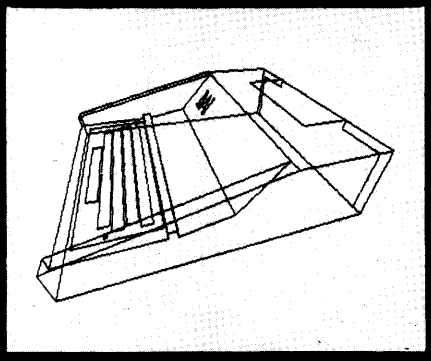
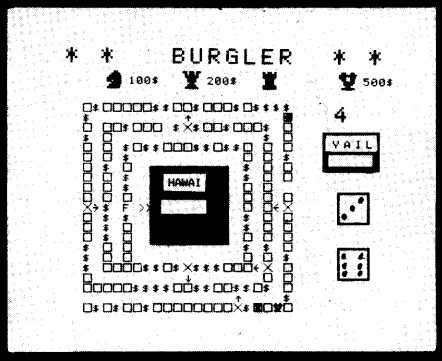
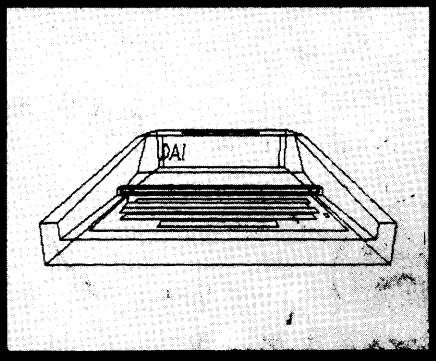
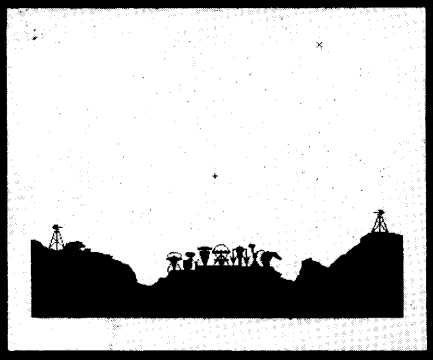
DAI NAMIC

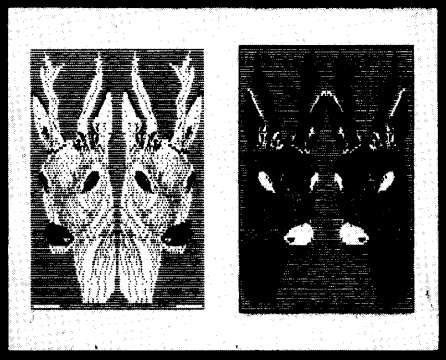
tweemaandelijks tijdschrift juli - augustus 1982

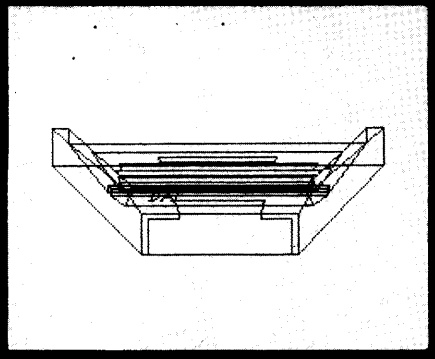


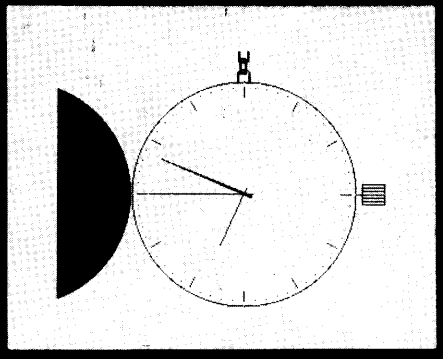


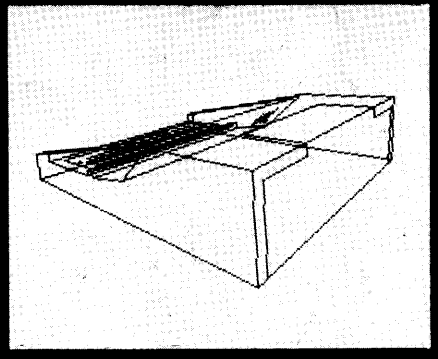












een uitgave van dainamic v.z.w.
verantw. uitgever w. hermans, heide 4 - 3171 westmeerbeek

COLOFON

DAInamic verschijnt tweemaandelijks. abonnementsprijs is inbegrepen in de jaarlijkse contributie : 750 Bfr. 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	ENQ	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

DAInamic

WESTMEERBEEK, 1 AUGUSTUS 1982

BESTE LEDEN,

ER IS DE LAATSTE 3 MAANDEN HEEL WAT GEBEURD ROND HET PRODUKT DAI.
DAT DE TOEKOMST VOOR DE DAI-GEBRUIKERS ER ERG GUNSTIG UITZIET LEEST U IN HET BERICHT VAN DIRK OOGJEN :

W.HERMANS

VOOR VELEN WAS DE DATUM 6 MEI 1982 EEN DATUM OM NIET LICHT TE VERGETEN. HET WAS DE DAG DAT DE RECHTER OFFICIEEL BESLISTE DAT DAI TE BRUSSEL GESLOTEN MOEST WORDEN EN DE STATUS 'FAILLIET' KREEG.
VANAF DIE DAG IS ER DAG EN NACHT GEWERKT AAN EEN OPLOSSING EN DIE IS NU GEVONDEN !
"PRODATA" BESLOOT DE AKTIVITEITEN VAN DAI VOORT TE ZETTEN EN STICHTTE HIERVOOR EEN NIEUWE FIRMA GENAAMD "INDATA NV".
DEZE NIEUWE FIRMA ZAL ALLE PRODUKTEN VAN DAI GAAN PRODUCEREN EN VERKOPEN ZODAT WIJ WERKELIJK KUNNEN ZEGGEN DAT DE PRODUKTEN 'TERUG OP DE MARKT' ZIJN. EN HOE! DE FINANCIELE BASIS VAN DEZE NIEUWE FIRMA IS DERMATE GROOT, DAT PLANNEN VOOR VERBETERING VAN DE ALGEMENE STRUKTUUR VAN DE FIRMA EN PRODUKTEN NU EINDELIJK VERWEZENLIJKT KUNNEN WORDEN. DIT MEDE DOOR DEELNAME VAN ANDERE FINANCIELE GROEPEN, ZODAT "INDATA NV" NU ZEER KRACHTIG OP DE MARKT KAN KOMEN.

ER IS EEN NIEUW MANAGEMENT, MET GOEDE EN TALENTVOLLE MENSEN, DIE ABSOLUUT VAN PLAN ZIJN VAN DEZE MAATSCHAPPIJ EEN SUCCES TE MAKEN.
HARDWAREMATIG ZIJN ER PLANNEN TOT VERBETERING EN UITBREIDING, EN OP SOFTWAREGEBIED WILLEN WIJ OOK VOORUIT, ZODAT ER ZEKER BINNEN NIET AL TE LANGE TIJD POSITIEVE ZAKEN TE VERWACHTEN ZIJN. WAT DIT PRECIES INHOUDT, KUNNEN WIJ NOG NIET VERKLAPPEN, MAAR DE GEBRUIKERS KRIJGEN BESLIST DE GELEGENHEID HUN MACHINE UIT TE GAAN BREIDEN.

OOK DE VROEGERE LEVERINGSPROBLEMEN ZULLEN TOT HET VERLEDEN BEHOREN.
VANZELFSPREKEND ZULLEN WIJ OPNIEUW MOETEN OPBOUWEN, EN ZO'N BEGIN IS ALTIJD MOEILIJK, MAAR DE BASIS IS GEZOND, EN DE WILSKRACHT GROOT.
WIJ ZIEN BESLIST UIT NAAR EEN GOEDE TOEKOMST VOOR DE DAI PRODUKTEN EN HOUDEN U OP DE HOOGTE VAN ONZE VORDERINGEN. HIERBIJ ENKELE GEGEVENS BETREFFENDE "INDATA NV":

MAY 6, 1982 WAS THE DAY THAT DAI WAS BANKRUPT. FROM THAT DAY WE WORKED VERY HARD TO GET THE COMPANY OPEN AGAIN, AND THAT WAS NOT EASY! "PRODATA", A COMPANY IN BRUSSELS, DECIDED TO BUY ALL THE RIGHTS OF DAI, AND SINCE THEN WE COULD PRODUCE AND SELL AGAIN.
AND HOW! THE FINANCIAL BASE OF THIS NEW COMPANY IS STABLE AND HIGH, SO THAT OLD PROBLEMS LIKE PRODUCTION DELAYS ARE SOLVED.
THE COMPANY GOT THE NAME "INDATA NV", WITH A WHOLE NEW MANAGEMENT WITH SPIRIT AND WILL TO LET THIS COMPANY GROW AS NEVER BEFORE.
THIS MEANS THAT WE CAN LOOK FORWARD TO NEW APPLICATIONS IN SOFT AND HARDWARE IN A SHORT TIME.
OF COURSE WE NEED TO START UP AGAIN, BUT THE FUTURE LOOKS GREAT.
WE WILL KEEP YOU INFORMED ABOUT OUR PROGRESS. HEREUNDER PLEASE FIND THE COMPANY INFORMATION.

INDATA NV, RUE DE LA FUSEE 60 (RAKETSTRAAT 60), 1130 BRUSSEL
TELEFOON (31)-2-2169720.

DIRK OOGJEN, SALES & MARKETING INDATA NV

BLADWIJZER CONTENTS SOMMAIRE IN DIESEM HEFT

NEW SOFTWARE IN OUR LIBRARY

TITLE	PRICE AUDIO	PRICE DCR
- GAMES COLLECTION 8	750 Bfr	900 Bfr
- GAMES COLLECTION 9	750 Bfr	900 Bfr
- TOOLKIT 3 (2 TAPES)	1000 Bfr	1300 Bfr

AVAILABLE SOON :

New 4K Character generator for DAIpc,Contains all DAI characters (some of them enhanced),greec alfabet, german and french characters (accentuees) and beautiful graphic symbols (card symbols,games symbols ....)

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

Ik ben van plan in dit blad een rubriek te gaan verzorgen over technieken en problemen bij het programmeren. Vaak zal het algemeen zijn maar regelmatig speciaal toegespitst op de DAI. Anders gezegd: de besproken methodes gelden altijd voor de DAI maar vaak ook voor vele/alle andere computers.
De onderwerpen die aan bod komen zullen bepaald worden aan de hand van uw suggesties en/of zwakke gedeeltes van ingestuurde programma's. Ik denk in dit verband bv aan het snel tekenen van cirkels, voorkomen van kleurconflicten, werken met arrays, snelheid en leesbaarheid van programma e.d.

Voor deze eerste keer wilde ik als onderwerp het verkrijgen van een lijst met bepaalde maar wel willekeurig verdeelde getallen nemen. Het idee voor dit onderwerp kwam bij mij op door het programma 'BINGO' van T. Groeneveld (blz 136 N10) Ik hoop dat mijnheer Groeneveld nu niet na dit artikel zal denken dat iedereen hem een slechte programmeur zal vinden. In ieder geval vind ik dat zelf niet en wel omdat hij zelf een oplossing heeft gevonden. Inderdaad niet de beste maar ik heb heel wat liever een domme,onhandige,lange,omslachtige,onduidelijke methode (Nee mijnheer Groeneveld dit slaat niet allemaal op U) dan een foute. En zoals ik de vorige keer al schreef krijgen wij nog regelmatig programma's binnen die domweg fout zijn; of haast nog erger alleen overgetikt.

Om het probleem nog eens te herhalen: In het programma hebben we een array nodig waar de getallen 1 t/m 75 in staan, deze moeten echter wel in een willekeurige volgorde staan. Als het programma trouwens niet werkt bij U, het volgende recept: intikken wat er staat maar dan wel na IMPINT.
Dat heeft U ook niet gedaan mijnheer Groeneveld (zie de .0's maar).
Het originele programma:

```
10 CLEAR 1000:DIM G(76.0)

25    R=RND(75.0)+1.0
26    ENVELOPE 1 15,9;0:SOUND 1 1 15 0 FREQ(RND(269.0)+31.0)
27    NOISE 1 15
30    FOR A=1.0 TO 75.0:IF G(A)<>R THEN NEXT A
40    IF A<76.0 AND G(A)=R THEN 25
50    IF G(I)=0.0 THEN G(I)=R
65    I=I+1.0:IF I=76.0 THEN 100
70    GOTO 25
```

We gaan even analyseren:
regel 25 we kiezen een geschikt getal
regel 26 en 27 we veraangenamen de wachttijd
regel 30 we controleren of we dit getal niet al eerder hadden gekozen
regel 40 als dit wel al eerder gekozen is kiezen we een ander getal
regel 50 als er nog niets ingevuld is zetten we ons getal in de tabel
regel 65 volgende keer getal op volgende plaats zetten en ophouden als we ze allemaal gehad hebben
regel 70 volgende getal als we nog niet klaar zijn

Zoals U zelf kunt vaststellen werkt dit programmadeel correct maar wel traag.
Op mijn machine (met matchchip) kwam ik tot een gemiddelde van 72 seconden.
Wat kunnen we hier nu aan doen? Eerst het programma licht veranderen.
Ik begon met het verwijderen van de .0's , liet de NOISE en ENVELOPE uit de loop en verving de RND bij de sound door R%4 wat veel sneller is.
Ik boekte inderdaad enige tijdwinst maar niet groots gemiddelde nu 66 seconden.

```
10    CLEAR 1000:DIM G(76.0)
16    ENVELOPE 1 15,9;0
17    NOISE 1 15
25    R=RND(75)+1
28    SOUND 1 1 15 0 FREQ(R*4+31)
30    FOR A=1 TO 75:IF G(A)<>R THEN NEXT A
40    IF A<76 AND G(A)=R THEN 25
50    IF G(I)=0 THEN G(I)=R
65    I=I+1:IF I=76 THEN 100
70    GOTO 25
```

Weer maar eens sleutelen. Als het kan meerdere regels tot een samen voegen. De test of A<76 is leek mij overbodig dus verwijderd. De eindtest bezorgt BASIC een slechte naam (spagetti-code) sprongconditie geinverteerd en tevens GOTO-adres.Gemiddelde tijd lag nu op 58 seconden.

```
10    CLEAR 1000:DIM G(76)
16    ENVELOPE 1 15,9;0:NOISE 1 15
25    R=RND(75)+1:SOUND 1 1 15 0 FREQ(R*4+31)
30    FOR A=1 TO 75:IF G(A)<>R THEN NEXT A
40    IF G(A)=R THEN 25:IF G(I)=0 THEN G(I)=R
65    I=I+1:IF I<76 GOTO 25
```

Nu alleen voor de test het geluid verwijderd hiermee kwam gemiddelde op zo'n 48 seconden.En ik vind ook dat je beter 58 seconden met geluid kunt wachten dan 48 seconden zonder. Later zag ik nog dat de loop maar tot I hoeft te gaan. We zitten echter nog steeds met een veel te grote wachttijd dus gaan we iets anders verzinen.

Het oorspronkelijke programma werkt daardoor zo traag omdat een willekeurig getal gekozen wordt en hierna gecontroleerd of dit er al is. Het is beter de getallen die we willen krijgen eerst te nemen en die dan op willekeurige plaatsen in de array neer te zetten. Ik heb deze methode niet uitgewerkt omdat we dan ons probleem omdraaien: heb ik in de eerste methode moeilijkheden een geschikt getal te vinden nu krijg ik het probleem een geschikte plaats te vinden.
De nieuwe methode zal wel iets sneller zijn omdat de controle simpeler is (alleen kijken of plaats leeg is ipv alle getallen controleren).
Maar bedenk ik nu als ik de array nu eens veel groter maak bv 256 dan zal ik gemakkelijk een leeg plaatsje kunnen vinden. Nadat ik dan in een hulparray de getallen 1 t/m 75 heb geplaatst moet ik ze alleen nog maar netjes achter elkaar te zetten in array G(). Voor liefhebbers we kunnen ook in de array zelf aanschuiven natuurlijk. De verkregen tijdwinst is enorm : minder dan 3.4 seconden. Inderdaad drie vier tiende en niet vierendertig.

```
10    CLEAR 10000:DIM G(75),S(255)
20    FOR I=1 TO 75
30    R=RND(256):IF S(R)<>0 GOTO 30:S(R)=I:NEXT
40    FOR I=0 TO 255:IF S(I)<>0 THEN J=J+1:G(J)=S(I)
50    NEXT
```

Maar we zijn nog niet tevreden. De methode is best voor een array met 75 elementen maar minder geschikt als we grotere hoeveelheden krijgen omdat we dan ook moeilijk een leeg plaatsje kunnen vinden. En soms belangrijk het kost heel wat extra geheugenruimte.
Er is dan als we niet erg vinden dat de lijst niet echt random is de volgende methode: we plaatsen de getallen 1 t/m 75 willekeurig in de array van 75 plaatsen. Is de plaats al bezet kiezen we niet random een nieuwe plaats maar nemen gewoon de eerste lege plaats vanaf onze keuze.Deze mogelijkheid heb ik hier niet uitgewerkt omdat het niet random is.

Nu de snelste methode die ook nog eens erg kort is.

```
10    CLEAR 10000:DIM G(75)
20    FOR I=1 TO 75:G(I)=I:NEXT
30    FOR I=1 TO 75:G(0)=G(I):R=RND(75)+1:G(I)=G(R):G(R)=G(0):NEXT
```

We zetten de getallen 1 t/m 75 gewoon op volgorde in de array en verwisselen dan elk getal met een willekeurig ander element. We krijgen een keurige random lijst en erg snel: 2.42 seconden als we G(0) als parkeerplaats gebruiken en zelfs 2.36 seconden als we A nemen.
Dit laatste komt omdat array variabelen voor de DAI moeilijker te vinden zijn.

Weet iemand een nog kortere of snellere methode houd ik mij daarvoor aanbevolen, ik zal er dan zeker op terugkomen.

RECTIFICATIE !!!
Neem mij gerust kwalijk. In DAInamic no 10 heb ik een steek laten vallen.
In mijn artikel over de voordelen van integers heb ik boven in de tweede kolom U trachten te overtuigen. Op de beschreven manier lukt dat niet. Ik ben vergeten er bij te zetten dat U na de reset en voor de tweede keer intikken van het programma nog IMP INT moet intikken. Dan zult U wel overtuigd worden.
Nogmaals mijn welgemeende excuses.

Frank H. Druijff

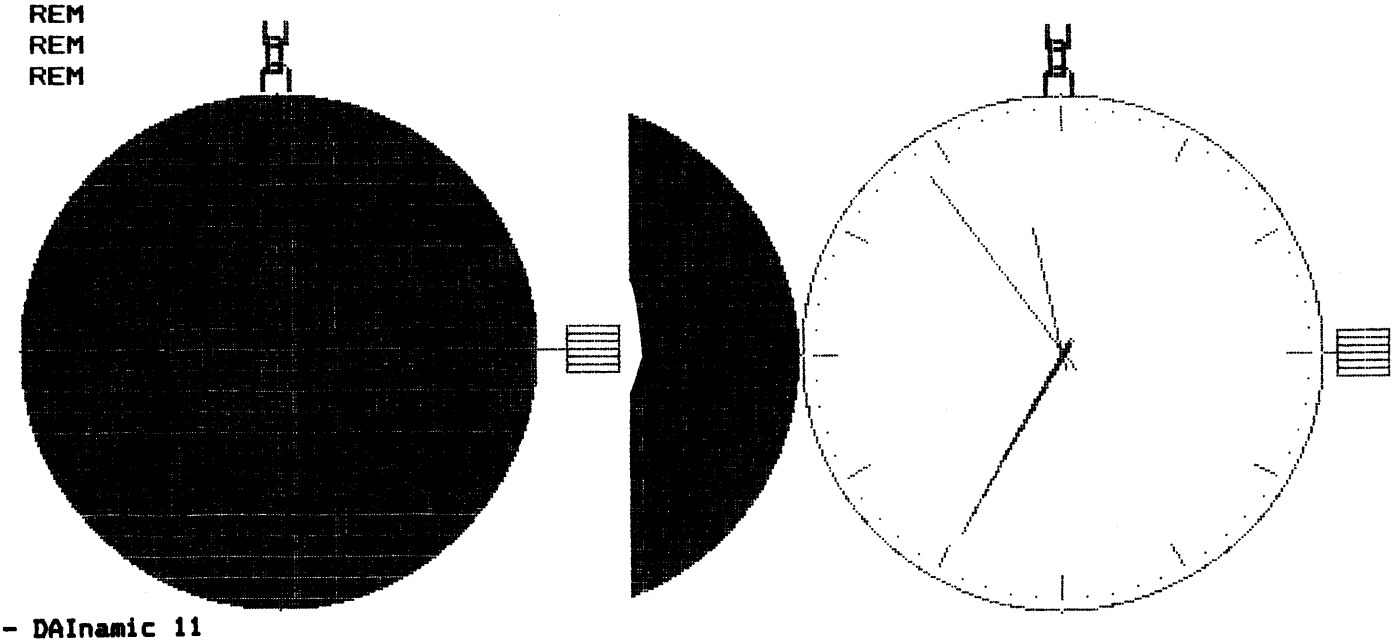
```
1     REM GRAFIK von ERNST JAECKLE,D-7218 Trossingen
2     REM SCHNELLER GEMACHT Frank H. Druijff
3     REM S,A UND B SIND FORMPARAMETER,SS.. STRICHDICHTE
4     REM FUER DRUCKEN "SCREEN COPY PROGRAM" LADEN
10    MODE 0:INPUT "Schrittweite als PI/S  S=(3 ODER 4,6 usw.)";S!:PRINT
20    DIM C(3):ZZ=1:SS!=0.2:IF S!>20.0 THEN SS!=0.125
30    C(1)=9:C(2)=14:C(3)=10:S!=PI/S!:A!=6.0:B!=-1.0
40    COLORG 0 9 14 10:MODE 6:FOR L!=-3.0 TO 3.0 STEP SS!
50    Z=Z+1:IF Z MOD 5=0 THEN ZZ=ZZ+1:IF ZZ=4 THEN ZZ=1
60    CO=C(ZZ):FOR I!=0.0 TO PI+PI+S! STEP S!:C!=A!+B!:H!=C!/B!*I!
70    X!=C!*COS(I!)-L!*B!*COS(H!):Y!=C!*SIN(I!)-L!*B!*SIN(H!)
80    XZ=167.5+20.88*X!:YZ=127.5+Y!*15.0:IF I!=0.0 THEN T=0
90    IF T=0 THEN XA=XZ:YA=YZ:T=1
100   DRAW XA,YA XZ,YZ CO:IF I!<>0.0 THEN XA=XZ:YA=YZ
110   NEXT:NEXT
120   INPUT "Drucken (J,N)";A$:PRINT
130   IF A$<>"J" GOTO 10:MODE 6:COLORG 0 15 14 10
140   POKE #FF06,17:CALLM #300:POKE #FF06,19:GOTO 10
```

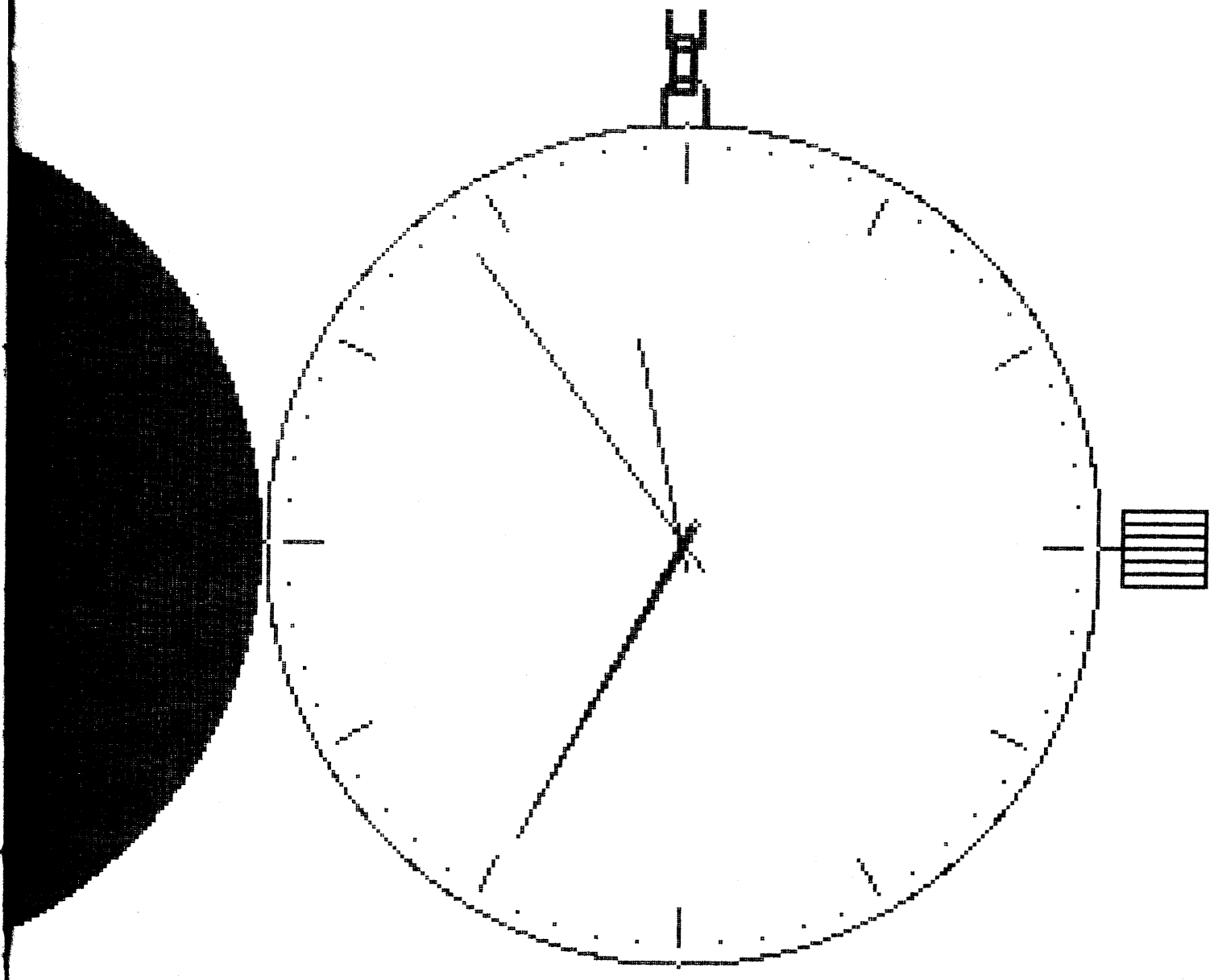
DAITIME

```
1     REM ************************************************
2     REM ***          D A I     T I M E               ***
3     REM ***                                          ***
4     REM ***  Programme concu et realise par:         ***
5     REM ***                        Thomas  BONDUELLE. ***
6     REM ***                        =================  ***
7     REM ***  Sur une idee originale de:              ***
8     REM ***                        Olivier Capon.    ***
9     REM ************************************************
10    CLEAR 1000:MODE 0:MODE 6:COLORG 0 0 0 0
20    DIM XP!(60.0),YP!(60.0),X!(60.0),Y!(60.0)
30    I=0:Q=1:K=10:XM!=XMAX/2.0:YM!=YMAX/2.0
40    EM!=PI/1800.0:EH!=EM!/12.0:ST!=PI/30.0:PRS=8:PRM=6:PRH=6
50    RS=YMAX/3:RM=RS-5:RH=RM-30:R1=RS+2:R2=R1+9:R3=R2+5
53    REM
54    REM
55    REM Trace de la montre:
56    REM
57    REM
60    L=R3:KL=23:GOSUB 600:L=R3-1:KL=21:GOSUB 600:KL=22
70    FOR A!=0.0 TO 2.0*PI STEP ST!:S!=SIN(A!):C!=COS(A!)
72    XP!(I)=XM!-PRS*S!:YP!(I)=YM!-PRS*C!:X!(I)=XM!+RS*S!:Y!(I)=YM!+RS*C!
74    IF I MOD 5.0=0.0 THEN DRAW XM!+R1*S!,YM!+R1*C! XM!+R2*S!,YM!+R2*C! KL:GOTO
 78
76    DOT XM!+R2*S!,YM!+R2*C! KL
78    I=I+1:NEXT
80    KL=22:XM3!=XM!+R3+10.0:GOSUB 700:GOSUB 800
90    XM2!=XM!-2.0*R3-1.0:COLORG 0 K K 0:KL=23
100   FOR I=0 TO 94:X!=SQR(R3*R3-I*I)
102   DRAW 0,YM!+I XM2!+X!,YM!+I KL
104   DRAW 0,YM!-I XM2!+X!,YM!-I KL:NEXT:I=0
110   GOSUB 650:KL=22
111   REM
112   REM Ouverture de la montre:
113   REM
114   REM
115   IF GETC<>0 THEN 115
116   IF GETC=0 THEN 116
117   GOSUB 240
120   L=R3-1:KL=16:GOSUB 600:KL=22:XM3!=XM!+R3+10.0:GOSUB 700
130   KL=23:GOSUB 800
140   FOR A!=0.0 TO 2.0*PI STEP ST!
142   IF I MOD 5.0=0.0 THEN DRAW XM!+R1*SIN(A!),YM!+R1*COS(A!) XM!+R2*SIN(A!),YM
!+R2*COS(A!) KL:GOTO 146
144   DOT XM!+R2*SIN(A!),YM!+R2*COS(A!) KL
146   I=I+1:NEXT
147   REM
148   REM Demarrage de la montre:
149   REM
150   IF GETC<>0 THEN 150
160   IF GETC<>18 THEN 160
170   FILL XM3!,YM!-10 XM3!+20,YM!+10 0
180   XM3!=XM!+R3:KL=23:GOSUB 700
195   REM
196   REM
197   REM Boucle principale:
198   REM
199   REM
200   IF GETC=19.0 THEN 400
210   IS=(IS+1) MOD 60:AM!=AM!+EM!:AH!=AH!+EH!
220   GOSUB 230:GOTO 200
```

```
225   REM
226   REM
227   REM SBR:Trace des nouvelles aiguilles
228   REM *                    et effacage des anciennes:
229   REM
230   KL=17+2*Q
240   X0!=XP!(IS):Y0!=YP!(IS):X!=X!(IS):Y!=Y!(IS)
250   C!=COS(AM!):S!=SIN(AM!):X60!=XM!-PRM*S!:Y60!=YM!-PRM*C!:Y6!=YM!+RM*C!:X6!=
XM!+RM*S!
260   C!=COS(AH!):S!=SIN(AH!):XH0!=XM!-PRH*S!:YH0!=YM!-PRH*C!:YH!=YM!+RH*C!:XH!=
XM!+RH*S!
270   DRAW X0!,Y0! X!,Y! KL
280   DRAW X60!+1,Y60! X6!,Y6! KL:DRAW X60!-1,Y60! X6!,Y6! KL
285   DRAW X60!,Y60!+1 X6!,Y6! KL:DRAW X60!,Y60!-1 X6!,Y6! KL
290   DRAW XH0!,YH0! XH!,YH! KL
300   COLORG 0 K*(1-Q) K*Q K
310   KL=18-2*Q
320   DRAW XA0!,YA0! XA!,YA! KL
330   DRAW XA60!+1,YA60! XA6!,YA6! KL:DRAW XA60!-1,YA60! XA6!,YA6! KL
335   DRAW XA60!,YA60!+1 XA6!,YA6! KL:DRAW XA60!,YA60!-1 XA6!,YA6! KL
340   DRAW XAH0!,YAH0! XAH!,YAH! KL
350   XA!=X!:YA!=Y!:XA0!=X0!:YA0!=Y0!:XAH!=XH!:YAH!=YH!:XAH0!=XH0!:YAH0!=YH0!
360   XA6!=X6!:YA6!=Y6!:XA60!=X60!:YA60!=Y60!:Q=1-Q
370   RETURN
395   REM
396   REM
397   REM Changement de l'heure:
398   REM
399   REM
400   FILL XM3!,YM!-10 XM3!+20,YM!+10 0
410   XM3!=XM!+R3+5.0:KL=23:GOSUB 700
420   I=GETC:IF I<16 OR I>19 THEN 420
430   IF I=18.0 THEN FILL XM3!,YM!-10 XM3!+20,YM!+10 0:XM3!=XM!+R3:KL=23:GOSUB 7
00:GOTO 210
440   IF I=16.0 THEN AM!=AM!+60.0*EM!:AH!=AH!+60.0*EH!:GOSUB 230:GOTO 420
450   IF I=17.0 THEN AM!=AM!-60.0*EM!:AH!=AH!-60.0*EH!:GOSUB 230:GOTO 420
460   FILL XM3!,YM!-10 XM3!+20,YM!+10 0
470   XM3!=XM!+R3+10.0:KL=23:GOSUB 700
480   I=GETC:IF I<16 OR I>19 THEN 480
490   IF I=18.0 THEN FILL XM3!,YM!-10 XM3!+20,YM!+10 0:XM3!=XM!+R3+5.0:KL=23:GOS
UB 700:GOTO 420
500   IF I=16.0 THEN AM!=AM!+67.0*60.0*EM!:AH!=AH!+67.0*60.0*EH!:GOSUB 230:GOTO
480
510   IF I=17.0 THEN AM!=AM!-67.0*60.0*EM!:AH!=AH!-67.0*60.0*EH!:GOSUB 230:GOTO
480
520   FILL XM!+R3,YM!-10 XM3!+20,YM!+10 0:DRAW 300,0 318,0 23
530   FOR I=300 TO 318 STEP 3:DRAW I,0 I,20 23:NEXT
540   DRAW 300,20 318,20 23:DRAW 309,20 309,40 23
550   FOR I=0 TO 2000:IF GETC=18 THEN FILL 300,0 320,50 0:GOTO 470
560   NEXT
570   GOTO 570
595   REM
596   REM
597   REM SBR:Trace du grand cercle:
598   REM
599   REM
600   FOR I=0 TO L:R=SQR(L*L-I*I)
610   DRAW XM!-R,YM!+I XM!+R,YM!+I KL
620   DRAW XM!-R,YM!-I XM!+R,YM!-I KL:NEXT
630   I=0:RETURN
```

```
645   REM
646   REM
647   REM SBR:Choix d'une heure:
648   REM
649   REM
650   P=RND(3600.0*12.0)
660   NH=P/3600:NM=(P-NH*3600)/60:NS=P-3600*NH-60*NM
670   IS=NS:AM!=(P-3600.0*NH)*PI/1800.0:AH!=P*PI/3600.0/6.0
680   XA60!=100.0:YA60!=100.0:XA6!=100.0:YA6!=100.0
690   RETURN
695   REM
696   REM
697   REM SBR:Trace du remontoir:
698   REM
699   REM
700   FOR I=0 TO 10 STEP 3:DRAW XM3!,YM!+I XM3!+20,YM!+I KL
710   DRAW XM3!,YM!-I XM3!+20,YM!-I KL:NEXT
720   DRAW XM3!+20,YM!-9 XM3!+20,YM!+9 KL:DRAW XM!+R3,YM! XM3!,YM! KL
730   DRAW XM3!,YM!-9 XM3!,YM!+9 KL
740   I=0:RETURN
795   REM
796   REM
797   REM SBR:Trace de la chainette:
798   REM
799   REM
800   DOT 162,237 KL:DOT 163,238 KL
810   DOT 170,238 KL:DOT 171,237 KL
820   FILL 161,228 162,236 KL:FILL 172,228 171,236 KL
830   FILL 163,235 164,248 KL:FILL 168,235 169,248 KL
840   FILL 164,235 168,236 KL:FILL 164,248 168,249 KL
850   FILL 164,245 169,246 KL:FILL 162,246 163,YMAX KL
860   FILL 170,246 171,YMAX KL:FILL 164,239 169,238 KL
870   I=0:RETURN
880   REM
881   REM
882   REM
883   REM
884   REM
885   REM *         Au revoir et merci....
886   REM
887   REM
888   REM
889   REM *                    Copyright T.Bonduelle 1982.
890   REM
891   REM
892   REM
893   REM
```





SCROLLED & UNSCROLLED

```
5     MODE 0:PRINT CHR$(12);
10    REM SCROLLED  & UNSCROLLED MODE 0
20    REM by P.H.DAL
30    REM A TOTAL 'LIST' RESET THE ORIGINAL SCREEN VALUES
40    REM DO NOT USE COLORT UNLESS IN FULL SCROLL (23)
50    REM I CAN HARDLY IMAGINE YOU WILL NOT TRY (red.)
100   REM (23)
110   REM . !
120   REM . !
130   REM . !--------UNSCROLLED
140   REM . !
150   REM . !
160   REM X
170   REM . !
180   REM . !
190   REM . !--------SCROLLED
200   REM . !
210   REM . !
300   INPUT " X VALUE ";X
310   GOSUB 1000
320   LIST -1020:GOTO 320
1000  IF X<1.0 OR X>23.0 THEN RETURN
1010  X=#B3E5+X*#86:POKE #8A,X MOD 256
1020  POKE #8B,X SHR 8:RETURN
```

DAIpc MEMORY MAP V4.4

```
**************************************
*                                    *
*        DAI pc    MEMORY MAP        *
*                                    *
**************************************
```

Basic V1.0/V1.1 Revision 4.4 Date: 28.4.82

INTERRUPT VECTOR ROUTINES: 0000 - 003F

```
0000-07          Interrupt vector routine 0:
                    Used by Utility (LOOK).
0008-0F          Interrupt vector routine 1:
                    Used by Utility and encoding Basic.
                    RST 1 + data: Switch to ROM-bank 3.
0010-17          Interrupt vector routine 2:
                    Used by stack interrupt.
0018-1F          Interrupt vector routine 3:
                    Used by sound interrupt.
0020-27          Interrupt vector routine 4:
                    Used for math. routines.
                    RST 4 + data: Switch to ROM-bank 1.
0028-2F          Interrupt vector routine 5:
                    Used for screen handling routines.
                    RST 5 + data: Switch to ROM-bank 2.
0030-37          Interrupt vector routine 6:
                    Used for keyboard service routines.
0038-3F          Interrupt vector routine 7:
                    Used to flash the cursor.

Interrupt vector    00  NOP
routines:           E5  PUSH H
                    2A  LHLD:
                    ..  ) vector address location
                    ..  ) see (#0062-#0071).
                    E9  PCHL
                    00  NOP
                    00  NOP
```

UTILITY WORK AREA: 0040 - 0061

```
0040     !POROM:  ) Memory of last outputs to output ports.
         !POR1M:  ) Duplicate of (#FD06).
          POROM:  )
0041/42  RSWK1:   Save (psw) during ROM bank switching
0043/44  RSWK2:   Save (H,L) during ROM bank switching
0045/46           Spare.
```

Used by LOOK:

0047		Store EI/DI instructions after using LOOK the first time (No clear occurs).
0048/49		High address trace window.
004A/4B		Low address trace window.
004C		Store EI/DI when LOOK is used.
004D-4F		Store current instruction of traced program.
0050		Flag for Look initialisation: #FF: init. Look, else: #00.
0051/52	IADR:	I: Address current instruction.
0053	AFSAV:	A: Contents A after execution of I.
0054		F: Idem status flags.
0055	BCSAV:	B: Idem B register.
0056		C: Idem C register.
0057	DESAV:	D: Idem D register.
0058		E: Idem E register.
0059	HLSAV:	H: Idem H register.
005A		L: Idem L register.
005B/5C	SPSAV:	S: Idem stackpointer.
005D/5E	PCSAV:	P: Address next instruction to be executed.
005F	TICIM:	M: Current interrupt mask. Duplicate of (#FFF8)
0060		T: Value TICC control word. (#FC after Z2).
0061		G: Value GIC control word. (#1B after Z2).

INTERRUPT VECTOR ADDRESSES: 0062 - 0071

0062/63	I0USA:	Vector address RST 0: set by UT (Z2): 3#EB5D.
0064/65	I1USA:	Vector address RST 1: utility/encode: #C70E.
0066/67	I2USA:	Vector address RST 2: stack interrupt: #D9E2.
0068/69	I3USA:	Vector address RST 3: sound interrupt: #D755.
006A/6B	I4USA:	Vector address RST 4: math. restart: #C6C0.
006C/6D	I5USA:	Vector address RST 5: screen restart: #C6FD.
006E/6F	I6USA:	Vector address RST 6: keyb. int. serv: #D578.
0070/71	I7USA:	Vector address RST 7: clock interrupt: #D9A9.
		By changing the vector addresses, other interrupt routines can be used.

SCREEN VARIABLES: 0072 - 00CF

Character mode variables:

0072/73	CURSOR:	Cursor position address.
0074	CURTY:	Cursor type: #00: cursor flashes in colour. #01: cursor alternates between actual character and contents #0075.
0075	CURIN:	Cursor information: If type = 0: Mask which is EXOR'ed with the colour byte for that character to flash it. If type = 1: Cursor alterates between actual character and this information.

```
0076/77  CURSV:  Contents screen RAM location indicated by
                 the cursor:
                    #0076 contains the colour byte,
                    #0077 contains the data.
0078/79  LNSTR:  Address line mode byte of currently used
                 line of the screen RAM.
007A     LNEND:  Lsbyte of end of cursor line.
                 Used to check if end of line is reached.
007B     LCONT:  Number of extended lines.
007C     COLMT:  ) colours for colour    #80+X3
007D             ) registers COLORT      #90+X2
007E             )                       #A0+X1
007F             )                       #B0+X0

         Variables set to describe the current state of
         the screen:

0080/81  SCREEN: Points to first byte of screen RAM (#BFFF).
0082/83  SCTOP:  Points after header (#BFEF).
0084/85  FFB:    First free byte in this mode.
0086/87  GRR:    Points to top of rolled area. Contains the
                 line mode byte of the line where split mode starts.
0088/89  GRE:    Points after end of graphics area.
008A/8B  CHS:    Points to start of character area.
008C/8D  GAE:    Unsplit: End archive area.
         CHE:    Split: After end of character area.
008E/8F  SCE:    End of screen (after trailer).
0090/91  GTE:    End area used splitting mode.
0092/93  GAS:    Unsplit: start archive area.
         GTS:    Split: start temporary save area.
0094/95  GRC:    Number of blobs horizontally in mode.
0096     GRL:    Number of lines of graphics in mode.
0097     GAL:    Number saved lines of graphics.
0098     GXB:    Number of bytes/line this mode.
0099/9A  GREQ:   Previous end of graphics.
009B/9C  CHSO:   Previous start characters:
                 Was split: previous mode byte of 1st text line.
                 Was graphics: Previous last COLORT-byte.
009D     SMODE:  Current screen mode (updated after mode
                 changed):
                    #00 mode 1          #08 mode 5
                    #01 mode 1A         #09 mode 5A
                    #02 mode 2          #0A mode 6
                    #03 mode 2A         #0B mode 6A
                    #04 mode 3
                    #05 mode 3A         #10 during init.
                    #06 mode 4
                    #07 mode 4A         #FF mode 0
                       bits 4-7 are ignored;
                       #0C,0D,0E,0F are inhibited.

         Graphics mode variables (From #00A2-#00B5 also
         used by the EDIT mode):

009E     COLMG:  ) colours for colour #80+X3
009F             ) registers COLORG   #90+X2
00A0             )                    #A0+X1
00A1             )                    #B0+X0
00A2     SCVR:
00A3-AA  SCXBUF: Buffer used to hold contents of an 8 bit
                 field during 16 colour updates.
00AB     SBGOU:  Flags when colour is being carried out to
                 next field.
00AC     SBGOC:  Colour being carried out.
00AD-B4  COLS:   Buffer for impossible requests.
```

!Edit variables:

00A2/A3	!EBUFR:	Address start EDIT buffer.
00A4/A5	!EBUFN:	Address end of text in EDIT buffer.
00A6/A7	!EBUFS:	End available space in EDIT buffer.
00A8	!EWINX:	Offset of left side of window.
00A9/AA	!EWINY:	Offset of top of window from start buffer.
00AB	!ECURX:	X-offset of cursor in document (current cursor position in text line).
00AC/AD	!ECURY:	Y-offset of cursor in document (count of current cursor line).
00AE/AF	!CURPT:	Pointer to cursor position in buffer.
00B0/B1	!CURLS:	Pointer to line mode byte of cursor line on screen.
00B2/B3	!CURLB:	Pointer to line mode byte of cursor line in buffer.
00B4/B5	!TABTP:	Address tab position table.

Line drawing variables:

00B5/B6	DELTA:	Amount to add into count.
00B7/B8	RT:	Count.
00B9/BA	COR:	Adjustments for long sectors.
00BB/BC	SECT:	Lower of 2 possible sector lengths.
00BD	SECTC:	Number of sectors.
00BE	TRIM:	Amount to trim off last sector.
00BF	DIRN1:	Set if Y-direction is negative.
00C0	DIRN2:	Set if swap X,Y directions.
00C1	ANIM:	Set if animate in 4 colour mode.
00C2/C3	FCOLR:	Details of colour required.
00C4/C5	ASMKRM:	Address memory management routine (#CA01). Checks available RAM space.
00C6/C7	AESTOP:	Address emergency stop routine (#CA25). Return-routine for 'Out of space for mode'.
00C8-CF		Spare.

MATH. WORKING AREA: 00D0 - 00FF

00D0/D1		Pointer to table with error routines (#C7F2).
00D2/D3		Pointer to input routine (#DDE0).
00D4	MVECA:	Math. chip flag: offset of start HW/SW vector; (offset for RST 4 restart routines): #00 No math. chip. #7B math. chip present.
00D5-D8	FPAC:	Arithmetic accumulator.
00D9-FF:		Used as scratch pad memory for math. package. Used in single and double byte configurations.
00DE	EXPDF:	
00DF-E2		Also used for data save during stack operations.
00E3	XN:	Length output string in #00E4-F0.
00E4		Sign output string.
00E4-F0		Used for output conversion.
00E7-EA		
00EB-EE		
00EF-F2	FTWRK:	
00F1		Digit count in output conversions.

DAIpc MEMORY MAP V4.4

BASIC VARIABLES: 0100 - 02EB

User state:

Following are saved by soft break: (SFRAME = SYSTOP - SYSBOT)

0100/01 SYSBOT:) Start of current line. Points to first
CURRNT:) byte of line number.
0102/03 BRKPT: Start of current command.
0104/05 LOPVAR: Points to current loop variable. Points to position of variable in symbol table.
#00 if no running loop.
0106 LSTPF: Flag for integer/fpt loop and implicit/explicit step.
bit 0: 0 = implicit step.
1 = expicit step.
bit 7: 0 = FPT loop variable.
1 = INT loop variable.
0107-0A LSTEP: Step value if explicit.
010B-0E LCOUNT: Loop iteration count.
010F-10 LOPPT: Pointer to start address loop.
0111/12 LOPLN: Pointer to start loop line.
0113/14 STKGOS: Stack level at last GOSUB.
#00 if no active call.
0115 SYSTOP:)
(STRFL:) Trace/step flag together)
TRAFL:) Trace flag (#FF when set).
0116 STEPF: Step flag (#FF when set).
0117 RDIPF: Flag set while running input (set: #FF).
0118 RUNF: Flag set while running program.
(Previous 2 bytes must be consecutive)

Runtime scratch area:

0119/1A GSNWK: Scratch area for GOSUB/NEXT (2 bytes).
Points to destination address last GOSUB.
LISW1: Startaddress of listed area.
COLWK: Scratch area for SCOLG, SCOLT (4 bytes).
Contains last selected COLORT/COLORG values.
011B/1C LISW2: End address listed area.

Save area for restart on error:

011D/1E ERSSP: Stack pointer.
011F-21
0122 ERSFL: Set if encoding a stored line (set: #01).

Data/read variables:

0123 DATAC: Offset of next character to encode.
0124/25 DATAP: Pointer to current data line.
!DATAQ: Pointer after current data line.
0126 CONFL: Set if there is a suspended program (set: #01).
0127/28 STACK: Current base stack level.

Scratch location for expression/function evaluation.

0129-2C WORKE: Scratch area. Contains also the argument A of the last software random RND(A).

Random number kernel:

012D-30 RNUM: Random number kernel.
!RNDLY: Random number delay count (1 byte).

Output switching:

0131 OTSW: #00 output to screen + RS232.
#01 output to screen only.
#02 output to edit buffer.
#03 output via DOUTC.

Encoding input source switching:

0132/33 EFEPT: Encoding input pointer. Points to start-address of Basic-line just being encoded.
0134 EFECT: Encoded input count. Counts length of line.
0135 EFSW: Encoded input switching:
#00 Input from keyboard/DINC.
#01 Input from string.
#02 From edit buffer to program area.

Variables used during expression encoding (could overlap with runtime variables):

0136 TYPE: Type of latest expression or item:
#00 FPT
#01 INT
#02 STR
0137 RGTOP: Latest priority operator:

#00	no operation	#6A	IOR
#38	AND	#6C	IXOR
#39	OR	#8D	SHL
#50	>=	#8E	SHR
#51	>	#A0	+
#52	<>	#A1	-
#53	<=	#C2	/
#54	<	#C3	*
#55	=	#CF	MOD
#69	IAND	#E4	^

0138 OLDOP: Old priority operator.
0139/3A HOPPT: Pointer to place in encoded input buffer for next operator.
013B/3C RGTPT: Pointer to place in encoded input buffer of operand latest operator.

Mask to select cassette 1 or 2:

013D CASSL: #10 Cassette 1 activated.
#20 Cassette 2 activated.

Encoded input buffer:

013E-BD EBUF: 128 bytes buffer. Also used by utility.

Interrupt handler variables:

01BE/BF TIMER: Timer location. Also used in WAIT TIME.
01C0 CTIMR: Cursor clock. Used for cursor flashing.
CTIMV: #0F: Flash time in 20 ms units.
When #00, cursor flashes.
01C1 KBXCT: Extend keyboard scan time counter. When #00, keyboard scan will be performed.
KBXCK: #02: Keyboard scan time (16 ms units). Also used by RAND routine.

Sound control block storage:

01C2-CF:		Sound control block 0. SCBL: Length of a sound block (14 bytes).
01C2	SCB0:	Duration count of present volume.
01C3/C4		Pointer to present envelope volume/duration in envelope table.
01C5/C6		Pointer to start envelope table.
01C7		Sound-volume *8.
01C8		Volume, calculated from sound-volume and present envelope volume.
01C9		Counter for tremolo.
01CA		Final volume, calculated from volume and tremolo fluctuations.
01CB		Glissando flag: #00 Endperiod reached. #02 Endperiod not reached.
01CC/CD		Present period.
01CE/CF		Final period (glissando).
01D0-DB	SCB1:	Sound control block 1 (see SCB0).
01DE-EB	SCB2:	Sound control block 2 (idem).
01EC-F4	NCB:	Noise control block. NCBL: Length of noise block (9 bytes). The noise control block is identical to the sound control block, but without period-values and tremolo.

Envelope storage:

01F5--0274	ENVST:	Envelope storage (128 bytes). ENVLL: #40: Number of bytes/envelope NUMENV: #02: Number of envelopes. Two envelope tables of each 64 bytes: #01F5-#0234 and #0235-#0274.
0275--28E	IMPTAB:	Implicit type table. #0275 A #027C H #0283 O #028A V #0276 B #027D I #0284 P #028B W #0277 C #027E J #0285 Q #028C X #0278 D #027F K #0286 R #028D Y #0279 E #0280 L #0287 S #028E Z #027A F #0281 M #0288 T #027B G #0282 N #0289 U
028F	IMPTYP:	Default number type. Selected by IMP command. #00 FPT #10 INT #20 STR
0290	REQTYP:	Required number type for present operation. #00 FPT #10 INT #20 STR #40 Array without arguments

Spare variable space:

0291/92	DATAQ:	Pointer to begin current data line.
0293	RNDLY:	
0294	POROM:	Duplicate of (#FD04).
0295	POR1M:	Duplicate of (#FD05).
0296	INSW:	Input switching: If #00, input from keyboard. If <>#00, input from DINC.
0297-9A		Spare.

Heap/text buffer/symbol table pointers:

```
029B/9C  HEAP:   Start address of HEAP.
029D/9E  HSIZE:  Size of HEAP.
                    HSIZD: #100: Default size.
029F/A0  TXTBGN: Start address of text buffer.
02A1/A2  TXTUSE: End text buffer and.
         STBBGN: Start symbol table.
02A3/A4  STBUSE: End of symbol table.
02A5/A6  SCRBOT: Bottom screen RAM area (48K):
                    mode 0:       #B350
                    mode 1/2(A): #B7A0
                    mode 3/4(A): #A65C
                    mode 5/6(A): #63B8
```

Keyboard variables + constants:

```
02A7/A8  KBTPT:  Pointer to table with ASCII-codes.
02A9-B0  MAP1:   Latest scan of keys (key-codes).
                 (row 0 in #02A9, row 7 in #02B0)
   02AF  RPLOC:  Byte containing REPT key.
                    RPMSK: #20: Rept key bit.
                    BRSEL: #40: Column select mask for BREAK.
                    BRMSK: #40: Break key bit.
   02B0  SHLOC:  Byte containing SHIFT.
                    SHMSK: #40: Shift key bit.
02B1-B8  MAP2:   Previous scanning of keyboard.
02B9     KNSCAN: Set to scan for BREAK only. When (#02B9)
                 is #FF: scan for BREAK only.
02BA-BD  KLIND:  4 byte circular buffer to store the ASCII
                 values for keys pressed.
                    KBLEN/KEYL: #04: length rollover buffer.
02BE/BF  KLIIN:  Next position for input to KLIND.
02C0/C1  KLIOU:  Next position for output from KLIND.
02C2     RPCNT:  Count for REPT. #01 when REPT is not
                 pressed. Else it is used as timer for the
                 repeat function.
02C3     SHLK:   Set to #FF when CTRL is pressed to
                 invert SHIFT. Else #00. Used to
                 calculate the offset for the ASCII code
                 table.
02C4     KBRFL:  Break flag. #FF indicates BREAK pressed
                 (Only if suspended program). When BREAK is pressed,
                 #02C4 counts from 00 to #0F before stopping
                 the program.
```

Data/cassette switching vectors:

Copy of ROM (#D7A4 - #D7CA) for cassette and RS232.
Can be loaded with other I/O vectors.

```
02C5-EB  IOVEC:  02C5  WOPEN:  C3 B8 D2  JMP: D2B8
                 02C8  WBLK:   C3 F1 D2  JMP: D2F1
                 02CB  WCLOSE: C3 27 D4  JMP: D427
                 02CE  ROPEN:  C3 25 D3  JMP: D325
                 02D1  RBLK:   C3 40 D3  JMP: D340
                 02D4  RCLOSE: C3 45 D4  JMP: D445
                 02D7  MBLK:   C3 A2 D3  JMP: D3A2
                 02DA  RESET:  C9 00 00  RET
                 02DD  DOUTC:  C9 00 00  RET
                 02E0  DINC:   C3 B4 DD  JMP: DDB4
                 02E3          C9 00 00  RET
                 02E6  TAPSL:  24 24     Tape speed leader.
                 02E8  TAPSD:  24 3C     Tape speed data.
                 02EA  TAPST:  24 18     Tape speed trailer.
```

DAIpc MEMORY MAP V4.4

```
HEAP, PROGRAM AREA, SCREEN RAM: 02EC - BFFF
===========================================

02EC-   (RAM:      HEAP (Strings + arrays)  - See (#029B/9C).
 -BFFF  (VAREND:   Program (compiled Basic) - See (#029F/A0).
        (VARLAST:  Symbol table             - See (#02A1/A2).
                   Not used RAM             - See (#02A3/A4).
                   Screen RAM               - See (#02A5/A6).


ROM AND CPU AREA: C000 - F8FF
=============================

C000-              24K ROM:
  -EFFF                #C000-#DFFF: 8K non-switched ROM.
          VECA:        #E000-#EFFF: 4 banks of each 4K ROM.
                                    (switchable).


F000-              Can be used for ROM extension (reading only).
  -F7FF            Is already completely used by Memocom MDCR-D.


F800-              Microcomputer stack.
  -F8FF            Incl. vector for MDS jump instructions.
                      #F800  SRBOT    Bottom of stack RAM.
                      #F900  STTOP    Top of stack RAM.



I/O  DEVICE ADDRESSES: F900 - FFFF
==================================

F900-              Spare I/O device addresses.
  -FAFF            (Not wired on pC board).


          MATH. CHIP AMD 9511: FB00 - FBFF
          ================================

FB00      MTHAD:   ) Data math.chip.
          MDATA:   )
FB02      MCOMD:   ) Command + status.
          MSTATUS:)

          AMD9511 operator and status bytes:
             ODADD:  #2C Int addition     OFADD:  #10 Fpt addition
             ODSUB:  #2D Int subtract     OFSUB:  #11 Fpt subtract
             ODMUL:  #2E Int multiply     OFMUL:  #12 Fpt multiply
             ODDIV:  #2F Int division     OFDIV:  #13 Fpt division
             OSQRT:  #01 Square root      OFIXD:  #1E Fix
             OSIN:   #02 Sine             OFLTD:  #1C Float
             OCOS:   #03 Cosine           OCHSD:  #34 Change sign int
             OTAN:   #04 Tangent          OCHSF:  #15 Change sign fpt
             OASIN:  #05 Arc sine         OPTOD:  #37 Push int/fpt
             OACOS:  #06 Arc cosine       OPOPD:  #38 Pop int/fpt
             OATAN:  #07 Arc tangent
             OLOG:   #08 Log base 10
             OLN:    #09 Log base e       MBUSY:  #80 Busy status bit
             OEXP:   #0A Expotential      MERRB:  #1E All error bits
             OPWR:   #0B X^Y              MZERO:  #20 Top of stack
```

PROGRAMMABLE INTERVAL TIMER 8253: FC00 - FCFF

Used for sound generator. 3 independent 16 bits down counters with programmable counter modes.

```
FC00/01  SNDAD:  )
         SND0:   ) Counter 0 (oscillator channel 0).
        !PDLCH:  Used as counter for paddle operations.
FC02/03  SND1:   Counter 1 (oscillator channel 1).
FC04/05  SND2:   Counter 2 (oscillator channel 2).
                 (16 bit data; LSB first)
FC06     SNDC:   Command 8253. To be loaded prior to freq.
                 selection with resp. #36, #76 and #B6.
                 Command word format:
                 bit 0    : 0   binairy counter 16 digits.
                            1   BCD counter (4 decades).
                     3,2,1: 000 mode 0: Int. on end count.
                            001 mode 1: Progr. one shot.
                            x10 mode 2: Rate generator.
                            x11 mode 3: Sq.wave rate gen.
                            100 mode 4: SW trig. strobe.
                            101 mode 5: HW trig. strobe.
                     5,4  : 00  Counter latch operation.
                            01  Read/load MSB only.
                            10  Read/load LSB only.
                            11  Read/load LSB first, then
                                MSB.
                     7,6  : 00  Select counter 0.
                            01  Select counter 1.
                            10  Select counter 2.
                            11  Illegal.
                 Several control words:
                    COFIX: #00  Fix count on channel 0.
                    COM0:  #30  Chan.0, mode 0, 2 byte op.
                    COM1:  #32  Chan.0, mode 1, 2 byte op.
                    COM3:  #36  Chan.0, mode 3, 2 byte op.
                    C1M3:  #76  Chan.1, mode 3, 2 byte op.
                    C2M3:  #B6  Chan.2, mode 3, 2 byte op.
```

DISCRETE I/O DEVICE ADDRESSES: FD00 - FDFF

```
FD00     PORI:   IN (1)  bit 0: -
                         1: -
                         2: PIPGE: Page signal
                         3: PIDTR: Serial output ready
                         4: PIBU1: Button on paddle 1
                                   (1 = closed)
                         5: PIBU2: Button on paddle 2
                                   (1 = closed)
                         6: PIRPI: Random data
                         7: PICAI: Cassette input data

FD01     PDLST:  OUT (3) Single pulse used to trigger
                         paddle timer circuit.
```

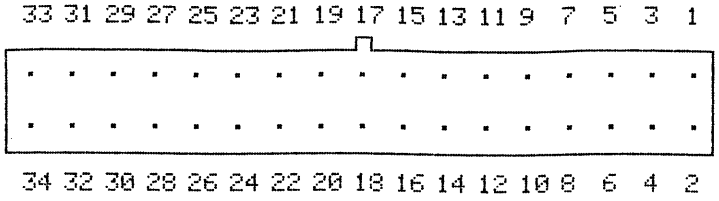
```
FD04      PORO:   OUT (2) bit 0 - 3: volume osc. channel 0
                              4 - 7: volume osc. channel 1
FD05      POR1:   OUT (2) bit 0 - 3: volume osc. channel 2
                              4 - 7: volume random noise
                                     generator.
FD06      PORO:   OUT (3) bit 0: POCAS: Cassette data output
                          1,2: PDLMSK: Paddle select
                            3: POPNA:  Paddle enable
                            4: POCM1:  Cassette 1 motor
                                       control.(0 = run)
                            5: POCM2:  Cassette 2 motor
                                       control.(0 = run)
                          7,6:         ROM bank switching:
                                          00  bank 0
                                          01  bank 1
                                          10  bank 2
                                          11  bank 3
```

PROGR. PERIPHERAL INTERFACE 8255 : FE00 - FEFF

Used for DCE-bus (GIC Controller).

```
FE00      GIC:    (1) I/O port A
FE01              (1) I/O port B
FE02              (1) I/O port C
FE03              (6) Command word 8255:
                  Contr.  PA   PCH  PCL  PB      (mode 0)
                   #80    out  out  out  out      RWMOP
                   #81    out  out  in   out
                   #82    out  out  out  in
                   #83    out  out  in   in
                   #88    out  in   out  out
                   #89    out  in   in   out
                   #8A    out  in   out  in
                   #8B    out  in   in   in
                   #90    in   out  out  out      RWMIP
                   #91    in   out  in   out
                   #92    in   out  out  in
                   #93    in   out  in   in
                   #98    in   in   out  out
                   #99    in   in   in   out
                   #9A    in   in   out  in
                   #9B    in   in   in   in
```

DAI DCE-BUS CONFIGURATION BACK VIEW



TICC: TIMER + INTERRUPT CONTROLLER 5501: FF00-FFFF

FFF0 (4) Serial input buffer. Contains the last character received on the RS232 interface.

FFF1 (4) Keyboard input port. Bottom 7 bits are data input from the keyboard. Bit 7 is the IN7 line from the DCE-bus and is attached to the page-blanking signal for the TV. Every 20 ms. an impulse is present.

FFF2 (5) Interrupt address register:
bits 5,4,3: Number of pending interrupt.
7,6 :)
2,1,0:) always '1'

FFF3 (4) Status register:
bit 0: Frame error. Set by a BREAK on the RS232 input.
1: Overrun error. Set if a character has been received but not taken by the CPU.
2: Serial input. Set if no data is received.
3: Receive buffer loaded. Set if a character has been received.
4: Transmit buffer empty. Set if RS232 output is ready to accept another character.
5: Interrupt pending. Set if one or more of the enabled interrupts has occured.
6: Full bit detected. Set if the first data bit of an incoming character has been detected.
7: Start bit detected. Set if the start bit of an incoming character has been detected.

FFF4 (5) Command register:
bit 0: TICC reset.
1: Send Break. If set, the serial output is high impedance.
2: Interrupt 7 select. A '1' selects IN7 of the DCE-bus, a '0' selects Timer 5.
3: Interrupt acknowledge enable. A '1' enables TICC to accept a INTA signal from the CPU.
4 - 7: Always 0.

FFF5 (6) Communications rate register:
bit 0: 110 baud
1: 150 baud
2: 300 baud
3: 1200 baud
4: 2400 baud
5: 4800 baud
6: 9600 baud
7: 1 - one stop bit
0 - two stop bits

```
FFF6      (6) Serial output buffer. Write byte to this
              location to send it on the RS232 output.
              Use only when #FFF3-bit 4 is high.
FFF7      (7) Keyboard output port. Data output to scan
              keyboard.
FFF8      (5) Interrupt mask register:
              bit 0: timer 1 has expired (UTIM).
                  1: timer 2 has expired.
                  2: External interrupt (STKIM).
                  3: Timer 3 has expired (SNDIM).
                  4: Serial receiver loaded.
                  5: Serial transmitter empty.
                  6: Timer 4 has expired (KBIM).
                  7: Timer 5 has expired or IN7 (CLKIM).
              (react only on low-high transition)
FFF9      (5) UTIAD:  Timer 1 address (UT).
FFFA      (5)         Timer 2 address.
FFFB      (5) SNDIAD: Timer 3 address (sound).
FFFC      (5) KBIAD:  Timer 4 address (keyboard).
FFFD      (5)         Timer 5 address.
FFFE          not used.
FFFF          not used.
```

NOTES:
(1) Read and write allowed.
(2) Reading allowed. Writing too, but may be overwritten by BASIC program.
(3) No writing allowed.
(4) Reading allowed, writing not.
(5) Should not be accessed.
(6) Writing allowed, reading not.
(7) Reading not allowed, writing is harmless but useless; keyboard scanner will overwrite it.

REMARKS:

ADDRESSES FB00 - FFFF:

The 2 highest bytes of the address are used for the chip select signal CS of the peripheral equipment 8253, 8255, 5501 etc. The lowest byte is used to address the several registers of the peripheral. The 2nd LSB does not have any value. So addresses in this range can be read as FBx0 - FFxF, in which x is a don't care.

For additional information and comments, please contact:

Jan Boerrigter
Fabritiusstraat 15
6174 RG Sweikhuizen, NL.
tel. 04493-2093

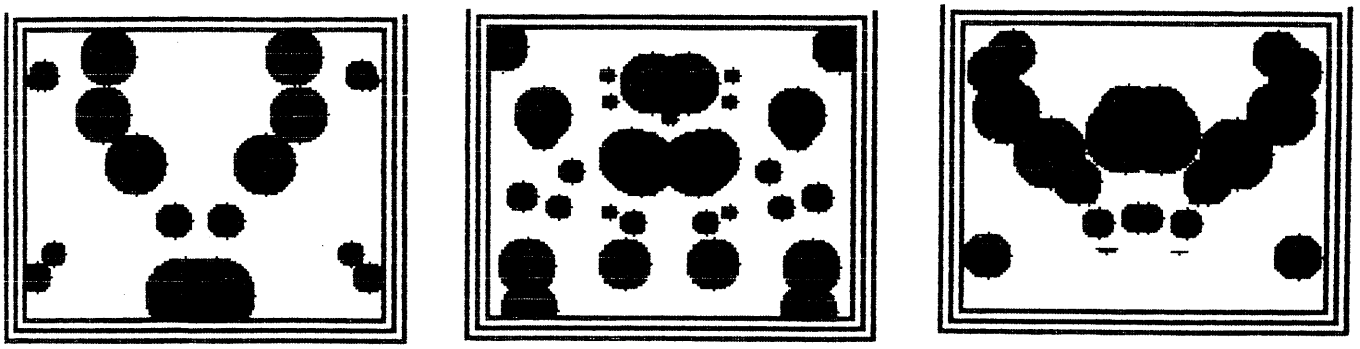
```
1     REM PROGRAMMA GEMAAKT DOOR J.G. van der HOEVEN.
4     MODE 0:PRINT CHR$(12):CURSOR 0,12:PRINT "Als U tydens het tekenen op de spatiebalk drukt
rygt U"
5     PRINT "informatie over de op dat moment in gebruik zynde kleuren."
6     CURSOR 10,1:PRINT "Druk op de balk om door te gaan."
7     G=GETC:IF G=0 THEN WAIT TIME 5:GOTO 7
10    MODE 0:COLORG 0 0 0 0:MODE 2
15    X1=0:X2=XMAX:X3=7:X4=0:X5=XMAX/2:Y1=0:Y3=YMAX/2:Y4=0:Y5=YMAX:XB=XMAX/2:YB=0
20    D=RND(15)+1:B=RND(15)+1:C=(D+B) MOD 16
30    COLORG 0 D B C
35    FOR N=0 TO XMAX
40    DRAW X1,Y1 X1,Y1+15 23:X1=(X1+1) MOD XMAX
50    V1=(X1+10) MOD XMAX:DRAW V1,Y1 V1,Y1+15 17
60    DRAW X2,Y1+5 X2,Y1+10 18:XA=(XA+1) MOD (XMAX-1):X2=XMAX-XA
70    VA=(XA+15) MOD (XMAX-1):V2=XMAX-VA:DRAW V2,Y1+5 V2,Y1+10 19
80    DRAW 0,Y3 XMAX,Y3 16:Y3=(Y3+1) MOD YMAX
90    DRAW X3,0 X3,YMAX 18:X3=(X3+1) MOD XMAX
100   DRAW X4,Y4 X4+5,Y4 21:X4=(X4+1) MOD (XMAX-5):Y4=(Y4+1) MOD YMAX
110   XB=(XB+1) MOD (XMAX-7):X5=XMAX-XB:Y5=(Y5+1) MOD (YMAX-4):DRAW X5,Y5 X5-7,Y5 23
140   IF GETC<>0 THEN GOSUB 200
150   NEXT N:Y1=(Y1+16) MOD (YMAX-15)
170   GOTO 20
200   MODE 1:MODE 2
210   FILL 0,3 71,13 D:FILL 0,21 71,31 B:FILL 0,39 71,49 C
220   KL=16:FOR X=2 TO 65 STEP 9:FILL X,0 X+4,YMAX KL:KL=KL+1:NEXT
230   PRINT "gebruikte kleuren:":PRINT "    16     17     18     19     20     21     22     23"
245   IF GETC=0 GOTO 245
250   MODE 1:MODE 2A
260   FILL 0,1 71,3 D:FILL 0,5 71,7 B:FILL 0,9 71,11 C
270   KL=16:FOR X=16 TO 48 STEP 4:IF X=32 THEN NEXT X
275   FILL 0,X 71,X+2 KL:KL=KL+1:NEXT
280   FILL 1,0 4,64 D:FILL 7,0 10,64 B:FILL 13,0 16,64 C
290   KL=16:FOR X=19 TO 67 STEP 6:IF X=43 THEN NEXT X
295   FILL X,0 X+3,64 KL:KL=KL+1:NEXT
300   PRINT "Gebruikte kleuren by COLORG A B C D"
310   PRINT "    B    C   D    16   17  18   19      20   21  22   23"
400   IF GETC=0 GOTO 400
500   WAIT TIME 10:MODE 2:RETURN
600   MODE 2:COLORG 0 0 0 0
610   FILL 0,0 XMAX,YMAX 18
620   FOR X=0 TO XMAX STEP 3:DRAW X,0 XMAX/2,YMAX 19:NEXT
630   COLORG 0 0 15 15
640   FILL 0,0 XMAX,YMAX 16
650   FOR X=0 TO XMAX STEP 3:DRAW X,YMAX XMAX/2,0 17:NEXT
660   COLORG 0 10 0 10
670   FILL 0,0 XMAX,YMAX 18
680   FOR Y=0 TO YMAX:DRAW XMAX/2-(Y MOD 2),Y XMAX/2+(Y MOD 30),Y 19:NEXT
690   COLORG 0 0 14 14
700   FILL 0,0 XMAX,YMAX 16
710   FOR Y=0 TO YMAX STEP 3:DRAW XMAX,Y 0,YMAX/2 17:NEXT
720   COLORG 0 5 0 5:GOTO 610
```

IAND IOR IXOR INOT SHL SHR

```
10000 A$="REAL BYTE MANIPULATIONS":GOSUB 900:PRINT
10010 PRINT "integer or   00101001 IOR 10001100 = 10101101":PRINT :PRINT "integer and  00011000 IAND
 11010111 = 00010000"
10030 PRINT :PRINT "exclusive or 10001111 IXOR 11111111 = 01110000":PRINT :PRINT "shift right  10110
00 SHR 4 = 0001011"
10050 PRINT :PRINT "shift left   00001010 SHL 4 = 1010000":GOTO 800
```

HOLLAND IN MODE 2

```
      REM holland in mode 2 ***
10        COLORG 8 5 3 0
20        CLEAR 1500
30        MODE 2
40        FOR X!=1.0 TO 109.0
50    1     READ A,B,C,D
70    1     FILL A,B C,D 5
75    1     WAIT TIME 25
80    1     NEXT
100       DATA 34,0,36,6,33,1,33,2,37,0,37,4,38,2,38,3,35,7,37,8,34,9,38
C         ,9,7,9,8,9
110       DATA 1,10,2,10,6,10,10,10,31,10,37,10,1,11,11,11,27,11,37,11,1
C         ,12,6,12
115       DATA 1,13,4,13
120       DATA 9,12,12,12,26,12,38,11,10,13,15,14,25,13,39,17,6,14,8,14,
C         20,14,22,14
130       DATA 4,15,10,15,14,15,15,19,17,15,18,15,21,15,23,15,2,16,5,16,
C         7,16,9,16
140       DATA 13,16,13,19,16,16,17,20,18,16,35,32,3,17,4,17,6,18,8,18,1
C         0,18,11,18
150       DATA 36,18,36,21,37,18,37,20,38,18,38,19,8,20,10,20,6,21,8,21,
C         11,21,13,21
160       DATA 10,22,11,22,8,23,10,23,17,22,17,39,16,22,16,33,15,22,15,3
C         1,14,23,14,30
170       DATA 13,23,13,28,12,24,12,28,10,25,11,25,36,24,38,41,39,25,41,
C         41,42,24,42,41
180       DATA 43,25,44,41,45,26,45,41,46,26,46,28,47,27,47,28,46,30,46,
C         38,47,31,47,37
190       DATA 48,32,49,37,50,33,50,36,18,33,22,43,23,33,26,33,23,34,24,
C         34,35,33,31,33
200       DATA 35,34,32,34,35,35,33,35,35,36,34,36,23,36,23,41,24,36,24,
C         38,23,42,25,42
210       DATA 23,43,26,43,19,44,19,48,20,44,20,47,21,44,21,46,22,44,23,
C         47
220       DATA 28,34,29,34,26,35,30,35,26,36,31,36,27,37,32,37,29,38,33,
C         38,30,39,34,39
230       DATA 30,40,33,40,37,42,45,58,33,42,32,42,33,43,36,56,32,43,31,
C         43,32,44,33,55
235       DOT 11,17 5:DOT 12,19 5:DOT 35,37 5
240       DATA 31,47,29,54,30,46,28,46,28,47,28,48,28,51,28,53,30,55,31,
C         55,34,57,35,57
250       DATA 36,57,36,58,42,59,46,59,46,58,47,43,48,43,49,56,50,55,50,
C         47,51,54,51,50
260       DATA 19,50,20,50,20,51,21,51,21,52,21,51,21,54,21,55,22,55,23,
C         55,23,56,23,55
270       DATA 25,57,25,58,26,58,28,58,28,59,30,59,32,60,34,60,38,61,40,
C         61,43,62,43,62
1000      GOTO 1000
```



```
100   REM *** RORSCHACH-INKTVLEKKEN
110   REM *** DE BONT CORNEEL : 19-10-1981
120   MODE 0:PRINT CHR$(12):COLORT 12 0 12 12:POKE #75,32
130   POKE #BFEF,#5A:CURSOR 0,23:PRINT "INKTVLEKKEN"
140   CURSOR 0,21
150   FOR I=0.0 TO 58.0:PRINT CHR$(255);:NEXT I:PRINT
160   FOR I=20.0 TO 5.0 STEP -1.0:PRINT CHR$(255);CHR$(255);
170   PRINT TAB(57);CHR$(255);CHR$(255):NEXT I
180   FOR I=0.0 TO 58.0:PRINT CHR$(255);:NEXT I
190   CURSOR 4,19:PRINT "DIT PROGRAMMA GENEREERT WILLEKEURIGE TEKENINGEN,"
200   CURSOR 4,18:PRINT "GEBASEERD OP GROTE EN KLEINE CIRKELS,WAARDOOR EEN"
210   CURSOR 4,17:PRINT "INKTVLEK-EFFECT ONTSTAAT.DE TEKENINGEN ZIJN PER-"
220   CURSOR 4,16:PRINT "FECT SIJMMETRISCH EN KUNNEN GEBRUIKT WORDEN IN DE"
230   CURSOR 4,15:PRINT "ALOM BEKENDE PSYCHOLOGISCHE 'RORSCHACH-TEST'."
240   CURSOR 4,14:PRINT "       (MEN VRAAGT DE PROEFPERSONEN DINGEN"
250   CURSOR 4,13:PRINT "        TE HERKENNEN IN DEZE INKTVLEKKEN)"
260   CURSOR 4,12:PRINT "ZODRA EEN TEKENING IS OPGEBOUWD,ZAL DE ACHTERGROND"
270   CURSOR 4,11:PRINT "SNEL VAN KLEUR VERANDEREN,EN EEN FLUITSIGNAAL ZAL"
280   CURSOR 4,10:PRINT "WEERKLINKEN.MEN KAN HET PROGRAMMA STOPPEN DOOR OP DE"
290   CURSOR 4,9:PRINT "TAB-TOETS TE DRUKKEN,OFWEL DRUKT MEN OP DE SPATIE-"
300   CURSOR 4,8:PRINT "BALK,EN ZAL ER EEN NIEUWE TEKENING GEMAAKT WORDEN."
310   CURSOR 4,6:PRINT "      DRUK OP EEN TOETS OM TE BEGINNEN ";
320   IF GETC=0.0 THEN 320
330   REM *** TEKENING
340   MODE 4:COLORG 12 8 4 0
350   FILL 0,0 158,129 0:FILL 2,2 156,127 12
360   FILL 4,4 154,125 0:FILL 6,6 152,123 4
370   FILL 8,8 150,121 8:FILL 10,10 148,119 12
380   S=INT(10.0*RND(1.0))+10.0
390   FOR N=1.0 TO S:A=INT(69.0*RND(1.0))+10.0
400   B=INT(109.0*RND(1.0))+10.0:D=INT(15.0*RND(1.0))+2.0
410   FOR Z=-D TO D:AZ=A-SQR(D*D-Z*Z)
420   IF AZ<10.0 THEN AZ=10.0
430   IF AZ>79.0 THEN AZ=79.0
440   CZ=A+SQR(D*D-Z*Z)
450   IF CZ<10.0 THEN CZ=10.0
460   IF CZ>79.0 THEN CZ=79.0
470   BZ=B+Z
480   IF BZ<10.0 THEN BZ=10.0
490   IF BZ>119.0 THEN BZ=119.0
500   DRAW AZ,BZ CZ,BZ 0:DRAW 158-AZ,BZ 158-CZ,BZ 0
510   NEXT Z:NEXT N
520   SOUND 1 0 15 0 FREQ(1000.0):WAIT TIME 10
530   COLORG 9 8 4 0:SOUND OFF :WAIT TIME 10
540   G=GETC
550   IF G=32.0 THEN COLORG 12 8 4 0:GOTO 370
560   IF G=9.0 THEN 580
570   COLORG 12 8 4 0:GOTO 520
580   MODE 4A:PRINT :PRINT :PRINT TAB(22);:POKE #75,95:END
```

INTERNAL BASIC REPRESENTATIONS IN THE DAI PERSONAL COMPUTER

SECTION 1 : USE OF MEMORY

In the DAI personal computer a BASIC program is stored in 2 areas :
- textbuffer : coded form of the inputted program (= sequence program execution)
- symboltable : table which describes the variables (name, type and value or pointer to value)

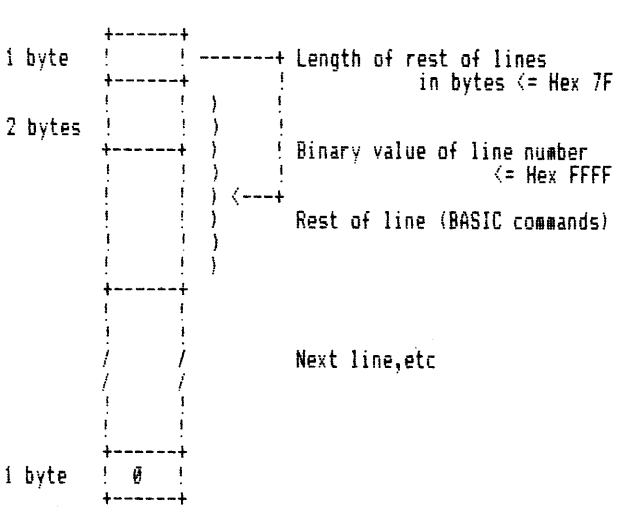
Location of these 2 areas is described in following system ram pointers :

HEX LOC	SYMBOL	DESCRIPTION	DEFAULT
29B-29C	HEAP	Pointer to start of heap	#2EC
29D-29E	HSIZE	Size of heap in bytes	#100
29F-2A0	TXTBGN	Pointer to start of textbuffer	#3EC
2A1-2A2	STBBGN	Pointer to start of symboltable	#3ED
2A3-2A4	STBUSE	Pointer after end of symboltabl	#3EE
2A5-2A6	SCRBOT	Bottom of screen ram	#B350

NOTES :
- default values after reset are given
- HEAP, TEXTBUF and SYMBOLTABLE use always successive parts of the RAM space -> HEAP + HSIZE = TXTBGN
- value of HSIZE is altered with a CLEAR command
- the area from STBUSE till SCRBOT is used during certain basic commands :
  - during SAVEA and LOADA of string arrays (to make a clean copy of all the strings in the array)
  - during program EDIT (the HEAP is used as editbuffer, before sending characters to it make the HEAP as large as possible by moving textbuffer and symboltable up to the screen ram)
- STBBGN and STBUSE are changed when entering program lines
- SCRBOT is dependend on the sreenmode
- direct command lines are tempory stored in a ram buffer starting at hex 13E ; used format same as normal line, but no linenumber is included.

SECTION 2 : FORMAT FOR PROGRAM STORAGE

A.Program is stored as a number (0 or more), followed by a single 0 byte



B.Commands are stored as a single byte code,followed by parameters as necessary.
The byte will have a value >= 81 hex.

Note : not-prgr : this command is only allowed as a direct command
not-comd : this command is not allowed in a direct comand line
no message : direct and stored allowed

NEW — 81 Hex — Not-prgr

CONT — 82 Hex — Not-prgr

STOP — 83 Hex — Not-comd

END — 84 Hex — Not-comd

RESTORE — 85 Hex

RETURN — 86 Hex — Not-comd

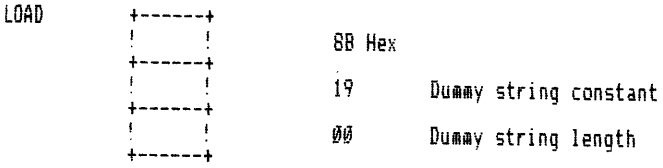
RUN — 87 Hex — Not-prgr

RUN < line number >
88 Hex — Not-prgr
2 bytes — Line number

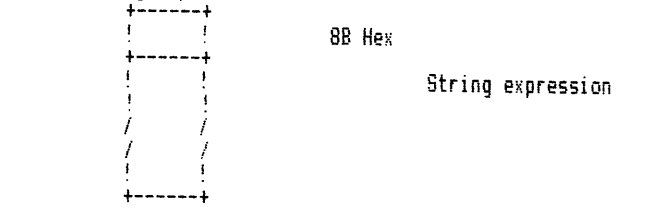
GOTO < line number >
89 Hex — Not-comd
Line number

GOSUB < line number >
8A Hex — Not-comd
Line number

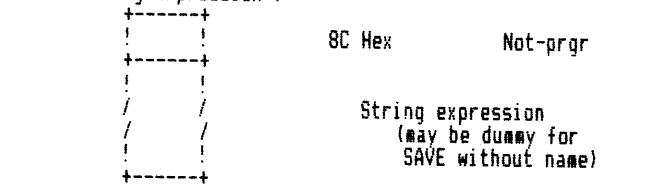
LOAD



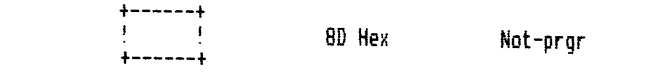
LOAD ⟨ string expression ⟩



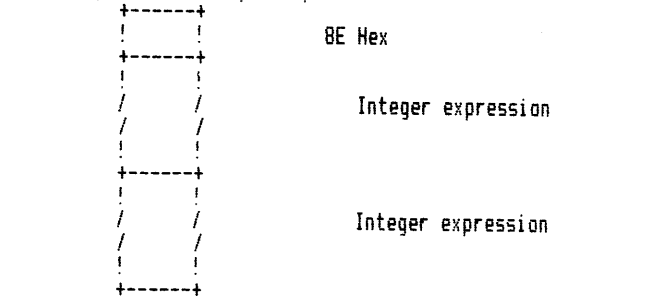
SAVE ⟨ string expression ⟩



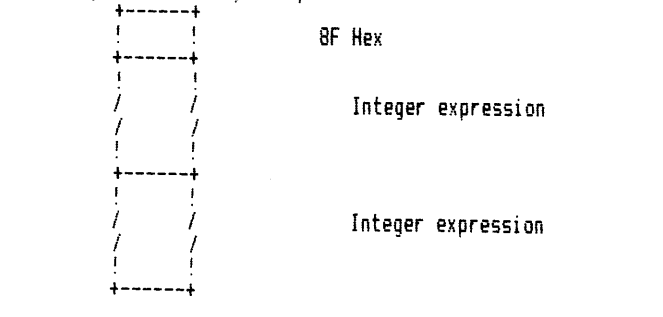
CHECK



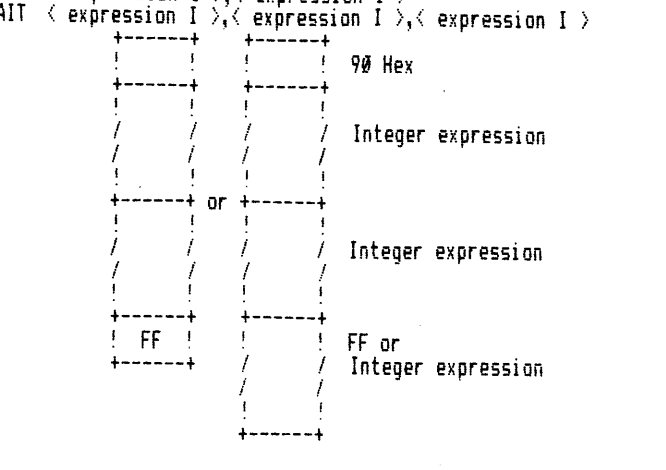
OUT ⟨ expression I ⟩ , ⟨ expression I ⟩



POKE ⟨ expression I ⟩ , ⟨ expression I ⟩



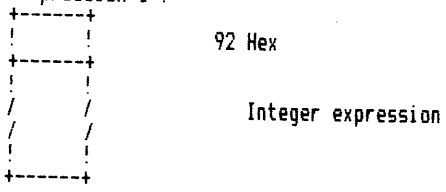
WAIT ⟨ expression I ⟩,⟨ expression I ⟩
WAIT ⟨ expression I ⟩,⟨ expression I ⟩,⟨ expression I ⟩



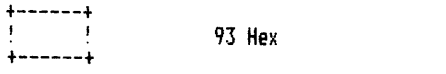
WAIT MEM
⟨ expression I ⟩,⟨ expression I ⟩
WAIT MEM
⟨ expression I ⟩,⟨ expression I ⟩,⟨ expression I ⟩

As WAIT , command byte = 91 Hex

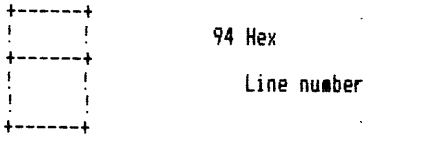
WAIT TIME ⟨ expression I ⟩



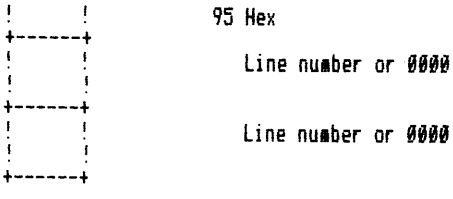
LIST



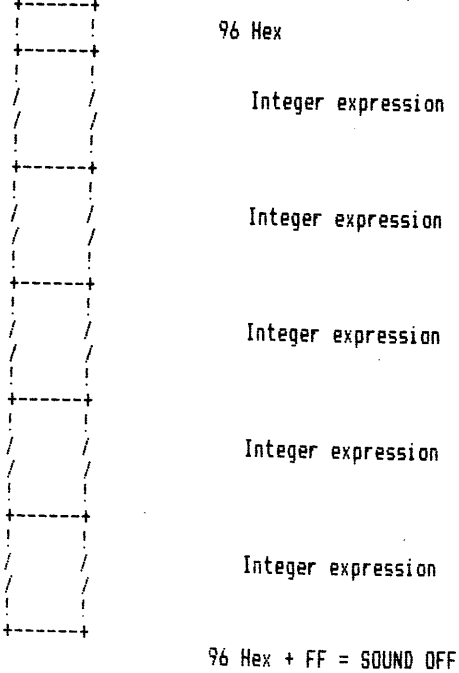
LIST ⟨ line number ⟩



LIST - ⟨ line number ⟩
LIST ⟨ line number ⟩ -
LIST ⟨ line number ⟩ - ⟨ line number ⟩

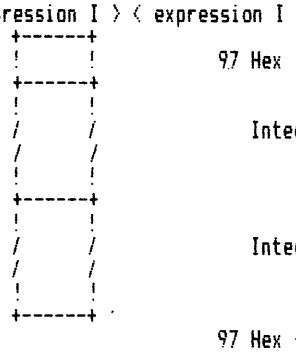


SOUND ⟨ expr I ⟩ ⟨ expr I ⟩ ⟨ expr I ⟩ ⟨ expr I ⟩
⟨ expr I ⟩

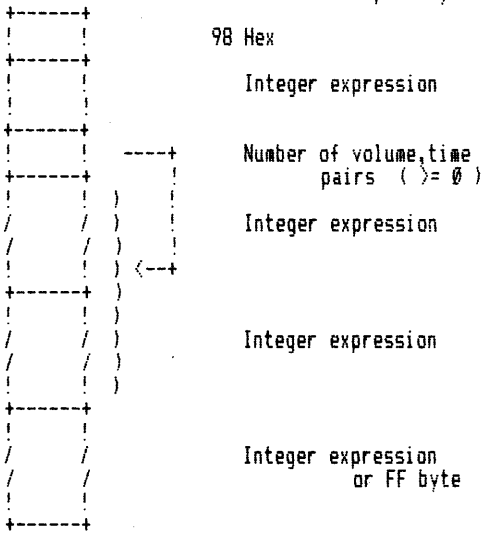


96 Hex + FF = SOUND OFF

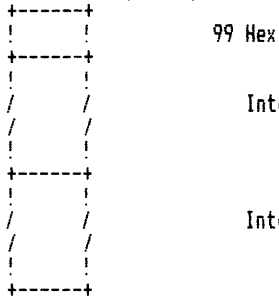
NOISE ⟨ expression I ⟩ ⟨ expression I ⟩



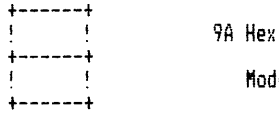
ENVELOPE
⟨ expr I ⟩ { ⟨ expr I ⟩,⟨ expr I ⟩ ; } ⟨ expr I ⟩
or ⟨ expr I ⟩ ⟨ expr I ⟩,⟨ expr I ⟩ ; { ⟨ expr I ⟩, ⟨ expr I ⟩; }



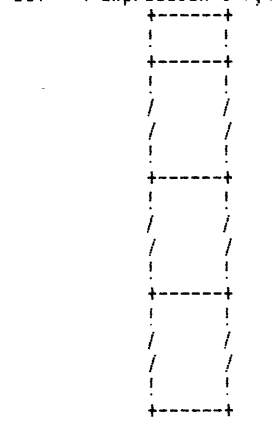
CURSOR
⟨ expression I ⟩ , ⟨ expression I ⟩



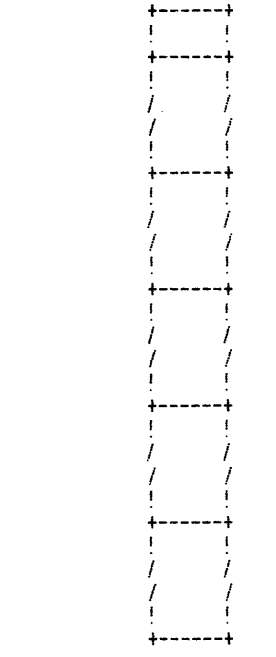
MODE ⟨ mode number ⟩



DOT ⟨ expression I ⟩,⟨ expression I ⟩ ⟨ expression I ⟩



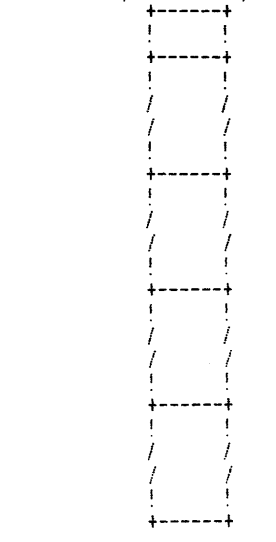
DRAW ⟨ expr I ⟩,⟨ expr I ⟩ ⟨ expr I ⟩,⟨ expr I ⟩ ⟨ expr I ⟩



FILL ⟨ expr I ⟩,⟨ expr I ⟩ ⟨ expr I ⟩,⟨ expr I ⟩ ⟨ expr I ⟩

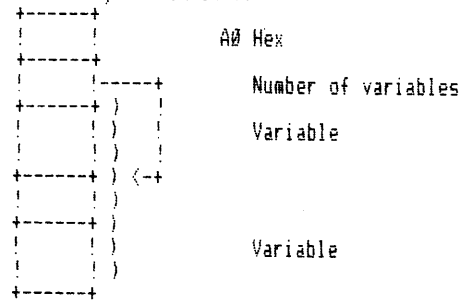
As DRAW , but command byte = 9D

COLORT
⟨ expr I ⟩ ⟨ expr I ⟩ ⟨ expr I ⟩ ⟨ expr I ⟩

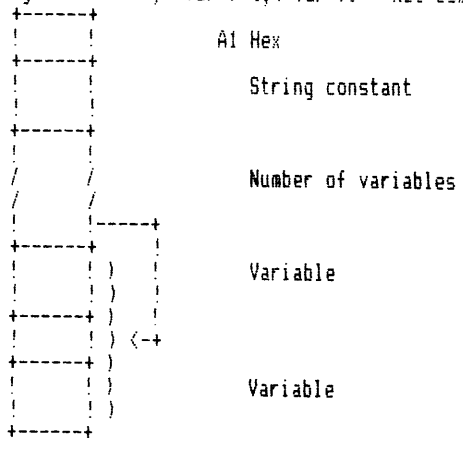


COLORG as COLORT but command code = 9F

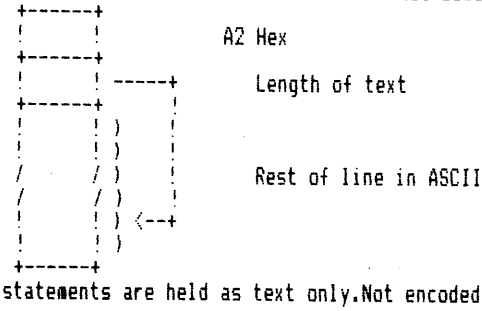
INPUT < variable > {,< variable >}



INPUT < string constant >;< var > {,< var >} Not-comd

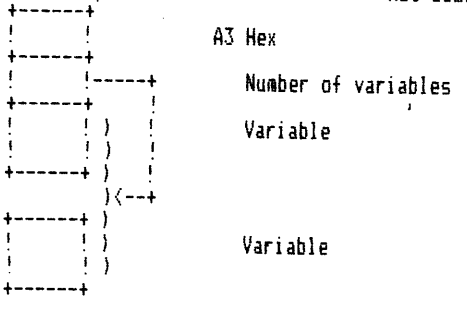


DATA < text > Not-comd

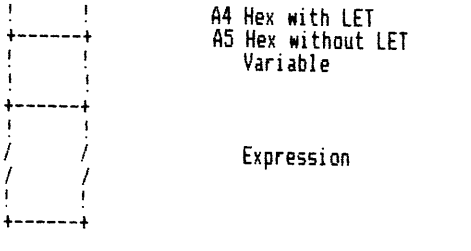


Data statements are held as text only.Not encoded.

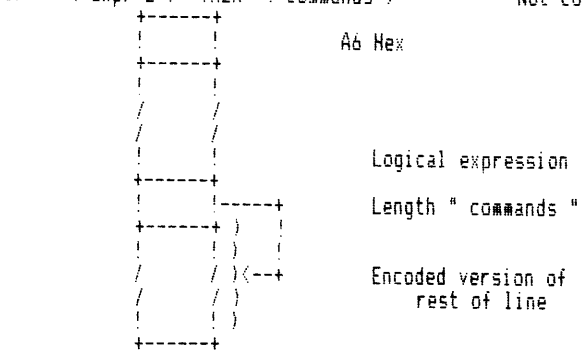
READ < variable > {,< variable > } Not-comd



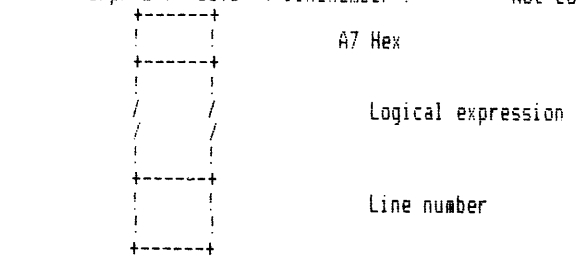
LET < variable > = < expression >
< variable > = < expression >



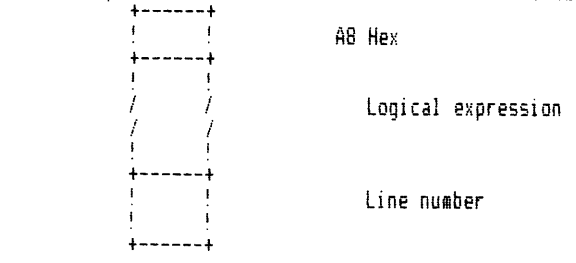
IF < expr L > THEN < commands > Not-comd



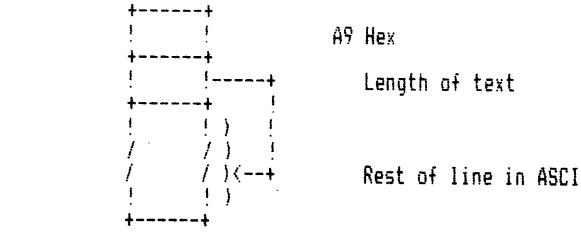
IF < expr L > GOTO < linenumber > Not-comd



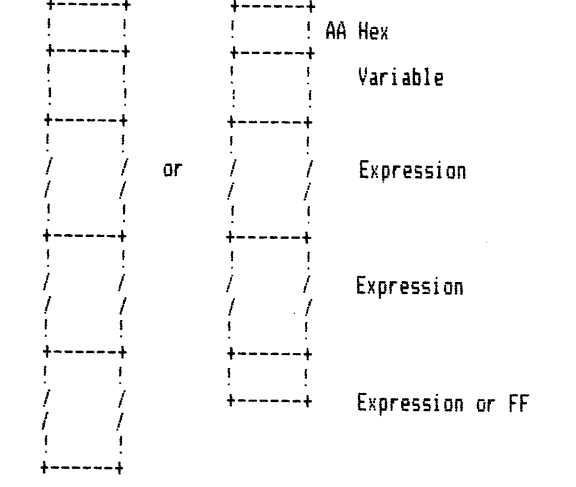
IF < expr L > THEN < linenumber > Not-comd



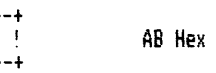
REM < text > Not-comd



FOR < var > = < expr > TO < expr > { STEP < expr > }

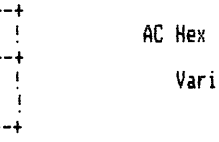


NEXT



AB Hex

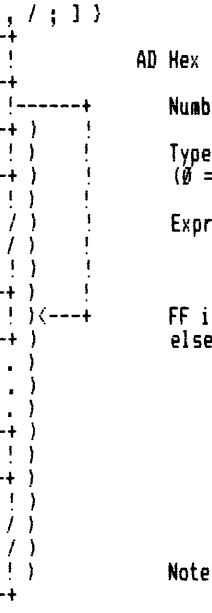
NEXT < variable >



AC Hex

Variable

PRINT { < expr > [, / ;] }



AD Hex

Number of expressions

Type of expression
(Ø = fpt, 1Ø = int
2Ø = str)

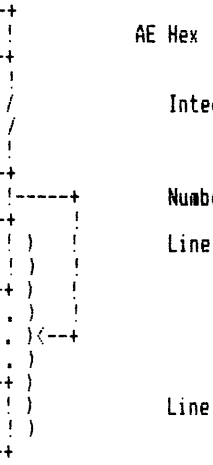
Expression

FF if no seperator,
else 2C or 3B for , ;

Note : PRINT = AD ØØ

ON < expr I > GOTO < linenumber > {,< linenumber >}

Not-comd



AE Hex

Integer expression

Number of line numbers

Line number

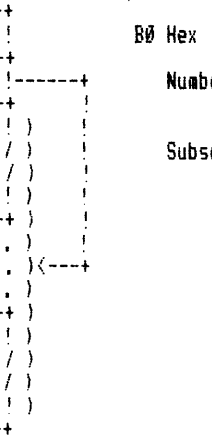
Line number

ON < expr I > GOSUB < linenumber > {,< linenumber >}

Not-comd

As ON...GOTO,except command code = AF

DIM < subscript variable > {,< subscript variable >}



BØ Hex

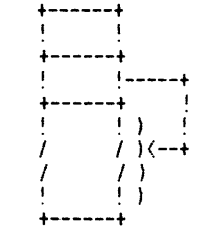
Number of variables

Subscript variable

200 - DAInamic 11

< text >

Not-comd

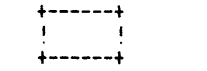


B1 Hex

Length of text

Rest of line in ASCII

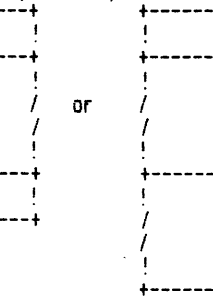
UT



Not-prgr

B2 Hex

CALLM < expr I > [,< variable >]

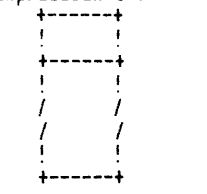


B3 Hex

Integer expression

FF or variable

CLEAR < expression I >



B4 Hex

Integer expression

IMP

This command is not encoded

Not-prgr

EDIT

Not-prgr

EDIT < line number >

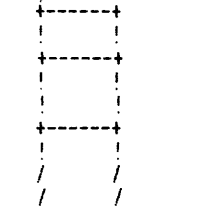
Not-prgr

EDIT < line number > - < line number >

Not-prgr

As LIST,except command code = B6 / B7 / B8

SAVEA < array name > [< string expression >]



B9 Hex

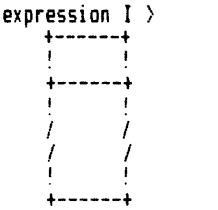
Array name

String expression
(may be dummy)

LOADA < arrayname > , [< string expression >]

As SAVEA,except command code = BA

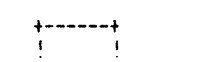
TALK < expression I >



BB Hex

Integer expression

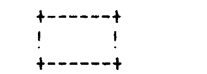
STEP



Not-prgr

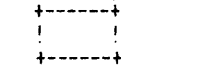
BC Hex

TRON



BD Hex

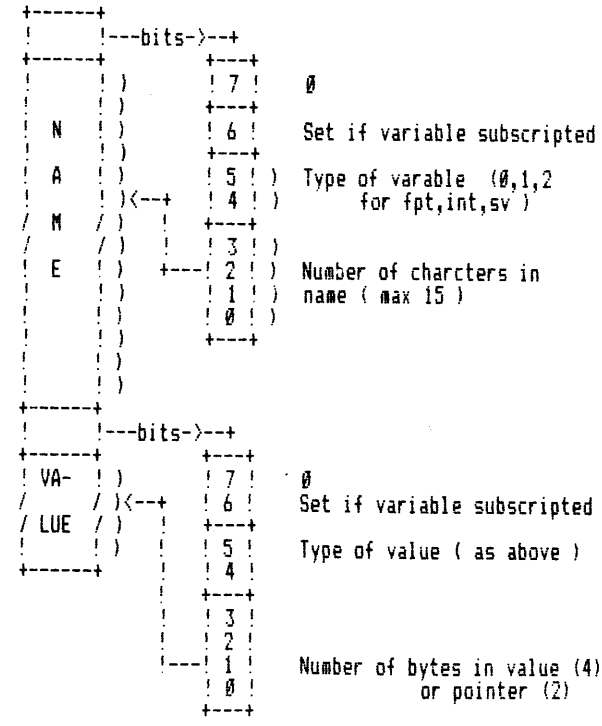
TROFF



BE Hex

SECTION 3 : FORMAT OF SYMBOL TABLE + ARRAY STORAGE

The symbol table is an integral part of a program, and holds the names of all variables.
In addition it holds the values for non-subscripted integer and floating-point variables.
For subscripted variables and strings the table holds only pointers.
The table is stored as a number of name and value or pointer pairs,followed by a single Ø byte.

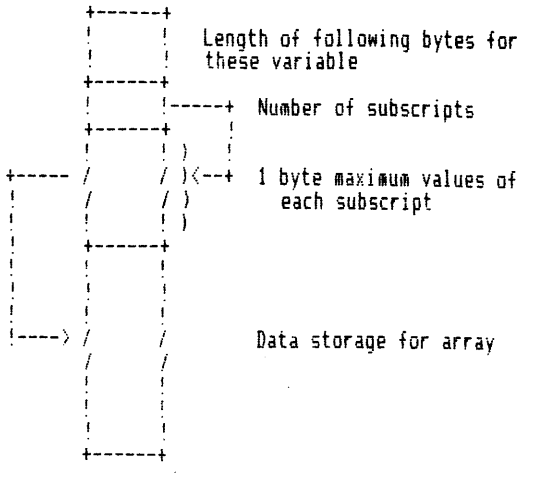


Name field :
The name is held in ASCII,first character at lowest address.

Value field :
Unsubscripted integer/fpt variables are held as 4 byte values.
Unsubscripted string variables are held as a 2 byte pointer to the string.

Subscripted variables of any type are held as a pointer to other information.
See next paragraph.
This pointer is Ø until the array has been dimensioned.

Array storage :
Array data and information are held in the "Heap".
The format is :



The entries in the data storage are 4 byte values (int/fpt) or 2 byte pointers (strings).

SECTION 4 : FORMAT FOR EXPRESSIONS / VALUES / VARIABLES

Expressions are held internally in operator prefix format.
That is to say the operator code precedes its operand or operands.
Each of these operands may be another operator and operands and so on.
Since the operator comes first, no syntax analysis or operator priorities are needed at run time.

Eg : A + B * C / D ^ E

Is held as + A * B / C ^ D E

Individual parts of an expression (items) are held as follows :

A.CONSTANTS

Integer + floating point values are held in binary and preceded by :

1Ø if floating point
14 if integer

String constants are held as a length (1 byte) plus the characters in ASCII and preceded by :

18

Hex constants are a special case of integer constants and are preceded by :

15

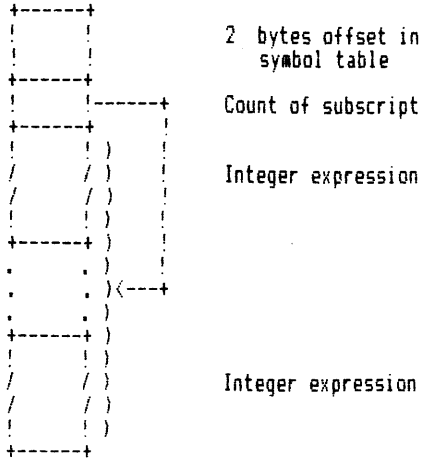
B.VARIABLES

All variables are held as a 2 byte value which hold 14 bit offset from the start of the symbol table with the 2 most significant bits of the top byte set to Ø1.
The offset is the distance from the first byte of the symbol table to the LENGTH BYTE of the VALUE FIELD for that variable.

C.SUBSCRIPTED VARIABLES

Subscripted variables are held just as are unsubscripted ones , but they are followed by a single byte giving the number of subscripts ,and that number of integer expressions.

Eg :



D.FUNCTIONS

System functions, such as SIN, VAL etc. are held as single byte 2Ø, plus a one byte code for the actual function,and the appropriate number of parameter expressions.

IMP INT-STR E.Couplet

Peut-être avez-vous déja essayé d'introduire une instruction telle que :
20 IMP INT dans un programme. Et vous avez obtenu "syntax error". On ne peut acceder a cette instruction que directement, comme RUN et EDIT.
Or il est intéressant de puvoir inclure cette fonction dans un programme, pour éviter son introduction à chaque utilisation,pour pouvoir eviter les signes % et $ et raccourcir le programma.
(à défaut le BASIC remplace une variable A$(4,2) par la variable A$(4.0,2.0) qui n'a pas réellement de sens .)
La solution est d'utiliser des POKE pour changer les indicateurs: ceux-ci sont renseignés dans le manuel DAI page 133 :

```
          0275      IMPTAB Impliciet type table
          027F      IMPTYP Defaut number type
```

mais sans explications.Les voici :

0275 définit la nature par défaut de toutes les variables dont le nom commence par A.
0276 idem pour B
0277 idem pour C

028E idem pour Z

028F est relatif à toutes les variables numériques non liées à un nom déterminé, par example les valeurs 4 et 2 relatives à la variable A$ précitée.
Le contenu de ces indicatuers est normalement 0.
Pour avoir des entiers (integer) il faut y mettre 1
Pour avoir des chaines (strings) if faut y mettre 2

Ainsi avec :

```
          110 POKE H277,2 : POKE H278,1
```

on obtiendra :

```
          B interpréte comme B!
          C     "            C$
          D     "            D%
```

OP 4 MEI 1982 IS IN UTRECHT EEN DAI-GEBRUIKERSGROEP OPGERICHT ALS ONDERDEEL VAN DE HOBBY COMPUTER CLUB.
DE GEBRUIKERSGROEP ZAL IN ALLE OPENHEID SAMENWERKEN MET DAInamic.
ER ZULLEN DIT JAAR NOG EEN AANTAL BIJEENKOMSTEN WORDEN GEHOUDEN. DE AGENDA IS ALS VOLGT:

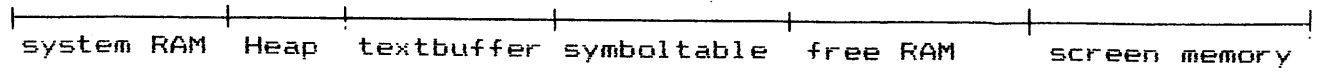
4 SEPTEMBER a.s. OM 10 UUR IN "DE BRON",VADERRIJNDREEF ,7 UTRECHT.
18 SEPTEMBER OM 10 UUR IN DE "ANTHONY FOKKERSCHOOL", BINCKHORSTLAAN,DEN HAAG.
20 SEPTEMBER OM 20 UUR IN GEBOUW ECA , FREDERIKLAAN 163,EINDHOVEN.

NADERE INLICHTINGEN ZIJN TE VERKRIJGEN BIJ :
W.NIJLAND,BUWITSACKER 21, 1902 AM CASTRICUM NEDERLAND.
TELEFOON : 02518-50360

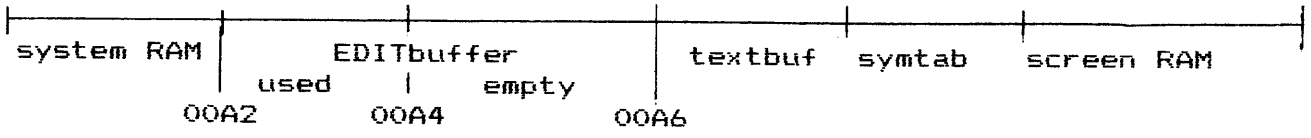
HOW WORKS THE EDIT BUFFER ?

If you examine the DAI memory map, you will not find a certain memory area reserved for the EDIT buffer. That is correct; the EDIT buffer is created the moment a EDIT command is given. How does it works? Lets have a look at the memory organisation first:

Normally, the RAM assignment is as follows:



When a EDIT command is given, the textbuffer and the symboltable are moved to the end of the free RAM space. Both Heap space and free RAM space are assigned to the EDIT buffer. The pointers for the EDIT buffer are now loaded: #00A2/A3 is start of EDIT buffer, #00A6/A7 is the end of the available buffer space.
Now the (part of the) program to be edited is listed into the EDIT buffer via a LIST routine. Then the pointer #00A4/A5 is set to the end of the used space of the EDIT buffer.
The memory map looks now as follows:



Because the output switch #0131 is set to the editbuffer, updates of the buffer contents can be made.

To finish working in the EDITbuffer, BREAK + space has to be typed in. Then the edited area in the textbuffer is deleted; a CLEAR is performed just big enough for the listed text in the EDITbuffer, and textbuffer and symboltable are moved to just behind this EDIT buffer.
Now the input switch #0135 is set to input from EDITbuffer, and the contents of this buffer is read into the textbuffer/symboltable.

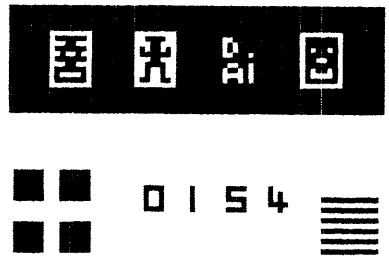
The EDITbuffer can also be used for merging programs:
CLEAR xxxx. Clear sufficient memory space for the program A still in the EDITbuffer (in other words: protect the EDITbuffer not to be overwritten when loading program B).
LOAD program A. Send it to the EDITbuffer and type BREAK/BREAK when the EDITlisting is ready. Program A is now in the EDITbuffer as described above.
LOAD program B. With LOAD, a program still in the textbuffer is deleted and replaced by the program read from tape.
POKE #135,2: Set the input switch for inputs from the EDITbuffer. The contents of the EDITbuffer is now read into the textbuffer. Both programs must have different linenumbers, otherwise program A will overwrite the same linenumbers of program B!!
Because the maximum CLEAR is #7FFF (about 32K), program A may not be longer than 32K!

Jan Boerrigter

GAMES COLLECTION 8

NEW SOFTWARE IN DAInamic LIBRARY

GAMES COLLECTION 8

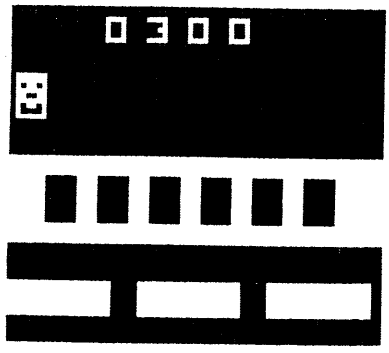


1. FRUIT MACHINE J.MOL

A VERY REALISTIC FRUIT MACHINE ,WITH ANIMATIONS IN MACHINE LANGUAGE.YOU CAN INSERT COINS,HOLD BARS.... YOU CAN PLAY THIS GAME FOR HOURS,WITHOUT GETTING BANKRUPT.

2. CATCH GOMPY II BY J.MOL

A NEW VERSION OF THE GOMPY OF GAMES COLLECTION 4. MORE ACTION,MORE COLOUR,..A COUNTER IS RUNNING TO SHOW YOUR SCORE.

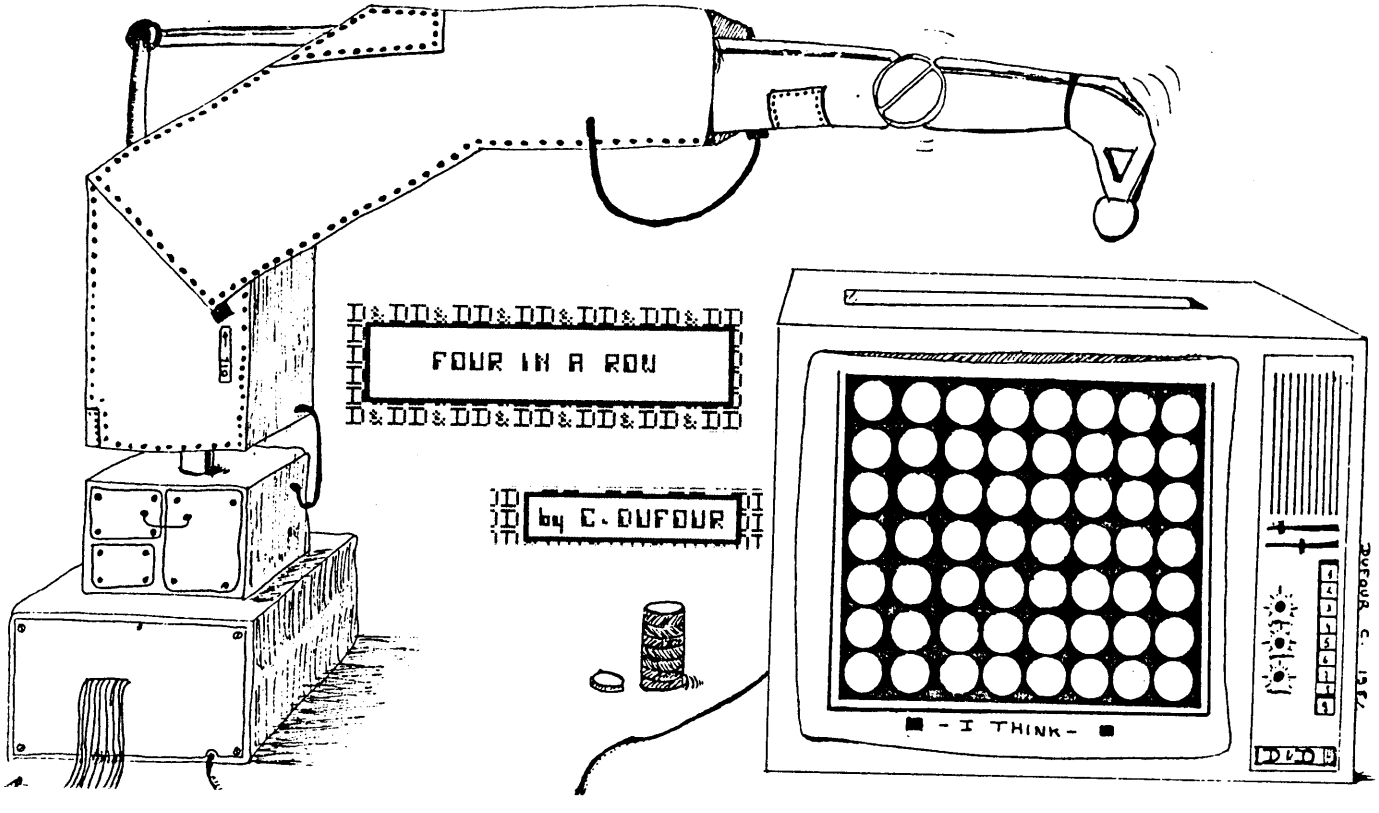


3. FOUR IN A ROW - PUISSANCE 4 BY DUFOUR

A WONDERFUL MACHINE LANGUAGE GAME IN MODE 6 WITH A VERY STRONG STRATEGY.YOU WILL LOVE THE SOUND,THE ACTION,THE COLOURS. THIS IS A 3 STARS PROGRAM !!!

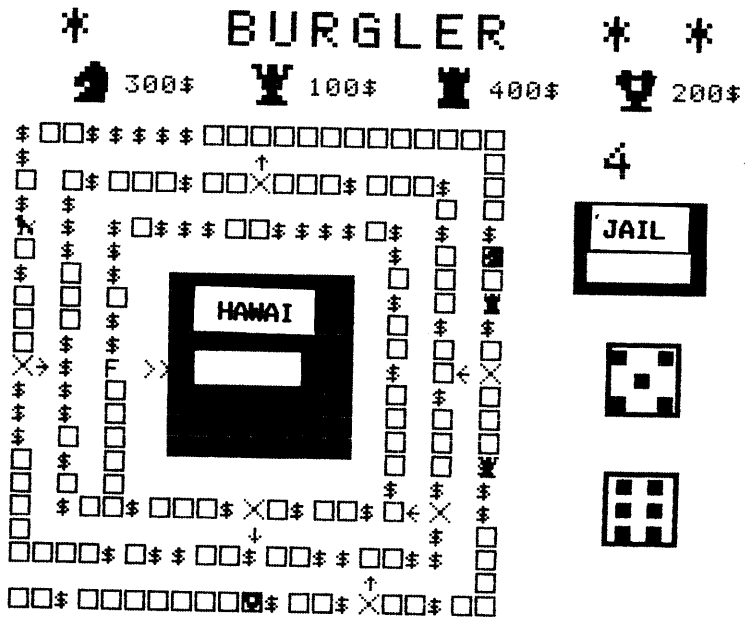
4. MENSCH AERGER DICH NICHT BY M.SIGG

THE FAMOUS BOARD GAME ON YOUR PERSONAL COMPUTER.HE CAN PLAY ON HIS OWN FOR DEMO OR WITH UP TO 4 PLAYERS IN COMPETITION.SUPERB GRAPHICS AND SOUND.



GAMES COLLECTION 8 : 750 BFR , ON DCR : 900 BFR

GAMES COLLECTION 9



NEW SOFTWARE IN DAInamic LIBRARY

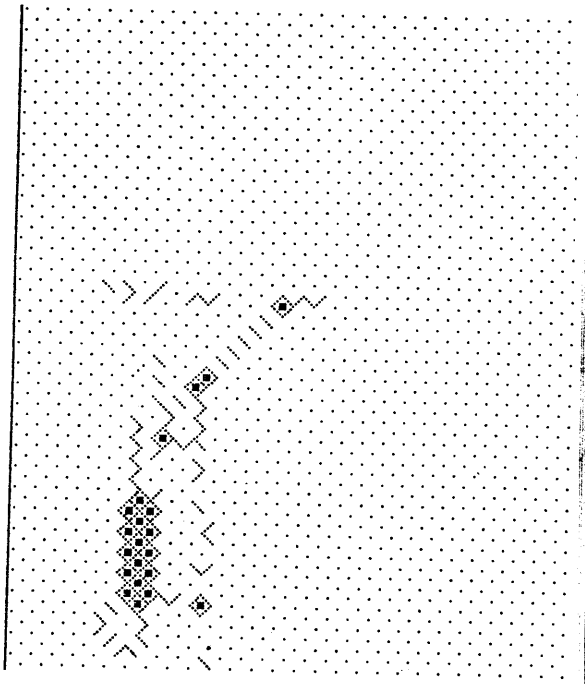
GAMES COLLECTION 9

1. SUPER MASTERMIND BY M.DE LA GRANGE

THE ULTIMATE MASTERMIND ... THAT'S ALL.

2. OUTHOUSE BY M.VERHOEVEN

TRY TO MAKE AS MANY OUTHOUSES AS POSSIBLE BY MOVING AROUND WITH YOUR PADDLE AND PUTTING WALLS. THE COMPUTERS DETECTS YOUR ACTIONS AND HOLDS SCORE.
THE GAME CAN BE PLAYED IN LOW,MEDIUM OR HIGH RESOLUTION.



3. BURGLER BY J.ROELANTS

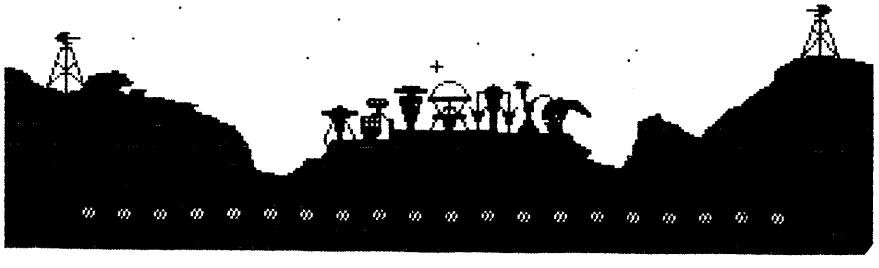
ESCAPE OUT OF THE JAIL, TRY TO ROB AS MANY BANC SAFES AS POSSIBLE AND REACH HAWAI ....
UP TO FOUR PLAYERS CAN HAVE FUN WITH THIS BEAUTIFUL ORIGINAL BOARD GAME.

4. ATOMIC ATTACK II

A SPLENDID ATTACK, BEAUTIFUL GRAPHICS AND ACTION IN MACHINE LANGUAGE.

5. MISSILE COMMAND

DEFEND YOUR CITY AGAINST INVASION OF UFO'S. AS THERE WILL BE MORE AND MORE INVADERS, THE GAME BECOMES MORE AND MORE DIFFICULT....



GAMES COLLECTION 9 : 750 BFR , ON DCR : 900 BFR

TOOLKIT 3

1. JUMBO SCREEN COPY ROUTINE

A JUMBO SCREEN COPY OF ALL GRAPHIC MODES ON YOUR EPSON PRINTER.(MX 80 WITH GRAFTRAX,MX 80 II,MX 82,MX 100).THE ROUTINE IS LOCATED IN MEMORY FROM #5300. CALLM#5300 GIVES STANDARD SCREEN COPY,CALLM#5500 GIVES JUMBO COPY. SEE EXAMPLE ELSE IN THIS ISSUE (DAITIME).

2. DAI TEST

TEST THE HARDWARE FUNCTIONS OF YOUR DAIpc.THE PROGRAM WILL REPORT IF SOMETHING IS WRONG. THERE IS A PROGRAM FOR BASIC ROMS V1.0 AND ROMS V1.1.

3. DAI AS TERMINAL

USE YOUR DAI AS TERMINAL,CONNECTED TO A LARGE COMPUTER.SEE ASSEMBLY PROGRAM IN THIS ISSUE.

4. MORSE

A VERY LARGE UNIVERSAL MORSE PROGRAM.LEARN MORSE WITH THIS BEAUTIFUL ROUTINE.

5. FFGT

A SUPERFAST VERSION OF FGT.THERE ARE SOME RESTRICTIONS IN COMPARISON WITH FGT, BUT THE GAIN IN SPEED IF IMPRESSIVE! ROUTINES ARE INCLUDED FOR MODES 3 & 5 AND SOME DEMO PROGRAMS.

6. 3-D PACKAGE

Some notes concerning 3-D-DARSTELLUNGEN:

The picture-data is stored seperated in points XP%,YP%,ZP% and connexions P1%,P2% defining which points are to connect.
The x-,y- and z-values must be integer-numbers in the range +/- 9999.
Up to 256 points and connexions can be stored.

The particular commands:

-L- load a picture; it is possible to integrate new data into the existing picture

-S- is used to save the whole stored data on tape;
when you want to save the data on a new mini-cassette which not yet contains some data, you first have to initialise it.
S and L use a subroutine which allows the user to control the DCR

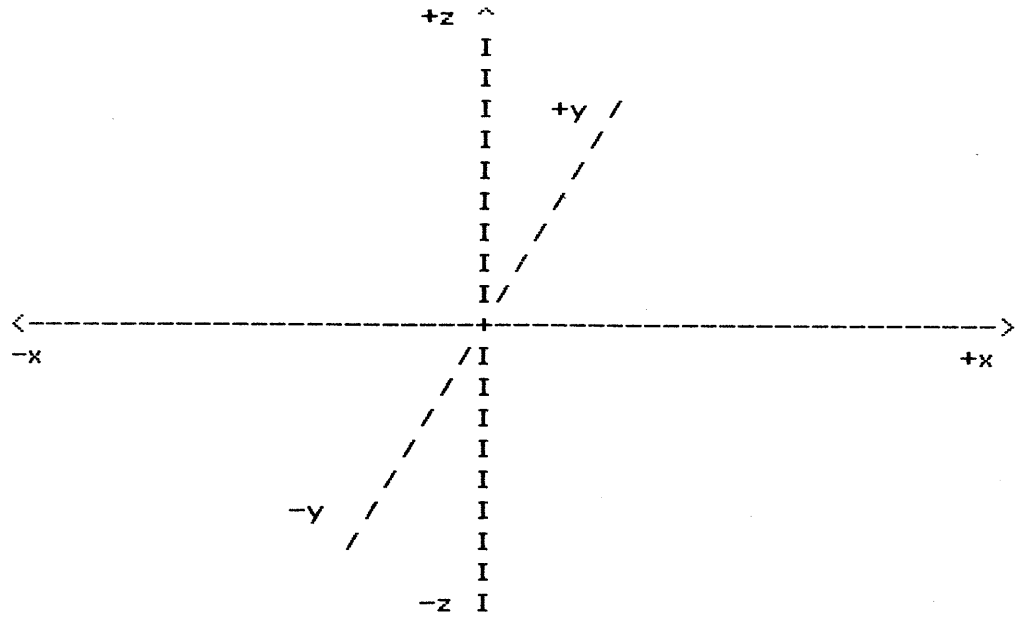
-E- offers several commands to treat the stored data:
-C- move or stretch points
-P- remove points/connexions which are defined twice ore more
(this can arise when C was used)
P takes very long when much data is to examine, so use it only when you have enough time and patience (eat something or take a walk in the meantime)
-E- the data can be revised in the Edit-mode
example: to create a pyramid you have to type
0:0,0,0 1:10,0,0 2:5,10,0 3:5,5,10
0/1 1/2 2/0 0/3 1/3 2/3

The Edit-mode is ended when TAB is pressed. Then the program asks for the number of points to read out. After this input you can get back into the Edit-buffer by typing E or keep the old data by typing A when. you made a mistake. When N is typed, the new data is read out, put in new order (if necessary) and printed out. The old data is erased now, but you can get back into the buffer by typing E if you discovered an error in the printed data. If the data is ok, type O.

-D- with this command the stored picture can be looked at from any direction by any angle. To look up the above defined pyramid from the front upper side, you could choose x,y,z = 7,-20,15 horizontal,vertical angle 0,-10
for a look from the below frontside x,y,z = 7,-20,-15, angles = 0,20
below backside x,y,z = 7,40,-15, angles = 180,20
upper backside x,y,z = 7,40,15, angles = 180,-10
upper left side x,y,z = -20,5,15 angles = 90,-10
below right side x,y,z= 40,5,-20 angles = -90,20
top side x,y,z = 5,5,30, angles = x,-90
...

horizontal angle: 0 in direction of positive y-axis (straight on)
90 in direction of positive x-axis (right)
-180=180 in direction of negative y-axis (backwards)
-90=270 in direction of negative x-axis (left)
and so on...

vertical angle: 0 in horizontal direction (straight on)
90 in direction of positive z-axis (upwards)
-90 in direction of negative z-axis (downwards)
and so on...



When P is pressed after the picture is ready (tone), it is printed out.

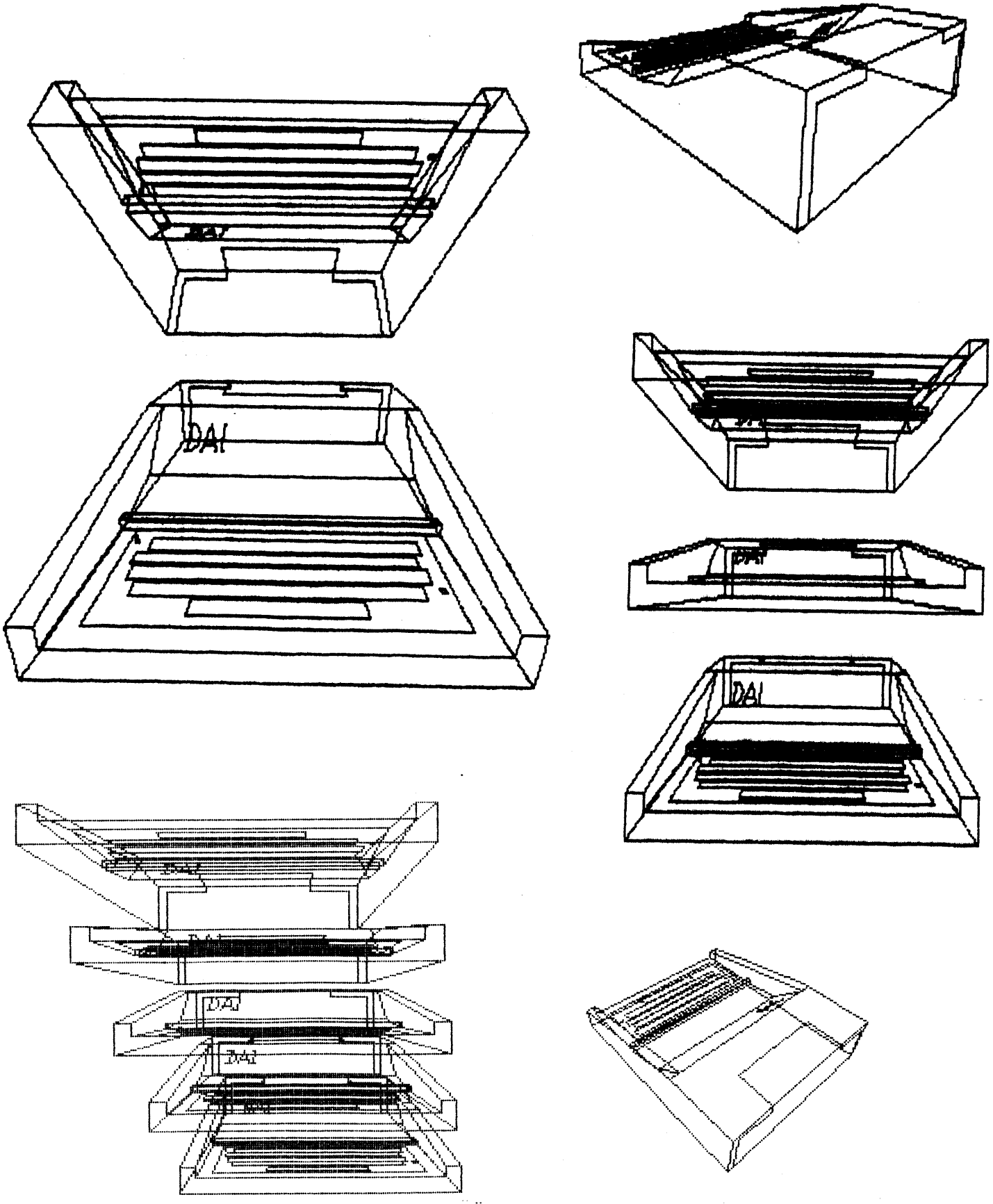
-M- draws a series of pictures which result from a movement trough the xyz-space. The initial and end standpoint and angles can be choosed. example: to make a movement towards the pyramid type inital x,y,z = 1,-80,7, end x,y,z = 4,5,6, all angles = 0, number of pictures = 15, erase the pictures after drawing. The movement is very slow and with complex pictures too slow to get the effect of moving. But other interesting effects can be reached, when the particular pictures are not erased (see enclosed examples).
The current picture is printed out when P is pressed (control-tone) while the computer is drawing. If P is pressed when the last picture is ready (tone), it too is printed.
The printer must be switched on !!

TOOLKIT 3

-C- can be used to change foreground/background colors.

-P- print the stored data in a clear, formatted manner

-N- clear data and start by new



TOOLKIT 3 : (2 TAPES) 1000 BFR, ON DCR : 1300 BFR

Sedert enige tijd bestaan er 2 versies van DAI-BASIC.
Er is V1.0 en de nieuwere V1.1.
In de V1.0 ROMs bleken enkele bugs te zitten die zoveel mogelijk en voor zover ze gekend waren, door DAI zijn verbeterd.
Door wat te experimenteren met de 2 ROM sets heb ik geprobeerd de verschillen te localiseren. Sommige inlichtingen heb ik rechtstreeks van bij DAI.
Ziehier mijn eerste voorlopige bevindingen.

1) V1.1 reageert anders op een RUN linenr.
In V1.0 werden alle veranderlijken geCLEARed.
Dat gebeurt niet meer in V1.1. Dit is een groot pluspunt voor de nieuwe ROM's. Het debuggen van BASIC programmas wordt dan eenvoudiger. Het is bvb. mogelijk om tijdens de uitvoering het prgr. te onderbreken ,te listen, evt. zelfs de waarde van veranderlijken in directe mode te veranderen om vervolgens het prgr. verder te laten lopen.

2) DIM(255,255) is toegelaten in V1.1.

3) Bij een hard reset wordt sound-channel 2 af gezet.
Dit was niet gegarandeerd in V1.0 zodat het mogelijk was dat bij het aanzetten bvb. reeds geluid werd geproduceerd

4) SAVEA en LOADA werken nu correct (zie vorige nummers van DAINAMIC-newsletter).

5) De bug in de SGN functie is eruit gehaald.

6) Een kleine bug in de EDIT mode is er ook uitgehaald.
Deze fout had echter geen desastreuse gevolgen en werd maar door 2 gebruikers gemeld! (de cursor verdween even onder sommige omstandigheden.)

7) De unaire operator ' - ' wordt in V1.1 correct herkend.
Het is dus niet meer nodig om haakjes rond de negatieve uitdrukking aan te brengen.

8) Er was een bug in het TALK commando. Meer weet ik er ook niet van.

9) GETC is 'debounced'. De toetsenbuffer wordt eerst geleegd.

10) Een gevolg is dat na het terugkeren uit de EDIT-mode het niet meer baat onmiddellijk RUN in te toetsen.

11) De TAB functie werkt nu correct onafhankelijk of de 'serial channel' (printer) al dan niet aan staat.

12) Er bleek ook soms een verkeerde foutenboodschap gegeven te worden (vb. ERROR IN LINE -zonder linenr.)

Dit zijn de mij gekende verschillen tussen V1.0 en V1.1 ROMs.
Wie een lijst wenst van de verschillen tussen de ROMs (in hexadecimale vorm) kan die op aanvraag tegen copie en verzendingskosten verkrijgen bij Jos Schepens.
Wegens examens zullen die pas begin juli opgestuurd worden.
Hopelijks zal dit kort overzicht u in staat stellen te oordelen of de nieuwe ROMs de moeite waard zijn.
De exacte prijs is mij ongekend, doch ik meen te weten dat die redelijk hoog is.

DAI pc als intelligente TERMINAL
****

Na in HCC nieuwsbrief no.39 een artikel gelezen te hebben over het gebruik van een APPLE als "verstandige terminal", dachten wij dat het voor DAI gebruikers zeker interessant zou zijn onze ervaringen op dit gebied op papier te zetten.
De hierna gegeven listing van een machine-taal programma schept de mogelijkheid om via de RS-232 uitgang met een mainframe of andere computer te converseren.
In principe kan van bijna alle faciliteiten,die standaard op een "grote computer" aanwezig zijn gebruik worden gemaakt.
Dit laatste geldt natuurlijk allen voor personen die toegang tot het systeem hebben
Het programma werkt zowel op 300 als op 1200 BAUD (zie comments in de source-listing).
Er is ook zonder probleem met diverse merken acoustische modems (300 BAUD , volgens CCITT-norm) gewerkt.
De RS232 uitgang van de DAI pc kan in dit geval rechtstreeks met het modem verbonden worden.
Aangezien ik een standaard-kabel had waarin pin 1 t/m 8 en pin 20 aan weerszijden van de pluggen doorverbonden waren is niet meer geprobeerd welke pinnen vervallen kunnen.
Vervolgens iets over het programma zelf.
De eerste versie was geschreven in BASIC en bleek niet te werken op 1200 BAUD.
Het omzetten in machine-taal (met dank aan P.Minten) loste dit probleem op.
Een klein probleemje kan er nog voorkomen met lijnen welke meer dan 60 karakters bevatten (alleen op 1200 BAUD).
Bij de door ons gebruikte VAX 11/780 computer gaan dan enkele karakters verloren
Het programma werd verder voorzien van enkele speciale karakters welke niet op het DAI key-board voorkomen maar essentieel zijn om met een groot systeem te kunnen werken.
Verder is er de mogelijkheid geschapen om vanuit de edit-buffer een DAI programma weg te schrijven in een file,welke later eventueel op een andere plek (terminal),ge-list of geprint kan worden.
Het wegschrijven gaat uitstekend, alleen wordt bij BASIC het 1e regelnummer verminkt.
Pascal en machinetaal(source) listings geven geen probleem.
Door tijdgebrek is het terugladen in de DAI nog niet helemaal rond.
Dit alles opent natuurlijk interessante perspectieven.
Bijvoorbeeld om thuis, via uw wordprocessor programma, op de DAI een brief te tikken en deze op een heel andere lokatie te laten listen of printen.
Of het werken in andere talen en het gebruiken van de data base management faciliteiten.

Het koppelen van 2 micro's hebben we nog niet geprobeerd.
Reacties of c.q verbeteringen aan het programma worden zeer op prijs gesteld
Voor de aardigheid is ook het oorspronkelijke BASIC programma ge-list en van comments voorzien.
Veel succes

Venray 31-4-1982 G.GRUITERS

```
1     PRINT CHR$(12):PRINT TAB(18);"DAI pc ALS TERMINAL"
5     PRINT TAB(18);"********************":PRINT
7     PRINT TAB(18);"(SNELHEID 300 BAUD)"
10    POKE #FFF5,#84:POKE #131,#1:F%=0:C%=16
20    K%=PEEK(#FFF3):IF K% IAND 8=8 THEN GOSUB 100
30    Z%=GETC
40    IF Z%<>0 THEN GOSUB 200
60    GOTO 20
100   X%=PEEK(#FFF0):IF X%>127 THEN X%=X%-128
110   IF X%=0 OR X%=13 THEN RETURN
111   IF X%=10 THEN 250
114   IF X%=8 THEN 260
120   PRINT CHR$(X%);:RETURN
200   K%=PEEK(#FFF3)
201   IF K% IAND C%=0 THEN 200
202   IF Z%=C% THEN F%=1
203   IF F%=1 AND Z%>64 THEN Z%=Z%-64
204   IF F%=1 AND Z%<>C% AND Z%<64 THEN F%=0
220   IF Z%=8 THEN Z%=127
230   POKE #FFF6,Z%
240   RETURN
250   PRINT :RETURN
260   A%=CURX:B%=CURY:A%=A%-1:CURSOR A%,B%:RETURN
300   REM *****COMMENTS*****
310   REM 10:BAUDRATE OP 300 BAUD, MET 1 STOPBIT.
320   REM OUTPUT ALLEEN NAAR SCREEN.
330   REM INITIALISATIE CONTROL-TOETS FLAG.
340   REM 20:LEES STATUS REGISTER.
350   REM BIT 3 (RECIEVE BUFFER LOADED) IS 1 ALS EEN
360   REM CHARACTER ONTVANGEN IS.
370   REM 30-40:INGETOETSTE CHARACTERS VERZENDEN MIDDELS
380   REM DE ROUTINE OP 200.
390   REM 60:HERHAAL LOOP 20-60.
400   REM *** INTERPRETEREN ONTVANGEN CHARACTERS ***
410   REM 100: HAAL ONTVANGEN CHARACTER.
420   REM VERWIJDER EVENTUELE 7e BIT.
430   REM 110: DOE NIETS ALS NULL OF CR ONTVANGEN IS.
440   REM 111: BIJ LINEFEED NAAR 250.
450   REM 114: BIJ CHAR DEL NAAR 260.
460   REM ***VERZENDEN CHARACTER***
470   REM 200:HAAL STATUS OP.
480   REM 201:BIT 4(TRANSMIT BUFFER EMPTY) IS 0 ALS VORIGE
490   REM CHAR. NOG NIET HELEMAAL VERSTUURD IS.WACHT!
500   REM 202:CURSOR-UP WORDT GEBRUIKT ALS CONTROL TOETS;
510   REM HET CHAR. WAT ER NA WORDT GE-TYPED WORDT ALS
520   REM CONTROL CHAR. VERSTUURD.
530   REM 203:MAAK CONTROL CHAR.
540   REM 204:RESET FLAG BIJ VERZENDEN CONTROL CHAR.
550   REM 230:VERZEND CHARACTER VIA RS232
560   REM 240:GA TERUG NAAR DE MAINLOOP.
570   REM 250:CARIAGE RETURN,LINEFEED EN TERUG MAINLOOP.
580   REM 260:BEPAAL CURSORPOS. ZET CURSOR EEN POS.TERUG.
590   REM TERUG NAAR MAINLOOP.
```

```
5     REM EASY BACKGROUND COLOR IN MODE 0
10    COLORT 8 0 5 0
15    FOR Y=5.0 TO 10.0
20    FOR X=0.0 TO 58.0
30    CURSOR X,Y:POKE #76,#FF:PRINT
40    NEXT
50    NEXT
100   FOR X=1.0 TO 4.0:PRINT "****************************":NEXT
```

DAI AS TERMINAL

PAGE 01

```
001                *       D.A.I.  USED AS A TERMINAL
002                *       ************************
003                *
004                *       Program written by P. MINTEN
005                *       Issue date wednesday 28/4/1982
006                *
007                *       This program is used to connect the D.A.I.
008                *       to a big computer (e.g a Digital Equipment
009                *       Corporations  VAX 11/780),via a full-duplex
010                *       modem.
011                *
012                *       Some of the keys of the D.A.I.-keyboard have
013                *       got different functions, to allow sending of
014                *       some non-standard characters.
015                *       If CURSOR LEFT is pressed, then ESC is send !
016                *       If   ß  is pressed , then UNDER-SCORE is send
017                *       If  ^ (power) is pressed ,the AT-sign is send
018                *       If  CURSOR UP is pressed, then the key which
019                *         is pressed next is send as CONTROL-char.
020                *         (e.g. CURSOR UP  then C  becomes CONTROL-C
021                *       If SHIFT-CURSOR LEFT is pressed,
022                *       the content of the edit-buffer is
023                *       send to the mainframe.
024                *
025                *       Problems still in the program:
026                *       1) If the D.A.I. has to create a secundary
027                *          line, because the received line is longer
028                *          then 60, some characters are lost
029                *          (at the breakpoint).
030                *          This is due to the high speed (1200 baud).
031                *          This problem doesn't exist with 300 baud.
032                *       2) A little waittime is required for moving
033                *          the screen of the D.A.I. This requires
034                *          one fill-character after a carriagereturn/
035                *          linefeed sequence. Most computers do this
036                *          automaticly , but the VAX needs the
037                *          command  $  SET TERMINAL  /CRFILL=1
038                *
039                *       If the program is used with the D.A.I.
040                *       assembler the command-sequence is as follows:
041                *       UT
042                *       >Z3
043                *       >R        READ THE ASSEMBLER
044                *       >G1100    START THE ASSEMBLER
045                *       BUFFERDIM DEC:  12
046                *       ÜR     READ THIS SOURCE PROGRAM (DAI-TERMINAL)
047                *       Ü#P.  ASSEMBLE AND LOAD
048                *        (SPACE)
049                *        (SPACE)
050                *       ÜU.
051                *       >G400
052                *       Disconnect for a moment the RS232 connection
053                *       to reset the modem.
054                *       Connect again and press RETURN
055                *       The communication should be operational now
056                *       To return to UTILITY ,just press BREAK
057                *
058                *
059                BAUD    EQU   :FFF5     BAUDRATE REGISTER IN TICC
060                BDRATE  EQU   :84       BAUDRATE 300 BAUD ONE STOP-
061                *                        BIT,(:88 FOR 1200 BAUD)!!!
062                OUTSCR  EQU   :131      OUTPUT-TO-SCREEN ADRESS
```

```
063                     STSREG  EQU   :FFF3     STATUS-REGISTER IN TICC
064                     INCHAR  EQU   :FFF0     SERIAL INPUTBUFFER IN TICC
065                     *                       CONTAINS LAST CHARACTER
066                     *                       RECEIVED
067                     CURPOS  EQU   :72        CURSOR POSITION
068                     *
069                     *
070                     *
071                             ORG   :400
072 0400 00             FLAG    DATA  0         FLAG USED FOR CONTROL CHARS.
073 0401 E5             START   PUSH  H
074 0402 D5                     PUSH  D         SAVE REGISTERS
075 0403 C5                     PUSH  B
076 0404 F5                     PUSH  PSW
077 0405 3E84                   MVI   A,BDRATE  DEFINE BAUDRATE
078 0407 32F5FF                 STA   BAUD      STORE IT IN TICC
079 040A 3E01                   MVI   A,:1      OUTPUT TO SCREEN ONLY
080 040C 323101                 STA   OUTSCR
081                     *
082                     *             MAINLOOP
083                     *             ========
084                     *       THIS LOOP IS CONTINUOUSLY EXECUTED.
085                     *       LOOKS FOR RECEIVED CHARS, AND
086                     *       TRANSMITS KEYS PRESSED.
087                     *
088                     *
089 040F 0608           MAINLP  MVI   B,8       MASK FOR "RECEIVE BUFFER
090 0411 3AF3FF                 LDA   STSREG    LOADED"
091 0414 A0                     ANA   B
092 0415 B8                     CMP   B         IF ZERO = CHAR RECEIVED
093 0416 CC3804                 CZ    INTREC    GOSUB TO HANDLE THIS
094 0419 CDBED6                 CALL  :D6BE     LOOK IF KEY PRESSED (GETC)
095 041C B7                     ORA   A
096 041D C45104                 CNZ   TRANSC    KEY PRESSED, GOSUB TO
097                     *                         TRANSMIT IT
098 0420 3AC402                 LDA   :02C4     FLAG FOR "BREAK PRESSED"
099 0423 FEFF                   CPI   :FF       #FF = BREAK PRESSED
100 0425 C20F04                 JNZ   MAINLP    LOOP AGAIN
101 0428 F1                     POP   PSW
102 0429 C1                     POP   B         RESTORE REGISTERS
103 042A D1                     POP   D
104 042B E1                     POP   H
105 042C C9                     RET             RETURN TO UTILITY
106                     *
107                     *
108                     *       HANDLE RECEIVED CHARACTERS
109                     *       --------------------------
110                     *       Routine starts at label INTREC
111                     *
112 042D E5             BACKSP  PUSH  H
113 042E 2A7200                 LHLD  CURPOS    POSITION IN VIDEO RAM
114 0431 23                     INX   H         1 BACKSPACE IS -
115 0432 23                     INX   H         2 INCREMENTS
116 0433 227200                 SHLD  CURPOS    CURX-1
117 0436 E1                     POP   H
118 0437 C9                     RET
119                     *
120 0438 3AF0FF         INTREC  LDA   INCHAR    ACCU GETS RECEIVED CHAR
121 043B FE7F                   CPI   127       IF PARITY BIT IS SET,
122                     *                        (BIT 7=1), STRIP IT!
123 043D FA4204                 JM    CHCKNU
124 0440 D680                   SUI   128
```

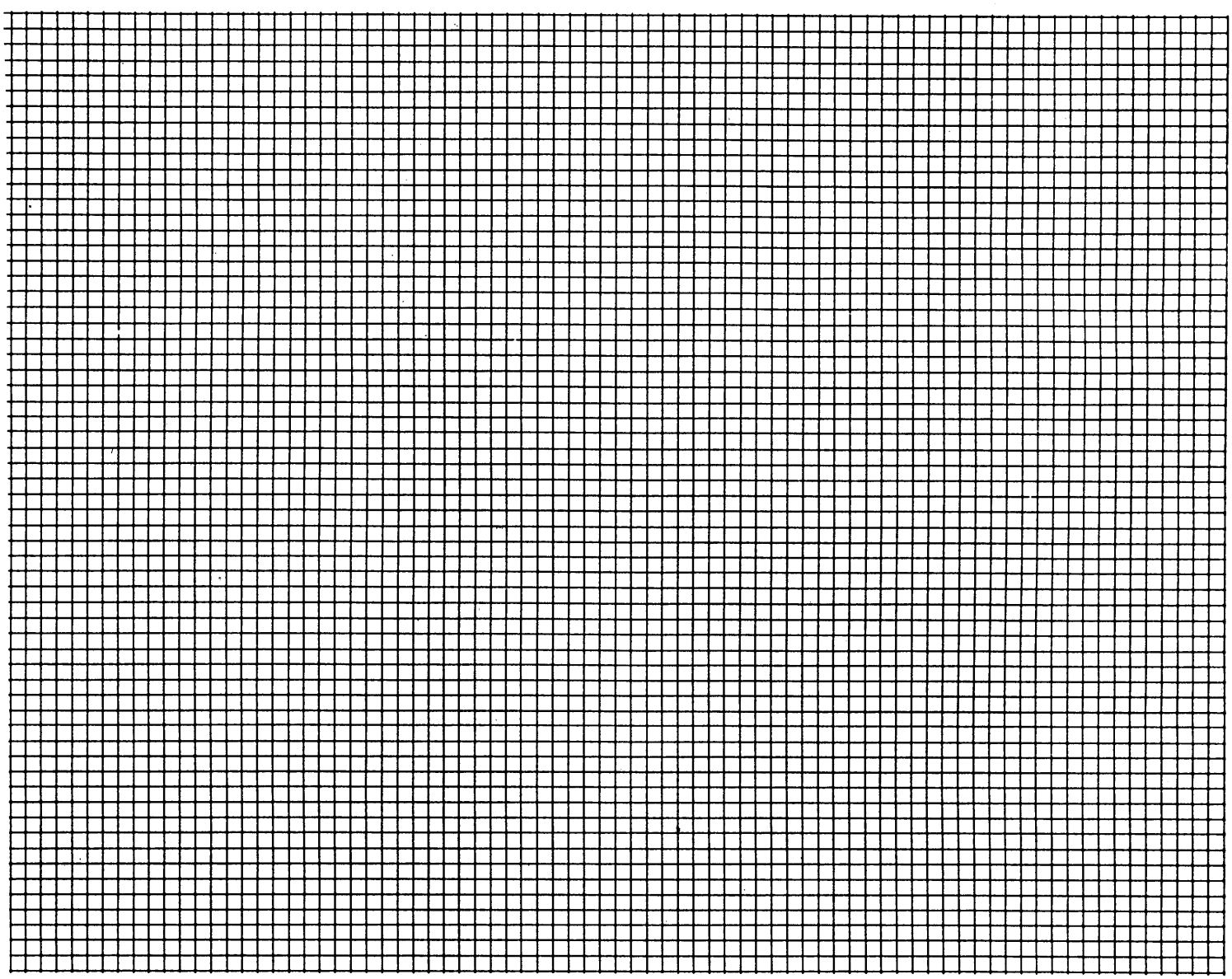
```
187 04AD CAB504            JZ    EDIT
188 04B0 7A         OUT    MOV   A,D
189 04B1 32F6FF            STA   :FFF6     SEND CHAR VIA RS232
190 04B4 C9                RET             RETURN TO MAINLOOP
191                 *
192                 *
193                 *       SEND EDIT-BUFFER
194                 *       ----------------
195                 *
196                 *
197                 *    To use this feature,read carefully
198                 *    how to do it !!
199                 *    To transmit an Assembly-language
200                 *    source listing:
201                 *
202                 *    (DNA Assembler already loaded!)
203                 *     üE1 F.
204                 *     (Break)
205                 *     (Break)   Two Breaks !!
206                 *     üU.       No other commands in between!
207                 *     >G400
208                 *               Enter the mainframe and put it
209                 *               in an Edit or Insert mode, as if
210                 *               you are entering from the keyboard
211                 *               As soon as you press SHIFT-
212                 *               LEFT-ARROW, the content of the
213                 *               Edit buffer is transmitted.
214                 *
215                 *     To transmit a Basic - program:
216                 *                  ---------------
217.                *
218                 *      *CLEAR xxxx  (depends on length of program)
219                 *      *EDIT
220                 *      (Break)
221                 *      (Break)   Two breaks !!!
222                 *      *UT
223                 *      >G400
224                 *               Rest of procedure is identical
225                 *               to that for an Assembly program
226                 *
227                 *
228 04B5 2AA200      EDIT   LHLD  :A2        START EDIT BUFFER
229 04B8 23               INX   H
230 04B9 23               INX   H
231 04BA 23               INX   H           FIRST THREE BYTES
232                 *                       IS RUBBISH
233 04BB 7E        NEXTCH  MOV   A,M        GET CHAR FROM MEMORY
234 04BC FE00             CPI   0           END OF EDIT-BUFFER
235 04BE C8               RZ                RETURN IF ZERO
236 04BF CD5104           CALL  TRANSC      TRANSMIT CHARACTER
237                 *THE LINES WITH $ IN
238                 *THE COMMENTS HAVE TO BE
239                 *DELETED FOR 1200 BAUD
240 04C2 1E03             MVI   E,3         $
241 04C4 16FF      LOOP   MVI   D,:FF       DELAY CONSTANT
242 04C6 0608      TEST   MVI   B,8         LOOK FOR RECEIVED
243 04C8 3AF3FF           LDA   STSREG      CHARACTER
244 04CB A0               ANA   B
245 04CC B8               CMP   B
246 04CD CC3804           CZ    INTREC
247 04D0 15               DCR   D           DECREMENT LOOP
248 04D1 C2C604           JNZ   TEST
```

DAI AS TERMINAL

PAGE 05

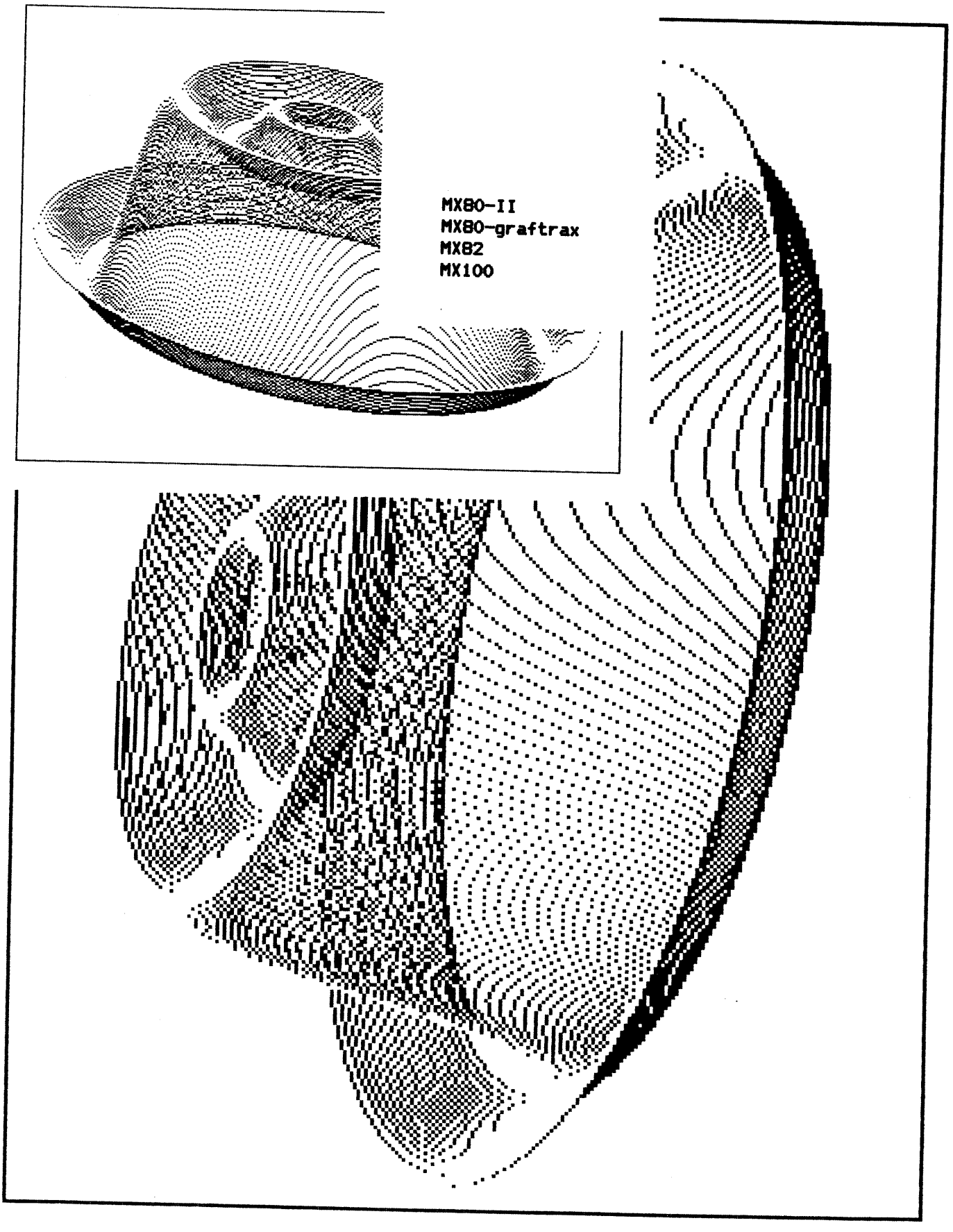
```
249 04D4 1D                 DCR   E          $
250 04D5 C2C404             JNZ   LOOP       $
251 04D8 23                 INX   H          NEXT CHARACTER
252 04D9 C3BB04             JMP   NEXTCH
253 04DC            LENGTH  END

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

BACKSP 042D   BAUD   FFF5   BDRATE 0084   CHCKBS 0448
CHCKNU 0442   CURPOS 0072   EDIT   04B5   FLAG   0400
INCHAR FFF0   INTREC 0438   LENGTH 04DC   LOOP   04C4
MAINLP 040F   NCTRLC 0466   NEXT   0477   NEXTCH 04BB
OUT    04B0   OUT1   048D   OUT2   0497   OUT3   04A1
OUT4   04AB   OUTSCR 0131   RESFLT 046E   SCREEN 044D
SEROUT 0482   START  0401   STSREG FFF3   TEST   04C6
TRANSC 0451
```

LAYOUT MATRIX



JUMBO-SCREENCOPY EN SCREENCOPY OP MX-82

Herman MOEYS
Pieter Nollekensstraat 109
B-3200 KESSEL-LO (LEUVEN)



Microbeam

Description :

MICROBEAM is a scientific program designed for solving structural problems involving beams (concrete, metal, wood, ...).

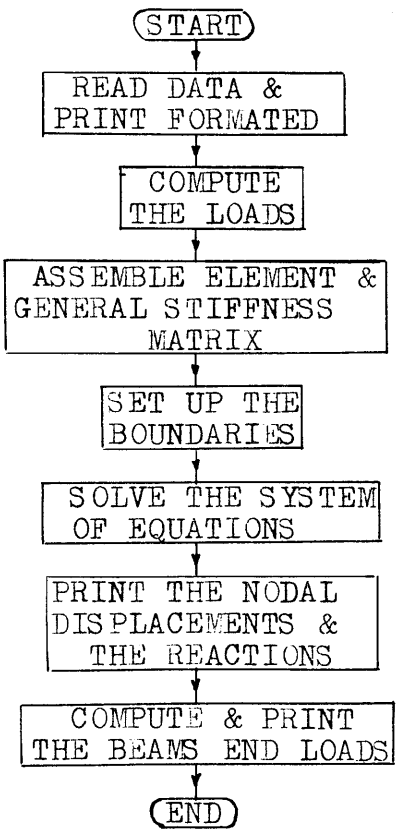
Its formulation is based on a mixture of Finite Element Method and Displacement Method, both well known by civil or structural Engineers.

Powerfull matrix compression algorithms allow our DAI computer to solve in a few minutes medium-size beams problems that would require hours to be solved by hand (when possible !).

The actual version is a bidimensional solver and has been tested with examples and solutions given by other microcomputer programs as well as by very large scale mainframes. The differences in the results do never exceed 1/100 and are usually in the order of 1/1000 !

The program in this actual version is a real professional tool and can be of great interest for civil engineers, building designers, architects,... It will be constantly upgraded to reflect new desires as well as more performing methods.

Software scheme :



Input and output :

All data input must be given as 'DATA' lines, starting at line 60000. This prevents the need of retyping the whole data set in case of error, as our DAI editor can be used for corrections.

The user must provide :

- the nodal coordinates in an X,Y system
- the beams connectivity and their family reference
- the beam families data (E, I, , Spec. Weight)
- the nodal restraints (nodes fixed)
- the loads applied (see below)

A plot of the geometry can be obtained to ensure the correctness of the coordinates and the connectivity, using the high-resolution graphics and a machine-language routine.

As output, the user will receive :

- the nodal displacements (Ux, Uy, Rotation)
- the reactions at the fixed nodes (Rx, Ry, Bending)
- the beams loads (Normal, Bending & Shear)

Under development :

- multi load cases problem solution
- restart with saved stiffness matrix
- more beam loading capabilities
- post-processor for deformed shape plots and beams diagrams (M, N, T)
- ...

Important note :

One should note that the program is running without any peripherals and thus can be used with the minimal configuration.

The resulting resolution times given in the examples are obtained without the Math chip.

A disc version should be started soon.

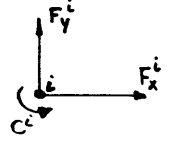
Informations :

For any informations about the 'MICROBEAM' software, please feel free to contact

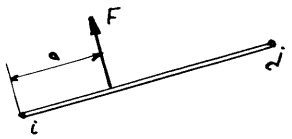
ir P. H. DAL
98, Bd E. MACHTENS, #29
1080 BRUSSELS
BELGIUM
Tel : 02/5228293

Library of loads :

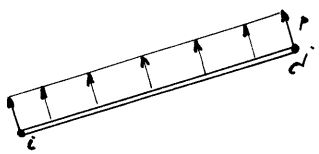
1. Nodal load : two forces and one couple applied to a node.



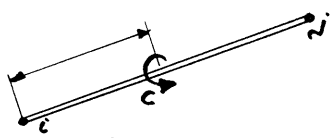
2. Concentrated force, normal : applied at distance 'a' from end 1.



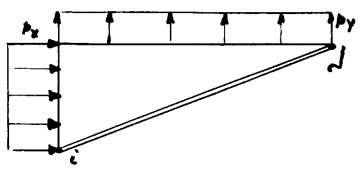
3. Constant pressure, normal : ---



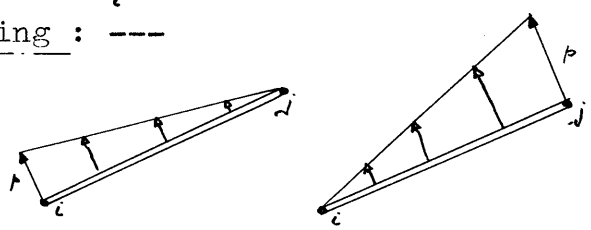
4. Concentrated couple : applied at distance 'a' from end 1.



5. Horizontal + Vertical uniform pressure : as wind, snow,...



6. Triangular loading : ---

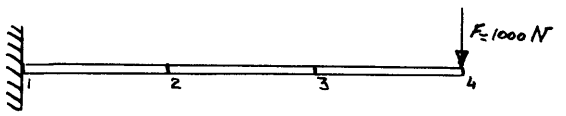


Note that combinations of these types can be realised for any beam.

Many examples have been run to ensure the quality of the software and none gave any problem.

Examples :

1. Clamped beam with vertical force at free end :



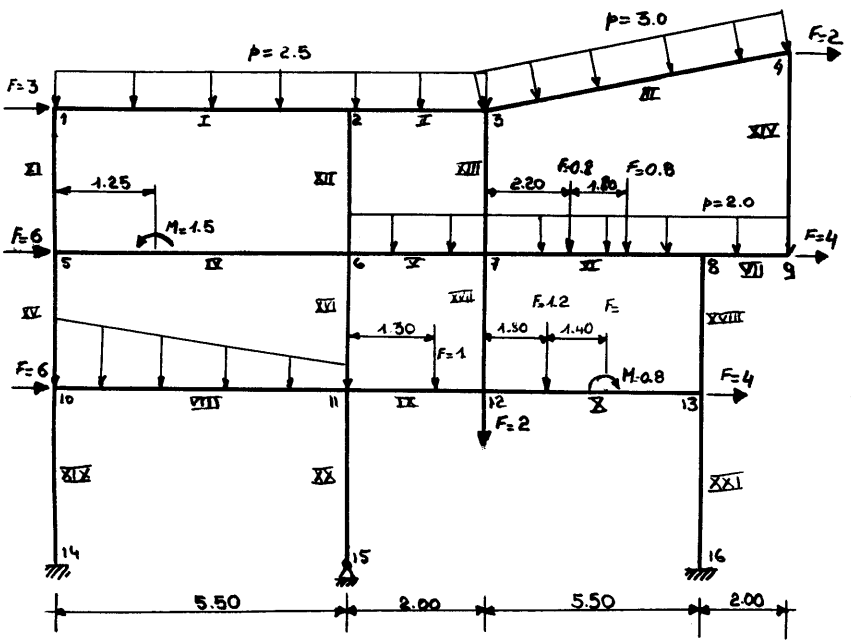
4 nodes
3 beams
1 beam family
1 nodal load

$E=21E10\ N/m^2$; $\Omega=0.02\ m^2$; $I=8.33E-5\ m^4$; $L=3\ m$

DAI/MICROBEAM solution match the theoretical solution and the computed solution from a commercial large-scale soft with a difference less than 5/10000.

Resolution time (from RUN to END OF PROGRAM) : 15 sec !

2. Hyperstatic beam assembly :



16 nodes
21 beams
5 beam fam.
6 nodal loads
15 beam loads

$E=3E6\ T/m^2$

$\Omega_1=.20\ m^2$
$\Omega_2=.24$ "
$\Omega_3=.28$ "
$\Omega_4=.32$ "
$\Omega_5=.16$ "

$I_1=.00417\ m^4$
$I_2=.00720$ "
$I_3=.01143$ "
$I_4=.01707$ "
$I_5=.00213$ "

Fam. 1 : beams 1 to 3
2 : beams 4 & 5, 8
3 : beams 6 & 7
4 : beams 9 & 10
5 : beams 11 to 21

The results are of course too long to be given here. Anyway, the MICROBEAM solution is 100% compatible with the one given by a professional soft tailored for the HP-85 computer (HEWLETT-PACKARD) and priced at BF 70000.- (tax excluded !!!).

Resolution time (same remark as above) : 4min 15 sec !

DNA ALTERNATIVE PRINTER ROUTINE

Voici les modifications a aPPorter a DNA (DAINAMIC assembler) Pour utiliser l'alternate Printer routine de Jan Boerri9ter (DAINAMIC No. 10 - Pa9es 83/84) via la commande]H. de l'assembleur

1. Char9er l'assembleur

2. >G1100
   Dimensionner les buffers de l'assembleur (valeur au choix)
   Revenir au basic via]B.

3. Rentrer le Pro9ramme suivant en machine (ensuite RUN) :

```
10 REM Ajustement des Pointeurs Pour buffers
20 FOR X=#2C00 TO #2FFF
30 IF PEEK(X)=0 AND PEEK(X+1)=#30 THEN POKE X,#50
40 NEXT
```

4. TaPer :
   *UT
   >S1259 00-01
   >G1100
   Dimensionner a nouveau les buffers (valeur au choix)
   Vous aurez a ce moment :
   START HEAP xxxx+50
   Revenir au basic via]B.

5. Rentrer le Pro9ramme suivant en machine (ensuite RUN) :

```
10 FOR X=#115F TO #1161
20 READ A:POKE X,A:NEXT
30 FOR X=#2C76 TO #2C78
40 READ A:POKE X,A:NEXT
50 FOR X=#3000 TO #3040
60 READ A:POKE X,A:NEXT
100 DATA #CD,#00,#30
110 DATA #CD,#3B,#30
120 DATA #E5,#21,#05,#FF,#36,#88,#21,#DD,#02,#36,#C3,#21,#13,#30,#22,#DE
130 DATA #02,#E1,#C9,#F5,#EF,#03,#3A,#00,#FD,#E6,#08,#CA,#16,#30,#3A,#F3
140 DATA #FF,#E6,#10,#CA,#1E,#30,#F1,#32,#F6,#FF,#FE,#0D,#C0,#F5,#3E,#0A
150 DATA #CD,#13,#30,#CD,#41,#DE,#CD,#41,#DE,#F1,#C9,#1F,#1F,#3F,#17,#17,#C9
```

6. Les modifications etant introduites,Prendre une coPie de la nouvelle version assembleur :
   TaPer :
   *UT
   >S1259 00-01
   >W1100 3050

   REM. : la valeur en #3005 Peut etre modfiee selon le baud rate de de l'imPrimante
   #3036 a #3038 Peuvent etre remPlaces Par #00 si le delai de #3033 a #3035 est suffisant

JOEL MENIER
PLACE DE VANNES,17
7000 MONS
BELGIQUE

ALTERNATIVE PRINTER ROUTINE

```
001                 *   DNA MODIFCATIONS FOR USING
002                 *   THE ALTERNATIVE PRINTER ROUTINE
003                 *   (DAINAMIC NR.10 P.83/84)
004                         ORG   :115F
005 115F CD0030             CALL  :3000
006                         ORG   :2C76
007 2C76 CD3830             CALL  :3038
008                         ORG   :3000
009 3000 E5                 PUSH  H
010 3001 2105FF             LXI   H,:FF05   BAUD RATE BYTE
011 3004 3688               MVI   M,:88     ACCORDING TO BAUD RATE
012 3006 21DD02             LXI   H,:2DD    ]
013 3009 36C3               MVI   M,:C3     ] SET DOUTC FOR
014 300B 211330             LXI   H,RS232   ] REPLACEMENT
015 300E 22DE02             SHLD  :2DE      ] ROUTINE
016 3011 E1                 POP   H
017 3012 C9                 RET
018 3013 F5         RS232   PUSH  PSW
019 3014 EF                 RST   5         OUTPUT TO SCREEN
020 3015 03                 DATA  :03
021 3016 3A00FD     OUTRDY  LDA   :FD00
022 3019 E608               ANI   :08
023 301B CA1630             JZ    OUTRDY    WAIT OUTPUT READY
024 301E 3AF3FF     BUFEMP  LDA   :FFF3
025 3021 E610               ANI   :10
026 3023 CA1E30             JZ    BUFEMP    WAIT BUFFER EMPTY
027 3026 F1                 POP   PSW
028 3027 32F6FF             STA   :FFF6     OUTPUT CHAR
029 302A FE0D               CPI   :0D       CAR. RET ?
030 302C C0                 RNZ             READY WHEN NOT
031 302D F5                 PUSH  PSW
032 302E 3E0A               MVI   A,:A
033 3030 CD1330             CALL  RS232     PRINT LINE FEED
034 3033 CD41DE             CALL  :DE41     DELAY
035 3036 CD41DE             CALL  :DE41
036 3039 F1                 POP   PSW
037 303A C9                 RET
038 303B 1F                 RAR             ]
039 303C 1F                 RAR             ] IF #131=1 THEN #131=3
040 303D 3F                 CMC             ]         OR
041 303E 17                 RAL             ] IF #131=3 THEN #131=1
042 303F 17                 RAL             ]
043 3040 C9                 RET
044 3041                    END
```

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

BUFEMP 301E   OUTRDY 3016   RS232  3013
```

```
] ?
]H.
]#P.
115F CD
1160 00 30

2C76 CD 3B 30

3000 E5 21 05 FF 36 88 21 DD 02 36 C3 21 13 30 22 DE
3010 02 E1 C9 F5 EF 03 3A 00 FD E6 08 CA 16 30 3A F3
3020 FF E6 10 CA 1E 30 F1 32 F6 FF FE 0D C0 F5 3E 0A
3030 CD 13 30 CD 41 DE CD 41 DE F1 C9 1F 1F 3F 17 17
3040 C9
```

INSIDE A LOOP

WHAT CAN HAPPEN INSIDE A LOOP ?

This article describes the way a Basic loop is executed by the DAI Basic, and a particular problem which may occur.

Lets have a look at a very simple loop in a Basic program:

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT
```

When line 10 is executed, the following happens:
a. The variable I is set to the value 1.
b. The address of the variable I in the symboltable is stored in the pointer LOPVAR (#0104/05).
c. From the Basic expression is calculated how many times the loop has to be executed. This value is stored in the pointer LCOUNT (#010B-#010E) in a 4 byte INT or FPT notation.

Now line 20 is executed. When the NEXT statement in line 30 is encountered, the value of the loopvariable I is incremented and the value of LCOUNT is decremented. As long as the latter is not 0, line 20 is executed again.

Lets now look at the following routine:

```
10 FOR I=1 TO 10
20 PRINT I
30 IF I=5 THEN I=I+5
40 NEXT
```

Execution of this routine gives a print out for I:
1,2,3,4,5,11,12,13,14,15
What happens ?

In line 10, the LOPVAR and LCOUNT pointers are set. During execution of the loop, LCOUNT is only decremented by the NEXT statement.
So when in line 30 the value of I is changed to I+5, the value of the variable I in the symboltable (address in LOPVAR) is updated, but not the value of LCOUNT!!! The loop will be executed 10 times, irrespective the momentary value of the loopvariable I (the FOR statement is only executed once to determine LCOUNT!!).

How to avoid this particular DAI-Basic problem? There are 3 possiblities:
a. Avoid the change of a loopvariable during loop execution.
b. Update LCOUNT by means of POKE routines.
c. Check the value of the loopvariabe before the NEXT instruction is is performed.

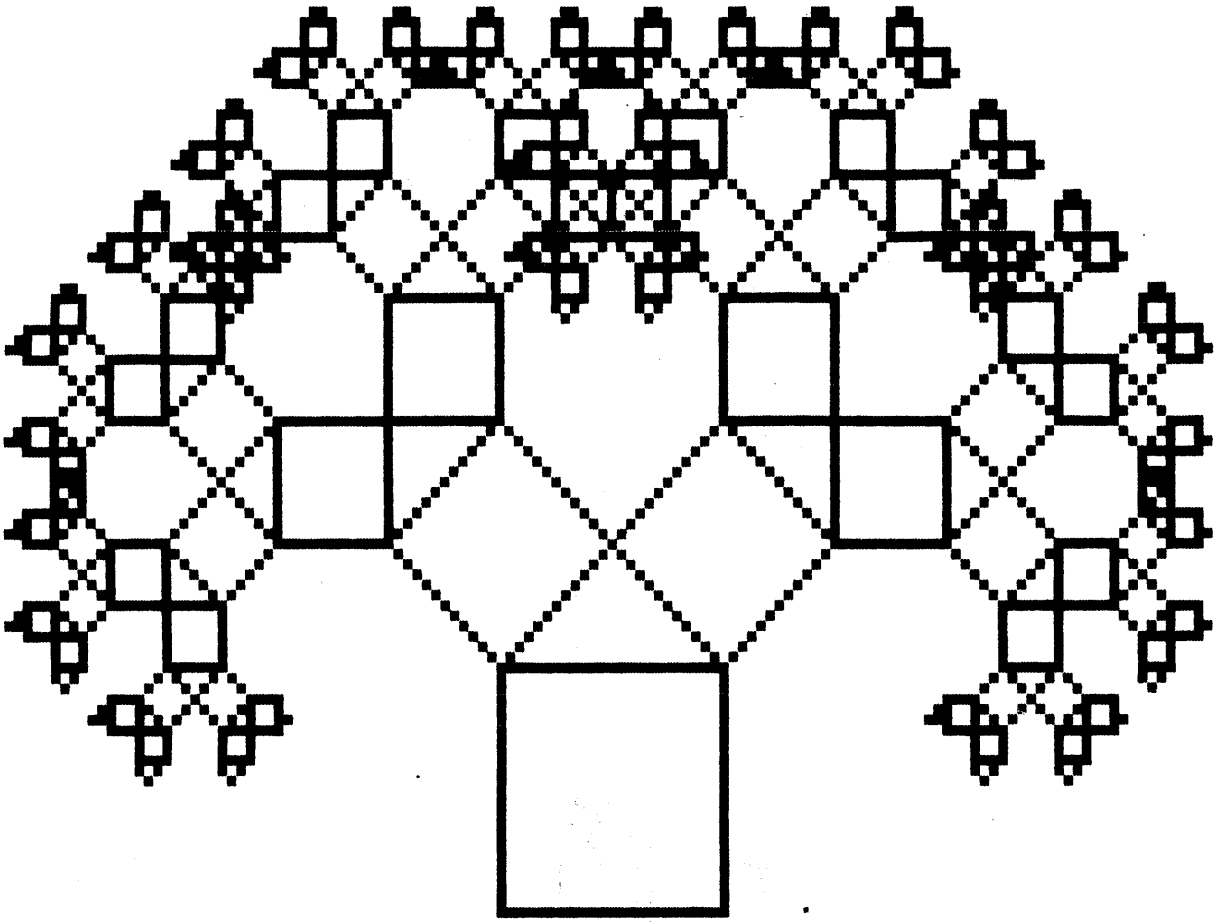
The first solution is undoubtly the best. But when it is unavoidable to change the value of a loopvariable during execution of a loop, solution b or c has to be used.
Solution b is possible but not practical. The value of LCOUNT is stored in 4 bytes in INT or FPT notation. Too many PEEK's and POKES's are required.
Solution c will be:

```
35 IF I >=10 GOTO 50
```

This seems the most suitable solution.

Jan Boerrigter

BOOM VAN PYTHAGORAS



```
4     REM IMPINT
5     PRINT "     *** DE BOOM VAN PYTHAGORAAS ***"
6     PRINT "Programma van Frits Regtien, leerling van het"
7     PRINT "Gymnasium St Willebrord te Deurne (NL), juni 1982"
10    Z=3000:GOTO 1000:REM 3000 IS STARTADRES
20    IF S<0 THEN S=4+S
30    IF S>3 THEN S=-4+S
40    DRAW A,B C,D 23:ON S GOTO 200,300,400
100   E=D+C-A:AC2=(A+C)/2:EEB2=E+(E-B)/2
110   DRAW C,D C,E 23:DRAW C,E A,E 23:DRAW A,E A,B 23
120   DRAW A,E AC2,EEB2 23:DRAW AC2,EEB2 C,E 23
130   POKE U,S:POKE U+1,A:POKE U+2,E:POKE U+3,AC2
140   POKE U+4,EEB2:POKE U+5,C:POKE U+6,E:U=U+7:RETURN
200   DDB=D+D-B:ACA=A-C+A
210   DRAW C,D A,DDB 23:DRAW A,DDB ACA,D 23:DRAW ACA,D A,B 23
220   DRAW ACA,D ACA,DDB 23:DRAW ACA,DDB A,DDB 23
230   POKE U,S:POKE U+1,ACA:POKE U+2,D:POKE U+3,ACA
240   POKE U+4,DDB:POKE U+5,A:POKE U+6,DDB:U=U+7:RETURN
300   E=C-D+B:EAE2=E-(A-E)/2:BD2=(B+D)/2
310   DRAW C,D E,D 23:DRAW E,D E,B 23:DRAW E,B A,B 23
320   DRAW E,B EAE2,BD2 23:DRAW EAE2,BD2 E,D 23
330   POKE U,S:POKE U+1,E:POKE U+2,B:POKE U+3,EAE2
340   POKE U+4,BD2:POKE U+5,E:POKE U+6,D:U=U+7:RETURN
400   CCA=C+C-A:BBD=B+B-D
410   DRAW C,D CCA,B 23:DRAW CCA,B C,BBD 23:DRAW C,BBD A,B 23
420   DRAW CCA,B CCA,BBD 23:DRAW CCA,BBD C,BBD 23
430   POKE U,S:POKE U+1,C:POKE U+2,BBD:POKE U+3,CCA
440   POKE U+4,BBD:POKE U+5,CCA:POKE U+6,B:U=U+7:RETURN
1000  MODE 6A:COLORT 6 11 0 0:P=Z:U=Z
1010  S=0:A=130:B=0:C=162:D=0:GOSUB 20:GOTO 2000
1020  Y=P:P=U:IF P>8000 GOTO 2100
1030  FOR G=Y TO U-1 STEP 7
1040  S=PEEK(G)+1:A=PEEK(G+1):B=PEEK(G+2):C=PEEK(G+3):D=PEEK(G+4):GOSUB 20
1050  S=PEEK(G)-1:A=C:B=D:C=PEEK(G+5):D=PEEK(G+6):GOSUB 20:NEXT
2000  WAIT TIME 100:GOTO 1020
2100  WAIT TIME 500:GOTO 1000
```

VARIABLE CURSOR

```
10    REM VARIABLE CURSOR / F.H. DRUIJFF 1/82
20    MODE 0:PRINT CHR$(12);
30    PRINT "VARIABLE CURSOR CHOICE."
40    PRINT "======================"
50    PRINT :PRINT "THIS PROGRAM WILL SHOW THAT YOU CAN FREELY CHOOSE YOUR OWN"
60    PRINT "CURSOR-SYMBOL."
70    PRINT "THE CURSOR-SYMBOL (THE LITTLE BLINKING LINE) IS SITUATED ON"
80    PRINT "ADDRESS #75 (117 IF YOU PREFER DECIMAL) IN MEMORY AND CAN BE"
90    PRINT "CHANGED IN JUST THE CHARACTER YOU WANT."
100   PRINT "YOU 'POKE #75,xx' WITH xx BEING A NUMBER IN THE RANGE 0-255"
110   PRINT "AND YOU WILL HAVE A NEW CURSOR BLINKING ON THE SCREEN."
120   PRINT :PRINT "BUT THERE IS MORE."
130   PRINT "ON ADDRESS #74 (116 DECIMAL) IS NORMAL 1. IF YOU CHANGE THIS"
140   PRINT "IN 0 THE CURSOR WILL NOT HAVE THE NORMAL COLORS BUT ALTER-"
150   PRINT "NATES IN COLOR THREE AND FOUR FROM THE 'COLORT' INSTRUCTION."
160   PRINT "TO SHOW THIS AND TO LET YOU APPRECIATE IT WE ASK YOU TO DO"
170   PRINT "SOME WORK. PLEASE FOLLOW INSTRUCTIONS."
180   PRINT SPC(10);"1 - TYPE: 'RUN300' FOLLOWED BY RETURN."
190   PRINT SPC(10);"2 - TYPE: 'EDIT300-390' AGAIN RETURN."
200   PRINT SPC(10);"3 - PLAY AROUND WITH CURSORKEYS (GREY KEYS)"
210   PRINT SPC(10);"     IN COMBINATION WITH REPT-KEY."
220   PRINT SPC(10);"4 - RETURN BY PRESSING 'BREAK' TWICE"
230   PRINT "5 - REPEAT FOR 400,500,...,900 SERIES.    ";:GOTO 999
300   POKE #74,1:POKE #75,1
310   COLORT 8 0 0 0
320   END
400   POKE #74,0:POKE #75,255
410   COLORT 8 0 0 8
420   END
500   POKE #74,1:POKE #75,65
510   COLORT 8 0 0 0
520   END
600   POKE 116,0:POKE 117,255
610   COLORT 8 0 15 0
620   END
700   POKE #74,0:POKE #75,255
710   COLORT 8 0 8 15
720   END
800   POKE #74,0:POKE #75,255
810   COLORT 8 0 3 14
820   END
900   POKE #74,0:POKE #75,#3F
910   COLORT 8 0 1 15
920   END
999   REM EINDE
```

RECTANGLES

```
10    REM RECTANGLES  /  F.H. DRUIJFF 2/82
20    MODE 4:COLORG 0 14 1 3:GOTO 50
30    DRAW X,Y 159-X,Y C:DRAW 159-X,Y 159-X,129-Y C
40    DRAW 159-X,129-Y X,129-Y C:DRAW X,129-Y X,Y C:RETURN
50    FOR I=0 TO 20:X=80-3*I:Y=65-2*I:C=21:GOSUB 30
60    F=1-F:C=22+F:FOR J=0 TO 20:X=80-2*J:Y=65-3*J
70    GOSUB 30:NEXT:NEXT
80    FOR J=0 TO 20:X=80-2*J:Y=65-3*J:F=1-F:C=22+F:GOSUB 30:NEXT:GOTO 80
```