DAI NAMIC

tweemaandelijks tijdschrift januari - februari 1983

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

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, machinetaal-specialisten 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 disc-drives 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 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 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:

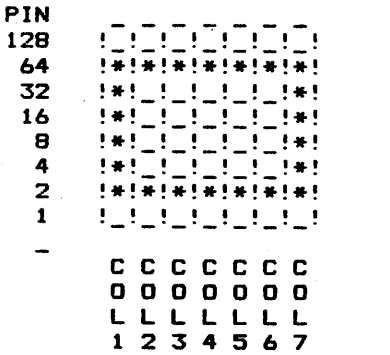
```
          TOP
          PIN   VALUE            In this example VALUE is
                                 the pin number.
PRINT      *      128
           *       64
HEAD       *       32            In selecting what pins to
           *       16            fire, add the values of the
---->      *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

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.



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

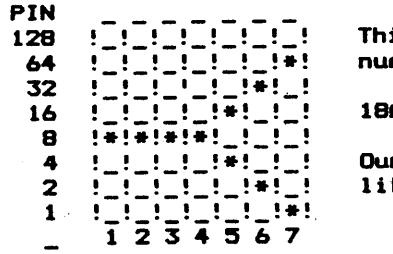
DATA 126,66,66,66,66,66,126

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.



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 0 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:

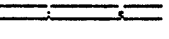
```
  5 GOTO 100
 10 YOUR SYSTEM'S SUBROUTINE
    . . .

 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 uitgebreide 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,

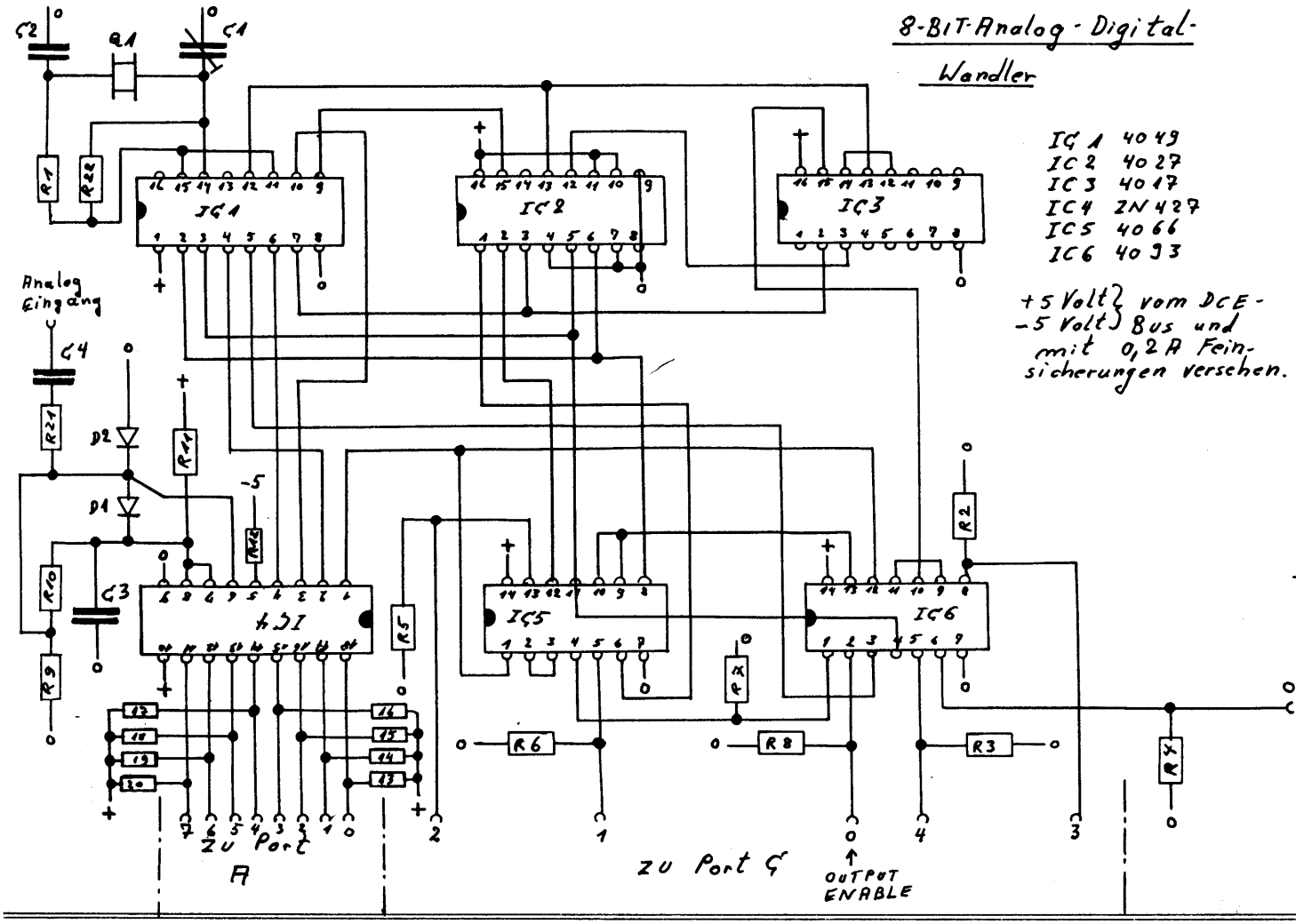
Um schallmessungen durchfuhren zu konnen habe ich eine schaltung mit dem ADW-IC ZN 427 von Ferranti entworfen und auf einer Veroboard lochrasterplatine aufgebaut. Diese schaltung passt in ein kunststoffgehause von 150x80x50 mm und ist met einem flachkabel an den DCE-bus angeschlossen. Mit dem beiliegenden 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.

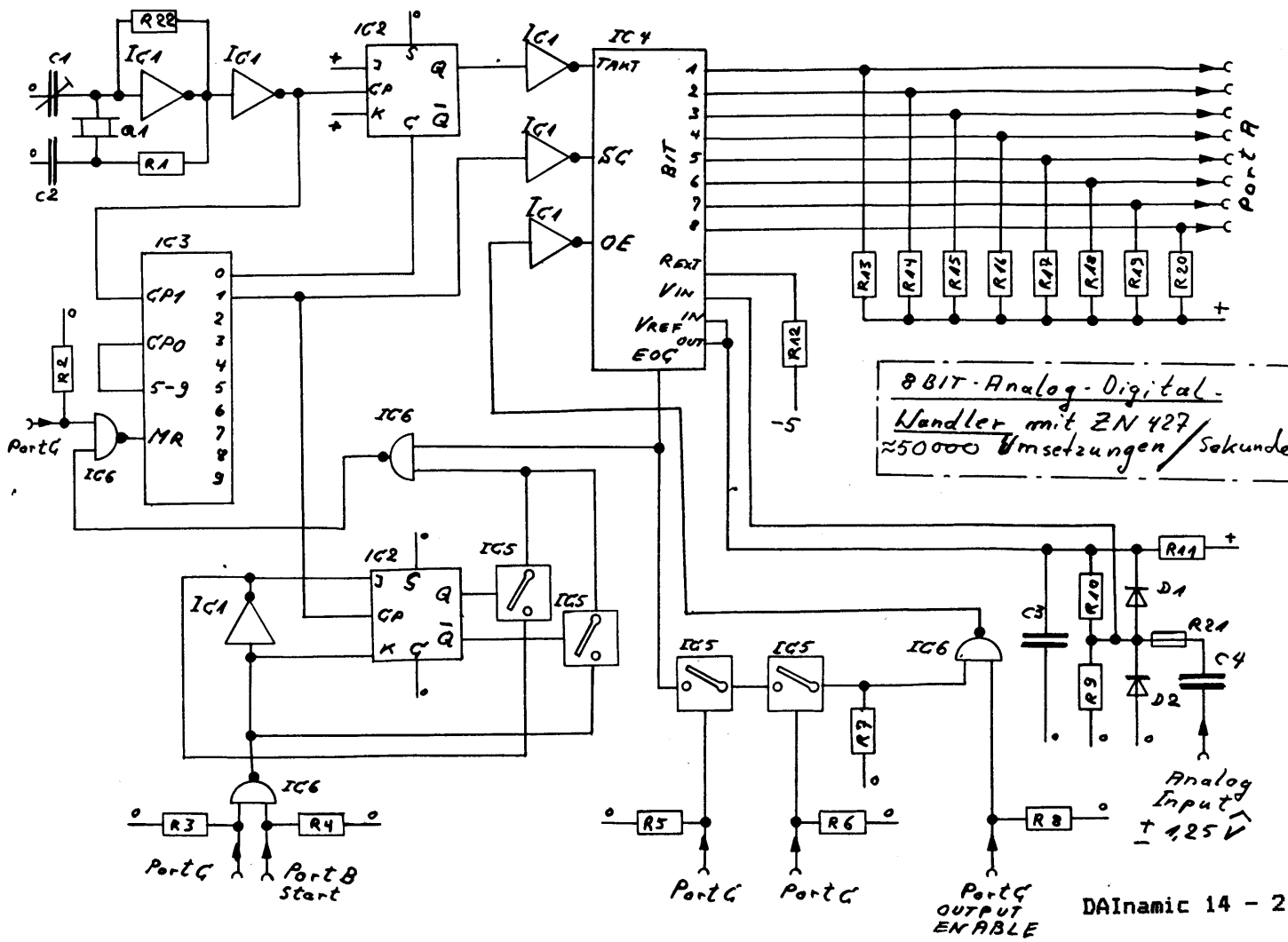
Bei dem aufbau der schaltung und dem uberprufen der im datenblatt fig.3 timing diagram vorgegebenen steuerimpulse fur den ZN 427 habe ich den DAIpc als logikanalysator eingesetz. Dabei wird der B und C port als ausgang und der A port als eingang programmiert. Mit einem der ausgange wird vom BASIC aus 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.

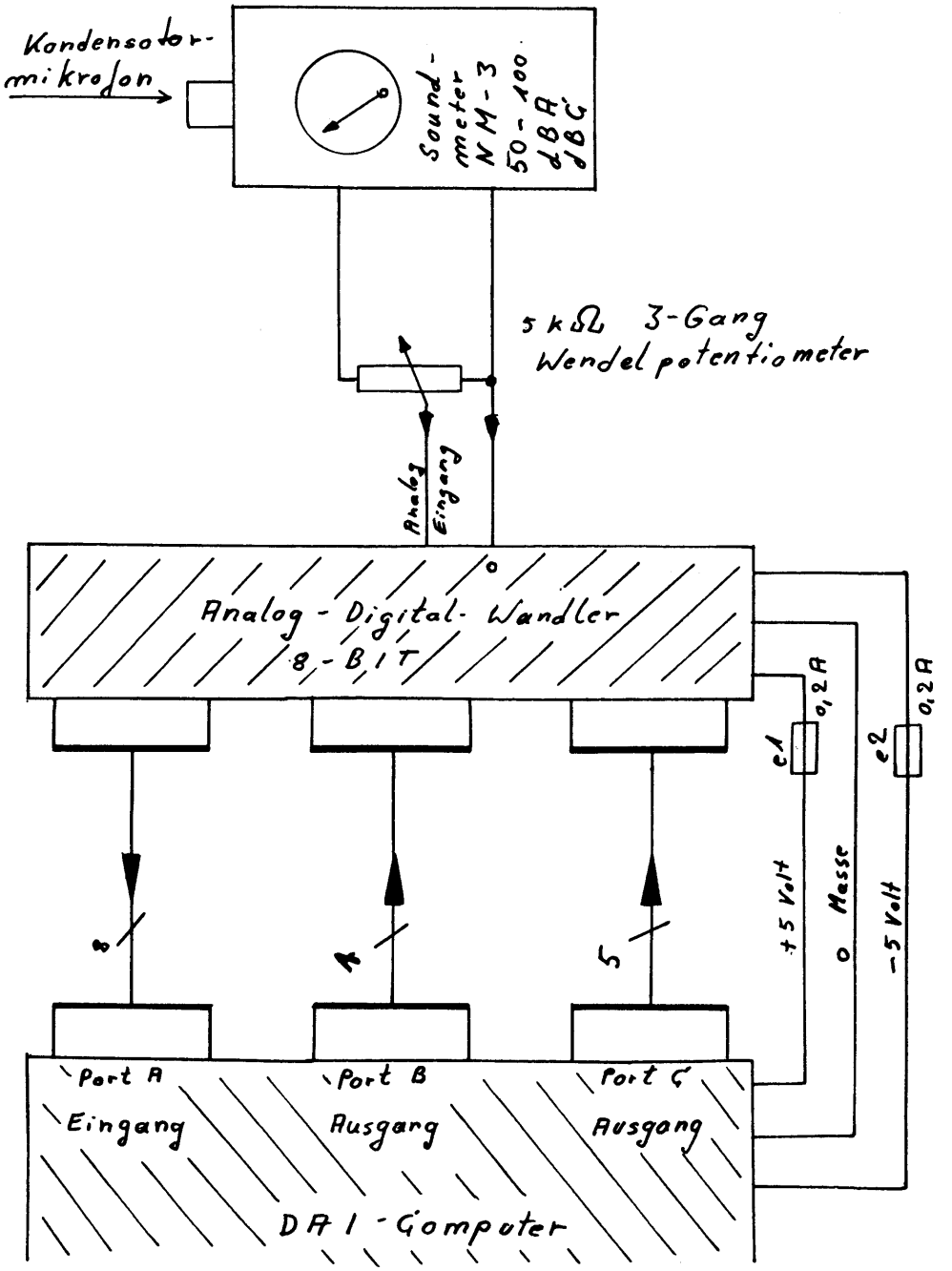
Viel spass beim nachbauen
mit freundlichen grussen

K.H.Kopp

8-BIT-Analog-Digital-Wandler







Stückliste für Analog Digital Wandler

IC 1 CD 4049
IC 2 CD 4027 CMOS
IC 3 CD 4019
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 nF Styroflex
C 4 2 nF 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 ⎫
: ⎬ 10K
R20 ⎭
~~R21~~
R21 1K
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 ! * !

****

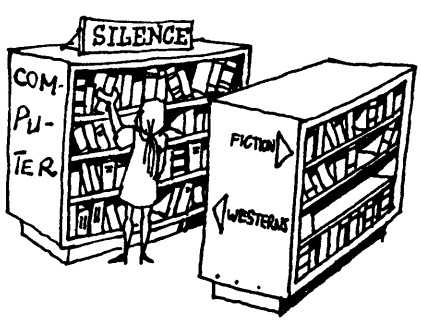
1)- I have a number of brandnew CASSETTE MECHANISMS for sale.
These units were especially manufactured for REMOTE CONTROL.
Each unit has 3 solenoids mounted on a base, to activate levers so as to enable FFW, FRW & RUN. A head and speed stabilised motor is also mounted.
Shematics for driving the solenoids 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.

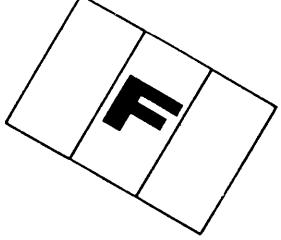
For using the DCE-bus, see also the article in the DAInamic Newsletter nr.1 of September 1980 and the Intel data sheets of the 8255 Programmable Peripheral Interface.
When using the power-on DCE initialisation routine, take good care of the several handshake signals on ports PA, PB and PC.

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

logiciels

Cassetes AUDIO

CENTIPEDE	85 Frs
DRIVER	85 Frs
GAMES 8	120 Frs
TOOLKIT 3	
DAInatexte	

(ajouter 15 Frs pour les cassettes DCR)

matériels

SNG	325 Frs
Caractères	150 Frs

Littérature

DAI Schémas	120 Frs
URGENT	125 Frs
the best of DAInamic (flamand)	90 Frs

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

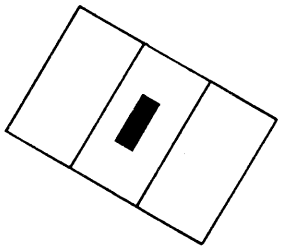
DAI pc FIRMWARE MANUAL

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