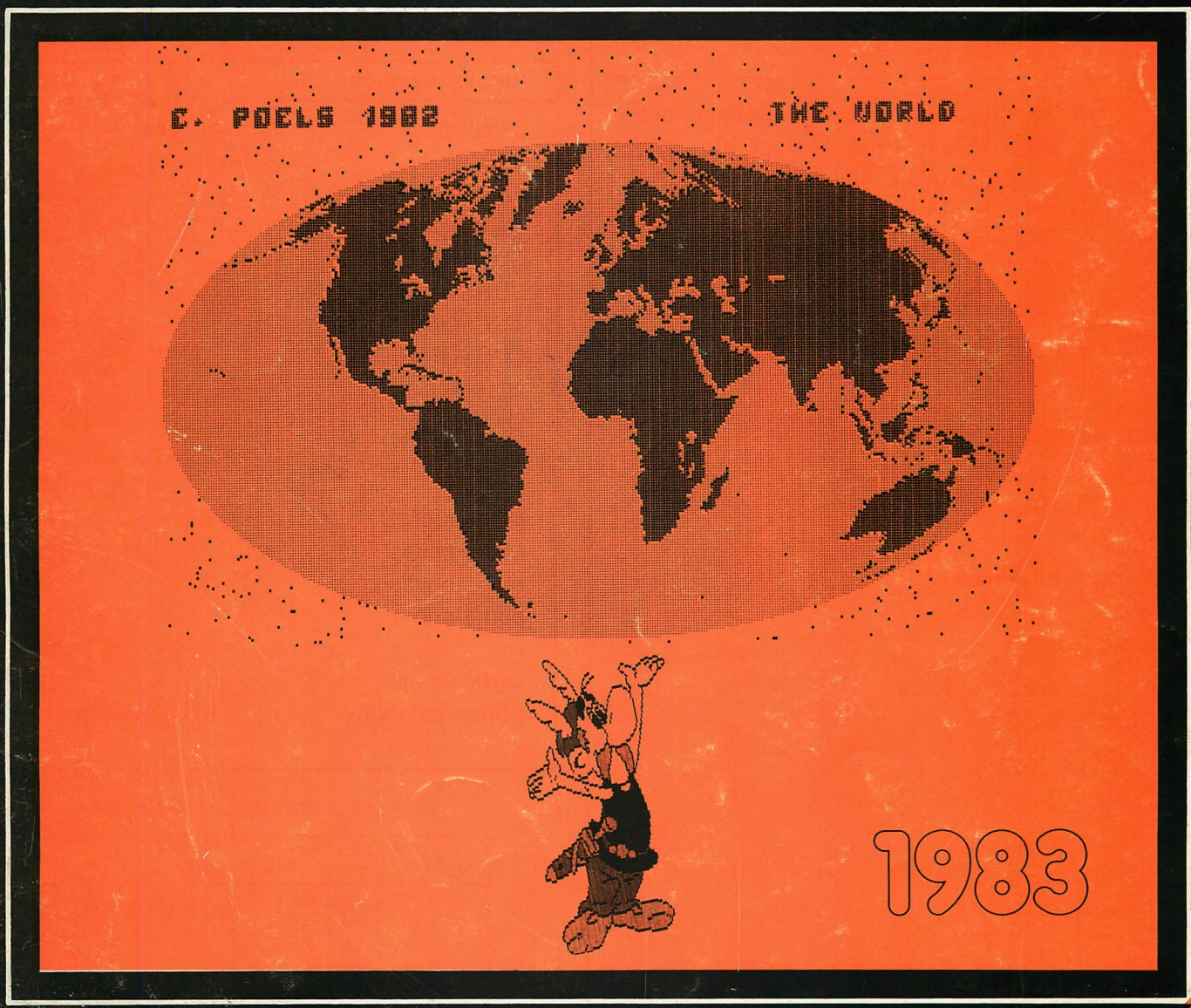
# DAI NAMIC

14

tweemaandelijks tijdschrift januari - februari 1983




## personal computer users club

een uitgave van dainamic v.z.w.
verantw. uitgever w. hermans, heide 4 - 3171 westmeerbeek

International

## COLOFON

DAInamic verschijnt tweemaandelijks.
Abonnementsprijs is inbegrepen in de jaarlijkse contributie .
Bij toetreding worden de verschenen nummers van de jaargang toegezonden.
DAInamic redactie :

Dirk Bonné
Freddy De Raedt
Wilfried Hermans
René Rens
Jos Schepens
Roger Theeuws
Bruno Van Rompaey
Jef Verwimp

Vormgeving : Ludo Van Mechelen.

---

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

---

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans
Heide 4
B 3171 Westmeerbeek
België
tel. : 016/69.86.23

Kredietbank Westmeerbeek
nr. 406-3016141-33
BTW : 420.840.834

Lidgelden

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

# DAINAMIC

PERSONAL COMPUTER USERS CLUB

| 4 | | 3 | | 2 | | 1 | |
|---|---|---|---|---|---|---|---|
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 1 | 4096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16384 | 4 | 1024 | 4 | 64 | 4 | 4 |
| 5 | 20480 | 5 | 1280 | 5 | 80 | 5 | 5 |
| 6 | 24576 | 6 | 1536 | 6 | 96 | 6 | 6 |
| 7 | 28672 | 7 | 1792 | 7 | 112 | 7 | 7 |
| 8 | 32768 | 8 | 2048 | 8 | 128 | 8 | 8 |
| 9 | 36864 | 9 | 2304 | 9 | 144 | 9 | 9 |
| A | 40960 | A | 2560 | A | 160 | A | 10 |
| B | 45056 | B | 2816 | B | 176 | B | 11 |
| C | 49152 | C | 3072 | C | 192 | C | 12 |
| D | 53248 | D | 3328 | D | 208 | D | 13 |
| E | 54344 | E | 3584 | E | 224 | E | 14 |
| F | 61440 | F | 3840 | F | 240 | F | 15 |

**belangrijke ASCII-waarden in DAIpc**

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

ASCII - HEX - ASCII CONVERSION TABLE

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

Westmeerbeek feb. 1983

Beste leden,

Traditiegetrouw komt het eerste nummer van onze nieuwe jaargang wat later in uw bus vallen. Spelbreker was deze keer een foutief banknummer in onze vorige publicatie. De juiste gegevens vindt U op pagina 4.
Vanwege het enorme aanbod aan materiaal was de selectie van de artikels deze keer niet eenvoudig. Wij vragen begrip voor deze (overigens kerngezonde) toestand: heeft U iets ingestuurd en vindt U dit nog niet terug in ons tijdschrift, dan komt uw bijdrage waarschijnlijk een volgende keer aan de beurt. Wij proberen telkens een evenwichtig nummer te brengen: bijdragen voor beginners, gevorderden, machinetaalspecialisten en al die andere activiteiten waarvoor een personal computer kan ingezet worden.
INDATA laat met de beloofde realisaties op zich wachten. Gelukkig tellen we onder onze leden een aantal erg actieve mensen: we vermoeden dat de realisatie van Kenneth Gooswit wel eens zou kunnen uitgroeien tot een standaard periferie voor DAIpc. Indien de testen meevallen zal DAInamic deze discdrives zeker supporteren, vooral omdat alle bestaande software meteen op dit formaat zal kunnen geleverd worden. Specificaties van KEN-DOS vindt U op bladzijde 56-58.
De volgende DAInamic bijeenkomst gaat door in TONGELSBOS (Bosstraat 2 3180 WESTERLO-BELGIE) op zaterdag 9 april 1983 van 10.00 tot 18.00 Hr.
Op de agenda volgende activiteiten:

* tweedehandsmarkt : iedereen mag te koop aanbieden wat hij graag kwijt wil.
* tijdschriften-info : van de meeste computertijdschriften hebben we ondertussen een drietal jaargangen. Deze kunnen geraadpleegd worden, we hebben ook een paar copieerapparaten gehuurd i.v.m. uitgaven die niet meer of moeilijk te verkrijgen zijn.
* demonstratie van 3-dimensionele beelden op DAI. Een realisatie van Jan Roelants, echt indrukwekkend.(zie ook p.64 en vlgd.)
* lezingen en demonstraties : deze worden ter plaatse aangekondigd.
* iedereen mag zijn hard|software realisaties komen voorstellen, gelieve wel eerst contact te nemen i.v.m. de indeling v.d. beschikbare ruimte.
* handelaars : we hebben een paar lokalen voorzien waar handelaars hun gamma kunnen voorstellen; geïnteresseerden gelieve contact te nemen.

Onze vertaaldienst wordt gereorganiseerd : wij zoeken mensen die artikels en|of programma's kunnen vertalen

Veel lees- en tikgenot, tot 9 april
W.Hermans

Dear members,
As usual the first edition of our magazine is rather late. The reason of this is a wrong banc number in Newsletter 13. You will find the right information for subscription op page 4.
The job of editing this issue was not easy : we receive a lot of articles and programs. If you do not find your contribution in this issue, please don't be disappointed, there are a lot of magazines to come.
We are very proud to announce the realisation of Kenneth Gooswit : Floppy-drives with all the specifications we have been waiting for : random access, sequential access, very large capacity. More about KEN-DOS on page 56-59.
On 9 april 1983, we have our next meeting. The place is : TONGELSBOS, Bosstraat 2 3180 WESTERLO-BELGIUM. The meeting is open from 10.00 to 18.00 Hr.
On the agenda of the meeting :

* secondhand-bargain market : anybody can offer hardware he wants to sell.
* magazine-info : from most of the computermagazines, we have issues of the last 3 years. You can read these magazines. We also will hire a few copying machines for the magazines that are no more available from the editors.
* demonstration of 3-dimensional pictures on DAI by Jan Roelants. You want believe it until you see this with your own eyes! See article and program in this issue on p. 64.
* lectures and demonstrations : these will be announced on the meeting.
* everyone can bring his machine to show hard- and software realisations. Please contact us, in order to arrange the available space.
* dealers : we reserve a few rooms for dealers to show their range of periferals for DAIpc. dealers should contact us before march 15.

We are arranging our translation service : please contact us if you can translate articles or programs.

We hope you enjoy newsletter 14,
see you on 9 april.
W.Hermans

# INHOUD — CONTENTS



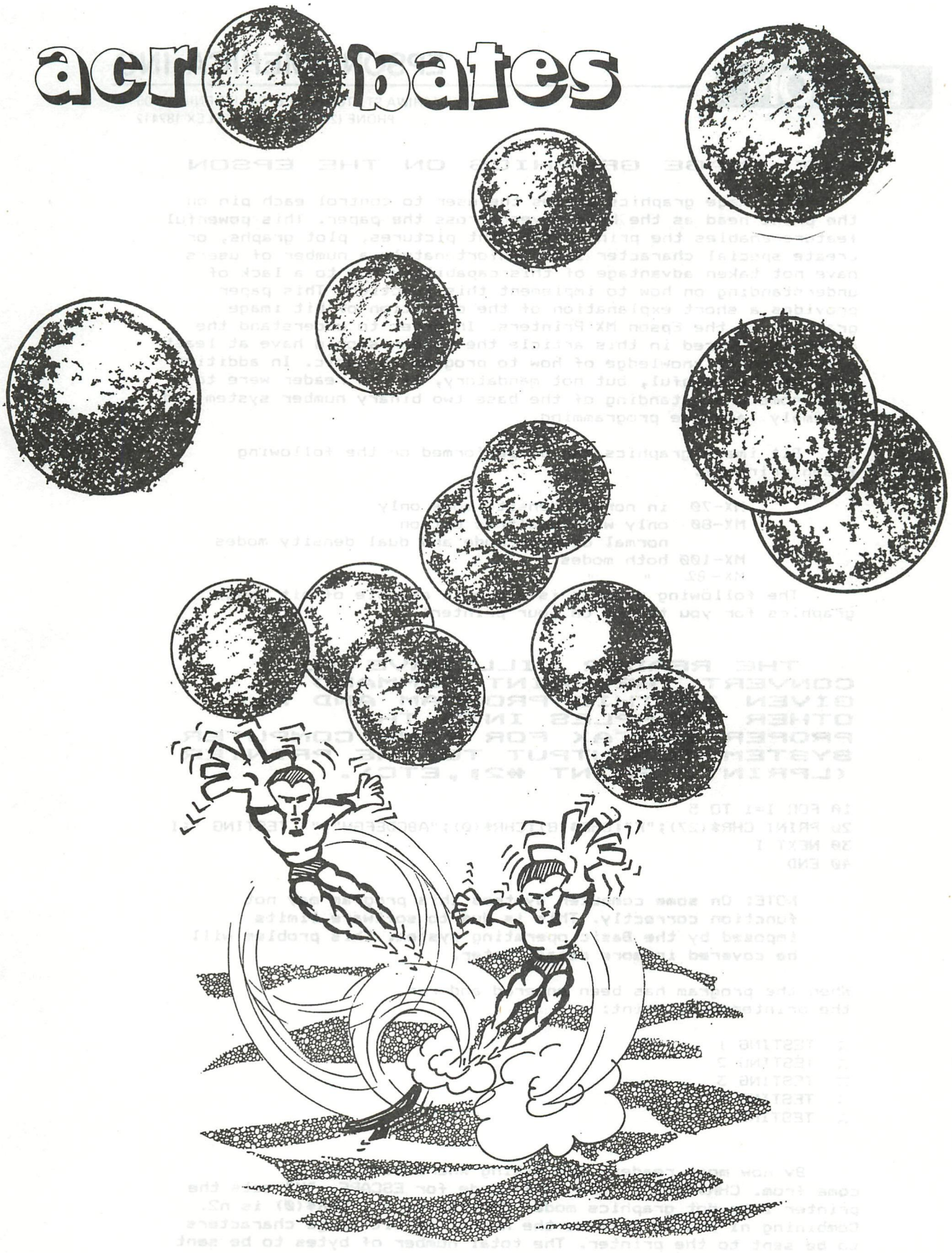
DAInamic subscription rates :

| | |
|---|---|
| Benelux | : 900 Bfr |
| Europe | : 1000 Bfr |
| Outside Europe (Air Mail) | : 1400 Bfr |

pay to : Dainamic SUBSCRIPTIONS
B.Van rompaey
Bovenbosstraat 4
3044 HAASRODE-BELGIUM

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

# acrobates



```
*****************************************
*                                       *
*  NOW AVAILABLE :                      *
*  ---------------                      *
*  ACROBATES : a fascinating new game,  *
*  totally in machine language, (over   *
*  7K of code). ONE/TWO PLAYER option,  *
*  BONUS SCORE, MUSICAL TUNES ...       *
*                                       *
*  PRICE : only 600 Bfr (750 on DCR)    *
*                                       *
*****************************************
```

EPSON

EPSON AMERICA, INC.

3415 KASHIWA ST., TORRANCE, CALIFORNIA 90505
PHONE (213) 539-9140 TELEX 182412

# BIT IMAGE GRAPHICS ON THE EPSON

Bit image graphics allows the user to control each pin on the print head as the head moves across the paper. This powerful feature enables the printer to print pictures, plot graphs, or create special character sets. Unfortunately a number of users have not taken advantage of this capability due to a lack of understanding on how to implement this function. This paper provides a short explanation of the operation of bit image graphics on the Epson MX Printers. In order to understand the material covered in this article the reader should have at least a fundamental knowledge of how to program in Basic. In addition it would be helpful, but not mandatory, if the reader were to have some understanding of the base two binary number system and assembly language programming.

Bit image graphics can be performed on the following Epson printers:

MX-70 in normal density mode only
MX-80 only with Graftrax option
normal density mode and dual density modes
MX-100 both modes
MX-82 " "

The following program is a simple example of bit image graphics for you to try on your printer:

**THE READER WILL HAVE TO CONVERT THE PRINT COMMAND GIVEN IN THIS PROGRAM AND ALL OTHER EXAMPLES INTO THE PROPER SYNTAX FOR YOUR COMPUTER SYSTEM TO OUTPUT TO THE PRINTER (LPRINT, PRINT #2;,ETC).**

```
10 FOR I=1 TO 5
20 PRINT CHR$(27);"K";CHR$(8);CHR$(0);"ABCDEFGH";"  TESTING ";I
30 NEXT I
40 END
```

**NOTE: On some computer systems this program may not function correctly. This is due to software limits imposed by the Basic operating system. This problem will be covered in more detail later.**

When the program has been entered and run the printer will print:

TESTING 1
TESTING 2
TESTING 3
TESTING 4
TESTING 5

By now most readers are asking where does the ABCDEFGH come from. CHR$(27) is the ASCII code for ESCAPE, "K" sets the printer into dot graphics mode, CHR$(8) is n1, CHR$(0) is n2. Combining n1 and n2 defines the number of bit image characters to be sent to the printer. The total number of bytes to be sent is figured in the following example :

n1 + (n2 * 256).

In the above program n1=8 and n2=0, thus 8+(256*0)=8. Had n2 = 1 then the printer would have been programed to receive 8+(256*1)=264 characters. In our example, the printer was programed to receive 8 characters.

Once the printer has been set into graphics mode, the next 8 characters sent to the printer will be printed as bit graphics. The characters *ABCDEFGH* are now printed in bit image mode rather then as letters.

When the computer sends a character to the printer, it is sent as a binary number. The printer then converts the number into a predefined symbol. The ASCII code defines each binary number up to the value 127. Each of these numbers has a predefined function or printable symbol as shown in the ASCII table in your Epson manual. This standard is followed by most computer and printer manufacturers.

When printing in bit image mode the printer does **NOT** print the letters sent to the printer, rather it prints the binary value of the ASCII code of the letter sent. The ASCII value of "A" is 65 decimal, Ø1ØØØØØ1 base 2. Compare the printed image with the ASCII table below.

| VALUE | A | B | C | D | E | F | G | H | |
|---|---|---|---|---|---|---|---|---|---|
| 128 BIT 8 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | top pin of the print |
| 64 BIT 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | head |
| 32 BIT 6 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | |
| 16 BIT 5 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | |
| 8 BIT 4 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | 1 | |
| 4 BIT 3 | Ø | Ø | Ø | 1 | 1 | 1 | 1 | Ø | |
| 2 BIT 2 | Ø | 1 | 1 | Ø | Ø | 1 | 1 | Ø | |
| 1 BIT 1 | 1 | Ø | 1 | Ø | 1 | Ø | 1 | Ø | bottom pin |

The Basic function CHR$(___) *should* allow the computer to send any number to the printer. On many computers there are certain numbers that are not transmitted to the printer correctly.

The TRS-8Ø Model I Computers will **NOT** run correctly on the above program. This is because the Model 1 cannot send a decimal Ø as shown in line 2Ø ( CHR$(Ø) ). Also the Model 1 can not send the decimal codes 1Ø, 11, or 12 from the Basic CHR$(___) command.

The TRS-8Ø Model III cannot send decimal 1Ø or 12 from the Basic CHR$(___) command.

The TRS-8Ø Color Computer cannot send any number above the value 127.

Apple II cannot send a decimal 9 or 13 nor any other number from 128 to 255. A decimal 9 is stopped by the firmware in the interface card. A decimal 13 (a CR) is sent to the printer, however the firmware then sends a decimal 1Ø (a LF). Numbers above 127 are not sent due to firmware limits within the computer.

Other computers MAY have some type of oddity in the operating system that stops a user from sending some numbers to the printer. Most problems occur in the control codes (numbers Ø to 31). A few may have an upper limit on the numbers sent, like the Apple or the TRS Color Computer. Some experimentation will be necessary to determine your system's limits.

A possible solution to the limits imposed by the system would be to write your own printer driver routine in assemply language, or use a short basic subprogram that talks to the printer port directly. The following subroutines MAY help to avoid problems:

**TRS-8Ø MODEL I**

```
1Ø IF PEEK(14312) <> 63 THEN GOTO 1Ø :REM WAIT FOR PRINTER TO BE
   READY
2Ø POKE 14312,X :REM SEND THE VALUE X TO THE PRINTER
3Ø IF PEEK(14312) <> 63 THEN GOTO 3Ø :REM SAME AS LINE 1Ø
4Ø RETURN
```

**TRS-8Ø MODEL III**

```
1Ø IF INP(251) <> 63 THEN GOTO 1Ø :REM INP(248) ALSO WORKS
2Ø OUT 251,X :REM IF THAT DOES NOT WORK TRY  OUT 248,X
3Ø IF INP(251) <> 63 THEN GOTO 3Ø :REM INP(248) ALSO WORKS
4Ø RETURN
```

## APPLE II WITH APPLESOFT BASIC

WITH THE EPSON 8131 PARALLEL INTERFACE BOARD

I do not know if this routine will work with other boards, if not, contact the manufacture of your board for PEEK and POKE locations.

```
10 IF PEEK(49601) >127 GOTO 10 :REM WAIT FOR PRINTER TO BE READY
20 POKE 49296,X                :REM SEND X TO THE PRINTER
30 IF PEEK(49601) >127 GOTO 30 :REM WAIT FOR PRINTER TO BE READY
40 RETURN
```

## ATARI

```
 1 OPEN #1,8,0,"P:"
10 PRINT #1;CHR$(X);
40 RETURN
```

## ALL OTHERS

```
10 LPRINT CHR$(X);
40 RETURN
```

This routine may not solve many of the limits imposed by your system. See your computer manual for the address of your printer port. IBM users should add the following command:

```
2 WIDTH "LPT1:",255
```

The next step is to combine your subprogram with a main program for printing graphics.

```
  5 GOTO 100
 10 YOUR SYSTEM'S SUBROUTINE
    . . .
100 REM GRAPHICS PROGRAM
105 PRINT CHR$(27);"K"; :REM SENDS AN ESCAPE FOLLOWED BY A K
110 LET X=60:GOSUB 10 :REM SEND n1
120 LET X= 0:GOSUB 10 :REM SEND n2
130 REM n1 + (256 * n2) = the total number of graphics
    characters the printer will be printing.
140 FOR I= 1 TO 60
150 LET X=65 :GOSUB 10
160 NEXT I
170 PRINT "  X = ";X
180 END
```

This program will print two parallel lines as follows:

```
=======   X = 65
```

Apple and TRS Color Computer owners will find that their system will not be able to send any number above 127 correctly. This problem is a firmware problem and cannot be corrected by any subroutine.

Some changes you might wish to try are:

```
100 FOR X2 = 0 TO 255:REM FOR X2=0 TO 127 with the Apple or
150 X=X2 : GOSUB 10  :REM              the TRS Color Computer.
175 NEXT X2
```

This prints all possible combinations of graphics characters. When the program has been run a clear pattern will appear. To help the reader understand this pattern see the figure below:

```
          TOP
          PIN   VALUE                  In this example VALUE is
                                       the pin number.
PRINT      *      128
           *       64                   In selecting what pins to
HEAD       *       32                   fire, add the values of the
           *       16                   pins to be fired.
---->      *        8
           *        4                   The pin on the bottom can not
           *        2                   be addressed when in bit
           *        1                   graphics mode.
           *      not used
```

```
If X = 0  then no pins will fire.
If X = 1  then pin 1 will fire.
If X = 2  then pin 2 will fire.
If X = 3  then pins 1 and 2 will fire.
If X = 4  then pin 4 will fire.
If X = 5  then pins 1 and 4 will fire.
If X = 65 then pins 64 and 1 will fire.
```

In order to get all 8 pins to fire X must = 255.

```
255 = pin 128 + pin 64 + pin 32 + pin 16 + pin 8 + pin 4 + pin 2
    + pin 1
```

When you want to draw a shape, the shape must be broken into numbers.

```
PIN
128    !_!_!_!_!_!_!_!
 64    !*!*!*!*!*!*!*!
 32    !*!_!_!_!_!_!*!
 16    !*!_!_!_!_!_!*!
  8    !*!_!_!_!_!_!*!
  4    !*!_!_!_!_!_!*!
  2    !*!*!*!*!*!*!*!
  1    !_!_!_!_!_!_!_!
  _
       C C C C C C C
       0 0 0 0 0 0 0
       L L L L L L L
       1 2 3 4 5 6 7
```

On a sheet of graph paper I've drawn a box. When broken into numbers, column one's number is 64+32+16+8+4+2=126. Columns 2 through 6 add up to 64+2=66. Column 7 adds up the same as column 1.

Thus the data for this drawing is

```
DATA 126,66,66,66,66,66,126
```

To draw this shape add the subroutine for your computer in the first part of this section to the program below.

```
  5 GOTO 100
 10 YOUR SUBROUTINE
    . . .
 40 RETURN
100 REM PRINT A SHAPE
105 PRINT CHR$(27);"K";
110 X=7 :GOSUB 10
120 X=0 :GOSUB 10
130 FOR I = 1 TO 7
140 READ X
150 GOSUB 10
160 NEXT I
170 PRINT
180 DATA 126,66,66,66,66,66,126
190 END
```

When this program is run, it will print: □

By changing line 180 you can change the shape printed.

```
PIN
128   !_!_!_!_!_!_!_!
 64   !_!_!_!_!_!_!*!
 32   !_!_!_!_!_!*!_!
 16   !_!_!_!_!*!_!_!
  8   !*!*!*!*!_!_!_!
  4   !_!_!_!_!*!_!_!
  2   !_!_!_!_!_!*!_!
  1   !_!_!_!_!_!_!*!
  _    1 2 3 4 5 6 7
```

This shape becomes the following numbers:

```
180 DATA 8,8,8,8,20,34,65
```

Our new shape will look like this : ≺

The syntax for entering graphics is:

```
                                ESC  K      + n1      + n2
In basic this becomes:  PRINT CHR$(27);"K";CHR$(n1);CHR$(n2);
```

Line 105 of our program sends the ESC K, line 110 sends n1, and line 120 sends n2. For beginners it is best to use a range of 1 to 255 for n1 (On an APPLE II use 1 TO 127) and set n2 to 0. A beginner is less likely to have problems or crash his program if shorter lengths are used.

Graphics using ESC K is called normal-density bit image graphics. Using ESC L gives dual-density bit image graphics. The ESC L command packs more dots per inch in the horizontal dirrection (120 dots/inch compared to 60 dots/inch). This causes the figure to become compressed. To see the effect, change line 105 as follows:

```
105 PRINT CHR$(27);"L";
□ becomes ▯ and
≺ becomes ≺ .
```

In the dual-density mode, the printer can print a form of emphasized graphics. Make the following changes to this demo program:

```
105 PRINT CHR$(27);"L";
110 X=14 :GOSUB 10
155 GOSUB 10
```

Line 105 puts the printer in a dual-density mode, lines 110 and 120 program the printer to receive 14 graphic characters. Lines 150 and 155 together will print the same graphics character twice. The program originally had 7 characters to the image. With the changes made, the 7 characters were printed twice resulting in a total of 14 graphics characters being printed (see line 110).

```
□ becomes □ and
≺ becomes ≺ .
```

For practice try taking a sheet of graph paper, draw your own shapes, and put the shape into numbers in a DATA statement. Remember, each number controls one column and the program given uses seven colums. When you understand how to draw a shape try expanding on this program and converting it to print 20 columns.

When printing bit-image graphics, a single number controls 8 rows in one column. This type of printing is useful for your own character set (such as Greek), but is somewhat limited for doing drawings or logos on a printout. The solution to this problem is to print the image in several passes. Thus if you were printing a drawing in three sections, you would print the top first, then the middle, and the bottom last. One additional command that must be given to the printer is to change the line feed spacing so that there will be no gaps between the passes.

The command in basic is:

```
              ESC  A      +n
   PRINT CHR$(27);"A";CHR$(7);:REM FOR APPLE AND COLOR COMPUTERS
   PRINT CHR$(27);"A";CHR$(8);:REM ALL OTHERS
```

Once you are done printing graphics remember to restore the printer to the normal line spacing with the command:

```
   PRINT CHR$(27);"A";CHR$(12);
or PRINT CHR$(27);"2";
```

The following program prints a simple bar chart in bit image mode. This program was written for an Apple II computer, however, with a few minor changes, the program can be made to work on other systems. Such changes are: Delete lines 1 and 440 (PR#1 and PR#0 turn the printer on and off for the Apple). Place the subroutine for your computer at lines 10 through 40. All PRINT commands should be changed to LPRINT , PRINT #2 , or whatever your system uses to talk to the printer.

To change the bars in the graph, change lines 600 to 630 to reflect the data you wish to plot. Another change you might wish to make is to add a routine to change the scale of your program to match your data.

This program takes about 12 minutes to run on the Apple II, I know of no way to speed it up, other than compiling it or rewriting it in machine code.

```
  1  PR# 1
  2  PRINT : PRINT : PRINT
  5  GOTO 300
  8  REM  PRINTER DRIVER ROUTINE
 10  IF  PEEK (49601) > 127 GOTO 10
 20  POKE 49296,X
 30  IF  PEEK (49601) > 127 GOTO 30
 40  RETURN
 45  REM  PRINT ARRAY P(I+K)
 50  PRINT "    "; CHR$ (27);"K";
 60  X = 127: GOSUB 10
 70  X = 1: GOSUB 10
 80  FOR I = 1 TO 383:X = P(I + K): GOSUB 10: NEXT I
 90  PRINT : RETURN
 95  REM  PUT IN THE VERTICAL LINES EVERY 32 BYTES.
100  FOR I = 1 TO 383:P(I + K) = 0: NEXT I
110  FOR I = 1 TO 383 STEP 32:P(I + K) = 126: NEXT I
120  P(383 + K) = 126: RETURN
195  REM  PRESET ARRAY TO  L   AND PRINT
200  FOR I = 1 TO 383:P(I + K) = L: NEXT I
210  PRINT "     ";: GOSUB 50
220  RETURN
300  REM  BAR CHART (MAIN PROGRAM)
310  GOSUB 500
320  FOR Q = 1 TO 12
330  K = 383: PRINT "     ";: GOSUB 50
340  PRINT M$(Q);" ";:K = 0: GOSUB 100
350  V = A(Q): IF V > 383 THEN V = 383
360  FOR I = 1 TO V:P(I) = 126: NEXT I
370  GOSUB 50
380  K = 383: PRINT "     ";: GOSUB 50
390  NEXT Q
400  REM  DRAW LAST LINE AND FINISH
410  L = 64:K = 383: GOSUB 200
420  PRINT CHR$ (27);"2"
430  PRINT : PRINT : PRINT
440  PR# 0
450  END
500  REM  SETUP ARRAYS
510  DIM P(766),A(12),M$(12)
520  PRINT "MONTH                                    SALES"
530  PRINT  CHR$ (27);"A"; CHR$ (6)
540  L = 2:K = 383: GOSUB 200
550  FOR I = 1 TO 12: READ M$(I),A(I): NEXT I
560  GOSUB 100
570  RETURN
600  DATA  "JAN",70,"FEB",120,"MAR",180
610  DATA  "APR",210,"MAY",143,"JUN",250
620  DATA  "JUL",289,"AUG",310,"SEP",290
630  DATA  "OCT",330,"NOV",380,"DEC",340
```

For those of you who would like this program to print the chart in a darker print, add the following changes:

```
  3  PRINT  CHR$ (27);"E";
 50  PRINT "    "; CHR$ (27);"L";
 60  X = 0: GOSUB 10
 70  X = 3: GOSUB 10
 80  FOR I = 1 TO 383:X = P(I + K): GOSUB 10: GOSUB 10: NEXT I
 85  X=0 : GOSUB 10 : GOSUB 10
```

The changes listed will slow down the program by a factor of two.

LINES 10-40 Are the subroutine listed for your computer.

LINES 50-90 Send the array P(I+K) to the printer. If K is set to 0, the first 383 bytes of the array go to the printer. If K is set to 383, the last 383 bytes of the array go to the printer.

LINES 100-120 Place the vertical lines for our plot inside the array P(I+K).

LINES 200-220 Preset the array P(I+K) to the number L and when the array is preset, it is printed.

LINES 300-390 Are the main part of the program that prints the plot by calling the necessary subroutines.

LINES 400-450 Finish the program by drawing the last line in the plot and return line feeds to normal.

LINES 500-570 Setup the arrays, change line feeds to 6/72 of an inch, and setup to print the top line of the plot.

LINES 600-630 Contain the data for our plot.

ARRAY P(I+K) Is where the graphics data is stored before it is printed. If K = 0 , the first 383 bytes of the array are used. If K = 383 , the last 383 bytes of the array are used

ARRAY A(Q) Stores the numbers for the length of the bars on the graph.

ARRAY M$(Q) Contains the names of the months in our chart.

X Is the number sent direct to the printer.

I Is used in the FOR - NEXT loops.

K Selects what half of the array P(I+K) is used.

L Used for preseting the array P(I+K) to a value.

Q Another variable used in a FOR - NEXT loop.

V Used in a FOR - NEXT loop to set the data in the array P(I+K) to the length of the bar.

Changing the scale of the chart is easy, the only change that needs to be made is at line 350. Thus line 350 becomes:

```
350 V=A(Q)*384/scale : IF V > 383 THEN V = 383
```

Some values for scale are:

| scale | each division becomes |
|---|---|
| 120 | 10 |
| 240 | 20 |
| 300 | 25 |
| 600 | 50 |
| 1200 | 100 |

## OTHER INFORMATION

An interesting trick that can be used to print special characters from a basic program is as follows:

```
100 P$=CHR$(27)+"K"+CHR$(6)+CHR$(0)+CHR$(32)+CHR$(62)+
      CHR$(32)+CHR$(62)+CHR$(32)+CHR$(0)
110 REM This puts the character Pi in P$.
120 LPRINT P$;" = 3.141592634..."
130 LPRINT P$;" R ^ 2      XL = 2";P$;"FL"
```

P$ will print out as Pi in this program.

P$ = π

**THIS TRICK WILL NOT WORK ON THE TRS-80 MODEL I. THE MODEL I CAN NOT SEND A CHR$(0) THROUGH THE OPERATING SYSTEM.**

As for the other systems, when you use this trick, make sure you do not try to send a number through the CHR$ function that your system can not pass.

A solution to this problem is a simple subroutine as follows:

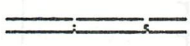
```
  5 GOTO 100
 10 YOUR SYSTEM'S SUBROUTINE
    . . .

 50 FOR I=1 TO LEN(A$)
 60 X=ASC(MID$(A$,I,1))
 70 GOSUB 10
 80 NEXT I
 90 RETURN
100 P$=CHR$(27)+"K"+CHR$(6)+CHR$(0)+CHR$(32)+CHR$(62)+
      CHR$(32)+CHR$(62)+CHR$(32)+CHR$(0)
110 REM Print P$
120 A$=P$ : GOSUB 50
130 LPRINT " = 3.141592634..."
140 END
```

π = 3.141592634 ...

A few final warnings about printing graphics. Some systems add a carriage return/line feed after 80, 127, 128, 132, 255, or 256 characters have been printed (the number depends on the operating system). If your system does that, you will find a small gap in your graphics printout. Should that occur either limit the length of your drawings or bypass your operating system in the computer.

For example you would see:



Or something similar to that.

Once the printer has been programed to receive a number of graphics characters, sending too few will make it appear as if the printer did not receive them. When the program is rerun several times the printer may start to print subsequent drawings on the same line. Sending more characters than the printer was programed for may put what looks like garbage at the end of the line. One thing to watch out for is if your Basic is adding or removing any data.

When printing long lines above 240 characters (80 characters on the MX-70), Basic's slow speed will cause the printer's buffer to fill, then stop, and print the first part of the line. Next the program will continue, home the head, and print the rest of the line, THIS IS NORMAL.

Printing graphics from a machine code program is desirible for 2 reasons.

(1) It runs faster.

(2) If it is written correctly you can bypass the computer's operating system and have less problems.

Machine language programs can print long lines in one pass if the program is operating fast enough (Basic takes several passes); however proper timing is critical. if the machine code program cannot meet the critical timing some data may be lost when the buffer fills. If you think that timing might be the source of your problem, try adding a short time delay to your assembly program to see if the missing data returns when using a delay.

Trying to set-up for more graphics characters than would normally fit on a line may cause the printer to enter an error condition.

It is best to print all your graphics with one programed setup per line; however this is not always possible. If you do not want a space between each setup, make your first setup a multiple of 6 if using ESC K, or a multiple of 12 if using ESC L.

Some practice and experimentation with the printer's graphics are required to obtian it's maximun capabilities, which are almost unlimited. The time spent in this effort will be amply repaid in terms of understanding and ease of operation.

# THE 'IF .... THEN .... ELSE ....' STATEMENT

In higher programming languages, the 'IF - THEN - ELSE' statement is possible. But did you know it can also be used in the DAI Basic ? Not in the style 'IF - THEN - ELSE', but via a special use of the 'IF - GOTO' statement !
This article describes how to do it.

Try the following example:

```
10 IF A=1 GOTO 30: GOTO 40
20 PRINT "20",
30 PRINT "30",
40 PRINT "40"
```

A RUN of this program prints:'30, 40' if A=1 and: '40' if A <> 1.
Line 20 is never executed !

Try the following example:

```
10 IF A=1 GOTO 30: PRINT "TEST"
20 PRINT "20",
30 PRINT "30",
40 PRINT "40"
```

This results in: '30, 40' if A=1, and: 'TEST, 20, 30, 40' if A <> 1.

The same results are obtained if in line 10 is written:

```
10 IF A=1 THEN 30: ........
```

You see, with 'IF - GOTO <linenr>: GOTO <linenr> an 'IF - THEN - ELSE' statement can be simulated !
But only if the first statement is an 'IF - GOTO <linenr>' or an 'IF - THEN <linenr> statement !

The reason for this behaviour of the DAI is the way the 'IF - GOTO' statement is executed.
The run-time routine can be found on address DF15. The logical expression ( A=1 ) is evaluated. The result of the evaluation is in accumulator A. If the condition is true ( A=1 ), the program jumps to the new linenumber via DF63.
If the condition is not true ( A<>1 ), the pointer in the textbuffer - the registers BC - is moved untill after the linenumber in the first 'GOTO' statement. There it finds the second 'GOTO' statement, which is performed accordingly !

Still one remark: Be very carefull in using this 'IF - THEN - ELSE' construction. Use it only in the form:

```
IF <logical expr> GOTO <linenr>: GOTO <linenr>
```

Then you can be sure that you do not create other problems in your programs.

# >>>>>>>>PROGRAMMEER - TECHNIEKEN<<<<<<<<

Ik wil in dit artikel eens ingaan op het programmeren van een programma. Op het eerste gezicht een misschien wat vreemd onderwerp; iedereen die een DAI bezit zal toch regelmatig programma's schrijven. Wat aanwijzingen om bepaalde zaken op een of andere manier aan te pakken -accoord- maar het programmeren zelf dat is een ieder wel duidelijk.
Jammer genoeg moeten we vaststellen (aan de hand van de inzendingen) dat juist het opzetten en vooral uitwerken van programma's en programma-ideeen vaak nog wat te wensen overlaat.

"Hoe zou de aanpak dan moeten zijn ?" zult U zich misschien afvragen. Er is echter geen standaardaanpak die voor alle programma's gebruikt kan worden. We moeten ons ten eerste afvragen wat wij met het programma willen doen.Pas als we daarop een zinvol antwoord kunnen geven heeft het zin om over een aanpak te gaan nadenken.

We gaan in gedachten eens na wat we (zouden) moeten doen voordat we achter het toetsenbord gaan zitten. Ik pretendeer echt niet alle relevante zaken nu aan de orde te laten komen, maar denk wel dat bij de overwegingen die ik zal bespreken er vele zullen zijn die vaak vergeten worden. Ook is het mogelijk dat men pas later aan bepaalde aspecten denkt zodat achteraf veel reparatiewerk nodig is. Goed, we hebben een idee voor een programma.Er zijn verschillende mogelijkheden

1) programma zelf verzonnen
   - we moeten alles zelf doen
   + we kunnen alles zelf doen
2) programma zien werken op ander toestel
   - we proberen misschien iets waar de DAI minder geschikt voor is
   + we kunnen werk van oorspronkelijke maker gebruiken
3) we hebben de listing van het programma maar wel voor ander toestel
   - we proberen misschien iets waar de DAI minder geschikt voor is
   - we zitten vast aan de denktrant van een andere programmeur dit betekent bijna altijd dat wij een minder goed programma maken als mogelijk zou zijn, omdat die ander rekening houdt met zijn machine en niet met onze
   - programma's uit de basicode (hobbyscoop) hebben niet de beperkingen van een machine maar van alle.
   + niemand kan u beschuldigen van originaliteit

Voor zover uit bovenstaande nog niet duidelijk: ik wil graag volledig zelfgemaakte programma's. Alleen dan is er een mogelijkheid dat er inderdaad programma's ontstaan die de mogelijkheden van de DAI ten volle benutten.
Dat u een leuk goed werkend programma van een andere machine ook op u eigen DAI wilt hebben is uw goed recht, maar kijk dan niet in de listing!!! Heus het komt het programma alleen maar ten goede.

De volgende fase in onze programmeergang is eens goed na te denken over o.a. de volgende zaken:
1) Voor wie is het programma bedoeld ?
2) Wil ik het programma ook in de toekomst nog gebruiken ?
3) Wil ik het programma of delen ervan later in andere programma's gebruiken ?
4) Is het nodig dat anderen de werking van mijn programma begrijpen ?
5) Is het nodig/wenselijk mijn programma's een eigen signatuur te geven ?
6) Is uitleg voor gebruik van programma nodig ?
7) Zijn varibelen in floating-point of integer nodig ?
8) Hoe snel moet programma klaar zijn
9) Hoe groot moet de snelheid van het programma zijn ?
enz.,enz,enz.....

Op de hier genoemde punten wil ik wat nader ingaan.

ad 1: Maakt u een programma enkel en alleen voor uzelf (dus ook geen vrienden en kennissen) en bent u niet van plan het programma vaker te gebruiken dan is uitleg of programmeerduidelijkheid overbodig. Mijn ervaring is dat op een enkel rekenprogramma na geen enkel programma aan deze eisen voldoet.

ad 2: Als u een programma later wilt gebruiken is een minimale uitleg een vereiste. Zelfs het eenvoudigste rekenprogramma is onduidelijk als het na RUN alleen een "?" geeft bij de invoervraag.Zet ook in het begin van het programma een datum en als modern mens (wat doet u anders met een computer) doe dat dan in de SI-vorm (In vele landen verplicht maar zelfs overheden zelf houden zich er niet aan) N.B. de SI-vorm is jaar-maand-dag voorbeeld 19830314
Een andere mogelijkheid is maand/jaar.

ad 3: Als u hier rekening mee wilt houden zult u modulair moeten programmeren. Later hierover meer.

ad 4: Een must voor docenten computerkunde (zoals ik), maar ook als u bepaalde programmeertrucs heeft verwerkt in ingzonden programma's is het prettig als anderen uw idee ook kunnen verwerken.

ad 5: Alleen als u dat zelf leuk vindt of als u commerciele bedoelingen hebt met uw programma. Leuk is het echter wel. Zorg er dan echter wel voor dat dit persoonlijke stempel aangenaam voor anderen is.
Sommige inzenders menen hun programma's herkenbaar te moeten maken door bv een aanslag te doen op mijn beeldbuis resp ogen en oren.
Zaken waar ik mij bv aan erger:
het beeld van een kader voorzien door:
FOR I=0 TO 10:J=1-J:FILL I,I XMAX-I,YMAX-I 15*J:NEXT
kleuren zodanig gekozen dat op een andere machine een onleesbar geheel ontstaat terwijl het niet simpel is te verhelpen door een KL1=.. of COLORT resp COLORG.
geluid bij "ongeluk" in programma niet af te zetten door gebruiker.
en nog vele maar daar zal ik u nu niet mee vervelen. (mogelijk in toekomst wel)

ad 6: Kleine uitleg bijna altijd gewenst. Hoe meer een programma voor gebruik door anderen - zeker als die anderen niet programmeren - gemaakt wordt hoe belangrijker de uitleg wordt. Probeer deze uitleg altijd zo duidelijk mogelijk te krijgen. Een ander kan dit meestal beter dan uzelf testen. Een niet programmerende en ook op de DAI geen ervaring hebbende kennis is hiervoor ideaal.
Voorbeeld : goede testers weten niet dat zij na een antwoord ook de returntoets nog moeten indrukken.Of dat zij voor RUN eerst MODE 0 hadden moeten intikken.
Geef uitleg eens door bij een spel bv een stuk voor te doen of zet uitgebreidde uitleg in een apart programma. Dit laatste scheelt inleestijd en geheugen

ad 7: Indexen van array's nooit met floating-point.Ook de cursor- en teken- opdrachten hebben nooit iets anders dan integers nodig.
Vele FOR-NEXT loops kunnen ook in integers.

ad 8: Als grote snelheid gewenst is; dan niet te veel naar uiterlijk van het programma kijken.Gebruik veel standaardroutines.Zorg als dit regelmatig gebeurt dat er een bibliotheek met zulke routines is waar u gebruik van kunt maken.

ad 9: Als de snelheid van het programma van belang is voorlopig een paar tips:
werk zoveel mogelijk in integers
test verschillende mogelijkheden van programmeren
zet het de snelheid het meest beinvloedende deel vooraan in het programma
schrijf gedeeltes van programma in machinetaal of vraag indin u dit (nog) niet kunt iemand anders dat voor u te doen.

Ik wil dit met een voorbeeld uit mijn eigen ervaring illustreren.

Als wiskundedocent had ik eens een lijst met gehele getallen nodig, die in groepjes van drie zouden voldoen als lengtes van zijden van een rechthoekige driehoek.(zgn pythagoreische getallen)
Ik wilde deze lijst gebruiken bij het samenstellen van een repetitie over de stelling van Pythagoras.Het is voor de leerlingen prettig als de uitkomsten gehele getallen zijn; getallen die ik wilde hebben zijn bv drie,vier,vijf omdat 3^2 +4^2 = 5^2. Andere bekende combinaties zijn 5,12,13 en 8,15,17.
Alle drie de getallen dus geheel en kleiner dan honderd.
Voor dit programma had ik alleen maar rekening te houden met het correct zijn van de antwoorden en de tijd waarin het programma klaar is. Met dit laatste bedoel ik de tijd die ik nodig had om het programma te schrijven en niet de tijd die het programma aan het rekenen is. (Als U zich wilt laten uitdagen ik had binnen een kwartier alle mogelijkheden onder de honderd op papier)
Ik hoefde in dit geval geen rekening te houden met anderen die mijn programma zouden willen gebruiken en ook niet met leesbaarheid en/of aanpassingen.
Dit geldt voor veel programma's maar vrijwel nooit voor programma's die geschreven worden voor gebruik door anderen.

Een niet altijd haalbaar ideaal zou het volgende hoofdprogramma kunnen zijn:

```
10 REM REIS OM DE WERELD / COPYRIGHT Demaeker jan-83
20 GOSUB 8000:REM Aankondiging en Inlezen data-1
30 GOSUB 7000:REM Uitleg en Inlezen data-2
40 GOSUB 6000:REM Initialisatie en Inlezen data-3
50 GOSUB 1000:REM Programma
60 PRINT "NOG EEN KEER SPATIEBALK"
70 H=GETC:IF H=0 GOTO 70:IF H=32 GOTO 40
80 END
```

Vanuit de subroutines 8000,7000 en 6000 wordt steeds de subroutine voor het inlezen van de data aangeroepen. Dit is iets wat slechts weinig programmeurs doen en dat is jammer. De gebruiker van een programma zit eerst naar een fraaie aankondiging van een programma te kijken en vervolgens de uitleg te lezen, de computer doet al die tijd niets dan wachten. Pas als de gebruiker het teken heeft gegeven dat hij klaar is met lezen (meestal spatiebalk) worden data ingelezen. Dan kan de gebruiker weer wachten. Een veel prettiger methode is het inlezen van data en/of opbouwen van tabellen te laten plaatsvinden tijdens de aankondiging en uitleg.Of lees de data in arrayvorm in, bv tijdens uitleg.
Daar de snelheid van de GOTO's en GOSUB's beinvloed wordt door de plaats van de regel in het programma is het zinvol de volgorde te kiezen uit het voorbeeld.
Voor alle duidelijkheid niet de getalwaarde van de regelnummers is van belang maar wel de plaats in de listing. Dit komt omdat de DAI als volgt reageert op een GOTO1000: 1-haal eerste regelnummer uit listing;
2-is dit 1000? ja ga daar verder en nee zoek lengte deze regel en bereken dan plaats volgende regel.ga terug naar 2.

We hebben hier een fraai voorbeeld van modulair programmeren. Het programma is met duidelijk herkenbare stukken geschreven, in dit geval in de subroutinevorm. Deze programmeeraanpak is in bijna alle gevallen goed. We zetten de voordelen even op een rij:
Onderdelen gemakkelijk te herkennen. Iemand die bv de aankondiging erg mooi vindt kan deze met aangepaste tekst zelf ook gebruiken.
Onderdelen gemakkelijk te vervangen door andere (betere) versies.
Onderdelen gemakkelijk in aangepaste vorm in andere programma's te gebruiken
Bij te lage snelheid op simpele wijze een basic-subroutineaanroep te vervangen door een CALLM.
Programma leesbaar en overzichtelijk.

Maar eerlijk is eerlijk nadelen zijn er ook.
Het programma zal nooit absolute topsnelheid halen.
Het programma is iets groter en zou in zeer extreme gevallen net niet meer in het geheugen passen.

Tot slot wil ik nog wat filosoferen over de beste manier om een programma te beginnen. Een ooit eens oa door mij aanbevolen begin is: 10 MODE0:?CHR$(12) dit omdat veel programma's in de begintijd begonnen zonder eerst een schoon beeld te maken en/of niet eerst in tekstmode kwamen.
Later hebben we ingezien dat hier soms nadelen aan kleven; een extreem groot programma dat in mode1 of 2 loopt geeft OUT OF MEMORY na de MODE0.
De ?CHR$(12) heeft twee nadelen. Als na deze opdracht het gehele scherm wordt vol gescheven zal bij de laatste regel het beeld hinderlijk opschuiven; de remedie is simpel :?CHR$(12); . We kunnen echter ook OUT OF STRING SPACE krijgen als we een CLEAR4 hadden. U zult zich misschien afvragen waarom iemand zo'n opdracht geeft; bv om de grootte van zijn programma te weten te komen.
Ook hier lijkt de remedie simpel: geef eerst een CLEAR... voor de PRINT
Ook zouden we een COLORT willen zien voor de MODE0.
Een REM in de eerste regel om programmanaam en maker aan te geven.
Voor deze REM een GOSUB.... om de snelheid bepalende delen van het programma zo veel mogelijk naar voren te kunnen zetten.
Nee, toch niet een GOTO ipv een GOSUB omdat er in een subroutine geen CLEAR mag staan.
Cursor zichtbaar of juist onzichtbaar maken enz.,enz.,....

Frank H.Druijff

P.S. 1 Ik wilde nog even de aandacht vestigen op een cirkelroutine van Fred van Amerongen, die ik naar aanleiding van een vorig artikel kreeg toegezonden

door Fred van Amerongen (knap!)

```
10    REM INCREMENTAL CIRCLE GENERATION.
20    COLORG 0 5 10 15:MODE 6:MODE 6
30    K1!=3.0:R!=140.0:XC=XMAX/2:YC=YMAX/2:REM ...CENTER
40    R!=R!-R!/10.0:REM ..............RADIUS
50    X1!=R!:Y1!=0.0:REM ..............STARTING POINT
60    K!=K1!/X1!:REM ..................INCREMENT
70    FOR I=0 TO (2*PI)/K!:X2!=X1!+K!*Y1!:Y2!=Y1!-K!*X2!
80    P=X1!:Q=Y1!:M=X2!:N=Y2!:DRAW P+XC,Q+YC M+XC,N+YC 21
90    X1!=X2!:Y1!=Y2!:NEXT:IF R!>2.0 GOTO 40
99    END
```

# program identification

title : A/D CONVERSION HARD & SOFT

author : K.H.Kopp

purpose :

comment :

K.H. Kopp
Irenenstr 93
4000 DUSSELDORF

Sehr geehrte Clubfreunde,

Um schallmessungen durchfuhren zu konnen habe ich eine schaltung mit dem ADW-IC ZN 427 von Ferranti entworfen und auf einer Veroboard lochrasterplatine aufgebaut. Diese schaltung passt in ein kunststoffgehause von 150x80x50 mm und ist met einem flachkabel an den DCE-bus angeschlossen. Mit dem beiliegenden demonstrationsprogramm sind +/- 13900 abtastungen pro/sek moglich. Vielleicht giebt es einige clubmitglieder die einen schnellen 8 bit analog-digital wandler nachbauen wollen und eine geeignete und Preiswerte losung suchen. Darum mochte ich ihnen die schaltungsunterlagen zur verfugung stellen. Moglicherweise sind mit dem ADW gerausch + sprachanalysen moglich mit deren hilfe man dit TALK funktion programmieren kan.

Bei dem aufbau der schaltung und dem uberprufen der im datenblatt fig.3 timing diagram vorgegebenen steuerimpulse fur den ZN 427 habe ich den DAIpc als logikanalysator eingesetz. Dabei wird der B und C port als ausgang und der A port als eingang programmiert. Mit einem der ausgange wird vom BASIC aus ein langsamer takt erzeugt der anstelle des quarzes die schaltung treibt. Die A port eingange werden an den zu messenden punkten der schaltung angeklemmt und nach jeder pegelanderung den bildschirm gebracht. Der takt und die pegel werden dann im grafik mode als rechtecksignal auf den bildschirm gebracht. Alle eingange und ausgange sind mit dem CMOS IC 4050 gepufferd und die eingange mit dioden geschutzt. Mit dem beiliegenden programm wird der DAI als analysator betrieben.

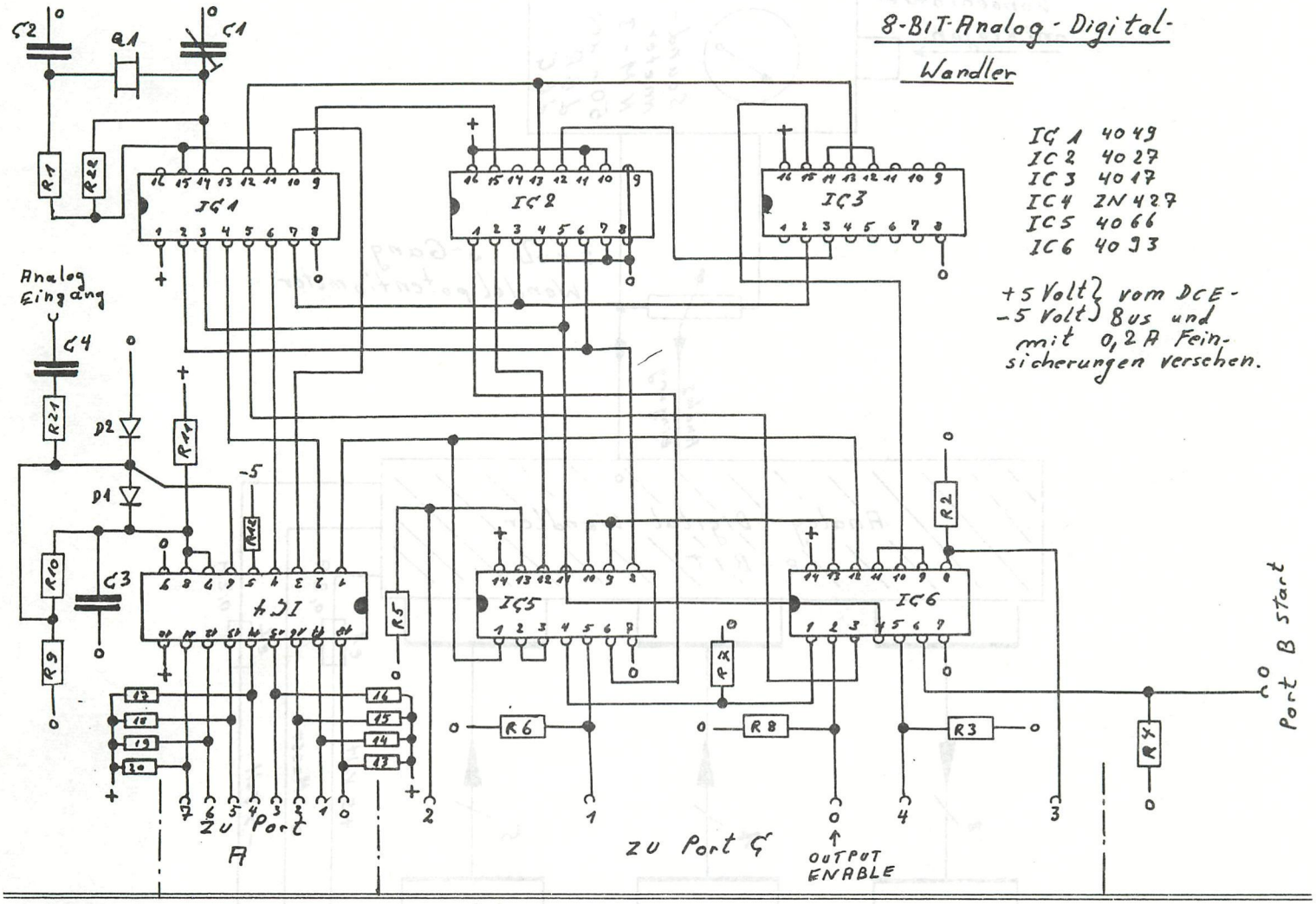
Viel spass beim nachbauen
mit freundlichen grussen

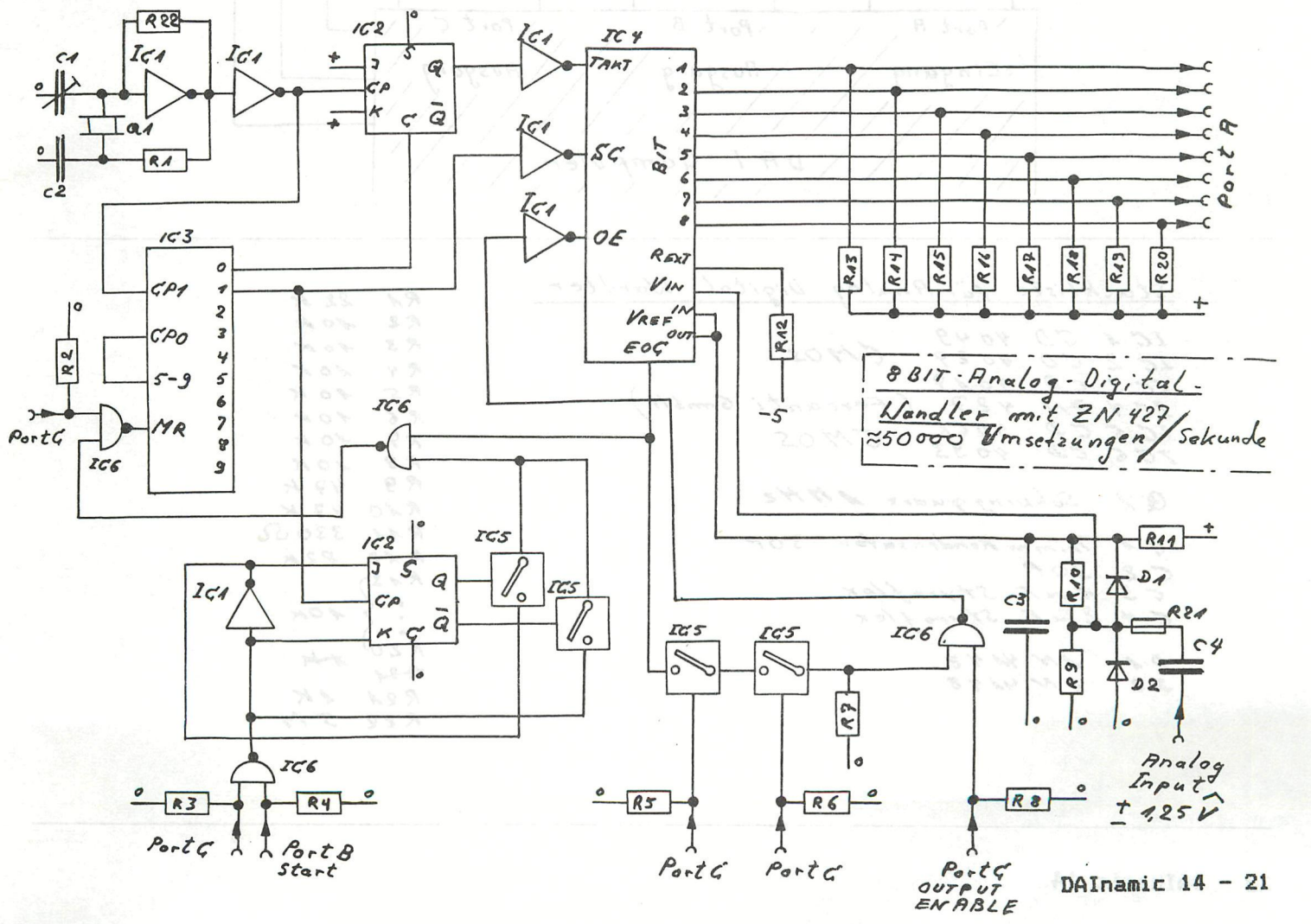
K.H.Kopp

# 8-BIT-Analog-Digital-Wandler

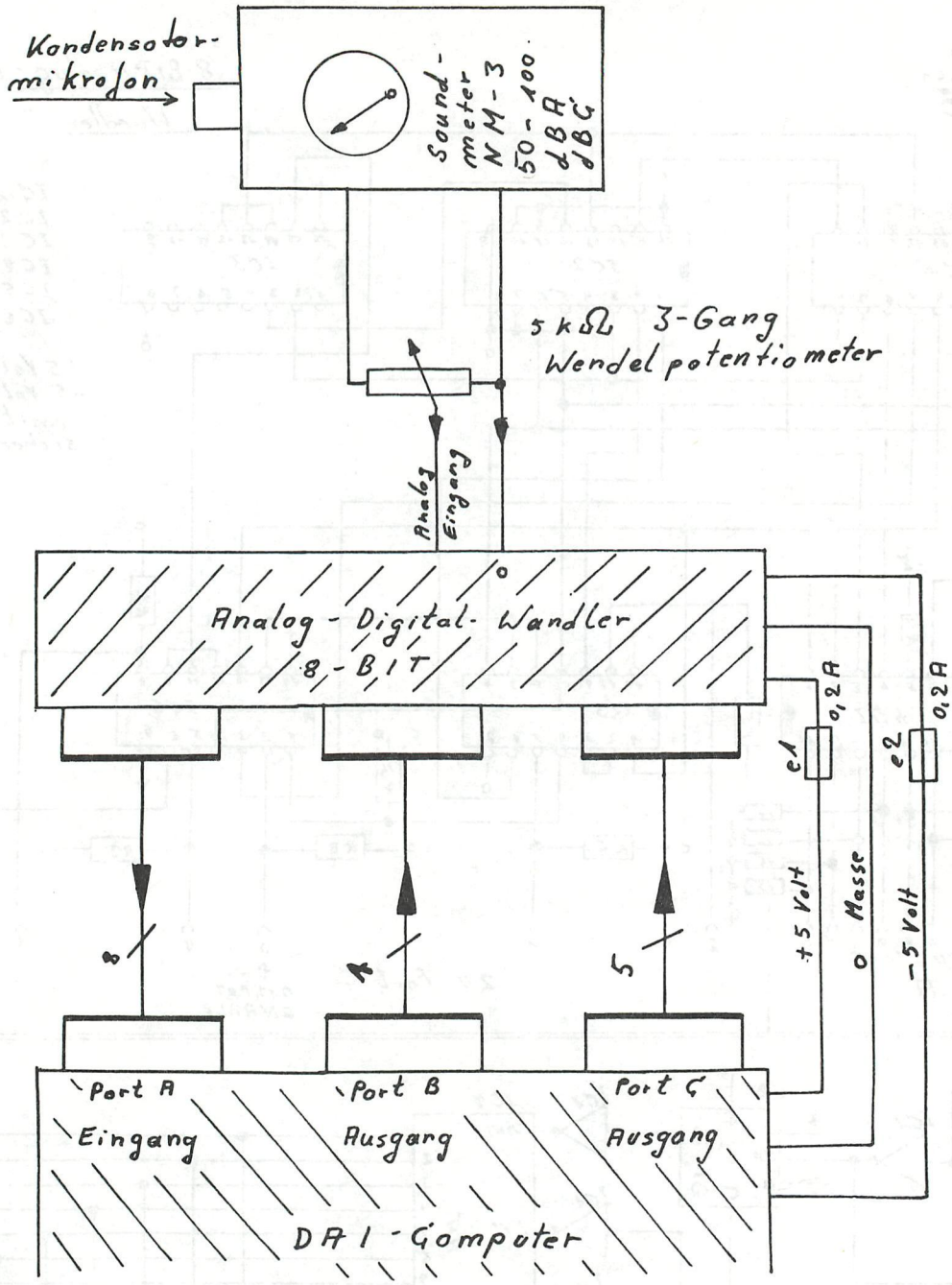
IC 1 4049
IC 2 4027
IC 3 4017
IC 4 ZN 427
IC 5 4066
IC 6 4093

+5 Volt } vom DCE-
-5 Volt } Bus und
mit 0,2 A Fein-
sicherungen versehen.




8 BIT-Analog-Digital-Wandler mit ZN 427
≈50000 Umsetzungen/Sekunde




---

## Stückliste für Analog Digital Wandler

| | | |
|---|---|---|
| IC 1 | CD 4049 | |
| IC 2 | CD 4027 | CMOS |
| IC 3 | CD 4017 | |
| IC 4 | ZN 427 | (Ferranti GmbH) |
| IC 5 | CD 4066 | CMOS |
| IC 6 | CD 4093 | |

| | |
|---|---|
| Q 1 | Schwingquarz 1 MHz |
| C 1 | Trimmkondensator 30P |
| C 2 | 30P |
| C 3 | 1 µF Styroflex |
| C 4 | 2 µF Styroflex |
| D 1 | 1N 4148 |
| D 2 | 1N 4148 |

| | |
|---|---|
| R1 | 22 K |
| R2 | 10 K |
| R3 | 10 K |
| R4 | 10 K |
| R5 | 10 K |
| R6 | 10 K |
| R7 | 10 K |
| R8 | 10 K |
| R9 | 47 K |
| R10 | 47 K |
| R11 | 330 Ω |
| R12 | 82 K |
| R13 : R20 | 10 K |
| ~~R21~~ | ~~1 K~~ |
| R21 | 1 K |
| R22 | 5 M |

```
1     MODE 0:REM ********** A D W  PROGRAMM  *********
2     REM DAS MASCHINEN PROGRAMM STARTET DEN ADW
3     REM UND LIEST DIE DATAEN VON PORT A IN DEN
4     REM RAM VON #4000 BIS #5000
5     REM DANN WERDEN DIE DATEN IN MODE 5 DARGESTELLT
6     REM DIE AUFLOESUNG KANN MIT DER ADDITION IN 230
7     REM VERAENDERT WERDEN 230 S%=S%+  1 2 3  ...12
10    DATA #00,#F3,#F5,#E5,#D5,#C5,#3E,#90,#32,#03
20    DATA #FE,#3E,#FF,#32,#02,#FE,#01,#00,#10,#11
30    DATA #00,#40,#00,#00,#3A,#00,#FE,#12,#13,#0B
40    DATA #78,#B1,#CA,#2B,#3F,#79,#32,#01,#FE,#C3
50    DATA #17,#3F,#00,#00,#C1,#D1,#E1,#F1,#FB,#C9
60    RESTORE
70    FOR A%=#3F00 TO #3F31
80    READ B%:POKE A%,B%
90    NEXT A%:CURSOR 15,12:PRINT " DRUECKE SPACE TASTE FUER A D W "
91    CURSOR 25,11:PRINT " DANN "
92    CURSOR 18,10:PRINT " S TASTE FUER NEUE A D W "
100   IF GETC=32.0 THEN CALLM #3F00:GOTO 200
105   GOTO 100
200   MODE 5:S%=#4000
210   FOR A%=0 TO XMAX
220   DRAW A%,PEEK(S%) A%,128 15
230   S%=S%+12:REM ************AUFLOESUNG DER GRAFIK
240   NEXT A%
250   IF GETC=83 THEN GOTO 1:REM ***** DREUCKE TASTE S
260   GOTO 250
```

```
10    PRINT CHR$(12)
20    PRINT TAB(20);" TAKTPEGEL PORT-C BIT-0 ":PRINT
22    PRINT TAB(20);" PORT-A  BIT-7 ":PRINT
24    PRINT TAB(20);" PORT-A  BIT-6 ":PRINT
26    PRINT TAB(20);" PORT-A  BIT-5 ":PRINT
28    PRINT TAB(20);" PORT-A  BIT-4 ":PRINT
30    PRINT TAB(20);" PORT-A  BIT-3 ":PRINT
32    PRINT TAB(20);" PORT-A  BIT-2 ":PRINT
34    PRINT TAB(20);" PORT-A  BIT-1 ":PRINT
36    PRINT TAB(20);" PORT-A  BIT-0 ":PRINT
38    PRINT " DRUECKE SPACE TASTE ":PRINT
40    IF GETC<>32.0 THEN 40
60    COLORG 10 10 10 10:MODE 5
70    POKE #FE03,#90:POKE #FE02,#FF:POKE #FE01,#1:REM *********PORT A,B,C PROGRAMMIEREN
71    T%=128:Y%=28
80    FOR X%=1 TO 320 STEP 16
90    AP%=PEEK(#FE00):REM ********PORT A ABFRAGEN
100   FOR A%=1 TO 8
120   IF AP% SHR A%-1 IAND 1=1 THEN GOSUB 1000:GOTO 140
130   GOSUB 900
140   NEXT A%
145   DRAW X%,227 X%,245 0
150   IF T%=128 THEN DRAW X%,245 X%+16,245 0:GOTO 170
160   DRAW X%,227 X%+16,227 0
170   POKE #FE01,T%
180   IF T%=0 THEN T%=128:GOTO 200
190   IF T%=128 THEN T%=0
200   NEXT X%
500   IF GETC=78 THEN MODE 0:GOTO 60:REM ******DRUECKE TASTE N
510   GOTO 500
890   REM *********SUBROTINE FUER HI PEGEL
900   IF SCRN(X%,Y%*A%-7)=0 THEN DRAW X%,Y%*A%-7 X%,Y%*A%-25 0
910   DRAW X%,Y%*A%-25 X%+16,Y%*A%-25 0
920   RETURN
990   REM ********SUBROTINE FUER LOW PEGEL
1000  IF SCRN(X%,Y%*A%-25)=0 THEN DRAW X%,Y%*A%-25 X%,Y%*A%-7 0
1010  DRAW X%,Y%*A%-7 X%+16,Y%*A%-7 0
1020  RETURN
```

# MAGAZINE NEWS

For the DAI fanatic, the last issues of most of the magazines were not too interesting. The regular column PATTERN which used to appear in Personal Computer World (UK), is not being continued. It's a shame because it used to be a very good series filled with innovative ideas.
The other magazines (British, French, Dutch, German, etc.) do not cover the DAI either. The DAINAMIC newsletter seems to be the only good source of information for the DAI-hacker. Sometimes it is interesting though to read other magazines, articles and programs even if these do not apply directly to our favorite computer.

Two magazine issues of American magazines are particularly interesting for they contain a lot of articles about graphics. The first one is BYTE (November 1982) and the second one is CREATIVE COMPUTING (January 1983).

The BYTE issue is interesting mainly for two articles one giving a fairly good general overview of what computer-graphics are all about (A Graphics Primer by Gregg Williams) The article is not really difficult or original though. A second article is more technical and deals with 3-D graphics on an Apple II. Programs in BASIC and Pascal are included. Some other articles talk about the various software packages that exist for ATARI, Apple II, etc.
Other articles in the November issue of BYTE:
- Graphics with Logo
- Build a video digitizer
- Computer graphics animation on the ATARI
- Microvec: a D.I.Y vector display system

The CREATIVE COMPUTING issue contains more articles but these tend to be shorter and more superficial than the BYTE articles. There is also a great number of evaluations of new soft- and hardware.
The more interesting articles are:
- Smooth graphics with pixel averaging
- Stereo graphics: 3D computer graphics
- The use of linked lists in computer graphics
- Computer art

Conclusion: both issues contain a lot of articles but the quality or originality of the articles is not always very good.
If you are looking for a D.I.Y. article on video-digitiser you should take a look at PCW (Janury 1983) page 168 -> 171.
I do expect to see a greater number of articles on computer-graphics as a great number of the newer computers have (limited) graphic possibilities.

New magazines

I recently discovered a couple of new magazines. I have no more than a couple of issues so it's a bit early to say what I think about them.
- Electronics and computing.
  This magazine tries to combine both hard- and software but allas! finds it also necessary to test every new computer on the market. If a new computer appears on the market, you can read about it in some 10 different magazines. Is this really necessary?
  price: 70 p per issue in the UK
  84 Bfr. in a bookstore in Belgium
  subscription: 9.50 pound (UK only)
  13.95 pound (overseas)
  address: Subscription dept. 40/42 OXFORD STREET
  DAVENTRY NORTHAMPTON NN11 4AD UK

Recently I ran acroos a couple of interesting American magazines featuring a majority of articles on CP/M. These magazines could be more and more interesting for DAI owners as CP/M is to be released "real soon".
One could also read in the last issue of the HCCN that the CP/M library will be converted to DAI-format.

It would be wrong to think that CP/M is the nec plus ultra in operating systems. It is also false to think that it isn't worth a dime. In fact it's a good compromise between an expensive, elegant, slow and bulky operating system and no operating system at all.
One of the greatest advantages of CP/M is that the software calls are well defined and compatible. So is most of the source and object code for various programs.
Total compatibility does not exist however in the fast growing field of computers and so it happens that the medium is not standard if one is working with 5 1/4 '' disks. There is a standard for the 8 '' disks but this standard is growing older and older and more people are presently using the smaller disks.
An advantage that CP/M has over any other O.S. is the great number of software packages that can run under CP/M. If you're looking for PASCAL, C, FORTRAN, LISP, COBOL or any other language interpreter or compiler there is quite a chance that you will find it in some CP/M library.
For the serious DAI user this is a very strong point.

But let's go back to the two magazines I mentioned earlier.( one could go on and on indefinitely about the benefits and the horrors of CP/M !)

1)- LIFELINES (THE SOFTWARE MAGAZINE)

| | |
|---|---|
| frequency | 12 issues a year |
| price | $ 3.00 for 1 issue |
| subscription | $ 50.00 for 12 issues (other countries than US, Can., Mex.) |

address: LIFELINES/ THE SOFTWARE MAGAZINE
1651 Third Avenue
New York, N.Y. 10028 USA

2)- MICROSYSTEMS (THE CP/M USERS'S JOURNAL)

| | |
|---|---|
| frequency | 6 issues a year |
| price | $ 3.95 for 1 issue (294.00 Bf.!!) |
| subscription | $ 32.97 for 12 issue ( 2 years !) |

address: MICROSYSTEMS
PO Box 1987
Morristown. NJ 07960 USA

Both magazines contain
- evaluations of new programs and software packages
- tutorials on CP/M, UNIX and other related subjects
- practical program listings for CP/M machines (with source code and fully commented)
- evaluation of new hardware items for the S100 bus (MICROSYSTEMS)
- programming techniques

Don't look for graphical Apple games in these magazines or don't expect to see homebrew hardware articles.
The level is pretty high and so is the quality of both magazines. MICROSYSTEMS is aimed a little more towards a professional public whereas LIFELINES is equaly useful for the amateur or hacker.
These two magazines are some of the best I have seen lately !

## ! ! ! ! ! SUPER BARGAIN ! ! ! ! !
## ! * ! FOR SALE ! * !

**********

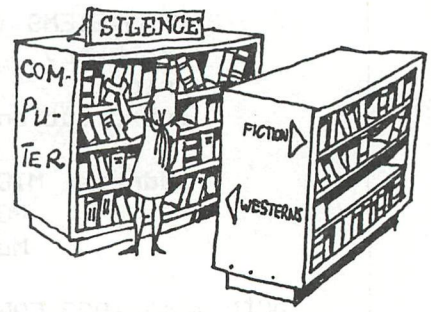
1)- I have a number of brandnew CASSETTE MECHANISMS for sale.
These units were especially manufactured for REMOTE CONTROL.
Each unit has 3 solenoids mounted on a base, to activate levers so as to enable FFW, FRW & RUN. A head and speed stabilised motor is also mounted.
Shematics for driving the solenoids are included. The components for amplifying the signal will have to be designed by yourself or you can use an existing unit. A shematic for saturation recording is also included in the very low and unique price of

**1500,— Bf. or 100,— Fl.**
+ post & packing

2)- Also for sale : a couple of modern Video Games with a cassette mechanism as described above, processor board, power supply unit and transformer, video display unit, modulator player keyboards (4 buttons) and many many more interesting components. Full set of shematics included !!!
Perfect for the hardware hacker !!! These games use a lot of the modern IC's . Most units are in working order and all are supplied with a couple of game-cassettes.
Used to cost 9000,- Bf. Now for

**only 3000,— Bf. (200,— Fl.)**
+ post & packing

3)- Must go !!! A large collection of magazines related to electronics and radio-controlled models. The number of items is too great even to give only a partial overview.
Also some books (electronics, engineering and model-building) are offered for sale. All for a very low price and in perfect shape.




For more information and availability please call
Jos Schepens at >>>> 052-21 67 43 <<<<
weekends only and not after 10 p.m. please !!!!
You can order at the following address:

Jos Schepens
Sint Jorisgilde 53
B-9330 DENDERMONDE
BELGIUM

Payments can be in any currency but preferably in Bfr.
You can pay by cheque, by GIRO, with an international money order or by money transfer on the account number 001-0855666-08

---

# CP/M® MICRO-COMPUTER CONTROL PROGRAM

**CP/M®** is a proprietary, general-purpose control program designed especially for microcomputers which use the Intel 8080, 8085, or Zilog Z80® microprocessor. CP/M has been in existence for over five years and has undergone extensive field testing in thousands of installations. CP/M has evolved into a sophisticated interactive program development system also serving as the basis for programming languages, word-processing software and business applications packages. Basic CP/M facilities include dynamic file management, fast assembler, general-purpose editor, and advanced debugger. Additional software packages include an Intel-compatible macro assembler, symbolic debugger, text formatter and spooler. High-level languages and applications packages are available from Digital Research and independent suppliers.

**BDOS**—The Basic Disk Operating System supports a named file system with a maximum of sixteen logical drives. Any particular file can contain from zero to eight megabytes, with space dynamically allocated and released. System calls enable an application program to create, rename, delete, read, and write files on any active disk drive, with both sequential and relative-record random access to data. The standard CP/M system is distributed for operation with IBM soft-sectored diskettes, setup with default values of 64 files per diskette, four diskette drives, and 240K bytes per drive. A field alteration manual gives procedures and program listings necessary to change these default values. Various peripheral devices are supported through the BDOS, including console, line printer, and paper or magnetic tape I/O device. With all these features, the BDOS uses less than 4K of memory.

**CCP**—The Console Command Processor interacts with the user via the built-in commands:

DIR—List all or selected directory entries.
TYPE—Type the contents of an ASCII file at the console.
REN—Rename a specific file.
ERA—Erase a specific file or set of files.
SAVE—Save the contents of memory on disk.
USER—Change the active user number.

In addition, the CCP allows CP/M system programs and user programs to be loaded and executed as though they were built-in commands.

**Utilities:**

**PIP**—The Peripheral Interchange Program provides file transfer between devices and disk files and performs various reformatting and concatenation functions. Formatting options include parity-bit removal, case conversion, Intel "hex" file validation, subfile extraction, tab expansion, line number generation, and pagination.

**ED**—The CP/M Text Editor allows creation and modification of ASCII files using extensive context editing commands: string substitution, string search, insert, delete, and block move. ED allows text to be located by context, line number, or relative position with a macro command for making extensive text changes with a single command line.

**ASM**—The CP/M Assembler is a fast 8080 assembler using standard Intel mnemonics and pseudo operations with free-format input, conditional assembly, and assembly-line expressions.

**DDT**—The CP/M Dynamic Debugging Tool is a powerful facility for 8080 program debugging; it contains an integral assembler/disassembler module that lets users patch and display memory in either assembler mnemonic or hexadecimal form. DDT allows the user to trace program execution with full register and status display. Instructions can be executed between breakpoints in real-time, or run fully monitored one instruction at a time.

**SUBMIT**—The submit utility allows the user to batch together a parameterized group of prototype CP/M commands in a file, and then "submit" them to the operating system with a single command.

**STAT**—The STAT utility alters and displays I/O device and file status including free-space computations, status of on-line diskettes, and physical-to-logical device assignment.

**LOAD**—The LOAD utility converts Intel "hex" format to absolute binary, ready for direct load and execution in the CP/M environment.

**SYSGEN**—The System Generation utility creates new CP/M system diskettes from existing diskettes for back-up purposes.

**MOVCPM**—MOVCPM provides regeneration of CP/M systems for various memory configurations and works in conjunction with SYSGEN to provide additional copies of CP/M.

The complete CP/M operating system is distributed on an IBM-compatible diskette, initially set to operate directly on an Intel MDS-800 Microcomputer Development System (with single-density drives). This standard CP/M system can be reconfigured to operate with any microcomputer hardware with the following characteristics:

- Intel 8080 or Zilog Z80 Central Processing Unit.
- At least 20K bytes of Read/Write main memory, contiguous from location zero.
- One to sixteen disc drives of up to eight megabytes capacity each.
- Some form of ASCII console device (normally a CRT).

Many popular hardware manufacturers distribute CP/M with necessary device-controller subroutines already installed for their equipment. The CP/M system is distributed in machine-code form only along with the complete documentation required. The software is licensed for use by the individual who purchases CP/M, and is registered and serialized to prevent unauthorized copying and distribution. The registered owner receives notice of updates and field changes to existing software.

**Documentation**

- CP/M 2.2 User's Guide—This manual gives an overview of CP/M 2.2 and introduces the features of this expanded operating system.
- An Introduction to CP/M Features and Facilities—This manual presents the organization of the CP/M system, along with the forms of file references, built-in commands, and transient commands including operation of the editor (ED), assembler (ASM), debugger (DDT), peripheral interchange program (PIP), and batch processor (SUBMIT).
- CP/M System Interface Guide—This manual gives the exact details for programming in the CP/M environment. All system calls are described, along with details of the CP/M file organization required to write programs which operate upon CP/M files.
- CP/M Assembler Guide, CP/M Debugger Guide

## THE BASIC-FUNCTION 'TAB'

As you know from the DAIpc handbook, the TAB-function can be used to print data in columns. As long as the cursor is not past the given tab-position, it is moved (by printing additional spaces) to the particular TAB-position.

But did you too find the TAB-function not working sometimes, appearently without any good reason? Well, here is the explanation. The story is a little bit different for the two Basic versions V1.0 and V1.1.

When performing the TAB-function, the DAI-Basic monitor checks the setting of the output switch DOUTC (#0131). When DOUTC=0 (output to screen and RS232), the TAB-function works as expected. For all other values of DOUTC (1 = output to screen only; 2 = output to edit-buffer; 3 = user output routine), the TAB-function is NOT performed, but only one (1) additional space is printed !!!!!!

When your machine has a Basic version V1.1, you are a little bit luckier. There the TAB-function is performed correctly for DOUTC is 0 (screen + RS232) and 1 (screen only).

Jan Boerrigter

## INITIALISATION DCE-PERIPHERALS

In the power-on routine of the DAI, an initialisation procedure for the DCE-bus is available. Via this routine, control can be handed over to a peripheral which is connected to the DCE-interface.

Somewhere in the power-on sequence, a jump is made to ROM-bank 3 via RST1/0C. Via 3EE0C is jumped to 3EF90.

```
EF90  MVI A,:98
EF92  STA :FE03      Set GIC cmd word for PA, PCH inputs,
                     PB, PCL outputs.
EF95  MVI A,:07
EF97  STA :FE03      PC bit 3=1.
EF9A  MVI A,:01
EF9C  STA :FE01      PB bit 1=1.
EF9F  MVI A,:01
EFA1  STA :FE03      PC bit 0=1.
EFA4  LXI B,:1000    Load time counter.
EFA7  LDA :FE02      Get inputs from PA.
EFAA  ANI :20        PA Bit 5 only (Data present).
EFAC  JNZ :EFB8      When the peripheral sets PA5=1, then a jump
                     to the DCE input routine is made.
EFAF  DCX B          Decrement time count.
EFB0  MOV A,B        )
EFB1  ORA C          ) Check if time count =0.
EFB2  JNZ :EFA7      Keep checking if PA5 becomes 1.
EFB5  MVI A,:EE      When not, abort DCE input routine.
EFB7  RET            Back to power-on routine.
```

When PA5=1, a jump is made to the input routine. Data is read from the peripheral and placed in the stackbottom. When ready, control is handed over to the (M.L.) program in the stackbottom.

```
EFB8  LXI D,:F800    Address stackbottom.
EFBB  MVI A,:05
EFBD  STA :FE03      PC bit 2=1 (Request for data).
EFC0  LDA :FE02      Get inputs from PC.
EFC3  ANI :80        PC bit 7 only (Data valid).
EFC5  JZ :EFC0       Wait for PC7 to become 1.
EFC8  LDA :FE00      Get inputs PA. This must be hex-bytes with
                     machine language instructions.
EFCB  STAX D         Store inputs in stackbottom.
EFCC  INX D          Points to next location in stack.
EFCD  MVI A,:04
EFCF  STA :FE03      PC bit 2=0.
EFD2  LDA :FE02      Get inputs from PC.
EFD5  ANI :80        PC bit 7 only.
EFD7  JNZ :EFD2      Wait for data valid signal to disappear.
EFDA  LDA :FE02      Get inputs from PC.
EFDD  ANI :20        PC bit 5 only.
EFDF  JNZ :EFBB      Continue reading inputs from PA (and store
                     them on stack) as long as PC5=1.
EFE2  MVI A,:06
EFE4  STA :FE02      Set PC2 and PC1=1, PCrest =0
                     [In ROM, 'FE3E' is found; this is decoded
                     as 'FE02'].
EFE7  LDA :FE02      Get inputs from PC.
```

```
EFEA  ANI :20        PC bit 5 only.
EFEC  JZ :EFE7       Wait for PC5=1.
EFEF  JMP :F800      Goto the routine loaded into the stack-
                     bottom.
                     This routine must end with a RET instruction!
```

After performing the routine loaded from the peripheral, the DAI continues with the normal power-on sequence.

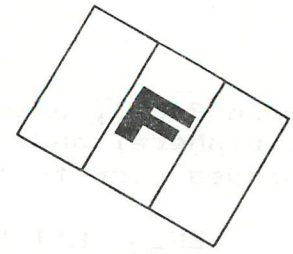
For using the DCE-bus, see also the article in the DAInamic Newsletter nr.1 of September 1980 and the Intel data sheets of the 8255 Programmable Peripheral Interface.
When using the power-on DCE initialisation routine, take good care of the several handshake signals on ports PA, PB and PC.

Jan Boerrigter

# DAInamic MEETING :

# on 9th April 1983 10.00 - 18.00 Hr
# in TONGELSBOS Bosstr. 2 3180 WESTERLO

# DAInamic INFO



Cher membre ,

Ce premier DAInamic INFO vous propose quelques traductions (resumés) des articles qui se trouvent dans le NO 12 de la revue. Ces traductions ne forment pas toujours un texte cohérents, Il faut alors s'aider des exemples et schémas de la revue. Je tiens à remercier toute l'équipe qui nous a permis de mettre au point ce numéro. Plus particulierement Mr Vim Van Dalfsen pour les traductions Flamands-Francais et Mr Wilfried Hermans pour le soutient qu'il apporte au club.

DAInamiquement votre
Cédric Dufour

EFFICIENT CIRCLE DRAWING (P268/271)

Dans cet article, plusieurs solutions (Numérotées ici de 1 a 14) vous sont proposées pour tracer un cercle de rayon 100 centre sur l'écran. Nous résumerons ici les caracteristiques de chaque méthode.

Ex 1 : C'est la méthode trivial. Le pas d'incrément est de 1 degré soit pi/180. Le cercle n'est pas continue

Ex 2 : Ici le pas est de 2*pi/200*pi. Il a été réduit pour avoir le maximum de points sans toutefois tracer deux fois le même (200*pi= Circonférence du cercle). L'utilisation d'une variable (R) a la place d'une constante (100) accelere l'execution.

Ex 3 : On ne vas plus faire chercher au DAI pres de 600 fois les valeurs X et YMAX

Ex 4 : L'utilisation d'entiers pour certaines variables va encore accelerer l'execution. pi+pi se calcule plus vite que 2.0*pi

Ex 5 : (Ligne 20 il faut lire :Y=R!*SIN(I) ) Pour ne plus calculer 628 fois le SIN et le COS utilisons une symetrie du dessin.

Ex 6 : (Ligne 30 seulement) Ici avec les deux symetries axiales. Il faut alors diminuer la boucle de la ligne 20 a pi/2. C'est vraiment plus rapide que notre solution triviale.

Ex 7 : Essais pour utiliser les symetries des bissectrices (diagonales).Ca va un peu plus vite et c'est beaucoup plus compliqué !!

Ex 8 : Une autre équation du cercle est X*X+Y*Y=R*R (X & Y etant les coordonées d'un point du cercle, R le rayon) Mais ici nous n'avons plus que deux points Y pour un point X il manque donc des points.

Ex 9/10 : La partie supérieur etant correct on essaie, par symetries, les autres cotés. R*R est plus rapide que R^2 et en plus on peut mettre des nombres négatifs.

Ex 11 : (Mettez plutot R que R!) Pour dessiner plusieurs cercles il sera interresant de mettre les valeurs des SIN & COS dans un tableau. Le PRINT P est superflu. Il est plus rapide d'utiliser deux tableau a deux dimensions qu'un seul a trois dimensions.

Ex 12 : On reprend ici l'idée des symetries (Il serait possible de mettre les valeurs des SIN et COS dans des lignes de DATAs)

Ex 13 : Voici enfin une méthode vraiment rapide. Nous tracons le cercle par une suite de petits traits.

Ex 14 : Comme en 13 mais combine avec un tableau. C'est plus difficile a programmer ! Il existe encore bien des ameliorations à apporter.(L'auteur vous laisse travailler maintenant !!)

## HOW TO EXTEND THE DAI BASIC (P264-265)

Il n'est à priori pas simple de rajouter des commandes au BASIC. Une idée de solution est ici propose.(C'est la methode employée par MEMOCOM pour le MDCR) (La partie en langage machine sera ici mise en M.E.V.)
Les nouvelles commandes auront la forme suivante :

```
10 CALLM (TESTREM): REM  COUNT
```

TESTREM est un programme en langage machine qui se charge de regarder si l'instruction qui suit le CALLM est un REM. Le code du REM est # A9 et la paire de registres BC 'pointe' cette donnée. Si il y a un REM alors TESTREM va regarder si la commande qui suit est dans sa table. (Table qui peut etre de la forme :

Longueur1,COUNT,Adresse1, Longueur2,Datas,Adresse2, ... ... 00=fin)

Si elle y est alors il execute cette commande (l'adresse est aussi donnée par la table) puis revient au BASIC, sinon il retourne directement au BASIC.

Organisation de la mémoire d'écran (p 256)

(1) mode graphique — schéma gauche MODE1
- A1 : En tête
- A2 : Donnés graph.
  - A (partie 1)
  - B (partie 2)
- A3 : Queu

(2) mode mixte — schéma droit MODE1A (p257)
- B1 : En tête
- B2 : Donnés (Partie B de A2)
- B3 : Intermédiaire
- B4 : Caractères
- B5 : Queu
- B6 : Archivage(Partie A de A2)

En passant d'un mode (1) a un mode (2), la partie A de A2 est mise en B6 et vice-versa. En cote des schémas un offset a #BFFF est porté. Dans la colonne separée les pointeurs en MEV contenant les données d'écran

- 512 x 244 - (p 261)

Vous obtiendrez, avec un CALLM #300 le mode 7 (ou 8). L'écran et ses pointeurs étant initialisés, vous pourrez alors utiliser les commandes BASIC de dessin (DRAW, DOT, FILL ...)
Le mode MIXTE (4 lignes de texte) n'est pas possible
Pour avoir du texte il vous faut revenir en mode 0



CONVERSION APPLE - ATARI - DAI (p 289)

En traduisant les programmes d'APPLE & d'ATARI pour le DAI vous vous trouvez confronté a quelques problèmes. Vous remarquez entre autres que l'origine des graphiques se situe en haut et à gauche de l'écran.
Ex : (p. 289) trois solutions proposées pour corriger ce 'défaut'...
L'avantage de la troisiéme solution est que vous pouvez presque copier mot pour mot le programme (Apres l'initialisation de la ligne 5).
Fonctionnement : Dans la solution NO 3 la partie en langage machine (lm) se substitue a la soustraction de la 2éme solution. ( 179-Y dans l'exemple ). Cela est possible car chaque commande BASIC en relation avec l'écran utilise la combinaison : RST 5 + DONNÉE Le RST 5 effectue en fait un CALL #28. En #28 se trouve la routine d'interruption 5 (Cf DAInamic NO 11 p180/181) celle-ci utilise un vecteur de saut qui se trouve en #6C/6D. Em mettant #300 dans #6C/6D a chaque ordre d'écran un call s'effectuera vers #300, avec dans les registres les valeurs a utiliser. Programme p290: En #300 une partie de la MEM a ete copiee, ce qui permettra un retour sans encombres au programme original. En #313 le programme vérifie qu'il a à faire à une fonction graphique. Si ce n'est pas le cas il retourne en MEM. Si il s'agit d'une fonction alors Y est soustrait à la valeur qui se trouve en #2FF.Après quoi on retourne en MEM, le BASIC n'ayant rien remarqué d'anormal ! Après avoir tapé le programme en lm il faut que vous adaptiez le pointeur BASIC ( #29B/C ). Apres cela vous retournez en BASIC et faites un NEW ou un CLEAR #100.

```
POKE #6C,0  :POKE #6D,3     Met en route le programme.
POKE #6C,#FD:POKE #6D,#C6   Revient en mode normal.
```

La valeur à laquelle est retranché Y peut etre POKEe en #2FF.
(remarque :sur APPLE YMAX=39 ou 179 en haute résolution)

Enregistrement en langage machine (UT)

Si en utilitaire vous tapez READ (au lieu de R) votre DAI essaiera quand meme de lire un programme! Ceci est du au fait que #EAD est une valeur hexadecimal licite. Le DAI lira le programme et le positionnera #EAD plus loin que son adresse initiale (Offset)

## SIMULATION DE 'REPT' EN BASIC

Dans la plupart des jeux d'actions, qui utilisent le clavier comme interface d'entrée, la touche REPT doit etre pressée si l'on veut avoir un mouvement continu. Pour eviter cet inconvenient, il est bien sur possible de faire sa propre scrutation du clavier en langage machine. Mais il y a aussi une solution simple en BASIC.

Lors d'une scrutation du clavier, les résultats du balayage précédent sont rangés aux adresses

# 2B1 à # 2B8

Résultats qui sont comparés avec ceux du balayage en cours, une touche n'étant prise en compte que si les résultats différent.Il faut donc imposer une difference entre ces deux scrutations, simplement en changeant les données du balayage précédent.

En ajoutant a votre programme une ligne telle que :

```
FOR I% = #2B1 TO #2B8 : POKE I%,0 : NEXT
```

Une touche pourra etre prise plusieurs fois en compte imitant ainsi l'action de REPT .

C. DUFOUR

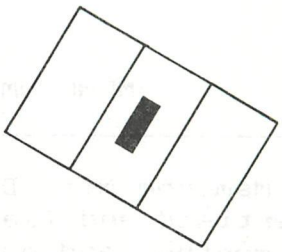
## DAI pc FIRMWARE MANUAL

The 'DAI pc FIRMWARE MANUAL' can be obtained from: 'Micro Service', Fabritiusstr.15, 6174 RG Sweikhuizen, The Netherlands.

It can be ordered by transferring Hfl. 68.- to bankaccount 13.05.78.754 of 'Micro Service' with the RABO-bank, Geleen, NL or by sending an Eurocheque to the above address.

When other cheques than Eurocheques are used, Hfl.75.- has to be paid due to transfer provision.

Nederlandse clubleden kunnen ook bestellen door overmaking van Hfl.68.- via postgirorekening 1037671 van de RABO-bank te Geleen ten gunste van bovenvermeld bankrekeningnummer.



HOW TO DUPLICATE LINES OF PROGRAM :

This trick comes from Roberto & Marco Bulgarelli, Rapallo (Genova-Italy), and it is useful when we have to write several lines equal between them apart from a little detail.

- 1) Edit the line
- 2) Change the number of line
  (be careful do not use number of line already existent)
- 3) Tape "BREAK" twice
- 4) Poke 309,2
- 5) List : the line is duplicated !

I am waiting for your answer, thank for wishes, I wish a Happy New Year to the whole DAInamic.

Yours faithfully,
Marco Di Martino

Marco Di Martino

This is the first (small) contribution of DAInamic-ITALY. More articles and programs will follow, cause we have a lot of new Italian members.
Members from Italy can send their contributions to the following address :

MARCO DI MARTINO
Casella Postale 31
20090 LINATE-AEROPORTO
MILANO-ITALY

Marco will eventual arrange translations and send them to us for publications.

A WARM WELCOME TO ALL ITALIAN DAI-USERS !

## MEMOCOM DCR REVIEW by Dave Atherton

The Memocom Mini-Digital Cassette Recorder (MDCR-D) had been advertised and featured in DAInamic several times in the last few months, and as I did not expect to be able to afford discs, I decided to buy one. Also the slight experience I have had of DAI discs left no doubt in my mind that the Memocom was preferable for a home user such as myself.

The unit arrived, being a small (approx 12cm) cube with a microcassette door at the front and a DCE bus at the back. The drive is a standard Philips mechanism, and Memocom's own OEM bit is a small card on the inside of the rear plate. Also supplied was a flat cable with connectors, a manual and an EPROM on a card, socketed at the edge. The EPROM card slots onto the 'X-bus' on the DAI (no soldering required) and the flat cable is then inserted to connect the unit to the DAI DCE bus. This took about 5 minutes and presents no problems

Software - The EPROM is 2K long, and fills the empty space between :F000 and :F7FF. The contents can be examined by the monitor display command in the normal way. Once installed, the system boots to operating with the DCR for the LOAD, SAVE, CHECK, R & W commands. There are also the following new high-level commands:

| | |
|---|---|
| REWIND | - Rewinds to tape start |
| REWIND n | - Rewinds 7 files |
| SKIP | - Fast forward to end of 'used' tape |
| SKIP n | - Fast forward 5 files |
| CASSETTE n | - Switch to audio cassette n: or n=0 none, n=3 both |
| CASSETTE | - Switch to audio cassette 1 |
| DCR n | - Switch to DCR n (logically 4 allowed) |
| DELETE | - Wipe tape from present pos. to end |
| LAST | - Mark previous file as last one |
| LOOK | - Read & print file name, then rewind to start of that file |
| VERIFY | - As CHECK for previous file only |

The commands REWIND, CASSETTE, DELETE and VERIFY can be abbreviated to their first three characters. The number n can be any ASCII number between 0 and 9, but not a variable. The LAST command works by erasing a few inches of tape. This is because the system does not allow you to go forward and leave gaps of unused tape.(You can of course do this manually) so the LAST mark effectively deletes from recognition all further files on tape. When loading, say a specific named file (Type 0 1 or 2) the system starts moving the tape forward until it finds the required file. If when the free tape is reached, the file has not been found, then an automatic rewind to the start is performed, and the search continues. Therefore it is possible to load files which have already been passed

**contactaddress U.K.** : Dave Atherton
16 Douglas Street
ATHERTON MANCHESTER M29 9FB
U.K. tel : 44-942 876210

on the tape. Thus the system does have a 'random access' facility. The drawback however is that in the worst situation where the full tape contained files, and you had just passed the file you required, the access time would be 2 x 95 secs. It should be noted that the fast forward and reverse speeds are the same as the loading speed, approx 11 i.p.s.

All the commands mentioned above can be entered and executed in immediate mode. However, as probably know, the DAI BASIC is encoded on entry and therefore it is not possible to enter the commands as they stand in BASIC program lines. To get round this problem, Memocom have devised an ingenious routine that reads and executes REM statements. The entrypoint for this routine is :F000 and therefore say, to skip 3 files the program line would be:

```
100 CALLM #F000:REM SKIP 3 files
```

Anything extra in the REM is ignored if a 'number' command is used. A cassette/DCR switch could be:

```
100 PRINT "Press C for CASS. or D for DCR"
110 G%=GETC:IF G%<67 OR G%>68 GOTO 110
120 IF G%=67 THEN CALLM #F000:REM CAS
130 IF G%=68 THEN CALLM #F000:REM DCR
```

Another feature of the Memocom software is error-trapping on I/O. The normal LOAD errors 0-3 exist and there are two saving errors 1 & 2 indicating 'tape door opened' and 'end of tape' respectively. Rather than crash, the error-trapper can be initialised by CALLM #F003,N% where N% is any integer variable Then when an error occurs, the variable N% will be loaded with the number of the error. A useful operation of this is described in the manual, viz several attempts to LOAD files i.e:

```
100 CALLM #F003,ERROR
110 TRIES=0
120 LOADA A$ "NAME"
130 TRIES=TRIES+1
140 IF ERROR=0 OR TRIES>4 GOTO 170
150 CALLM #F000:REM REW
160 GOTO 120
170 REM File has loaded OK or 4 attempts have failed
```

This program does not distinguish between the errors but of course a program could if desired, if only to print messages such as "CHECKSUM ERROR" etc.

For assembly language programmers, the manual, although short (20 pages) provides the addresses to call for each software routine. Also the system variable addresses (below :2EC are also all provided. The ROM has a jump table like the RST calls to paged ROM, and the calls are :F000 to :F01E at 3 byte intervals of course. I have done a couple of short assembled routines to test this, and everything worked fine. The ROPEN, RBLK etc vectors are loaded with EPROM addresses when the system boots, so special R/W routines written for cassette will work provided they call those vectors and not

the direct ROM addresses. For hardware geniuses it may even be possible to use the custom software to drive their own peripherals through the DCE Bus. (an EPROM Burner?). Also the table for the CALLM #F000 routines is pointed to by :297/8 so if this were altered other new words would be a possibility. See Mr Boerrigters article on this in DAInamic 12. p.264. I have not had a chance to experiment with this yet, but if I do, I will report the results. (If terribly successful I'll submit them to DAInamic for larger circulation)

The system contains a facility for auto-start of programs. When the DAI is hard reset, as part of the reset routine, it checks the DCR, and if it finds a write-disabled (i.e. plug removed) cassette in the drive, rewinds it. It then attempts to load the first file. If and only if this is a type 1 file called "USER" it loads it without offset, and commences execution from the first address loaded. If the cassette is absent, or write-enabled, or the first file is not Type 1 or not called "USER" then no load takes place. Of course the first file can be a routine to load a BASIC program, and indeed Mr De Raedt has written one - see DAInamic 12, p.283.

I have had a couple of problems with this procedure however, both booting the assembler direct and loading the Word Processor using Mr De Raedt's program. I think they are something to do with stack levels. However, this facility has great possibilities in user-friendliness, in that all a user has to do is put a tape into the drive and switch on, the software on the tape giving operating instructions etc.

Finally, the command list above shows that logically four DCRs may co-exist each being software selectable. The number of the DCR is determined by DIL switches inside the case Memocom supply cables with multi-DCE bus connectors for this purpose and again here there are possibilities. With two DCRs it would be possible to have a read tape and a write tape, and at the end of a job, the latter holds the up-to-dat records, and becomes the read tape for future jobs. The old read tape would then be erased.

The Memocom MDCR-D is manufactured by Memocom Computer-Controlled Systems BV, Postbus 2294, 3000 CX, Rotterdam. Tel: 010-148284. The TOS software is copyrighted to them. The system is available in the UK from Codified Computer Systems, 255 Archway Road, London N6 5BS. Tel: 01-340 4582 Price (October 1982) £175.00 + VAT + carriage. The tapes used are Philips Certified Digital which are £23.50 + VAT for a box of 6 from Codified. Each holds 64K on each side. These are quite dear and I have used dictation machine cassettes which are about £2.00 each but they are prone to errors especially when they have been recorded on several times. The digital tapes are guaranteed to have an error rate of less than 1 bit in 100 million, so these can be treated as faultless. I think the Memocom is good value and provides an economical and powerful substitute for discs, for the home user. I also think that careful study of the firmware will provide some very useful routines.

# EXPLANATION OF THE FULL 32 DISPLAY MODES AND 136 COLOURS

by Louis Gidney, 114 Houghton St Giles, Walsingham Norfolk

The display mode of every line on screen is individually controllable and can be put into one of 32 available modes. Lines in different modes can be freely intermixed to give an enormous number of richly coloured screen formats. In many of these formats it is possible to make wide-scale changes on the screen by altering just one byte. For example it is quite easy to construct pictures in which the colours of 50 or more complicated shapes can be individually switched by single bytes between the 16 colours (or in certain circumstances 136 mixed colours). These many formats give rise to an assortment of animation techniques for, games, re-configurable patterns to accompany music, etc. which can give fast-moving results even when controlled by BASIC programs.

It is possible to put character-mode lines (in 4 widths) in the middle of graphics and although a single line cannot be in two modes at the same time (eg: chars and graphics), it is possible to freely superimpose characters (including user-defined characters up to 16x16 dots or bigger) on graphics in any position (even overlapping) using the FGT facility.

The height of any line can be chosen to be any even number of TV raster scans from 2 to 32. (A line of 2 scans is the minimum controllable line). If the height of all the lines on the screen is set to 2 scans, the total number of controllable lines is 625/2 = 312 (260 on US TV's). If all the lines are made 32 scans high, the number of lines over the full raster height becomes 625/32 = 20 (17 on US TV's). In practice, unless a TV is modified, the edges of the raster are hidden by the bezel, so 256 is a practical number of lines to use.

The "mode" of a line is fully specified, regardless of its height, by a combination of four attributes:

a). The number of colours desired (16 / or any 4 of 16)

b). Characters / or Graphics.

c). Horizontal resolution. (Low/ Medium/ High/ or Very High
ie 88,176,352 or 528 dots per raster scan-width
or 11, 22, 44, or 66 characters.)

d). Normal / or "Unit" Mode. In normal mode the display across one line varies along the line, and is controlled by many bytes. "Unit mode" makes it possible for two bytes to control a character or a horizontal pattern of 8 dots (vertical bars if the line-height is increased) which is repeated 11,22,44 or 66 times along a line. This line-mode has many uses. One example is the compacting of pictures in which horizontal uniformity or repetition occurs. For example a strip of uniformly blue sky 32 raster scans high in high resolution would normally be controlled by 90x16=1440 bytes. Compacted to a unit-mode line 32 scans high it can be controlled by only 4 bytes. (In some conditions only two!).

Here is a chart which summarises the relevant bits in the line control byte high nibble.

LINE CONTROL BYTE. High Nibble (Bits 7,6,5,4)

| MODE OF LINE | Bits 7,6 | Bits 5,4 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|
| 4 Colour Graphics | 00 | 0 | 1 | 2 | 3 |
| 4 Colour Characters | 01 | 4 | 5 | 6 | 7 |
| 16 Colour Graphics | 10 | 8 | 9 | A | B |
| 16 Colour Characters | 11 | C | D | E | F |
| Horizontal Resolution - | | LOW | MED | HI | V.HI |
| Dots per Full Raster Width - | | 88 | 176 | 352 | 528 |
| Dots used by BASIC - | | 72 | 160 | 336 | 480 |
| Characters per Full Raster Width - | | 11 | 22 | 44 | 66 |
| Characters over BASIC Width - | | 9 | 20 | 42 | 60 |
| Interval Line Ctl Bytes Graphics - | | 24 | 46 | 90 | 134 |
| Interval Line Ctl Bytes Characts - | | 26 | 48 | 90 | 134 |

To set any given line to the desired mode from the above table poke the high nibble of the line control byte with the correct hex digit from the matrix, and the low nibble with the number of scans of the TV screen that are desired for that line. The number of scans can be between 1 and 16. (i.e 0=1 scan, :F=16)

So where are the line control bytes. Well, the first one is always :BFEF, for the top line on screen. If you peek this you will see that it contains :7A i.e 4 colour characters, and 11 (not 10) scans. If you are not sure what scans are, poke :BFEF with :73 (4 scans) while the top line is full of text. Now the subsequent line control bytes are harder to find. Working down the screen from the top, take each line control byte, test both the resolution and whether it is Graphics or Characters then subtract the appropriate figure (last two lines on chart) plus one (for the line colour byte) to find the new line control byte. So thats how the control bytes work. In Part 2 of this article I will explain how the line colour byte works and show how 136 different colours can be generated on screen simultaneously.

Louis Gidney

10.

## ADVERTISEMENTS

If you have something to sell please give details and your ad will be carried. Due to the tiny circulation at present, no charge will be made, and probably never for private members. If you want to do a full page ad, please submit it on the paper size that this newsletter is printed on (9.5" x 11.5") Please do the photocopying yourself and send me enough copies for the next issue. (Phone for an estimate). However you may wish to sell your product by writing an article about it, in which case it should be submitted in the normal way. You can also lend your wares to someone else, for the purpose of writing a review.

Inclusion of an article for sale in these pages does not mean endorsement by the Group, and all transactions to be conducted directly between vendor and purchaser (except where outsiders offer club discounts in which case special arrangements will be made).

---

## HINTS & TIPS

This section is for small items discovered that can be useful in program writing and use. I will start the ball rolling this time with some of my own. Please send your own bits and pieces for this page.

*Many of your programs will display some text, and wait for the spacebar to be pressed before moving on. Instead of the typing: 100 G=GETC:IF G<>32 GOTO 100 just use CALLM #D6DA which has the same effect.

*If you want to disable the whole keyboard during a BASIC program, including the BREAK key, just POKE #5F,#84. The keyboard will re-enable at the program end, or when you POKE #5F,#C4.

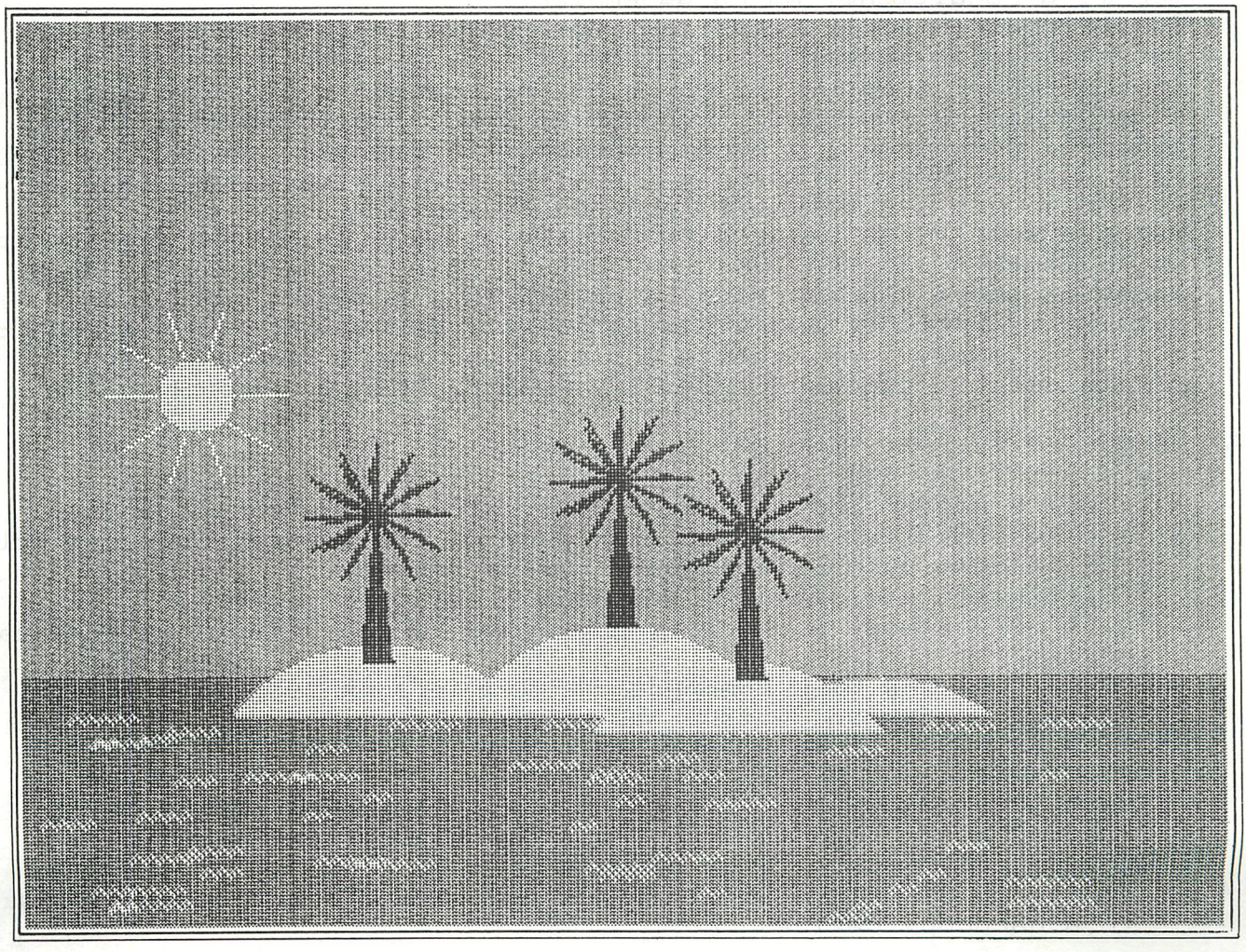
*A common feature of programs is to test GETC for a particular key but to disregard whether it is upper or lower case. Rather than G=GETC:IF G=#41 OR G=#61 GOTO ... to test for the 'A' key, just use G=GETC:IF G IOR #20=#61 GOTO ... The IOR converts the value to lower case if it is upper. (i.e. sets Bit 5).

*For a neater printed menu etc, blank out the screen i.e with COLORT 8 8 8 8 before your printing lines, then reset the colours i.e. COLORT 8 0 8 0 after the printing, for an apparently instantaneous image.

*You will know that POKE #75,32 blanks out the cursor by making it a SPACE character. However this is not always satisfactory as if the cursor slides over screen area that already contains characters, it can be seen momentarily blanking out the characters. To avoid this, ignore the number in #75 and POKE #74,0. Your 3rd and 4th COLORT must be the same as the first two. i.e. COLORT 8 0 8 0. The cursor will then truly disappear. To see the difference try sliding the cursor over a line in the editor in each mode.

```
100   REM *** GROETEN UIT HAWAI ALOHA OE *********
110   REM *** GESCHREVEN DOOR : DE BONT CORNEEL **
120   REM ********************** OOSTEINDE 10 *****
130   REM ***************** 2338 BAARLE-HERTOG ****
140   REM ********************** 28 - 1 - 1983 ****
150   REM ****************************************
160   GOTO 1000
200   REM *** HELE OF HALVE CIRKEL
210   FOR Z=D1 TO D2:S=SQR(D2*D2-Z*Z)
220   DRAW XC-S,YC+Z XC+S,YC+Z 22:NEXT:RETURN
300   REM *** GOLVEN
310   NG=INT(5.0*RND(1.0))+1.0
320   FOR X=0.0 TO NG:DOT XG,YG-1 22:XG=XG+1.0
330   DOT XG,YG+0 22:XG=XG+1.0:DOT XG,YG+1 22:XG=XG+1.0
340   DOT XG,YG+0 22:XG=XG+1.0:NEXT:RETURN
400   REM *** PALMBOOM
410   DI=8.0:FOR Y=YP TO YP+40.0:DRAW XP,Y XP+DI,Y 21
420   DI=DI-0.2:XP=XP+0.1:NEXT:ST=PI/60.0
430   FOR X=0.0 TO 2.0*PI STEP PI/6.0
440   CO=COS(X):SI=SIN(X):DRAW XP,Y XP+20*CO,Y+20*SI 21
450   CO=COS(X+ST):SI=SIN(X+ST)
460   DRAW XP+3.0*CO,Y+3.0*SI XP+17*CO,Y+17*SI 21
470   CO=COS(X+ST+ST):SI=SIN(X+ST+ST)
480   DRAW XP+6.0*CO,Y+6.0*SI XP+14*CO,Y+14*SI 21
490   NEXT:RETURN
1000  REM *** INLEZEN NOTENARRAY
1010  CLEAR 3000:DIM F(60.0)
1020  MODE 0:PRINT CHR$(12):POKE #75,32:RESTORE:SOUND OFF
1030  ENVELOPE 0 15
1040  ENVELOPE 1 12,5;1,10;0
1050  SOUND OFF :NOISE OFF :PRINT CHR$(12);:COLORT 9 14 9 9
1060  COLORG 12 5 14 9:MODE 6A:POKE #FF05,2
1070  POKE #7556,222:POKE #74D0,211
1080  POKE #744A,213:POKE #73C4,220
1090  FOR I=141.0 TO 211.0:J=#BFEF-I*90.0-2.0:POKE J,119
1100  POKE J-1,143:POKE J-86,255:POKE J-87,255:NEXT
1110  FILL 0,0 335,70 23:I=0.0
1120  CURSOR 11,3:PRINT "GROETEN UIT DE STILLE ZUIDZEE."
1130  XG=INT(300.0*RND(1.0)):YG=INT(60.0*RND(1.0))+2.0
1140  GOSUB 300:G=GETC:IF G<>0.0 THEN 1160
1150  I=I+1.0:IF I<50.0 THEN 1130
1160  CURSOR 16,2:PRINT "GROETEN VANUIT HAWAI"
1170  D1=25.0:D2=50.0:XC=167.0:YC=35.0:GOSUB 200
1180  D1=30.0:D2=50.0:XC=100.0:YC=30.0:GOSUB 200
1190  D1=30.0:D2=50.0:XC=200.0:YC=25.0:GOSUB 200
1200  D1=40.0:D2=50.0:XC=240.0:YC=20.0:GOSUB 200
1210  D1=-10.0:D2=10.0:XC=50.0:YC=150.0:GOSUB 200
1220  FOR X=0.0 TO 2.0*PI STEP PI/5.0:S=SIN(X):C=COS(X)
1230  DRAW 50+12*C,150+12*S 50+25*C,150+25*S 22
1240  NEXT
1250  XP=200.0:YP=70.0:GOSUB 400
1260  XP=164.0:YP=85.0:GOSUB 400
1270  XP=96.0:YP=75.0:GOSUB 400
1280  CURSOR 14,1:PRINT "OF ZOALS MEN DAAR ZEGT :"
1290  CURSOR 21,0:PRINT CHR$(34);"ALOHA OE";CHR$(34);
1300  SOUND OFF :NOISE OFF :POKE #FF05,255
1500  REM *** MUZIEK
1510  RESTORE:FOR I=0.0 TO 57.0:READ F(I):NEXT:TE=7.0
1520  FOR I=1.0 TO 84.0:READ T,W:S=F(T)
1530  SOUND 0 0 15 0 FREQ(S*4.0):SOUND 1 0 15 0 FREQ(S)
1540  SOUND 2 1 15 0 FREQ(S*2.0):WAIT TIME TE*W:NEXT
1550  SOUND OFF
1560  FOR I=0.0 TO 3.0:CURSOR 0,I:PRINT SPC(58);:NEXT
1570  POKE #FF05,2:CURSOR 27,0:PRINT "EINDE";:CURSOR 27,1
1580  PRINT "ALOHA":CURSOR 25,2:PRINT "TOT ZIENS"
1590  POKE #75,95:POKE #FF05,255:CURSOR 0,3
```

```
2000  REM *** NOTEN VAN 4 OKTAVEN F(57)
2010  DATA 20000,20000
2020  DATA 065,069,073,078,082,000,087
2030  DATA 092,098,104,110,116,123,000
2040  DATA 131,138,147,155,165,000,175
2050  DATA 185,196,208,220,233,247,000
2060  DATA 262,277,294,311,330,000,349
2070  DATA 370,392,415,440,466,494,000
2080  DATA 523,554,587,622,659,000,698
2090  DATA 740,784,831,880,932,988,000
2100  REM *** F,T (5 * PER REGEL)
2110  DATA 24,2,30,2,34,2,34,4,32,2
2120  DATA 30,3,28,1,30,2,26,2,24,8
2130  DATA 24,4,00,2,34,2,34,2,32,2
2140  DATA 32,4,31,2,32,2,30,2,36,3
2150  DATA 34,1,32,8,32,4,00,2,24,2
2160  DATA 30,2,34,2,34,4,32,2,30,3
2170  DATA 28,1,30,2,26,2,24,8,24,4
2180  DATA 00,2,30,2,28,2,26,4,32,2
2190  DATA 30,2,28,4,34,2,32,2,30,8
2200  DATA 30,6,00,2,24,2,26,4,30,4
2210  DATA 36,6,26,2,24,4,30,4,34,6
2220  DATA 00,2,30,2,28,3,26,1,28,2
2230  DATA 30,2,32,2,32,2,36,4,34,8
2240  DATA 30,6,00,2,24,2,26,4,30,4
2250  DATA 36,6,26,2,24,4,30,4,34,6
2260  DATA 00,2,30,2,28,6,30,2,34,2
2270  DATA 32,4,28,2,30,8,30,4,00,2
2280  REM --------- EINDE ---------
```



```
*****************************************
*                                       *
*  S P L : A NEW ASSEMBLER FOR THE DAI PC  *
*                                       *
*****************************************
```

As announced in DAInamic 13, workgroup SPHYNX (located in the Netherlands) has designed an assembler with many special features. SPL was created exclusive for the DAI pc with the intention to allow easy assembly of large machine language programs.

## 1) FEATURE LIST

### 1.1) MEMORY USAGE

Unlike DNA the SPL is a lot more economical with memory. This is accomplished by using a single buffer storage and a special format for source code store.
8080 instructions are stored as single bytes codes. Symbols are kept in a symboltable and in the sourceline only references into the symboltable are included (this is the same method as used by DAI basic). The only thing which takes as many bytes as characters shown on a listing are comments.
As illustration the whole source for SPL (which generate 12 kbyte code) can be loaded, editted and assembled with SPL.

### 1.2) CONDITIONEL MACRO ASSEMBLER

SPL is a conditionel macro assembler.
Conditional assembly means that some parts of the source code are ignored during assembly. Control of this is done by using IF ELSE ENDIF constructs (which may be nested). As condition any comparation may be used. Purpose of conditional assembly is to generate more than 1 program version with the same source (eg make a game with keyboard or paddle input, or include multi-language menu's in the source and select 1 language at assembly time).
Macro assembly can be used to declare a sequence of mlp instructions, which perform a given function, only 1 time in the source. Each location where the function is needed in a program, only a macro request is included and SPL will insert the sequence of instructions during assembly. An example could be the macro PSHALL, which references the sequence PUSH PSW, PUSH B, PUSH D, PUSH H.

### 1.3) SOURCE EDITOR AND HANDLER

Full-scale editor and source handler, with followimg commands :
- source code list from whole or part of the source, where selection can be done using linenumbers or/and labels
- full-screen editing (DAI editmode), selection idem as for list ; on exit of editmode a check is performed to catch typing and syntax errors
- MOVE, COPY and KILL commands for sourcelines
- replacing a symbol with a new symbolname
- find in which lines a label or a symbol is used
- READ, WRITE and MERGE commands
- configure memory-map (dimension and location of source-buffer)
- display memory-map to verify spare area source-buffer
- inline call from all dcr commands
- configure hardcopy output (lines/pages) and send control characters to a hardcopy device

## 1.4) ASSEMBLY COMMANDS

Assembly monitor with following commands :
- object list (hex + source), with LST UNL control
- poke object code (offset possible)
- generate an error list (eg not-defined symbols)
- symbol table list
- and other commands to help program design and test

Due to the way SPL stores his source, assembly runs very fast.

## 1.5) SPL LANGUAGE

- SPL uses the normal INTEL 8080 mnemonics for opcodes and pseudo-operations
- SPL implements a large subset of the MACRO-80 (Mikrosoft) or ISIS-II 8080 MACRO-ASSEMBLER (Intel) features
- operators for operands : add, subtract, dived, multiply, modulo, shift, and, or, xor
- operators for conditions : <, >, =, <>, or, and
- all operators are single characters (eg * " #)
- symbols may be 6 characters long and lower-case characters must be used for naming macros

# 2) REQUIREMENTS

## 2.1) MEMORY AND ROM VERSIONS

SPL as delivered will only run in DAI systems with 48k RAM.
Generic code of SPL starts at hex address B400 and ends nearly at begin of screenram. When using SPL switching to a graphic mode is not possible, as it would destroy the SPL code.
A version for 32k RAM will only be supplied if sufficient requests reach us.
SPL was tested with ROM V1.0 and V1.1.

## 2.2) PERIPHERALS

SPL is tested with DCR software DTOS1.2.
Due to location of SPL no memory overwrite problem should exist for using a floppy system which uses lower part of RAM to store the disk-operating system. However as no high-quality floppy system is available until now, Dainamic had not tested SPL-floppy interface.
As delivered SPL will use the RS232 output to drive a printer, but SP can be configured to drive a parallel printer interface (interface description in the manual).

# 3) EVALUATION

## 3.1) BY DAINAMIC MEMBERS

In the past 2 months SPL was distributed to a selected group of members, this to perform a pre-release test and evaluation.
Reactions were channeled to the designers and updates were made to code or/and manual.
In general the results of the evaluation were very good and we think SPL will became an important enhancement for machine language programming on the DAI pc.

## 3.2) COMPARE WITH MACRO-80

For those members who know the MACRO-80 assembler, an (incomplete) list of M80 features not implemented in SPL :

- ASEG, CSEG, DSEG location counters
- EXTRN, PUBLIC, COMMON, NAME , .REQUEST : SPL doesn't generate linkable object modules
- IFDEF, IFB special conditional tests
- IRP, IRPC repetition controls (normal REPT is implemented in SPL with ### pseudo)
- LOCAL labels in macros (SPL uses a kind of SET pseudo to avoid multi-defined errors, however this seems not to have some restrictions)
- parameters for macros cannot be register names

NOTE : MACRO-80 is an extremely powerfull assembler which requires a floppy-system (eg CP/M) and which is sold for about 10.000 Bfrs. Also MACRO-80 does only the assembly job. Sourcecode editting and objectcode linking must be done with other programs.
Most of the above mentioned differences can be overcome by using some other instructions. Problem of module linking is not so important, because most applications will fit in 1 SPL source file (see 1.1).

## 3.3) SOME POINTS ABOUT USE

Numeric constants must always be entered with a B, D or H appended, no default is provided. In a list appearing of these constants in decimal or hexadecimal format must be controlled by using PUT directives.
Due to single character commands (the whole alfabet including some lower case characters) some skill is needed to use them without looking back at the manual.
Same thing applies for single character operands and meaning of single or double character error reports.

## 3.4) CONCLUSIONS

SPL is a good compromise between a professional full-line assembler and a system environment as the DAI pc.
SPL don't require floppy's to retrieve records of source but allows that the source is stored in ram in a compacted format.
Used solutions to minimalise SPL generic code (such as single character commands and operators) are easy to live with.

# 4) PACKAGE

SPL is delivered with a dutch and english manual.
Included on tape are following source-programs :

- IMPLEM : to patch SPL (eg parallel printer, screen colors, lines/page)
- RESTORE XXXX : example SPL source
- DISPL : a very powerfull 8080 disassembler which directly generates SPL source

Planned to be included :

- CONVERT : convert DNA source to SPL format

It is also planned to make the SPL source listing available on request.

# 5) DELIVERY

PRICE : audio tape : 1.100 Bfrs
dcr tape : 1.250 Bfrs
SCHEDULE : delivered from march 83 on
COPYRIGHT : Workgroup SPHYNX Netherlands
SELLING RIGHTS AND ONLY PLACE FOR ORDERS :
DAInamic vzw Westmeerbeek Belgium

Geachte heren,

Met belangstelling heb ik het artikel FGT-DISK-PEEK-POKE in nummer dertien gelezen.De methode die de heer Gesp gebruikt vind ik nogal omslachtig en maakt geen gebruik van de mogelijkheden die het DOS ons bieden.

Wanneer we het FGT programma deel laten uitmaken van het DOS zal dit bij het inschakelen samen met het DOS ingeladen worden en zal er blijven zitten zolang we willen.Ook brengt het splitsen van het programma voordelen met zich mee.Het is immers enkel de tabel die veranderd als we een ander lettertype willen.Wanneer we van de tabellen D-files maken kan een programma over meer dan een lettertype beschikken.

Voor we zover zijn moeten we echter een beetje goochelen maar er zijn geen trucjes mee gemoeid.
Het nieuwe DOS bestaat uit :

| | | |
|---|---|---|
| 1. | Het originele DOS | Van 02E3 tot 19CB |
| 2. | Het FGT programma | Van 19D0 tot 1C05 |
| 3. | De FGT tabel | Van 1C10 tot TBLEND |
| 4. | Vrije ruimte | Van TBLEND tot 27E0 |

TBLEND is het einde van een FGT tabel.
De vrije ruimte is nodig om ook de langste tabel in het DOS te laten passen.TBLEND van de langste tabel is dus 27E0.
Wanneer het nieuwe DOS geladen is zal de HEAP op 27E1 beginnen.Er is dus nog plaats voor een fors programma.
Op de systeem schijf staat buiten het DOS ook de verschillende tabellen zodat ze beschikbaar zijn voor de programmas die er gebruik van gaan maken.

Hoe gaan we nu te werk ?
Na HARD=RESET laden we de FGT met de langste tabel en gaan met UT kijken waar deze eindigd.Daarna laden we de FGT met standaard tabel en gaan nu het geheel van DOS en FGT op een systeem-schijf zetten waarvan $MSTRDOS is ge-deleted.
Tik in : DSAVE $MSTRDOS:0 2E3 27E0 13EC [RET]
en onze nieuwe DOS is klaar.Het adres 27E0 kan in uw geval anders zijn,het hangt van de lengte van de door u als langste tabel beschouwde tabel af en van de locatie van uw FGT.
We laden nu achtereenvolgens de programmas met de verschillende tabellen en DSAVEn enkel de tabellen op de schijf met een passelijke naam.

Hoe laad een programma zijn tabel ?
We gebruiken hiervoor het commando DLOAD maar plaatsen het tussen de gebruikelijke poke's.Voorbeeld :

```
100   POKE#131,3:PRINT "DLOAD MATH:0":POKE#131,1
```

Deze programma regel hoeft slechts eenmaal uitgevoerd en kan desgewenst in een zelfde programma herhaald worden om over een ander lettertype te kunnen beschikken.Het ganse programma hoeft niet langer in een array geladen te worden en op z'n plaats gePOKEd te woden :tijdwinst en plaatswinst.

Om het geheel te proberen stel ik voor dat u nu een beetje speelt met de demo-programmas die aangepast zijn en eveneens op deze schijf staan.

Met de meeste hoogachting,

Couwberghs Frans
Boekdonkstraat 13
3980 Tessenderlo
013/666340

## program identification

| | | |
|---|---|---|
| title | : | DAI-FLOPPIES (the "old" ones) |
| author | : | Mr. Baptiste |
| purpose | : | Explains how to solve some DOS-problems |
| comment | : | This information should be in the manuals ! |

APRES 8 MOIS DE TRAVAIL SUR DISK ET PLUS SPECIALEMENT EN TRAVAILLANT SUR DES FICHIERS, JE VOUS LIVRE MES CONCLUSIONS.

UNE DISQUETTE DOIT ETRE STOCKEE A L'ABRI DE LA POUSSIERE, DANS UNE BOITE FERMEE,PAS A PROXIMITE D'UN CHAMP MAGNETIQUE (ENREGISTREUR,HAUT PARLEUR,ARRIERE DE L'ORDINATEUR,SUR L'UNITE DE DISK CAR UN TRANSFO SE TROUVE A DROITE). J'AI PERSONNELLEMENT PERDU 1 DISQUETTE EN LA DEPOSANT SUR LE DISK DRIVE A L'ARRIERE DROIT OU SE TROUVE LE TRANSFORMATEUR.

QUAND LE DISK EST FROID, IL PEUT PROVOQUER DES RATES DE LECTURE .LES RATES SONT PLUS FREQUENTS DANS LES LECTURES DE FICHIERS QUE DANS LES LECTURES DE PROGRAMMES.
POUR EVITER LES ENNUIS,IL VAUT MIEUX CHAUFFER LES AIGUILLES EN EFFECTUANT UN VERIFY SUR LE DISK 0 ET DISK 1 AVANT DE TRAVAILLER.

L'ADRESSE #9B7 SIGNALE SI UNE ERREUR A ETE COMMISE SUR LE DISK DRIVE.MAIS AVANT TOUTE INSTRUCTION, ON DOIT METTRE LE BYTE #9B7 A 0;LE DOS NE LE FAIT PAS LUI-MEME.

LE NOM DES FICHIERS NE PEUT CONTENIR DE BLANCS, CAR LE DOS ARRETE SA LECTURE DES LE PREMIER ESPACE ET IGNORE LES RENSEIGNEMENTS SUIVANTS (SAUF POUR LES INSTRUCTIONS SAVE,SAVEA,LOAD,LOADA,QUI NE FONT PAS PARTIE DU DOS).

LE DOS EMPLOIE UNE TECHNIQUE QUI PEUT INTERESSER LES PERSONNES N'AYANT PAS DE DISK DRIVE:L'APPEL D'UN ASSEMBLEUR AVEC UNE INSTRUCTION :L'APPEL SE FAISANT EN DONNANT LE NOM DE L'INSTRUCTION AU CLAVIER OU PAR PROGRAMME
PAR 10 POKE #131,3:PRINT "INSTRUCTION":POKE #131,1
POUR CELA IL FAUT CHANGER LE VECTEUR KEYBOARD PAR
E5 2A 6E PAR C3 4A 0A
L'INITIALISATION DANS LE DOS SE FAIT PAR LA ROUTINE 13EC
L'ADRESSE DE LA TABLE SE MET A 297 (VOIR 13FC)

```
13FC LXI H(#1837)
     SHLD (#297)

0A4A PUSH H
0A4B LXI H((A51)
0A4E XTHL
0A50 PCHL
```

CETTE ROUTINE SERT A EXECUTER #A51 SI LE MOT ENCODE NE SE TROUVE PAS DANS LA TABLE DE ROM.
LA ROUTINE #A51 SERT A CONSULTER LA TABLE DU DOS.
LE VECTEUR 2DD (CHARACTER OUT) EST ARME ET CONTIENT C3 CF 0A.
SI ON LE COUPE,L'EXECUTION D'UNE INSTRUCTION DU DOS PAR PROGRAMME EST IGNOREE.(JE CROIS QU'IL Y AURAIT INTERET A BIEN COMPRENDRE LES ROUTINES A4A-ADA ET 1746-1834 POUR PERMETTRE AUX UTILISATEURS SANS DISK D'AVOIR LA FACULTE D'APPEL D'UNE INSTRUCTION PAR CETTE TECHNIQUE.

POUR LA PARTIE PROGRAMMATION,LES VECTEURS 2C5 A 2EO SONT CHANGES
2C5 JUMP 6A7
2C8 JUMP 6D7
2CB JUMP 751
2CE JUMP 865
2D1 JUMP 879
2D4 JUMP 963
CETTE TABLE SE TROUVE EN C6E-C7D ET SE TRANFERE PAR TRANFERT BLOCK DE LA ROM PAR 32A JUMP (DE4F) (REF DAINAMIC 82/15)
LES VECTEURS CASSETTE SWITCH SONT RAPPELLES EN 32D-33C

L'INSTRUCTION CREATE QUI N'A PAS ETE BIEN EXPLIQUEE EST LIBELLEE COMME SUIT :
CREATE FILEMANE.STR:O 7 OU CREATE FILENAME.STR:O D (D=VARIABLE)
OU POKE #131,3:PRINT "CREATE "+D$+" "+D AVEC D$="FILENAME.STR:O"
CECI EXECUTE L'OUVERTURE D'UN FICHIER STR DE NOM FILENAME ET DE 7 SECTEURS DE LONGUEUR.

POUR SAUVER UN FICHIER,PLUSIEURS ENNUIS PEUVENT SURVENIR.
-LE FICHIER EST RATE A L'ECRITURE PARFOIS PAR UNE TETE DE LECTURE TROP FROIDE OU TROP CHAUDE OU UNE CAUSE INDETERMINEE (TRAVAIL INTENSE ET RAPIDE MAIS RARE;JE SUIS PARVENU A TROP CHAUFFER LA TETE PAR UN PROGRAMME DE TRANSFERT RAPIDE D'UN DISK VERS L'AUTRE EN PASSANT PAR LE DIRECTORY DU DISK 1 ET AVEC UNE LONGUEUR DE FICHIER DE 1 SECTEUR: LA PREMIERE FOIS JE SUIS PARVENU A FAIRE PASSER 80 SECTEURS PUIS UN ARRET ;JE L'AI RELANCE AUSSITOT,IL N A FAIT QUE 10 SECTEURS ET AU TROISIEME ESSAI IL N'EN A MEME PAS FAIT UN).LE DIRECTORY DANS CE CAS N'EST PAS TOUCHE.
-LE FICHIER EST CREE MAIS IL Y A UN FICHIER VIDE DE LONGUEUR 1 AVEC LE NOM INSCRIT DANS LE DIRECTORY ET UN VERIFY EXACT.
-LE FICHIER EST CREE ET LE NOMBRE DE SECTEURS EST EXACT MAIS LE FICHIER NE PEUT ETRE LU (PEUT ETRE CONTRE PAR UN VERIFY)
-SI UN FICHIER A UNE LONGUEUR EGALE A UN MULTIPLE DE 128 BYTES, IL Y A DANGER DE BLOQUAGE.LE DOS OUVRE UN SECTEUR DE TROP OU IL SE TROUVE DES #55 .A LA LECTURE,CE SECTEUR EST LU COMME UN FICHIER ET IL APPARAIT DES "UUUUU".
MAIS PARFOIS LA LECTURE EST IMPOSSIBLE;LE FICHIER EST PERDU. POUR RETABLIR LA LECTURE,IL FAUT REMPLIR CE SECTEUR DE CODES OO (VOIR PLUS LOIN POUR LA TECHNIQUE)
POUR S'ASSURER QU'UN FICHIER A ETE BIEN ENREGISTRE,LE MEILLEUR MOYEN QUE JE VOIS EST DE CREER UN AUTRE FICHIER, DE FAIRE LOADA ET DE COMPARER LES DEUX FICHIERS. CETTE TECHNIQUE NE PEUT ETRE FAITE QU'AVEC DE PETITS FICHIERS.

SI ON SAUVE UN FICHIER QUI EST DEJA SUR LA DISQUETTE ET QUE LE FICHIER QUE L'ON SAUVE EST PLUS GRAND QUE LE FICHIER SUR DISQUETTE,IL Y A APPARITION DU FATIDIQUE "END OF FILE" QUI SIGNIFIE ICI FICHIER DETRUIT.POUR EVITER CECI,ON DOIT FAIRE UN "RENAME" DU FICHIER SUR DISQUE,PUIS "DELETE" ET ENFIN SAUVER LE NOUVEAU FICHIER.
REMARQUE:LE NOM D'UN FICHIER DETRUIT PAR "DELETE" RESTE DANS LE $DKDIR ET EST RECONNU LORS D'UN SAVEA.

UN AUTRE PROBLEME,LES MESSAGES D'ERREUR NE BLOQUENT PAS LE PROGRAMME BASIC. AVEC LES INSTRUCTIONS DE LA ROM,UN NUMBER OUT OF RANGE,... ETC ARRETE L'EXECUTION DU BASIC. ICI PAS, LE PROGRAMME PASSE A L'INSTRUCTION SUIVANTE.

POUR LIRE OU ECRIRE UN SECTEUR,ON EXECUTE CE QUI SUIT SOIT EN UTILITY SOIT EN ASSEMBLEUR:
A=NUMERO DU DISK DRIVE #30 OU #31 ("0" OU "1" EN ASCII)
B=NUMERO DE PISTE #0 A #23 (35 DECIMAL)
C=NUMERO DE SECTEUR #1 A #12
HL=ADRESSE OU ON DESIRE PLACER LE SECTEUR (LE DOS LE PLACE EN #1945 (ref #D70)POUR SON BACKUP OU COPY .LE NUMERO DU DISK DRIVE EN #19C5 (ref #D6D);MAIS L'ADRESSE POUR CETTE ROUTINE EST LAISSEE AU BON VOULOIR DU PROGRAMMEUR.

POUR LIRE UN SECTEUR,EXECUTER #A1C
POUR ECRIRE UN SECTEUR,EXECUTER #A32
EN CAS D'ERREUR ON AURA AU RETOUR LE REGISTRE A<>0
UNE AUTRE ZONE MEMOIRE SE TROUVE EN #977- #9B9 AVEC
EN 992-99C LE NOM EN 8 LETTRES D'UN FICHIER OU D'UN PROGRAMME
SI - DE 8 LETTRES,LE DOS COMPLETE AVEC DES BLANCS (#20)
SUIVIS DE BAS,BIN,STR,INT,FPT(CES NOMS SE TROUVENT
EN #761-#76F DANS LE DOS)
EN 9A7 NUMERO DU DISK EN ASCII
EN 9A8 TYPE DE FICHIER QUE L'ON A :
0 BAS
1 BIN
2 FPT
3 INT
4 STR
EN 9A9 LONGUEUR REELLE DU NOM SANS LES ESPACES
EN 9AC-9AD : COPIE DE 2A3-2A4
EN 9B1 NOMBRE DE BITS A CONSIDERER POUR ARRETER LA LECTURE/ECRITURE
DES MOTS D'UN FICHIER (POUR UN FICHIER STR,#9B1=#FF;POUR UN
FICHIER BAS,#9B1=#0F (JE N'AI PAS COMPRIS POURQUOI #0F)).
EN 9B7 SI <> 0 ERREUR
LA ROUTINE #14EF SERT A ALLER CHERCHER UN PROGRAMME SUR
DISQUETTE ET A LE LANCER.
CALLM#300,N$:AVEC N$=NOM DU PROGRAMME DEMANDE
HL POINTE VERS LE NOM DU PROGRAMME
EN 300 JUMP (14EF)
LA ROUTINE #5CE SERT A IMPRIMER LES MESSAGES D'ERREUR
ELLE SE TROUVE A LA FIN DE TOUTES LES INSTRUCTIONS ET EST
APPELEE PAR CNZ(5CE)
LES ROUTINES DE DISCUSSION AVEC LE DISK DRIVE SONT
AU NOMBRE DE 5.
387 SE TROUVE EN PREMIER LIEU DANS LES INSTRUCTIONS
PROBABLEMENT UNE ROUTINE D'INITIALISATION
3B3 ENVOI D'UN CARACTERE VERS LE DISK DRIVE
3D6 RECEPTION D'UN CARACTERE DU DISK DRIVE
4AA NON COMPRISE
3A5 NON COMPRISE
PAR LA ROUTINE 3B3,LE DOS ENVOIE UN CODE CORRESPONDANT A CERTAINES
INSTRUCTIONS:CELA SE FAIT PAR LA ROUTINE 402 AVEC UNE SEQUENCE
DE CODES STOCKEE EN 980 981 982 ...ETC
5A EXECUTE LE RESETD
49 A :A=NUMERO DU DISK DRIVE:IDISK(LE IDISK PEUT AUSSI ETRE
EXECUTE PAR 9C7 AVEC A=NUMERO DU DISK
4E A B C :A=NUMERO DU DISK
B C NOMBRE DE SECTEURS CREES
CELA SEMBLE ETRE UNE SORTE DE CREATE
53 LECTURE
54 ECRITURE
4F A SUIVI DE SOIT 49 SOIT 4F
52
43
4B
JE NE COMPRENDS PAS ENCORE BIEN CES DIFFERENTS CODES ET LEUR
FONCTION.
CONCERNANT LE PROBLEME DU FICHAGE, DEUX SOLUTIONS
SE PRESENTENT:
SI L'ON A UN FICHIER AVEC UN NOM ET DES VALEURS
SATELLITES IMPORTANTES

EXEMPLE:
BAPTISTE / alain/ 6 PLACE DE DINANT/ 1000/ BRUXELLES
NOM ********VALEUR SATELLITE**********
ET QUE CES VALEURS ET LE NOM FONT PLUS DE 612/107=+-6 SECTEURS
ON CREE UN FICHIER A$(4)
AVEC A$(0)="BAPTISTE"
A$(4)="BRUXELLES"
ET ON FAIT SAVEA A$ "BAPTISTE". ATTENTION,LE NOM <= 8 LETTRES
SI PAR CONTRE,LA TAILLE DU FICHIER EST < 6*128 BYTES ET SI ON A
PLUS DE 107 NOMS,ON OUVRE UN FICHIER A$(10,4) PAR EXEMPLE AVEC 11 NOMS
ET ON SAUVE EN BLOCS DE PLUSIEURS NOMS.
SI L'ON A QUE DES NOMS,ON OUVRE UN FICHIER A$(255)
OU A$(10) ET ON SAUVE EN AYANT EU SOIN DE NE SAUVER QUE LES
MEMOIRES OCCUPEES.SI ON A A$(255) AVEC SEULEMENT 20 NOMS ON
PERD (255-20) = 235 BYTES OU 2 SECTEURS QUI SONT POURTANT VIDES.
(FAUTE VUE DANS LE WORD PROCESSOR)
POUR EVITER CELA,ON PEUT SOIT CREER UN FICHIER DE LONGUEUR A$(19)
ET TRANSFERER LES DONNEES DANS CE FICHIER SOIT CHANGER LES
CARACTERISTIQUES DU FICHIER PAR DES POKES SUR VARPTR(A$(0))-3 -2 ET -1
ET LES REMETTRE EN PLACE APRES L'AVOIR SAUVE.
SI UN FICHIER DOIT ETRE LU,CORRIGE PUIS ETRE REPLACE
SUR LA MEME DISQUETTE, IL EST SAGE DE CALCULER AU MOMENT
DE LA LECTURE LE NOMBRE DE SECTEURS QU'IL PREND SUR LA DISQUETTE
POUR CELA ON FAIT :

```
30000 LONG=2:FOR I=0 TO 255
30010 IF A$(I)="" THEN 30040
30020 LONG=LONG+1+LEN(A$(I))
30030 NEXT I
30040 SECTOR=(LONG+1)/128
```

AVANT DE SAUVER,ON REFAIT LA MEME CHOSE AVEC LE FICHIER
A SAUVER. SI LONG = UN MULTIPLE EXACT DE 128,ON DOIT RAJOUTER
UN NOM DE PASSE SOUS PEINE DE PERDRE LE FICHIER.PUIS ON CALCULE
LE NOMBRE DE SECTEUR DU NOUVEAU FICHIER A SAUVER.SI CE NOMBRE
EST <= A L'ANCIEN,PAS DE PROBLEME.PAR CONTRE,S'IL
EST SUPERIEUR ON "RENAME" L'ANCIEN PUIS "DELETE" ET ENFIN
ON SAUVE LE NOUVEAU SUR DISQUETTE.CE FICHIER SE METTRA A LA
FIN DE LA DISQUETTE.POUR RECUPERER LA PLACE PERDUE ON EXECUTERA UN
COMPACT A LA FIN DU TRAVAIL.
EN CONCLUSION,LE DISK A DES AVANTAGES ET DES
INCONVENIENTS PAR RAPPORT AUX AUTRES DOS DES MARQUES CONCURRENTES.
AVANTAGES
---
ON N'A PAS BESOIN DE DONNER UNE LONGUEUR FIXE PAR DONNEE,
CE QUI FAIT QU'ON NE PERD PAS DE PLACE SUR LA DISQUETTE.
CECI EST UN GROS AVANTAGE QUI DOIT AU MOINS FAIRE GAGNER
UNE QUARANTAINE DE K PAR DISQUETTE (JE DIS BIEN 40000 BYTES
SUR LES 78000 QUE PEUT CONTENIR UNE DISQUETTE !).
INCONVENIENTS
---
LE DIRECTORY EST LIMITE A 107 NOMS.
LE SAVEA/LOADA TRANSFERE LE FICHIER DANS LE BUFFER APRES LE
PROGRAMME AVANT DE LE SAUVER/LIRE,CE QUI ENTRAINE UNE PERTE
DE PLACE ET DE TEMPS.CECI ETAIT INDISPENSABLE POUR LA CASSETTE,
CAR LE FLOT DE DONNEES DOIT ETRE REGULIER,MAIS ICI LE PROCESSEUR
DU DISK EST ESCLAVE ET PEUT ETRE LEGEREMENT FREINE.
MAIS AU TOTAL L'INSTRUCTION SERAIT PLUS RAPIDE.
POUR L'INSTANT UN FICHIER DE 6 SECTEURS NECESSITE 10 SECONDES,
CE QUI EST BEAUCOUP TROP LENT (J'AI CREE UNE ROUTINE EN ASSEMBLEUR
QUI NE NECESSITE QUE 4" ET IL Y A MOYEN DE FAIRE BEAUCOUP MIEUX).
LA DISKETTE,QUAND ELLE TRAVAILLE,BLOQUE L'UNITE CENTRALE
CONTRAIREMENT A CE QUE RENSEIGNE DAI DANS SA PUBLICITE.
LA DISQUETTE POURRAIT EFFECTIVEMENT LE FAIRE MAIS LE DOS N'A PAS
ETE PREVU POUR CELA.

CHER MR HERMANS

JE VOUS ENVOIE ENCORE QUELQUES TRUCS CONCERNANT PARTICULIEREMENT LES DISK I/O.

LE DSAVE OU LE SAUVETAGE DES PROGRAMMES BINAIRES EST LIBELLE COMME SUIT:EXEMPLE DSAVE ESSAI.BIN:0 2000 2EFF 2345
2000 DEBUT DU PROGRAMME.
2EFF FIN DU PROGRAMME.
2345 POINT D ENTREE DE LA ROUTINE D INITIALISATION.
IL EST PREFERABLE DE TOUJOURS INDIQUER FILENAME.BAS:1, CE QU INDIQUE QUEL FICHIER VOUS SAUVEZ (BAS,BIN,STR,FPT,INT) ET DE PREFERENCE LE NUMERO DU DISK CAR IL PEUT Y AVOIR DES PROBLEMES.
SI L ON SAUVE UN FICHIER BINAIRE SOUS LE NOM $USER SUR UNE DISQUETTE CONTENANT UN DOS, A L' INITIALISATION, LE COMPUTER LIRA LE $USER.BIN ET EXECUTERA LA ROUTINE D INITIALISATION:DANS CETTE ROUTINE ON CHANGERA LE 29B-29C.LE CHANGEMENT DES AUTRES POINTEURS SE FERA PAR LA ROUTINE D INITIALISATION DU DOS ET NE DOIT DONC PAS ETRE CHANGEE ICI.ON VEILLERA A TOUJOURS AVOIR LE BOOTSTRAP ET LE MSTRDOS AVEC LE MEME NOMBRE DE SECTEURS QUE SUR LA DISQUETTE NORMALE.(PAS DE SECTEUR VIDE)
EN METTANT UN $USER.BIN ET UN $USER.BAS EN MEME TANT QUE LE DOS ON PEUT A L INITIALISATION CHARGER UN BINAIRE, METTRE LES POINTEURS EN PLACE SANS MANIPULATION, ENSUITE L'ORDINATEUR IRA CHERCHER LE BASIC ET LE LANCERA.CELUI-CI A SON TOUR POURRA ALLER CHERCHER DES FICHIERS FPT,INT OU STR.

LE NOM D'UN FICHIER NE DOIT PAS NECESSAIREMENT AVOIR 8 BYTES DE LONGUEURS . ON PEUT L'APPELER PAR EXEMPLE:LOAD "TAR.BAS:1" QUI VEUT DIRE PROGRAMME BASIC "TAR" SUR LE DISK 1.

L APPEL CALLM #300 NE FNCTIONNE QU'UNE FOIS.SI ON REFAIT UN AUTRE APPEL,IL Y A UN PLANTAGE MONUMENTAL.

IL EST TRES FACILE D'AJOUTER DES INSTRUCTIONS A DOS:
EXEMPLE:
CHANGER LE POINTEUR 0297-0298 PAR 00-20 (#2000) PUIS ENTRER LE PETIT PROGRAMME SUIVANT
2000 03 41 41 41 77 04 00 37 18
PUIS REVENER EN BASIC ET TAPER AU CLAVIER AAA PUIS RETURN
LE RESETD S'EXECUTE.
ON PEUT AUSSI L'APPELER PAR PROGRAMME:
10 POKE #131,3:PRINT "AAA":POKE #131,1
EXPLICATION: LA TABLE DU DOS EST DEVIEE VERS 2000
3=LA LONGUEUR DU NOM DE L'INSTRUCTION.
41 41 41 CODE # DE A
77 04 =0477 ADRESSE DE L'INSTRUCTION RESETD
00 ARRET DE CETTE TABLE
37 18 ADRESSE DE LA TABLE DU DOS(#1837)

LE BACKUP ET LE COMPACT EXECUTENT LE IDISK AVANT D EFFECTUER LE RECOPIAGE.

LE COPY D'UN FICHIER OU D'UN PROGRAMME AJOUTE UN SECTEUR VIDE A LA FIN.POUR L EVITER ON FAIT UN CREATE AVANT LE COPY
EXEMPLE
FILENAME.BAS:0 A RECOPIER SUR DISK 1.IL A 50 SECTEURS
SI ON FAIT COPY FILENAME.BAS:0 ON AURA SUR LE DISK 1 51 SECTEURS POUR FILENAME.BAS
PAR CONTRE SI ON FAIT:
CREATE FILENAME.BAS:1 50
COPY FILENAME.BAS:0
ON N'AURA QUE 50 SECTEURS SUR LE DISK 1

LE BYTE #9B7, QUI EST LE BYTE D'ERREUR POUR LES INSTRUCTIONS DU DOS, PEUT SERVIR D UNE SORTE DE IF ERROR.

EXEMPLE:

```
5 DIM A$(255)
10 POKE #9B7,0:POKE #131,3:PRINT "VERIFY FILENAME.STR:0":POKE#131,1
15 IF PEEK(#9B7)=0 THEN 50
20 PRINT "LE FICHIER FILENAME NE SE TROUVE PAS SUR LE DISK,VEUILLEZ CHANGER LE D
ISK"
30 RJZ=GETC:IF RJZ=0 THEN 30:GOTO 10
50 LOADA A$ "FILENAME.STR:0"
```

STOCKAGE DES DISQUETTES: ON VEILLERA A TOUJOURS COLLER SUR LE COTE LA PETITE ETIQUETTE POUR BLOQUER L'ECRITURE ;UN IDISK EST SI VITE ARRIVE.

L'INSTRUCTION RESETD SERT A REINITIALISER LE PROCESSEUR DE LA DISQUETTE :IL SERA UTILISE QUAND ON UTILISE LES FICHIERS ASCII ,QU' ON A LA DONNEE QUE L ON VEUT ET QUE L ON VEUT ARRETER LA LECTURE.

SI ON FAIT UN BREAK DANS UNE INSTRUCTION DU DOS OU DANS UN LOADA,SAVEA,LOAD,SAVE ,IL VAUT MIEUX,DES QU'ON A LE CONTROLE, EXECUTER UN RESETD SINON LA PROCHAINE INSTRUCTION DU DISK RISQUE D'ETRE FAUTIVE (SURTOUT POUR LE DIR QUI N'EFFECTUE PAS UN RESETD).

SI VOUS AVEZ UN FICHIER QUE VOUS MODIFIE REGULIEREMENT (FICHIER CORRESPONDANCE,STOCK,COMMANDE),VOUS DEVEZ TROUVER UNE TECHNIQUE POUR NE PAS RISQUER DE LE PERDRE.

LE SAUVETAGE TRIANGULAIRE

---

SOIT DISK "A" LE FICHIER FAIT LE 1 NOVEMBRE
VOUS FAITES UN BACKUP SUR LE DISK "B"
VOUS MODIFIER LE DISK "A" LE 5 NOVEMBRE
VOUS FAITES UN BACKUP DE "A" SUR "C"
VOUS AVEZ DONC
"B" DISK 1 NOVEMBRE
"A"-"C" DISK 5 NOVEMBRE
VOUS MODIFIEZ LE DISK "A" LE 10 NOVEMBRE
VOUS FAITES UN BACKUP DE "A" VERS "D"
VOUS AVEZ DONC
"B" DISK 1 NOVEMBRE
"C" DISK 5 NOVEMBRE
"A"-"D" DISK 10 NOVEMBRE
QUANT VOUS ETES CERTAIN QU'IL N Y A PAS EU D'ERREUR SUR "A"-"D" VOUS DETRUISE LE DISK "B".ON DOIT EN FAIT VEILLEZ A TOUJOURS AVOIR LE FICHIER, SA COPIE ET SA VERSION PRECEDENTE.
CECI EST UN MINIMUM.

POUR CEUX QUI FONT DES MODIFICATIONS QUOTIDIENNEMEN, IL VAUT MIEUX ADOPTER UN CYCLE SUR 7 JOURS. VOUS DETRUISEZ LE FICHIER DU JOUR DE LA SEMAINE PRECEDENTE ET GARDEZ DES LORS A TOUT MOMENT 6 VERSIONS.
LE LUNDI, ON DETRUIT LA VERSION DU LUNDI PRECEDENT
D'AUTRE PART, UNE COPIE NE SE STOCKE PAS DANS LA MEME BOITE QUE L ORIGINAL ET DE PREFERENCE PAS DANS LA MEME PIECE.CECI POUR EVITER DES ACCIDENTS (VOLS,CHAMP MAGNETIQUE ACCIDENTEL,ETC..)

CONCERNANT L'IMPRIMANTE (EPSON MX80 F/T),IL FAUT EVITER DE LUI ENVOYER TROP SOUVENT DES CARACTERES DE CONTROLE, CAR ELLE NE FAIT DES LORS PLUS QUE L'ECRITURE DE GAUCHE A DROITE ET EST DE CE FAIT LORS PLUS LENTE.

EXEMPLE:

```
10 FOR I=0 TO 200
20 PRINT CHR$(15);A$(I)
30 NEXT I
```

LE CHR$(15) EST RAPPELE A CHAQUE LIGNE.IL EST NON SEULEMENT INUTILE MAIS L'IMPRIMANTE NE TRAVAILLERA QUE DE GAUCHE A DROITE
IL VAUT MIEUX FAIRE

```
10 PRINT CHR$(15);:FOR I=0 TO 200
20 PRINT A$(I)
30 NEXT I
```

CONCERNANT LE PERSONAL COMPUTER,PLUSIEURS REMARQUES:
L INSTRUCTION FRE EST LE NOMBRE DE BYTES ENTRE LA FIN DE
LA TABLE DES SYMBOLES ET LE DEBUT DE LA ZONE D'ECRAN
ELLE FAIT LA DIFFERENCE ENTRE LA VALEUR DE (A5-A6) ET (A3-A4)
ELLE NE REPRESENTE EN RIEN LE RESTANT DE MEMOIRE.
SACHANT QUE L'INSTRUCTION SAVEA LOADA UTILISE CE BUFFER
POUR SAUVER SES FICHIERS, ON DOIT AVOIR LE FRE QUI DOIT
ETRE EGAL OU SUPERIEUR A LA DIMENSION DU FICHIER.
POUR LES FICHIERS STRINGS, ON CALCULE LA DIMENSION
DE LA FACON SUIVANTE:EXEMPLE: DIM A$(20,10)
1 BYTE POUR LE NOMBRE DE DIMENSION
2 BYTE POUR LES CARACTERISTIQUES (20,10)
(20+1)*(10+1)*2 POUR LE TABLEAU DE POINTEURS
1 BYTE DE LONGUEUR DE STRING+ LA VALEUR PAR MEMOIRE OUVERTE
SI CE NOMBRE EST PLUS GRAND QUE FRE, IL APPARAIT UN LOADING ERROR 1
ON PEUT A LA RIGUEUR,SI L ON N A PAS BESOIN DE L'ECRAN
A CE MOMENT CHANGER A3-A4 PAR #BFFF,PUIS LOADA ET REMETTRE
A3-A4 EN PLACE APRES

```
5 DIM A$(255)
10 AZ=PEEK(#A3):BZ=PEEK(#A4)
20 POKE #A3,#FF:POKE #A4,#BF
30 LOADA A$ "FICHIER"
40 POKE #A3,AZ:POKE #A4,BZ
50 MODE0:PRINT CHR$(12)
```

L'ECRAN PENDANT LE LOADA SE REMPLIT DE DONNEES CODEES,IL
FAUT L EFFACER PAR LA LIGNE 50.
AVEC CE SYSTEME, ON PEUT AVOIR UNE HEAP PLUS GRANDE MAIS
IL NE FAUT PAS AVOIR BESOIN DE L'ECRAN A CE MOMENT LA.

CONCERNANT L'ORGANISATION DES PROGRAMMES BASIC,
JE RESERVE DES ADRESSES POUR CERTAINES FONCTIONS,MEME
SI JE N EN AI PAS BESOIN
1-99 UNIQUEMENT L'INITIALISATION
100-199 CONTROLEUR GENERAL DOUT TOUT PART ET REVIENT
200-999 ZONES DE TRAVAIL OU LES TACHES SONT SEPAREES TOUS LES 100
TACHE 1 DE 200 A 299
TACHE 2 DE 300 A 399 ETC
1000 EDITEUR DE VALEURS
30000 LOADA DE FICHIERS
31000 SAVEA DE FICHIERS
32000 LECTURE DU DIRECTORY DU DISK
10000 ROUTINE D IMPRESSION SUR L'IMPRIMANTE
CECI PERMET DE CONSTRUIRE DE NOUVEAUX PROGRAMMES SANS
RELOCATER CERTAINES ZONES ET D'UTILISER LE MERGE POUR
FAIRE UN PROGRAMME. C'EST LE PRINCIPE DU MODULE.
JE RESERVE EGALEMENT CERTAINS NOMS POUR DES PROBLEMES
PARTICULIERS.
RJZ TOUJOURS POUR LE GETC
Z,ZZ,Z%,ZZ%,Z$,ZZ$ TOUJOURS POUR DES CALCULS INTERMEDIAIRES
QUI VONT ETRE UTILISES TRES RAPIDEMENT DANS
LE PROGRAMME OU DES DONNEES TRANSMISES A UNE
SOUS- ROUTINE
EXEMPLE Z=3:GOSUB 2000:PRINT A$(Z) LE Z N'ETANT PLUS
UTILISE PAR APRES.CECI PERMET DE LIMITER LE NOMBRE DE
VARIABLES

I,J,K,I%,J%,K% UNIQUEMENT POUR LES BOUCLES FOR NEXT
DI$( ) UNIQUEMENT POUR LE COPIAGE DU DIRECTORY($DKDIR)
AVEC DI$(0) OU DI$(0,0) ET DI$(0,1) LE NOMBRE DE FICHIERS OUVERTS
A$(),B$(),C$() A() AZ() ETC POUR LES FICHIERS
AVEC CETTE ORGANISATION, LA LISIBILITE DES
PROGRAMMES EST PLUS GRANDE.
JE VOUDRAIT SAVOIR SI VOUS AVEZ UN ASSEMBLEUR
POUR UTILISER DISK,ET CONTENANT EGALEMENT DES MACROS

# didaisoft

Nieuwe onderwijssoftware ... met het diDAIsoft-kwaliteitslabel.

## 1. diDAIsoft-talentape

### a. Test your tenses level 1

5 lijsten van telkens 10 onregelmatige engelse werkwoorden worden ter studie aangeboden. Per lijst van 10 worden de hoofdtijden opgevraagd. Foutieve antwoorden worden onthouden , verbeterd en opnieuw aangeboden. Basiskennis.

### b. Test your tenses level 2

6 lijsten van telkens 10 onregelmatige engelse werkwoorden worden op dezelfde wijze als in level 1 aangeboden en verwerkt. Uitbreiding van de basiskennis.

### c. Test your tenses level 3

De 110 werkwoorden uit level 1 en level 2 worden at random opgevraagd.

### d. Test your structures

De leerlingen moeten een correcte engelse zin vormen met een gegeven verb, subject, object en time adjunct. De aard van de gewenste zin (question, statement...) wordt eveneens opgegeven. Na twee pogingen wordt het correcte antwoord aangeboden.

### e. Test your words: animals

De computer presenteert woorden en uitdrukkingen in verband met:
1. animal sounds
2. young animals
3. male ... and... female
4. groups of animals
5. animals we eat

Na een studietijd worden deze uitdrukkingen ondervraagd. Correctie van foute antwoorden.

### f. Test your words: quantities and qualities

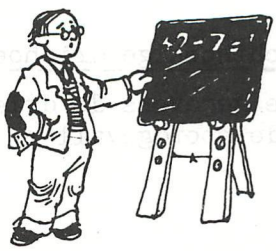
Analoog als bij animals. De woorden en uitdrukkingen hebben betrekking op quantities and qualities: synoniemen , tegengestelde begrippen....

### g. synoniemen en antoniemen

De computer presenteert een woord tesamen met een ganse reeks andere. De leerling moet het synoniem of het antoniem uit de lijst aanstippen. Vlot te gebruiken. Eenvoudig aan te passen naar andere toepassingen.

### h. Nederlandse items

Gebruiken jou leerlingen ook zo vaak foutieve uitdrukkingen zoals "wedden voor 20 frank" in plaats van "wedden om 20 frank" ? Laat ze dan dit diDAIsoft-programma uittesten. Dit programma hamert 40 analoge uitdrukkingen op een correcte wijze vast in hun taalgebruik.



didaisoft

## 2. Het diDAIsoft-familiebudget

Voortaan geen problemen meer met uw familiebudget. De DAI-personal computer houdt al uw inkomsten en uitgaven (zeer sterk gerubriceerd) bij, maakt alle gewenste totalen en overzichten. Het geeft u een duidelijk beeld (grafisch) van de diversiviteit van uw jaarlijkse uitgaven en berekent vanaf een gewenste maand het cumulatieve verschil van uw uitgaven en inkomsten, zodat u zelf uw eigen budget kritisch kan bekijken (grafisch) en beslissingen treffen in verband met uw maandelijkse spaaractiviteiten. Alle bedragen worden opgeslagen in arrays die op informatiedrager worden opgeslagen.
In het onderwijs als toepassing mogelijk in de lessen economie, handel... en in de lessen informatica waar bepaalde routines die in het programma uitgewerkt zijn kunnen bestudeerd worden.

## 3. Grafische hulp

Dit diDAIsoft-programma laat toe op een eenvoudige wijze tekeningen te maken en deze tekeningen tesamen met de tekst die erbij werd geschreven in files weg te schrijven en snel op te roepen. Op deze wijze kan men gemakkelijk de schermlayout van educatieve programma's verzorgen.
Geleverd met een uitgebreide handleiding (20 blz.)

## 4. diDAIsoft-wiskunde tape

### a. Studie van de neperiaanse logaritmische functie

Zeer didactische grafische presentatie van de definitie en het verloop van de neperiaanse logaritmische functie.

### b. Stelling van Pythagoras

Zeer duidelijke grafische illustratie bij het bewijs van de stelling van Pythagoras.

### c. Methode van Gauss : oplossen van nxn stelsels

Dit programma lost nxn stelsels op met de methode van Gauss en laat toe voor n kleiner dan 6 deze methode stap per stap te volgen.

### d. Grootste gemene deler van 2 natuurlijke getallen

Dit programma toont duidelijk het algoritme van de opeenvolgende delingen om de GGD te berekenen. Het laat bovendien toe dit algoritme zelf ook toe te passen.

### e. Priemgetallen : Zeef van Eratosthenes

Dit programma bestaat uit 3 delen:
1. demo: het zeefalgoritme wordt toegelicht voor de priemgetallen kleiner dan 200
2. m.b.v. deze methode kan de leerling de priemgetallen kleiner dan 5000 bepalen deze worden genummerd, zodat kan gevraagd worden naar priemgetal nummer n.
3. onderzoeken of een getal kleiner dan 25000000 een priemgetal is en zo nee een priemdeler laten afdrukken

### f. Oplossen van willekeurige en rechthoekige driehoeken ( 2 programma's)

Dit programma laat toe met concrete gegevens de driehoek volledig op te lossen. Het behandelt meer dan de vertrouwde hoofdgevallen.

# didaisoft

5 diDAIsoft-fysica_tape

a. Veldlijnen A1

Dit programma laat toe de veldlijnen uit te tekenen veroorzaakt door twee ladingen. Doordat het programma aan de gebruiker toelaat waarden voor parameters in te voeren is het programma voldoende flexibel om alle aspecten van deze theorie toe te lichten. Een uitgebreide handleiding wordt bijgeleverd.

b. Tijd-weg

Het doel van dit programma is:
1. meten van tijden tussen N opeenvolgende sluitingen van de event ingang op paddle 1
2. in de onderstelling dat dit sluiten gebeurt door de beweging van een voorwerp (wagentje) voorbij kontakten op onderling regelmatige afstanden A langs een baan, uitzetten van de beweging in weg-tijd-grafiek
3. uitzetten in grafiek van gemiddelde snelheid en gemiddelde versnelling in elk interval. Met handleiding, o.a. t.b.v. het practisch uitvoeren van de kontakten.

c. Stopwatch (kort)

Dit programma laat toe de DAI p.c. als chronometer te gebruiken en een aflezing te doen met grote cijfers. Paddle noodzakelijk.

d. Stopwatch (lang)

Zoals stopwatch kort maar met mogelijkheid om het tijdsverloop te volgen.

e. Hydraulische pers

Principe en werking van de hydraulische pers wordt grafisch toegelicht.

f. U-vormige buis

Enkele concrete voorbeelden (water - kwik en water-olie) worden gebruikt om op een zeer mooie grafische wijze de wetmatigheden die gelden bij een U-vormige buis te illustreren.

Wat is diDAIsoft ?

diDAIsoft is een softwaregroep die werkt binnen DAInamic en bestaat uit leerkrachten uit alle niveaus van het Belgisch onderwijs (lager,secundair,hoger). De groep heeft als doelstelling voor alle vakken van het (hoofdzakelijk secundair) onderwijs didactische software te ontwikkelen. Op dit ogenblik bestaat de groep uit 27 leden, die tweemaandelijks samenkomen in het CMO-vormingscentrum te Haasrode Geldenaaksebaan 327, telkens op zaterdag van 10u tot 16u. Volgende bijeenkomst 26-2-1983. Bent U leraar en heeft U interesse, kom eens kijken. Vooraf toch graag een telefoontje aan Bruno Van Rompaey (016/461085). Waar de groep geïnteresseerde collega's te woord zal staan is op zaterdag 9 april 1983 op onze DAInamic dag in Westerlo-Tongerlo. Kom daar zeker eens kennis maken. We hebben ook uw bijdrage nodig om van de diDAIsoft-programma's kwaliteitssoftware te maken. We verwachten U, collega.

| program identification | |
|---|---|
| title | : "KEN-DOS", a new floppy-standard for DAIpc |
| author | : Kenneth Gooswit |
| purpose | : |
| comment | : We will bring test results in the next edition |

Dear User,

About a year ago I was annoyed by the fact that there was no fast memory-unit available for the DAI-PC. Then Dai came with a twin-drive floppyunit, but in my opinion this unit doesn't meet all the needs of the common user. In fact, this unit is very slow.
To overcome this problem I then started to do some research.
After several months of hard working and talking to the right people, I managed to construct a controller-board that made it possible to operate four 80-track floppy drives.
At that moment only in single-density.
The problem was that I couldn't adapt any software to my system and I was forced to write mine own operating system.
Now most of the work is done, and I can proudly announce the birth of 'Ken-Dos'. I have contacted a few firms and it is now possible to supply a complete unit with fast accestime and a lot of storage capabilities. Up to 3,2 megbyte. A back-up is done within one minute and fifteen seconds (400kbyte)

## THE HARDWARE

The system consists of a controller-PCB, powersupply with ringcore transformer,enclosure for two drives, one or two drives and an eprom-PC-Board with 14 banks of 2kbyte that has to be connected to the X-bus; thus 28kbyte extra memory.
The software resides totally in eprom (12kbyte) and there is an option of using 4kbyte of static rams (6116), which is then used as an buffer for the directory and the sectormap. Normally 2560 bytes of the DAI-ramspace from #A950 to #B350 is used for this purpose. This buffer moves automatically with the screenbuffer, so saving a picture won't be a problem. There are programs however, like the new assembler/editor, which use those addresses and in that case the user has the option of using the extra 4kbyte buffer. To do so however, the stack-interrupt circuit has to be disabled by putting the base of transistor T1 to ground.
The user has 16kbyte extra romspace at his disposal, so it is possible to put several programs in eprom.
Since Ken-Dos is totally in eprom, any program ever written for the DAI-PC can be loaded and run without relocation.
The controller PC-Board can handle four mini 80-trackdrives (double sided/ double density). That means that the user can put 3,2megbyte on line.
To use the double density mode, the user has to make a minor hardware modification. Pin 49 of the X-bus has to be connected to pin 4 of the DCE-bus.
The powersupply can service two drives and is automatically switched on and off by the DAI-PC.
The controller PC-Board, powersupply and drives are horizontally mounted in a cabinet. On the rear is an extra connector for a printer or DCR.

## THE SOFTWARE

The software can handle sequential- and randomacces-files.
In single density the user can store 200kbytes formatted and in double density 400kbytes on a single side of a diskette (DS/DD).
Since Ken-Dos is entirely in eprom, there is no need for a systemdisk so all memoryspace on the diskette is at the users disposal.
After coldstart or reset Ken-Dos will be in search of com-files on the disk resided in drive-0. The names of those files are placed in the extended command-table and are then handled as commands. This means that those files can be loaded and eventually executed by just entering the name of the file.

After this search another search takes place. Ken-Dos will look for a file with an unique mark telling that this file has to be loaded and eventually executed. If no such file is found it returns by printing name of the inserted disk,date, free sectors, status of protection and density on top of the screen. Ken-Dos will then ask for date and password. To keep the password secret, it is scrambled on the screen. If the user doesn't know the password, he will not have acces to protected disks or files and utilities.
With Ken-Dos it is possible to link a basicprogram to a ML-part.On loading the basicprogram the ML-program will automatically be loaded. First Ken-Dos loads the ML-part and then the basicprogram. Pointers are corrected.
When loading a file, the user could enter the full name of the file, but it is also possible to enter just a part of the name. Ken-Dos will find and load the desired file. If an autostartmark is placed before the name (+), the file will be executed.
Every file is saved with date of creation and date of the latest modification.
The disk has a name, a date of formatting, and can be software protected agains writing or reading. Every disk has a directory of the created files. In single density the user can create 64 files; in double density 128 files. After entering the command 'DIR1', Ken-Dos will display the directory of the disk in drive-1. The name, date etc. of the disk will be printed on top of the screen followed by the filenames. On hitting the spacebar the filenames will scroll on the screen. The diskname etc. will remain stable.
To display all files ,including deleted files, just type 'DIRi1'.

The following COMMANDS are available:

| | |
|---|---|
| 'DIRO" | : display directory on screen(drive-0 is selected) |
| 'LIB1' | : all commands are displayed on screen |
| 'LIB2' | : all extentioncommands (com-files) are displayed |
| 'HELP' | : information about using Ken-Dos is given on the screen |
| 'LOAD".."' | : normal Dai-load |
| 'SAVE".."' | : normal Dai-save |
| 'DLOAD".."' | : any file or a part can be loaded (ML-files etc.) |
| 'DSAVE".."' | : any file or a part can be saved |
| 'DISK1' | : assign system to disk (drive-1 is selected) |
| 'CAS' | : assign system to casset |
| 'RDCAS' | : read-only of casset |
| 'WRCAS' | : write-only |
| 'DCR' | : assign DCE-bus to DCR |
| 'LPRINT' | : assign DCE-bus to printer |
| 'PASS' | : to enter the password |
| 'DATE' | : to enter the date |
| 'BASIC' | : basic pointers are shown and can be modified |
| 'COM1' | : com-files are retrieved from disk (drive-1 is selected) |
| 'BANK5' | : to select an eprombank (bank-5 is selected) |

The following commands are only available to the user who knows the right password:

```
'RENAME".."': to change the filename
'DNAME1'    : to change the diskname(drive-1 is selected)
'LOCK".."'  : to protect file
'UNLOCK".."':
'DELETE".."': to delete a file
'RESTORE"."':
'KILL".."'  : to destroy a file
'COMPACT"."': to copy all files (except deleted files)
'COPY".."'  : copy a file
'BACKUP".."': total copy of the disk
'OPEN"."..": open a file for writing
'CLOSE".."' : close it
'VERIFY".."': verify a file and print informations about lenght,startaddress,
'FORMAT".."': format a disk                                          etc...
'MANUAL'    : To read or write any sector on the disk
```

It is possible to link most commands to basicprograms. A basic-line with 'CALLM#F000:REM ...(command)' will do the job.
There are more features, but explaining them all goes beyond the purpose of this article. For detailed information I refer to the USERS-MANUAL

The whole system complete with one or two double-sided drives (resp. 800kb or 1,6 megbyte of memoryspace) and the USERS-MANUAL can be ordered.
Just write to:

Ken-dos dev.
P.O.B. 40
1616 ZG  HOOGKARSPEL
HOLLAND
TEL: 02285/1 28 93

or:

MIKRO SHOP
Bennenbergweg 1
3221 NIEUWRODE (Bij AARSCHOT)
BELGIUM
TEL: 016/56 87 70

It will be possible to adapt the system to other minidrives or microdrives.
For further information write to us.
There are also entrypoints to use with modified CPM
A utility program to read standard DAI-disks will be available.
Ken-Dos is written in a way that it can be extended. All extentions will be available at minor costs

Kenneth Gooswit

**KEN-DOS**

# AUDIO CASSETTE INTERFACE

## PART 1: WRITING TO TAPE

### 1. FORMAT:

#### 1.1. DATA FORMAT:

Many personal and hobby computers use the so-called Kansas-City format for writing data to an audio cassette tape. This is a FSK (frequency shift keying) method, using 1200 Hz and 2400 Hz tones to write '0' and '1' data bits to the tape.
The DAI uses a different format. For each bit written to tape, it uses 2 impulses of different duration, a short one and a long one. The signal send to tape for a '0' and for a '1' bit are given in figure 1.

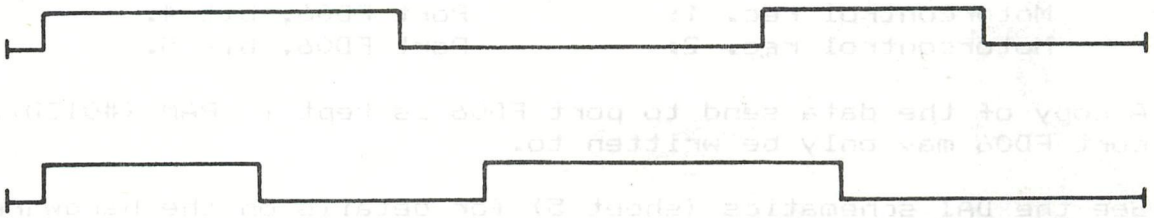


figure 1: Data '1' and '0' bits

#### 1.2. LEADER FORMAT:

At the beginning of a program (a 'file'), written on tape, a 'leader' tone is written in order to enable synchronisation when the file has to be read from the tape. The leader consists of a great number of bits, for which the duration of the 'long' and the 'short' impulse are equal (see figure 2). This results in a constant tone of about 890 Hz for ca. 2.3 secs.

figure 2: a leader bit

At the end of the constant tone, one '1' bit is written to tape to indicate the end of the leader tone.

#### 1.3. TRAILER FORMAT:

Figure 3 shows the format of a bit of the trailer tone, which is written after a file on tape to indicate its end.
Although the DAI writes a trailer tone on the tape, it does not use it when reading the file from tape.

figure 3: a trailer bit

A data '1' bit ends the trailer tone.

### 1.4. TAPE SPEED DATA:

The information for the impulse duration of the impulses to be written on tape is moved from ROM (D7C5-D7CA) into RAM 02E6-02EB during system reset. These constants are:

| | | |
|---|---|---|
| 02E6/7 | 24/24 | Impulse,duration leader bits |
| 02E8/9 | 24/3C | Impulse duration data bits |
| 02EA/B | 24/18 | Impulse duration trailer bits |

## 2. AUDIO CASSETTE RECORDER CONNECTIONS:

Two audio cassette recorders can be connected. Their motors can be switched on/off under program control. The audio input and output channels of both recorders are parallel connected.

| | |
|---|---|
| Output DAI to recorders: | Port FD06, bit 0. |
| Input recorders to DAI: | Port FD00, bit 7. |
| Motorcontrol rec. 1: | Port FD06, bit 4. |
| Motorcontrol rec. 2: | Port FD06, bit 5. |

A copy of the data send to port FD06 is kept in RAM (#013D), because port FD06 may only be written to.

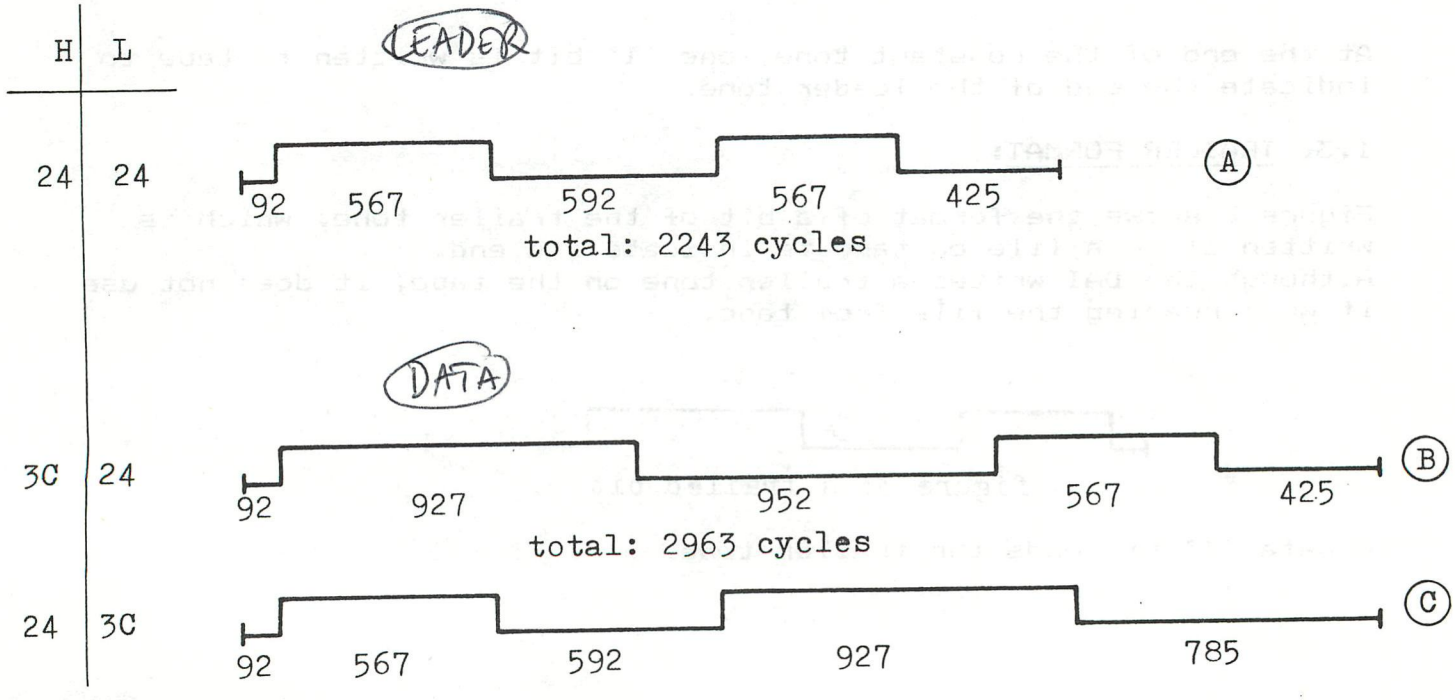
See the DAI schematics (sheet 5) for details on the hardware.
Details on the software can be found in the 'DAI pc FIRMWARE MANUAL'.

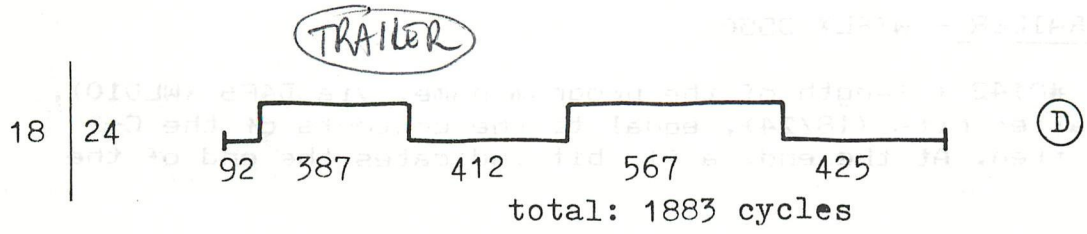
## 3. SUBROUTINES FOR WRITING TO TAPE:

### 3.1. WRITE A BIT - WBIT/D524:

On entry, the HL-registers contain the relevant impulse duration information from 02E6-02EB. One cycle (= 1 impulse = high/low) is written to port FD06/bit 0 for the duration of the value loaded into H. Then the value in L is moved into H, and the second cycle (with other impulse duration) is written to tape (see figure 1).

This routine is used for writing data bits, leader bits and trailer bits to tape.






Duration given in machine cycles (1 cycle = 0,5 usec)

Tape speed data:

24/24 - Used for the leader.

3C/24, 24/3C - Used for data bytes: '1', '0'

18/24 - Used for the trailer.

Figure 4: Bit patterns and timing

### 3.2. WRITE A BYTE - WBYTE/D509:

On entry, the byte to be written is in the accumulator. The HL registers contain the impulse duration values, and the DE registers are loaded with the inversed values.
To write a byte (= 8 bits), the highest bit is moved into the CY-flag. If the bit is '1', fig.4b is written to tape; if the bit is '0', fig.4c is written to tape. This sequence is done 8 times.

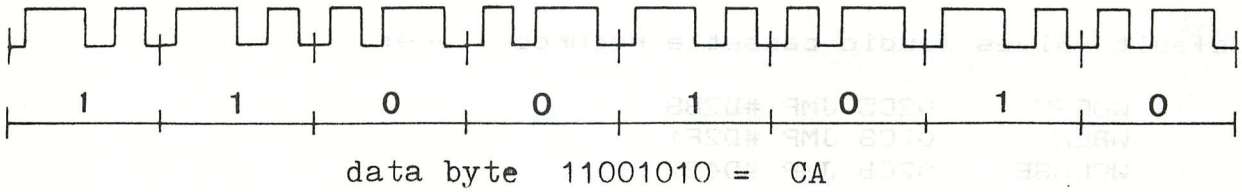



figure 5: A data byte pattern

### 3.3. WRITE A LEADER - WLEAD/D4ED:

The HL registers are loaded with the tape speed data 24/24. In BC, the number of leader bits to be written are loaded (#07E8 = 2024). Now 2024 times 24/24 impulses are written via WBIT. When all leader bits are written, HL is loaded with the tape speed data for a data bit, and a '1' is written, indicating the end of the leader.

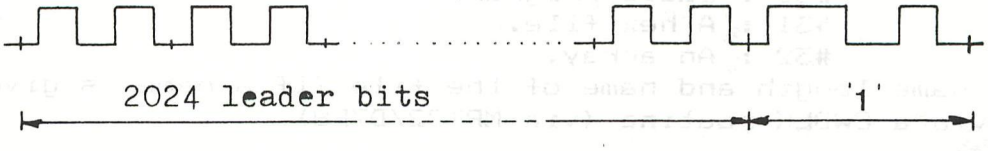



figure 6: Leader pattern

3.4. WRITE A TRAILER - WTRLX/D550:

On entry, BC = #0142 + length of the program name. Via D4F6 (WLD10), a number of trailer bits (18/24), equal to the contents of the C-register is written. At the end, a '1' bit indicates the end of the trailer.

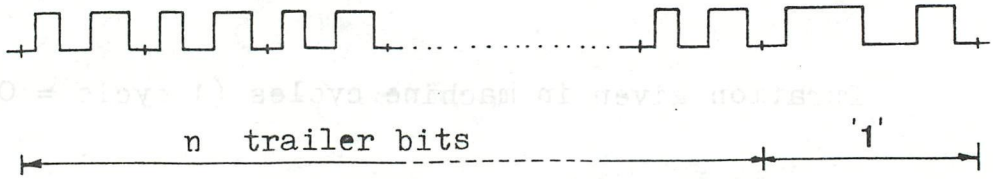



figure 7: Trailer bit pattern

## 4. FILE HANDLING:

Now all subroutines are known, the file set-up will be explained. (A program written to tape can be considered as a file). Normally, a file consists of one or more blocks of data, which are written to tape sequentially. At the beginning of the file on tape, a kind of a 'header' is written, containing the leader tone and some information on the type of file. The end of the file is marked with a trailer tone.

For audio cassette operation, subroutines are available in the resident software to open a tape file (CWOPEN), write data blocks to tape (CWBLK) and to close a tape file (CWCLOSE).
The startaddresses of these audio tape file handling routines are moved from ROM (D7A4-D7AC) into RAM 02C5-02CD during system reset. These RAM pointers WOPEN, WBLK, WCLOSE can be loaded with other - user defined - pointers in order to address write routines for other mass storage devices, like floppy disk or digital cassette recorder (like the Memocom MDCR).

The default values (audio cassette recorder) are:

```
WOPEN    02C5 JMP #D2B8
WBLK     02C8 JMP #D2F1
WCLOSE   02CB JMP #D427
```

### 4.1. OPEN A TAPE FILE - CWOPEN/D2B8:

- Evaluate if writing to tape during a program run or in direct mode (MPT21/D720).
- If in direct mode: Print 'SET RECORD,START TAPE,TYPE SPACE' and wait for the spacebar to be pressed.
- Start selected cassette motor(s) (CASST/D42E).
- Write leader tone (WLEAD/D4ED).
- Write a 'flag byte' #55 to tape. This is the first byte after the leader tone.
- Write a 'file type byte':
  - #30 : Basic program.
  - #31 : A hex file.
  - #32 : An array.
- Write name length and name of the file (if a name is given) to tape via a CWBLK routine (via MPT22/D7F8).

### 4.2. WRITE A DATA BLOCK TO TAPE - CWBLK/D2F1:

- Write the length of the block, followed by its checksum.
- Write the contents of the block, plus its checksum.

The checksum-byte is a control byte on all transmitted data bytes. It is intially set to #56, and then EXOR-ed with the data byte, followed by a RLC (rotate left). This is done again and again for all data bytes transmitted. The final checksum byte is written on tape after the last data byte.

### 4.3. CLOSE A TAPE FILE - CWCLOSE/D427:

- Write trailer bits (WTRLX/D550).
- Switches off cassette motors (CASSP/D445).

## 5. RESIDENT DATA FILING ROUTINES:

The DAI firmware knows 3 data filing routines:

| | | |
|---|---|---|
| 'SAVE' | file type 0 | Save Basic programs. |
| 'UT/write' | file type 1 | Save Hex files |
| 'SAVEA' | file type 2 | Save Arrays |

The use of the routines WOPEN, WBLK and WCLOSE is slightly different for each of the 3 methods.

### 5.1. BASICCOMMAND 'SAVE' - RSAVE/D23D:

- D23D: Clear all variables, calculate length of textbuffer and symboltable, evaluate a (evt) given name.
- D263: WOPEN (see 4.1).
- D268: WBLK: write textbuffer length + checksum.
  idem for textbuffer contents.
- D26C: WBLK: write symboltable length + checksum.
  idem for symboltable contents.
  WCLOSE (see 4.3).

### 5.2. UTILITY COMMAND 'W(rite)' - 3EEE4:

- EEE4: Get low- and highaddress on stack.
- EEED: Get an evt. name in the encoded input buffer.
- EEF8: WOPEN.
- EEFE: WBLK: write startaddress (length + checksum, contents + checksum).
- EF08: WBLK: write hex block (length + checksum, contents + checksum)
- EF0B: WCLOSE.

### 5.3. BASICCOMMAND 'SAVEA' - RSAVA/D81D:

- D81D: Test type of array.
- D824: If stringarray: Arrange stringelements in the free RAM space (see previous article on SAVEA/LOADA).
- D82F: WOPEN.
- D837: WBLK: write array type (length + checksum, byte + checksum).
  WBLK: write array contents (length + checksum, contents + checksum).
  WCLOSE.

| program identification | |
|---|---|
| title | 3-D pictures (with colored glasses) |
| author | J.Roelants |
| purpose | |
| comment | you need glasses with one red and one green filter. |

Als waarnemer geplaatst in de ruimte ontvangen we twee verschillende beelden via ons linker-en rechter oog . Deze twee beelden versmelten in onze hersenen tot één enkel beeld zodanig dat er een diepte-indruk ontstaat, m.a.w. we verkrijgen informatie omtrent de drie dimensionele ruimte die ons omringt .
Willen we een 3D-ruimte suggereren op een plat vlak dan kan dit in principe gebeuren door twee perspectievisch verschoven beelden te creëren . Het probleem stelt zich nu dat er door één oog slechts één beeld mag worden waargenomen . We rekenen er op dat onze hersenen deze twee beelden integreren tot één enkel 3D-beeld .
Een eerste bekende methode bestaat er uit een stereo-kijker te ontwerpen die er voor zorgt dat elk oog slechts één van de voorgestelde figuren waarneemt . Het voordeel van deze methode bestaat erin dat de voorgestelde figuren allerlei kleuren kunnen bevatten .

Een tweede methode bestaat erin de twee perspectievisch verschoven figuren over elkaar te plaatsen . Beide figuren bestaan uit één kleur , bijvoorbeeld rood en groen . Daar waar de gemeenschappelijke delen elkaar overlappen gebruiken we bijvoorbeeld geel .
Plaatsen we nu over het rechter oog een rode filter dan zal het rechter oog de groene figuur niet waarnemen .
Plaatsen we over het linker oog een groene filter dan wordt de rode figuur niet waargenomen .
Zijn de twee figuren op de juiste manier perspectievisch t.o.v. elkaar verschoven dan ontstaat een 3D-figuur .
Het voordeel van deze methode bestaat erin dat we gans het beeldscherm kunnen benutten . We dienen echter over een bril, voorzien van een rood en groen glas, te beschikken .

Er bestaan ondertussen technieken die toelaten drie dimensionele beelden op te wekken door het beeld te onderwerpen aan vibraties , hierbij kan het 3D-effect worden waargenomen zonder enig hulpmiddel .

Laten we de tweede methode nader toelichten .
We dienen te beschikken over 4 kleuren .
Namelijk zwart,rood,groen en geel .
Dit kan bij de DAI-gebeuren met COLORG 0 3 5 14 .
Hierin is kleur 3 (rood) voor het rechterbeeld .
kleur 5 (groen) voor het linkerbeeld
kleur 14 (geel) voor de gemeenschappellijke delen .
Elk beeldelement dient te beschikken over twee bits die we afzondelijk moeten kunnen beïnvloeden .
Het rechterbeeld onstaat door de eerste bit te setten , het linkerbeeld door de tweede bit te setten .
Bij de gemeenschappellijke delen is nu zowel de eerste als de tweede bit op 1 gebracht .

| | X1 | XO | COLORG | |
|---|---|---|---|---|
| 0 | 0 | 0 | achtergrond | 0 |
| 1 | 0 | 1 | rechterbeeld in rood | 3 |
| 2 | 1 | 0 | linkerbeeld in groen | 5 |
| 3 | 1 | 1 | gemeenschappellijk in geel | 14 |

Bij DAI onstaat het rechterbeeld door kleur 17 te gebruiken, hierbij wordt de eerste bit geset en behoudt de tweede zijn oorspronkelijke inhoud .
Het linkerbeeld onstaat door kleur 19 te gebruiken, waarbij de laatste bit wordt geset .

Het laatste probleem dat ons rest is de coördinaten te bepalen van het linker-en rechter beeldpunt XEL,YE en XER,YE . Om deze te kunnen vinden dienen de coördinaten van het weer te geven punt P te zijn gegeven (XP,YP,ZP) en de positie van de twee waarnemingspunten .
Het tafereel bevindt zich in het xy-vlak en de waarnemingspunten in het xz-vlak symmetrisch t.o.v. het yz vlak . De coördinaten van het rechterwaarnemingspunt zijn R(L,0,-V) , deze van het linkerwaarnemingspunt L(-L,0,-V) .

Het snijpunt van de rechte PR met het tafereel kunnen we vinden door de vergelijking van deze rechte op te stellen in de ruimte en de z-waarde gelijk aan nul te stellen .

$$\frac{x_{E_R} - x_R}{x_P - x_R} = \frac{y_{E_R} - y_R}{y_P - y_R} = \frac{z_E - z_R}{z_P - z_R}$$

$$\frac{x_{E_R} - L}{x_P - L} = \frac{y_E}{y_P} = \frac{+V}{z_P + V}$$

of $x_{ER} = \frac{V}{ZP+V} . (x_P - L) + L \qquad y_E = \frac{V}{ZP+V} \cdot y_P$

stellen we $\boxed{K = \frac{V}{ZP+V}}$ dan :

$$x_{ER} = (1-K).L + K.x_P \qquad y_E = K.y_P$$

stellen we $\boxed{D = (1-K).L}$ dan :

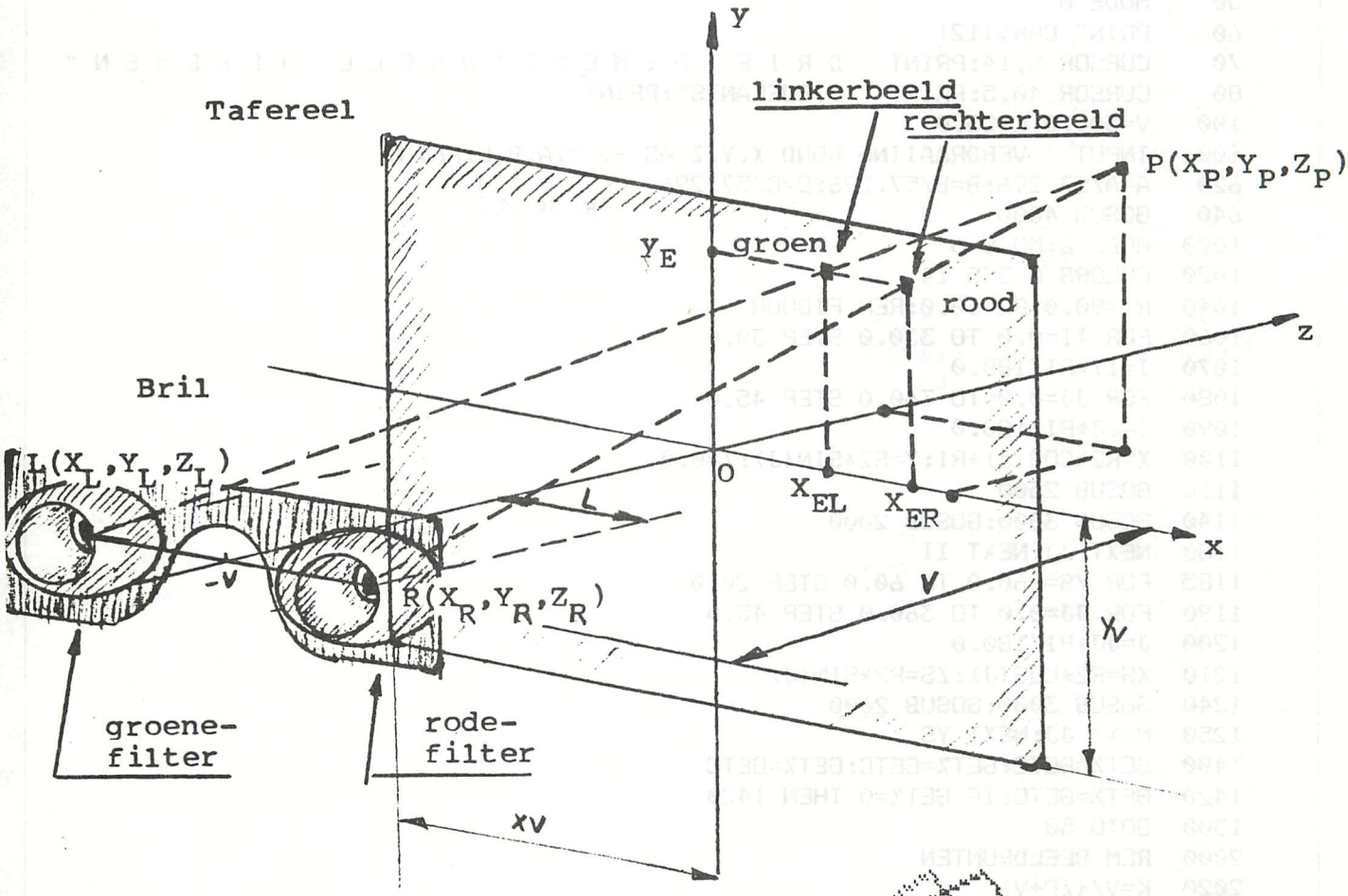
| $x_{ER} = K.x_P + D$ | $y_E = K.y_P$ |
|---|---|

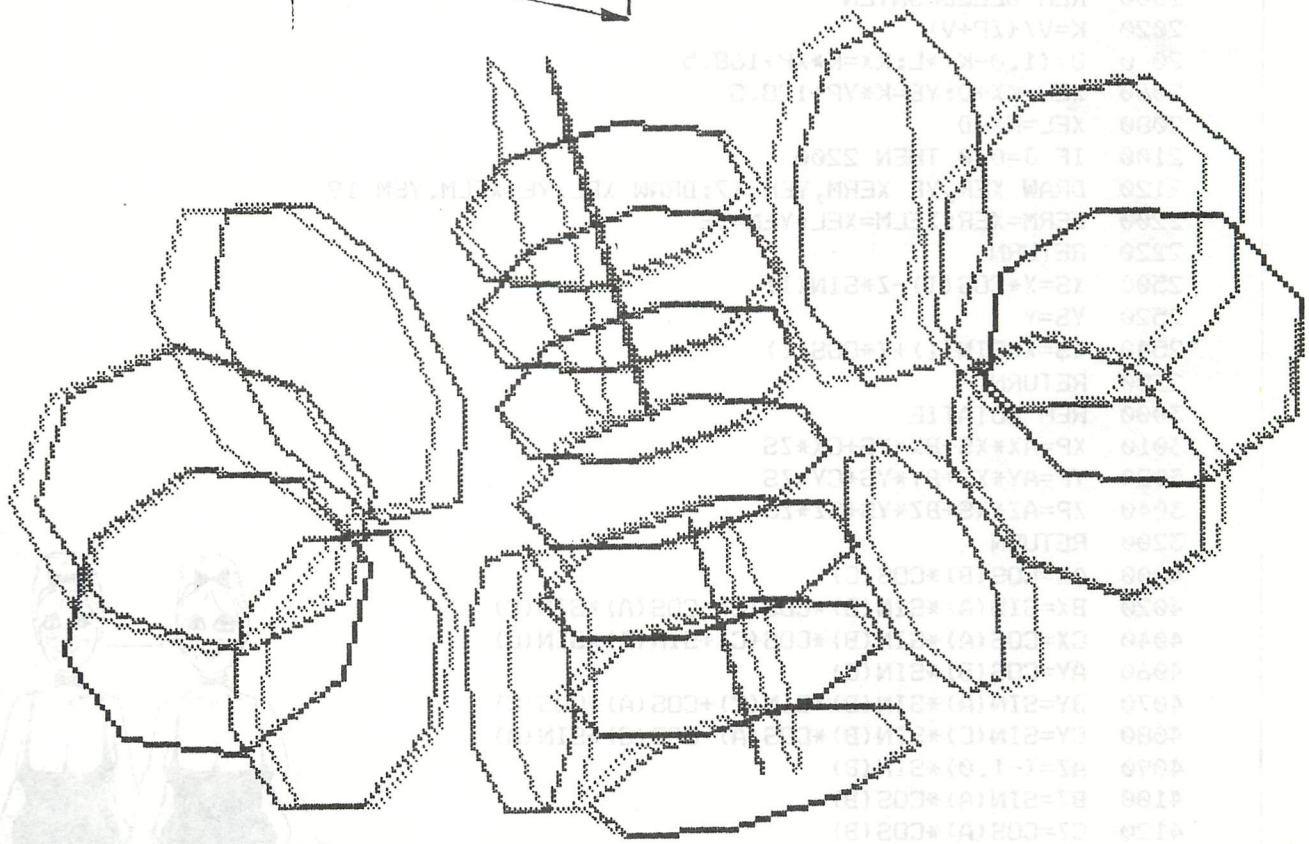
Voor het linkerbeeldpunt :

| $x_{EL} = K.x_P - D$ | $y_E = K.y_P$ |
|---|---|

Daar de oorsprong van ons assenkruis in het centrum van ons beeld is gelegen dienen we aan de gevonden waarden een translatie toe te voegen van XMAX/2 voor de x-waarden en YMAX/2 voor de y-waarden .

Nu dienen we de coördinaten van de gewenste figuur te genereren deze indien gewenst te onderwerpen aan de nodige translatie en rotatiebewegingen om uiteindelijk de waarden van XP,YP en ZP vast te leggen .
Daarna bepalen we de coördinaten van het linker-en het rechter beeldpunt zoals hierboven is beschreven . Passen eventueel een window toe indien bepaalde gedeelten van de figuur buiten het toegestane gebied vallen en wachten naar het resultaat dat we bekijken met onde bril .
De waarden van de afstand V tot het tafereel bedraagt 500 en deze van L = 25 indien XMAX=336 en YMAX=256 in MODE 6 .



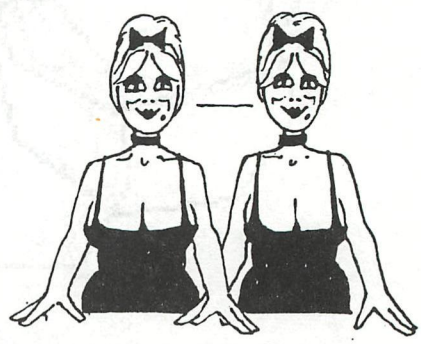
YELLOW

RED

GREEN

```
50    MODE 0
60    PRINT CHR$(12)
70    CURSOR 5,14:PRINT " D R I E   D I M E N S I O N E E L   T E K E N E N "
80    CURSOR 40,5:PRINT " J.ROELANTS":PRINT
100   V=500.0:L=25.0
600   INPUT " VERDRAAIING ROND X,Y,Z AS -> ";A,B,C:PRINT
620   A=A/57.296:B=B/57.296:C=C/57.296
640   GOSUB 4000
1000  MODE 6:MODE 6
1020  COLORG 0 3 5 14
1040  R1=90.0:R2=30.0:REM FIGUUR
1060  FOR II=0.0 TO 330.0 STEP 30.0
1070  I=II*PI/180.0
1080  FOR JJ=0.0 TO 360.0 STEP 45.0
1090  J=JJ*PI/180.0
1100  X=R2*COS(J)+R1:Y=R2*SIN(J):Z=0.0
1110  GOSUB 2500
1140  GOSUB 3000:GOSUB 2000
1180  NEXT JJ:NEXT II
1185  FOR YS=-60.0 TO 60.0 STEP 20.0
1190  FOR JJ=0.0 TO 360.0 STEP 45.0
1200  J=JJ*PI/180.0
1210  XS=R2*COS(J):ZS=R2*SIN(J)
1240  GOSUB 3000:GOSUB 2000
1250  NEXT JJ:NEXT YS
1400  GET%=GETC:GET%=GETC:GET%=GETC
1420  GET%=GETC:IF GET%=0 THEN 1420
1500  GOTO 50
2000  REM BEELDPUNTEN
2020  K=V/(ZP+V)
2040  D=(1.0-K)*L:KX=K*XP+168.5
2060  XER=KX+D:YE=K*YP+128.5
2080  XEL=KX-D
2100  IF J=0.0 THEN 2200
2120  DRAW XER,YE XERM,YEM 17:DRAW XEL,YE XELM,YEM 19
2200  XERM=XER:XELM=XEL:YEM=YE
2220  RETURN
2500  XS=X*COS(I)-Z*SIN(I)
2520  YS=Y
2540  ZS=X*SIN(I)+Z*COS(I)
2580  RETURN
3000  REM ROTATIE
3010  XP=AX*XS+BX*YS+CX*ZS
3020  YP=AY*XS+BY*YS+CY*ZS
3040  ZP=AZ*XS+BZ*YS+CZ*ZS
3200  RETURN
4000  AX=COS(B)*COS(C)
4020  BX=SIN(A)*SIN(B)*COS(C)-COS(A)*SIN(C)
4040  CX=COS(A)*SIN(B)*COS(C)+SIN(A)*SIN(C)
4060  AY=COS(B)*SIN(C)
4070  BY=SIN(A)*SIN(B)*SIN(C)+COS(A)*COS(C)
4080  CY=SIN(C)*SIN(B)*COS(A)-COS(C)*SIN(A)
4090  AZ=(-1.0)*SIN(B)
4100  BZ=SIN(A)*COS(B)
4120  CZ=COS(A)*COS(B)
4200  RETURN
```

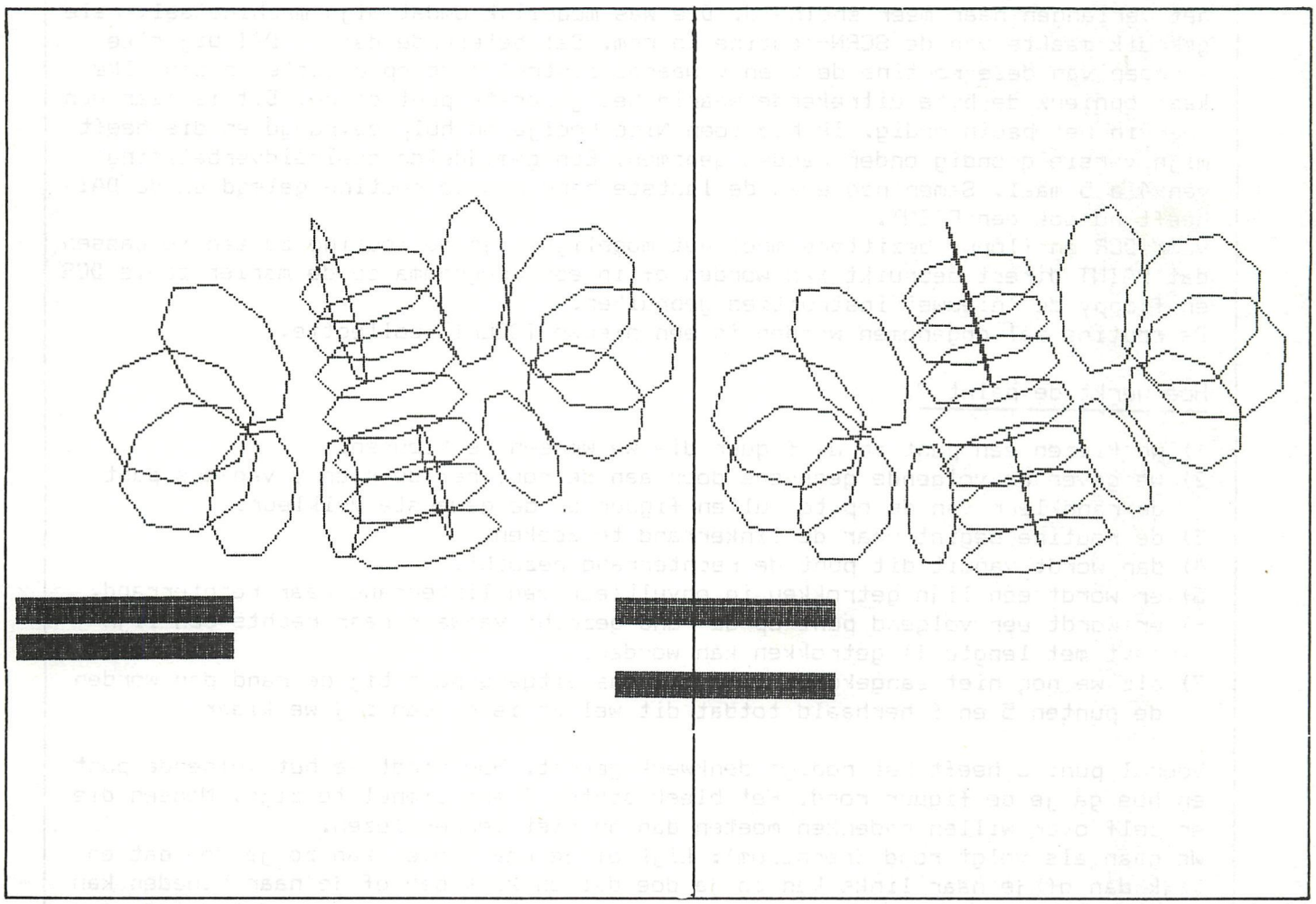
↑ ↑ ↑
e.g. 30,30,30



Bij het waarnemen van het beeld dient men de helderheid van het beeld zo hoog mogelijk te nemen zodanig dat de groene en rode kleur even sterk doorkomen .

Men kan het beeld bewaren door een foto te nemen van het scherm of door twee screen coppy's te nemen .
Bij de eerste screen coppy drukt men enkel de tweede en vierde kleur af , bij de tweede de derde en de vierde kleur .
Nu bekomt men twee figuren waargenomen door het linker - en het rechter oog . Bekijkt men deze twee beelden door een stereo-kijker dan onstaat het drie dimensionele beeld opnieuw .

Veel genoegen met het waarnemen van de drie-dimensionele beelden .



# program identification

| | | |
|---|---|---|
| title | : | PAINT |
| author | : | Frank Druijff |
| purpose | : | To fill a shape with a specified color |
| comment | : | a machine-language version will be available on TOOLKIT 5 |

* * * * p a i n t * * * * p a i n t * * * * p a i n t * * * *

De instructie PAINT is een grafische instructie die de DAI moest ontberen. De instructie die wel in de standaard instructieset van sommige andere machines zit kleurt een willekeurig vlak in de door de gebruiker opgegeven kleur. Wij wilden die mogelijkheid ook graag voor de DAI realiseren en dat is gelukt. Na wat gefilosofeer over een mogelijk methode kwam Hans Peters met de gouden tip. Het schrijven van een programma om deze tip te gebruiken kostte toen ook nog het nodig denkwerk maar was uit te voeren. Toen de basicversie goed bleek te werken (nog geeneens zo erg traag) kwam de volgende stap: het programma in machinetaal schrijven. Er was een aanzienlijke snelheidswinst maar toch bleef het verlangen naar meer snelheid. Die was mogelijk omdat mijn machinetaalversie gebruik maakte van de SCRN-routine in rom. Dat betekende dat de DAI bij elke aanroep van deze routine de x en y waarde controleerde op grootte en dan elke keer opnieuw de byte uitrekende waarin het gewenste punt stond. Dit is maar een keer in het begin nodig. Ik heb toen Nico Looije om hulp gevraagd en die heeft mijn versie grondig onder handen genomen. Een gemiddelde snelheidverbetering van 4 a 5 maal. Samen nog even de laatste hand aan de routine gelegd en de DAI heeft nu ook een PAINT.
Voor DCR en floppy bezitters moet het mogelijk zijn de routine zo aan te passen dat PAINT direct gebruikt kan worden of in een programma op de manier zoals DCR en floppy de 'nieuwe' instructies gebruiken.
De routine zal opgenomen worden in een nieuwe TOOLKIT-collectie.

Hoe werkt de paint ?

1) we kiezen een punt in de figuur die we wensen te kleuren.
2) we geven de volgende gegevens door aan de routine: de x en y van ons punt de randkleur van de op te vullen figuur en de gewenste vulkleur.
3) de routine begint naar de linkerrand te zoeken.
4) dan wordt vanuit dit punt de rechterrand gezocht.
5) er wordt een lijn getrokken in opvulkleur van linkerrand naar rechterrand.
6) er wordt een volgend punt op de rand gezocht vanwaar naar rechts een lijn (evt met lengte 1) getrokken kan worden.
7) als we nog niet aangekomen zijn bij ons uitgangspunt bij de rand dan worden de punten 5 en 6 herhaald totdat dit wel zo is en dan zij we klaar

Vooral punt 6 heeft het nodige denkwerk gekost. Hoe vindt je het volgende punt en hoe ga je de figuur rond. Het bleek achteraf erg simpel te zijn. Mensen die er zelf over willen nadenken moeten dan nu niet verder lezen.
We gaan als volgt rond (rechtsom): kijk of je naar boven kan zo ja doe dat en kijk dan of je naar links kan zo ja doe dat en kijk dan of je naar beneden kan en zo ja doe dat dan en kijk dan of je naar rechts kan en zo ja doe dat dan en kijk dan weer of je naar boven kan enz.. Kan je niet naar boven gaan dan naar rechts gaan, kan je niet naar rechts dan naar beneden, kan je niet naar beneden dan naar links. En ..... kun je niet naar links dan heb je een punt van waaruit je een lijn naar rechts kunt trekken. Goed geprogrammeerd zijn dit vier regels. (20,30,40 en 50) In regel 60 wordt gecontroleerd of we al klaar zijn en regel 70 wordt de lijn getrokken.

In het basicprogramma zit echter nog een belangwekkend deel: regel 100 t/m 170 Het is een zeer simpele, maar wel fraaie methode om een punt op het beeld te laten bewegen. Met de cursortoetsen gewoon een naar links,rechts,boven of onder maar met cursortoetsen en ingedrukte shift met stappen van tien. Dit aantal kan ook anders gemaakt worden. Dit programmadeel loont de moeite om op te nemen in een set standaardroutines.
Het basicprogramma voorziet verder in een mogelijkheid om zelf uw figuur te tekenen of de standaardfiguur te kiezen.(uit de DATA lijnen)

Nog wat laatste opmerkingen over de twee machinetaal routines. Eerste versie werkt in alle modes,maar controleert niets. De tweede versie is sneller en controleert op juiste gebruik kleuren en geeft OFF SCREEN als de beeldrand bereikt wordt. Het nadeel dat deze versie alleen in vier-kleuren modes werkt is niet een echte beperking, door kleurconflicten is het programma nauwelijks zinvol te gebruiken in de zestien-kleuren modes. Probeer de basicversie maar eens in een zestien-kleuren mode.

Al de oplossingen hebben echter een nadeel; als in de figuur nog een andere figuur ligt dan werpt deze een 'schaduw' naar rechts. Dit is nauwelijks op te lossen, tenminste niet op een zinvolle manier. Mogelijkheden zijn een tweede keer linksom rond lopen en dan juist naar links een lijn trekken maar dan is gedeelte tussen twee binnenfiguren weer ongevuld en het programma wordt met een factor twee vertraagd. Andere mogelijkheden als het 'oversteken' naar zo'n binnenfiguur leveren grote problemen op bij het bijhouden van terugkeer adressen.
Een paar manieren om het probleem op te lossen.
1) trek een lijn tussen buitenrand en binnenfiguur. Alles wordt dan goed gevuld en trek dan de verbindingslijn opnieuw in de vulkleur.
2) teken buitenfiguur, kleur die, teken binnenfiguur, kleur met andere kleur.
3) Kies nieuw startpunt in het nog niet gevulde gedeelte en dan wordt tekening daar alsnog gevuld. (de binnen figuur is dan rand !)

Hierna het basic-programma en als u het intikt vergeet u dan niet eerst IMPINT te tikken ?

```
10    CLEAR 256:GOTO 400:REM PAINT / F.H. DRUIJFF 12/82
20    IF SCRN(X,Y+1)<>G THEN Y=Y+1:GOTO 50:REM BOVEN
30    IF SCRN(X+1,Y)<>G THEN X=X+1:GOTO 20:REM RECHTS
40    IF SCRN(X,Y-1)<>G THEN Y=Y-1:GOTO 30:REM BENEDEN
50    IF SCRN(X-1,Y)<>G THEN X=X-1:GOTO 40:REM LINKS
60    K=X-1:IF X=BX THEN IF Y=BY THEN IF F=1 GOTO 80:F=1
70    K=K+1:IF SCRN(K,Y)<>G GOTO 70:DRAW X,Y K-1,Y W:GOTO 20
80    END
100   C=SCRN(X,Y):DOT X,Y 23:H=GETC:DOT X,Y C:IF H=0 GOTO 100
110   IF H<16 GOTO 170:IF H>23 GOTO 170:V=H/20*9+1
120   ON H MOD 4 GOTO 140,150,160
130   Y=Y+V:IF Y<YMAX GOTO 170:Y=YMAX:GOTO 170
140   Y=Y-V:IF Y>0 GOTO 170:Y=0:GOTO 170
150   X=X-V:IF X>0 GOTO 170:X=0:GOTO 170
160   X=X+V:IF X<XMAX GOTO 170:X=XMAX
170   RETURN
300   PRINT "Move blinking dot. If inside field to color press space ."
310   GOSUB 100:IF H<>32 GOTO 310:X=X+1
320   X=X-1:IF SCRN(X,Y)<>G GOTO 320:X=X+1:BX=X:BY=Y:F=0:GOTO 60
400   G=14:W=1:COLORG 0 W G 15:MODE 4A:REM INITIALISATION
410   INPUT "Standard/Own drawing ";VE$:PRINT
420   X=33:Y=33:IF VE$="O" GOTO 500
430   READ P,Q:IF P+Q=0 GOTO 300:DRAW P,Q X,Y 22:X=P:Y=Q:GOTO 430
500   GOSUB 100:IF H=13 GOTO 300:IF H<>32 GOTO 500
510   IF F=0 THEN DOT X,Y 22:F=1:P=X:Q=Y
620   DRAW X,Y P,Q 22:P=X:Q=Y:GOTO 500
900   DATA 147,12,147,45,90,45,80,31,80,60,120,70
910   DATA 85,70,75,60,66,45,45,55,33,33,0,0
```

veel plezier
Frank H. Druijff

## UNIVERSELE PC. DCE-BUS INTERFACEKAART (GICC

Deze kaart heeft hetzelfde doel en dezelfde logische opbouw als de DCE-F module van DAI, met dit verschil dat bij deze kaart alle aansluitingen in volgorde liggen zoals bij de personal computer.(de industriele kaarten hebben een afwijkende volgorde.)
Het gevolg hiervan is dat deze "GICC" (General Interface Control Card) met een kaartadapter rechtstreeks op de DCE-bus van DAIpc kan worden aangesloten door middel van een gewone bandkabel als het een enkele kaart betreft.
Voor het aansluiten van meerdere kaarten doen we een beroep op de busprint van Elektuur (EPS80024).
Iedere busprint biedt plaats voor max 5 interface kaarten, terwijl meerdere busprinten plaats bieden voor max. 15 kaarten volledig in BASIC programmeerbaar met de OUT I,J en de INP(I) instructies.
Voor de niet-ingewijden in de DCE-BUS techniek verwijzen we naar DAInamic No 1 p 3,4,5 en 6 , of naar "THE BEST OF DAInamic" p 4,5,6 en 7.
De GICC is een eurokaart met een 64-polige DIN 41612 connector een volledige adresdecodering, 3 8-bits I/O poorten met een command register (8255A), een wrapping area van 110mm x 110mm en alle DCE-BUS signalen.
Het kaartadres is door middel van 4 DIP-switches door de gebruiker zelf instelbaar (1 tot 15).
Van de 64-polige DIN connector zijn er nog een aanzienlijk aantal lijnen onbezet, die we kunnen gebruiken om onze bus-structuur tussen de verschillende interfacekaarten verder uit te bouwen.(oa bijkomende voedingsspanningen +/- 15V)

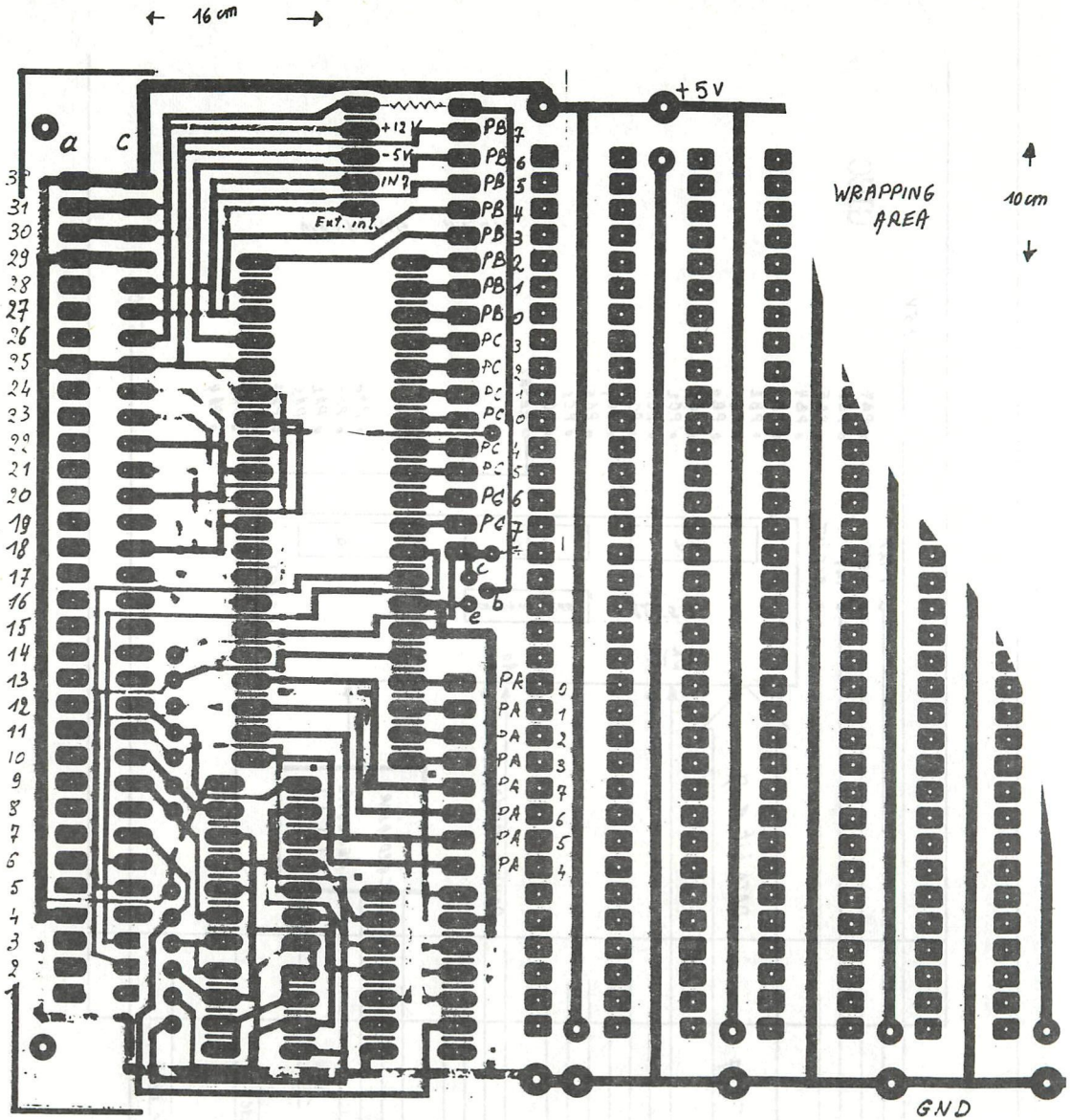
## THE DCE-CONCEPT

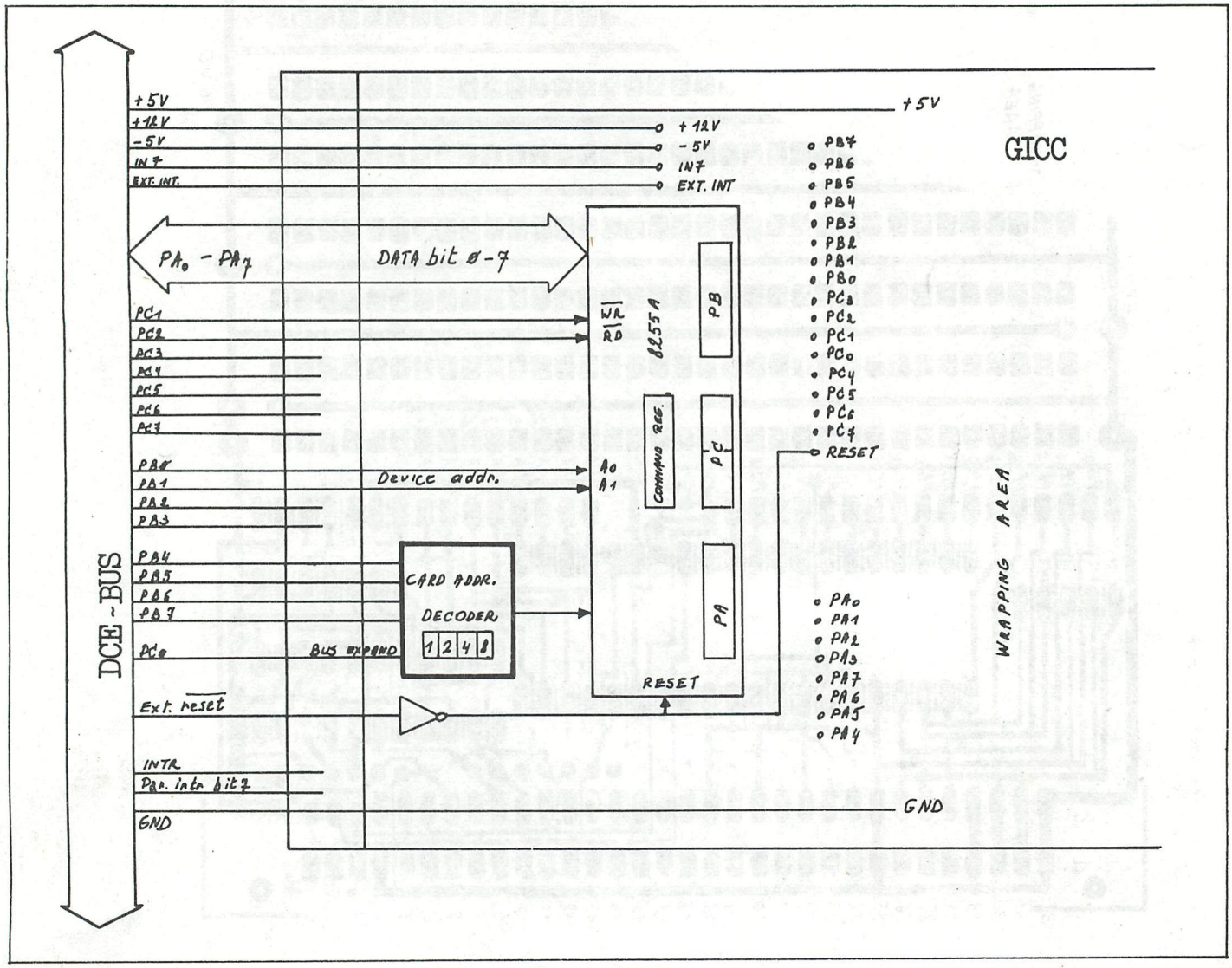
The DCE-BUS provides an easy way for the exchange of data and control information between the DCE-BUS of the personal computer and the interface card(s).
The cards are driven by the GIC (8255) on the personal computer, configured to provide 8-bit input/output transfer, read & write signals, two external interrupts requests and eight address control lines for selective access to the cards on the bus.
Each card is plugged into the DCE-BUS (through Elektuur 80024 motherboard) and given a unique address via 4 dipswitches on the card.
Read and Write subroutines (IN/OUT) are provided (in DAI-BASIC) to simplify the tranfer of data between the personal computer and the interface cards.
GIC PORT 0 is used as a bidirectional data path for data transfer between the computer and the cards.
GIC PORT 1 is used to specify the address of the connecting card.
GIC PORT 2 issues the control signals to complete the transfer logic.
PORT 0 must be alternated between input and output by software command.

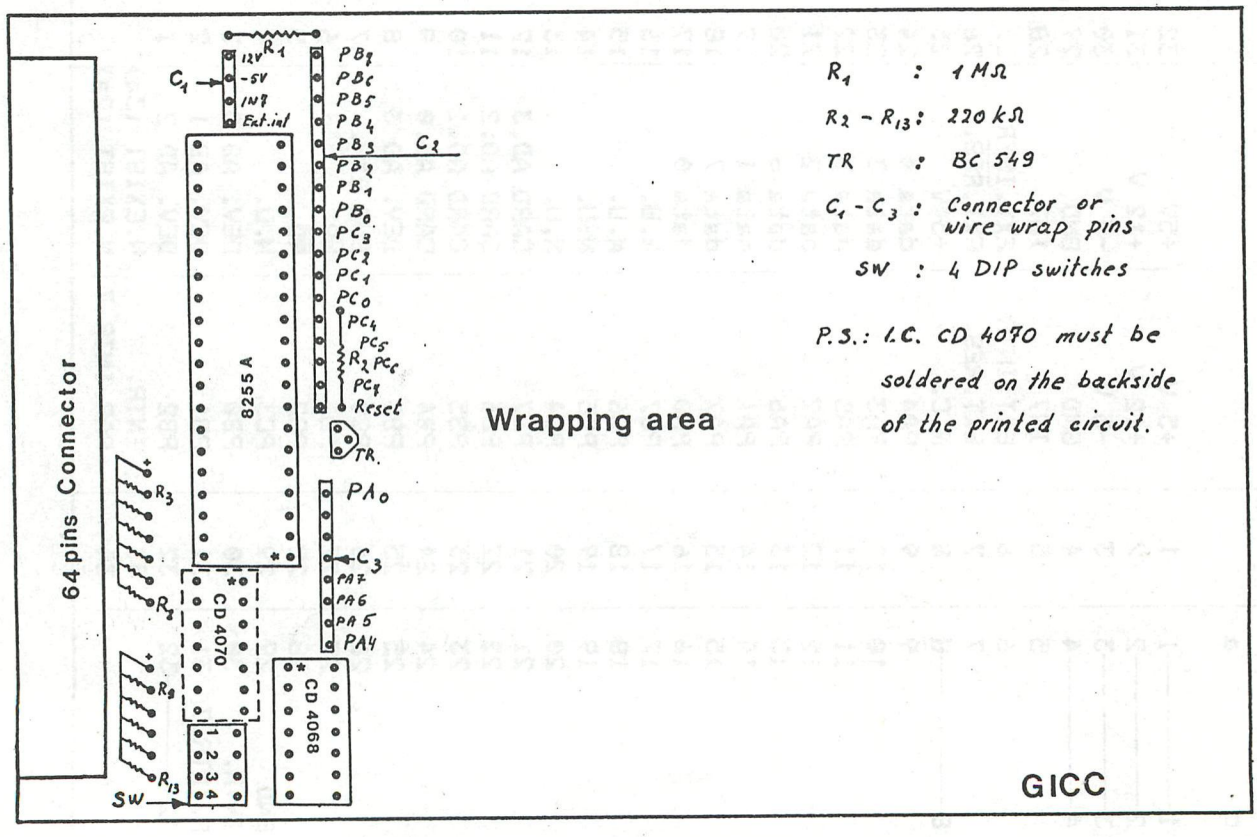
*For more information, please consult :*

" DCE MICROCOMPUTER SYSTEM DESIGNER'S HANDBOOK"

| PRINT EPS80024 (Elektuur) | | pin | DAIpc Output | DCE-BUS Function | CONNECTOR |
|---|---|---|---|---|---|
| c | a | | | | |
| 1 | 1 | 1 | +5 V | +5V | 32 |
| 2 | 2 | 2 | +12 V | +12 V | 31 |
| 3 | 3 | 3 | -5 V | -5 V | 30 |
| 4 | 4 | 4 | GND | GND | 29 |
| | 5 | 5 | IN7 | IN7 | 28 |
| | 6 | 6 | EXT.INTR | EXT.INTR. | 27 |
| | 7 | 7 | EXT.RES. | EXT.$\overline{\text{RES}}$. | 26 |
| 8 | 8 | 8 | N.C. | +5 V | 25 |
| | 9 | 9 | PA4 | data 4 | 24 |
| | 10 | 10 | PA3 | data 3 | 23 |
| | 11 | 11 | PA5 | data 5 | 22 |
| | 12 | 12 | PA2 | data 2 | 21 |
| | 13 | 13 | PA6 | data 6 | 20 |
| | 14 | 14 | PA1 | data 1 | 19 |
| | 15 | 15 | PA7 | data 7 | 18 |
| | 16 | 16 | PA0 | data 0 | 17 |
| | 17 | 17 | PC7 | N.U. | 16 |
| | 18 | 18 | PC6 | N.U. | 15 |
| | 19 | 19 | PC5 | N.U. | 14 |
| | 20 | 20 | PC4 | N.U. | 13 |
| | 21 | 21 | PB7 | CARD AD.3 | 12 |
| | 22 | 22 | PB6 | CARD AD.2 | 11 |
| | 23 | 23 | PB5 | CARD AD.1 | 10 |
| | 24 | 24 | PB4 | CARD AD.0 | 9 |
| | 25 | 25 | PB3 | DEV. AD 3 | 8 |
| | 26 | 26 | PC0 | BUS EXP. | 7 |
| | 27 | 27 | PC1 | $\overline{\text{WR}}$ | 6 |
| | 28 | 28 | PC2 | $\overline{\text{RD}}$ | 5 |
| GND | 29 | 29 | PC3 | N.U. | 4 |
| INTR. | 30 | 30 | PB0 | DEV. AD 0 | 3 |
| P.INTR.7 | 31 | 31 | PB1 | DEV. AD 1 | 2 |
| 32 | 32 | 32 | PB2 | DEV. AD 2 | 1 |
| | | 33 | INTR. | N.EXIST (3a) | |
| | | 34 | PAR.INTR 7 | N.EXIST (2a) | |






$R_1$ : 1 MΩ

$R_2 - R_{13}$ : 220 kΩ

TR : Bc 549

$C_1 - C_3$ : Connector or wire wrap pins

SW : 4 DIP switches

P.S.: I.C. CD 4070 must be soldered on the backside of the printed circuit.

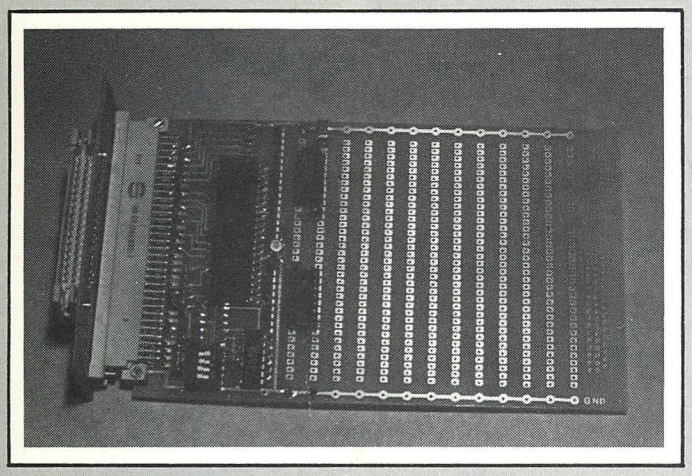


photo 1 Assembled interface-card

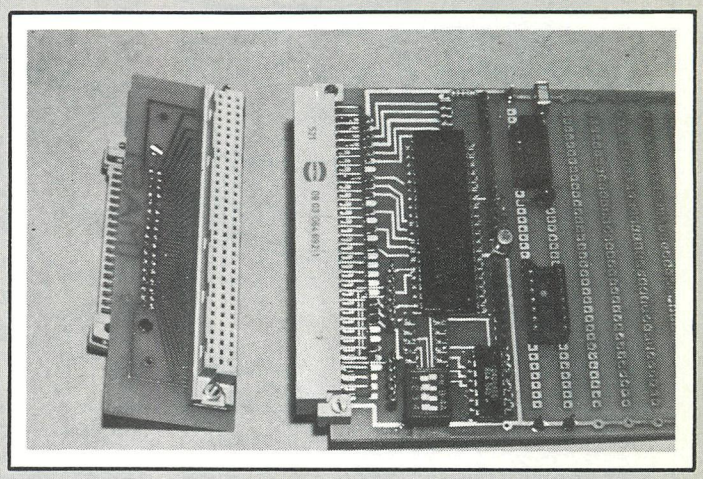


photo 2 Assembled interface-card + adapter card

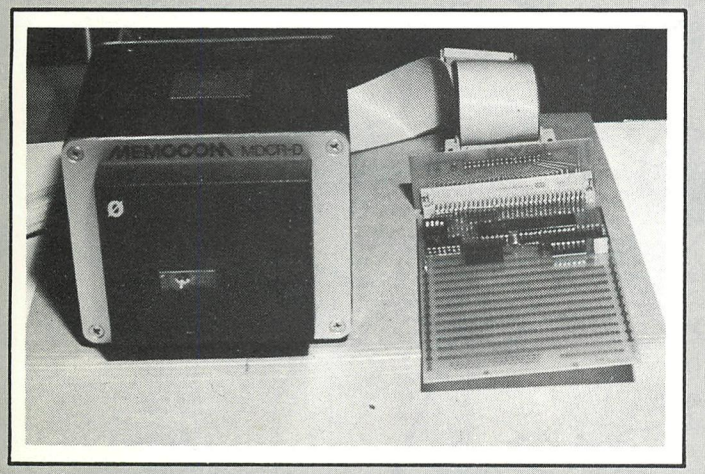



photo 3 MDCR + interface-card

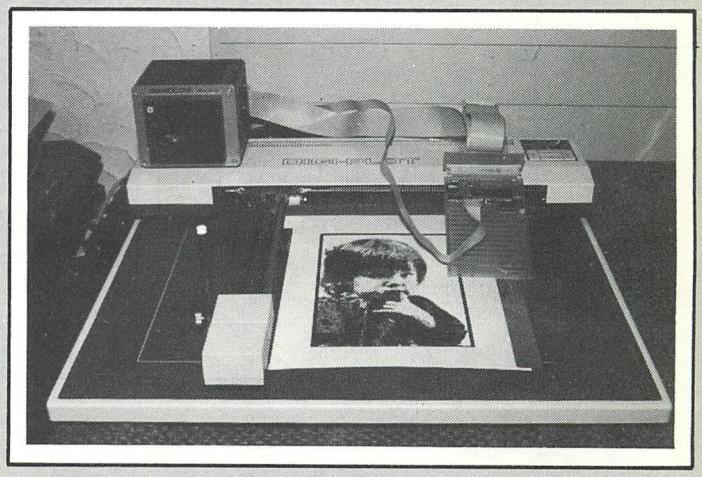


photo 4 MDCR + plotter, together on the DCE-bus

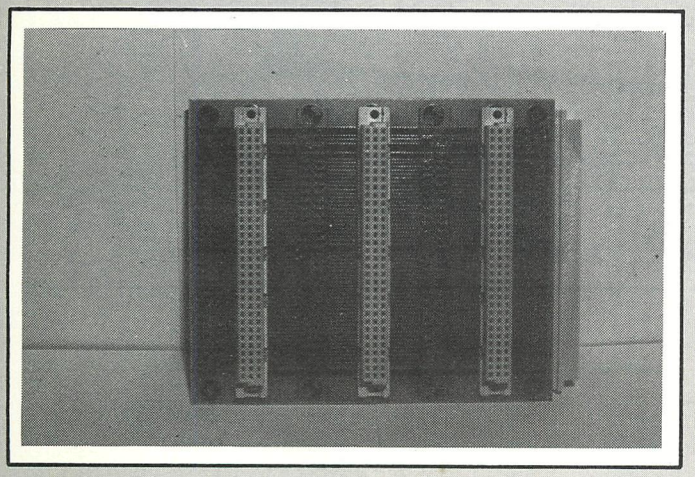


photo 5 Elektuur EPS80024,
motherboard for interface-cards

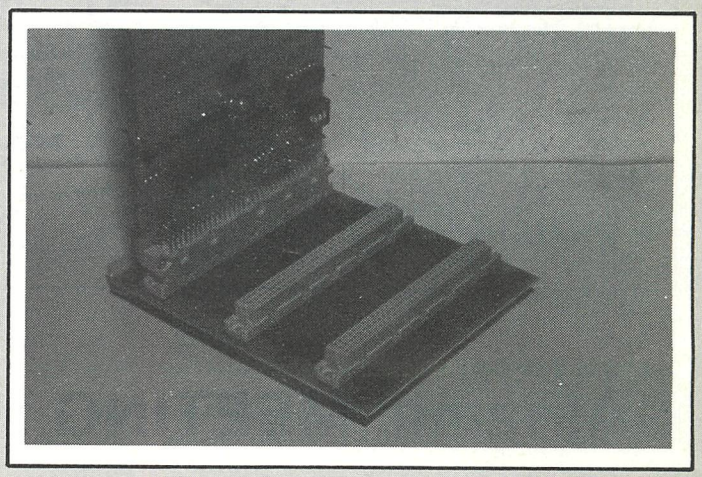
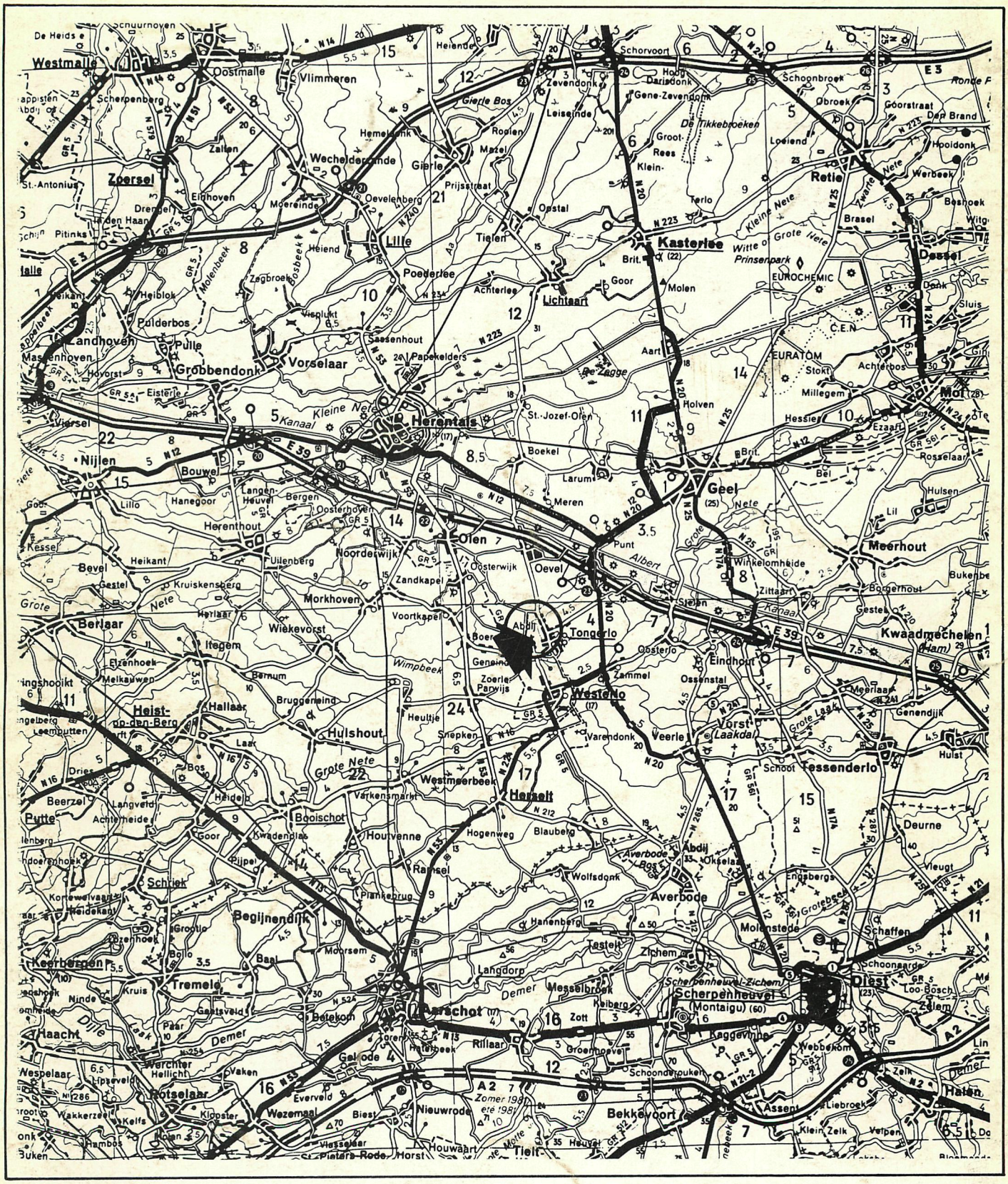


photo 6 Elektuur EPS80024 with 1 interface-card




# DAInamic MEETING :

# on 9th April 1983 10.00 - 18.00 Hr
# in TONGELSBOS Bosstr. 2 3180 WESTERLO