

personal computer users club

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

International

COLOFON

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

DAInamic redactie :

Dirk Bonné	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	Kredietbank Herselt
3170 Herselt	nr. 401-1009701-46
Tel. 014/54 59 74	BTW : 420.840.834

Lidgelden / Subscriptions

Bruno Van Rompaey	Generale
Bovenbosstraat 4	Bankmaatschappij
B 3044 Haasrode	Leuven
België	nr. 230-0045353-74
tel. : 016/46.10.85	

Voor Nederland : Pour la France :

GIRO : 4083817	DAInamic FRANCE
t.n.v. J.F. van Dunne'	C. Dufour
Hoflaan 70	Rue Lavoisier 9
3062 JJ ROTTERDAM	59149 DUNKERQUE
Tel. : (010) 144802	Tel. 02866 3339

Inzendingen : Games & Strategy

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

DAInamic
 PERSONAL COMPUTER USERS CLUB

4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15

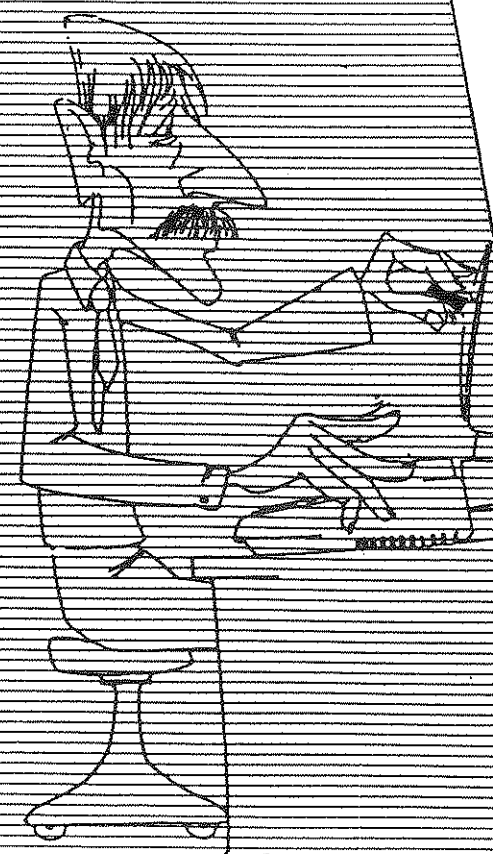
belangrijke ASCII-waarden in DAInpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

Remark



Herselt, aug 85

Beste leden,

DAInamic bestaat 5 jaar. Om dit te
 vieren organiseren we de jaarlijkse
 software-aanbieding met korting. Voor de
 trouwe lezers die al de uitgaven netjes
 hebben bewaard zit er op p.255 een
 wedstrijd. Naast de leuke van het
 zoekwerk heeft U ook de kans om een RGB-
 monitor te winnen of een van de andere
 prijzen. Ondertussen is de redactie
 verhuisd naar de nieuwe lokalen. Op
 afspraak kan U hier terecht voor
 demonstratie, uitwisseling van
 programma's of gewoon een praatje. We
 reserveren voor U de woensdagnamiddag en
 de zaterdagvoormiddag. De voorbije
 vakantie gaf ons de gelegenheid om
 DAInamic een degelijke facelift toe te
 dienen. We hopen dat U tevreden bent met
 het resultaat. In dit nummer worden een
 7-tal nieuwe pakketten aangekondigd. In
 combinatie met de reuzekorting kan uw
 DAInbibliotheek weer wat rijker worden.
 Na zijn D-BASIC project waren de ideeën
 van Willy Coremans nog niet uitgeput :
 in dit nummer een eerste beschrijving
 van zijn nieuwste realisatie : een
 volwaardige DISK BASIC als D-BASIC
 extensie. Spoedig verkrijgbaar voor de
 INDATA-systemen 2 X 320, 2 X 160 en 2 X
 80. Voor een korte bespreking van de
 inhoud van nummer 29 verwijzen we naar
 bladzijde 201. Dit zijn de spelregels
 voor onze verjaardagsactie :

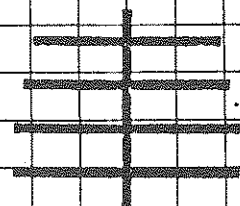
2000 fr of meer: -10%
 4000 fr -20%
 6000 fr -30%
 8000 fr -40%
 meer dan 10000 fr : -50%
 De korting wordt berekend op het
 totaal bedrag, toeslag voor DCR niet
 inbegrepen.

met vriendelijke groeten,
 W.Hermans

DAInamic redaction

5th Anniversary Contest

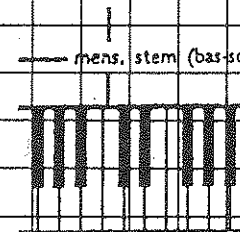
Inhoudstafel - Contents



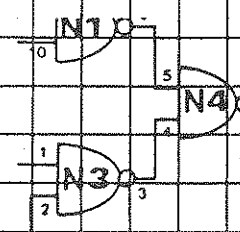
9



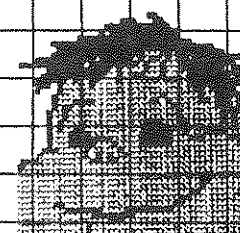
10



12



3



14

199	VOORWOORD	REDACTION
200	INHOUDSTAFEL - CONTENTS	REDACTION
202	PROGRAMMEERTECHNIEKEN	F. DRUIJFF
206	CASSETTEPROGRAMMEUR	S. DUBOURG
208	TIJD	F. DE JONGHE
210	SELFDESTROY	R. SCHALL
211	INTEGREX COLOURJET 132	L. GIDNEY
212	HIGH SPEED LOADER	H. RISON
215	DAI DOS 1541 RELATIVE FILES	J. BOERRIGTER
220	CRITICAL PATH ANALYSIS	M. ANTROP
221	FWP IN DUBBELKOLOMS	F. DRUIJFF
223	VIDEO REPORTER	G. PONS
224	UNIPROM	A. J. BATHE
226	NEW SOFT	REDACTION
228	DEMO TURTLE-BASIC	O. PATTINIEZ
229	DISABLE BREAK	W. HERRMANN
230	DBASIC GOES DISCO	W. COREMANS
234	CHAMPIONNAT OTHELLO	O. I.
236	RANDOM NUMBERS	J. HARLOW
237	PROGRAMMOTEEK	L. DIETZ
245	LOW COST DIGITIZER	P. DE LAET
255	5th ANNIVERSARY CONTEST	REDACTION

© DAIInamic

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt in gedrukte, akoestische, filmische of welke andere vorm dan ook zonder geschreven toestemming van de uitgever.

5 in other words...

DAInamic is 5 years young. We celebrate this birthday with a special reduction offer and an anniversary **Contest**. See p 255 for the contest-info. In his article **Programmeertechnieken** Frank Druijff explains the use of VARPTR and how DAI stores variables in RAM. The program cassette programmeur by S. Dubourg offers the solution to program audio recordings. The power of the recording system should be connected to the remote control of DAI cassette connector. (By means of a relay we assume). In his program **Tijdberekeningen**, Frank De Jonghe calculates calendars for a chosen year and calculates the day for a given date. Be careful with the program **Selfdestroy** of Rolf Schall: he can ruin the result of a whole weekend of typing... We regret we can not publish the contribution of Louis Gidney in multicolour. We can only assure you that the colours of the printout of his **Integrex Colourjet 132** are beautiful! We ask Louis to supply more information about his activities with this colourprinter. Henk Rison gives you all the details on his **High Speed Loader**. If you have some programs you need very often and you want no loading delays, then this project could be your solution. If you contact him, maybe he can supply you the print layout as well. By the way, I hope you like DAIInamic with the new facelift... Jan Boerrigter, another member of the Limburg-clan, continues to explore the facilities of the Commodore drive. Today's subject is **Relative Files** (well known as RANDOM ACCESS). If you have a lot of jobs to do and you don't know where to begin, the **Critical Path Analysis** program from Marc Antrop could be your solution. The computer will create the best possible solutions for you... This program is available from our software library.

With our anniversary reduction, this could be another bargain! As we mentioned a few issues ago, publishing in 2 columns is much more comfortable to read. Frank Druijff created this small BASIC program to split a **FWP** text file in 2 columns. He even suggests that a machine-language routine might be available soon. Gilbert Pons asks everyone with cinema-allures to join his video team. This could be your way to Hollywood? J. Bathe offers another Epromprogrammer with the necessary soft. If you charge your TURTLE-BASIC program, you can see the demo created by Olivier Pattiniez: **Demo Turtle-Basic Disable Break** during program run, and activate break in command mode: that's the result of the very small routine by Willi Herrmann. Willy Coremans has extended his **D-Basic** with a complete new DISK BASIC. Now you can use your drive with maximum results for databases, file handling and so on. Soon available! The **Othello-Reversi** freaks can test the stenght of their programs at the O.I championnat in Paris. Send a photocopy of the subscription leaflet to O.I. and join the competition ... good luck! Referring to the article of Raymond Vanlathem, Jonathan Harlow gives some extra information about **Random Numbers**. With **Programmoteek** you can make an inventory of all your DAI programs. As the program is rather long, you will find the program in **Toolkit 7**. Due to the many activities during the past months, we can not offer you a printed circuit layout of the **Low Cost Digitizer** but here is the complete story. The result on page 253 equals the quality of our unit we bought last year! The digitizer programs are also included in toolkit 7. I hope you didn't throw away the old issues of DAIInamic: if you find the solution, and you are lucky, we will invite you to come and get a beautiful RGB-monitor! till next issue...

De vorige keer eindigde ik met een aankondiging van de VARPTR. Nu zal ik deze vollediger behandelen en tevens daarbij van de gelegenheid gebruik maken U eens mee te laten kijken hoe deze zaken uitgevonden kunnen worden door een iegelijk die maar doorzet.

VARPTR

Het woord VARPTR is een samentrekking van de woorden VARIABLE PointER. Deze naam wijst er al op dat we met de functie een variabele kunnen laten aanwijzen. We kennen bij de DAI verschillende soorten variabelen en voor dit verhaal is het nodig daar onderscheid tussen te maken.

We kennen de integer of geheel getals variabelen en de floating point of drijvende komma getallen. Deze variabelen kunnen zowel op zich zelf staan als met andere samen in een array zitten. Naast deze getalvariabelen kennen we ook nog de tekstvariabelen en ook die kunnen zowel alleenstaand als in een array voorkomen.

Opfrissen

Om het onderzoeksverhaal niet al te lang te maken zal ik eerst een aantal relevante zaken (zo nodig) opfrissen.

Bij de DAI zijn, in tegenstelling tot bijna alle andere computers, zowel voor de integer als de floating point variabelen vier bytes gereserveerd. Bij de meeste andere computers is er voor de integers maar een twee bytes ruimte, maar voor de floating point vaak vijf of meer, tot wel tien toe. Aantallen van zes of meer worden vaak met double precision (=dubbele nauwkeurigheid) aangeduid. Dit is ook de reden dat de DAI veel grotere getallen in integer aankan dan de meeste andere computers, maar met de nauwkeurigheid bij de floating point getallen het laat afweten.

De namen en de type-identificatie van alle variabelen staan in de zogenaamde symbol table. De waarde van de

variabele staat bij de losse integer en floating pointer eveneens in de symbol table. De tekstvariabelen worden echter in de zogenaamde heap opgeslagen. Interpreteer dit laatste wel goed: de waarde van de variabele (dus de tekst die er in staat), die staat in de heap maar de naam van de variabele staat in de symbol table. De waarden, die getalsvariabelen hebben, die in een array zitten, staan ook in de heap.

Onderzoek

We zetten de machine uit (ON op OFF! zetten of omgekeerd voor de bezitters van revisie 4) en niet RESET want dat is in dit geval echt iets anders. Dan zetten we de computer weer aan. We tikken UT[RETURN] in en komen zo in de utility. Om een aantal pointers te weten te komen tikken we in:

```
D29B 2A6
```

We krijgen dan:

```
029B EC 02 00 01 EC
02A0 03 ED 03 EF 03 50 B3
```

Hierin kunnen we het volgende aflezen

```
29B-29C start heap is . . . . . 2EC
29D-29E size heap is . . . . . 100
29F-2A0 start basic tekst is . 3EC
2A1-2A2 start symbol table is . 3ED
2A3-2A4 end synbol table is . . 3EF
2A5-2A6 bottom screen is . . .B350
```

Alle bovenstaande waarden zijn in het zestientallig of hexadecimaal stelsel gegeven en staan zoals bij de 8080 gebruikelijk is laag-hoog genoteerd. We vervolgen met D2EC 400 en zullen dan zien dat alleen aan het begin en aan het eind van de heap wat anders dan nullen staat. De twee maal twee bytes worden gebruikt voor de interne organisatie van de heap en doen hier verder niet ter zake. Weet echter wel dat ze kunnen veranderen, zoals we later ook zullen zien. Is de rest niet nul dan heeft U de machine niet uitgezet!

We keren met B terug naar BASIC en tikken daar in:

```
A%=3
?VARPTR(A%)
1008 is het antwoord dat we krijgen
vervolgens tikken we
?HEX$(VARPTR(A)) of ?HEX$(1008)
en krijgen
3F0
We gaan weer naar utility en laten
daar het stuk rond het adres 3F0 displayen met:
```

```
D3EC 3FF
```

Het antwoord van de DAI luidt:

```
00 01 41 14 00 00 00 03 en verder 00
↑↑
Hier is 3F0
```

We gaan nu de procedure herhalen met A! in plaats van A%. Ook nu blijkt de VARPTR(A!) 1008 te zijn en de hexadecimale notatie daarvan dus weer 3F0. In utility krijgen we nu echter bij

```
D3EC 3F3
```

```
00 01 41 04 02 C0 00 00
↑↑
Hier is 3F0
```

Schoonmaken

De DAI moest weer opnieuw uitgezet worden anders krijgen we echt andere resultaten; al moet ik toegeven dat hier RESET of NEW ook voldoende was geweest.

Wilt U om welke reden dan ook de DAI liever niet steeds aan/uit zetten kan bij dit onderzoek worden volstaan met de volgende procedure:

```
UT
F2EC 4FF 0
B
NEW
```

Maar bij andere onderzoeken zal dit niet gelden, denk maar eens aan het onderzoek uit DAInamic 28 naar de RND van Raymond Vanlathem.

U ziet dat er voor een gericht onderzoek vaak veel voorkennis is vereist, maar bedenk wel, dat anderen die voor enige jaren ook niet hadden.

Analyse

Goed nu de analyse van de verkregen resultaten.

Gedachtig het feit dat een integer in de DAI vier bytes inneemt is het m.i. redelijk te veronderstellen dat die vier bytes die de waarde van de variabele bevatten, staan op de adressen 3F0, 3F1, 3F2 en 3F3. De inhoud van 3F3 namelijk 03 wijst daar al op. We kunnen een en ander controleren door de variabele A% b.v. 2,4 of 7 te maken. Meer dan tien mag ook maar dan is enige kennis van het zestientallig stelsel vereist.

Geef echter eens de instructie: A%=#1234567 en we kunnen dat simpel vanaf 3F0 terugvinden.

Als we onze ASCII-codes goed kennen zullen we vast en zeker ook de 41 herkennen als de ASCII-code van 'A'.

Met deze laatste wetenschap kunnen we iets aardig doen waarbij we tevens een toepassing van de VARPTR zien. We zullen een programma 'onleesbaar' maken door alle variabelen dezelfde naam te geven. Vraag mij echter niet naar het nut hiervan.

```
10 LIST
20 A=2
30 B=3
40 PRINT A+A
50 PRINT A+B
60 PRINT B+B
70 POKE VARPTR(A)-2,88
80 POKE VARPTR(B)-2,88
90 LIST -
```

Vanzelfsprekend kunnen we ook andere ASCII-codes dan 88 nemen. 'Grappige' voorbeelden zijn 8 of 12 en ook de spatie (32) kan verwarring geven.

De andere drie bytes die staan op 3ED 3EE en 3EF geven informatie over naam lengte en de type-identificatie.

We gaan verder met de getallenarray's. Zo'n array is in feite niets anders dan een aantal variabelen, die alle dezelfde naam hebben. Zoals reeds vermeld werd staat die naam in de symbol table, maar de waarden van de variabelen uit die array in de heap. Als we logisch nadenken zullen we inzien dat er meer moet instaan. In de symbol table dient te staan waar in

de heap de gewenste array te vinden is. Vervolgens dient ergens ook nog de informatie over de array (hoeveel dimensies en hoe groot in elk van de dimensies) te staan. Bij de DAI staat deze laatste informatie in de heap.

Voor de VARPTR is dit alles alleen voor de DAI een probleem. Voor de gewone programmeur is het niet echt van belang.

Begin weer zoals we steeds begonnen en tik vervolgens in :

```
DIM A$(3)
?VARPTR(A$(0))
We krijgen dan :
752
?VARPTR(A$(1))
geeft
756
en
?HEX$(752)
2F0
We vervolgen met
POKE 756,4
?A$(1)
en krijgen dan als antwoord
67108864
schrijven we dit binair zien we :
```

```
00000100 00000000 00000000 00000000
```

en wat staat er binair op de adressen 756,757,758 en 759 ????. Precies !!!! Trouwens had U ook kunnen bedenken dat 67108864 precies 2↑26 is en er staan 26 nullen achter de 1.

Door dus ijverig te poken en steeds te kijken wat er gebeurde kunnen we ons een deel van de hexadecimale code namelijk de notatie zelf aanleren.

Tekstvariabelen

Als laatste de moeilijkste te weten de tekstvariabelen. Het probleem bij deze variabelen schuilt in het feit dat we niet van tevoren weten hoe lang de tekst is. De inhoud staat altijd in de heap maar ook hier staat de naam van de variabele in de symbol table.

In de symbol table zal dus net als bij de array's een verwijzing naar de heap moeten zijn. Deze verwijzing is een adres en dus maar twee bytes lang en hebben we niet van tevoren het geheugen ge'schoon'd zal een 'oude'

variabele in de symbol table ook nog zichtbaar kunnen zijn. Alleen NEW of een RESET is nu echt niet voldoende. We tikken in :

```
A$="3"
?VARPTR(A$)
en dit laatste levert tot onze verba-
zing 1008 op dus hexadecimaal weer de
3F0. Dit adres ligt duidelijk in de
symbol table en er was net gezegd dat
de variabele in de heap zit.
```

```
UT
D3EC 3F3
Dit levert op :
00 21 41 22 ED 02
↑↑
Hier staat 3F0
```

We herkennen de naam van de variabele in de 41 en we verwachten de inhoud te vinden vanaf het adres 2ED dat in de symbol table wordt genoemd.

```
D2ED 2EF
levert echter op :
01 33 80 F9
```

De inhoud van de laatste twee bytes herkennen we na enige oefening wel als de bij de heap behorende pointers Maar nu de 01 en de 33. Ik wil niet langdradig worden met mijn uitleg maar als het niet bekend is lees dan nog even niet verder en tracht het zelf op de aangegeven methode te vinden. Welnu de 01 geeft de lengte van de variabele (de inhoud niet de naam!) aan en de 33 is die inhoud; de '3' maar dan in de ASCII-code.

Om hiemeer wat te oefenen tikken we het volgende in :

```
NEW
A$="3"
Probeer nu eens met een geschikte
POKE de inhoud van A$ te veranderen.
Het is dit geval het gemakkelijkst
als we beginnen met :
```

```
A=VARPTR(A$)
en dan
POKE PEEK(A)+PEEK(A+1)*256+1,ASC("*")
```

of als we het met alle geweld ineens willen doen met

```
POKE PEEK(VARPTR(A))+PEEK(VARPTR(A)+
+1)*256+1,ASC("*")
```

Verklaring

Met A=VARPTR(A\$) is de variabele A 1008 geworden. Op de adressen 1008 en 1009 (decimaal) of 3F0 en 3F1 (hexadecimaal) staat het adres in voor de 8080 gebruikelijke omgekeerde notatie in hexadecimale vorm. Om het adres te berekenen gaan we als volgt te werk : eerst nemen we inhoud van 3F0 en 3F1 met PEEK(A) en PEEK(A+1). De inhoud van 3F1 vermenigvuldigen we hierna met 256 of #FF. Dan deze twee waarden bij elkaar optellen en we hebben het adres in de heap, waar de tekstvariabele staat. Op dat adres staat echter niet de inhoud maar de lengte van de variabele.

De lengte staat in een byte genoteerd en kan dus niet groter dan 255 zijn. Bij ons berekende adres moet nog een geteld worden om inderdaad de eerste byte van de inhoud van de tekstvariabele te krijgen.

POKE'n we dan in de byte op dat adres kunnen we zo de inhoud veranderen.

Tekstarray's

De laatste mogelijkheid om de VARPTR te gebruiken. Hopelijk naar het voorafgaande niet meer zo moeilijk te begrijpen. Voor de hand liggend vind ik dat er een soort combinatie van de array-behandeling en de tekstvariabele-behandeling. We gaan kijken :

We beginnen vanzelfsprekend weer met een schone machine en tikken dan in :

```
DIM A$(3)
?VARPTR(A$(0))
geeft
752
en dat is hexadecimaal genoteerd 2F0.
```

We gaan in utility eens kijken wat er in de symbol table staat.

```
UT
D3EC 3F3
levert
00 61 41 62 EE 02 00 00
```

De bytes 3ED, 3EE en 3EF geven ook hier weer de type-identificatie en de naam. Maar het adres 2EE dat we zien

op de adressen 3F0 en 3F1 verwacht misschien de onderzoeker. Toen we met de VARPTR van A(0) vroegen kregen we toch 2F0 en niet 2EE? Laten we eens kijken wat er in de heap op deze plaatsen staat.

```
D2EC 2F9
Levert ons op
00 0A 01 03 00 00 .. .. 00 00 80 F0
↑↑ ↑↑
↑↑ Hier staat 2F0
^↑↑
Hier staat 2EE
```

Na enig denkwerk en het eventueel veranderen van b.v. de dimensie van de array of het aantal dimensies van de array zien we in dat op 2EE het aantal dimensies van de array gemeld wordt, gevolgd door de gereserveerde grootte in die dimensies.

Daarna volgt de array zelf, startend op adres 2F0.

De 00 0A die op 2EC en 2ED staat geeft aan dat voor de er op volgende array A(=10) bytes zijn gereserveerd. Hier zijn dat dus twee bytes voor de identificatie en (4 x 2) acht bytes voor de inhoud.

Het is echter niet zo simpel als hier geschetst; we hebben namelijk met een tekst-array te maken en bij tekst weten we niet hoelang die tekst is. Net als bij de tekstvariabelen zal de VARPTR alleen maar het adres aangeven waar we de tekst zelf kunnen vinden.

samenvattend:

De VARPTR geeft een adres. Op dit adres begint een nieuw adres. Dit laatste adres geeft aan waar de lengte gevolgd door de tekst staat.

Als we nu met A\$(0)="2" het eerste array element gaan vullen zien we op 2F0 en 2F1 het adres 2F9 genoemd worden (in de volgorde F9 02). Vanaf 2F9 vinden we 01 32 80 ED 00 00 met als eerste de lengte van onze tekst (=1) en daarna de tekst zelf in ASCII-notatie is 2 namelijk 32. Erachter stond 80 F0 de verandering komt door de verandering in gebruik van de heap.

Frank H. Druijff

Cassette Programmeur

```
10 PRINT CHR$(12):POKE #29C,5:POKE #29B,#FF
15 CLEAR 3000:DIM F(3.0),D(3.0)
20 GOSUB 1020:GOSUB 1020
25 GOSUB 1020
30 GOSUB 1020:GOSUB 1020
35 GOSUB 1020
40 GOSUB 1020
41 COLORT 9 14 5 3
50 CALLM #3A1:INPUT "HH,MM,SS";HH,MM,SS:PRINT
55 POKE #300,0:POKE #301,SS MOD 10:POKE #302,INT(SS/10.0)
60 POKE #303,MM MOD 10:POKE #304,INT(MM/10.0)
65 POKE #305,HH MOD 10:POKE #306,INT(HH/10.0)
70 REM "#300-#306:COMTEURS"
75 REM " CALLM#"
80 REM "30D:AFFICHAGE;34B:COMPTEUR*****"
85 REM "3A1:MARCHE ; 3B8:ARRET"
90 REM "3E6: AFF ON ;3EF: AFF OFF"
95 INPUT "MOMENT DU DECLICHEMENT : HH,MM";HH,MM:PRINT
100 D(3.0)=INT(HH/10.0):D(2.0)=HH MOD 10.0
105 D(1.0)=INT(MM/10.0):D(0.0)=MM MOD 10.0
106 INPUT " MOMENT DE FIN D'ENREGISTREMENT : HH,MM";HH1,MM1:PRINT
115 F(3.0)=INT(HH1/10.0):F(2.0)=HH1 MOD 10.0
120 F(1.0)=INT(MM1/10.0):F(0.0)=MM1 MOD 10.0
125 PRINT CHR$(12)
130 CURSOR 6,12:PRINT "* CASSETTE PROGRAMMEUR *"
135 PRINT TAB(6);"*****"
140 CURSOR 6,13:PRINT "*****"
145 CURSOR 43,23:PRINT "TIME:"
150 CURSOR 0,1:PRINT "DECLICHEMENT a' : ";HH;".";MM
155 PRINT TAB(40);"FIN : ";HH1;".";MM1;
160 FOR I%=0 TO 3:IF PEEK(#303+I%)<>D(I%) THEN 160
165 NEXT:CURSOR 0,1:PRINT SPC(12);"ENREGISTREMENT EN COURS"
170 POKE #40,#28
175 FOR I%=0 TO 3:IF PEEK(#303+I%)<>F(I%) THEN 175
180 NEXT:POKE #40,#30:CALLM #3B8
185 PRINT CHR$(12):CURSOR 0,20:PRINT "ENREGISTREMENT EFFECTUER:"
190 PRINT " A P U Y E R S U R L E S T O P D U M A G N E T O"
195 PRINT " S U R L E P O W E R D E L A C H A I N E"
200 PRINT :PRINT " eteindre la TV et l'ORDINATEUR(interrupteur derriere)"
202 ENVELOPE 0 15,50;0,30;
205 COLORT 0 3 0 0
210 SOUND 0 0 15 2 FREQ(100.0):WAIT TIME 100
215 COLORT 9 14 0 0
220 SOUND 0 0 15 2 FREQ(10000.0):WAIT TIME 100:GOTO 205
999 WAIT TIME 100:END
1020 READ POINT%:POINT=POINT%
```

```
1030 PH=INT(POINT/#100):PL=POINT-PH*#100
1040 READ C%:C=C%:IF C>#2FF THEN RETURN
1050 IF C<#100 THEN POKE POINT,C:GOTO 1100
1060 IF C<#200 THEN POKE POINT,(C-#100+PL) IAND #FF:RET=(C-#100+PL) IAND #100
1070 POKE POINT,C-#200+PH+RET
1100 POINT=POINT+1.0:GOTO 1040
10000 DATA #300
10005 DATA 0,0,0,0,0,0,#C6,#30,#77,#23,#23,#C9
10010 DATA #C5,#D5,#E5,#F5,#2A,#8A,#00,#01,#88,#FF,#09
10015 DATA #3A,#101,#200,#CD,#107,#200,#3A,#102,#200
10020 DATA #CD,#107,#200,#3A,#103,#200,#23,#23,#CD,#107,#200
10025 DATA #3A,#104,#200,#CD,#107,#200,#23,#23,#3A,#105,#200
10030 DATA #CD,#107,#200,#3A,#106,#200,#CD,#107,#200
10035 DATA #C3,#F8,#03,#C1,#C9
10040 DATA #FFF,#345
10045 DATA #36,#00,#23,#34,#7E,#C9
10050 DATA #C5,#D5,#E5,#F5,#21,#00,#03
10055 DATA #34,#7E,#FE,#32,#C2,#E1,#03,#CD,#100,#200
10060 DATA #FE,#0A,#C2,#11,#03,#CD,#100,#200,#FE,#06
10065 DATA #C2,#11,#03,#CD,#100,#200,#FE,#0A,#C2,#11,#03
10070 DATA #CD,#100,#200,#FE,#06,#C2,#11,#03,#CD,#100,#200
10075 DATA #FE,#04,#CA,#147,#200,#FE,#0A,#C2,#11,#03,#CD,#100,#200
10080 DATA #C3,#11,#03
10085 DATA #23,#7E,#FE,#02,#C2,#11,#03,#36,#00,#2B,#36,#00
10090 DATA #C3,#11,#03
10095 DATA #FFF,#39B
10100 DATA #CD,#4B,#03,#C3,#A9,#B9
10105 DATA #FFF,#3A1
10110 DATA #F3,#C5,#D5,#E5,#F5,#2A,#70,#00,#22,#9F,#03
10115 DATA #21,#9B,#03,#22,#70,#00,#F1,#E1,#D1,#C1
10120 DATA #FB,#C9
10125 DATA #FFF,#3B8
10130 DATA #F3,#C5,#D5,#E5,#F5,#2A,#9F,#03,#22,#70,#00
10135 DATA #F1,#E1,#D1,#C1,#FB,#C9
10140 DATA #FFF,#3C9
10145 DATA #32,#5F,#03,#32,#67,#03,#32,#6F,#03,#32,#77,#03
10150 DATA #32,#84,#03,#32,#8A,#03,#32,#91,#03,#32,#99,#03
10155 DATA #F1,#E1,#D1,#C1,#C9
10160 DATA #C5,#D5,#E5,#F5,#3E,#11,#C3,#100,#200
10165 DATA #C5,#D5,#E5,#F5,#3E,#E1,#C3,#100,#200
10170 DATA #FFF,#3F8
10175 DATA #2A,#8A,#00,#01,#85,#FF,#09,#06,#FF,#0E,#2E
10180 DATA #70,#23,#23,#70,#23,#23,#70,#23,#23,#70,#23,#71
10185 DATA #23,#70,#23,#23,#70,#23,#23,#70,#23,#71,#23,#70
10190 DATA #F1,#E1,#D1,#C1,#C9
19999 DATA #FFF
65510 REM
```

Tijdberekeningen

```

10 CLEAR 5000:MODE 0:PRINT CHR$(12):COLORT 9 14 0 15:POKE #75,32
20 FOR I%=#BFFF TO #BFF3 STEP -4:POKE I%,#30:NEXT
30 FOR I%=#BFEF-24*#B6 TO I%-12 STEP -4:POKE I%,#3F:NEXT
40 CURSOR 6,10:PRINT ">>>>> A L L E R L E I i.v.m. T I J D <<<<<";
50 FOR I%=1 TO 11:COLORT 9 14*(I% MOD 2) 0 15:WAIT TIME 15:NEXT
60 FOR I%=#BFFF TO #BFF3 STEP -4:FOR J%=#31 TO #3F:POKE I%,J%:WAIT TIME 2:NEXT:NEXT
70 FOR I%=#BFEF TO I%-23*#B6 STEP -#B6:FOR J%=#7B TO #7F:POKE I%,J%:WAIT TIME 2:NEXT:NEXT
80 CURSOR 10,8:PRINT "1. Kalender van een jaar";
90 CURSOR 10,6:PRINT "2. Bepalen welke dag een bepaalde datum was ";
120 CURSOR 5,0:PRINT "door Frank De Jonghe          Nummer ?";
130 FOR I%=#BFFF TO #BFF3 STEP -4
140 FOR J%=#3E TO #30 STEP -1:POKE I%,J%:WAIT TIME 3:NEXT:NEXT
150 FOR I%=#BFEF TO I%-8*#B6 STEP -#B6
160 FOR J%=#7E TO #70 STEP -1:POKE I%,J%:WAIT TIME 3:NEXT:NEXT
200 A%=GETC:IF A%<49 OR A%>50 THEN 200
210 DIM DAMAX(12,0),MAAND$(12,0)
215 FOR I=1.0 TO 12.0:READ DAMAX(I):NEXT I
220 FOR I=1.0 TO 12.0:READ MAAND$(I):NEXT I
230 DATA 31,28,31,30,31,30,31,31,30,31,30,31
240 DATA Januari,Februari,Maart,April,Mei,Juni,Juli,Augustus,September,Oktober,November,December
250 ON A%-48 GOTO 300,1500
300 REM *** Kalender maken ***
310 MODE 0:COLORT 12 0 12 12
320 PRINT CHR$(12)
330 PRINT TAB(20);"Kalender"
340 PRINT TAB(20);"[ I I I ]"
350 PRINT :PRINT
360 INPUT "Jaartal (niet afkorten) ";JAX:PRINT
370 D=1.0:M=1.0:J=JAX:GOSUB 2000
420 IF JAX/400.0=INT(JAX/400.0) THEN DAMAX(2,0)=29:GOTO 450
430 IF JAX/100.0=INT(JAX/100.0) THEN DAMAX(2,0)=28:GOTO 450
440 IF JAX/4.0=INT(JAX/4.0) THEN DAMAX(2,0)=29:GOTO 450
445 DAMAX(2,0)=28
450 DIM KAL%(12,0,7,0,6,0)
455 DX=DX-1
460 FOR MAX=1 TO 12
465 RX=1
470 FOR DAX=1 TO DAMAX(MAX)
480 DX=DX+1:IF DX>7 THEN DX=1:RX=RX+1
490 KAL%(MAX,DX,RX)=DAX
500 NEXT DAX:NEXT MAX
510 PRINT CHR$(12)
520 CURSOR 10,15:PRINT "Afdruk op printer (J/N) ?"
530 A%=GETC:IF A%<>74 AND A%<>78 THEN 530
560 IF A%=74 THEN 750
570 POKE #131,1:LIJN$="~~~~~"
580 FOR I%=1 TO 12
590 PRINT CHR$(12)
600 PRINT TAB(15);MAAND$(I%)
610 PRINT TAB(15);LEFT$(LIJN$,LEN(MAAND$(I%)))
620 PRINT :PRINT
630 PRINT TAB(7);"Zo";TAB(14);"M";TAB(21);"D";TAB(28);"W";
640 PRINT TAB(35);"D";TAB(42);"V";TAB(49);"Z"
650 FOR J%=1 TO 6
660 FOR K%=1 TO 7
670 IF KAL%(I%,K%,J%)=0.0 THEN PRINT TAB(K%*7-1);"":GOTO 690
680 PRINT TAB(K%*7-1);KAL%(I%,K%,J%);
690 NEXT K%

```



```

700 PRINT
710 NEXT J%
720 IF GETC<>32.0 THEN 720
730 NEXT I%
740 GOTO 1100
750 POKE #131,1:LIJN$="~~~~~"
760 PRINT CHR$(12)
770 PRINT "Hoeveel afdrucken wil u (<=20) ?"
780 INPUT "--> ";AF%:PRINT
790 IF AF%<1.0 OR AF%>20.0 THEN PRINT "Een beetje ernst A.U.B.":GOTO 770
800 POKE #131,0
810 FOR H%=1 TO AF%
812 AFDRUKDAG$="Zo M D W D V Z"
815 PRINT :PRINT :PRINT :PRINT
820 A$=SPC(15)
830 HH$=A$+"* K A L E N D E R van "+MID$(STR$(JAX),1,LEN(STR$(JAX))-3.0)+"*"
835 FOR HP=1.0 TO LEN(HH$)-15.0:H$=H$+"#":NEXT
836 H$=A$+H$:PRINT H$
840 PRINT HH$
870 PRINT H$
880 FOR HP=1.0 TO 5.0:PRINT :NEXT
890 FOR I%=1 TO 12
900 A$=SPC(10)
910 H$=A$+MAAND$(I%)
920 PRINT H$
923 PRINT
925 PRINT AFDRUKDAG$
930 PRINT :PRINT
940 FOR J%=1 TO 6
945 R$=""
950 FOR K%=1 TO 7
960 IF KAL%(I%,K%,J%)=0.0 THEN R$=R$+SPC(8):GOTO 1000
970 HU$=STR$(KAL%(I%,K%,J%))
980 HU$=MID$(HU$,1,LEN(HU$)-3)
990 R$=R$+HU$+SPC(8-LEN(HU$))
1000 NEXT K%
1010 PRINT R$
1020 NEXT J%
1030 PRINT
1035 NEXT I%
1036 PRINT :PRINT :PRINT
1037 A$=SPC(20)+"U aangeboden door Frank De Jonghe"
1038 PRINT A$
1039 PRINT :PRINT :PRINT :PRINT :PRINT
1040 POKE #131,1
1050 PRINT CHR$(12):PRINT "Zet het papier goed."
1060 PRINT "Druk <RETURN> ":INPUT RET$:PRINT
1070 POKE #131,0
1075 H$=""
1080 NEXT H%
1090 POKE #131,1
1100 PRINT CHR$(12)
1110 PRINT "Druk 'K' voor nog een kalender,"
1120 PRINT "en 'M' om naar het menu terug te keren."
1130 A%=GETC:IF A%<>77 AND A%<>75 THEN 1130
1140 IF A%=77 THEN 10
1150 GOTO 300
1500 REM *** Dagbepaling ***
1510 MODE 0:COLORT 0 10 0 0
1520 PRINT CHR$(12)
1530 PRINT TAB(20);"Dagbepaling"
1540 PRINT TAB(20);"#####"
1550 PRINT :PRINT
1560 INPUT "Jaar (niet afkorten) ";J:PRINT

```

```

1570 IF J<=0.0 THEN 1560
1580 INPUT "Maandnummer ";M:PRINT
1590 IF M<1.0 OR M>12.0 THEN 1580
1600 INPUT "Dagnummer ";D:PRINT
1610 IF J/4.0=INT(J/4.0) AND M=2.0 AND D=29.0 THEN 1630
1620 IF D<=0.0 OR D>DAMAX(M) THEN 1600
1630 GOSUB 2000
1640 PRINT
1650 PRINT "Deze dag was/is/zal zijn een ";D$
1660 CURSOR 1,3:PRINT "Druk 'M' voor terugkeer naar het menu."
1670 CURSOR 1,2:PRINT "en 'D' voor een andere dagbepaling."
1680 AX=GETC:IF AX<>77 AND AX<>68 THEN 1680
1690 IF AX=77 THEN 10
1700 GOTO 1500
2000 REM *** Subroutine voor dagbepaling ***
2010 REM D=dag/M=maand/J=jaar
2020 F=J+D+3.0*M-3.0
2030 X=INT(SGN(M-3.0)/2.0)
2040 J=J+X
2050 F=F-(X+1.0)*INT(0.4*M+2.3)
2060 F=F+INT(J/4.0)-INT(3.0*INT(J/100.0)/4.0)-1.0
2070 F=F-7.0*INT(F/7.0)
2080 ON F+1 GOTO 2090,2100,2110,2120,2130,2140,2150
2090 D$="Zaterdag":D%=7:RETURN
2100 D$="Zondag":D%=1:RETURN
2110 D$="Maandag":D%=2:RETURN
2120 D$="Dinsdag":D%=3:RETURN
2130 D$="Woensdag":D%=4:RETURN
2140 D$="Donderdag":D%=5:RETURN
2150 D$="Vrijdag":D%=6:RETURN

```

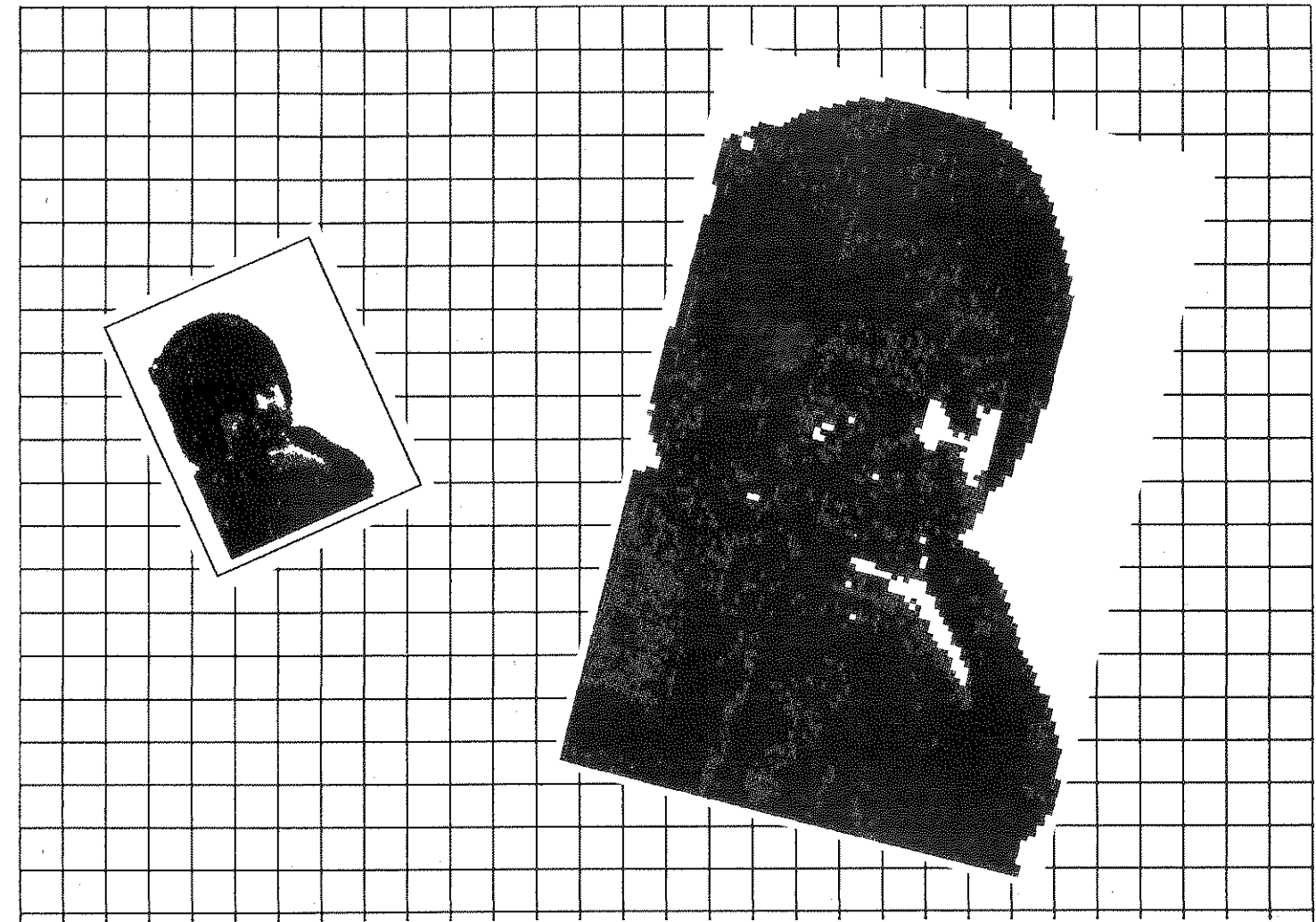
Selfdestroy

```

10 REM .....@.BY ROLF SCHALL (Wem sonst).....
20 REM ..... SELBSTVERNICHTUNGS-PROGRAMM .....
30 REM ..... MIT ***** GARANTIE ***** .....
50 MODE 0:PRINT CHR$(12):COLORT 5 15 5 5
60 FOR I!=0.0 TO 10.0:POKE #BFEF-134*I!,#5A:NEXT
70 PRINT :PRINT "PROGRAM DESTROYED":PRINT
75 PRINT " IN";" SECONDS":FOR I=3.0 TO 0.0 STEP -1.0:CURSOR 7,19:
PRINT I:WAIT TIME 50:NEXT:PRINT
80 COLORT 5 15 0 0:PRINT CHR$(12);:WAIT TIME 2:COLORT 8 0 0 0:PRINT "BASIC
V1.0"
100 POKE #29F,#EC:POKE #2A0,3:POKE #2A1,#ED:POKE #2A2,#3:POKE #2A3,#EE:POKE
#2A4,#3

150 REM
160 REM ..... GEHEIME BOTSCHAFT .....
170 REM
200 REM HABEN SIE'S ENDLICH GEMERKT, SIE BLINDGANGER ?!
210 REM ..... ENDE DES PROGRAMMS .....

```

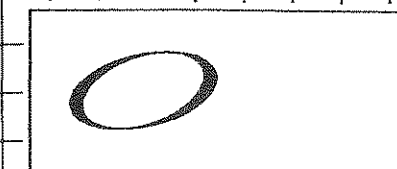


PRELIMINARY COLOUR MIXING EXPERIMENTS ON THE
'INTEGREX COLOURJET 132' PRINTER AND DAI PERSONAL COMPUTER
Louis Gidney, July 1984, Tel: 032872-248

DAI PAL RANGE FOR GENERAL PURPOSE SCREEN DUMPS
(faster than the "Quality" set)

(7 = white 6 = cyan 5 = magenta 3 = yellow 0 = black
Example:- Each dot of patch '655333' is a mixture of: 1 part cyan, 2 pts magenta and 3 pts yellow)

777700 777665 777655 777553 777750 776633 777530 776533 777770 777766 777533 777775 777776 777763 7777



High Speed Loader

The High Speed Loader was developed a few years ago by Hein Kop and myself, to make it possible to load often used programs in a fast way by means of reading from an Eeprom bank, which was connected to the DCE bus. The system was built around an Eeprom card which was available at Elektuur Printservice.

It could contain four Eeproms of the type 2716 (4x2kByte) or 2732 (4x4kByte).

This Eepromprint was placed on a basic printed circuit board with an 32 pins connector which contained the rest of the electronics.

The result was that we could read a program of 10kByte in one second.

For programs like SPL an ideal solution.

Also several games were stored in Eeprom and children loved it.

An Eeprom card with the game was plugged in, and one command was enough to load the game. Like Atari.

However due to the high cost of Eeproms at that time only a few were made and they worked very satisfactory.

Another disadvantage was that the Eeprom card was laid out only for 2716 or 2732 Eeproms.

With the development of the DAI-DOS 1541 system we decided to include the software of the HSL into the software of the 1541 system, because SPL was used during this project and available on an Eeprom bank of the HSL.

However from time to time people are asking us again about this project so we gave it a go over and got rid of several items like Power On Initialization which was no longer required because this is now on our Floppy Interface.

The final result is on a drawing together with this description.

We have not the intention to develop a basic printed circuit board or the request for it must be so demanding that we have no choice.

The HSL can be connected to the DCE bus together with an MDCR(s) or VC1541 disk drive(s).

It has its own device address selection and will not interfere with other equipment.

TECHNICAL DESCRIPTION

The device selection is done with PB4 and decoded in such away, with the use of an 74LS138, that the HSL is only selected when PB4 is high.

Only one output of the 74LS138 is used, the others can be used for addressing other devices connected to the DCE bus.

The address counter for addressing the Eeproms consist of two 74LS393 clocked with PC0 and a reset pulse PC1.

Both signals are twice inverted by an 74LS14.

This is done to reshape the clock and reset signal which are through the length of the flat cable and other devices in bad shape.

For the same reason are both C's of InF included.

In some cases it is necessary to place an 390 ohm resistor in series in both lines.

The data lines are extra buffered with an 74LS245 which

is only active during the time the HSL is selected. This way unwanted signals on the DCE bus are avoided. The selection of the Eeproms is done with a combination of an 74LS157 and an 82S23 Prom.

Due to this combination, together with PB2 and BP3, we have the possibility to split a program over more than one Eeprom and read it without interruption as a whole.

Example:

Eeprom 1 (2732) contains a program called "Spooler"

Eeprom 2,3 and 4 contains "SPL"

With the command HSL0 the "Spooler" program stored in the first Eeprom is loaded.

With the command HSL1 "SPL" is loaded, this program is stored in three Eeproms which will be selected one after the other.

When you give the command HSL2 an error "NOT AVAILABLE" will appear on the screen.

Due to the fact that a address preset is not possible, empty space within one Eeprom cannot be used for another program.

At the beginning of every new program in an Eeprom or bank five bytes are used to provide the software with the proper start information.

1st. Byte	:#FD This a not used Opcode. Indicates a NEW PROGRAMM.
2nd.+3th. Byte	:Start address of the program within the DAI memory. Lowbyte- Highbyte.
4th.+5th. Byte	:Total length of the program. Highbyte- Lowbyte

The contents of the used Prom type 82S23 is as follows:

Address: 0: 1: 2: 3: 4: 5: 6: 7: 8: 9: A: B: C: D: E: F

Data : E: D: B: 7: D: B: 7: F: B: 7: F: F: 7: F: F: F

The Eeprom print card was available at Elektuur printservice, Postbus 75, 6190 AB Beek(L) number EPS 82558-2.

The used connector is available at almost every good electronics shop.

However if you want to make your own arrangement you do not need this card and connector.

This is the way the HSL is set up.

Variations, changes and improvements are possible we realize this. So if you have a good idea let us know.

On paper we have a version which can handle Eeproms of the following types: 2716, 2732, 2764 and 27128.

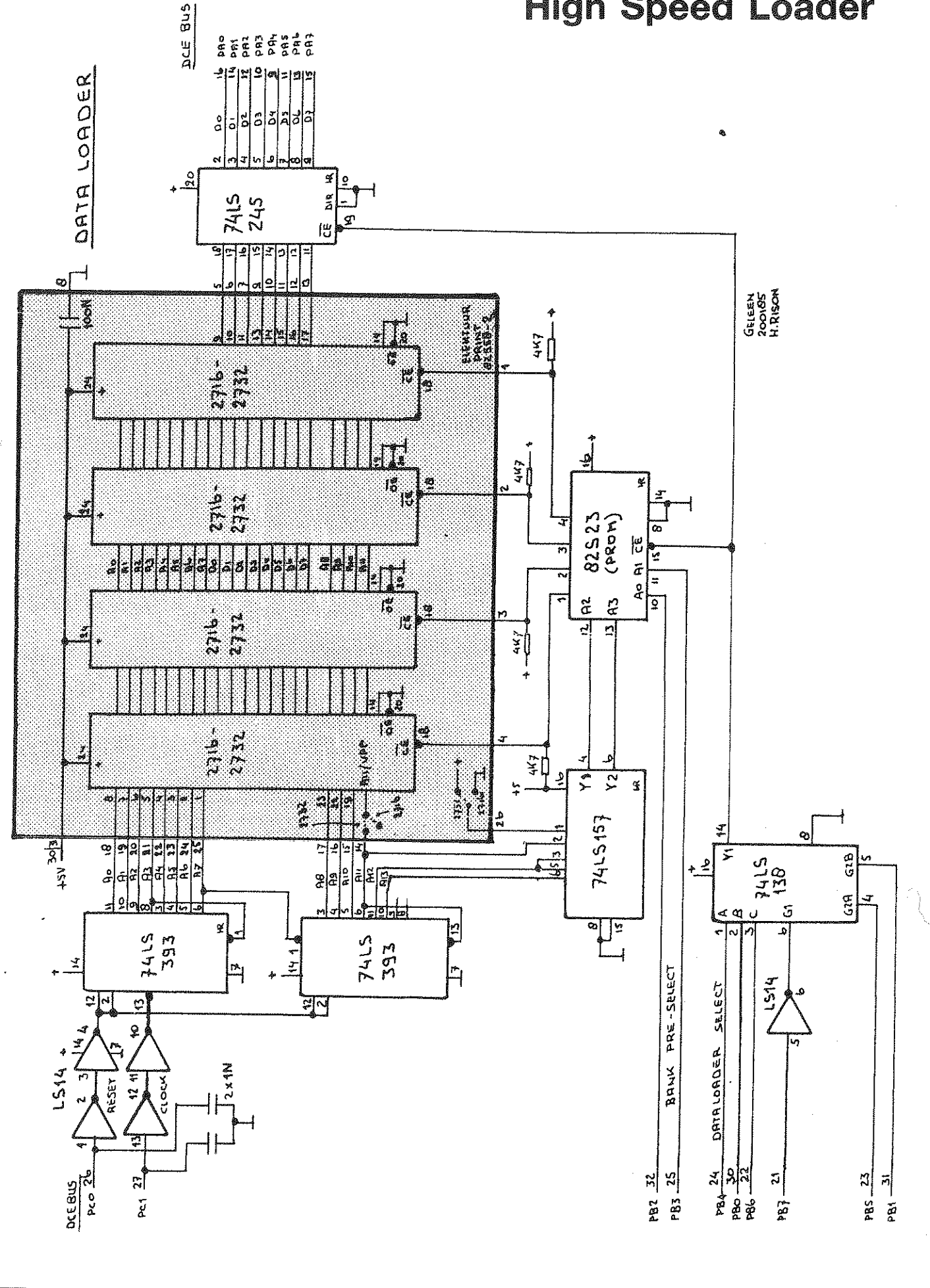
However we do not have a small Eeprom card which can contain 28 pins IC sockets with the same pin arrangement as the used connector on the Elektuur PCB.

For extension to use more than four programs on an Eeprom card a change in the DOS software must be made because this is not foreseen.

H. Rison
Luxemburgstraat 17
6164 BS Geleen NL

DAI DOS 1541 : Relative Files

High Speed Loader



This article describes the principle of relative files and the way they are handled on the DAI-DOS 1541 with the Commodore VC1541 disk drive.

PRINCIPLES/ADVANTAGE OF RELATIVE FILES:

Fast access to each record - Large files can be handled.

Relative files are so-called random access files. They are disk-oriented. This means, that the contents of the file is not loaded into the RAM space of the computer, but remains on the diskette. Only the parts needed to be handled are read from the diskette into the computer memory.

The advantage of this system of file handling is, that very large files can be handled. The sequential files - for the DAI this means: the arrays - can be max. 32KB, due to the maximum heap space in the DAI. String arrays can be max. about 20KB, due to the double RAM space required for writing and reading. Relative files on the diskette can be max. 164KB, limited only by the max. storage space available on a diskette.

A relative file consists of a number of records. A record is a block of information. Let us take as example a mailing list. For everyone on that list, we need to store the name, the address and the town. These 3 items ('FIELDS') together form a 'RECORD'. The records of all people in this mailing list together, are called a 'FILE'.

Another advantage is the manner in which updates are made. A relative file is random-access, which means that each record of the file can be addressed without loading the whole file in the computer memory. To update a sequential file (array), the whole file must be read, updates can be made, and the file has to be saved again. For relative files, only the record to be updated has to be loaded into the computer and can be saved afterwards. This saves a lot of time.

To handle relative files on the DAI-DOS1541, the additional monitorprogram 'FDDMON/O' is required (see below). This small object code program allows the use all additional options of the VC1541 disk drive.

FILE/RECORD STRUCTURE:

Relative files must have records of a fixed length. This is dictated by the way they are handled by the VC1541 operating system. In addition, only character strings are allowed as records. The maximum record length is 254 characters.

The maximum number of records is 65335 (#FFFF), which means the practical number of useable records is limited by the capacity of the diskette.

In our example of the mailing list, let us take:

```

name      20 characters
address   20 characters
town      15 characters

```

The total length of a record is now 20+20+15=55 characters.

The record length must be determined in advance. Once fixed, it can not be changed anymore! It is used by the disk drive for the file organisation on the diskette.

To enable the handling of each single field, we must take care, that each field has exactly these number of characters. If e.g. a name is only 14 characters long, 6 additional spaces have to be added to the name. Later, we will see how we form the fields together to a record, and how a single field can be separated from a record.

Furthermore, the file must have a file name of maximum 16 characters.

COMMUNICATION DAI - VC1541:

The relative files can only be handles in combination with the monitor program FDDMON/O. This program must loaded together with your BASIC program. If the name of the BASIC program is 'MAILING LIST/B', then load both parts with:

```
BOOT"FDDMON/O,MAILING LIST/B"
```

By means of the object code routines in the FDDMON/O program the whole communication is handled. The easiest way is to declare (INT) variables in your BASIC program with the addresses of the various routines, e.g.:

```

OPEN3=#400      PRNT3=#406      ERROR=#F2E8
CLOSE3=#421     INF3=#418       etc.

```

See the manual of the 'FDD-TOOLKIT/1' (see below) for all addresses.

In deviation of the normal save and load procedures on the DAI-DOS 1541, for relative files the opening and closing of the file must be done by the user. For all communication for relative files, the channel 3 of the VC1541 is used, in combination with channel 15 (command/error channel) for system instructions.

Also the error channel is not read automatically.

Only one relative file can be handled at the same time. If you want to handle data from two different relative files, always close the first one before opening the second file, otherwise an error message 'NO CHANNEL' will occur.

OPEN/SET-UP/CLOSE A RELATIVE FILE:

The format to OPEN a relative file is the following:

```

10  FILENAME$="MAILING LIST"
20  RECLENGTH=56      (1 more than the real length due
                      to the CR send after the string)
30  OPEN#=FILENAME$+",L,"+CHR$(RECLENGTH)
40  CALLM OPEN3,OPEN#

```

When the relative file is opened the first time, it is advised to reserve already a number of records. This is not a must, but it increases the speed of the file handling. Reserving a number of records means, that on the diskette a number of 'empty' records are written. If e.g. you think that you will need 200 records, it is sufficient to write data to the 200th record. All records 1-199 will then be initialised too. Later writing into the records 1-199 will now be done much faster. This principle can be repeated always. If e.g. the 200 records are used, you can write into the 400th record, and now the records 201-399 are set-up too.

To give free this last record, it is sufficient to write #FF - CHR\$(255) - into it.

At first, the VC1541 must be POSITIONED on this record. This is done as follows:

```

50  RECNR=200      (as example)
60  HB=RECNR SHR 8
70  LB=RECNR IAND 255
80  POS$="P"+CHR$(3)+CHR$(LB)+CHR$(HB)+CHR$(0)
90  CALLM PRNT15,POS$

```

The string POS\$ tells the VC1541 that it is a 'P'osition command on channel 3, the record number, and that it should point to the first byte (byte 0) of the record. This is done via the command channel 15.

To write #FF in this record, the following statements are sufficient:

```

100 OP#=CHR$(255)
110 CALLM PRNT3,OP#

```

The print statement in line 110 will cause an error message 'RECORD NOT PRESENT', because record 200 does not yet exist the moment #FF is written into it. This error message can be ignored.

If a large number of records is declared free, it may take some time. For 500 records of 40 characters, it lasts about 2.5 minutes.

CLOSING a relative file is done via:

```
120 CALLM ERROR: CALLM CLOSE3
```

Always read the error channel before closing the relative file. An eventual error message will be displayed on the screen.

WRITE DATA TO RELATIVE FILE:

Once the relative file is set-up like described above, data can be entered. In our example, we have 3 variables: NAM\$, ADDR\$ and TOWN\$, each with a defined length. These lengths are stored in the variables LN, LA and LT:

```

name      :  NAM$      LN=20
address:  ADDR$      LA=20
town      :  TOWN$     LT=15

```

A small program will describe in steps how data is send to the file (use of FDDMON/O is required !!).

```
10 INPUT NAM$: PRINT
20 IF LEN(NAM$) >LN THEN 10
30 NAM$=NAM$+SPC(LN-LEN(NAM$))
40 INPUT ADDR$: PRINT
50 IF LEN(ADDR$)>LA THEN 40
60 ADDR$=ADDR$+SPC(LA-LEN(ADDR$))
70 INPUT TOWN$: PRINT
80 IF LEN(TOWN$)>LT THEN 70
90 TOWN$=TOWN$+SPC(LT-LEN(TOWN$))
```

This routine enters the data for each field of the record. Too long fields are not accepted. Too short fields are made exactly the correct length by adding additional spaces.

The number of the record, to which the data must be send, has to be entered now:

```
100 INPUT RECNR: PRINT
110 HB=RECNR SHR 8: LB=RECNR IAND 255
```

Now the relative file can be opened:

```
120 FILENAME$="MAILING LIST": RECLENGTH=56
130 OPEN$=FILENAME$+",L,"+CHR$(RECLENGTH)
140 CALLM OPEN3,OPEN$
```

After opening the file, the pointer must be set to the correct record:

```
150 POS$="F"+CHR$(3)+CHR$(LB)+CHR$(HB)+CHR$(0)
160 CALLM PRNT15,POS$
```

Now the data can be send to the specified record:

```
170 OP$=NAM$+ADDR$+TOWN$
180 CALLM FRNT3,OP$
```

This input procedure can be repeated for other records, e.g. via a loop. When ready, the relative file must be closed:

```
190 CALLM ERROR: CALLM CLOSE3
```

READ DATA FROM RELATIVE FILE:

Data can be read from a relative file in two ways: character by character (GET) or as a string (INPUT). The GET-method is useful if only one character or a part of a record is required. If the whole record must be read, the INPUT-method must be used.

In the example, both methods will be used. The GET-method to see if the record is empty, the INPUT-method to read the record.

The handling is a little bit 'artificial', due to the way the link between the object code program FDDMON/O and the BASIC. Two variables have to be declared at the start of the program:

```
1 IN$=""
2 IP$="": FOR I=0 TO 254: IP$=IP$+" ": NEXT
```

The variable IN\$ is used for temporary storage of the character read by the GET-command. In the 255 spaces long variable IP\$, the characters read by the INPUT-command are stored.

The relative file is opened in the same way as done for the write function:

```
10 CALLM OPEN3,OPEN$ ) see 'write data',
20 CALLM PRNT15,POS$ ) lines 100-160
30 CALLM GET3,IN$
40 IF ASC(IN$)=255 THEN 100
```

If the first character of the record is #FF, then the record is empty.

```
50 CALLM FRNT15,POS$
60 CALLM INP3,IP$
```

This second position command is required to point again to the start of the record. Now the whole record is stored in IP\$, and must be divided into 3 parts: the name, the address and the town:

```
70 NAM$ =MID$(IP$,0,LN)
80 ADDR$=MID$(IP$,LN,LA)
90 TOWN$=MID$(IP$,LN+LA,LT)
```

The various strings can be updated and written back again as described above.

Do not forget to close the file, once all reading and writing is finished:

```
100 CALLM ERROR: CALLM CLOSE3
```

REFERENCES:

For a detailed explanation, with a lot of examples, is referred to 'Das grosse Floppy-Buch' by Englisch/Szczepanowski, published by Data Becker GmbH, Duesseldorf (BRD). This book is also translated in other languages.

The diskette 'FDD TOOLKIT/1' with (amongst others) the program FDDMON/O can be ordered from 'Micro Service', Fabritiusstr.15, 6174 RG Sweikhuizen, The Netherlands. It costs Dfl.60.- to be paid in advance by means of an international postal money order. Please indicate the program name on your order. Dutch members can transfer this amount on bank account 13.05.78.754 of Micro Service. This toolkit contains also other useful programs for extended use of the VC1541 disk drive.

Jan Boerrigter, June 1985

Critical Path Analysis

FWP in dubbelkoloms

DAINAMIC

Marc ANTROP - 1981

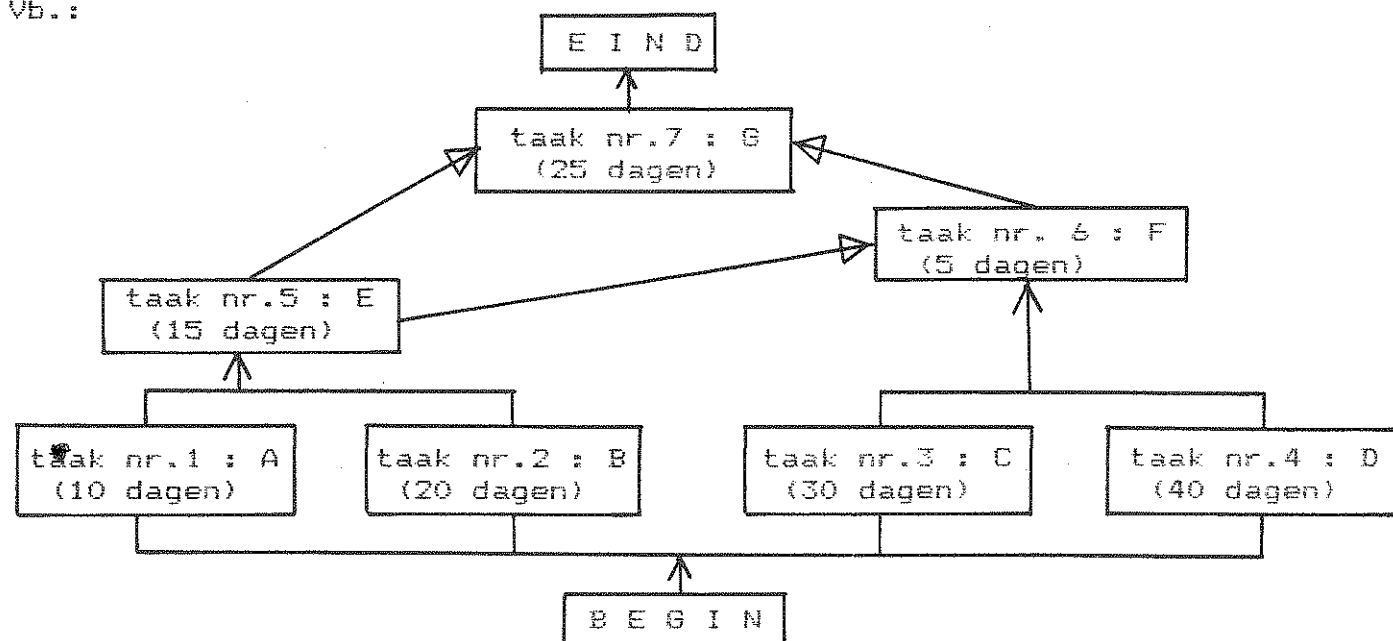
1. DOEL

Critical path analysis is een techniek om in een project, bestaande uit een complex netwerk van taken die elk een bepaalde minimale duur hebben en onderling afhankelijk zijn, die taken te detecteren die een vertraging kunnen oorzaken in de afwerking van het gehele project (de zgn. 'bottle necks'). Daarnaast wordt ook de totale minimale duur van het gehele project berekend evenals de speelruimte in tijd voor iedere taak afzonderlijk. Dit is dan zgn. 'slack time': de vertraging die een welbepaalde taak mag oplopen om de totale tijdsduur van het project niet in gevaar te brengen.

2. GEBRUIK

Eerst moet een stroomschema van de verschillende taken opgesteld worden. Dit bevat de definitie van de taken (naam of code), hun geschatte uitvoeringsduur en hun onderlinge volgorde.

Vb.:



De gegevens die moeten ingevoerd worden zijn :

- aantal taken
- naam of code van de taak
- nummer van de taak
- geschatte duur in tijdseenheden naar keuze
- voor iedere taak de taak die er aan vooraf gaat
- of er een 'BEGIN'-taak (nr.0) vooraf gaat en of de 'EIND'-taak (nr. aantal+1) volgt.

Probleem

Een van de beperkingen van FWP is het feit dat er, althans niet gemakkelijk, in twee kolommen mee is te werken.

Ik heb dit probleem zelf ook meerdere malen ontmoet en ik wilde er iets aan doen. Nu is het werken op een zestig-karakter-scherm zoals de DAI heeft het prettigst als we een tekst hebben die niet meer en ook niet tijdelijk meer dan zestig karakters op een regel heeft. Ik heb zelf gekozen voor een kolombreedte van 37 karakters en dat past ruimschoots in de zestig.

Oplossing

Ik heb me er toen toe gezet een programma te schrijven, dat ons hier kan helpen. Voorlopig nog in Basic maar waarschijnlijk binnenkort in een veel snellere en uitgebreidere versie in machinetaal.

Wat doet dit programma? Wel simpel gezegd komt het op het volgende neer: We tikken de tekst in op kolombreedte 37 (kan aangepast worden). We maken enige voorbereidingen en laten dan het programma lopen. Onze tekst staat dan 'keurig' netjes in twee kolommen.

Vorbereidingen

Willen we de regels allemaal even lang hebben moeten we dat nu doen. Tevens dienen we er voor te zorgen dat in de tekst voldoende returns staan. Als bij een paginalengte van 60 de tekst uit 234 regels bestaat zullen de laatste zes regels van de linkerkolom verdwijnen daar er geen rechterkolom bij is. De oplossing is simpel; we voegen gewoon minstens zes returns toe en het probleem is weg. We zetten de tekst voor de zekerheid wel weg op diskette of band.

Aanpassingen

Zijn we dan helemaal klaar dan kijken we op de adressen A2 A3 A4 en A5 om te weten waar de edit buffer (nu dus tekstbuffer 1) begint en eindigt. Vervolgens verplaatsen we de tekst naar een hoger adres. Bijna altijd

zal #1000 voldoende zijn maar zolang we niet boven A000 komen werkt alles uitstekend.

N.B. Houdt er rekening mee dat boven 9000 buffer 2 vernield wordt.

Voorbeeld

```
UT
DA2 A5
00A2 00 30 7A 43
```

Geeft A2 A3 iets anders dan 00 30 dan staat FWP niet op STEP 1. In dit geval kunnen we kiezen voor

```
M3000 437A 7000
```

waardoor onze tekst komt te staan van 7000 tot 837A. Noteer deze waarden om ze straks in het basicprogramma aan te kunnen brengen.

We laden het dubbelkolomprogramma in en passen, indien nodig, enige constanten aan. In bovenstaand voorbeeld moeten we de eindconstante dus veranderen in 837B. (mag meer zijn) Geef dan RUN en na enige tijd krijgen we 'STOPPED IN LINE xx'. Nu is de tekst in twee kolommen in de buffer geplaatst.

Complicaties

Grote letters in de linkerkolom en/of de rechterkolom zijn opgelost. Ik zet grote letters aan met g en uit met u waarbij die codes staan in de headings. Dit uitzetten gebeurt bij de Epson echter met 20 en dat is een code die FWP zelf gebruikt. In de heading staat dan ook en in de convert character table staat 64 wordt 20.

De laatste nabehandeling die we de tekst moeten geven is het aanpassen van het aantal spaties na een grote tekst. Als de vergrote tekst b.v. 13 karakters lang is dienen er 13-6 = 7 spaties verwijderd te worden en alles loopt naar wens.

Frank H. Druijff

Programma FWP dubbelkoloms

```
10 REM F.W.P.-DUBBELKOLOMS / F.H. DRUIJFF - 7/85
20 REM
30 REM POINTERS

40 BL=#7000;REM START ADRES VERPLAATSTE TEKST
      bij voorkeur hoger dan #4000
50 E=#88FF: REM EINDADRES (+1) VERPLAATSTE TEKST
      het mag meer zijn maar kost nodeloos extra tijd
60 D=#3000: REM BEGINADRES BUFFER 1
70 L=41: REM REGELLENTE LINKERKOLOM + TUSSENSTUK
      onbegrensd, maar is alleen zinnig als de
      regel ook echt korter is dan deze grens.

80 P=60: REM PAGINALENGTE

100 REM
110 REM START TWEDE KOLOM
120 REM
      Zoek P regels verder naar begin van de eerste
      rechterkolom idem voor voldoende pagina's.

130 R=0;BR=BL-1
140 BR=BR+1:IF PEEK(BR)<>13 GOTO 140;R=R+1:IF R<P GOTO 140;BR=BR+1
150 REM BL= START LINKERKOLOM & BR= START RECHTERKOLOM
160 REM
200 REM LINKERKOLOM
      Verplaats een regel linkerkolom.
      Bij einde regel de rechterregel gaan verplaatsen.
      Stop bij einde tekst.

210 BC=PEEK(BL):BL=BL+1:IF BC=13 GOTO 230
220 POKE D,BC:D=D+1:A=A+1:IF BL<=E GOTO 210:STOP
230 IF PEEK(BR)<>13 THEN IF A<L THEN FOR I=A TO L:POKE D,32:D=D+1:NEXT
250 REM RECHTERKOLOM
      Verplaats een regel rechterkolom.
      Stop bij einde tekst.

260 BC=PEEK(BR):BR=BR+1:IF BC=13 GOTO 280
270 POKE D,BC:D=D+1:IF BR<=E GOTO 260:STOP
280 POKE D,13:D=D+1:A=0
290 Q=Q+1:IF Q<P GOTO 210
      Pas pointers aan en zoek nieuwe start rechterkolom
      als einde pagina (P regels) nog niet werd bereikt.

300 Q=0:BL=BR:GOTO 100
```

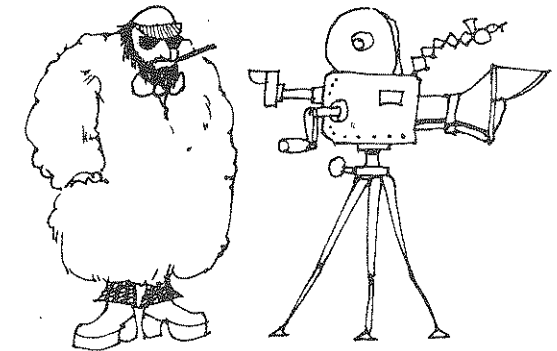
VIDEO REPORTER

Gilbert PONS

3, rue Mozart

34500 BEZIERS

Tél. 67.30.79.69



Bonjour,

C'est dans le but d'établir avec les adhérents de votre club une collaboration «artistique» que je m'adresse à vous aujourd'hui.

Je recherche des programmeurs intéressés par la réalisation d'un «dessin animé en 16 ou 35 mm» sur micro-ordinateur.

Le court métrage sera présenté aux différents festivals.

Un contrat au millième liera chaque créateur.

2 SOLUTIONS

1 Thème/Club avec la réalisation d'une séquence 40 s.

1 Thème/Créateur 2 à 5 s.

THEME : «HYTEC ou le Tunnel sous la Manche»

HYTEC, en collaboration avec la COMEX, va réaliser prochainement un relevé topographique des fonds marins de la Manche, en vue d'une expertise d'évaluation des coûts d'une potentielle édification d'ouvrages d'ART immergés pour relier le Royaume-Uni à la France.

Ci-dessous une perspective du module HYTEC version chenille. (Il est possible de remplacer les chenilles par deux vis sans fin, ou par trois tuyères propultrices : 2 horizontales, 1 verticale).

Le module HYTEC est équipé de deux caméras (une fixe N/B, une mobile couleur), trois projecteur (2 fixes, 1 mobile).

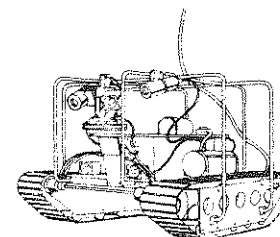
En outre, j'offre la possibilité à vos adhérents de devenir correspondants VIDEO REPORTER et de bénéficier à ce titre du transfert gratuit (super 8 - vidéo / photos vidéo / diapos vidéo) avec à leur charge les frais de transport et de cassettes (VHS ou Beta).

(Vidéo Reporter est une entreprise de Reportages Vidéo spécialisée sur le mariage, spectacles, etc...)

Dans la mesure de mes connaissances informatiques, j'assurerai une assistance à la programmation pour l'accès au langage machine spécialisée au graphisme.

Dans l'attente de vous lire, et de vous compter parmi nos futurs correspondants, soyez assuré de mes sentiments distingués.

HYTEX



Uniprom-1

Nun gibt es fuer alle DAI-Fans die Software zu dem universellen EPROM-Programmieraerat UNIPROM-1. Interessierte Leser seien auf den Artikel 'Davor ist kein EPROM sicher' im mc-Heft 8/1984 hingewiesen.

Das Steuerprogramm wurde vollstaendig in Assembler geschrieben und befaest 2,5 KBytes. Um bei langwieriger Datenarbeit im Monitor einen klaren Blick zu behalten, wird der Schriftmodus mit 44 Zeichen pro Zeile erzeugt.

Im DRIVER stehen folgende Unterprogramme zur Verfuegung :

- 1) EPROM-Auswahl
- 2) Loeschtest
- 3) Lesen des EPROM-Inhalts
- 4) MONITOR-Aufruf
- 5) Programmieren
- 6) Ueberpruefen

Folgende EPROM-Typen sind 'direkt' programmierbar ('direkt' deshalb, da praktisch auch andere EPROM-Typen vom Programm aus bedient werden koennen) :

2508, 2516, 2532, 2564
2758A(B), 2716, 2732(A), 2764, 27128
sowie alle C-Typen der 27xx-Serie

Der UNIPROM-1 MONITOR U1.5 verfuegt ueber 5 Befehle :

- | | |
|-------------------|---|
| 1) C adr1-adrh | berechnet die Checksumme im Speicherbereich adr1 bis adrh |
| 2) D adr1-adrh | displayed Bereich adr1 bis adrh und berechnet wiederum von jeder Zeile die entsprechende Checksumme |
| 3) F adr1-adrh xy | gleiche Funktion wie in Utility |
| 4) P adr1-adrh xy | printed alle Adressen im Bereich adr1 bis adrh mit Inhalt xy |
| 5) S adr | gleiche Funktion wie in Utility |
- B, Pfeil oben Zurueck zum Driver, Clear-Screen

Das Programm laesst nur moegliche Eingaben zu und ignoriert alle Eingaben, welche zum Absturz fuehren koennten. Datenzugriff ist nur im Bereich 0000h+Offset bis max. 7FFFh+Offset erlaubt, der Offset betraegt 1000h.

Das Programm kann auf Audio-Cassette bezogen werden bei :
Andreas J. Bathe, Helmstrasse 8 in 1000 Berlin 62
Es kostet DM 35,- zuzueglich Versandkosten.

Uniprom-1 Hardware Info

Um die Hardware aufzubauen, muss man auch nicht tief in die Tasche greifen: alle ICs kosten weniger als DM 30,- :

- 1 x 4021, 4 x 4094
- 1 x 7406, 1 x 555
- 8 x Transistoren
- 9 x Dioden (2 LEDs)
- 4 x Kondensatoren
- 23 x Widerstaende

Die Verbindung DAI - UNIPROM-1 wird ueber den DCE-Bus realisiert. Folgende Pinbelegungen ergeben sich dabei :

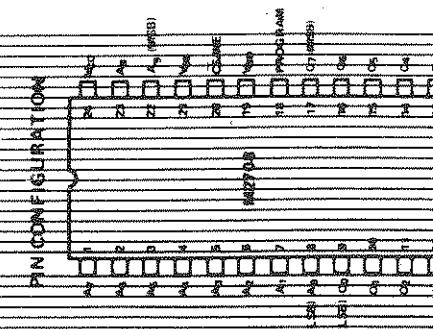
- +5V - Pin 1
- GND - Pin 4
- IN - Pin 26 (P2B0)
- Ck - Pin 27 (P2B1)
- S - Pin 28 (P2B2)
- OUT - Pin 20 (P2B4)

Die Platine (gebohrt, DM 19,80) kann beim Autor bestellt werden; ebenfalls alle notwendigen Bauteile.

Der UNIPROM-1 zeichnet sich durch seine Flexibilitaet und seiner Preisguenstigkeit aus. Die 50 ms-langen Programmierimpulse und die 5 Schieberegister sind aber Grund eines Nachteils: die Geschwindigkeit: um den 27128 (16K) beispielsweise zu lesen o. ae. werden schon 120 Sekunden benoetigt, das Programmieren verlangt allerdings schon fast 16 Minuten.

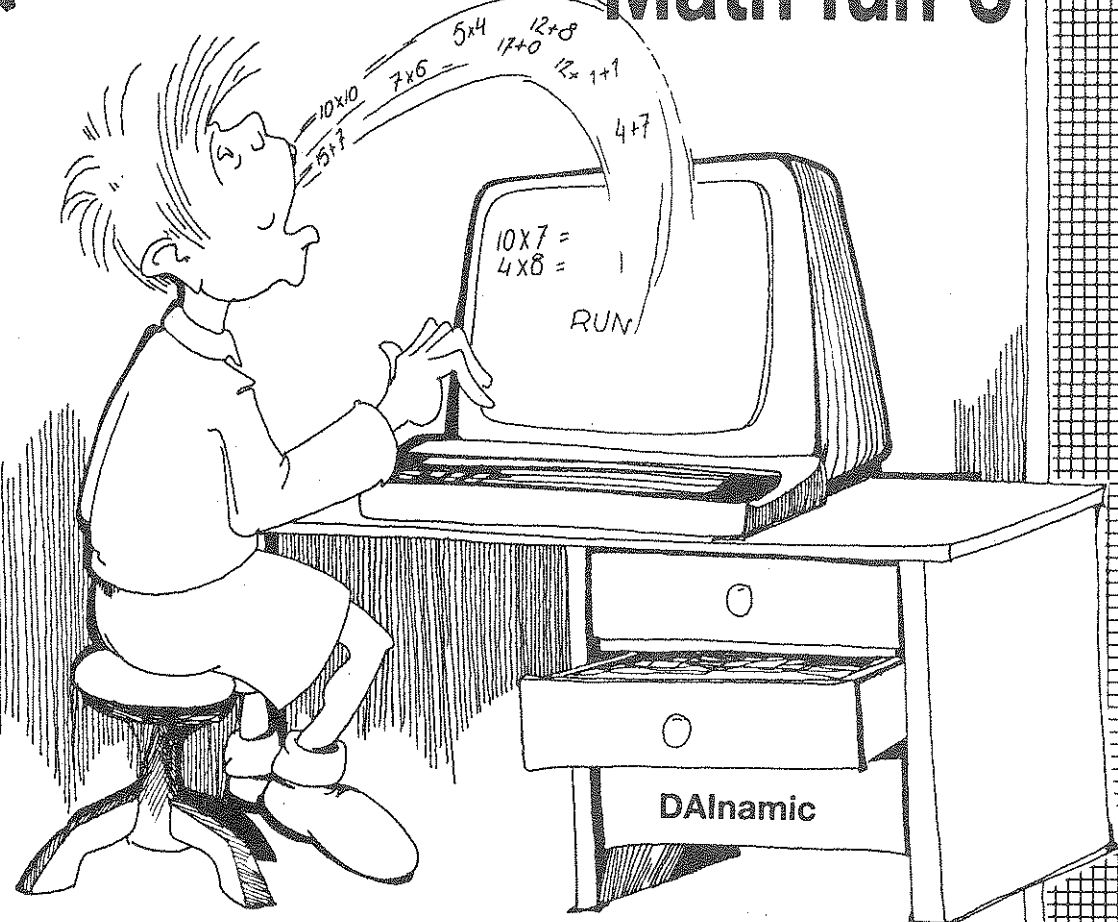
Andreas J. Bathe

The program is delivered with English documentation.



New Software

Math'fun 3



- Potjes
- Herkennen
- Brug over 10
- Groter-Kleiner

Games collection 15

Nim
 Viergewinnt
 Memory
 Cubit

Games collection 16

Hase & Igel
 zeeslag
 Satellit
 Missile Command

Toolkit 7

MSX - EXTENSIONS
 NEW ZOOM
 VARIABLE PRECISION
 PROGRAMMOTEEK
 DIGITIZER SOFT for :
 - P. DE LAET project
 - commercial unit
 FWP DUBBELKOLOMS

HAAS EN EGEL

Een wedren voor verstandige spelers. En zoals de eeuwenoude fabel reeds aantoonde, niet alleen de rapste kan winnen ook de trage als hij het maar slim genoeg weet aan te pakken.

In het spel leveren wortelen de nodige energie om vooruit te komen. (elke zet die men doet vraagt een aantal wortelen) Bij de start van het spel krijgt iedere speler (minimum 2 spelers - maximum 6) een bepaald aantal wortelen en 3 kroppen sla. Met het aantal wortelen komt men echter niet toe om het einddoel te bereiken. Er bestaan echter verschillende mogelijk heden om aan wortelen te geraken. Het doel van het spel is om als eerste de eindmeet te halen zonder salade en met slechts een beperkt aantal wortelen. Dus denk aan de fabel en verzin voor je speelt.

MISSILE COMMAND

Je opdracht bestaat erin om de stad te verdedigen tegen aanvallers uit de ruimte. De besturing van het luchtafweervizier gebeurt door middel van de paddels. Je moet ervoor zorgen dat het vizier exact gericht wordt op de aanvallende ufo. Door het drukken op de event-knop wordt voor een ontploffing gezorgd die de ufo vernietigd.

NEEM EN WIN

Dit spel wordt gespeeld tegen de computer. Er worden een aantal groepjes lucifers op het scherm getekend. De bedoeling is dat je om beurt een bepaald aantal lucifers van een groepje wegneemt. Wie de laatste lucifer kan nemen wint. Vooreerst kan je bepalen of je zelf of de computer begint. Met de cursortoetsen kan een van de groepjes geselecteerd worden. Uit dit groepje moet minimum een lucifer genomen worden (druk 1/2/3/4/5/6).

Wie verslaat de computer ??

Demo Turtle-Basic

```

1  REM
2  REM DEMO TURTLE-BASIC
3  REM
4  REM Olivier PATTINIEZ
5  REM
6  REM
10  COLORG 0 3 12 14:TORTUE:MODE 5
20  CACHE
30  COULEUR PEN 12
40  DEPLACE 210.0,YMAX/2.0
100  FOR I%=0 TO 90:AVANCE 10.0
105  DROITE 20.0:RECULE 12.0:GAUCHE 24.0
110  NEXT:DEPLACE 221.0,YMAX/2.0
115  COULEUR PEN 3:TOURNE 90.0
120  AVANCE 55.0:TOURNE 180.0
125  AVANCE 144.0
130  TOURNE 270.0:AVANCE 145.0
135  TOURNE 0.0:AVANCE 144.0:TOURNE 90.0:AVANCE 90.0
140  DEPLACE 165.0,125.0:GOSUB 200
145  DEPLACE 115.0,125.0:GOSUB 200
150  COULEUR PEN 12:DEPLACE 52.0,130.0
155  X%=34:Y%=20:GOSUB 210
160  DEPLACE 140.0,90.0:X%=7:Y%=30:GOSUB 210:X%=260:Y%=200
165  GOSUB 230:COULEUR PEN 8
170  X%=260:Y%=50:GOSUB 230
175  WAIT TIME 100:RECOMMENCE:CACHE:COULEUR PEN 8
180  X%=40:Y%=120:GOSUB 260
185  WAIT TIME 100:RECOMMENCE:CACHE
190  COULEUR PEN 7
195  X%=160:Y%=100:GOSUB 240
199  GOTO 199
200  FOR I%=0 TO 30:AVANCE 2.0:DROITE 12.0
205  NEXT:RETURN
210  FOR I%=0 TO 19:AVANCE X%:DROITE Y%
215  COULEUR PEN (I% MOD 14)+1
220  NEXT:RETURN
230  FOR I%=0 TO 70:DEPLACE X%,Y%
235  DROITE I%+10.0:AVANCE 20.0
237  GAUCHE 30.0:AVANCE 10.0:NEXT:RETURN
240  FOR I%=0 TO 480 STEP 3:DEPLACE X%,Y%:TOURNE I%+12.0
245  COULEUR PEN 8:AVANCE 35.0:GAUCHE 60.0
247  COULEUR PEN 10:AVANCE 30.0:GAUCHE 60.0
250  COULEUR PEN 9:AVANCE 30.0
255  NEXT:RETURN
260  FOR I%=0 TO 890 STEP 15:DEPLACE X%,Y%:X%=X%+5
270  TOURNE I%:AVANCE 40.0:GAUCHE 120.0
280  AVANCE 40.0:GAUCHE 120.0
290  AVANCE 40.0:NEXT:RETURN

```

Disable Break

PAGE 01 DISABLE BREAK

```

001  *
003  *
004  *****
005  * WILLI HERRMANN                MAI 1985 *
006  * RICHARD-DEHMEL-STR.4          *
007  * D-4320 HATTINGEN              *
008  *                                *
009  * Dieses Maschinenprogramm schaltet innerhalb von *
010  * laufenden BASIC-Programmen die BREAK-Taste ab. *
011  * Auch bei Eingaben mit INPUT oder GETC ist die *
012  * BREAK-Taste abgeschaltet.          *
013  * Wenn das BASIC-Programm durch END beendet wird, *
014  * braucht das Maschinenprogramm nicht gestoppt *
015  * werden; die BREAK-Taste funktioniert wieder !! *
016  *****
017  *
018  *           ORG      :300      BELIEBIG
019  *
020  * Diesen Programmteil nur einmal mit
021  * CALLM INIT starten.
022  *
023  0300 F3      INIT      DI
024  0301 E5                PUSH  H
025  0302 210B03          LXI   H,START  RST-VEKTOR 6
026  0305 226E00          SHLD  :006E  UMLEITEN
027  0308 E1                POP   H
028  0309 FB                EI
029  030A C9              RET
030  *
031  * ERWEITERUNG DES NORMALEN INTERRUPTS
032  *
033  030B 211303          START  LXI   H,RETURN  RETURN-ADRESSE
034  030E E3                XTHL          AUF DEN STACK
035  030F E5                PUSH  H
036  0310 C378D5          JMP   :D578  NORMALE ROUTINE
037  *
038  *
039  0313 F5              RETURN  PUSH  PSW
040  0314 3A0001          LDA   :100  J
041  0317 B7                ORA   A      J LAEUFT EIN BASIC
042  0318 C22203          JNZ  JUMP1  J PROGRAMM ?
043  031B 3A0101          LDA   :101  J
044  031E B7                ORA   A      J
045  031F CA2603          JZ   JUMP2
046  0322 AF              JUMP1  XRA   A      J WENN JA, LOESCHE
047  0323 32C402          STA   :2C4  J BREAK FLAG
048  0326 F1              JUMP2  POP   PSW
049  0327 C9              RET
050  *
051  *
052  0328                END

```

* SYMBOL TABLE *

INIT 0300 JUMP1 0322 JUMP2 0326 RETURN 0313
START 030B

D-Basic goes DISCO

A disk based DBASIC

A. general

A lot of people have been asking me to implement a DBASIC version on disk. Since there are several disk drive systems available for DAI, each operated by one or more disk operating systems, adapting DBASIC for each system separately would require too much work and would increase the risk of incompatibility between different DBASIC versions. Writing one extension for each disk drive system would generate exactly the same problem.

This is why I hesitated so long to come up with a disk based DBASIC.

Finally, in May, I was encouraged to proceed when the INDATA company ordered a DBASIC implementation on their 2*320 kbyte system.

To avoid incompatibility with DBASIC versions to be implemented on other disk drive systems in future, I designed the following DBASIC extensions :

1. The file handler (DBFIL.SYS file)

-all extended commands and functions concerning disk access go into one extension file : the file handler.

This extension is totally independent on the hardware connected to the DAI. This means that DBASIC will always use the same commands and functions to access files, regardless of the disk drive system used (INDATA floppy disk drives, KENDOS drives, hard disk?...etc...).

Except for the FILES command and the DSKF function, the file handler is also completely independent on the way files are stored on diskette (handled by the basic dos).

Further on in this text, most commands and functions of the file handler are explained.

2. The basic dos (DBDOS.SYS file)

The basic dos takes care of the layout of files, directory entries and disk space allocation information on disk. Like the file handler the basic dos is completely independent on hardware. The current DBDOS works on a CPM V2.2 filing structure :

i.e. -16 logical drives A-P can be installed

- a drive can contain up to 8 megabytes
- a file can contain up to 8 megabytes
- the disk layout is table defined

If you are the owner of both the CPM V2.2 and the DBASIC V2.2 packages for your disk drive, the same files can be accessed by both systems if you specify identical disk parameter headers and disk parameter blocks in both BIOS's (see CPM operating system manual).

Note that other filing systems could be developed for DBASIC to accomplish file compatibility with MSX-DOS, MS-DOS or other disk operating systems.

3. The basic input/output system (BIOS640.SYS, BIOS160.SYS...)

The bios part handles the physical input from and output to the disk drive. At this moment I have completed a bios for the INDATA 2*320 kbyte floppy drive and a RAMdisk (which I used for testing) I am about to complete a bios for the INDATA 2*80 kbyte floppy drive and other bios's will follow soon.

B. Booting and linking

When switching on your DAI, DBASIC V2.2 will be loaded automatically, followed by the configuration file "CONFIG.CFG". The standard configuration file will execute the following commands :

```
$EXTEND "DBDOS"  
$EXTEND "DBFIL"  
$EXTEND "<bios name>"  
DBDOS="DBDOS"  
DBIOS="<bios name>"  
DISK "A:"  
$INPUT="AUTOEXEC.BAT"  
LOAD "AUTOEXEC.BAS"
```

i.e. DBASIC will be extended with the basic dos, the file handler and the bios. Because these extensions are relocated they have to be linked together :

DBDOS="DBDOS" will link the "DBDOS" extension to the file handler.

DBIOS="<bios name>" will link the bios to the basic dos.

DISK "A:" will select drive A.

Then, if present, the "AUTOEXEC.BAT" batch file will be loaded and executed.

If not present the DBASIC program "AUTOEXEC.BAS" will be loaded and

executed if it is a compiled program. Note that the configuration file may be changed, but ALWAYS include the first 6 commands.

C. File names

The syntax for file names is compatible with the CPM file name syntax. In general : <file reference>=

[<drive>:]<filename>[.<file ext>]

with :

<drive> =A,B,...P

<file name>=up to 8 alphanumeric characters or !, ", #, \$, %, & or '.

If <file name> contains more than 8 characters it will be truncated.

<file ext> =up to 3 alphanumeric characters or !, ", #, \$, %, & or '.

If <file ext> contains more than 3 characters it will be truncated.

Lower case characters will be converted to uppercase.

-Omitting <drive>: means the default drive selected by the DISK command will be accessed.

-Omitting .<file ext> means the default file extension for the command using <file reference> will be supplied :

command	default file ext	file type
LOAD	.BAS	0
SAVE		
R (utilities)	.UTY	1
W (utilities)		
LOADA	.ARR	2
SAVEA		
\$EXTEND	.SYS	\$
FILES	.DAT	3
OPEN		
CREATE		
KILL		
PROTECT		
UNPROTECT		
\$OUTPUT		
\$INPUT		
SPL files	.SPL	&
DNA files	.DNA	#
/	.TXT	4
/	.HLP	5

Note : file types correspond to a default file extension but are not used by DBDOS (ex. SAVE "MYPRG.CPL" is valid)

D. File access

DBASIC can access files in sequential, random, relative and random-relative mode. Records in files can have both fixed and variable length.

-files : before a file can be opened, the file must be declared in DBASIC using the FILE statement :

ex. 100 FILE INFILE,OUTFILE,WORKFILE

The FILE statement reserves the variables INFILE,OUTFILE and WORKFILE as file variables and allocates space for their file control blocks and dma-buffers. Note that DBASIC references files by a file variable (cfr. pascal) rather than by a file number as in other disk basics.

-Open a file : before any access can be made on a file it has to be opened with the open statement.

ex. 110 OPEN OUTFILE,"B:TEST.TST"
120 OPEN ADDRESSES,AD\$+".ADR"

Note that the OPEN statement does not need a mode specification like "FOR INPUT", "FOR OUTPUT" or "FOR APPEND". If you want to extend a file (append mode), locate the end of file first :

ex. 200 OPEN MYF,F\$:LOEOF MYF

1.--- VARIABLE LENGTH RECORDS ---

After opening a file you can write data into it using the #PRINT statement :

ex. 130 I=1:A\$="good "
140 #PRINT OUTFILE;I,A\$,"day"

In the file the different elements of the record will be separated by ',' and the record will be completed with carriage return, line feed.

i.e. 1 , g o o d , d a y Odh Oah

#PRINT writes sequential because of the variable record length.

The end of file mark is written by :

150 #PRINT OUTFILE;CHR\$(#1A)
160 CLOSE OUTFILE:END

IMPORTANT : at the end of a program

you always have to close those files data has been written in (#PRINT or PUT), to update the directory.

Variable length records can be read by :

```
200 #INPUT INFILE;NUM,A$,B$
```

The following example illustrates the use of #PRINT and #INPUT.

```
10 TITLE "EX001"
20 FILE TEST
30 OPEN TEST,"TEST"
40 ON BREAK GOTO "CLOS:A=0"
50 WHILE A=0 DO INPUT "TYPE TEXT OR
BREAK";TEXT$
60 #PRINT TEST,TEXT$
70 WEND
100 "CLOS CLOSE TEST
110 OPEN TEST,"TEST"
120 WHILE EOF(TEST)=0 DO #INPUT TEST
;TEXT$
130 PRINT TEXT$:WEND
140 END
```

2. --- FIXED LENGTH RECORDS ---

To access files in random, relative or random-relative mode, fixed length records have to be used. They can be defined by the FIELD statement :

```
ex. 50 FIELD NAME;LF$(NAME$,20),LF$(NAME$,10)
60 FIELD NAMAD;FLD(NAME),LF$(ADDRESS$,20),RG$(NR$,5),AGE,PCT!
```

In this example the FIELD statement will reserve NAME and NAMAD as field variables.

LF\$(NAME\$,20) reserves 20 bytes in the record for the NAME\$ variable, NAME\$ will be justified left.

RG\$(NR\$,5) reserves 5 bytes in the record for the NR\$ variable, NR\$ will be justified right.

FLD(NAME) includes the NAME field into the NAMAD field.

For integer and floating point variables 4 bytes will be reserved in the field.

Writing the records into a file will be done by the PUT statement. In general :

```
PUT <file var>[,<random file position>];
<field var>[.<relative record>]
```

```
ex. 90 PUT MYF,1000;NAMAD
```

will assemble in the file MYF at random

file position 1000 (bytes) the record defined by the field NAMAD.

```
ex. 90 PUT MYF;NAMAD.1
```

will write relative record 1.

NAMAD.0 will start at file position 0

NAMAD.1 will start at file position

$0 + \text{FLDLEN}(\text{NAMAD}) = 0 + 20 + 10 + 20 + 5 + 4 + 4 = 63$

NAMAD.I will start at file position $I * \text{FLDLEN}(\text{NAMAD})$. FLDLEN(NAMAD) represents the field length)

```
ex. 90 PUT MYF,OFFSET;NAMAD.J
```

NAMAD.J will be written starting at file position OFFSET+J*FLDLEN(NAMAD)

```
ex. 90 PUT MYF;NAMAD
```

will write records sequential into the file.

IMPORTANT : at the end of a program you always have to close those files data has been PUT in.

Reading records will be done by GET :

```
GET <file var>[,<random file position>];
<field var>[.<relative record>]
```

GET performs exactly the opposite operation of PUT.

E. System output/input

-System output : you can redirect all data normally printed on screen or printer to a disk file.

```
ex. 10 $OUTPUT="AUTOEXEC.BAT"
20 PRINT "$EXTEND ";
30 PRINT CHR$(#22);"KEY";CHR$(#22)
40 PRINT "KEYON"
50 PRINT CHR$(#1A)
60 END
```

This small program creates a "AUTOEXEC.BAT" file which will extend DBASIC with function keys.

```
ex. $OUTPUT="PRINT.PRN":$LIST
```

This command will generate a cross reference listing into the PRINT.PRN file.

-system input : Input normally coming from keyboard comes from a file.

```
ex. 10 $INPUT="AUTOEXEC.BAT"
20 WHILE A=0 DO INPUT A$
30 PRINT A$:WEND
```

Input coming from AUTOEXEC.BAT will be placed in A\$.

```
ex. $INPUT="AUTOEXEC.BAT"
```

Input coming from AUTOEXEC.BAT will be interpreted by DBASIC.

```
ex. $INPUT="HOME.TXT"
```

with HOME.TXT containing the text :

```
100 PROCEDURE HOME
110 ?CHR$(12);
120 END PROC
```

will merge the HOME procedure to the program currently in memory.

F. Listing the directory

General : FILES[<file reference>]

```
ex. FILES
```

will show all files on the disk in the selected drive.

```
ex. FILES "b:*.BAS"
```

will show all DBASIC program files on the disk in drive B.

```
ex. FILES "AUTO?X??.BA*"
```

will show all files whose name begins with AUTO, have a X on position 6 and have a file extension beginning with BA. Note that wildcards (? and *) are only allowed in the files command.

G. Utility commands

Utility commands other than FILES are :

-CREATE <file reference> : creates a file on disk : ex. CREATE "A:GENINFO"

-KILL <file reference> : deletes a file from disk : ex. KILL FNAM\$

-PROTECT <file reference> : write/delete disables a file : ex. PROTECT "XREF.SYS"

-UNPROTECT <file reference> : write/delete enables a file : ex. UNPROTECT F\$

-DISK[<drive>:] : selects a disk drive for further operations :

ex. DISK "B:" : selects drive B

DISK : reselects a disk drive after the DCR or CAS command has been given.

-MOUNT[<drive>:] : mounts the disk in <drive>, i.e. ready a disk for access.

ex. MOUNT "A:" : mounts the disk in drive A

MOUNT : mounts the disk in the selected drive.

note : The FILES command automatically mounts the disk.

H. Functions

Besides the functions already discussed (LF\$,RG\$,FLD and FLDLEN) there are :

-DSKF(<drive>:) : returns the free disk space in kbytes.

```
ex. PRINT DSKF("A:")
```

PRINT DSKF("") : returns the free disk space of the disk in the currently selected drive.

-LOC(<file var>) : returns the current file position in bytes.

```
ex. LOEOF TEST:EOFPOS=LOC(TEST)
EOFPOS contains the length of the file.
```

-EOF(<file var>) : returns 1 if the end of file has been reached or 0 else.

```
ex. 50 WHILE EOF(TEST)=0 DO #LINEINPUT
TEST;A$
```

```
60 PRINT A$:WEND
```

Willy Coremans

7^e Championnat International de programmes d'Othello-Reversi
7th Othello-Reversi Programs World Championship

Organisé par

**L'ORDINATEUR
L'INDIVIDUEL**

21-22 septembre 1985

CATÉGORIE :
ORDINATEURS
DE POCHE
(INTERPRÈTES
ET COMPILES)

7^e Championnat International de
programmes d'Othello-Reversi
7th Othello-Reversi Programs World Championship

BULLETIN D'INSCRIPTION

Prénom, Nom :
Adresse :
Téléphone :
Age :
Matériel utilisé (y compris taille, mémoire et périphériques) :

Nom du programme :
Auteur du programme :
Taille du programme :
Langage utilisé :
Le matériel utilisé est-il un ordinateur de poche ?
Le programme est-il entièrement interprété ?
Ou bien comporte-t-il au moins une partie compilée
ou assemblée ?

Autres membres de l'équipe :

ATTENTION : Les concurrents doivent disposer de leur matériel.

Tout concurrent non présent à 8H 45 sera exclu du concours.

L'alimentation électrique fournie sera du 220 V avec prise de terre.

Le Championnat a lieu

le SAMEDI 21 SEPTEMBRE 1985, pour les ordinateurs de table,
les SAMEDI 21 et DIMANCHE 22 SEPTEMBRE 1985, pour les ordinateurs
de poche, au SICOB, au CNIT-LA DEFENSE.

Le rendez-vous est fixé à 8H 30 dans le hall d'arrivée du R.E.R.
l'endroit exact vous sera confirmé ultérieurement.

Le présent bulletin est à retourner à L'ORDINATEUR INDIVIDUEL
(OTHELLO) 5 place du Colonel-Fabien 75491 PARIS Cedex 10.

L'ORDINATEUR INDIVIDUEL (Othello) 5 place du Colonel-Fabien 75491 Paris Cedex 10 ☎ (1) 240 22 01

RÈGLEMENT

1. - Chaque partie se déroule sur un espace de $6 \times 6 = 36$ cases, les coordonnées pour référencer chaque carré lors de la communication des coups entre les joueurs sont exprimées de même manière qu'aux échecs : Blanc aura le carré A1 dans son coin gauche et H1 dans son coin droit. Noir aura A6 dans son coin droit et F 6 dans son coin gauche.
2. - Pour des raisons de simplicité, la position de départ est celle d'Othello : en début de partie, Noir a des pièces sur les carrés C3 et D4, Blanc sur les carrés C4 et D3. Noir joue le premier à partir de cette position.
3. - Chaque joueur dispose d'un certain temps pour jouer tous les coups d'une partie. Ce temps est de 60 minutes par joueur et par partie. Si au cours d'une partie l'un ou l'autre des adversaires dépasse son temps alloué, l'arbitre doit obligatoirement être appelé. L'arbitre pourra alors soit déclarer vainqueur le joueur qui n'a pas excédé son temps, soit demander la continuation de la partie (le temps étant toujours décompté) s'il estime qu'elle peut être terminée rapidement. Dans ce dernier cas, il fixera à chacun des joueurs une limite de temps que celui-ci ne pourra dépasser faute d'être déclaré perdant.
4. - Si un joueur perd par manque de temps, le nombre de point marqué par chaque joueur sera calculé par péréquation à 36, à partir du nombre de pions présents sur l'échiquier. Toutefois, si le gagnant a moins de pions que le perdant, le score sera alors de 31 points pour ce dernier et de 33 points pour le gagnant.
5. - A la fin d'une partie achevée normalement (passe de deux adversaires, ou 36 pions posés), chacun des deux adversaires se voit attribuer un nombre de pions calculé comme indiqué ci-dessous :
A) Si tous les pions ont été posés, le nombre de pions retenu pour chacun est égal au nombre de pions dont il dispose sur l'échiquier.
B) Si moins de 36 pions figurent sur l'échiquier, on calcule d'abord le nombre de pions à attribuer au vainqueur (celui qui en a le plus), soit R. Ce nombre est calculé à partir du nombre de pions sur l'échiquier, soit V pour le vainqueur et P pour le perdant, suivant la formule $R = \text{Partie entière de } (V \times 36 / (V + P) + 0.5)$. Le nombre de pions retenus pour le perdant est alors de $36 - R$.
6. - Une victoire vaut deux points, un match nul un point et une défaite zéro. Si deux programmes ou plus sont ex aequo, ils sont départagés par les totaux (pions propres - pions de l'adversaire) des parties qu'ils ont gagnées dont l'on soustraira les totaux des parties perdues. Cette méthode avantage les programmes qui gagnent avec les plus grands écarts. S'il y a encore des ex aequo, ils seront dits ex aequo sauf s'il est possible de jouer une nouvelle partie (ou des nouvelles parties) pour les départager.
7. - Chaque participant doit apporter son PSI, ainsi que tous les accessoires nécessaires à son fonctionnement (prises multiples notamment).
8. - Chaque participant est responsable de son PSI et de son bon fonctionnement.
9. - Si un joueur (par maladresse), ou sa machine (par plantage du matériel) interrompt la partie et que celle-ci ne puisse plus être reconstituée dans un délai raisonnable, ce joueur sera déclaré perdant. Son adversaire est déclaré vainqueur et marque 2 points. L'attribution des nombres de pions se fait suivant la procédure décrite au paragraphe 4.
10. - En cas de coup irrégulier d'un programme, ce programme est déclaré perdant de la partie. Son adversaire est déclaré vainqueur et marque 2 points. Il sera attribué 36 pions au vainqueur et 0 à son adversaire.
11. - Chaque participant dispute 6 parties sur 2 jours.
12. - Chaque ronde se dispute selon le système suisse, c'est-à-dire qu'à l'issue de chaque tour, l'arbitre fait jouer les concurrents qui totalisent le même nombre de points. Deux concurrents ne peuvent s'affronter dans une partie s'ils ont déjà disputé un match ensemble. En conséquence, l'arbitre les fera jouer contre des concurrents ayant le total le plus approchant.
13. - Si un joueur ne peut jouer faute de prise à réaliser, il passe son tour, mais s'il peut prendre, il est obligé de prendre.
14. - Les coups joués par chaque concurrent sont écrits par lui-même à chaque coup sur la feuille de marque ambulante qui circule entre les deux adversaires. La notation utilisée est obligatoirement celle définie au paragraphe 1.
15. - Dans la mesure du possible, un jeu complet d'Othello Reversi sera placé entre les opérateurs humains qui auront la responsabilité de la pose des pions et du retournement des pions capturés. A part les opérateurs et l'arbitre, personne ne devra toucher damier et pions pendant la partie.
16. - On ne peut changer aucun paramètre du programme en cours de partie, mais on peut indiquer au programme le temps dont il dispose encore, s'il requiert cette information.
17. - Il est prévu deux catégories :
- Interpréteurs ne faisant appel à aucun sous-programme en langage machine ;
- Langage machine, assembleur ou compilateur.
Certains détails à ce sujet ne peuvent être décidés qu'en fonction du nombre d'engagés dans chaque catégorie. L'arbitre les explicitera au début de l'épreuve.
18. - Les arbitres de la rencontre sont seuls juges pour régler tout litige.
19. - Le niveau de jeu des machines ou le temps de réflexion par coup est fixé avant le début de chaque partie par les personnes qui les ont engagées dans le tournoi. Il ne peut en aucun cas être modifié en cours de jeu à l'initiative des manipulateurs. Dans le cas où le programme demande lui-même un changement, celui-ci pourra se faire après que l'arbitre ait été appelé pour en constater la validité.
20. - La participation au tournoi sous-entend l'acceptation du présent règlement.

L'ORDINATEUR INDIVIDUEL tient à rappeler aux participants de ce tournoi qu'il s'agit avant tout d'un jeu. Le climat se doit d'être celui d'une grande fête de l'informatique individuelle.
C'est donc l'esprit et non la lettre du règlement qui sera pris en considération par l'arbitre pour trancher tout litige.

Random Numbers

Hardings Cottage, Swan Lane
Winterbourne, Avon BS17 1RJ
Winterbourne (0454) 775731
ENGLAND

23 June 1985

Dear Sir,

RANDOM NUMBERS

I was interested in Raymond Vanlathem's article in Issue No 28 of DAInamic. When I write programmes for students of Economics, I usually randomise the initial values of variables; so that, by repetition, they learn the basic principles, not just a set of moves (or that's what I hope). So it was disturbing to realise that they would always start from the same point after start-up. (RND(1) is always 0.571069 first time round, on my machine, by the way.)

In this situation, what matters is that the user cannot control or repeat the initial values. The routines proposed by Raymond Vanlathem would not be suitable - in effect, they require the user himself to supply the random element. A solution which has occurred to me is to use the internal Timer over an interval which is difficult to predetermine e.g. where an input or key entry is called for. A self-contained example follows. Although the choice for the value of RND(1) is confined to the first 255 members of the pseudo-random series, I think it is pretty hard for the user to determine which it will be.

```
10 REM * * * RANDOM NUMBER SEEDER * * *
20 POKE #1BE,#FF
30 PRINT "Press any key for a nearly random number."
40 A=GETC:IF A=0.0 GOTO 40
50 FOR I=1.0 TO PEEK(#1BE):RANDOM=RND(1.0):NEXT
60 PRINT "Your nearly random number is";RND(1.0)
```

Yours sincerely,

Jonathan Harlow
Jonathan Harlow

Programmoteek

```
80 PRINT CHR$(12):CLEAR 20000:A=24:A1=A-1:L=0:N=0:S=0:T=0:Z=0
90 FOR I=#275 TO #28F:POKE I,#10:NEXT:REM IMPINT
100 CURSOR 27,14:PRINT "D C R"
110 CURSOR 17,12:PRINT "P R O G R A M M O T E E K"
120 CURSOR 16,10:PRINT "voor";A;" cassetteband-kanten."
130 CURSOR 20,7:PRINT "Wilt u uitleg.? J/N":GOSUB 3920:REM GETC
140 IF Q<>74 AND Q<>78 THEN 130
150 IF Q=78 THEN 170
160 GOSUB 5500:GOTO 80
170 DIM AB(A1),AP(A1),NB(200),NP(200)
180 DIM PL(A1,15),PT(A1,15),LP(200),TP(200),TL(A1),VL(A1)
190 DIM NP$(A1,15),PN$(200),D$(0)

200 REM ***** MENUTEKST *****
205 PRINT CHR$(12):CURSOR 0,23
210 PRINT " Hier is uw PROGRAMMOTEEK. Wenst u:":PRINT
215 PRINT " A. een BESTAND in te lezen.?"
220 PRINT " B. een OVERZICHT van de banden.?"
225 PRINT " C. een OVERZICHT van de programma 's op 'een' band.?"
230 PRINT " D. een OVERZICHT van alle programma's per band.?"
235 PRINT " E. een WIJZIGING van een overzicht.?"
240 PRINT " F. een NIEUW programma in te schrijven.?"
245 PRINT " G. het BEWAREN van het bestand.?"
250 PRINT " H. het PRINTEN van het bestand.?"
255 PRINT " I. het ZOEKEN naar een programma.?"
260 PRINT " J. het SAMENSTELLEN van een programma lijst.?"
265 PRINT " K. het SORTEREN van de programma lijst.?"
270 PRINT " L. een OVERZICHT van de programma lijst.?"
275 PRINT " M. het BEEINDIGEN van dit programma.?"
280 PRINT
285 PRINT " Wat kan ik voor u doen.?"
290 PRINT " Type de gewenste letter A-M in. "
300 W=0:GOSUB 3920
310 IF Q>64 AND Q<78 THEN Q=Q-64
320 IF Q<1 OR Q>13 THEN 300
330 IF Q=1 OR Q=6 OR Q=13 THEN 400
340 IF Z=0 THEN GOSUB 3600:GOTO 300
350 IF Q<>11 AND Q<>12 THEN 400
360 IF S=0 THEN GOSUB 3700:GOTO 300

370 REM ----->A B C D E F G H I J K L M
400 ON Q GOSUB 500,700,900,1000,1100,1500,1900,2200,2400,2700,2800,3100,3500
410 IF W=1 THEN 300
420 GOTO 200

500 REM ***** BESTAND INLEZEN *****
510 IF Z=1 THEN GOSUB 3800:W=1:RETURN
520 GOSUB 4400:REM BLANCO REGELS
530 CURSOR 9,5:PRINT "Is het bestand op de band aanwezig.? J/N":GOSUB 3920
540 IF Q<>74 AND Q<>78 THEN 520
550 IF Q=78 THEN GOSUB 4400:W=1:RETURN
560 CURSOR 9,3:PRINT "Dient de band te worden gespoeld.? J/N ":GOSUB 3920
570 IF Q<>74 AND Q<>78 THEN 560
580 IF Q=78 THEN W=1:GOTO 600
590 GOSUB 4600:RETURN
600 CURSOR 21,1:PRINT "Even geduld a.u.b."
610 LOADA D$ " Datum."
620 LOADA AB " Aantal banden."
630 LOADA PL " Programmalengte."
640 LOADA PT " Programmatype."
```

```

650  LOADA AP " Aantal programma's per band."
660  LOADA NP$ " Namen van de programma's per band."
670  CURSOR 9,1:PRINT "Het bestand bevindt zich in het geheugen."
680  Z=1:RETURN

700  REM ***** HET PRINTEN VAN HET BANDEN-OVERZICHT. *****
710  POKE #131,1:PRINT CHR$(12):POKE #131,0
730  PRINT "      Banden overzicht dd. ";D$(0)
740  PRINT "      Vlg- Aant. Vry ! Vlg- Aant. Vry ! Vlg- Aant. Vry"
750  PRINT "      no. prgr. lgt.! no. prgr. lgt.! no. prgr. lgt."
760  PRINT "      -----"
770  GOSUB 4100:REM Vullen van de array's.
780  A3=A/3:FOR I=0 TO A3-1
790  1  LI=INT(LEN(STR$(I+1))):LA=INT(LEN(STR$(AP(I)))):LV=
1  INT(LEN(STR$(VL(I))))
800  1  LI1=INT(LEN(STR$(I+A3+1))):LA1=INT(LEN(STR$(AP(I+A3)))):LV1=
1  INT(LEN(STR$(VL(I+A3))))
810  1  LI2=INT(LEN(STR$(I+2*A3+1))):LA2=INT(LEN(STR$(AP(I+2*A3)))):LV2=
1  INT(LEN(STR$(VL(I+2*A3))))
820  1  PRINT TAB(9-LI);I+1;TAB(15-LA);AP(I);TAB(21-LV);VL(I);" !";
830  1  PRINT TAB(26-LI1);I+A3+1;TAB(32-LA1);AP(I+A3);TAB(38-LV1);VL(I+A3);"
1  !";
840  1  PRINT TAB(43-LI2);I+2*A3+1;TAB(49-LA2);AP(I+2*A3);TAB(55-LV2);VL(I+
1  2*A3)
850  NEXT I:PRINT :GOSUB 3900:RETURN

900  REM ***** OVERZICHT VAN DE INHOUD VAN EEN BAND. *****
910  GOSUB 4400:REM BLANCO REGELS
920  CURSOR 9,5:PRINT "Type het volgnummer,>0<";A+1;
930  INPUT ",van de band in. ";I:PRINT
940  IF I<1 OR I>48 THEN 920
950  I=I-1:PRINT CHR$(12):GOSUB 4200:REM Printen van overzicht.
960  GOSUB 3900:REM GETC
970  RETURN

1000 REM ***** TOTAALOVERZICHT VAN DE PROGRAMMA'S, PER BAND. *****
1010 GOSUB 4400:CURSOR 9,3:PRINT "Voor beëindiging Type return.":WAIT TIME
100
1020 POKE #131,1:PRINT CHR$(12):POKE #131,0
1030 FOR I=0 TO AB(0)-1:POKE #131,1:PRINT CHR$(12):POKE #131,0
1040 1  GOSUB 4200:REM Printen van overzicht.
1050 1  POKE #131,1:GOSUB 3890:IF Q=13 THEN RETURN
1060 POKE #131,0:PRINT :NEXT I:RETURN

1100 REM ***** HET WIJZIGEN VAN EEN OVERZICHT. *****
1110 GOSUB 4400:REM BLANCO REGELS
1120 CURSOR 9,5:PRINT "Wenst u het programmabestand te wijzigen.?J/N ":PRINT
:GOSUB 3920:REM GETC
1130 IF Q<>74 AND Q<>78 THEN 1120
1150 IF Q=78 THEN W=1:GOSUB 4400:RETURN
1160 GOSUB 900:REM Overzicht van de inhoud van een band.
1170 INPUT "Oud programma-volgnummer:          ";K:PRINT
1180 IF K<1 OR K>16 THEN 1170
1190 J=K-1
1200 PRINT "Hieronder kunt u de wijziging intypen."
1210 PRINT "Hetgeen ongewijzigd blijft,beantwoordt u met 0 "
1220 INPUT "Nieuw programma-volgnummer,>0<17:      ";L:PRINT
1230 IF L=0 THEN 1260
1240 IF L<1 OR L>16 THEN 1220
1250 J=L-1

```

```

1260 INPUT "Nieuwe programmalengte in sec.          ";L:PRINT
1270 IF L=0 THEN 1300
1280 IF L<0 OR L>92 THEN 1260
1290 PL(I,J)=L
1300 PRINT "Voor ongewijzigd type 3 in."
1310 INPUT "Nieuw type programma (0,1),-data (2). ";L:PRINT
1320 IF L=3 THEN 1350
1330 IF L<0 OR L>3 THEN 1310
1340 PT(I,J)=L
1350 INPUT "Nieuwe programmaam. ";L$:PRINT
1360 IF L$="" THEN 1350
1370 IF L$="0" THEN 1390
1380 NP$(I,J)=L$
1390 IF AB(0)>I+1 THEN 1410
1400 AB(0)=I+1
1410 RETURN

1500 REM ***** HET INSCHRIJVEN VAN EEN NIEUW PROGRAMMA. *****
1505 GOSUB 4400:REM BLANCO REGELS
1510 CURSOR 9,5:PRINT "U wenst een nieuw programma in te schrijven.? "
1520 CURSOR 9,3:PRINT "Betreft het een bestaande band.? J/N ":GOSUB 3920:REM
GETC
1530 IF Q<>74 AND Q<>78 THEN 1520
1540 IF Q=74 THEN 1630
1550 GOSUB 700:REM Bandenoverzicht.
1560 PRINT CHR$(12):PRINT "Voor einde inschrijving type 0 in.":PRINT
1570 PRINT "Type het gewenste bandvolgnummer >0<";A+1.0;
1580 INPUT " in. ";L:PRINT
1590 IF L=0 THEN RETURN
1600 IF L<0 OR L>A THEN 1570
1610 I=L-1:IF AB(0)=I THEN AB(0)=I+1
1620 GOTO 1690
1630 IF Z=0 THEN GOSUB 3600:W=1:RETURN
1640 PRINT CHR$(12):PRINT "Voor einde inschrijving type 0 in.":PRINT
1650 INPUT "Type het bandvolgnummer in. ";L:PRINT
1660 IF L=0 THEN RETURN
1670 IF L<0 OR L>A THEN 1650
1680 I=L-1
1690 INPUT "Type het programmavolgnummer in.          ";L:PRINT
1700 IF L=0 THEN RETURN
1710 IF L<0 OR L>16 THEN 1690
1720 AP(I)=L:J=L-1
1730 INPUT "Type de programmalengte in sec. in.          ";L:PRINT
1740 IF L=0 THEN RETURN
1750 IF L<0 OR L>92 THEN 1730
1760 PL(I,J)=L
1770 INPUT "Type het type progr. (0,1),-data (2) in. ";L:PRINT
1780 IF L<0 OR L>2 THEN 1770
1790 PT(I,J)=L
1800 INPUT "Type de programmaam in. ";NP$(I,J):PRINT
1810 IF NP$(I,J)=" " THEN 1800
1820 Z=1
1830 PRINT :GOTO 1690

1900 REM ***** HET BEWAREN VAN HET BESTAND OP DE BAND. *****
1910 GOSUB 4400:REM BLANCO REGELS
1920 CURSOR 9,5:PRINT "U wilt het bestand bewaren.? J/N          ":GOSUB 3920
1930 IF Q<>74 AND Q<>78 THEN 1920
1940 IF Q=78 THEN GOSUB 4400:W=1:RETURN
1950 CURSOR 9,3:PRINT "Dient de band te worden gespoeld.? J/N":GOSUB 3920

```

```

1960 IF Q<>74 AND Q<>78 THEN 1950
1970 IF Q=78 THEN W=1:GOTO 1990
1980 GOSUB 4600:RETURN
1990 CURSOR 9,1:INPUT "Type de datum in. ";D$(0)
2000 GOSUB 4400:REM BLANCO REGELS
2010 CURSOR 21,3:PRINT "Even geduld a.u.b."
2020 SAVEA D$ " Datum."
2030 SAVEA AB " Aantal banden."
2040 SAVEA PL " Programmalengte."
2050 SAVEA PT " Programmatype."
2060 SAVEA AP " Aantal programma's per band."
2070 SAVEA NP$ " Namen van de programma's per band."
2080 CURSOR 9,3:PRINT "Het bestand staat op de band. "
2090 RETURN

2200 REM ***** HET UITPRINTEN VAN DE TEKST. *****
2210 PRINT CHR$(12)
2240 PRINT "Indien u tekst wilt uitprinten, zet dan de"
2250 PRINT "printer aan 'voor' het intypen van de letter"
2260 PRINT "B, C, D of L voor het uitprinten van het over-"
2270 PRINT "zicht van de programma's."
2280 GOSUB 3900:REM GETC
2290 RETURN

2400 REM ***** HET OPZOEKEN VAN EEN PROGRAMMA. *****
2410 T=0:GOSUB 4400:REM BLANCO REGELS
2420 CURSOR 9,3:PRINT "Naar welk programma moet worden gezocht.?"
2430 CURSOR 9,1:INPUT "Type een sleutelwoord van 3 letters. ";S$:PRINT :
PRINT
2440 IF S$="" THEN 2430
2450 F=LEN(S$):IF F>3 THEN 2430
2460 FOR I=0 TO AB(0)-1:FOR J=0 TO AP(I)-1
2470 2 IF LEN(NP$(I,J))<F THEN F=LEN(NP$(I,J))
2480 2 IF LEFT$(NP$(I,J),F)=S$ THEN GOSUB 2530
2490 NEXT J:NEXT I:IF T>0 THEN 2590
2500 CURSOR 9,3:PRINT "Sleutelwoord komt in het bestand niet voor. "
2510 CURSOR 9,1:PRINT "Wilt u een ander sleutelwoord invoeren.!" "
2520 WAIT TIME 200:GOTO 2600
2530 IF T=0 THEN PRINT CHR$(12)
2540 PRINT "Programma met sleutelwoord ";S$;". "
2550 PRINT "Bandvolgnummer is ";I+1
2560 PRINT "Programmavolnummer is: ";J+1
2570 PRINT "Naam van het programma is: ";NP$(I,J);". "
2580 T=T+1:T1=T MOD 4:PRINT :IF T1=0 THEN 3900:RETURN
2590 GOSUB 3900:REM GETC
2600 RETURN

2700 REM ***** SAMENSTELLEN VAN PN$. *****
2710 N=1:GOSUB 4400:CURSOR 21,3:PRINT "Even geduld a.u.b."
2720 FOR I=0 TO AB(0)-1:FOR J=0 TO AP(I)-1
2730 2 NB(N)=I+1:NP(N)=J+1:PN$(N)=NP$(I,J)
2740 2 LP(N)=PL(I,J):TP(N)=PT(I,J)
2750 N=N+1:NEXT J:NEXT I:N=N-1:S=1
2760 CURSOR 9,3:PRINT "De lijst is samengesteld. "
2770 W=1:RETURN

2800 REM ***** SORTEREN VAN PN$. *****
2810 GOSUB 4400:CURSOR 21,3:PRINT "Even geduld a.u.b."
2820 FOR Z1=INT(N/2) TO 2 STEP -1:X=Z1
2830 1 K$=PN$(X):L=NB(X):M=NP(X):O=LP(X):P=TP(X)

```

```

2840 1 Y=X+X
2850 1 IF Y>N THEN 2910
2860 1 IF Y=N THEN 2880
2870 1 IF PN$(Y+1)>PN$(Y) THEN Y=Y+1
2880 1 IF K$>=PN$(Y) THEN 2910
2890 1 PN$(X)=PN$(Y):NB(X)=NB(Y):NP(X)=NP(Y)
2900 1 LP(X)=LP(Y):TP(X)=TP(Y):X=Y:GOTO 2840
2910 PN$(X)=K$:NB(X)=L:NP(X)=M:LP(X)=O:TP(X)=P:NEXT
2920 K$=PN$(1):L=NB(1):M=NP(1):O=LP(1):P=TP(1)
2930 FOR Z1=N TO 2 STEP -1:X=1
2940 1 Y=X+X
2950 1 IF Y>Z1 THEN 3010
2960 1 IF Y=Z1 THEN 2980
2970 1 IF PN$(Y+1)>PN$(Y) THEN Y=Y+1
2980 1 IF K$>=PN$(Y) THEN 3010
2990 1 PN$(X)=PN$(Y):NB(X)=NB(Y):NP(X)=NP(Y)
3000 1 LP(X)=LP(Y):TP(X)=TP(Y):X=Y:GOTO 2940
3010 1 PN$(X)=K$:NB(X)=L:NP(X)=M:LP(X)=O:TP(X)=P
3020 1 K$=PN$(Z1):L=NB(Z1):M=NP(Z1):O=LP(Z1):P=TP(Z1)
3030 1 PN$(Z1)=PN$(1):NB(Z1)=NB(1):NP(Z1)=NP(1)
3050 LP(Z1)=LP(1):TP(Z1)=TP(1):NEXT
3060 PN$(1)=K$:NB(1)=L:NP(1)=M:LP(1)=O:TP(1)=P
3070 CURSOR 9,3:PRINT "De lijst is gesorteerd. "
3080 W=1:RETURN

3100 REM ***** PRINTEN VAN PN$. *****
3110 GOSUB 4400
3120 CURSOR 9,3:INPUT "Hoeveel regels wenst u per bladzijde. ";V:PRINT
3130 IF V>60 THEN GOTO 3120
3140 POKE #131,1:PRINT CHR$(12):POKE #131,0:N1=1:N2=N
3150 PRINT " Programma overzicht dd. ";D$(0)
3160 PRINT " Volg- Band- Progr- Progr- Naam van het programma."
3170 PRINT " no. no. no. lgt."
3180 PRINT " -----"
3190 IF N2-N1>V THEN N2=N1+V-1
3200 FOR Z1=N1 TO N2
3210 1 LZ=INT(LEN(STR$(Z1))):LB=INT(LEN(STR$(NB(Z1)))):LP1=
1 INT(LEN(STR$(NP(Z1))))
3220 1 LLP=INT(LEN(STR$(LP(Z1)))):LTP=INT(LEN(STR$(TP(Z1))))
3230 1 PRINT TAB(7-LZ);Z1;TAB(13-LB);NB(Z1);TAB(19-LP1);NP(Z1);
3240 1 PRINT TAB(26-LLP);LP(Z1);TAB(30-LTP);TP(Z1);" ";PN$(Z1)
3250 NEXT Z1:N1=N2+1:N2=N
3260 POKE #131,1:PRINT :GOSUB 3900:POKE #131,0
3270 IF N-N1>0 THEN PRINT :GOTO 3160
3280 RETURN

3500 REM ***** BEVEILIGING TEGEN VERLIES VAN DATA. *****
3510 GOSUB 4400:REM BLANCO REGELS
3520 CURSOR 9,3:PRINT "Heeft u niets te bewaren.? J/N"
3530 GOSUB 3920:REM GETC
3540 IF Q=74 THEN 1900:REM Bewaren bestand.
3550 IF Q=78 THEN 8000:REM END.
3560 GOTO 3500

3600 REM ***** Mededeling van geen bestand aanwezig *****
3610 GOSUB 4400:FOR D=1 TO 6
3620 1 CURSOR 9,3:PRINT "Er is geen bestand in het geheugen.!" "
3630 1 WAIT TIME 20
3640 1 CURSOR 9,3:PRINT " "
3650 1 WAIT TIME 10

```

```

3660 NEXT D:RETURN

3700 REM ***** Mededeling over programmalijs. *****
3710 GOSUB 4400:FOR D=1 TO 6
3720 1 CURSOR 9,3:PRINT "Eerst de programmalijs samenstellen (J). "
3730 1 WAIT TIME 20
3740 1 CURSOR 9,3:PRINT " "
3750 1 WAIT TIME 10
3760 NEXT D:RETURN

3800 REM ***** Mededeling over aanwezigheid van bestand. *****
3810 GOSUB 4400:FOR D=1 TO 6
3820 1 CURSOR 9,3:PRINT "Er is reeds een bestand in het geheugen.!"
3830 1 WAIT TIME 20
3840 1 CURSOR 9,3:PRINT " "
3850 1 WAIT TIME 10
3860 NEXT D:RETURN
3890 CURSOR 0,1:PRINT "Type space of return.":GOTO 3920

3900 REM ***** GETC ROUTINE *****
3910 CURSOR 0,1:PRINT "Type space. ";
3920 Q=GETC:Q=GETC:Q=GETC
3930 Q=GETC:IF Q=0 THEN 3930
3940 IF Q>96 AND Q<123 THEN Q=Q-32
3950 CURSOR 0,1:PRINT " ";RETURN

4100 REM ***** HET VULLEN VAN DE VRYE RUIMTE ARRAY *****
4110 FOR C=0 TO A1:TL(C)=0:FOR D=0 TO AP(C)-1
4120 TL(C)=TL(C)+PL(C,D):NEXT D:VL(C)=92-TL(C):NEXT C:RETURN

4200 REM ***** PRINTEN VAN EEN OVERZICHT *****
4210 GOSUB 4100:REM Vullen vrye ruimte array.
4220 PRINT " Programma overzicht. dd. ";D$(0)
4230 PRINT " Bandvolgno: ";I+1;" ";TAB(21);"Aantal programma's:";
4240 PRINT AP(I);" "
4250 PRINT " Volg- Progr- Vrije bandlengte: ";VL(I);" sec."
4260 PRINT " nummer. lengte. Naam van het programma."
4270 PRINT " -----"
4280 FOR J=0 TO AP(I)-1
4290 1 LJ=INT(LEN(STR$(J+1))):LPL=INT(LEN(STR$(PL(I,J)))):LPT=
1 INT(LEN(STR$(PT(I,J))))
4300 1 PRINT TAB(9-LJ);J+1;TAB(19-LPL);PL(I,J);TAB(24-LPT);PT(I,J);" ";NP$(I,
1 J)
4310 NEXT J:RETURN

4400 REM ***** BLANCO REGELS MAKEN *****
4410 CURSOR 0,5:PRINT "
"
4420 CURSOR 0,3:PRINT "
"
4430 CURSOR 0,1:PRINT "
"
4440 RETURN

4600 REM ***** BAND-SPOEL-ROUTINE *****
4640 PRINT CHR$(12):PRINT "Kies dan hiervoor het juiste nummer."
4650 PRINT
4660 PRINT "0 = geen spoelen.: 1=REW1 : 2=REW2 : 3=REW3 : 4=REW4"
4670 PRINT "5=REW : 6=SKIP1 : 7=SKIP2 : 8=SKIP3 : 9=SKIP":PRINT
4680 Q=10:PRINT "Type het juiste nummer in. "

```

```

4690 GOSUB 3920:REM GETC
4700 IF Q<49 THEN RETURN
4710 IF Q>57 THEN 4680
4720 Q=Q-48
4730 ON Q GOSUB 4810,4820,4830,4840,4850,4860,4870,4880,4890
4740 GOTO 4690
4810 CALLM #F000:REM REW1
4815 RETURN
4820 CALLM #F000:REM REW2
4825 RETURN
4830 CALLM #F000:REM REW3
4835 RETURN
4840 CALLM #F000:REM REW4
4845 RETURN
4850 CALLM #F000:REM REW
4855 RETURN
4860 CALLM #F000:REM SKIP1
4865 RETURN
4870 CALLM #F000:REM SKIP2
4875 RETURN
4880 CALLM #F000:REM SKIP3
4885 RETURN
4890 CALLM #F000:REM SKIP
4895 RETURN

5000 REM TOEWIJZING VARIABELEN.
5010 REM AX :=Maximaal aantal cassettebanden i.v.m DIN.
5020 REM AX dient een veelvoud van 3 te zijn.
5030 REM AB%:=Aantal banden.
5040 REM AP%:=Array van aantallen programma's op een band.
5050 REM D$ :=Datum string.
5060 REM LX :=Hulp variabele.
5070 REM LP%:=Sorteerarray van PL%.
5080 REM NX :=Aantal programma's.
5090 REM NB%:=Sorteer array voor nummer-band.
5100 REM NP%:=Sorteer array voor nummer-programma.
5110 REM NP$:=Array van namen van de programma's.
5120 REM PL%:=Array van programmalingten in seconden.
5130 REM PN$:=Sorteer array van NP%.
5140 REM PT%:=Array van programmatypen (0,1) en data (2).
5150 REM SX :=Teller bij het samenstellen van de programmalijs.
5160 REM TX :=Teller bij het printen na zoeken.
5170 REM TL%:=Array van totale lengten op de band in seconden.
5180 REM TP%:=Sorteer array van PT%.
5190 REM VX :=Aantal regels per bladzijde bij printen.
5200 REM VL%:=Array van vrye ruimten op de banden in sec.
5210 REM WX :=Teller tegen nieuwe menu tekst.
5220 REM ZX :=Teller voor al of niet aanwezigheid bestand in geheugen.
5500 REM ***** UITLEG EN AUTEUR NAAM *****
5510 PRINT CHR$(12):PRINT " ***** UITLEG PROGRAMMA *****"
5520 PRINT
5530 PRINT "Dit programma voorziet in een complete en overzichtelyke"
5540 PRINT "administratie van het DCR cassettebanden bestand, bestaan-"
5550 PRINT "de uit programma's, zowel in BASIC (0),als in ASSEMBLERTAAL"
5560 PRINT "(1) en DATA (2). "
5570 PRINT "Het programma kent als cassetteband-eenheid de cassetteband-"
5580 PRINT "kant, dus kant A of B. De tijdsduur voor het spoelen (CHECK)"
5590 PRINT "van een kant is vastgesteld op 96 seconden. Daarvan wordt 4"
5600 PRINT "seconden gereserveerd voor de, aan het begin van de band,"
5610 PRINT "aanwezige DCR Tapecontrol. Rest bandlengte bedraagt dus 92"

```



```

5620 PRINT "seconden. Elk programma of data bezet een plaats op de band,"
5630 PRINT "waarvan de lengte wordt uitgedrukt in seconden looptijd van"
5640 PRINT "de band."
5650 POKE #131,1:GOSUB 3900:PRINT CHR$(12):POKE #131,0
5660 PRINT "Geadministreerd worden:"
5670 PRINT
5680 PRINT "      Datum."
5690 PRINT "      Aantal banden (lees band-kanten)."
5700 PRINT "      Programma lengte (of data lengte)."
5710 PRINT "      Programma type (0, 1 of 2)."
5720 PRINT "      Aantal programma's per band."
5730 PRINT "      Namen van de programma's per band."
5740 PRINT
5750 PRINT "Het DCR Tapecontrol programma is herzien en aangepast aan"
5760 PRINT "dit programma. In de kop van dit programma is een constante"
5770 PRINT "% gedeclareerd, die geheugenruimte voor % cassetteband-"
5780 PRINT "kanten data reserveert. Al naar behoefte kan de waarde van"
5790 PRINT "% worden aangepast, rekening houdend met de beschikbare"
5800 PRINT "geheugencapaciteit."
5810 PRINT "% dient een drievoud te zijn."
5820 PRINT
5830 PRINT "Veel succes ermee."
5840 POKE #131,1:GOSUB 3900:PRINT CHR$(12):POKE #131,0
6000 PRINT
6010 PRINT "      *****"
6020 PRINT "      *                               *"
6030 PRINT "      *   P R O G R A M M O T E E K   *"
6040 PRINT "      *                               *"
6050 PRINT "      * AUTHOR: Ing. L.C.J. Dietz      *"
6060 PRINT "      *       St. Josephstraat 12     *"
6070 PRINT "      *       5961 GM Horst (L)       *"
6080 PRINT "      *       The Netherlands         *"
6090 PRINT "      *       Phone 04709-1642        *"
6100 PRINT "      *       DAI Personal Computer   *"
6110 PRINT "      *       Horst, januari 1984.    *"
6120 PRINT "      *                               *"
6130 PRINT "      *****"
6140 GOSUB 3900:PRINT CHR$(12):RETURN
8000 GOSUB 4400:CURSOR 21,3:END

```

Low Cost Digitizer

A LOW COST DIGITIZER

Cher monsieur hermans

Je vous remercie pour l'interet que vous avez porte a mon convertisseur. Suite a votre demande je vous envoie les plans ainsi que le software necessaire pour le faire fonctionner.

CARACTERISTIQUES GLOBALES:

- utilise le mode 6 (336*256 points, 4 niveaux de gris)
- temps de conversion = 12 sec (durant lesquelles l'image doit rester immobile)
- le balayage de l'image se fait de gauche a droite
- utilise le DCE-BUS (connecteur de 34 pins)
- utilise un signal video standard (*) accessible sur tout les magnetoscopes (1V, 75 OHMS = VIDEO OUT)

(*) REM: les meilleurs resultats ont ete obtenus avec des images en noir et blanc.

COMMENTAIRES GENERAUX SUR LE HARDWARE:

La simplicite de la realisation a pu etre obtenue en adoptant les principes suivants :

- utiliser un systeme de declenchement analogique
- faire executer le maximum d'operation par le software
- convertir sur 4 niveaux seulement puisque le DAI est limite a 4 niveaux (pour les modes sans restrictions)
- ne recourir a aucune RAM externe (utilisation directe de la RAM video du DAI)

COMMENTAIRES GENERAUX SUR LE SOFTWARE

Le role du software est de :

- gerer les signaux de synchronisation de ligne et de trame
- imprimer au fur et a mesure sur l'ecran le resultat de la conversion
- supprimer les problemes d'entrelacement entre les lignes paire et impaire en ne prenant qu'une image sur deux

SA REALISATION

Ne disposant que d'un prototype du convertisseur sur plaquette d'essais (**), je ne peux donc pas vous fournir les masques necessaires a la realisation du circuit imprime.

(**) Ce convertisseur n'exige par consequent aucune precaution HF particuliere.

Remarque sur l'alimentation :

Personnellement, j'ai prefere ne pas risquer d'utiliser directement les tensions (-5,+5,+12V) accessibles sur le connecteur du DCE-bus (l'alimentation du DAI etant assez mal protegee) . Il me semble plus sage d'utiliser une alimentation stabilisee exterieure (-6,+5,+12V).

A titre de renseignement voici les consommations du convertisseur sur les differentes tensions :
 I(12V)=23 mA : I(5V)=14 mA : I(-6V)=17 mA

LES REGLAGES

Le convertisseur possede en tout deux types de reglages :

- a) le reglage des niveaux de gris
- b) le reglage de la largeur de l'image

a) Ce reglage est indispensable pour obtenir des resultats visibles. Il concerne les 3 TRIMPOTS de 10K que nous appellerons B1, B2 et B3. Avec :

B3=transitions blanc - gris clair.
 B2=transitions gris clair - gris fonce.
 B1=transitions gris fonce - noir.

Par consequent B1 devra commuter sur des niveaux plus faibles que B2 et B2 sur des niveaux plus faibles que B3 (soit B3 > B2 > B1)

Afin de faciliter ce reglage a tout ceux qui ne possede pas d'oscilloscope, j'ai egalement realise une mire de test ainsi qu'un programme utilitaire BASIC qui imprime l'etat des 3 comparateurs a differents endroits de l'image.

Methode de reglage

- enregistrer la mire sur un magnetoscope.
- connecter le magnetoscope au convertisseur (mode PLAY).
- charger le programme "REGLAGE" + RUN
- controler l'activite des 3 comparateurs.
 c'est a dire 1) verifier B3 >= B2 >= B1
 2) essayer d'obtenir pour les differentes nuances :

B1	B2	B3	
1	1	1	(noir)
0	1	1	(gris fonce)
0	0	1	(gris clair)
0	0	0	(blanc)

- modifier en consequence les TRIMPOTS B1, B2, B3
- ETC...

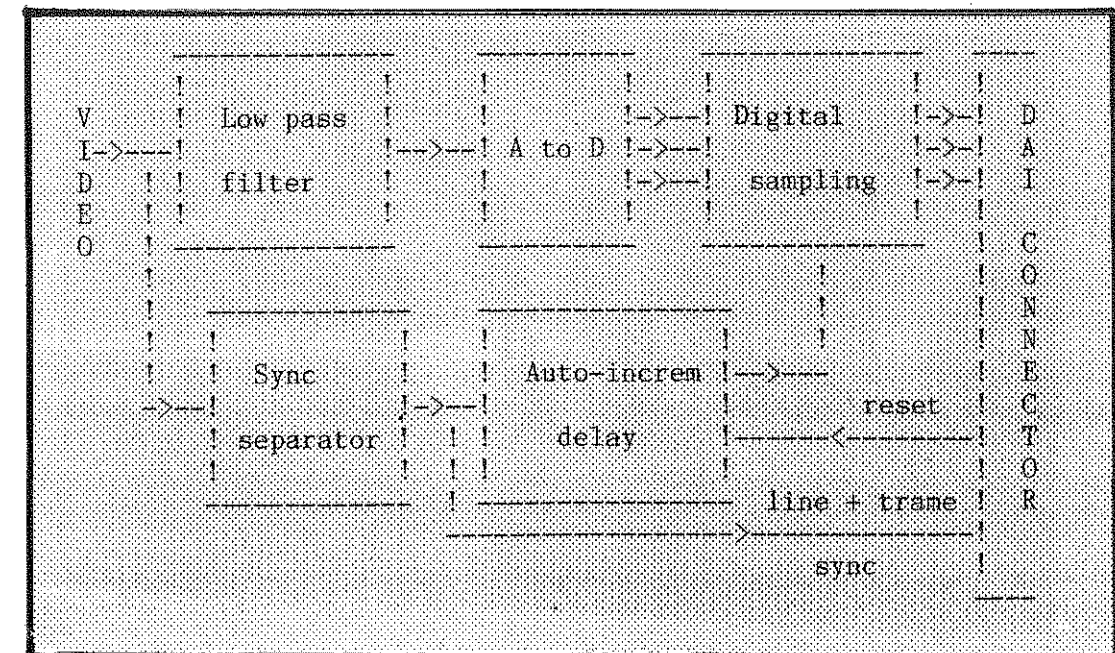
rem:

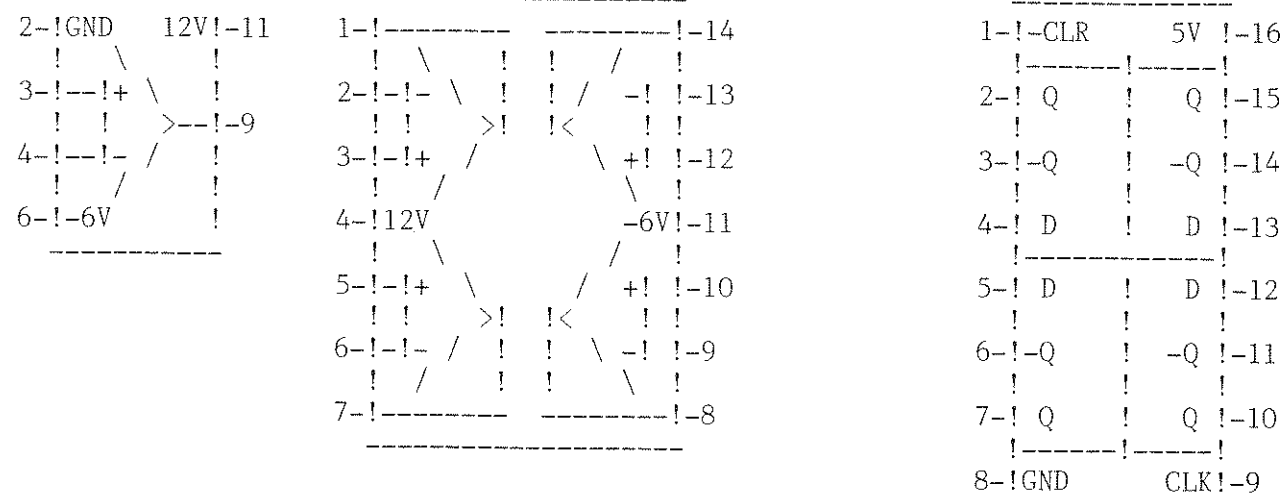
Il est possible que le reglage de la largeur de l'image (TRIMPOT 100K) interfere avec celui ci. Il sera par consequent preferable de passer aux essais de fonctionnement reel des que les comparateurs commencent a commuter (c'est a dire lorsque les triades de chiffres changent de valeurs) et de regler la largeur de l'image (b).

CHARGEMENT DU SOFTWARE

LOAD "DIGITIZER"
 RUN

BLOCK DIAGRAM





```

5     REM SOFTWARE FOR DIGITIZER
10    POKE #FE03,#8B:REM INIT 8255
20    COLORG 3 7 11 13
30    MODE 6
40    FOR ADD=#E00 TO #EOF:READ A:POKE ADD,A:NEXT
50    FOR ADD=#F800 TO #F88F:READ A:POKE ADD,A:NEXT
60    POKE #FE00,#FF:WAIT TIME 10:POKE #FE00,#0:REM RESET
70    WAIT TIME 70
80    CALLM #F800
90    GOTO 60
1000  DATA #0,#0,#1,#1,#0,#0,#80,#80,#1,#1,#80,#80,#81,#81,#81
1010  DATA 245,229,213,197,243,33,0,0,34,2,13,33,235,191,34,0
1020  DATA 13,33,1,254,6,6,62,255,166,250,24,248,175,182,250,20
1030  DATA 248,5,194,29,248,175,182,242,38,248,17,163,0,27,122,179
1040  DATA 194,45,248,6,14,17,0,15,62,255,166,250,58,248,126,230
1050  DATA 14,79,10,18,28,194,56,248,17,0,15,1,167,255,42,0
1060  DATA 13,26,7,126,23,119,43,26,15,126,23,119,9,28,194,81
1070  DATA 248,1,0,90,9,34,0,13,235,42,2,13,35,34,2,13
1080  DATA 235,123,230,7,194,17,248,1,254,255,9,34,0,13,123,254
1090  DATA 80,194,17,248,122,254,1,194,17,248,251,193,209,225,241,201
    
```

```

2     REM CE PROGRAMME EST DESTINE AU REGLAGE DES 3 COMPARETEURS
5     POKE #FE03,#8B:REM INITIALISE 8255
6     POKE #FE00,#FF:WAIT TIME 5:POKE #FE00,#0:REM RESET INTERFACE
7     WAIT TIME 100
8     PRINT "B1","B2","B3":PRINT

9     REM IMPRIME 21 VALEURS D'ECHANILLONS SUR TOUTE L'IMAGE
10    FOR I=0 TO 20
12    1  WAIT TIME 26
15    1  A=PEEK(#FE01) IXOR #FF
20    1  PRINT (A SHR 1.0) IAND 1.0,(A SHR 2.0) IAND 1.0,(A SHR 3.0) IAND 1.0
30    NEXT
40    GOTO 5

1000  REM CORRECT VALUES ARE :
1010  REM 1 1 1 (NOIR)
1020  REM 0 1 1 (GRIS FONCE)
1030  REM 0 0 1 (GRIS CLAIR)
1040  REM 0 0 0 (BLANC)
1050  REM

2000  REM EX OF UNCORRECT VALUES :
2010  REM 1 0 1 B1 TRIMPOT > B2 TRIMPOT LEVEL
2020  REM -> REDUCE B1 OR INCREASE B2 TRIMPOT LEVEL
2030  REM 1 1 0 B3 TRIMPOT < B1 AND B2 TRIMPOT
2040  REM B1=B2=B3=ALWAY 0 ->ALL TRIMPOT ARE TOO LOW
2050  REM B1=B2=B3=ALWAY 1 ->ALL TRIMPOT ARE TOO HIGHT
    
```

```

2     REM MIRE DE TEST POUR LES REGLAGES DES COMPARETEURS
5     MODE 3
10    FOR X1=0.0 TO XMAX*3.0/4.0 STEP XMAX/4.0
20    1  READ COL!:FILL X!,0 X1+XMAX/4,YMAX COL!
30    NEXT
40    GOTO 40
1000  DATA 3,5,8,11
    
```

New Software

VEROVERING VAN MEMORY ALPHA

Vanuit een ruimteschip wordt je alleen achtergelaten op een donkere planeet. De bedoeling is om de basis Memory Alpha te bereiken via ingetypte commando's op de computer. De computer geeft op elk commando een ge-past antwoord of aanwijzing.

**Dutchmansgold
Conquest of Memory Alpha**

ADVENTURE!

Games collection 17

```

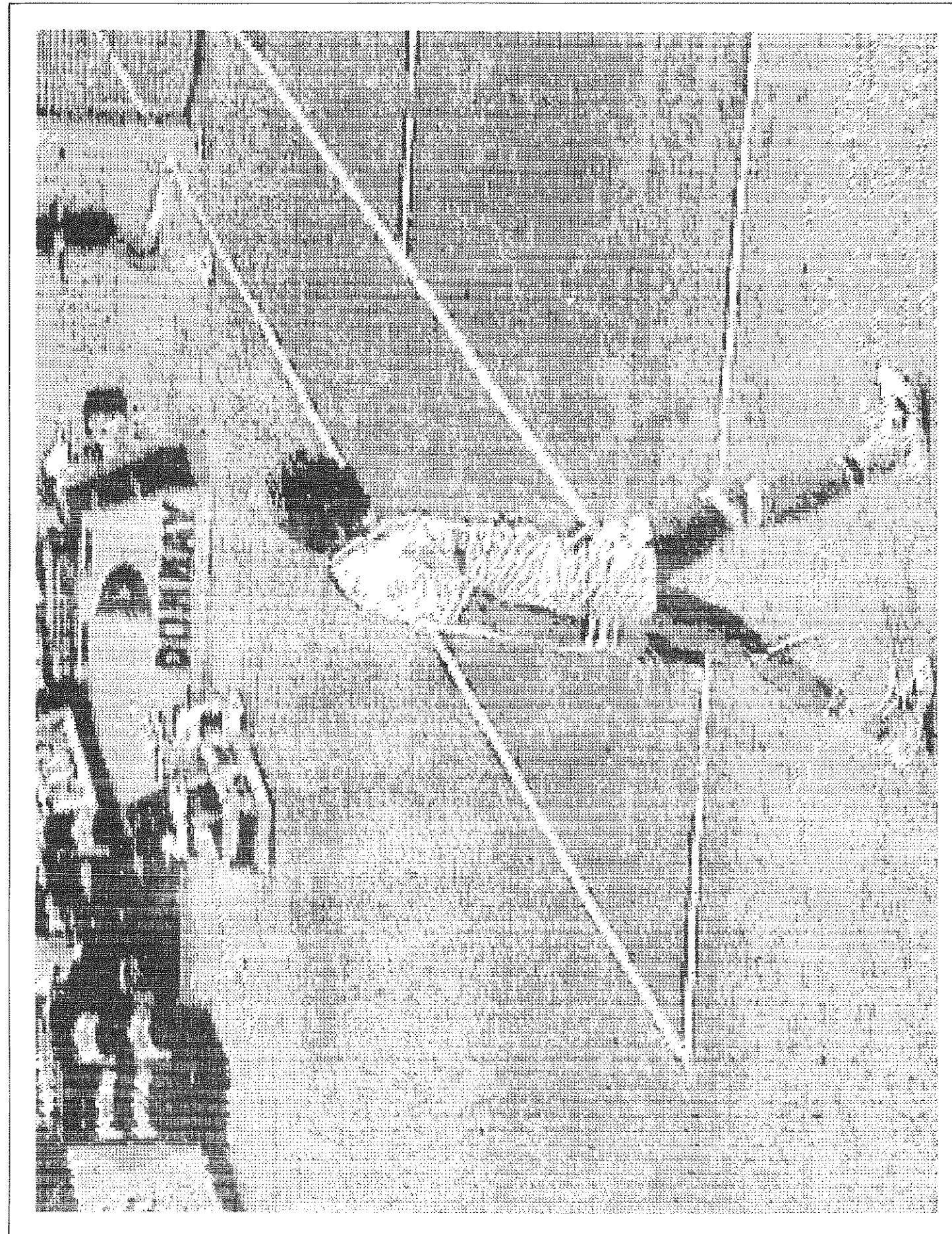
0000 ;
0000 ; *****
0000 ; * DIGITIZER SOFTWARE *
0000 ; *
0000 ; * WRITTEN BY PASCAL DE LAET *
0000 ; *****
0000 ;
0000 @=FE01 PORTB EQU 0FE01H
0000 @=0F00 BUF EQU 0F00H ;WORKING AREA (=256 BYTES)
0000 @=FFA7 DLTYM1 EQU 1H-5AH ;1 - LINE SIZE (MODE 6)
0000 @=BFEB BEGVID EQU 0BFEBH ;SCREEN TOP
0000 @=5A00 SETUP EQU 5A00H ;SCREEN SIZE (MODE 6)
0000 @=00A3 TEMPO EQU 0A3H ;VERTICAL POSITION OF WINDOW
0000 ;
0000 ORG 0D00H
0D00 0000 VIEDO DW 0H ;VIDEO RAM POINTER
0D02 0000 COMPT DW 0H ;VERTICAL LINE COUNTER
0D04 ;
0D04 ORG 0E00H
0E00 000001 TABLE DB 0H,0H,1H,1H,0H,0H,80H,80H,1H,1H,80H,80H,81H,81H,81H
0E10 ;
0E10 ; MAIN PROGRAM
0E10 ;
0E10 ORG 0F800H
F800 F5 PUSH PSW
F801 E5 PUSH H
F802 D5 PUSH D
F803 C5 PUSH B
F804 F3 DI
F805 ;
F805 210000 LXI H 0H
F808 22020D SHLD COMPT ;RESET LINE COUNTER
F80B ;
F80B 21EBBF LXI H BEGVID
F80E 22000D SHLD VIEDO ;LET VIEDO POINTER=SCREEN TOP
F811 ;
F811 ;
F811 2101FE DEBUT LXI H PORTB ;DEF HL=PORT B ADD (8255)
F814 ;
F814 ; WAIT BLANKING SIGNAL
F814 ;
F814 0606 LIGNE MVI B 6H ;LOAD BLANKING SIGNAL DELAY
F816 3EFF MVI A 0FFH ;SET BIT 7
F818 ;
F818 A6 LOOP ANA M
F819 FA18F8 JM LOOP ;WAIT B7 DOWN (8255)
F81C ;
F81C AF XRA A ;RESET BIT 7
F81D B6 LOOP1 ORA M
F81E FA14F8 JM LIGNE ;JUMP IF LINE SYNC DETECTED
F821 ;
F821 05 DCR B
F822 C21DF8 JNZ LOOP1
F825 ;
F825 ; NOW BLANKING SIGNAL IS DETECTED
F825 ;
F825 AF XRA A ;WAIT THE BLANKING SIGNAL END
F826 B6 WAITUP ORA M
    
```

```

F827 F226F8 JP WAITUP
F82A ;
F82A 11A300 LXI D TEMPO ;ADJUST THE WINDOW POSITION
F82D 1B TEMP DCX D
F82E 7A MOV A,D
F82F B3 ORA E
F830 C22DF8 JNZ TEMP
F833 ;
F833 ; LOAD 256 PIXELS FROM DIGITIZER TO BUFFER
F833 ;
F833 060E MVI B TABLE/100H
F835 ;
F835 11000F LXI D BUF
F838 ;
F838 3EFF ST MVI A 0FFH ;WAIT LINE SYNC
F83A A6 LOP ANA M
F83B FA3AF8 JM LOP
F83E ;
F83E 7E MOV A,M ;CAPTURE THE PIXEL
F83F E60E ANI 0EH
F841 ;
F841 4F MOV C,A ;CONVERT IT WITH TABLE
F842 0A LDAX B
F843 ;
F843 12 STAX D ;STORE IT IN BUF
F844 1C INR E
F845 C238F8 JNZ ST
F848 ;
F848 ;
F848 ; PRINT FROM BUFFER TO SCREEN IN MODE 6
F848 ;
F848 11000F LXI D BUF ;LET DE=BUFFER ADD
F84B ;
F84B 01A7FF LXI B DLTYM1 ;DEF BC=1-LINE SIZE
F84E ;
F84E 2A000D LHLD VIEDO ;RESTORE VIEDO RAM POINTER
F851 ;
F851 1A BOUCL LDAX D ;LET CY=PIXEL HIGHT BIT
F852 07 RLC
F853 ;
F853 7E MOV A,M ;STORE IT IN VIEDO RAM
F854 17 RAL
F855 77 MOV M,A
F856 ;
F856 2B DCX H ;IDEM WITH LOW BIT
F857 ;
F857 1A LDAX D
F858 0F RRC
F859 ;
F859 7E MOV A,M
F85A 17 RAL
F85B 77 MOV M,A
F85C ;
F85C 09 DAD B ;ADAPT VIEDO RAM POINTER
F85D 1C INR E ;ADAPT BUFFER POINTER
F85E C251F8 JNZ BOUCL ;TREAT THE NEXT PIXEL
F861 ;
F861 01005A LXI B SETUP ;Y=0 -> Y=YMAX
    
```

```

F864 09          DAD B
F865           ;
F865 22000D      SHLD  VIDEO      ;SAVE VIDEO RAM POINTER
F866           ;
F868 EB         XCHG             ;INCREMENT LINE COUNTER
F869 2A020D     LHL  COMPT      ;AND PUT IT IN DE REG PAIR
F86C 23         INX H
F86D 22020D     SHLD  COMPT
F870 EB         XCHG
F871           ;
F871 7B         MOV  A,E
F872 E607       ANI   7H        ;TEST IF THE FIELD ROW IS FULL
F874 C211F8     JNZ   DEBUT
F877 01FEFF     LXI  B  0H-2H   ;THEN FILL THE NEXT FIELD ROW
F87A 09         DAD  B
F87B 22000D     SHLD  VIDEO
F87E           ;
F87E 7B         MOV  A,E        ;TEST IF SCREEN FULL
F87F FE50       CPI   50H
F881 C211F8     JNZ   DEBUT
F884 7A         MOV  A,D
F885 FE01       CPI   1H
F887 C211F8     JNZ   DEBUT      ;THEN END
F88A           ;
F88A FB         EI
F88B C1         POP  B
F88C D1         POP  D
F88D E1         POP  H
F88E F1         POP  PSW
F88F C9         RET
F890           END
    
```

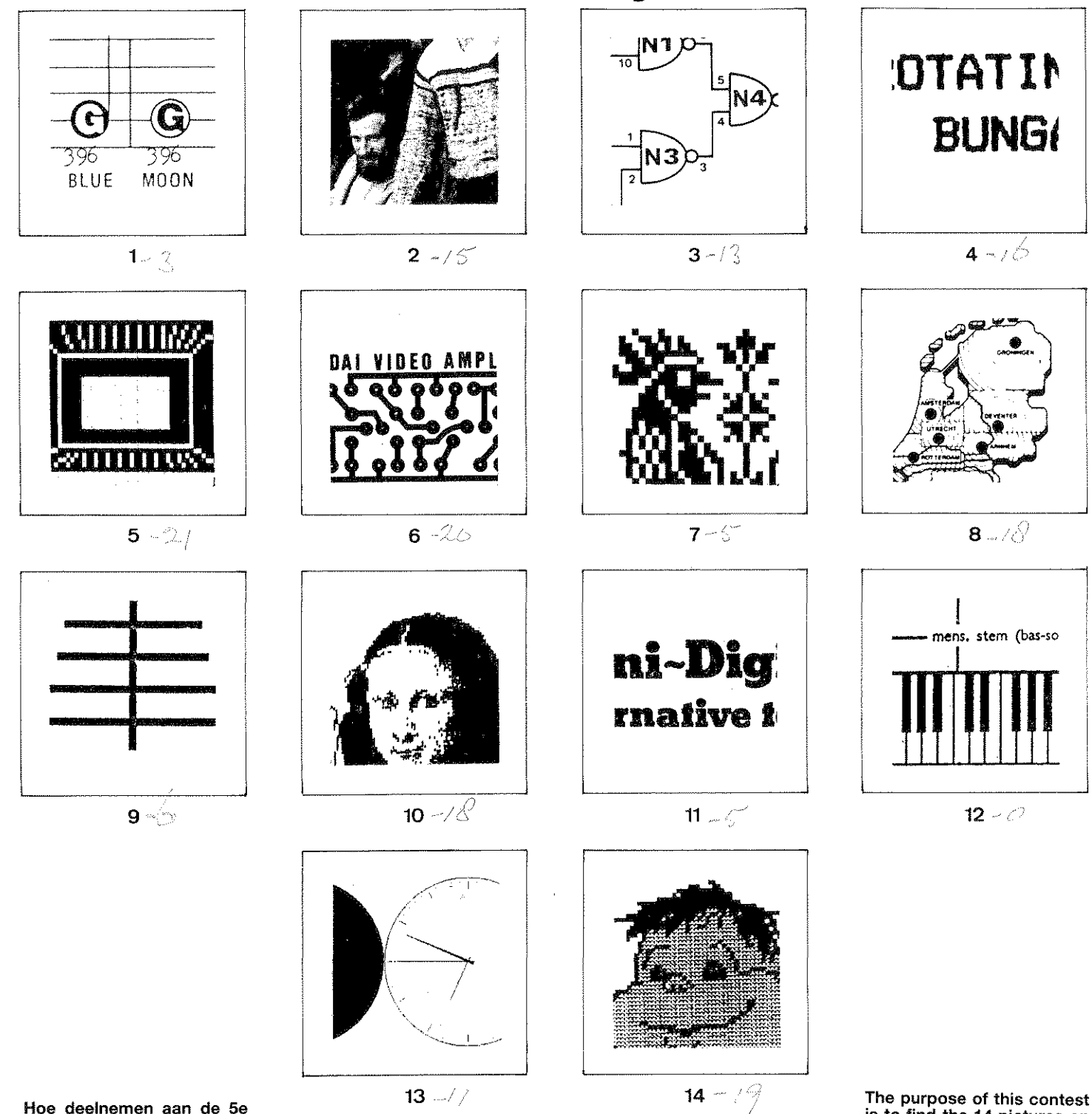
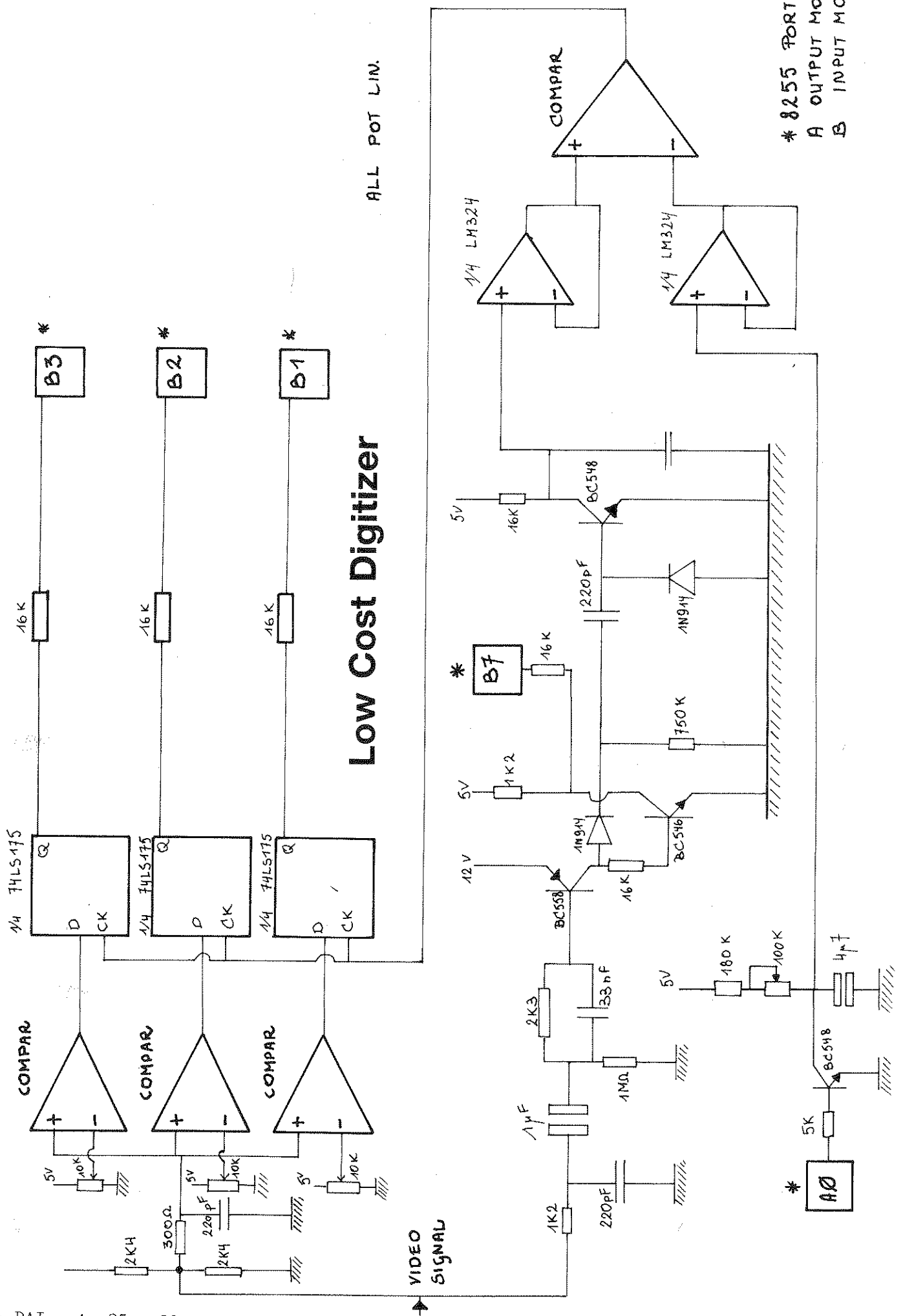


New Software

ASTRO INVADERS

Een mooi en kleurrijk uitgevoerde invadersspel. Verhinder dat de ruimte-eieren afgeworpen worden of je raken. Het spel wordt gespeeld met behulp van de cursortoetsen (bewegen links-rechts) en de eventknop om te schieten. Je kan extra bonuspunten verdienen door op tijd de Ufo te raken die af en toe op het scherm verschijnt. Het spel past de moeilijkheidsgraad aan (wordt sneller) in evenredigheid met de tijd dat je het speelt.

5th Anniversary Contest



Hoe deelnemen aan de 5e verjaardags wedstrijd van DAInamic

U zoekt in de verschillende DAInamic jaargangen de 14 afbeeldingen of teksten van pagina. Met uitgavenummer geeft U via de conversietabel een bepaalde letter. vb.: U heeft illustratie 1 gevonden in de uitgave nummer 12. De conversietabel bepaalt dat in hokje 1 de letter L dient ingevuld te worden. Als de 14 hokjes correct zijn ingevuld komt uw oplossing te voorschijn. Stuur uw kaart volledig ingevuld naar de redactie. Tussen de juiste oplossingen worden een kleurenmonitor en andere waardevolle prijzen verloot.

1 = A	14 = N
2 = B	15 = O
3 = C	16 = P
4 = D	17 = Q
5 = E	18 = R
6 = F	19 = S
7 = G	20 = T
8 = H	21 = U
9 = I	22 = V
10 = J	23 = W
11 = K	24 = X
12 = L	25 = Y
13 = M	26 = Z

The purpose of this contest is to find the 14 pictures on page in one of the DAInamic magazines. The magazine number gives you a character mentioned in the conversion table ex.: you found illustrations 1 in magazine 12, so this give you the character 'L'. All 14 characters give you the solution. Put this solution on the (compleet filled out) attached card and send it to the redaction. A color monitor and other valuable prices can be won. good luck !