DAI NAMIC

25

tweemaandelijks tijdschrift november - december 1984

personal computer users club

een uitgave van dainamic v.z.w.
verantw. uitgever w. hermans, mottaart 20 - 3170 herselt

International

DAINAMIC
PERSONAL COMPUTER USERS CLUB

COLOFON

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

DAInamic redactie :

Dirk Bonné	wdw
Freddy De Raedt	Herman Bellekens
Wilfried Hermans	Frans Couwberghs
René Rens	Guido Govaerts
Bruno Van Rompaey	Daniël Govaerts
Jef Verwimp	Frank Druijff
Cedric Dufour	Willy Coremans

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
Mottaart 20
3170 Herselt
Tel. 014/54 59 74

Kredietbank Herselt
nr. 401-1009701-46
BTW : 420.840.834

Lidgelden / Subscriptions

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

Generale
Bankmaatschappij
Leuven
nr. 230-0045353-74

Voor Nederland :
GIRO : 4083817
t.n.v. J.F. van Dunne
Hoflaan 70
3062 JJ ROTTERDAM
Tel. : (010) 144802

Pour la France :
DAInamic FRANCE
C. Dufour
Rue Lavoisier 9
59149 DUNKERQUE
Tel. 02866 3339

Inzendingen : Games & Strategy

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

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

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

DAInamic

NEWSLETTER 25

Herselt, december '84

Beste Leden,

Met een extra dik nummer (88 pagina's + mini-poster) hebben we deze 25-ste uitgave iets feestelijker willen maken. We zijn ervan overtuigd dat er genoeg materiaal in zit om heel actief de feestdagen door te brengen.

Een terugblik op de voorbije manifestaties :

20 october Nijvel : een 250-tal bezoekers bevestigden dat het een goed idee was om deze dag te organiseren. We mochten een dertigtal nieuwe leden noteren en verschillende DAI-gebruikers maakten voor de eerste keer kennis met onze uitgebreide software-bibliotheek. Onze gelukwensen aan de initiatiefnemers C. Poels en F. Duluins.

17 november Utrecht : groot, druk, onoverzichtelijk... we hebben 's morgens een half uurtje moeten zoeken om onze eigen kramen te vinden ! We hopen dat U ons toch heeft kunnen vinden. Zoals in Nijvel was er veel belangstelling voor onze experimenten met de video camera interface, ook vanwege niet-DAI gebruikers.

24 november Diepenbeek : Het onderwijssymposium georganiseerd door SCHOOL en COMPUTER kende een groot succes. Onze DAI computer was ruim vertegenwoordigd : in de software catalogus vonden wij demonstraties door : Marc Antrop, Bruno Van Rompaey, INDATA, TRON, J. Mergaerts, M. De Jaeger, F. Cerulus en ondergetekende.

Ondanks de aangekondigde posttarief verhogingen kunnen wij de contributie ongewijzigd houden
Subscription prices will be the same in 1985 :

Benelux : 1000 Bfr — Europe : 1100 Bfr — Air Mail : 1500 Bfr

Collective subscriptions (for schools only) :

2 and more subscriptions on the same address and paid together : - 30 % from the second subscription on.

voor België en andere landen :
DAInamic subscriptions
B. Van Rompaey
Bovenbosstraat 4
B-3044 HAASRODE
Tel. (016) 46.10.85
Bancaccount : 230-0045353-74 of Gen. Bank Leuven

voor Nederland :
J.F. Van Dunne
Hoflaan 70
NL-3062 JJ ROTTERDAM
Tel. (010) 144802
giro : 4083817

pour la France :
DAInamic FRANCE
C. Dufour
Rue Lavoisier 9
59149 DUNKERQUE
Tel. 02866 3339

Noot : Maakt U zich geen zorgen om wisselkoersen, U kan gerust 1000 Belgische Franken overschrijven via de Nederlandse giro, F. Druijff heeft het voor ons uitgeprobeerd.

NEW SOFTWARE :

BOEKHOUDPROGRAMMA :	audio2000 Bfr DCR2150 Bfr
DBASIC V2.2 + ext. update :	250 Bfr
FYSISCHE GEOGRAFIE :	audio1000 Bfr DCR1150 Bfr

SOON AVAILABLE :

COMPILATION 83-84 : We don't know yet how many tapes it will take (DCR), but as this collection will offer programs from Newsletter 16 to Newsletter 25, it will be a lot of software !

SUPER-BASE : At last a 100 % machine-language database program, using the total amount of RAM to hold up to 1000 records, depending on the field-size. Ultrafast sorting, build-in mail-merge, formatting for all kind of labels and so on...

PATROUILLER : The latest creation of P. Janin, if you ever saw PHOENIX, you know what we are talking about... guide your CAR on the surface of the moon and encounter a lot of missions to complete your race...

The first issue of 1985 will deal specially with DCE-bus and other I/O projects. If you have some projects ready, please send description to be published in Newsletter 26.

We invite you to dive into this jubilee-issue, till next year,

W. Hermans

PROGRAMMEERTECHNIEKEN

Structuren in basicprogramma's zijn het onderwerp van deze keer. Langzaamaan begint ook basic steeds meer instructies te kennen die niet echt noodzakelijk zijn, maar die programmeren direct of later prettiger maken. Met instructies die direct het programmeren prettiger maken bedoel ik bv FOR-NEXT, WHILE-WEND, REPEAT-UNTIL. Instructies die vooral later hun vruchten afwerpen als U het programma nogeens doorneemt om er een paar kleine veranderingen in aan te brengen zijn bv PROCEDURE en FUNCTION en labels. Deze mogelijkheden zitten jammer genoeg niet alle in de standaard DAI-basic. Maar gelukkig brengt D-BASIC daar nu verandering in. Hiermee verandert de basic in een soort PASCAL. Sommige fabrikanten hebben een soortgelijke BASIC al verheven tot nieuwe taal nl COMAL.

Maar laten we bij het begin beginnen en eerst de loops of lussen bekijken. De naamgeving is duidelijk afkomstig uit de tijd dat er alleen nog maar in machinetaal of assembler werd geprogrammeerd. Het is als volgt te zien : als je het programma stap voor stap volgt met een vingertje langs de listing, je vaststelt dat er programmadelen zijn die meermalen worden uitgevoerd. Heb je dan een pen in de hand zie je vanzelf een lus (in het engels loop) ontstaan. Vaak wordt het dan een volledige warboel. Zeker als er meerdere loops in elkaar en helemaal door elkaar zitten. De structuur moet dan geheel van de programmeur komen. Maar om in minimum basic dus zelfs zonder FOR-NEXT een loop op te zetten zijn er vele methodes. Eens kwam ik een artikel tegen met de kop '24 ways to program a loop'. Wat zijn deze methodes dan en waarin verschillen ze van de andere ? De body bestaat uit een of meer instructies die een zeker aantal malen (N) uitgevoerd dienen te worden. Er zijn een paar principiele verschillen aan te wijzen : de teller die het aantal bijhoudt telt op tot N of telt af vanaf N. De teller telt tot N of tot en met N; bij dalen tot 0 of tot en met 0 De controle, of de grens reeds bereikt is, wordt vooraf gedaan of achteraf.

Een aantal voorbeelden :

```
   -A-                    -B-                    -C-                    -D-
10 T=0                 10 T=1                 10 T=0                 10 T=0
20 REM begin           20 REM begin           20 T=T+1               20 IF T=N GOTO 90
-- body                -- body                30 IF T>N GOTO 90      30 REM begin
50 REM eind            50 REM eind            40 REM begin           -- body
60 T=T+1               60 T=T+1               -- body                60 REM eind
70 IF T<N GOTO 20      70 IF T<=N GOTO 20     70 REM eind            70 T=T+1
                                              80 GOTO 20             80 GOTO 20
                                              90 REM vervolg         90 REM vervolg
```

Op het eerste gezicht weinig verschil totdat we ons bedenken hoevaak de body nu wordt uitgevoerd. Mijn leerlingen krijgen ALTIJD deze structuur te leren en dan later pas bv FOR-NEXT. Ik vraag ook altijd hoeveel zij eronder durven te verwedden dat de body inderdaad N maal is uitgevoerd en niet N-1 of N+1 maal. Een programma, waarvan U niet zeker weet dat het goed werkt, is nog slechter dan een programma, dat niet werkt. Dat laatste geeft geen antwoorden maar het eerste misschien foute antwoorden. Controleer de werking vooraf met kleine waarden voor N, zodat U dat zelf eenvoudig kunt controleren . Vervang de body daartoe door bv 'PRINT "TEST ";T' en neem voor N een getal onder de tien. Nu de verschillen : Bij de versies A en B zal de loop minimaal eenmaal doorlopen worden, ook als N kleiner dan een is en bij de versies C en D zal dit niet gebeuren. Daar wordt de loop (bij niet-negatieve N tenminste) precies N maal doorlopen. In dit opzicht hebben C en D dus de voorkeur boven A en B. Maar er zijn meer regels gebruikt en ook meer GOTO's en dat maakt het programma niet overzichtelijker. Wel is nu het begin en het einde van de loop gemarkeerd met

een GOTO. Bij D kunnen we trouwens door de test op gelijkheid met N grandioos de mist ingaan als N en/of T een breukdeel bevatten. Een ander verschil tussen C en D zit in de plaats van de verhoging van T. Bij C wordt er altijd verhoogt en bij D alleen als de loop daadwerkelijk doorlopen wordt. Merk op dat de verhoging van T ook voor de body, maar na de test, geplaatst kan worden. Bij A en B is het voornaamste verschil in de waarde van T tijdens het uitvoeren van de body. Wordt de waarde van T in de body gebruikt is dat dus een criterium om te kiezen tussen A en B. Andere varianten zijn nog te bedenken: N van 11 t/m 40 of van -10 t/m 20. (Klopt dit wel ?) In plaats van T van 1 t/m 30 te laten gaan hem van 10 t/m 300 te laten gaan in stappen van tien in plaats van een. Merk op dat onze taal bij 10 t/m 300 wel duidelijk is in het er bij horen van 300 maar niet met het er bij horen van 10.

De verschillen komen neer op de volgende zaken :

1) vanaf startwaarde of van en met startwaarde
2) tot eindwaarde of tot en met eindwaarde
3) eerst uitvoeren dan controle of eerst controle en dan eventueel uitvoeren
4) verhoging voor controle of verhoging na controle
5) verhoging voor uitvoering body of er na
6) controle op gelijkheid (gevaarlijk !) of ongelijkheid
7) controle op < of <= ;respectievelijk > of >=
8) T is zuivere loopteller (dwz telt de doorgangen) of is reeds voorbewerkt voor gebruik in de body.

Alle combinaties hiervan leveren dus meer dan de al genoemde 24 mogelijkheden. We komen nu toe aan de FOR-NEXT instructie . Dit programmeert vanzelfsprekend veel simpeler en soms ook eleganter. We hebben geen GOTO meer nodig dus wordt het geheel overzichtelijker. MAAR als U geen antwoord weet te geven op de volgende vragen mag u hem niet gebruiken !!!! Hoe is de volgorde bij de FOR-NEXT eerst controle of eerst uitvoeren van de body ? Eerst de body en dan de verhoging of net omgekeerd ? Eerst de controle of eerst de verhoging ? Als het U lastig valt op zulke theoretische vragen te antwoorden kijk dan naar de practische vragen aan de hand van de volgende voorbeelden.

```
10 FOR I=0 TO 20:...........:NEXT    wat is de waarde van I na regel 10 ?
20 FOR I=5 TO 12:...........:NEXT    hoe vaak wordt de body uitgevoerd ?
30 FOR I=6 TO 6:............:NEXT    hoe vaak wordt de body uitgevoerd ?
40 FOR I=8 TO 5:............:NEXT    hoe vaak wordt de body uitgevoerd ?
```

Moeilijker :

```
50 FOR I=1 TO 20:........:I=I+1:NEXT  hoe vaak wordt de body uitgevoerd ?
60 FOR I=1 TO 10:........:I=I-1:NEXT  hoe vaak wordt de body uitgevoerd ?
```

Nog lastiger :

```
Wat is het verschil tussen FOR I%=0 TO 6 en FOR I!=0 TO 6 ?
Wat is het verschil tussen FOR I%=0 TO 6 EN FOR I%=0.0 TO 6.0 ?
Wat is het verschil tussen eindigen met NEXT en NEXT I ?
```

En tot slot de vragen waar bijna niemand correct antwoord op blijkt te kunnen geven: Hoeveel niveaus van FOR-NEXT kent de DAI ? Mag men straffeloos uit en FOR-NEXT loop springen ?

We zullen een aantal van de hierboven beschreven gevallen bespreken. Niet alle want de eerste vier kunt U vanzelfsprekend zelf intikken en dan controleren. Foei als het nodig is en driewerf foei als U het niet zeker weet en dan toch niet controleert. De problemen met regelnummers 50 en 60 zijn lastiger. De DAI bepaalt aan het begin van de loop hoeveel maal deze doorlopen moet worden. Hij

doet dit door het verschil eindwaarde min beginwaarde te delen door de grootte van de stap. Is de stapgrootte 0 krijgen we de foutmelding 'DIVISION BY ZERO'. Een interne (integer) teller houdt het aantal doorgangen van de loop bij. Vervolgens wordt de body eenmaal uitgevoerd en dan zowel de interne teller als de door de programmeur opgegeven variabele verhoogd (of verlaagd) . Veranderen we in de body dus de eigen teller (loopcounter) zal deze wel veranderen, maar het aantal malen dat de loop doorlopen wordt, wordt met de interne teller bijgehouden en zal onveranderd blijven . Het resultaat is dat bij regel 50 de body 20 maal wordt uitgevoerd en I via de oneven getallen uitkomt op 41. Bij regel 60 zal de body 10 maal uitgevoerd worden en I op 1 blijven. (steeds een er af en een er af aan het eind) Het verschil tussen gebruik van een integer teller I% of een floating point I! is alleen de snelheid van werken, vooral te merken met kleine body's. Opgelet trouwens bij het werken met integertellers in dit verband : FOR I%=20 TO 30 STEP .5:.....:NEXT geeft de foutmelding 'DIVISION BY ZERO', evenals de berekening van de stapgroottevariabele, die een breukdeel oplevert. Bij integervariabelen worden die dan afgerond; wordt het 0 krjgen we een foutmelding, maar in de andere gevallen een onjuist werkend programma . Ik heb geen verschil kunnen vinden in het gebruik van NEXT en NEXT I.

Het aantal niveaus dat de DAI aankan is wisselend en dit veroorzaakt dan ook een aantal vreemde zaken. Om te onthouden waar de FOR-NEXT body begint zet de DAI een aantal gegevens op een speciale plaats in het geheugen (de stack) Deze stack wordt echter ook gebruikt door de ROM-routines en bv de interrupt routines. Tik ter testing het volgende programma in :

```
10  FOR A=0 TO 2
20  FOR B=0 TO 2
30  FOR C=0 TO 2          Niet erg dat er geen NEXT'n staan
::  ::: ::: :: :
::  ::: ::: :: :
140 FOR N=0 TO 2          het programma moet ontsporen voor de test
150 FOR O=0 TO 2
160 FOR P=0 TO 2
```

U zult zien dat U na RUN bijna altijd de melding 'STACK OVERFLOW IN LINE 160'. Voegt U aan het programma toe 155 PRINT "ufghtf" komt U al bij 155 op stack overflow. Vaak intikken van RUN levert ook soms een stack overflow op regels met lagere nummers. Bij mij 140 als minimum. Dit verschijnsel is te verklaren door het gebruik van de stack door ROM-routines en de interrupt handling. Bv zal de 20ms interrupt (en de cursor-'interrupt' als onderdeel daarvan) niet altijd op hezelfde moment in het programma komen. Het maximum is dus bij een normale stack 15 niveaus, maar het blijft gevaarlijk zo diep te gaan , daar op andere plaatsen dit problemen met stackruimte kan geven. Heeft het programma zelf al subroutines (GOSUB) levert het helemaal vrijwel zeker problemen op. Een aardige tip voor luie (= slimme ?) programmeurs : de teller mag best een arrayvariabele zijn. dus kan de gewenste informatie als volgt gevonden worden.

```
10 DIM T(20)
20 FOR T(I)=0 TO 2:I=I+1:GOTO 20
```

RUN geeft zoals verwacht 'STACK OVERFLOW IN LINE 20' en PRINT I levert 14 op.

En dan nu het grote misverstand. Velen denken dat ze bij de DAI ongestraft uit een FOR-NEXT loop mogen springen. D I T I S N I E T W A A R ! ! ! ! ! De goede programmeur wist reeds dat het principieel nooit kan . Een zeer grote stack kan het probleem enige tijd uitstellen maar het komt gegarandeerd. Het laatste voorbeeld liet het eigenlijk al zien. We springen daar uit de FOR-NEXT en we krijgen inderdaad een stack overflow. Maar de ontwerpers van de DAI hebben iets heel slims bedacht. Als er een FOR-NEXT loop opgestart wordt en er is al eerder een FOR-NEXT loop gestart met dezelfde teller dan wordt die teller

opnieuw gebruikt. De stack wordt daarmee dus niet extra belast en zo is het in veel gevallen dus toch mogelijk geworden om uit een FOR-NEXT te springen . Ter controle kunt U even intikken '10 FOR I=0 TO 2:GOTO 10'. U krijgt bij RUN geen stack overflow. De consequenties hiervan zijn, zoals hiervoor geschetst, dat U slechts zelden stack overflow zult krijgen door springen uit een FOR-NEXT loop Alleen die gevallen waar steeds uit FOR-NEXT loops wordt gesprongen en al die loops een andere loopcounter hebben, zal het bij een groter aantal fout gaan . Conclusie : Het is aan te bevelen steeds dezelfde variabelenamen te gebruiken als loopcounter. Voor de duidelijkheid van het programma is het niet aan te raden, maar het kan zonder problemen :

```
100 FOR H=1 TO 1000:H=GETC:IF H=0 THEN NEXT          Binnen 1000 keer reageren
110 PRINT "KLAAR"                                      anders gaan we verder.
```

U begrijpt dat met maar een soort loop nl de FOR-NEXT, terwijl er zoveel mogelijk zijn er naar nieuwe constructies werd gezocht. Ik behandel de WHILE-WEND en de REPEAT-UNTIL. Een belangrijk verschil met de FOR-NEXT is het feit dat nu bij aanvang nog niet bekend is hoe vaak de loop doorlopen moet worden. Dit impliceert dus tevens, dat we, bij een foutieve programmering, in een eindeloze loop kunnen komen te zitten.

Om te beginnen de REPEAT-UNTIL. De syntax voor deze instructie is voor D-BASIC als volgt : REPEAT [statement(s)] : UNTIL <logical expression> De dubbele punt voor UNTIL is noodzakelijk, maar zoals aangegeven met de rechte '[ ]' haken de statements niet. Net zoals bij de FOR-NEXT, worden de statements minstens eenmaal uitgevoerd. Enkele voorbeelden voor het gebruik :

```
10 REPEAT H=GETC:UNTIL H<>0                              wachten op toetsaanslag
20 REPEAT T=T+1:UNTIL GETC<>0        controleer de reactietijd tot toetsaanslag
30 REPEAT DOT X,Y 23:Y=Y+1:UNTIL Y>W     maak langzaam een kolom van hoogte W
40 REPEAT GOSUB 2000:UNTIL T=0       doe de subroutine op 2000 tot dat T nul is
```

Over WHILE-WEND is iets meer te vertellen. De syntax ervoor is in wezen niet correct en ook nog eens verschillend bij D-BASIC en TURTLE-BASIC . Let dus op het verschil :

*** D-BASIC : WHILE <logical expression> DO [statement(s)] : WEND

De 'DO' is hier in wezen overbodig. Ook zonder dat woord is het te begrijpen wat er gedaan moet worden. De body (= [statement(s)]) wordt uitgevoerd zolang aan de voorwaarde (= <logical expression>) wordt voldaan. Is de voorwaarde al bij de start onjuist wordt de loop NIET doorlopen. Voorbeelden voor gebruik :

```
10 WHILE GETC=0 DO WEND        Wacht op een toetsaanslag
*** Volgens de syntax van D-BASIC zou er tussen DO en WEND een : moeten staan.
20 WHILE T>0 DO T=T-1:WEND     Tel af tot nul
```

TURTLE-BASIC WHILE <logical expression> DONOT <linenumber>:[statement(s)]:WEND

De WEND is hier in wezen overbodig, daar bij logisch programmeren na de WEND juist het regelnummer volgt dat achter de 'DONOT' staat. Ik vind het niet een bepaald mooie instructie in TURTLE-BASIC. Zolang aan de voorwaarde wordt voldaan zal niet naar de regel achter de 'DONOT' gesprongen worden, maar de statements tot de WEND uitgevoerd worden. Een nadeel van de WHILE-WEND in TURTLE-BASIC is het feit dat ze niet genest kunnen worden.

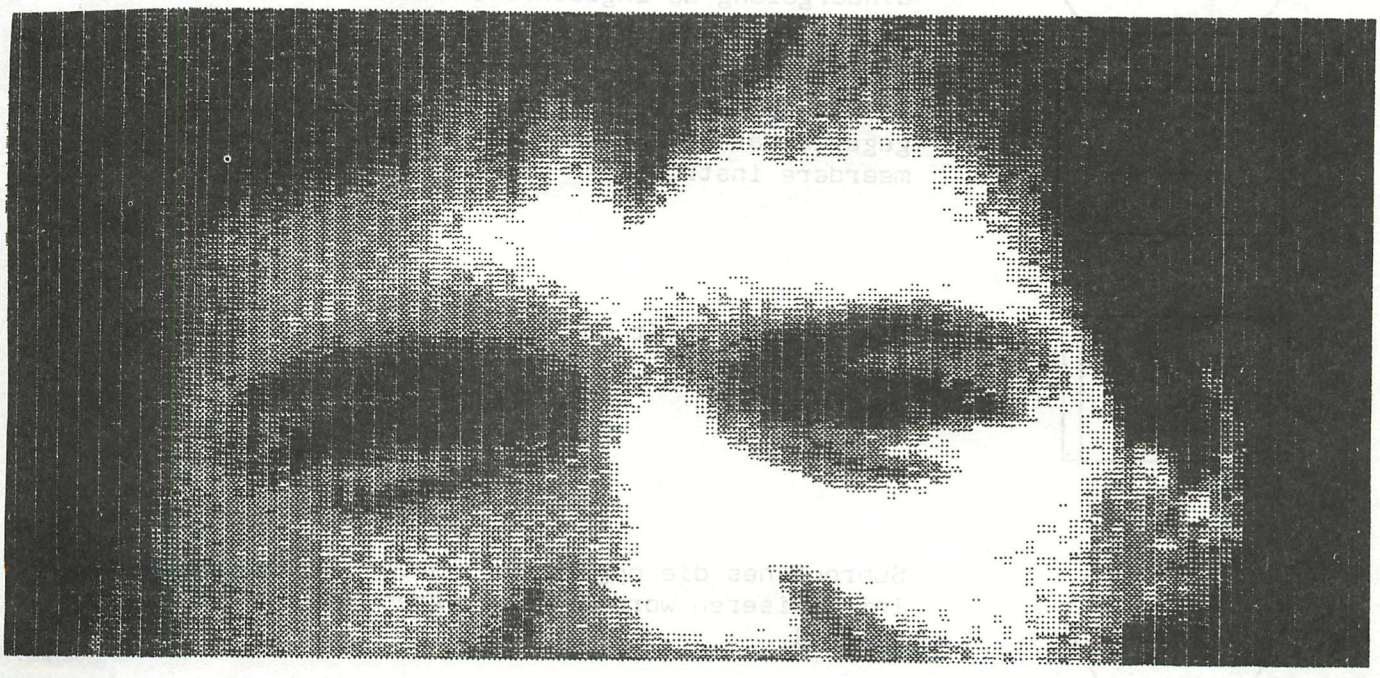
Het volgend nummer kom ik hier nog op terug. Evenals de opmerking, dat er geen verschil is tussen NEXT en NEXT I.

Frank H. Druijff

BANKOVERSCHRIJVINGEN

```
10    REM *** BANKOVERSCHRIJVINGEN ***
20    POKE #131,1:POKE #75,#FF:PRINT CHR$(12):COLORT 14 0 14 14
30    A%=1.0:T%=0.0
40    N%=0.0:T%=T%+B%
45    REM ----------------------------- SCHERMOPBOUW
50    PRINT :PRINT :PRINT
100   PRINT TAB(15);"*** INVOER VAN DE GEGEVENS ***"
110   PRINT :PRINT
120   PRINT " 1. DATUM (DDMMJJ)              :"
130   PRINT " 2. REK.NR van de BEGUNSTIGDE :"
140   PRINT " 3. BEDRAG                      :"
150   PRINT " 4. NAAM van de BEGUNSTIGDE   :"
160   PRINT " 5. STRAAT + HUISNUMMER       :"
170   PRINT " 6. POSTCODE + GEMEENTE       :"
180   PRINT " 7. MEDEDELING                  :"
190   REM ----------------------------- GEGEVENSINVOER
200   CURSOR 31,16:INPUT DA$:DA$=LEFT$(DA$,2.0)+"."+MID$(DA$,2,2)+"."+RIGHT$(DA$,2):CURSOR 31,16:PRINT DA$;"    "
205   IF X%=1.0 GOTO 410
210   CURSOR 31,15:INPUT NB$
215   REM * PRINTEN van het REK.NR in STANDAARDVORM en CONTROLE op de CHECK-DIGIT
220   CURSOR 31,15:PRINT LEFT$(NB$,3);"-";MID$(NB$,3,7);"-";RIGHT$(NB$,2);" "
230   IF VAL(RIGHT$(NB$,2))=((VAL(LEFT$(NB$,6)) MOD 97)*10000+VAL(MID$(NB$,6,4))) MOD 97 GOTO 280
240   CURSOR 46,15:PRINT "REK.NR ONJUIST"
250   FOR Z%=#BBBF-50.0*2.0-3.0 TO Z%-29.0 STEP -2.0:POKE Z%,#FF:NEXT
260   COLORT 14 0 15 0
270   GOTO 210
280   COLORT 14 0 14 14
285   IF X%=1 GOTO 410
290   CURSOR 31,14:INPUT B$
300   B%=VAL(B$):L%=LEN(B$)
305   REM * PRINTEN van het BEDRAG en PLAATSEN van het PUNTTEKEM
310   IF L%>3.0 AND L%<7.0 GOTO 330
320   IF L%>6.0 GOTO 340
325   GOTO 360
330   B$=LEFT$(B$,L%-3)+"."+RIGHT$(B$,3):GOTO 360
340   LL%=LEN(LEFT$(B$,L%-6))
350   B$=LEFT$(B$,L%-6)+"."+MID$(B$,LL%,3)+"."+RIGHT$(B$,3)
360   CURSOR 31,14:PRINT B$;"    "
365   IF X%=1.0 GOTO 410
370   CURSOR 31,13:INPUT NA$:CURSOR 31,13:PRINT NA$;"    "
375   IF X%=1.0 GOTO 410
380   CURSOR 31,12:INPUT AD$:CURSOR 31,12:PRINT AD$;"    "
385   IF X%=1.0 GOTO 410
390   CURSOR 31,11:INPUT WO$:CURSOR 31,11:PRINT WO$;"    "
395   IF X%=1.0 GOTO 410
400   CURSOR 31,10:INPUT MED$:CURSOR 31,10:PRINT MED$;"    "
405   IF X%=1.0 GOTO 410
410   X%=0.0
420   PRINT
495   REM ----------------------------- CORRECTIEROUTINE
500   CURSOR 0,8:INPUT "CORRECTIES NODIG (J/N) ";COR$:PRINT
510   IF COR$="J" GOTO 530
520   IF COR$<>"N" AND COR$<>"J" GOTO 500
525   GOTO 600
530   INPUT "TIK GEWENST NUMMER IN ";N%:PRINT
```

```
540  IF N%<1 OR N%>7 GOTO 530
550  X%=1.0
560  ON N% GOTO 200,210,290,370,380,390,400
600  CURSOR 0,2:PRINT "ZET DE PRINTER AAN"
610  PRINT "DRUK OP DE SPATIEBALK OM VERDER TE GAAN"
620  IF GETC<>32.0 GOTO 620
625  REM ------------------------------ PRINTROUTINE
630  PRINT CHR$(12):PRINT :PRINT :PRINT
640  PRINT "============================================================":PRINT
650  POKE #131,0
660  PRINT TAB(2);DA$:PRINT :PRINT
670  PRINT TAB(24);LEFT$(NB$,3);"-";MID$(NB$,3,7);"-";RIGHT$(NB$,2);TAB(42);B$:PRINT :PRINT
680  PRINT TAB(28);NA$:PRINT
690  PRINT TAB(28);AD$:PRINT
700  PRINT TAB(28);WO$:PRINT
710  PRINT MED$
720  POKE #131,1
725  PRINT "============================================================":POKE #131,0
730  FOR P%=0 TO 10:PRINT :NEXT
740  POKE #131,1
760  CURSOR 0,0:INPUT "NOG EEN OVERSCHRIJVING (J/N) ";NOG$
770  IF NOG$="N" GOTO 800
780  IF NOG$<>"N" AND NOG$<>"J" GOTO 760
790  PRINT CHR$(12):A%=A%+1:GOTO 40
800  IF A%=1 GOTO 890
810  PRINT :PRINT :PRINT
820  PRINT "============================================================":PRINT
830  POKE #131,0
840  PRINT TAB(2);DA$:PRINT :PRINT
850  PRINT TAB(24);"***-*******-***";TAB(42);T%+B%:PRINT :PRINT
860  PRINT TAB(28);"GEZAMENLIJKE OVERSCHRIJVING":PRINT :PRINT :PRINT :PRINT :PRINT
870  PRINT A%;" BIJLAGEN"
875  POKE #131,1:PRINT "============================================================":POKE #131,0
880  PRINT CHR$(12)
890  POKE #131,1:END
```



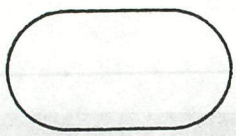
NOP	00000000	- - - - -	1	4

No operation
Deze opdracht heeft geen enkel resultaat tenzij het laten voorbijgaan van 4 klokperiodes. Deze opdracht kan bijvoorbeeld gebruikt worden om een overbodige instructie te vervangen zonder dat daarbij het hele programma dient herschreven te worden. De NOP instructie laat eveneens toe in een programma plaatsen te voorzien waar achteraf een instructie kan toegevoegd worden.

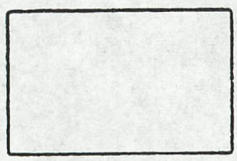
4.3. Opbouw van een programma

Als we een programma wensen op te bouwen, gaan hieraan enkele belangrijke stadia vooraf.
Het probleem of de toepassing waarvoor een programma dient geschreven te worden, dient vooraf in al zijn facetten grondig bestudeerd en geanalyseerd te worden.
Als resultaat van deze analyse wordt een organigram opgebouwd, waarin aan de hand van tekensymbolen de verschillende fasen gesymboliseerd worden.
We merken op dat een programma zoveel mogelijk in kleine modules dient geschreven te worden. Onder een module verstaat men een programmagedeelte dat een logisch geheel vormt en zelfstandig bepaalde akties uitvoert.
Het voordeel van deze modules is dat ze afzonderlijk kunnen getest worden op foutloze werking. Ze kunnen tevens door andere programma onderdelen opgeroepn worden als subroutines.
Men dient een programma steeds te voorzien van de nodige commentaar (documenteren), niet alleen om achteraf het programma vlot te kunnen lezen maar tevens om aan derden toe te laten uw programma te begrijpen. Men noemt dit een programma documenteren.

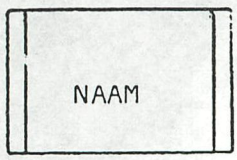
4.4. Symbolen gebruikt in organigrammen



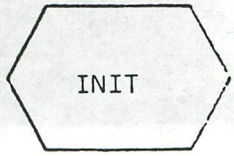
Dit symbool duidt het begin van een programma (START) of het einde van een programma (END) aan, alnaargelang de ingeschreven tekst.



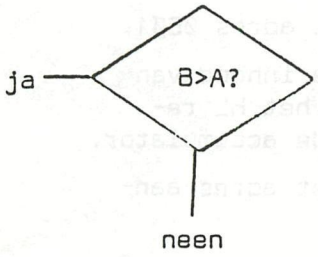
In dit symbool wordt een programma onderdeel aangegeven dat kan afgehandeld worden door één of meerdere instructies.



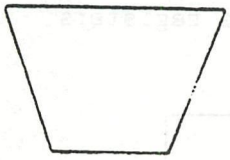
Dit symbool geeft het oproepen aan van de subroutine NAAM.



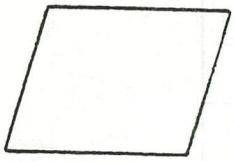
Subroutines die opgeroepen worden om bouwstenen te initialiseren worden ook voorgesteld door dit symbool.



De in dit symbool geschreven vergelijking of tekst wordt getest. Het programma wordt verdergezet in de richting van de 'ja-uitgang' of de 'neen-uitgang' alnaargelang het resultaat van de test.



Symbool voor manuele invoer (bijvoorbeeld van een klavier).



Symbool voor machinale in- en uitvoer (bijvoorbeeld naar een regeldrukker).

Als het organigram meerdere pagina's lang is, geeft dit symbool aan dat de lijnen met hetzelfde nummer dienen doorverbonden te worden.

4.5. Programmavoorbeelden voor de 8080/8085

Voorbeeld 1 : optelling van twee 8 bits getallen

Veronderstel dat in RAM op adres A12D het hexadecimaal getal B6 staat en op het adres ØEØ1 het getal 1A.
We wensen deze getallen op te tellen en het resultaat terug te schrijven op het adres ØEØ1.
Het programma bevindt zich bijvoorbeeld in ROM vanaf het adres ØØ21 en gebruik makend van de mnemonics, luidt het :

```
LDA    A120    H
LXI    H,E01   H
ADD    M
MOV    M,A
```

LDA ØA12D H — laadt de accumulator met de inhoud van de geheugencel A12D. In hexadecimale notatie moeten getallen steeds beginnen met een cijfer (vandaar de Ø voor A12D en eindigen met H. Zonder H wordt het een decimaal getal, met een B wordt het een binair getal.

LXI H,ØEØ1 H — laadt het registerpaar onmiddellijk met het adres ØEØ1.

ADD M — telt de inhoud van de accumulator op bij de inhoud van de geheugencel die geadresseerd wordt door het HL registerpaar (ØEØ1). Het resultaat komt in de accumulator.

MOV M,A — Brengt de inhoud van de accumulator naar het adres aangegeven door het HL register (ØEØ1).

Om duidelijk te zien wat er gebeurt in functie van de tijd, hebben we onderstaande tabel voor elke instructie de inhoud gegeven van de registers en geheugencellen die bij dit programma betrokken zijn.

ROM			RAM			
adr	Mnemonic	Hexacode	HL	ACCU	A12D	ØEØ1
ØØ21	LDA ØA12D H	3A	?(1)	?	B6	1A
ØØ22		2D	?	?	B6	1A
ØØ23		A1	?	B6	B6	1A
ØØ24	LXI H,ØEØ1 H	21	?	B6	B6	1A
ØØ25		Ø1	?	B6	B6	1A
ØØ26		ØE	ØEØ1	B6	B6	1A
ØØ27	ADD M	86	ØEØ1	DØ(2)	B6	1A
ØØ28	MOV M,A	77	ØEØ1	DØ	B6	DØ

(1) ? wil zeggen dat de inhoud niet nader bekend is en een gevolg is van een vorig programma of van een startprocedure van de microprocessor.

(2) DØ is de hexadecimale som van B6 en 1A.

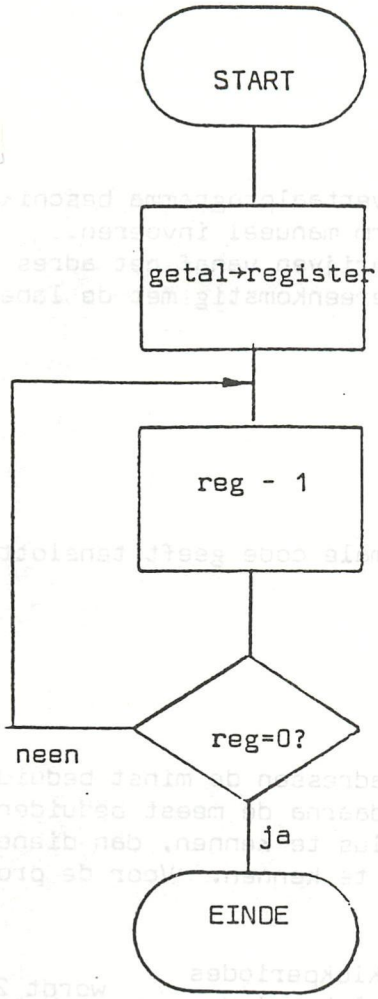
Het programma kan niet onder de vorm van mnemonics in het programmatiegeheugen gebracht worden, maar wel onder de vorm van de hexadecimale code van kolom 3.

Stel zelf het programma op voor het optellen van twee 16 bits getallen
a) met 8 bits optelinstructies (ADA, ADC)
b) met de 16 bits optelinstructie DAD.

Voorbeeld 2 : programmatie van een korte vertragingslus

In een register wordt een getal ingeschreven. De registerinhoud wordt met één verminderd en er wordt getest of de inhoud nul is. Indien de inhoud niet nul is, wordt de registerinhoud opnieuw met één verminderd en terug getest.

ORGANIGRAM :



PROGRAMMA :

```
        MVI     B,FF      H
LUS:    DCR B
        JNZ LUS
```

LUS staat in dit programma als symbolisch adres LABEL genoemd. (LUS is het adres waar de instructie DCR B is ondergebracht.
Als we over een assembler vertaalprogramma beschikken, kunnen we het programma rechtstreeks invoeren in mnemonics, waarbij gebruik gemaakt kan worden van labels, dit zijn symbolische namen (adressen) waaraan door de assembler een bepaalde waarde wordt toegekend. Deze mag tot zes alfanumerische karakters bevatten met als eerste een letter, een vraagteken of een @. De label BEGIN wordt gevolgd door een dubbelpunt. Labels mogen geen mnemonics zijn die gebruikt worden als opcode. Andere assembler directieven zijn ORG adres (PC=adres), END (afsluiting) en EQU uitdrukking of adres waarbij de waarde toegekend wordt aan een label.
Voorgaand programma wordt dus :

```
BEGIN:  ORG     1000      H
TEL     EQU     FF        H
        MVI     B,TEL
LUS:    DCR     B
        JNZ     LUS
        END
```

Als we echter niet over een assembler vertaalprogramma beschikken, moeten we het programma onder hexadecimale vorm manueel invoeren.
Veronderstel dat we dit programma inschrijven vanaf het adres 1ØØØ H in RAM, dan wordt het symbolisch adres overeenkomstig met de label LUS gelijk aan 1ØØ2 H.

```
        MVI     B,FF     H
1002    DCR     B
        JNZ     1002
```

Omzetting van de mnemonics in hexadecimale code geeft tenslotte :

```
1000   04 FF
1002   05
1003   02 02 10
```

Let er op dat bij het invoeren van de adressen de minst beduidende byte het eerste komt en slechts daarna de meest beduidende byte.
Wensen we de vertragingstijd van deze lus te kennen, dan dienen we eerst het aantal klokperiodes per instructie te kennen. Voor de processor 8080 wordt dit :

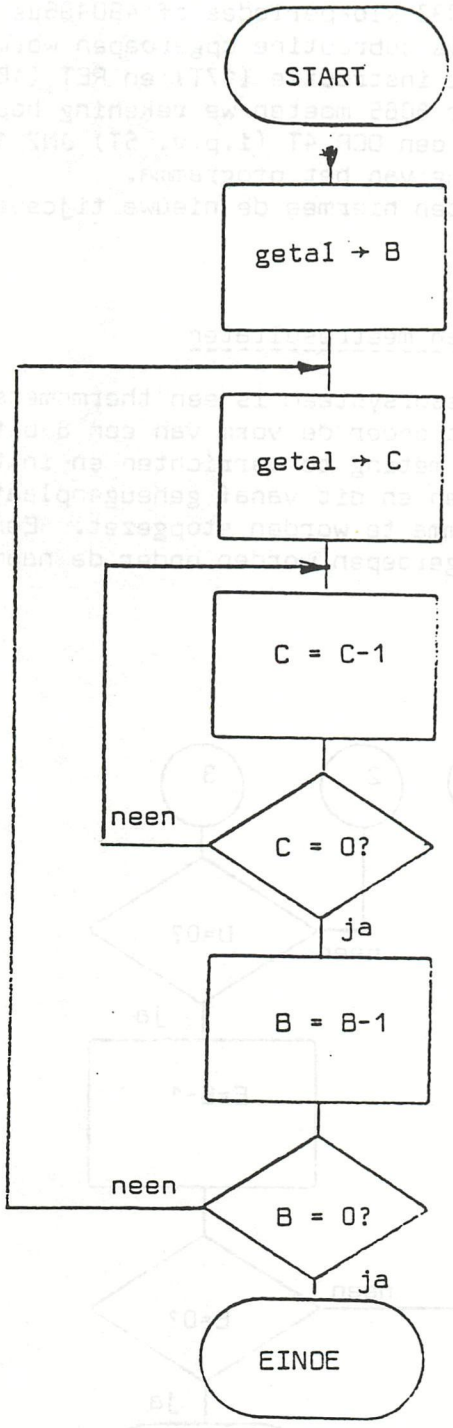
```
        MVI     B,FF     H      7 klokperiodes
LUS:    DCR     B               5 klokperiodes     wordt 255 (FF)
        JNZ     LUS            10 klokperiodes     maal uitgevoerd
```

De totale vertragingstijd is : 7 + (5+10) 255 = 3822 klokperiodes.
Indien we werken met een klok van 2MHz (periode 500ns) dan is de vertragingstijd 3832 x 0,5 = 1916µs.
Indien kleine tijdsaanpassingen gewenst zijn, kunnen we enkele instructies inlassen die geen invloed hebben op de tellerwerking (bv. NOP, MOV A,A ; ORA A,A ; XCHG).
Bij het inlassen van 1 NOP na DCRB wordt de vertragingstijd 2,4ms.

Voorbeeld 3 : programmatie van een dubbele vertragingslus

Het gebruik van een dubbele vertragingslus in genestelde lussen, kan aanzienlijk langere vertragingstijden geven.

ORGANIGRAM :



PROGRAMMA :

P102

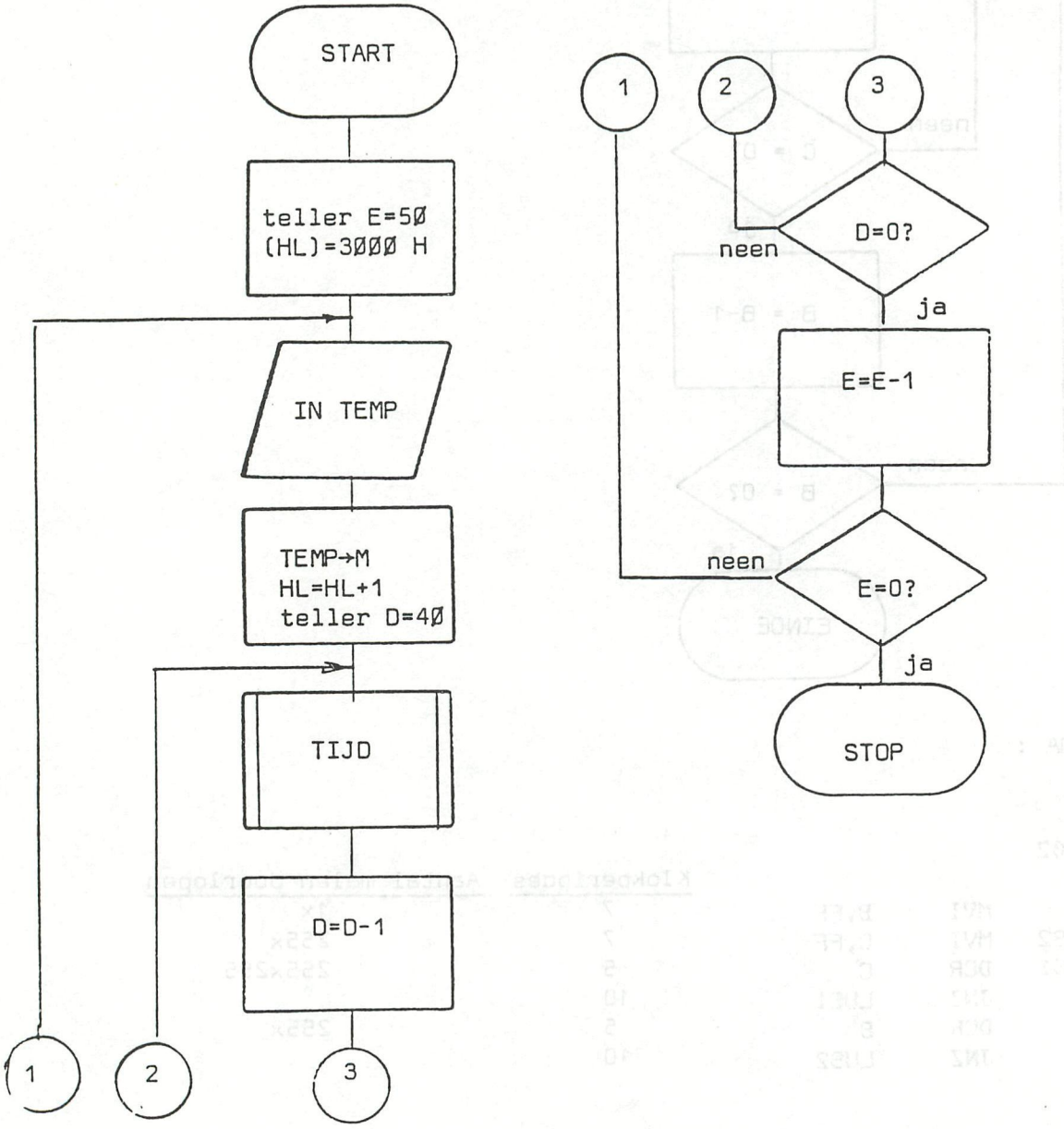
			Klokperiodes	Aantal malen doorlopen
	MVI	B,FF	7	1x
LUS2	MVI	C,FF	7	255x
LUS1	DCR	C	5	255x255
	JNZ	LUS1	10	
	DCR	B	5	255x
	JNZ	LUS2	10	

Totale vertragingstijd :
$7+255(10+5+7)+255^2(5+10)$ = 980.992 klokperiodes of 490496µs = 0,49sec.
Indien deze vertragingslussen als subroutine opgeroepen worden, vergroot de tijd met de duur van een CALL instructie (17T) en RET (10T).
Bij het gebruik van de processor 8085 moeten we rekening houden met enkele tijdsaanpassingen. Zo duurt een DCR 4T (i.p.v. 5T) JNZ 10T bij een sprong en 7T bij een voortzetting van het programma.
Het wordt aan de lezer overgelaten hiermee de nieuwe tijdsvertragingen te berekenen.

Voorbeeld 4 : Het inschrijven van meetresultaten

Aan kanaal 2 van een microprocessorsysteem is een thermometer aangesloten die de gemeten temperatuur geeft onder de vorm van een 8-bits code.
We wensen om de 20 seconden een meting te verrichten en in te schrijven in opeenvolgende geheugenplaatsen en dit vanaf geheugenplaats 3ØØØ.
Na 50 metingen dient het programma te worden stopgezet. Een vertragingsroutine van 0,5 seconden kan opgeroepen worden onder de naam TIJD.

ORGANIGRAM :



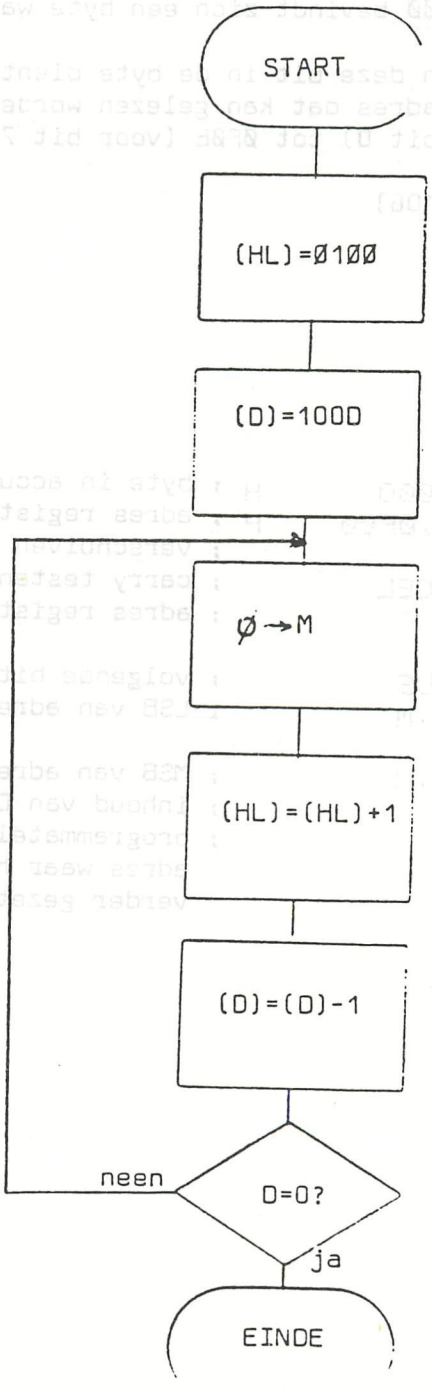
PROGRAMMA :

```
        MVI     E,50            ; teller laden met 50
        LXI     H,3000          ; adres register laden met 3000
LUS3    IN      2               ; thermometer lezen
        MOV     M,A             ; lezing inschrijven in geheugen
        INX     H               ; adresregisters incrementeren
        MVI     D,40
LUS4    CALL    TIJD            ; vertragingslus 20sec
        DCR     D
        JNZ     LUS4
        DCR     E               ; teller decrementeren
        JNZ     LUS3            ; testen of teller op nul staat
        HLT
```

Voorbeeld 5 : op 0 stellen van geheugen

We wensen 100 opeenvolgende geheugenplaatsen vanaf adres Ø1ØØ H op nul te zetten.

ORGANIGRAM :



PROGRAMMA :

```
        LXI     H,100           ; adres register laden met Ø1ØØ
        MVI     D,100           ; teller laden met 1ØØ
WISSEN  MVI     M,ØØ            ; geheugencel op nul zetten
        INX     H               ; adres register incrementeren
        DCR     D               ; teller decrementeren
        JNZ     WISSEN          ; teller testen op nul
```

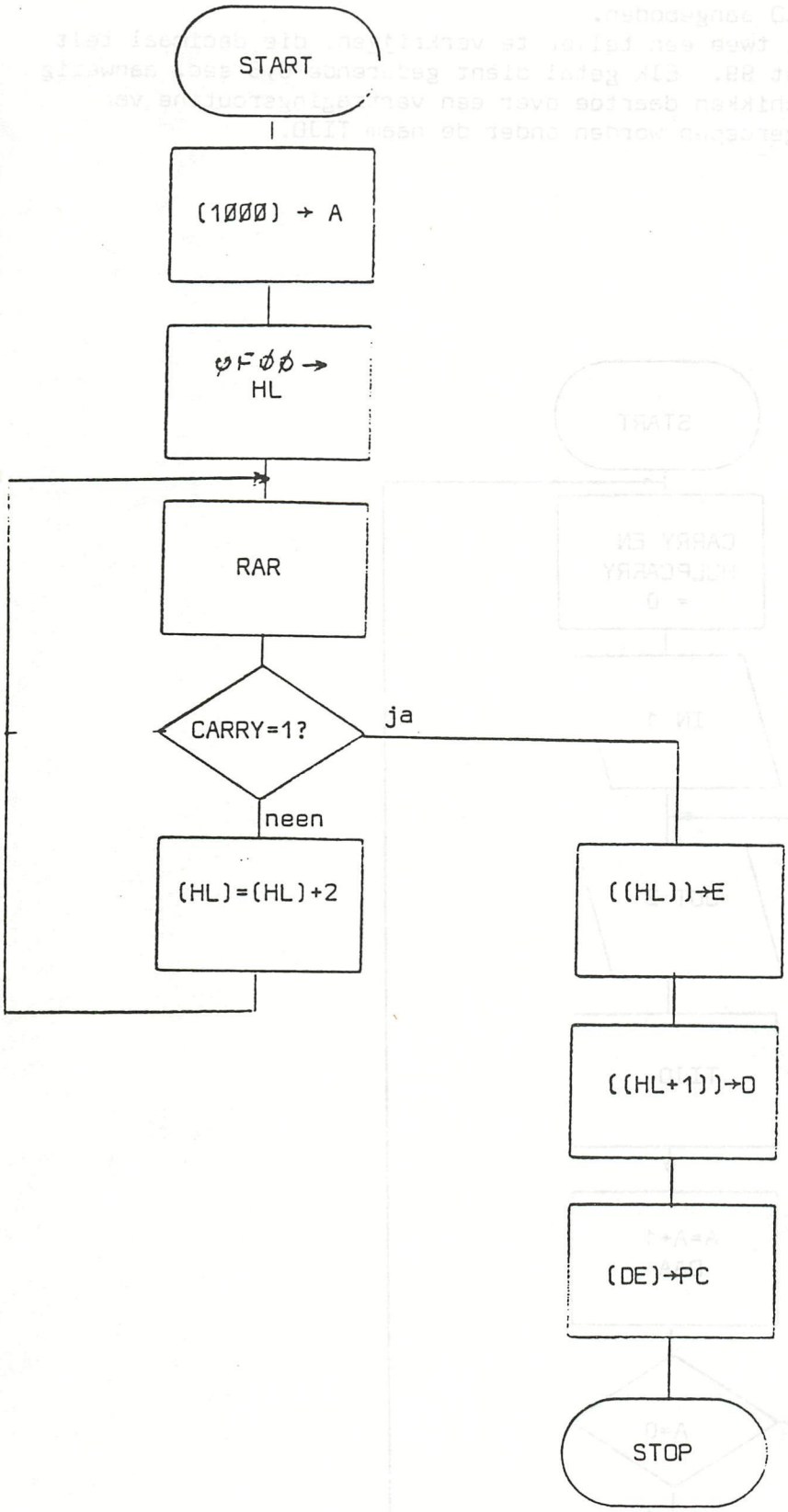
Voorbeeld 6 : programmavertakking

In de geheugenplaats 1ØØØ bevindt zich een byte waarvan slechts één bit gelijk is aan 1.
Naargelang de plaats van deze bit in de byte dient het programma verder gezet te worden op een adres dat kan gelezen worden in de geheugentabel vanaf adres ØFØØ (voor bit 0) tot ØFØE (voor bit 7).

ORGANIGRAM : (zie pag. 106)

PROGRAMMA :

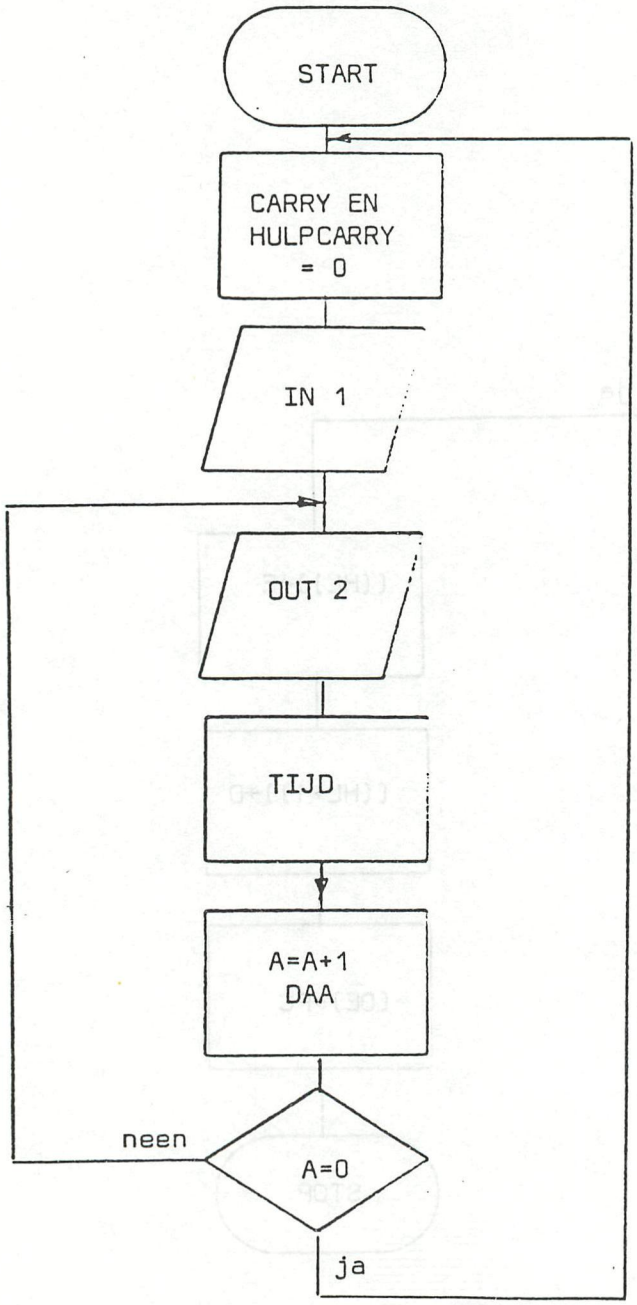
```
0001            LDA     1000      H ; byte in accumulator
0002            LXI     H,ØF00    H ; adres register laden met ØFØØ
0003 LUS:       RAR                 ; verschuiven naar rechts
0004            JC      DOEL        ; carry testen
0005            INX     H           ; adres register 2x ophogen
0006            INX     H
0007            JMP     LUS         ; volgende bit testen
0008 DOEL:      MOV     E,M         ; LSB van adres in E
0009            INX     H
0010            MOV     D,M         ; MSB van adres in D
0011            XCHG                ; inhoud van DE naar HL
0012            PCHL                ; programmateller laden met
                                      adres waar het programma
                                      verder gezet dient te worden.
```


START
(1ØØØ) → A
ØFØØ →
HL
RAR
CARRY=1?
ja
neen
(HL)=(HL)+2
((HL))→E
((HL+1))→D
(DE)→PC
STOP

Voorbeeld 7 : teller

Aan kanaal 1 van een microprocessorsysteem wordt een getal (N) van twee decimalen in BCD aangeboden.
We wensen aan kanaal twee een teller te verkrijgen, die decimaal telt vanaf het getal N tot 99. Elk getal dient gedurende 0,5 sec. aanwezig te blijven. We beschikken daartoe over een vertragingsroutine van 0,5 sec. die kan opgeroepen worden onder de naam TIJD.

ORGANIGRAM :



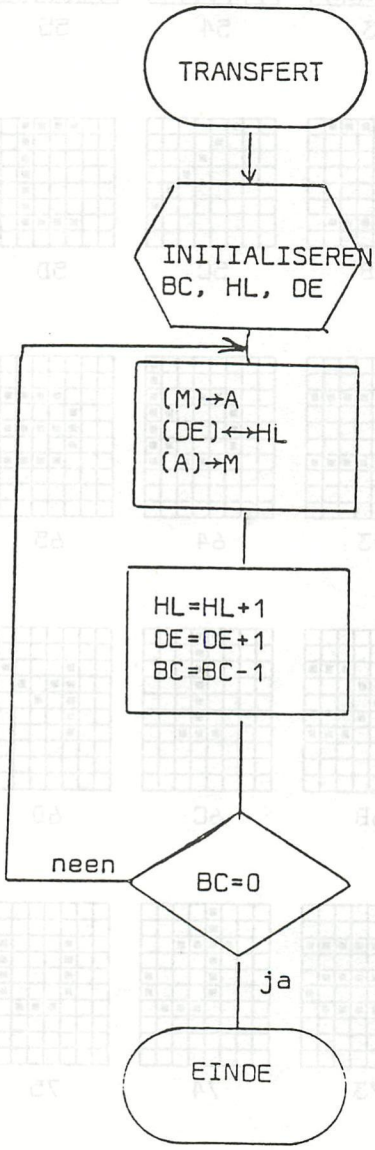
PROGRAMMA :

```
        XRA     A               ; carry en hulpcarry op nul
BEGIN   IN      1               ; getal lezen
UIT     OUT     2               ; getal afdrukken
        CALL    TIJD            ; 0,5 sec. wachten
        INR     A               ; accumulator ophogen
        DAA                     ; accu. omzetten in BCD
        CPI     0               ; testen overgang 99/0
        JNZ     UIT             ; indien ≠ 0 verder tellen
        JMP     BEGIN           ; indien = 0 herbeginnen
```

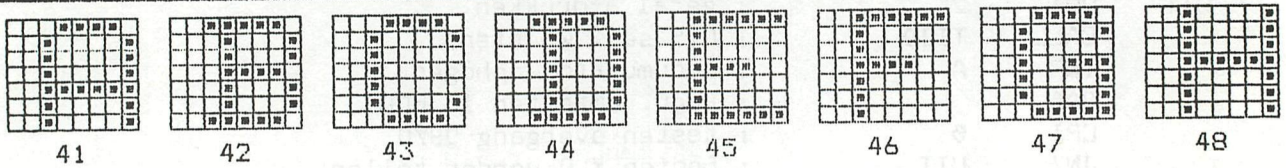
Voorbeeld 8 : verplaatsing van een blok informatie in geheugen

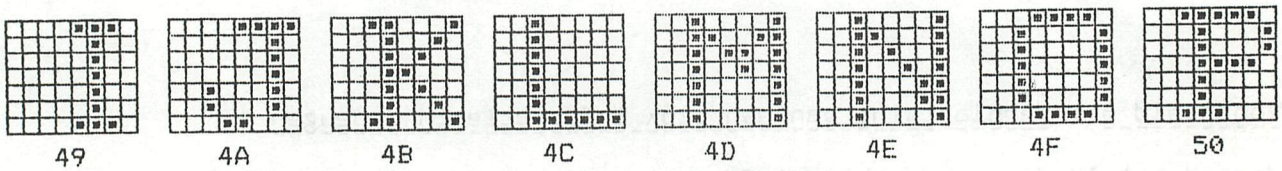
Verondersteld dat we een blok informatie, waarvan de lengte zich in het registerpaar BC bevindt, wensen te verplaatsen van het beginadres dat zich in HL bevindt naar het beginadres dat zich in DE bevindt.

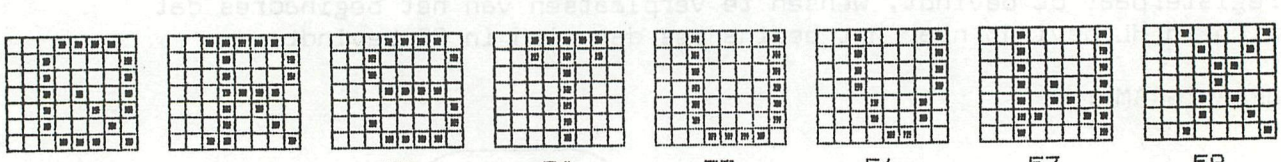
ORGANIGRAM :

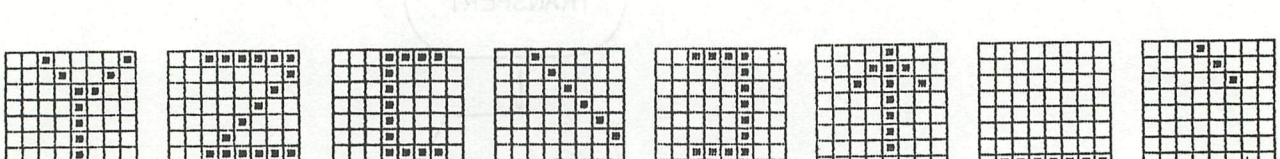


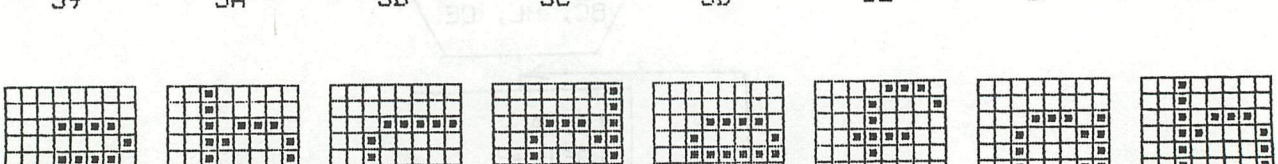
DAI CHARACTER SET

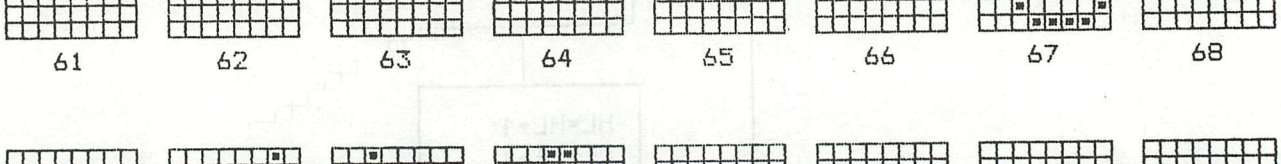


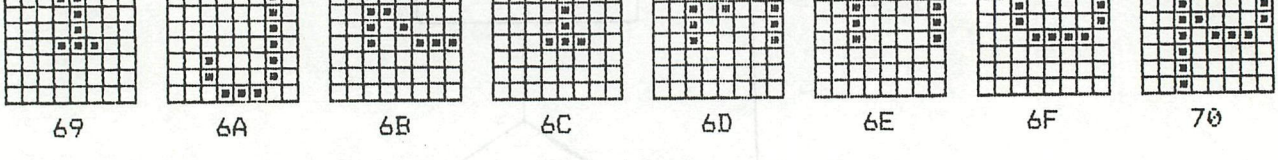


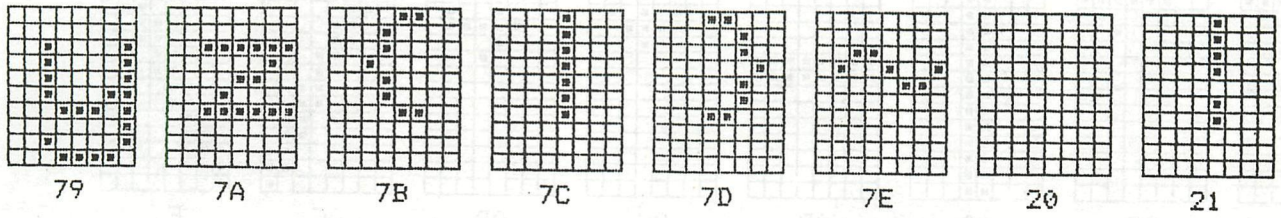


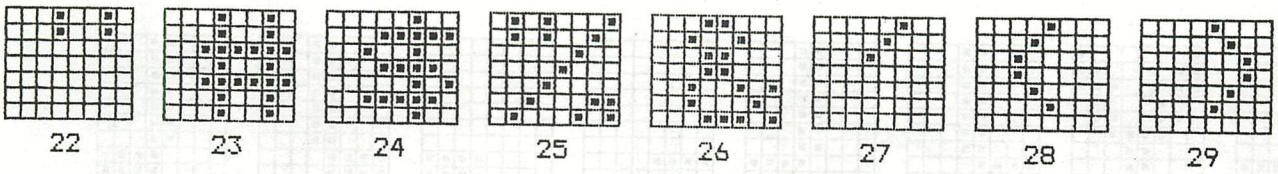


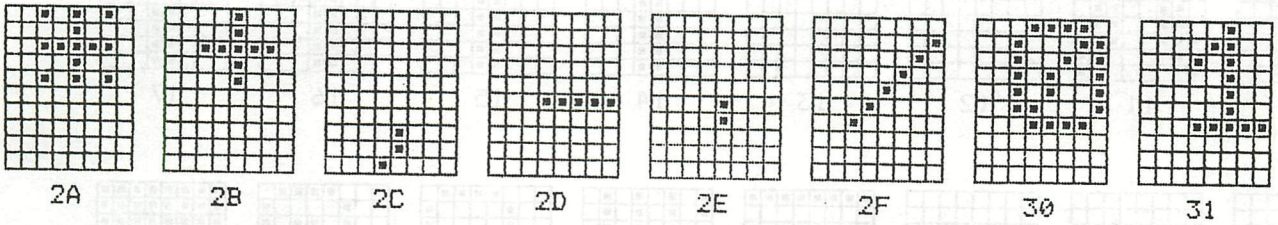


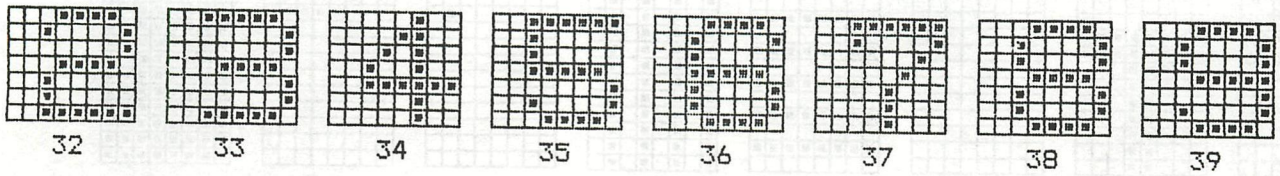


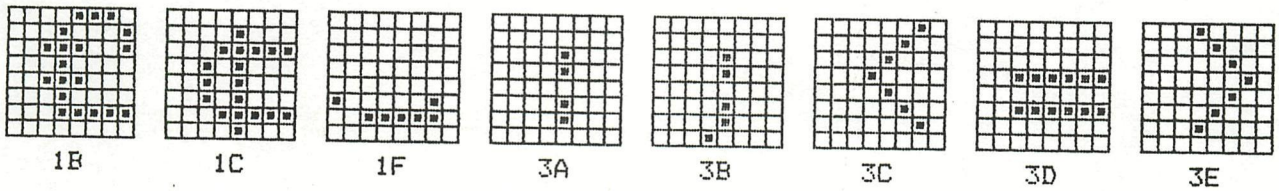


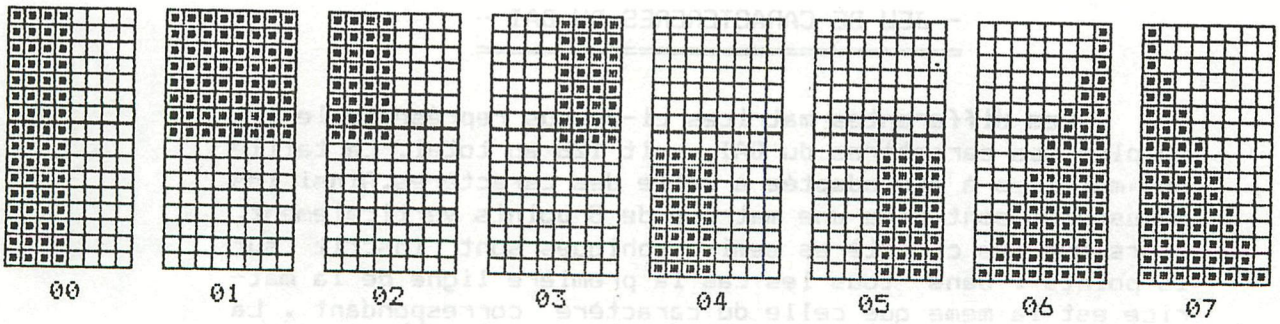

79
7A
7B
7C
7D
7E
20
21

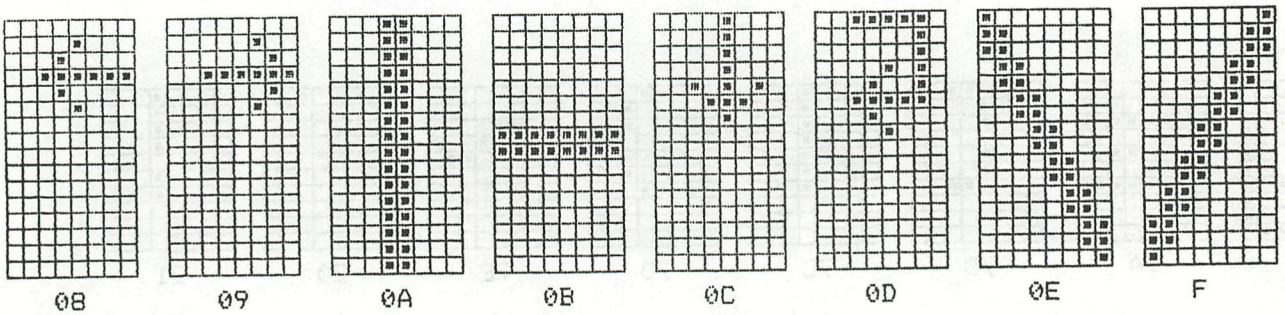

22
23
24
25
26
27
28
29

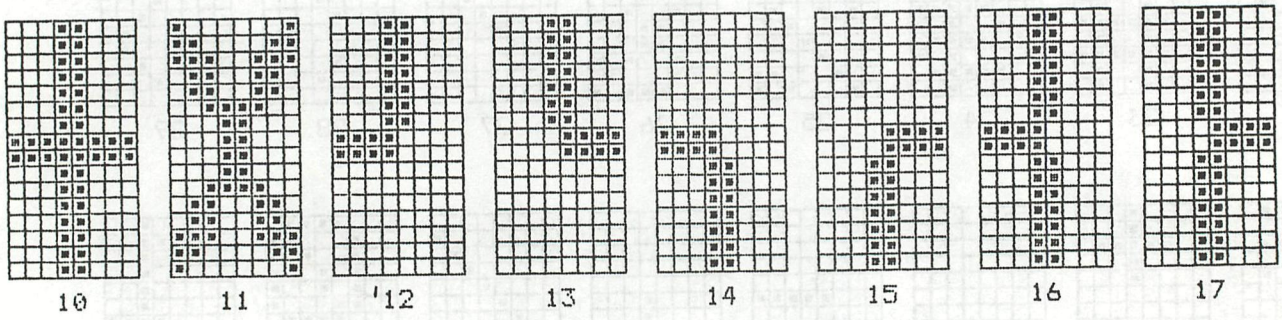

2A
2B
2C
2D
2E
2F
30
31

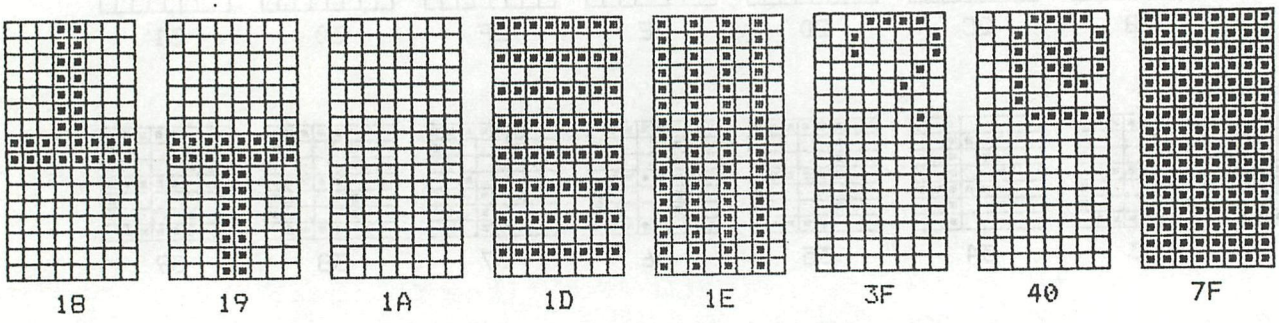

32
33
34
35
36
37
38
39


1B
1C
1F
3A
3B
3C
3D
3E


00
01
02
03
04
05
06
07







- JEU DE CARACTERERES DU DAI -

Les differentes matrices ci-dessus représente le jeu complet des caractères du DAI, soit 128 au total. La taille des matrices à été adaptée à celle des caractères.Ainsi les majuscules sont dans une matrice de 8 points verticalement, alors que les caractères semi-graphiques sont inscrit sur 16 points . Dans tous les cas la première ligne de la matrice est la meme que celle du caractère correspondant . La lecture des dessins des caractères ayant été réalisée manuellement, si vous decouvrez une erreur faites le moi savoir !

Cédric Dufour

DAIminiCALC

AVANT DE CHARGER LE PROGRAMME TAPER 'IMPINT'

Ce programme permet le calcul de formules utilisant :

1. Le symbole # pour introduire une formule
   des lettres A-Z ou les symboles] [< >
   qui désignent une ligne du tableau
   des opérateurs +,-,* ou /
   des constantes entières ou décimales
   des parenthèses ()
   La touche return pour terminer la formule
   Exemple : #(A*100+B)/(C*1.89+D)
   Le résultat est attribué à la ligne ou la formule est écrite
   La formule est valable pour toute une ligne(1-14).
   Après analyse chaque formule est memorisée et transformée en ligne de programme(dank-u Mr Looije)
   C'est la forme algébrique usuelle (priorité de * et /)

2. Les données sont introduites dans le tableau en déplaçant la fenêtre colorée avec les flèches
   et taper espace + la valeur de la donnée + return.
   Les données numeriques ont priorité sur les alphanumériques.
   Une donnée numérique peut être entrée comme alphanumérique en lui incorporant un symbole par ex /: 12/ 3/82
   Une donnée <>0 peut être recopiée dans la fenêtre voisine:
   il faut employer curseur + shift
   La rapidité du calcul dépend du nombre de formules
   La rapidité de l'affichage dépend des dimensions des tableaux

3. En plus du mode introduction des données et formules il existe des commandes (tapez /) :
   /L pour Load un écran from DCR
   le titre apparait
   N=skip au suivant
   space=load
   /S pour save un écran sur DCR avec titre obligatoire
   /C pour effectuer les calculs
   /T pour recopier une colonne dans une autre
   /D pour recopier une ligne dans une autre
   /P pour imprimer le tableau avec le titre du save
   /R ou F pour revenir en mode entrée des données
   /I pour insérer une ligne avec décalage du tableau vers le bas
   les formules sont aussi décalées
   /Z pour annuler une rangée
   /! position supérieure gauche de l'écran
   /= position supérieure droite
   /z position inférieure gauche
   /? position inférieure droite par rapport au tableau 30x14

4. Pour changer un symbôle dans une formule (chiffre ou lettre) on se place sur la ligne ou se trouve la formule à introduire
   Exemple: en ligne M se trouve la formule #(A*G)/1.383
   on veut que la formule de la ligne T devienne: #(A*K)/1.282
   on place la fenêtre sur la ligne T et on tape #T:GK:32

5. En cas d'arrêt du programme avec erreur taper RUN 100
   et ensuite /= pour récupérer le tableau
   et poursuivre le programme.

6. Il existe une version disque (KEN-DOS) plus rapide pour les chargements de tableaux mais qui utilise plus de mémoire de masse.

```
1       FOR DD=0 TO 8:READ C:POKE #F800+DD,C:NEXT
2       DATA #C5,#CD,#79,#D8,#CD,#18,#C9,#C1,#C9
4       GOSUB 8900
5       II=1
6       GOSUB 52800:II=II+1:IF II<=30 GOTO 6

10      REM DAIminiCALC  /Ph. WANET 3/83 ----   IMP INT !!!
20      CLEAR 16000:DIM R$(30.0),A(30.0,15.0),A$(30.0,15.0),PIL$(30.0),
        PR$(10.0),PR(10.0),P(10.0),T$(0.0),E$(21.0)
23      TTA$="  TEXTES  ":FOR IK=1 TO 14:AA$=STR$(IK):AA$=LEFT$(AA$,LEN(AA$)-2)
24      TTA$=TTA$+SPC(8-LEN(AA$))+AA$:NEXT:TTA$=TTA$+"          "
25      POKE #131,1:PRINT :PRINT :INPUT "DATE : ";DATE$:FOR JJ=1 TO 60:S$=S$+"
        ":NEXT
27      FOR IK=0 TO 21:E$(IK)=S$:NEXT:FOR IK=0 TO 30:R$(IK)=S$+S$+S$:NEXT
30      COLORT 8 0 14 3
40      POKE #75,32:CC=#FF:PC=31:YY=1:YZ=1:GOSUB 5000:INIT=1:GOSUB 3700
98      CURSOR 0,1:PRINT "'Space'=entr. donnee.'Shift+cur.'=recopier donnee":
        PRINT "'#'=formule.'/'=commandes";:CURSOR 0,0:WAIT TIME 20:GOSUB 1500
99      CX=CURX:CY=CURY:GOSUB 1000:CURSOR CX,CY
100     QK=GETC:IF QK=0 THEN 100
110     IF QK>15.0 AND QK<20.0 THEN CC=0:GOSUB 1000
112     IF QK>19 AND QK<24 THEN 2000
115     IF QK=35.0 THEN CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:PRINT "#";:A$(YZ,
        15.0)="":GOSUB 3000:GOTO 99
120     IF QK=17.0 THEN GOSUB 4000
130     IF QK=16.0 THEN GOSUB 4100
140     IF QK=18.0 THEN GOSUB 4200
150     IF QK=19.0 THEN GOSUB 4300
160     IF QK>15.0 AND QK<20.0 THEN CC=#FF:GOTO 99
170     IF QK=32 THEN 500
195     IF QK=47 THEN CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:PRINT "/";:GOSUB
        6000:GOTO 98
299     GOTO 99
500     CURSOR 0,0:INPUT S$
510     LI=8:IF XZ=0 THEN LI=12
520     IF LEN(S$)>LI THEN S$=RIGHT$(S$,LI)
530     LK=LEN(S$):SS=0:FN=0
535     IF LK=0.0 THEN 590
537     SI=1:IF LEFT$(S$,1)="-" THEN SI=-1:LK=LK-1:S$=RIGHT$(S$,LK)
540     FOR IK=0 TO LK-1
550   1   A$=MID$(S$,IK,1):IF A$>="0" AND A$<="9" THEN SS=SS*10+ASC(A$)-48:GOTO
      1   570
560   1   FN=1:SS=0:GOTO 580
570     NEXT IK
580     IF FN=0 THEN A(YZ,XZ)=SI*SS:IF SI=(-1) THEN S$="-"+S$
590     A$(YZ,XZ)=S$

600     REM IF SI%=(-1.0) THEN S$="-"+S$
603     LK=LEN(S$):S$=SPC(LI-LK)+S$:R$(YZ)=LEFT$(R$(YZ),12+(XZ-1)*LI)+S$+
        RIGHT$(R$(YZ),(14-XZ)*8)
605     E$(YY)=MID$(R$(YZ),(XZ-XX)*8,60)
610     CURSOR 0,23-YY:PRINT E$(YY);:CURSOR 0,0:PRINT SPC(58);:CURSOR 0,0
620     GOTO 99
700     CURSOR 0,0:PRINT S$;:RETURN
800     LK=LEN(S$):SS=0
810     IF LK=0.0 THEN 870
820     SI=1:IF LEFT$(S$,1)="-" THEN SI=-1:LK=LK-1:S$=RIGHT$(S$,LK)
830     FOR II=0 TO LK-1
840   1   A$=MID$(S$,II,1):IF A$>="0" AND A$<="9" THEN SS=SS*10+ASC(A$)-48:GOTO
      1   860
```

```
845   1  IF A$=" " THEN 860
850   1  SS=0:GOTO 870
860      NEXT II
865      IF SI=(-1.0) THEN S$="-"+S$
870      SS=SI*SS:RETURN
1000     POKE #75,32:POKE #74,1:X1=12+XX*8:DD=0:IF XX=0 THEN DD=6
1010     AA=#BFEF-YY*#86-X1*2+5+DD
1030     FOR IK=AA TO AA-14-DD STEP -2
1040  1  POKE IK,CC
1050     NEXT IK
1060     CURSOR 0,1:PRINT SPC(60);:CURSOR 0,0:PRINT SPC(59);:CURSOR 0,1:PRINT
         A$(YZ,15.0)
1070     POKE #75,#FF:POKE #74,0:RETURN
1500     SOUND 0 0 15 0 FREQ(500.0):WAIT TIME 2:SOUND OFF :RETURN
2000     SS=A(YZ,XZ):S$=A$(YZ,XZ)
2005     IF QK=21.0 THEN GOSUB 4000
2010     IF QK=20.0 THEN GOSUB 4100
2020     IF QK=22.0 THEN GOSUB 4200
2030     IF QK=23.0 THEN GOSUB 4300
2032     LI=8:IF XZ=0 THEN LI=12
2034     IF LEN(S$)>LI THEN S$=RIGHT$(S$,LI)
2035     A$(YZ,XZ)=S$
2037     IF S$<>"" AND SS=0 THEN 2050
2040     A(YZ,XZ)=SS:GOSUB 20000
2050     GOTO 603

3000     REM ENTREE DES FORMULES & NPI
3005     INPUT EX$:IF EX$="" THEN 3020
3010     IF LEFT$(EX$,1)="~" THEN 3800
3020     EX$="#"+EX$+",":PP=1:TR$="":PL=0:C2$="":PAR=0
3030     GOSUB 3440:REM LIRE
3040     PIL$(PP)=C2$:PP=PP+1
3050     GOSUB 3440:REM LIRE
3055     IF C2$="<" THEN C2$=CHR$(92)
3056     IF C2$=">" THEN C2$=CHR$(94)
3060     IF C2$=" " THEN 3050
3070     IF C2$="(" THEN 3140
3080     IF C2$>="A" AND C2$<=CHR$(94) THEN 3170
3090     IF C2$="+" OR C2$="-" OR C2$="*" OR C2$="/" THEN 3200
3100     IF C2$=")" THEN 3260
3110     IF C2$>="0" AND C2$<="9" THEN 3340
3120     IF C2$="," THEN 3310
3130     ER=1:GOTO 3430

3140     REM (
3150     PAR=PAR+1:IF C1$>="A" AND C1$<=CHR$(94) THEN ER=2:GOTO 3430
3160     PIL$(PP)=C2$:PP=PP+1:GOTO 3050

3170     REM OPERANDE
3180     IF (C1$>="A" AND C1$<=CHR$(94)) OR C1$=")" THEN ER=3:GOTO 3430
3190     TR$=TR$+C2$:GOTO 3050

3200     REM OPERATEUR
3210     IF C1$="+" OR C1$="-" OR C1$="*" OR C1$="/" OR C1$="(" OR C1$="#" THEN
         ER=4:GOTO 3430
3220     C$=C2$:GOSUB 3570:SPRI=PRI
3230     C$=PIL$(PP-1.0):GOSUB 3570
3240     IF SPRI>PRI THEN PIL$(PP)=C2$:PP=PP+1:GOTO 3050
3250     TR$=TR$+PIL$(PP-1.0):PP=PP-1:GOTO 3230
```

```
3260    REM )
3270    PAR=PAR-1:IF C1$="+" OR C1$="-" OR C1$="*" OR C1$="/" THEN ER=5:GOTO
        3430
3280    IF PIL$(PP-1.0)="(" THEN PP=PP-1:GOTO 3050
3290    IF PIL$(PP-1.0)="#" THEN ER=6:GOTO 3430
3300    TR$=TR$+PIL$(PP-1.0):PP=PP-1:GOTO 3280
3310    IF PAR<>0 THEN ER=9:GOTO 3430
3320    IF PIL$(PP-1.0)="#" THEN A$(YZ,15.0)=EX$:II=YZ:GOSUB 52800:CURSOR 0,0:
        PRINT SPC(58);:CURSOR 0,0:RETURN
3330    TR$=TR$+PIL$(PP-1.0):PP=PP-1:GOTO 3320

3340    REM CONSTANTE
3350    FLP=0
3360    GOSUB 3440:IF C2$>="0" AND C2$<="9" THEN 3360
3390    IF C2$="." THEN FLP=FLP+1:GOTO 3360
3400    IF FLP<=1 THEN 3060
3410    ER=10
3430    PRINT " Erreur de syntaxe ";ER;:WAIT TIME 300:CURSOR 0,0:PRINT SPC(58);:
        CURSOR 0,0:RETURN

3440    REM LIRE  C2
3450    C1$=C2$
3460    IF C1$="," THEN RETURN
3470    C2$=MID$(EX$,PL,1):PL=PL+1:RETURN

3570    REM PRIOR(C$)
3580    PRI=0:FOR IK=1 TO 8
3590  1  IF PR$(IK)=C$ THEN PRI=PR(IK):RETURN
3600    NEXT IK:ER=8:RETURN

3700    REM TABLE DE PRIORITE
3710    PR$(1.0)="#":PR(1.0)=0
3720    PR$(2.0)="(":PR(2.0)=1
3730    PR$(3.0)="+":PR(3.0)=2
3740    PR$(4.0)="-":PR(4.0)=2
3750    PR$(5.0)="*":PR(5.0)=3
3760    PR$(6.0)="/":PR(6.0)=3
3770    PR$(7.0)=")":PR(7.0)=4
3780    PR$(8.0)=",":PR(8.0)=5
3790    RETURN
3800    L$=MID$(EX$,1,1):LK=LEN(EX$)-3:EX$=RIGHT$(EX$,LK)
3805    IF RIGHT$(EX$,1)<>":" THEN EX$=EX$+":"
3810    LS=ASC(L$)-64:IF LS=(-4) OR LS=(-2) THEN LS=LS+32
3820    S$=A$(LS,15.0):IF S$="" THEN 3840
3825    GOSUB 3900:A$(YZ,15.0)=S$
3830    II=YZ:EX$=S$:GOSUB 52800
3840    CURSOR 0,0:PRINT SPC(58);:CURSOR 0,0:RETURN
3900    LL=LEN(S$)-1:FOR IK=0 TO LL
3910  1  FOR JJ=0 TO LK STEP 3:C1$=MID$(EX$,JJ,1):C2$=MID$(EX$,JJ+1,1)
3915  2    IF MID$(EX$,JJ+2,1)<>":" THEN PRINT " erreur de syntaxe ";:S$="":GOTO
      2    3950
3920  2    IF MID$(S$,IK,1)=C1$ THEN S$=LEFT$(S$,IK)+C2$+RIGHT$(S$,LL-IK)
3930  1  NEXT JJ
3940    NEXT IK
3950    RETURN
4000    IF YY=21.0 THEN 4040
4010    IF YY<21.0 THEN YY=YY+1
4020    IF YZ<30 THEN YZ=YZ+1
4030    RETURN
4040    IF YZ<30 THEN YZ=YZ+1:GOSUB 4500
```

```
4050    RETURN
4100    IF YY=1 THEN 4140
4110    IF YY>1 THEN YY=YY-1
4120    IF YZ>1 THEN YZ=YZ-1
4130    RETURN
4140    IF YZ>1 THEN YZ=YZ-1:GOSUB 4500
4150    RETURN
4200    IF XX=0 THEN 4240
4210    IF XX>0 THEN XX=XX-1
4220    IF XZ>0 THEN XZ=XZ-1
4230    RETURN
4240    IF XZ>0 THEN XZ=XZ-1:GOSUB 4500
4250    RETURN
4300    IF XX=6 THEN 4340
4310    IF XX<6 THEN XX=XX+1
4320    IF XZ<14 THEN XZ=XZ+1
4330    RETURN
4340    IF XZ<14 THEN XZ=XZ+1:GOSUB 4500
4350    RETURN
4500    FOR KK=1 TO 21
4510  1  GOSUB 4600
4520    NEXT KK
4530    GOSUB 5000:RETURN
4600    E$(KK)=MID$(R$(KK+YZ-YY),(XZ-XX)*8,60):RETURN
5000    POKE #75,32:POKE #74,1:PRINT CHR$(12);
5001    COLORT 9 9 9 9:FOR HK=#BFE8 TO #BF6D STEP -2:POKE HK,#FF:NEXT
5002    PRINT MID$(TTA$,(XZ-XX)*8,60);
5003    FOR IK=1 TO 22:AA=#BFEF-#86*(IK-1)-4:JJ=63+IK+YZ-YY:IF JJ=92 OR JJ=94
        THEN JJ=JJ-32
5004    POKE AA,JJ:POKE AA-3,#FF:NEXT IK:PRINT
5005    IF INIT=0 THEN 5075
5008    FOR JJ=1 TO 21:PRINT E$(JJ):NEXT JJ:GOTO 5075
5010    PRINT :FOR JJ=1 TO 30
5020  1  LK=LEN(A$(JJ,0.0)):A$=SPC(12-LK)+A$(JJ,0.0)
5025  1  FOR IK=1 TO 14:IF A(JJ,IK)>0.0 THEN AA=A(JJ,IK):SA$="":GOSUB 5500
5030  2   IF A(JJ,IK)<0.0 THEN AA=-A(JJ,IK):SA$="-":GOSUB 5500
5040  2   IF A(JJ,IK)=0.0 THEN 5060
5050  2   A$=A$+SPC(8-LK)+AA$+ESP$:GOTO 5065
5060  2   LS=LEN(A$(JJ,IK)):IF LS>8.0 THEN LS=8:A$(JJ,IK)=RIGHT$(A$(JJ,IK),8):
      2   GOTO 5065
5061  2   IF LS>0.0 THEN A$=A$+SPC(8-LS)+A$(JJ,IK)+ESP$:GOTO 5065
5062  2   A$=A$+SPC(8)+ESP$
5065  1  NEXT:PRINT A$;:IF LEN(A$)<80.0 THEN PRINT
5066  1  IF FLFOR=1.0 THEN PRINT A$(JJ,15.0)
5068  1  IF FLFOR=0.0 THEN PRINT
5070    NEXT:CURSOR 0,0:PRINT SPC(50);
5075    COLORT 9 15 14 3
5080    POKE #74,0:POKE #75,#FF:RETURN
5500    AA$=""
5510    QK=AA/10:RK=AA-10*QK:AA$=CHR$(48+RK)+AA$:AA=QK:IF QK>0 THEN 5510
5520    AA$=SA$+AA$:LK=LEN(AA$):IF LK>8 THEN AA$="overflow":LK=8
5530    RETURN
6000    CURSOR 0,1:PRINT "Load.Save.Calcul.Decal.Transla.Ins.Print.Fin.Zero ! =
        z ? <>";:CURSOR 0,0
6005    PK=GETC:IF PK=0 THEN 6005
6010    PRINT CHR$(PK);
6015    IF PK=80 THEN 6300
6020    IF PK=83 THEN GOSUB 7000:GOTO 6200
6030    IF PK=76 THEN GOSUB 8000:GOTO 6200
6040    IF PK=67 THEN GOSUB 9000:GOTO 6200
```

```
6050    IF PK=84 THEN GOSUB 12000:GOTO 6200
6060    IF PK=68 THEN GOSUB 13000:GOTO 6200
6070    IF PK=33 THEN XZ=XX:YZ=YY:GOSUB 4500:GOTO 6200
6080    IF PK=61 THEN XZ=XX+8:YZ=YY:GOSUB 4500:GOTO 6200
6090    IF PK=63 THEN XZ=XX+8:YZ=YY+9:GOSUB 4500:GOTO 6200
6100    IF PK=122 THEN XZ=XX:YZ=YY+9:GOSUB 4500:GOTO 6200
6110    IF PK=73.0 THEN GOSUB 10000:GOTO 6200
6120    IF PK=90 THEN GOSUB 11500:GOTO 6200
6130    IF PK=62 THEN GOSUB 6500:GOTO 6200
6140    IF PK=60 THEN GOSUB 6600:GOTO 6200
6200    CURSOR 0,1:PRINT SPC(59);:CURSOR 0,0:RETURN
6300    PRINT :INPUT " AVEC FORMULES O/N ";T$:FLFOR=0:IF T$="O" OR T$="O " THEN
        FLFOR=1
6301    POKE #131,0:PRINT :PRINT CHR$(27);"-";CHR$(1);CHR$(14);T$(0.0);:PRINT
        CHR$(27);"-";CHR$(0);
6310    PRINT :ESP$=" ":PRINT CHR$(15);:GOSUB 5010:POKE #131,1:ESP$="":GOSUB
        5000:GOTO 6200
6500    INPUT "Recopier (a droite) jusqu'a ";T$:IF T$="" THEN T$="14"
6510    TK=VAL(T$):KK=YZ:SA=A(KK,XZ):FOR JJ=XZ+1 TO TK:A(KK,JJ)=SA:SS=SA:GOSUB
        20000:A$(KK,JJ)=S$:GOSUB 15000:NEXT JJ
6520    KK=YY:GOSUB 4600:CURSOR 0,23-KK:PRINT E$(KK);:RETURN
6600    INPUT "Recopier (a gauche) jusqu'a ";T$:IF T$="" THEN T$="1"
6610    TK=VAL(T$):KK=YZ:SA=A(KK,XZ):FOR JJ=XZ-1 TO TK STEP -1:A(KK,JJ)=SA:SS=
        SA:GOSUB 20000:A$(KK,JJ)=S$:GOSUB 15000:NEXT JJ
6620    KK=YY:GOSUB 4600:CURSOR 0,23-KK:PRINT E$(KK);:RETURN
7000    CURSOR 0,1:PRINT SPC(59);:CURSOR 0,1:PRINT "titre  I=IDEM(que
        precedent)  S=SKIP  R=REW 1  N=annul.save";
7001    CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:INPUT T$:CURSOR 0,2:IF T$="I" AND
        T$(0)<>"" THEN 7006
7002    IF T$="R" THEN CALLM #F000:REM REW2:LOOK
7003    IF T$="S" THEN CALLM #F000:REM SKIP
7004    IF T$="I" OR T$="S" OR T$="R" THEN 7000
7005    T$(0)=T$:IF T$="" OR T$="N" THEN RETURN
7006    SAVEA T$ T$(0.0)+" "+DATE$
7010    SAVEA A$
7020    RETURN

8000    REM CURSOR 0,1:PRINT SPC(59);
8005    LOADA T$ :CURSOR 0,1:PRINT SPC(58);:CURSOR 0,1:PRINT T$(0.0);
8006    CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:PRINT " <S>=SKIP 1   <R>=REWIND 1
        <C>=GARDER FORM  <space>=LOAD ";
8007    P!=GETC:IF P!=0 THEN 8007
8008    IF P!=67 THEN GOSUB 8600:GOTO 8014
8009    FLAGFO=0
8012    IF P!=83.0 THEN 8050
8013    IF P!=82 THEN 8070
8014    LOADA A$ :GOSUB 8900
8016    FOR IK=1 TO 30:LK=LEN(A$(IK,0.0)):R$(IK)=SPC(12-LK)+A$(IK,0.0):FOR JJ=1
        TO 14:S$=A$(IK,JJ):LK=LEN(S$):GOSUB 800:A(IK,JJ)=SS
8017  2   IF LK=0 THEN S$=SPC(8)
8018  2   GOSUB 20030:R$(IK)=R$(IK)+S$
8019    NEXT JJ:NEXT IK
8020    GOSUB 4500
8021    IF FLAGFO=1 THEN GOSUB 8700:GOTO 8030
8022    GOSUB 8900:II=1
8023    EX$=A$(II,15.0):GOSUB 52800
8025    II=II+1:IF II<31 THEN 8023
8030    RETURN
8050    CALLM #F000:REM SKIP1
8060    GOTO 8000
```

```
8070    CALLM #F000:REM REW3
8075    GOTO 8000
8600    FOR II=0 TO 30:PIL$(II)=A$(II,15):NEXT II:FLAGFO=1
8610    RETURN
8700    FOR II=0 TO 30:A$(II,15)=PIL$(II):NEXT II
8710    RETURN
8900    CURSOR 0,0:PRINT SPC(59);:CURSOR 0,0:PRINT "Busy    Even geduld
        Attendre svp!";:RETURN

9000    REM CALCULE
9005    CURSOR 0,0:PRINT " Je suis occupe ";
9030    FOR JJ=1 TO 14:PP=0
9100  1  GOSUB 55000
9300    NEXT JJ
9450    CURSOR 0,0:PRINT SPC(58);:CURSOR 0,0
9460    GOSUB 4500
9500    RETURN
10000   INPUT "NSERER A PARTIR DE LA LIGNE No ";I$:IF I$<="9" THEN IK=VAL(I$):
        I$=CHR$(IK+64):GOTO 10010
10005   IK=ASC(I$)-64:IF IK=(-2) OR IK=(-4) THEN IK=IK+32
10010   FOR JJ=30 TO IK+1 STEP -1
10020 1  FOR KK=0 TO 14:A$(JJ,KK)=A$(JJ-1.0,KK):A(JJ,KK)=A(JJ-1.0,KK):NEXT KK:
      1  R$(JJ)=R$(JJ-1.0)
10025 1  IMIN=IK
10030   T$=A$(JJ-1.0,15.0):GOSUB 10500:A$(JJ,15.0)=T$:NEXT JJ:GOSUB 4500
10040   II=IMIN
10050   EX$=A$(II,15):GOSUB 52800
10060   II=II+1:IF II<31 THEN 10050
10100   RETURN
10500   LK=LEN(T$):IF LK=0 THEN RETURN
10505   FOR KK=0 TO LK-1:C$=MID$(T$,KK,1)
10510 1  IF C$>=I$ AND C$<=CHR$(94) THEN T$=LEFT$(T$,KK)+CHR$(ASC(C$)+1)+
      1  RIGHT$(T$,LK-KK-1)
10520   NEXT KK:RETURN
11000   FOR KK=0 TO 14:A$(JJ,KK)=A$(IK,KK):A(JJ,KK)=A(IK,KK):NEXT:RETURN
11500   CURSOR 1,0:PRINT SPC(56);:CURSOR 0,0:INPUT "ZERO LIGNE OU COLONNE
        (L/C/ R=fin )";T$
11505   IF T$<>"L" AND T$<>"C" THEN CURSOR 0,0:PRINT SPC(59);:RETURN
11510   IF T$="L" THEN GOSUB 11600:GOTO 11500
11520   IF T$="C" THEN GOSUB 11700:GOTO 11500
11600   INPUT L$:IF (L$<"A" AND L$<>"<" AND L$<>">") OR L$>CHR$(94) THEN RETURN
11610   JJ=ASC(L$)-64:IF JJ=(-2) OR JJ=(-4) THEN JJ=JJ+32
11620   FOR KK=0 TO 14:A$(JJ,KK)="":A(JJ,KK)=0:NEXT:R$(JJ)=SPC(124):GOSUB 4500:
        RETURN
11700   INPUT JJ:FOR KK=1 TO 30:A$(KK,JJ)="":A(KK,JJ)=0:S$=SPC(8):GOSUB 15000:
        NEXT:GOSUB 4500:RETURN
12000   INPUT IK:INPUT " vers ";JJ:CURSOR 0,0:IF IK>14 OR JJ>14 OR IK<0 OR JJ<0
        THEN 12000
12010   FOR KK=1 TO 30:SS=A(KK,IK):A(KK,JJ)=SS:S$=A$(KK,IK):A$(KK,JJ)=S$:IF S$=
        "" THEN GOSUB 20000
12012 1  IF LEN(S$)<>8.0 THEN GOSUB 21000
12015   GOSUB 15000:NEXT KK
12020   GOSUB 4500:RETURN
13000   INPUT DA$:INPUT " vers ";AA$:IF (DA$<"A" AND DA$<>"<" AND DA$<>">") OR
        DA$>CHR$(94) OR (AA$<"A" AND AA$<>"<" AND AA$<>">") OR AA$>CHR$(94)
        THEN RETURN
13010   IK=ASC(DA$)-64:IF IK=(-2) OR IK=(-4) THEN IK=IK+32
13020   JJ=ASC(AA$)-64:IF JJ=(-2) OR JJ=(-4) THEN JJ=JJ+32
13030   GOSUB 11000
13035   R$(JJ)=R$(IK)
```

```
13040   GOSUB 4500:RETURN

14000   REM LISTE DES TTITRES
14005   DIM T$(0,0):IK=0
14010   IK=IK+1:POKE #131,0:LOADA T$ :PRINT IK;" - ";T$(0,0)
14020   CALLM #F000:REM SKIP1
14030   POKE #131,1
14040   GOTO 14010
15000   IF LEN(R$(KK))<124.0 THEN R$(KK)=R$(KK)+SPC(124-LEN(R$(KK)))
15010   R$(KK)=LEFT$(R$(KK),JJ*8+4)+S$+RIGHT$(R$(KK),112-8*JJ):RETURN
20000   S$="":IF SS=0.0 THEN 20030
20005   SI=1:IF SS<0 THEN SI=-1:SS=-SS
20010   QQ=SS/10:RR=SS MOD 10:S$=CHR$(RR+48)+S$:IF QQ<>0 THEN SS=QQ:GOTO 20010
20015   IF SI<0.0 THEN S$="-"+S$
20020   IF LEN(S$)>8 THEN S$="overflow":LK=8:RETURN
20030   S$=SPC(8-LEN(S$))+S$:LK=8:RETURN
21000   IF LEN(S$)>8.0 THEN S$=RIGHT$(S$,8):GOTO 21020
21010   S$=SPC(8-LEN(S$))+S$
21020   RETURN
25000   S$="":IF SS=0 THEN RETURN
25010   SI=1:IF SS<0 THEN SI=-1:SS=-SS
25020   QQ=SS/10:RR=SS MOD 10:S$=CHR$(RR+48)+S$:IF QQ<>0 THEN SS=QQ:GOTO 25020
25030   IF SI<0 THEN S$="-"+S$
25040   IF LEN(S$)>8 THEN S$="overflow"
25050   RETURN
52800   LEI=LEN(EX$)-1:IF LEI<2 THEN LINE$=LEFT$(STR$(55005.0+II*10),6):GOSUB
        53000:LINE$=LEFT$(STR$(55006.0+II*10),6):GOSUB 53000:RETURN
52805   IK=0
52810   C$=MID$(EX$,IK,1):IF C$="#" THEN EX$=RIGHT$(EX$,LEI):LEI=LEI-1:GOTO
        52810
52815   IF C$="," THEN EX$=LEFT$(EX$,LEI):LEI=LEI-1
52816   IF C$="[" THEN EX$=LEFT$(EX$,IK)+"A0%"+RIGHT$(EX$,LEI-IK):LEI=LEI+2
52820   IF C$="<" THEN EX$=LEFT$(EX$,IK)+"A1%"+RIGHT$(EX$,LEI-IK):LEI=LEI+2
52830   IF C$="]" THEN EX$=LEFT$(EX$,IK)+"A2%"+RIGHT$(EX$,LEI-IK):LEI=LEI+2
52840   IF C$=">" THEN EX$=LEFT$(EX$,IK)+"A3%"+RIGHT$(EX$,LEI-IK):LEI=LEI+2
52850   IK=IK+1:IF IK<=LEI+1 THEN 52810
52860   LINE$=LEFT$(STR$(55005.0+II*10.0),6)+"SS%="+EX$:GOSUB 53000
52865   LINE$=LEFT$(STR$(55006+II*10),6)+"A%("+STR$(II)+",JJ%)=SS%:GOSUB 25000:
        A$("+STR$(II)+",JJ%)=S$"
52867   LINE$=LINE$+":GOSUB 21000:KK%="+STR$(II)+":GOSUB 15000"
52870   GOSUB 53000:RETURN
52900   IJ=1
52910   LINE$=LEFT$(STR$(55500+IJ*10),6):GOSUB 53000:LINE$=LEFT$(STR$(55000+
        IJ*10),6)
52920   IF IJ>26 THEN LINE$=LINE$+"A"+CHR$(21+IJ):GOTO 52940
52930   LINE$=LINE$+CHR$(64+IJ)
52940   LINE$=LINE$+"=A%("+STR$(IJ)+",JJ%)"
52945   GOSUB 53000
52950   IJ=IJ+1:IF IJ<31 THEN 52910
52960   STOP
53000   LINE$=LINE$+CHR$(13)+CHR$(0):A=VARPTR(LINE$):A=PEEK(A)+PEEK(A+1)*256+1:
        B=A+LEN(LINE$)
53010   POKE #A2,A MOD 256:POKE #A3,A SHR 8:POKE #A4,B MOD 256:POKE #A5,B SHR 8:
        POKE #135,2:CALLM #F800:POKE #135,0:RETURN

55000   REM *******
55010   A=A(1.0,JJ)
55020   B=A(2.0,JJ)
55030   C=A(3.0,JJ)
55040   D=A(4.0,JJ)
```

```
55050   E=A(5.0,JJ)
55060   F=A(6.0,JJ)
55070   G=A(7.0,JJ)
55080   H=A(8.0,JJ)
55090   I=A(9.0,JJ)
55100   J=A(10.0,JJ)
55110   K=A(11.0,JJ)
55120   L=A(12.0,JJ)
55130   M=A(13.0,JJ)
55140   N=A(14.0,JJ)
55145   SS=J+K
55146   A(14.0,JJ)=SS:GOSUB 25000:A$(14.0,JJ)=S$:GOSUB 21000:KK=14.0:GOSUB 15000
55150   O=A(15.0,JJ)
55155   SS=((B*D/100+E)*I*F/1000/J+G)*103.42/C+H+0.9
55156   A(15.0,JJ)=SS:GOSUB 25000:A$(15.0,JJ)=S$:GOSUB 21000:KK=15.0:GOSUB 15000
55160   P=A(16.0,JJ)
55165   SS=((B*D/100+E)*I*F/1000/K+G)*103.42/C+H+0.9
55166   A(16.0,JJ)=SS:GOSUB 25000:A$(16.0,JJ)=S$:GOSUB 21000:KK=16.0:GOSUB 15000
55170   Q=A(17.0,JJ)
55175   SS=((B*D/100+E)*I*F/1000/L+G)*103.42/C+H+0.9
55176   A(17.0,JJ)=SS:GOSUB 25000:A$(17.0,JJ)=S$:GOSUB 21000:KK=17.0:GOSUB 15000
55180   R=A(18.0,JJ)
55185   SS=(B*D/100+E)*F/J/C*0.10342+0.9
55186   A(18.0,JJ)=SS:GOSUB 25000:A$(18.0,JJ)=S$:GOSUB 21000:KK=18.0:GOSUB 15000
55190   S=A(19.0,JJ)
55195   SS=(B*D/100+E)*F/K/C*0.10342+0.9
55196   A(19.0,JJ)=SS:GOSUB 25000:A$(19.0,JJ)=S$:GOSUB 21000:KK=19.0:GOSUB 15000
55200   T=A(20.0,JJ)
55205   SS=(B*D/100+E)*F/L/C*0.10342+0.9
55206   A(20.0,JJ)=SS:GOSUB 25000:A$(20.0,JJ)=S$:GOSUB 21000:KK=20.0:GOSUB 15000
55210   U=A(21.0,JJ)
55215   SS=(B*D/100+E)*F*I/J/C*0.10342+0.9
55216   A(21.0,JJ)=SS:GOSUB 25000:A$(21.0,JJ)=S$:GOSUB 21000:KK=21.0:GOSUB 15000
55220   V=A(22.0,JJ)
55225   SS=(B*D/100+E)*F*I/K/C*0.10342+0.9
55226   A(22.0,JJ)=SS:GOSUB 25000:A$(22.0,JJ)=S$:GOSUB 21000:KK=22.0:GOSUB 15000
55230   W=A(23.0,JJ)
55235   SS=(B*D/100+E)*F*I/L/C*0.10342+0.9
55236   A(23.0,JJ)=SS:GOSUB 25000:A$(23.0,JJ)=S$:GOSUB 21000:KK=23.0:GOSUB 15000
55240   X=A(24.0,JJ)
55250   Y=A(25.0,JJ)
55260   Z=A(26.0,JJ)
55270   A0=A(27.0,JJ)
55280   A1=A(28.0,JJ)
55290   A2=A(29.0,JJ)
55300   A3=A(30.0,JJ)
56000   RETURN
```

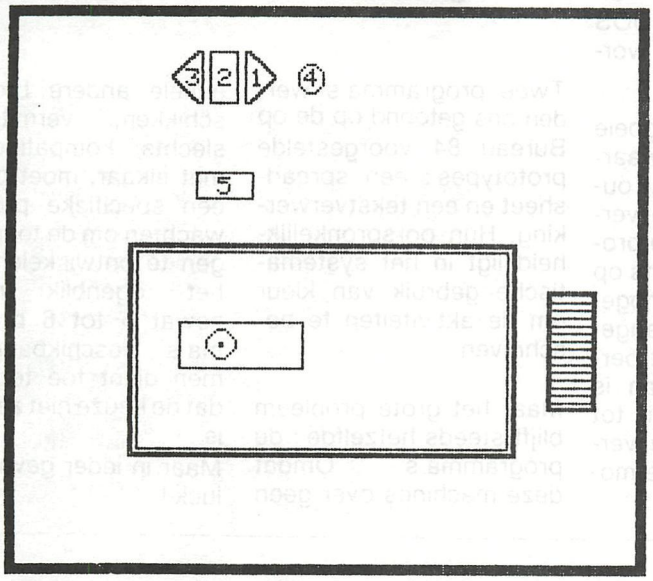
DEMO

TARASCON ete 83

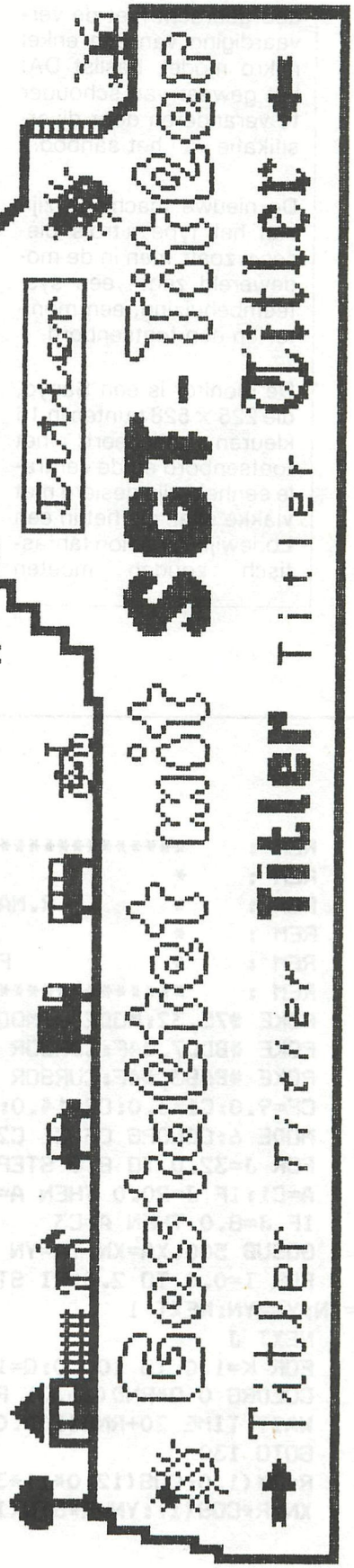
```
DEBUT        1/ 4     5/ 5     8/ 6     1/ 9     1/10    15/10     1/ 4    25/ 5     1/ 9
  FIN        4/ 5     7/ 6    30/ 8    30/ 9    14/10    31/10    24/ 5    30/ 8    30/ 9
    AVION   12500    12500    13230    12500    12500    12500    12500    13230    12500
 TRANSFERT   2000     2000     2000     2000     2000     2000      800      800      800
 COEFF.BEN.    82       82       82       82       82       82       82       82       82
   CHANGE      56       56       56       56       56       56       56       56       56
    NUITS       4        4        4        4        4        4        4        4        4
HOTELS   :  SEKIS    SEKIS    SEKIS    SEKIS    SEKIS    SEKIS   DEURTH   DEURTH   DEURTH
   DOUBLE    1600     2300     2900     2900     2300     1600     1600     1650     1650
#,
   SINGLE    2000     2800     3400     3400     2800     2000     2100     2200     2200
#,
 SUP REPAS    600      600      600      600      600      600      500      500      500
#,

#,
 TOTAL DBL  22098    24075    26691    25770    24075    22098    21250    22312    21391
#(C+(D+I*G)*F/100)*103.42/E+400,
   SINGLE   23228    25488    28103    27183    25488    23228    22663    23866    22945
#(C+(D+J*G)*F/100)*103.42/E+400,
  DBL/jour   1130     1624     2048     2048     1624     1130     1130     1165     1165
#I*F*1.0342/E,
  SGL/jour   1412     1977     2401     2401     1977     1412     1483     1553     1553
#J*F*1.0342/E,
 SUP REPAS    423      423      423      423      423      423      353      353      353
#K*F*1.0342/E,
 DEBUT      1/ 4     5/ 5     8/ 6     1/ 9    15/10     1/ 4     8/ 6     1/ 9     1/10
#,
 FIN        4/ 5     7/ 6    30/ 8    14/10    31/10     7/ 6    30/ 8    30/ 9    31/10
#,
  HOTELS    BECH     BECH     BECH     BECH     BECH    SEKIM    SEKIM    SEKIM    SEKIM
#,
 TRANSFERT  2000     2000     2000     2000     2000     4000     4000     4000     4000
#,
   DOUBLE    1400     2050     2750     2750     1400      550      600      600      550
#,
   SINGLE    1600     2700     3400     3400     1600      800      900      900      800
#,
 SUP REPAS    750      750      750      750      750
#,
   AVION    12500    12500    13230    12500    12500    12500    13230    12500    12500
#,
 TOTAL DBL  21532    23369    26267    25346    21532    20544    21606    20685    20544
#(Y+(U+V*G)*F/100)*103.42/E+400,
   SINGLE   22098    25205    28103    27183    22098    21250    22453    21532    21250
#(Y+(U+W*G)*F/100)*103.42/E+400,
  DBL/jour   988     1447     1942     1942      988      388      423      423      388
#V*F*1.0342/E,
  SGL/jour  1130     1906     2401     2401     1130      565      635      635      565
#W*F*1.0342/E,
 SUP REPAS    529      529      529      529      529
#X*F*1.0342/E,
```

Hardy → DCR-Funktionsanzeige ← Strobel

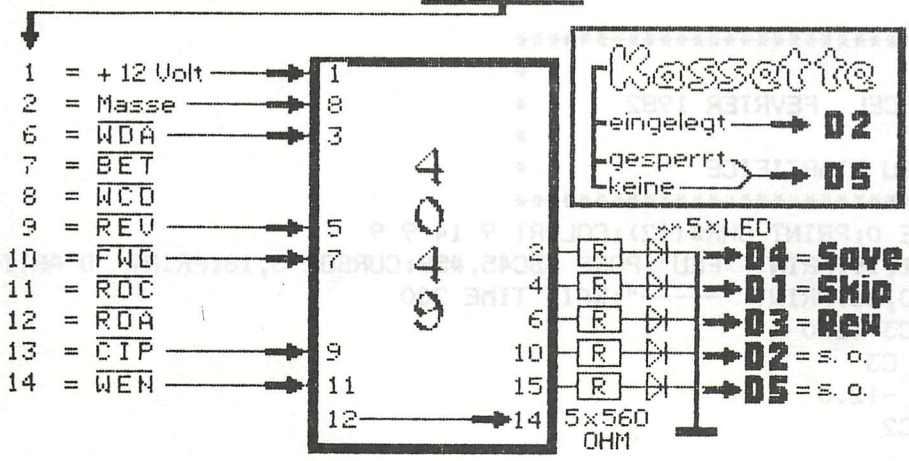
Mit 5 Leuchtdioden alles in Blick ! ! !



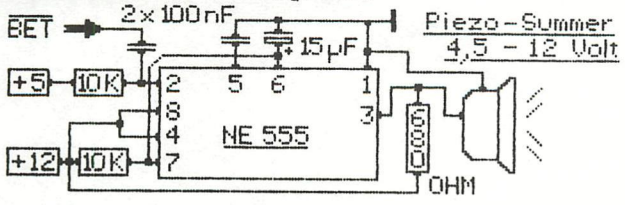
DCR-FRONTANSICHT



In der DCR geht ein Flachbandkabel von der Zusatzplatine zur Laufwerksplatine, dort ist es angelötet Die Anschlußpunkte sind numeriert.



Bandende-Piepser :



Layout : Hardy Strobel, Neuselsbrunn 51, 8500 Nürnberg 50

Ausführung : Jean Marchand

De nieuwe DAI's

Na lange jaren te hebben doorgebracht met de vervaardiging van een enkel mikro model, beslist DAI het geweer van schouder te veranderen door diversifikatie van het aanbod.

De nieuwe machines zijn van het type « trois pièces » zoals men in de modewereld zegt : een systeembehuizing, een monitor en een toetsenbord.

De monitor is een Sanyo, die 225 × 528 punten in 16 kleuren afficheert; het toetsenbord en de centrale eenheid zijn gesierd met vlakke skai, die het in een Lodewijk XV salon fantastisch zouden moeten doen. Vier modellen worden voorgesteld : de DAI-T, die eeneenvoudige terminal is, en de DAI-T 1, 2 of 3, die monoposten zijn met 1, 2 of 3 diskette-drives van 5,25'', die door de DOS V3.0 elk met 640 K worden toebedeeld.

De CPU is steeds die goeie ouwe 8080. Naar waarheid krijgt hij wel wat ouderdom, maar bij Intel verzekert men, dat deze processor meestal slechts op 5 of 10 % van zijn mogelijkheden wordt aangewend; indien DAI dit percentage opvoert, dan is het mogelijk, dat hij tot prestaties komt, die vergelijkbaar zijn met de moderne kringen.

Twee programma's werden ons getoond op de op Bureau 84 voorgestelde prototypes : een spreadsheet en een tekstverwerking. Hun oorspronkelijkheid ligt in het systematische gebruik van kleur om de aktiviteiten te beschrijven.

Maar het grote probleem blijft steeds hetzelfde : de programma's. Omdat deze machines over geen enkele andere bron beschikken, vermits ze slechts kompatibel zijn met elkaar, moet men op een specifieke produktie wachten om de toepassingen te ontwikkelen. Voor het ogenblik moeten zowat 5 tot 6 programma's beschikbaar zijn; men dient toe te geven, dat de keuze niet zeer ruim is.

Maar in ieder geval, good luck ! □

```
2     REM :    ****************************************
3     REM :    *                                      *
4     REM :    *        R.MARCEL  FEVRIER 1982        *
5     REM :    *                                      *
6     REM :    *            FEU D'ARTIFICE            *
7     REM :    ****************************************
100   POKE #75,32:MODE 6:MODE 0:PRINT CHR$(12):COLORT 9 14 9 9
110   POKE #BDD7,#4F:CURSOR 1,19:PRINT "FEU":POKE #BC45,#5F:CURSOR 3,16:PRINT "D'ARTIFICE"
115   POKE #BAB3,#4F:CURSOR 0,13:PRINT "-----":WAIT TIME 200
120   CF=9.0:C1=3.0:C2=14.0:C3=15.0
130   MODE 6:COLORG CF C1 C2 C3
140   FOR J=32.0 TO 8.0 STEP -12.0
150   A=C1:IF J=20.0 THEN A=C2
160   IF J=8.0 THEN A=C3
170   GOSUB 500:XA=XN:YA=YN
180   FOR I=0.0 TO 2.0*PI STEP PI/128.0:GOSUB 500:DRAW XMAX/2-XA,YMAX/2-YA XMAX/2-XN,YMAX/2-YN A
:XA=XN:YA=YN:NEXT I
190   NEXT J
220   FOR K=1.0 TO 100.0:Q=1.0-Q
230   COLORG 0 Q*RND(16.0) RND(16.0) RND(16.0)
240   WAIT TIME 20+RND(100.0):NEXT
250   GOTO 130
500   R=J*(1.0-COS(12.0*I)*3.0)
510   XN=R*COS(I):YN=R*SIN(I):RETURN
```

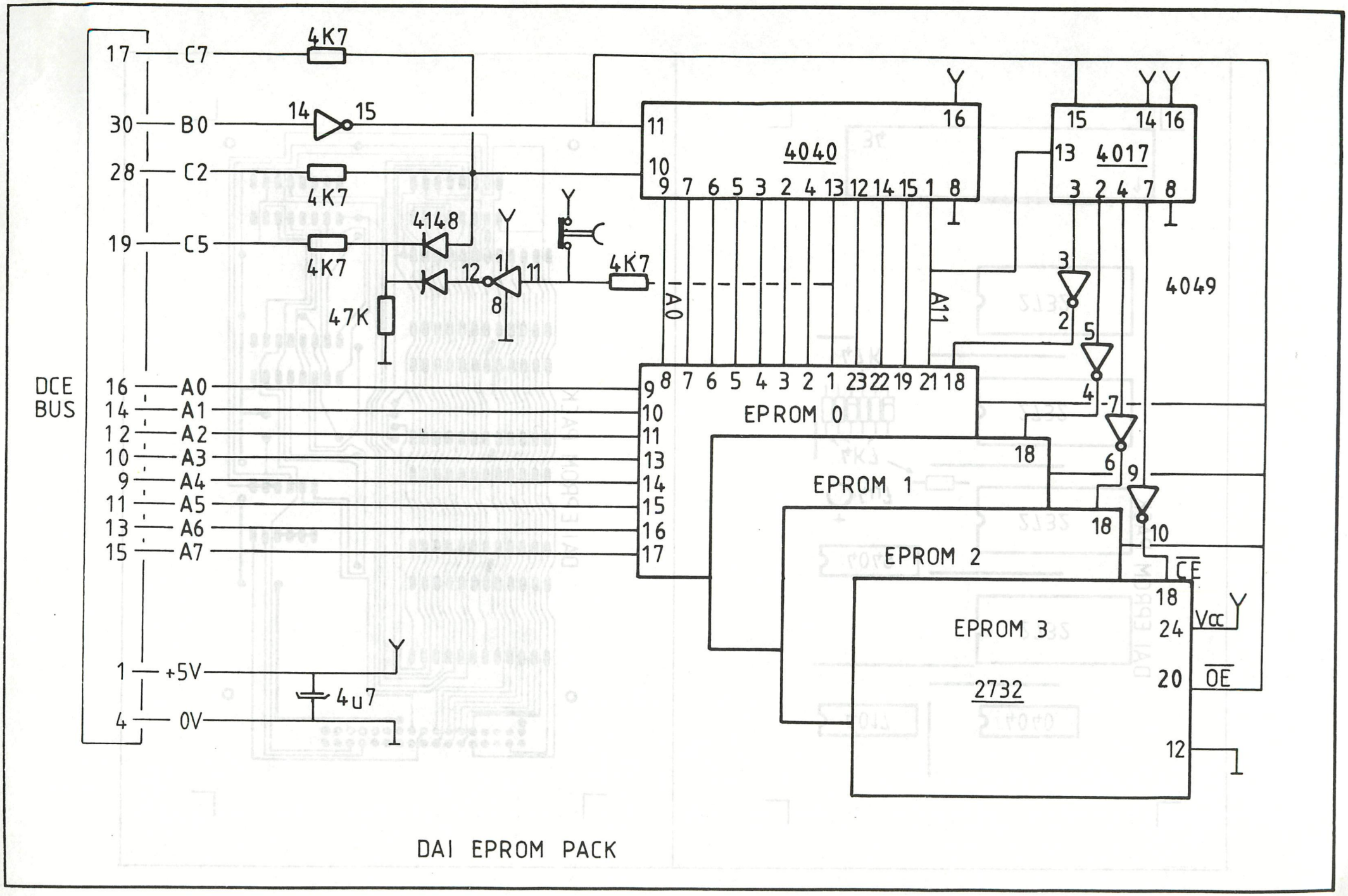
EPROM PACK

```
1       ;
2       ;
3       ;
4       ;WAARSCHIJNLIJK HEEFT IEDERE DAI-GEBRUIKER ZICH WEL
5       ;EENS GEERGERD AAN HET STEEDS WEER MOETEN LADEN VAN DE
6       ;VEEL GEBRUIKTE ROUTINES OF PROGRAMMA'S. MET NAME
7       ;DIEGENEN DIE VEEL MET ASSEMBLER WERKEN EN EEN AUDIO-
8       ;CASSETTERECORDER ALS OPSLAGMEDIUM GEBRUIKEN.
9       ;DE DCE-BUS WORDT TIJDENS DE POWER-UP ROUTINE OF NA
10      ;EEN RESET DOOR MIDDEL VAN EEN IN DE FIRMWARE OPGENO-
11      ;MEN ROUTINE GECONTROLEERD OP EEN EVENTUEEL AANGESLO-
12      ;TEN APPARAAT DAT WIL COMMUNICEREN.
13      ;INDIEN DIT ZO IS, ZAL DE OP DE BUS AANGEBODEN DATA OP
14      ;DE STACK GEPLAATST WORDEN. DEZE DATA MOET EEN PRO-
15      ;GRAMMA ZIJN, DAT IN DE RUIMTE VAN DE STACK PAST (F800
16      ;-F900). DIT PROGRAMMA WORDT DOOR DE DCE-BOOTSTRAP GE-
17      ;START. MET BEHULP VAN DIT PROGRAMMA, DAT OP ZIJN
18      ;BEURT OOK WEER EEN BOOTSTRAP KAN ZIJN, WORDT DE DATA
19      ;OPGEHAALD VAN DE GEWENSTE ROUTINE OF HET GEWENSTE
20      ;PROGRAMMA.
21      ;NATUURLIJK MOET DEZE BOOTSTRAP DE DATA OP DE JUISTE
22      ;PLAATS GAAN SCHRIJVEN. DUS ZAL HET BEGIN-ADRES EN
23      ;HET EIND-ADRES VAN DE ROUTINE OF VAN HET PROGRAMMA
24      ;MEEGEGEVEN MOETEN WORDEN.
25      ;VERDER IS ER EEN STUK HARDWARE NODIG OM DE IN DE
26      ;EPROM OPGESLAGEN DATA OP DE DCE-BUS TE DOEN ZETTEN.
27      ;DE BOVENGENOEMDE FILOSOFIE HEEFT GELEID TOT DE ONT-
28      ;WIKKELING VAN HET DAI-EPROM-PACK.
29      ;DEZE ONTWIKKELING WORDT DOOR ENKELE DAI-GEBRUIKERS,
30      ;DIE ZICH MET ASSEMBLER (DNA EN SPL) BEZIG HOUDEN,
31      ;AL ENIGE TIJD TOEGEPAST. NA HET AANZETTEN VAN DE DAI-
32      ;MACHINE IS BIJVOORBEELD SPL DIRECT BESCHIKBAAR.
33      ;DE LAADTIJD WORDT IN DE PRAKTIJK NIET BEMERKT, OMDAT
34      ;DEZE SLECHTS 1-2 SEC. BEDRAAGT
35      ;ER ZIJN DIVERSE BOOTSTRAP PROGRAMMA'S IN GEBRUIK.
36      ;EEN TWEE STUKS ZIJN HIER AFGEDRUKT.
37      ;DE EERSTE VERSIE IS MINDER DAN 64 BYTES GROOT EN
38      ;LAADT EEN AANEENGESLOTEN BLOK VAN EEN OF MEERDERE
39      ;PROGRAMMA'S. DE WEERSTAND VAN 4K7 BIJ HET VERBREEK-
40      ;CONTACT IS DAN OP PIN 4 VAN DE CD4040 AANGESLOTEN.
41      ;DE TWEEDE VERSIE IS GROTER DAN 64 BYTES EN IS BEDOELD
42      ;OM GESCHEIDEN PROGRAMMA'S OF ROUTINES TE LADEN OP
43      ;TEVOREN BEPAALDE ADRESSEN. DE WEERSTAND VAN 4K7 IS
44      ;DAN AANGESLOTEN ZOALS HET SCHEMA AANGEEFT OP PIN 13
45      ;VAN DE CD4040.
46      ;ER WORDT GEWERKT AAN EEN BOOTSTRAP WAARBIJ ER DIRECT
47      ;EEN MENU VERSCHIJNT, WAARDOOR MEN KAN KIEZEN UIT
48      ;BIJVOORBEELD: A-BASIC  B-SPL  C-TEKSTVERWERKEN.
49      ;EEN ADVIES MET BETREKKING TOT DE HARDWARE:
50      ;DE C-MOS IC'S EN DE EPROMS ZIJN SOMS MOEILIJK
51      ;VERKRIJGBAAR, WAARDOOR DE PRIJZEN HOOG KUNNEN LIGGEN.
52      ;ONDERZOEK DUS EERST DE VERKRIJGBAARHEID VAN DEZE COM-
53      ;PONENTEN. HET IS NIET NOODZAKELIJK DIRECT ALLE EPROMS
54      ;TE INSTALLEREN, MET EEN WERKT HET OOK.
55      ;
```

```
56      ;DE HARDWARE.
57      ;DE MAXIMALE GEHEUGENCAPACITEIT IS 16K. (4X(4KX8))
58      ;HIER GAAT DE BOOTSTRAP VAN AF.

SPL

59      ;HET SCHEMA GEEFT DE PENAANSLUITINGEN VAN DE DCE-BUS
60      ;WEER MET DE FUNCTIE ER VAN.
61      ;POORT  A(0-7)      DATABUS
62      ;       C(2,5,7)    CONTROLEERT DE COMMUNICATIE
63      ;       B(0)        RESET
64      ;
65      ;C7 (IN)            IS ER DATA GELDIG? INIEN C7="1" DAN
66      ;                   WORDT DE DATA OP POORT A INGELEZEN.
67      ;B0 (OUT)           RESET (SOFTWARE)
68      ;C2 (OUT)           DATA-REQUEST
69      ;C5 (IN)            DATA-PRESENT.
70      ;                   INDIEN C5="1" WORDT ER NAAR DE
71      ;                   DCE-INPUTROUTINE GESPRONGEN.
72      ;A0-A7 (IN)         DATA IN
73      ;
74      ;DRUKTOETS          INDIEN HET CONTACT IS GEOPEND TIJDENS
75      ;                   DE POWER-UP ROUTINE OF EEN RESET WORDT
76      ;                   DE EPROM INHOUD GELADEN.
77      ;
78      ;4017B              DECADE COUNTER
79      ;4040B              BINARY COUNTER (12-STAGE  2↑12=4096)
80      ;4049B              BUFFER (6X)
81      ;2732               EPROM (4KX8=4096)
82      ;
83      ;BRONNEN
84      ;DAINAMIC           '80 NR.1  BLZ 3-4   DE DCE-BUS
85      ;                   N.B.: IN DE TEKENING DE 1 EN DE 2 VAN
86      ;                         P1 EN P2 VERWISSELEN.
87      ;
88      ;                   '82 NR.9  BLZ 9     DCE-CONNECTIONS
89      ;                                       JOS SCHEPENS.
90      ;
91      ;                   '83 NR.14 BLZ 28-29 INITIALISATION
92      ;                                       DCE-PERIPHERALS
93      ;                   N.B.: ADRES #EFA7-#EFB2 PORT A
94      ;                         MOET ZIJN PORT B
95      ;
96      ;INTEL              2732 EPROM
97      ;DATA-SHEETS        8255 P.P.I.
98      ;
99      ;FIRM-WARE          DCE-INPUTROUTINE
100     ;MANUAL             B.J. BOERRIGTER
101     ;
102     ;
```

```
103             TITL    'BOOTSTRAP A.J.'
104     ;
105     HEAP    EQU     2ECH
106     PPI     EQU     0FE00H
107     STACK   EQU     0F800H
108     ;
109             ORG     0F800H
110     ;
111             LHLD    ENDA            ;EXECUTED AFTER
112             XCHG                    ;RESET
113             LHLD    STARTA
114     ;
115     START   LDA     PPI             ;LOAD DATA
116             MOV M,A                 ;STORE DATA
SPL V1.1  PAGE 3        BOOTSTRAP A.J.
117             MOV A,H                 ;)CHECK
118             SUB D                   ;)
119             JZ      NOT             ;)FOR
120             MOV A,L                 ;)
121             SUB E                   ;)END-
122             JZ      ENDR            ;)ADDRESS
123     ;
124     NOT     MVI A   4H              ;RESET PC2
125             STA     PPI+3H
126     NOG     LDA     PPI+2H          ;)WAIT
127             ANI     80H             ;)C7
128             JNZ     NOG             ;)LOW
129             MVI A   5H              ;←PC2
130             STA     PPI+3H          ;←HIGH
131             INX H
132             JMP     START
133     ENDR    LXI H   HEAP
134             RET
135             ORG     0F82FH
136     STARTA  DW      8400H
137     ENDA    DW      0B2FFH
138             END
139     ;
140     ;
141             TITL    'BOOTSTRAP J.A.P. 840109'
142     ;
143     ;*1* IF HEAPPOINTERS MUST NOT BE CHANGED,
144     ;*1* PUT IN THIS 2 INSTRUCTIONS.
145     ;
146             ORG     0F800H
147     COUNT1  EQU     297H
148     ;
149             NOP                     ;(E5) PUSH H  *1*
150             LXI H   TABLE
151             PUSH H
152     ;
153     BLOCK   MOV C,M                 ;LOW-
154             INX H                   ;--START ADDRESS
155             MOV B,M                 ;HIGH-
156             INX H
157             MOV E,M                 ;LOW-
158             INX H                   ;--END ADRESS
159             MOV D,M                 ;HIGH-
160             INX H
161             SHLD    COUNT1
162             PUSH B                  ;(
163             POP H                   ;( BE TO HL
164             CALL    COMPAR
```

```
                                                ;HL=DE
165             JNZ     START1                  ;NOT THEN START1
166             POP H                           ;YES THEN POP END ADDRESS
167             LDA     0FE01H                  ;!
168             ANI     0FEH                    ;!
169             STA     0FE01H                  ;!RESET EPROMPACK
170             NOP                             ;(E1) POP H  *1*
171             RET                             ;END OF ROUTINE
172     ;
173     START1  POP B                           ;KILL SAVED END ADDRESS
174             LXI B   0FE03H
SPL V1.1  PAGE 4        BOOTSTRAP J.A.P. 840109

175     START2  LDA     0FE00H                  ;GET DATA
176             MOV M,A                         ;STORE DATA
177             MVI A   4H                      ;) RESET PC2
178             STAX B                          ;)        =INCR. COUNTER
179             INR A                           ;) A=05
180             STAX B                          ;) SET PC2
181             INX H                           ;) NEXT ADDRESS
182             CALL    COMPAR                  ;END ADDRESS REACHED?
183             JNZ     START2                  ;NOT THEN START2
184             PUSH H                          ;YES THEN PUSH END ADDRESS
185             LHLD    COUNT1
186             JMP     BLOCK
187     ;
188     ;COMPARE HL-DE.
189     COMPAR  MOV A,H
190             SUB D
191             RNZ
192             MOV A,L
193             SUB E
194             RET
195     ;
196     ;START- AND END-ADDRESS TABLE.
197     ;
198     TABLE   DW      800H                    ;STARTADDRESS (FIRST PART)
199             DW      810H                    ;EN ADDRESS+1
200             DW      83EH                    ;STARTADDRESS (SECOND PART)
201             DW      843H                    ;END ADDRESS+1
202             DW      900H                    ;STARTADDRESS (THIRD PART)
203             DW      999H                    ;END ADDRES+1
204     ;       ''      ''                      ;''     ''     ''
205     ;       ''      ''                      ;''     ''     ''
206             DW      0H                      ;) AT THE END ALWAYS
207             DW      0H                      ;) 8X ZERO
208     ;
209             END
```



DAI EPROM PACK

DAI EPROM PACK
4040
2732
2732
2732
2732
1
4017
4049
4u7
4K7
47K
34

DAI EPROM PACK

ERRATA DATAFLEX

ERRATTA DATAFLEX ERRATTA DATAFLEX ERRATTA DATAFLEX ERRATTA DATAF

Mr Siegfried Brys meld ons een bug in het Dataflex programma op de Dataflex diskette. Er zijn problemen wanneer een bestand vol is en men toch probeert het aan te vullen.(Ø - ENTER) Volgende wijzigingen worden voorgesteld :

```
1000 FOR P%=USED%+1 TO LENGTH%:R$=""
```

wordt :

```
1000 IF USED%=LENGTH% GOTO 1075
1005 FOR P%=USED%+1 TO LENGTH%:R$=""
```

en :

```
1070 GOSUB  950:NEXT P%:GOSUB 300:PRINT "FILE IS FULL,  IT NEEDS
EXTENDING !";
```

wordt :

```
1070 GOSUB 950:NEXT P%
1075 GOSUB 300:PRINT "FILE IS FULL, IT NEEDS EXTENDING !";
```

!!! Type : *UT, >Z3, >B voor U editeert. !!!

Mocht U problemen hebben hiermee stuur Uw copy terug, wij zenden U de gewijzigde versie terug.

Gedurende de boodschap 'FILE IS FULL ....' moet U de spatiebalk indrukken om terug in het menu te komen.

Mocht er nog iemand op een ei zitten, aarzel niet in Uw pen te kruipen.

Couwberghs Frans
Boekdonkstraat 13
3980 Tessenderlo
013/666340

FEESTWENSEN

1985

```
2     MODE 0:PRINT CHR$(12)
3     COLORT 8 5 6 14
4     FOR I=#BFEE TO #B3E4 STEP -#86:POKE I,#F0+RND(15):NEXT
5     A$="          +-                                                  ":GOSUB 100
10    A$="         +$*-                                                 ":GOSUB 100
20    A$="        +*****-                    +-                        ":GOSUB 100
30    A$="       +******$*-                 +**-                       ":GOSUB 100
40    A$="      +$********-    +-           +***$*-                    ":GOSUB 100
50    A$="         !!        +$*-         +*******-                   ":GOSUB 100
60    A$="                  +*****-      +$********-                  ":GOSUB 100
70    A$="                 +******$*-   +**********$**-                ":GOSUB 100
80    A$="    +-           +********$**-     !!                        ":GOSUB 100
81    A$="   +**-         +*********$******-                           ":GOSUB 100
83    A$="  +*$**-       +**********$*****-       PRETTIGE            ":GOSUB 100
85    A$=" +*******-    +*$*************$***-       KERSTDAGEN         ":GOSUB 100
86    A$="+*$******$*- +*****************$**-        & EEN VOORSPOEDIG  ":GOSUB 100
87    A$="*****$*************$*************$******-          1985      ":GOSUB 100
90    A$="    !!    +**********************-                            ":GOSUB 100
91    A$="                  !!                                         ":GOSUB 100
92    A$="                                                             ":GOSUB 100
94    A$="                                                             ":GOSUB 100
95    A$="=============================================================":GOSUB 100
96    A$="door u toegewenst door:                    Jeroen Overvoorde ":GOSUB 100
98    A$="=============================================================":GOSUB 100
99    GOTO 99
100   FOR I=0 TO 59:W=ASC(MID$(A$,I,1)):IF W=32 THEN PRINT " ";:NEXT
110   IF W=42 THEN PRINT CHR$(127);:NEXT
120   IF W=43 THEN PRINT CHR$(6);:NEXT
130   IF W=45 THEN PRINT CHR$(7);:NEXT
140   IF W=33 THEN PRINT " ";:XX=CURX*2:YY=CURY*#86:POKE #B3E5-XX+YY-9,#FF:NEXT
145   IF W=61 THEN PRINT CHR$(1);:NEXT
150   IF W=36 THEN PRINT CHR$(127);:XX=CURX*2:YY=CURY*#86:POKE #B3E5-XX+YY-9,56:NEXT
155   IF I>59 THEN PRINT :RETURN
160   POKE #B3E4+CURY*#86,#CD:POKE #B3E4+(CURY-1)*#86,#C8
170   PRINT CHR$(W);:NEXT
```

DAInamic software a program for e

☐ Games collection 1	400 Bfr
☐ Games collection 2	400 Bfr
☐ Games collection 3	400 Bfr
☐ Games collection 4	800 Bfr
☐ Games collection 5	400 Bfr
☐ Games collection 6	750 Bfr
☐ Games collection 7	750 Bfr
☐ Games collection 8	750 Bfr
☐ Games collection 9	750 Bfr
☐ Games collection 10	750 Bfr
☐ Games collection 11	750 Bfr
☐ Games collection 12	750 Bfr
☐ DNA assembly pack	1100 Bfr
☐ Fast graph text	1000 Bfr
☐ FGT applications	1000 Bfr
☐ Toolkit 1	1000 Bfr
☐ Toolkit 2	1000 Bfr
☐ Toolkit 3	1000 Bfr
☐ Toolkit 4	1000 Bfr
☐ Toolkit 5	1000 Bfr
☐ Primary Education 1	1000 Bfr
☐ Math' fun 1	1000 Bfr
☐ Secundary Education 1	1000 Bfr
☐ Secundary Education 2	1000 Bfr
☐ Mathematics 3	1000 Bfr
☐ Bits & Bytes	750 Bfr
☐ Mailing List	1000 Bfr
☐ Graphic Tablet	1000 Bfr
☐ Music collection 1	300 Bfr

☐ Music collection 2	
☐ Music collection 3	
☐ DAI Tiny Pascal	
☐ English-German	
☐ DAI DEMO + Ba	
☐ Sargon Chess	
☐ Space Invaders	
☐ Tape 80-81	
☐ Newsletter 10	
☐ Newsletter 11-12	
☐ Newsletter 13-14	
☐ Centipede	
☐ Driver	
☐ Super Invader	
☐ DAI PANIC	
☐ MICRO'S-Onder	
☐ SPL Macro-asse	
☐ Taal-tape 1	
☐ Fysica 1	
☐ Familiebudget	
☐ Grafische Hulp	
☐ Acrobates	
☐ Character Gener	
☐ Fast Word Proce	
☐ PAC-MAN	
☐ DAYLAXIANS	
☐ PUZZLY	
☐ DUEL	
☐ C.L.I.O.	

very application

2	300 Bfr
3	300 Bfr
	1000 Bfr
rainer	1000 Bfr
ic tutor	500 Bfr
	1500 Bfr
	800 Bfr
	850 Bfr
	500 Bfr
	650 Bfr
-15	650 Bfr
	600 Bfr
	600 Bfr
	600 Bfr
	800 Bfr
vijs	990 Bfr
nbler	1100 Bfr
	750 Bfr
	750 Bfr
	500 Bfr
	500 Bfr
	600 Bfr
tor	1750 Bfr
sor	2000 Bfr
	1100 Bfr
	2100 Bfr
	2100 Bfr
	2100 Bfr
	3000 Bfr

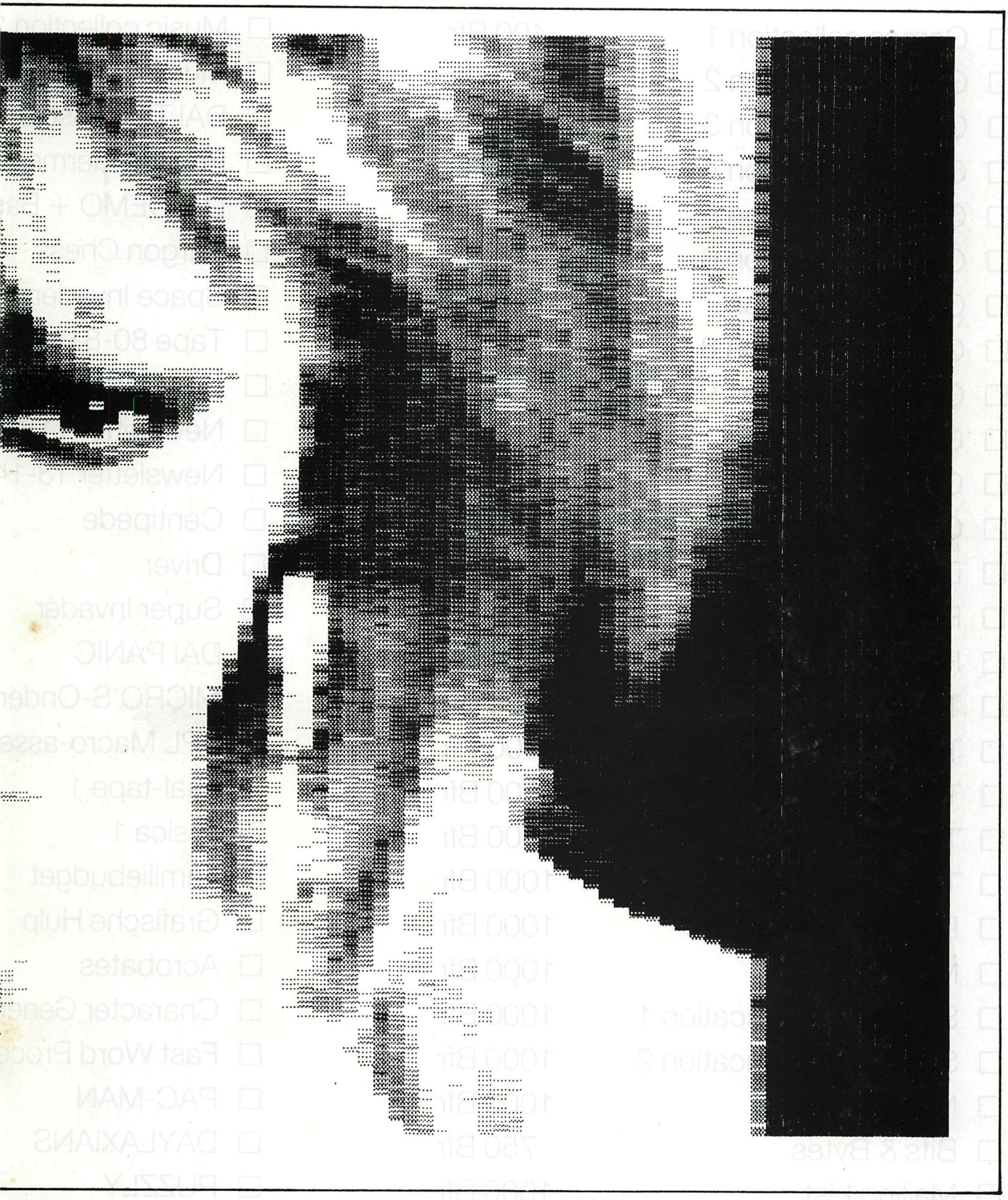
☐ DAIpc SCHEMATICS	850 Bfr
☐ BEST of DAInamic (80-81)	500 Bfr
☐ DAInibble	800 Bfr
☐ Education 6	1000 Bfr
☐ Education 7	1000 Bfr
☐ Education 8	1000 Bfr
☐ Frogger	1250 Bfr
☐ Eagles	1000 Bfr
☐ Phoenix	1250 Bfr
☐ Tangram	750 Bfr
☐ Turtle-Basic	1250 Bfr
☐ Toolkit 6	1000 Bfr
☐ MIX 1	500 Bfr
☐ QUEST adventure	600 Bfr
☐ Sociale Geografie	1000 Bfr
☐ D-BASIC	2000 Bfr
☐ Math'fun 2	1000 Bfr
☐ DOS-TOOLKIT (disquette)	1250 Bfr
☐ DATAFLEX (disquette)	1500 Bfr
☐ Fysische Geografie	1000 Bfr
☐ Boekhoudprogramma	2000 Bfr
☐ Games collection 13	750 Bfr
☐ Games collection 14	750 Bfr
☐ SFGT	1000 Bfr
☐ Firmware manual	1350 Bfr
☐ Superbase	—
☐ Patrouiller	—
☐ Compilation 83-84	—

indicated prices are for audio, DCR + 150 Bfr

A HAPP

from DAIna

Y 1985



mic TEAM

FRACTALS

```
PAGE 01  --  FRACTALS SOUND

10      REM fractals with sound
15      MODE 0:PRINT CHR$(12):COLORT 0 5 0 0
20      INPUT "Geef A met 3<A<3.95";A!:PRINT
30      IF A!<3.0 THEN A!=3.0:IF A!>3.95 THEN A!=4.0
40      INPUT "DUUR VAN TONEN 0<TIJD<10";TIJD:PRINT
50      IF TIJD<0.0 THEN TIJD=0
60      IF TIJD>10 THEN TIJD=10
65      PRINT "START/STOP met spatiebalk"
70      G=GETC:IF G<>32 THEN GOTO 70
75      Z!=RND(1.0)

80      REM REPEAT UNTIL SPACE-BAR IS PRESSED
90      Z!=A!*Z!*(1.0-Z!)
100     SOUND 0 1 15 0 FREQ(1000.0*Z!)
105     WAIT TIME TIJD
110     G=GETC:IF G<>32 THEN GOTO 90
120     SOUND OFF
130     GOTO 15
```

```
PAGE 01  --  FRACTALS GRAPHIQUE

5       MODE 0:PRINT CHR$(12):COLORT 0 5 0 0
10      INPUT "Geef beginwaarde A met 1<A<4 bv 2.8 of 3.7";AINT!:PRINT
20      IF AINT!<1 OR AINT!>4 THEN GOTO 5
25      PRINT CHR$(12)
30      PRINT "Geef maximum van A met ";AINT!;:INPUT "<A<4 bv 3.99 of 3.9";
        AMAX!:PRINT
40      IF AMAX!<AINT! OR AMAX!>4.0 THEN GOTO 25
50      COLORG 0 5 10 15:MODE 6
70      SCALE!=(AMAX!-AINT!)/XMAX
80      Z!=RND(1.0)
90      FOR X=0 TO XMAX
100   1  A!=X*SCALE!+AINT!
110   1  IF A!<2.95 THEN CYCLE=2
120   1  IF A!>=2.95 THEN CYCLE=20
130   1  FOR Y=1 TO CYCLE:Z!=A!*Z!*(1.0-Z!):NEXT Y
140   1  FOR Y=1 TO CYCLE
150   2    Z!=A!*Z!*(1.0-Z!)
160   2    Z1=Z!*YMAX:DOT X,Z1 5
170   1  NEXT Y
180     NEXT X
200     END
```

FWP PATCHES 2

The advantage of the following patches is that the "K" and the "X" command can be executed now, whilst marker 01 is also present in buffer 1 (see last DAInamic, FWP TIP).

The following addresses, starting from 0959, have to be changed.

0959 79 FE 07 C8 FE 06 C8 37 C9

It is not necessary to change address 0966.

The above changes still leave the possibility of setting more than once the same marker, so you are more flexible, however multiple markers can introduce errors. For example in case you give a "K" command with 2 markers 07, be aware that all the text between marker 06 and the last marker 07 will be deleted !

If you don't want to have the markers more than once (without getting an error message), change the the addresses below.
The above mentioned feature is also incorporated in these patches.

0941 0C 00

0948 0C 0C 00 00

0951 0C 0C 0C 00

0959 79 FE 05 C8 FE 06 C8 37 C9

In case you prefer "start printing" always from the first line in the buffer, instead of the top of the current STEP, change the next 3 addresses.

073E **21 00 30** (**21 00 90** for buffer 2)

Note: If you ever experienced a stack-overflow condition during working with FWP it may help to change the content of address 1313 in 00.
This is a small bug in the program (1313 is not a joke !!)

Ger Gruiters 25-10-1984

HEAP ORGANISATIE

Numansdorp
29.05.84

Beste DAInamic vrienden.

Hier een klein programma, dat de vrije en gebruikte heapruimte bepaalt. De opzet was dit als subroutine in een groter programma te gebruiken om te zien hoeveel heapruimte er nog over was. Wij vinden het nog steeds een nadeel van de DAI dat de FRE functie alleen werkt op de totale programmaruimte en niet ook apart op de heap. Tijdens het uitproberen van dit programma kwamen wij tot een beter inzicht in de oranisatie van de heap. Tezamen met het artikel van Jan Boerrigter in DAInamic Nr. 7 Blz. 188 e.v. geeft dit een beter inzicht in de manier waarop de DAI met de heap omspringt. Dat dit niet altijd even economisch is zal blijken. Type het programma in zoals het is, geef RUN en RETURN en na het vraagteken een paar maal alleen RETURN, daarna bijv. een A en RETURN en een paar maal alleen RETURN. Hier uit blijkt, dat bij een INPUT de heap pas na de derde INPUT op de zelfde string wordt gereorganiseerd. Dit verbetert aanmerkelijk als we de string eerst leeg maken met A$="":INPUT A$, in regel 20. Verander het begin ook eens als volgt:

```
 5 CLEAR 1000
10 DIM A$(10)
20 FOR A=0 TO 10:INPUT A$(A):NEXT
```

en verander de GOTO op regel 2000 beurtelings in GOTO 10 en GOTO 20. Hopend dat U net als wij hierdoor een beter gebruik van de heap kunt maken volgt nu het programma:

```
*IMP INT
*   5 CLEAR #100
*  10 INPUT A$
*1000 PRINT
*1010 HEAP=PEEK(#29B)+PEEK(#29C)*256
*1020 HSIZE=PEEK(#29D)+PEEK(#29E)*256
*1030 HEND=HEAP+HSIZE
*1040 PP=PEEK(HEND-1)+PEEK(HEND-2)*256
*1050 PRINT "HEAP  ","HSIZE ","H-END ","ENDPTR"
*1060 PRINT HEAP,HSIZE,HEND,PP
*1070 PRINT " #";HEX$(HEAP)," #";HEX$(HSIZE)," #";
     HEX$(HEND)," #";HEX$(PP)
*1080 PRINT
*1090 FOR HP=HEAP TO HEND
*1100 PO= PEEK(HP+1)+PEEK(HP)*256
*1110 FH=(PO+PP) IAND #FFFF
*2000 IF FH=HEND-HP-5 THEN PRINT "BYTES LEFT: ";FH;
     "  USED: ";HSISE-FH:GOTO 10
*2010 NEXT:PRINT "NO POINTERS LEFT !!!":GOTO 5
*RUN
```

De DAI gebruikt soms wel de ruimte voor de pointers voor dat de melding OUT OF STRING SPACE wordt gegeven.
Met vriendelijke groeten:

Fredrik en Frits Chabot

```
002                     ORG   :300
003                VAR  EQU   :2EC
004                HEND EQU   VAR+2
005                PP   EQU   HEND+2
006                HP   EQU   PP+2
007                FH   EQU   HP+2
008                *
009 0300 F3        START DI
010 0301 C5              PUSH  B
011 0302 D5              PUSH  D
012 0303 E5              PUSH  H
013 0304 F5              PUSH  PSW
014 0305 3600            MVI   M,:0
015 0307 23              INX   H
016 0308 3600            MVI   M,:0
017 030A 23              INX   H          HET ADRES VAN DE VAR.
018 030B 22EC02          SHLD  VAR        OPSLAAN
019                * HEND BEREKENEN EN OPSLAAN VOOR LATER GEBRUIK
020 030E 2A9B02          LHLD  :29B
021 0311 EB              XCHG
022 0312 2A9D02          LHLD  :29D
023 0315 19              DAD   D
024 0316 22EE02          SHLD  HEND
025                * DE POINTER OPHALEN DIE AAN HET EIND V/D HEAP STAAT
026 0319 2B              DCX   H
027 031A 5E              MOV   E,M
028 031B 2B              DCX   H
029 031C 56              MOV   D,M
030 031D EB              XCHG
031 031E 22F002          SHLD  PP
032                * HP=PEEK(#29B ...       ( VOOR DE LOOP VAN START HEAP
033 0321 2A9B02          LHLD  :29B         TOT EIND HEAP )
034 0324 22F202          SHLD  HP
035                *DIT IS DE LOOP WAARIN GEKEKEN WORDT OF DE
036 0327 2AF202    FOR   LHLD  HP           AANGETROFFEN BYTE'S OVEREEN
037 032A 56              MOV   D,M          KOMEN MET HET ADRES WAAR ZE
038 032B 23              INX   H            STAAN
039 032C 5E              MOV   E,M          DE 2 BYTE'S IN  (DE)
040 032D 2AF002          LHLD  PP           (HL)=PP
041 0330 19              DAD   D            (HL)=(HL)+(DE)
042 0331 22F402          SHLD  FH           FH=(HL)
043 0334 EB              XCHG               (HL) <=> (DE)     [ (DE)=FH ]
044 0335 2AF202          LHLD  HP           (HL)=HP
045 0338 CD7903          CALL  SUB          (BC)=-(HL)
046 033B 2AEE02          LHLD  HEND         (HL)=HEND
047 033E 09              DAD   B            ;                 [ (HL)=HEND-HP ]
048 033F 01FBFF          LXI   B,:0-:5
049 0342 09              DAD   B            ;                 [ (HL)=(HL)-5 ]
050 0343 CD7903          CALL  SUB          (BC)=-(HL)
051 0346 EB              XCHG               (HL) <=> (DE)     [ (HL)=FH ]
052 0347 09              DAD   B            ;    [ (HL)=FH-(HEND-HP-5) ]
053 0348 CD8003          CALL  HL           (HL) 0 ?
054 034B CA6803          JZ    TVAR         JA  GOTO TVAR
055 034E 2AEE02    NEXT  LHLD  HEND         NEE HET VOLGENDE ADRES
056 0351 CD7903          CALL  SUB          PROBEREN
057 0354 2AF202          LHLD  HP
058 0357 23              INX   H            HP=HP+1
059 0358 22F202          SHLD  HP
060 035B 09              DAD   B
061 035C CD8003          CALL  HL           EIND HEAP AL BEREIKT ?
062 035F C22703          JNZ   FOR          NEE  NEXT
063 0362 21FFFF          LXI   H,:FFFF      JA.DAN ONTBREKEN DE POINTERS
```

```
PAGE 02    FRE HEAP       F.H. CHABOT Jr. & Jr.

064 0365 22F402           SHLD  FH
065                  *
066 0368 2AF402      TVAR LHLD  FH
067 036B 23               INX   H
068 036C EB               XCHG
069 036D 2AEC02           LHLD  VAR
070 0370 72               MOV   M,D        FRE IN DE VAR ACHTER DE
071 0371 23               INX   H          CALLM
072 0372 73               MOV   M,E
073                  *
074 0373 F1               POP   PSW        EN RETURN
075 0374 E1               POP   H
076 0375 D1               POP   D
077 0376 C1               POP   B
078 0377 FB               EI
079 0378 C9               RET
080                  *
081                  *
082 0379 7C          SUB  MOV   A,H        (BC)=-(HL)
083 037A 2F               CMA
084 037B 47               MOV   B,A
085 037C 7D               MOV   A,L
086 037D 2F               CMA
087 037E 4F               MOV   C,A
088 037F C9               RET
089                  *
090 0380 7C          HL   MOV   A,H        (HL)=0 ?
091 0381 FE00             CPI   :0
092 0383 C0               RNZ
093 0384 7D               MOV   A,L
094 0385 FE00             CPI   :0
095 0387 C9               RET
096                  *
097 0388                  END

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

FH      02F4   FOR     0327   HEND    02EE   HL      0380
HP      02F2   NEXT    034E   PP      02F0   START   0300
SUB     0379   TVAR    0368   VAR     02EC

]#P.
0300 F3 C5 D5 E5 F5 36 00 23 36 00 23 22 EC 02 2A 9B
0310 02 EB 2A 9D 02 19 22 EE 02 2B 5E 2B 56 EB 22 F0
0320 02 2A 9B 02 22 F2 02 2A F2 02 56 23 5E 2A F0 02
0330 19 22 F4 02 EB 2A F2 02 CD 79 03 2A EE 02 09 01
0340 FB FF 09 CD 79 03 EB 09 CD 80 03 CA 68 03 2A EE
0350 02 CD 79 03 2A F2 02 23 22 F2 02 09 CD 80 03 C2
0360 27 03 21 FF FF 22 F4 02 2A F4 02 23 EB 2A EC 02
0370 72 23 73 F1 E1 D1 C1 FB C9 7C 2F 47 7D 2F 4F C9

0380 7C FE 00 C0 7D FE 00 C9

]B.BASIC V1.0
*LIST
10    CLEAR 10000
15    DIM F(10,10,10)
20    INPUT A$
30    CALLM #300,X
40    PRINT "FRE ";X
```

JEROEN DEMO

PAGE 01 -- JEROEN DEMO 7

```
10      REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
20      REM Jeroen Overvoorde Helmbloem 5 3068 AC Rotterdam
30      REM Telefoon 010-210426 Nederland  datum 1-10-1983
40      REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
50      REM ------------------------------------------------
60      REM Logo Nederlandse Omroep Stichting  NOS
70      REM ------------------------------------------------
100     MODE 6:COLORG 0 0 0 9:C=17:YM=YMAX-20:XM=XMAX:R=19
110     POKE #BFEE-13*90,#FF:POKE #BFEE-15*90,#F9
120     POKE #BFEE-209*90,#FF
200     FILL 22,YM-80 61,YM C
205     R=19:K=C:G=72:MY=YM-19:MX=92:GOSUB 1000
210     FILL 72,YM-80 111,YM-19 C
215     R=19:K=C:G=167:MY=YM-61:MX=192:GOSUB 2000:MY=YM-19:MX=141:GOSUB 3000
220     FILL 122,YM-19 167,YM-80 C:FILL 167,YM-61 211,YM C
225     R=29:K=0:MY=YM-40:MX=166:GOSUB 5000:R=20:K=C:GOSUB 5000
230     R=19:K=C:G=267:MY=YM-61:MX=292:GOSUB 2000:MY=YM-19:MX=241:GOSUB 3000
235     FILL 222,YM-19 267,YM-80 C:FILL 267,YM-61 311,YM C
240     R=29:K=0:G=222:MY=YM-71:MX=250:GOSUB 1000:G=311:MY=YM-9:MX=287:GOSUB
        4000
245     FILL 222,YM-80 279,YM-71 0:FILL 258,YM-9 311,YM 0
250     R=20:K=C:MY=YM-71:G=222:MX=250:GOSUB 1000:G=311:MY=YM-9:MX=287:GOSUB
        4000
255     FILL 222,YM-71 270,YM-80 C:FILL 267,YM-9 311,YM C
260     R=19:K=0:G=221:MY=YM-61:MX=241:GOSUB 4000:G=312:MY=YM-19:MX=292:GOSUB
        1000
265     FILL 22,YM+6 311,YM+7 23:FILL 22,YM-220 311,YM-219 23
300     FOR I=22 TO 112:DRAW 0,I+128 114,I+128 19:WAIT TIME 2:NEXT
305     FOR I=22 TO 112:DRAW I,150 I,YM 18:DRAW 114,I+128 214,I+128 19:NEXT
310     FOR I=22 TO 112:DRAW I+100,150 I+100,YM 18:DRAW 214,I+128 314,I+128 19:
        NEXT:TE=TE+1:IF TE=2 THEN GOTO 330
315     FOR I=22 TO 112:DRAW I+200,150 I+200,YM 18:DRAW 14,I+128 114,I+128 19:
        NEXT
320     GOTO 305
330     FOR I=150 TO 240:DRAW 14,I 114,I 19:DRAW 114,I-45 214,I-45 19:DRAW I+72,
        150 I+72,YM 18:NEXT
340     FOR I=195 TO 240:DRAW 114,I 214,I 19:DRAW 214,I-45 314,I-45 19:WAIT
        TIME 2:NEXT
350     FOR I=195 TO 240:DRAW 214,I 314,I 19:WAIT TIME 2:NEXT
400     WAIT TIME 200:LOAD "JEROEN DEMO 8"
1000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
1010    DRAW G,MY+Y MX+X,MY+Y K:NEXT Y
1020    RETURN
2000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
2010    DRAW G,MY-Y MX+X,MY-Y K:NEXT Y
2020    RETURN
3000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
3010    DRAW MX-X,MY+Y G,MY+Y K:NEXT Y
3020    RETURN
4000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
4010    DRAW MX-X,MY-Y G,MY-Y K:NEXT Y
4020    RETURN
5000    D=R*R:FOR Y=0 TO R:X=SQR(D-Y*Y)
5010    DRAW MX-X,MY-Y MX+X,MY-Y K:DRAW MX+X,MY+Y MX-X,MY+Y K:NEXT Y
5020    RETURN
```

LE MONITEUR BASIC

DAInamic INFO

LE MONITEUR BASIC (#C80C-#C956)

1er PARTIE : GENERALITES

Cet article décrit les grandes lignes du fonctionnement des routines du moniteur BASIC du DAI. De prochains articles permettront de détailler certaines parties du moniteur. Le DAI PC FIRMWARE MANUAL sert de référence à cet article. L'organigramme simplifié ci-joint peut vous guider au long de cet article.

Le moniteur BASIC peut etre considéré comme le coeur du système d'exploitation du DAI-BASIC. Sous son controle, des lignes de programme peuvent etre entrées dans le TEXT-BUFFER, des commandes directes peuvent etre éxécutées, l'éxecution d'un programme est controlée, un BREAK est géré,...

Le moniteur est appelé par le DAI après un RESET ou la mise sous tension à l'adresse #C818. D'autres adresses (#C80C, #C814 et #C823) sont aussi utilisées à la fin certaines routines (ceci sera explicité dans la 2em partie)

INITIALISATION (#C818-#C843) :

Le moniteur doit etre initialisé. Ceci signifie qu' une situation bien définie de départ est à creer. Le STACKPOINTER est initialisé, les flags controlants l' éxecution des INPUT, appel de programmes et de ss prgms, et de l'encodage des lignes stockées sont remis à zéro. Résultat: un DAI pret. Les interruptions clavier et horloge sont 'en ligne', sans lesquelles aucune entrée, ni aucune sortie sur écran n'est possible.

ENTREE A PARTIR DE L'EDIT BUFFER (#C846-#C84E) :

Le SWITCH d'encodage des entrées #0135 est controlé. Ce switch définie la provenance de l'entrée (l'information) qui doit etre encodée. Encodé signifie : traduction des lignes de BASIC (commandes directes ou lignes de programme) du BASIC 'lisible' en un BASIC codé lisible par le semi-compilateur du DAI permettant ainsi L1'execution de ces lignes. C'est dans ce code q'un programme est stocké dans le TEXT-BUFFER. Des détails sur ce pseudo-code peuvent etre trouvés dans le NEWSLETTER 11, page 196...

Si la source est l'EDIT-BUFFER, alors la saisie des informations à partir de ce buffer est préparé via #D879, et l'encodage intervient via #C867. Lorsqu'une ligne est encodée, le moniteur recommence de meme jusqu'a l'epuisement de l'edit-buffer.

A voir également sur ce sujet un précédent article 'EDITOR STORY'.

REMARQUE: L'encodage d'une 'cochonnerie indéfinie' (c.a.d. le contenu momentané du buffer d'entrée d'encodage) intervient aussi si le switch #0135 est >2 !!! Cette faiblesse du FIRMWARE s'explique sur la conviction que #0135 est toujours <=2.

ENTREE A PARTIR DU CLAVIER (#C851-#C864) :

Si la source d'entrée n'est pas l'edit-buffer, mais le clavier , le signe ('*') est

affiché à l'écran et une entrée de ligne BASIC est attendue via #DD1A. En toutes lettres: le DAI attend une ligne de commandes directes ou une ligne de programme BASIC à stocker dans le TEXT-BUFFER, à partir du clavier uniquement. La séquence de saisie ne peut s'achever que par un CR ('RETURN')
ou par un BREAK. Tous les caractères tapés sont affichés à l'écran La mémoire d'écran est alors à ce moment le seul emplacement ou existe l'information entrée. Lorsqu'un retour chariot CR est entré, le moniteur commence alors à travailler. Il saisie le premier caractère de la ligne entrée dans la mémoire d'écran, pour l'analyser.

La différence entre une ligne de commandes directes, et une ligne de programme à stocker dans le TEXT-BUFFER est la présence d'un no de ligne en tete de la ligne de programme. Et

si ce no existe, alors la ligne doit se coder dans le text-buffer. Dans le cas contraire, et son execution doit suivre immédiatement son encodage. Enfin, si le premier caractere est un CR, la ligne d'entrée est non valide , et le moniteur attend après une autre ligne.

ENCODAGE DE LIGNE DE PROGRAMME (#C867-#C86A) :

Si une ligne de programme a été entrée, elle est encodée et stockée dans le TEXT-BUFFER, à un emplacement correct; ce qui signifie dans l'ordre des numéros de ligne. Le moniteur est relancé pour saisir la nouvelle ligne. Des détails sur le processus d'encodage seront apportés dans un prochain article.

ENCODAGE DE LIGNE DE COMMANDES DIRECTES (#C86D-#C87D) :

La ligne de commandes directes est encodée via une routine d'encodage d'ordres BASIC via RST1/00. Le résultat de ce processus, le code 'semi-compilé' , est stocké dans le buffer d'entrées encodées EBUF #013E-01BD. Ce buffer de 128 octets explique l'erreur 'LINE TOO COMPLEX' qui intervient pour une ligne d'instructions encodées de plus de 128 octets

Le flag #0117 est activé (#FF), simulant ainsi un RUN de lignes de programme. Le registre BC, utilisé pour indiquer ou le moniteur est actif dans dans la ligne de commandes en cours, pointe sur le debut de la ligne dans le buffer d'entrées encodées EBUF, et la ligne de commandes est prete à s'executer.

EXECUTION DE COMMANDES DIRECTES OU DE LIGNES DE PROGRAMMES

Ce chapitre concerne aussi bien l'execution de commandes directes que celle de lignes de programme sockées dans le TEXT-BUFFER. La seule différence concerne le contenu du registre pair BC. Il pointe vers le buffer d'entrées codées EBUF pour l'execution de commandes directes.Et dans le cas de l'execution d'un programme stocké dans le buffer de texte, il pointe sur le début d'une ligne de texte dans ce buffer. Le premier caractère d'une ligne est alors saisi.Si c'est un code (NDT: Jan Boerrigter parle d'un 'token', soit un signe, une marque...) -une commande BASIC sous sa forme compilée- l'adresse de départ de la routine d'exécution de la commande BASIC correspondant à ce code est trouvée dans la table #CF02, et cette routine est executée via #C8A9. Dans le cas ou le premier octet n'est pas un code (#C8E5), il peut alors etre soit un '0', soit l'octet de longueur placé au début

d'une ligne de programme. Un '0' indique alors la fin du contenu du TEX-BUFFER,(ou un '0' est inséré après la dernière instruction), ou la fin d'une ligne de commandes directes (voir #C876). La rencontre d'un '0' relance le MONITEUR. Si le premier octet lu par le moniteur est l'octet de longueur d'une ligne de programme stockée, alors le TRACE/STEP FLAG est (#0115-#0116). Si l'un de ces flags est ACTIF, la nouvelle ligne de programme est affichée sur l'écran. Dans le cas du FLAG STEP, l'enfoncement de la touche SPACE est attendue.Si la touche 'BREAK' n'est pas enfoncée, le moniteur continue en #C87F.

LA FIN DE CERTAINES ACTIONS (#C8AA-#C8B5) :

Quand le code a été trouvé, quand l'execution de la routine est terminée, le moniteur retourne en #C88F. Il controle alors l'éventualité d'une fin spéciale de la routine BASIC en cours. Cette 'fin spéciale' es signalée par un code dans l'accumulateur A et la mise sur ACTIF du CARRY-FLAG après l'execution de la routine BASIC. Voici par exemple #DF03: 'STOP' : CY=1, A=3

Quatre actions sont possibles:

0: #C908-#C915 : Après 'LOAD' : if 'LOAD' n'est pas une commande directe, le programme chargé sera démarré immédiatement.

1: #C818 : CAN'T CONTINU :Après certaines commandes qui provoquent unemodification des pointeurs de programme, ou qui altèrent le contenu du TEXT-BUFFER le moniteur est relancé.

2: #C8C0-#C8C8 : Après un 'soft break' (BREAK provoqué par le programme),le BREAK doit etre pris en compte et traité.

3: #C8B8-#C8BD : Après la commande 'STOP' d'une ligne BASIC . L'endroit ou s'arrete le programme doit etre mémorisé et un 'STOPPED IN LINE....' est affiché. D'autres traitements interviennent comme pour le 'soft break'

PROGRAMMEERTECHNIEKEN (NO 13, page 301)

Voyons maintenant le sujet d'aujourd'hui: les couleurs 16 à 19 d'une part et les couleurs 20 à 23 d'autres part. Je commencerais avec ces derniers, l'utilisation des couleurs 20 à 23 servant dans le dessin en mode 4 couleurs . C'est pourtant simple . Au tout debut d'un programme , nous placons l'ordre COLORG A B C D ou A B C D sont nos quatres couleurs desirees.Dans notre programme, chaque appel de la couleur 20 donnera la Couleur A ,21 la couleur B , et ainsi de suite... C'est d'ailleurs une bonne habitude de commencer par une instruction COLORG,plutot que par une instruction MODE. Nous eviterons ainsi de voir l'ecran prendre une voliente teinte jaune (provenant du precedent ordre COLORG) plutot que le noir souhaite. Dans le mode 16 couleurs,la premiere couleur de l'ordre COLORG devient la couleur de fond,et le bord 'obtient' aussi cette couleur.Ceci non pas avec un ordre FILL 0,0 XMAX,YMAX K , mais seulement avec un nouvel ordre COLORG suivi de MODE 1,3 ou 5.L'avantage de l'habitude de surtout utiliser les codes couleurs 20 à 23 est dans le fait que, pour un nouvel usager, une mauvaise combinaison de couleurs est facile à modifier. Sur mon Moniteur RGB, j'ai l'excellente habitude de prendre les couleurs 0 1 2 3,mais je comprend que l'utilisateur d'un moniteur noir et blanc me maudisse si je choisis les couleurs 4 5 8 13 !!.

MOD. DNA-FWP

Cette petite modification apportée à DNA permet de récupérer dans le buffer de FWP le résultat de la commande]#L. de DNA; ceci en vue de l'insérer dans un texte ou à tout autre usage.

Marche à suivre :

1. Lire DNA.

2. Introduire la modification par SUBSTITUTE.

3. TRES IMPORTANT : initialiser le pointeur PTRBUF (selon le buffer que l'on veut utiliser) avec les valeurs suivantes :
   BUFFER 1 -> #1175/#1176 = 00-30
   BUFFER 2 -> #1175/#1176 = 00-90

4. >G1100

5. Lire le programme que l'on veut éditer.

6. Taper la commande]#L.

7. Taper]U. et à l'adresse pointée par PTRBUF (#1175/#1176) substituer #00 (FLAG DE FIN pour buffer de FWP).

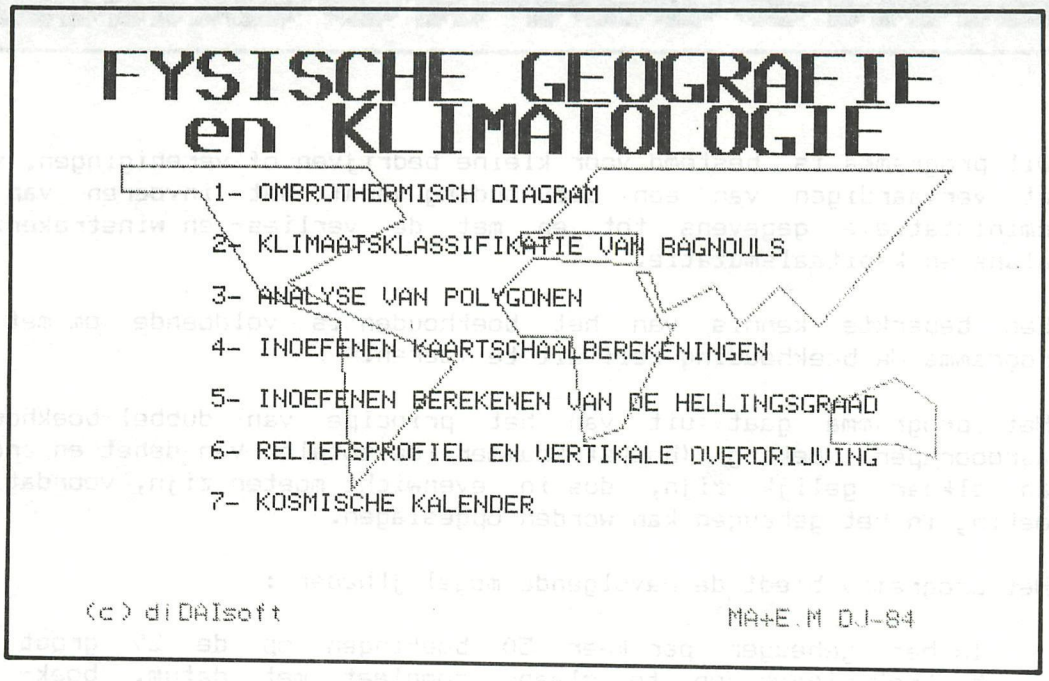
8. Remplacer le contenu de #3FF par #FF (Bypass clear buffer).

9. >G400

Ci-dessous le programme et les commentaires :

```
PAGE 01     DNA ---> BUFFER FWP

002                     PTRBUF  EQU   :1175
003                             ORG   :1EEA
004 1EEA CD5EDD                 CALL  :DD5E
005 1EED 7E             LOOP    MOV   A,M
006 1EEE E67F                   ANI   :7F
      Il faut masquer le bit 7 car FWP
      n'affiche pas les caractères >#7F.
007 1EF0 23                     INX   H
008 1EF1 E5                     PUSH  H
009 1EF2 2A7511                 LHLD  PTRBUF
010 1EF5 77                     MOV   M,A
      On garnit ici le buffer de FWP.
011 1EF6 23                     INX   H
012 1EF7 227511                 SHLD  PTRBUF
      On remet à jour le pointeur.
013 1EFA E1                     POP   H
      On restitue le pointeur pour DNA
      et on continue le traitement normal.
014 1EFB FE0D                   CPI   :0D
015 1EFD CA061F                 JZ    :1F06
016 1F00 CD7711                 CALL  :1177
017 1F03 C3ED1E                 JMP   LOOP
018 1F06                        END

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

LOOP   1EED   PTRBUF 1175
```

NEW



FYSISCHE GEOGRAFIE

Een nieuwe collectie onderwijs-programma's , verzameld door Marc Antrop.

audio : 1000 Bfr
DCR : 1150 Bfr

D - BASIC

NEW

DBASIC version 2.2 and how to get it...

W.Coremans has released a new version of DBASIC.(see also his article on p.427-430).
A few small bugs have been corrected but what is more important, 2 beautiful extensions for DBASIC are available now:

1/ a very special format listing , with extra list of extended commands & functions, procedures,functions,labels,arrays and variables. (see sample program on p.429-430).
2/ another extension, offering true programmable function keys.

If you already own the DBASIC program and you want these new facitilies together with the new version of DBASIC, send us your cassette together with 250 Bfr for mailing and administration costs.
For compatibitily reasons, only version 2.2 will be delivered in the future.

note : we look forward to receive the first DBASIC programs!

BOEKHOUDPROGRAMMA

new!

Dit programma is bestemd voor kleine bedrijven of verenigingen, voor het vervaardigen van een boekhouding vanaf het invoeren van de administatieve gegevens tot en met de verlies- en winstrekening, balans en kapitaalsmutatie.

Een beperkte kennis van het boekhouden is voldoende om met dit programma Uw boekhouding zelf uit te voeren.

Het programma gaat uit van het principe van dubbel-boekhouden waardoor per boeking (boekstuknummer) de totalen van debet en credit aan elkaar gelijk zijn, dus in evenwicht moeten zijn, voordat die boeking in het geheugen kan worden opgeslagen.

Het programma biedt de navolgende mogelijkheden :

* In het geheugen per keer 50 boekingen op de 29 groot boekrekeningen op te slaan, compleet met datum, boekstuknummer en 50 posities voor de omschrijving voor elke boeking.

* Boekingen op zoveel grootboekrekeningen als nodig tegen te boeken.

* Vergissingen onmiddellijk en eenvoudig op het scherm te corrigeren voordat wordt doorgeboekt.

* Het saldo automatisch naar een van de de BTW rekeningen te boeken door deze als sluitpost te gebruiken

* Reeds opgeslagen gegevens te controleren en naar behoefte te wijzigen.

* Gecontroleerde gegevens op papier of op het scherm of op beide af te drukken als controlelijst of journaal.

* Overzichten per grootboekrekening te vervaardigen.

* Gegevens op tape of DCR op te slaan voor later gebruik en om verder te gaan met boeken.

* Om fouten te voorkomen is printen op papier of wegladen op tape of DCR niet mogelijk als de gegevens niet in evenwicht zijn. U moet dan eerst controleren en wijzigen.

* Op elk gewenst moment een tussentijdse proef- en saldibalans op te vragen.

* Een volledige (tussentijdse) balans met resultatenoverzicht te vervaardigen, zonder dat er iets met de andere gegevens gebeurt, anders dan transporteren. Hierdoor kan bijvoorbeeld elk kwartaal een overzicht vervaardigd worden, wat voor een goede bedrijfsvoering erg belangrijk kan zijn.

* Een automatische kapitaalsmutatie, waarbij het resultaat direct naar het kapitaal geboekt wordt.(Ook tussentijds zonder gevolgen voor de gegevens).

audio:2000
dcr :2150

RAUTEN

```
PAGE 01  --  RAUTEN

30      MODE 6:COLORG 1 1 1 1

50      REM ......... FENSTERLN .... BY ROLF SCHALL
100     REM ........ BLAU-WEISS .......
110     FILL 33,21 XMAX-33,YMAX-21 21:COLORG 1 15 15 15
112     FILL XMAX/2-36,YMAX/2-24 XMAX/2+36,YMAX/2+24 23
115     FOR X0=0 TO 1000-59 STEP 59:X=X0
120   1  FOR A=0.0 TO YMAX-20.0 STEP 7.0
130   2   IF X<7 THEN 150:X=X-8:IF X<XMAX-32 THEN GOSUB 500
140   1  NEXT A
150     NEXT X0

200     REM ........ FARBWECHSEL .......
210     R=20.0:GOSUB 600
220     FOR I=1.0 TO 20:COLORG 1 15 15 1:WAIT TIME 40-I:COLORG 1 15 1 15:WAIT
        TIME 20-I:NEXT
230     COLORG 1 15 15 1:R=23:GOSUB 600
240     FOR I=1.0 TO 20:COLORG 1 15 15 1:WAIT TIME 40-I:COLORG 15 1 15 1:WAIT
        TIME 20-I:NEXT
250     COLORG 1 15 15 1:R=20:GOSUB 600
260     FOR I=1 TO 15:COLORG 1 15 RND(15) 15:WAIT TIME 30:NEXT
270     FOR I=1 TO 20:C=RND(15):COLORG C 15 C 15:WAIT TIME 30:NEXT
280     COLORG 15 15 1 1
290     GOTO 290

500     REM ....... RAUTENDRUCK .......
520     FOR I=0.0 TO 13.0:DRAW I+X,I+A I+X+20,I+A+7 16:NEXT:RETURN

600     REM ........ RAND ........
610     FILL 0,0 XMAX,20 R:FILL 0,YMAX-20 XMAX,YMAX R
620     FILL 0,21 32,YMAX-21 R:FILL XMAX-32,21 XMAX,YMAX-21 R:RETURN
```

ERRATA : numbers up

```
470   Q=RND(4.0):IF (Q+QO) MOD 4=1 GOTO 470:GOSUB 740
510   COLORT 8 8 8 9:FOR Y=0 TO B:S$="":FOR X=0 TO B*7:S$=S$+CHR$(11):NEXT
```

80 KOLOMMEN TEKST

****
*
* Wijziging: 80 kolommen tekst op de DAI
*
****

Omdat bij mij de belangstelling groot is om op korte termijn "CPM" software te gaan "draaien" op een apart systeem, waarbij de DAI als terminal moet gaan dienen. Hierbij is echter wel raadzaam dat men over een terminal beschikt met >= 80 kolommen- en >=24 rijen tekst. Als men in de toekomst op de DAI zelf "CPM" gaat "draaien" lijkt mij deze wijziging ook uiterst zinvol. Om de wijzigingen beperkt te houden zijn alle screen modes onveranderd gebleven, behalve de laagste resolutie tekst modes. Naar mijn weten wordt deze tekst modes door niemand gebruikt, zodat ik deze modes (10 kolommen tekst) omgebouwd heb naar 80 kolommen tekst, met als praktische mogenlijkheid van 24- of 25- of 26- rijen tekst. Met normaal gebruik van de DAI merkt niemand iets van de hard-ware ingreep.

Alvorens men de beslissing neemt om de DAI te gaan wijzigen moet men vast stellen dat zijn of haar machine voorzien is van snelle type dynamische Ram geheugens. Het type Ram moet zijn: 416-2 of 4116-2 of 16k2 of dergelijke 150 nS Ram ic's. Is dit echter niet het geval dan kan men de wijziging niet uitvoeren of men moet natuurlijk de geheugen ic's vervangen door alreeds 1 van aangegeven typen. Mijn machine was al voorzien van het snelle type Ram ic's. (DAI Rev. 4)

Door de nog te omschrijven wijziging ondergaat de betekenis van de lijn control words van het video screen geheugen een kleine aanpassing:

High addr.byte MODE	7 6 5 4	Video screen mode Voor wijziging	Na wijziging
2	0 0 0 0	88 Kol. 4 Kl. Graf.	Idem
4	0 0 0 1	176 Kol. 4 Kl. Graf.	Idem
6	0 0 1 0	352 Kol. 4 Kl. Graf.	Idem
(8)	0 0 1 1	528 Kol. 4 Kl. Graf.	Idem
???	0 1 0 0	11 Kol. 4 Kl. Tekst	88 Kol. 4 Kl. Tekst
-	0 1 0 1	22 Kol. 4 Kl. Tekst	Idem
-	0 1 1 0	44 Kol. 4 Kl. Tekst	Idem
0	0 1 1 1	66 Kol. 4 Kl. Tekst	Idem
1	1 0 0 0	88 Kol.16 Kl. Graf.	Idem
3	1 0 0 1	176 Kol.16 Kl. Graf.	Idem
5	1 0 1 0	352 Kol.16 Kl. Graf.	Idem
(7)	1 0 1 1	528 Kol.16 Kl. Graf.	Idem
-	1 1 0 0	11 Kol.16 Kl. Tekst	88 Kol.16 Kl. Tekst
-	1 1 0 1	22 Kol.16 Kl. Tekst	Idem
-	1 1 1 0	44 Kol.16 Kl. Tekst	Idem
-	1 1 1 1	66 Kol.16 Kl. Tekst	Idem

Om de wijzigingen hardware-matig beperkt te houden wordt er tijdens de nieuwe 80 (88) koloms tekst mode wordt er gebruik gemaakt van een andere character set. deze set is gebaseerd op een 6*10 matrix (spacing) in plaats van de in de DAI gebruikte 8*11 matrix (spacing). Dit houdt in dat de huidige character generator prom moet worden vervangen door 4k eprom van het type 2732(A) of 2532. In de ene helft van de prom zit dan de normale character set van de DAI, terwijl in de andere helft dan de aangepaste character set bevindt. Dit laatste wordt specifiek voor de 80 (88) koloms tekst modes gebruikt. De nieuwe character prom dient op een ic-voet geplaatst te worden in verband met de nog eventueele latere wijzigingen aan de character sets. De omschakeling van de character sets gebeurd hardware-matig.

Vanzelf sprekend moet de snelheids selectie prom (74S288 bruin ic 5) (resolutie besturing) vervangen worden door een nieuwe prom van een dergelijk type met een aangepaste inhoud (74S288 of 82S123).

Tevens wordt er een 74LS10 (3-voudig-input nand) circuit toegevoegd om het cycle-skipping proces (2 uit 3 overslaan) welk specifiek wordt gebruikt voor de 80 (88) koloms tekst modes te kunnen bewerkstelliggen. Het genoemde ic wordt boven op een ander ic geplaatst

De RAS timing van de geheugen chips moet worden teruggebracht van ongeveer 250

nS naar ongeveer 160 nS. Dit houdt een vervanging in van een timing conde; ator.

Verder moet een aantal print-sporen worden doorgesneden en moet er een aantal draadjes worden gelegd.

Kosten plaatje:

- Eventueel snellere geheugen ic's (150 nS)
  HCC Hardware service 4116-2 (24#) (: fl 182,40)
- Nieuwe character generator ic 2732(A) of 2532
  HCC Hardware service 2732 : fl 18,95
- Nieuwe snelheids selectie prom ic 74S288 of
  82S123 : fl 10,90
- Een ic t.b.v. cycle skipping 74LS10 : fl 2,50
- Enventueel ic voet t.b.v. character gen. (: fl 2,--)
- 1 weerstand + 3 condensatoren : fl 2,--
- montagedraad, soldeertin en verzendkosten : fl 12,--
- 1 a 2 avonden montage- en testwerk (: fl ???,??)

Is de machine van het snelle type Ram ic's voorzien, beschikt men over hardware ervaring, test- en meet- en programmeer apparatuur, is de wijziging goedkoop uit te voeren. Niet is meegerekend de kosten welke eventueel zijn verbonden aan de bijbehorende (firm-) (soft-) ware aanpassingen / toevoegingen.

De hiervoor gobaal aangegeven wijzigingen wil ik pas in details publiceren (ook in DAInamic.) en vervolgens vrijgeven nadat ik een aantal gewillige "slachtoffers" heb gevonden om hun of haar machines tegen kostprijs aan te laten passen om zodoende wat meer ervaring op te kunnen doen. Tevens wil ik dan van hun bevindingen op de hoogte gesteld te worden.

Voordat ik deze brief heb geschreven heb ik al enkele weken met de nieuwe tekst modes gespeeld. Hierbij is de machine niet "plat" gegaan ten gevolge van hardware fouten. Mijn aanbeveling is om een nieuwe tekst mode te definieeren, bestaande uit 80 kolommen en 26 rijen tekst, gebruik makend van de 4 kleuren mode. De naamgeving voor deze nieuwe screen mode stel ik dan voor: "MODE 9".(Al reeds aangegeven in tabel.)

Voor diegenen die de 80 (88) koloms tekst geheel software matig willen oplossen in een "MODE 8" omgeving door middel van een speciale "FGT" wil ik de door mij gebruikte character set beschikbaar stellen. De software oplossing heeft als belangrijke nadelen: Bijna 32 Kb geheugen in gebruik in plaats van ruim 4,6 Kb met de hardware oplossing ((88#2+2)#26+2#16), en zal bijzonder traag zijn in verband met het scrollen en het clear screen commando. Hierbij is praktisch slechts 24 rijen tekst mogenlijk.

De (K)TV- en video monitor interface kaarten behoeven niet te worden aangepast. Nodig is misschien wel dat de resolutie van met name de "pal color tv interface chart" verbetert moet worden als men de 80 (88) koloms tekst mode op een gewone KTV wil gebruiken. Bij de overige interface kaarten is de resolutie al groot genoeg. De wijzigingen die ik een ieder aanbeveel ten aanzien van de "pal color tv chart" heb ik al reeds omschreven in DAInamic 10. (Alleen uit te voeren, welke zijn omschreven in hoofdstukken 1 en 2.)

Ik hoop met de hiervoor omschreven 80 koloms aanpassing een kleine impuls te kunnen geven ten voordelen van de DAI personal computer, waardoor misschien de levenskans van deze overigs schitterende machine wat toeneemt.

Gaarne ontvang ik opmerkingen en suggesties ten aanzien van dit onderwerp.

Ik hoop op de volgende landelijke bijeenkomst van de DAI-GG te Utrecht het een en ander te kunnen demonstreren.

Anton Doornenbal,
Oud AA 39A,
3621 LA Breukelen.
Tel. Prive : 03462-63237
Kantoor : 035-891036

SAVEV / LOADV

SAVEV / LOADV (c) Ch. POELS 25/12/83

The instructions LOADA and SAVEA of BASIC have a few limitations which can become very troublesome if one desires to handle arrays of important dimensions, for instance in the case of files on cassettes.

Indeed, when we save an array on cassette, BASIC reorganizes all the data of the array before recording them. This reorganization requires a memory area which lies immediately behind the SYMBOL TABLE. This is a major inconvenience if our BASIC program is very big and if too little memory is left over for this operation. In that case, an error message is displayed:"OUT OF MEMORY".

Similarly, when we load an array in memory, data are first saved in bulk behind the SYMBOL TABLE and then reorganized and transferred into the HEAP. This operation can be very lenghty, and the fact that digital cassettes are used does not, unfortunately, change anything to this state of affairs.....

In order to cope with this problem, I have written a routine which saves directly on cassette the whole of the HEAP and the SYMBOL TABLE. This routine has also its drawbacks, but it can be very useful in certain cases. Indeed, it is not necessary anymore to have a free memory area available for reorganization.Moreover, the waiting time after LOADA is cancelled. These 2 reasons seemed enough to me for writing this routine. In order to be able to use it, two limitations have to be kept in mind: when loading the variables, the CLEAR and the BASIC program have to be exactly the same as when the recording of variables was done!Moreover, after loading the variables, all their values are updated (because the whole of the variables has been loaded). It is therefore advised to perform the reading of the variables at the beginning of the progam, during the initializations.

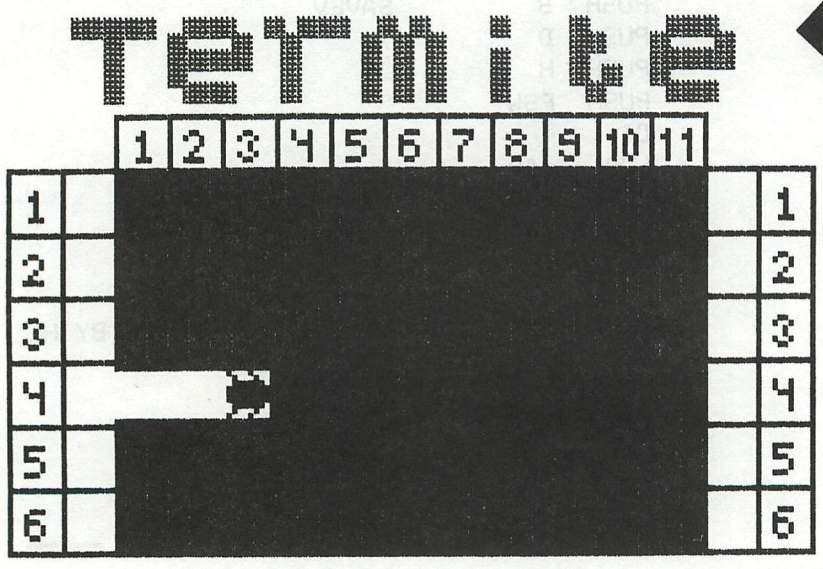
Use of the routine: saving of the variables occurs as follows: CALLM #34A,A$ (A$ being an alphanumeric variable containing the name of the file). The FILE TYPE of this file is "3".

To reload the variables: CALLM #300,A$.

```
>D300 390
0300 C5 D5 E5 F5 F3 3A 40 00 F6 C0 32 40 00 32 06 FD
0310 5E 23 56 EB 06 33 0E FF CD CE 02 2A 9B 02 11 00
0320 F9 CD D1 02 D2 AD D2 2A A1 02 11 00 F9 CD D1 02
0330 D2 A8 D2 22 A3 02 CD D4 02 3A 40 00 E6 3F 32 40
0340 00 32 06 FD FB F1 E1 D1 C1 C9 C5 D5 E5 F5 F3 3A
0350 40 00 F6 C0 32 40 00 32 06 FD 3E 33 5E 23 56 EB
0360 CD C5 02 2A 9D 02 EB 2A 9B 02 CD C8 02 2A A3 02
0370 EB 2A A1 02 7B 9D 5F 7A 94 57 CD C8 02 CD CB 02
0380 3A 40 00 E6 3F 32 40 00 32 06 FD FB F1 E1 D1 C1
0390 C9
```

```
]#L.
PAGE 01

001
002 0300 C5                    PUSH  B          LOADV
003 0301 D5                    PUSH  D
004 0302 E5                    PUSH  H
005 0303 F5                    PUSH  PSW
006 0304 F3                    DI
007 0305 3A4000                LDA   :40
008 0308 F6C0                  ORI   :C0
009 030A 324000                STA   :40
010 030D 3206FD                STA   :FD06
011 0310 5E          DLECT     MOV   E,M        FILENAME POINTED BY HL
012 0311 23                    INX   H
013 0312 56                    MOV   D,M
014 0313 EB                    XCHG
015 0314 0633                  MVI   B,'3'      FILE TYPE
016 0316 0EFF                  MVI   C,:FF      FILE NAME PRINTED
017 0318 CDCE02                CALL  :2CE       READ FILE NAME
018 031B 2A9B02                LHLD  :29B       READ HEAP
019 031E 1100F9                LXI   D,:F900
020 0321 CDD102                CALL  :2D1
021 0324 D2ADD2                JNC   :D2AD      RUN L.E. WITHOUT NEW
022 0327 2AA102                LHLD  :2A1       READ SYMBOL TABLE
023 032A 1100F9                LXI   D,:F900
024 032D CDD102                CALL  :2D1
025 0330 D2A8D2                JNC   :D2A8      RUN LOADING ERROR
026 0333 22A302                SHLD  :2A3       END OF S. TABLE UPDATED
027 0336 CDD402                CALL  :2D4       RCLOSE
028 0339 3A4000                LDA   :40
029 033C E63F                  ANI   :3F
030 033E 324000                STA   :40
031 0341 3206FD                STA   :FD06
032 0344 FB                    EI
033 0345 F1                    POP   PSW
034 0346 E1                    POP   H
035 0347 D1                    POP   D
036 0348 C1                    POP   B
037 0349 C9                    RET
038 034A C5                    PUSH  B          SAVEV
039 034B D5                    PUSH  D
040 034C E5                    PUSH  H
041 034D F5                    PUSH  PSW
042 034E F3                    DI
043 034F 3A4000                LDA   :40
044 0352 F6C0                  ORI   :C0
045 0354 324000                STA   :40
046 0357 3206FD                STA   :FD06
047 035A 3E33                  MVI   A,'3'      FILE TYPE
048 035C 5E                    MOV   E,M        FILE NAME POINTED BY HL
049 035D 23                    INX   H
050 035E 56                    MOV   D,M
051 035F EB                    XCHG
052 0360 CDC502                CALL  :2C5       SAVE FILE NAME
053 0363 2A9D02                LHLD  :29D       SAVE HEAP
```

```
PAGE 02

054 0366 EB          XCHG
055 0367 2A9B02      LHLD  :29B
056 036A CDC802      CALL  :2C8
057 036D 2AA302      LHLD  :2A3      SAVE SYMBOL TABLE
058 0370 EB          XCHG
059 0371 2AA102      LHLD  :2A1
060 0374 7B          MOV   A,E
061 0375 9D          SBB   L
062 0376 5F          MOV   E,A
063 0377 7A          MOV   A,D
064 0378 94          SUB   H
065 0379 57          MOV   D,A
066 037A CDC802      CALL  :2C8
067 037D CDCB02      CALL  :2CB      WCLOSE
068 0380 3A4000      LDA   :40
069 0383 E63F        ANI   :3F
070 0385 324000      STA   :40
071 0388 3206FD      STA   :FD06
072 038B FB          EI
073 038C F1          POP   PSW
074 038D E1          POP   H
075 038E D1          POP   D
076 038F C1          POP   B
077 0390 C9          RET
078 0391             END

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

DLECT  0310
```



Games collection 13

```
00000   SCREEN TABULATOR & EPSON 80/2 III, replaces print tab();

50010 REM E. ZAHNER, CH 8910 AFFOLTERN MARCH 25,1983
50020 REM PERMITS SCREEN TABLE WITH RS232C OFF
50030 REM PRINTER COMMANDS:
50040 REM CHR$(27);"D";CHR$(FIRST);CHR$(SECOND);..CHR$(0) TO CONCLUDE
50050 REM CHR$(9) TO MOVE PRINT HEAD TO NEXT TAB POSITION
50060 REM
50100 CLEAR 1000
50110 POKE #131,1:PRINT CHR$(12);
50120 V24=#FF05:BAUD0=#C0:BAUD1=#90:REM RS232C V24 SERIAL OUTPUT 9600 2400 BAUD
50130 WRITER=#131:SWON=0.0:SWOFF=1.0:REM PRINTER ON, OFF
50132 REM PRI= PREVIOUS STATUS WRITER
50140 REM VARIABLES
50150 REM TABUZ% NBR OF TABS.  TABULA%(..) TAB SETTINGS
50160 REM CX CY LAST CURSOR POSITION
50170 REM TABULA$ = TAB INSTRUCTION TO PRINTER (EPSON III)
50180 REM WO = NEW TAB POSITION.
50200 GOTO 50300
50210 REM SUBROUTINES
50220 PRI=PEEK(WRITER):POKE WRITER,SWOFF:CX=CURX:CY=CURY
50230 INPUT "<RETURN> ";RETU$:PRINT :POKE WRITER,PRI:RETURN
50240 REM WIPE SCREEN, NO FORM FEED
50250 PRI=PEEK(WRITER):POKE WRITER,SWOFF:INPUT "<RETURN WHEN OK > ";RETU$
50260 PRINT CHR$(12);:CX=CURX:CY=CURY:POKE WRITER,PRI:RETURN
50300 TABU$="":TABULA$="":REM TAB POSITION EXAMPLE
50310 DATA 5,15,25,35,45,55
50320 TABUZ%=6:DIM TABULA%(TABUZ%):FOR I%=1 TO TABUZ%
50330 READ TABULA%(I%):TABU$=TABU$+CHR$(TABULA%(I%))
50360 NEXT I%:TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
50370 PRINT TABULA$;
50400 POKE V24,BAUD1:POKE WRITER,SWON
50410 PRINT TABULA$
50420 PRINT "0";:FOR I%=1 TO TABUZ%:PRINT CHR$(9);"*";
50430 NEXT I%:PRINT
50440 FOR I%=1 TO TABUZ%:PRINT CHR$(9);I%;
50450 NEXT I%:PRINT
50460 FOR I%=-1 TO TABUZ%*(-1) STEP -1:PRINT CHR$(9);I%;
50470 NEXT I%:PRINT
50480 FOR I%=1 TO TABUZ%:PRINT CHR$(9);TABULA%(I%);
50490 NEXT I%:PRINT :POKE WRITER,SWOFF
50495 LIST 50495:REM NOTE: THE TAB POS. IS IN FRONT OF THE   FIRST CHARACTER ON
THE PRINTER
50500 LIST 50500:REM NOW A TABLE SHALL BE SHOWN ON THE SCREEN
50510 GOTO 50700
50520 REM SUBROUTINES FOR SCREENTABLE
50530 CX=CURX:CY=CURY:REM RETURN:REM STORES OLD CURSOR POSITION
50540 WO=TABULA%(I%):REM GETS NEXT TAB POSITION INTO WO
50550 IF WO<CX THEN WO=CX:REM RETURN:REM OLD<NEW
50560 PRINT CHR$(9);:PRI=PEEK(WRITER)
50570 POKE WRITER,SWOFF:PRINT SPC(WO-CX);:POKE WRITER,PRI
50580 RETURN:REM IF 50530/50550 RETURNS ARE USED, 50560 COMES TWICE ??
50700 POKE WRITER,SWON
50710 PRINT TABULA$
50720 TABUZ%=6
50730 FOR I%=1 TO TABUZ%
50732 GOSUB 50530
50734 REM GOSUB 50540
50736 REM GOSUB 50560:REM NOTE 50580
50740 PRINT CHR$(124);:NEXT I%:PRINT
50750 POKE WRITER,SWOFF
50800 LIST 50800:REM INPUT TAB POSITIONS
50805 PRINT "NBR","POS","AVOID 12 or 13"
50810 INPUT "NUMBER OF TAB-STOPS ";TABUZ%:PRINT
50820 DIM TABULA%(TABUZ%):TABU$="":TABULA$="":T=0.0
50830 FOR I%=1 TO TABUZ%:PRINT I%,
50840 INPUT TABULA%(I%):PRINT
50850 IF TABULA%(I%)<T THEN 50840
50860 T=TABULA%(I%)+1.0:IF T<3.0 THEN T=3.0
50870 TABU$=TABU$+CHR$(T-1):NEXT I%:PRINT
50880 TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
50900 REM
50910 GOSUB 50240:REM WIPE
50920 PRINT "NOW INPUT A STRING SUITABLE FOR THE COLUMN WIDTH"
50930 POKE WRITER,SWON:PRINT TABULA$:STRING$=""
50940 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT CHR$(124);:NEXT I%:PRINT
50950 FOR I%=1 TO TABUZ%:GOSUB 51530:PRINT STRING$;:NEXT I%:PRINT
50960 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT I%*(-1.0);:NEXT I%:PRINT
50970 FOR I%=1 TO TABUZ%:GOSUB 50530:PRINT I%;:NEXT I%:PRINT
50980 FOR I%=1 TO TABUZ%:GOSUB 50530:I=I%:PRINT I;:NEXT:PRINT
50990 REM GOTO 50940
51000 REM GOSUB 50240:REM WIPE
51010 POKE WRITER,SWOFF
51020 DATA 10,20,30,40
51030 TABUZ%=4:TABU$="":TABULA$="":DIM TABULA%(TABUZ%)
51040 PRINT "NEW TAB SETTING";:FOR I%=1 TO TABUZ%:READ TABULA%(I%):PRINT TABULA%
(I%);:NEXT:PRINT
51050 FOR I%=1 TO TABUZ%:TABU$=TABU$+CHR$(TABULA%(I%))
51070 TABULA$=CHR$(27)+"D"+TABU$+CHR$(0)
51080 PRINT "NOW INPUT A INTEGER NUMBER 1 TO 5 DIGITS"
51085 POKE WRITER,SWON:PRINT TABULA$
51090 FOR J%=0 TO 2:FOR I%=1 TO TABUZ%:GOSUB 51900:REM INPUT
51100 PRINT CHR$(124);SPC(LEER);NBR%;:NEXT I%:PRINT CHR$(124.0)
51110 NEXT J%
51120 POKE WRITER,SWOFF:GOSUB 50240:POKE V24,BAUD0:END
51500 REM SUBROUTINE STRING INPUT
51530 CX=CURX:CY=CURY:PRI=PEEK(WRITER):POKE WRITER,SWOFF
51535 CURSOR 0,1:PRINT SPC(58);:CURSOR 25,1
51540 INPUT "STRING ";STRING$
51550 WO=TABULA%(I%):IF WO<CX THEN WO=CX
51560 CURSOR CX,CY:POKE WRITER,PRI
51570 PRINT CHR$(9);:POKE WRITER,SWOFF
51580 PRINT SPC(WO-CX);:POKE WRITER,PRI
51590 RETURN
51700 REM FORMATTING
51710 NBR%=INT(NBR)
51756 LE=7.0
51760 IF NBR%<1E5 AND NBR%>(-1E5) THEN LE=6.0
51770 IF NBR%<10000.0 AND NBR%>(-10000.0) THEN LE=5.0
51780 IF NBR%<1000.0 AND NBR%>(-1000.0) THEN LE=4.0
51790 IF NBR%<100.0 AND NBR%>(-100.0) THEN LE=3.0
51800 IF NBR%<10.0 AND NBR%>(-10.0) THEN LE=2.0
51820 RETURN
51900 CX=CURX:CY=CURY:PRI=PEEK(WRITER):POKE WRITER,SWOFF
51910 CURSOR 0,1:PRINT SPC(58);:CURSOR 20,1
51920 INPUT "INTEGER ";NBR:GOSUB 51700
51925 IF I%=TABUZ% THEN COLWID=12.0:GOTO 51940:REM LAST COLUMN =12
51930 COLWID=TABULA%(I%+1.0)-TABULA%(I%):REM COLUMNWIDTH
51940 LEER=COLWID-LE-2.0:REM 1+CHR$(24) & 1 SPACE
51970 POKE WRITER,PRI:CURSOR CX,CY
51980 PRINT CHR$(9);
51990 RETURN
51995 REM END OF PROGRAM
END PROGRAM
```

SPL
UN ASSEMBLEUR POUR LE DAI PC

LE logiciel SPL, comme disent les initiés, est un des cinq éditeurs-assembleurs de langage-machine qui « tournent » sur le DAI PC. Sans doute est-il le plus puissant, car il se veut très proche de MACRO 80, un assembleur tout à fait professionnel.

SPL est livré sous forme de cassette audio (ou micro-cassette numérique) avec plusieurs utilitaires, dont DISPLAY, un très puissant désassembleur et TRANSLATOR, qui permet de récupérer des sources écrites avec les autres assembleurs. Voilà qui est bien pratique. La notice, en français, est longue de 34 pages ; c'est dire que les commandes sont nombreuses et qu'il convient de lire et relire le texte avant de prétendre maîtriser ce logiciel. Cela étant, *SPL* offre une grande souplesse d'emploi.

Le droit à l'erreur

D'abord, sa façon de compacter le fichier-source autorise la compilation de programmes très longs, avantage décisif pour toute application « sérieuse » : on peut assembler en une seule fois jusqu'à 12 Koctets de codes-machine ! Ensuite, l'écriture de la source sous éditeur plein écran (avec tabulation automatique pour chacune des zones labels, opérandes, etc.) simplifie grandement le travail : il est possible de déplacer le texte-source dans toutes les directions, de se positionner n'importe où pour corriger.

Le premier déverminage (ou débogage) se fait en sortie d'édition : en cas d'erreur de syntaxe, *SPL* réédite la source à partir de la ligne fautive, et le curseur est remplacé par une lettre clignotante mnémonique du type d'erreur détecté. Ce système fait gagner beaucoup de temps lors de l'écriture d'un programme. Les possibilités de travail sont très grandes, et il est hors de question, en quelques lignes, de les passer toutes en revue. Les grands « classiques » sont là, avec souvent un « plus » : copie ou déplacement de paragraphes, recherche ou remplacement d'étiquettes, paramétrage de la liste (nombre de lignes par page, affi-

chage en décimal, hexadécimal, octal, binaire, au choix, des adresses et/ou des opérandes). J'ai beaucoup apprécié la possibilité de demander un segment de liste ou d'édition par numéro de ligne ou par étiquette. Ainsi, la commande L 10 FIN déclenche-t-elle la liste de la source de la ligne 10 jusqu'à l'endroit où se trouve l'étiquette FIN. L'inconvénient d'une telle richesse est l'abondance des commandes : il vaut mieux garder le manuel près de soi pour s'y retrouver, du moins au début !

La liste 1 montre un exemple de source *SPL*, utilisant l'assemblage conditionnel, la gestion dynamique d'étiquettes, les macro-instructions, avec ou sans passage de paramètres. Ce programme est destiné à mesurer le temps que le Dai met à parcourir 255 instructions NOP (1). Les lignes 21 à 25 montrent comment expanser 255 instructions en peu de place : c'est l'assembleur qui fait la boucle en gérant dynamiquement l'étiquette COMPT.

Le début de liste (lignes 9 à 14) est un exemple d'assemblage conditionnel, utile pour faire plusieurs versions d'une même source. Ici, l'étiquette VERS (version) vaut 1, le début du code-machine sera compilé à partir de 400 (hex), sinon, à partir de 500 (hex). Ceci peut être encore affiné, car *SPL* permet les tests booléens sur les étiquettes, du genre IF VERS =1 AND LABEL=2 OR VERS < > 5, etc. Les macro-instructions peuvent aussi passer des paramètres (exemple : store passe l'adresse RESULT dans l'étiquette RE lors de l'expansion de la macro). La ligne 2 montre comment donner des directives à l'imprimante, lors du listage de la source. Ici, on commande le passage en écriture grasse sur une Epson.

L'Assembleur n'est pas tout

L'utilitaire DISPLAY, livré avec *SPL*, ajoute des fonctions à l'assembleur : il permet d'obtenir la liste hex et ASCII d'une zone mémoire. Il désassemble aussi un code-machine en recréant une source utilisable par *SPL*, avec des étiquettes, s'il vous plaît ! Sur la liste 2, on voit comment apparaît le désassemblage de la routine « sortie série » du Dai. Remarquez les étiquettes placées aux points de branchement de la routine. DISPLAY peut désassembler *SPL* lui-même (12 Ko de codes-machines !) et le tout (code et source) tient encore en mémoire vive. Enfin,

*(1) Si vous désirez savoir combien de temps le Dai met à parcourir 255 NOP, faites, sous Basic : PRINT (#FFFF - PEEK (#300) - (PEEK (#301) *256)) *20 ; "millisecondes".*

En fait, ce n'est qu'à partir de plusieurs milliers de NOP que la décrémentation devient visible.

Liste 1 : programme-source obtenu avec SPL

```
1     ;
2               PRT       1BH,45H          ;directives imprimante
3     ;
4               TITL       ESSAI SPL
5     ;
6               PUT       "H"              ;affichage hexadecimal
7               ORG       300H
8     TEMPS     DW        0FFFFH           ;initialisation
9     VERS      SET       1H               ;version 1
10              IF        VERS=1H
11              ORG       400H
12              ELSE
13              ORG       500H
14              ENDIF
15    ;
16    TIMER     EQU       1BEH             ;horloge du DAI
17    RESULT    EQU       300H             ;tampon du resultat
18    ;
19              push                       ;appel de macro
20              time
21    COMPT     SET       0FFH             ;label gere dynamiquement
22    loop      ###                        ;pt d'entree de l'expansion
23              NOP
24    COMPT     SET       COMPT-1H         ;decrement du label
25              loop      COMPT>0H         ;boucle tant que label>0
26    ;
27              store     RESULT           ;macro avec parametre
28              pop
29              RET
30    FIN       END
31    ;
32    ;zone des macro-instructions
33    ;
34    push      MACRO
35              PUSH B
36              MEND
37    time      MACRO
38              LHLD      TEMPS
39              SHLD      TIMER
40              MEND
41    store     MACRO     RE               ;passage du parametre
42              LHLD      TIMER
43              SHLD      RE
44              MEND
45    pop       MACRO
46              POP B
47              MEND
48    ;
```

Liste 2 : exemple de désassemblage d'une routine

```
0000              ;
0000              ;
0000                                PRT       1BH,45H
0000                          ;
0000                          ORG       0DD94H
DD94   F5       LHDD94       PUSH PSW
DD95   3A00FD   LHDD95       LDA       LHFD00
DD98   E608                  ANI       8H
DD9A   CA95DD                JZ        LHDD95
DD9D   3AF3FF   LHDD9D       LDA       LHFFF3
DDA0   E610                  ANI       10H
DDA2   CA9DDD                JZ        LHDD9D
DDA5   F1                    POP PSW
DDA6   32F6FF                STA       LHFFF6
DDA9   FE0D                  CPI       0DH
DDAB   C0                    RNZ
DDAC   F5                    PUSH PSW
DDAD   3E0A                  MVI A     0AH
DDAF   CD94DD                CALL      LHDD94
DDB2   F1                    POP PSW
DDB3   C9                    RET
DDB4   à=FD00   LHFD00       EQU       0FD00H
DDB4   à=FFF3   LHFFF3       EQU       0FFF3H
DDB4   à=FFF6   LHFFF6       EQU       0FFF6H
DDB4                         END
```

LES COUPS D'ŒIL DE LIST

SPL, UN ASSEMBLEUR POUR DAI

l'utilitaire IMPLEM permet de paramétrer *SPL*, si les options par défaut ne vous conviennent pas, et ce, sans POKEs fastidieux.

La fonction LINK (« accrochage », lors de la compilation, de sous-routines en bibliothèque) est absente de *SPL*. Elle est remplacée par un MERGE (commande Y) qui inclut dans la source de travail des segments provenant de la mémoire de masse.

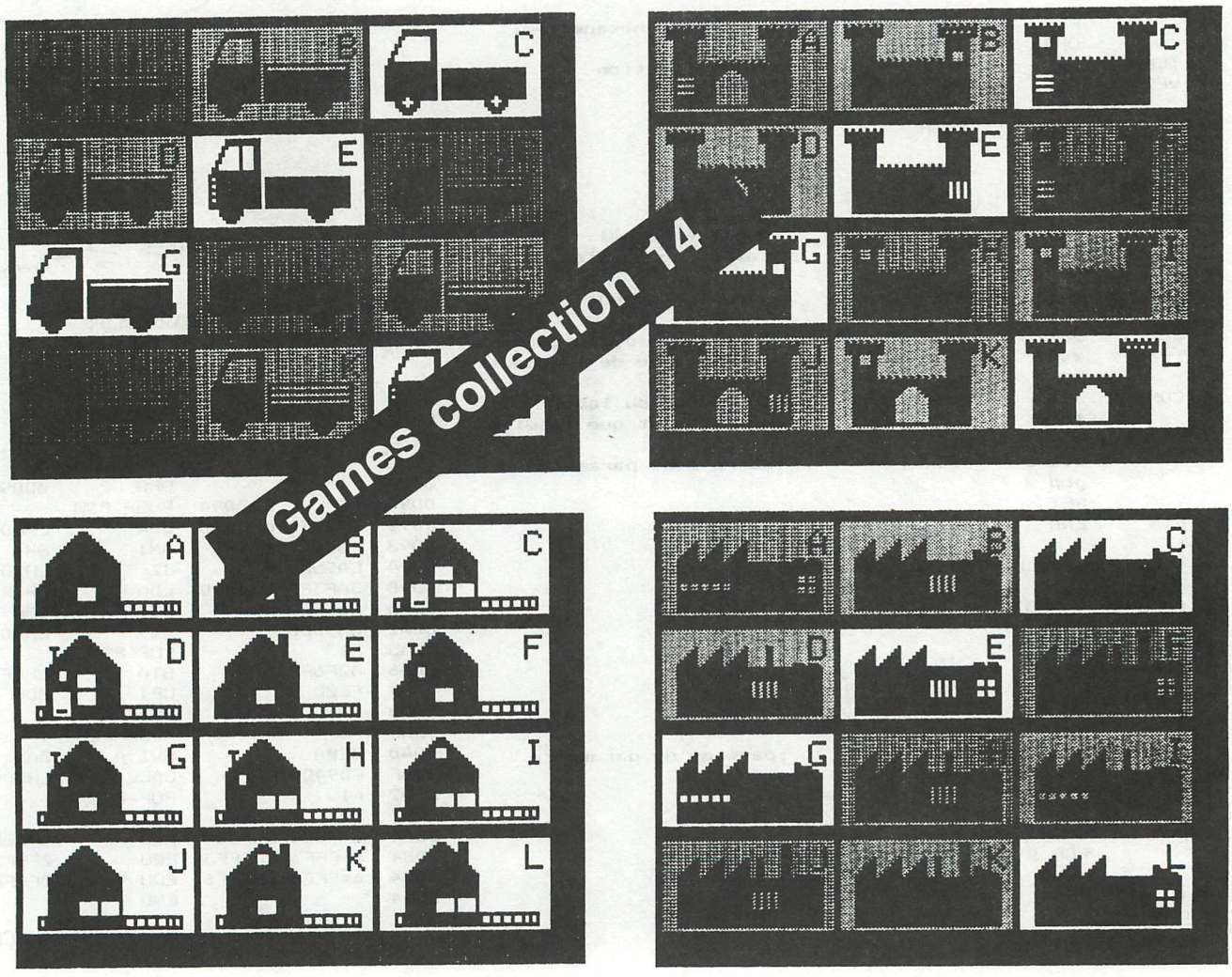
L'interactivité de *SPL* se traduit en (nombreux !) messages signalant telle ou telle anomalie : fautes de syntaxe, fautes de structure (lors de la compilation), fautes d'introduction (commande erronée), etc. De plus, à chaque retour à *SPL*, un *check sum* de la source, du code et de *SPL* lui-même vérifie l'intégrité de ce qui est en mémoire (vous savez : un programme « pas tout à fait au point » et que l'on essaie, peut occasionner des ravages sournois !).

Voilà donc un assembleur très réussi et puissant qui ne dépayserait pas un utilisateur venant de machines n'ayant rien à voir avec un micro-ordinateur. Associé à *DDT (DAInamic debugging tool,* un excellent utilitaire de mise au point), *SPL* se devrait de figurer dans la panoplie de tout Daïste féru de langage-machine.

Alain MARIATTE

Le logiciel en quelques lignes

Nom : Assembleur 8080
Ordinateur : Dai PC et Dai T (version professionnelle)
Forme : cassette audio ou cassette numérique
Edité et distribué par : Dainamic Mottaart - 20 3170 Herselt Belgique
ou : Dainamic France - 9, rue Lavoisier, 59140 Dunkerque
Prix public : 1 100 F belges (165 F français, environ)
Orientation principale : assemblage de langage-machine
Autres orientations : désassembleur, translateur de sources, dump-mémoire



DIM STATEMENT

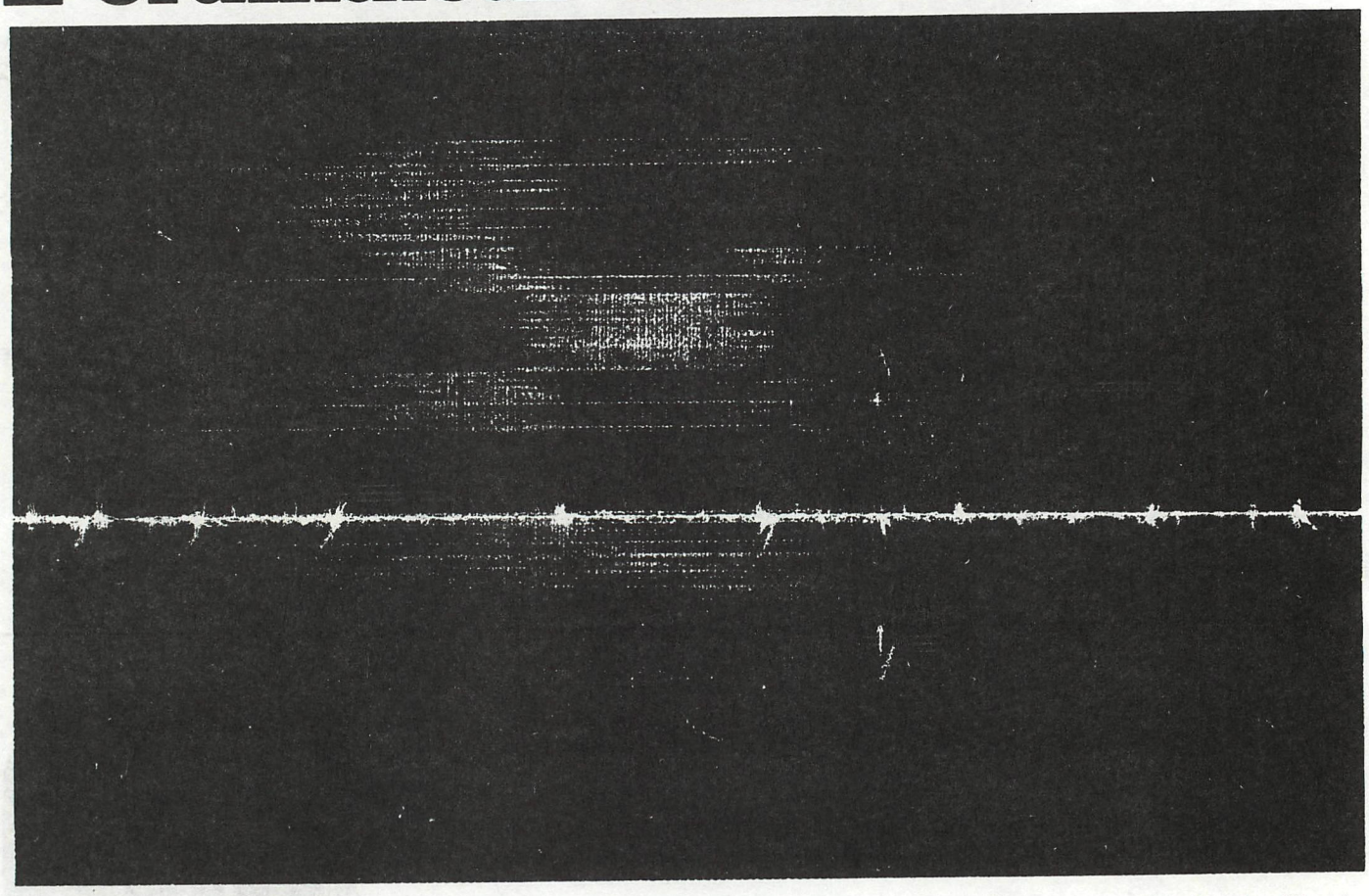
In the DAI-manual (Basic version V1.0), it is stated that the maximum dimension which can be declared is 254. But if in a program a dimension > 254 is used, no error report occurs. A DIM N(2,255) will be executed, appearently without any problems.

But is seems only without problems. As soon as the array is filled with data, the problem will show up. A value declared to N(1,0) or to N(2,0) will be found in N(0,0) too !! What happened ?

If we examine the pointer to N(1,0) via HEX$(VARPTR(N(1,0))), we will find the same VARPTR as for N(0,0) ! The DAI makes a big mess with the data stored in this array. Effectively, it has declared an array N(0,254), and the high order subscripts point to the same memory area.

The Basic version V1.1 enables the dimensioning of an array with the maximum subscript 255. So programs developped on a V1.1 machine may cause problems on a V1.0 machine !

TISSUS ARMURÉS
L'ordinateur tisserand



Tissé à partir d'une armure créée à l'aide d'un des programmes informatiques, un échantillon où apparaît le jeu des fils pris et laissés.

Le tissu imprimé, plus facile et moins cher à réaliser, a supplanté de nos jours le tissu armuré qui tire son relief du mode d'entrecroisement des fils. Celui-ci revient à la mode dans les haut de gamme des textiles d'ameublement.

La manière d'entrelacer fils de chaîne et fils de trame procède d'une certaine répétition. C'est ce qui a amené *Rémi Prin,* créateur textile et ingénieur informaticien, à créer des programmes de constitution d'armures. A partir des résultats obtenus, *Monique Prin* et *Rémi* tissent des échantillons que des industriels textiles vont ensuite utiliser pour leur fabrication.

Deux sortes de programmes ont ainsi été mis au point : l'un part d'un motif de base et lui fait subir toute sortes de déformations, c'est donc un programme *analytique* de traitement de texture. L'autre joue avec les paramètres constituant l'armure. Pour comprendre les manipulations réalisées, il faut connaître les opérations essentielles de tissage (voir encadré).

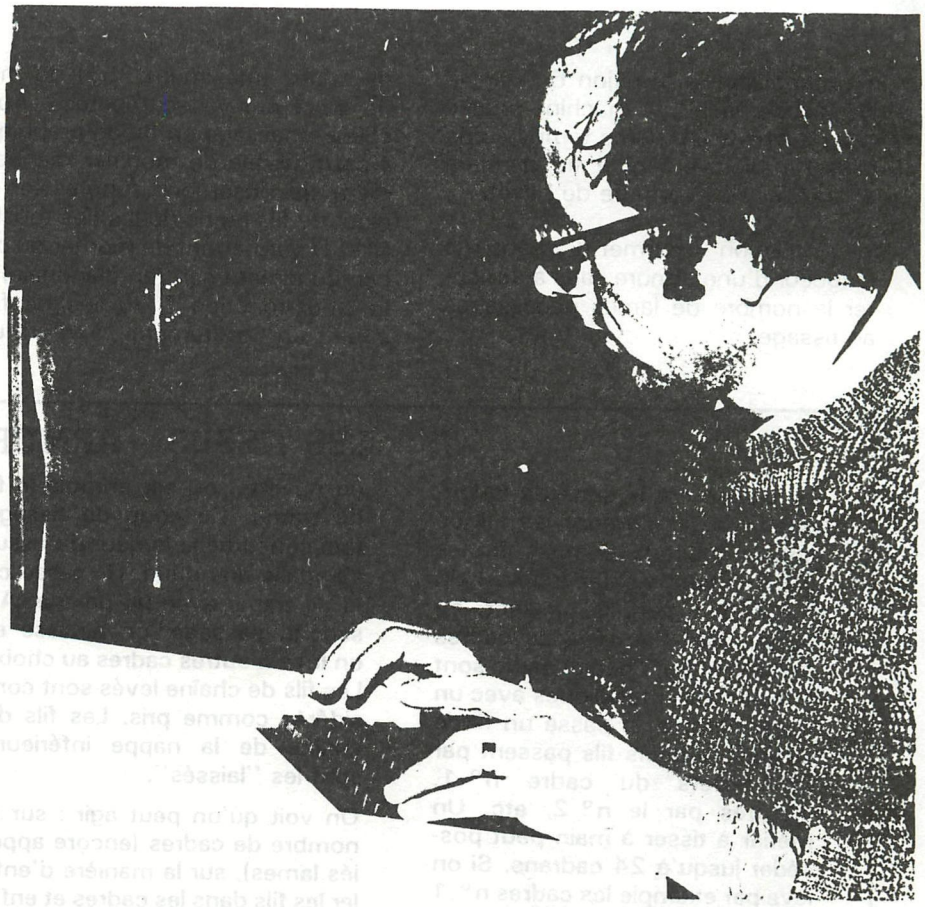
Rémi Prin travaille sur un micro-ordinateur type *DAI* de 48 K caractères de mémoire pour l'utilisateur. Les programmes sont écrits en langage de programmation appelé BASIC VI, sur cassette. L'unité centrale est reliée à un écran TV couleur équipé d'un câble de péritélévision. La partie édition de chaque programme est écrite pour adaptation à une imprimante graphique (de type *AXIOM* IMP 2-Q).

Appelé "Textile 3A", ce programme part donc du motif de base et comporte huit fonctions :

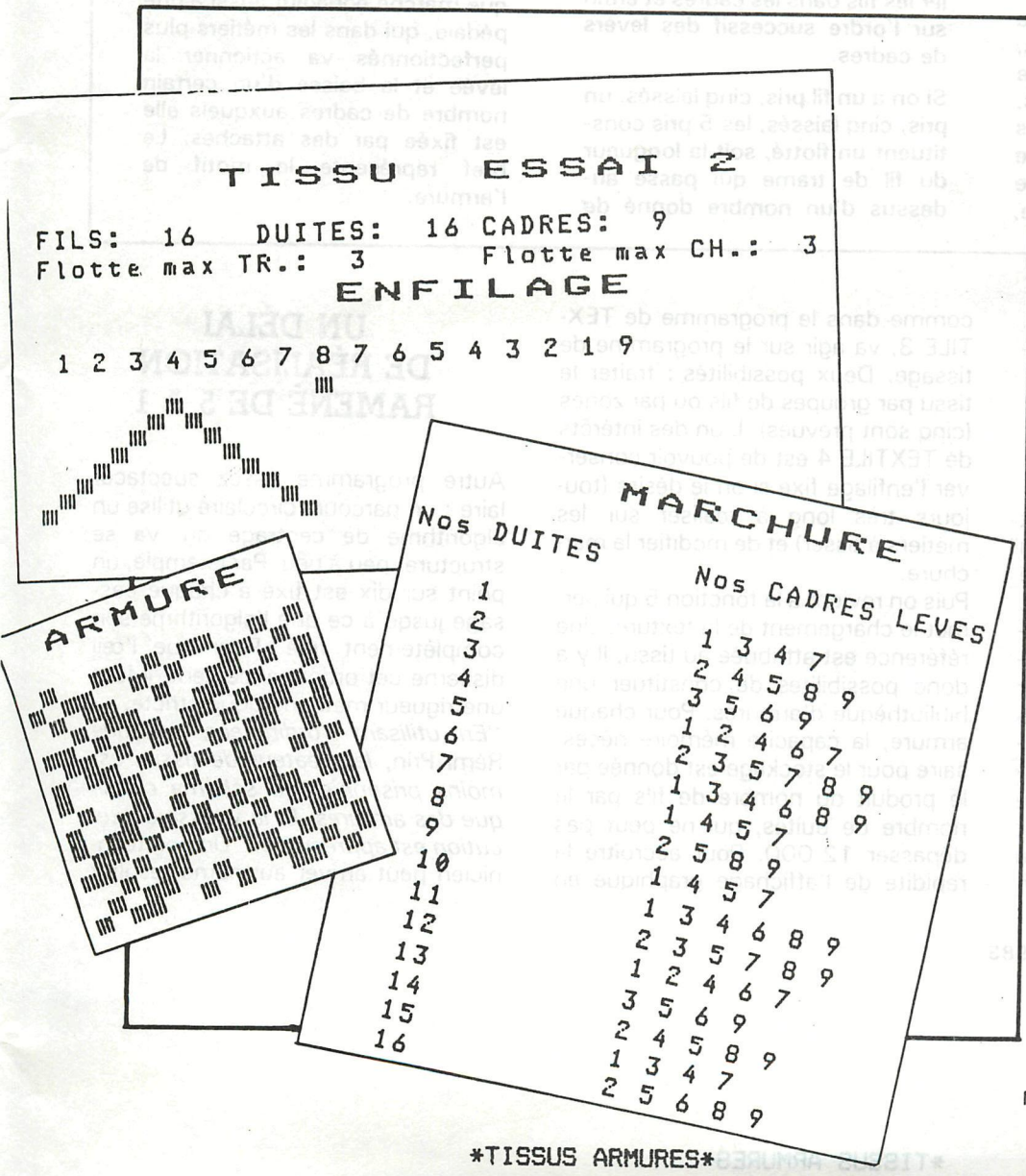
- Après appel de la fonction 1 "Entrée armure de base", l'opérateur définit les dimensions de l'armure (nombre de fils (1) et de duites) qui est saisie graphiquement.
- La fonction 2 permet de modifier

l'armure précédente par retrait ou ajout soit de fils, soit de duites, toujours en saisie graphique, les fils de chaîne apparaissant en bleu foncé sur le rectangle bleu ciel aux dimensions de l'armure. Les laissés restent bleu clair.

• A l'appel de la fonction 3 d'analyse/édition, l'ordinateur effectue l'analyse de l'armure et édite à l'écran, ou sur imprimante au choix, les données suivantes : nombre de fils, nombre de duites, nombres de cadres nécessaires au tissage, longueur des flottés maximum en chaîne et en trame, l'enfilage, la marchure et enfin l'armure.



Rémi Prin élabore une nouvelle armure au clavier de son Daï.



Edition à l'imprimante des différentes caractéristiques d'une armure créée à l'écran.

• La fonction 4 donne l'aspect du tissu. Elle offre trois possibilités de résolution graphique, G pour 72 × 65 points, M pour 160 × 130 points et F pour 336 × 256 points. Second choix : les couleurs (4 parmi 16 disponibles). Troisième choix : la hauteur du dessin écran. Quelques répétitions de motifs étant suffisantes au début de l'élaboration d'un tissu, on peut l'afficher sur plein, moitié, quart d'écran..., etc. On peut aussi afficher une armure équilibrée (50/50) ou à chaîne élargie (ch. 100/tr. 50) ou trame élargie (ch. 50/tr. 100).

• La fonction 5 de "traitement de l'armure" offre des possibilités intéressantes de traitement du motif ou d'agrandissement du motif de base. Quatre zones de travail sont définies avec, pour chacune, onze traitements possibles dont des rotations de 90, 180 ou 270°, des symétries verticales ou horizontales, des décalages dans les quatre directions ou une inversion des pris et des laissés.
On peut revenir au début de cette fonction cinq, plusieurs fois de suite, ce qui multiplie la capacité de traiter des armures plus complexes.

• En appelant la fonction 6 "Rappel armure de base", la machine stocke deux armures en même temps sur la cassette digitale rapide (la dernière effectuée, plus l'armure de base).

• La fonction 7 permet d'agrandir la longueur d'une armure sans augmenter le nombre de lames nécessaires au tissage.

de duites (maximum 255) du motif de base qui va se répéter ? Après chaque passage en mode graphique, il est possible de modifier manuellement par insertion, suppression ou ajout de fils ou de duites (les touches M et D permettent de monter ou descendre directement en diagonale).
La sous-fonction 5 au lieu de faire subir un traitement à l'armure,

haute résolution, certaines routines ont été réalisées en langage "bas niveau" dit assembleur.
Rémi Prin devrait élaborer une version encore plus perfectionnée qui fusionnera ces deux sortes de programmes, mais elle nécessitera de posséder un double lecteur de disquettes pour augmenter les capacités mémoires insuffisantes sur cassette.

LES TISSUS ARMURÉS

La chaîne est le sens de la longueur du fil. Ce sont les fils de chaîne qui sont tendus entre les deux extrémités (ensouples) du métier à tisser. Ils passent, au milieu, par une série de cadres parallèles. Sur chaque cadre sont fixés des fils métalliques avec un œillet central où passe un fil de chaîne. Certains fils passent par les œillets du cadre n° 1, d'autres par le n° 2, etc. Un métier à tisser à main peut posséder jusqu'à 24 cadrans. Si on lève par exemple les cadres n° 1 et 5, la nappe de fils de chaîne se sépare en deux sous-nappes. Celle du dessus est constituée de tous les fils de chaîne soulevés, celle du dessous de tous les autres fils. La 1re opération de tissage consiste à passer entre ces deux nappes, une cannette, ou navette, où est enroulé le fil de trame. Ce coup de tissage (qui constitue la largeur du tissu) s'appelle une duite. On tasse ce fil de trame avec un peigne. Au second passage, on abaisse et on lève d'autres cadres au choix. Les fils de chaîne levés sont considérés comme pris. Les fils de chaîne de la nappe inférieure sont les "laissés".

On voit qu'on peut agir : sur le nombre de cadres (encore appelés lames), sur la manière d'enfiler les fils dans les cadres et enfin sur l'ordre successif des leviers de cadres.

Si on a un fil pris, cinq laissés, un pris, cinq laissés, les 5 pris constituent un flotté, soit la longueur du fil de trame qui passe au-dessus d'un nombre donné de fils de chaîne (ici 5). (Certains flottés peuvent se retrouver sur l'envers ou être des flottés de chaîne.)

Suivant la répartition des pris et des laissés et compte tenu d'un enfilage donné, on va obtenir une armure bien définie, l'armure étant la représentation de l'entrelacement des fils. Cette suite de combinaisons (où si l'on veut la marche à suivre) constitue la marchure, et chacune de ces combinaisons, une marche. Chaque marche équivaut aussi à une pédale, qui dans les métiers plus perfectionnés va actionner la levée et la baisse d'un certain nombre de cadres auxquels elle est fixée par des attaches. Le bref représente le motif de l'armure.

• La fonction 8 est d'anayse-édition. Elle sort sur imprimante l'armure réalisée (voir schéma).

CRÉATION SYNTHÉTIQUE

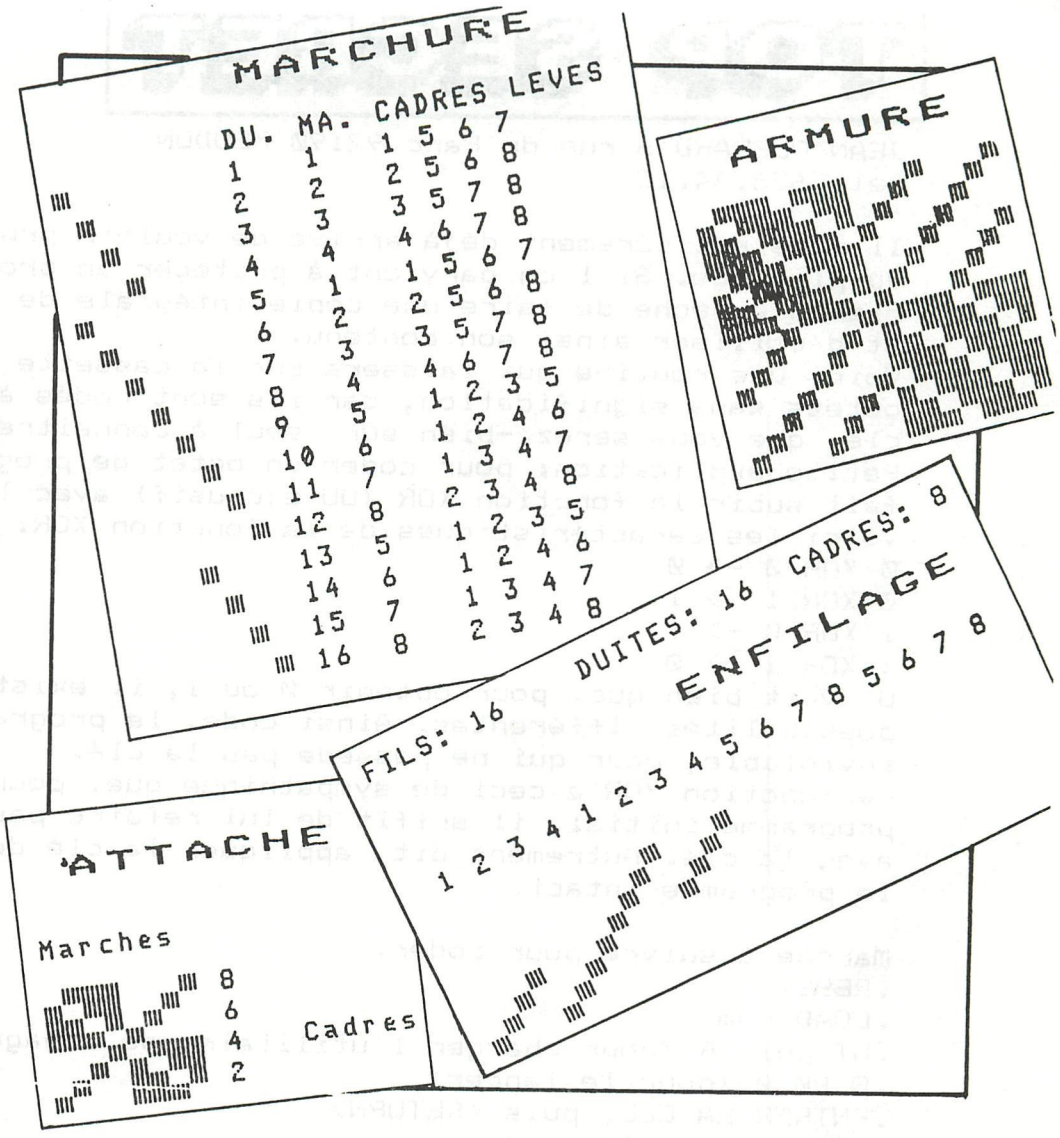
Le programme "TEXTILE 4", lui, est orienté vers la création synthétique, à partir des données de l'enfilage et de la marchure. Il possède six fonctions. La fonction 1 de "création/modification texture qui comporte sept sous-menus, pose d'abord à l'utilisateur des questions de définition de l'armure. Combien de cadres ? (maximum 24). Combien de fils ? (maximum 255). On obtient dans un premier temps l'affichage de l'enfilage et le pavé rectangulaire correspondant à la dimension choisie. Puis, combien

comme dans le programme de TEXTILE 3, va agir sur le programme de tissage. Deux possibilités : traiter le tissu par groupes de fils ou par zones (cinq sont prévues). L'un des intérêts de TEXTILE 4 est de pouvoir conserver l'enfilage fixe si on le désire (toujours très long à réaliser sur les métiers à tisser) et de modifier la marchure.
Puis on revient à la fonction 5 qui permet le chargement de la texture. Une référence est attribuée au tissu. Il y a donc possibilités de constituer une bibliothèque d'armures. Pour chaque armure, la capacité mémoire nécessaire pour le stockage est donnée par le produit du nombre de fils par le nombre de duites, qui ne peut pas dépasser 12 000. Pour accroître la rapidité de l'affichage graphique en

UN DÉLAI DE RÉALISATION RAMENÉ DE 5 A 1

Autre programme assez spectaculaire : un parcours circulaire utilise un algorithme de centrage qui va se structurer peu à peu. Par exemple, un point sur dix est fixé à chaque passage jusqu'à ce que l'algorithme soit complètement figé. Pour que l'œil discerne cet ordonnancement, il faut une rigueur mathématique stricte.
"En utilisant l'ordinateur, explique Rémi Prin, *le créateur de tissus est moins prisonnier du schéma classique des armures. Et la vitesse d'exécution est appréciable."* Un bon technicien peut arriver au même résultat

MICRO PROGRAMMES



mais ses crayonnés lui prennent un temps fou. C'est ainsi qu'un autre programme "écossais-rayures" réduit le délai de réalisation dans la proportion de 5 à 1. *"Cependant,* reconnaît-il, *l'informatique n'est pas tout : on réalise aussi un travail plastique indépendant de l'utilisation du micro-ordinateur. Il y a toute une recherche de matière due à la densité des fils qui décale l'aspect visuel du tissu."*

DU TEXTILE A LA CRÉATION ARTISTIQUE

Le grand danger, selon lui, serait qu'industriels et créateurs textiles récupèrent l'outil informatique dans l'état actuel sans essayer d'en développer les potentialités. Avec des moyens appropriés, il serait possible d'effectuer des recherches approfondies sur le langage textile, en remettant en cause les classifications traditionnelles des armures très rigoureuses. Lorsqu'il s'agissait, autrefois, de tisser des étoffes d'armures compliquées, il fallait bien connaître les modes d'enfilage et de flottement précis pour que le tissu tienne bien. De trop grands flottés, ou des irrégularités importantes dans la répartition des points d'aiguillage (l'endroit où les fils se croisent) peuvent faire que le tissu poche ou soit peu solide. Et cela, l'ordinateur ne le dit pas. Tout au moins pas encore, car il serait possible de réaliser des programmes sur la densité à respecter. Sur les extensions envisageables de l'utilisation du micro, Rémi Prin est intarissable : *"L'idéal serait d'effectuer des recherches approfondies au niveau de la combinatoire, de l'armure et du langage textile. Remettre en cause les classifications, ce qui implique une recherche informatique sur les réseaux des pris et des laissés. Un livre édité en 1938 à Lyon écrit par Brandon et Guillet traite de* "La méthode des initiales". *C'est l'approche du tissage à lames à l'aide d'un formalisme mathématique. Il permet de relier une forme continue d'une courbe que l'on désire obtenir dans le tissu et le caractère discontinu du tissage ; des contours analytiques à des réseaux de tissage. Peu de gens maîtrisent encore bien cette méthode. Il y aurait là des travaux informatiques approfondis à envisager."*

Rémi Prin rêve de recherches pluridisciplinaires dont le textile serait le nœud. Il a animé, en mars 1982, un stage de *"Micro-informatique en création textile"* au Centre Art et Industrie de Tourcoing. Un second est prévu à Chambéry en avril 1983 auprès d'un groupe de créateurs textiles de Savoie et d'Isère. Une semaine, sur les deux que dure le stage, sera réservée à un travail plastique autour de la technique de l'ikate (2) pour laquelle Rémi Prin a également créé un autre type de programme.

Préoccupations identiques pour une démonstration réalisée par Philippe Dujardin et Rémi Prin lors des trois journées "Micro-informatique et création textile", qui ont eu lieu aux Filteries DMC les 12, 13 et 14 janvier dernier. Organisées par la filothèque DMC et l'association Textile / Art / Langage, ces rencontres de créateurs et d'industriels ont permis d'envisager les développements possibles de ces recherches.

Dans l'industrie, il existe bien quelques systèmes informatiques, mais trop lourds et donc incomplètement exploités, ou qui sont utilisés pour déchiffrer des motifs existants et les reproduire. En somme un travail de compilation et non de création. C'est pourquoi, en France, n'existe-t-il quasiment pas de programme exploitable par les tisserands-artisans eux-mêmes. Rémi Prin a voulu combler ce manque avec ses différents programmes qu'il continue d'améliorer et qui coûtent 1 200 F chacun.

Quant à l'enseignement de l'utilisation du micro, il est dommage que seuls des rares stages ponctuels soient mis sur pied à l'heure actuelle. L'outil informatique au service du textile pourrait d'abord être utilisé avec profit dans les écoles d'arts appliqués en section textile.

Micheline DOMANCICH

---

(1) Fils : il s'agit des fils de chaîne quand ce n'est pas précisé.

(2) Ikate : les fils sont teints avant tissage. Sur chaque fil, il existe une fragmentation des couleurs par réserve. Le programme permet d'élaborer des contours des zones à colorier à partir de ces contours et de les mélanger.

TOP SECRET

JEAN GUERARD 6,rue du Parc 92190 MEUDON
tél: 626.34.18

Il vous est sûrement déjà arrivé de vouloir protéger un programme du piratage. Si l'on parvient à protéger un programme du BREAK, rien n'empêche de faire une copie intégrale de la cassette et d'utiliser ainsi son contenu.
Voici une routine qui laissera sur la cassette une suiite d'octets sans signification, car ils sont codés à l'aide d'une clé, que vous serez -bien sûr- seul à connaitre.
Petite explication: pour coder un octet de programme, on lui fait subir la fonction XOR (OU exclusif) avec la clé.
Voici les caractéristiques de la fonction XOR:
0 XOR 0 -> 0
0 XOR 1 -> 1
1 XOR 0 -> 1
1 XOR 1 -> 0
On voit bien que, pour obtenir 0 ou 1, il existe *deux* possibilités différentes. Ainsi codé, le programme est donc inviolable, pour qui ne possède pas la clé.
La fonction XOR a ceci de sympathique que, pour retrouver le programme initial, il suffit de lui refaire passer l'encodage avec la clé. Autrement dit, appliquer la clé deux fois laisse le programme intact.

Marche à suivre pour coder:
.RESET
.LOAD pgm
.UT puis R (pour charger l'utilitaire de codage)
.G B000 (pour le lancer)
.ENTRER LA CLE, puis <RETURN>
.SAVE pgm basic codé

C'est tout ! Essayez donc de lister ensuite, pour voir !

Pour décoder un programme crypté, la marche à suivre est exactement la même. Il est -bien sûr- inutile de refaire le SAVE.

Pour vous rassurer quant à la sécurité d'encodage, sachez que l'utilitaire efface toute trace de la clé après codage ou décodage. De plus, avec 100 caractères au clavier, il existe 100^n clés possibles de n lettres !
Il faudrait une patience séculaire au pirate qui tenterait d'essayer les 10^32 clés de 1 à 16 lettres.

SUITE

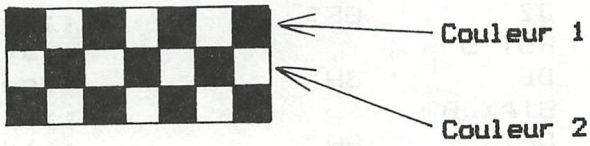
```
B04E            ;*
B04E            ;*------ messages -----
B04E            ;*
B04E  5C        MESSG   DB      5CH             ;;nbr caracteres
B04F  0C0D              DB      0CH,0DH
B051  202050            DB      ' Pour coder ou decoder '
B069  6C6520            DB      'le programme en memoire,'
B081  0D                DB      0DH
B082  202065            DB      '  entrez la cle'
B091  206465            DB      ' de decryptage:'
B0A0  0D                DB      0DH
B0A1  202020            DB      '       >>> '
B0AB            END     END
```

```
0000
0000            ;****************************************
0000            ;*          T O P - S E C R E T         *
0000            ;*     Pour proteger vos programmes     *
0000            ;*    (c) 1984 Jean GUERARD pour l'OI   *
0000            ;****************************************
0000            ;*
0000                    ORG       0B000H        ;;implantation
B000            ;*
B000            ;*------- presentation -------
B000            ;*
B000  214EB0            LXI H     MESSG         ;;adr message
B003  CD32DB            CALL      0DB32H        ;;aff. message
B006            ;*
B006            ;*----- entree de la cle -----
B006            ;*
B006  à=F800 CLE        EQU       0F800H        ;;stockage cle
B006  0100F8            LXI B     CLE           ;;debut de la cle
B009  CDBBD6 GET        CALL      0D6BBH        ;;attente d'un
B00C  CA09B0            JZ        GET           ;;caractere.
B00F  EF                RST 5                   ;;affichage de
B010  03                DB        3H            ;;ce caractere.
B011  02                STAX B                  ;;stockage
B012  FE08              CPI       8H            ;;)decr. pointeur
B014  C21BB0            JNZ       TEST          ;;)si DELCHAR
B017  0B                DCX B                   ;;)est pressee.
B018  C309B0            JMP       GET
B01B  03      TEST      INX B
B01C  FE0D              CPI       0DH           ;;fin de la cle
B01E  C209B0            JNZ       GET
B021            ;*
B021            ;*--- preparation des adresses ---
B021            ;*
B021  2AA302            LHLD      2A3H          ;;fin de prgm
B024  EB                XCHG                    ;; dans DE
B025  21EC03            LXI H     3ECH          ;;debut de prgm
B028            ;*
B028            ;*--- codage / decodage du programme ---
B028            ;*
B028  0100F8 DEBUT      LXI B     CLE           ;;debut de la cle
B02B  0A      MASQUE    LDAX B                  ;;1 elt de la cle
B02C  AE                XRA M                   ;;<== C'EST ICI
B02D  77                MOV M,A
B02E  23                INX H
B02F  CD14DE            CALL      0DE14H        ;;HL=DE?
B032  CA3FB0            JZ        FIN           ;;codage termine
B035  0A                LDAX B                  ;;) si la cle est
B036  FE0D              CPI       0DH           ;;) finie,on
B038  CA28B0            JZ        DEBUT         ;;) recommence;
B03B  03                INX B                   ;;sinon,element
B03C  C32BB0            JMP       MASQUE        ;;suivant
B03F            ;*
B03F            ;*-- fin :effacer cle,retour au basic --
B03F            ;*
B03F  2100F8 FIN        LXI H     CLE           ;;debut de la cle
B042  3E80              MVI A     80H           ;;
B044  3600    EFFACE    MVI M     0H            ;;on efface la
B046  23                INX H                   ;;zone reservee
B047  3D                DCR A                   ;;a la cle,
B048  C244B0            JNZ       EFFACE        ;;puis
B04B  C3A0C7            JMP       0C7A0H        ;;retour au basic
```

TRAMES

L'utilisation des trames est trés en vigueur dans le dessin industriel, la photo ... et aussi l'informatique. En effet c'est un procédé qui, utilisé à bon escient sur votre DAI permet d'accroitre ses possibilitées

D'une part, une trame simple peut servir à fusionner deux couleurs. C'est, si elle est assez fine, au travers de cette trame que sera crée l'impression d'une troisiéme couleur , mixage des deux couleurs de bases de la trame. Ainsi une trame Rouge et Bleu donnera un effet Violet.
Le motif retenue pour la trame est le suivant :

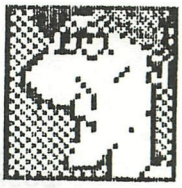


C'est le quadrillage le plus fin que l'on puisse obtenir avec deux couleurs. Car c'est la finesse qui crée l'illusion c'est aussi pourquoi on emploiera , de préference, un mode graphique haute résolution tel que les modes 5,6,7 ou 8. A ce moment la trame devient moins percetible à l'oeuil et le fondu des couleurs est plus réaliste.

Le programme ci-dessous utilise les critéres sus-citée pour afficher les 120 couleurs secondaires, par l'utilisation de toutes les combinaisons possibles de trames bicolores.

IMPINT

```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      MODE 5:DIM A(15)
20      FOR I=0 TO 15 : READ A(I) : NEXT
30      FOR Y=0 TO 15
40      FOR X=0 TO 15
50      COL=A(Y)+16*A(X)
60      AD=#BC21+X*4-Y*1350
70      A=#55
80      FOR J=0 TO 15
90      FOR I=0 TO 3 STEP 2
100     POKE AD-J*90+I,A
110     POKE AD-J*90+I-1,COL
120     NEXT
130     A= INOT (A) IAND #FF
140     NEXT:NEXT:NEXT
200     DATA 0,4,8,15,14,10,3,6
210     DATA 12,1,9,5,13,11,2,7
220     REM                          C.D.
```

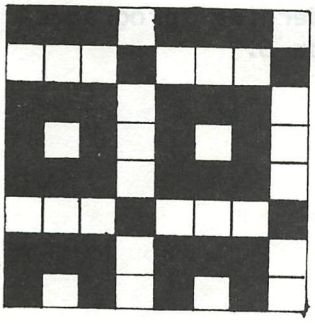


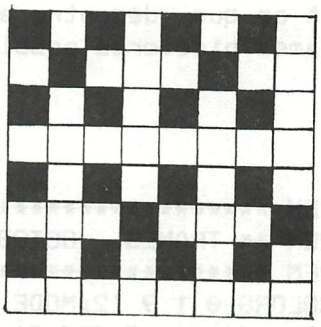
Le tramage des couleurs peut aussi se réaliser à partir de trois couleurs de bases, mais les choix des couleurs ainsi que celui de la trame devront etre judicieux afin d'avoir un mixage convenable.

D'autre part, une trame peut servir à representer une valeur ou une matiere. L'effet recherché est alors differents de celui de la fusion. Ici le fait que la trame soit visible ou non importe peu, ce qui compte c'est l'effet qui s'en dégage. On peut dés lors, dans les modes 'pauvres' en couleurs, tels les modes 6 & 8, faire appel à ces trames pour donner de trés nombreux effets differents.

Le programme suivant démontre mieux qu'un long discour ces effets de textures. Les trames employées ont été déssinées dans un cadre de 8x8, point par point.

Par exemple :





IMPINT

```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      CLEAR 500:DIM A(11,7)
20      COLORG 0 1 9 12:MODE 6
30      FOR M=0 TO 11:FOR N=0 TO 7:READ A(M,N):NEXT:NEXT
40      FOR I=#BFEC TO #8000 STEP -720
50      FOR J=0 TO 68 STEP 2
60      FOR K=0 TO 7:POKE I-90*K-J,A(J/6,K):NEXT
70      NEXT:NEXT
1000    DATA #AA,#44,#AA,#00,#AA,#11,#AA,#00
1010    DATA #55,#EE,#55,#BB,#55,#EE,#55,#BB
1020    DATA #00,#55,#00,#AA,#00,#55,#00,#AA
1030    DATA #44,#40,#5F,#40,#04,#F5,#04,#44
1040    DATA #88,#44,#22,#11,#11,#22,#44,#88
1050    DATA #99,#66,#99,#66,#99,#66,#99,#66
1060    DATA #EE,#AA,#EE,#00,#BB,#AA,#BB,#00
1070    DATA #AA,#55,#AA,#55,#AA,#55,#AA,#55
1080    DATA #00,#77,#00,#BB,#00,#DD,#00,#EE
1090    DATA #AA,#99,#55,#66,#AA,#99,#55,#66
1100    DATA #EE,#11,#EE,#AA,#EE,#11,#EE,#AA
1110    DATA #FF,#FF,#FF,#FF,#FF,#FF,#FF,#FF
1120    REM
```

C.D.

Vous voyez donc que meme avec deux couleurs seulement il est possible de representer :un mur puis une route, une veste et des chaussures ...! avec les trames les plus adaptées, et à chaque matiére sa trame.

Pour obtenir des resultats encore plus beau, vous pouvez mixer les deux methodes et créer des trames de matiéres multicolores. Ces méthodes sont trés interessante dans les mode 6 & 8 car il est trés facile d'y dessiner mais le nombre de couleurs est réduits à 4 (Par lignes).

Enfin pour ceux qui n'aiment pas les Pokes un procédé rapide d'obtention de trames multicolores est le tracé d'une diagonale sur deux. La trame obtenue ainsi est exactement la meme que celle decrite au début. C'est ce que démontre aussi ce dernier programme qui affiche les 6 trames bicolores possibles en mode 6.

IMPINT

```
1       REM ************************
2       REM ** TRAMES - OCTOBRE 84 **
3       REM ************************
10      COLORG 0 1 9 12:MODE 6
20      FILL 40,0 295,255 21
30      FOR I=40 TO 295 STEP 2
40      DRAW I,0 295,295-I 22:DRAW 335-I,255 40,I-40 22
50      NEXT
60      FILL 50,10 150,110 20
70      FOR I=50 TO 150 STEP 2
80      DRAW I,10 150,160-I 21:DRAW 50,I-40 200-I,110 22
90      NEXT
100     FILL 50,245 150,145 21
110     FOR I=50 TO 150 STEP 2:DRAW I,145 150,295-I 23:NEXT
120     FILL 285,10 185,110 22
130     FOR I=185 TO 285 STEP 2:DRAW I,10 285,295-I 23:NEXT
140     FILL 285,245 185,145 23
150     FOR I=185 TO 285 STEP 2:DRAW I,145 285,430-I 20:NEXT
160     COLORG 6 10 14 3
170     REM                                          C.D.
```

N'hesitez pas à me faire parvenir vos plus belles réalisations à partir des trames, ainsi que les trames de matiéres que vous avez construites. Bon 'tramages' !

Cédric DUFOUR

TRUCS

****
Voici deux petits trucs pour le DAI qui vous seront
(je l'espère) très utiles.
****

1> Comment charger un programme BASIC en mémoire sans effacer celui
qui est déjà en mémoire ?

C'est très simple : Vous devez tapez :

```
1 : LOAD <return>   (REM: Chargement du 1er programme)
2 : EDIT <return>   (REM: Ne surtout rien changer dans le programme !!!)
3 : BREAK + BREAK   (REM: Vous avez bien lu : taper BREAK puis BREAK)
4 : NEW <return>    (REM: OUI,j'ai bien dit 'NEW')
5 : LOAD <return>   (REM: Chargement du 2eme programme)
6 : POKE 309,2 <return>
7 : LIST <return> ==> Les deux programmes sont en mémoire !!!!
```

= N.B : Il faut évidement qu'aucun des numéros de lignes ne soient =
= identiques dans les deux programmes !!!) =

****

2> Comment supprimer une partie d'un programme ?

C'est également très simple : Vous devez taper :

```
1 : LOAD <return>          (REM: Charger un programme en mémoire)
2 : EDIT x-y <return>      (REM:les lignes de x à y sont celles que vous
                               désirez garder en mémoire.)
3 : BREAK + BREAK          (REM : sortir de l'éditeur)
4 : NEW <return>           (REM : efface le programe BASIC en mémoire)
5 : POKE 309,2 <return>    (REM: Fait passer la valeur du BUFFER d'édition
                               en mémoire)
6 : LIST <return> (REM : Dans la plupart des cas : OH ! ca marche !!!)
```

= N.B : Ce deuxième petit truc est nettement moins sur que le premier,il =
= ne vaut mieux pas l'employer tout le temps !!! =

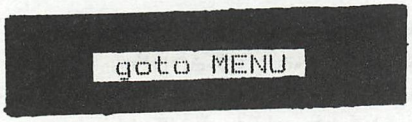
... A Bientot.....Marc Vandermeersch.....

SPECIAL LISTING

```
*****************************************************************************
***                                                                       ***
***      COMMENT INTRODUIRE UN CHR$() DANS UN  LISTING   ? ? ?            ***
***                                                                       ***
*****************************************************************************
```

Plusieurs d'entres-vous se sont, sans doute, déjà demandés comment on plaçait des caractères graphiques (= CHR$<>)dans un listing.

ex : 10 REM
     20 REM
     30 REM



En fait, il s'agit simplement de placer un CHR$(x) dans un"REM". Pour plus de facilités, nous allons toujours travailler avec le CHR$(127) qui est comme tout le monde le sait, un petit carré noir.

Pour parvenir à entrer ce petit carré noir dans un "REM", il faudrait changer la valeur d'une touche du clavier (nous prendrons la touche 'A') par ce CHR$(127).

Voici la marche à suivre : il suffit de taper :

```
1 : UT                                  <return> (REM: aller en UTILITY)
2 : ME8C5 E9C5 5000                     <return> (REM: la mémoire clavier est
                                                 maintenant placée en #5000
                                        (REM: retour au BASIC)
3 : B
4 : POKE#2A7,0:POKE#2A8,#50 <return> (REM: Les 2 'POKE's en 1 seule ligne
                                           sinon on perd le controle du
                                           clavier)
```

Maintenant, nous avons donc déplacé l'espace mémoire du clavier de manière à ce que cet espace mémoire commence à partir de l'adresse hexadécimale #5000

Maintenant, la valeur ASCII de la touche 'A' se trouve à l'adresse #5011 (valeur ASCII de la touche 'B' = #5012 etc...)

Nous pouvons donc remplacer la valeur ASCII de la touche 'A' par un simple petit 'POKE'.

Si on tape "POKE#5011,127",la valeur de la touche 'A' sera le CHR$(127) soit le petit carré noir.(Pour ceux qui ne me croient pas, enfoncer la touche 'A' et vous verrez)

Voilà, avec cela, nous savons donc faire de beaux cadres.

Petite liste de CHR$() intérréssant pour ce truc:

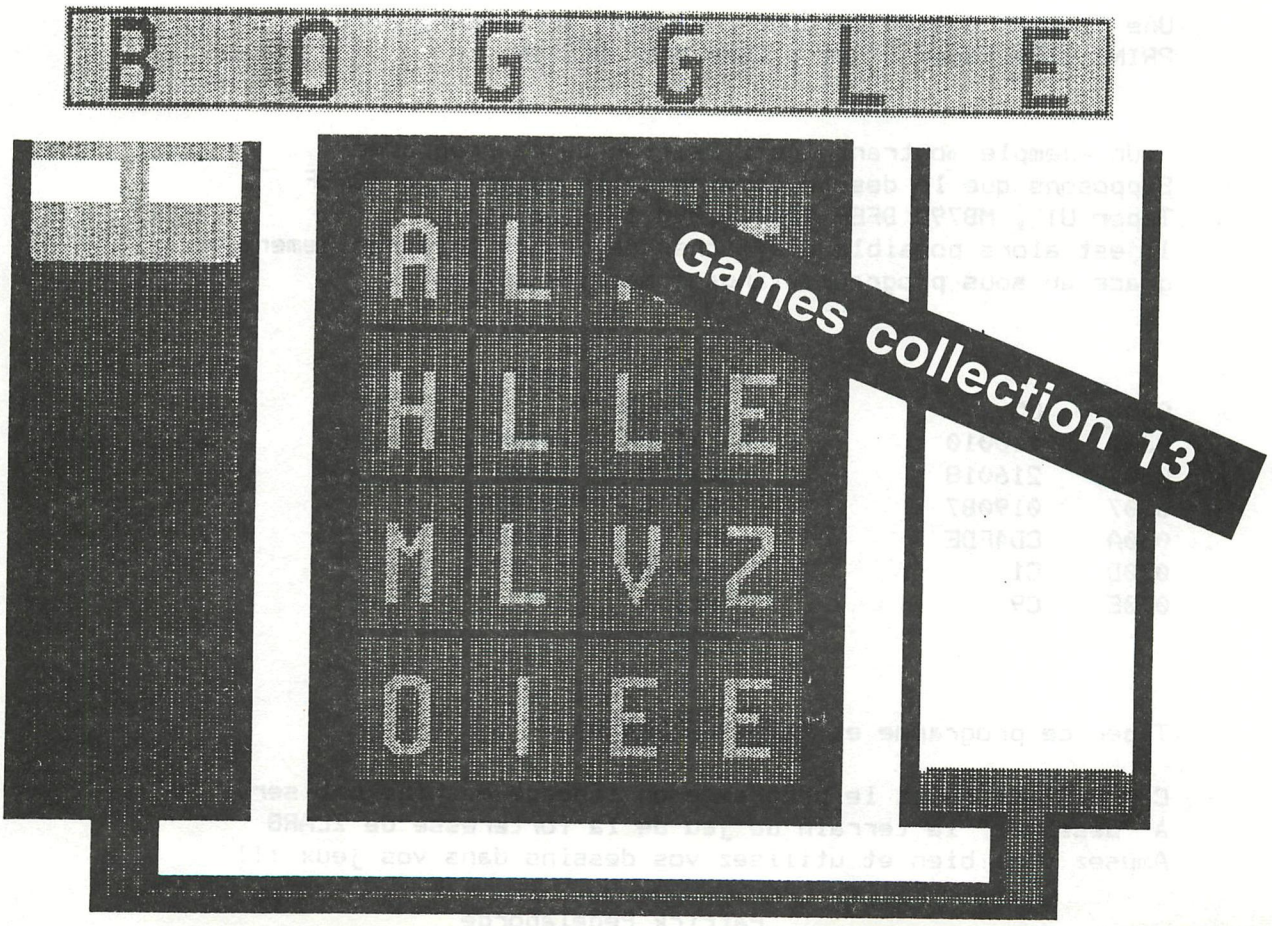
CHR$(29) = petites lignes verticales
CHR$(10) = ligne verticale
CHR$(11) = ligne horizontale
CHR$(30) = petite carré d'une autre couleur
CHR$(12) = vous ne verrez rien si vous poussez si la touche 'A' en mode commande, si vous passer en éditeur et que vous tapez 'A',vous verrez apparaitre une petite flèche vers le bas.

Si vous placez ce CHR$(12) dans un listing, (ex 1000 REM 'CHR$(12),vous verrez que dès que que vous demanderez à l'ordinateur un "LIST" de votre programme, il imprimera les lignes normalement à l'écran mais dès que il rencontre notre petite flèche vers le bas (CHR$(12), l'écran va s'effacer et l'ordinateur reprendra la suite du listing en haut de l'écran.

!!! Une chose à ne pas oublier : pour remettre la valeur ASCII de 'A'!!!
!!! sur la touche 'A',vous devz taper POKE #5011,#41 !!!

Il y a encore bien d'autres petits trucs à trouver avec ces CHR$()...

.....A Bientot.....Marc Vandermeersch.....



UTILITAIRE "DESSIN" SEMI-GRAPHIQUE 16 COULEURS

Cet utilitaire permet de "dessiner" en mode texte 16 couleurs en utilisant n'importe quel caractère semi-graphique du DAI .

Le caractère sera determiné par 3 conditions :

a)La couleur de son fond
b)La couleur du caractère
c)Son code ASCII

Le programme demande au départ sur combien de lignes vous voulez dessiner,la couleur de ces lignes (couleur fond du dessin) et enfin la couleur du cadre qui entoure le dessin. Il suffit ,si vous ne voulez pas de cadre, de le mettre de la couleur des lignes .

Vous aurez besoin d'un paddle 3 dimensions et d'un peu d'imagination !!!
PDL(0) et PDL(1) déplacent le caractère sur l'écran .
Appuyer sur le bouton du paddle fixera le caractère sur l'écran .
Mettre PDL(2) sur 255 change le caractère à afficher .
Le programme demande la redetermination du caractère
(Ne pas oublier de remettre PDL(2) sur 0)

Une fois le dessin terminé , appuyer sur BREAK,faire PRINT DEP$,taper UT et W DEP$ BFEF DESSIN.

Un exemple montrant l'utilisation de ce programme
Supposons que le dessin terminé aille de #B790 a #BFEF
Taper UT , MB790 BFEF 1000,W1000 185F dessin .
Il est alors possible d'afficher ce dessin trés rapidement grace au sous programme langage machine suivant

```
                        ORG     300H
0300    C5              PUSH B
0301    110010          LXI D   1000H
0304    216018          LXI H   1860H
0307    0190B7          LXI B   0B790H
030A    CD4FDE          CALL    0DE4FH
030D    C1              POP B
030E    C9              RET
                        END
```

Taper ce programme et faire CALLM#300

Cet utilitaire et le programme en langage machine ont servi à "dessiner" le terrain de jeu de la forteresse de ZLARG
Amusez vous bien et utilisez vos dessins dans vos jeux !!!

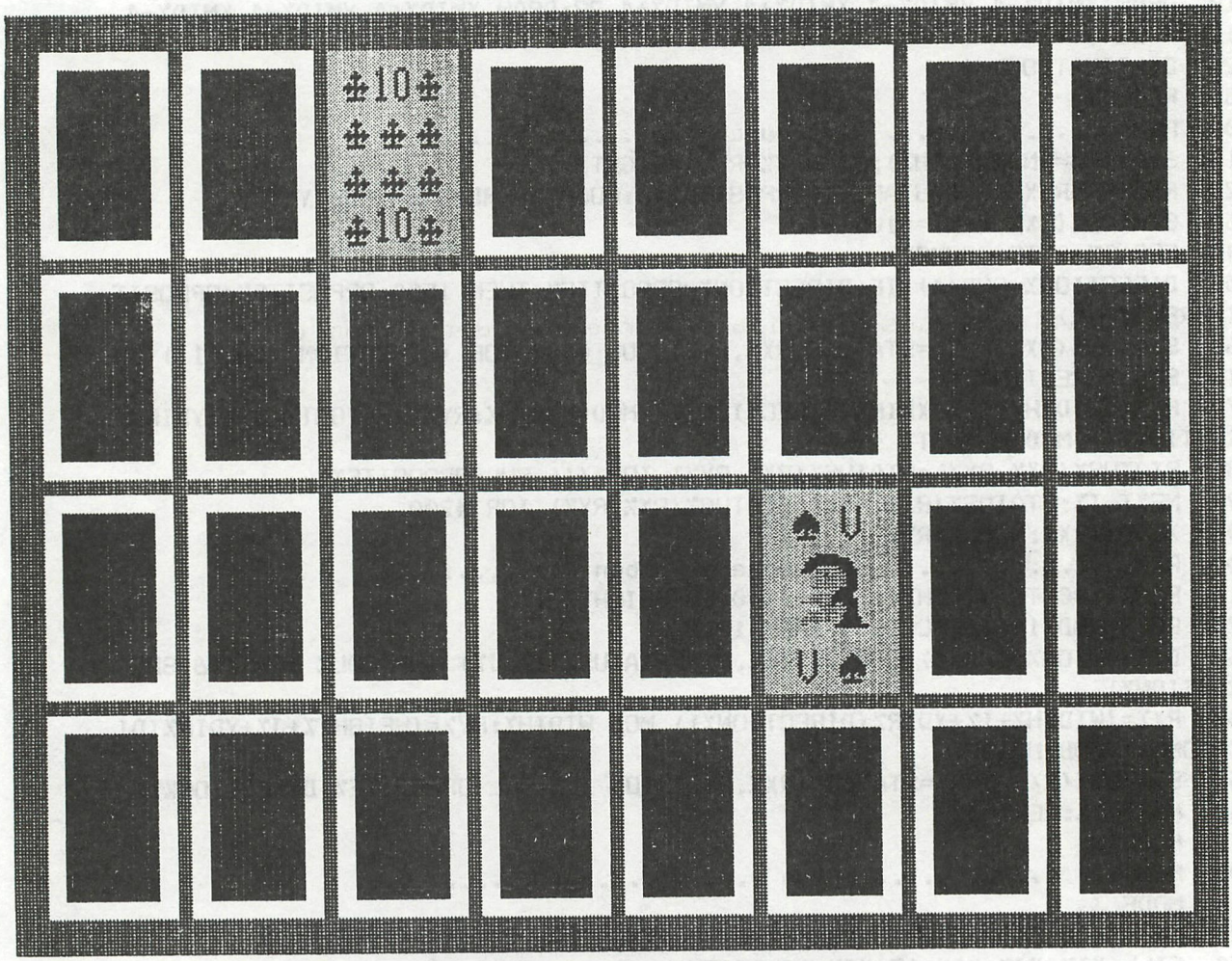
Patrick Pedelaborde

```
2     REM ******************************
3     REM * UTILITAIRE SEMI-GRAPHIQUE *
4     REM ******************************
5     REM
6     REM IMPINT
7     REM
10    MODE 0:PRINT CHR$(12):COLORT 8 0 0 0:POKE #75,95
20    CURSOR 2,20:INPUT "NOMBRE DE LIGNES (MAX20/MIN3)";NL:PRINT
30    IF NL<3 OR NL>20 THEN 20
40    CURSOR 2,19:INPUT "COULEUR DU FOND (DE 0 A 15)";CDF:PRINT
50    IF CFD<0 OR CFD>15 THEN 40
60    CURSOR 2,18:INPUT "COULEUR DU CADRE (DE 0 A 15)";CDC:PRINT
70    IF CDC<0 OR CDC>15 THEN 50
80    CURSOR 20,15:PRINT "C'EST PARTI !!!":WAIT TIME 50:PRINT CHR$(12);
90    DEP$=HEX$(#BFEF-NL*#86+1)
197   REM
198   REM ON MET LES LIGNES EN MODE 16 COULEURS
199   REM
200   FOR I=0 TO NL-1:POKE #BFEF-I*#86,#FA:NEXT
297   REM
298   REM DESSIN FOND ET CADRE AVEC COULEURS DEMANDEES
299   REM
300   FOR I=65 TO 3 STEP -1:A=#BFEF-(NL-1)*#86-I*2:POKE A,#FF:POKE A-3,CDC*16+CDC:NEXT
310   FOR K=1 TO NL-2
320   FOR I=65 TO 64 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDC*16+CDC:NEXT
330   FOR I=63 TO 5 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDF*16+CDF:NEXT
340   FOR I=4 TO 3 STEP -1:B=#BFEF-(NL-1-K)*#86-I*2:POKE B,#FF:POKE B-3,CDC*16+CDC:NEXT
350   NEXT
360   FOR I=65 TO 3 STEP -1:A=#BFEF-I*2:POKE A,#FF:POKE A-3,CDC*16+CDC:NEXT
397   REM
398   REM RESERVATION FENETRE
399   REM
400   FEN=#BFEF-NL*#86:POKE #8A,FEN MOD 256:POKE #8B,FEN SHR 8
497   REM
498   REM QUESTIONS
499   REM
500   CURSOR 0,23-NL:INPUT "COULEUR DU FOND (0 a 15)";CF:PRINT
510   IF CF<0 OR CF>15 THEN PRINT CHR$(12);:GOTO 500
520   CURSOR 0,22-NL:INPUT "COULEUR DU CARACTERE (0 a 15)";CC:PRINT
530   IF CC<0 OR CC>15 THEN 520
540   INPUT "CODE  ASCII DU CARACTERE (0 a 255)";ASCII
550   IF ASCII<0 OR ASCII>255 THEN 540
597   REM
598   REM DEPLACEMENT DU CARACTERE ET DESSIN
599   REM
600   LIGNE=#BFEF-(NL-1)*#86+INT(PDL(1)*(NL+1)/255)*#86
610   POSCAR=LIGNE-INT(PDL(0)/3.9)*2
620   X1=PEEK(POSCAR):Y1=PEEK(POSCAR-3)
630   X2=ASCII:Y2=CC*16+CF
640   POKE POSCAR,X2:POKE POSCAR-3,Y2:WAIT TIME 5
650   POKE POSCAR,X1:POKE POSCAR-3,Y1:WAIT TIME 5
660   IF PDL(2)>200 THEN PRINT CHR$(12);:GOTO 500
670   IF PEEK(#FD00) IAND #20=0 THEN 600
680   X1=X2:Y1=Y2:GOTO 650
```

```
10    REM ................................................
20    REM ..................... WEGWEISER ..................
30    REM ................................................
40    REM ..................... (c) 1984 by Rolf Schall ....
50    REM ................................................
100   REM ............... Erklaerung ........................
110   COLORT 8 0 0 0:MODE 0:PRINT CHR$(12)
120   PRINT TAB(16);"Wegweiserspiel V2.0":PRINT
130   PRINT "Aufgabe: Der gelbe Fleck ist durch ein Labyrinth von Raeumen"
140   PRINT "unterschiedlicher Farbe zu lenken."
150   PRINT "Folgen Sie mit Hilfe der  Cursortasten den Pfeilen  bis  zum"
160   PRINT "durch ein  Kreuz  gekennzeichneten Zielpunkt.  Versuchen Sie"
170   PRINT "danach, zum Ausgangsort mit moeglichst wenigen Schritten zu-"
180   PRINT "rueckzufinden.  Wenn Sie einen Raum verlassen haben,  werden"
190   PRINT "die Wegweiser weggewischt."
200   PRINT "Wenn Sie aufgeben wollen,  druecken Sie SPACE.  Es erscheint"
210   PRINT "dann der Grundriss des Labyrinth mit allen Durchgaengen."
220   PRINT :PRINT "Darin bedeuten:"
230   PRINT "  Schwarzer Fleck: Anfangs- und Zielpunkt"
240   PRINT "      Roter Fleck: Endpunkt"
250   PRINT "    Gruener Fleck: Ihr Standort"
260   PRINT "  Schwarze Punkte: Hinweg"
270   PRINT "    Weisse Punkte: Ihr Weg"
280   PRINT :PRINT TAB(16);"Bitte einen Moment warten..."
300   REM .......... Initialisierung ........................
310   CLEAR 2000:DIFFICULTY%=12:WIDTH%=10:HEIGHT%=8:DIM STATUS%(WIDTH%-1,HE
IGHT%-1),PASSAGE%(3)
320   DIM XDIR%(3),YDIR%(3),XBEG%(3),YBEG%(3),OPPOSITE%(3),XMOV%(3),YMOV%(3
)
330   YDIR%(0)=1:YDIR%(1)=-1:XDIR%(2)=-1:XDIR%(3)=1
340   OPPOSITE%(0)=1:OPPOSITE%(1)=0:OPPOSITE%(2)=3:OPPOSITE%(3)=2
350   FOR I%=0 TO 3:XMOV%(I%)=XDIR%(I%)*8:YMOV%(I%)=YDIR%(I%)*8:NEXT
360   GOSUB 1500:REM ........ Grundriss und Wegberechnung ..
370   PRINT TAB(16);"Spiel startet mit SPACE":CALLM #D6DA
380   REM ............. Init. der Bildkonstanten .............
390   MODE 6:COLORG 8 1 2 14
400   XMID%=XMAX/2+1:YMID%=YMAX/2+1
410   XS%=XMID%+8:YS%=YMID%+8
420   XBEG%(0)=XMID%-16:XBEG%(1)=XMID%+8:XBEG%(2)=XMAX-31:XBEG%(3)=32
430   YBEG%(0)=32:YBEG%(1)=YMAX-31:YBEG%(2)=YMID%-16:YBEG%(3)=YMID%+8
440   FILL 0,0 XMAX,YMAX 1:FILL 8,8 XMAX-8,YMAX-8 8
450   FILL XMID%-16,YMID%-16 XMID%+16,YMID%+16 8
460   REM ............. Spielbeginn ........................
470   RX%=STARTX%:RY%=STARTY%:STATUS%=STATUS%(RX%,RY%) IOR #8000
480   STATUS%(RX%,RY%)=STATUS%
490   GOSUB 1010:GOSUB 1200
600   REM ............... INKEY-Routine .....................
610   KEY%=GETC:IF KEY%=32 THEN 2000:IF KEY%<16 OR KEY%>19 THEN 610
620   KEY%=KEY%-16
630   FILL XS%,YS% XS%+7,YS%+7 8
640   XS%=XS%+XMOV%(KEY%):YS%=YS%+YMOV%(KEY%)
650   IF SCRN(XS%+4,YS%+4)<>8 THEN XS%=XS%-XMOV%(KEY%):YS%=YS%-YMOV%(KEY%)
660   FILL XS%,YS% XS%+7,YS%+7 14
670   IF XS%>7 AND XS%<XMAX-7 AND YS%>7 AND YS%<YMAX-7 THEN 610
800   REM ............... Wechsel des Raumes ................
810   DIRECTION%=KEY%:IF DESTINATION%=1 THEN STPS%=STPS%+1
820   RX%=(WIDTH%+RX%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+RY%+YDIR%(
DIRECTION%)) MOD HEIGHT%
830   STATUS%=STATUS%(RX%,RY%):GOSUB 1000
840   IF STATUS% IAND #2400=#400 THEN GOSUB 1200
850   IF STATUS% IAND #100>0 THEN GOSUB 1300
860   IF STATUS% IAND #200>0 THEN GOSUB 1400
870   XS%=XBEG%(DIRECTION%):YS%=YBEG%(DIRECTION%)
880   FILL XS%,YS% XS%+7,YS%+7 14
```

```
890   STATUS%(RX%,RY%)=STATUS% IOR #2000:REM Raum betreten
900   GOTO 610
910   REM ................. Zurueckgefunden .................
920   PRINT "Schritte hin:";ROOMS%,"Zurueck";STPS%-1
930   END
1000  REM ................. Neuen Raum zeichnen ............
1010  OPPOSITE%=OPPOSITE%(DIRECTION%):COLORG 8 STATUS% IAND #F 2 14
1020  FOR I%=0 TO 3:PASSAGE%(I%)=(STATUS% SHR I%+4) IAND 1:NEXT
1030  FILL XMID%-16,YMAX XMID%+16,YMAX-7 21-PASSAGE%(0)
1040  FILL XMID%-16,0 XMID%+16,7 21-PASSAGE%(1)
1050  FILL 0,YMID%-16 7,YMID%+16 21-PASSAGE%(2)
1060  FILL XMAX,YMID%-16 XMAX-7,YMID%+16 21-PASSAGE%(3)
1070  FILL XMID%-8,YMID%-8 XMID%+8,YMID%+8 20
1080  RETURN
1200  REM ................ Wegweiser zeichnen ..............
1210  WAY%=(STATUS%(RX%,RY%) SHR 11) IAND 3
1220  WX%=XMOV%(WAY%):WY%=YMOV%(WAY%)
1230  DRAW XMID%-WX%,YMID%-WY% XMID%+WX%,YMID%+WY% 23
1240  DRAW XMID%-WY%/2,YMID%-WX%/2 XMID%+WX%,YMID%+WY% 23
1250  DRAW XMID%+WY%/2,YMID%+WX%/2 XMID%+WX%,YMID%+WY% 23
1260  RETURN
1300  REM ................ Start erreicht ..................
1310  FILL XMID%-4,YMID%-4 XMID%+4,YMID%+4 22:FILL XMID%-3,YMID%-3 XMID%+3,
YMID%+3 20
1320  IF DESTINATION%=1 THEN 910
1330  RETURN
1400  REM ................ Ende erreicht ..................
1410  DRAW XMID%-4,YMID%-4 XMID%+4,YMID%+4 22:DRAW XMID%+4,YMID%-4 XMID%-4,
YMID%+4 22
1420  DESTINATION%=1
1430  RETURN
1500  REM ................ Hauptdurchgang .................
1510  STARTX%=RND(WIDTH%):STARTY%=RND(HEIGHT%)
1520  RX%=STARTX%:RY%=STARTY%:OPPOSITE%=5:ROOMS%=RND(DIFFICULTY%)
1530  STATUS%(RX%,RY%)=#100
1540  FOR I%=0 TO ROOMS%
1550  DIRECTION%=RND(4):IF DIRECTION%=OPPOSITE% THEN 1550:OPPOSITE%=OPPOSIT
E%(DIRECTION%)
1560  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR #400 IOR (DIRECTION% SHL 11) IO
R (16 SHL DIRECTION%)
1570  RX%=(WIDTH%+RX%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+RY%+YDIR%(
DIRECTION%)) MOD HEIGHT%
1580  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR (16 SHL OPPOSITE%)
1590  NEXT I%:STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR #200
1600  XEND%=RX%:YEND%=RY%
1800  REM ............. Durchgaenge & Farben ..............
1810  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1
1820  COL%=RND(16):IF COL%=8 THEN 1820
1830  DIRECTION%=RND(4):STATUS%(I%,J%)=STATUS%(I%,J%) IOR COL% IOR (16 SHL
DIRECTION%)
1840  RX%=(WIDTH%+I%+XDIR%(DIRECTION%)) MOD WIDTH%:RY%=(HEIGHT%+J%+YDIR%(DI
RECTION%)) MOD HEIGHT%
1850  STATUS%(RX%,RY%)=STATUS%(RX%,RY%) IOR (16 SHL OPPOSITE%(DIRECTION%))
1860  NEXT J%:NEXT I%
1870  RETURN
2000  REM ................. Plan ........................
2010  MODE 3
2020  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1:XI%=I%*16:XJ%=J%*16
2030  FILL XI%,XJ% XI%+15,XJ%+15 STATUS%(I%,J%) IAND #F
2040  FILL XI%+2,XJ%+2 XI%+13,XJ%+13 8
2050  NEXT J%:NEXT I%
2060  FOR I%=0 TO WIDTH%-1:FOR J%=0 TO HEIGHT%-1:XI%=I%*16:XJ%=J%*16:STATUS
%=STATUS%(I%,J%)
2070  IF STATUS% IAND 16>0 THEN FILL XI%+6,XJ%+14 XI%+10,XJ%+15 8
2080  IF STATUS% IAND 32>0 THEN FILL XI%+6,XJ% XI%+10,XJ%+1 8
2090  IF STATUS% IAND 64>0 THEN FILL XI%,XJ%+6 XI%+1,XJ%+10 8
```

```
2100  IF STATUS% IAND 128>0 THEN FILL XI%+14,XJ%+6 XI%+15,XJ%+10 8
2110  IF STATUS% IAND #400>0 THEN DOT XI%+8,XJ%+6 0
2120  IF STATUS% IAND #2000>0 THEN DOT XI%+8,XJ%+10 15
2130  NEXT J%:NEXT I%
2140  XI%=STARTX%*16:XJ%=STARTY%*16
2150  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 0:REM ..............  START
2160  XI%=XEND%*16:XJ%=YEND%*16
2170  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 2:REM ..............  ENDE
2180  XI%=RX%*16:XJ%=RY%*16
2190  FILL XI%+2,XJ%+2 XI%+12,XJ%+12 5:REM ............  STANDORT
2200  GOTO 2200
3000  REM ................................................
3010  REM ............  BIT-Belegung STAT ................
3020  REM .....  0- 3 : Farben .........  4- 7 : Tueren ...
3030  REM .....     8 : Start ...........     9 : Ende .....
3040  REM .....    10 : Weg ..............  11-12 : Richtung .
3050  REM .....    13 : Vom Spieler betretenes Feld ........
3060  REM ................................................
```



Games collection 13

DBASIC part 4

DBASIC V2.2

1. A new version of DBASIC

In using DBASIC V2.1 you may have noticed the presence of some bugs, specially in error-trapping or extension-evaluation.
All the errors which have been detected and communicated to me have been corrected in DBASIC V2.2. Besides this corrections, some new statements have been implemented. Some of them will be mentioned later on in this article.

2. New extensions

Besides a new version of the DBASIC language, a few extensions are available. One of them generates a structured listing and/or cross-reference : this extension will be the main subject of this article.
Another extension allows you to define and use function keys. See PFK and PFK$ in the program listing.

3. Structured listing and cross-reference

The program listing on the next pages is an example of the output produced by the $XREF extension. By simply typing $LIST a structured listing of the complete program, followed by a cross-reference is produced.
Another command, XREF, will skip the structured listing and will directly output a cross-reference.

3.A Structured listing

As can be seen in the program listing, DBASIC program lines are spread out on several listing lines to enable correct indentation of :

- .iteration structures (ex. FOR/NEXT,REPEAT/UNTIL,WHILE/WEND).
- .selection structure (ex. IF/ELSIF(new in DBASIC V2.2)/ELSE/END IF).
- .definition structures (ex. PROCEDURE(DEF PROC)/END PROC,FUNCTION(DEF FN)/END FN).

A counter also shows the level of indentation.

Special care has been taken for some details :

- .line numbers are printed rigth justified
- .keywords, symbols or text will not be splitted at the end of the line
- .the title will be truncated if it is to long
- .pages are numbered

3.B cross-reference

A cross-reference could be a very usefull help in debugging your software. For clarity the DBASIC cross-reference groups all symbols into 6 different classes :

```
.extended commands & functions..(X option)
.procedures.....(P option)
.functions.....(F option)
.labels.....(L option)
.arrays.....(A option)
.variables.....(V option)
```

Special care has been taken for some details :

- .for functions,arrays and variables the type is indicated
- .for functions and arrays the () indicates the necessity of arguments
- .line numbers are rigth justified
- .the first line number indicates the line where the symbol is defined (ex. procedures,functions,labels,arrays) or used the first time

3.C General syntax

The general syntax is :

```
$LIST[ first line][ last line][,title][;options]    and
 XREF[ first line][ last line][,title][;options]    with
```

```
first line : line number from where to start the structured listing
last line  : line number where to end the structured listing
title      : the heading on each page
options    : options for the cross-reference output
```

ex. $LIST 100 500,"TEST PROGRAM";"FP"

This command asks for a structured listing starting with line number 100 up to line number 500 with heading TEST PROGRAM and produces a cross-reference of functions and procedures

3.D Default values

```
default range   : complete program
default header  : the string expression following the DBASIC TITLE
                  statement. If the program does not contain a TITLE state-
                  ment the default header is an empty string.
                  (note : the TITLE statement is also used as a default for
                  SAVE)
defaut1 options : "XPFLAV" i.e. first print the cross-reference of all
                  extended commands & functions, then the cross-reference
                  of all procedures...and so on...
```

4. Other DBASIC aprovements

Besides error correction and implementation of new statements, I have implemented extended functions in DBASIC V2.2 (see PFK$). The execution time of extended commands and functions has been diminished quite a bit due to the addition of an extra compilation pass. For more information see the documentation of DBASIC V2.2 and the extension's documentation.

Willy Coremans

```
DBASIC V2.2
PAGE      1   MANIPULATE PROGRAMMABLE FUNCTION KEYS


    10        ON BREAK GOTO "TRAPBREAK
    20        TITLE "MANIPULATE PROGRAMMABLE FUNCTION KEYS"
    30        REM to enable/disable function keys--type PFKON/PFKOFF
    40        REM to use function keys--type <ctrl> and 0 to 9 at the same time
   100        PROCEDURE HOME :
         1     PRINT CHR$(12);:
               END PROC
   200        PROCEDURE PFKTOARRAY  ARR A$:
         1     LOCAL I
   210   1     FOR I=0 TO 9:
         2      A$(I)=PFK$(I):
         1     NEXT
   220         END PROC
   300         PROCEDURE ARRAYTOPFK  ARR A$:
         1     LOCAL I
   310   1     FOR I=0 TO 9:
         2      PFK I,A$(I):
         1     NEXT
   320         END PROC
   400         PROCEDURE CLRPFK :
         1     LOCAL J
   410   1     FOR J=0 TO 9:
         2      PFK J,"":
         1     NEXT
   420         END PROC
   500         FUNCTION PRNTPFK$(J):
         1     LOCAL I:HOME
   510   1     FOR I=0 TO J:
         2      PRINT PFK$(I):
         1     NEXT
   520   1     FN = "":
              END FN
  1000        "START
              DIM AR$(9)
  1005        REPEAT
         1     HOME
  1010   1     CURSOR 10,20:PRINT "1--ARRAY TO FUNCTION KEYS"
  1020   1     CURSOR 10,19:PRINT "2--FUNCTION KEYS TO ARRAY"
  1030   1     CURSOR 10,18:PRINT "3--CLEAR FUNCTION KEYS"
  1035   1     CURSOR 10,17:PRINT "4--PRINT FUNCTION KEYS & END PROGRAM"
  1040   1     CURSOR 0,0:PRINT "SELECT CHOICE ";
  1050   1     REPEAT
         2      I=GETC-ASC("0"):
         1     UNTIL I>0 AND I<5
  1055   1     CURSOR 14,0:PRINT I;:WAIT TIME 10
  1060   1     IF I=1 THEN
         2      ARRAYTOPFK AR$()
  1070   1     ELSIF I=2 THEN
         2      PFKTOARRAY AR$()
  1075   1     ELSIF I=3 THEN
         2      CLRPFK
  1080   1     ELSE
         2      PRINT PRNTPFK$(9)
  1090   1     END IF
  1100        UNTIL I=4
  2000        END
  3000       "TRAPBREAK
              CONTINUE
```

extended commands & functions :

PFK	310	410
PFK$	210	510

procedures :

ARRAYTOPFK	300	1060	
CLRPFK	400	1075	
HOME	100	500	1005
PFKTOARRAY	200	1070	

functions :

PRNTPFK$()	500	1080

labels :

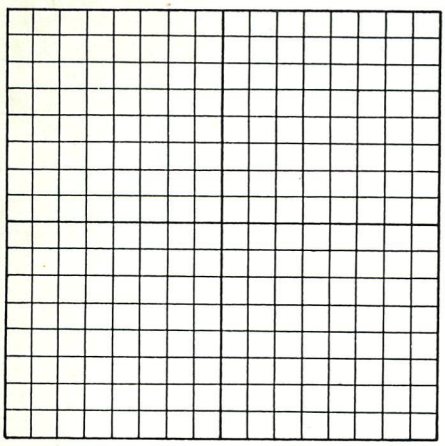
START	1000	
TRAPBREAK	3000	10

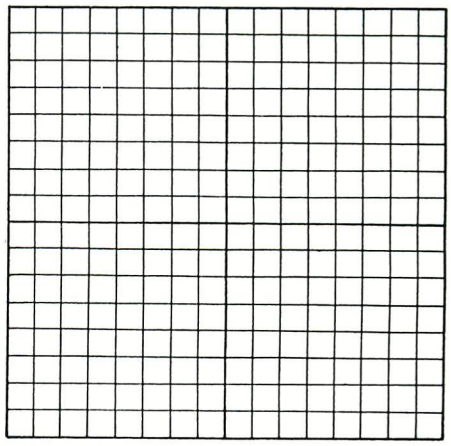
arrays :

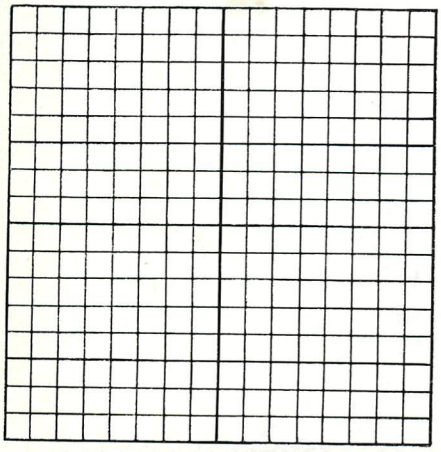
AR$()	1000	1060	1070	
A$()	200	210	300	310

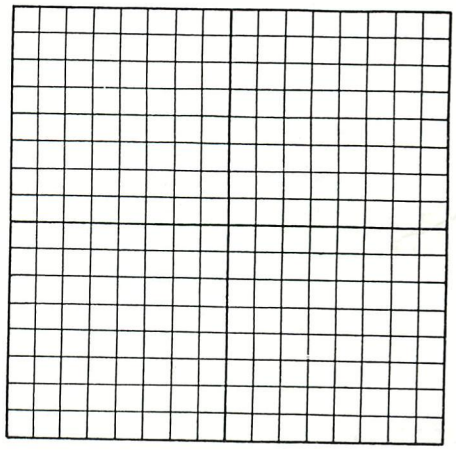
variables :

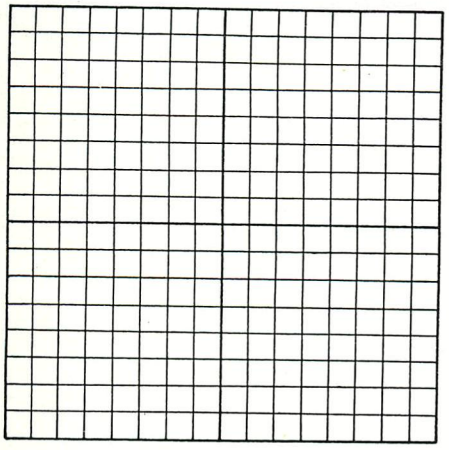
I%	200	210	300	310	500	510	1050	1055	1060
	1070	1075	1100						
J%	400	410	500	510					

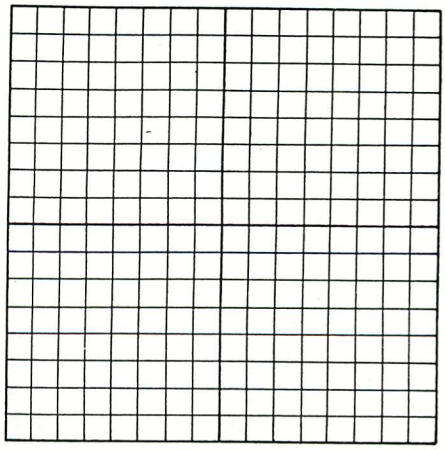












16 x 16 design

FYSISCHE GEOGRAFIE en KLIMATOLOGIE

1- OMBROTHERMISCH DIAGRAM

2- KLIMAATSKLASSIFIKATIE VAN BAGNOULS

3- ANALYSE VAN POLYGONEN

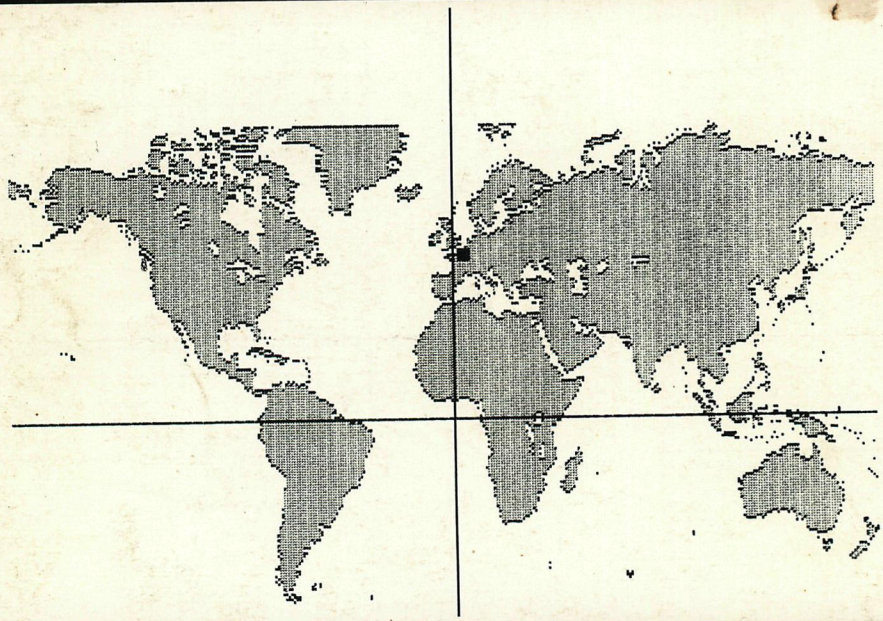
4- INOEFENEN KAARTSCHAALBEREKENINGEN

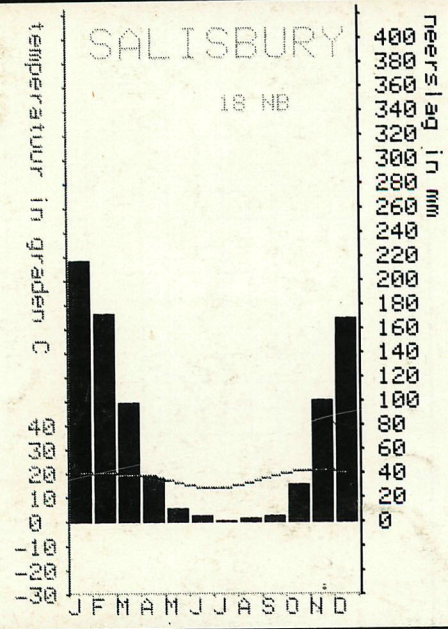
5- INOEFENEN BEREKENEN VAN DE HELLINGSGRAAD

6- RELIEFSPROFIEL EN VERTIKALE OVERDRIJVING

7- KOSMISCHE KALENDER

MA+E.M DJ-84





BAGNOULS spreekt van een KOUDE MAAND (K.M) als het maandgemiddelde < 0 °C

Tik het aantal koude maanden in (0,1,2,...,12) en noteer het cijfer

K.M. = ?