (1 SHL
BITPOSITION%):RETURN
55    REM TITEL + CONTROLE OP 16 COLOR MODE
60    PRINT CHR$(12):PRINT :PRINT " THIS PROGRAM CHANGES ANY 16 COLORMODE IN AN 4
 COLORMODE"
62    IF GETC<>32 THEN 62
65    REM CONTROLEER OF 16 KLEUREN MODE GEZET IS
70    SMODE%=PEEK(#9D):IF SMODE%=#FF OR SMODE%=2 OR SMODE%=3 OR SMODE%=6 OR SMOD
E%=7 OR SMODE%=#A OR SMODE%=#B THEN PRINT :PRINT " YOU ARE IN NO 16 COLOR MODE":
PRINT :END
75    REM ONDERZOEK WELK CONTROLE WOORD GEBRUIKT MOET WORDEN
80    IF SMODE%=1 THEN MBYTE%=3
82    IF SMODE%=5 THEN MBYTE%=#11
84    IF SMODE%=9 THEN MBYTE%=#20
90    REM VUL TAFEL OP DIE HET VERBAND BEPAALD TUSSEN DE KLEUR IN 16 KLEUREN MOD
E EN DE VIER KLEURENREGISTERS IN 4 KLEUREN MODE
95    COUNTER%=-1
100   PRINT :COUNTER%=COUNTER%+1:IF COUNTER%>15 THEN 140
110   PRINT " COLOR ";COUNTER%;" WORDT COLORREGISTER ";:INPUT A%
120   IF A%<4 THEN TABLE%(COUNTER%)=A%:GOTO 100
130   COUNTER%=COUNTER%-1:GOTO 100
135   REM SELECTEER JUISTE MODE
140   IF SMODE%=1 THEN MODE 1
142   IF SMODE%=5 THEN MODE 3
144   IF SMODE%=9 THEN MODE 5
145   REM VERANDER DE CONTROL WOORDEN VAN ALLE LIJNEN IN DE WAARDE VOOR 4 COLOR
MODE
150   FOR ADRESS%=#BFEF TO #BFEF-(XMAX/4+7)*(YMAX+1) STEP -(XMAX/4+7):POKE ADRES
S%,MBYTE%:NEXT
390   REM CONTROLEER DE DATA WOORDEN OP VOOR- EN ACHTERGROND KLEUR
391   REM IN REGEL 410 WORDT HET KLEUR REGISTER VOOR DE VOORGROND KLEUR BEPAALD
392   REM IDEM VOOR REGEL 420 MAAR DAN VOOR DE ACHTER GROND KLEUR
393   REM IN REGEL 430 WORDT DE BYTE DIE DE EIGENLIJKE DOTS BEVAT OPGEROEPEN
394   REM REGEL 440 BEPAALD WELKE BIT WE GAAN TESTEN EN AAN DE HAND VAN DEZE BIT

395   REM DIE OFWEL VOOR- DAN ACHTERGROND BEPAALD GAAN WE TWEE NIUWE DATA WOORDE
N SAMEN STELLEN
396   REM DIE DUS DE TEKENING VERANDEREN IN EEN 4 KLEUREN MODE
400   COUNTER%=0:OFFSET%=(XMAX/4+7)/2:FOR ADRESS%=#BFEF-2 TO #BFEF-(XMAX/4+7)*(Y
MAX+1) STEP -2:COUNTER%=COUNTER%+1:IF COUNTER%=OFFSET% THEN COUNTER%=0:NEXT:GOTO
 500
410   COLORF%=TABLE%((PEEK(ADRESS%-1) IAND #F0) SHR 4)
420   COLORB%=TABLE%(PEEK(ADRESS%-1) IAND #F)
430   SCREENBYTE%=PEEK(ADRESS%):HIGHBYTE%=0:LOWBYTE%=0
440   FOR BITPOSITION%=0 TO 7
450   IF SCREENBYTE% IAND (1 SHL BITPOSITION%)>0 THEN GOSUB 10:GOTO 470
460   GOSUB 20
470   NEXT
480   POKE ADRESS%,HIGHBYTE%:POKE ADRESS%-1,LOWBYTE%
490   NEXT
500   IF SMODE%=1 THEN POKE #9D,2
505   IF SMODE%=5 THEN POKE #9D,6
510   IF SMODE%=9 THEN POKE #9D,#A
520   PRINT CHR$(12):PRINT :PRINT " YOU ARE NOW IN 4 COLOR MODE  !!!!":END
```

2.15.3

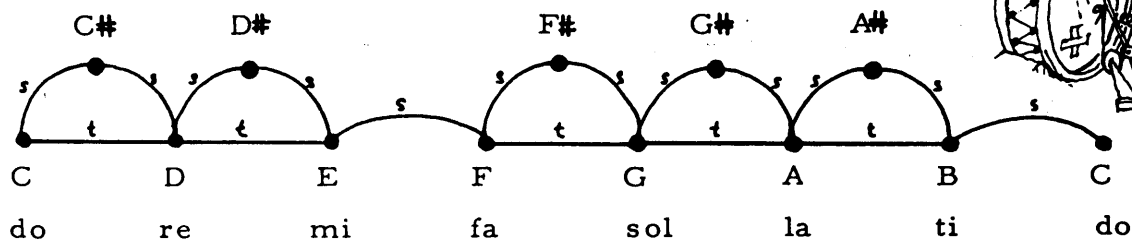
Generation of Music

This section describes how the DAI Graphical Sound Generator may be used to produce musical tones. The mathematical relationship between notes is described for the major scale. A procedure for the generation of scales is outlined.



2.15.3.1

Notes and the Major Scale



The above diagram shows the composition of the Major scale. The natural notes (e.g. C, D, E etc.) represent the white keys on a piano keyboard whilst the sharps (#) represent the black keys.

From C to C, over the scale, represents one OCTAVE. An OCTAVE represents a 2 : 1 frequency ratio. Thus to move from from middle C up 3 OCTAVEs requires an increase in frequency of 8 times (i.e. $2 * 2 * 2 = 2 \uparrow 3$).

To increase the pitch from C to C# requires that the pitch be increased by one SEMITONE. One SEMITONE represents an increase in frequency in the ratio $\sqrt[12]{2} : 1$.

Thus freq (C#) = freq (C) $* \sqrt[12]{2}$

To increase the pitch from C to D requires that the pitch be increased by one TONE. One TONE represents an increase of frequency in the ratio $\sqrt[6]{2} : 1$.

Thus freq (D) = freq (C) $* \sqrt[6]{2}$

It should be noted that

1 TONE = 2 SEMITONES

($\sqrt[6]{2} = \sqrt[12]{2} * \sqrt[12]{2}$)

more about talk & music

2.15.3.2

Automatic generation of scales

It is possible to generate the notes in a scale or a number of scales by increasing or decreasing the frequency by the desired number of SEMITONES. Thus to move from C to F requires an increase of

2 TONES + 1 SEMITONE = 5 SEMITONES

Thus freq (F) = freq (C) $*\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2}$

$= \text{freq (C)} * (\sqrt[12]{2})^5$

A convenient reference note to take when generating scales is A = 44Ø Hz. By moving down the scale 9 SEMITONES C can be generated. Generation of a scale then merely involves a move of an integer number of SEMITONES up from C.

As an example consider a program to generate the irregular do, re, mi sequence:

```
1ØØ   CLEAR 1ØØØØ : SOUND OFF
11Ø   SEMI = 2.Ø ↑ (1.Ø/12.Ø)
12Ø   LOWC = 44Ø/(SEMI ↑ 9) : REM C 9 SEMITONES DOWN
13Ø   ENVELOPE Ø 16
14Ø   FOR N% = 1 TO 8
15Ø   READ D%
16Ø   FRQ = LOWC *(SEMI ↑ D%) : REM D% SEMITONES UP FROM C
17Ø   SOUND Ø Ø 15 Ø FREQ (FRQ)
18Ø   WAIT TIME 2Ø : SOUND OFF
19Ø   NEXT N%
2ØØ   STOP
21Ø   DATA Ø,2,4,5,7,9,11,12
```

Volume control of a particular sound generator is attained by using the appropriate TALK code followed by one byte giving the volume level between Ø → #F (i.e. 16 level).

Frequency control of a particular sound generator is given by the two bytes of data following the frequency TALK code. The frequency is obtained by specifying the number of 1/2 µS units in the same manner as for the SOUND command.

e.g. Determine the hexadecimal number required to give a frequency of 8ØØHz

?HEX$(FREQ(8ØØ)) gives 9C4.

more about talk & music

Delay of up to 0.9 seconds is possible. The unit delay is 13.5 μS. The two bytes of data specify the number of unit delays required.

As TALK codes are interpretive codes relating to sound generation, they are much more restrictive than machine code. Should more sophisticated functions be needed, TALK has the facility to call a subroutine written in absolute machine code. Thus, facilities such as loop counts, etc. become possible with TALK. Two bytes of data specify the 16 bit destination address.

Note: The TALK interpreter, in order to execute TALK code statements quickly, uses very low level coding. One feature of the 8080 microprocessor is the way in which it collects double byte data from memory, collecting the lower value byte first and the higher value byte last.

This convention is observed for all double byte data in the TALK code format. Thus the 3 byte TALK command to call the subroutine at location #5ØBC would be #ØD (call routine), #BC (lower byte address), #5Ø (upper byte address).

Consider the following example of a TALK program:

Address	TALK code	DATA	FUNCTION
#AØØØ	Ø	#C4,9	Set channel Ø frequency to 8ØØHz (#9C4).
AØØ3	8	#F	Set channel Ø volume maximum.
AØØ5	# C	#FF, #FF	Set maximum delay (.88 secs).
AØØ8	8	Ø	Set channel Ø volume minimum.
AØØA	2	#A, #1A	Set channel 1 frequency to 3ØØ Hz (#1AØA)
AØØD	9	#F	Set channel 1 volume maximum.
AØØF	# C	#FF, #FF	Set maximum delay (.88 secs).
AØ12	9	Ø	Set channel 1 volume minimum.
AØ14	4	#2Ø, #4E	Set channel 2 frequency to 1ØØHz (#4E2Ø).
AØ17	#A	#F	Set channel 2 volume maximum.
AØ19	#C	#FF, #FF	Set maximum delay (.88 secs).
AØ1C	#A	Ø	Set channel 2 volume minimum.
AØ1E	#ØD	#21, #AØ	Jump to machine code subroutine (#AØ21)

For the purposes of this example we will include a small machine code subroutine which will return to the TALK interpreter, causing it to start executing TALK instructions from address #AØØØ.

Address	Code	
#AØ21	#21, #ØØ, #AØ	LXI,H with AØØØ
AØ24	#C9	Return

The H,L register pair is loaded with the address of the next instruction for the TALK interpreter.

The following program will run the TALK statements given.

```
1Ø      CLEAR 1ØØØ:B%= #AØØØ
2Ø      READ A%:IF A% < Ø GOTO 4Ø
3Ø      POKE B%,A%:B%=B%+1:GOTO 2Ø
4Ø      TALK #AØØØ
5Ø      STOP
1ØØ     DATA Ø, #C4,9,8, #F, #C, #FF, #FF,8,Ø
11Ø     DATA 2, #A, #1A,9, #F, #C, #FF, #FF,9,Ø
12Ø     DATA 4, #2Ø, #4E, #A, #F, #C, #FF, #FF, #A,Ø
13Ø     DATA #ØD, #21, #AØ, #21,Ø, #AØ, #C9,-1
```

The next program will generate the Major scale over 5 OCTAVES. The values may be compared with the table which follows the program.

```
1ØØ     CLEAR 1ØØØ:SOUND OFF
11Ø     SEMI=2.Ø↑(1.Ø/12.Ø)
12Ø     LOWC=44Ø.Ø/(SEMI↑33.Ø)
13Ø     ENVELOPE Ø 15
14Ø     FOR N%=Ø TO 59
15Ø     IF N% MOD 12=Ø THEN RESTORE
16Ø     READ A$:FRQ=LOWC*(SEMI ↑N%)
17Ø     SOUND Ø Ø 15 Ø FREQ(FRQ)
18Ø     PRINT A$, "FREQ=";FRQ
19Ø     WAIT TIME 2Ø:SOUND OFF
2ØØ     NEXT N%
21Ø     STOP
22Ø     DATA DO,DO# ,RE, RE# , MI, FA, FA# , SOL, SOL# , LA, LA# ,TI
```

Note	Freq. Hz Octave 1	Freq. Hz Octave 2	Freq. Hz Octave 3	Freq. Hz Octave 4	Freq. Hz Octave 5
DO	66	131	262	523	1Ø47
DO#	69	139	277	554	11Ø9
RE	73	147	294	587	1175
RE#	78	156	311	622	1245
MI	82	165	33Ø	659	1319
FA	87	175	349	698	1397
FA#	92	185	370	74Ø	148Ø
SOL	98	196	392	784	1568

SOL#	1Ø4	2Ø8	415	831	1661
LA	11Ø	22Ø	44Ø	88Ø	176Ø
LA#	117	233	466	932	1865
TI	123	247	494	988	1976

2.15.4

Synthesising Vocal Sound

Due to the flexibility of change in volume and frequency it is quite feasible to explore the possibilities of vocal sound generation. The BASIC of the DAI Personal Computer gives full control to the programmer who wishes to develop experimentally a burst of sounds that may result in vocal sounds. A further feature aiding experimentation is the TALK facility, which allows for fast bursts of sound to be programmed in a low level code.

2.15.4.1

TALK

EXAMPLES

```
TALK  #BØØØ
TALK   VARPTR(X(Ø))
```

TALK is both a command in DAI BASIC and a subsystem in its own right. The above TALK commands cause the TALK code interpreter to start to interpret the special low level TALK code at the address specified. The following table outlines the TALK codes:

Instruction Code	Bytes of Data	Function
#Ø	2	Frequency control, channel Ø
#2	2	" " " 1
#4	2	" " " 2
#8	1	Volume control, channel Ø
#9	1	" " " 1
#A	1	" " " 2
#B	1	" " , white noise generator
#C	2	Delay in 13 μS units.
#D	2	Call absolute machine code routine.
#FF	Ø	END TALK/RETURN TO BASIC.

An alternative way of achieving the same result but with more safety, is to set up space on the HEAP by dimensioning an array and writing the TALK codes into it. Any addresses which need to be set up can be done with reference to position in the DATA table. The following program demonstrates this possibility.

```
10      CLEAR 1000:DIM Q%(100.0):B%=VARPTR(Q%(0.0))
20      READ A%:IF A%<0.0 GOTO 40
30      POKE B%,A%:B%=B%+1:GOTO 20
40      READ F%:IF F%<0 GOTO 50:READ S%:GOSUB 1000:GOTO 40
50      TALK VARPTR(Q%(0.0))
60      STOP
100     DATA 0,#C4,9,8,#F,#C,#FF,#FF,8,0
110     DATA 2,#A,#1A,9,#F,#C,#FF,#FF,9,0
120     DATA 4,#20,#4E,#A,#F,#C,#FF,#FF,#A,0
130     DATA #0D,0,0,#21,0,0,#C9,-1
140     DATA 32,34,35,1,-1
1000    J%=4:UF%=VARPTR(Q%(F%/J%))+(F% MOD 4.0)-1.0
1010    US%=VARPTR(Q%(S%/J%))+(S% MOD 4.0)-1.0
1020    POKE UF%,US% IAND #FF
1030    POKE UF%+1,US% SHR 8
1040    RETURN
```

floating point numbers

PART 2

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

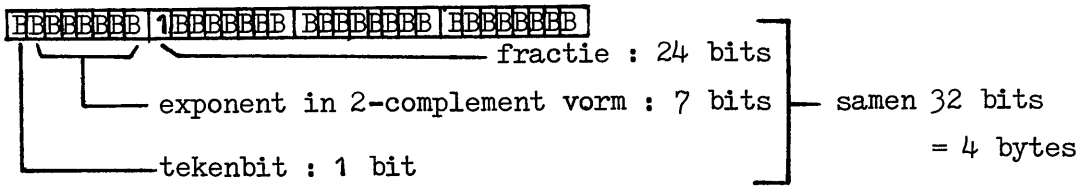
$$2^{63} \approx 9.2 \times 10^{18}$$

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

bingo

```
1     REM BINGOSPEL       PROGR: T.GROENEVELD LEEUWARDEN NEDERLAND
2     REM GEBRUIKSAANWIJZING
3     REM DIT SPEL MAAKT GEBRUIK VAN EEN DRUKKNOP, MET DAARAAN
4     REM PARRALLEL EEN WEERSTAND VAN 47K EN IS AANGESLOTEN OP PDL(2)
5     REM NA GEREEDMELDING OP DRUKKNOP DRUKKEN VOOR EERSTE GETAL.
6     REM VOOR ELK VOLGEND GETAL KORTSTONDIG OP DRUKKNOP DRUKKEN.
7     REM ALS ER BINGO GEROEPEN WORDT, KRUKKNOP NET ZOLANG INDRUKKEN
8     REM TOT HET BINGOVELD VERSCHIJNT MET DE GEWEESTE GETALLEN.
10    MODE 0:COLORT 8 0 8 0:CLEAR 1000:DIM G(76.0):L=0.0:PRINT CHR$(12):POKE #75
,32:COLORT 8 0 8 0
11    CURSOR 10,20:PRINT "##########################################"
12    CURSOR 10,19:PRINT "#                                        #"
13    CURSOR 10,18:PRINT "#          B   I   N   G   O             #"
14    CURSOR 10,17:PRINT "#          ================              #"
15    CURSOR 10,16:PRINT "#                                        #"
16    CURSOR 10,15:PRINT "##########################################"
17    CURSOR 10,14:PRINT "uitgevoerd met een == D A I computer =="
18    CURSOR 10,10:PRINT "DE COMPUTER IS NU BEZIG EEN WILLEKEURIG"
19    CURSOR 10,9:PRINT "--------------------------------------"
20    CURSOR 12,7:PRINT "BINGO GETALLEN BESTAND OP TE BOUWEN"
22    CURSOR 12,6:PRINT "----------------------------------"
23    REM OPBOUWEN GETALLENBESTAND
25    R=RND(75.0)+1.0
26    ENVELOPE 1 15,9;0:SOUND 1 1 15 0 FREQ(RND(269.0)+31.0)
27    NOISE 1 15
30    FOR A=1.0 TO 75.0:IF G(A)<>R THEN NEXT A
40    IF A<76.0 AND G(A)=R THEN 25
50    IF G(I)=0.0 THEN G(I)=R
65    I=I+1.0:IF I=76.0 THEN 100
70    GOTO 25
80    REM GEREEDMELDING VAN GETALLENBESTAND
100   PRINT CHR$(12):COLORT 5 15 5 5:POKE #75,32:POKE #BA2D,95
110   SOUND OFF :CURSOR 0,12:PRINT "BINGO IS GEREED"
115   REM WACHTEN OP TOETSENBORD
120   REM R%=GETC:IF R%=0 THEN 120
130   REM WACHTEN OP PADDLE
140   P!=PDL(2):IF PDL(2)>10 THEN 140
160   L=0:COLORT 8 0 8 0:POKE #BA2D,122
162   REM GETALLENBESTAND EEN VOOR EEN LEEGLEZEN
163   MODE 1:COLORG 1 10 0 0:PRINT CHR$(12)
165   L=L+1.0:IF L=76 THEN 185
180   PRINT "            Getal is ==";G(L);" == Is er BINGO (J/N) ?"
181   FILL 5,5 65,55 2:GOSUB 300
182   SOUND 0 0 15 0 FREQ(800.0):WAIT TIME 5:SOUND OFF
184   REM WACHTEN MET GETC OP TOETSENBORD
185   REM G=GETC:IF G=0.0 THEN 185
186   REM IF G=74.0 THEN 194
188   REM WACHTEN MET GETC OP PADDLE
189   IF PDL(2)>10 THEN 189
190   WAIT TIME 100
191   IF PDL(2)<10 THEN 194
193   GOTO 165
194   REM GEGEVEN BINGOGETALLEN WEERGEVEN
195   MODE 0:PRINT CHR$(12)
196   IF L=76.0 THEN L=75
```

```
197   PRINT "     ==========================================================="
198   PRINT "     # # # #    B    #    I    #    N    #    G    #    O  # # # #"
199   PRINT "     ==========================================================="
200   FOR T=1 TO L
205   P=INT(G(T)-1.0)/15.0
210   CURSOR P*8.0+12,20-G(T)+P*15.0
230   PRINT G(T):NEXT T
240   CURSOR 5,4:PRINT "============================================================"
250   CURSOR 5,2:PRINT "GOEDE BINGO (J/N) ?"
254   CURSOR 23,2
255   R=GETC:IF R=0 THEN 255
260   IF R=74 THEN 600
262   FOR W!=500.0 TO 40.0 STEP -1.0
264   SOUND 0 0 15 2 FREQ(W!):NEXT:SOUND OFF
265   CURSOR 5,1:PRINT "VOOR VOLGENDE GETALLEN OP SPATIEBALK DRUKKEN !!!"
266   G!=GETC:IF G!=0.0 THEN 266
267   CURSOR 5,1:PRINT "                                                    ":GOTO 1
63
300   N=30:X=INT(G(L)/10.0)
310   IF X=0.0 THEN 330
320   GOTO 470
325   REM BINGOGETALLEN GROOT WEERGEVEN
330   A=G(L)
340   ON A+1 GOTO 350,360,370,380,390,400,410,420,430,440
350   GOSUB 500:GOSUB 510:GOSUB 520:GOSUB 530:GOSUB 540:GOSUB 550:GOTO 450
360   GOSUB 540:GOSUB 550:GOTO 450
370   GOSUB 530:GOSUB 540:GOSUB 560:GOSUB 510:GOSUB 500:GOTO 450
380   GOSUB 530:GOSUB 540:GOSUB 550:GOSUB 500:GOSUB 560:GOTO 450
390   GOSUB 520:GOSUB 560:GOSUB 540:GOSUB 550:GOTO 450
400   GOSUB 530:GOSUB 520:GOSUB 560:GOSUB 550:GOSUB 500:GOTO 450
410   GOSUB 530:GOSUB 520:GOSUB 510:GOSUB 500:GOSUB 550:GOSUB 560:GOTO 450
420   GOSUB 530:GOSUB 540:GOSUB 550:GOTO 450
430   GOSUB 530:GOSUB 520:GOSUB 510:GOSUB 500:GOSUB 550:GOSUB 540:GOSUB 560:GOTO
 450
440   GOSUB 530:GOSUB 520:GOSUB 560:GOSUB 540:GOSUB 550:GOSUB 500:GOTO 450
450   IF X=0.0 THEN RETURN
460   A=FRAC(G(L)/10.0)*10.0+0.5:X=0:N=30:GOTO 340
470   A=X:N=0:GOTO 340
500   DRAW 10+N,10 30+N,10 8:RETURN
510   DRAW 10+N,10 10+N,30 8:RETURN
520   DRAW 10+N,30 10+N,50 8:RETURN
530   DRAW 10+N,50 30+N,50 8:RETURN
540   DRAW 30+N,50 30+N,30 8:RETURN
550   DRAW 30+N,30 30+N,10 8:RETURN
560   DRAW 30+N,30 10+N,30 8:RETURN
600   ENVELOPE 0 15
602   FOR J!=1.0 TO 15.0
605   SOUND 1 0 15 2 FREQ(800.0):WAIT TIME 5
610   SOUND 1 0 15 2 FREQ(600.0):WAIT TIME 5
615   NEXT:SOUND OFF
620   CURSOR 5,1:PRINT "WILT U NOG EEN KEER SPELEN (J/N) ?"
625   CURSOR 38,1
630   G!=GETC:WAIT TIME 3:IF G!=0.0 THEN 630
640   IF G!=78.0 THEN 660
650   GOTO 10
660   POKE #75,95:PRINT :END
```

grue + chrono

```
1     REM
2     REM GRUE + CHRONO - D'APRES DAINAMIC NO 8 .
3     REM ---------------------------------------
4     REM ECRIT PAR MOENS JACQUES - NOVEMBRE 1981
5     REM ---------------------------------------
10    MODE 0:PRINT CHR$(12):COLORT 12 0 12 12:PRINT :PRINT :PRINT
11    PRINT TAB(20);"GRUE + CHRONO":PRINT TAB(20);"-------------":PRINT
12    PRINT TAB(10);"LE BUT DU JEU EST DE RAMENER LE COLIS ":PRINT TAB(10);"SUR
LE QUAI SITUE A GAUCHE DE L'ECRAN"
13    PRINT TAB(9);"EN UN MINIMUM DE MOUVEMENTS ET DE TEMPS.":PRINT :PRINT TAB(1
5);"LES MOUVEMENTS SONT :"
14    PRINT TAB(15);CHR$(94);"  ";CHR$(140);"  ";CHR$(136);"  ";CHR$(137)
15    PRINT TAB(15);"1 = SERRAGE DES PINCES"
16    PRINT TAB(15);"2 = ECARTEMENT DES PINCES"
17    PRINT TAB(7);"POUR ACCELERER APPUYEZ SUR LA TOUCHE 'REPT'"
18    PRINT :PRINT :PRINT TAB(10);"POUR COMMENCER APPUYEZ SUR UNE TOUCHE"
19    IF GETC=0.0 THEN 19
20    GOTO 140
30    YM1%=YMAX-1:YP1%=YMAX-Y1%+1:DRAW X1%,YM1% X1%,YP1% C%
35    XM1%=X1%-D1%:YM1%=YMAX-Y1%-1:YM5%=YMAX-Y1%-5
40    DRAW XM1%,YM1% XM1%,YM5% C%
50    XP1%=X1%+D1%:DRAW XP1%,YM1% XP1%,YM5% C%
60    YM1%=YMAX-Y1%:DRAW X1%-10,YM1% X1%+10,YM1% C%
61    IF AC=0 THEN RETURN
62    FILL XG1,YG1 XD1,YD1 D%
70    RETURN
140   MODE 2:COLORG 0 1 10 5
150   DRAW 10,YMAX XMAX-10,YMAX 21
151   FILL 50,0 60,10 10
152   DRAW 0,30 29,30 5:DRAW 29,30 29,0 5
160   C%=21:X1%=11:Y1%=5:D1%=1:AC=0.0:PO=0.0
161   XG1=50.0:YG1=0.0:XD1=60.0:YD1=10.0:GOSUB 30
170   A%=GETC:S%=S%+1:IF A%=0 THEN 170
180   S%=S%+10:M%=M%+1:X2%=X1%:Y2%=Y1%:D2%=D1%
181   XG2=XG1:XD2=XD1:YG2=YG1:YD2=YD1
182   IF AC=1.0 THEN 190
186   IF SCRN(XG1-1,YD1)=0 THEN 190
187   IF SCRN(XD1+1,YD1)=0 THEN 190
188   IF SCRN(XG1+6,YD1+2)=0 THEN 190
189   AC=1.0
190   IF SCRN(10,31)<>10 AND SCRN(19,31)<>10 THEN 200
195   PO=1.0:GOTO 290
200   IF A%=50.0 THEN D2%=D1%+1:IF D2%<11 THEN 270
205   IF A%=49 THEN D2%=D1%-1:IF D2%>0 THEN 270
210   IF A%<>16 THEN 220
211   Y2%=Y1%-1
212   IF AC=1 THEN YG2=YG1+1.0:YD2=YD1+1.0
213   IF Y2%>1.0 THEN 260
220   IF A%<>17.0 THEN 230
221   Y2%=Y1%+1:Y5%=YMAX-Y2%-5
222   IF AC=0 THEN 226
225   YG2=YG1-1.0:YD2=YD1-1.0:IF YG2<=0.0 THEN 250
226   IF SCRN(XD2,YD2+1)=1 THEN 250
228   IF (Y2%<59.0) AND (SCRN(X1%+D1%,Y5%)=0) AND (SCRN(X1%-D1%,Y5%)=0) THEN 260
230   IF A%<>18.0 THEN 240
231   X2%=X1%-1
232   IF AC=1 THEN XG2=XG1-1.0:XD2=XD1-1.0
233   IF X2%>10.0 THEN 260
240   IF A%<>19.0 THEN 250
241   X2%=X1%+1
242   IF AC=1 THEN XG2=XG1+1.0:XD2=XD1+1.0
243   IF X2%<XMAX-10.0 THEN 260
250   SOUND 1 0 15 0 FREQ(100.0):WAIT TIME 10:SOUND OFF :GOTO 170
260   IF X2%<31.0 AND Y2%>28.0 THEN 250
261   IF X2%<40.0 AND Y2%>33.0 THEN 250
262   IF AC=0.0 THEN 265
263   IF SCRN(XG2-1,YG2)=5 THEN 250
```

240 × 528 resolution

```
001                 ****************************************************
002                 * CE PROGRAMME PERMET DE METTRE L'ECRAN EN MODE 8
003                 * C'EST A DIRE EN 240 X 528 4 COULEURS.
004                 * IL EST DONNE AVEC SA REPLIQUE 'BASIC'
005                 * POUR CEUX QUI NE SONT PAS FAMILIARISE
006                 * AVEC LE LANGUAGE MACHINE.
007                 * POUR LANCER LE PROGRAMME IL SUFFIT DE FAIRE:
008                 * MODE 6:CALLM#300
009                 ****************************************************
010                 START   ORG   :300
011                 * LANGAGE MACHINE        LE MEME EN BASIC
012                 * --------------         ---------------
013                 * MODE6:CALLM#300        * 1   MODE6
014 0300 F5                 PUSH  PSW        * 10  PPSW%=PSW%
015 0301 C5                 PUSH  B          * 20  PB%=BC%
016 0302 D5                 PUSH  D          * 30  PD%=DE%
017 0303 E5                 PUSH  H          * 40  PH%=HL%
018 0304 21EFBF             LXI   H,:BFEF    * 50  HL%=#BFEF
019 0307 1EF0               MVI   E,240      * 60  E%=240
020 0309 1684       LOOP0   MVI   D,:84      * 70  D%=#84
021 030B 3630               MVI   M,:30      * 80  POKE HL%,#30
022 030D 2B                 DCX   H          * 90  HL%=HL%-1
023 030E 3640               MVI   M,:40      * 100 POKE HL%,#40
024 0310 2B         LOOP1   DCX   H          * 110 HL%=HL%-1
025 0311 3600               MVI   M,:00      * 120 POKE HL%,#00
026 0313 15                 DCR   D          * 130 D%=D%-1
027 0314 C21003             JNZ   LOOP1      * 140 IF D%<>0 THEN 110
028 0317 2B                 DCX   H          * 150 HL%=HL%-1
029 0318 1D                 DCR   E          * 160 E%=E%-1
030 0319 C20903             JNZ   LOOP0      * 170 IF E%<>0 THEN 70
031 031C E1                 POP   H          * 180 HL%=PH%
032 031D D1                 POP   D          * 190 DE%=PD%
033 031E C1                 POP   B          * 200 BC%=PB%
034 031F F1                 POP   PSW        * 210 PSW%=PPSW%
035 0320 C9                 RET              * 220 RETURN
036 0321            FIN     END

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

FIN    0321   LOOP0  0309   LOOP1  0310   START  0000
```

grue + chrono

```
265   C%=20:D%=20:GOSUB 30:C%=21:D%=10:X1%=X2%:Y1%=Y2%:D1%=D2%
267   IF AC=1 THEN XG1=XG2:XD1=XD2:YG1=YG2:YD1=YD2
268   GOSUB 30:IF P=1.0 THEN 290
269   GOTO 170
270   IF (SCRN(D2%+X1%,YM5%)=0) AND (SCRN(X1%-D2%,YM5%)=0) THEN 262
271   GOTO 250
290   PRINT "TERMINE EN";M%;" MOUVEMENTS ET";S%/50.0;" SEC"
291   INPUT "UN NOUVEL ESSAI (O/N) ";R$
292   IF R$="O" THEN MODE 0:PRINT CHR$(12):M%=0:S%=0:GOTO 20
293   IF R$="N" THEN 999
294   GOTO 291
999   PRINT :PRINT "AU REVOIR ET A BIENTOT J'ESPERE":PRINT :END
1000  REM PROGRAMME ECRIT PAR JACQUES MOENS
1010  REM ..................... CLOS FONTAINE DES DUCS,6
1020  REM ..................... B - 1310  LA HULPE
1030  REM ..................... TEL. 02/657.95.60.
```

```
##############################
#DAI TINY PASCAL#
#     COMPILER     #
##############################
```

PASCAL

1. INTRODUCTION

On tape following software modules are at your disposal:

A. PASCAL SYSTEM V3.1 consisting of:
- # an editor to generate "source code"
- # the Tiny Pascal compiler which translates the sourcecode into "P-code"
- # the 8080 translator which converts P-code into 8080 statements
- # the runtime modules

B. The Pascal Library which contains a set of procedures that can be merged with the PASCAL programs, enabling to make use of DAI's grafic features. If desired you can enlarge the contents of this liberary by yourself.

C. Convertor, which is a program that enables Pascal V1.X users to transform programs ,made with DAI's BASIC editor int the V3.1 compatible format.

D. A set of examples

It is important to remark that the "runtime modules" are not saved as a separate file on the tape. However, such a file is important when the object code of a Pascal program is directly loaded into memory.
To get the separate "runtime module" perform:

```
# UT
> R cr
> Z3 cr
> W2EC 9C4 RUNTIME-MODULES cr
```

Object files normally always start at address #9C5. To start an object file perform:

```
# UT
> Z3
> R RUNTIME-MODULES
> R OBJECTFILE
> G9C5
```

2. MEMORY MAP

adddress		
2EC .. 9C4		RUNTIME MODULES
9C5 .. 1FFF		OBJECT CODE
2000 .. 2FFF		EDIT BUFFER (SOURCE CODE)
3000 .. 612A		PASCAL SYSTEM V3.1
612B .. 6FFF		RUNTIME STACK
7000 .. 7FFF		P-CODE BUFFER
8000 .. B34F		FREE

3. HOW GETTING STARTED.

Following actions have to be peformed:

```
#UT
>Z3
>R PASCAL SYSTEM V3.1
>G3000
```

Now a menu appears on the screen with follwing options

<E> :enables you to create the PASCAL Source File. The normal DAI editing features are applicable. To return from edit mode, just type <break>.
<D> :Deletes the contents of the editbuffer. To avoid undesired loose of a sourcefile, before typing 'D', '/' has to be entered first.
<L> :Loads the editbuffer with a source file from tape
<M> :Adds a "source file" to the contents of the edit buffer. This option must be selected when the already mentioned PASCAL LIBRARY has to be linked with a created source file. The position of the cursor determines the insertion of the file.
<S> :Saves the contents of the editbuffer on tape.
<C> :Compilation - the source file in the editbuffer is converted into p-code. When a syntax error has beeen detected, compilation is halted. On hitting the spacebar a return to the primery menu is accomplished. Possible errors can now be corrected. The compiler provides an option that when selected displays the mnemonics of the generated P-code statements.
<T> :Translation of P-code into 8080 code. In principle, the translator polls for a set addresses. To avoid problems only respond by <return>. Note begin and end addresses mentioned by the translator
<R> :loads object file from tape
<W> :writes bject file to tape
<G> :start execution of the object code. When no specific address is filled in , execution will start from #9C55 When the prgram finishes a returnal to utility mode is performed. Just accomplish G<return> to return to to the primery menu.
<U> :return to utility.
<B> :return to basic.
<P> :-not displayed on the primery option menu. The listing of the source file is displayed on the screen and printed on hardcopy equipment.

4. TINY PASCAL STATEMENTS

Following keywords will be understood by the compiler

AND	END	PROC (:procedure)
ARRAY	FOR	READ
BEGIN	FUNC(:function)	REPEAT
CALL	IF	SHR (:shift right)
CASE	INTEGER	SHL (:shift left)
CONST	MEM	THEN
DIV	MOD	TO
DO	NOT	UNTIL
DOWNTO	OF	VAR
ELSE	OR	WHILE
		WRITE

Before starting creating Pascal programs, first read both appendixes.
.APPENDIX I gives the Tiny Pascal Syntax diagrams
.APPENDIX II gives a lot of examples on all thes keywords

PASCAL

```
  ###############
! #DECLARATIONS#
  ###############

  CONSTANTS (SINGLE BYTE) ARE PERMITTED
!
 CONST LF=13;FF=12;


!
  TINY PASCAL ONLY UNDERSTANDS INTEGERS
  BY THIS  A VARIABLE CALLED "TEST" ALWAYS
  MUST BE DECLARED AS
!
  VAR TEST1,TEST2 : INTEGER;

! WHILE AN ARRAY OF ELEMENTS "VECTOR" AS
!
 VECTOR :ARRAY[300] OF INTEGER;
! REMARK THAT
 . THIS COMPILER ONLY UNDERSTANDS
   SINGLE DIMENTION DECLARATIONS
 . DECLARATIONS MUST BE PERFORMED IN FOLLOWING
   SEQUENCE
    a. CONSTANTS
    b. VARIABLES
    c. ARRAYS

!
  ###############
  #INPUT^OUTPUT#
  ###############

  A VARIABLE MAY BE PRESENTED IN
   -INTEGER  FORMAT      :  <VARIABLE>%
   -HEX INTEGER FORMAT   :  <VARIABLE>#
   -ASCI                 :  <VARIABLE>


! VAR A IS SET TO 13,
 THIS PROGRAM RESPECTIVELY WRITES
  . HEX VALUE OF A (4 DIGITS)
  . DEC VALUE OF A
  . ASCII VALUE OF A (CR^LF)
!
  VAR A,B :INTEGER;
  BEGIN
   A:=13;
   WRITE('   ',A#,A,'   ',A%,A);
   WRITE('   ','that is all folks')
  END. ! '.' MARKS END OF PROOGRAM
```

```
! THIS PROGRAM DEMONSTATES
  THE USAGE OF THE 'READ COMMAND'
  THE PROGRAM ASKS FOR FOUR DECIMAL
  INPUTS AND DISPLAYS THE HEX REPRE-
  SENTATION
!
VAR I,A :INTEGER;
BEGIN
 I:= 0;
 REPEAT
    WRITE ('      ');
    READ (A%);! INTEGER INPUT!
    WRITE('       ',A#,13);!WRITE IN HEX FOMAT!
    I:=I+1
 UNTIL I=4
END.



 ! FOR ... DO EXAMPLE
   100 FUNNY A'S ARE GOING TO BE
   DISPLAYED ON THE SCREEN
   .. IN STEAT OF A SINGLE STATEMENT, ALSO
      A COMPOUND STATEMENT MAY BE USED
 !
 CONST FUNNYA=#40;
 VAR I :INTEGER;
 BEGIN
   FOR I:=0 TO 99 DO  !<-- NO  SEMI-COLOMN !
       WRITE(FUNNYA);
 END.
!
 ########################
 # REPETITIVE STATEMENTS#
 ########################

 FOR ... DO
 REPEAT .. UNTIL
 WHILE ..  DO

 FOR ... DO EXAMPLE
   100 FUNNY A'S ARE GOING TO BE
   DISPLAYED ON THE SCREEN
   .. IN STEAT OF A SINGLE STATEMENT, ALSO
      A COMPOUND STATEMENT MAY BE USED
 !
 CONST FUNNYA=#40;
 VAR I :INTEGER;
 BEGIN
   FOR I:=0 TO 99 DO  !<-- NO  SEMI-COLOMN !
       WRITE(FUNNYA);
 END.
!
  SAME EXAMPLE WITH THE DO..WHILE STATEMENT
!
  CONST FUNNYA=#40;
  VAR I :INTEGER;
  BEGIN
    I:=0;
    WHILE I<>100 DO!AS LONG AS I DIFFERS FROM 100!
    BEGIN
       WRITE(FUNNYA);
       I:=I+1; !INCREMENT I!
    END;
  END.
!
```

```
  SAME EXAMPLE WITH THE REPEAT UNTIL STATEMENT
!
  CONST FUNNYA=$40;
  VAR I :INTEGER;
  BEGIN
    I:=0;
    REPEAT !NO ";"!
        WRITE(FUNNYA);
        I:=I+1; !INCREMENT I!
    UNTIL I=99
  END.



!
  ************************
  * CONDITIONAL STATEMENTS *
  ************************

  THERE EXISTS :
     IF ...THEN (..ELSE)
     THE CASE STATEMENT


  THE IF STATEMENT
!
  VAR A,B :INTEGER;
  BEGIN
   B:=1;
   WHILE B DO  !ALSO MAY BE B=1!
    BEGIN
     READ(A);
     IF A='A' THEN WRITE('   HELLO BUDDY',13)
       ELSE IF A='B' THEN WRITE('   NICE WHETHER',13)
           ELSE IF A='S' THEN
              BEGIN
                 B:=0;
                 WRITE('   THIS IS ALL FOLKS',13)
              END
           ELSE WRITE('   YOU FOOL',13)
   END
  END.



!
  THE SAME FUNCTION PERFORMED BY THE
  CASE STATEMENT
!
  VAR A,B :INTEGER;
  BEGIN
   B:=1;
   WHILE B DO  !ALSO MAY BE B=1!
    BEGIN
     READ(A);
     CASE A OF
     'A' : WRITE('   HELLO BUDDY',13);
     'B' : WRITE('   NICE WHETHER',13);
     'S' : BEGIN
             B:=0;
             WRITE('   THIS IS ALL FOLKS',13)
           END
     ELSE WRITE('   YOU FOOL',13)
   END END
  END.
```

```
!
  ************************
  * PROCEDURES AND FUNCTIONS*
  ************************

  A. PROCEDURES

  FORMAT: PROC <NAME>(a1,a2,..an);
    WHERE a1.. an represent parameters: I/O TOWARDS
    THE PROCEDURE BODY.
    PARAMETERS NEED NOT TO BE DECLARED.
    A PROCEDURE IS CALLED BY ITS NAME.
    PROCEDURES MAY HAVE LOCAL VARIABLES

  FOLLOWING PROGRAM READS A SET OF INTEGERS AND DISPLAYS
  IT IN REVERSED ORDER
!
  VAR I,J :INTEGER;
      A: ARRAY[60] OF INTEGER;
  !PROCEDURE READSTRING!
  PROC READSTRING;
    VAR I :INTEGER;
    BEGIN
     I:=0;J:=0;READ(I%);
     WHILE I<>9999 DO
      BEGIN
        A[J]:=I;J:=J+1;READ(I%)
      END
    END;
  !PROCEDURE REVERSE STRING!
  PROC REVERSESTRING(X);
  VAR I:INTEGER;
  BEGIN
   FOR I:=X-1 DOWNTO 0
    DO WRITE(A[I]%,'  ');
  END;
  ! PROCEDURE NEWLINE!
  PROC NEWLINE;
  BEGIN
   WRITE(13,'          ')
  END;
   ! MAIN!
  BEGIN
   NEWLINE;READSTRING;NEWLINE;REVERSESTRING(J)
  END.
!


   B. FUNCTONS

  FORMAT: FUNC <NAME>(a1,a2,..an);
    WHERE a1.. an represent parameters: I/O TOWARDS
    THE FUNCTION BODY.
    PARAMETERS NEED NOT TO BE DECLARED.
    A PROCEDURE IS ALWAYS CALLED IN AN EXPRESSION I.E.
      A:=<NAME>(a1,...an)
    FUNCTIONS MAY HAVE LOCAL VARIABLES

  FOLLOWING PROGRAM READS A SET OF 4 INTEGERS AND DISPLAYS
  THE GREATEST VALUE
!
```

```
VAR A1,A2,A3,A4,A,GREATEST :INTEGER;
FUNC MAX4(X1,X2,X3,X4);
 FUNC MAX2(X1,X2);
   BEGIN
    IF X1>X2 THEN MAX2:=X1
            ELSE MAX2:=X2
   END;
  BEGIN
  MAX4:=MAX2(MAX2(X1,X2),MAX2(X3,X4))
  END;
  BEGIN
   A:='Y';
   WHILE A='Y' DO
    BEGIN
     WRITE(13,'  A NEW GAME ?  [Y/N]');READ(A);
     WRITE(13,'  =======> input   ');
     IF A='Y' THEN
       BEGIN
        READ(A1%,A2%,A3%,A4%);
        GREATEST:=MAX4(A1,A2,A3,A4);
        WRITE('       ',GREATEST%);
       END
    END
   END.
```

```
**********************
* BOOLEAN EXPRESSIONS*
**********************
```

BOOLEAN EXPRESSIONS MAY BE USED IN
. THE "IF" STATEMENT
. THE "WHILE" STATEMENT
. THE "REPEAT UNTIL" STATEMENT

```
EX. IF A*(B+C)/D > E THEN ...
                    ELSE ...
```

FOLLOWING EXPRESSIONS ARE VALID

<	LESS THEN
>	GREATER THEN
<=	LESS OR EQUAL
>=	GREATER OR EQUAL
<>	DIFFERENT

```
*************************
* ARITHMETIC OPERATORS *
*************************
```

. + ADD	EX. A:=B+C;
. - SUBSTRACT	EX. A:=B-D;
. * MULTIPLY	EX. A:=A*B;
. DIV DIVIDE (INTEGER)	EX. A:=B DIV C
. MOD MODULUS	EX. A:=B MOD 8

FOLLOWING EXAMPLE DEMONSTATES THESE OPERATORS

```
VAR A,B,C :INTEGER;
BEGIN
 WRITE(13,'INPUT B,C :  ');READ(B%,C%);
 WRITE(13,'       B + C = ',(B+C)%,13,
       '       B - C = ',(B-C)%,13,
       '       B x C = ',(B*C)%,13,
       '       B : C = ',(B DIV C)%,13,
       '       B mod C=',(B MOD C)%,13)
END.
```

```
**********************
* BOOLEAN OPERATORS *
**********************
```

THESE ARE:

. OR	EX. A:=B OR C
. AND	EX. A:=B AND C
. NOT	EX. A:= NOT B
. SHL SHIFT LEFT	
. SHR SHIFT RIGHT	EX. A:=B SHR C

```
VAR I,J :INTEGER;
BEGIN
 WRITE(13,'INPUT 2 HEX VARIABLES   ');READ(I#,J#);
 WRITE(13,'    I OR J    = ',(I OR J)#,13,
         '    I AND J   = ',(I AND J)#,13,
         '    NOT I     = ',(NOT I)#,13,
         '    I SHL 2   = ',(I SHL 2)#,13,
         '    J SHR 2   = ',(J SHR 2)#,13)
END.
```

```
***************************
* PEEK AND POOK AND CALLM*
***************************
```

. PEEK : A:=MEM #131;
. POKE : MEM #131:=2;
. CALLM: CALL(#D6BE)

PASCAL

```
! Eratosthenes Sieve Prime Number Program in PASCAL !

VAR  FLAGS : ARRAY[1000] OF INTEGER;
     SIZE,I,PRIME,K,COUNT : INTEGER;

BEGIN
WRITE(12,' FROM 1 TILL  '); READ(SIZE%);
SIZE := SIZE DIV 2; WRITE(13,13,'  2    '); COUNT := 1;
FOR I := 0 TO SIZE DO FLAGS[I] := 1;
FOR I := 0 TO SIZE DO
     IF FLAGS[I] THEN BEGIN
               PRIME := I+I+3;  K := I + PRIME;
               WHILE K <= SIZE DO
                  BEGIN FLAGS[K]:=0; K:=K + PRIME END;
               COUNT := COUNT + 1; WRITE(PRIME%,' ');
               IF PRIME <   10 THEN WRITE(' ');
               IF PRIME <  100 THEN WRITE(' ');
               IF PRIME < 1000 THEN WRITE(' ');
               IF COUNT MOD 11 = 0 THEN WRITE(13,'  ')
               END;
WRITE(13,13,' ',COUNT%,' PRIME NUMBERS GENERATED')
END.



! PROGRAM TOWERS OF HANOI !

VAR I, X, Y, N, NDISKS, TIME : INTEGER;
    D : ARRAY[33]      OF      INTEGER;
    Z : ARRAY[ 2]      OF      INTEGER;

PROC MODE(M);
   BEGIN
   MEM[#7D0]:=M; CALL(#7FD)
   END;

PROC COLORG(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#82B)
   END;

PROC DRAW(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#886)
   END;

FUNC XMAX;
   BEGIN
   MEM[#7D2]:=0; MEM[#7D5]:=0; MEM[#7D6]:=0;
   CALL(#912); XMAX:=256*MEM[#7D4]+MEM[#7D3]
   END;

PROC WAITTIME(T);
   BEGIN
   MEM[#1BE]:=T AND 255; MEM[#1BF]:=T SHR 8;
   REPEAT UNTIL (MEM[#1BE]=0) AND (MEM[#1BF]=0)
   END;
```

PASCAL

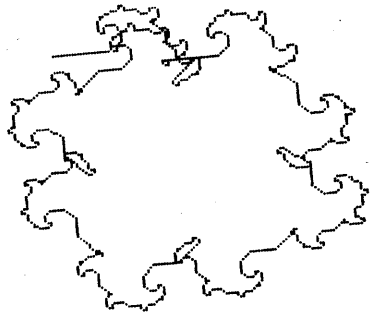
```
PROC HANOI(N,F,T,C);

   PROC MOVEDISK(F,T) ! LOCAL declared within hanoi !;
      VAR X1, X2, Y, H1, H2 : INTEGER;
      BEGIN ! procedure body of movedisk !
      WAITTIME(TIME);
      X1:=D[Z[F]+F*NDISKS]+F*24;
      X2:=11+F*24; Y:=Z[F]*4;
      DRAW(X1,Y,X2,Y,7);              ! whipe out    !
      DRAW(X2+2,Y,48*F+24-X1,Y,7);    ! "from" disk  !
      X1:=D[Z[F]+F*NDISKS]+T*24;
      X2:=11+T*24; Y:=Z[T]*4+4;
      DRAW(X1,Y,X2,Y,1);              ! draw         !
      DRAW(X2+2,Y,48*T+24-X1,Y,1);    ! "to" disk    !
      Z[T]:=Z[T]+1;
      H1:=Z[T]+T*NDISKS; H2:=Z[F]+F*NDISKS;
      D[H1]:=D[H2]; Z[F]:=Z[F]-1
      END ! movedisk !;

   BEGIN ! procedure body of hanoi !
   IF N=1 THEN MOVEDISK(F,T)
          ELSE BEGIN
               HANOI(N-1,F,C,T);     ! RECURSIVE def. !
               MOVEDISK(F,T);
               HANOI(N-1,C,T,F)      !       "        !
               END ! else !
   END ! hanoi !;

BEGIN ! *** main program *** !
MODE(3); COLORG(7,4,5,1);
REPEAT
   WRITE(12,'   ENTER NUMBER OF DISKS (1 - 11) : ');
   READ(NDISKS%)
   UNTIL (NDISKS>0) AND (NDISKS<12);
DRAW(0,0,XMAX,0,4);
FOR I:=1 TO 3 DO DRAW(I*24-12,0,I*24-12,60,5);
FOR X:=1 TO NDISKS DO
   BEGIN
   Y:=4*X; D[X]:=X;
   DRAW(X,Y,11,Y,1); DRAW(13,Y,24-X,Y,1)
   END ! x-loop !;
Z[0]:=NDISKS; Z[1]:=0; Z[2]:=0;
WRITE(13,'   ENTER TIME BETWEEN MOVES ');
WRITE('(UNITS OF 1/50 SEC) : '); READ(TIME%);
WRITE(13,'   PRESS ANY KEY TO START');
READ(X); WRITE(12);
HANOI(NDISKS,0,2,1);
N:=1; FOR I:=1 TO NDISKS DO N:=2*N; N:=N-1;
WRITE(13,13,'   TOTAL NUMBER OF MOVES MADE = ',N%)
END ! main !.
```

```
10    REM ONE TEXT LINE IN MODE 5/6
20    COLORG 0 5 10 15
30    MODE 6:FOR X=1 TO 20
35    DRAW RND(XMAX),20+RND(200) RND(XMAX),20+RND(200) 21+RND(3):NEXT
40    POKE #680B,#7A:REM CONTROL BYTE
50    POKE #680A,#40:REM COLOR   BYTE
60    FOR X=#6805 TO #6787 STEP -2:POKE X,#40+RND(26):POKE X-3,0:NEXT
70    IF GETC=32 THEN END
80    COLORG 0 RND(16) RND(16) RND(16):GOTO 60
```

PASCAL

```
!PROGRAM DECODE!
CONST STOP=#FF; EXIT=#3000;
VAR STADR, INDX, NUM: INTEGER;
PROC CRLF;
  BEGIN WRITE (13) END;
PROC FMT (VAL, LEN);
  VAR I, J: INTEGER;
  BEGIN !FMT!
    J:=VAL;
    FOR I:=2 + (J=0) TO LEN DO
      IF J=0 THEN WRITE (32) ELSE J:=J DIV 10;
    IF J>9 THEN WRITE ('&');
    WRITE (VAL%)
  END !FMT!;
BEGIN! MAIN!
  WRITE (12,'START DECODING AT ');READ (STADR#);CRLF;
  CRLF;INDX:=0;
  WHILE (STADR < #7FFF ) AND (MEM[STADR] <> STOP ) DO
    BEGIN FMT(INDX, 10);WRITE ('    ');INDX:=INDX+1;
      NUM:=MEM[STADR];
      CASE NUM OF
      0      : WRITE ('LIT');
      1      : WRITE ('OPR');
      2,#12  : WRITE ('LOD');
      3,#13  : WRITE ('STO');
      4      : WRITE ('CAL');
      5      : WRITE ('INT');
      6      : WRITE ('JMP');
      7      : WRITE ('JPC');
      8      : WRITE ('CSP')
      ELSE BEGIN WRITE ('ILL'); MEM[STADR]:= STOP END
      END;  !CASE!
      IF (NUM=#12) OR (NUM=#13) THEN WRITE ('X')
                                ELSE WRITE (32);WRITE (32);
      WRITE (MEM[STADR+1]%,',');
      NUM:=MEM[STADR+3] SHL 8 + MEM[STADR+2];
      WRITE (NUM%);CRLF;
      IF INDX MOD 15 = 0 THEN
        BEGIN READ (NUM);IF NUM='S' THEN CALL(EXIT) END;
      IF MEM[STADR] <> STOP THEN STADR:= STADR + 4;
    END; !WHILE!
END.!MAIN!
```

```
10    REM DRAGON KURVE
15    REM NAAR DRAGON CURVE
16    REM H4 RECURSION AND ITTERATION  BOEK : LISP DOOR P.H  WINSTON
20    REM RECURSIEF GEDEFINIEERDE FIGUUR
25    CLEAR 3000
30    DIM RETST(20.0),ST(60.0):SP=0.0:RETSP=0.0
40    ANGLE=0.0
45    MODE 6:COLORG 8 10 13 15:MIDX=XMAX/2.0:MIDY=YMAX/2.0
50    LENGTH=12.0
60    P=1.0:GOTO 1000:REM CALL FUNCTION
70    IF LENGTH>1.0 THEN X=MIDX+LENGTH*COS(ANGLE):Y=MIDY+LENGTH*SIN(ANGLE):DRAW
X,Y MIDX,MIDY 21:MIDX=X:MIDY=Y:GOTO 90
80    GOTO 2000:REM FUNCTION RETURN
90    LENGTH=LENGTH/SQR(2.0):ANGLE=ANGLE+SIGN*PI/4.0:SIGN=1.0
100   P=2.0:GOTO 1000:REM CALL FUNCTION
110   LENGTH=LENGTH/SQR(2.0):ANGLE=ANGLE-SIGN*PI/4.0:SIGN=-1.0
120   P=3.0:GOTO 1000:REM CALL FUNCTION
130   GOTO 2000:REM FUNCTION RETURN
1000  REM ======= CALL FUNCTION ======
1005  ST(SP)=SIGN:SP=SP+1.0
1010  ST(SP)=LENGTH:SP=SP+1.0
1020  ST(SP)=ANGLE:SP=SP+1.0
1030  RETST(RETSP)=P:RETSP=RETSP+1.0
1040  GOTO 70
2000  REM ======= RETURN FUNCTION  =====
2010  IF RETSP=0.0 GOTO 50
2030  SP=SP-1.0:ANGLE=ST(SP)
2040  SP=SP-1.0:LENGTH=ST(SP)
2045  SP=SP-1.0:SIGN=ST(SP)
2050  RETSP=RETSP-1.0
2060  ON RETST(RETSP) GOTO 70,110,130
```



PASCAL

```
PROC COLORT(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#7DE)
   END;

PROC MODE(M);
   BEGIN
   MEM[#7D0]:=M; CALL(#7FD)
   END;

PROC COLORG(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#82B)
   END;

PROC DOT(X,Y,C);
   BEGIN
   MEM[#7D5]:=X AND 255; MEM[#7D6]:=X SHR 8;
   MEM[#7D2]:=Y; MEM[#7D0]:=C; CALL(#84A)
   END;

PROC DRAW(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#886)
   END;

PROC FILL(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#8CC)
   END;

FUNC SCRN(X,Y);
   BEGIN
   MEM[#7D5]:=X AND 255; MEM[#7D6]:=X SHR 8;
   MEM[#7D2]:=Y; CALL(#912); SCRN:=MEM[#7D0]
   END;

FUNC XMAX;
   BEGIN
   MEM [#7D2]:=0;MEM [#7D5]:=0;MEM[#7D6]:=0;
   CALL(#912); XMAX:=256*MEM[#7D4]+MEM[#7D3]
   END;

FUNC YMAX;
   BEGIN
   MEM [#7D2]:=0;MEM [#7D5]:=0;MEM[#7D6]:=0;
   CALL(#912); YMAX:=MEM[#7D1]
   END;

FUNC CURX;
   BEGIN
   CALL(#938); CURX:=MEM[#7D5]
   END;

FUNC CURY;
   BEGIN
   CALL(#938); CURY:=MEM[#7D6]
   END;

PROC CURSOR(X,Y);
   BEGIN
   MEM[#7D5]:=X; MEM[#7D6]:=Y; CALL(#93E)
   END;

FUNC PDL(I);
   BEGIN
   MEM[#7D0]:=I; CALL(#94F); PDL:=MEM[#D8]
   END;

PROC WAITTIME(T);
   BEGIN
   MEM[#1BE]:=T AND 255; MEM[#1BF]:=T SHR 8;
   REPEAT UNTIL (MEM[#1BE]=0) AND (MEM[#1BF]=0)
   END;
```

DAI-RS232

Volgende ervaringen had ik (als DAI-user-beginner) bij het aansluiten van een printer...

DAI RS-232 (rev.5)

Bij het aansluiten van een EPSON MX-82 op mijn 2 maand jonge DAI (via RS-232-parallel interface), bleek een en ander niet naar behoren te funktioneren: het listen op de DAI liep rustig verder zonder op de printer te wachten.
Vermits printer en interface bij aankoop uitgetest waren, belandde de bewuste DAI spoedig op de operatie-tafel voor een inwendig onderzoek.
Bijgaande figuren tonen print-layout en schema van de RS-232 DTR ingang (main-board rev.5).

INTERRUPT: hangt de 56k weerstand bij vorige revisies niet aan massa ipv aan +5V ?

Uit metingen bleek dat bij een 'NOT READY' signaal (0.13 V), de spanning op ingangspin 10 van ic 74LS373 (latch) 0.95 V bedroeg wat te hoog is (volgens de ic-specs moet V(IL) < 0.8 V, low-level input voltage).
De spanningsval over de 3k3 weerstand is veroorzaakt door de stroom via de 56k weerstand (0.072 mA) en de 'low-level input current' van het ic (0.176 mA) die nog ruim binnen de ic-specs valt (I(IL) < -0.4 mA) .
Door de 3k3 R te verlagen tot 2k (met parallelweerstand van 5k6) was de patient weer in uitstekende gezondheid.

EPSON MX-82 met VERBORGEN TALENTEN

80 karakters ipv de in de manual aangekondigde 96 spoorden mij aan tot een tweede operatie: bij inwendig nazicht stond DIP-switch 1-5 in de OFF-stand.
Over de funktie hiervan zwijgt de manual in alle talen en vermeldt alleen:
'Never set this pin to the OFF position, always leave it in the ON position'.
En inderdaad, in de ON-stand krijg je de volle 96 karakterbreedte, in de OFF-stand slechts 80 (+ 16 marge achteraan).

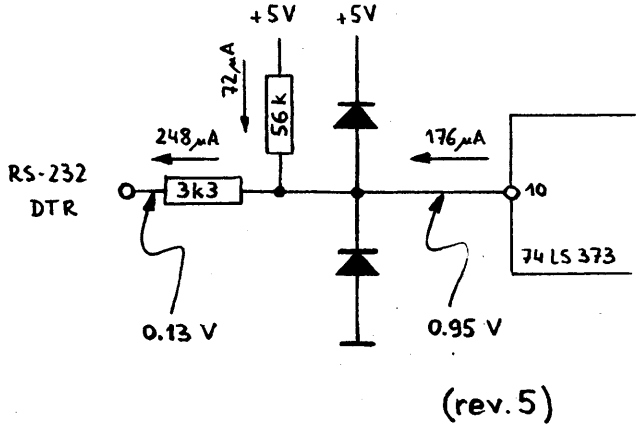
Tot slot nog een happy end !
Bij een vergelijking met de control-codes van de MX-100 waren 3 hiervan niet terug te vinden in de MX-82 manual: uitproberen dus en... met positief resultaat !

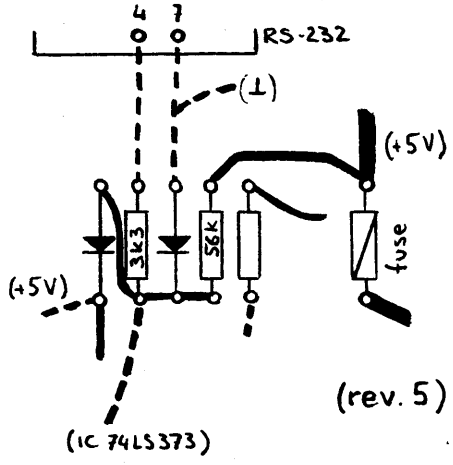
ESC G	double print (advance paper 1/216'' and repeat line)
ESC H	cancels ESC G
ESC M	elite print (letterbreedte tussen normal en condensed)

Deze codes staan dus vanaf nu ook in mijn MX-82 manual.

Herman MOEYS

3200 KESSEL-LO (LEUVEN)





machinetaal in een rem-statement

(Mijn eerste "eigen" machinetaal programma)

In MC 4 (Die Mikrocomputer Zeitschrift nov/dec 1981) wordt op de mogelijkheid gewezen om (korte) stukje machinetaal onder te brengen in een basic REM regel
Voordeel : Alles wordt in een keer geladen op de gewone manier van een basic programma.

Nadeel: Die stonden er niet bij maar ....

Mogelijkheden: Die komen in het navolgende aan de orde.

① Hoe weet je waar de gebruikte REM regel in het geheugen zit?

b.v. 10 REM *****

Op de adressen $29F, $2A0 vinden we EC, 03 dat wil zeggen dat het begin van de basic text te vinden is op adres $03EC. We vinden daar:

+5 ←

$03E0 FF FF FF FF FF FF FF FF FF FF 1F FF 09 00 0A 8F

→

$03F0 05 2A 2A 2A 2A 2A 00 00 FF FF FF FF FF FF FF FF

1F FF : einde van de heap (zie DAI nawic nr 7 blz 188)
09 : de komende basic regel bevat 9 ~~tekens~~ bytes van 00 t/m 2A
00 0A : regelnummer 10
8F : code voor REM
05 : na de REM komen 5 tekens
2A....2A : 5 maal asterix

② In plaats van 5 maal 2A kunnen we ook 5 andere bytes kiezen die een machinetaalprogramma vormen. Dit programma kan dan aangeroepen worden d.mv CALLM #3F1 mits aan twee voorwaarden is voldaan :

- het programma begint met 10 REM ....
- de text begint op adres $03EC ; dit is zeker zo na power on of hard reset. voorafgaande aan de eerste basic regel mag geen CLEAR gegeven worden

In de listing bevat de eerste regel een reeks willekeurige karakters. Soms is de regel niet meer kompleet. In het stukje machinetaal zit dan b.v. $0C ; bij het listen wordt dan het scherm schoongemaakt.

③ Een voorbeeld

Het volgende programma zet alle karakters op het scherm van CHR$(#FF) aftellend tot CHR$(#0C) = CHR$(12) ⇐ scherm schoon:

```
1   0 3 F 1   C 5      PUSH B        } inhoud v.
2   0 3 F 2   D 5      PUSH D        } registers bewaren op stack
3   0 3 F 3   E 5      PUSH H        }
4   0 3 F 4   F 5      PUSH PSW      }
5   0 3 F 5   0 6 0 C  MVI B,0C      CHR$(12) in B
6   0 3 F 7   3E FF    MVI A,FF      #FF in A
7   0 3 F 9   CD 6 0 DD CALL :BD60   CHR$(<A>) → scherm <--.
8   0 3 F C   3D       DCR A         <A> ← <A> - 1         :
9   0 3 F D   B 8      CMP B         } als <A> - <B> ≠ 0 dan :
A   0 3 F E   C2 F9 0 3 JNZ          } naar 03F9 ----------'
B   0 4 0 1   F1       POP PSW       }
C   0 4 0 2   E1       POP H         } register van stack halen
D   0 4 0 3   D1       POP D         } inhoud
E   0 4 0 4   C1       POP B         }
F   0 4 0 5   C9       RET
```

Het programma bevat 21 bytes. Daarom begint het basic programma met
```
10 REM *** 21 x asterix ***
20 CALLM #3F1
30 END
```

Met het Substitute kommando worden in UT op adres 03F1 t/m 0405 alle 2A bytes vervangen door de programma byte's.

Het geheel kan nu als basic gesaved en geladen worden.

Een REM statement kan tot 122 tekens bevatten; het machinetaal programma kan dus een overeenkomstige maximale lengte hebben (zonder al te grote kunstgrepen)

(1) De tweede voorwaarde genoemd onder punt (2) (start of text at 03EC) kan problemen geven.
Als door een CLEAR de heap ruimte groter is gemaakt dan #100 bytes, dan zal het begin van de text buffer zijn opgeschoven naar een hoger adres.
Dit adres is weer te vinden op #29F en #2A0 deze bevatten resp het lage en hoge aangepaste adres byte van het begin van de text buffer.

Het voorbeeld zou nu als volgt aangepast kunnen worden voor eventuele heap vergroting:

```
10 REM ------------
20 ADRES = PEEK(#2A0)*256+PEEK(#29F)+5
30 CALLM ADRES
40 END
```

We kijken dan eerst op #29F en #3A0 waar de text buffer begint. Door hij het begin adres 5 op te tellen (zie (1)) hebben we het begin adres van het machinetaal programma!

⑤ Helaas, als we een en ander proberen met ons voorbeeld programma dan loopt de zaak vast, zodat alleen het bekende witte knopje (de rode mag ook) uitkomst biedt.

In het machinetaal programma zit een sprong terug van $ regel A naar regel 7.
In de sprong instruktie wordt het adres $3FG vermeldt.
Maar dat klopt niet meer als de text buffer en daarmee het machinetaalprogramma naar hogere adressen wordt verplaatst.

Een eenvoudige oplossing hiervoor zou zijn een instruktie in de vorm: sprong terug, aantal adressen

Zo'n relatieve sprong instruktie kent de 8080A helaas niet!

En dan zit je als beginner met machinetaal goed vast
Na een paar telefoontjes kreeg ik de volgende tips van Freddy de Raat:

- bereken het adres waarnaar je terug springt en zet dat in reg's H, L
- zet daarna de inhoud van H,L op stack: PUSH H
- komt nu de sprong voorwaarde bv een test op nul, dan kun je springen door RETURN NON ZERO
- of verder gaan met POP H waardoor de stack weer op zijn oude hoogte terug is.

De truc is dat de berekening van het sprong adres uitgaat van het adres dat op #29F en #2AØ als wijzer naar de text buffer staat.
Omdat deze wijzer wordt aangepast bij een CLEAR, wordt het sprong adres nu ook aangepast.

⑥ Voorbeeld

1	Ø 3 F 1	C5	PUSH B	
2	Ø 3 F 2	D5	PUSH D	
3	Ø 3 F 3	E5	PUSH H	
4	Ø 3 F 4	F5	PUSH PSW	
5	Ø 3 F 5	2A 9F Ø2	LHLD :Ø29F	} berekening sprong adres
6	Ø 3 F 8	11 14 Ø Ø	LXI D ØØ14	
7	Ø 3 F B	19	DAD D	
8	Ø 3 F C	Ø6 ØC	MVI B, ØC	
9	Ø 3 FE	3E FF	MVI A, FF	
A	Ø 4 Ø Ø	CD 6Ø DD	CALL DDØ6	
B	Ø 4 Ø 3	3D	DCR A	
C	Ø 4 Ø 4	B8	CMP B	
D	Ø 4 Ø 5	E5	PUSH H	} relatieve sprong
E	Ø 4 Ø 6	CØ	RNZ	
F	Ø 4 Ø 7	E1	POP H	
1Ø	Ø 4 Ø 8	F1 E1 D1 C1	POP ALL	
11	Ø 4 Ø C	C9	RET	

Toelichting

Regel nr.

5 Startadres van text buffer wordt gelezen op #2gF. #2Aφ en geplaatst in regs H, L

6 Het adres waar naar gesprongen wordt ligt #4φφ - #3EC = #14 adressen verder dan het start adres van de text buffer

daarom φφ #, 14 in de regs D, E

7 na optellen staat het resultaat (dus het sprongadres) in de regs H, L

D Het sprong adres wordt vanuit H, L op stack gezet.

E Is <A> - <B> ≠ φ dan return naar adres dat bovenaan op stack staat

Wordt nu voor het laden een CLEAR gegeven of is de HEAP door een voorafgaand programma vergroot, dan loopt het programma nu wel.

En dan ben je zo blij dat het werkt, dat je alles nog eens opschrijft om het voor al niet te vergeten.

Mogelijk heeft een ander er dan ook nog wat aan.

TOEGIFT :

Groetjes Thijs Berke
Ulissingen

```
1φ  MODE 6: COLOR 4 φ 5 φ φ
15  FOR SH = φ TO 5φ STEP 2
2φ  V = φ : Y = 25 - SH
25  FOR X = φ TO XMAX - SH
3φ    A = -1φ * Y
4φ    V = V + A * φ.φ2          } "snelle sinus"   1φ regelt de periode
5φ    Y = Y + V * φ.φ2
6φ    DOT X+SH, 75+Y+2*SH  5
7φ  NEXT X: NEXT SH
1φφ GOTO 1φφ
```

Probeer ter vergelijking dit programma eens met SIN !

TEKENS NAAR SCHERM

REM + MLP + REL. JMP

HEX DUMP

```
03 0E 00  FF FF FF FF FF FF FF FF FF FF 7F FF 20 00 00 0A A9
03 0F 00  1C C5 D5 E5 F5 2A 9F 02 11 14 00 19 06 0C 3E FF
04 04 00  CD 60 DD 3D B8 E5 C0 E1 F1 E1 D1 C1 C9 21 00 14
04 11 00  A5 40 06 BF A0 A0 A3 20 17 15 00 00 02 A0 14 00
04 20 00  00 01 00 20 17 15 00 00 02 9F 14 00 00 00 05 07
04 30 00  00 1E B3 9F 40 06 FF 03 00 28 84 00 05 41 44 52
04 40 00  45 53 04 0A FC 40 00 05 4C 49 49 53 54 04 00 00
04 50 00  00 00 01 54 04 00 00 00 00 00 00 00 00 00 00 00
```

```
02 9F  03   } pointer naar start of text
02 A0  EC   }
```

```
10 REM ..... oorspronkelijk 28 x asterix ....
20 ADRES = PEEK(#2A0)*256 + PEEK(#29F)+5
30 CALL ADRES
40 END
```

INTERACTIEVE VIDEOTEX

BEKNOPTE BESCHRIJVING

1. WAT IS INTERACTIEVE VIDEOTEX?

VIDEOTEX is een communicatiesysteem dat o. a. toelaat informatie, opgeslagen in een computergeheugen, te verspreiden en op te vragen. Daartoe heeft de gebruiker een telefoon nodig evenals een terminal bestaande uit een beeldscherm en een klavier. Het kan hier gaan om een specifieke terminal, een aangepast kleurentelevisietoestel, een persoonlijke microcomputer enz...

Terloops weze opgemerkt dat het coderingsschema van de interactieve videotex eveneens gebruikt wordt voor de omgeroepen videotex, genoemd Teletekst. Het betreft hier een éénwegsysteem voor het verspreiden van informatie, die verzonden wordt bij middel van het televisiesignaal.

Interactieve videotex is een systeem met individuele tweeweg-transmissie dat aldus een dialoog tussen gebruiker en computer toelaat. De opslagmogelijkheden van het ganse systeem zijn slechts beperkt door de geïnstalleerde stockeringscapaciteit. De bediening is zeer eenvoudig en door enkele toetsen in te drukken krijgt de gebruiker alle gekozen informatie bijna onmiddellijk op het scherm. Dank zij de interactie zijn ook bewerkingen en transacties mogelijk.

REGIE VAN TELEGRAFIE
EN TELEFONIE

Departement Informatieverwerking

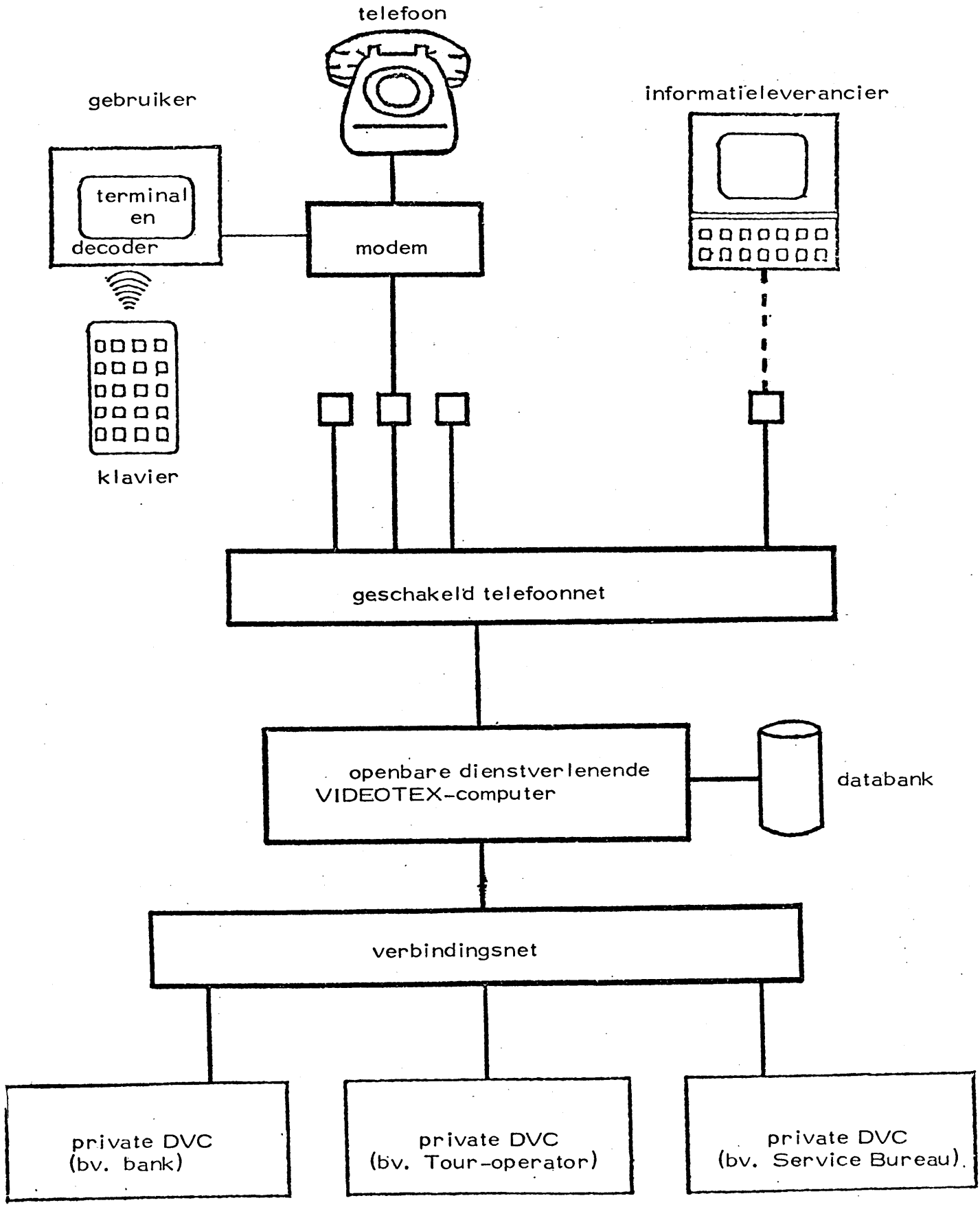
Projectgroep Videotex
Secretariaat : 02/216 38 27

April 1982

Projectgroep Videotex R. T. T.

2. DE SAMENSTELLENDE ONDERDELEN

2. 1. Algemeen schema



Alle onderdelen die deel uitmaken van de R. T. T. -infrastructuur zijn in een dikkere lijnsoort weergegeven.

2. 2. De dienstverlenende computers (DVC)

De DVC's bevatten gegevens en programma's.

De (voorlopig enige) openbare DVC die door de R. T. T. zal uitgebaat worden, zal de volgende functies bevatten :

- kontrole van de verrichtingen (o. a. toegangsbeveiliging, taxatie)
- programma's voor de zeer eenvoudige ondervraging van de databank
- faciliteiten voor de inleiding en bijwerking van gegevens in deze databank
- een brievenbusdienst voor de gebruikers
- een dienst voor gegevensvastlegging
- doorschakeling naar private DVC's.

De private DVC's bevatten programma's en gegevenstructuren die door hun exploitant vrij bepaald worden.

2. 3. Gebruiker

Hij of zij die bij middel van een terminal gebruik maakt van de diensten van een DVC van het videotexnet.

2. 4. Dienstaanbieder of informatieleverancier

Diegene die over gegevens beschikt of een toepassing beheert welke hij ter beschikking wil stellen van bepaalde of alle gebruikers. Hij kan deze dienst onderbrengen :

- op de openbare DVC of,
- op zijn eigen private DVC of,
- op een private DVC van een service bureau.

De dienstaanbieder kan zijn toepassing, zo gewenst, commercialiseren door de gebruiker een vergoeding aan te rekenen voor de geconsumeerde diensten. Deze bedragen kunnen door de R. T. T. worden geïnd.

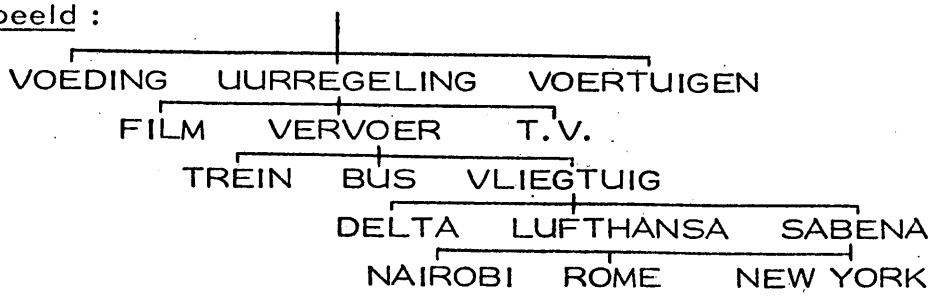
2. 5. De terminal

Deze kan teksten en eenvoudige grafismen weergeven in 8 kleuren. Hij kan eventueel van een drukkertje of een recorder worden voorzien voor de vastlegging van de dialoog.

2. 6. De programma's voor ondervraging van de databank uitgebaat op de openbare DVC

- De hier ontwikkelde videotex-logica laat een snelle en precieze opzoeking van informatie toe.
- De informatie wordt opgeslagen in een eenvoudige boomstructuur :

Voorbeeld :



- De opgeslagen informatie kan ook langs een bepaald "trefwoord" worden opgezocht of via de naam van de dienstaanbieder.

3. WAT IS KARAKTERISTIEK AAN INTERACTIEVE VIDEOTEX?

3.1. In het algemeen

Videotex is een snel en eenvoudig informatiesysteem dat iedereen kan gebruiken, zonder speciale opleiding.
Informatie kan op een zeer eenvoudige manier worden opgezocht, dank zij een aangepaste zoekstructuur.
De informatie kan zeer gemakkelijk "up to date" gehouden worden.
In principe is de opslagcapaciteit onbeperkt.
Bij videotex kan een goedkope terminal (TV scherm) gebruikt worden.
Met videotex zijn heel wat grafische mogelijkheden voorzien zodat een aantrekkelijk beeld opgebouwd kan worden (kleuren, knipperen, ...).
Twee richtingscommunicatie tussen gebruiker en videotex-systeem laten alle types van interactie toe.
De gebruiker kan ook boodschappen sturen naar het videotex-centrum of andere gebruikers.
De "besloten gebruikersgroepen" kunnen de informatie beperken tot hun leden.
De informatie is beveiligd met persoonlijke paswoorden.
Beperkte dataverwerking is mogelijk op de openbare DVC o. a. gegevensvastlegging.
Overdrachtsnelheid :

- van abonnee naar videotex-systeem : 7, 5 tekens per seconde
- van videotex-systeem naar abonnee : 120 tekens per seconde

3.2. Voor het bedrijfsleven

Naast een informatiebron is videotex een goed communicatiesysteem, om relaties met geografisch gespreide filialen, verdelers, kantoren of personeelskernen te onderhouden.
Een snelle verspreiding van veelgebruikte en vaak wijzigende gegevens naar vele medewerkers is dan mogelijk.
Ook intern voor een groot bedrijf met vele afdelingen en diensten kan videotex de communicatieproblemen oplossen.
Door het opzetten van "besloten gebruikersgroepen" blijft de vaak vertrouwelijke informatie beperkt tot de leden.
De informatieleverancier bepaalt zelf wie toegang heeft tot zijn informatie en aan welke prijs.
Het is mogelijk met de videotex-dienst toegang te krijgen of te verlenen tot uw eigen of andere computers. Daarenboven kan op termijn een internationaal schakelnet tot stand komen.
Videotex is een ideaal publicitair medium voor grote verspreiding met attractieve grafische mogelijkheden.
De informatieleverancier kan op een eenvoudige en efficiënte manier zijn informatie "te koop" aanbieden.

4. VOORBEELDEN VAN TOEPASSING.

Financieel : Agentschappen krijgen diverse informatie ter beschikking, bv. beursberichten, stand der rekeningen, financiële, juridische, fiscale informatie, boodschappendienst. Home-banking.

Verzekeringen : Opvolgen van contracten van de klanten. Speciale reglementeringen, berichten, tarieven.

Toerisme, transport : - Informatie aan de reisbureau's, catalogus, actuele tarieven, beschikbare zitplaatsen of kamers, onmiddellijke reservaties, boekingen.
- Uurregelingen allerlei : treinen, vliegtuigen, ...

Automobielsector : Videotex-net bij dealers, invoerders, garages, voor berichten, catalogen, prijslijst, wisselstukken, voorraad wagens, ...

Openbare dienst : Het omvangrijke pakket aan reglementen, wetteksten, procedures.

Uitgeverijen : Verspreiden van vakinformatie ...

Bedrijfsnieuws : Groepering van diverse gegevens over de bedrijfswereld.

Distributiesector : Communicatie met filialen, franchises. Winkelketens.

Immobiliën : Informatie voor en tussen makelaars. Aanbod aan het kliënteel.

Wetgeving : Gestructureerd bijhouden van wetten, reglementen, arresten, staatsblad, ...

Verenigingen : Informatie voor de leden van de beroepsvereniging. Aankondigingen, vergaderingen, brievenbus, ...

faculteit

```
2     CLEAR 3000:PRINT CHR$(12)
5     INPUT "  FACULTEIT";F%:PRINT
10    DIM N%(255)
15    S%=-1:V%=1:N%(0)=1:REM INIT. 1!
19    PRINT :PRINT CHR$(#E);F%;" !":POKE #131,1
20    S%=S%+1:T%=0:W%=0
25    CURSOR 10,20:PRINT V%
30    B%=N%(T%)/10000
40    A%=N%(T%)-10000*B%
50    A%=A%*V%
60    B%=B%*V%
70    A%=A%+W%
80    A2%=A%/10000
90    A1%=A%-A2%*10000
100   B%=B%+A2%
110   W%=B%/10000
120   B1%=B%-W%*10000
130   N%(T%)=A1%+B1%*10000
140   T%=T%+1
150   IF T%-1<>S% GOTO 30
200   IF N%(S%)=0 THEN S%=S%-1
203   V%=V%+1:IF F%<>V%-1 THEN 20
210   POKE #131,0:T%=S%:P$=""
220   B%=N%(T%)/10000:A%=N%(T%)-B%*10000
230   S$=STR$(B%):S$=RIGHT$(S$,LEN(S$)-1):S$=LEFT$(S$,LEN(S$)-2)
235   IF S$="0" THEN S$=""
240   T$=STR$(A%):T$=RIGHT$(T$,LEN(T$)-1):T$=LEFT$(T$,LEN(T$)-2)
250   IF B%<>0 THEN IF LEN(T$)<4 THEN T$="0"+T$:GOTO 250
260   P$=S$+T$:T%=T%-1:IF -T%=1 GOTO 350
270   FOR T%=T% TO 0 STEP -1
280   B%=N%(T%)/10000:A%=N%(T%)-B%*10000
290   S$=STR$(B%):S$=RIGHT$(S$,LEN(S$)-1):S$=LEFT$(S$,LEN(S$)-2)
300   IF LEN(S$)<4.0 THEN S$="0"+S$:GOTO 300
310   T$=STR$(A%):T$=RIGHT$(T$,LEN(T$)-1):T$=LEFT$(T$,LEN(T$)-2)
320   IF LEN(T$)<4 THEN T$="0"+T$:GOTO 320
330   P$=P$+S$+T$:IF LEN(P$)>65 THEN GOSUB 1000
340   NEXT T%
350   IF LEN(P$)<5 GOTO 358
351   FOR U%=5 TO LEN(P$) STEP 5
352   D$=D$+LEFT$(P$,5)+" "
354   P$=RIGHT$(P$,LEN(P$)-5)
356   NEXT U%
358   PRINT D$;:PRINT P$:P$="":D$=""
999   END
1000  O$=RIGHT$(P$,LEN(P$)-65)
1010  P$=LEFT$(P$,65)
1020  FOR U%=5 TO 65 STEP 5
1030  D$=D$+LEFT$(P$,5)+" "
1035  P$=RIGHT$(P$,65-U%)
1040  NEXT U%
1050  PRINT D$:P$=O$:D$=""
1060  RETURN
```

```
  FACULTEIT?100

   100   !
 93326 21544 39441 52681 69923 88562 66700 49071 59682 64381 62146 85929 63895
 21759 99932 29915 60894 14639 76156 51828 62536 97920 82722 37582 51185 21091
 68640 00000 00000 00000 00000 000
 END PROGRAM
```

cassette list

```
5     REM CASSETTE LIST rev 5. ARTS
6     CURS=20.0
10    MODE 0:PRINT CHR$(12);:POKE #131,1:AD$=""
30    FOR A%=1 TO 21:READ B%:AD$=AD$+CHR$(B%):NEXT
40    AD%=PEEK(VARPTR(AD$)) IOR (PEEK(VARPTR(AD$)+1) SHL 8)+1
50    PRINT "DIT PROGRAMMA"
51    PRINT "GEEFT EEN INHOUDSOPGAVE VAN EEN CASSETTE"
52    PRINT "A    OP   110 BAUD VOOR DE MATRIX-PRINTER"
53    PRINT "B    OP   300 BAUD VOOR EEN MODEM VERBINDING"
54    PRINT "C    OP  1200 BAUD VOOR DE P880 VERBINDING"
55    PRINT "D    OP  2400 BAUD VOOR HET INTEL-SYSTEEM"
56    PRINT "E    OP  9600 BAUD VOOR HET DISPLAY"
60    PRINT " WAT WORDT GEWENST ? MAAK UW KEUZE !  ";
70    K=GETC:IF K<#41 OR K>#45 THEN GOTO 70
73    REM WAIT TIME 20
75    PRINT CHR$(12)
80    ON K-#40 GOSUB 91,92,93,94,95
81    POKE #FF05,BAU:GOTO 100
91    BAU=1.0:RETURN
92    BAU=#84:RETURN
93    BAU=#88:RETURN
94    BAU=#90:RETURN
95    BAU=#C0:RETURN
100   INPUT " CODE NUMMER VAN DE CASSETTE IS :";CN$:PRINT :PRINT
111   PRINT "WAT IS DE DATUM VAN VANDAAG   YYMMDD"
112   INPUT "VB 14 MAART 1982 IS  820314  ";DAT$:PRINT
113   IF CN$="" OR LEN(DAT$)<>6.0 THEN GOTO 1998
115   PRINT CHR$(12)
120   POKE #131,0:PRINT "DATUM :  "+DAT$:PRINT CN$:FOR A%=0 TO LEN(CN$)-1:PRINT
-";:NEXT:PRINT CHR$(#D):CPT%=1
1000  CURSOR 0,23:POKE #FF05,#C0:POKE #131,1
1001  PRINT "                                                                    "
1010  CURSOR 0,23:CALLM AD%:CURS=CURS-1.0
1011  IF CURS=0.0 THEN CURS=20.0
1012  CURSOR 1,CURS
1013  POKE #FF05,BAU:POKE #131,0:POKE #40,#30:PRINT CPT%;" - ";
1030  FOR A%=#BFE7 TO #BF85 STEP -2:PRINT CHR$(PEEK(A%));:NEXT:PRINT CHR$(#D)
1035  IF GETC<>0 THEN 2000
1040  CPT%=CPT%+1:GOTO 1000
1998  PRINT :PRINT "DIT PROGRAMMA WERKT NIET ZONDER GOEDE GEGEVENS"
1999  PRINT "START HET PROGRAMMA OPNIEUW MET 'RUN'":PRINT :END
2000  POKE #131,1:PRINT :STOP
2010  GOTO 1040
10000 DATA #F5,#C5,#D5,#E5,#01,#40,#00,#11,#B1,#80,#21,#9E,#E6,#CD,#CE,#02,#E1,#
D1,#C1,#F1,#C9
```

```
100   REM /** DNE 241 MODIFIER 21MRT82 **/
200   REM /* FILE TYPE (RD/WR) : #23 (#) OR #31 (1)
210   POKE #2BE1,#23:POKE #2BDC,#23
300   REM /* #L PAGE FORMAT
310   POKE #1F40,#42:REM PAGE 1
320   POKE #1F3A,#42:REM FOLLOWING PAGES
400   REM /* HARDCOPY : WAITSPACE (#L AND #P)
410   REM POKE #1F36,#CD:POKE #1F37,#CE:POKE #1F38,#2C:REM YES
420   POKE #1F36,0:POKE #1F37,0:POKE #1F38,0:REM NO
900   REM /*** LAB USE ONLY ***/
910   POKE #2F01,1:REM EPSON AUTO-LF
1000  FOR I=0 TO 2500:IF GETC=#20 GOTO 1100:NEXT
1100  POKE #100,0:POKE #101,0:CALLM 12000
```

weersatelliet foto's op uw tv-scherm

Sinds kort gebruik ik mijn DAI computer voor de weergave van foto's , afkomstig van weersatellieten. Het blijkt, dat de grafische mode 6 geschikt is voor het zichtbaar maken van de foto's van wolkenformaties, welke dagelijks door een aantal weersatellieten naar de aarde worden gestuurd.

Op de plaatjes die tot nu toe via de DAI computer zijn geproduceerd zijn deze wolken min of meer duidelijk te onderscheiden, maar ik denk dat zowel de software als de hardware nog verder verbeterd kunnen worden.

Wanneer er meer DAInamic leden zijn die hun computer willen gaan gebruiken voor het weergeven van satellietfoto's, of die geinteresseerd zijn in het geheel van ontvangen en weergeven van foto's , kunnen zij kontakt opnemen met mij. Als veel mensen meer willen weten over deze fascinerende hobby, dan kan ik wat informatie toesturen aan de redaktie van DAInamic voor plaatsing in de nieuwsbrief.

Voor geinteresseerden: de kosten voor het ontvangen en weergeven van weersatellietfoto's bedragen ongeveer f 800,- (BFR 11500) en daarvoor heeft u dan een ontvanger, antenne en interface voor de DAI.

Voor meer informatie:

H. Bakker
Rijnstraat 28
6811 EW Arnhem
tel. 085-451394

INTERFACE HARDWARE

Op het blokschema in Fig. 2 komt linksboven het 2400 Herz signaal binnen van de tuner. Dit signaal wordt eerst gefilterd en versterkt (A1 resp. A2), daarna gelijkgericht en de condensator die hierbij wordt opgeladen, wordt met een constante stroom ontladen. De Comparator A3 zorgt ervoor dat het zaagtand-achtige signaal over de condensator C wordt omgezet in een blokspanning met variable pulsbreedte.
Verder zorgt het timer IC NE 555 ervoor dat de puls na ca. 300 micro seconden in ieder geval weer laag wordt, ook al is het ingangssignaal te hoog in volume.
Bij weergave van het opgenomen signaal zet OP AMP A4 de misvormde videopulsen weer om in een nette blokspanning, die via buffers N4 en N5 naar de DAI gaan.
Tevens triggert de positieve flank van deze pulsen een oscillator gevormt door N2 en N3. De blokspanning van deze oscillator heeft dus dezelfde frequentie als de videopulsen en tevens vallen van beide signalen de positieve flank samen.

Deze positieve flank zal later in het machinetaal programma worden gebruikt als referentiestart voor het nemen van samples.

Voor we de interface kunnen gebruiken, moeten we eerst met potmeter P3 de frequentie van de oscillator instellen op iets minder dan 2400 Herz. Wie geen frequentiemeter heeft, kan op het gehoor afgaan door eerst te luisteren naar de 2400 Herz van een weersatelliet, en de oscillatorfrequentie iets lager dan deze frequentie in te stellen (ca. 2360 Herz).

Als we nu een ontvanger aansluiten op de interface en we wachten tot we het geluid van een weersatelliet horen, dan moeten we eerst P1 en P2 zo instellen, dat de pulsbreedte varieert tussen ca. 100 en 300 microseconden. Ook dit is eventueel op het gehoor af te regelen door P1 zo in te stellen dat het signaal naar de Tape rec. nog net zuiver klinkt. Maar een oscilloscoop is hierbij toch wel handig ...
P1 bepaalt de versterking en P2 het contrast.

WEERSATELLIETEN FOTO'S OP DE DAI COMPUTER.

De DAI Computer kan voor veel dingen gebruikt worden, maar bovengenoemde toepassing zal velen van u vreemd voorkomen. Toch is het met vrij eenvoudige middelen mogelijk om foto's, afkomstig van weersatellieten, zichtbaar te maken op een TV scherm. Wat hiervoor nodig is, zal in dit verhaal aan de orde komen.

Eerst iets over weersatellieten. Zij kunnen verdeeld worden in globaal twee groepen. De eerste soort draait met de aarde mee en staat daardoor stil ten opzichte van de aarde (dit heet Geostationair). Deze satellieten vinden we op ca. 36000 Km hoogte loodrecht boven de evenaar.
Doordat ze ten opzichte van de aarde stil staan, fotograferen ze steeds hetzelfde gedeelte van deze aarde.

De tweede soort draait rond de aarde via noord- en zuidpool (ongeveer) en staan daardoor niet stil t.o.v. de aarde. Bovendien is hun vlieghoogte 'slechts' 800 - 1500 Km. Met deze tweede soort wil ik me verder bezighouden.

Een gevolg van deze noord-zuid baan is dat dit soort satellieten slechts dan te ontvangen is, wanneer ze over West Europa 'vliegen' en dan nog maar gedurende maximaal 20 minuten. Bij iedere omwenteling verschuift de baan van de satelliet 15 graden, zodat na een halve dag weer opnieuw ontvangst mogelijk is. De satelliet vliegt dan echter in omgekeerde richting.

Om weersatellieten te kunnen ontvangen hebben we (u raadt het al) een ontvanger nodig. Wanneer u handig bent is zo'n ontvanger zelf te maken. Later zal een leverancier genoemd worden waar bouwpakketten en kant-en-klaar ontvangers te koop zijn.

Het signaal wat we ontvangen bestaat uit een toon van 2400 Herz en de eigenlijke video informatie zit in de amplitude van dit signaal (Amplitude Modulatie AM).
Wanneer we dit signaal gelijkrichten houden we sinusvormige pulsen over waarvan de hoogte een maat is voor een licht- of donker niveau .
De frequentie van deze toon is echter niet continu. Twee maal per seconde wordt deze onderbroken door een toon van een andere frequentie. Deze korte toon geeft het begin aan van een nieuwe beeldlijn aan, want weersatellieten maken eigenlijk geen foto van de aarde, maar zij 'scannen' deze lijn voor lijn af, net zoals een TV scherm wordt opgebouwd.
Per seconde zendt een satelliet dus twee beeldlijnen uit, maar dat geldt alleen voor de Russische satellieten. Amerikaanse satellieten zenden per seconde vier beeldlijnen uit, nl twee beeldlijnen van het zichtbare gedeelte van de aarde +wolken, en twee lijnen van het Infra Rode gedeelte . Hierdoor kunnen tevens foto's welke 'snachts ontvangen worden zichtbaar gemaakt worden.
De mooiste foto's komen echter van de Russische satellieten Meteor 120 en - 240, zodat we ons eerst met de weergave van deze foto's zullen bezighouden.

Het begin van iedere beeldlijn wordt door de satelliet aangegeven door een sink-puls. Gedurende een zekere tijd wordt het video signaal onderdrukt (vergelijk het TV signaal) en hierop zal een later te behandelen machinetaal programma reageren door te beginnen op een nieuwe regel.

Goed, wat we dus ontvangen is een toon van 2400 herz welke in amplitude (sterkte) varieert. Om later de foto meerdere malen te kunnen weergeven nemen we deze toon tijdens de ontvangst op een cassette recorder of (beter) bandrecorder op.
Als we de toon rechtstreeks op de band zetten, zullen we echter later bij de weergave last kunnen krijgen van zg. Drop-outs.
Dit is het kort weg vallen van het geluid tijdens weergave door stof of slechte plekken op de band. Als het geluid wegvalt zien we dat meteen in een kleurverandering op het TV-scherm.
Om dit te voorkomen vorm ik het signaal eerst om tot pulsen die in lengte afhankelijk zijn van de amplitude van het ontvangen signaal. Met andere woorden: als ik na het gelijkrichten van het 2400 Herz signaal een 'halve sinusvormige puls' krijg met een HOGE amplitude, dan zet ik een puls met een LANGE pulslengte op de band.

De hard-ware die ervoor zorgt dat de video informatie wordt omgezet naar pulsen van variabele lengte, is niet erg ingewikkeld. Ik zal hier later verder op ingaan.
Als we een geslaagde opname hebben gemaakt, staan op de band pulsen, die bij weergave naar de computer gaan. De computer heeft behalve deze pulsen nog een tweede signaal nodig, nl een klok signaal. Dit tweede signaal wordt in de weergave hardware opgewekt door een oscillator die precies dezelfde frequentie heeft als de video puls-frequentie.
Wat de computer nu tijdens weergave van de foto moet doen is het volgende:

1 - Wacht tot de video-informatie gedurende enige tijd laag blijft (trigger sink puls),

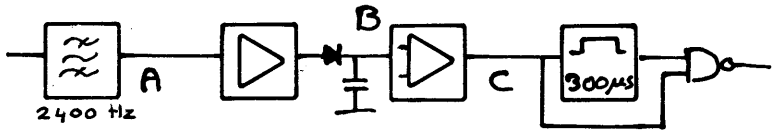
2 - Kijk hierna steeds na het hoog worden van de klok hoe lang de video puls duurt,

3 - Maak afhankelijk van de puls-lengte een witte, donker-grijze, licht-grijze of zwarte punt als dit kan (zie verder..).

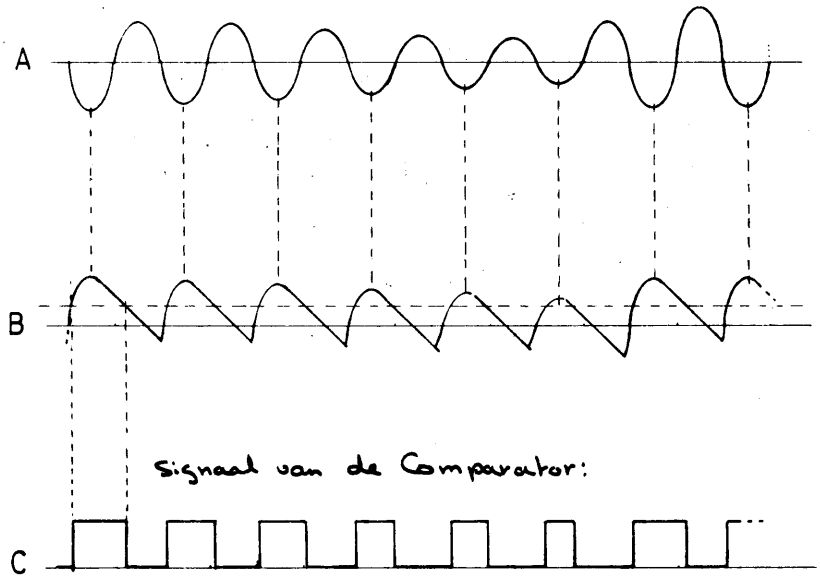
Verder moet de computer natuurlijk de aldus gemaakte punten (dot's) op de juiste plaats op het beeldscherm plaatsen, en stoppen als het scherm vol is.
Het aantal pulsen (halve sinusvormige video-pulsen) per beeld lijn is echter groter dan het aantal dot's wat de DAI op een lijn kan plaatsen.
Twee beeldlijnen p/sec met een pulsfrequentie van 2400 Herz betekent per beeldlijn een kleine 1200 pulsen (2400/2 - sink puls). De DAI komt niet verder dan 336 dots.
Daarom plaatsen we ten eerste een gedeelte van de beeldlijn niet op het scherm ,en ten tweede plaatsen we van het wel zichtbare gedeelte slechts de helft (1 video puls WEL, de volgende NIET enz.) . Met een INPUT statement is de grootte van het linker gedeelte van het totale beeld wat overgeslagen moet worden, in te voeren.
De computer zal na de sink-puls eerst dit gedeelte overslaan, vervolgens de helft van het vervolg van de beeldlijn displayen tot de regel op het scherm vol is, en tenslotte weer wachten op de volgende sink-puls.
Na 256 lijnen keert het machine taal programma terug naar BASIC en wacht tot een toets wordt ingedrukt.

wordt vervolgd...

weersatelliet foto's op uw tv-scherm



Signaal van A1:



Signaal afkomstig van Taperec:

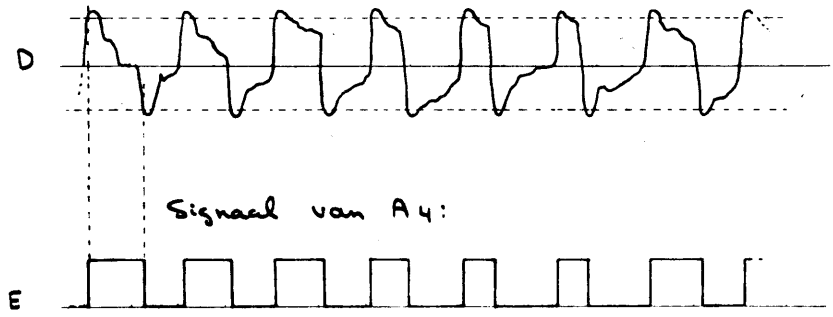


fig. 1

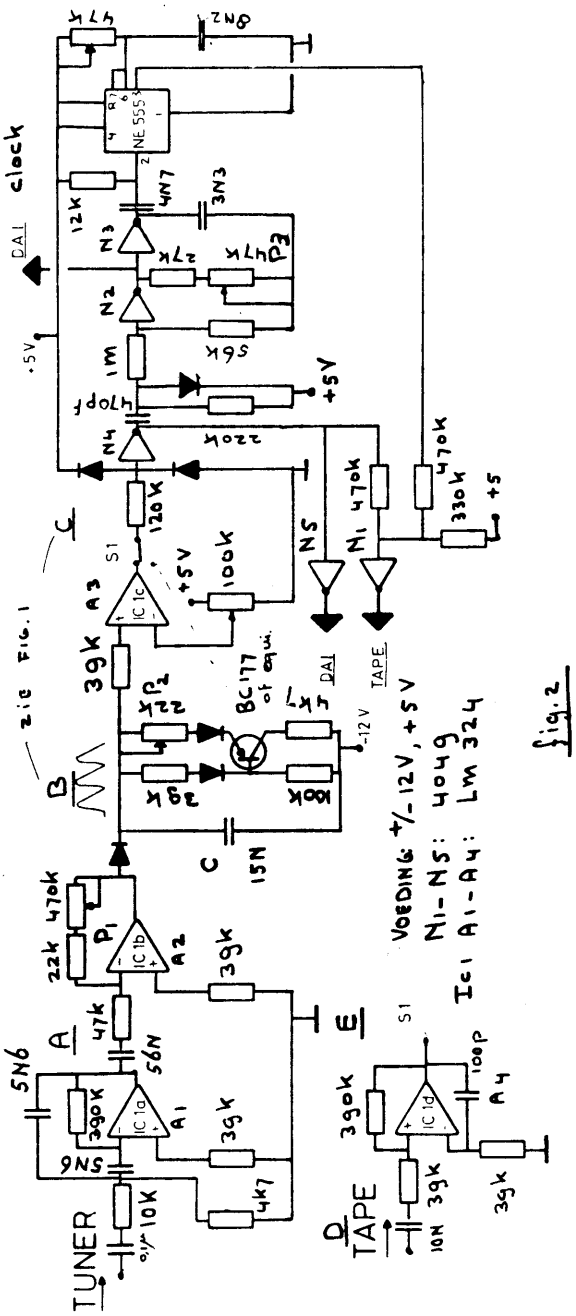
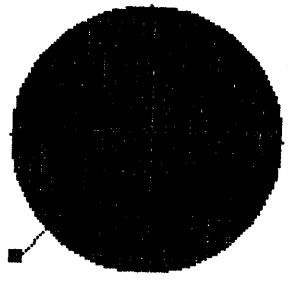


fig. 2

```
T
10    REM ************************************************
20    REM *    Demonstratie programma voor het weer-      *
30    REM *       geven van foto's afkomstig van          *
40    REM *              weersatellieten                  *
45    REM *                                               *
50    REM *          (c)   H. Bakker   1982               *
60    REM *                085-451394                     *
70    REM ************************************************
75    REM * Een paar gedeelten van dit DEMO"programma     *
77    REM * gebruiken FGT (hoofd+kleine letters )         *
78    REM * FGT is being used in this program !!          *
79    REM ************************************************
80    PRINT CHR$(12):MODE 0:COLORT 8 0 8 0:PRINT :CLEAR 5000:DIM A%(100.0),F(16.0):COLORG 1 5 10 15
90    PRINT "  Weergave van weersatellietfoto's via de DAI computer"
100   PRINT :PRINT :ENVELOPE 1 1,10;3,20;5,30;7,40;9,50;11,60;13,70;15,100;
110   PRINT " Er zijn minstens vier weersatellieten, die rond de aarde"
120   PRINT "draaien met een omlooptijd van ongeveer 100 minuten."
130   PRINT "Na iedere omwenteling is zo'n weersatelliet ongeveer 25 gra-"
140   PRINT "den in westelijke richting verschoven en zal daardoor eens"
150   PRINT "in de 12 uur over W. Europa 'vliegen'. Aangezien weer-"
160   PRINT "satellieten een baan om noord- en zuidpool volgen, zal een"
170   PRINT "satelliet eens per 12 uur van noord naar zuid, en de vol-"
180   PRINT "gende keer van zuid naar noord overvliegen."
190   PRINT :PRINT " Dit brengt ons bij het eerste probleem : wanneer een satel-"
200   PRINT "liet richting zuid vliegt, is het beeld in orde (noord-"
210   PRINT "pool 'boven'), maar in het andere geval moeten we bij het"
220   PRINT "weergeven van de foto het beeld 'omdraaien'.":PRINT
230   PRINT "Weersatellieten maken eigenlijk geen foto's van de aarde,"
240   PRINT "maar 'scannen' de aarde lijn voor lijn. Men kan het verge-"
250   PRINT "lijken met de manier, waarop een televisiebeeld is opge-"
260   PRINT "bouwd.";:INPUT "   (return)";D$:PRINT CHR$(12)
270   PRINT "Deze lijnen worden continu naar de aarde gezonden in de"
280   PRINT "vorm van een 2400 Herz toon, welke in amplitude is gemodu-"
290   PRINT "leerd met het videosignaal. Kleine amplitude betekent"
300   PRINT "lichte kleur, grote amplitude geeft de kleur zwart weer.":PRINT
310   PRINT "In het algemeen zullen de zee en het land een donkere kleur"
320   PRINT "geven (behalve sneeuw op b.v. bergen), en zijn wolken"
330   PRINT "zichtbaar als witte of grijze vlekken.":PRINT
331   PRINT "Het satellietsignaal wordt tijdens ontvangst eerst omgezet"
332   PRINT "naar een blokspanning met variabele lengte (duty cycle), en"
333   PRINT "wordt daarna opgenomen op de band. Dit omzetten naar een"
334   PRINT "blokspanning vermindert het effekt van drop-out's.":PRINT
340   PRINT "Aan het begin van iedere lijn stuurt de satelliet heel "
350   PRINT "even een andere toon, waarop de weergave apparatuur rea-"
360   PRINT "geert met het beginnen op een nieuwe regel (vergelijk RE-"
370   PRINT "TURN & LINE FEED)."
380   PRINT "Goed, laten we aannemen dat we erin zijn geslaagd met een "
390   PRINT "ontvanger het signaal van een weersatelliet hoorbaar te ma-"
395   PRINT "ken."
```



```
400  PRINT " Dit geluid klinkt ongeveer zo....: (return)";
410  FOR X%=1 TO 100:A%(X%)=SQR((10000.0-(100.0-X%)*(100.0-X%)-X%*X%)/2.0):NEXT X%:INPUT D$
420  MODE 6A:Q%=XMAX/2.0:PRINT CHR$(12):COLORT 1 8 8 0
430  PRINT :PRINT "              ** GELUID OP KANAAL 1 & 2 **"
440  FOR X%=1 TO 50:DOT RND(XMAX),RND(YMAX) 15:NEXT X%
450  FOR X%=1.0 TO 100.0:SOUND 1 1 10 3 FREQ(50.0*X%)
460  DRAW Q%-A%(X%),75+X% Q%+A%(X%),75+X% 10:NEXT X%
470  FOR X%=2 TO 100:Z%=15:IF X%=75 THEN GOSUB 580
480  SOUND 1 1 10 1 FREQ(2400.0):FILL Q%-A%(X%)-30.0,X%+42+X%/3 Q%-A%(X%)-5.0,X%+73+X%/4 1
490  FILL Q%-A%(X%)-10.0,X%+X%/3+58 Q%-A%(X%)-14.0,X%+X%/3+62 5
500  IF X%=25.0 THEN GOSUB 550
510  DRAW Q%-A%(X%)-10.0,X%+X%/3+60 Q%-A%(X%)-1.0,X%+75 Z%
520  IF Z%=15.0 THEN WAIT TIME 8:Z%=1:GOTO 510
530  SOUND 1 1 8 0 FREQ(1300.0):NEXT X%:SOUND OFF
540  WAIT TIME 100:GOTO 620
550  PRINT :PRINT " Dit is de 2400 Herz toon waarvan de amplitude veranderd"
560  PRINT "overeenkomstig de beeldinformatie (Amplitude geModuleerd)."
570  WAIT TIME 250:RETURN
580  PRINT CHR$(12):PRINT " Dit is een toon van een andere frequentie, welke het begin"
590  PRINT "van iedere video lijn aangeeft (trigger burst)."
600  PRINT "Deze toon is kort hoorbaar aan het begin van iedere lijn.";
610  WAIT TIME 250:RETURN
620  REM
621  FOR X%=1.0 TO 16.0:F(X%)=SIN((X%*PI)/8.0):SOUND 1 1 15 3 FREQ((17.0-X%)*100.0)
622  SOUND 2 1 15 3 FREQ(X%*100.0):NEXT X%:Q%=XMAX/2.0:SOUND OFF
625  Q=XMAX:SP=6.0:CC=15.0:X=1.0
626  MODE 5:ENVELOPE 0 15
630  GOSUB 2000
640  C=GETC:WAIT TIME 10
650  C=GETC:IF C=0.0 THEN 650
660  PRINT CHR$(12):COLORT 8 0 8 0:MODE 0
670  PRINT "De computer ontvangt dus bij weergave van de band +/- 2400"
680  PRINT "maal p/sec. een puls van wisselende lengte.  De taak van de"
690  PRINT "computer is te herkennen wanneer de puls begint (herkennen"
700  PRINT "van de positieve flank) en dan te kijken hoelang de puls"
710  PRINT "hoog blijft. Afhankelijk van de pulslengte vormt de computer"
720  PRINT "een zwarte, donkergrijze, lichtgrijze of witte punt."
730  PRINT :PRINT "Het zal duidelijk zijn dat dit vormen van de punten niet"
740  PRINT "te veel tijd in beslag mag nemen omdat de computer klaar"
750  PRINT "moet zijn als de volgende puls begint. Daarom zal het niet"
760  PRINT "eenvoudig zijn een machinetaal programma te schrijven wat"
770  PRINT "zo snel is dat de computer gebruik kan maken van de 16 kleu-"
780  PRINT "ren mode. Maar zelfs met de vier kleuren die wel mogelijk"
790  PRINT "zijn, zien de resultaten er best acceptabel uit.":PRINT
800  PRINT "Bij het machinetaal programma hoort een kort BASIC progra-"
810  PRINT "ma. In dit programma wordt u gevraagt hoeveel punten van"
820  PRINT "iedere beeldlijn u wilt overslaan (SKIP LEFT PART) en ver-"
830  PRINT "volgens dient u in te voeren van welk type (Amerikaanse of"
840  PRINT "Russische satelliet) u een foto wilt weergeven."
850  PRINT :PRINT :INPUT "    (return)";D$:PRINT CHR$(12)
855  PRINT "Hierna worden door dit BASIC programma diverse va-"
860  PRINT "riabelen ten behoeve van het machinetaalprogramma gepoked."
```

```
870  PRINT "Dan gaat de computer naar MODE 6 en begint het machinetaal-"
880  PRINT "programma met het starten van de bandrecorder  en wacht"
890  PRINT "even tot de bandrecorder op snelheid is gekomen. Na ongeveer"
900  PRINT "1 seconde begint het weergeven nadat de computer een juist"
910  PRINT "triggersignaal heeft herkent."
920  PRINT "Als het beeld wat verschijnt te licht of te donker is, dan"
930  PRINT "kunt u tijdens het weergeven met de toets CURSOR LEFT het"
940  PRINT "beeld donkerder, en met CURSOR RIGHT het beeld lichter"
950  PRINT "krijgen."
960  PRINT "Het indrukken van de TAB toets stopt het machinetaal pro-"
970  PRINT "gramma. Druk niet op de BREAK toets, want dit stopt wel"
980  PRINT "de bandrecorder, maar niet het programma."
990  PRINT "Als het beeld vol is, stopt de band en springt het program-"
991  PRINT "ma terug naar BASIC."
992  PRINT "De klokpuls moet u aansluiten op BIT 0 van PORT 0 (pin 14"
993  PRINT "van de DCE bus) en het video signaal komt binnen op pin"
994  PRINT "16 van de DCE bus. Verbindt de andere 6 bits van port 0 met"
995  PRINT "nul via een weerstand van 1000 ohm.":PRINT :INPUT "    (return)";D$
1000 MODE 6:FOR X=1.0 TO 100.0:SOUND 1 1 13 3 FREQ(50.0*X):NEXT X:COLORG 13 9 6 2
1010 A$="* WEERSATELLIET ONTVANGST *":X=1.0:Y=200.0:SP=6.0:CC=2.0:DF=1.1:FF=1.0:FC=2.0:GOSUB 5000
1020 A$="Voor informatie over ontvangers :":X=10.0:Y=150.0:CC=9.0:DF=0.9:FF=1.0:GOSUB 5000
1030 X=20.0:CC=2.0:A$="fa.  MECOM":Y=130.0:GOSUB 5000
1040 A$="     Coendersstraat   24":Y=110.0:GOSUB 5000
1050 A$="     9780 AA  Bedum":Y=90.0:GOSUB 5000
1060 A$="Tel: 05900 - 14390":Y=70.0:GOSUB 5000:FF=0.0
1070 X=10.0:CC=6.0:A$=" of : H. Bakker  Rijnstraat 28  6811 EW  Arnhem":Y=40.0:GOSUB 5000
1080 A$="      Tel: 085 - 451394":Y=30.0:GOSUB 5000
1090 A$="=":FOR X=1.0 TO XMAX-6.0 STEP 10.0:Y=1.0:GOSUB 5000:Y=YMAX-10.0:GOSUB 5000:NEXT
1091 A$="Bedankt voor uw aandacht - Tot ziens":X=70.0:Y=10.0:CC=6.0:FF=1.0:GOSUB 5000
1095 C=GETC:WAIT TIME 20:C=GETC
1098 C=GETC:IF C=0.0 THEN 1098
1099 END
2000 COLORT 0 8 8 0
2001 Y=245.0:A$="Kleine amplitude geeft witte kleur ,":GOSUB 5000
2002 Y=230.0:A$="Grote amplitude geeft zwarte kleur :":GOSUB 5000
2003 Y=145.0:A$="Klok voor de synchronisatie :":GOSUB 5000
2004 Y=95.0:A$="Signaal afkomstig van Tape recorder :":GOSUB 5000
2005 Y=50.0:A$="Overeenkomstige kleur van de punt op het scherm :":GOSUB 5000
2008 Y=2.0:A$="( DRUK OP 'RETURN' VOOR VERVOLG )":X=50.0:CC=8.0:GOSUB 5000
2020 DRAW 0,195 XMAX,195 10:R%=19:GOSUB 5000
2030 FOR X%=-9.0 TO 0:P=ABS(2.0*X%)-9.0
2035 FILL R%+15,20 R%+43,40 X%+13.0:SOUND 1 0 ABS(X%) 1 FREQ(2400.0)
2040 DRAW R%+2,115 R%+16,115 15:DRAW R%+16,115 R%+16,135 15:DRAW R%+16,135 R%+32,135 15
2042 DRAW R%+32,135 R%+32,115 15:DRAW R%+4+P,65 R%+16,65 15:SOUND OFF
2044 DRAW R%+16,65 R%+16,85 15:DRAW R%+16,85 R%+32+P,85 15
2045 DRAW R%+32+P,85 R%+32+P,65 15
2050 FOR Y%=1.0 TO 16.0:R%=15*X%*2+2*Y%+Q%+120
2070 DOT R%+X%+5,F(Y%)*X%*3.0+195.0 15
2080 NEXT:NEXT
2999 WAIT TIME 200:RETURN
5000 SOUND 2 1 10 3 FREQ(10.0*Y+100.0):C=FC**40+CC:F=FF**80+PF**40+ZF**20+VF**10+VF**10+DF
5002 POKE #2F0,C:POKE #2F1,F
5004 POKE #2F2,X MOD 256:POKE #2F3,X/256:POKE #2F4,Y
5006 POKE #2F5,SP:POKE #2F6,ID
5008 CALLM #300,A$:SOUND OFF :RETURN
```

DAI PERSONAL COMPUTER SCHEMATICS

Sheet 1 CPU
Sheet 2 RAM DRIVE
Sheet 3 RAM
Sheet 4 TIMING
Sheet 5 ROM + I/O
Sheet 6 MAIN BOARD GENERAL LAY-OUT
Sheet 7 SOUND
Sheet 8 PADDLE
Sheet 9 B/W TV CARD
Sheet 10 PSU
Sheet 11 TIMING
Sheet 12 PAL COLOUR CARD LAY-OUT
Sheet 13 PAL COLOUR CARD
Sheet 14 RGB CARD LAY-OUT
Sheet 15 RGB CARD

Now available from your club : the complete DAIpc schematics,
drawn by Mr Boerrigter and co during many,many sleepeless nights...
Most sheets measure about 40cm x 60 cm,in professional styling.
order from :

DAInamic
Heide 4
3171 WESTMEERBEEK
BELGIUM

price of the complete package : 850 Bfr including mailing and VAT.
This hardware information is available for personal use only !

How to order DAInamic software :
send cheque in belgian francs or cash together with your order to
DAInamic Heide 4 3171 WESTMEERBEEK BELGIUM.

Contact address for contribution,membership,backnumbers,the best of
DAInamic 80/81 :
Bruno Van Rompaey Bovenbosstraat 4 3044 HAASRODE BELGIUM

"The BEST of DAInamic" will be available in august 82. You can order
now all programs that have been published in these issues :
"T80/81" : price : 850 Bfr. (57 programs on cassette/DCR) see contents p. 101

Also available on tape : the 23 programs published in this issue:
save yourself hours of typing for 500 Bfr.
name of the package : "N10".