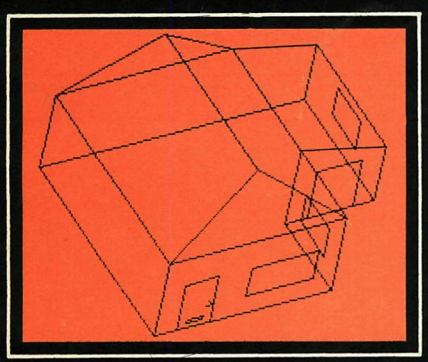
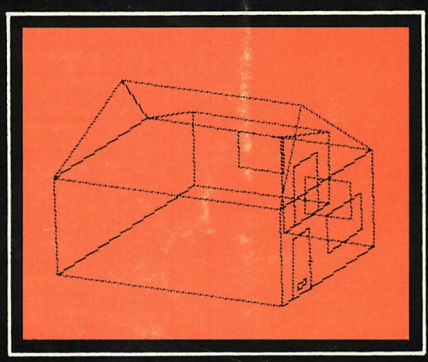
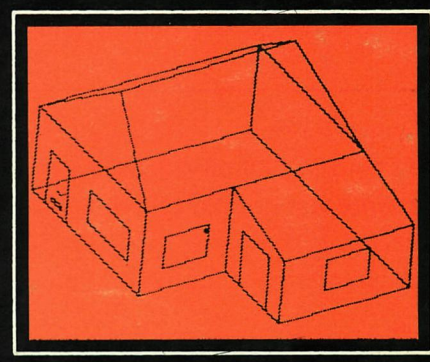
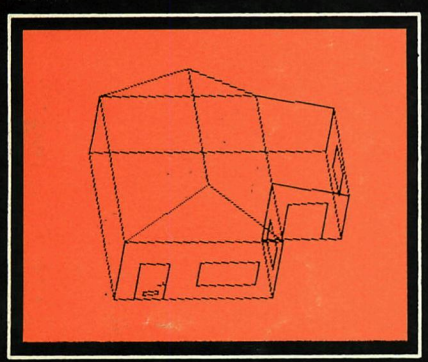
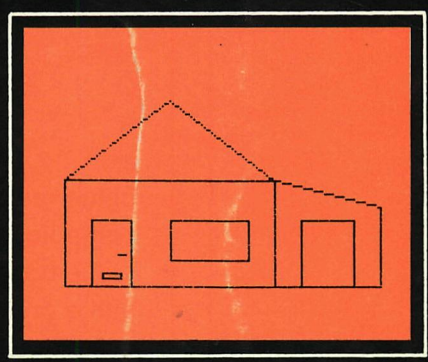
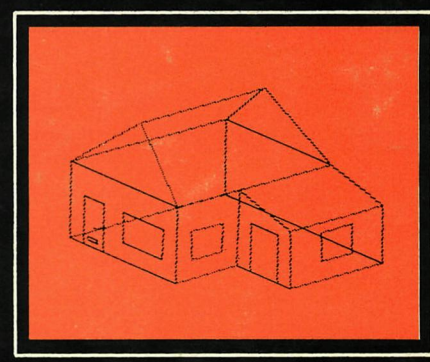
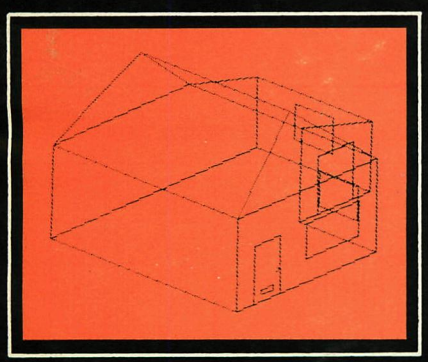
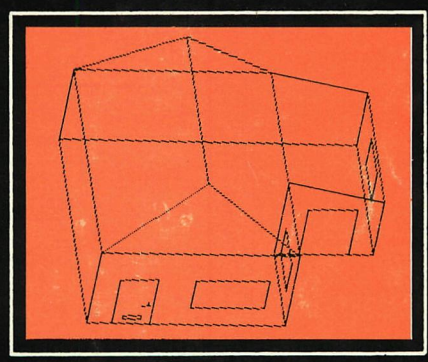
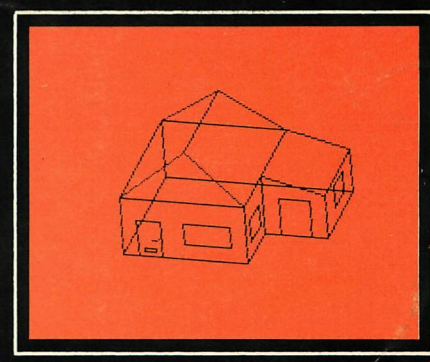
DAI NAMIC

16

tweemaandelijks tijdschrift mei - juni 1983

personal computer users club

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

International

COLOFON

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

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

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

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

Lidgelden

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

DAINAMIC

PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

NEWSLETTER 16

Westmeerbeek, juni 1983

Beste leden,

Met Newletter 16 bieden we U weer een bonte verzameling programma's, artikels en tips. Aan de namen van de auteurs kan je merken dat er bijdragen zijn uit heel Europa, de vertaaldiensten zorgen er verder wel voor dat dit alles voor de diverse nationaliteiten te begrijpen is. Voor de Nederlandse leden wordt bestellen en betalen een stuk eenvoudiger : Just van Dunne is zo vriendelijk geweest voor DAInamic een POSTGIRO nummer in Nederland te openen. Hier kan je voortaan terecht voor alle verrichtingen i.v.m. software,hardware en abonnementen.

J.F. van Dunne
Hoflaan 70
3062 JJ ROTTERDAM

GIROnummer : 4083817 Telefoon : 010/144802

Wij wensen U nog een prettige vakantie, tot binnen 2 maanden.

Dear members,

With Newsletter 16 we offer you a lot of programs, articles and hints for your DAI computer. Many thanks to the authors all over Europe. Special thanks to Bill Read, Colin Hards, Dave Atherton, Cedric Dufour and all those who take care of the translations. BASICODE II is a new version of the HOBBYSCOOP transmission standard, written for DAI by Th. van Lieshout. Maybe you can capture Hilversum radio on the short waves ? In next issue we hope to bring you an extended test-report on KEN-DOS, the system should be ready by that time. Rumours are that INDATA is also preparing new drives , we will let you know.

We wish you sunny holidays, enjoy newsletter 16.
p.s. don't take WOM too serious ...
Wilfried Hermans

NEWSLETTER 16

INHOUD — CONTENTS

DAInamic subscription rates :

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

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

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

DAINIBBLE

NEW SOFTWARE

********** DO YOU LIKE ACTION ?

Five little men are waiting till they can jump into the maze of DAINIBBLE, where they will be running around under your control till they are nibbled by a monster. After his death a man gets a decent funeral and the place is marked with a cross. A man cannot walk over the place where another man has been buried.

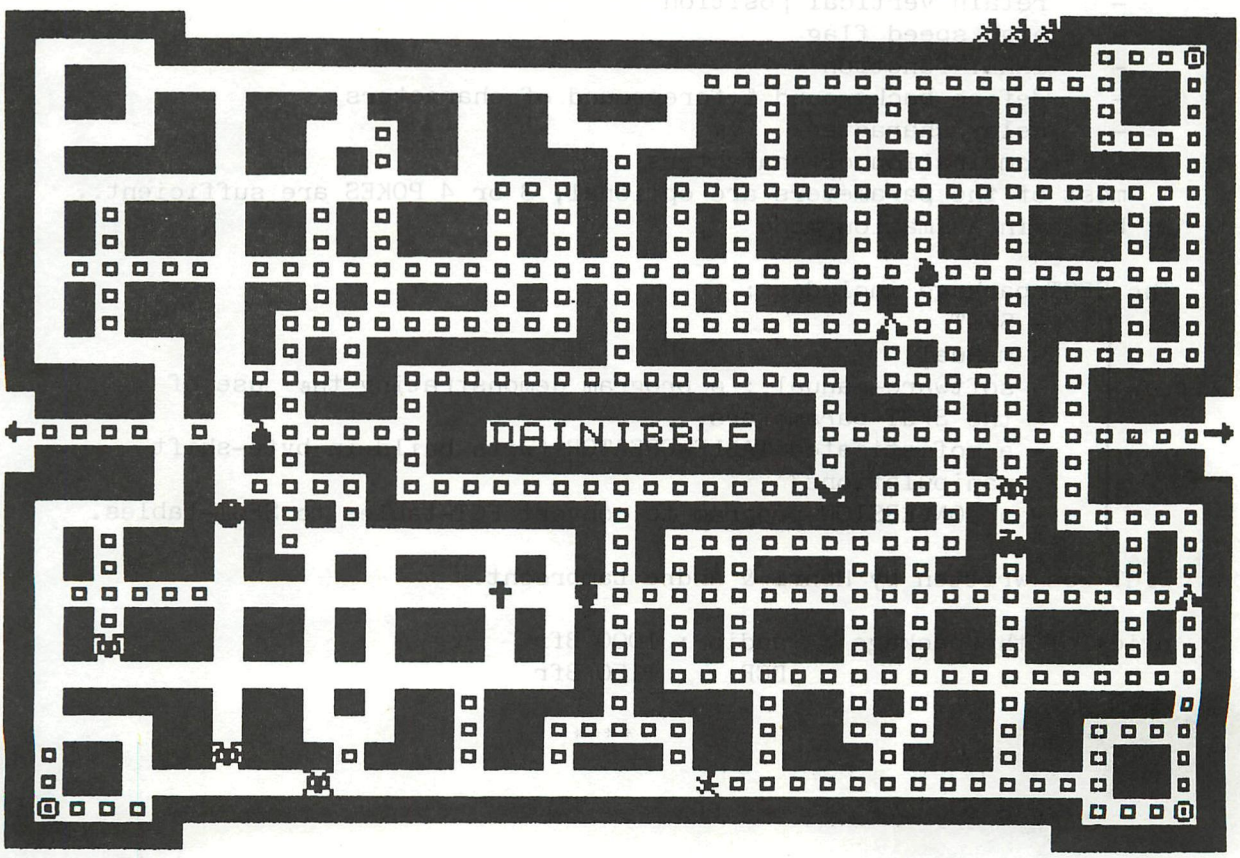
You can nibble the fruit that is moving around the maze, but you will have to nibble also dots to let the fruit appear. You get points for everything you nibble
Four monsters are trying to nibble you, but you can nibble them if a vitamine is nibbled by you and you are still extra strong because of that.

So, if you like it to nibble around, you better play DAINIBBLE.

This game is completely in machinelanguage, so it's as fast as you want it to be.

price : audio : 800 Bfr
DCR : 950 Bfr

****
* *
* Author Hendrik-Jan van Randen *
* *
****



SFGT

First there was FGT, with many facilities and medium speed, due to the restrictions of the DRAW-routine in DAI-BASIC. Then there was FFGT, high speed, but with restrictions of bit-access.
Now there is SFGT, a high-speed text-in-graphics program, offering the best of both previous programs.

NEW SOFTWARE

SPECIFICATIONS of SFGT :

* compatible for all graphic modes
* speed : up to 300 characters/second
* the same call-procedure as FGT : CALLM 800,A$
* up to 10 tables of characters in memory at the same time, choice of table with 1 POKE !
* parameters:
  - horizontal position
  - vertical position
  - horizontal step/character
  - vertical step/character
  - vertical step on OFF/SCREEN (no error)
  - horizontal step on OFF/SCREEN (no error)
  - left margin of screen WINDOW
  - right margin of screen WINDOW
  - top of screen WINDOW
  - bottom of screen WINDOW
  - matrix start
  - matrix lenght
  - string-start (implied MID$)
  - string-length (implied MID$)
  - up/down/left/right contruction of strings
  - retain horizontal position
  - retain vertical position
  - high speed flag
  - INKEY-function
  - define background & foreground of characters
  - delay/character
  - combination of characters
* most of the parameters are optional, 3 or 4 POKES are sufficient..
* build-in animation mode

The SFGT package includes :

- SFGT
- manual
- software-manual : a program demonstrating the use of the SFGT-parameters
- a sofisticated TABLE-CREATOR (with build-in byte-shift manipulations)
- a CONVERSION-program to convert FGT-tables to SFGT-tables.

SFGT was written by Henri & Andre Lambrecht.

price of the package : audio : 1000 Bfr
DCR : 1150 Bfr

*****

DAInamic DEBUGGING TOOL

NEW SOFTWARE

DDT USERS-GUIDEpage 01

Dainamic Debugging Tool (DDT) by W. COREMANS

1.DESCRIPTION :

DDT is a program ment for testing machine language programs
The functions this program can handle are :

1. Putting up to 255 (default = 100) breakpoints in a mlp (Machine Language Program). This is done in an edit-like way with the possibility to change, insert or delete breakpoints
2. Testing the mlp in 3 different modes :
   -A trace mode to test the mlp step by step
   -A breakpoint mode which runs your mlp in real time until a breakpoint is reached
   -A breakpoint mode which gives you also breaks at rom-entrypoints (ex. bankswitching)
3. Ability to change all the 8080 processor's registers and flags during testing
4. Full control of the stack. You can display the stack-contents and change it in a similar way as you do with breakpoints (i.e. change, insert or delete 16-bit data, returnaddresses or anything else which is on the stack)
5. Disassemble your mlp or the firmware in any rom-bank, with an option to display bytes after RST1, RST4 and RST5 as DATA
6. A number of options concerning the display of mnemonics and or registers, tracing RST instructions or not, disable/enable interrupts a.o.

DDT is partly written in BASIC ,partly in machine language. The machine language part is placed in 2 stringarrays which are loaded and relocated by the BASIC program. This means it has no fixed place in memory, so the mlp to be tested can be located anywhere

Running DDT is very easy : first load your mlp, eventualy adapt the heap pointer and at last LOAD and RUN DDT. When your mlp is located in higher ram addresses (ex. near to the screen-ram) you don't have to change the heap pointer, so your mlp will be located in the free ram space behind the BASIC program

Breakpoints can be set at will into the mlp you want to test, but be sure the breakpoint points to the begin of an 8080 instruction. If this is confusing then just keep in mind that you may specifie all addresses displayed in a disassembling run, exept when the address is followed by DATA

Special care has been taken concerning interrupt handling. Your mlp can change the interrupt mask at will or even reset the interrupt system, DDT will always find its way back.

For more specific information see the complete DDT USERS-GUIDE delivered with TOOLKIT 5

SUPER DCR-LIST

	NEWS 13-14-15		
	*** DCR 123 ***	FF	
	<<<<<<<<<< A >>>>>>>>>>	<<<<<<<<<< B >>>>>>>>>>	
	01-2Programs of Newsletters 13-14-15 (c) DAInamic	20-2Newsletters 13-14-15 (c) DAInamic 1983	
123	02-0DEMO C.POELS	21-0ASTERIX (cover of N14)	
	03-0MULTI PUZZLE C.POELS	22-0 CHARACTER INVASION N15 p101	
	04-0 EPROM-PROGRAMMER V3 G.KNOOPS	23-0 3-D : eenbladige omwentelingshyperboloide N15 p 129	
	05-1B000 B195 mlp EPROM-PROGRAMMER KNOOPS	24-0 BLOK IN REVERSE N15 p 137	
	06-1SOURCE EPROM-PROGRAMMER G. KNOOPS	25-0 SYMMETRISCHE WAAIERS N15 p 137	
	07-0 ROTATION ECRAN C.POELS	26-0 ZANDLOPER N15 p 137	
	08-0HISTOIRES	27-0 CUBE ROTATION AND TRANSLATING IN SPACE N15 p 134	
	09-0 SCHNELLER ANALOG DIGITAL WANDLER KOPP	28-2.	
	10-0 LOGIK ANALYSATOR KOPP		
	11-0 GROETEN UIT HAWAI - ALOHA OE.		
	12-0 LINEMODEBYTES & COLOURTYPEBYTES N.LOOIJE		
	13-0FGT-DEMO (FGT not included)		
123	14-0MODE 7 INITIALISTIE V0.0		
	15-0MODE 8 INITIALISTIE V0.0		
	16-0MODE 7 RELOCATABLE		
	17-0MODE 8 RELOCATABLE		
	18-03-D GENERATOR J.ROELANTS		
	19-0BASIC PAINT DEMO F.DRUIJ		

TOOLKIT 5

NEWS 13-14-15

DCR 123 A

NEW SOFTWARE

```
01-2-ARR-Programs of Newsletters 13-14-15 (c) DA
        Inamic
02-0-BAS-DEMO C.POELS
03-0-BAS-MULTI PUZZLE C.POELS
04-0-BAS- EPROM-PROGRAMMER V3 G.KNOOPS
05-1-OBJ-B000 B195 mlp EPROM-PROGRAMMER KNOOPS
06-1-OBJ-SOURCE EPROM-PROGRAMMER G. KNOOPS
07-0-BAS- ROTATION ECRAN C.POELS
08-0-BAS-HISTOIRES
09-0-BAS- SCHNELLER ANALOG DIGITAL WANDLER KOPP
10-0-BAS- LOGIK ANALYSATOR KOPP
11-0-BAS- GROETEN UIT HAWAI - ALOHA OE.
12-0-BAS- LINEMODEBYTES & COLOURTYPEBYTES N.LOOIJ
        E
13-0-BAS-FGT-DEMO (FGT not included)
14-#-DNA-MODE 7 INITIALISTIE V0.0
15-#-DNA-MODE 8 INITIALISTIE V0.0
16-0-BAS-MODE 7 RELOCATABLE
17-0-BAS-MODE 8 RELOCATABLE
18-0-BAS-3-D GENERATOR J.ROELANTS
19-0-BAS-BASIC PAINT DEMO F.DRUIJFF
```

update 83-05-10

NEWS 13-14-15

DCR 123 B

```
01-2-ARR-Newsletters 13-14-15 (c) DAInamic 1983
02-0-BAS-ASTERIX (cover of N14)
03-0-BAS- CHARACTER INVASION N15 p101
04-0-BAS- 3-D : eenbladige omwentelingshyperboloi
        de N15 p 129
05-0-BAS- BLOK IN REVERSE N15 p 137
06-0-BAS- SYMMETRISCHE WAAIERS N15 p 137
07-0-BAS- ZANDLOPER N15 p 137
08-0-BAS- CUBE ROTATION AND TRANSLATING IN SPACE
        N15 p 134
09-2-ARR-.
```

update 83-05-10

TIMING PROBLEMS

PROBLEMEN BIJ (KORTE) TIJDSMETING OP DE DAIpc

Bij het ontwikkelen van de BASICODE is het mij opgevallen dat de machine-instructietijden van de 8080 microprocessor niet kloppen. Bij uitzoeken bleek dit (volgens mij) de volgende oorzaak te hebben :
De computer is hardwarematig zo opgebouwd dat de microprocessor en de videoprocessor om de beurt gebruik maken van het RAM-geheugen. Echter op het moment van de rasterterugslag (dit is het moment dat de electronenstraal van rechtsonder naar linksboven in het beeld gaat), maakt de videoprocessor geen gebruik van de RAM en draait de microprocessor op volle toeren. Mogelijk treedt dit effect ook op bij de linesync (lijnterugslag). Effectief werkt de microprocessor op gemiddeld 1.17 Mhz. Normaal zult U hiervan weinig merken. Echter bij korte tijdmeting of frequentiemeting, zoals de ontvangst van BASICODE, telex of morse dan wel satellietontvangst, dan zal er niet die frequentie uitkomen die U berekend had. Bovendien zal er een interferentie ontstaan met de 50 Hz van de rasterterugslag (dit is geen brom o.i.d. van het lichtnet). De interferentie is wellicht te verminderen door grbruik te maken van subroutines in het ROMgeheugen (vb delays).
Het genoemde effect treedt nl niet op in de ROM, daar de videoprocessor hier geen gebruik van maakt. Bij tijdsmetingen van meer dan 50 à 100 mS zult U weinig last hebben van het genoemde effect omdat dan de rasterterugslag weggemiddeld wordt. Ik hoop hiermee enkele mensen slapeloze nachten bespaard te hebben.

met vriendelijke groeten,
Th. van Lieshout
Postgalei 5
1687 VP WOGNUM
NEDERLAND
tel 02297-2648

DAINIBBLE

After DAIPANIC,CENTIPEDE,ACROBATES, another 100% machine language game : high resolution (MODE 5) , very fast and very nibble ....

audio : 800 Bfr
DCR : 950 Bfr

SFGT

The very best of text-in-graphics , superspeed and all the options you want ...

audio : 1000 Bfr
DCR : 1150 Bfr

NEW SOFTWARE

TOOLKIT 5

* BASICODE II
* DAInamic Debugging Tool (see description in this issue)
* DAICOM communication program (see Newsletter 15)
* SUPER DCR-LIST (see sample printout in this issue)
* PAINT-routine in machine language (a very fast one)
* SOFTWARE PRINTER SPOOLER (see this issue)
* Gothic characters with FGT

audio : 1000 Bfr

DCR : 1150 Bfr

> > > > > PROGRAMMEERTECHNIEKEN < < < < <

Ik wilde deze keer weer eens de werksnelheid van een programma onder de loep nemen. Ik weet heus wel dat ook overzichtelijkheid en veranderbaarheid van een programma belangrijk zijn, vaak zelfs van doorslaggevend belang, maar over de overzichtelijkheid van een programma valt weinig te vertellen. Aanwijzingen kunnen in het algemeen wel gegeven worden maar detailzaken hangen te veel van het programma in kwestie af en hangen ook ten nauwste samen met de smaak van de betrokken programmeur. In snelheidszaken ligt dat anders: als ik een bepaalde methode voorstel kan iedereen zelf nagaan dat die methode sneller werkt dan een andere. Werkt hij niet sneller dan de door U gebruikte methode; ook dan is er geen probleem, ik kan vaststellen dat die methode beter (sneller) is en zal er dan in een volgend artikel op terugkomen. U kunt er natuurlijk ook zelf een artikel over schrijven.

Het eerste probleem is natuurlijk: Hoe test ik de snelheid van een bepaalde instructie of bepaald programmadeel ? Nu, hierbij kan de DAI ons zelf erg goed van dienst zijn. Het best is het om een apart timer-programma te maken dat indien gewenst een onderdeel kan zijn van het grote programma zolang dat nog in het ontwerpstadium is. Hieronder het geraamte van dit programma:

```
5     WAIT TIME 2:POKE #1BF,#FF:POKE #1BE,#FF
95    T1BE=PEEK(#1BE):T1BF=PEEK(#1BF):PRINT(#FFFF-T1BE-T1BF*256)/50.0
```

Iemand die programmeert met de goede gewoonte om de regels te nummeren met 10-vouden kan de regels 5 en 95 eenvoudig om het te testen deel zetten.
Toch behoeft dit programma enige toelichting. De variabelen T1BE en T1BF zijn natuurlijk integer. We maken gebruik van de 20msec klok die de waarde op #1BE en #1BF elke 20 milliseconde met 1 vermindert tot dat die nul is.
Met de POKE's uit regel 5 vullen we deze twee bytes met #FFFF ,eerst de high en dan de low byte; de WAIT TIME ervoor zorgt voor een schone start.
Aangezien ook de WAIT TIME instructie gebruik maakt van deze adressen mag in het te testen programma (deel) geen WAITTIME voorkomen. Andere WAIT's wel maar lijkt mij niet handig omdat de tijd dan door die WAIT's beinvloed wordt.
Het uitlezen van de adressen in regel 95 is het best in deze volgorde; bij andere volgorde kan de timing verkeerd gaan. Voorbeeld: #1BE staat op #35 en #1BF op #F6; als er geen aftelling is tussen de twee PEEK's zal de volgorde er niet toe doen, als er echter tussen de eerste en tweede PEEK in wel een vermindering plaatsvindt zal de uitkomst 20 msec kleiner zijn dus 50.10 ipv 50.12 seconden Geen erg grote fout misschien, maar onjuist; we moeten niet de snelheid beinvloeden met het timerprogramma zelf. Toch is er wel een risico aan verbonden : eens op de tweehonderdzesenvijftig keer staat #1BE op nul en wordt dan met een verminderd dan zal de klok niet 20 ms maar 256 * 20 ms is ruim vijf seconden verkeerd aangeven. Dit verschil is echter zo groot dat dat moet opvallen.
De deling M O E T door 50.0 en niet door 50 omdat we een fpt-uitkomst willen.
Als we dit niet doen kappen we alles naar beneden af op hele seconden.

We gaan wat experimenteren met deze timer en komen dan tot vermoedelijk voor velen opzienbarendende resultaten. Ik vermeld met opzet de resultaten niet. Geinterresseerden kunnen gemakkelijk zelf de tests uitvoeren en dan verrast door de resultaten zelf verder gaan proberen. Voor programmadelen die erg belangrijk zijn voor de snelheid zal het bijna altijd lonen een aantal mogelijkheden op snelheid te testen. Veel mensen zien echter niet dat er verschillende methoden zijn. Vandaar enige voorbeelden, waarbij de uitkomsten soms veel verschillen maar soms ook nauwelijks of niet.
Aangezien een enkele instructie zo snel gedaan wordt dat dit met deze methode niet te timen is, zullen we er een FOR - NEXT loop omheen zetten. Het timen van deze FOR - NEXT loop zelf.

```
      10    FOR I=1 TO 5000:NEXT
```

en doe die zowel na IMPFPT als na IMPINT.
daarna proberen we:

```
      10    FOR I=1 TO 10000 STEP 2:NEXT
```

en dit is toch ook een loze loop die 5000 maal doorlopen wordt ?

We gaan er eens iets tussen zetten b.v. A=3 of A=B. De tijden lopen echter zo op dat we de loop tot 1000 gaan beperken. En misschien wel tot 500 of 100 zeker als functies zoals SQR,SIN,COS,TAN en dergelijke gebruikt gaan worden.
Het is wel nuttig om een keer een loop zowel 1000 als 10000 maal te laten uitvoeren om eens te zien dat het inderdaad tien maal zolang duurt. Om redenen van praktische aard is een tijd tussen 10 en 30 seconden het meest aanbevelenswaardig. Korter maakt resultaat onbetrouwbaarder en langer kost te veel tijd. Daarbij deze tijdmeting kan maximaal een periode van #FFFF * 20 ms (dus ruim 20 minuten) meten.

We gaan verder experimenteren met een eenvoudige toewijzing: binnen de loop zetten we : A=8 of A=8.0 of A=B of A=B+4 of A=4+B
dit alles na IMPINT of na IMPFPT. Dit is op zich niet zo belangrijk maar vergelijkt U niet oude FPT-resultaten met nieuwe INT-resultaten. Pas op niet altijd is integer sneller! Gelooft U mij niet ? Probeer A%=SQR(P%) en A!=SQR(P!) maar.

We willen in A de waarde van twee B plus 4 krijgen. Vergelijkt U de volgende mogelijkheden eens.

```
A=2*B+4      erg voor de hand liggend niet waar?
A=B*2+4      anders ?
A=B+B+4      dit is sneller !!!
A=B+4+B of A=4+B+B
A=(B+B)+4    maakt dit iets uit ?
A=B+(B+4)    moet vergissing zijn.
A=(((((B)))))+((((B))))+(((4))) valt mee.
```

We gaan verder: A=B SHL 1 dit is sneller dan A=B*B maar pas op A=B SHL 5 is trager dan A=B*32. Voor de BASIC V1.1 bezitters A=-B of A=0-B
Probeert U ook eens het verschil tussen de gehele FOR - NEXT loop op een regel of op twee of drie regels, dat kan soms ook heel wat uitmaken.In principe altijd proberen de FOR en de NEXT op dezelfde regel te krijgen.

In het tweede deel wil ik wat dieper ingaan op de instructies die het programma op een andere plaats laten vervolgen. De instructies zijn GOTO , GOSUB , IF xx GOTO , IF xx THEN , ON W GOTO en ON W GOSUB.
Ik zal de instructie IF xx THEN gevolgd door een regelnummer nooit gebruiken. Ik vind een verplaatsopdracht zo belangrijk dat je dat duidelijk moet kunnen zien door een GOTO. De logica is voor mij ook: bij GOTO geef je duidelijk aan ergens anders te vervolgen; bij THEN zou men kunnen denken dat alleen die instructies die op de opgegeven regel staan dienen te worden uitgevoerd om daarna weer op het oude punt door te gaan. Dit is wel niet zo maar ik kies toch om de bovengenoemde logische redenen voor IF xx GOTO.

De GOTO is de meest vervloekte instructie in BASIC, vooral de tegenstanders van de italiaanse keuken zijn vaak erg fel. Met GOTO kun je namelijk heerlijke spagetti maken. De voorstanders vinden dit juist een voordeel evenals het totaal onleesbaar (begrijpelijk) worden van programma's. Dit alles onder het motto: 'Schaf nu space-invaders aan en U krijgt er een labyrint bij cadeau. Of zoals ik het zelf graag uitdruk : jump as jump can.

Ik geef even een voorbeeld dat ik reeds in vele inzendingen ben tegengekomen:

```
120     IF A=3 GOTO 140
130     GOTO 160
140     P=2
150     GOTO 160
160     END:REM of hopeless program
```

Natuurlijk werkt het. Deed het 't maar niet dan zou de maker misschien wat gaan nadenken. Als A gelijk aan drie is wordt P twee gemaakt en anders blijft P onveranderd. Maar dit kan toch wel een stuk beter:

```
120    IF A<>3 GOTO 160
140    P=2
160    END:REM of better program
```

Of als we het kort willen houden:

```
120    IF A<>3 GOTO 160:P=2
160    END:REM of shorter version
```

Of als we de GOTO willen vermijden

```
120    IF A=3 THEN P=2
```

Zelfs de IF kunnen we vermijden (ik beweer niet dat dit handiger is)

```
120    P=P+P*(SGN(ABS(A-3))-1)-2*(SGN(ABS(A-3))-1)
```

Gaat U deze regel maar eens na en probeert u ook eens de IF te vermijden, het is een zeer goede oefening.

In een vorig artikel is al eens uitgelegd waarom we sprongen naar hogere regelnummers (dwz ver naar achteren in de listing) moeten trachten te voorkomen in snelheid bepalende delen. Maar wist U dat de ON W GOTO xx sneller werkt dan de IF xx GOTO ? Voorbeeld IF PEEK(I)=1 GOTO 200 is trager dan ON PEEK(I) GOTO 200.

Niet altijd zal echter de te testen variabele geschikt zijn om met de ON W GOTO te gebruiken. Een extra berekening kan de winst dan weer geheel teniet doen. Controleer zeker aan het eind of er geen GOTO's te veel in een programma staan. Het staat zo dom om in regel 50 aan te treffen GOTO 170 en dan op 170 alleen GOTO 220 en dan op 220 alleen GOTO 30. Dit is echt geen fantasie; ik heb programma's (inderdaad meervoud) waar zoits in staat. Maar ik heb er ook begrip voor. Normaal zal niemand zoiets programmeren, maar op regel 170 stond bij een eerdere versie nog een andere instructie voor de GOTO. Later bleek die overbodig maar de GOTO niet. Om een logische programmavoortgang te krijgen keerde men alleen vanaf regel 220 terug naar het begin van het programma. Dus vanaf regel 170 naar regel 220 en niet naar regel 30. Nu beter te begrijpen maar nog steeds niet de snelste of overzichtelijkste versie.

Nu wat de GOSUB betreft. Ook hier tegenstrijdige belangen. De overzichtelijkheid van het programma is gebaat met een uitgang (RETURN) en een ingang per subroutine maar de snelheid kan soms gebaat zijn met meerdere in en uitgangen van een subroutine.
Voorbeeld:

```
70     GOSUB 2000
::
130    GOSUB 2010
::
2000   IF A=5 THEN RETURN
2010   P=2
2020   RETURN
```

Ik kies zelf in zulke gevallen bijna altijd voor 2000 IF A=5 GOTO 2020, dus laat ik overzichtelijkheid preveleren.
De instructie ON xx GOSUB kom ik zo zelden tegen dat ik me afvraag of de mensen hem wel kennen. Probeert U het gerust als de gelegenheid zich voordoet.

Elders in dit blad zult U een artikeltje van Just van Dunne' aantreffen over de GOTO. Wat hij daar uitlegt kan soms zeer handig zijn maar kost wel meer tijd dan de normale GOTO. Het voordeel is dan ook gelegen in gebruik in bijzondere gevallen. Ook zult U nog een klein (2 regels) programmaatje van mij kunnen vinden waar ik de timer om heen heb gezet. In dit programma offer ik echt alles op aan de snelheid en pas verschillende 'gemene' trucs toe om de snelheid te verhogen. Ondermeer laat ik een NEXT door twee verschillende FOR's gebruiken en of dit nog niet 'gemeen' genoeg is ook nog eens een FOR twee NEXTen gebruiken.

Verder zijn er reacties binnengekomen op het 'cirkel'programma van nummer 14. Meerderen hebben vastgesteld dat er snel ronde ellipsen getekend werden en geen echte cirkels. Dank voor uw bijdrage, die wij niet alle kunnen publiceren maar maar wel een ervan of een combinatie. Als uw bijdrage niet expliciet gebruikt is laat dit U er dan niet van weerhouden nogmaals in te zenden. Van alle inzendingen wordt nota genomen.

Succes met de timer en vindt U iets zeer opzienbarends laat het dan even weten.

Frank H. Druijff

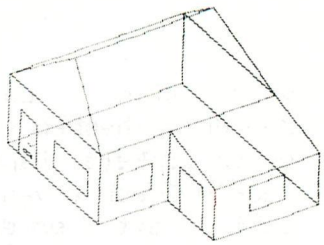
PROBLEM PROGRAM

Problem - program - problem - program - problem - program - problem - program

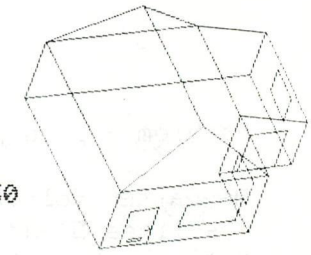
Hereafter follows a little 'basic'program of only two lines (10 and 20).
The lines 5 and 95 are used for timing purpose. Can you do it faster ?
But what does it do? I challenge you, try to understand what this program does before you type it in. But be careful it runs only in a clean memory. So if you have typed it in, put it on cassette, turn your machine off and switch it on again, load the program and run it. Isn't it fast ?

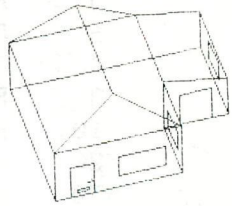
```
5     WAIT TIME 1:POKE #1BE,#FF:POKE #1BF,#FF:POKE #1BE,#FF
10    CURSOR 1,22:POKE #BF61,#32:POKE #131,1:FOR G=0 TO 14:ON PEEK(G+#32F0)
 GOTO 20:FOR I=(G+G+6)*G+#32F3 TO #34E1 STEP G+G+3:POKE I,1:NEXT
20    NEXT:FOR I=#32F0 TO #34E1:ON PEEK(I) GOTO 20:PRINT I+I-#65DD:POKE #7B
,0:NEXT
95    B=PEEK(#1BE):A=PEEK(#1BF):PRINT :PRINT (#FFFF-B-A*256)/50.0
100   REM ****** IMPINT ******
```

```
2     REM ***********************************************
3     REM ***    HARDCOPY SOURCE PASCAL           ******
4     REM ***    (c) ALAIN MARIATTE 1983          ******
5     REM *** pour imprimante serie,vitesse ajustable *
6     REM ***********************************************
7     REM
8     REM
10    REM LA COMMANDE P  DE TINY PASCAL NE PERMET PAS D'
20    REM UTILISER UNE IMPRIMANTE DONT LA VITESSE EST <> DE
30    REM 9600 Bauds.IL SUFFIT DE VIDER LE BUFFER D'EDITION
40    REM PAR CE COURT PROGRAMME BASIC,IMPLANTE A PARTIR DE
50    REM #8000, COEXISTANT AVEC PASCAL,POUR AVOIR L'EFFET
60    REM DESIRE.
70    REM
80    REM
90    REM
100   POKE #FFF5,#A0:REM VITESSE RS 232.ICI,4800 Bauds
110   POKE #131,0:REM OUVERTURE DU PORT RS 232
120   I=#2000-1
130   I=I+1:REM DEBUT DE L'EDIT BUFFER
140   A=PEEK(I):IF A=0 THEN POKE #131,1:END
150   A$=CHR$(A)
160   PRINT A$;
170   GOTO 130
```



ROTATING BUNGALOW

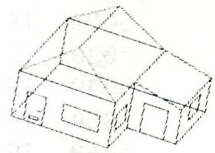


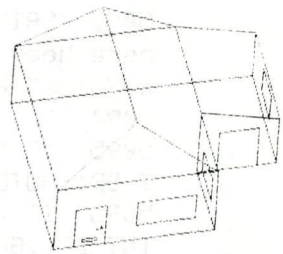


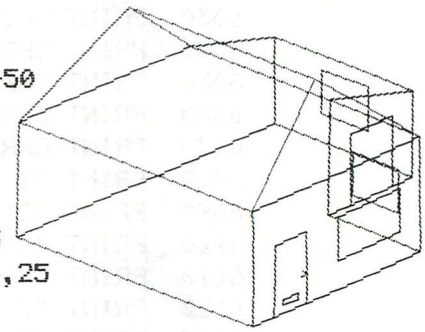
```
1     REM ***********************************
2     REM ***** 3DPLOT VERSIE 19.06.82 *****
3     REM *****  A.VINGERLING, R'DAM   *****
4     REM ***********************************
100   CLEAR 2750:PRINT CHR$(12);:GOSUB 7000:GOTO 103
101   COLORT 5 15 0 0:MODE 0:CURSOR 13,12
102   PRINT "INITIALISATIE, EVEN GEDULD SVP"
103   SF=1.0:GOSUB 2000:REM DATA INLEZEN
104   GOSUB 6000:REM TOON MOGELIJKHEDEN
120   COLORG 15 0 15 0:MODE 6:MODE 6
130   GOSUB 600:REM NIEUWE PLOT TEKENEN
140   GOSUB 810:REM KLEUR WIJZIGEN
150   GOSUB 700:REM OUDE PLOT WISSEN
160   Q%=1-Q%:KS=ABS(KS)
170   IF FL%=1.0 THEN 210
180   A%=GETC:GOSUB 880:IF A%<16 THEN 180
190   IF A%=ASC("H") THEN GOSUB 4000:GOTO 130
200   GOSUB 5000:ON P210% GOTO 210
205   IF A%=ASC("/") AND AA%=0 THEN 180
206   GOTO 230
210   P210%=0:IF AA%>0.0 THEN A%=AAA%(AA%):AA%=AA%-1:GOTO 230
220   AA%=0:FL%=0:GOTO 180
230   IF A%=ASC(".") THEN VF=1.0/0.8:GOTO 250:REM >
240   VF=0.8:REM <
250   IF A%=ASC("1") THEN A%=1:GOTO 280
260   IF A%=ASC("2") THEN A%=2:GOTO 280
270   IF A%<=19 THEN A%=A%-13
280   REM FOR P=1 TO NP:XX!(P)=X!(P):YY!(P)=Y!(P):NEXT
290   IF A%=ASC(",") OR A%=ASC(".") THEN GOSUB 900:GOTO 130
300   IF A%>=0 THEN ON A% GOTO 320,330,400,410,500,510
305   IF A%=ASC("C") THEN GOSUB 800:CF%=1-CF%:GOTO 130
310   GOTO 180
320   KS=-KS
330   FOR P%=1 TO NP%:X=X(P%):Y=Y(P%):X(P%)=X*KC+Y*KS:Y(P%)=Y*KC-X*KS:NEXT
340   GOTO 130
400   KS=-KS
410   FOR P%=1 TO NP%:Y=Y(P%):Z=Z(P%):Y(P%)=Y*KC+Z*KS:Z(P%)=Z*KC-Y*KS:NEXT
420   GOTO 130
500   KS=-KS
510   FOR P%=1 TO NP%:Z=Z(P%):X=X(P%):Z(P%)=Z*KC+X*KS:X(P%)=X*KC-Z*KS:NEXT
520   GOTO 130
600   REM NIEUWE PLOT TEKENEN
610   FOR L%=1 TO NL%:PA%=LA%(L%):PB%=LB%(L%)
620   DRAW X(PA%)+XC%,Y(PA%)+YC% X(PB%)+XC%,Y(PB%)+YC% 17+Q%*2
630   NEXT:RETURN
700   REM OUDE PLOT WISSEN
710   FILL 0,0 XMAX,YMAX 18-2*Q%:RETURN
800   IF CF%=0 THEN FILL 306,0 300,6 15:GOTO 802
801   FILL 306,0 300,6 0
802   RETURN
810   IF CF%=0 THEN COLORG 0 15*(1-Q%) 15*Q% 15:GOTO 812
811   COLORG 15 15*Q% 15-Q%*15 0
812   RETURN
880   QT%=QT%+1:IF QT%<15 THEN DRAW 305,1 300,6 15:DRAW 305,6 300,1 15:RETU
RN
881   DRAW 305,1 300,6 0:DRAW 305,6 300,1 0:IF QT%<30 THEN RETURN
882   QT%=0:RETURN
900   FOR QQ%=1 TO NP%:X(QQ%)=VF*X(QQ%):Y(QQ%)=VF*Y(QQ%):Z(QQ%)=VF*Z(QQ%):N
EXT:RETURN
2000  REM -- DATA INLEZEN --
2005  POKE #75,32:CURSOR 0,5:PRINT "data inlezen:";
2006  CURSOR 15,5:DPI=12.0:GOSUB 5200
2010  XC%=165:YC%=127:Q%=1
2020  READ NP%,NL%:PRINT 1.0
2030  DIM X(NP%),Y(NP%),Z(NP%)
```

```
2040  DIM XX(NP%),YY(NP%)
2050  DIM LA%(NL%),LB%(NL%),AAA%(100.0)
2060  FOR P%=1 TO NP%:READ X(P%),Y(P%),Z(P%)
2070  X(P%)=X(P%)*SF
2080  Y(P%)=(Y(P%)-20.0)*SF
2090  Z(P%)=Z(P%)*SF
2100  CURSOR 20,5:PRINT NP%-P%;:NEXT
2101  CURSOR 15,5:PRINT 0.0
2110  FOR L%=1 TO NL%:READ LA%(L%),LB%(L%)
2111  CURSOR 20,5:PRINT NL%-L%;:NEXT
2120  POKE #75,95:RETURN
4000  GOSUB 6000:MODE 6:FF=1.0:GOSUB 900:RETURN
5000  IF A%=47.0 AND AA%>0.0 THEN FL%=1:GOSUB 800:P210%=1:RETURN
5010  IF A%=47 AND AA%=0 THEN RETURN
5015  AA%=AA%+1
5020  IF A%=49 THEN AAA%(AA%)=50:GOSUB 800:RETURN
5030  IF A%=50 THEN AAA%(AA%)=49:GOSUB 800:RETURN
5040  IF A%=17 THEN AAA%(AA%)=16:GOSUB 800:RETURN
5050  IF A%=16 THEN AAA%(AA%)=17:GOSUB 800:RETURN
5060  IF A%=19 THEN AAA%(AA%)=18:GOSUB 800:RETURN
5065  IF A%=18 THEN AAA%(AA%)=19:GOSUB 800:RETURN
5070  IF A%=46 THEN AAA%(AA%)=44:GOSUB 800:RETURN
5080  IF A%=44 THEN AAA%(AA%)=46:GOSUB 800:RETURN
5081  IF A%=ASC("H") THEN MODE 0:GOSUB 6000:MODE 6:GOSUB 800:RETURN
5090  IF A%<>ASC("S") THEN RETURN
5091  DP=180.0/DPI:PRINT "Hoekverdraaing is nu";:CURSOR 20,3:PRINT 180.0/DP
I;"   (uitgangshoek =";180.0/DPI;" graden)"
5092  PRINT "grotere hoek : bovenste cursortoets + repeattoets":PRINT "klei
nere hoek: onderste cursortoets + repeattoets"
5093  A%=GETC:IF A%=0 THEN 5093
5094  IF A%=94 THEN MODE 6:A%=0:RETURN
5095  IF A%=16 THEN DP=DP+1.0:CURSOR 20,3:PRINT "        ";:CURSOR 20,3:PRIN
T DP;:GOTO 5093
5096  IF A%=17.0 THEN DP=DP-1.0:CURSOR 20,3:PRINT "        ";:CURSOR 20,3:PR
INT DP;:GOTO 5093
5097  IF A%<>13.0 OR DP=0.0 THEN 5093:DPI=180.0/DP
5098  MODE 6:GOSUB 5200:GOSUB 800:A%=0:RETURN
5200  PHI=PI/DPI:KS=SIN(PHI):KC=COS(PHI):RETURN
6000  COLORT 5 5 5 5:MODE 0:PRINT CHR$(12);:IF A%=ASC("H") THEN CURSOR 0,20
6010  PRINT "============================================================"
6011  PRINT "Dit programma plot een ruimtelijke figuur en maakt het"
6020  PRINT "mogelijk deze te wentelen, te vergroten, of te verkleinen"
6030  PRINT "m.b.v de cursortoetsen en met 1,2,<,> en /":PRINT
6040  PRINT CHR$(136);" : roteren naar links                +XZ"
6050  PRINT CHR$(137);" : roteren naar rechts               -XZ"
6060  PRINT CHR$(94);" : kantelen achterover               +YZ"
6070  PRINT CHR$(140);" : kantelen voorover                -YZ"
6080  PRINT "1 : wentelen met de klok mee          +XY"
6090  PRINT "2 : wentelen tegen de klok in         -XY"
6100  PRINT "> : vergroten (geen shift gebruiken)   +Y"
6110  PRINT "< : verkleinen(geen shift gebruiken)   -Y"
6120  PRINT "/ : alles in omgekeerde volgorde tot de oorspr. figuur"
6125  PRINT "H : HELProutine, toont de verschillende commando's"
6126  PRINT "S : Stapgrootte hoekverdraaing instellen m.b.v cursortoetsen"
6127  PRINT "C : Change; inverteert de kleuren"
6128  IF A%=ASC("H") THEN 6170
6129  PRINT "============================================================"
6130  PRINT "IN DE DATASET : NP= aantal knooppunten"
6140  PRINT SPC(16);"NL= aantal lijnen"
6150  PRINT SPC(16);"x1,y1,z1,x2,y2,z2,= knooppuntscoordinaten"
6160  PRINT SPC(16);"b1,e1,b2,e2,= begin/eind lijn 1,2, enz."
6170  PRINT "============================================================"
6180  PRINT "Het programma begint na aantippen van een toets";
6185  COLORT 5 15 0 0
6190  IF GETC=0.0 THEN 6190
6200  MODE 6:RETURN
```

```
7000  REM MELDING
7001  COLORT 5 5 5 5:MODE 0:PRINT CHR$(12);:CURSOR 0,15
7002  PRINT "00000   000000        000000    0        00000    0000000"
7003  PRINT "     0   0    0         0    0   0       0     0      0   "
7004  PRINT "     0   0    0         0    0   0       0     0      0   "
7005  PRINT " 0000    0    0  000    00000    0       0     0      0   "
7006  PRINT "     0   0    0         0        0       0     0      0   "
7007  PRINT "     0   0    0         0        0       0     0      0   "
7008  PRINT "00000   000000        000       0000000  00000      000   "
7009  COLORT 5 15 0 0:RETURN
7010  DATA 43,49
7020  REM x1,y1,z1,x2,y2,z2..knooppuntscoordinaten
7030  DATA -60,0,-50,20,0,-50,-60,0,50,-60,40,-50
7040  DATA 20,0,-10,20,40,-50,60,0,50,-60,40,50
7050  DATA -20,70,-40,60,0,-10,20,40,-10,20,40,50
7060  DATA 60,30,50,-20,70,40,60,30,-10
7070  DATA -50,0,-50,-50,25,-50,-35,25,-50,-35,0,-50
7080  DATA -20,10,-50,-20,25,-50,10,25,-50,10,10,-50
7090  DATA 20,10,-40,20,25,-40,20,25,-20,20,10,-20
7100  DATA 30,0,-10,30,25,-10,50,25,-10,50,0,-10
7110  DATA 60,10,10,60,25,10,60,25,30,60,10,30,0,0,0
7111  DATA -46,3,-50,      -46,5,-50, -39,5,-50, -39,3,-50
7112  DATA -40,12,-52, -37,12,-52, -37,12,-50
7120  REM b1,e1,b2,..begin- en eindpunt v.d.staven
7130  DATA 1,2,1,3,1,4,2,5,2,6,3,7,3,8,4,6
7140  DATA 4,8,4,9,5,10,5,11,6,9,6,12,7,10,7,13
7150  DATA 8,12,8,14,9,14,10,15,11,15,12,13,12,14,13,15
7160  DATA 16,17,17,18,18,19,20,21,21,22,22,23,23,20,24,25
7170  DATA 25,26,26,27,27,24,28,29,29,30,30,31,32,33
7180  DATA 33,34,34,35,35,32,36,36,37,38
7190  DATA 38,39,39,40,40,37,41,42,42,43
8000  REM NP,NL   aantal knooppunten/lijnen
8010  DATA 43,49
8020  REM x1,y1,z1,x2,y2,z2..knooppuntscoordinaten
8030  DATA -60,0,-50,20,0,-50,-60,0,50,-60,40,-50
8040  DATA 20,0,-10,20,40,-50,60,0,50,-60,40,50
8050  DATA -20,70,-40,60,0,-10,20,40,-10,20,40,50
8060  DATA 60,30,50,-20,70,40,60,30,-10
8070  DATA -50,0,-50,-50,25,-50,-35,25,-50,-35,0,-50
8080  DATA -20,10,-50,-20,25,-50,10,25,-50,10,10,-50
8090  DATA 20,10,-40,20,25,-40,20,25,-20,20,10,-20
8100  DATA 30,0,-10,30,25,-10,50,25,-10,50,0,-10
8110  DATA 60,10,10,60,25,10,60,25,30,60,10,30,0,0,0
8111  DATA -46,3,-50,      -46,5,-50, -39,5,-50, -39,3,-50
8112  DATA -40,12,-52, -37,12,-52, -37,12,-50
8120  REM b1,e1,b2,..begin- en eindpunt v.d.staven
8130  DATA 1,2,1,3,1,4,2,5,2,6,3,7,3,8,4,6
8140  DATA 4,8,4,9,5,10,5,11,6,9,6,12,7,10,7,13
8150  DATA 8,12,8,14,9,14,10,15,11,15,12,13,12,14,13,15
8160  DATA 16,17,17,18,18,19,20,21,21,22,22,23,23,20,24,25
8170  DATA 25,26,26,27,27,24,28,29,29,30,30,31,32,33
8180  DATA 33,34,34,35,35,32,36,36,37,38
8190  DATA 38,39,39,40,40,37,41,42,42,43
9999  END
10000 REM MELDING
10001 COLORT 5 5 5 5:MODE 0:PRINT CHR$(12);:CURSOR 0,15
10002 PRINT "00000   000000        000000    0        00000    0000000"
10003 PRINT "     0   0    0         0    0   0       0     0      0   "
10004 PRINT "     0   0    0         0    0   0       0     0      0   "
10005 PRINT " 0000    0    0  000    00000    0       0     0      0   "
10006 PRINT "     0   0    0         0        0       0     0      0   "
10007 PRINT "     0   0    0         0        0       0     0      0   "
10008 PRINT "00000   000000        000       0000000  00000      000   "
10009 COLORT 5 15 0 0:RETURN
```







FGT-DEMO

```
10    GOTO 1500
80    REM ..............................................
100   REM .     Here is a small demonstration program     .
110   REM .     which should hopefully solve your FGT     .
120   REM .     problems. Read through the listing        .
130   REM .     carefully after observing what happens    .
140   REM .     on the screen. You should then be able    .
150   REM .     to duplicate the effects in your own      .
160   REM .     programs. The subroutine from lines       .
170   REM .     1030-1050 should be used in all your      .
180   REM .     BASIC programs that use FGT. You may      .
190   REM .     of course compress the lines by using     .
200   REM .     colons.                                   .
210   REM .                              Dave Atherton    .
220   REM ..............................................
1001  REM FGT BASIC CONTROL SUBROUTINE
1002  REM CC=TEXT COLOUR (0-15)
1003  REM X=X START POSN (0-335):Y=Y START POSN (0-256)
1004  REM SP=SPACING BETWEEN CHARACTERS(0-255)
1005  REM DF=DIMENSION OF CHARACTERS(0-15)
1006  REM VF=TEXT PRINTED VERTICALLY(1) OR NOT(0)
1007  REM ZF=LEFT TO RIGHT(0) OR RIGHT TO LEFT(1)
1008  REM PF=NEW X,Y USED(0),EXISTING POSN USED NEW XY IGNORED(1)
1009  REM T$=TEXT PRINTED (UP TO 255 CHARACTERS)
1010  REM FF=BACKGROUND FILLED IN (1) OR NOT (0)
1011  REM FC=FILLCOLOUR FOR BACKGROUND (0-15)
1012  REM ID=VERTICAL SCALE(0-255)
1030  POKE #2F0,FC%*#40+CC%
1035  POKE #2F1,FF%*#80+PF%*#40+ZF%*#20+VF%*#10+DF%
1040  POKE #2F2,X% MOD 256
1041  POKE #2F3,X%/256
1042  POKE #2F4,Y%
1045  POKE #2F5,SP%
1047  POKE #2F6,ID%
1050  CALLM #300,T$:RETURN
1500  MODE 0:PRINT CHR$(12):CLEAR 1000
1510  DIM A$(3):FOR AA%=0 TO 2:READ A$(AA%):NEXT
1512  DATA "THIS SHOWS THE","FGT PROGRAM","WITH THE"
1530  GOSUB 2000:SOUND 0 0 15 0 FREQ(600):WAIT TIME 10:SOUND OFF
1550  GET%=GETC:IF GET%=0 THEN 1550:END
2000  REM FGT PARAMETERS
2010  X%=20:Y%=130:SP%=9:FC%=1:CC%=0:DF%=0:ID%=15:PF%=0:ZF%=0:FF%=0:ST%=15:
VF%=0
2013  COLORG 8 15 5 2
2014  A$(3)="STANDARD CONDITIONS":GOSUB 3000
2015  VF%=1:A$(3)="VERTICAL FLAG SET":GOSUB 3000:VF%=0
2016  DF%=1:A$(3)="DIMENSIONS ALTERED":GOSUB 3000:DF%=0
2017  ZF%=1:X%=200:Y%=10:A$(3)="DIRECTION FLAG SET":GOSUB 3000:ZF%=0
2018  SP%=18:A$(3)="SPACING ALTERED":GOSUB 3000:SP%=9
2019  PF%=1:A$(3)="PRINT FLAG SET":GOSUB 3000:PF%=0
2020  CC%=RND(16):A$(3)="COLOUR ALTERED":GOSUB 3000:CC%=0
2021  FC%=7:A$(3)="FILLCOLOUR UNSET":GOSUB 3000:FC%=1
2022  ST%=10:A$(3)="LINE SPACING ALTERED":GOSUB 3000:ST%=15
2023  FF%=1:A$(3)="FF FLAG SET":GOSUB 3000:FF%=0
2024  ID%=180:A$(3)="ID ALTERED":GOSUB 3000:ID%=15
2070  REM WITH THIS YOU SHOULD BE ABLE TO DO YOUR OWN FGT
2080  RETURN
3000  MODE 5
3005  FOR F%=X% TO X%+3*ST% STEP ST%
3010  T$=A$(S%):S%=S%+1:Y%=Y%-ST%*(-ZF%*2+1)
3030  GOSUB 1030:NEXT:WAIT TIME 50:S%=0
3060  X%=20:Y%=130:RETURN
```

SORTING DEMO

```
1     CLEAR 1000:DIM S$(10.0),A$(10.0)
10    REM SORTEERDEMO
15    COLORT 8 0 8 8:RF$="N"
19    POKE #75,32
20    GOSUB 1500
30    PRINT "           # VISISORT #"
31    PRINT TAB(20);"bert maertens ":FOR SX!=1.0 TO 37.0:A$=A$+CHR$(64):NEXT:PRI
NT A$
32    A$=""
40    PRINT :PRINT "       1) BUBBLE SORT":PRINT "       2) VERTRAAGD SORTEREN"
50    PRINT "       3) SHELL-METZNER SORT":PRINT "       4) EINDE PROGRAMMA"
80    PRINT :PRINT "Uw keuze :"
81    CH!=GETC:IF CH!=0.0 THEN 82
82    CH!=CH!-48.0
90    IF CH!<1.0 OR CH!>4.0 OR CH!<>INT(CH!) THEN 81
100   IF CH!=4.0 THEN PRINT CHR$(12):END
110   GOSUB 1500:GOSUB 160
120   Y!=22.0:FOR K=1 TO NS:A$(K)=S$(K):CURSOR 2,Y!:PRINT A$(K):Y!=Y!-1.0:NEXT
130   NE=0:NC=0:CURSOR 20,22:PRINT "# ITEMS =";NS:CURSOR 20,21:PRINT "# VERGELIJ
KEN =";NC:CURSOR 20,20:PRINT "# UITWISSELEN =";NE
140   ON CH! GOSUB 310,400,540
150   CURSOR 0,12:PRINT "LIJST OPNIEUW SORTEREN <J/N>"
151   RF!=GETC:IF RF!=0.0 THEN 151
152   RF$=CHR$(RF!):GOTO 20
160   IF LEFT$(RF$,1)="J" THEN 250
170   NS$="10":PRINT "Hoeveel items sorteren (MAX 9)";
171   NS!=GETC:IF NS!=0.0 THEN 171
172   NS=NS!-48.0:IF NS<2.0 OR NS>10.0 OR NS<>INT(NS) THEN 171
173   PRINT NS
180   F1$="C":PRINT "Door (G)ebruiker of (C)omputer"
181   R1!=GETC:IF R1!=0.0 THEN 181
182   R1$=CHR$(R1!)
190   IF LEFT$(R1$,1)="G" THEN 240
200   R$="N":PRINT "(N)ummers of (L)etters"
201   R!=GETC:IF R!=0.0 THEN 201
202   R$=CHR$(R!)
203   IF R$="L" OR R$="N" THEN 210
204   GOTO 201
210   FOR K=1 TO NS
220   IF LEFT$(R$,1)="L" THEN S$(K)=CHR$(RND(26.0)+65.0)
222   IF LEFT$(R$,1)="N" THEN S$(K)=CHR$(RND(10.0)+48.0)
230   NEXT:GOTO 250
240   GOSUB 1500:PRINT "maximaal 1 teken per item":FOR K=1 TO NS:PRINT "ITEM #";
K;" = ";
241   S!=GETC:IF S!=0.0 THEN 241
242   S$(K)=CHR$(S!):PRINT S$(K):NEXT
250   RF$="N":GOSUB 1500
260   RETURN
270   CURSOR 0,12:PRINT "duw een toets om verder te gaan"
271   IF GETC=0.0 THEN 271
290   CURSOR 0,12:PRINT SPC(31)
300   RETURN
310   CURSOR 9,23:PRINT "-BUBBLE SORT-":GOSUB 270
320   FOR I=1 TO NS-1
330   FOR J=I+1 TO NS
340   X=I:Y=J:GOSUB 710:IF A$(I)<=A$(J) THEN 360
350   GOSUB 750
360   NEXT J
370   NEXT I
380   CURSOR 0,13:PRINT " SORTEREN BEEINDIGD....":GOSUB 270
390   RETURN
400   CURSOR 3,23:PRINT "  -VERTRAAGDE SORTERING-":GOSUB 270
410   J=0:R=0:I=0
420   I=I+1
```

```
430   IF I=NS THEN 520
440   J=I:R=J+1
450   X=J:Y=R:GOSUB 710:IF A$(R)>=A$(J) THEN 470
460   J=R
470   R=R+1
480   IF R<=NS THEN 450
490   IF I=J THEN 420
500   GOSUB 750
510   GOTO 420
520   CURSOR 0,13:PRINT "SORTEREN BEEINDIGD...":GOSUB 270
530   RETURN
540   CURSOR 7,23:PRINT "-SHELL-METZNER SORT-":GOSUB 270
550   M=NS
560   M=INT(M/2.0)
570   IF M=0.0 THEN 690
580   P=NS-M
590   H=1
600   I=H
610   J=I+M
620   X=I:Y=J:GOSUB 710:IF A$(I)<=A$(J) THEN 660
630   GOSUB 750
640   I=I-M
650   IF I>=1 THEN 610
660   H=H+1
670   IF H>P THEN 560
680   GOTO 600
690   CURSOR 0,13:PRINT "SORTEREN BEEINDIGD....":GOSUB 270
700   RETURN
710   NC=NC+1:CURSOR 35,21:PRINT NC
720   CURSOR 5,23-X:PRINT CHR$(136):CURSOR 5,23-Y:PRINT CHR$(136):CURSOR 0,0:WAI
T TIME 50
730   CURSOR 5,23-X:PRINT " ":CURSOR 5,23-Y:PRINT " "
740   RETURN
750   FOR K=2 TO 8 STEP 2
760   CURSOR K,23-I:PRINT " ":CURSOR K+2,23-I:PRINT A$(I)
770   CURSOR K,23-J:PRINT " ":CURSOR K+2,23-J:PRINT A$(J)
780   WAIT TIME 5:NEXT K
790   CURSOR 10,23-I:PRINT " ":CURSOR 12,23-I:PRINT A$(I):DF=J-I
800   FOR K=1 TO DF
810   CURSOR 12,23-(I+K-1):PRINT " ":CURSOR 12,23-(I+K):PRINT A$(I)
820   CURSOR 0,0:WAIT TIME 20
830   CURSOR 10,23-(J-K+1):PRINT " ":CURSOR 10,23-(J-K):PRINT A$(J)
840   NEXT K
850   FOR K=12 TO 4 STEP -2
860   CURSOR K,23-J:PRINT " ":CURSOR (K-2),23-J:PRINT A$(I)
870   CURSOR K,23-I:PRINT " ":CURSOR (K-2),23-I:PRINT A$(J)
880   WAIT TIME 5:NEXT K
890   NE=NE+1:CURSOR 35,20:PRINT NE
900   TEMP$=A$(I):A$(I)=A$(J):A$(J)=TEMP$
910   RETURN
1499  GOTO 1499
1500  MODE 0:PRINT CHR$(12):FOR X!=#BFEF TO #BA2D STEP -#86:POKE X!,#6A:NEXT:RET
URN
```

DAI VIDEO HARDWARE

DAI - VIDEO hardware: messcherpe plaatjes in zwart-wit!

Aangezien het werken met de DAI in combinatie met een gewone TV een zeer vermoeiende bezigheid bleek, heb ik me nogal wat moeite getroost hierin verbetering te brengen.

De problemen waar het hier om gaat treden vooral op bij het verwerken van tekst (ontwikkelen van programma's). Als hoofdschuldige moet de karaktergrootte worden aangemerkt: 64 tekens per regel is gewoon teveel voor een normale TV. Verder bleken die karakters niet zuiver stil te staan. Bovendien hadden de letters last van allerlei "rafelige" uitwaaiers vooral bij het gebruik van verkeerde kleuren. Wie een beeld voltikt met dezelfde letters kan 'genieten' van een diagonaal en lopend strepenpatroon. Het totaal geeft ongeveer de indruk van een plaatje waar een watergordijn voor langs stroomt.

Overigens: bij een spelletje 'Spaceinvader' heb ik geen enkel probleem met de beeldkwaliteit. Vooral bij het ontwikkelen van programma's is echter uiterste concentratie vereist. De extra belasting van de hierboven beschreven matige beeldkwaliteit zal m.i. elke serieuze programmeur spoedig de hoop doen opgeven: hij/zij ziet letterlijk door de bomen het bos niet meer !!!

Om een lang verhaal kort te maken: het is uiteindelijk gelukt een zeer bevredigende èn betaalbare beeldkwaliteit te krijgen m.b.v. een zwart-wit monitor en enige eenvoudige wijzigingen aan de DAI-print. Voordat ik die ga beschrijven wil ik nog graag in het kort mijn niet-succesvolle pogingen verwoorden:

-K.T.V. gebruiken als monitor. Ik had tot mijn beschikking een Blaupunkt met FM 121 chassis. Op het schema vond ik het IC TDA 3300. Deze heeft extra RGB-aansluitingen (pen 24, 25, 26), met pen 23 als vermoedelijke schakelaar.
Tot op heden heb ik echter niemand kunnen vinden die me op dit punt verder kan helpen (ik ben geen electronicus); ik ben er echter bijna zeker van dat via dit IC een eenvoudige RGB-aansluiting te realiseren moet zijn.
In dit verband denk ik dat er vele computerhobbyisten zijn die hun K.T.V. van een monitor aansluiting willen voorzien: een gat in de markt dus. Wie zich erin wil werpen kan ik in ieder geval helpen aan het FM 121 schema...

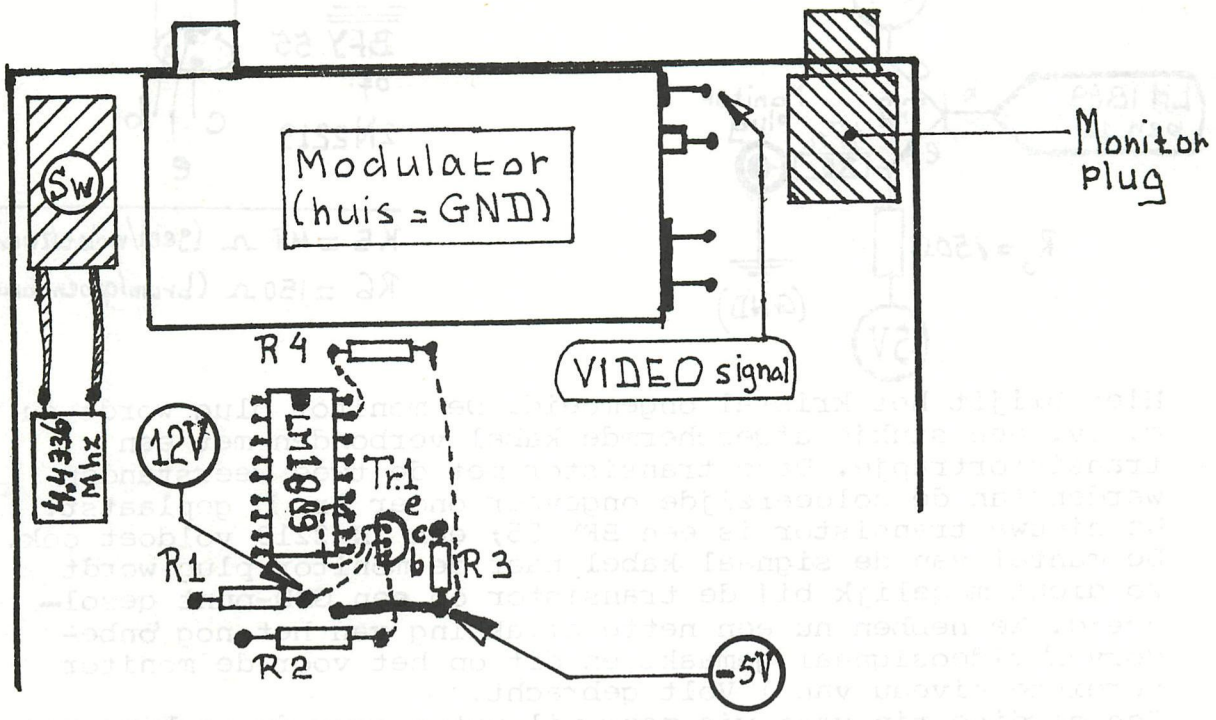
-DAI-video kaart verbeteren.
Als voorbeeld hiervoor gebruikte ik het verhaal van Anton Doornenbal (DAInamic maart/juni 1982). Als eerste werd de ZNA 134 aansluiting gewijzigd (interliniering). Dat gaf een kleine verbetering.
Vervolgens de zaken die verband houden met bandbreedte aangepakt (kleur en helderheid). Het bleek dat kleur- en helderheidsbandbreedte niet afzonderlijk konden worden gewijzigd. Hoewel ik me bij Anton persoonlijk heb kunnen overtuigen van de prima beeldkwaliteit van zijn installatie, ben ik van verbetering in mijn eigen geval niet zo zeker. De twijfel is ook weer niet zo groot dat ik de zaak weer in de oorspronkelijke toestand terugbreng...

-Gebruik zwart-wit monitor. Een demonstratie bij een handelaar, waarbij het signaal afgetakt werd van de modulator leverde niet het monitor beeld zoals ik dat kende van andere computers.

Over het beeld liep een diagonaal patroon van lijnen, welke door contrastverhoging enigszins te onderdrukken waren. Hoewel men probeerde de indruk te wekken dat dit ècht normaal en ik een beetje abnormaal was ging de koop dus niet door... Toch kocht ik later tweedehands zo'n kubus uit Hong-Kong; voor honderd gulden: dat kùn je niet weigeren!
Teneinde het geheel goed te laten samenwerken ging ik te rade bij Anton Doornenbal. De hierna weergegeven zwart-wit monitoraansluitingen komen dan ook van zijn hand. Beide werken ze perfect. Hoewel tevens de eerder genoemde wijzigingen interliniering/helderheid/kleur zijn doorgevoerd zijn die toch van ondergeschikt belang.
Het lopende diagonaallijnenpatroon bleek te worden veroorzaakt door een kleurpuls welke de kleurenontvanger doet omschakelen op kleurenweergave. Dit signaal is bij een zwart-wit monitor uiteraard overbodig, ja blijkt zelfs uitermate storend te werken.

ZWART-WIT MONITOR AANSLUITINGEN

Onderstaande figuur toont de PAL-videokaart met enkele van de belangrijkste componenten. Ik hoop dat ook de minder ervaren knutselaars hiermee uit de voeten kunnen.

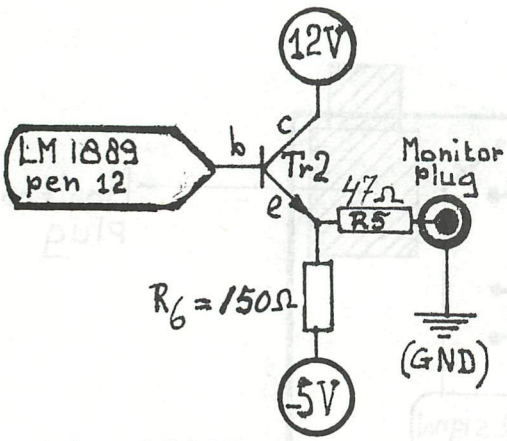


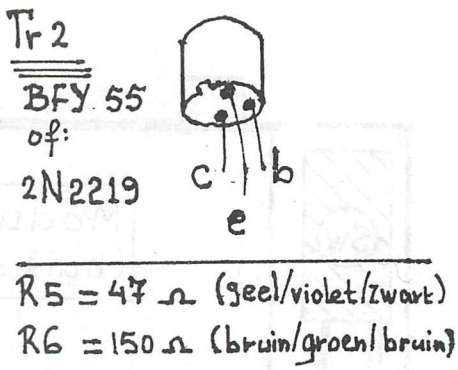
R1 = 820 Ω (grijs/rood/bruin)
R2 = 1k5 (bruin/groen/rood)
R3 = 1 kΩ (bruin/zwart/rood)
R4 = 2k7 (rood/violet/rood)
Tr1 = 2N 3704

1-SUPERSIMPEL-aansluiting

Installeer de monitorplug op de (gearceerd) aangegeven plaats. Verbind het modulatorhuis (=GND) met de buiten kant van de plug. De binnenkant van de plug wordt verbonden met het punt "VIDEOsignal". Dit is de 'normale' monitoraansluiting.
De gewraakte kleurpuls is gesitueerd rond het kristal 4,4336 Mhz. De puls wordt effectief gesmoord door dit kristal eenvoudigweg kort te sluiten. Maak de bedrading absoluut niet langer dan nodig is, daar het kristal anders zou kunnen worden beschadigd!
In de figuur is de kortsluiting uitschakelbaar gemaakt d.m.v. een lampdrukknopschakelaartje (Sw). Via een gaatje in het deksel van de DAI kan deze m.b.v. een potlood o.i.d. bediend worden. K.T.V. en zw-monitor mogen ook samen gebruikt worden. Echter met de schakelaar geopend is de kleurpuls aanwezig, en dus de monitor weergave niet optimaal; met gesloten schakelaar verdwijnt de kleurpuls en levert de K.T.V. slechts zwart-wit beelden. Dit is dus een nadeel.
De volgende oplossing laat volledig onafhankelijk gebruik van K.T.V. en zw-monitor toe.

2-SUPERDELUX-aansluiting





Hier blijft het kristal ongemoeid. De monitor plug wordt nu m.b.v. een stukje afgeschermde kabel verbonden met een transistortrapje. Deze transistor met de twee weerstanden werden aan de soldeerzijde ongeveer onder Tr. 1 geplaatst. De nieuwe transistor is een BFY 55; een 2N 2219 voldoet ook. De mantel van de signaal kabel naar de monitor plug wordt zo dicht mogelijk bij de transistor op een GND-punt gesoldeerd. We hebben nu een nette aftakking van het nog 'onbedorven' videosignaal gemaakt en dit op het voor de monitor vereiste niveau van 1 Volt gebracht.
Een aardige tip voor wie meer wil weten over de werking van een dergelijke videokaart: kijk eens in Elektuur nr 231 (jan 1983). Hierin wordt een VideoAudioModulator beschreven, waarin de LM1889 is gebruikt.

Geert Hospers (met dank aan A. Doornenbal)
Hoffmannlaan 555
5011 WL Tilburg (NL) tel.013-561844

CONTINUATION LINES

SCREEN CONTINUATION LINES

The DAI monitorprogram enables the use of extended lines on the screen. Up to 3 extended lines are possible; each line is indented 6 spaces and a continuation 'C' is placed at its beginning.

Mr. Markus Sigg, Worblingen, Germany, designed the following ML routine to manipulate with these extended lines.
It enables the use of more than 3 extended lines, it cancels the continuation 'C' and it enables the use of more or less spaces on the extended lines.

```
INIT        PUSH H
            LXI H,START        addr. alternative routine
            SHLD :006C         load RST5 vectoraddr.
            POP H
            RET

START       PUSH H             On entry, A contains last char
            PUSH PSW
            CPI :0D            Car.ret?
            JZ READY           Then abort
            LDA :0072          Get lsbyte cursorpos.
            LHLD :007A         Get lsbyte last pos on line
            CMP L              End of line reached?
            JNZ READY          Abort when not
            CALL :DD5E         Else: print car.ret

SPACE       MVI A,:20          Load Ascii 'space'
            MVI H,:06          Nr. of indented spaces
OUTC        RST5               Output to screen
            DATA :03
            DCR H
            JNZ OUTC           Next space when not ready

READY       POP PSW
            POP H
            JMP :C6FD          Perform original RST5
```

The INIT routine must be performed once at the beginning of the program. It replaces the RST5 vector address on #006C/D with with the address of the alternative routine. [This can also be done under Utility with V5=START].

The rest of the routine will be runned by the DAI monitor program every time a RST5 (screen handling) routine is performed.
If no indent is required, the part 'SPACE' can be skipped.
The number of spaces on the indented lines can be changed by changing the contents of the H-register in the 'SPACE'-routine.
After this routine is performed, the original RST5 routine on address #C6FD is performed.

This alternative routine has no influence on the RS232 output.

INIT E5 21 00 50 22 6C 00 E1 C9

Jan Boerrigter

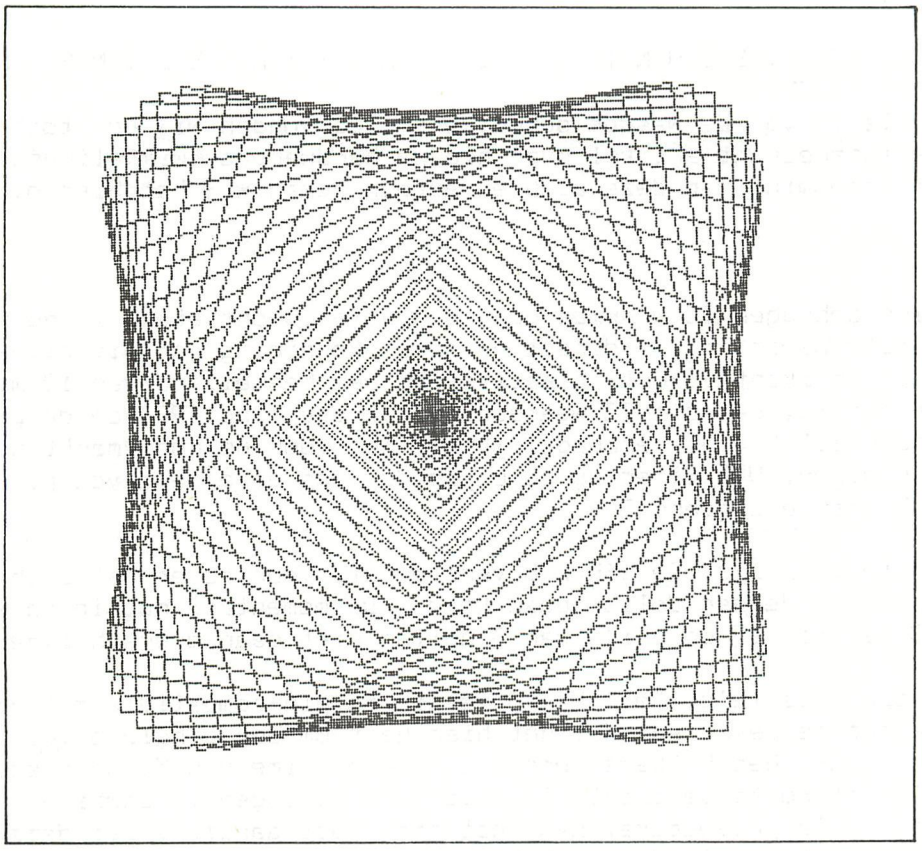
5000 E5 F5 FE 0D CA 20 50 3A 72 00 2A 7A 00 B0 C2 20
5010 50 CD 5E DD 3E 20 26 06 EF 03 25 C2 18 50
5020 F1 E1 C3 FD C6

program identification

title	:	SCREEN TO BUFFER / BUFFER TO SCREEN
author	:	W.Hermans
purpose	:	To hold an entire picture in buffer,and restore it
comment	:	actual program is for MODE 3/4, could be adapted for other MODES(lines 7/8/9 & 28/29/30)

```
001                         ORG   :D00
002                  *MOVE SCREEN TO BUFFER
003 0D00 E5                 PUSH  H
004 0D01 C5                 PUSH  B
005 0D02 D5                 PUSH  D
006 0D03 F5                 PUSH  PSW
007 0D04 112E17             LXI   D,5934     : TIMES TO MOVE
008 0D07 21EFBF             LXI   H,:BFEF    : SOURCE
009 0D0A 010090             LXI   B,:9000    : DESTINATION
010 0D0D 7E           CONT  MOV   A,M
011 0D0E 02                 STAX  B
012 0D0F 2B                 DCX   H
013 0D10 0B                 DCX   B
014 0D11 1B                 DCX   D
015 0D12 7A                 MOV   A,D
016 0D13 B3                 ORA   E
017 0D14 C20D0D             JNZ   CONT
018 0D17 F1                 POP   PSW
019 0D18 D1                 POP   D
020 0D19 C1                 POP   B
021 0D1A E1                 POP   H
022 0D1B C9                 RET
023                  *MOVE  BUFFER TO SCREEN
024 0D1C E5                 PUSH  H
025 0D1D C5                 PUSH  B
026 0D1E D5                 PUSH  D
027 0D1F F5                 PUSH  PSW
028 0D20 112E17             LXI   D,5934     : TIMES TO MOVE
029 0D23 210090             LXI   H,:9000    :SOURCE
030 0D26 01EFBF             LXI   B,:BFEF    :DESTINATION
031 0D29 7E           CONT2 MOV   A,M
032 0D2A 02                 STAX  B
033 0D2B 2B                 DCX   H
034 0D2C 0B                 DCX   B
035 0D2D 1B                 DCX   D
036 0D2E 7A                 MOV   A,D
037 0D2F B3                 ORA   E
038 0D30 C2290D             JNZ   CONT2
039 0D33 F1                 POP   PSW
040 0D34 D1                 POP   D
041 0D35 C1                 POP   B
042 0D36 E1                 POP   H
043 0D37 C9                 RET
044 0D38                    END
```

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

CONT   0D0D   CONT2  0D29
```



2-D TRANSFORMATIONS

```
10    REM 2D TRANSFORMATIONS.
20    CLEAR 256:MODE 0
50    DIM CO!(5.0,2.0):REM CO(i,1)=Xi, CO(i,2)=Yi
51    FOR I=1 TO 5:FOR J=1 TO 2:READ CO!(I,J):NEXT:NEXT
52    DATA 1,1,-1,1,-1,-1,1,-1,0,0
60    S!=1.318:REM .....................SCALING
62    R!=2.82:RAD!=180.0/PI
63    CS!=COS(R!/RAD!):SN!=SIN(R!/RAD!):REM ..ROTATION
64    XT!=168.0:YT!=128.0:REM ............TRANSLATION
100   COLORG 8 0 5 15:MODE 6:SF!=(-1E-2)
110   FOR N=1 TO 64
120   REM PLOT
122   CO!(5.0,1.0)=CO!(1.0,1.0):CO!(5.0,2.0)=CO!(1.0,2.0)
124   FOR I=1 TO 4:J=I+1
126   DRAW CO!(I,1.0)+XT!,CO!(I,2.0)+YT! CO!(J,1.0)+XT!,CO!(J,2.0)+YT! 0:NE
XT
130   REM ROTATION
131   FOR I=1 TO 4
132   X1!=CO!(I,1.0)*CS!+CO!(I,2.0)*SN!
133   Y1!=CO!(I,2.0)*CS!-CO!(I,1.0)*SN!
134   CO!(I,1.0)=X1!:CO!(I,2.0)=Y1!:NEXT
140   REM SCALING
142   FOR I=1 TO 4:FOR J=1 TO 2:CO!(I,J)=CO!(I,J)*S!:NEXT:NEXT
150   S!=S!+SF!:IF S!<0.0 THEN SF!=1E-2
199   NEXT
999   END
```

***** N I E U W E T E C H N I S C H E O N T W I K K E L I N G E N *****

In het Info-bulletin van de Nascom gebruikers club vonden wij een toch ook voor DAI-gebruikers interessant artikel over nieuwe hardware ontwikkelingen. Met medeweten en toestemming van dezs club en auteur nemen wij het hier over.

De WOM

Dit is een nieuw geheugen IC waarmee een revolutie is ontketend in de wereld van de solid state memory's. Al direct na de aankondiging van dit nieuwe IC door de japanse fabrikant Jijusti plaatste IBM een bestelling van 10 miljoen dollar. Het gevolg is, dat dit IC waarschijnlijk pas eind 1983 op de Europese markt beschikbaar zal komen. Vermoedelijk zal het voor de hobbymarkt nog wel wat langer gaan duren. Wat is er nu zo bijzonder aan deze memorychip, dat zelfs IBM zo gretig heeft gereageerd ?

Welnu, ten eerste de prijs. De 16K x 8 versie zal, voor zover het zich nu laat aanzien, zeker onder de een dollar gaan komen. De momenteel nog in ontwikkeling zijnde 64K x 8 versie zal de prijs van 3 dollar niet gaan overschrijden.

Ten tweede de specifcatie's. De 16K x 8 versie is pin-compatible met de 2764 eprom. De data-sheets geven aan dat het hier gaat om een single 5 volt memory-chip met in de huidig beschikbare versie een accesstime van 50 nano-seconde zodat deze IC's uitermate geschikt zijn voor toepassingen in combinatie met snelle of zeer snelle proseccors. Naar het zich laat aanzien, zal deze nieuwe ontwikkeling een grote invloed gaan hebben op de toepassing van Winchester disk systemen. Solid state blijft namelijk altijd betrouwbaarder dan een systeem met mechanische (dwz bewegende) componenten. Geheugenbanken van vele Mega bytes op een printplaat zijn nu eenvoudig te verwezelijken en zullen grote concurrenten gaan worden voor de hard-disk systemen.

Ten derde de power dissipatie. Indien geselecteerd neemt de 16K x 8 versie slechts 85 milliwatt op en in de mode -niet geselecteerd- daalt het verbruik zelfs tot 15 milliwatt. Ook deze gegevens wijzen op de mogelijkheid van grote geheugenbanken met een minimum aan warmteontwikkeling.

Hoe is dit nu allemaal mogelijk ? Welnu,voor zover het te lezen was in de specificaties zijn een aantal nieuwe vindingen gecombineerd op zodanige wijze dat bovenstaande resultaten een feit werden. Deze vindingen betreffen ondermeer:

- een op de chip uitgevoerde AC/AC converter
- een schuifregister volgens het FINO-principe
- twee schuifregisters volgens het FISO-principe
- een parallelle input die intern bi-directioneel wordt gecomplementeerd
- een refresh systeem gebaseerd op het 4711 principe
- een non-directionele interne bus structuur
- een a-synchrone zichzelf continu corrigerende baudrate

De toepassing van de WOM wordt vooral gezien in systemen waarbij data-opslag voor langere tijd en in grote kwantiteit noodzakelijk is.

Gebruikte afkortingen: FINO - First In ,Never Out
FISO - First In ,Sometimes Out
4711 - bekend Keuls patent
WOM - Write Only Memory

Tot zover dit nieuwtje over de WOM. Ik zal trachten iedereen op de hoogte te houden over nieuwe ontwikkelingen op dit gebied en nodig alle leden uit om bijdragen hierover aan de redactie op te sturen.

FIAT

```
100   REM
110   REM
120   REM
130   REM SOMEBODY NEEDS A FIAT MARK ?
140   REM
150   REM
160   REM ... Gianni Uliana
170   REM
180   REM
190   REM
200   REM
10010 COLORG 14 0 0 0
10015 MODE 5
10020 FILL 55,90 290,150 0
10040 DRAW 96,90 111,150 14
10050 DRAW 97,90 112,150 14
10060 DRAW 98,90 113,150 14
10070 DRAW 155,90 172,150 14
10080 DRAW 156,90 173,150 14
10090 DRAW 157,90 174,150 14
10100 DRAW 215,90 231,150 14
10110 DRAW 216,90 232,150 14
10120 DRAW 217,90 233,150 14
10130 FOR I=40.0 TO 55.0
10131 DRAW I,90 I+15,150 0:NEXT
10140 FOR I=275.0 TO 290.0
10141 DRAW I,90 I+15,150 14:NEXT
10149 REM F
10150 FOR I=55.0 TO 60.0
10151 DRAW I,105 I+10,140 14:NEXT
10152 DRAW 66,140 92,140 14:DRAW 66,139 92,139 14
10153 DRAW 66,138 91,138 14:DRAW 66,137 91,137 14
10154 DRAW 65,125 86,125 14:DRAW 65,124 86,124 14
10155 DRAW 65,123 85,123 14:DRAW 65,122 85,122 14
10159 REM I
10160 FOR I=125.0 TO 130.0
10161 DRAW I,103 I+10,140 14:NEXT
10169 REM A
10170 FOR I=195.0 TO 200.0
10180 DRAW I-25,103 I,140 14:DRAW I+10,103 I,140 14:NEXT
10181 FOR I=115.0 TO 119.0
10182 DRAW 182,I 203,I 14:NEXT
10189 REM T
10190 FOR I=245.0 TO 250.0
10191 DRAW I,103 I+10,140 14:NEXT
10195 DRAW 240,136 271,136 14
10196 DRAW 240,137 271,137 14
10197 DRAW 241,138 272,138 14
10198 DRAW 241,139 272,139 14
10199 DRAW 241,140 272,140 14
10200 WAIT TIME 100:GOTO 10020
```

FIAT

ON ERROR GOTO

```
SPL V1.1  PAGE 1

0000                    TITL       'ON ERROR GOTO V2.0'
0000            ;To activate:
0000            ;*UT <RETURN>
0000            ;>V5 C6FD-300 <CURSOR LEFT>
0000            ;>B
0000            ;*POKE #6,LINENUMBER IAND #FF:POKE #7,LINENUMBER SHR 8
0000            ;To deactivate:
0000            ;*POKE #6,0:POKE #7,0
0000            ;-------------------------------------------------------------------------
0000            ;  STACK DURING ERRORHANDLING
0000            ;
0000            ;When an error occurs a jump is performed to the errorprint-
0000            ;routine which goes via ODA50H. This routine is as follows:
0000            ;
0000            ;             ADDRES          STACKP    TOS         ROUTINE
0000            ;ODA3DH     CALL ODA50H       SP+2H     ODA40H      Runtime error
0000            ;     Stackbase for all errors;
0000            ;ODA50H     CALL ODD55H       SP-00H    ODA53H      Print CR before
0000            ;ODD55H     PUSH D            SP-02H    DE          errormsg is
0000            ;ODD56H     PUSH H            SP-04H    HL          printed.
0000            ;ODD57H     RST5
0000            ;ODD58H     DB   0CH          SP-06H    ODAE1H      get CURX,CURY
0000            ;0028H      NOP
0000            ;0029H      PUSH H            SP-08H    HL          Change to ROMbank
0000            ;002AH      LHLD 6CH                                2 6C=300H
0000            ;002DH      PCHL
0000            ;0300H      PUSH PSW          SP-0AH    PSW         On error routine
0000            ;
0000            ;-------------------------------------------------------------------------
0000  @=0006 ERRLIN   EQU        6H              ;Free bytes after RST 0
0000            ;
0000                  ORG        300H
0300  F5              PUSH PSW                   ;Save PSW
0301            ;                                ;HL may be corrupted because
0301            ;                                ;contents is already on stack.
0301  2A0600          LHLD       ERRLIN          ;Check if line to goto
0304  7C              MOV A,H                    ;on error
0305  B5              ORA L
0306  CA2103          JZ         OUT             ;quit if 0
0309            ;
0309  210A00          LXI H      0AH             ;Offset to stackbase
030C  39              DAD SP                     ;Add stackpointer to find
030D            ;                                ;original caller
030D  7E              MOV A,M                    ;check if caller is ODA53H
030E  FE53            CPI        53H
0310  C22103          JNZ        OUT             ;if not continu RST 5
0313  23              INX H
0314  7E              MOV A,M
0315  FEDA            CPI        0DAH
0317  C22103          JNZ        OUT             ;if caller is NOT ODA50H
031A  23              INX H                      ;Check if runtime error
031B  7E              MOV A,M
031C  FE40            CPI        40H
031E  CA2503          JZ         ONERR           ;If no running program
0321  F1        OUT   POP PSW                    ;restore stack continu
0322  C3FDC6          JMP        0C6FDH          ;with normal RST 5
0325            ;
0325  2A0600  ONERR   LHLD       ERRLIN          ;Get linenumber to goto
```

ON ERROR GOTO

```
SPL V1.1  PAGE 2      ON ERROR GOTO V2.0

0328            ;                                   ;in HL
0328  CDF6CA              CALL        0CAF6H        ;find this line in BASIC
032B  D24D03              JNC         UNDEFL        ;text buffer, if undefined
032E            ;                                   ;line run error
032E  44                  MOV B,H                   ;BC start line in BASIC
032F  4D                  MOV C,L                   ;text buffer
0330  210001              LXI H       100H          ;start of basic pointers
0333  3E15                MVI A       15H           ;in HL, clear evt.GOSUB and
0335  3600    LOOP       MVI M       0H            ;FOR NEXT levels
0337  23                  INX H
0338  3D                  DCR A
0339  C23503              JNZ         LOOP          ;loop till ready
033C  F3                  DI
033D  323101              STA         131H          ;Output to screen
0340  324000              STA         40H           ;Force ROMbank 0
0343  3206FD              STA         0FD06H
0346  FB                  EI
0347  3100F9              LXI SP      0F900H        ;reset stack pointer continu
034A  C392C8              JMP         0C892H        ;with BASIC execution
034D            ;
034D  210000  UNDEFL      LXI H       0H            ;If the lineno in ERRLIN does
0350  220600              SHLD        ERRLIN        ;not exist errors are enabled.
0353  3E04                MVI A       4H            ;code for UNDEFINED LINENUMBER
0355  C3F5D9              JMP         0D9F5H        ;print this message. Back to
0358            ;                                   ;BASIC monitor.
0358                      END
```

```
0300 F5 2A 06 00 7C B5 CA 21 03 21 0A 00 39 7E FE 53
0310 C2 21 03 23 7E FE DA C2 21 03 23 7E FE 40 CA 25
0320 03 F1 C3 FD C6 2A 06 00 CD F6 CA D2 4D 03 44 4D
0330 21 00 01 3E 15 36 00 23 3D C2 35 03 F3 32 31 01
0340 32 40 00 32 06 FD FB 31 00 F9 C3 92 C8 21 00 00
0350 22 06 00 3E 04 C3 F5 D9 00 00 00 00 00 00 00 00
```

```
1     REM ON ERROR GOTO DEMO/N.P. LOOIJE
2     REM all levels of FOR NEXT/GOSUB will be cleared
10    MODE 0:PRINT CHR$(12)
20    POKE 6,10:POKE 7,0:REM ON ERROR GOTO 10
30    A=A+1:PRINT A:IF A>10 THEN 50:REM 10 times ON ERROR
40    DOT A,B C:REM not possible in MODE 0
50    POKE 6,0:POKE 7,0:REM ON ERROR off
60    GOTO 40:REM now an errormessage will be printed
```

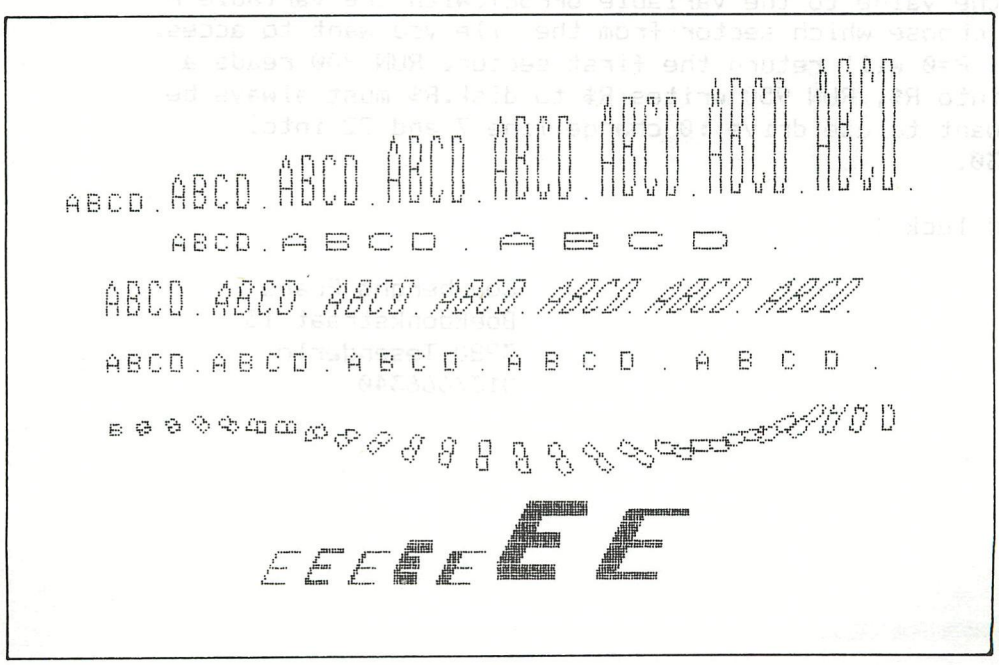
UPPER TO LOWER CASE

```
002                     *
003                     *
004                     * CONVERTS UPPER CASE CHARACTERS TO LOWER
005                     * CASE WHEN PLACED IN DATA STATEMENTS.
006                     *
007                     * HL: ADDRESS IN TEXTBUFFER
008                     * DE: END TEXTBUFFER
009                     * BC: OFFSET OF START NEXT TEXTLINE
010                     *  C: NR OF DATABYTES TO BE CHECKED
011                     *
012                           ORG   :400
013                     *
014 0400 F5             INIT  PUSH  PSW
015 0401 C5                   PUSH  B
016 0402 D5                   PUSH  D
017 0403 E5                   PUSH  H
018 0404 2AA102               LHLD  :02A1       GET STBBGN
019 0407 EB                   XCHG
020 0408 2A9F02               LHLD  :029F       GET TXTBGN
021 040B 0600                 MVI   B,:00
022                     *
023 040D CD14DE         LNCNT CALL  :DE14       COMPARE HL-DE
024 0410 D23B04               JNC   READY       ) ABORT WHEN
025 0413 CA3B04               JZ    READY       ) FINISHED
026                     *
027 0416 4E             LKDATA MOV  C,M         LENGTH TEXTLINE IN C
028 0417 23                   INX   H
029 0418 23                   INX   H
030 0419 23                   INX   H           POINTS TO TOKEN
031 041A 7E                   MOV   A,M         GET TOKEN
032 041B FEA2                 CPI   :A2         DATA STATEMENT?
033 041D CA2604               JZ    DATA
034 0420 2B                   DCX   H
035 0421 2B                   DCX   H
036 0422 09             NEXTLN DAD  B           GET ADDR NEXT TEXTLINE
037 0423 C30D04               JMP   LNCNT
038                     *
039 0426 23             DATA  INX   H           POINTS TO LENGTH DATA
040 0427 4E                   MOV   C,M
041 0428 0C                   INR   C
042 0429 23             NXTBYT INX  H           ) UPDATE ADDR +
043 042A 0D                   DCR   C           ) LENGTH POINTERS
044 042B CA0D04               JZ    LNCNT       NEXT LINE WHEN DONE
045 042E 7E                   MOV   A,M         GET BYTE
046 042F CD02DE               CALL  :DE02       CHECK IF UPPER CASE CHAR
047 0432 D22904               JNC   NXTBYT
048 0435 C620           CHANGE ADI  :20         CHANGE TO LOWER CASE
049 0437 77                   MOV   M,A
050 0438 C32904               JMP   NXTBYT
051                     *
052 043B E1             READY POP   H
053 043C D1                   POP   D
054 043D C1                   POP   B
055 043E F1                   POP   PSW
056 043F C9                   RET
057                     *
058 0440                      END
```

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

CHANGE 0435   DATA   0426   INIT   0400   LKDATA 0416
LNCNT  040D   NEXTLN 0422   NXTBYT 0429   READY  043B
```

```
1      REM GRAFTEXT, MAAKT GEBRUIK VAN EEN FGT-TABEL.
2      COLORG 8 0 0 0:MODE 5A:COLORG 8 0 0 0
5      GTCS$="GRAFTEXT":C=0:X0=60:Y0=0:H!=6.0:W!=3.0:G!=0.0:R!=45.0:S!=45.0:
TH=1:TW=6:GOSUB 40000
6      WAIT TIME 100:MODE 5A        ***** FIRST LOAD FGT *****
9      GTCS$="ABCD."
10     X0=5:Y0=150:W!=1.0:G!=0.0:R!=0.0:S!=0.0:TH=0:TW=0
11     FOR H!=1.0 TO 8.0:GOSUB 40000:NEXT
12     X0=45:Y0=135:H!=1.0:FOR W!=1.0 TO 3.0:GOSUB 40000:NEXT
13     X0=20:Y0=110:H!=2.0:W!=1.0:FOR S!=0.0 TO 90.0 STEP 15.0:GOSUB 40000:N
EXT
14     X0=20:Y0=90:H!=1.0:W!=1.0:S!=0.0:FOR G!=0.0 TO 8.0 STEP 2.0:GOSUB 400
00:NEXT
15     X=25:X0=X:Y0=60:H!=0.6:W!=1.0:S!=0.0:G!=0.0:GTCS$="B"
16     FOR R!=0.0 TO 360.0 STEP 15.0:GOSUB 40000:X=X+12:X0=X:Y0=60:H!=H!+0.1
:NEXT
17     X0=80:Y0=2:H!=2.0:W!=2.0:S!=30.0:R!=0.0:G!=0.0:GTCS$="E":GOSUB 40000
18     TH=1:TW=0:GOSUB 40000:TH=0:TW=1:GOSUB 40000
19     TH=4:TW=0:GOSUB 40000:TH=0:TW=4:GOSUB 40000
20     H!=4.0:W!=4.0:TH=5:TW=5:GOSUB 40000:TH=2:TW=6:GOSUB 40000
99     END
999    END
40000  REM GRAF TEXT
40002  RAD!=57.2958:HS!=H!:SW!=W!:SG!=G!/SW!
40004  HX!=COS(R!/RAD!):HY!=SIN(R!/RAD!)
40008  VX!=-HY!+HX!*SIN(S!/RAD!):VY!=HX!+HY!*SIN(S!/RAD!)
40010  DHX!=HX!*SW!:DHY!=HY!*SW!:DVX!=VX!*HS!:DVY!=VY!*HS!
40014  DSX!=DHX!*(8.0+SG!):DSY!=DHY!*(8.0+SG!)
40100  FOR M=0 TO LEN(GTCS$)-1:GTC=ASC(MID$(GTCS$,M,1)):GTI=#580
40120  GTT1=PEEK(GTI):GTT2=PEEK(GTI+1):IF GTT1=#5A AND GTT2=#FF GOTO 40140
40122  IF GTT1<>GTC THEN GTI=GTI+2*GTT2+2:GOTO 40120
40130  GTI=GTI+2
40131  FOR N=GTI TO GTI+2*(GTT2-1) STEP 2
40132  GTX=(PEEK(N) IAND #F0) SHR 4:GTY=PEEK(N) IAND #F
40133  GTX1=X0+GTY*DVX!+GTX*DHX!:GTY1=Y0+GTY*DVY!+GTX*DHY!
40135  GTX=(PEEK(N+1) IAND #F0) SHR 4:GTY=PEEK(N+1) IAND #F
40136  GTX2=X0+GTY*DVX!+GTX*DHX!:GTY2=Y0+GTY*DVY!+GTX*DHY!
40137  FOR T1=0 TO TW:FOR T2=0 TO TH
40138  DRAW GTX1+T1,GTY1+T2 GTX2+T1,GTY2+T2 C:NEXT:NEXT
40139  NEXT:X0=X0+DSX!+TW:Y0=Y0+DSY!
40140  NEXT
40150  RETURN
```

GRAFTEXT



RANDOM-ACCESS

Tessenderlo 27 april, 1983

Dear Dai-Diskunit-owners,

As owner of a dual Disk-drive of the firm Dai I noticed that to complete the Disk Operating System there was need for a Random Acces posibility.

Thanks to the artikel in Newsletter 14 I developed a method that is close to the original Random acces feature.
When we read a sector from disk, this sector will occupy 128 bytes in memory adressed by the HL register.When the content of the HL register is a VARPTR from a string with a length of 128 the information from the sector is stored in this string.We only have to set the length of the string,define which track and sector we need and call a routine.Upon return the string contains the sectors information.The same applys for writing a sector to disk.The only important thing is the length of the string.It always must be 128 bytes.When not you can destroy parts of other variables.
By now you will ask: How do I now which track and sector I have to poke ? To get this information we look in the directory. Whe therefore use the ASCII-FILE method used in the Auto-Menu on ower system disk.Here are the results:

```
Example:   FILENAMEBAS-----N#####
           11111111222333345677
```

1: 8 Bytes for the filename.
2: 3 Bytes for the extension.
3: 5 Bytes not used.
4: 1 byte for the attribute
5: 8 bit number for first track of file.
6: 8 bit number for first sector of file.
7: 16 bit number for length of file

With the statement ASC() we retrieve the desired information. Multiply the track with #12 and add the sectors to it.Then assign the value to the variable OFFSET.With the variable P you can choose which sector from the file you want to acces. Caution: P=0 will return the first sector. RUN 900 reads a sector into R$, RUN 950 writes R$ to disk.R$ must always be If you want to use drive :0 change line 7 and 22 into:
MVI A,:30.

Good luck !

Couwberghs Frans
Boekdonkstraat 13
3980 Tesenderlo
013/666340

RANDOM-ACCESS

```
1010  PRINT "   PAGE 01    RANDOM GET PUT"
1070  PRINT
1080  PRINT "   002                          ORG   :19CD"
1090  PRINT "   003 19CD F5                  PUSH  PSW"
1100  PRINT "   004 19CE C5                  PUSH  B"
1110  PRINT "   005 19CF D5                  PUSH  D"
1120  PRINT "   006 19D0 E5                  PUSH  H"
1130  PRINT "   007 19D1 3E31                MVI   A,:31"
1140  PRINT "   008 19D3 0600                MVI   B,0"
1150  PRINT "   009 19D5 0E00                MVI   C,0"
1160  PRINT "   010 19D7 210000              LXI   H,0"
1170  PRINT "   011 19DA 23                  INX   H"
1180  PRINT "   012 19DB CD1C0A              CALL  :A1C"
1190  PRINT "   013 19DE E1                  POP   H"
1200  PRINT "   014 19DF D1                  POP   D"
1210  PRINT "   015 19E0 C1                  POP   B"
1220  PRINT "   016 19E1 F1                  POP   PSW"
1230  PRINT "   017 19E2 C9                  RET"
1240  PRINT "   018 19E3 F5                  PUSH  PSW"
1250  PRINT "   019 19E4 C5                  PUSH  B"
1260  PRINT "   020 19E5 D5                  PUSH  D"
1270  PRINT "   021 19E6 E5                  PUSH  H"
1280  PRINT "   022 19E7 3E31                MVI   A,:31"
1290  PRINT "   023 19E9 0600                MVI   B,0"
1300  PRINT "   024 19EB 0E00                MVI   C,0"
1310  PRINT "   025 19ED 210000              LXI   H,0"
1320  PRINT "   026 19F0 23                  INX   H"
1330  PRINT "   027 19F1 CD320A              CALL  :A32"
1340  PRINT "   028 19F4 E1                  POP   H"
1350  PRINT "   029 19F5 D1                  POP   D"
1360  PRINT "   030 19F6 C1                  POP   B"
1370  PRINT "   031 19F7 F1                  POP   PSW"
1380  PRINT "   032 19F8 C9                  RET"
1390  PRINT "   033 19F9                     END"
1400  PRINT
1410  PRINT
1420  PRINT "900   REM GET A SECTOR FROM DISK"
1430  PRINT "910   TRACK=(P+OFFSET)/#12:SECTOR=(P+OFFSET) MOD #12"
1440  PRINT "920   IF SECTOR=0 THEN SECTOR=18:TRACK=TRACK-1"
1450  PRINT "930   POKE #19D4,TRACK:POKE #19D6,SECTOR:POKE #19D8,PEEK(VARPT
R(R$))"
1460  PRINT "950   REM PUT A SECTOR ON DISK"
1470  PRINT "960   TRACK=(P+OFFSET)/#12:SECTOR=(P+OFFSET) MOD #12"
1480  PRINT "970   POKE #19EA,TRACK:POKE #19EC,SECTOR:POKE #19EE,PEEK(VARPT
R(R$))"
1490  PRINT "980   POKE #19EF,PEEK(VARPTR(R$)+1):CALLM #19E3:RETURN"
```

NEGATIEVE CURSOR

```
100   REM *** DE NEGATIEVE CURSOR ***************
110   REM *** GESCHREVEN DOOR : DE BONT CORNEEL *
120   REM ********************* 14 - 5 - 1983 ***
130   REM ****************************************
140   GOTO 400
200   REM *** BUITEN-MARGE SUBROUTINE
210   IC=#BFEE-((23-YP)*#86):POKE IC,208+ZP
215   IF YP>0 THEN POKE IC-#86,208
220   FOR IX=0 TO LEN(G$)-1:T$=MID$(G$,IX,1):GX=ASC(T$)
230   IP=#BFE7-((23-YP)*#86)-(XP*2):POKE IP,GX
240   XP=XP+1:NEXT
250   RETURN
300   REM *** GETC-ROUTINE
310   CURSOR 0,0:PRINT SPC(58);:POKE #75,95
320   CURSOR 10,0:PRINT "DRUK OP EEN TOETS OM TE VERVOLGEN";
330   CC=INT(15*RND(1)):IF CC=8 THEN 330
340   POKE #B3E4,CC+208:SOUND 1 0 15 0 FREQ((CC*50)+50)
350   WAIT TIME 3:SOUND OFF :G=GETC:G=GETC:G=GETC
360   G=GETC:IF G=0 THEN WAIT TIME 3:I=I+1:IF I<30 THEN 360
370   I=0:IF G=0 THEN 330
380   POKE #75,32:RETURN
400   REM *** TITEL
410   MODE 0:COLORT 8 0 8 8:PRINT CHR$(12);:POKE #75,32
420   CURSOR 55,7:ZP=-1:FOR I=0 TO 23:XP=-3+I:YP=23-I
430   G$="DEMONSTRATIE VAN NEGATIEVE CURSOR"
440   ZP=ZP+1:IF ZP=16 THEN ZP=0
450   IF ZP=8 THEN ZP=ZP+1
460   GOSUB 200:NEXT:PRINT
470   GOSUB 300:RESTORE
500   PRINT CHR$(12);:COLORT 8 0 8 8:CURSOR 55,7
510   FOR YP=23 TO 1 STEP -1:READ G$:ZP=0
520   XP=INT(3*RND(1)):IF RND(10)>5 THEN XP=XP*(-1)
530   GOSUB 200:NEXT
540   GOSUB 300:GOTO 400
600   REM *** INFO
610   DATA "DEMONSTRATIE VAN EEN NEGATIEVE CURSOR"
620   DATA "**************************************"
630   DATA "DIT PROGRAMMA DEMONSTREERT U EEN SUB-"
640   DATA "ROUTINE,WAARIN EEN NEGATIEVE CURSOR  "
650   DATA "WORDT GEBRUIKT.EEN OPGEGEVEN STRING  "
660   DATA "WORDT OP ELKE GEWENSTE PLAATS OP HET "
670   DATA "SCHERM WEGGESCHREVEN,NET ALS BIJ HET "
680   DATA "GEBRUIK VAN EEN CURSOR-STATEMENT !!  "
690   DATA "HET GROTE VERSCHIL IS DAT MEN HIER   "
700   DATA "CURSOR X-WAARDEN TOT -3 KAN OPGEVEN. "
710   DATA "OOK BESTAAT DE MOGELIJK DE REGELS TE "
720   DATA "KLEUREN.DIT KAN MEN KLAARSPELEN DOOR "
730   DATA "BIJ HET AANROEPEN DER SUBROUTINE DE  "
740   DATA "VARIABELE 'ZG' EEN WAARDE TUSSEN 0   "
750   DATA "EN 15 MEE TE GEVEN. (0=ZWART)        "
760   DATA "DEZE SUBROUTINE KAN ALS VOLGT AANGE- "
770   DATA "SPROKEN WORDEN.MEN BEPAALT EERST:    "
780   DATA "XG : (X-AS) EEN GETAL TUSSEN -3 EN 58"
790   DATA "YG : (Y-AS) EEN GETAL TUSSEN 0 EN 23 "
800   DATA "ZG : (KLEUR) EEN GETAL TUSSEN 0 EN 15"
810   DATA "G$ : DE STRING (TOT MAX 62 LETTERS)  "
820   DATA "WAARNA MEN EEN 'GOSUB 200' UITVOERT  "
830   DATA "(XG=-3:YG=9:ZG=7:G$='DEMO':GOSUB 200)"
```

FLASHING IN 4 COLOR

Here is a convenient way of making items flash on the DAI display. The method uses a small machine code routine which is triggered by the RST7 interrupt. At intervals controlled by a parameter the routine switches the value in the colour control byte at location £BFF2. This is the one which is used to set up colour register 3 in a 4-colour mode. As a result anything which is drawn on the screen using colour '3' will flash (change colour regularly).

Here is the machine code routine that I use:-

```
FLASH:    PUSH H
          LXI  H,SWITCH
          SHLD £70                ; INTERCEPT RST7
          POP  H
          RET
SPEED:    DC   10                 ; FLASH RATE PARAMETER
REMAIN    DC   0                  ; WORKING REMAINDER VALUE
DIFFER    DC   0                  ; COLOUR CHANGE PARAMETER
SWITCH:   PUSH PSW
          LXI  H,SPEED            ; EACH TIME SWITCH IS EXECUTED
          MOV  A,M                ; SPEED IS ADDED TO REMAIN
          INX  H                  ; IF A CARRY OCCURS IT IS TIME
          ADD  M                  ; TO SWAP COLOURS IN £BFF2
          MOV  M,A
          JNC  NOCARRY
          INX  H
          LDA  £BFF2              ; 'DIFFER' IS TAKEN AS THE
          XRA  M                  ; LOGICAL DIFFERENCE BETWEEN
          STA  £BFF2              ; THE TWO COLOUR CODES
NOCARRY:  POP  PSW
          LXI  H,£D9A9            ; NORMAL RST7 CONTINUATION
          PCHL
```

Here is a simple example program using the technique.

```
10 CLEAR 1000
20 POKE £29B,3: POKE £29D,PEEK(£29D)-1: REM make space
30 FOR I=£2EC TO £30F: READ J: POKE I,J: NEXT
40 MODE 6A: COLORG 15 5 1 5
50 CALLM £2EC: REM Activate FLASH routine
60 POKE £2F7,10: REM Colour register will alternate 15 - 5
60 FILL 100,50 110,60 21: REM Ordinary green blob
70 FILL 150,50 160,60 23: REM Flashing green blob
80 DATA £E5,£21,£F8,£02,£22,£70,£00,£E1,£C9
90 DATA £0A,£00,£00,£F5,£21,£F5,£02,£7E,£23
100 DATA £86,£77,£D2,£0B,£03,£23,£3A,£F2,£BF
110 DATA £AE,£32,£F2,£BF,£F1,£21,£A9,£D9,£E9
```

Now you can try different speeds and colours by POKING values into £2F5 and £2F7- (£2F5 is the speed control and is preset at 10 giving a flash time in milliseconds of 20*(256/10) or about half a second in each colour; - £2F7 is the logical difference between the colour codes). The difference should not exceed 15 or it will destroy the screen format

IMPORTANT NOTE: flashing at a rate of between 7 and 10 cycles per second can be unpleasant and even dangerous! so avoid values for SPEED which are greater than 50 or so.

Harry Harte
28.4.82

SOFTWARE PRINTER-SPOOLER

```
002                 *
003                 * Peter Lelie, 19-Oct-82
004                 * D-4047 Dormagen-Zons
005                 * Wilhelm Busch Strasse 36
006                 * Tel.: (02106) 46852
007                 *
008                 *
009                 * Program Description
010                 *
011                 * The printer spooler consists of three parts:
012                 *
013                 * 1. Entry point ENTER
014                 * Called via DOUTC (:02DD). If you want text to
015                 * be printed, POKE #131,3 . From now on all the
016                 * PRINTed stuff will be entered into the printer
017                 * buffer (inside RAM). If the buffer is filled,
018                 * the program waits for buffer to become free.
019                 * To switch off the entering of output into the
020                 * buffer, just POKE #131,1 .
021                 *
022                 * 2. Entry point SPOOL
023                 * Called via RST7 every 20 msec. As the printer
024                 * is ready and there are characters inside the
025                 * buffer area, they will be printed, one after
026                 * another. After a character has been printed,
027                 * the used bufferspace will be freed for further
028                 * use by ENTER. SPOOL takes attention on the
029                 * 'DATA TERMINAL READY'-signal from the printer.
030                 *
031                 * 3. Entry point INIT
032                 * To be called from BASIC (only one time) to
033                 * install the spooler. BEFORE loading a BASIC
034                 * program, do a CALLM #300 to init the spooler
035                 * and to adjust the heap pointer and a NEW is
036                 * generated. After the first INIT, nothing bad
037                 * will happen if you do another CALLM #300,
038                 * as this will only execute a RETurn instruction.
039                 *
040                 *
041                 * DECLARATIONS
042                 *
043                 BAUD    EQU   :C0       ;9600 Bd
044                 *
045                 BUFSIZ  EQU   :800      ;buffer size (2k)
046                 DOUTC   EQU   :02DD     ;output vector
047                 HEAP    EQU   :029B     ;heap ptr loc.
048                 JUMP    EQU   :C3       ;code for JMP
049                 PRISW   EQU   :0131     ;printer switch
050                 RETUR   EQU   :C9       ;code for RET
051                 RST7    EQU   :0070     ;vector addr.
052                 *
053                 * DAI hardware related declarations
054                 BRREG   EQU   :FF05     ;Baudrate reg.
055                 DTR     EQU   :FD00     ;data term rdy
056                 V24DA   EQU   :FFF6     ;ser. data
057                 V24ST   EQU   :FFF3     ;ser. status
058                 *
059                 * DAI ROM entrypoints:
060                 CMPDH   EQU   :DE14     ;compare DE, HL
061                 OPSYS   EQU   :D9A9     ;rest of RST7
062                 PMSG    EQU   :DAD4     ;print message
063                 RNEW    EQU   :DEB8     ;Basic NEW
064                 *
065                 * Literals
```

```
066                     CR      EQU   :0D       ;car ret
067                     FF      EQU   :0C       ;formfeed
068                     LF      EQU   :0A       ;linefeed
069                     *
070                     *
071                             ORG   :0300
072 0300 C37A03         INIT    JMP   SETUP     ;RET after init
073                     *
074                     * ENTRY POINT VIA RST7
075                     *
076 0303 D5             SPOOL   PUSH  D         ;SAVE REG'S
077 0304 E5                     PUSH  H
078 0305 F5                     PUSH  PSW
079                     *
080                     * avoid output to screen and printer:
081                     *
082 0306 213101                 LXI   H,PRISW   ;if = 0
083 0309 7E                     MOV   A,M       ;then
084 030A FE00                   CPI   :00       ;make
085 030C C21003                 JNZ   OK        ;PRISW=1
086 030F 34                     INR   M         ;
087                     *
088                     * is RS232 ready for output?
089                     *
090 0310 3A00FD         OK      LDA   DTR       ;term ready?
091 0313 E608                   ANI   :08       ;
092 0315 CA3E03                 JZ    RETRN     ;
093 0318 3AF3FF                 LDA   V24ST     ;UART ready?
094 031B E610                   ANI   :10       ;
095 031D CA3E03                 JZ    RETRN     ;
096                     *
097                     * ready for output;
098                     * do we have any output?
099                     *
100 0320 2A7803                 LHLD  PTR2      ;
101 0323 EB                     XCHG            ;
102 0324 CD6003                 CALL  COMPAR    ;is PTR1<>PTR2?
103 0327 CA3E03                 JZ    RETRN     ;RET if not
104 032A 7E                     MOV   A,M       ;get char
105 032B 32F6FF                 STA   V24DA     ;print it
106 032E FE0D                   CPI   CR        ;carriage return ?
107 0330 C23803                 JNZ   NOCR      ;no
108 0333 360A                   MVI   M,LF      ;replace CR into LF
109 0335 C33E03                 JMP   RETRN     ;no change of pionters
110 0338 CD6703         NOCR    CALL  INCPTR    ;move pointer
111 033B 227603                 SHLD  PTR1      ;and restore
112 033E F1             RETRN   POP   PSW       ;restore reg's
113 033F E1                     POP   H         ;
114 0340 D1                     POP   D         ;
115 0341 C3A9D9                 JMP   OPSYS     ;back to opsys
116                     *
117                     * The following routine is called via
118                     * 'DOUTC'
119                     * Purpose: enter the received character
120                     * into printer buffer
121 0344 D5             ENTER   PUSH  D         ;save reg's
122 0345 E5                     PUSH  H         ;
123 0346 F5                     PUSH  PSW       ;
124                     *
125 0347 2A7803                 LHLD  PTR2      ;
126 034A CD6703                 CALL  INCPTR    ;
127 034D EB                     XCHG            ;
128 034E CD6003         LOOP    CALL  COMPAR    ;try to find
129 0351 CA4E03                 JZ    LOOP      ;buffer space
```

```
130                     *
131                     * buffer space available:
132                     * now put char into buffer:
133 0354 F1                     POP   PSW       ;
134 0355 2A7803                 LHLD  PTR2      ;
135 0358 77                     MOV   M,A       ;stor in buff
136 0359 EB                     XCHG            ;
137 035A 227803                 SHLD  PTR2      ;stor new PTR2
138 035D E1                     POP   H         ;restore reg's
139 035E D1                     POP   D         ;
140 035F C9                     RET             ;to sender!
141                     *
142                     * SUBROUTINES
143                     *
144                     * compare values in HL and DE
145                     *
146 0360 2A7603         COMPAR  LHLD  PTR1      ;get 1st ptr
147 0363 CD14DE                 CALL  CMPDH     ;comp DE, HL
148 0366 C9                     RET             ;
149                     *
150                     * move PTR in HL to next position
151                     *
152 0367 11790B         INCPTR  LXI   D,BUFEND  ;get end of buff
153 036A CD14DE                 CALL  CMPDH     ;compare
154 036D C27403                 JNZ   NXT       ;OK if not end
155                     * reached end of buffer:
156                     * goto start of buffer
157 0370 217A03                 LXI   H,BUFFER  ;PTR = BUFFER
158 0373 C9                     RET             ;
159 0374 23             NXT     INX   H         ;PTR = next pos.
160 0375 C9                     RET             ;
161                     *
162                     * Buffer area following:
163                     *
164 0376 0000           PTR1    DBL   :0000
165 0378 0000           PTR2    DBL   :0000
166                     *
167 037A                BUFFER  RES   BUFSIZ-1
168 0B79 00             BUFEND  DATA  :00
169 0B7A 00             FINI    NOP
170                     *
171                     * the following code is located inside
172                     * the printer buffer. After INIT it
173                     * will be overwritten!
174                     *
175                             ORG   BUFFER
176                     *
177 037A E5             SETUP   PUSH  H         ;save reg's
178 037B F5                     PUSH  PSW       ;
179 037C 3EC3                   MVI   A,JUMP    ;set up DOUTC
180 037E 32DD02                 STA   DOUTC     ;
181 0381 214403                 LXI   H,ENTER   ;
182 0384 22DE02                 SHLD  DOUTC+1   ;
183                     *
184 0387 3EC9                   MVI   A,RETUR   ;kill entrypoint
185 0389 320003                 STA   INIT      ;
186                     *
187 038C 217A03                 LXI   H,BUFFER  ;ini PTR1 & PTR2
188 038F 227603                 SHLD  PTR1      ;
189 0392 227803                 SHLD  PTR2      ;
190                     *
191 0395 3EC0                   MVI   A,BAUD    ;init baudrate
192 0397 3205FF                 STA   BRREG     ;
193                     *
```

```
194 039A 3E01              MVI   A,1       ;printer off
195 039C 323101            STA   PRISW     ;
196                  *
197 039F 217A0B            LXI   H,FINI    ;set heap ptr
198 03A2 229B02            SHLD  HEAP      ;
199 03A5 CDB8DE            CALL  RNEW      ;run NEW
200                  *
201 03A8 21B703            LXI   H,TXT$    ;print msg
202 03AB CDD4DA            CALL  PMSG      ;
203                  *
204 03AE 210303            LXI   H,SPOOL   ;init RST7
205 03B1 227000            SHLD  RST7      ;
206                  *
207 03B4 F1                POP   PSW       ;
208 03B5 E1                POP   H         ;
209 03B6 C9                RET             ;
210                  *
211                  *
212 03B7 0C          TXT$  DATA  FF
213 03B8 507269            ASC   'Printer spooler '
214 03C8 696E69            ASC   'initialized.'
215 03D4 0D0D              DATA  CR,CR
216 03D6 507269            ASC   'Printer ON : POKE #131,3'
217 03EE 0D                DATA  CR
218 03EF 202020            ASC   '        OFF: POKE #131,1'
219 0407 0D0D00            DATA  CR,CR,0
220                  *
221                  LENGTH EQU   FINI-SPOOL
222 040A                   END
```

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

BAUD	00C0	BRREG	FF05	BUFEND	0B79	BUFFER	037A
BUFSIZ	0800	CMPDH	DE14	COMPAR	0360	CR	000D
DOUTC	02DD	DTR	FD00	ENTER	0344	FF	000C
FINI	0B7A	HEAP	029B	INCPTR	0367	INIT	0300
JUMP	00C3	LENGTH	0877	LF	000A	LOOP	034E
NOCR	0338	NXT	0374	OK	0310	OPSYS	D9A9
PMSG	DAD4	PRISW	0131	PTR1	0376	PTR2	0378
RETRN	033E	RETUR	00C9	RNEW	DEB8	RST7	0070
SETUP	037A	SPOOL	0303	TXT$	03B7	V24DA	FFF6
V24ST	FFF3						

READ/WRITE IN UTILITY

HOW TO USE UTILITY 'WRITE' AND 'READ' IN M.L.PROGRAMS

The DAI utility package doesnot have a simple entry to enable writing to and reading from tape in combination with machine language programs. The onliest possibility to write and read M.L.programs is via the direct Utility commands 'W' and 'R'.
Annexed routines give an entry to the 'W'- and 'R'-routines in combination with M.L.programs.

When you are working with the Utility package, ROM-bank 3 of the firmware is used. Here also the write and the read routines for file type 1 (hex-files) can be found, resp. on 3EEE4 and 3EF0F. These are used in the routines below.

The WRITE routine (to be used via CALL WRITE), prints a 'W' on the screen. Then the startaddress and the endaddress of the file to be saved can be typed in, as well as the name of the file. Then the file is written to tape in exactly the same way as always.

The READ routine (CALL READ) works the same way. It prints a 'R', and then an eventual offset and the file name can be typed in. Then the file is read from tape.

```
002                     *
003                     *
004                             ORG    :300
005                     *
006                     *
007 0300 E5             WRITE   PUSH   H
008 0301 D5                     PUSH   D
009 0302 C5                     PUSH   B
010 0303 F5                     PUSH   PSW
011 0304 CD5EDD                 CALL   :DD5E      CAR.RET
012 0307 3E57                   MVI    A,:57
013 0309 CD60DD                 CALL   :DD60      PRINT 'W'
014 030C CDE4EE                 CALL   :EEE4      'WRITE'
015 030F F1                     POP    PSW
016 0310 C1                     POP    B
017 0311 D1                     POP    D
018 0312 E1                     POP    H
019 0313 C9                     RET
020                     *
021                     *
022                     *
023 0314 E5             READ    PUSH   H
024 0315 D5                     PUSH   D
025 0316 C5                     PUSH   B
026 0317 F5                     PUSH   PSW
027 0318 CD5EDD                 CALL   :DD5E      CAR.RET
028 031B 3E52                   MVI    A,:52
029 031D CD60DD                 CALL   :DD60      PRINT'R'
030 0320 CD0FEF                 CALL   :EF0F      'READ'
031 0323 F1                     POP    PSW
032 0324 C1                     POP    B
033 0325 D1                     POP    D
034 0326 E1                     POP    H
035 0327 C9                     RET
036                     *
037                     *
038                     *
039 0328                        END
```

The routine WRITE1 is a second possibility to write files on tape under control of a m.l.program.
Before calling this routine, the low-address of the file to be saved must be in the HL-register and the high-address in the DE-register. This is useful when the boundaries of the file depend on the action performed by your m.l.program.
The routine asks for the filename to be typed in before the file is written to tape.

```
001                             ORG   :300
002                     *
003 0300 E5             WRITE1  PUSH  H
004 0301 D5                     PUSH  D
005 0302 C5                     PUSH  B
006 0303 F5                     PUSH  PSW
007 0304 CD0C03                 CALL  WRITE2
008 0307 F1                     POP   PSW
009 0308 C1                     POP   B
010 0309 D1                     POP   D
011 030A E1                     POP   H
012 030B C9                     RET
013                     *
014 030C E5             WRITE2  PUSH  H
015 030D D5                     PUSH  D
016 030E 211703                 LXI   H,NAME
017 0311 CD2FED                 CALL  :ED2F       PRINT NAME:
018 0314 C3EDEE                 JMP   :EEED       'WRITE'
019                     *
020 0317 0D             NAME    DATA  :0D
021 0318 4E                     DATA  :4E         N
022 0319 41                     DATA  :41         A
023 031A 4D                     DATA  :4D         M
024 031B 45                     DATA  :45         E
025 031C 20                     DATA  :20
026 031D 3A                     DATA  :3A         :
027 031E 20                     DATA  :20
028 031F 00                     DATA  :00
029                     *
030 0320                        END
```

The use of the CHECK-routine in m.l.programs is difficult. Not only because it is a routine which can only be aborted by a BREAK, but also because the program- or filenames are given via a 'fast-print' routine. This routine 'prints' the filenames on the screen by poking the name directly into the screen RAM area.
Maybe someone wants to try to write a routine for this purpose. The article 'Cassette list' in DAInamic 82/1-p.48 can be helpful.
Please let me know when you succeeded.

Jan Boerrigter

program identification

title	:	LARGE TEXT (PRINTER-POSTER)
author	:	T. GROENEVELD
purpose	:	generates jumbo print-out
comment	:	control-characters are for EPSON-printers

```
1     REM CHANGE LINE SPACING ON EPSON : ?CHR$(27);"A";CHR$(9)
2     MODE 0:CLEAR 5000:COLORT 15 0 0 0:POKE #131,1
5     PRINT CHR$(12):PRINT
6     PRINT "              ############################"
7     PRINT "              #      LARGE TEXT          #"
8     PRINT "              #      ==========          #"
9     PRINT "              ############################":PRINT
10    INPUT "   Hoeveel karakters vertikaal........ ";X%:PRINT :PRINT
20    INPUT "   Hoeveel karakters horizontaal...... ";Y%:PRINT :PRINT
21    INPUT "   Wilt u tabulator gebruiken (J/N)... ";L$:PRINT :PRINT
22    G1%=0.0:IF L$="J" THEN G1%=1.0
23    PRINT "   Met welke tekst dienen karakters opgebouwd"
24    INPUT "   te worden.......................... ";M$:PRINT :PRINT
29    INPUT "   Wat zijn de gewenste karakters..... ";A$:PRINT :PRINT
30    IF M$="" OR A$="" THEN 5
35    INPUT "   STEL PAPIER IN EN GEEF RETURN";0$:PRINT :POKE #131,0
40    A%=ASC(LEFT$(A$,1))
65    DIM S%(10.0),J%(10.0),F%(10.0)
70    FOR T%=0.0 TO LEN(A$)-1.0
80    P$=MID$(A$,T%,1)
90    FOR Q%=1.0 TO 50.0
94    READ S$,S%(1.0),S%(2.0),S%(3.0),S%(4.0),S%(5.0),S%(6.0),S%(7.0)
98    IF P$=" " THEN 812
100   IF P$=S$ THEN 200
120   NEXT Q%
200   RESTORE
201   X$=M$
205   FOR U%=1.0 TO 7.0
210   FOR K%=8.0 TO 0.0 STEP -1.0
230   IF INT(2.0^K%+0.5)<INT(S%(U%)) THEN 270
235   REM PRINT INT(2.0^K),INT(S(U)),U
240   J%(9.0-K%)=0.0
250   GOTO 280
270   J%(9.0-K%)=1.0:S%(U%)=S%(U%)-INT(2.0^K%+0.5)
272   IF INT(S%(U%))=1.0 THEN 815
280   NEXT K%
445   FOR T1%=1.0 TO X%
447   PRINT TAB((63.0-4.5*Y%)*G1%/(LEN(X$))+1.0);
450   FOR B%=1.0 TO F%(U%)
460   IF INT(J%(B%))=0.0 THEN 500
465   FOR I%=1.0 TO Y%:PRINT X$;:NEXT I%
470   GOTO 600
500   FOR I%=1.0 TO Y%
510   FOR I1%=1.0 TO LEN(X$)
520   PRINT " ";:NEXT I1%
530   NEXT I%
600   NEXT B%
```

```
620   PRINT
630   NEXT T1%
700   NEXT U%
750   FOR H%=1.0 TO 2.0*X%:PRINT :NEXT H%
800   NEXT T%
806   REM FOR N=1.0 TO 5.0:PRINT :NEXT N
810   POKE #131,1:GOTO 5
812   FOR N%=1.0 TO 7.0*X%:PRINT :NEXT N%
813   GOTO 800
815   F%(U%)=9.0-K%:GOTO 445
899   DATA " ",0,0,0,0,0,0,0
900   DATA "A",505,37,35,34,35,37,505
901   DATA "G",125,131,258,258,290,163,101
902   DATA "E",512,274,274,274,274,258,258
903   DATA "T",2,2,2,512,2,2,2
904   DATA "W",256,257,129,65,129,257,256
905   DATA "L",512,257,257,257,257,257,257
906   DATA "S",69,139,274,274,274,163,69
907   DATA "O",125,131,258,258,258,131,125
908   DATA "N",512,7,9,17,33,193,512
909   DATA "F",512,18,18,18,18,2,2
910   DATA "K",512,17,17,41,69,131,258
911   DATA "B",512,274,274,274,274,274,239
912   DATA "D",512,258,258,258,258,131,125
913   DATA "H",512,17,17,17,17,17,512
914   DATA "M",512,7,13,25,13,7,512
915   DATA "?",5,3,2,354,18,11,5
916   DATA "U",128,129,257,257,257,129,128
917   DATA "R",512,18,18,50,82,146,271
918   DATA "P",512,18,18,18,18,18,15
919   DATA "Q",125,131,258,258,322,131,381
920   DATA "Y",8,9,17,481,17,9,8
921   DATA "V",64,65,129,257,129,65,64
922   DATA "X",388,69,41,17,41,69,388
923   DATA "Z",386,322,290,274,266,262,260
924   DATA "I",258,258,258,512,258,258,258
925   DATA "C",125,131,258,258,258,131,69
926   DATA "J",65,129,257,257,257,129,128
927   DATA "1",0,0,261,259,512,257,257
928   DATA "2",261,387,322,290,274,267,261
929   DATA "*",69,41,17,512,17,41,69
930   DATA "3",66,130,258,274,266,150,100
931   DATA "4",33,49,41,37,35,512,33
932   DATA "5",160,274,274,274,274,274,226
933   DATA "6",194,291,293,297,305,289,193
934   DATA "7",258,130,66,34,18,10,8
935   DATA "8",69,171,274,274,274,171,69
936   DATA "9",263,138,74,42,26,10,7
937   DATA "=",41,41,41,41,41,41,41
938   DATA "!",1,1,1,384,1,1,1
939   DATA "0",57,69,131,258,131,69,52
940   DATA ".",1,1,129,449,129,1,1
1000  GOTO 5
1010  REM MET DEZE LARGE TEXT KUNT U TEKSTEN NAAR GEWENSTE GROOTTE MAKEN.
1020  REM VOORBEELD 1 IS: hor=4  vert=5 TAB=N  tekst=X   kar=DAI
1030  REM VOORBEELD 2 IS: hor=2  vert=4 TAB=N  tekst=DAI kar=HALLO
1040  REM m.b.v TAB BEGINT DE PRINTER TAB(40) VERDER
1050  END
```

BASICODE II

HET BASICODE MACHINETAAL PROGRAMMA

MET DIT PROGRAMMA IS HET MOGELIJK PROGRAMMA'S MET ANDERE COMPUTERS UIT TE WISSELEN IN BASICODE FORMAAT.
HET PROGRAMMA WORDT GELADEN DOOR ACHTEREENVOLGENS IN TE TOETSEN :'UT','R' EN RETURN. NA HET LADEN TOETST U 'B' DAN IS DE COMPUTER TERUG IN DE BASIC MODE.
HET PROGRAMMA WORDT GESTART DOOR CALLM #300.
DAN IS ER KEUS UIT 7 MOGELIJKHEDEN:

1 HET LADEN VAN EEN BASICODE PROGRAMMA: HIERBIJ WORDT AUTOMATISCH HET MID$-STATEMENT GECORRIGEERD (DEZE IS BIJ DE DAI AFWIJKEND VAN ANDERE COMPUTERS).
2 WEGSCHRIJVEN VANAF REGEL 1000 VAN HET AANWEZIGE PROGRAMMA. OOK HIER WORDT HET MID$-STATEMENT GECORRIGEERD;BOVENDIEN WORDEN DE SPATIES TUSSEN REGELNUMMER EN REGEL ONDERDRUKT.
3 HET LADEN VAN EEN BASICODE PROGRAMMA (ZONDER CORRECTIE).
4 WEGSCHRIJVEN VAN HET BASIC PROGRAMMA (ZONDER CORRECTIE).
5 WISSEN VAN HET AANWEZIGE BASIC PROGRAMMA EN HET (OPNIEUW) INVOEREN VAN DE BASICODE SUBROUTINES.
6 HET MERGEN VAN DE BASICODE SUBROUTINES MET HET AANWEZIGE PROGRAMMA.
7 HET STARTEN VAN HET (NIEUWE) BASICODE PROGRAMMA.

VOOR HET LADEN VAN EEN BASICODE PROGRAMMA,WORDT HET AANWEZIGE PROGRAMMA NAAR DE EDITBUFFER GESTUURD (DIT DUURT ENIGE TIJD). ALS DIT KLAAR IS VERSCHIJNT DE MEDEDELING 'TYPE SPACE'. DAN START U DE RECORDER EN DRUKT U OP DE SPATIEBALK ALS U DE FLUITTOON (= LEADER) HOORT.
DE REST GEBEURT AUTOMATISCH.
ALS HET PROGRAMMA GELADEN IS VERSCHIJNT EEN VAN DE VOLGENDE MEDEDELINGEN:
1 DATA DROPOUT ERROR (BIJ EEN DROPOUT OP DE BAND).
2 CHECKSUM ERROR (STORING OP DE BAND).
3 NO TAPE ERROR (GEEN BANDFOUT).
OP DIT MOMENT WORDT DE EDITBUFFER GELEEGD,WAARNA HET PROGRAMMA GEREED VOOR GEBRUIK IS. MOGELIJK ZIJN ER ENKELE KLEINE FOUTJES BV SYNTAX ERROR,OVERFLOW ETC.DEZE ZIJN OP DE GEBRUIKELIJKE MANIER IN DE EDITMODE TE VERBETEREN.

VOOR HET WEGSCHRIJVEN VAN HET PROGRAMMA,WORDT DIT VERPLAATST NAAR EEN BUFFER. DIT DUURT ENIGE TIJD (AFHANKELIJK VAN DE LENGTE VAN HET PROGRAMMA),WAARNA DE MEDEDELING 'SET RECORD,START TAPE TYPE SPACE'VERSCHIJNT. NU DRUKT U OP DE SPATIEBALK,WAARNA DE BASICODE UIT DE COMPUTER (CASSETTE INTERFACE) KOMT.

BASICODE II

THE BASICODE MACHINE LANGUAGE PROGRAM.

WITH THIS PROGRAM IT IS POSSIBLE TO EXCHANGE PROGRAMS WITH OTHER COMPUTERS.
THIS PROGRAM CAN BE LOADED BY KEYING (FROM BASIC):
'UT','R' AND RETURN . WHEN THIS IS DONE TYPE 'B'
THE COMPUTER IS THEN BACK IN BASIC MODE.
THIS TRANSLATION PROGRAM IS STARTED BY CALLM #300.
THERE IS THEN A CHOICE OF 7 POSSIBILITIES:

1 LOADING A BASICODE PROGRAM FROM TAPE:HERE,THE MID$-
  STATEMENT WILL BE AUTOMATICALLY CORRECTED (THIS
  STATEMENT IS DIFFERENT FROM OTHER COMPUTERS).
2 SAVING FROM LINE 1000 OF THE PRESENT BASICODE PROGRAM:
  ALSO HERE THE MID$-STATEMENT WILL BE CORRECTED;BESIDES,THE
  SPACES BETWEEN LINE NUMBER AND LINE WILL BE SUPPRESSED.
3 LOADING OF THE BASICODE PROGRAM (WITHOUT ANY CORRECTION).
4 SAVING OF THE TOTAL PROGRAM (WITHOUT ANY CORRECTION).
5 CANCELLING THE PRESENT PROGRAM AND (RE)STORING THE
  BASICODE SUBROUTINES.
6 MERGING THE BASICODE SUBROUTINES WITH THE PRESENT
  PROGRAM.
7 STARTING THE (NEW) BASICODE PROGRAM.

BEFORE LOADING A BASICODE PROGRAM, THE PRESENT PROGRAM
WILL BE SENT TO THE EDIT BUFFER (THIS TAKES SOME TIME).
WHEN THIS IS READY THE ANNOUNCEMENT 'TYPE SPACE' APPEARS.
THEN YOU CAN START THE RECORDER AND TYPE SPACE WHEN THE
LEADER IS HEARD.THE REST TAKES PLACE AUTOMATICALLY.
WHEN THIS ALL IS DONE AND LOADED THE FOLLOWING ANNOUNCE-
MENTS CAN APPEAR:
1 DATA DROPOUT ERROR (IF THERE IS A DROPOUT ON TAPE).
2 CHECKSUM ERROR (FAILURE ON TAPE).
3 NO TAPE ERROR (IF EVERYTHING ON TAPE IS OK).
AT THIS MOMENT THE EDITBUFFER WILL BE EMPTIED AND AFTER
THIS THE PROGRAM IS READY FOR USE.THERE ARE POSSIBLY
A FEW SMALL ERRORS SUCH AS E.G. SYNTAX ERROR ,OVERFLOW ETC.
THESE WILL BE CORRECTABLE AS USUAL IN THE EDITMODE.

BEFORE SAVING THE PROGRAM, IT WILL BE MOVED INTO A BUFFER.
THIS ALSO TAKES SOME TIME (DEPENDING ON THE LENGTH OF
PROGRAM), WHEREAFTER THE ANNOUNCEMENT 'SET RECORD ,START
TAPE AND TYPE SPACE'APPEARS. THE SPACE KEY SHOULD NOW BE
PRESSED IN. THE BASICODE PROGRAM THEN COMES OUT OF THE
COMPUTER (CASSETTE INTERFACE).

```
002                     POPALL  EQU   :C14D     POP ALL REGISTERS AND RETURN
003                     CASST   EQU   :D42E     START CASSETTE MOTORS
004                     CASSP   EQU   :D445     STOP CASSETTE MOTORS
005                     TIMER   EQU   :D54B     PRECISION TIMER TIME IN L
006                     WSPACE  EQU   :D6DA
007                     SRSTTS  EQU   :DAFF     STRT TAPE,SET REC,TYPE SPACE
008                     PSTR    EQU   :DB32     PRINT A STRING
009                     FILLBU  EQU   :DD75     FILL EDIT BUFFER
010                     ALFA    EQU   :DE02     TEST A-Z
011                     COMPAR  EQU   :DE14     COMPARE  DE - HL
012                     NEW     EQU   :DEB8
013                     LIEDBU  EQU   :E19F     LIST IN EDIT BUFFER
014                     DELEDI  EQU   :E228     MOVE EDIT BUFFER TO BASIC
015                     INITED  EQU   :E26C     INIT EDITMODE
016                     CURSOR  EQU   :0075
017                     EDBBGN  EQU   :00A2     STARTADDRESS EDITBUFFER
018                     EDBPTR  EQU   :00A4     (EDIT) INPUT POINTER
019                     EDBMAX  EQU   :00A6     END OF EDITBUFFER
020                     INSW    EQU   :296
021                     INPMEM  EQU   :2EC      CASS.INP.MEMORY
022                     CHKSUM  EQU   :2ED      CHECKSUM
023                     ENDFLG  EQU   :2EE
024                     BASTAB  EQU   :2EF
025                     BTPTR   EQU   :2F1
026                     MID$FG  EQU   :2F3
027                     PTRIN   EQU   :2F4
028                     PTROUT  EQU   :2F6
029                     OLDPTR  EQU   :2F8
030                     STRPTR  EQU   :2FA
031                     PORMEM  EQU   :2FC
032                     TMPPTR  EQU   :2FD
033                     SPCFLG  EQU   :2FF
034                     INPORT  EQU   :FD00     CASSETTE INPUT PORT
035                     OUTPRT  EQU   :FD06     CASSETTE OUTPUT PORT
036                             ORG   :300
037 0300 C32C07     BASTRT  JMP   SLOT
038 0303 C36703     READ    JMP   R         READ BASICODE
039 0306 C37203     READCO  JMP   RCO       READ W MID$ CORR
040 0309 C35103     WRITE   JMP   W         WRITE BASICODE
041 030C C35C03     WRITCO  JMP   WCO       WRITE W MID$&SPACE CORR
042 030F C35706     WFINAL  JMP   BUFEND
043 0312 C3F005     HORSEW  JMP   PRESUB    HORSEW+1=POINTER
044 0315 C36604     HORSER  JMP   FIN       HORSER+1=POINTER
045 0318 CDB8DE     NE&BAS  CALL  NEW       CANCEL WHOLE PROGRAM
046 031B C5         PREBAS  PUSH  B         RESTORE BASIC PART
047 031C D5                 PUSH  D
048 031D E5                 PUSH  H
049 031E F3                 DI
050 031F 213303             LXI   H,BASIC
051 0322 22E102             SHLD  :2E1
052 0325 AF                 XRA   A
053 0326 3C                 INR   A
054 0327 329602             STA   INSW
055 032A 2AEF02             LHLD  BASTAB    START OF BASICODE PROGR
056 032D 22F102             SHLD  BTPTR
057 0330 C33603             JMP   Y
058 0333 C5         BASIC   PUSH  B
059 0334 D5                 PUSH  D
060 0335 E5                 PUSH  H
061 0336 2AF102     Y       LHLD  BTPTR
062 0339 7E                 MOV   A,M
```

```
063 033A 23                 INX   H

PAGE 02    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

064 033B 22F102             SHLD  BTPTR
065 033E FE00               CPI   :00
066 0340 C24D03             JNZ   POPEXP
067 0343 329602             STA   INSW
068 0346 21B4DD             LXI   H,:DDB4
069 0349 22E102             SHLD  :2E1
070 034C FB                 EI
071 034D E1         POPEXP  POP   H
072 034E D1                 POP   D
073 034F C1                 POP   B
074 0350 C9                 RET
075 0351 E5         W       PUSH  H
076 0352 21F005             LXI   H,PRESUB
077 0355 221303             SHLD  HORSEW+1
078 0358 E1                 POP   H
079 0359 C38405             JMP   CSTWRT
080 035C E5         WCO     PUSH  H
081 035D 21B905             LXI   H,SPCABS
082 0360 221303             SHLD  HORSEW+1
083 0363 E1                 POP   H
084 0364 C38405             JMP   CSTWRT
085 0367 E5         R       PUSH  H
086 0368 216604             LXI   H,FIN       WITHOUT MID$-
087 036B 221603             SHLD  HORSER+1    CORRECTION
088 036E E1                 POP   H
089 036F C37A03             JMP   INIT
090 0372 E5         RCO     PUSH  H
091 0373 214904             LXI   H,COR       WITH MID$-
092 0376 221603             SHLD  HORSER+1    CORRECTION
093 0379 E1                 POP   H
094 037A F5         INIT    PUSH  PSW
095 037B C5                 PUSH  B
096 037C D5                 PUSH  D
097 037D E5                 PUSH  H
098 037E 3EFF               MVI   A,:FF       *PLACE BLACK SQUARE
099 0380 327500             STA   CURSOR      *ON CURSOR
100 0383 CD6CE2             CALL  INITED      INIT EDITBUFFER
101 0386 CD9FE1             CALL  LIEDBU      LIST IN EDIT BUFFER
102 0389 AF                 XRA   A
103 038A 323101             STA   :131        RESET OUTPUT POINTER
104 038D 32EE02             STA   ENDFLG      RESET ENDFLG
105 0390 CD2ED4             CALL  CASST       CASS.START ROUTINE IN DAI
106 0393 215305             LXI   H,MESTYS    MESSAGE 'TYPE SPACE'
107 0396 CD32DB             CALL  PSTR
108 0399 CDDAD6             CALL  WSPACE      WAIT FOR SPACEBAR PRESSED
109 039C 216005             LXI   H,MESLDB    MESSAGE 'LOADING BASICODE'
110 039F CD32DB             CALL  PSTR
111 03A2 CD2ED4             CALL  CASST       CASS.START ROUTINE IN DAI
112 03A5 2100FD     START   LXI   H,INPORT    ON FIRST TIME (ON START OF
113                 *                         PROGRAM)WAIT FOR CHANGING
114                 *                         INPUT.NO CHECK FOR TIME.
115 03A8 4E                 MOV   C,M
116 03A9 1E00       STARTS  MVI   E,:00
117 03AB 7E                 MOV   A,M
118 03AC A9                 XRA   C
119 03AD F2AB03             JP    STARTS+2
120 03B0 F3                 DI
121 03B1 4E                 MOV   C,M
122 03B2 1C         WAITFL  INR   E           WAIT FOR AN EDGE
123                 *                         ON THE INPUT PORT
124 03B3 CA1404             JZ    ERROR       TOO LONG ERROR
```

```
125 03B6 7E                MOV   A,M

PAGE 03    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

126 03B7 A9                XRA   C
127 03B8 F2B203            JP    WAITFL     WAIT FOR CHANGING INPUT
128 03BB 4E                MOV   C,M
129 03BC 1E78       STRTBT MVI   E,:78
130 03BE 2E0D              MVI   L,:0D      ****************************
131 03C0 CD4BD5            CALL  TIMER
132 03C3 1C         COUNT1 INR   E
133 03C4 CA1404            JZ    ERROR      TOO LONG ERROR
134 03C7 7E                MOV   A,M
135 03C8 A9                XRA   C
136 03C9 F2C303            JP    COUNT1
137 03CC 4E                MOV   C,M
138 03CD 1C                INR   E
139 03CE F2BC03            JP    STRTBT     TOO SHORT FOR STARTBIT
140 03D1 1E8A              MVI   E,:8A
141 03D3 06FF              MVI   B,:FF
142 03D5 C30905            JMP   SB
143 03D8 7B         BYTE   MOV   A,E
144 03D9 C620              ADI   :20        CARRY SET :BIT='1'
145 03DB 78                MOV   A,B
146 03DC 1F                RAR              SHIFT
147 03DD 47                MOV   B,A
148 03DE DAF804            JC    BIT        CARRY SET = NO STARTBIT
149                 *                       SHIFTED IN CARRY
150 03E1 FB         CSUM   EI
151 03E2 3AEE02            LDA   ENDFLG
152 03E5 B7                ORA   A
153 03E6 C21E04            JNZ   STOTEX     IF ENDFLAG SET
154 03E9 3AED02            LDA   CHKSUM
155 03EC A8                XRA   B          MODIFY CHECKSUM
156 03ED 32ED02            STA   CHKSUM
157 03F0 78                MOV   A,B
158 03F1 E67F              ANI   :7F
159 03F3 FE03              CPI   :03        ASCII 3 = END OF TEXT
160 03F5 CA0804            JZ    SETFLG
161 03F8 FE02              CPI   :02        CLEAR CHECCKSUM
162 03FA CA0E04            JZ    CLCHSM
163 03FD CD75DD     RETURN CALL  FILLBU     STORE CHARACTER IN
164                 *                       (EDIT)BUFFER AND MODIFY
165                 *                       POINTER
166 0400 3E20              MVI   A,:20
167 0402 327500            STA   CURSOR
168 0405 C3A503            JMP   START
169 0408 32EE02     SETFLG STA   ENDFLG     SET ENDFLAG
170 040B C3A503            JMP   START
171 040E 32ED02     CLCHSM STA   CHKSUM
172 0411 C3A503            JMP   START
173 0414 FB         ERROR  EI
174 0415 214005            LXI   H,MESERR   MESSAGE 'DATA DROPOUT ERROR'
175 0418 CD32DB            CALL  PSTR       PRINT MESSAGE
176 041B C33604            JMP   LAST
177 041E 3AED02     STOTEX LDA   CHKSUM
178 0421 90                SUB   B
179 0422 E67F              ANI   :7F
180 0424 CA3004            JZ    GOOD
181 0427 212205     FALSE  LXI   H,MESFAL   MESSAGE 'CHECKSUM ERROR'
182 042A CD32DB            CALL  PSTR       PRINT MESSAGE
183 042D C33604            JMP   LAST
184 0430 213105     GOOD   LXI   H,MESGD    MESSAGE ' OK '
185 0433 CD32DB            CALL  PSTR       PRINT MESSAGE
186 0436 3E0D       LAST   MVI   A,:0D
```

```
187 0438 EF                     RST   5

PAGE 04    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

188 0439 03                     DATA  :3
189 043A 3E0D                   MVI   A,:0D
190 043C EF                     RST   5
191 043D 03                     DATA  3
192 043E 3E5F                   MVI   A,:5F
193 0440 327500                 STA   CURSOR    ' - 'IN CURSOR
194 0443 CD45D4                 CALL  CASSP     STOP CASSETTE MOTORS
195 0446 C31503                 JMP   HORSER
196 0449 0600         COR       MVI   B,:00
197 044B 217505                 LXI   H,MID$
198 044E 22FA02                 SHLD  STRPTR
199 0451 2AA400                 LHLD  EDBPTR    EDIT INPUT POINTER
200 0454 22F802                 SHLD  OLDPTR    STORE IN POINTER
201 0457 EB                     XCHG
202 0458 2AA200                 LHLD  EDBBGN    START OF EDITBUFFER
203 045B 7E           CHKWRD    MOV   A,M
204 045C 23                     INX   H
205 045D CD7004                 CALL  WDRCZR    CALL WORDRECOGNIZER
206 0460 CD14DE                 CALL  COMPAR    END OF EDITBUFFER?
207 0463 C25B04                 JNZ   CHKWRD
208 0466 AF           FIN       XRA   A         A=00
209 0467 CD75DD                 CALL  FILLBU    LAST BYTE IN BUFFER
210                     *                       (MUST BE A '00')
211 046A CD28E2                 CALL  DELEDI    DELETE EDIT BUFFER
212 046D C34DC1                 JMP   POPALL    POP REGISTERS AND RETURN
213                     *
214 0470 E5           WDRCZR    PUSH  H         SUBROUTINE WORDRECOGNIZER
215                     *                       TO RECOGNIZE THE MID$-
216                     *                       STATEMENT
217 0471 FE0D                   CPI   :0D       END OF ROW
218 0473 CA9B04                 JZ    CKM$FG    CHECK MID$FLAG (IN ROW)
219 0476 FE20                   CPI   :20       SPACE
220 0478 CA9604                 JZ    STORE     NO FURTHER ACTION
221 047B 4F                     MOV   C,A       C=CHARACTER
222 047C 78                     MOV   A,B       B=COUNTER
223 047D FE05                   CPI   :5
224 047F CAAD04                 JZ    COMMA     5 COUNTS AND A COMMA MEANS
225                     *                       FILL WITH  '- 1+'
226                     *                       E.G. MID$("ABC",-1+N,2)
227 0482 79           NCOMMA    MOV   A,C
228 0483 2AFA02                 LHLD  STRPTR    NO COMMA - STRINGPOINTER
229 0486 BE                     CMP   M
230 0487 CAD004                 JZ    INRB      INCREMENT STRINGPOINTER
231 048A 78                     MOV   A,B
232 048B FE05                   CPI   :5
233 048D FADC04                 JM    CLRB
234 0490 79                     MOV   A,C
235 0491 FE29                   CPI   :29       ')' EG MID$(A$(0,5),N,2)
236 0493 CAD804                 JZ    DCRB
237 0496 CD75DD       STORE     CALL  FILLBU
238 0499 E1                     POP   H
239 049A C9                     RET
240                     *
241 049B 3AF302       CKM$FG    LDA   MID$FG    MID$FLAG MEANS THIS ROW
242                     *                       CONTAINS A STATEMENT 'MID$'
243 049E B7                     ORA   A
244 049F C2E804                 JNZ   ENCODE
245 04A2 2AF802                 LHLD  OLDPTR    NO MID$ STATEMENT  OLD POINT
246 04A5 22A400                 SHLD  EDBPTR    IN EDITBUFFERPOINTER
247 04A8 0600                   MVI   B,:00
248 04AA C39904                 JMP   STORE+3
```

```
249 04AD 79          COMMA   MOV   A,C

PAGE 05    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

250 04AE FE2C                CPI   :2C       CHECK','
251 04B0 C28204              JNZ   NCOMMA
252 04B3 32F302              STA   MID$FG    SET MID$FLAG
253 04B6 217505              LXI   H,MID$
254 04B9 22FA02              SHLD  STRPTR
255 04BC 0600                MVI   B,:00
256 04BE CD75DD              CALL  FILLBU
257 04C1 3E2D                MVI   A,:2D
258 04C3 CD75DD              CALL  FILLBU
259 04C6 3E31                MVI   A,:31
260 04C8 CD75DD              CALL  FILLBU
261 04CB 3E2B                MVI   A,:2B
262 04CD C39604              JMP   STORE
263                  *
264 04D0 23          INRB    INX   H
265 04D1 04                  INR   B
266 04D2 22FA02              SHLD  STRPTR
267 04D5 C39604              JMP   STORE
268 04D8 05          DCRB    DCR   B
269 04D9 C39604              JMP   STORE
270 04DC 217505      CLRB    LXI   H,MID$    DEFAULT VALUE $POINTER
271 04DF 22FA02              SHLD  STRPTR
272 04E2 0600                MVI   B,:00
273 04E4 79                  MOV   A,C
274 04E5 C39604              JMP   STORE
275 04E8 AF          ENCODE  XRA   A
276 04E9 32F302              STA   MID$FG    CLEAR MID$FLAG
277 04EC 2AA400              LHLD  EDBPTR
278 04EF 23                  INX   H
279 04F0 22F802              SHLD  OLDPTR    POINTER INCL CARR RET
280 04F3 3E0D                MVI   A,:0D
281 04F5 C39604              JMP   STORE
282                  *
283 04F8 1E7A        BIT     MVI   E,:7A
284 04FA 4E          BITSEC  MOV   C,M       SECOND TIME (=SECOND CYCLE
285                  *                       OF 2400 HZ)
286 04FB 2E0D                MVI   L,:0D     ****************************
287 04FD CD4BD5              CALL  TIMER
288 0500 1C          COUNT2  INR   E
289 0501 CA1404              JZ    ERROR
290 0504 7E                  MOV   A,M
291 0505 A9                  XRA   C
292 0506 F20005              JP    COUNT2
293 0509 4E          SB      MOV   C,M
294 050A 2E0D                MVI   L,:0D     ****************************
295 050C CD4BD5              CALL  TIMER
296 050F 53                  MOV   D,E
297 0510 1D          COUNT3  DCR   E
298 0511 CA1404              JZ    ERROR
299 0514 7E                  MOV   A,M
300 0515 A9                  XRA   C
301 0516 F21005              JP    COUNT3
302 0519 14                  INR   D
303 051A FAD803              JM    BYTE
304 051D 1EEF                MVI   E,:EF
305 051F C3FA04              JMP   BITSEC
306 0522 0E          MESFAL  DATA  :0E       LENGTH OF STRING
307 0523 434845              ASC   'CHECKSUM ERROR'
308 0531 0E          MESGD   DATA  :0E
309 0532 4E4F20              ASC   'NO TAPE ERRORS'
310 0540 12          MESERR  DATA  :12
```

```
311 0541 444154              ASC   'DATA DROPOUT ERROR'


PAGE 06    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

312 0553 0C          MESTYS  DATA  :0C
313 0554 0C0D                DATA  :0C,:0D
314 0556 545950              ASC   'TYPE SPACE'
315 0560 14          MESLDB  DATA  :14
316 0561 0C0D                DATA  :0C,:0D
317 0563 4C4F41              ASC   'LOADING BASICODE'
318 0573 0D0D                DATA  :0D,:0D
319 0575 4D4944      MID$    ASC   'MID$((((((((((('
320                  *      ************************
321                  *      * BASICODE WRITE PART *
322                  *      ************************
323 0584 F5          CSTWRT  PUSH  PSW        COLD START OF BASICODE
324 0585 C5                  PUSH  B          WRITE ROUTINE
325 0586 D5                  PUSH  D
326 0587 E5                  PUSH  H
327 0588 F3                  DI
328 0589 2AA302              LHLD  :2A3       END OF SYMBOL TABLE
329 058C 24                  INR   H
330 058D 22F402              SHLD  PTRIN      SET INPUT &
331 0590 22F602              SHLD  PTROUT     OUTPUT POINTER
332 0593 AF                  XRA   A
333 0594 32FF02              STA   SPCFLG     RESET SPACEFLAG
334 0597 3EC3                MVI   A,:C3      *SET POINTER TO
335 0599 32DD02              STA   :2DD       *(WARM)START OF
336 059C 21B205              LXI   H,SWRITE   *WRITEROUTINE
337 059F 22DE02              SHLD  :2DE       *
338 05A2 3E03                MVI   A,:03      OUTPUT SWITCH
339 05A4 323101              STA   :131
340 05A7 FB                  EI
341 05A8 21ED02              LXI   H,CHKSUM   ALSO USED BY READROUTINE
342 05AB 3E82                MVI   A,:82      START OF TEXT (02)
343                  *                        8TH BIT MUST BE A '1'
344 05AD 3683                MVI   M,:83      CHECKSUM INCLUSIVE
345                  *                        END OF TEXT (03)
346 05AF C3B605              JMP   SWRITE+4   FIRST TIME NO PUSHING
347 05B2 F5          SWRITE  PUSH  PSW
348 05B3 C5                  PUSH  B
349 05B4 D5                  PUSH  D
350 05B5 E5                  PUSH  H
351 05B6 C31203              JMP   HORSEW
352                  * SUBROUTINE TO ABSORB THE SPACES
353                  * BETWEEN LINENUMBER AND LINE
354 05B9 FE0D        SPCABS  CPI   :0D        RETURN
355 05BB CC3A06              CZ    RSFLAG     RESET FLAG
356 05BE CD02DE              CALL  ALFA       TEST A-Z
357 05C1 DC4706              CC    STFLAG     SET FLAG
358 05C4 FE20                CPI   :20        SPACE?
359 05C6 CA4B06              JZ    ABSORB
360 05C9 2AFD02              LHLD  TMPPTR
361 05CC 6F                  MOV   L,A        L =CHARACTER
362 05CD 7C                  MOV   A,H        H=STRINGCOUNTER
363 05CE FE05                CPI   :5
364 05D0 CA0806              JZ    WCOMMA
365 05D3 EB          WNCOMA  XCHG
366 05D4 2AFA02              LHLD  STRPTR
367 05D7 7B                  MOV   A,E        E=CHARACTER
368 05D8 BE                  CMP   M
369 05D9 CA2906              JZ    INCREG     INCREMENT COUNTER
370 05DC 7A                  MOV   A,D        D=STRINGCOUNTERC
371 05DD FE05                CPI   :5
372 05DF FA3206              JM    CLRREG     CLEAR COUNTER
```

```
373 05E2 7B                 MOV   A,E

PAGE 07    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

374 05E3 FE29               CPI   :29
375 05E5 CA2E06             JZ    DECREG     DECREMENT COUNTER
376 05E8 22FA02     WSTORE  SHLD  STRPTR
377 05EB EB                 XCHG
378 05EC 22FD02             SHLD  TMPPTR
379 05EF 7D                 MOV   A,L
380 05F0 CDF605     PRESUB  CALL  SUB
381 05F3 C34DC1             JMP   POPALL     RETURN
382 05F6 F680       SUB     ORI   :80
383 05F8 47                 MOV   B,A        SAVE CHARACTER IN B
384 05F9 21ED02             LXI   H,CHKSUM   OLD CHECKSUM
385 05FC AE                 XRA   M          MODIFY
386 05FD 77                 MOV   M,A        STORE CHECKSUM
387 05FE 2AF402             LHLD  PTRIN      INPUT POINTER
388 0601 70                 MOV   M,B        STORE CHARACTER
389 0602 23                 INX   H          INCREMENT /
390 0603 22F402             SHLD  PTRIN      MODIFY POINTER
391 0606 78                 MOV   A,B        RESTORE CHARACTER IN ACCU
392 0607 C9                 RET
393 0608 7D         WCOMMA  MOV   A,L
394 0609 FE2C               CPI   :2C        CHECK','
395 060B C2D305             JNZ   WNCOMA     NO COMMA
396 060E 217505             LXI   H,MID$
397 0611 22FA02             SHLD  STRPTR
398 0614 3E2C               MVI   A,:2C      (=',')
399 0616 CDF605             CALL  SUB
400 0619 3E20               MVI   A,:20      (='-')
401 061B CDF605             CALL  SUB
402 061E 3E31               MVI   A,:31      (='1')
403 0620 CDF605             CALL  SUB
404 0623 212B00             LXI   H,:002B    H=STRINGCOUNTER(=00)
405 0626 C3D305             JMP   WNCOMA     L=CHARACTER(2B='+')
406 0629 14         INCREG  INR   D
407 062A 23                 INX   H
408 062B C3E805             JMP   WSTORE
409 062E 15         DECREG  DCR   D
410 062F C3E805             JMP   WSTORE
411 0632 217505     CLRREG  LXI   H,MID$
412 0635 1600               MVI   D,:00
413 0637 C3E805             JMP   WSTORE
414 063A F5         RSFLAG  PUSH  PSW
415 063B AF                 XRA   A
416 063C 32FF02             STA   SPCFLG
417 063F F1                 POP   PSW
418 0640 217505             LXI   H,MID$
419 0643 22FA02             SHLD  STRPTR
420 0646 C9                 RET
421 0647 32FF02     STFLAG  STA   SPCFLG
422 064A C9                 RET
423 064B 3AFF02     ABSORB  LDA   SPCFLG
424 064E B7                 ORA   A
425 064F 3E20               MVI   A,:20
426 0651 CA4DC1             JZ    POPALL
427 0654 C3F005             JMP   PRESUB
428 0657 F5         BUFEND  PUSH  PSW        BUFFER END
429 0658 C5                 PUSH  B
430 0659 D5                 PUSH  D
431 065A E5                 PUSH  H
432 065B 213101             LXI   H,:131     OUTPUT SWITCH
433 065E 3600               MVI   M,:00      BACK
434 0660 21ED02             LXI   H,CHKSUM
```

```
435 0663 46                     MOV   B,M        CHECKSUM IN B

PAGE 08    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

436 0664 2AF402                 LHLD  PTRIN
437 0667 3683                   MVI   M,:83      END OF TEXT IN BUFFER
438 0669 23                     INX   H          INCREMENT A POINTER
439 066A 70                     MOV   M,B        CHECKSUM IN BUFFER
440 066B 22F402                 SHLD  PTRIN
441 066E CDFFDA                 CALL  SRSTTS     PRINT SET RECORD ,START
442                     *                        TAPE TYPE SPACE
443 0671 9CDB                   DBL   :DB9C
444 0673 CDDAD6                 CALL  WSPACE     WAIT FOR SPACEBAR/CR
445 0676 CD2ED4                 CALL  CASST      START CASSETTE MOTORS
446 0679 F3                     DI
447                     * THE TIMING IS CALCULABLE :
448                     * FOR PROGRAM IN RAM : EVERY STATE ~1uS
449                     * FOR PROGRAM IN ROM : EVERY STATE 0.5uS
450                     * EXPLANATION :THE VIDEO PROCESSOR USES ALSO
451                     * THE RAM IN TURN:THAT MEANS A FACTOR 2 SLOWER
452                     * EXEPT WHEN SCREENSPOT GOES BACK FROM UNDERSIDE
453                     * TO UPPERSIDE OF THE SCREEN ,THE VIDEOPROCESSOR
454                     * IS NOT USING THE RAM.
455                     *
456                     *                      STATES   TIME AFTER LAST FALLING
457                     *                               EDGE   ( uS )
458                     *                         *      *      *      *
459 067A 3A4000                 LDA   :0040
460 067D F601                   ORI   :01
461 067F 32FC02                 STA   PORMEM
462 0682 1106FD                 LXI   D,OUTPRT   10
463 0685 010018                 LXI   B,:1800    10
464                     *
465                     * TIMER   L=(208-T-42)/7.5
466                     *
467 0688 2E0C           LEADER  MVI   L,:0C      7              >58
468 068A CDF506                 CALL  BITONE     17     >27      75>
469 068D 0B                     DCX   B          5       32
470 068E 78                     MOV   A,B        5       37
471 068F B7                     ORA   A          4       41
472 0690 C28806                 JNZ   LEADER     10      51>
473 0693 2E0E                   MVI   L,:0E      7       46
474 0695 CDF506                 CALL  BITONE     17      63>   >27
475 0698 2AF602         DATA    LHLD  PTROUT     16             43
476 069B D5                     PUSH  D          11             54
477 069C 0609                   MVI   B,:09      7              61
478 069E 7E                     MOV   A,M        7              68
479 069F 2E07                   MVI   L,:07      7              75
480 06A1 F5             BITW    PUSH  PSW        11             86     73
481 06A2 D41507                 CNC   BITZER     11/17  >15     103>   >90
482 06A5 DCF506                 CC    BITONE     11/17   26            >101
483 06A8 F1                     POP   PSW        10      36
484 06A9 1F                     RAR              4       40
485 06AA 05                     DCR   B          5       45
486 06AB 2E09                   MVI   L,:09      7       52
487 06AD C2A106                 JNZ   BITW       10      62>
488 06B0 2AF602                 LHLD  PTROUT     16      78
489 06B3 23                     INX   H          5       83
490 06B4 22F602                 SHLD  PTROUT     16      99
491 06B7 2B                     DCX   H          5      104
492 06B8 E5                     PUSH  H          11     115
493 06B9 2E03                   MVI   L,:03      7      122
494 06BB 00                     NOP              4      126
495 06BC CDF506                 CALL  BITONE     17     143>
496 06BF E1                     POP   H          10      >37
```

```
497 06C0 EB                XCHG          4        41

PAGE 09    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

498 06C1 2AF402            LHLD  PTRIN   16       58
499 06C4 7C                MOV   A,H     5        63
500 06C5 BA                CMP   D       7        70
501 06C6 7D                MOV   A,L     5        75
502 06C7 C2CB06            JNZ   POPD    10       85>
503 06CA BB                CMP   E       7        92
504 06CB D1        POPD    POP   D       10       102
505 06CC F5                PUSH  PSW     11       113
506 06CD 2E04              MVI   L,:04    7        120
507 06CF CDF506            CALL  BITONE  17       137>
508 06D2 F1                POP   PSW     10       >37
509 06D3 C29806            JNZ   DATA    10        47
510 06D6 01000F            LXI   B,:0F00  10        57
511 06D9 2E0B              MVI   L,:0B    7         64
512 06DB CDF506    TRAILR  CALL  BITONE  17        81>  >75>
513 06DE 0B                DCX   B       5      >32
514 06DF 78                MOV   A,B     5       37
515 06E0 B7                ORA   A       4       41
516 06E1 2E0C              MVI   L,:0C    7       48
517 06E3 C2DB06            JNZ   TRAILR  10      58>
518 06E6 3EC9              MVI   A,:C9
519 06E8 32DD02            STA   :2DD
520 06EB 210000            LXI   H,:0000
521 06EE 22DE02            SHLD  :2DE
522 06F1 FB                EI
523 06F2 C34DC1            JMP   POPALL
524 06F5 CD4BD5    BITONE  CALL  TIMER
525 06F8 3AFC02            LDA   PORMEM
526 06FB 12                STAX  D
527 06FC 3D                DCR   A
528 06FD 2E16              MVI   L,:16
529 06FF CD4BD5            CALL  TIMER
530 0702 12                STAX  D
531 0703 3C                INR   A
532 0704 2E16              MVI   L,:16
533 0706 CD4BD5            CALL  TIMER
534 0709 12                STAX  D
535 070A 3D                DCR   A
536 070B 2E16              MVI   L,:16
537 070D CD4BD5            CALL  TIMER
538 0710 12                STAX  D
539 0711 00                NOP
540 0712 00                NOP
541 0713 00                NOP
542 0714 C9                RET
543 0715 00        BITZER  NOP
544 0716 00                NOP
545 0717 00                NOP
546 0718 CD4BD5            CALL  TIMER
547 071B 2E1A              MVI   L,:1A
548 071D CD4BD5            CALL  TIMER
549 0720 3AFC02            LDA   PORMEM
550 0723 12                STAX  D
551 0724 3D                DCR   A
552 0725 2E32              MVI   L,:32
553 0727 CD4BD5            CALL  TIMER
554 072A 12                STAX  D
555 072B C9                RET
556 072C E5        SLOT    PUSH  H
557 072D 21800E            LXI   H,:0E80
558 0730 229B02            SHLD  :29B
```

```
559 0733 E1                POP   H

PAGE 10    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

560 0734 C31803            JMP   NE&BAS
561 0737             END   END             END

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

ABSORB 064B   ALFA   DE02   BASIC  0333   BASTAB 02EF
BASTRT 0300   BIT    04F8   BITONE 06F5   BITSEC 04FA
BITW   06A1   BITZER 0715   BTPTR  02F1   BUFEND 0657
BYTE   03D8   CASSP  D445   CASST  D42E   CHKSUM 02ED
CHKWRD 045B   CKM$FG 049B   CLCHSM 040E   CLRB   04DC
CLRREG 0632   COMMA  04AD   COMPAR DE14   COR    0449
COUNT1 03C3   COUNT2 0500   COUNT3 0510   CSTWRT 0584
CSUM   03E1   CURSOR 0075   DATA   0698   DCRB   04D8
DECREG 062E   DELEDI E228   EDBBGN 00A2   EDBMAX 00A6
EDBPTR 00A4   ENCODE 04E8   END    0737   ENDFLG 02EE
ERROR  0414   FALSE  0427   FILLBU DD75   FIN    0466
GOOD   0430   HORSER 0315   HORSEW 0312   INCREG 0629
INIT   037A   INITED E26C   INPMEM 02EC   INPORT FD00
INRB   04D0   INSW   0296   LAST   0436   LEADER 0688
LIEDBU E19F   MESERR 0540   MESFAL 0522   MESGD  0531
MESLDB 0560   MESTYS 0553   MID$   0575   MID$FG 02F3
NCOMMA 0482   NE&BAS 0318   NEW    DEB8   OLDPTR 02F8
OUTPRT FD06   POPALL C14D   POPD   06CB   POPEXP 034D
PORMEM 02FC   PREBAS 031B   PRESUB 05F0   PSTR   DB32
PTRIN  02F4   PTROUT 02F6   R      0367   RCO    0372
READ   0303   READCO 0306   RETURN 03FD   RSFLAG 063A
SB     0509   SETFLG 0408   SLOT   072C   SPCABS 05B9
SPCFLG 02FF   SRSTTS DAFF   START  03A5   STARTS 03A9
STFLAG 0647   STORE  0496   STOTEX 041E   STRPTR 02FA
STRTBT 03BC   SUB    05F6   SWRITE 05B2   TIMER  D54B
TMPPTR 02FD   TRAILR 06DB   W      0351   WAITFL 03B2
WCO    035C   WCOMMA 0608   WDRCZR 0470   WFINAL 030F
WNCOMA 05D3   WRITCO 030C   WRITE  0309   WSPACE D6DA
WSTORE 05E8   Y      0336
```

```
10    REM
20    REM
30    REM ~~~~~~~~~ NOTTE DI FUOCO IN MODE 1 ~~~~~~~~~~~~~
40    REM
50    MODE 1:DIM A%(40.0)
52    FOR I%=0 TO 40:READ A%:POKE (#2F1+I%),A%:NEXT
57    DOT RND(XMAX),1 RND(16.0)
60    CALLM #2F1
70    GOTO 57
100   DATA #C5,#D5,#E5,#F5,#01,#D7,#BF,#11,#EF,#BF,#00
110   DATA #00,#00,#00,#00,#00,#00,#00,#00,#00,#26,#18
120   DATA #2E,#40,#0A,#12,#0B,#1B,#2D,#C2,#09,#03,#25
130   DATA #C2,#07,#03,#F1,#E1,#D1,#C1,#C9
```

COMMANDES DANS UN PROGRAMME

```
>tSPL V1.0  PAGE 1

0000            ;                  ****************************
0000            ;                  Commandes dans un programme
0000            ;                  Instructions  en  commandes
0000            ;                  ****************************
0000            ;       Grace a ce petit programme, vous pourrez
0000            ;       utiliser des Commandes dans un programme
0000            ;       et vice-versa.
0000            ;       Par exemple :   *100 RUN 10
0000            ;Mise en route :
0000            ;  Changer les vecteurs et taper le programme(#AE0-#AFA)
0000            ;  SI ce n'est deja fait
0000            ;  (*POKE #29C,11      *NEW)
0000            ;  Taper    *UT
0000            ;           >V1 C7E0-0AE0 (space+curseur gauche)
0000            ;           >B
0000            ;Mise hors service :
0000            ;  Taper    *UT
0000            ;           >Z3
0000            ;           >B
0000            ;
0000                    ORG       0AE0H          ; Compatible S.P.U
0AE0 E1                 POP H                    ; *****************
0AE1 F3                 DI                       ; * Recopie de    *
0AE2 224300             SHLD      43H            ; *   C70E a C714 *
0AE5 F5                 PUSH PSW                 ; *               *
0AE6 3EC0               MVI A     0C0H           ; *****************
0AE8 E1                 POP H                    ; * Recopie de    *
0AE9 224100             SHLD      41H            ; *   C705 a C70A *
0AEC 67                 MOV H,A                  ; *               *
0AED AF                 XRA A                    ; *****************
0AEE E3                 XTHL                     ; Data apres RST 1
0AEF B6                 ORA M                    ; Si = 0 : Codage
0AF0 CCF80A             CZ        DATA0          ;
0AF3 E3                 XTHL                     ; Restore pile
0AF4 AF                 XRA A                    ; A=0
0AF5 C3CFC6             JMP       0C6CFH         ; Continue RST
0AF8 16C0       DATA0   MVI D     0C0H           ; Masque de test
0AFA C9                 RET                      ; pour
0AFB            ;                                  COMMAND INVALID
0AFB            ;
0AFB            ; AE0  E1 F3 22 43 00 F5 3E C0 E1 22 41 00 67 AF
0AFB            ; AEE  E3 B6 CC F8 0A E3 AF C3 CF C6 16 C0 C9
0AFB            ;
0AFB            ;Le test qui affiche le message d'erreur
0AFB            ;'COMMAND INVALID' se trouve aux adresses
0AFB            ; E035 ... de la banque 3 (Apres un RST1 DATA 0)
0AFB            ;Le test est fait comme suit :
0AFB                    ORG       0E035H
E035 7E                 MOV A,M                  ;Dans A code de l'instruction
E036 E6C0               ANI       0C0H           ;On garde les deux derniers bits
E038 A2                 ANA D                    ;Masque de test
E039 3E18               MVI A     18H
E03B CAF5D9             JZ        0D9F5H         ;message d'erreur
E03E            ;En imposant a D la valeur #C0 avant d'executer
E03E            ;ces instructions (En deviant le vecteur du RST1)
E03E            ;le jump au message d'erreur ne pourra plus se faire.
E03E            ;(Le resultat du ET n'etant jamais nul)
E03E                    END
```

Cédric
-DUFOUR-

CALLM

La fonction CALLM du basic peut etre appelée de deux façon :

1) CALLM nn
2) CALLM nn,var

Avec nn qui représente une adresse (éventuellement dans une variable) et var qui est le nom d'une variable
Dans les deux cas cértains renseignements sont placés dans les registres avant d'exécuter le sous-programme en langage machine. Ces renseignements sont donnés dans les deux cas çi-desous:

```
      1) CALLM nn
* A  contient la valeur #FF
* BC contient l'adresse de l'instruction qui suit
             le CALLM
* HL contient l'adresse nn


      2) CALLM nn,var
* A  contient une indication sur la nature de 'var'
             soit :     #04  =  FPT (Virgule flottante)
                        #14  =  INT (Entier)
                        #22  =  STR (Chaine)
* BC Idem a 1)
* HL contient l'adresse de 'var' soit:
         - Adresse de la variable si var est numérique
         - Adresse du pointeur de la variable (Vecteur)
                        si var est une chaine
```

Il est à remarquer que seul le contenu de BC est important pour le BASIC. Vos sous programmes peuvent alors s'ecrire plus simplement.

```
Ex:    C5  -   PUSH B
       xx  -   ...Programme...
       C1  -   POP B
       C9  -   RET
```

Cédric DUFOUR

```
IMPINT
10    REM MONTAGNES
12    REM F.DEBROUWERE & C.DUFOUR
14    MODE 6:COLORG 15 9 12 1:Y1=101:Z1=130
18    FOR J=1 TO 55
20    A!=RND(6)-3:B!=RND(6)-3
22    FOR I=1 TO 6
24    X=X+1
26    Y=A!*I+Y1:IF Y<1 THEN Y=1
28    Z=B!*I+Z1:IF Z<1 THEN Z=1
30    DRAW X,Z X,Y 23:DRAW X,Z+2 X,Z 22
32    DRAW X,Y X,0 21:DRAW X,Y+2 X,Y 22
34    NEXT:Y1=Y:Z1=Z:NEXT
36    I=GETC:WAIT TIME 3:IF I=0 THEN 36
38    END
```

INTEGERS

```
*DNA
    ]#L.
    PAGE 01

    001                 ************************
    002                 *    I N T E G E R S   *
    003                 ************************
    004                 *
    005                 *
    006                 *    Dit  machinetaalprogramma zorgt  ervoor dat
    007                 *    INTEGER-variabelen  naar  rechts  geschikt
    008                 *    worden.  Dit in tegenstelling tot wat  DAI
    009                 *    personal computer doet.
    010                 *    Het kan aangeroepen worden vanuit BASIC met
    011                 *        CALLM#300,VARIABLE%.
    012                 *    De integervariabele wordt uitgeprint in een
    013                 *    formaat  van  6  cijfers  met  ervoor  een
    014                 *    spatie  voor  de  positieve  en een minteken
    015                 *    voor de negatieve integers.
    016                 *    Dit kan gewijzigd worden vanuit  BASIC  met
    017                 *        POKE #371,AANT%.
    018                 *    Hierbij is  AANT%  minstens  even groot als
    019                 *    het aantal cijfers van het  langste  uit te
    020                 *    printen getal.
    021                 *    Als AANT% kleiner is dan het aantal cijfers
    022                 *    van het langste getal dan wordt het  BASIC-
    023                 *    programma onderbroken met vermelding :
    024                 *        "FORMAT ERROR".
    025                 *    Als  VARIABLE% = 0  dan  worden  er  alleen
    026                 *    spaties  ( aantal = AANT% + 1 )  geprint.
    027                 *    Dit kan  eveneens  gewijzigd  worden vanuit
    028                 *    BASIC met
    029                 *        POKE #370,#30   er  worden  een  aantal
    030                 *                        spaties en een  "0" ge-
    031                 *                        geprint.
    032                 *        POKE #370,#20   er worden enkel spaties
    033                 *                        geprint.
    034                 *
    035                 *    With   this   machinelanguageprogramm   the
    036                 *    INTEGERvariables  will  be  ordered  to the
    037                 *    right.
    038                 *    From BASIC called with
    039                 *        CALLM#300,VARIABLE%.
    040                 *    The maximum lenght  of the  integervariable
    041                 *    is  6.  This lenght  can  be  changed  from
    042                 *    BASIC with
    043                 *        POKE #371,NUMB%.
    044                 *        NUMB%  >=  lenght  of longest  integer-
    045                 *                   variable
    046                 *    If  NUMB% < the lenght of the variable then
    047                 *    there will be  a BREAK  of the BASICprogram
    048                 *    and an ERROR report : "FORMAT ERROR".
    049                 *    If  VARIABLE% = 0  then only spaces will be
    050                 *    printed.
    051                 *    This can be changed by
    052                 *        POKE #370,#30 : print a "0"
    053                 *        POKE #370,#20 : print spaces
```

```
055                     *
056                     NUL     EQU   :370      *  #20=SPACE, #30=0
057                     AANT    EQU   :371      *  Number of figures to
058                     *                      *  printout
059                     XIBC    EQU   :C027     *  Convert integer for
060                     *                      *  OUTPUT
061                     PMSG    EQU   :DAD4     *  Print a message
062                     OUTCPR  EQU   :D695     *  OUTPUT character
063                     KBRFL   EQU   :2C4      *  BREAK flag
064                             ORG   :300
065 0300 C5             POUT    PUSH  B
066 0301 D5                     PUSH  D
067 0302 F5                     PUSH  PSW
068 0303 E7                     RST   :4        *  Convert integer variable
069 0304 0C                     DATA  :0C       *  for OUTPUT
070 0305 CD27C0                 CALL  XIBC      *
071 0308 217103                 LXI   H,AANT
072 030B 56                     MOV   D,M       *  Number of digits to print
073                     *                      *  in D
074 030C 14                     INR   D         *  +1
075 030D 3AF100                 LDA   :F1
076 0310 BA                     CMP   D
077 0311 DA2303                 JC    NEW
078 0314 218003                 LXI   H,MSG$
079 0317 CDD4DA                 CALL  PMSG      *  If D<F1 then print
080 031A 21C402                 LXI   H,KBRFL   *  message 'FORMAT ERROR'
081 031D 36FF                   MVI   M,:FF     *  and BREAK BASICprogram
082 031F F1                     POP   PSW
083 0320 D1                     POP   D
084 0321 C1                     POP   B
085 0322 C9                     RET
086 0323 3AF100         NEW     LDA   :F1       *  Lenght of the integer in
087                     *                      *  in #F1
088 0326 15                     DCR   D
089 0327 BA                     CMP   D
090 0328 CA3303                 JZ    PSTART
091 032B 3E20                   MVI   A,:20
092 032D CD95D6                 CALL  OUTCPR    *  OUTPUT character
093 0330 C32303                 JMP   NEW
094 0333 3AE400         PSTART  LDA   :E4       *  Sign of the integer
095 0336 FE2D                   CPI   :2D       *  if #E4="-" then CONT 2
096 0338 C23E03                 JNZ   CONT1     *  if #E4=space then CONT 1
097 033B C34003                 JMP   CONT2
098 033E 3E20           CONT1   MVI   A,:20
099 0340 CD95D6         CONT2   CALL  OUTCPR    *  OUTPUT character
100 0343 3AF100                 LDA   :F1
101 0346 21E600                 LXI   H,:E6     *  ASCII-string from #E6 to
102 0349 47                     MOV   B,A       *  #E6 + (#F1)
103 034A FE01                   CPI   :1
104 034C CA5C03                 JZ    ZERO
105 034F 7E             PRLOOP  MOV   A,M       *  PRINT-Routine
106 0350 CD95D6         RETURN  CALL  OUTCPR    *  OUTPUT character
107 0353 23                     INX   H         *
108 0354 05                     DCR   B         *
109 0355 C24F03                 JNZ   PRLOOP    *
110 0358 F1                     POP   PSW
111 0359 D1                     POP   D
112 035A C1                     POP   B
113 035B C9                     RET
114 035C 7E             ZERO    MOV   A,M       *  If #F1=1 and Integer=0
115 035D FE30                   CPI   :30       *  then print "0" or space
116 035F C24F03                 JNZ   PRLOOP    *  NUL=#30 or NUL=#20
```

```
PAGE 03

117 0362 3A7003              LDA   NUL
118 0365 C35003              JMP   RETURN
119                          ORG   :380
120 0380 0D          MSG$    DATA  :0D       *  MESSAGE
121 0381 464F52              ASC   'FORMAT ERROR'
122 038D 00                  DATA  0
123 038E                     END

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

AANT   0371   CONT1  033E   CONT2  0340   KBRFL  02C4
MSG$   0380   NEW    0323   NUL    0370   OUTCPR D695
PMSG   DAD4   POUT   0300   PRLOOP 034F   PSTART 0333
RETURN 0350   XIBC   C027   ZERO   035C
```

```
PC UTILITY V3.3
>D300 38F

0300 C5 D5 F5 E7 0C CD 27 C0 21 71 03 56 14 3A F1 00
0310 BA DA 23 03 21 80 03 CD D4 DA 21 C4 02 36 FF F1
0320 D1 C1 C9 3A F1 00 15 BA CA 33 03 3E 20 CD 95 D6
0330 C3 23 03 3A E4 00 FE 2D C2 3E 03 C3 40 03 3E 20
0340 CD 95 D6 3A F1 00 21 E6 00 47 FE 01 CA 5C 03 7E
0350 CD 95 D6 23 05 C2 4F 03 F1 D1 C1 C9 7E FE 30 C2
0360 4F 03 3A 70 03 C3 50 03 FF FF FF FF FF FF FF FF
0370 FF FF FF FF FF FF FF FF FF FF FF FF BF BF FF FF
0380 0D 46 4F 52 4D 41 54 20 45 52 52 4F 52 00 FF FF
```

```
10    PRINT CHR$(12):CLEAR 100:DIM A(10.0):B=1
15    PRINT CHR$(14);"PRINT OUT OF INTEGERVARIABLES"
16    PRINT CHR$(14);"-----------------------------"
20    PRINT :PRINT " FORMAT = 6 figures"
30    PRINT " If A%=0 then print spaces":PRINT :GOTO 100
40    PRINT " DAI          DAInamic":PRINT " PRINT A%;   CALLM#300,A%":RETURN
50    FOR P=0 TO 5
55    PRINT
60    PRINT A(P),:REM DAI PRINT
65    CALLM #300,A(P):REM PRINT WITH MLP
70    NEXT P
75    RETURN
80    POKE #131,1:PRINT TAB(30);"Push any key to continue.";:POKE #131,0:RETURN
90    GC=GETC:IF GC=0 THEN 90:PRINT :PRINT CHR$(12)
92    RETURN
100   FOR P=0 TO 5
102   A(P)=INT(RND(10.0)*(10.0^P))
104   IF A(P)=0.0 THEN 102
106   NEXT P
110   FOR Q=0 TO 1
120   FOR P=0 TO 5:A(P)=INT(A(P)*(100.0^Q)+0.5):NEXT P
130   GOSUB 40
140   GOSUB 50
160   PRINT :PRINT "+--------   +--------"
170   A(10.0)=0:FOR P=0 TO 5:A(10.0)=A(10.0)+A(P):NEXT P
180   PRINT " ????????",:CALLM #300,A(10.0)
185   GOSUB 80
186   POKE #371,8:REM CHANGE FORMATLENGHT : 8 figures
187   GOSUB 90
188   IF Q=0.0 THEN PRINT :LIST 186:PRINT
189   NEXT Q
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures
210   LIST 190
215   PRINT :GOSUB 40
220   FOR P=0 TO 5:A(P)=B*P:B=B*10:NEXT P
230   GOSUB 50:PRINT :PRINT
240   LIST 250
245   PRINT
250   POKE #370,#30:REM IF VARIABLE% IS ZERO THEN PRINT: 0
260   GOSUB 40:FOR P=0 TO 5:PRINT :PRINT A(P),:CALLM #300,A(P):NEXT P
270   POKE #370,#20:REM IF VARIABLE% ( A%(P%) ) IS ZERO THEN PRINT SPACES
275   GOSUB 80
280   GOSUB 90
290   LIST 190:PRINT
300   A=12345678
305   LIST 310:PRINT
310   PRINT A,:REM DAI PRINT
311   PRINT TAB(15);"( Lenght of A% = 8 figures )"
312   PRINT
315   LIST 320
320   CALLM #300,A:REM PRINT WITH MLP
```

PRINT OUT OF INTEGERVARIABLES

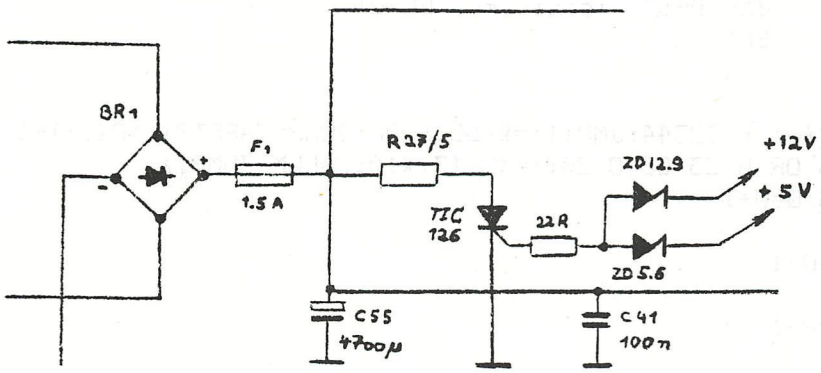
```
 FORMAT = 6 figures
 If A%=0 then print spaces

 DAI           DAInamic
 PRINT A%;     CALLM#300,A%

 5                5
 65              65
 451            451
 5311          5311
 69630        69630
 395732      395732
+--------    +--------
 ????????    471194

186   POKE #371,8:REM CHANGE FORMATLENGHT : 8 figures

 DAI           DAInamic
 PRINT A%;     CALLM#300,A%

 500              500
 6500            6500
 45100          45100
 531100        531100
 6963004      6963004
 39573224    39573224
+--------    +--------
 ????????    47119428
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures

 DAI           DAInamic
 PRINT A%;     CALLM#300,A%

 0
 10               10
 200             200
 3000           3000
 40000         40000
 500000       500000

250   POKE #370,#30:REM IF VARIABLE% IS ZERO THEN PRINT: 0

 DAI           DAInamic
 PRINT A%;     CALLM#300,A%

 0                0
 10              10
 200            200
 3000          3000
 40000        40000
 500000      500000
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures

310   PRINT A,:REM DAI PRINT

 12345678       ( Lenght of A% = 8 figures )

320   CALLM #300,A:REM PRINT WITH MLP

FORMAT ERROR
BREAK IN LINE 320
```

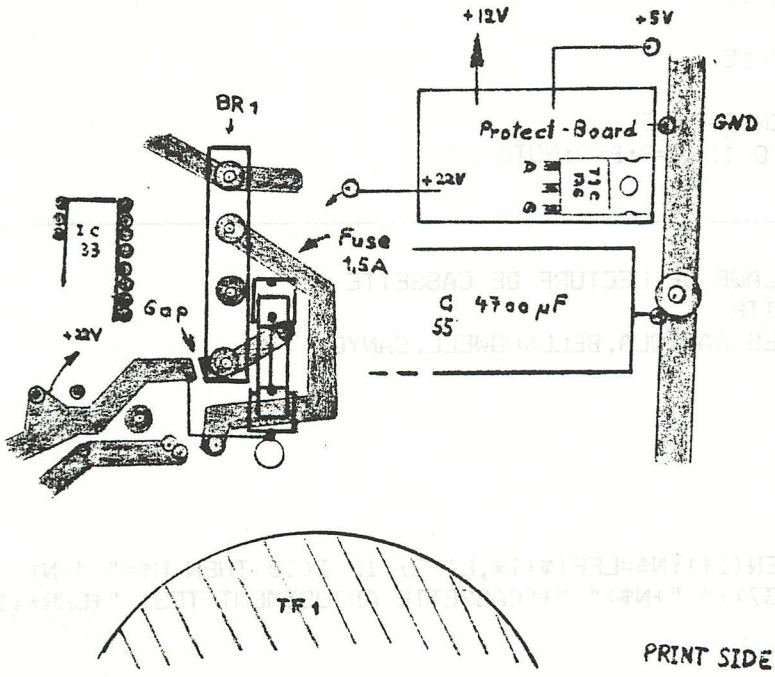
Netzteil u.Überspannungsschutz

Powersupply and Overvoltage-Protection.

Diese Schaltung und die Sicherung F1 verhindern im Falle eines Netzteildefekts große Schäden im Rechner,die durch die ungeregelte Gleichspannung von 22V verursacht werden könnte. Ebenso wird das bislang ungesicherte Netzteil vor Überlastung geschützt.
Überschreitet die Versorgunsspannung mehr als 0,9V ihres Normalwertes, zündet der Thyristor und die Sicherung wird ausgelöst.
Der Aufbau kann auf einer kleiner Veroplatine erfolgen.Diese kann vor dem Elko C55 im Netzteilkäfig eingebaut werden. Durch zwei kleine Bohrungen werden die Zuleiungen für Masse und die Eingangsspannung 22V mit der Grundplatine verbunden.Auf der Bestückungsseite umläuft eine breite Leiterbahn die Grundplatine.Von dieser Leiterbahn kann +5V zur Schutzschaltung geführt werden.Durch diese Anschlüsse wird die Schutzplatine gleichzeitig an ihrem Platz fixiert.
Die Sicherung wird neben dem Gleichrichter BR1 eingebaut.Dazu werden 4 kleine Löcher für die Halterung gebohrt.Die U+ Leiterbahn vom Gleichrichter BR1 wird unterbrochen und die Sicherung (1.5 Amp.) dazwischen geschaltet.

02.1983
r.corswandt





SIMULATION OF GOTO X / RUN X ONLY 7 BYTE

With the following program you can simulate GOTO X and it's also possible to do RUN (LINE NUMBER) in BASIC V1.0 without emptying the symbol table.
To get the program into the computer type in the following basicline:

```
*10 DIM JMP(1):JMP(0)=#232344:JMP(1)=#4DC363DF:JUMP=VARPTR(JMP(0))+1
```

Now JUMP contains the address and JMP(0),JMP(1) the program.
!!! Be careful if you have a CLEAR after executing the basicline JMP(0), JMP(1) and JUMP are zero !!!

You can use my program in the following ways:
1) as RUN X then type e.g. *X=100:CALLM JUMP,X
2) as GOTO X then type e.g. *10 X=100
*20 CALLM JUMP,X:REM GOTO X

!!!!! Note: X, JUMP, JMP(0) and JMP(1) have to be integers. !!!!!

Just van Dunne'

Here follows the machine language program and a basic example.

```
23          INX H
23          INX H
44          MOV B,H
4D          MOV C,L
C3 63 DF    JMP DF63  (basiccmd. GOTO)
            END
```

```
5     MODE 4
10    DIM JMP(1):JMP(0)=#232344:JMP(1)=#4DC363DF:JUMP=VARPTR(JMP(0))+1
20    H=GETC:IF H<16 OR H>23 GOTO 20:X=(H-13)*10:CALLM JUMP,X
30    IF Q<YMAX THEN Q=Q+1
35    GOTO 110
40    IF Q>0 THEN Q=Q-1
45    GOTO 110
50    IF P>0 THEN P=P-1
55    GOTO 110
60    IF P<XMAX THEN P=P+1
65    GOTO 110
70    IF Q<YMAX-15 THEN Q=Q+15
75    GOTO 110
80    IF Q>15 THEN Q=Q-15
85    GOTO 110
90    IF P>15 THEN P=P-15
95    GOTO 110
100   IF P<XMAX-15 THEN P=P+15
110   DOT A,B 0:DOT P,Q 15:A=P:B=Q:GOTO 20
```

GOTO X / RUN X

```
30    REM TEST DE REGLAGE DE LECTURE DE CASSETTE
40    REM ALAIN MARIATTE
50    REM MAGNETOPHONES RADIOLA,BELL&HOWELL,SANYO
60    REM
70    REM
80    REM
90    REM
100   DIM A(0.0)
120   FOR I=1 TO 20
130   I$=STR$(I):L!=LEN(I$):N$=LEFT$(I$,L!-2):IF I<10 THEN N$=" "+N$
140   A$="   "+CHR$(137)+" "+N$+" "+"CASSETTE ADJUSTMENT TEST "+CHR$(136)
150   SAVEA A A$
160   NEXT
```

PROGRAMMING TECHNIQUES

(from DAInamic 11, page 173)

I am planning to provide a chapter in this journal about the problems and techniques of programming. The methods to be given will always be valid for the DAI and sometimes also for many other computers. The subjects offered arise from your suggestions and/or from weak spots in submitted programs. In this connexion I am thinking of such things as drawing circles quickly, avoiding conflicting colours, working with arrays, the speed and readability of programs, etc. For this first occasion my subject will be the construction of a list of finite but randomly dispersed numbers. The idea came from T. Groeneveld's BINGO program (DAInamic 10, page 136). I hope that Mr. Groeneveld, after reading this, will not imagine that people think he is a poor programmer. I certainly do not, particularly as he himself has found a solution. In his program an array is needed with the numbers from 1 to 75 inclusive, but the numbers must be stored in a random order. If you cannot make the program work properly try inputting it after an IMP INT command; that was not done originally as can be seen from the .0s From the original program :

```
10 CLEAR 1000: DIM G(76.0)
25 R=RND(75.0)+1.0
26 ENVELOPE 1 15,9;0: SOUND 1 1 15 0 FREQ(RND(269.0)+31.0)
27 NOISE 1 15
30 FOR A=1.0 TO 75.0: IF G(A)<>R THEN NEXT A
40 IF A<76.0 AND G(A)=R THEN 25
50 IF G(I)=0.0 THEN G(I)=R
65 I=I+1.0: IF I=76.0 THEN 100
70 GOTO 25
```

Let us analyse these lines:

In 25 we choose a suitable number
26 and 27 the waiting time is made pleasant
30 checks that the number chosen has not previously been selected
40 if so, we choose another
50 if the number is not already in the table we put it there
65 allocate the next number to the next location and check to see if we have them all yet
70 if not go back and choose the following one

As you can see the program works all right, but very slowly. On my machine (with the maths chip) I recorded an average of 72 seconds. What can be done about it ? Firstly change the program a little: I started by removing the .0s, took the NOISE and ENVELOPE out of the loop and coloured the sound with R*4 which is much faster than another RND. I improved the time a little, the average now being 66 seconds.

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

Back again to the rebuilding! Multiple statements can be combined on the same line. The test of A<76 seems to me superfluous. The end test is the thing that is giving BASIC a bad name (spaghetti code) with its inverted jump combined with a GOTO address. The average time is now 58 seconds.

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

Now only running the test, with the sound removed, brings the average to 48 seconds, but I find it better to wait 58 seconds with sound than 48 without. Later I saw that the loop only had to go to I. We still have too long to wait and so must try another tack.

The original program is sluggish because a random number has to be chosen and then checked to see if it has appeared before. It is better firstly to get the numbers we want and then to put them in a random place in the array. I have not worked this method out because it would only be turning our problem around. In the first method there is the difficulty of obtaining a suitable number while in the suggested alternative there is the difficulty of obtaining a suitable location. The new method certainly would be quicker because of the simpler checking (only looking to see if the place is empty instead of checking all the numbers). If I were to make the array much larger, say 256, then it should be easier to find an empty place in it. Afterwards, when I have all the numbers 1 to 75 in this auxilliary array I only have to transfer them neatly one after the other into array G(). The time gained is vast, resulting in a period of 3.4 seconds.

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

But we have not finished yet. The method is best for an array with 75 elements, but less suitable as the quantity increases because it then becomes more difficult to find an empty place. Sometimes too the penalty of all that extra memory space is significant. If we do not mind the list not being truly random, then the following method may do: Place the numbers 1 to 75 randomly into the array of 75 locations. If the location is already occupied, do not choose a new random place but take the next empty one. I have not worked out these possibilities because of the non-random features. Now the fastest method which still has a short program is:

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

We put the numbers 1 to 75 inclusive successively into the array and then exchange each number with another random element. We get a nice random list in a nice quick time: 2.4 seconds if we use G(0) as a parking place and even better, 2.36 seconds, when we use A. This latter is because the DAI has more difficulty in finding array variables. If anyone knows of a shorter or quicker method I would like to hear of it and will then certainly come back to the subject.

Frank H Druijff

```
 0 zwart
 1 blauw
 2 d.rood
 3 rood
 4 paars
 5 groen
 6 d.bruin
 7 l.bruin
 8 grijs
 9 blauw
10 oranje
11 rose
12 l.blauw
13 l.groen
14 geel
15 wit
```

alle adressen in HEXvorm!

```
29B-29C   start heap
29D-29E   size  heap
29F-2A0   start text buffer
2A1-2A2   start symbol table
2A3-2A4   end of symbol table
2A5-2A6   bottom screen ram

75        cursor symbol
74        cursor mode
72-73     cursor position

40,28 cass motor 1 ON
40,18 cass motor 2 ON
40,30 1 and 2 OFF
```

```
131,0   output scrn+
        RS232
131,1   screen only
131,2   edit buffer
135,2   read from
        edit buffer
```

MODE	XMAX	YMAX
1/2	71	64
3/4	159	129
5/6	335	255

```
MERGE
°CLEAR XXX
°LOAD"A"
°EDIT BREAK/BREAK
°LOAD"B"
°POKE 135,2
```

```
COLORG R1 R2 R3 R4
       20 21 22 23
16 :R2*R1 R4*R3
17 :R1*R2 R3*R4
18 :R3*R1 R4*R2
19 :R1*R3 R2*R4
```

```
32K 7XXX
12K 2XXX
 8K 1XXX
```

```
IMP INT *** IMP FPT
°IMP FPT
°CLEAR XXXX
°EDIT BREAK/BREAK
°IMP INT
°POKE 135,2
```

LIJN	CTRL	COLOR	LIJN	CTRL	COLOR
23	BFEF	BFEE	11	B9A7	B9A6
22	BF69	BF68	10	B921	B920
21	BEE3	BEE2	9	B89B	B89A
20	BE5D	BE5C	8	B815	B814
19	BDD7	BDD6	7	B78F	B78E
18	BD51	BD50	6	B709	B708
17	BCCB	BCCA	5	B683	B682
16	BC45	BC44	4	B5FD	B5FC
15	BBBF	BBBE	3	B577	B576
14	BB39	BB38	2	B4F1	B4F0
13	BAB3	BAB2	1	B46B	B46A
12	BA2D	BA2C	0	B3E5	B3E4

CTRL&COLOR BYTES IN A-MODE

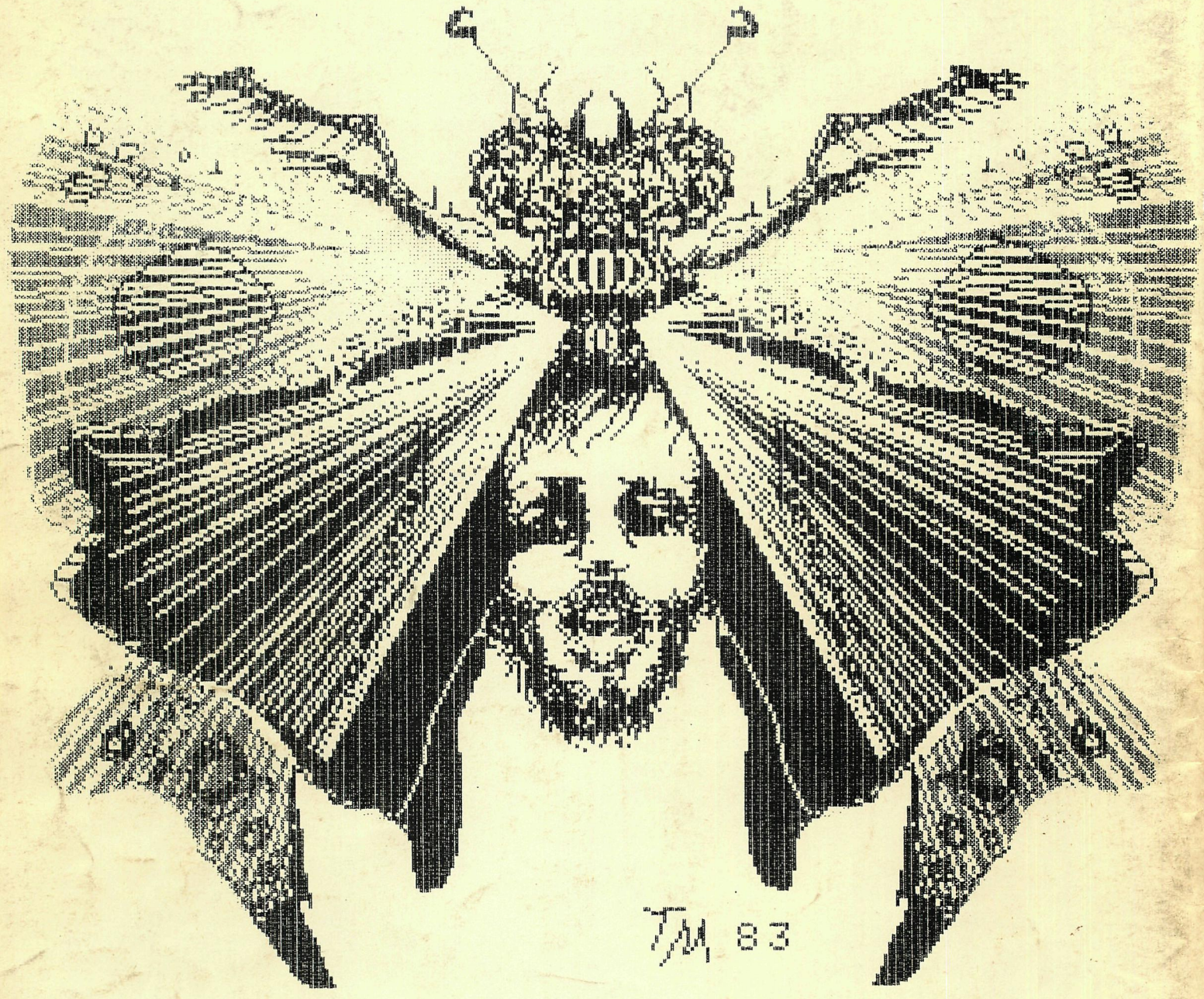
MODE	CTRL	COLOR	LIJN
1A/2A	BAE7	BAE6	3
	BA61	BA60	2
	B9DB	B9DA	1
	B955	B954	0
3A/4A	ACD3	ACD2	3
	AC4D	AC4C	2
	ABC7	ABC6	1
	AB41	AB40	0
5A/6A	7557	7556	3
	74D1	74D0	2
	744B	744A	1
	73C5	73C4	0

```
FD00 b2 page signal
     b3 serial out rdy
     b4 right paddle
     b5 left  paddle
     b6 random data
     b7 cass. input
FD01    Trigger paddle
FD04 0-3 volume ch.1(0)
     4-7 volume ch.2(1)
FD05 0-3 volume ch.3(2)
     4-7 volume noise
FD06 b0 cass.out
     b1/2 paddle select
     b3 paddle enable
     b4 cass motor 1
     b5 cass motor 2
     b6/7 ROM BANK SWITCH
```

```
FF00 ser.inp.buff
FF01 b0-6 keyb.inp.
     b7 in7 DCE
FF02 Interr.reg.
FF03 b1 frame error
     b2 overrun error
     b3 rec.buf.loaded
     b4 trans.buf.empty
FF04 COMMAND REGISTER
FF05 BAUD RATE REGISTER
FF06 ser.out buf.
FF07 keyb.output
FF08 interr.mask reg.
```

```
TEST EVENT
PEEK(éFD00) IAND 32
PEEK(éFD00) IAND 16
PEEK(éFD00) IAND 48
```

```
FF09 TIMER 0
FF0A TIMER 1
FF0B TIMER 2
FF0C TIMER 3
FF0D TIMER 4
8253
CH 0 FC00/FC01
CH 1 FC02/FC03
CH 2 FC04/FC05
STATUS FC06/FC07
```


TM 83