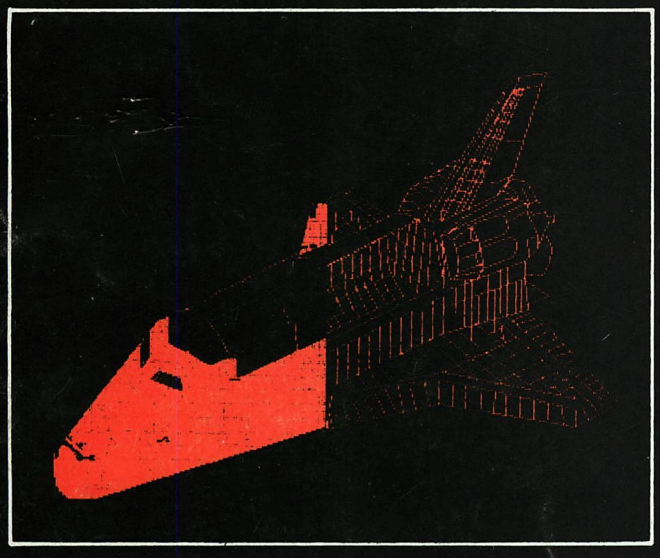
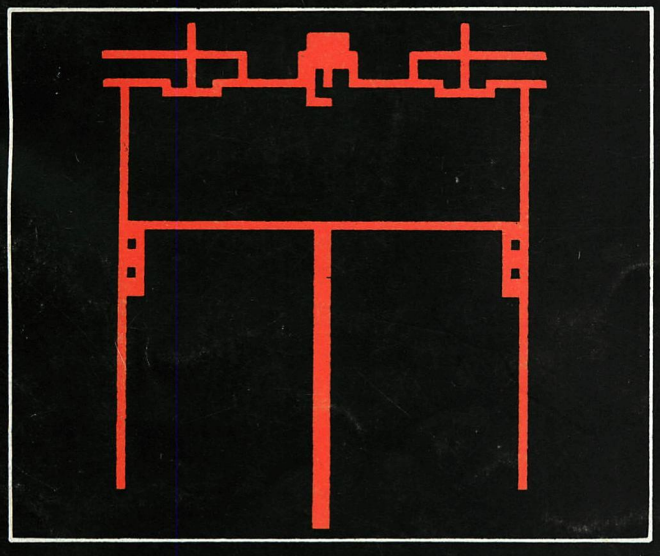
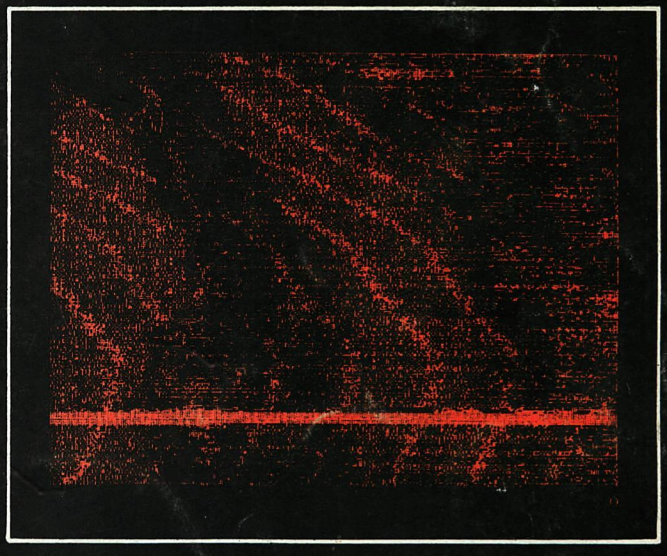
DAI NAMIC

10
dubbel nummer

tweemaandelijks tijdschrift maart - juni 1982







personal computer users club

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

COLOFON

DAInamic verschijnt tweemaandelijks.
abonnementsprijs is inbegrepen in de jaarlijkse contributie : 750 Bfr.
Bij toetreding worden de verschenen nummers van de jaargang toegezonden.
DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans
Heide 4
B 3171 Westmeerbeek
België

tel. : 016/69.86.23

Kredietbank Westmeerbeek
nr. 406-3016141-33

BTW : 420.840.834

Lidgelden

Bruno Van Rompaey
Bovenbosstraat 4
B 3044 Haasrode
België

tel. : 016/46.10.85

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

Frank Druijff
's Gravendijkwal 5A
NL 3021 EA Rotterdam
Nederland

tel. : 010/25.42.75

DAINAMIC
PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

DAINAMIC

Westmeerbeek mei 82

Beste leden,

Door de vele extra clubactiviteiten van de laatste maanden waren we achter op ons publicatieschema.Met dit dubbelnummer doen we een inhaalmaneuver , zodat we voortaan ons blad weer op tijd kunnen versturen.We hopen dat kwaliteit en volume van nummer 10 het wachten ruimschoots compenseren.

Hierbij geven we een overzicht van de nieuwe publicaties en producties voor de komende maanden :

* hardware schema's DAIpc :nu verkrijgbaar
* tape 80-81 (57 programma's van de vorige jaargangen) :nu verkrijgbaar
* de 23 programma's uit dit nummer op tape :nu verkrijgbaar
* "the best of DAInamic " in boekvorm : augustus 82
* "DAI firmware " door J.Boerrigter : december 82

"softwarepiraat slaat toe via de ether...":dit zou een leuke krantentitel kunnen zijn.Op originele manier hadden we de laatste weken last van piraterij : onze copieerapparaten staan op een 500m van de zendmast van de plaatselijke vrije radio "TELSTAR". Deze jongens zenden er erg enthousiast op los en komen met hun programma's ongevraagd op de aanloopstrook van onze software-bandjes... Gelukkig kan het computersignaal de populaire plaatjes en praatjes volledig onderdrukken.

Onze dank gaat andermaal naar alle medewerkers en auteurs die deze uitgave mogelijk maakten.

We willen iets gaan doen aan het taalprobleem binnen onze vereniging:mensen die artikels en programma's zouden kunnen vertalen naar het engels,duits of frans worden verzocht contact te nemen met onze vertaal-coordinator:

Filip Peeters
Vlinderstraat 7
2440 GEEL Belgie tel 014/583018

Uw werk compenseren we graag met gratis software,papier en telefoonkosten worden vergoed.

Onze bijeenkomst op 10 april was andermaal een succes,vooral de lezingen vielen erg in de smaak.Ergens in oktober gaan we een volgende bijeenkomst plannen, we zullen de voordrachten nog beter voorbereiden en aankondigen zodat U deze dag optimaal zal kunnen benutten.

Bij de samenstelling van dit nummer hebben we getracht de beginners niet te vergeten:we vermoeden dat de demo-programma's van Frank het handboek mooi aanvullen.Volgende keer gaan we de EDIT-buffer bestuderen en leggen we de loupe op de BASIC-token structuur.

we hopen dat nummer 10 uw creativiteit stimuleert,uw bijdragen zjjn steeds welkom..

---

dear members,

Because we were running out of time,we offer you this "double issue",we hope that the many programms and articles can compensate for the waiting.

This is the time-schedule of our new publications and productions:

* hardware schematics DAIpc :now available
* tape 80-81 (57 programms) :now available
* the 23 programms of this issue:now available
* "the best of DAInamic"(book) :august 82
* "DAI firmware "by J.Boerrigter:december 82

We are looking for co-editors/translators : if you can assist in translating articles to english,french or german,please contact our language-coordinator:

Filip Peeters
Vlinderstraat 7
2440 GEEL Belgium tel 014/583018

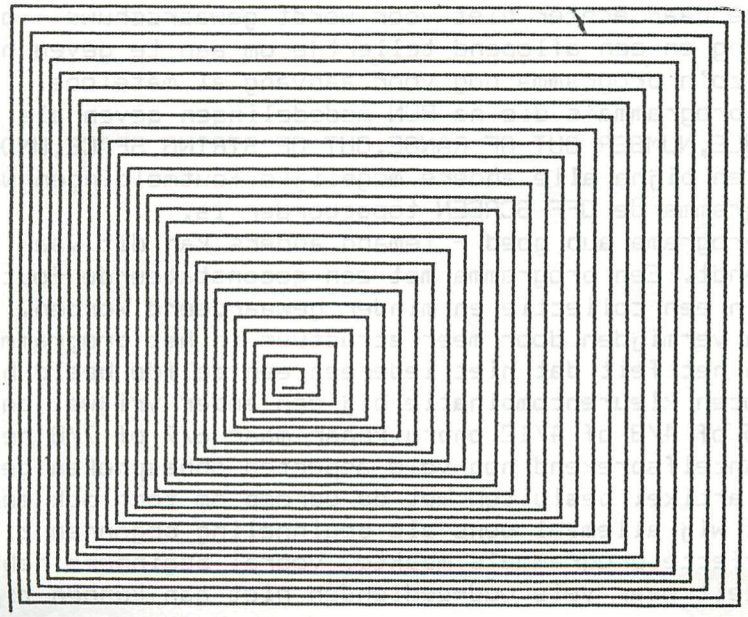
We will compensate your efforts with free software,we will pay your mailing and telephonecosts for this job.

Our next meeting will be in october ,more information in future issues.

we hope you will enjoy this king size issue,we appreciate all contributions...

Wilfried Hermans

BLADWIJZER CONTENTS SOMMAIRE IN DIESEM HEFT



*RUN

***** INZENDEN VAN PROGRAMMA'S *****

Veel dainamicers weten niet, is mij gebleken, dat programma's opgestuurd kunnen worden. De programma's worden dan beoordeelt en U krijgt behalve dit commentaar ook nog eens een aantal programma's uit de bibliotheek. Dit laatste geldt alleen als de programma's een ruil rechtvaardigen en de opgezonden programma's -indien zij daarvoor in aanmerking komen- opgenomen mogen worden in de bibliotheek van Dainamic.
De eisen die wij aan programma's stellen om voor ruil in aanmerking te komen zijn echter niet hoog; ook de beginner moet de kans hebben zijn programma's in te kunnen sturen. U kunt trouwens zelf uw voorkeur voor bepaalde collecties opgeven zodat wij U zo goed mogelijk kunnen helpen.
Mensen die vaak insturen zullen begrip hebben dat zij soms niets krijgen en een volgend keer veel meer dan gerechtvaardigd door de inzending van dat moment. Een tweede beloning (blijk van waardering) krijgt U als uw programma opgenomen wordt in een van de collecties - U ontvangt deze collectie dan geheel gratis en automatisch (klaag dus als wij het vergeten).

Nu over het inzenden zelf.
Liefst ontvangen wij uw programma's op DCR-cassette omdat wij dan nooit laadproblemen hebben. Iets minder lief maar nog steeds gewaardeerd op disk of op gewone cassette. De hogere kosten van minicassette/diskette behoeven geen bezwaar te zijn bij insturen daar U terugkrijgt wat U instuurt.
Vooral bij gewone cassettes zijn er nog regelmatig laadproblemen, de snelheid waarmee U antwoord krijgt kan hier vaak mee samenhangen.
U kunt deze problemen enigzins verlichten door uw band te beginnen met een twintigtal tests waar wij dan op kunnen afregelen. U doet dit als volgt:

```
NEWreturn
DIM A(0)return
FOR I=1 TO 20:SAVEA A "TEST":NEXTreturn
```

en vanzelfsprekend de recorder op opname. Vervolgens kunt U ons erg helpen met een begeleidend schrijven waarin vermeld staat wat wij op de band kunnen verwachten het spaart ons het afspelen van een band waar niets meer op staat. Doe dit begeleidend schrijven inderdaad letterlijk op papier, als wij laadproblemen hebben kunnen wij aanwijzingen op band natuurlijk ook niet lezen. En dan het meest belangrijke : de programma's zelf.

Ik zou bijna elke inzender de raad willen geven om eerst iemand die niets van computers weet eens met het programma te laten werken/spelen.
Ik heb tientallen programma's die na een RUN mij lange tijd lieten wachten en dan bij controle in een eeuwige loop bleken te zitten. Soms in afwachting van een bepaalde code, die mij echter op dat moment nog onbekend was.Soms bleken overigens correcte programma's lange tijd bezig met het inlezen van data. Als daar echter niet voor wordt gewaarschuwd - bij voorkeur met een aankondiging en een aflopend tellertje om aan te geven hoever het staat - irriteert zo'n programma nog voor aanvang al mateloos.
Ook bezit ik vele programma's die na RUN mededelingen geven als OUT OF MEMORY, COLOUR NOT AVAILABLE,NUMBER OUT OF RANGE,OUT OF STRING SPACE,UNDEFINED ARRAY,OUT OF DATA en bijna alle andere mogelijke foutmeldingen waar iets verder in het programma de OFF SCREEN topscoorder is.
Controleert U uw programa aub goed - iemand anders kan dit vaak beter - voordat U het inzendt. Een programma met een geconstateerde fout zal niet opgenomen worden in een collectie en minder gewaardeerd worden.
U kunt vele fouten vermijden door meer structuur in het programma te brengen, te denken aan het feit dat niet iedereen een zelfde beeld heeft dus dat de door U gekozen kleurencombinatie bij een ander vrijwel onleesbaar is. Bij mij is 4/5 of 4/8 of 4/12 onbruikbaar maar 6/7 of 7/8 heel duidelijk. Er zijn vanzelfsprekend nog vele aspecten onbesproken gebleven maar ik wilde dit artikel besluiten met de oproep om toch ook eens in te zenden. Bedenk dat wij allemaal eens als beginneling begonnen zijn en dat niemand ineens perfecte programma's maakte. Vooral beginners hebben vaak nog een originaliteit waar vele oud-gedienden niet aan kunnen tippen.

Frank H. Druijff

A A N D A C H T + + + + O N D E R W I J S + + + S O F T W A R E + + +

diDAIsoft

De DAI pc doet zijn intrede in het onderwijs. DAInamic start daarom met een didactische softwaregroep diDAIsoft.

Doelstelling :-didactische software ontwikkelen voor alle vakken uit het secundair onderwijs
-kant en klare handleidingen bij deze programma's samenstellen
-een vaste onderwijsrubriek in de nieuwsbrief verzorgen

Werkwijze : op tweemaandelijkse bijeenkomsten worden concrete afspraken gemaakt in verband met de programma's die gedurende de volgende twee maanden zullen ontwikkeld worden; individuele engagementen voor één van de voorgestelde projecten; op een volgende bijeenkomst worden dan de ontwikkelingen bekeken, besproken en (eventueel) aangepast; op deze wijze komen per periode een aantal zinvolle programma's tot stand, waarvoor de ganse softwaregroep garant staat; publicatie van de software onder het label diDAIsoft.

Wat bieden wij: -toegang tot de reeds beschikbare bibliotheekroutines;
-reisonkostenvergoeding en lunch op de dagen van de bijeenkomsten;

Plaats van de bijeenkomsten: Bruno Van Rompaey
Bovenbosstraat 4
3044 HAASRODE
016 - 461085

DATUM VAN DE EERSTE BIJEENKOMST:

dinsdag 6 juli 1982 vanaf 10u tot 16u

Om schikkingen te treffen voor de lunch graag vooraf telefonisch of schriftelijk verwittigen voor 1 juli: contactadres Bruno Van Rompaey.
Geinteresseerde medewerkers die op deze "stichtingsvergadering" niet aanwezig kunnen zijn, toch maar contact op nemen. We melden hen dan persoonlijk de datum van de volgende werksessie.
Met uw hulp wordt diDAIsoft een garantie voor kwaliteitssoftware.

Bruno

DCE-microline

```
67        *** ========================================
68        *** *              CONNECTIONS
69        *** *  D.C.E.BUS -------- OKI microline 80
70        *** ========================================
71        *** Port Bit Pin ----------------Func.-Pin
72        *** *    GND   4 ------------        0V   26
73        *** P0 bit 0  16 ------------ data b1    2
74        *** P0     1  14 ------------      b2    3
75        *** P0     2  12 ------------      b3    4
76        *** P0     3  10 ------------      b4    5
77        *** P0     4   9 ------------      b5    6
78        *** P0     5  11 ------------      b6    7
79        *** P0     6  13 ------------      b7    8
80        *** P0     7  15 ------------      b8    9
82        *** P2     7  17 ---- <          Busy   11
83        *** P2     1  27 ---- >        Strobe    1
84        *** P2     6  18 ---- <   Acknowledge   10 (not used)
85        *** P2     5  19 ---- < Paper out/low   12 (not used)
86        *** P2     4  20 ---- <    reserve       (not used)
90        ***
100        *** ::***  M/L ROUTINE FOR PRINTER ON DCE BUS  ***::
110        *** ------------------------------------------------
111        *** Go in UTILITY,type in next program,save on tape.
120        *** ------------------------------------------------
130        *** 029B  20 03       :HEAP
140        *** 029D  00 01       :HSIZE
150        *** 029F  20 04       :TXTBGN
160        *** 02A1  21 04       :TXTUSE
170        *** 02A3  22 04       :STBUSE
180        *** 02DD  C3 F0 02    :DOUTC
190        *** ********************************
200        *** 02F0  E5          PUSH H
210        *** 02F1  D5          PUSH D
220        *** 02F2  C5          PUSH B
230        *** 02F3  F5          PUSH PSW
240        *** 02F4  57          MOV  D,A
250        *** 02F5  21 03 FE    LXI  H,dble
260        *** 02F8  36 8A       MVI  M,byte
270        *** 02FA  2B          DCX  H
280        *** 02FB  06 20       MVI  B,byte
290        *** 02FD  7E          MOV  A,M
300        *** 02FE  E6 80       ANI    byte
310        *** 0300  C2 FD 02    JNZ    addr
320        *** 0303  7A          MOV  A,D
330        *** 0304  32 00 FE    STA    addr
340        *** 0307  36 FD       MVI  M,byte
350        *** 0309  05          DCR  B
360        *** 030A  C2 09.03    JNZ    addr
370        *** 030D  36 FF       MVI  M,byte
380        *** 030F  F1          POP  PSW
390        *** 0310  C1          POP  B
400        *** 0311  D1          POP  D
410        *** 0312  E1          POP  H
420        *** 0313  C9          RET
430        ***
440        ***
450        REM : USE  POKE#131,3 TO ACTIVATE PRINTER
460        REM : USE  POKE#131,1 TO DEACTIVATE PRINTER
470        REM : Aut.J.A.P.
```

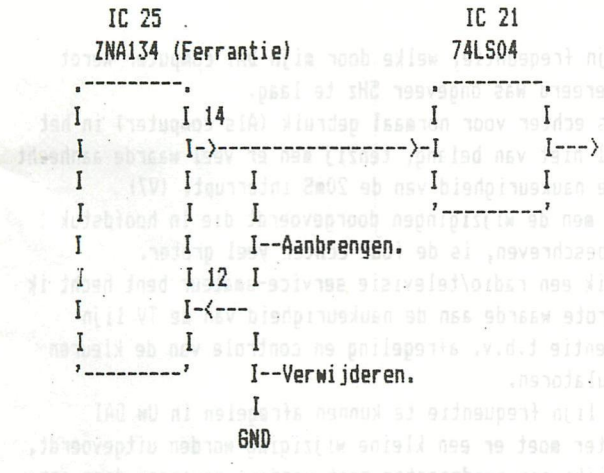
DAI video-hardware

INHOUD OPGAVE (1)

1 Wijziging DAI computer van interliniering naar niet interliniering. (2)

Een TV beeld wordt opgebouwd met 625 lijnen.
Deze lijnen worden niet in 1 keer op Uw scherm geschreven, maar in 2 keer.
Met andere woorden: eerst wordt op Uw scherm de oneven lijnen geschreven, vervolgens de even lijn nummers.
Van de 625 lijnen zijn er ongeveer 550 zichtbaar.
Uw DAI computer doet het op dezelfde wijze,echter de informatie van de oneven- en even lijnnummers zijn hierbij gelijk.
Stel: de nu volgende schrijfslag zijn de oneven lijn nummers.De de hierna volgende schrijfslag bevat hetzelfde beeld informatie met dien verstande dat het iets lager op Uw TV scherm geschreven wordt.(Afstand tussen beide schrijfslagen is ongeveer van 0.5mm tot 1.0 mm.)
Een schrijfslag duurt 20ms.(50Hz)
Samenvattend: de door Uw DAI computer gegenereerde beeld staat iets te dansen in vertikale zin,met een periode tijd van 2 * 20ms = 40ms
Dit is de oorzaak dat Uw DAI computer een enigsinds onrustig beeld geeft.

Wijzigings voorstel interliniering - niet interliniering (3)



Aangezien poot 12 van IC 25 onder deze chip aan de komponent zijde van de print met de 'GND' is verbonden, is het moeilijk om deze verbinding te verbreken.
Om deze poot toch vrij te maken kan men het beste vlak boven de print (komponent zijde) deze doorknippen, om vervolgens enigsinds omhoog te buigen en met een draadje te kunnen doorverbinden met poot 14.

Voordeel van de hiervoor aangegeven wijziging zal een ieder wel duidelijk zijn. (4)
Deze wijziging heeft ook een nadeel:
Indien men gebruik maakt van de 20ms interrupt (vector 7) b.v. een real time clock, zal deze klok iets te snel gaan lopen.
f(kristal (zna 134)) = 2,562,500 Hz
f(lijn) = f(kristal) / 164 = 15625 Hz
f(raster) = f(lijn) / 312.0 (was 312.5) = (50 + 25/312) Hz
De '20ms' interrupt komt nu iedere 19.968 ms.
Verder omgerekent komt het op neer dat de real time clock per 24 uur 138 sec te snel zal lopen,voorop gesteld dat de kristal frequentie gelijk is aan de opgegeven waarde.

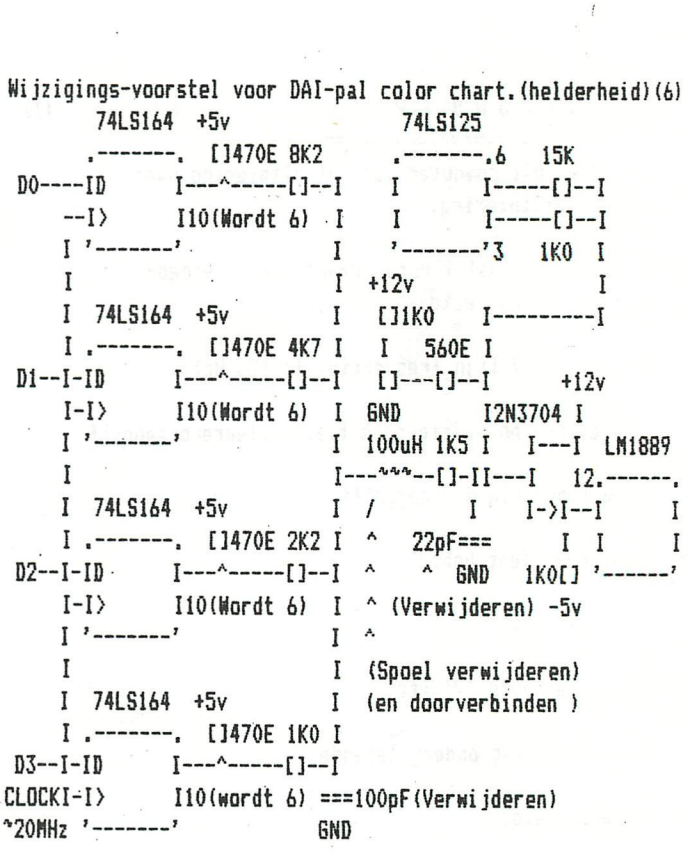
zie hiervoor verder hoofdstuk 3 van deze brief.

DAI video-hardware

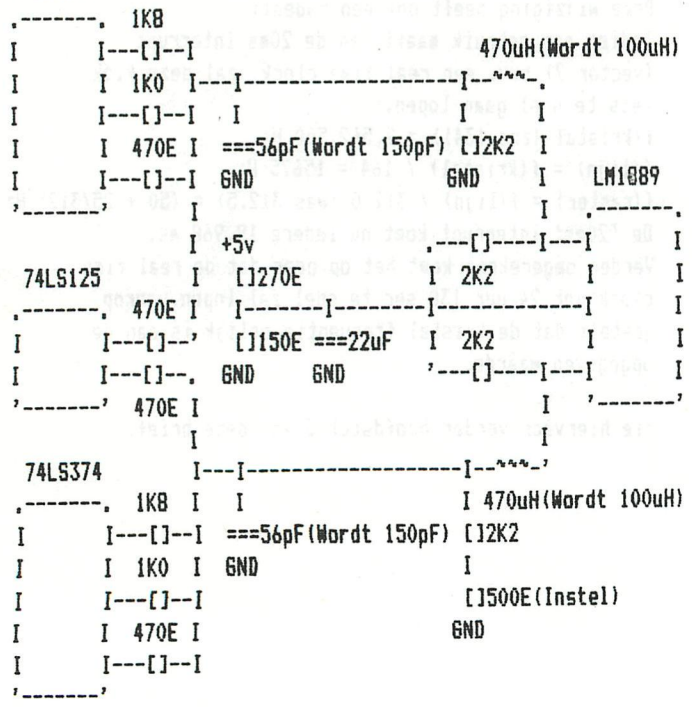
2 Wijziging DAI pal kleuren kaart t.a.v video- en kleur-bandbreedte (5)

Het viel mij op dat het testbeeld, uitgezonden door de nederlandse televisie zenders een hogere resolutie had dan de door mijn DAI computer gegenereerde plaatjes. Oorzaak hiervan is dat de video-bandbreedte van de DAI-pal color-chart maar slechts 3MHZ is.(-6dB) Idem voor de kleur-bandbreedte namelijk 0.75MHZ. Waarom heeft de firma 'DAI' deze bandbreedtes zo klein gekozen? Misschien om interferenties te voorkomen? De bandbreedtes van een televisie toestel zijn resp. 5 MHZ en 1.5 MHZ. Na de op de volgende bladzijden beschreven wijzigingen krijgt de TV interface band-breedtes van resp 7MHZ en 2.5 MHZ. Dit geeft een enorme verbetering t.a.v.het oplossend vermogen. Na de wijzigingen heb ik in het geheel geen last van interferenties en dergelijke.Bij enkele andere merken TV's echter wel.De TV die ik gebruikt als monitor is een philips met 'KT3' chassis,43 cm 90 graden beeld-buis.De toepassing van een 90 graden beeld-buis maakt deze TV zeer geschikt als monitor voor uw DAI computer. omdat het oplossend vermogen van de beeld-buis groter is (focus) dan vanuitgaande van de 5MHZ bandbreedte.

Wijzigings-voorstel voor DAI-pal color chart.(helderheid)(6)



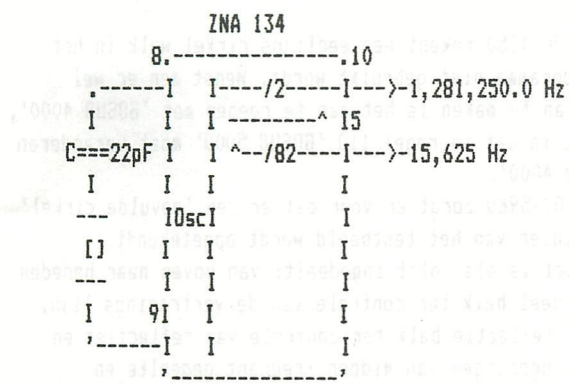
Wijzigings-voorstel DAI-pal color chart.(Kleur) (7)



3 Afregeling TV lijn freqeuntie (15625 Hz) (8)

De lijn freqeuntie, welke door mijn DAI computer werdt gegenereerd was ongeveer 5Hz te laag.
Dit is echter voor normaal gebruik (Als computer) in het geheel niet van belang, tenzij men er veel waarde aanhecht aan de naukeurigheid van de 20mS interrupt. (V7)
Heeft men de wijzigingen doorgevoerdt die in hoofdstuk 1 zijn beschreven, is de fout echter veel groter.
Daar ik een radio/televisie service-amateur bent hecht ik een grote waarde aan de naukeurigheid van de TV lijn frequentie t.b.v. afregeling en controle van de kleuren demodulatoren.
Om de lijn frequentie te kunnen *afregelen in Uw DAI* computer moet er een kleine wijziging worden uitgevoerdt, namelijk: een condensator moet worden vervangen door een instelbare type.(Zie volgende bladzijde.)
Nogmaals: een ieder die de DAI computer niet als testbeeld generator gebruikt, heeft het in het geheel geen zin om de in dit hoofdstuk beschreven wijziging en afregeling uit te voeren.

Wijziging en afregeling TV lijn frequentie: (9)



Condensator 'C' wordt een instelbare type van 50pF.
Hiermede regelt men de frequentie op pin 10 van IC25
(ZNA 134) af op 1,281,250.0 Hz met een frequentie teller.

(11)

Stemt Uw TV toestel af op een TV zender en kontroleer
of het testbeeld nog in kleur wordt weergegeven.
Laat nu de computer en de TV een half uur opwarmen.
Vervolgens regelt U met de erbij geplaatste instel
condesator zodanig af dat het lisajoux figuur op de scoop
stilstaat.

DAI video-hardware

4 Afregeling- en wijzigingen t.a.v. kleuren draaggolf. (10)

Het gebruikt van de DAI computer als testbeeld generator
heeft mij snel geleerdt dat de frequentie van de kleuren
draaggolf te hoog is, zodanig dat enkele TV toestellen
het niet kan 'pakken' zodat er niet op kleur weergave
wordt overgeschakeld.Dit is echter al eerder op gemerkt,
zie hiervoor een DAInamic nieuwsbrief.
Om de genoemde frequentie lager en afgeregeld te krijgen
moet er instelbare condensator van 22pF parralel aan de
kristal geplaatst te worden.(Kristal, welke op het
modulator plaat bevindt)
Deze frequentie moet heel nauwkeurig afgeregeldt worden
en omdat de door mijn gebruikte frequentie teller niet
nauwkeurig genoeg is het ik een ander methode gebruikt
namelijk om de frequentie te vergelijken met de van een
TV zender door middel van lisajoux figuren op een scoop.
De ene kanaal wordt op Uw DAI computer aangesloten
op de ingang van de HF modulator (Kleine blikken doosje,
welke op de modulator plaat bevindt, pin 4.) terwijl de
ander op de uitgang van de kleuren referentie generator
van Uw TV toestel wordt aangesloten.
Typ op Uw DAI computer:'100 COLORG 1 0 0 0:MODE5:GOTO 100'
vervolgens: 'RUN'.

5 Afregeling geluid draaggolf (12)

De frequentie van de geluid draaggolf is eveneens van
belang als de DAI computer wordt gebruikt als test beeld
generator.
Om deze af te regelen heeft men een scherp geslepen
lucifer stokje nodig.
Dit laat zich makkelijk afregelen. Men sluit een
frequentie teller aan op pin 1 van de HF modulator.
In de deksel van deze modulator bevindt een tweetal gaatjes,
men neme de dichtst bijzijnde bij de aansluit zijde
en regel de frequentie af op 5,500 KHz.
Hierbij dient 'SOUND OFF' te zijn.
Bij mijn DAI computer was naregeling niet nodig.

DAI video-hardware

6 Programma 'Testbeeld' (13)

Zoals U al weet ben ik een service amateur, dus had ik een behoefte aan een eigen testbeeld generator.
Dit is ook de hoofdreden geweest om mijn DAI computer op diverse punten te wijzigen.(Zie alle voorgaande hoofdstukken.)
Als voorbeeld voor het te generen plaatje heeft het test beeld plaatje van de nederlandse zenders gestaan
Probleem was echter dat de DAI basic V1.1 in de grafische bewerkingen niet de onzichtbare gedeelten support, hetgeen voor een testbeeld van uiters belang is.
Dat is de reden dat in het programma 'testbeeld'(verder op deze cassette.) erg veel naar het scherm gePOKEt wordt.
De cirkel is wat kleiner uitgevallen omdat ik dit keer wel gebruik wilde maken van de basic support.
Dit programma is niet voorzien van allerlei instel mogenlijkheden, gebleken is in de praktijk dat ik toch iedere keer het zelfde plaatje wilde hebben
Een korte beschrijving van het programma:
Regels 100-999 is het hoofd programma welke kort is.
Regels 1000-1060 is de initializatie van het beeld scherm geheugen, hierbij wordt het scherm enigsinds omhoog geschoven en van onder wat toegevoegd. (14)
Regels 2000-2804 zorgen er voor dat er de witten blokken rondom het plaatje geplaatst worden.(De meesten zijn niet of gedeeltelijk zichtbaar.)
Regels 3000-3070 deelt het beeld op in kleine vierkanten blokjes.
Regels 4000-4050 tekent een eenlijns cirkel welk in het hoofd programma niet gebruikt wordt. Wenst men er wel gebruik van te maken is het aan te roepen met 'GOSUB 4000', het houdt in dat in regel 110 'GOSUB 5000' moet veranderen in 'GOSUB 4000'.
Regels 5000-5960 zorgt er voor dat er een 'gevulde cirkel' in het midden van het testbeeld wordt opgetekendt.
Deze cirkel is als volgt ingedeelt: van boven naar beneden

- Rood/geel balk ter controle van de vertragings lijn.
- Zwart reflectie balk ter controle van reflecties en uitslingeringen van midden frequent gedeelte en dergelijke.
- Grijs gradatie balk ter controle van de lineairietiet van de video eindversterker(s).
- Kleuren balken ter controle van de kleur weergave.
- Het midden van de cirkel is voorzien van een kruis ten behoeve van de convergentie.
- Frequentie balken ter controle van de frequentie

karakteristieken van onder ander MF gedeelte en video eind versterker(s). (15)

- Grijs gradatie balk.
- Geel/rood balk ter controle van de vertragings lijn.

Tenslotte van dit programma bevindt een eigen variatie van de SGT.(Slow Graf Text)
Dit programma is voor een groot gedeelte overgenomen uit het DAI handboek.Allerlei beveiligingen die moeten voorkomen dat de text string van het beeld af kan lopen zijn verwijderdt.Daar dan tegen is dit programma uitgebreidt met de optie dat men in staat stelt text onder allerlei veschillende hoeken te doen optekenen,dit in tegenstelling van andere versies van graf text waar dit allen mogenlijk is met hoeken met veelvoud van 90 graden.
SGT is aan te roepen met 'GOSUB 40000'.
In het hoofd programma moet men ten behoeve deze subroutine een extra 'CLEAR 1400' plegen.

- Variabele 'A$' bevat de te printen text string.
- Variabelen 'X' en 'Y' bevatten de coordinaten van het startpunt van te printen string.
- Variabele 'DEV' is de hoek waarmede de string wordt afgedrukt, in radialen, tegen de klok in.
- 'C' bevat de kleur code waarmede . het wordt opgetekend
- 'F' bepaalt de vergrotings factor.

7 Cassette 'FGT' (16)

Ik ben al een enige tijd in het bezit van het programma 'FGT'.het wordt tot volle tevredenheid gebruikt.
Een nadeel van dit programma is dat het een machinetaal programma is.Het is altijd lastig om deze met een basic programma te laden,hetzij apart of deze bevindt als data opslag in het basic programma met alle gevolgen van dien.
In het nummer van het blad DAInamic geeft U een redelijke oplossing voor dit probleem, hetgeen ik nog niet uit geprobeert hebt.
Het grote voordeel dat het een machine taal programma is vindt ik ook de snelheid, welke belangrijker is dan de hier voor gegeven bezwaren.
De nu volgende file's ben ik niet in geslaagt om deze in te lezen: - Obj trigisch alfabet,
- Obj morse alfabet.

Wilt U de genoemde files mij alsnog doen toekomen?

DAI video-hardware

8 Ruilen cassette 'Viditel'. (17)

Op de HCC dag in Utrecht heb ik het programma viditel gekocht.Met dit programma heb ik nog geen ervaring omdat ik nog geen modem tot mijn beschikking heb.
In de half-duplex mode heb ik al reeds er mee gespeelt.
De belangrijkste functies die naar mening nog missen zijn:
- Het laten knipperen van een symbool.
- De mogenlijkheid van het opslaan van de pagina's.
- De mogenlijkheid van het maken van een hard copy van een pagina.

Bijgvoegd is de cassette die ik heb gekocht.
Deze zou ik graag willen ruilen voor Viditel versie 3.0.
In de bespreking van Viditel in DAInamic jan./feb '82 neem ik aan dat de genoemde functies zijn verweselijkt.

10 Algemeen, slot. (19)

Veel lof heb ik voor het clubblad DAInamic.
Praktisch elke letter wordt door mij gelezen.
De software krijgt veel aandacht in het blad, b.v. het werk van de heer Boerrigter t.a.v het uitzoeken van de DAI firmware.Ik wacht al ongeduldig op het boek over het genoemde onderwerp.
Hardware matig gebeurt er vrijwel niets.
Graag zou ik willen zien dat er een kompleet schema van de DAI computer wordt uitgegeven.
Sinds kort wordt het clubblad DAInamic gebonden, hetgeen veel beter bevalt dan de losbladige, er mee door gaan dus.
In afwachting verblijf ik

Hoogachtend

A. Doornenbal

A. Doornenbal,
Oud Aa 39a
3621 LA Breukelen.
Tel. 03462-3237

```
5     REM +++ DRAWING WITH PADDLES R.RENS +++
10    MODE 6:K%=8
20    COLORG 8 6 4 2:MODE 5
30    XM%=XMAX/2:YM%=YMAX/2:Y3%=YMAX/3:Y23%=YMAX*2/3
40    XN%=XM%:YN%=Y23%:UN%=XM%:VN%=Y3%
1000  XO%=XN%:YO%=YN%:UO%=UN%:VO%=VN%
1010  XD%=(127-PDL(1))*10/256:YD%=(PDL(2)-127)*10/256:UD%=(PDL(5)-127)*10/256:VD
%=(PDL(4)-127)*10/256
1100  XN%=XO%+XD%:IF XN%<0 THEN XN%=0
1110  IF XN%>XMAX THEN XN%=XMAX
1120  YN%=YO%+YD%:IF YN%<Y3% THEN YN%=Y3%
1130  IF YN%>YMAX-1 THEN YN%=YMAX-1
1140  UN%=UO%+UD%:IF UN%<0 THEN UN%=0
1150  IF UN%>XMAX-1 THEN UN%=XMAX-1
1160  VN%=VO%+VD%:IF VN%<0 THEN VN%=0
1170  IF VN%>Y23% THEN VN%=Y23%
1200  DRAW XO%,YO% XN%,YN% K%:DRAW UO%,VO% UN%,VN% K1%
1220  EV1%=(PEEK(#FDOO) IAND #20) SHR 5:IF EV1%=1 THEN FILL 0,Y3% XMAX,YMAX 8
1230  EV2%=(PEEK(#FDOO) IAND #10) SHR 4:IF EV2%=1 THEN FILL 0,0 XMAX,Y23% 8
1300  Z%=PDL(0):ZD%=(Z%-1)/16:K%=ZD%
1310  W%=PDL(3):WD%=(W%-1)/16:K1%=WD%
1400  GOTO 1000
```

large size characters

```
10    REM LARGE SIZE CHARACTERS / F.H. DRUIJFF 1/82
20    PRINT CHR$(12);
30    PRINT "LARGE SIZE CHARACTERS."
40    PRINT "======================"
50    PRINT :PRINT "THIS PROGRAM SHOWS ANOTHER POSSEBILITY OF DAI."
60    PRINT "YOU CAN CREATE LARGE SIZE CHARACTERS BY CHANGING 7 IN THE"
70    PRINT "FIRST HALF OF THE CONTROLBYTE IN 6,5 OR 4."
80    PRINT :PRINT "SINCE THE REAL INFORMATION IS ONLY TWO BITS OF THIS BYTE"
90    PRINT "IT IS USELESS TO TRY OTHER NUMBERS THEN 7,6,5 OR 4."
100   PRINT "IF THE CHARACTERS HAVE A LARGER SIZE IT WILL BE CLEAR THAT"
110   PRINT "LESS CHARACTERS WILL FIT ON ONE LINE. IF YOU USE 7 THERE IS"
120   PRINT "PLACE FOR 66, WITH 6 44,WITH 5 22 AND WITH 4 11 CHARACTERS.":PRINT
130   PRINT "IF YOU WANT THE WHOLE SCREEN TO USE WITH NONSTANDARD CHARAC-"
140   PRINT "TERS, YOU'LL HAVE TO REBUILD THE SCREENAREA BY YOURSELF."
150   PRINT "STARTING EACH LINE WITH TWO(!) CONTROL BYTES; FILLING THE"
160   PRINT "LINE WITH CHARACTER AND COLOR INFORMATION."
170   PRINT "ALSO WILL THE NEW CONTROLBYTE HAVE TO FOLLOW ON THE EXPECTED"
180   PRINT "DISTANCE. (DEPENDING ON THE SIZE YOU USE)"
190   PRINT "NORMALY YOU ONLY USE LARGER SIZE FOR HEADINGS."
200   PRINT "A SIMPLE WAY OF USING THIS IS: KEEP NORMAL LENGTHS AND WIPE"
210   PRINT "UNDESIRED INFORMATION WITH A 'COLORT x y x x' INSTRUCTION."
220   PRINT :PRINT "START DEMONSTRATION BY TYPING SPACE.";
230   IF GETC=0 GOTO 230
240   PRINT CHR$(12):PRINT
250   COLORT 8 0 8 8
260   T$="* DAI *"
270   PRINT T$
280   WAIT TIME 100
290   POKE #BFEF-2*#86,#6A
300   WAIT TIME 100
310   POKE #BFEF-2*#86,#5A
320   WAIT TIME 100
330   POKE #BFEF-2*#86,#4A
340   PRINT :PRINT :PRINT :PRINT "NOW ON ONE PAGE."
350   WAIT TIME 100
360   PRINT CHR$(12)
370   PRINT SPC(20);T$
380   PRINT :PRINT SPC(12);T$
390   PRINT :PRINT SPC(4);T$
400   PRINT :PRINT T$
410   POKE #BFEF-3*#86,#6A
420   POKE #BFEF-5*#86,#5A
430   POKE #BFEF-7*#86,#4A
440   WAIT TIME 100
450   PRINT :PRINT :PRINT :PRINT "RESTART DEMONSTRATION WITH SPACEBAR AND STOP W
ITH S."
460   H=GETC
470   IF H=32 GOTO 240
480   IF H<>83 GOTO 460
490   END
```

beer

```
10    GOTO 200:REM BEER / F.H. DRUIJFF  2/82
20    H=(J+4)/10:DRAW X-H,J X29+H,J C:RETURN
30    FOR X=11 TO 291 STEP 56:X10=X+10:X18=X+18:X29=X+29
40    FOR Y=147 TO 2 STEP -3:FILL X10,Y X18,Y+2 22:NEXT
50    K=0:C=23:FOR J=2 TO 7:WAIT TIME 2:GOSUB 20:NEXT
60    FOR I=8 TO 63:C=22:J=I-6:GOSUB 20:WAIT TIME 2
70    C=23:J=I+K:GOSUB 20:IF I MOD 4=0 THEN K=K+1:J=I+K:GOSUB 20
80    NEXT:FOR Y=147 TO 80 STEP -3:IF Y=Y/24*24 THEN J=J+1:GOSUB 20
90    FILL X10,Y X18,Y+2 20:NEXT:J=J+1:GOSUB 20:X1=X+16:X2=X+18:X3=X+17
100   FOR Y=149 TO 147 STEP -1:DOT X3,Y 22:WAIT TIME 2
110   DRAW X1,Y X2,Y 22:WAIT TIME 2:NEXT:DOT X3,146 22
120   FOR Y=145 TO 83 STEP -2:FILL X1,Y X2,Y+1 22
130   DOT X3,Y-1 22:FILL X1,Y+3 X2,Y+4 20:NEXT
140   DRAW X+10,82 X+24,82 23:FILL X1,84 X2,85 20:T=1-T
150   IF T=1 GOTO 100:DRAW X+14,83 X+22,83 23:NEXT:END
200   MODE 6A:COLORT 5 5 5 5:COLORG 6 0 14 15:FOR X=10 TO 290 STEP 56
210   DRAW X,0 X-8,81 21:FILL X,0 X+30,1 21:DRAW X+31,0 X+39,81 21
220   DRAW X-1,0 X-9,81 21:DRAW X+32,0 X+40,81 21:FILL X+10,154 X+20,211 21
230   DRAW X+13,154 X+13,211 22:FILL X+8,150 X+22,153 21:NEXT:GOTO 30
```

alternative printer routine

The RS232 interface on your DAI is designed for operation of a printer via a handshake protocol. The serial output interface chip 5501 sends only data to the printer if a 'ready' signal is received. The software routine for serial output can be found on the addresses #DD94-#DDB3.

Several times I have been called by people who wanted to use a printer without the handshake facilities as expected by the DAI. Therefore I designed the annexed alternative printer routine.

This specific routine is written for a printer which had timing problems with the line feed signal. But we will see that the routine can easily be adapted to cope with other problems.

The principle of the routine is based on the DAI feature that input and output routines, available in the ROM's, can be replaced by any other, user specified, routine. As can be found in the memory map, the output direction is determined by the output direction pointer OTSW on address #131. For normal RS232 operation, it has the value #00. But if the value is #03, it points to the instruction on address #02DD (DOUTC). Normally, you will find a return instruction on this address, but this can be replaced by a jump instruction to a user specified routine in RAM.

The alternative routine starts with a initialisation routine. It changes the startaddress of the Heap to after this ML program. Then on address #02DD a jump to the alternative printer routine is loaded. If you load the routine directly after switching on the DAI, it must be followed immediately by UT > G300 before loading a Basic program.
(Note: For machines with Basic V1.1 the NEW routine is on address #DEB8!).

The alternative printer routine starts on #0317. It is the same routine as on #DD94, except the additional delay on #0337. When required, another delay routine can be used for more or for less delay time.
The comments annexed to the ML routine indicate how to use this routine.

If your printer doesnot have any handshake facilities at all, the instructions on #0322-#0327 (waiting for a ready signal from the printer) can be cancelled. If required, a delay time can be inserted in stead of this 'wait for printer ready'.

It is possible of course to integrate this alternative printer routine in a Basic program. See therefore the article 'Combination Basic and ML programs/rules for standardization'
But then the lines #301-#309 must be skipped!

Jan Boerrigter - Febr.'82

alternative printer routine

```
002                     *
003                             ORG   :300
004                     *
005 0300 E5             INIT    PUSH  H
006 0301 213C03                 LXI   H,LAST
007 0304 229B02                 SHLD  :029B     HEAP POINTER AFTER ROUTINE
008 0307 CDB5DE                 CALL  :DEB5     RUN NEW (SHIFT TEXTBUF)
009 030A 21DD02                 LXI   H,:02DD   )
010 030D 36C3                   MVI   M,:C3     ) SET DOUTC FOR
011 030F 211703                 LXI   H,RS232   ) REPLACEMENT
012 0312 22DE02                 SHLD  :02DE     ) ROUTINE
013 0315 E1                     POP   H
014 0316 C9                     RET
015                     *
016                     *
017                     *
018 0317 F5             RS232   PUSH  PSW
019 0318 EF                     RST   5         OUTPUT TO SCREEN
020 0319 03                     DATA  :03
021 031A 3A00FD         OUTRDY  LDA   :FD00
022 031D E608                   ANI   :08
023 031F CA1A03                 JZ    OUTRDY    WAIT OUTPUT READY
024 0322 3AF3FF         BUFEMP  LDA   :FFF3
025 0325 E610                   ANI   :10
026 0327 CA2203                 JZ    BUFEMP    WAIT BUFFER EMPTY
027 032A F1                     POP   PSW
028 032B 32F6FF                 STA   :FFF6     OUTPUT CHAR
029 032E FE0D                   CPI   :0D       CAR.RET?
030 0330 C0                     RNZ             READY WHEN NOT
031 0331 F5                     PUSH  PSW
032 0332 3E0A                   MVI   A,:0A
033 0334 CD1703                 CALL  RS232     PRINT LINE FEED
034 0337 CD41DE                 CALL  :DE41     DELAY
035 033A F1                     POP   PSW
036 033B C9                     RET
037                     *
038 033C 00             LAST    NOP
039                     *
040                     *
041                     * After loading this routine: UT >G 0300,
042                     * to run initialisation.
043                     *
044                     * In Basic program: POKE #131,3 to activate
045                     * printer; POKE #131,1 to switch off.
046                     *
047                     * For use with machine language programs, the ORG
048                     * must refer to an address outside the ML program!
049                     *
050 033D                        END

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

BUFEMP 0322   INIT   0300   LAST   033C   OUTRDY 031A
RS232  0317
```

spot on text

```
10    REM SPOT ON TEXT / F.H. DRUIJFF 1/82
20    PRINT CHR$(12);:COLORT 8 0 8 14
30    PRINT "SPOT ON TEXT"
40    PRINT "============":PRINT
50    PRINT "IN THIS PROGAM WILL BE SHOWN THAT YOU CAN CHANGE TEXTCOLOR"
60    PRINT "AND BACKGROUND COLOR OF ANY PART OF THE TEXT."
70    PRINT :PRINT "YOU PERFORM THIS BY FILLING THE COLORBYTES OF THE CHARACTERS
"
80    PRINT "YOU WANT TO CHANGE WITH #FF (DECIMAL 255)."
90    PRINT "IF YOU CHOOSE ANYTHING ELSE THEN #FF THEN ONLY THE COLUMS OF"
100   PRINT "THE CHARACTERMATRIX WILL CHANGE OF WHICH THE CORRESPONDING"
110   PRINT "BIT IN THE COLORBYTE IS 1."
120   PRINT "IF THIS IS TOO COMPLICATED AT THE MOMENT JUST TRY SOMETHING"
130   PRINT "AND REMEMBER TO CONVERT THE NUMBER YOU INTEND TO POKE IN BI-"
140   PRINT "NARY NOTATION."
150   PRINT "THE BYTE TO POKE IS FOUND IN THE FOLLOWING WAY:"
160   PRINT "?#BFEF-Y*#86-X*2-3 WITH X=POSITION IN LINE AND Y=LINE-1."
170   PRINT "REMEMBER THAT FIRST 3 CHARACTERS ARE NOT PRINTED BY PRINT."
180   PRINT :PRINT "THE COLORS USED ARE GIVEN BY A 'COLORT' INSTRUCTION."
190   PRINT "THE POKED COLORBYTES WILL FORCE THE SECOND SET."
200   FOR I=#BFEF-6*#86-36*2-3 TO I-18 STEP -2
210   POKE I,#FF:NEXT
220   FOR I=#BFEF-9*#86-8*2-3 TO I-28 STEP -2
230   POKE I,#FF:NEXT
240   FOR I=#BFEF-11*#86-19*2-3 TO I-20 STEP -2
250   FOR J=0 TO 8:POKE I,2^J:NEXT:NEXT
260   PRINT :PRINT "CONTINUE BY SPACE STOP WITH S."
270   C3=RND(16):C4=RND(16):IF C3=C4 GOTO 270
280   COLORT 8 0 C3 C4
290   CURSOR 0,1:PRINT "YOU ARE USING NOW COLORT 8 0";C3;C4;"    ";
300   G=GETC:IF G=32 GOTO 270:IF G<>83 GOTO 300
310   COLORT 8 0 0 0
320   PRINT CHR$(12):END
```

tumbling lines

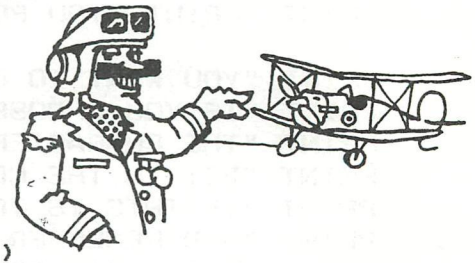
```
5     COLORG 0 15 15 15
10    REM TUMBLING LINES  F.H. DRUIJFF  2/82
20    MODE 4:CLEAR 2000:DIM X(18),Y(18):GOSUB 90:GOTO 50
30    FOR I=17 TO 0 STEP -1:P=X-X(I):DRAW X,0 P,Y(I) 22:DRAW X,0 P,Y(I) 0:NEXT:R
ETURN
40    FOR I=0 TO 18:P=X+X(I):DRAW X,0 P,Y(I) 22:DRAW X,0 P,Y(I) 0:NEXT:RETURN
50    FILL 0,0 24,26 22:FOR A=24 TO 0 STEP -1
60    X=A:GOSUB 40:E=XMAX-28:IF A=0 THEN E=E-26
70    FOR X=A+26 TO E STEP 26:GOSUB 30:GOSUB 40:NEXT
80    GOSUB 30:DRAW X,0 P,Y(0) 22:NEXT:END
90    FOR I=0 TO 18:READ X(I):Y(18-I)=X(I):NEXT:RETURN
100   DATA 0,2,4,6,8,10,12,14,16,18,20,21,22,23,24,25,25,26,26
```

patchwork

T

```
10    REM PATCHWORK /  F.H. DRUIJFF  2/82
20    MODE 1:DIM C(3):REM PROGRAM RUNS ALSO IN MODE 3 OR 5
30    FOR I=XMAX-7 TO 0 STEP -8:FOR J=0 TO YMAX-3 STEP 4
40    FOR K=0 TO 3:C(K)=RND(16):NEXT:FILL I,J I+7,J+3 RND(16)
50    FOR Y=J TO J+3:M=2-M:FOR X=I+7 TO I STEP -1:N=1-N
60    K=M+N:DOT X,Y C(K):NEXT:NEXT:NEXT:NEXT:GOTO 30
```

flug simulator



```
10    REM FLUG-SIMULATOR        A.MEYSTRE  ANDLAUERSTR. 10
15    REM FEB. 82               CH-4132 MUTTENZ    SCHWEIZ
20    REM ------------------------------------------------
25    REM
30    REM DIESER EINFACHE FLUG-SIMULATOR ZEIGT EINIGE
35    REM VIDEO-EIGENSCHAFTEN DES DAI-COMPUTERS AUF.
40    REM
50    REM WERT MODUS              PUNKTE/ZEILE  BYTES/ZEILE
52    REM 0    UNIT-COLOR-MODE         528           4
55    REM 2    LOW                      88          24
58    REM 4    MEDIUM                  176          46
60    REM 6    HIGH                    352          90
62    REM 8    ULTRA                   528         134
65    REM
68    REM FALSCHE EINGABE ERGIBT WERT 0
70    REM
72    REM TASTEN-FUNKTIONEN IM FLUG  ( mit REPT-Taste )
75    REM
78    REM CURSOR AUF/AB              zum STEIGEN und FALLEN
80    REM CURSOR AUF/AB + SHIFT      langsamer / schneller
85    REM CURSOR AUF+AB              Ruettelflug
90    REM M-Taste                    Dieses Menue
95    REM
110   REM Bis zum fertigen Flugsimulator mit Piste und Berge
120   REM ist noch ein weiter Weg. Diese einfache Version
130   REM soll nur die Video-Moeglichkeiten durch
140   REM Manipulation der Zeilenkontrollbytes aufzeigen.
150   REM Die Wirkung ist augenblicklich !
160   REM Die Darstellung braucht in allen MODES 31 Zeilen.
170   REM Weitere Zeilen verschwinden am unteren Bildrand.
180   REM
190   REM Je nach MODE sind   LI  Anzahl Bytes/Zeile
200   REM .                  MCO  Colorbyte-Maske
210   REM .                 MASK  Modebyte-Maske
220   REM zu setzen.
1000  CLEAR 1000
1010  PRINT CHR$(12):LIST -100:MCO=#C0:MIX=0
1020  INPUT "99    REM GEWUENSCHTER MODUS EINGEBEN ";GMO:PRINT
1030  IF GMO=2 THEN MODE 2:LI=24:MASK=#0:GOTO 1090
1040  IF GMO=4 THEN MODE 4:LI=46:MASK=#10:GOTO 1090
1050  IF GMO=6 THEN MODE 6:LI=90:MASK=#20:GOTO 1090
1051  REM
1052  REM Fuer die Darstellungen 0 und 8, die im BASIC nicht
1053  REM unterstuetzt werden, sind stoerende Kontrollbytes
1054  REM von MODE 6 auf 00 zu setzen.
1055  REM LCB zeigt jeweils auf das Kontrollbyte.
1056  REM
1060  MODE 6:LCB=#BFEF:FOR I=0 TO 60:POKE LCB,0:POKE LCB-1,0:LCB=LCB-90:NEXT
1070  IF GMO=8 THEN LI=134:MASK=#30:GOTO 1090
1080  LI=4:MCO=#80:MASK=#30
1090  COLORG 1 12 12 1:REM Grundfarben ( Himmel )
1099  REM Tabellen fuer 31 Zeilen (von oben nach unten)
1100  DIM BL(30.0),BC(30.0),MIX(7.0)
1199  REM Zeilenbreite
1200  DATA 15,10, 1, 2, 3, 4, 4, 3, 2, 1,10,15,15,15, 1, 0, 1, 2, 3, 4, 5, 6, 7,
 8, 9,10,11,12,13,14,15
1299  REM Hintergrundfarbe der Zeile
1300  DATA 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,12,12,12,12,15,15,14,14,13,13,13, 6, 6,
5, 5, 5, 5, 7, 7, 5, 7
1310  REM Muster zur Farbmischung Dunkelblau/Hellblau
1320  DATA #20,#84,#51,#AA,#55,#BA,#6F,#FD
1400  RESTORE:REM Einlesen von Zeilenbreite und Farbe
1500  FOR I=0 TO 30:READ BL(I):NEXT:FOR I=0 TO 30:READ BC(I):NEXT:FOR I=0 TO 7:R
EAD MIX(I):NEXT
```

```
1505  REM
1510  REM In der Bildmitte ist der duenne, weisse Horizont.
1520  REM Im oberen Bildteil wird der Himmel durch
1530  REM verschiedene Blautoene dargestellt. Die untere
1540  REM Bildhaelfte zeigt dahingleitende Felder in den
1550  REM Farben gruen, gelb und braun. Das Grundbild
1560  REM wird nun aufgebaut durch Veraenderung der
1570  REM Kontrollbytes-Informationen. LCB ist jeweils
1580  REM die Kontrollbyte-Adresse und LI ist die
1590  REM Differenz bis zum naechsten Kontrollbyte.
1595  REM
1600  LCB=#BFEF:FOR I=0 TO 30
1700  REM In den Zeilen 2 bis 9 wird ein Mischmuster eingesetzt
1800  IF I>1 AND I<10 THEN MLCB=LCB-2:FOR J=1 TO LI-2 STEP 2:POKE MLCB-J,MIX(MIX
):NEXT:MIX=MIX+1
1900  POKE LCB,BL(I) IOR MASK:POKE LCB-1,BC(I) IOR MCO:LCB=LCB-LI:NEXT
1950  REM
1960  REM U ist die Anfangsfluggeschwindigkeit
1970  REM H, H4, H8 und HC sind die Adressen der Bildkopfkontrollbytes
1980  REM Durch Veraenderung der Zeilenbreite in diesen 4 Kopfzeilen
1990  REM wird die Horizonthoehe verschoben.
1992  REM M ist eine Maske zum einsetzen der Zeilenbreite.
1995  REM
2000  U=6:H=#BFFF:H4=H-4:H8=H-8:HC=H-12:M=#30
2010  REM
2020  REM Ab hier entsteht die ganze Bewegung der unteren Bildhaelfte.
2030  REM Die Farbe der Zeile wird jeweils zum naechsttieferen Kontrollbyte
2040  REM vesetzt. Dadurch entsteht die Illusion einer Vorwaertsbewegung.
2050  REM
2100  LCB=#BFEE-LI*16:F1=PEEK(LCB)'
2110  FOR I=2 TO 15:LCB=LCB-LI:F2=PEEK(LCB):POKE LCB,F1:F1=F2:NEXT
2115  REM Am Horizont erscheint ein neues Feld mit Zufallsfarbe
2120  POKE #BFEE-LI*16,BC(15.0+RND(15.0)) IOR MCO
2122  REM
2125  K=GETC:REM einlesen der gewuenschten Bewegung
2127  REM Das Flugzeug steigt, die Geschwindigkeit nimmt dabei ab.
2128  REM Die Breite der obersten Zeilen nimmt zu, so dass der Horizont
2129  REM nach unten gleitet und Sie somit hoeher fliegen.
2130  REM
2132  IF K=16 AND U<15 THEN U=U+1:U3=U IOR M:POKE H,U3:POKE H4,U3:POKE H8,U3:POK
E HC,U3
2135  REM
2136  REM Das Flugzeug sinkt, die Geschwindigkeit nimmt zu. Die Breite der
2137  REM obersten Zeilen nimmt ab, dadurch wandert der Horizont
2138  REM gegen den oberen Bildrand. Die Erde kommt naeher und naeher.
2139  REM
2140  IF K=17 AND U>0 THEN U=U-1:U3=U IOR M:POKE H,U3:POKE H4,U3:POKE H8,U3:POKE
 HC,U3
2145  REM
2146  REM Die Geschwindigkeit nimmt ab ( die Wartezeit S wird groesser ).
2147  REM
2150  IF K=20 OR K=16 THEN S=S+1
2155  REM
2156  REM Die Geschwindigkeit nimmt zu ( die Wartezeit S wird kleiner, aber nich
t negativ ).
2157  REM
2160  IF K=21 OR K=17 THEN S=S-1:IF S<0 THEN S=0
2165  REM
2166  REM wurde M eingegeben, so erscheint das Menue.
2167  REM
2170  IF K=ASC("M") THEN MODE 0:GOTO 1010
2175  REM
2176  REM Vor der naechsten Bewegung wird umgekehrtproportional zur Geschwindigk
eit gewartet.
2177  REM
2180  WAIT TIME S:GOTO 2100
2200  REM
2210  REM Viel Spass ! Versuchen Sie mal dieses Programm zu erweitern,
2220  REM z.B. mit Motorgeraeusch, Piste, Berge, Wolken, Bordinstrumente ...
2230  REM Bitte berichten Sie mir falls Sie etwas erreichen, viel Erfolg !
```

SPACE INVADER

On some machines, and for an unknown reason,the program doesn't initiate the values of POINTS,LASER and HIGHSCORE to the right contents.
This problem can be fixed with the following patch:The area from D4-DE must be cleared before starting the game:

```
49A     JMP     :3E0        jump to patch
3E0     LXI H   :0          old instr in 49A
3E3     SHLD    :D4         clear D4-DE
3E6     SHLD    :D6
3E9     SHLD    :D8
3EC     SHLD    :DA
3EF     SHLD    :DC
3F2     SHLD    :DD
3F5     JMP     :49D        cont program
```

If you have problems with SPACE INVADER on your machine,and you cannot make the patch yourself,please return the tape,we will send a new one,at no costs.(This applies to the whole library)

Ondergetekende zou gaarne in contact treden met DAI-gebruikers die bezig zijn met,c.q. reeds ontwikkeld hebben,een programma voor het samenstellen van roosters en/of clusters voor het gebruik op scholen.
Liefst z.s.m.

R.A.B.FABER tel : 01856:2865
Keizersdijk 36
3291 CE STRIJEN NEDERLAND

OUTPUT TO RS232 ONLY

POKE H2DD,H3C : POKE H2DE,H94 : POKE H2DF,HDD : POKE H131,3

switch back with : POKE H131,Ø

---

revision 7 of DAIpc has been released : DCR_users cannot use the first version of Eprom-print on this machine,Memocom is already supplying adapted prints.

INPUT FROM KEYBOARD

```
INPUT  KEYBOARD
OUTPUT CHR IN A

IN   PUSH B
     PUSH D
     PUSH H
     CALL :D6BB
     POP  H
     POP  D
     POP  B
     RET
```

OUTPUT TO VIDEO

```
INPUT  CHR IN A
OUTPUT VIDEO

OUT  PUSH B
     PUSH D
     PUSH H
     CALL :D695
     POP  H
     POP  D
     POP  B
     RET
```

INPUT FROM RS232

```
INPUT  RS232
OUTPUT CHR IN A

INM  PUSH B
     PUSH D
     PUSH H
     LDA  :FF03
     ANI  :04
     JZ   WAIC
     MVI  A,0
     JMP  RETM
WAIC LDA  :FF03
     ANI  :08
     JZ   WAIC
     LDA  :FF00
RETM POP  H
     POP  D
     POP  B
     RET
```

OUTPUT TO RS232

```
INPUT  CHR IN A
OUTPUT RS232

OUTM PUSH B
     PUSH D
     PUSH H
     PUSH PSW
RED1 LDA  :FD00
     ANI  :08
     JZ   RED1
RED2 LDA  :FFF3
     ANI  :10
     JZ   RED2
     POP  PSW
     STA  :FFF6
     POP  H
     POP  D
     POP  B
     RET
```

8-BIT MULTIPLY

```
INPUT  D=MULTIPLICAND
       C=MULTIPLIER
OUTPUT BC=RESULT

MULT    MVI  B,:00
        MVI  E,:09
MULTO   MOV  A,C
        RAR
        MOV  C,A
        DCR  E
        JZ   DONE
        MOV  A,B
        JNC  MULT1
        ADD  D
MULT1   RAR
        MOV  B,A
        JMP  MULTO
DONE    RET
```

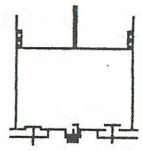
8-BIT DIVIDE

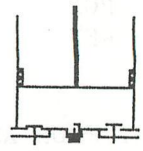
```
INPUT  B=DIVIDEND
       C=QUOTIENT
OUTPUT D=DIVISOR
       E=REMAINDER

DIV     MOV  A,D
        CMA
        MOV  D,A
        MOV  A,E
        CMA
        MOV  E,A
        INX  D
        LXI  H,:00
        MVI  A,:17
DVO     PUSH H
        DAD  D
        JNC  DV1
        XTHL
DIV1    POP  H
        PUSH PSW
        MOV  A,C
        RAL
        MOV  C,A
        MOV  A,B
        RAL
        MOV  B,A
        MOV  A,L
        RAL
        MOV  L,A
        MOV  A,H
        RAL
        MOV  H,A
        POP  PSW
        DCR  A
        JNZ  DVO
        ORA  A
        MOV  A,H
        RAR
        MOV  D,A
        MOV  A,L
        RAR
        MOV  E,A
        RET
```

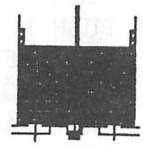
4-takt motor

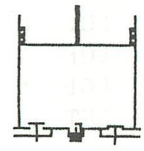
```
10    REM DIT PROGRAMMA SIMULEERT EEN 4 TAKT MOTOR
20    POKE #75,32
110   REM VAST GEDEELTE
120   PRINT CHR$(12)
133   PRINT
135   COLORT 0 8 10 15:COLORG 0 8 15 6
136   MODE 4A
137   DRAW 37,10 43,10 21
138   DRAW 49,10 59,10 21
139   FILL 40,15 85,11 23
140   DRAW 39,10 39,56 21:DRAW 86,10 86,56 21
150   DRAW 37,7 52,7 21
160   DRAW 72,7 88,7 21
170   DRAW 65,10 75,10 21
180   DRAW 81,10 88,10 21
190   DRAW 52,7 52,9 21:DRAW 59,7 59,9 21
195   DRAW 65,7 65,9 21:DRAW 73,7 72,9 21
200   FILL 60,5 64,8 21
250   DOT 62,10 21:DOT 62,9 21
260   DRAW 64,9 64,12 21
270   DOT 62,12 21:DOT 63,12 21
280   REM KLEPPEN BEGINSTAND
290   DRAW 46,4 46,11 22:DRAW 78,4 78,11 22
300   DRAW 43,11 49,11 22:DRAW 75,11 81,11 22
320   REM ZUIGER BEGINSTAND
330   DRAW 40,15 85,15 22:DRAW 40,14 85,14 23
340   FILL 41,15 40,22 22:FILL 85,15 84,22 22
345   DOT 40,17 20:DOT 40,20 20:DOT 85,17 20:DOT 85,20 20
350   REM DRIJFSTANG
360   FILL 62,15 63,60 22
400   REM #### WERKING VAN DE MOTOR ####
500   REM INLAATSLAG
510   PRINT CHR$(12);"INLAATSLAG"
520   DRAW 43,12 49,12 22:DRAW 43,11 49,11 23
530   DOT 46,11 22:DOT 46,4 20
540   GOSUB 1000
550   REM COMPRESSIESLAG
560   DRAW 43,11 49,11 22:DRAW 43,12 49,12 23:DOT 46,4 22
570   PRINT CHR$(12);"COMPRESSIESLAG"
580   GOSUB 2000
620   REM ARBEIDSLAG
640   PRINT CHR$(12);"ARBEIDSLAG"
645   COLORG 0 8 15 10
650   GOSUB 1000
655   COLORG 0 8 15 6
670   DRAW 75,12 81,12 22:DRAW 75,11 81,11 23
680   DOT 78,11 22:DOT 78,4 20
710   PRINT CHR$(12);"UITLAATSLAG"
720   GOSUB 2000
730   DRAW 75,11 81,11 22:DRAW 75,12 81,12 23
740   DOT 78,4 22
750   GOTO 500
1000  POKE #305,#5B:POKE #306,#B6
1010  POKE #308,#2D:POKE #309,#B6
1020  POKE #31C,#D2:POKE #31D,#FF
1030  FOR J%=0 TO 28:CALLM #300:NEXT:RETURN
2000  POKE #31C,46:POKE #31D,0
2010  POKE #305,#88:POKE #306,#AF
2020  POKE #308,#B6:POKE #309,#AF
2030  FOR J%=0 TO 28:CALLM #300:NEXT:RETURN
```

```
0300 E5 D5 C5 F5 21 88 AF 11 B6 AF D5 06 26 0E 0E 1A
0310 77 2B 1B 0D 79 B7 C2 0F 03 E1 E5 11 2E 00 19 5D
0320 54 E3 05 78 B7 C2 0D 03 F1 F1 C1 D1 E1 C9 20 20
```









```
*********************************************
*                                           *
* COMBINATION OF MACHINE LANGUAGE AND BASIC *
* ========================================= *
*      - rules for standardization -        *
*                                           *
*********************************************
```

Several programs, written in Basic, require that some parts are written in machine language to obtain a higher execution speed. There are several ways to combine these two parts. The most common methods are:

- Keep them seperate. Before program execution, both routines must be loaded into the machine.
- Insert the ML program into DATA statements. Via a READ command, the instructions will be moved into an array during program execution.

The first method is not very useful when you just learned your kids how to load a simple Basic program; the second one needs a lot of typing with a high failure rate (although this can be overcome by using a data statement generator; but that requires more memory).

In order to get a normalisation in this matter, the following procedure is proposed as a standard for combining Basic and ML into one program.

Some standards are required:

- The startaddress of the Heap will always be #02EC (its default value).
- The ML routine starts always at #0300.

The length of the ML program is called MLPLEN (use hex-value!).

The method of combining Basic and ML routines is based on the principle that the ML program is moved into the text buffer. When you save a Basic program, both textbuffer and symboltable contents are written on tape, so now the ML routine is saved too. After loading, the ML part must be moved back to its original location.

The Basic program must contain the following lines at the very beginning of the program:

```
1 POKE #29B,#EC: POKE #29C,#02        Default start Heap
2 CLEAR xxxx                          xxxx > MLPLEN
3 MLPLEN=yy                           yy = length ML program
4 QQ=INT(SQR(MLPLEN/4+4))
5 DIM MLP(QQ,QQ)
6 STMLP=PEEK(#2A1)+PEEK(#2A2)*256-MLPLEN-1
7 FOR I=0 TO MLPLEN-1
8 POKE #300+I,PEEK(STMLP+I)
9 NEXT
```

These lines reserve by means of a dummy array memory space for the ML program. A 2-dimensional array is used to cope with ML programs of more than 1K. From line 7 onwards, the ML bytes are moved from the textbuffer into this area. This may last some time (FGT 8 secs). The ML routine can be called by Basic with CALLM #300.

The procedure (to be used during program development) for moving the ML program into the textbuffer is as follows:

```
Hard RESET
* CLEAR xxxx                          xxxx > MLPLEN
* UT
> R MLprogram
> B
```

```
* LOAD "Basic program"
* UT
> D2A1 2A4                                  (2A2/1)=STBBGN
                                            (2A4/3)=STBUSE
> M(2A2/1) (2A4/3) (2A2/1)+MLPLEN+1         Shift symboltable
> S(2A2/1) into (2A2/1)+MLPLEN+1            Update STBBGN
> S(2A4/3) into (2A4/3)+MLPLEN+1            Update STBUSE
> M300 300+MLPLEN (2A2/1)
> B
* SAVE "Basic program incl. MLP"
```

The CLEAR reserves space for the ML program. After loading both parts, the symboltable is moved to make place for the MLP. Then the pointers for the symboltable are updated. Now the MLP is moved into the now free area after the textbuffer. After saving, the program now on tape contains both Basic and ML programs. It can be loaded, executed, updated, and listed (listing the MLP part produces rubbish of course). Editing is only allowed until the last Basic linenumber!! Editing the ML part destroys your program!!!!

As an example, the transfer of FGT will be demonstrated. FGT is written on the memory locations #300-#900, but saved on tape with the Heap pointers: #29B-#900.

```
> R FGT(29B-900)
> W300 900 FGT(300-900)                     Save FGT only
Hard RESET
* CLEAR #700                                MLPLEN=#600
* UT
> R FGT(300-900)
> B
* LOAD "Basic program"
* UT                                        Suppose you find:
> D2A1 2A4                                     STBBGN = #126A
                                               STBUSE = #1325
> M126A 1325 186B                           #186B=#126A+#600+1
> S2A1 6A-6B 12-18 25-26 13-19              #1926=#1325+#600+1
> M300 900 126A
> B
* SAVE "Complete program"
```

If the ML routine needs RAM-space for pointers, the area #2EE-#2FF can be used. If you need more space, place them above the ML routine. This area must be also counted for MLPLEN!

A final remark: This method is a proposal for standardization. You don't have to follow it. As long as you only write programs to be used by yourself, nobody cares about any standard. But if you intend to send your programs to DAInamic for the software library, to be used by other people, then certain rules are necessary.
I don't know your opinion. But I hate programs that require the loading of seperate routines. I hate to find out after loading the routines that I should have moved the Heap before loading. I also hate all those different startaddresses of ML programs. I do like those programs which require only a simple LOAD command after the machine is switched on. And I am sure you like that too!

Jan Boerrigter

DAI interface module

```
174                 * IF FIRST CHAR IS INPUTTED ON RS232 DEVICE THEN
175                 * INSW IS SET TO 1.
176                 * NOTE THAT INSW IS ALSO USED FOR OTHER PURPOSES
177                 * IN DAI DOS AND MEMOCOM DCR SOFTWARE.
178                 *
179                 GETFRC  EQU   :D6BB     FORCE 1 SCAN TO GET CHAR
180                 * ENTRY : INPUT SWITCH INSW SET TO 0 OR 1
181                 * FUNC  : IDEM GETC, KNSCAN = 0 NOT NEEDED, BUT
182                 *         GETFRC HAS POOR DEBOUNCING.
183                 * EXIT  : AS GETC
184                 *
185                 OUTC    EQU   :DD60     OUTPUT A CHAR
186                 * ENTRY : ASCI VALUE IN A
187                 *         OTSW SET TO 0, 1, 2 OR 3
188                 * FUNC  : OUTPUT CHAR TO DEVICE OR BUFFER AS
189                 *         SPECIFIED IN OTSW
190                 * EXIT  : A, F CORRUPTED
191                 *         OTSW = 0 : CHAR ON SCREEN (AT CURSOR)
192                 *                    AND TO RS232
193                 *         OTSW = 1 : CHAR ON SCREEN ONLY
194                 *         OTSW = 2 : CHAR APPENDED TO TEXT IN EDIT-
195                 *                    BUFFER (DON'T USE, EDITBUFFER
196                 *                    POINTERS MUST BE SET UP FIRST)
197                 *         OTSW = 3 : CHAR SEND TO A ROUTINE VIA
198                 *                    A VECTOR ON DOUTC
199                 *                    (NORMALLY DO NOTHING,JUST RET
200                 *                    USED FOR DISK OPERATING SYSTEM)
201                 OTSW    EQU   :131      OUTPUT SWITCHING
202                 *
203                 OUTCPR  EQU   :D695     OUTPUT CHAR, PRESERVE PSW
204                 * ENTRY : IDEM OUTC
205                 * FUNC  : IDEM OUTC
206                 * EXIT  : IDEM OUTC, BUT A, F NOT CORRUPTED
207                 *
208                 CRLF    EQU   :DD5E     OUTPUT A CARRIAGE RETURN
209                 * ENTRY : IDEM OUTC, BUT CHAR IS FORCED TO
210                 *         ASCI VALUE :D
211                 * FUNC  : FOR CRLF IDEM OUTC
212                 * EXIT  : IDEM OUTC
213                 *
214                 *
215                 COLO    EQU   :DD55     OUTPUT CR IF CURX <> 0
216                 * ENTRY : OTSW SET TO 0, 1, 2 OR 3
217                 * FUNC  : IF CURX <> 0 THEN OUTPUT CAR RET
218                 * EXIT  : IDEM OUTC
219                 *
220                 *
221                 * PART 3 : MESSAGE OUTPUT ROUTINES
222                 * ------------------------------
223                 *
224                 PMSG    EQU   :DAD4     PRINT A MESSAGE
225                 * ENTRY : POINTER TO MESSAGE IN HL
226                 * FUNC  : PRINT A MESSAGE, WHICH MAY INCLUDE
227                 *         INTERNAL REFERENCES TO OTHER SUBMESS.
228                 * EXIT  : POINTER AFTER STRING IN HL
229                 * MESSAGE FORMAT :
230                 *   A SERIES OF BYTES :
231                 *     0      -> END OF STRING
232                 *     1-:7F  -> ASCI VALUE CHAR TO PRINT
233                 *      >:80  -> BITS 0-13 ARE OFFSET FROM :C000 TO ..
234                 *                 IF BIT 14 = 1 THEN
235                 *                   .. TO A MESSAGE (CHARS TERMINATED
236                 *                      BY A 0)
237                 *                 IF BIT 14 = 0 THEN
```

```
238                     *                    .. TO A STRING (1 BYTE LENGTH
239                     *                      + CORRECT NUMBER OF CHAR)
240                     * REF : MESSAGES IN DAI ROM STARTING AT ADDR :DB6F
241                     *       AND POINTERS TO ERROR MESSAGES AT :DA94
242                     *
243                     PMSGR   EQU   :DAFF     PRINT A MESSAGE
244                     * ENTRY : POINTER TO MESSAGE IN THE 2 MEMORY LOCA-
245                     *         TIONS AFTER THE CALL TO PMSGR (LOW-HIGH)
246                     * FUNC  : IDEM PMSG
247                     * EXIT  : IDEM PMSG
248                     *
249                     PSTR    EQU   :DB32     PRINT A STRING
250                     * ENTRY : POINTER TO STRING IN HL
251                     * FUNC  : PRINT A STRING
252                     * EXIT  : HL POINTS AFTER STRING
253                     * STRING FORMAT :
254                     *   1 BYTE LENGTH ( 0 =  NO DATA )
255                     *   N BYTES DATA (ASCI VALUES)
256                     *
257                     PSTRM   EQU   :DB44     PRINT A STRING
258                     * ENTRY : POINTER TO STRING IN HL
259                     *         LENGTH OF STRING IN A
260                     * FUNC  : PRINT A STRING
261                     * EXIT  : HL POINTS AFTER STRING
262                     * FORMAT :
263                     *   N BYTES DATA (ASCI VALUES)
264                     *
265                     *
266                     * PART 4 : GETTING INPUT AND ENCODE IT
267                     * ------------------------------------
268                     *
269                     * NOTE : DAI BASIC USE A SPECIAL WAY TO HANDLE
270                     *        INPUT OF COMMANDS, PROGRAM LINES AND
271                     *        'INPUT' VALUES.
272                     *        FIRST ALL INPUTTED KEYS ARE SEND TO THE
273                     *        SCREEN (=TO THE CURRENT LINE).
274                     *        THIS CONTINUES UNTIL A CARRIAGE RETURN IS
275                     *        PRESSED.
276                     *        NOW THE PROPER ROUTINE FETCHES THE
277                     *        CHARACTERS BACK FROM THE SCREEN (=FROM THE
278                     *        CURRENT LINE) TO COMPILE OR TRANSLATE THEM.
279                     *        ADVANTAGE : NO NEED FOR A EXTRA INPUTBUFFER
280                     *        TO STORE CHARACTERS ; CHAR DEL IS HANDLED
281                     *        BY THE SCREEN PACK.  A CURRENT LINE MAY
282                     *        HAVE 3 CONTINUATION LINES.
283                     *        ABOVE METHOD IS NOT USED IN PC UTILITY.
284                     *
285                     INLINE  EQU   :DD1A     GET CHARACTERS UNTIL CR
286                     * ENTRY : PROMPTCHARACTER IN A
287                     * FUNC  : IF CURX<>0 THEN PRINT CR (DO COLO)
288                     *         PRINT CHARACTER IN A
289                     *         ENABLE FULL KEYBOARDSCAN (KNSCAN->0)
290                     *         INPUT CHARACTERS FROM KEYBOARD OR RS232
291                     *         ( REF TO GETC FUNCTION )
292                     *         IF CHAR = CAR RET THEN TERMINATE
293                     *         IF BREAK PRESSED THEN TERMINATE
294                     *         IF PRINTABLE OR CHARDEL THEN SEND
295                     *                 CHARACTER TO SCREEN
296                     *         INPUT NEXT CHARACTER
297                     * EXIT  : CY = 0 : VALID TERMINATE WITH CAR RET
298                     *                  REG C IS SET TO 1 ( POSITION
299                     *                  FIRST INPUTTED CHARACTER )
300                     *                  NOTE : NO CR IS OUTPUTTED TO
301                     *                  THE SCREEN -> CURSOR AFTER
```

DAI interface module

```
302                 *                   END OF INPUTTED CHARACTERS
303                 *          CY = 1 : BREAK PRESSED
304                 *                   LINE IS TERMINATED WITH A '!'
305                 *                   CHARACTER AND A CAR RET
306                 *          KNSCAN IS ALWAYS SET FOR SCAN BREAK ONLY
307                 * NOTE  : A CHAR IS PRINTABLE IF ASCI VAL >= :20
308                 *
309                 INLINX  EQU   :DD1F     GET CHARACTERS, NO COLO
310                 * ENTRY : IDEM INLINE
311                 * FUNC  : IDEM INLINE, BUT NO COLO PERFORMED AT
312                 *         BEGIN ( ALLOWS EG. XYZ> PROMPTING )
313                 * EXIT  : IDEM INLINE, BUT POSITION C=1 NOT VALID
314                 * EXAMPLE OF USAGE :
315                 *         LXI  H,MSG$     MESSAGE BEFORE INPUT
316                 *         CALL PMSG
317                 *         RST  5          GET CURSOR POSITION
318                 *         DATA :0C        X POS IN REG L
319                 *         INR  L          INCREMENT FOR PROMPT
320                 *         PUSH H          SAVE CURX
321                 *         MVI  A,'>'      BLANK PROMPT
322                 *         CALL INLINX
323                 *         POP  B          GET CURX IN REG C
324                 *         JC   BREAK      EXIT IF BREAK PRESSED
325                 *         REM : READY TO USE ENCINP OR LOOKC
326                 *
327                 ENCINP  EQU   :DDE0     GET CHARACTER TO ENCODE
328                 * ENTRY : C : POSITION ON CURRENT LINE OF FIRST
329                 *             CHAR TO INPUT  (MAX 219)
330                 *         EFSW : SOURCE FOR CHARACTERS
331                 *         SOURCE : CURRENT SCREENLINE OR
332                 *                  STRING OR EDITBUFFER
333                 * FUNC  : CHECK EFSW FOR SOURCE
334                 *          = 0 : FETCH CHARACTER FROM CURRENT LINE
335                 *                (=CURSOR LINE) OF SCREEN, A CAR RET
336                 *                CHAR IS RETURNED IF AT OR PAST
337                 *                CURSOR
338                 *          = 1 : FETCH CHARACTER FROM STRING
339                 *          = 2 : FETCH CHARACTER FROM EDITBUFFER
340                 *         NOTE : FOR 1 AND 2 EFEPT (:132) AND EFECT
341                 *                (:134) MUST BE SET UP CORRECTLY,
342                 *                THIS WILL BE DESCRIBED ANOTHER TIME
343                 * EXIT  : CHARACTER IS RETURNED IN REG A
344                 *         A CAR RET CHAR FLAGS END OF SOURCELINE,
345                 *         POSITION IN REG C IS NOT CHANGED, THIS
346                 *         MUST BE DONE BY THE USER (REASON : SAME
347                 *         POSITION MAY BE FETCHED MORE THAN ONCE)
348                 EFSW    EQU   :135      ENCODING INPUT SWITCH
349                 *
350                 IGNB    EQU   :DDD2     GET CHAR, IGNORE BLANKS
351                 * ENTRY : IDEM ENCINP
352                 * FUNC  : IDEM ENCINP, BUT SPACE AND TAB (ASCI :9)
353                 *         ARE IGNORED (NOT RETURNED)
354                 * EXIT  : IDEM ENCINP, BUT POSITION IN REG C MAY
355                 *         BE INCREMENTED IF SPACE(S) OR TAB(S)
356                 *         FOUND
357                 *
358                 LOOKC   EQU   :CA34     LOOKUP INPUT IN A TABLE
359                 * ENTRY : C  : POSITION ON CURRENT LINE
360                 *         HL : STARTADDRESS TABLE
361                 *         E  : (NUMBER OF CODEBYTES IN TABLE) - 1
362                 * FUNC  : FETCH CHARACTERS VIA EFSW, STARTING AT
363                 *         POSITION IN C
364                 *         LEADING BLANKS OR TABS ARE IGNORED
365                 *         MATCH A SEQUENCE OF SOURCE CHARACTERS
```

```
                    *          WITH KEYWORDS IN THE TABLE
                    *          THE WHOLE TABLE IS SCANNED, RESULTBYTES
                    *          AFTER THE KEYWORDS ARE SKIPPED
                    * EXIT   : CY = 1 : MATCHING KEYWORD FOUND
                    *            HL : POINTS TO BYTE AFTER KEYWORD
                    *                 = FIRST RESULTBYTE
                    *             C : POSITION AFTER LAST CHAR OF
                    *                 MATCHED SEQUENCE
                    *             A : LAST CHAR FETCHED FROM SCREEN
                    *          CY = 0 : NO MATCHING KEYWORD FOUND
                    *             C : MAY BE CHANGED IF SPACE OR TABS
                    *                 SKIPPED, IN EACH CASE POSITION
                    *                 OF FIRST NON BLANK/TAB CHAR
                    *            HL : POINTS AFTER END OF TABLE
                    *             A : ZERO
                    *
                    * TABLE FORMAT :
                    * A NUMBER OF RECORDS, EACH IN FORMAT :
                    *      1 BYTE   = LENGTH OF KEYWORD
                    *      N BYTES  = KEYWORD IN ASCI
                    *      M BYTES  = RESULTBYTES, VECTOR TO ROUTINE
                    *                 OR A CODEVALUE
                    *      NOTE : M MUST BE SAME FOR ALL RECORDS (M-1 IS
                    *             A INPUT (REG C) TO THE LOOKC FUNCTION)
                    * END OF TABLE : DUMMY RECORD WITH LENGTH OF
                    *                KEYWORD = 0
                    *
                    * EXAMPLE : (SET REG E TO 2)
                    *     DATA  3
                    *     ASC   'DNA'
                    *     DATA  :B0        CONTROLVALUE
                    *     DBL   :1100      POINTER TO ROUTINE
                    *     DATA  7
                    *     ASC   'NOTHING'
                    *     DATA  :FF
                    *     DBL   :FFFF
                    *     DATA  0          FLAGS END OF TABLE
                    *
                    * REF : BASIC COMMAND TABLE STARTING AT ROM ADDRESS
                    *       :CBBF, RESULTBYTES : 1 BYTE COMMANDCODE,
                    *       2 BYTES VECTOR TO COMPILE ROUTINE
                    *
0000                      END

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

ADDA   DE30   ADDMI  DE39   ALFA   DE02   ALFNUM DE09
COLO   DD55   COMPDE DE14   CRLF   DD5E   DELAY  DE41
EFSW   0135   ENCINP DDE0   FILL   DE7C   GETC   D6BE
GETFRC D6BB   IGNB   DDD2   INLINE DD1A   INLINX DD1F
INSW   0296   KNSCAN 02B9   LOOKC  CA34   MOVE   DE4F
MULA   DE8F   NEGHL  DE26   NUMER  DE0D   OTSW   0131
OUTC   DD60   OUTCPR D695   PMSG   DAD4   PMSGR  DAFF
POPRET C14D   PSTR   DB32   PSTRM  DB44   RSTART C80C
SUBDE  DE1A   SUBDED D790
```

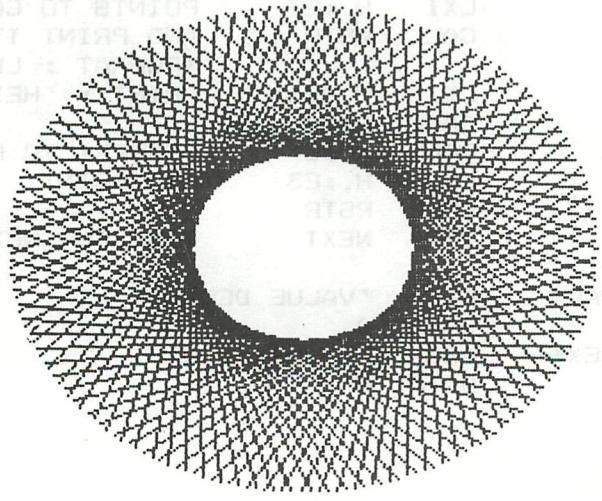
math-pack & rom-interface

```
002                     ********************************************************
003                     *                                                      *
004                     *  THIS DEMO SHOWS USE OF INFO DESCRIBED IN :          *
005                     *     DAI MATH-PACK (REF TO DAINAMIC 82/1,PAGE 7)      *
006                     *     DAI ROM INTERFACE (82/1,PAGE 14 AND 82/2)        *
007                     *                                                      *
008                     *  WITH COMMANDS ENTERED AFTER A SPECIAL PROMPT        *
009                     *  IT IS POSSIBLE TO PERFORM SOME FUNCTIONS ON A       *
010                     *  4 BYTE FLOATING POINT VALUE.                        *
011                     *  AFTER EACH FUNCTION THE VALUE IS PRINTED IN         *
012                     *  DECIMAL (FLOATING POINT) AND HEXADECIMAL FORMAT.    *
013                     *  THE VALUE IS STORED IN A RAM AREA OF 4 BYTES        *
014                     *  STARTING AT LABEL 'NUMBER'.                         *
015                     *                                                      *
016                     ********************************************************
017                     *
018                     * ORG : P. JONGEN AND F. DE RAEDT
019                     *
020                     * INTERFACE WITH DAI ROM (REF DAI INTERFACE) :
021                     POPRET  EQU   :C14D     POP REGS, RET
022                     LOOKC   EQU   :CA34     LOOKUP CMD IN TABLE
023                     PMSG    EQU   :DAD4     PRINT A MESSAGE
024                     PSTR    EQU   :DB32     PRINT A STRING
025                     INLINE  EQU   :DD1A     GET CHARACTERS UNTIL CR
026                     COLO    EQU   :DD55     OUTPUT CR IF CURX<>0
027                     CRLF    EQU   :DD5E     OUTPUT CR
028                     IGNB    EQU   :DDD2     GET CHAR, IGN BLANKS
029                     NUMER   EQU   :DE0D     CHECK 0..9
030                     *
031                     * FOLLOWING ENTRYPOINTS FOR NUMBER INPUT/OUTPUT
032                     * ARE TAKEN FROM THE DAI MANUAL PAGE 126 :
033                     XFCB    EQU   :C01E     INPUT FPT
034                     XFBC    EQU   :C021     CONVERT FPT
035                     XHBC    EQU   :C02D     CONV HEX
036                     XPRTY   EQU   :C030     PRETTY UP
037                     *
038                     * DECLARE LITERALS :
039                     FF      EQU   :0C       FORMFEED
040                     CR      EQU   :0D       CAR RET
041                     PRCH    EQU   :40       PROMPTCHARACTER
042                     *
043                     *
044                     * SET BEGIN OF PROGRAMCODE :
045                             ORG   :300      THIS ALSO ENTRY OF PROGRAM
046                                             UT>Z3>G300 OR *CALLM #300
047                     *
048                     * PART 1 : COMMAND HANDLER
049                     *
050 0300 F5             ENTRY   PUSH  PSW       SAVE ALL REGISTERS
051 0301 C5                     PUSH  B         ON STACK
052 0302 D5                     PUSH  D
053 0303 E5                     PUSH  H
054 0304 214203                 LXI   H,TITL$   PRINT TITLE
055 0307 CDD4DA                 CALL  PMSG
056 030A 212D04                 LXI   H,HELP$   PRINT MENU
057 030D CDD4DA                 CALL  PMSG
058 0310 C39603                 JMP   ZER01     INIT NUMBER TO 0
059                     * CODE ENTERTED WITH PREVIOUS JUMP TRANSFERS CONTROL
060                     * TO THE FOLLOWING LINE
061 0313 3E40           NEXT    MVI   A,PRCH
062 0315 CD1ADD                 CALL  INLINE    GET CHAR UNTIL CAR RET
063 0318 DA1303                 JC    NEXT      CANCEL LINE IF BREAK
064                     * INLINE RETURNS WITH C = 1 : POINTS TO FIRST
065                     * INPUTTED CHARACTER ON CURRENT LINE
```

```
066 031B CDD2DD              CALL  IGNB       IF LINE HAS ONLY BLANKS
067 031E FE0D                CPI   CR         THEN NO VALID INPUT
068 0320 CA1303              JEQ   NEXT       AND START AGAIN
069                  * NOW LOOK FOR A VALID COMMAND
070 0323 216603              LXI   H,CMDTBL   BEGIN OF COMMANDTABLE
071 0326 1E01                MVI   E,CMDLNG-1 LENGTH RESULT BYTES TABLE
072 0328 CD34CA              CALL  LOOKC      LOOKUP CMD IN TABLE
073 032B D23603              JNC   SYNERR     NO VALID COMMAND FOUND
074 032E 5E                  MOV   E,M        GET ROUTINE ADDR
075 032F 23                  INX   H          FROM TABLE
076 0330 56                  MOV   D,M
077 0331 EB                  XCHG             AND TRANSFER CONTROL
078 0332 E9                  PCHL             TO THE ROUTINE
079                  * THIS IS A JUMP, NOT A CALL ; IF NO ERRORS OCCUR
080                  * THE ROUTINE SHOULD JUMP BACK TO 'NEXT'
081                  *
082                  EXIT                     CMD : EXIT TO BASIC OR UTIL
083 0333 C34DC1              JMP   POPRET     RESTORE ALL REGISTERS
084                  *
085                  SYNERR                   ROUTINE HAS FOUND INVALID
086                                           SYNTAX IN COMMAND LINE
087 0336 CD55DD              CALL  COLO
088 0339 2123DC              LXI   H,:DC23    MSG 'SYNTAX ERROR'
089 033C CDD4DA              CALL  PMSG
090 033F C31303              JMP   NEXT
091                  *
092 0342 0C          TITL$   DATA  FF
093 0343 44454D              ASC   'DEMO ROM INTERFACE AND '
094 035A 4D4154              ASC   'MATH PACK'
095 0363 0D0D00              DATA  CR,CR,0
096                  *
097                  * TABLE FOR COMMAND LOOKUP :
098                  CMDLNG  EQU   2          LENGTH OF RESULT FIELD
099 0366 04          CMDTBL  DATA  4          LENGTH OF COMMAND$
100 0367 5A45524F            ASC   'ZERO'     COMMAND$
101 036B 8E03                DBL   ZERO       RESULT FIELD
102 036D 03                  DATA  3          NEXT COMMAND ...
103 036E 414444              ASC   'ADD'
104 0371 9F03                DBL   ADD
105 0373 04                  DATA  4
106 0374 45584954            ASC   'EXIT'
107 0378 3303                DBL   EXIT
108 037A 03                  DATA  3
109 037B 4E4547              ASC   'NEG'
110 037E BB03                DBL   NEG
111 0380 03                  DATA  3
112 0381 535152              ASC   'SQR'
113 0384 CD03                DBL   SQR
114 0386 04                  DATA  4
115 0387 48454C50            ASC   'HELP'
116 038B 2404                DBL   HELP
117 038D 00                  DATA  0          FLAG END OF TABLE
118                  *
119                  ZERO                     CMD : ZERO OUT NUMBER
120 038E CDD2DD              CALL  IGNB       TEST OTHER CHARACTERS ON
121 0391 FE0D                CPI   CR         SAME LINE
122 0393 C23603              JNE   SYNERR     ERROR IF TRUE
123 0396 AF          ZERO1   ZAR              MAKE A 0 FPT VALUE
124 0397 47                  MOV   B,A        IN REGS A,B,C,D
125 0398 4F                  MOV   C,A
126 0399 57                  MOV   D,A
127 039A E7                  RST   4          SEND IT TO FPT ACC
128 039B 12                  DATA  :12
129 039C C3DF03              JMP   DONE       DUMP RESULT
```

```
130                     *
131                     ADD                             CMD : ADD INPUTTED VALUE
132 039F CDD2DD                 CALL   IGNB             IF CHARACTER AFTER CMD
133 03A2 CDODDE                 CALL   NUMER            NOT NUMBER THEN
134 03A5 D23603                 JNC    SYNERR           SYNTAX ERROR
135 03A8 CD1EC0                 CALL   XFCB             INPUT FPT NUMBER TO ACC
136 03AB CDD2DD                 CALL   IGNB             CHECK FOR ILLEGAL CHARS
137 03AE FE0D                   CPI    CR               AFTER NUMBER
138 03B0 C23603                 JNE    SYNERR
139 03B3 218E05                 LXI    H,NUMBER         POINTS TO NUMBER
140 03B6 E7                     RST    4                ADD NUMBER TO
141 03B7 00                     DATA   :00              FPT ACC
142 03B8 C3DF03                 JMP    DONE             DUMP RESULT
143                     *
144                     NEG                             CMD : NEGATE NUMBER
145 03BB CDD2DD                 CALL   IGNB             ERROR IF CHARS AFTER
146 03BE FE0D                   CPI    CR               COMMAND
147 03C0 C23603                 JNE    SYNERR
148 03C3 218E05                 LXI    H,NUMBER         POINTS TO NUMBER
149 03C6 E7                     RST    4
150 03C7 0C                     DATA   :0C              FETCH NUMBER IN FPT ACC
151 03C8 E7                     RST    4
152 03C9 1B                     DATA   :1B              CHANGE SIGN OF FPT ACC
153 03CA C3DF03                 JMP    DONE             DUMP RESULT
154                     *
155                     SQR                             CMD : SQR OF NUMBER
156 03CD CDD2DD                 CALL   IGNB             ERROR IF CHARS AFTER
157 03D0 FE0D                   CPI    CR               COMMAND
158 03D2 C23603                 JNE    SYNERR
159 03D5 218E05                 LXI    H,NUMBER         POINTS TO NUMBER
160 03D8 E7                     RST    4
161 03D9 0C                     DATA   :0C              FETCH NUMBER IN FPT ACC
162 03DA E7                     RST    4
163 03DB 33                     DATA   :33              TAKE SQR OF FPT ACC
164 03DC C3DF03                 JMP    DONE             DUMP RESULT
165                     *
166                     * DUMP RESULT :
167                     *     PUT RESULT IN NUMBER
168                     *     DISPLAY RESULT IN  FPT AND HEX FORMAT
169                     *     GOTO NEXT FOR NEW COMMAND
170                     *
171 03DF 218E05         DONE    LXI    H,NUMBER         POINTS TO NUMBER
172 03E2 E7                     RST    4
173 03E3 0F                     DATA   :0F              STORE FPT ACC IN NUMBER
174 03E4 CD5EDD                 CALL   CRLF             NEW LINE
175 03E7 210D04                 LXI    H,VAL$           PRINT 'DEC :'
176 03EA CDD4DA                 CALL   PMSG
177 03ED CD21C0                 CALL   XFBC             CONVERT FPT FOR OUTPUT
178 03F0 3E06                   MVI    A,6              PRECISION
179 03F2 CD30C0                 CALL   XPRTY            PRETTY UP
180 03F5 21E300                 LXI    H,:E3            POINTS TO CONVERTED VALUE
181 03F8 CD32DB                 CALL   PSTR             AND PRINT IT
182                                                     (FORMAT : LENGTH+CHARS)
183 03FB 211904                 LXI    H,HEX$           PRINT : 'HEX : #'
184 03FE CDD4DA                 CALL   PMSG
185 0401 CD2DC0                 CALL   XHBC             CONVERT TO HEX
186 0404 21E300                 LXI    H,:E3            AND PRINT
187 0407 CD32DB                 CALL   PSTR
188 040A C31303                 JMP    NEXT             TIME FOR NEXT COMMAND
189                     *
190 040D 56414C         VAL$    ASC    'VALUE DEC :'
191 0418 00                     DATA   0
192 0419 202020         HEX$    ASC    '    HEX : #'
193 0423 00                     DATA   0
```

```
194                     *
195                     HELP                    CMD : PRINT MENU
196 0424 212D04                 LXI   H,HELP$
197 0427 CDD4DA                 CALL  PMSG
198 042A C31303                 JMP   NEXT
199                     *
200 042D 0D             HELP$   DATA  CR
201 042E 544849                 ASC   'THIS DEMO ALLOWS SOME MATHEMATIC'
202 044E 204655                 ASC   ' FUNCTIONS ON A VALUE SAVED'
203 0469 0D                     DATA  CR
204 046A 494E20                 ASC   'IN THE LOCATION NUMBER.'
205 0481 0D                     DATA  CR
206 0482 414654                 ASC   'AFTER EACH FUNCTION THE VALUE '
207 04A0 495320                 ASC   'IS DISPLAYED IN FPT AND HEX '
208 04BC 0D                     DATA  CR
209 04BD 464F52                 ASC   'FORMAT.'
210 04C4 0D                     DATA  CR
211 04C5 56414C                 ASC   'VALID COMMANDS ARE :'
212 04D9 0D                     DATA  CR
213 04DA 20205A                 ASC   '  ZERO : RESET VALUE TO 0'
214 04F3 0D                     DATA  CR
215 04F4 202041                 ASC   '  ADD XXX.XX : ADD TO VALUE'
216 050F 202850                 ASC   ' (POSITIVE AND START WITH 0..9)'
217 052E 0D                     DATA  CR
218 052F 20204E                 ASC   '  NEG  : CHANGE SIGN'
219 0543 0D                     DATA  CR
220 0544 202053                 ASC   '  SQR  : TAKE SQR'
221 0555 0D                     DATA  CR
222 0556 202048                 ASC   '  HELP : SHOW COMMANDS'
223 056C 0D                     DATA  CR
224 056D 202045                 ASC   '  EXIT : RETURN TO UT OR BASIC'
225 058B 0D0D00                 DATA  CR,CR,0
226                     *
227                     *
228 058E                NUMBER  RES   4,0       AREA FOR FPT NUMBER
229                     *
230 0592                        END

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

ADD     039F    CMDLNG  0002    CMDTBL  0366    COLO    DD55
CR      000D    CRLF    DD5E    DONE    03DF    ENTRY   0300
EXIT    0333    FF      000C    HELP    0424    HELP$   042D
HEX$    0419    IGNB    DDD2    INLINE  DD1A    LOOKC   CA34
NEG     03BB    NEXT    0313    NUMBER  058E    NUMER   DE0D
PMSG    DAD4    POPRET  C14D    PRCH    0040    PSTR    DB32
SQR     03CD    SYNERR  0336    TITL$   0342    VAL$    040D
XFBC    C021    XFCB    C01E    XHBC    C02D    XPRTY   C030
ZERO    038E    ZERO1   0396
```



math-pack & rom-interface

```
    ]#P.
0300 F5 C5 D5 E5 21 42 03 CD D4 DA 21 2D 04 CD D4 DA
0310 C3 96 03 3E 40 CD 1A DD DA 13 03 CD D2 DD FE 0D
0320 CA 13 03 21 66 03 1E 01 CD 34 CA D2 36 03 5E 23
0330 56 EB E9 C3 4D C1 CD 55 DD 21 23 DC CD D4 DA C3
0340 13 03 0C 44 45 4D 4F 20 52 4F 4D 20 49 4E 54 45
0350 52 46 41 43 45 20 41 4E 44 20 4D 41 54 48 20 50
0360 41 43 4B 0D 0D 00 04 5A 45 52 4F 8E 03 03 41 44
0370 44 9F 03 04 45 58 49 54 33 03 03 4E 45 47 BB 03

0380 03 53 51 52 CD 03 04 48 45 4C 50 24 04 00 CD D2
0390 DD FE 0D C2 36 03 AF 47 4F 57 E7 12 C3 DF 03 CD
03A0 D2 DD CD 0D DE D2 36 03 CD 1E C0 CD D2 DD FE 0D
03B0 C2 36 03 21 8E 05 E7 00 C3 DF 03 CD D2 DD FE 0D
03C0 C2 36 03 21 8E 05 E7 0C E7 1B C3 DF 03 CD D2 DD
03D0 FE 0D C2 36 03 21 8E 05 E7 0C E7 33 C3 DF 03 21
03E0 8E 05 E7 0F CD 5E DD 21 0D 04 CD D4 DA CD 21 C0
03F0 3E 06 CD 30 C0 21 E3 00 CD 32 DB 21 19 04

03FE CD D4
0400 DA CD 2D C0 21 E3 00 CD 32 DB C3 13 03 56 41 4C
0410 55 45 20 44 45 43 20 3A 00 20 20 20 48 45 58 20
0420 3A 20 23 00 21 2D 04 CD D4 DA C3 13 03 0D 54 48
0430 49 53 20 44 45 4D 4F 20 41 4C 4C 4F 57 53 20 53
0440 4F 4D 45 20 4D 41 54 48 45 4D 41 54 49 43 20 46
0450 55 4E 43 54 49 4F 4E 53 20 4F 4E 20 41 20 56 41
0460 4C 55 45 20 53 41 56 45 44 0D

046A 49 4E 20 54 48 45
0470 20 4C 4F 43 41 54 49 4F 4E 20 4E 55 4D 42 45 52
0480 2E 0D 41 46 54 45 52 20 45 41 43 48 20 46 55 4E
0490 43 54 49 4F 4E 20 54 48 45 20 56 41 4C 55 45 20
04A0 49 53 20 44 49 53 50 4C 41 59 45 44 20 49 4E 20
04B0 46 50 54 20 41 4E 44 20 48 45 58 20 0D 46 4F 52
04C0 4D 41 54 2E 0D 56 41 4C 49 44 20 43 4F 4D 4D 41
04D0 4E 44 53 20 41 52 45 20 3A 0D

04DA 20 20 5A 45 52 4F
04E0 20 3A 20 52 45 53 45 54 20 56 41 4C 55 45 20 54
04F0 4F 20 30 0D 20 20 41 44 44 20 58 58 58 2E 58 58
0500 20 3A 20 41 44 44 20 54 4F 20 56 41 4C 55 45 20
0510 28 50 4F 53 49 54 49 56 45 20 41 4E 44 20 53 54
0520 41 52 54 20 57 49 54 48 20 30 2E 2E 39 29 0D 20
0530 20 4E 45 47 20 20 3A 20 43 48 41 4E 47 45 20 53
0540 49 47 4E 0D 20 20 53 51 52 20 20 3A 20 54 41 4B
0550 45 20 53 51 52 0D

0556 20 20 48 45 4C 50 20 3A 20 53
0560 48 4F 57 20 43 4F 4D 4D 41 4E 44 53 0D 20 20 45
0570 58 49 54 20 3A 20 52 45 54 55 52 4E 20 54 4F 20
0580 55 54 20 4F 52 20 42 41 53 49 43 0D 0D 00

058E 00 00
0590 00 00
```

TAPE 1980-1981

```
0 GETALCONVERSIES DAInamic 0-1 1980
0 4 OP EEN RIJ DAInamic 0-1 1980
0 RAKETSPEL DAInamic 0-1 1980
0 PROPELLER DAInamic 0-1 1980
0 4 COLOR DEMO DAInamic 0-1 1980
0 CITROEN B14 DAInamic 0-1 1980
0 BLOXY COLOR DEMO DAInamic 01 1980
0 MUSIC DAInamic 2 1980
 0 REAL-TIME KLOK DAInamic 2 1980
 0 DIGITAAL-ANALOOG KLOK DAInamic 2 1980
 0 FORMAAT LISTING DAInamic 2 1980
 0 LIST-PRINT DAInamic 2 1980
 0 ZAKDOEK LEGGEN DAInamic 2 1980
 0 BLUE MOON DAInamic 3 1981
 0 TUBELAR BELLS DAInamic 3 1981
 0 RADAR-SIMULATIE DAInamic 3 1981
 0 VIDEORAM IN MODE 0 DAInamic 3 1981
 0 VIDEO TEXT DAInamic 3 1981
 0 VARIABELEN ATLAS DAInamic 1981
 0 BARRICADE DAInamic 3 1981
 0 GEKLEURDE ACHTERGRONDEN IN MODE 0 DAInamic 3
 0 KANTEN KLEEDJE DAInamic 4 1981
 0 ALLE TEKENS DAInamic 4 1981
 0 PRIEMGETALLEN 110 S DAInamic 4 1981
 0 ROTERENDE ELLIPSEN DAInamic 4 1981
 0 FILM TITLE SIMULATION DAInamic 4 1981
 0 PADDELEN MET FGT DAInamic 5 1981
 0 TIME OUT DAInamic 5 1981
 0 RESPONTIE TIJD DAInamic 5 1981
 0 REAL TIME CLOCK DAInamic 5 1981
 0 RANDOM GENERATOR TEST DAInamic 5 1981
 0 FASING KEYBORD MET BOOTSTRAP LOADER DAInamic 5
 1FASING KEYBORD
 0FASING KEYBORD
 0 WILHELMUS DAInamic 6 1981
 0 TORENS VAN HANOI DAInamic 6 1981
 0 VIJFDE GRAADS POLYNOMEN DAInamic 6 1981
 0 8080 SIMULATOR BOOTSTRAP LOADER DAInamic 7 1981
 1ML 8080 SIMULATOR
 0BASIC 8080 SIMULATOR
 0 BITRIJGENERATOR DAInamic 7 1981
 0 16 KLEUREN IN 4 KLEUREN MODE DAInamic 7 1981
 0 TALK EDITOR DAInamic 7 1981
 0 CIJFERTABEL DAInamic 7 1981
 0 CARPENTERS MYSTERY DAInamic 8 1981
 0 BASICODE MET BOOTSTRAPLOADER DAInamic 8 1981
 1UTILITY BASICODE 2EE 527
 0DOCUMENTATIE BASICODE
  0 DIGITALE CLOCK DAInamic 8 1981
  0 MEMORY-MAP MODE 4 DAInamic 8 1981
  0 THE HAT DAInamic 8 1981
  0 POKE ACTION DAInamic 8 1981
  0 LETTER DAInamic 8 1981
  0 SIENTJE DAInamic 8 1981
  0 DEMO DAInamic 8 1981
  0 TELEX IN BASIC DAInamic 8 1981
  0 KRAAN DAInamic 8 1981
```

the story of anna list

Little Anna List was e real character.DAInamic but a bit fixed in her ideas,she executed her daily routine without ever understanding its function or ever passing a remark.She lived in a software house,a multi-level structure comfortable but basic. At the back was a data field where she kept a pet ram tied up with a bit string.
One day, an operator in Fort Ranfow noticed that Anna had not returned any data. He had nothing in his programma for that day, so he decided to give her a call. He took the DCE bus to Anna's address and was surprised to find the house surrounded by an array of bins fixed to the wall. The only entry point was an access port on the left.
He entered and found Anna sitting at a symbol table just about to take a byte out of an apple.

He addresses Anna, "What are all those bins?"

"I like apples",she declared," and those are the core dumps."

"I'm from I.B.M.(Internal Buffer Maintenace)" the operator commented,and continued "Why haven't you computed your data?"

Anna displayed anguish."I can't read the code ",she sobbed,

"I can only read ASCII and the formats are all in EBCDIC."

The operator's response was unprintable,he cursed and cursed (he was a real cursor)."If data returned then okay else I shall compile a report" he ended.
After this exit, anna decided to call her friend Pascal Bull. She called his linenumber and he told her to go to his address. Anna repeated what the I.B.M. man had said.

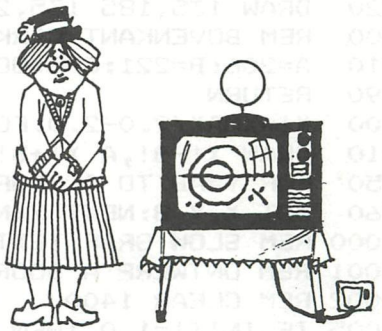
"Don't worry," Pascal said "no need to poke around;I'll get my brother Franco to take a peek at the problem."

The next day, Pascal gave Anna a conversion routine written by Franco Bull for translating EBCDIC into ASCII.Anna has compiled her data returns happily ever after.

Barry Hunt

testbeeld

```
100   REM TESTBEELD V1.0
110   REM ONTWERP A DOORNENBAL.
120   REM TEKENEN DUURT 2 MIN.
200   CLEAR 300+1400
210   GOSUB 1000:GOSUB 2000:GOSUB 3000:GOSUB 5000
220   REM GOSUB 3000 WORDT GOSUB 4000 VOOR LEGE CIRKEL
300   A$="TESTBEELD":X!=130.0:Y!=190.0:DEV!=0.0:C!=0.0
310   F!=1.0:GOSUB 40000
400   ENVELOPE 0 15,10;0,10;:SOUND 0 0 15 0 2000
999   GOTO 999
1000  REM BEELD OPMAAK
1005  COLORG 8 15 15 15
1010  MODE 5:POKE #BFFF,#B3:POKE #BFFB,#B2:POKE #BFF7,#B3:POKE #BFF3,#B3
1015  POKE #2A5,#AF:POKE #2A6,#5A
1020  FOR X=#BFEF-256*90 TO #BFEF-288*90 STEP -2
1030  POKE X,#FF:POKE X-1,#80:NEXT X
1040  FOR X=#BFEF-256*90 TO #BFEF-288*90 STEP -90
1050  POKE X,#A0:POKE X-1,#40:NEXT X
1060  RETURN
1799  DATA BLOK
2000  REM WITTE BLOKKEN OM BEELD
2010  DIM BLOK(43.0):REM CLEAR 300
2015  RESTORE
2016  READ A$:IF A$<>"BLOK" THEN 2016
2020  FOR X=0 TO 43:READ BLOK(X):NEXT X
2110  FOR LIJN=0 TO 7:GOSUB 2500:NEXT LIJN
2120  FOR LIJN=282 TO 288:GOSUB 2500:NEXT LIJN
2130  FOR LIJN=8 TO 280:A=#6F:B=#8F:C=#C0:D=#80
2140  IF ((LIJN-8)/21) MOD 2=1 THEN A=#90:B=#F8:C=#C0:D=#8F
2150  GOSUB 2600:NEXT LIJN:GOSUB 2600:RETURN
2500  Y=#BFEF-LIJN*90-2:FOR X=0 TO 43
2510  POKE Y-2*X,BLOK(X) IXOR #FF:POKE Y-2*X-1,#F:NEXT X
2520  RETURN
2600  Y=#BFEF-LIJN*90-2:POKE Y,A:POKE Y-1,B
2610  POKE Y-86,C:POKE Y-87,D:RETURN
2800  DATA #80,0,1,#FF,#FF,#F0,0,1,#FF,#FF,#F0,0
2801  DATA 1,#FF,#FF,#F0,0,1,#FF,#FF,#F0,0
2802  DATA 1,#FF,#FF,#F0,0,1,#FF,#FF,#F0,0
2803  DATA 1,#FF,#FF,#F0,0,1,#FF,#FF,#F0,0
2804  DATA 0,#3F
3000  REM RUITEN TESTBEELD
3010  FOR LIJN=8 TO 281:Y=#BFEF-LIJN*90-6
3020  FOR A=0 TO 7:POKE Y-A*10,#FE:POKE Y-A*10-1,#8F
3030  POKE Y-A*10-6,#EF:POKE Y-A*10-7,#8F:NEXT A
3040  POKE Y-A*10,#FE:POKE Y-A*10-1,#8F:NEXT LIJN
3050  FOR LIJN=8 TO 281 STEP 21:Y=#BFEF-90*LIJN-2
3060  FOR A=0 TO 43:POKE Y-A*2,#FF:POKE Y-A*2-1,#F8:NEXT A:NEXT LIJN
3070  RETURN
4000  REM CIRKEL
4010  X!=XMAX/2.0-2.0:D!=0.0:C!=0.0
4020  FOR A=0 TO 221:B!=20.0*SQR(221.0*A-A*A)/21.0
4030  IF ABS(B!-C!)<1.0 THEN NEXT A
4040  DRAW X!-C!,D! X!-B!,A 15:DRAW X!+C!,D! X!+B!,A 15
4050  C!=B!:D!=A:NEXT A:RETURN
5000  REM GEVULDE CIRKEL
5100  REM GEEL/ROOD BALK
5110  A=0:B=16:C=14:GOSUB 5900:FILL 155,0 175,16 3
5200  REM WIT REFLECTIE BALK
5210  A=17:B=37:C=15:GOSUB 5900:FILL 115,17 215,37 0
5220  DRAW 135,17 135,37 15
5300  REM GRIJS GRADATIES
5310  AB=38:AC=58:GOSUB 5320:AB=164:AC=184:GOSUB 5320:GOTO 5400
5320  C=0:FOR X=AB TO AC:A=X:B=X:GOSUB 5900
5330  DRAW 185,X X!+B!,X 15:NEXT X
5340  A=105:B=145:GOSUB 5950:FILL 145,AB 185,AC 8
5350  A=185:B=223.0:GOSUB 5950:RETURN
```



testbeeld

```
5400  REM FREQUENTIE BALKEN
5410  A=59:B=100:C=0:GOSUB 5900
5420  FOR X=85 TO 105 STEP 10:FILL X,59 X+4,100 15:NEXT X
5430  FOR X=117 TO 141 STEP 6:FILL X,59 X+2,100 15:NEXT X
5440  FOR X=149 TO 177 STEP 4:FILL X,59 X+1,100 15:NEXT X
5450  FOR X=181 TO 208 STEP 3:FILL X,59 X+1,100 15:NEXT X
5460  FOR X=213 TO 245 STEP 2:DRAW X,59 X,100 15:NEXT X
5500  REM KLEUREN BALKEN
5510  C=14:FOR X=122 TO 163:A=X:B=X:GOSUB 5900
5520  DRAW 185,X X!+B!,X 1:NEXT X
5530  FILL 95,122 130,163 13:FILL 130,122 165,163 12
5540  FILL 165,122 200,163 11:FILL 200,122 235,163 3
5600  REM HET MIDDEN VAN DE CIRKEL
5610  FILL 155,79 175,142 0:A=101:B=111:C=0:GOSUB 5900
5620  DRAW X!-B!,111 X!+B!,111 15:B=121:GOSUB 5900:DRAW 165,79 165,142 15
5630  FOR X=75 TO 275 STEP 20:DRAW X,101 X,121 15:NEXT X
5700  REM ZWART REFLECTIE BALK
5710  A=185:B=205:C=0:GOSUB 5900:FILL 115,185 215,205 15
5720  DRAW 135,185 135,205 0
5800  REM BOVENKANT CIRKEL & ROOD/GEEL BALK
5810  A=206:B=221:C=3:GOSUB 5900:FILL 155,206 175,221 14
5890  RETURN
5900  X!=XMAX/2.0-2.0:FOR A=A TO B:B!=20.0*SQR(221.0*A-A*A)/21.0
5910  DRAW X!-B!,A X!+B!,A C:NEXT A:RETURN
5950  FOR Y=AB TO AC:FOR X=A+Y MOD 2 TO B+Y MOD 2 STEP 2
5960  DOT X,Y 8:NEXT X:NEXT Y:RETURN
40000 REM SLOW GRAF TEXT SGT
40001 REM ONTWERP A DOORNENBAL
40002 REM CLEAR 1400
40005 IF INIT!=1.0 THEN 40040:DIM CAR$(90.0)
40010 RESTORE
40012 READ B$:IF B$<>"SGT" THEN 40012
40015 FOR Z!=32.0 TO 91.0:READ B$
40025 IF B$="STOP" THEN INIT!=1.0:GOTO 40040
40030 READ CAR$(Z!):NEXT:INIT!=1.0
40040 CO!=COS(DEV!)*F!:SI!=SIN(DEV!)*F!:X!=X!+0.5:Y!=Y!+0.5
40042 FOR M!=0.0 TO LEN(A$)-1.0:GR$=CAR$(ASC(MID$(A$,M!,1)))
40060 FOR N!=0.0 TO LEN(GR$)-1.0 STEP 4.0
40065 IF MID$(GR$,N!,1)="/" THEN X!=X!+CO!*8.0:Y!=Y!+SI!*8.0:NEXT M!:X!=X!-0.5:Y
!=Y!-0.5:RETURN
40080 JC1!=VAL(MID$(GR$,N!,1)):JC2!=VAL(MID$(GR$,N!+1,1))
40082 JC3!=VAL(MID$(GR$,N!+2,1)):JC4!=VAL(MID$(GR$,N!+3,1))
40090 DRAW X!+JC1!*CO!-JC2!*SI!,Y!+JC2!*CO!+JC1!*SI! X!+JC3!*CO!-JC4!*SI!,Y!+JC4
!*CO!+JC3!*SI! C!
40095 NEXT N!
40999 DATA SGT
41000 DATA BLANCO,/,UITROEP!,31313337/,QOUTES,25274547/,#
41001 DATA 1353155521274147/,$,12424253244415262656313 7/
41002 DATA %,172716261256415142 52/,&,121321315331155116273536/,'
41003 DATA 3537/,(,131513311537/
41004 DATA ),315353555537/,*,125616523137/,+,32361454/,COMMA,21323233/
41005 DATA -,1454/,.,31423241/,/,1256/,0,121621415256274712 56/
41006 DATA 1,214131372637/,2,1151123344445556472716 27/,3
41007 DATA 122121415253345617574453/,4,414713531447/
41008 DATA 5,122121415254154515171757/,6,2141121514445253153737 57/,7
41009 DATA 212223561757/,8,2141244427471213151652535556/
41010 DATA 9,113131535356245415162747/,:,33333535/,;,213232333535/,<
41011 DATA 14471441/
41012 DATA =,13531555/,>,21545427/,?,162727473433313134 56/,APE,/
41013 DATA A,111551551353153737 55/,B,111717471444114152535556/,C
41014 DATA 121627474456214141 52/,D,1117114152561747/
41015 DATA E,1117115114441757/,F,111714441757/,G,121627572151515353 43/,H
41016 DATA 111714545157/
41017 DATA I,214131372747/,J,122121415257/,K,111713572451/,L,11171151/
41018 DATA M,111717355353435575751/,N,111751571652/,O,121627475652214 1/,P
41019 DATA 1117144417475556/
41020 DATA Q,121627475653213133 51/,R,111717475655144424 51/,S
41021 DATA 122121415253244415162747475 6/,T,17573137/
41022 DATA U,111721415157/,V,1317535713313153/,W,111751571133335133 34/,X
41023 DATA 111217165152575612561652/
41024 DATA Y,161756571634345631 34/,Z,175712561151/
41025 DATA STOP
```

BESTE DAI VRIENDEN

Naar aanleiding van enkele reacties op mijn bijdrage in DAI-namic jan/feb is gebleken dat enige uitleg over de getal-conversie routines welkom zou zijn.
Op deze tape is zowel uitleg over het gebruik van de ROM routines bij het omzetten van ASCII input naar de Math-accumulator als de omgekeerde bewerking opgenomen.
Mogelijk vindt u een en ander geschikt voor publikatie in een volgende DAI-namic. Met behulp van deze routines hoeft de assembler programmeur geen uitstapjes meer te maken naar BASIC voor zijn output.

Met vriendelijke groet,

Peter Jongen
Zeemanhof 25
NL 2871JW SCHOONHOVEN
tel. 01823-5170

ASCII-BINARY-ASCII

If you want to use ASCII input for your M.L.-programs the DAI-encoding routines can be verry usefull to you.
Following an explanation how to convert ASCII input to internal binary format. There are 3 calls:

```
CALL :C01E   Floating point input to MATH. ACCUMULATOR
     :C024   Integer input to MATH ACC.
     :C02A   Hex input to MATH ACC.
```

If you call one of these routines they will need the ASCII input characters one at a time. Therefore a CALL will be made to a routine whose address is in RAM loc :D2/:D3.
This is the pointer to the DAI-input-routine for expresion encoding. You should supply a subroutine in your program that can put one (1) ASCII character in the A-reg on call.
Store the address of your routine in :D2 before calling the conversion routine.
Conversion will stop if any non numeric (0-9) character is entered. (exept for first: . E and - in case of FPT)
The conversion-routine will add 1 for every character to the C-register. Register C is available on the call to your input routine.

EXAMPLE:

```
 * INITIATE INPUT POINTER FOR EXPRESION ENCODING.
        LXI     H,INPUT     ADDR OF INPUT SUBR
        SHLD    :D2         STORE IT IN POINTER
        .
 *  CONVERT ASCII INPUT TO MATH.ACC
        MVI     C,0         CLEAR CHAR COUNTER
        CALL    :C01E       INPUT FPT NUMBER TO ACC
        .
           On return the converted integer number
           is in the MATH-ACC.
        .
 * INPUT SUBROUTINE.
INPUT   MOV     A,C         CHAR COUNT
        CPI     10          ALLOW ONLY 10 CHAR.
        JZ      INRET       RETURN WITH ILLEGAL CHAR TO STOP
GETC    CALL    :6DBE       GETC A CHARACTER
        JZ      GETC        WAIT FOR IT
INRET   RET
```

If you already have the ASCII-number in a string in mememory
the input routine could look like:

```
INPUT    PUSH    H
         PUSH    B
         LXI     H,STRING      GET FIRST ADDRES OF STRING
         MVI     B,0           ADD C (OFFSET) TO ..
         DAD     B             HL TO GET NEXT ASCI DIGIT
         MOV     A,M           GET THE DIGIT IN A
         POP     B
         POP     H
         RET                   RETURN WITH IT IN A-REG
STRING   ASC     '12.345000'   SPACE TO STORE INPUT
NEXT     ASC     'S'           ANY NON-NUM. TO STOP CONV.
          .
          .
```

In above program conversion will stop if the 'S' is passed
to the conversion routine.
All register exept C preserved.
Note:
-The conversion routines will not recognize a leading '-' or
 '+'sign. For negative numbers use 'RST 4 DATA :60 or :1B
 For FPT input 'E', '.' and '-' for (neg-)exponent are
 allowed. (e.g. 1.234E-5)
-If you want to return to BASIC after M.L. programs using
 above, be sure to save and restore :D2/D3

DAI-BINARY TO ASCII

Following a description how to use the DAI-ROM-routines
to convert the result in the MATH-ACC to ASCII for output.

On return from these CALL's you will find a string of
characters starting in mememory location :E4.
For both INT and FPT the string will have following format:

```
    +.XXXXXXX or    -.XXXXXXX          X=ASCII numeric
```

As you can see the number is always <1 AND >-1.
The power of 10 to apply is stored at loc :F1 in binary
This string is not nice for output as it is. A second CALL
is provided to pretty it up for printing.

CALL :C021 (XFBC)

Convert FPT-number in MATH-ACC to ASCII-string.

On return:

```
          String in :E4 - - - - - - - :EC
          Format      + . X X X X X X X
```

X=ASCII-numeric.
String is always <1 and >-1 and 7 digits precision.
The power of 10 to apply in :F1.
EXAMPLE:
FPT 100.5 in ACC after conversion will look like:

```
            + . 1 0 0 5 0 0 0      :F1=03
```

CALL :C027 (XIBC)

Convert INT-number in MATH=ACC to ASCII-string.

On return:

String in :E4 and on, same format as above exept
Number of digits is min 3 and max 10.

EXAMPLE:
INT 100 in ACC after conversion will look like:

```
            + . 1 0 0 0 0 U U U    :F1=03
                                     U=unchanged
```

ASCII-BINARY-ASCII

```
      CALL :C030      (XPRTY)
      ======================
      Pretty-up FPT or INT string.

Before CALLing load the A-reg with the number of digits
for precision. (max 7)
If you use this CALL for an INT number, load A-reg with
contence of :F1
ON RETURN:
          A nice printable string is in memory lokation
          :E4 and on. The number of valid digits is in :E3.


      CALL :C02D      (XHBC)
      ======================
      Convert MATH-ACC to ASCII-Hex for output.

ON RETURN:
          String in :E4 - :EB (max 8 digits)
          The number of vallid digits in :E3


EXAMPLE:
          .
          CALL      :C027          CONVERT INTEGER FOR OUTPUT
          LDA       :F1            PRECISION
          CALL      :C030          PRETTY IT UP
          LXI       H,:C4          START OF STRING
          LDA       :C3            NUMBER OF DIGITS
          MOV       B,A
PRLOOP    MOV       A,M            GET CHARACTER
          CALL      :D695          PRINT IT
          INX       H
          DCR       B              MORE CHARACTERS
          JNZ       PRLOOP         LOOP FOR PRINTING

  for FPT numbers in ACC first two lines will be:
          CALL      :C021          CONVERT FPT
          MVI        A,:7          SET PRECSION


GOOD LUCK
Peter Jongen. Zeemanhof 25. 2871JW SCHOONHOVEN. 01823-5170
```

electric energy

```
100   REM ONTLADING ELECTRIC ENERGY +++ SMIT +++
150   MODE 6
160   COLORG 0 0 5 0
170   DRAW 50,50 100,100 21:DRAW 100,100 100,80 21:DRAW 100,80 150,130 21
180   FILL 52,52 48,48 22:FILL 152,132 148,128 22
200   FOR A=1.0 TO RND(4.0)
205   COLORG 0 3 5 0
210   SOUND 1 0 15 0 FREQ(1200.0)
220   SOUND OFF
225   COLORG 0 0 5 0
230   WAIT TIME RND(6.0)
240   NEXT
250   GOTO 200
```

oscilloscope

```
J#L.
PAGE 01

001                     *
002                     * OSCILLOSCOPE
003                     * LE SIGNAL EST ENTRE PAR LE PDL(0)
004                     *
005                     START   ORG   :300
006                     * START FROM BASIC WITH : MODE 6 :CALLM#300
007                     * CLEAR TABLE
008 0300 21D803                 LXI   H,TABLE
009 0303 015001                 LXI   B,336
010 0306 3600           CLEAR   MVI   M,0
011 0308 23                     INX   H
012 0309 0B                     DCX   B
013 030A 78                     MOV   A,B
014 030B B1                     ORA   C
015 030C C20603                 JNZ   CLEAR
016                     *
017 030F 21D803         RESTAR  LXI   H,TABLE
018 0312 015001                 LXI   B,336     * NOMBRE DE POINTS
019 0315 110000                 LXI   D,0       * COMPTEUR X
020 0318 7B             OSCIL   MOV   A,E
021 0319 32CF03                 STA   X
022 031C 7A                     MOV   A,D
023 031D 32D003                 STA   X+1
024 0320 7E                     MOV   A,M       * ANCIEN X
025 0321 32D103                 STA   Y
026 0324 3E00                   MVI   A,0
027 0326 32D703                 STA   COL
028 0329 CD5103                 CALL  DOT       * EFFACE L'ANCIEN POINT
029 032C C5                     PUSH  B
030 032D D5                     PUSH  D
031 032E E5                     PUSH  H
032 032F 3E00                   MVI   A,0
033 0331 CDC6EB                 CALL  :EBC6     * PDL(0)
034 0334 3AD800                 LDA   :D8       * VALEUR DU PDL(0)
035 0337 E1                     POP   H
036 0338 D1                     POP   D
037 0339 C1                     POP   B
038 033A 32D103                 STA   Y
039 033D 77                     MOV   M,A       * NOUVELLE VALEUR DE Y
040 033E 3E01                   MVI   A,1
041 0340 32D703                 STA   COL
042 0343 CD5103                 CALL  DOT       * ALLUME LE NOUVEAU POINT
043 0346 23                     INX   H         * INCREMENTE LA TABLE
044 0347 13                     INX   D         * INCREMENTE X
045 0348 0B                     DCX   B         * DECREMENTE LE COMPTEUR X
046 0349 78                     MOV   A,B
047 034A B1                     ORA   C
048 034B C21803                 JNZ   OSCIL     * POINT SUIVANT
049 034E C30F03                 JMP   RESTAR
050                     ************************************
051                     * SOUS-ROUTINE DE DESSIN D'UN POINT
052                     ************************************
053                     * PLACE MEMOIRE = #6644+(Y*#5A)-(X/4)IAND#7E (3E
```

oscilloscope

```
054 0351 C5          DOT     PUSH  B
055 0352 D5                  PUSH  D
056 0353 E5                  PUSH  H
057 0354 3ACF03              LDA   X
058 0357 32D303              STA   XAT
059 035A 3AD003              LDA   X+1
060 035D 32D403              STA   XAT+1
061 0360 3AD103              LDA   Y
062 0363 115A00              LXI   D,:5A
063 0366 210000              LXI   H,:0000
064                  *
065 0369 37          BCL2    STC           * MULTIPLIE A*D>HL
066 036A 3F                  CMC
067 036B 1F                  RAR
068 036C D27003              JNC   BCL1
069 036F 19                  DAD   D
070 0370 B7          BCL1    ORA   A
071 0371 CA7A03              JZ    CONT1
072 0374 EB                  XCHG
073 0375 29                  DAD   H
074 0376 EB                  XCHG
075 0377 C36903              JMP   BCL2
076                  *
077 037A 114466      CONT1   LXI   D,:6644   * PREMIERE PLACE MEMOIRE 6
078 037D 19                  DAD   D         * HL=(YAT*#5A)+#6644
079 037E C5                  PUSH  B
080 037F 0602                MVI   B,2
081 0381 37          LOPI    STC
082 0382 3F                  CMC
083 0383 3AD403              LDA   XAT+1
084 0386 1F                  RAR
085 0387 32D403              STA   XAT+1
086 038A 3AD303              LDA   XAT
087 038D 1F                  RAR
088 038E 32D303              STA   XAT
089 0391 05                  DCR   B
090 0392 C28103              JNZ   LOPI
091 0395 C1                  POP   B
092 0396 E67E                ANI   :7E       * X=X SHR 2 IAND #7E
093 0398 47                  MOV   B,A
094 0399 7D                  MOV   A,L
095 039A 90                  SUB   B
096 039B 6F                  MOV   L,A
097 039C 7C                  MOV   A,H
098 039D DE00                SBI   :00
099 039F 67                  MOV   H,A       * DANS HL: POSITION MEMOIRE
100 03A0 3ACF03              LDA   X
101 03A3 E607                ANI   :07       * POSITION DU POINT
102 03A5 47                  MOV   B,A
103 03A6 3E00                MVI   A,:00
104 03A8 04                  INR   B
105 03A9 37                  STC             * CARRY=1
106 03AA 1F          FINPT   RAR
107 03AB 05                  DCR   B
108 03AC C2AA03              JNZ   FINPT
109 03AF 47                  MOV   B,A       * DANS B MASQUE 00100000
110 03B0 EEFF                XRI   :FF
111 03B2 4F                  MOV   C,A       * DANS C MASQUE 11011111
112 03B3 1602                MVI   D,:2
113 03B5 3AD703              LDA   COL
114 03B8 5F                  MOV   E,A
115 03B9 1F          ROT     RAR
116 03BA 5F                  MOV   E,A
```

oscilloscope

```
117 03BB 7E                MOV   A,M
118 03BC DAC303            JC    SET
119 03BF A1                ANA   C          * EFFACE UN POINT
120 03C0 C3C403            JMP   CONTI
121 03C3 B0          SET   ORA   B          * DESSINE UN POINT
122 03C4 77          CONTI MOV   M,A        * DESSINE LE POINT EN MEM
123 03C5 7B                MOV   A,E
124 03C6 23                INX   H
125 03C7 15                DCR   D
126 03C8 C2B903            JNZ   ROT
127 03CB E1                POP   H
128 03CC D1                POP   D
129 03CD C1                POP   B
130 03CE C9                RET
131 03CF 0000        X     DBL   :0000
132 03D1 0000        Y     DBL   :0000
133 03D3 0000        XAT   DBL   :0000      * X POUR ROUTINE DOT
134 03D5 0000        YAT   DBL   :0000      * Y POUR ROUTINE DOT
135 03D7 00          COL   DATA  :00
136 03D8 0000        TABLE DBL   :0000
137                  FIN   ORG   TABLE+337
138 0529                   END
```

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

BCL1	0370	BCL2	0369	CLEAR	0306	COL	03D7
CONT1	037A	CONTI	03C4	DOT	0351	FIN	03DA
FINPT	03AA	LOPI	0381	OSCIL	0318	RESTAR	030F
ROT	03B9	SET	03C3	START	0000	TABLE	03D8
X	03CF	XAT	03D3	Y	03D1	YAT	03D5

mandala

```
5     REM +++ MANDALA  ....who is the author ? +++
10    T=9.0:MODE 6:COLORG 3 5 9 14
30    GOSUB 400:MODE 4:GOSUB 400:MODE 2:GOSUB 400:GOTO 10
35    END:REM ------------------------------------------
40    FOR S%=0 TO 20
41    L=RND(16.0):M=RND(16.0):N=RND(16.0):O=RND(16.0)
42    COLORG L M N O
44    FOR A=0.0 TO 700.0:K=GETC:IF K=0.0 THEN NEXT A
50    IF K=32.0 THEN GOSUB 202
55    IF K>0.0 AND K<>32.0 THEN GOSUB 200
60    NEXT S%:RETURN
90    REM ------------------------------------------------
200   L=9.0:M=L+1.0:N=L+3.0:O=L+2.0
202   FOR X=0.0 TO 20.0
205   COLORG L M N O:WAIT TIME T
210   COLORG O L M N:WAIT TIME T
220   COLORG N O L M:WAIT TIME T
225   COLORG M N O L:WAIT TIME T:NEXT
230   RETURN
390   REM ------------------------------------------
400   D=YMAX:E=D/2.0:F=E+0.5:G=F/4.0:H=(YMAX+1.0)/8.0
470   M=INT(H*4.0-1.0):FOR X=0.0 TO M:MD=INT(XMAX/2.0)
500   DRAW MD+X,O MD-X,M*2 20+X MOD 4
505   DRAW MD-X,O MD+X,M*2 20+X MOD 4
510   DRAW (MD-M),M+X MD+M,M-X 20+X MOD 4
515   DRAW (MD-M),M-X MD+M,M+X 20+X MOD 4
520   NEXT X:GOSUB 40:RETURN
```

16 COLOR CHARACTERS IN MODE 0 AND PICTURE STABILISATION MODIFICATION

For these modifications you need:

- A good soldering iron, preferably temperature controlled, around 40W.
  A soldering pistol won't do because you damage both print and components.
- A sharp knife.
- Some thin, isolated wire.
- Some solder, pliers and cutters.
- A resistor of 33K (orange/orange/orange).
- A capacitor of 1n5 polyester.
- A silicium diode 1N4148 or 1N914 (the yellow bar is the cathode).

These modifications are to be made on the main board rev.3, rev.4 or rev.5 only.
Some soldering experience is a necessity. We are not responsible for damage on your machine because of bad soldering or improperly executing these modifications.

1. 16 color characters in mode 0

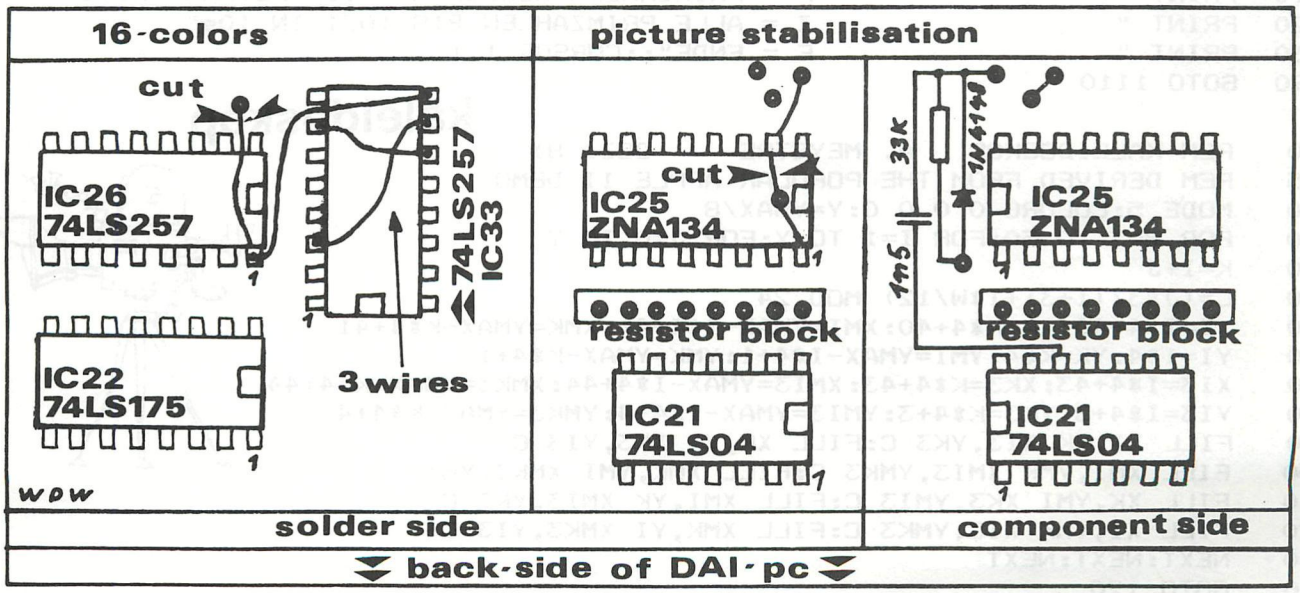
There is one cut to be made on the solder side of the board. It is on the track coming from pin 1 of the 74LS257 next to the ZNA134 to a plated through hole below this chip.
Cut it next to the hole.
Then solder 3 wires under the chip on the left of the former one.

- From pin 3 to pin 10.
- From pin 7 to pin 11.
- From pin 9 to pin 1 of the first chip (where you cutted the track).

2. Picture stabilisation

Again there is a cut to be made on the solder side of the board. It is on the big track underneath the ZNA134. Cut it between pin 15 and the plated through hole in the middle of the chip.
The components are soldered on the parts side of the board.
The capacitor is soldered from pin 12 of the 74LS04 above the ZNA134 to the plated through hole below the ZNA134 (the big hole on the right of the link).
Both the resistor and the diode are soldered from pin 1 of the ZNA134 to the hole mentioned before. The cathode of the diode must be on the bottom side of the board.

Now you can turn on your machine again. The jumping of your picture must be gone now.
A 16 color characters demo program is available.
For further information please contact the club, they will give you my address or phone number.
Good luck.



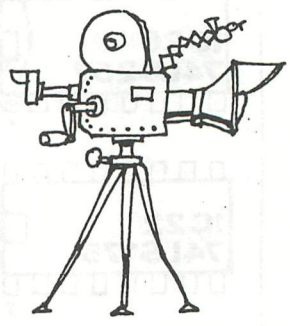
T

primzahlen

```
10    REM PRIMZAHLEN VON 1 BIS 1024 IN 10 SEKUNDEN
12    REM A. MEYSTRE , CH-4132 MUTTENZ / APRIL 81
15    CLEAR (2000):DIM S(255.0):S=VARPTR(S(0.0)):GOTO 100
20    M=1023.0:E=S+M:T=0:U=1:Z=2
25    A=#1010101:B=#1010001:C=#10101:D=#101:P=0
30    FOR I=1 TO 17:S(P)=C:S(P+1)=B:S(P+2)=B:S(P+3)=C:S(P+4)=D:S(P+5)=B:S(P+6)=A
34    S(P+7)=D:S(P+8)=A:S(P+9)=C:S(P+10)=D:S(P+11)=B:S(P+12)=C:S(P+13)=C:S(P+14)
=B
35    P=P+15:NEXT:S(255)=C
40    PRINT CHR$(12),2,3,5,:A=S+4
50    FOR N=7 TO M STEP Z:A=A+Z:IF PEEK(A)<>T THEN NEXT:SOUND 1 0 15 0 FREQ(750)
:GOTO 100
70    PRINT N,:IF N<32 THEN FOR I=A TO E STEP N+N:POKE I,U:NEXT
90    NEXT
100   WAIT TIME 20:SOUND OFF :WAIT TIME 200:GOTO 2200
1000  REM GRAPHISCHE DARSTELLUNG DER PRIMZAHLENGENERIERUNG
1010  REM A. MEYSTRE, CH-4132 MUTTENZ  /  FEB. 82
1015  F1=0:F2=5:F3=8:F4=15
1020  COLORG F1 F2 F3 F4:MODE 2A:MODE 2A:POKE #75,#20
1024  PRINT CHR$(12);TAB(4);"GRAPHISCHE DARSTELLUNG DER PRIMZAHLENGENERIERUNG"
1025  PRINT TAB(4);"NACH DER METHODE VON ERATOSTHENES (275-194 v.Chr.)"
1026  PRINT TAB(24);CHR$(#5F);CHR$(#5F);CHR$(#5F);CHR$(#5F);CHR$(#5F)
1028  PRINT TAB(4);"SETZEN VON 2 BIS  ";CHR$(142);CHR$(143);" 1024";
1029  PRINT TAB(38);"PRIMZAHL :";
1030  FILL 16,8 55,47 F3:FILL 20,12 51,43 F2:DOT 20,12 F1:DOT 21,12 F1
1040  FOR I=2.0 TO 32:J=I:GOSUB 2000:IF SCRN(X,Y)<>F2 THEN 1100
1050  CURSOR 48,0:PRINT I;
1060  FOR L=I+I TO 1023 STEP I:J=L:GOSUB 2000
1065  IF SCRN(X,Y)=F2 AND I>3 THEN SOUND 1 0 15 0 FREQ(750.0):DOT X,Y F4:WAIT TI
ME 10
1070  DOT X,Y F4:WAIT TIME SQR(I)+1.0:DOT X,Y F1:SOUND OFF :NEXT
1100  NEXT:GOTO 2200
1110  K=GETC:WAIT TIME 5:IF K=0 THEN 1110
1120  IF K=ASC("G") THEN 1000
1125  IF K=ASC("Z") THEN 20
1130  IF K=ASC("P") AND F4>0 THEN GOSUB 2100:GOTO 2200
1135  IF K=ASC("P") AND F4=0 THEN PRINT :PRINT "ZUERST GRAPHIK ERSTELLEN !!!":GO
TO 1110
1140  IF K=ASC("E") THEN POKE #75,#5F:END
1150  GOTO 1110
2000  X=J MOD 32:Y=J/32:X=X+20:Y=Y+12:RETURN
2100  PRINT CHR$(12);TAB(23);"DIE PRIMZAHLEN"
2130  FOR J=2 TO 1023:GOSUB 2000:IF SCRN(X,Y)=F1 THEN NEXT:RETURN
2140  CURSOR 28,1:PRINT J:DOT X,Y F4:INPUT Z$:IF Z$<>"" THEN 2200:DOT X,Y F2:NEX
T:RETURN
2200  PRINT CHR$(12);"TASTEN-FUNKTION: G = GRAPHISCHE DARSTELLUNG"
2210  PRINT "                  P = PRIMZAHLEN AUSGEBEN  (VIA RETURN-TASTE)"
2220  PRINT "                  Z = ALLE PRIMZAHLEN BIS 1021 IN 10s"
2230  PRINT "                  E = ENDE";:CURSOR 1,1
2290  GOTO 1110
```

kaleidoskop

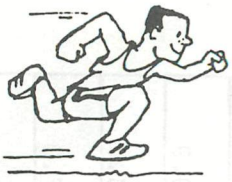
```
100   REM KALEIDOSKOP   A. MEYSTRE  /  DEZ. 81
105   REM DERIVED FROM THE POPULAR APPLE II DEMO
110   MODE 5:COLORG 0 0 0 0:Y=YMAX/8
120   FOR W=3 TO 50:FOR I=1 TO Y:FOR J=0 TO Y
130   K=I+J
140   C=(J*3/(I+3)+I*W/12) MOD 24
150   XI=I*4+40:XK=K*4+40:XMI=YMAX-I*4+41:XMK=YMAX-K*4+41
160   YI=I*4:YK=K*4:YMI=YMAX-I*4+1:YMK=YMAX-K*4+1
170   XI3=I*4+43:XK3=K*4+43:XMI3=YMAX-I*4+44:XMK3=YMAX-K*4+44
180   YI3=I*4+3:YK3=K*4+3:YMI3=YMAX-I*4+4:YMK3=YMAX-K*4+4
190   FILL XI,YK XI3,YK3 C:FILL XK,YI XK3,YI3 C
200   FILL XMI,YMK XMI3,YMK3 C:FILL XMK,YMI XMK3,YMI3 C
210   FILL XK,YMI XK3,YMI3 C:FILL XMI,YK XMI3,YK3 C
220   FILL XI,YMK XI3,YMK3 C:FILL XMK,YI XMK3,YI3 C
230   NEXT:NEXT:NEXT
240   GOTO 120
```



benchmarks

All Timings in seconds. All variables FPT

COMPUTER		BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8
DAI with 9511	8080A	0.93	4.78	10.05	9.78	11.20	18.12	30.11	2.14
APPLE II	6502	1.3	8.5	16.0	17.8	19.1	28.6	44.8	10.7
TRS 80	Z80	2.5	18.0	34.5	39.0	45.0	67.0	109.0	
COMPUCOLOR II	8080A	2.0	10.9	22.4	23.9	25.7	38.7	55.2	10.8
SORCEROR (EXIDY)	Z80	1.8	10.0	20.7	22.2	24.3	37.6	53.7	9.6
PET	6502	1.7	9.9	18.4	20.4	21.0	32.5	50.9	12.3
SUPER PET	6502	1.7	10.0	18.4	20.3	21.9	32.4	51.0	11.9
CHALLENGER C2-4P	6502	1.4	7.8	15.0	16.5	17.8	27.0	39.5	7.5
ATARI 800	6502	2.35	7.41	19.89	23.16	26.78	40.75	61.51	43.08
SUPERBRAIN	2XZ80	1.56	5.25	14.01	13.91	14.77	26.26	43.24	5.56
SHARP MZ-80K	Z80	1.4	9.4	16.3	22.5	25.4	36.8	51.1	10.2
SINCLAIR ZX81 (FAST MODE)	Z80	4.5	6.9	16.4	15.8	18.6	49.7	68.5	22.9
SINCLAIR ZX81 (SLOW MODE)	Z80	17.7	27.2	65.3	63.0	74.2	199.3	275.6	91.6
CROMEMCO Z-2D	Z80	2.0	5.8	16.4	19.1	20.1	31.5	42.5	23.0
COMMODORE VIC	6502	1.4	8.3	15.5	17.1	18.3	27.2	42.7	9.9
TRS 80 COLOR COMPUTER	6809E	2.0	11.5	22.2	23.9	27.0	41.5	61.1	13.0
NEC PC-8001	Z80A	1.7	8.3	18.1	17.8	18.6	29.5	49.2	7.0
TRS 80 MODEL 11 (S/PRECISION)	Z80	1.0	5.0	13.0	13.0	14.0	23.0	35.0	6.0



DAI OP DE PISTE

Op 22 Mei j.l. werd in Eindhoven een scholen-sportdag gehouden, waarbij met groot succes een tweetal DAI-PC's werd ingezet.
Er werd deelgenomen door twaalf zesde klassen van lagere scholen, waarbij jongens en meisjes elk een zestal individuele sportprestaties moesten leveren zoals 60 mtr hardlopen, kogelstoten, verspringen.
Dus 24 groepen van gemiddeld vijftien kinderen.
Vanuit het gemiddelde per groep werden punten voor die groep berekend met een omrekeningsformule per sport. Daarnaast werd in twee pools voetbal en kastie gespeeld. Ook van deze uitslagen werden punten toegekend, doelsaldi berekend etc.

Een DAI 48K zorgde voor het berekenen van de punten, punten totalen, overzicht van verwerkte uitslagen, overzichten van de poolstanden, en natuurlijk meteen aan het einde van de dag voor elke school een totaal-lijst.

Bovendien werd op geschikte momenten tijdens de dag een array gevuld met de schoolnamen en de bijbehorende tussenstanden, waarna een SAVEA.
De tweede DAI stond met LOADA te wachten, zijn recorderingang was via een tussenversterkertje op de recorderuitgang van de eerste gekoppeld.
Zodra de LOADA was ontvangen gaf deze PC de tussenstanden in kleur weer op een grootbeeld TV op het veld.
En met het knipperlicht op de momenteel hoogste tussenstand. Dat bleef zo actief tot de volgende LOADA.

Waar vorige jaren vier mensen bezig waren met rekenen en lijsten kon nu met twee mensen, een om de binnenkomende uitslagenlijsten op te lezen en een tikker, veel meer, veel sneller en met overzicht tussen de inputruns gewerkt worden.

Bij het ontwikkelen van het programma bleek het rekenende gedeelte nauwelijks problemen te geven.
Foutmeldingen tijdens de run, zowel vanuit het programma als door foutieve input, waren natuurlijk absoluut ontoelaatbaar. Het zal maar gebeuren dat in de loop van de middag, met 80% van de gegevens in het geheugen, een ERROR optreedt !!
Aan beveiliging tegen inputfouten en een gedegen controle van het programma in alle mogelijke takken moest dus erg veel tijd besteed worden, en een regelmatige tussentijdse SAVEA van diverse arrays moest worden veilig gesteld.

Een paar weken vooraf werden per school, per groep (jongens/meisjes),per sport ook de alfabetische naamlijsten (144) geprint.
Er namen in totaal 174 jongens en 195 meisjes deel, zodat het nu ook de moeite waard is statistisch nog wat gegevens te berekenen. Gemiddelde, spreiding, hoogste prestatie individueel, en vooral of de toegepaste puntenberekening nog op de werkelijke prestaties klopt.

Mocht er belangstelling bestaan voor dit soort programma's dan zullen we ze graag aanbieden.

Met vriendelijkê groeten,

Hans Vollinga
Chopinlaan 44
5653 EW Eindhoven

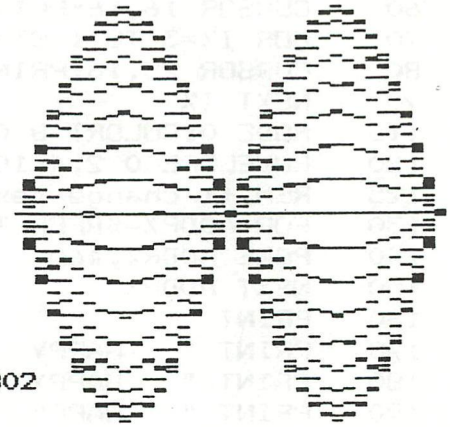
Ger Esselink
Nederhoven 11
5655 BR Eindhoven

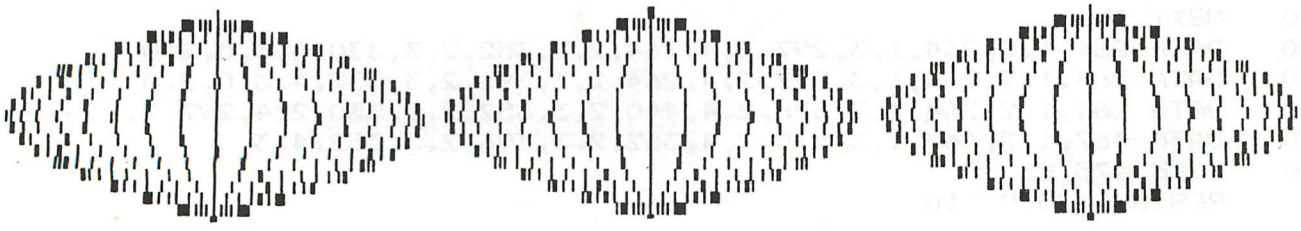
birthday song

```
5     REM +++ BIRTHDAY SONG - Barry Hunt - 4/3/82 +++
6     REM +++ The name of the person whose birthday +++
7     REM +++ is being celebrated should be substituted +++
8     REM +++ for "CLAUDIE" (the name of my wife) +++
9     REM +++ in the third line of the song +++
10    REM +++ and the third DATA  statement should be +++
11    REM +++ changed accordingly +++
15    PRINT CHR$(12):POKE #75,32
20    CURSOR 20,20:PRINT "BIRTHDAY  SONG"
30    CURSOR 20,19:FOR I%=1 TO 14:PRINT CHR$(11);:NEXT I%
40    CURSOR 16,16:PRINT "Press any key to start."
50    IF GETC=0.0 THEN 50
60    CURSOR 16,16:PRINT "                    "
70    FOR I%=3 TO 1 STEP -1
80    CURSOR 25,16:PRINT I%:WAIT TIME 30
90    NEXT I%
110   MODE 0:COLORT 8 0 8 12:PRINT CHR$(12)
120   ENVELOPE 0 2,4;10,6;15,20;12,30;8,40;4,50;0
125   REM ++ change text letter size
130   FOR ADDR%=#BFEF TO #BCCB STEP -#86
140   POKE ADDR%,#6A
150   NEXT ADDR%
160   PRINT
170   PRINT "   HAPPY  BIRTHDAY  TO  YOU"
180   PRINT "   HAPPY  BIRTHDAY  TO  YOU"
190   PRINT "   HAPPY  BIRTHDAY  DEAR  CLAUDIE"
200   PRINT "   HAPPY  BIRTHDAY  TO  YOU"
205   REM ++ set start of line and current position
210   START%=#BED2:POSN%=START%
220   FOR I%=1 TO 28
225   REM ++ read note frequncy, note length and (length of syllable - 1)
230   READ NOTE%,NL%,CL%
235   REM ++ Note of zero means new line
240   IF NOTE%=0 THEN 380
250   SOUND 0 0 15 0 FREQ(NOTE%)
260   SOUND 0 0 8 0 FREQ(NOTE%*2.0)
270   SOUND 2 0 4 0 FREQ(NOTE%*4.0)
275   REM ++ change colour of syllable
280   FOR ADDR%=POSN% TO (POSN%-2*CL%) STEP -2
290   POKE ADDR%,#FF
300   NEXT ADDR%
310   WAIT TIME 8*NL%
320   SOUND OFF
325   REM ++ restore colour of syllable
330   FOR ADDR%=POSN% TO (POSN%-2*CL%) STEP -2
340   POKE ADDR%,#0
350   NEXT ADDR%
355   REM ++ update current position
360   POSN%=POSN%-2*CL%-2
370   GOTO 400
375   REM ++ new line
380   WAIT TIME NL%*8
390   START%=START%-#86:POSN%=START%
400   NEXT I%
410   DATA 264,1,2,264,1,3,297,2,4,264,2,3,352,2,3,330,4,3,0,2,0
420   DATA 264,1,2,264,1,3,297,2,4,264,2,3,396,2,3,352,4,3,0,2,0
430   DATA 264,1,2,264,1,3,528,2,4,440,2,3,352,2,5,330,2,4,297,2,2,0,0,0
440   DATA 467,1,2,467,1,3,440,2,4,352,2,3,396,2,3,352,4,3
450   POKE #75,95
500   RESTORE:GOTO 110
```

lopende & staande golven

```
20    CLEAR 200
30    COLORT 14 0 0 0
40    PRINT CHR$(12):PRINT
50    PRINT " Staande GOLVEN II  -  Natuurkunde ":PRINT
60    PRINT "                       Roelants J.":PRINT :PRINT
75    PRINT "  Druk op de SPACE BAR om het bewegend beeld te bevriezen."
76    PRINT "  Druk op de B-toets om een andere tijdvertraging "
77    PRINT
80    EE%=0
100   DATA #E5,#C5,#D5,#F5,#11,#00,#10,#0E,#24
140   DATA #06,#25,#D5,#CD,#00,#35,#36,#FF
180   DATA #CD,#00,#35,#23,#36,#FF,#CD,#00,#35
220   DATA #36,#FF,#23,#36,#FF,#05,#C2,#0C,#30
270   REM TIJDVERTRAGING
271   DATA #26,#FF,#2E,#80,#2D,#C2,#27,#30
272   DATA #CD,#BB,#D6,#B7,#FE,#42,#CA,#5B,#30,#FE,#20,#CA,#07,#35
273   DATA #25,#C2,#25,#30
300   REM
310   DATA #D1,#06,#25,#CD,#00,#35,#36,#00
340   DATA #CD,#00,#35,#23,#36,#00
360   DATA #CD,#00,#35,#36,#00,#05,#C2,#40,#30,#0D
410   DATA #C2,#09,#30,#C3,#04,#30
415   REM BASIC
420   DATA #D1,#F1,#D1,#C1,#E1,#C9,-1
490   REM SUB TEKEN
500   DATA #1A,#6F,#13,#1A,#67,#13,#C9
560   REM GETC
570   DATA #CD,#BB,#D6,#B7,#CA,#07,#35,#C3,#3D,#30,-1
800   X%=#3000
802   READ T%:IF T%<>(-1) THEN POKE X%,T%:X%=X%+1:GOTO 802
805   X%=#3500
806   READ T%:IF T%<>(-1) THEN POKE X%,T%:X%=X%+1:GOTO 806
810   INPUT " --> Tijdvertraging 1-255 ";V%:PRINT
812   IF V%>255 OR V%<1 THEN GOTO 810
815   POKE #3024,V%:POKE #3026,V%
900   INPUT " --> Berekening van de golven J/N";A$:PRINT
905   IF A$="J" GOTO 1000
910   IF A$<>"J" THEN 1500
920   PRINT " --> Eerste maal steeds berekenen ":GOTO 900
1000  REM CREEER DATA
1010  INPUT " --> Aantal golflengten per beeld 1...3 ";U:PRINT
1020  A=2.0*PI/36.0:F%=35:EE%=1
1030  PRINT CHR$(12):PRINT "-->  Even wachten om de golf te berekenen .":PRINT
1040  J%=#1000
1060  FOR T%=0 TO 35:FOR X%=0 TO 36
1080  Y1%=42*SIN((U*X%-T%)*A):Y%=Y1%+42:GOSUB 2000
1100  Y2%=42*SIN((U*X%+T%)*A):Y%=Y2%+42:GOSUB 2000
1120  Y%=170+Y1%+Y2%:GOSUB 2000
1140  NEXT X%:CURSOR 30,2:PRINT " --> countdown ";F%-T%;" #":NEXT T%
1500  MODE 6:MODE 6
1510  COLORG 0 5 10 15:REM TRY OTHER VALUES..
1520  CALLM #3000
1540  GOTO 810
1990  END
2000  AD%=26098+2*(40-X%)+Y%*90
2020  LO%=AD% IAND 255:HI%=AD% SHR 8
2040  POKE J%,LO%:POKE J%+1,HI%:J%=J%+2
2060  RETURN
```





lopende & staande golven

LOPENDE EN STAANDE GOLVEN

Bespreking van het programma

Het programma demonsteert de mogelijkheid om zeer snel een aantal opeenvolgende beelden op het scherm te plaatsen om zo een bewegend effect te bekomen .
Daartoe worden vanuit een M.T. programma rechtstreeks een aantal bytes in de video RAM (MODE6 - start 26098) geplaatst .

Het BASIC programma berekent eerst de opeenvolgende beelden, en pookt de gevonden waarden weg in een tabel die start op adres :3000 en ongeveer:2000 bytes in beslag neemt .

Het M.T. programma is geplaatst vanaf :5000 tot :5071 en kan via het S commando in UT. worden ingevoerd (SAVE via W5000 5071).

Om te beweging te vertragen is er een tijdlus ingebouwd .
De SPACE BAR werkt als aan-uit schakelaar om het beeld te bevriezen of verder te laten lopen . Door de B-toets in ter drukken keren we terug van de machinitaal subroutine naar het BASIC programma van waaruit we andere beelden kunnen berekenen of een andere tijds-vertraging kiezen .

De berekening van de lopende golven gebeurt op de regelnummers 1080 en 1100, de staande golf op 1120 .
Subroutine 2000 zorgt er telkens voor dat de coordinaten X% en Y% worden omgerekend naar hun overeenstemmend adres AD% in de VIDEO RAM. AD% wordt nu opgesplitst in LO% en HI% en weggepookt op de adressen J% en J%+1 in de tabel .

In totaal beschikken we hier over 36 beelden waarin drie figuren bestaande uit 37 bytes zijn opgeslagen .

Voeren we andere waarden in voor X% en Y% dan zal er een andere reeks van beelden ontstaan .

J. Roelants

DCR tape control

```
*LIST
1000  REM dcr tapecontrol
1001  REM ref. ward adriaens wilfried hermans inno broekman
1020  CLEAR 5000:COLORT 8 0 8 0
1040  PRINT CHR$(12):CURSOR 10,16:PRINT "D C R  T A P E C O N T R O L"
1060  WAIT TIME 40:PRINT CHR$(12)
1100  PRINT CHR$(12):RESTORE
1120  READ N%:READ P$:IF P$="###" GOTO 1220
1140  IF N%<10.0 THEN PRINT " ";N%;">>       ";P$
1160  IF N%>9.0 THEN PRINT N%;">>       ";P$
1180  GOTO 1120
1220  REM wandelende cursor
1240  POKE #74,3:POKE #75,#20
1260  REM start keuze
1280  FOR X%=23 TO 0 STEP -1
1300  CURSOR 5,X%:PRINT CHR$(1)
1340  FOR Z%=1 TO 200
1360  IF GETC<>0 GOTO 1480
1380  NEXT Z%
1400  CURSOR 5,X%:PRINT CHR$(32)
1440  NEXT X%
1460  GOTO 1100
1480  CN%=24.0-X%
1500  RESTORE
1520  FOR I=1.0 TO CN%:READ N%:READ P$
1540  IF P$="###" THEN PRINT CHR$(12):CURSOR 0,2:PRINT CHR$(1);"   MEMOCOM kent
dit programma niet !  ";CHR$(1):WAIT TIME 200:GOTO 1100
1560  IF CN%=X% THEN GOTO 1100
1580  IF P$="geen programma" GOTO 1520
1590  IF CN%>22 THEN GOTO 1670
1600  NEXT I
1660  PRINT CHR$(12):CURSOR 0,2:PRINT CHR$(1);" MEMOCOM is nu aan het zoeken naa
r : ";P$;".";CHR$(1):GOTO 2000
1670  PRINT CHR$(12):CURSOR 0,2:PRINT CHR$(1);" MEMOCOM draait nu naar het einde
 van de band. ";CHR$(1):GOTO 2070
2000  IF N%=1.0 THEN GOTO 2050
2005  IF CN%>22 THEN GOTO 2060
2010  FOR LOOP%=0 TO N%-2
2020  CALLM #F000:REM DCR0:SKIP1
2040  NEXT LOOP%
2050  LOAD
2070  CALLM #F000:REM DCR0:SKIP9:SKIP9:SKIP9
2080  END
3000  DATA 1,"DAMMEN"
3010  DATA 2,"SCHAKEN"
3020  DATA 3,"ROTERENDE PYRAMIDE"
3030  DATA 4,"CURSORTEKENEN"
3040  DATA 5,"BARRICADE"
3050  DATA 6,"VIER OP EEN RIJ"
3060  DATA 7,"RAKETSPEL"
3070  DATA 8,"TORENS VAN HANOI"
3080  DATA 9,"ACIGOL"
3090  DATA 10,"OMKEREN"
3100  DATA 11,"DOOLHOF"
3110  DATA 12,"SKIPISTE"
3120  DATA 13,"BLACKJACK"
3130  DATA 14,"ANDROIDENSPEL"
3140  DATA 15,"LANDMIJNEN"
3150  DATA 16,"geen programma"
3160  DATA 17,"geen programma"
3170  DATA 18,"geen programma"
3180  DATA 19,"geen programma"
3190  DATA 20,"geen programma"
3200  DATA 21,"geen programma"
3210  DATA 22,"geen programma"
3220  DATA 23,"naar bandeinde"
3230  DATA 24,"###"
```

```
100   REM +++ PHONE RINGING +++
120   ENVELOPE 0 10,2;15,2;14,4;12,5;8,10;0
130   ENVELOPE 0 15,10;0,10;15,10;0,10;0
150   SOUND 0 1 15 0 FREQ(1000.0)
160   SOUND 0 0 13 1 FREQ(2000.0)
170   SOUND 0 0 15 2 FREQ(5000.0)
1000  GOTO 120
```

Freiburg, 25.2.82

Dr. Wolfgang Werbeck
D-78 Freiburg
Innsbrucker Straße 2
Telefon 49 45 77

letter from mr Werbeck

Herrn
W. Hermans
Heide 4
B-3171 Westmeerbeek

Lieber DAInamic-Freund.

Beim Blättern in älteren Heften der Elektronik-Zeitschrift "Elektor" (in den Niederlanden "Elektuur") fand ich in der Dezember-Nummer 1981 einen interessanten Artikel mit dem Titel "IPROM". Ich dachte dabei sofort an den leeren 24-poligen Sockel auf dem Board meines DAI P.C. Denn der Arithmetic Processor AM9511 ist mir viel zu teuer.
Beim IPROM wird ein 2 K x 8 Bit - RAM (HM 6116 LP, Preis DM 40,00) als "Pseudo-ROM" verwendet. Man kann - nach entsprechendem Aufbau mit zusätzlichen Bauteilen - ein Programm wie bei einem RAM einfach laden. Es ist dann durch einen kleinen Schalter vor Überschreiben geschützt und bleibt mit Hilfe von zwei Knopfzellen auch nach Abschalten des Computers erhalten. Man benötigt also weder ein Programmiergerät noch ein Löschgerät und hat dennoch ein ROM, in das man jederzeit ein neues Programm einschreiben kann.
Leider ist die Anschlußbelegung des freien Sockels völlig ungeeignet für die Aufnahme eines IPROMs.
Fragen Sie doch einmal einige Hardware-Spezialisten im DAInamic Club, ob sie eine Lösung für dieses Problem wissen.

Ich habe mir übrigens die vier zusätzlichen Tasten nach der Bauanleitung von Herrn Dessart (DAInamic Nr.7,S.207 ff.) eingebaut und bin sehr zufrieden damit.

Mit besten Grüßen

W. Werbeck

IMP INT ******* G E T A L L E N ******* IMP FPT

We kennen in programma's vele plaatsen waar gebruik gemaakt wordt van getallen. Deze getallen kunnen echter op verschillende manieren voorkomen en niet iedereen is bekend met of zich bewust van deze verschillen.

I) Getallen kunnen voorkomen als constanten of als variabelen Voorbeeld: DOT X,Y+2 9 hier zijn X en Y variabelen en 2 en 9 constanten.
II) Getallen kunnen voorkomen in drie vormen in een programma als geheel getal (integer) of als breuk (floating point); de derde verschijningsvorm is hier niet interessant namelijk de geschreven vorm (string).

Om alle misverstanden te voorkomen wil ik nu reeds van tevoren zeggen dat de programmeur in alle gevallen bepaalt welk type een getal zal zijn.

U weet waarschijnlijk al uit de foutmeldingen dat getallen in elk type binnen bepaalde grenzen moet liggen.
Bij gehele getallen (integers) zijn deze grenzen +/- 2^31, bij drijvende komma getallen (brrrr floating point dus) heb ik destijds vastgesteld dat de grenzen hier afhankelijk zijn van de omstandigheden. In een programma kon ik de variabele op de waarde 9.22E18 krijgen maar bij invoer van C=8E18 kreeg ik reeds overflow.
Als we in een programma trouwens de waarde van een variabele steeds laten toenemen krijgen we op den duur OVERFLOW; ook als we een getal steeds kleiner laten worden (zonder de minervoor steeds groter) krijgen we OVERFLOW.
Als we een variabele echter door delen met een getal groter dan 1 kleiner laten worden (het getal komt steeds dichter bij nul) krijgen we geen waarschuwing van de DAI voor deze UNDER FLOW maar wordt de variabele gewoon nul gemaakt.
Hier dient in bepaalde gevallen wel degelijk rekening mee gehouden te worden:Als we een getal eerst honderd maal halveren en dan honderd maal verdubbelen zouden we het oorspronkelijke getal terug moeten krijgen, we zullen afhankelijk van de startwaarde iets anders krijgen, meestal nul.
programma:

```
IMPFPT
10 A=123
20 FOR I=1 TO 100:A=A/2:NEXT
30 FOR I=1 TO 100:A=A*2:NEXT
40 ?A
```

Bij het bestuderen van programma's van anderen heb ik vaak moeten vaststellen dat het niet bij iedereen bekend is dat programma's sneller werken als integerwaarden gebruikt worden. Een ander voordeel van het gebruik van integers ligt in de logica: ik vind het vreselijk als iemand (via zijn/haar DAIprogramma beweert dat ik door een bepaalde slag of zet 3.0 pionnen heb veroverd, alsof je er ook 2.74653 zou kunnen veroveren. Ik hoor nu al de protesten: Ja maar dat wilde ik niet dat deed de DAI. Klopt omdat U hem daartoe de opdracht gaf al geef ik toe dat het vaak een impliciete opdracht is geweest en waar U niet aan gedacht hebt. De meest vergeten opdracht van dit type is de opdracht IMP FPT die U (ja U) geeft door de DAI aan te zetten of reset te geven.
Na deze IMP FPT zijn alle variabelen en vele (niet alle) constanten van het fpt-type.
Eerst gaan we wat in op het snelheidsaspect. Ik hoop dat U mij niet gelooft en ik zal ook niet trachten iemand te overtuigen als de DAI dat voor mij kan doen.

Ga achter de DAI zitten, zet hem aan of reset en tik in:

```
10 MODE 1:K=10
20 A=0:B=0
30 DRAW XMAX-1,YMAX-2 A+1,B+1 K
40 B=B+2:IF B+1<YMAX-1 GOTO 30
50 IF K<>0 THEN B=0:K=0:GOTO 30
60 END
```

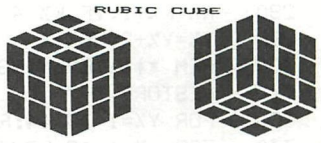
Geef RUN en tracht de tijd vast te stellen die de DAI hiervoor nodig heeft, eventueel programma uitbreiden met 5 J=0 en 55 J=J+1:IF J<50 GOTO 10.
geef LIST en zie dat het programma niet meer is wat U intikte.
Geef nu reset of machine uit/aan en tik weer precies hetzelfde in, tik RUN en ik denk dat U overtuigd bent.
N.B. het progamma is ter demonstatie het kan heus sneller:

```
10 MODE2:M=XMAX-1:N=YMAX-1:FOR K=10 TO 0 STEP -10
20 FOR B=1 TO N STEP 2:DRAW M,N 1,B K:NEXT:NEXT:END
```

Als U de listing van de fpt-versie nog herinnert zult U opgemerkt hebben dat er 20 A=0.0:B=0.0 stond. De ingevoerde constanten blijken automatisch in fpt-notatie te staan.
Vreemd genoeg blijkt echter in regel 30 dat de daar ingevoerde constanten wel in de door ons ingetikte vorm staan.
We kunnen dit begrijpen als we in het handboek lezen dat getallen voor de bepaalde gebruikstoepassingen vaak in een standaard vorm verwacht worden.
Bij een toewijzing zoals regel 20 kijkt DAI naar IMPINT resp IMPFPT maar bij een DRAW als regel 30 verwacht hij integers.
Zijn variabelen of constanten niet in de verwachtte vorm, zet de DAI ze voor ons om naar het juiste type. Dit kost echter wel tijd!!!!
We analyseren de hopeloze regel 30:
Voor elke keer wordt de XMAX en YMAX aangeroepen.
Elke keer wordt deze constante met een constante verminderd het resultaat is dus ook ... constant.
Bij de variabele A ,die overigens een constante waarde heeft wordt volkomen overbodig elke keer 1 opgeteld, we hadden A vanzelfsprekend 1 groter kunnen definieren. Maar het is nog erger: bij A een fpt-variabele wordt 1 een int-constante opgeteld. Dit is onmogelijk en dus wordt eerst de integer omgezet in fpt, daarna de optelling gedaan en dan wordt het geheel weer omgezet in int voor de DRAW-opdracht.
U kunt zelf nagaan dat zeer veel instructies integers verwachten of daar ook mee kunnen werken, als U ze echter fpt voert protesteren ze niet maar straffen U wel af door het programma trager af te werken of zelfs andere dingen te doen dan U bedoelde.
Ter illustratie geen protest bij COLORG 0.0 5.3 9.8 3.4 of FOR I=0 TO XMAX maar wat gaat dat allemaal veel sneller in integers.
U begrijpt dat ik er een voorstander van ben om standaard te beginnen met IMPINT (inderdaad spatie niet nodig) en dan wel ! te zetten (ook .0) indien dit gewenst is.
N.B. er is een verschil tussen IMPINT en IMPFPT gevolgd door IMPINT A-Z en omgekeerd, speelt er wat mee en verbaast U.

*** aanwijzingen gebruik RUBIK'S CUBE ***

De gekozen kleuren zijn de standaard kleuren zoals op de originele kubus te vinden zijn. De gekozen volgorde is die zoals de kubus in de verpakking zit.
Komen de kleuren bij U niet zo goed over (ik gebruik RGB) kunt U de kleuren wijzigen in regel 2010.
Pas op geef eerst IMPINT voordat U EDIT2010 doet.
De linker kubus in beeld geeft voor,rechter en bovenvlak, de rechter kubus is de kubus in de hand "horizontaal" gedraaid zo dat U onder,linker en achtervlak ziet. Speel wat na "I" met de vlakken zodat U weet wat U doet.
EDe Equator zit tussen boven en ondervlak, het Middenvlak tussen linker en rechtervlak en het Staande vlak tussen voor en achtervlak, te besturen met E,e,M,m,S en s.
Andere kleuren voor de vlakken zijn te geven in regel 2013 t/m 2019. zie de voorbeelden in regel 2011 en regel 2012.
Het middelste blokje van elk vlak kan ook een andere kleur krijgen (dan wel alle zijvlakken dezelfde kleur middenblokje) door de REM's in regel 2080 en 2090 te verwijderen.
Het middenblokje is dan zwart; andere kleur door 2080 te beginnen met KM=#70 als kleur 7 gebruikt moet worden.
In de stand W is het mogelijk de teller te resetten door C
Het draaitempo kan veranderd worden door de loop op regel 50 te veranderen. De draaiingen kunnen net zoals bij LIST beinvloed worden. Veel plezier. fhd



TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS-TIPS

Begin programmeren in bijna alle gevallen met IMPINT

Zet de uitleg van een programma wel in het begin van de "RUN" maar niet in het begin van de LIST.
Dwz begin bv met 10 CLEAR 3000:GOSUB 10000: enz
terwijl we de uitleg dan zetten vanaf regel 10000; het programma wordt sneller en overzichtelijker.

Doe niet mee aan de domme amerikaanse gewoonte te beginnen met HOE HEET JE ? om dat gegeven vervolgens nooit meer te gebruiken.

Begin niet met "WILT U UITLEG ?"
Er zijn twee mogelijkheden:
I) uitleg klein geef die dan ongevraagd.
II) uitleg groot maak een apart uitlegprogramma.
    voordelen 1-als U geen uitleg nodig hebt minder inlezen
              2-programma mogelijk iets sneller
              3-hele grote programma's passen door splitsing
                wel in het geheugen.
              4-inlezen hoofdprogramma tijdens geven uitleg

Begin elk groter programma met een "leader" of "header"
Hier bedoel ik mee een beeld waarin de naam van het programma staat en de naam van de auteur.
Liefst op een enigszins verantwoorde manier.

Laat de aankondiging en de uitleg niet zolang in beeld staan als U goeddunkt maar laat de gebruiker van het programma deze vrijheid, door bv te wachten op indrukken spatiebalk.

Geef bij het vragen om een moeilijkheidsgraad aan wat gemakkelijk en wat moeilijk is.

```
200   MODE 6A:PRINT CHR$(12);:POKE #75,32
210   COLORT 12 0 12 12:COLORG 12 0 10 15
220   POKE #7557,#4A:POKE #7556,223
230   PRINT "SNOOPY":POKE #74D0,208
240   DIM C%(4)
250   C%(1)=12:C%(2)=0:C%(3)=10:C%(4)=15:K%=1
260   FOR X%=0 TO 48 STEP 2
270   FILL X%,Y% 330-X%,210-Y% C%(K%)
280   K%=K%+1:IF K%>4 THEN K%=1
290   Y%=Y%+2:NEXT
300   REM *** TEKENING
310   RESTORE
320   FOR Y%=1 TO 44:READ A$:A$=RIGHT$(A$,LEN(A$)-1):KY%=Y%*2
330   FOR X%=1 TO LEN(A$)-1:T$=MID$(A$,X%,1):IF T$=" " THEN 390
340   IF T$="X" THEN C%=0:GOTO 370
350   IF T$=":" THEN C%=10:GOTO 370
360   IF T$="-" THEN C%=15
370   DOT 100+X%,150-KY% C%:DOT 100+X%,149-KY% C%
380   DOT 230-X%,150-KY% C%:DOT 230-X%,149-KY% C%
390   NEXT:NEXT
400   REM *** AUTOMATISCHE KLEURWISSELINGEN
410   CURSOR 2,2:PRINT "OM DE KLEUREN TE DOEN STILHOUDEN,DRUK OP DE SPATIEBALK ";
420   C1%=RND(16):C2%=RND(16):COLORG 12 0 C1% C2%:WAIT TIME 5
430   G%=GETC:IF G%=32 THEN 470
440   WAIT TIME 5:G%=GETC:IF G%=32 THEN 470
450   WAIT TIME 5:G%=GETC:IF G%=32 THEN 470
460   GOTO 420
470   CURSOR 2,2:PRINT "OM DE KLEUREN TE DOEN WISSELEN,DRUK OP DE SPATIEBALK  ";
480   G%=GETC:WAIT TIME 3:IF G%<>32 THEN 480
490   GOTO 410
1000  REM *** SNOOPY
1010  DATA .                XXXX
1020  DATA .               X----XX
1030  DATA .              X-XXXXX-X                XXXXX
1040  DATA .             X-XXXXXXX-X             XXX-----XX
1050  DATA .          XXXX-XXXXXXX-XXX       XXXX----------XX
1060  DATA .        XX:::X-XXXXXXX-XXXXXXXXX----------------XX XXX
1070  DATA .      XX::::::X-XXXXX-X-----------------------------XXX-X
1080  DATA .     X:::::::::XX----XX-----XX---------------------XXX-X
1090  DATA .    X:::::::::XXXXXX-------XX---------------------XXXXX
1100  DATA .   X:::::::::XX:::X-----------------------------XX
1110  DATA .  X:::::::::XX::::X----------XXXXXXXXXXXXXXXXXX
1120  DATA .  X:::::XXXXX::::X----------X
1130  DATA .  X::::X---X:::::X---------X
1140  DATA .  X::::X-XX-X:::::X--------X
1150  DATA .  X::::X-XX-X:::::X--------X                  XXXX
1160  DATA .   X:::X-XX-X:::::X--------X                XX----XX
1170  DATA .   X::::X---X:::::X---------X              X--X-X---X
1180  DATA .    X::::XXX:::::::X---------X             XX--X--XXXX
1190  DATA .     X:::::::::::::X---------XXXXXXXX      XX---XX--X
1200  DATA .      XX:::::::::::::XX-------------X      X----X--XX
1210  DATA .        XX:::::::::::::XXXX---XXXXXX      X-----XXX
1220  DATA .          XXX:::::::::::::XXXXX          X-----X
1230  DATA .             XXXXXXXXXXXXX X:::X        X-----X
1240  DATA .                          X::::XX      X-----X
1250  DATA .                         X::::X--XXX X-----X
1260  DATA .                         X::::X-----XXX---X
1270  DATA .                        X:::X----------XXX
1280  DATA .                        X:::X--X----------XXX
1290  DATA .                       X:::X----X-----------XX
1300  DATA .                       X:::XX----X-------------X
1310  DATA .                      X:::X--X----X-------------X
1320  DATA .                      X::XX---X----X-------------X
1330  DATA .                     X::X X----X---XXXX----------X
1340  DATA .                     X::X X----XX-----X----------X
1350  DATA .                    X:::X X---X-----XX----------X
1360  DATA .                    X:::X  X---XXX---X---------X
1370  DATA .                   X:::X    XX---XXXX-------XXX
1380  DATA .                  X::::X     XXXX-------X-----X
1390  DATA .                 X:::::X          X------X-----X
1400  DATA .  XXXXXXXXXXXXXX:::::X           X------X------XXXXXXXXX
1410  DATA .         X:::::::::::X        XXXXXX-------XXXXXXXXX------X
1420  DATA .    XXXXX:::::::::::::X     X----------------------X--X--X
1430  DATA .      X:::::::::::::X     X-----------------X--X--XXXXXX
1440  DATA .XXXXXXXXXXXXXXXXXXX       XXXXXXXXXXXXXXXXXXXXXX
1450  REM ------------------------------ EINDE ----------------------
```

```
REM *** SNOOPY ************************************
REM *** UITGEZONDEN IN NOS-BASICODE 21-3-1982 ****
REM *** GEWIJZIGD EN AANGEPAST DOOR : ************
REM ********************* DE BONT CORNEEL *********
REM ********************* OOSTEINDE 10 ************
REM ***************** 2338 BAARLE - HERTOG *********
REM ********************************************
```

snoopy

change 16 color mode in 4 color mode

```
1     REM ##################################################
2     REM ##### CHANGE 16 COLOR MODE IN 4 COLOR MODE #####
3     REM ##### PROGRAM  BY  W.De Winter  82-04-27   #####
4     REM ##################################################
5     CLEAR 100:DIM TABLE%(15):GOTO 60
9     REM CONSTRUEER 4 COLOR DATA WOORDEN (SUBROUTINE)
10    ON COLORF% GOTO 30,40,50:RETURN
20    ON COLORB% GOTO 30,40,50:RETURN
30    LOWBYTE%=LOWBYTE% IOR (1 SHL BITPOSITION%):RETURN
40    HIGHBYTE%=HIGHBYTE% IOR (1 SHL BITPOSITION%):RETURN
50    HIGHBYTE%=HIGHBYTE% IOR (1 SHL BITPOSITION%):LOWBYTE%=LOWBYTE% IOR (1 SHL
BITPOSITION%):RETURN
55    REM TITEL + CONTROLE OP 16 COLOR MODE
60    PRINT CHR$(12):PRINT :PRINT " THIS PROGRAM CHANGES ANY 16 COLORMODE IN AN 4
 COLORMODE"
62    IF GETC<>32 THEN 62
65    REM CONTROLEER OF 16 KLEUREN MODE GEZET IS
70    SMODE%=PEEK(#9D):IF SMODE%=#FF OR SMODE%=2 OR SMODE%=3 OR SMODE%=6 OR SMOD
E%=7 OR SMODE%=#A OR SMODE%=#B THEN PRINT :PRINT " YOU ARE IN NO 16 COLOR MODE":
PRINT :END
75    REM ONDERZOEK WELK CONTROLE WOORD GEBRUIKT MOET WORDEN
80    IF SMODE%=1 THEN MBYTE%=3
82    IF SMODE%=5 THEN MBYTE%=#11
84    IF SMODE%=9 THEN MBYTE%=#20
90    REM VUL TAFEL OP DIE HET VERBAND BEPAALD TUSSEN DE KLEUR IN 16 KLEUREN MOD
E EN DE VIER KLEURENREGISTERS IN 4 KLEUREN MODE
95    COUNTER%=-1
100   PRINT :COUNTER%=COUNTER%+1:IF COUNTER%>15 THEN 140
110   PRINT " COLOR ";COUNTER%;" WORDT COLORREGISTER ";:INPUT A%
120   IF A%<4 THEN TABLE%(COUNTER%)=A%:GOTO 100
130   COUNTER%=COUNTER%-1:GOTO 100
135   REM SELECTEER JUISTE MODE
140   IF SMODE%=1 THEN MODE 1
142   IF SMODE%=5 THEN MODE 3
144   IF SMODE%=9 THEN MODE 5
145   REM VERANDER DE CONTROL WOORDEN VAN ALLE LIJNEN IN DE WAARDE VOOR 4 COLOR
MODE
150   FOR ADRESS%=#BFEF TO #BFEF-(XMAX/4+7)*(YMAX+1) STEP -(XMAX/4+7):POKE ADRES
S%,MBYTE%:NEXT
390   REM CONTROLEER DE DATA WOORDEN OP VOOR- EN ACHTERGROND KLEUR
391   REM IN REGEL 410 WORDT HET KLEUR REGISTER VOOR DE VOORGROND KLEUR BEPAALD
392   REM IDEM VOOR REGEL 420 MAAR DAN VOOR DE ACHTER GROND KLEUR
393   REM IN REGEL 430 WORDT DE BYTE DIE DE EIGENLIJKE DOTS BEVAT OPGEROEPEN
394   REM REGEL 440 BEPAALD WELKE BIT WE GAAN TESTEN EN AAN DE HAND VAN DEZE BIT

395   REM DIE OFWEL VOOR- DAN ACHTERGROND BEPAALD GAAN WE TWEE NIUWE DATA WOORDE
N SAMEN STELLEN
396   REM DIE DUS DE TEKENING VERANDEREN IN EEN 4 KLEUREN MODE
400   COUNTER%=0:OFFSET%=(XMAX/4+7)/2:FOR ADRESS%=#BFEF-2 TO #BFEF-(XMAX/4+7)*(Y
MAX+1) STEP -2:COUNTER%=COUNTER%+1:IF COUNTER%=OFFSET% THEN COUNTER%=0:NEXT:GOTO
 500
410   COLORF%=TABLE%((PEEK(ADRESS%-1) IAND #F0) SHR 4)
420   COLORB%=TABLE%(PEEK(ADRESS%-1) IAND #F)
430   SCREENBYTE%=PEEK(ADRESS%):HIGHBYTE%=0:LOWBYTE%=0
440   FOR BITPOSITION%=0 TO 7
450   IF SCREENBYTE% IAND (1 SHL BITPOSITION%)>0 THEN GOSUB 10:GOTO 470
460   GOSUB 20
470   NEXT
480   POKE ADRESS%,HIGHBYTE%:POKE ADRESS%-1,LOWBYTE%
490   NEXT
500   IF SMODE%=1 THEN POKE #9D,2
505   IF SMODE%=5 THEN POKE #9D,6
510   IF SMODE%=9 THEN POKE #9D,#A
520   PRINT CHR$(12):PRINT :PRINT " YOU ARE NOW IN 4 COLOR MODE  !!!!":END
```

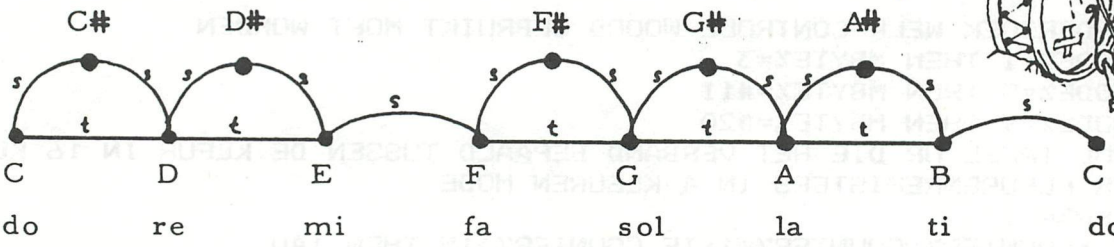
2.15.3

Generation of Music

This section describes how the DAI Graphical Sound Generator may be used to produce musical tones. The mathematical relationship between notes is described for the major scale. A procedure for the generation of scales is outlined.

2.15.3.1

Notes and the Major Scale



The above diagram shows the composition of the Major scale. The natural notes (e.g. C, D, E etc.) represent the white keys on a piano keyboard whilst the sharps (#) represent the black keys.

From C to C, over the scale, represents one OCTAVE. An OCTAVE represents a 2 : 1 frequency ratio. Thus to move from from middle C up 3 OCTAVEs requires an increase in frequency of 8 times (i.e. $2 * 2 * 2 = 2 \uparrow 3$).

To increase the pitch from C to C# requires that the pitch be increased by one SEMITONE. One SEMITONE represents an increase in frequency in the ratio $\sqrt[12]{2}$: 1.

Thus freq (C#) = freq (C) $* \sqrt[12]{2}$

To increase the pitch from C to D requires that the pitch be increased by one TONE. One TONE represents an increase of frequency in the ratio $\sqrt[6]{2}$: 1.

Thus freq (D) = freq (C) $* \sqrt[6]{2}$

It should be noted that

1 TONE = 2 SEMITONES

($\sqrt[6]{2} = \sqrt[12]{2} * \sqrt[12]{2}$)

2.15.3.2

Automatic generation of scales

It is possible to generate the notes in a scale or a number of scales by increasing or decreasing the frequency by the desired number of SEMITONES. Thus to move from C to F requires an increase of

2 TONES + 1 SEMITONE = 5 SEMITONES

Thus freq (F) = freq (C) $*\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2} *\sqrt[12]{2}$

$= \text{freq (C)} * (\sqrt[12]{2})^5$

A convenient reference note to take when generating scales is A = 44Ø Hz. By moving down the scale 9 SEMITONES C can be generated. Generation of a scale then merely involves a move of an integer number of SEMITONES up from C.

As an example consider a program to generate the irregular do, re, mi sequence:

```
1ØØ   CLEAR 1ØØØØ : SOUND OFF
11Ø   SEMI = 2.Ø ↑ (1.Ø/12.Ø)
12Ø   LOWC = 44Ø/(SEMI ↑ 9) : REM C 9 SEMITONES DOWN
13Ø   ENVELOPE Ø 16
14Ø   FOR N% = 1 TO 8
15Ø   READ D%
16Ø   FRQ = LOWC *(SEMI ↑ D%) : REM D% SEMITONES UP FROM C
17Ø   SOUND Ø Ø 15 Ø FREQ (FRQ)
18Ø   WAIT TIME 2Ø : SOUND OFF
19Ø   NEXT N%
2ØØ   STOP
21Ø   DATA Ø,2,4,5,7,9,11,12
```

Volume control of a particular sound generator is attained by using the appropriate TALK code followed by one byte giving the volume level between Ø → #F (i.e. 16 level).

Frequency control of a particular sound generator is given by the two bytes of data following the frequency TALK code. The frequency is obtained by specifying the number of 1/2 µS units in the same manner as for the SOUND command.

e.g. Determine the hexadecimal number required to give a frequency of 8ØØHz

?HEX$(FREQ(8ØØ)) gives 9C4.

Delay of up to 0.9 seconds is possible. The unit delay is 13.5 μS. The two bytes of data specify the number of unit delays required.

As TALK codes are interpretive codes relating to sound generation, they are much more restrictive than machine code. Should more sophisticated functions be needed, TALK has the facility to call a subroutine written in absolute machine code. Thus, facilities such as loop counts, etc. become possible with TALK. Two bytes of data specify the 16 bit destination address.

Note: The TALK interpreter, in order to execute TALK code statements quickly, uses very low level coding. One feature of the 8080 microprocessor is the way in which it collects double byte data from memory, collecting the lower value byte first and the higher value byte last.

This convention is observed for all double byte data in the TALK code format. Thus the 3 byte TALK command to call the subroutine at location #5ØBC would be #ØD (call routine), #BC (lower byte address), #5Ø (upper byte address).

Consider the following example of a TALK program:

Address	TALK code	DATA	FUNCTION
#AØØØ	Ø	#C4,9	Set channel Ø frequency to 8ØØHz (#9C4).
AØØ3	8	#F	Set channel Ø volume maximum.
AØØ5	# C	#FF, #FF	Set maximum delay (. 88 secs).
AØØ8	8	Ø	Set channel Ø volume minimum.
AØØA	2	#A, #1A	Set channel 1 frequency to 3ØØ Hz (#1AØA)
AØØD	9	#F	Set channel 1 volume maximum.
AØØF	# C	#FF, #FF	Set maximum delay (. 88 secs).
AØ12	9	Ø	Set channel 1 volume minimum.
AØ14	4	#2Ø, #4E	Set channel 2 frequency to 1ØØHz (#4E2Ø).
AØ17	#A	#F	Set channel 2 volume maximum.
AØ19	#C	#FF, #FF	Set maximum delay (. 88 secs).
AØ1C	#A	Ø	Set channel 2 volume minimum.
AØ1E	#ØD	#21, #AØ	Jump to machine code subroutine (#AØ21)

For the purposes of this example we will include a small machine code subroutine which will return to the TALK interpreter, causing it to start executing TALK instructions from address #AØØØ.

Address	Code	
#AØ21	#21, #ØØ, #AØ	LXI,H with AØØØ
AØ24	#C9	Return

The H,L register pair is loaded with the address of the next instruction for the TALK interpreter.

The following program will run the TALK statements given.

```
1Ø      CLEAR 1ØØØ:B%= #AØØØ
2Ø      READ A%:IF A% < Ø GOTO 4Ø
3Ø      POKE B%,A%:B%=B%+1:GOTO 2Ø
4Ø      TALK #AØØØ
5Ø      STOP
1ØØ     DATA Ø, #C4,9,8, #F, #C, #FF, #FF,8,Ø
11Ø     DATA 2, #A, #1A,9, #F, #C, #FF, #FF,9,Ø
12Ø     DATA 4, #2Ø, #4E, #A, #F, #C, #FF, #FF, #A,Ø
13Ø     DATA #ØD, #21, #AØ, #21,Ø, #AØ, #C9,-1
```

The next program will generate the Major scale over 5 OCTAVES. The values may be compared with the table which follows the program.

```
1ØØ     CLEAR 1ØØØ:SOUND OFF
11Ø     SEMI=2.Ø↑(1.Ø/12.Ø)
12Ø     LOWC=44Ø.Ø/(SEMI↑33.Ø)
13Ø     ENVELOPE Ø 15
14Ø     FOR N%=Ø TO 59
15Ø     IF N% MOD 12=Ø THEN RESTORE
16Ø     READ A$:FRQ=LOWC*(SEMI↑N%)
17Ø     SOUND Ø Ø 15 Ø FREQ(FRQ)
18Ø     PRINT A$, "FREQ=";FRQ
19Ø     WAIT TIME 2Ø:SOUND OFF
2ØØ     NEXT N%
21Ø     STOP
22Ø     DATA DO,DO# ,RE, RE# , MI, FA, FA# , SOL, SOL# , LA, LA# ,TI
```

Note	Freq. Hz Octave 1	Freq. Hz Octave 2	Freq. Hz Octave 3	Freq. Hz Octave 4	Freq. Hz Octave 5
DO	66	131	262	523	1Ø47
DO#	69	139	277	554	11Ø9
RE	73	147	294	587	1175
RE#	78	156	311	622	1245
MI	82	165	33Ø	659	1319
FA	87	175	349	698	1397
FA#	92	185	370	74Ø	148Ø
SOL	98	196	392	784	1568

SOL#	1Ø4	2Ø8	415	831	1661
LA	11Ø	22Ø	44Ø	88Ø	176Ø
LA#	117	233	466	932	1865
TI	123	247	494	988	1976

2.15.4

Synthesising Vocal Sound

Due to the flexibility of change in volume and frequency it is quite feasible to explore the possibilities of vocal sound generation. The BASIC of the DAI Personal Computer gives full control to the programmer who wishes to develop experimentally a burst of sounds that may result in vocal sounds. A further feature aiding experimentation is the TALK facility, which allows for fast bursts of sound to be programmed in a low level code.

2.15.4.1

TALK

EXAMPLES

```
TALK   #BØØØ
TALK    VARPTR(X(Ø))
```

TALK is both a command in DAI BASIC and a subsystem in its own right. The above TALK commands cause the TALK code interpreter to start to interpret the special low level TALK code at the address specified. The following table outlines the TALK codes:

Instruction Code	Bytes of Data	Function
#Ø	2	Frequency control, channel Ø
#2	2	" " " 1
#4	2	" " " 2
#8	1	Volume control, channel Ø
#9	1	" " " 1
#A	1	" " " 2
#B	1	" " , white noise generator
#C	2	Delay in 13 µS units.
#D	2	Call absolute machine code routine.
#FF	Ø	END TALK/RETURN TO BASIC.

more about talk & music

An alternative way of achieving the same result but with more safety, is to set up space on the HEAP by dimensioning an array and writing the TALK codes into it. Any addresses which need to be set up can be done with reference to position in the DATA table. The following program demonstrates this possibility.

```
10      CLEAR 1000:DIM Q%(100.0):B%=VARPTR(Q%(0.0))
20      READ A%:IF A%<0.0 GOTO 40
30      POKE B%,A%:B%=B%+1:GOTO 20
40      READ F%:IF F%<0 GOTO 50:READ S%:GOSUB 1000:GOTO 40
50      TALK VARPTR(Q%(0.0))
60      STOP
100     DATA 0,#C4,9,8,#F,#C,#FF,#FF,8,0
110     DATA 2,#A,#1A,9,#F,#C,#FF,#FF,9,0
120     DATA 4,#20,#4E,#A,#F,#C,#FF,#FF,#A,0
130     DATA #0D,0,0,#21,0,0,#C9,-1
140     DATA 32,34,35,1,-1
1000    J%=4:UF%=VARPTR(Q%(F%/J%))+(F% MOD 4.0)-1.0
1010    US%=VARPTR(Q%(S%/J%))+(S% MOD 4.0)-1.0
1020    POKE UF%,US% IAND #FF
1030    POKE UF%+1,US% SHR 8
1040    RETURN
```

floating point numbers

PART 2

3. Floating point getallen

3.1 Definitie en notatie van floating point getallen in het decimaal en binair talstelsel

Een floating point getal bestaat uit een geheel deel en een fractiedeel. Beide delen worden in de gewone notatie naast elkaar geschreven en zijn gescheiden door een punt. Het deel links van de punt is het gehele deel, terwijl de cijfers rechts ervan het fractiedeel vormen.

voorbeelden: in het decimaal talstelsel 2.75 - 0.678 567.0

in het binair talstelsel 1011.010 -101.011 0.011

Een floating point getal kan geschreven worden als de som van de producten van elk van zijn cijfers met een positieve of een negatieve macht van het grondtal

van het talstelsel waarin het floating point getal is uitgedrukt.

voorbeeld 1: in het decimaal stelsel

$$376.14 = 3 \times 10^2 + 7 \times 10^1 + 6 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}$$

voorbeeld 2: in het binair stelsel

$$1011.101 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$(= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125)$$

$$(= 11.625)$$

De plaats van de punt geeft de overgang aan tussen de positieve en de negatieve machten van het grondtal.

3.2 De wetenschappelijke notatievorm voor floating point getallen

Omdat in een computer niet met een onbeperkt aantal cijfers kan worden gewerkt, gebruikt men meestal de wetenschappelijke notatievorm om floating point getallen voor te stellen. In deze representatie van de floating point getallen onderscheidt men 3 delen:

a. een geheel deel
b. een fractiedeel
c. een(expliciet vermelde) macht van het grondtal van het talstelsel

Het voordeel van deze notatievorm is dat zeer grote en zeer kleine getallen met een minimum aantal cijfers kunnen geschreven worden.

voorbeeld: 17.15×10^9 i. p. v. 17150000000 in het decimaal stelsel

11.01×10^{101} i. p. v. 1101000 in het binair stelsel

3.3 De genormaliseerde wetenschappelijke notatievorm: definitie

Door de exponent van de macht van het grondtal op de juiste wijze aan te passen kan er steeds voor gezorgd worden dat het gehele deel in de wetenschappelijke notatievorm nul is.

voorbeeld: $17.15 \times 10^9 = 0.1715 \times 10^{11}$ in het decimaal stelsel

$11.01 \times 10^{101} = 0.1101 \times 10^{111}$ in het binair talstelsel

Indien het fractiedeel uit een vast aantal cijferposities bestaat, zal een floating point getal bijgevolg met de grootste nauwkeurigheid kunnen worden voorgesteld, indien geeist wordt dat het cijfer dat onmiddellijk na de punt volgt verschillend is van nul. Ook aan deze eis kan voldaan worden door de exponent van

het grondtal op een gepaste wijze te veranderen.

voorbeeld: $0.0023 \times 10^{4} = 0.23 \times 10^{2}$ (decimaal)

$0.0011 \times 10^{1010} = 0.11 \times 10^{1000}$ (binair)

Indien vorige twee voorwaarden bij de voorstelling van een floating point getal (in om 't even welk talstelsel) voldaan zijn, noemt men de notatie genormaliseerd. Samengevat kan gesteld worden: een wetenschappelijke notatie van een floating point getal is genormaliseerd dan en slechts dan als:

1. het gehele deel ervan nul is
2. het eerste cijfer van het fractiedeel verschillend is van nul; het enige getal dat op deze eis een inbreuk mag maken is het getal nul zelf.

3.4 Voorstelling in het geheugen van de binaire genormaliseerde wetenschappelijke notatievorm van een floating point getal

Rekening houdend met de definitie van een genormaliseerde wetenschappelijke notatie kan de standaardvorm hiervan in basis twee (binair 10 genoteerd) als volgt worden opgeschreven:

$$0.BBBBBBBBB \times 10^{bbbbb}$$

Hierin stellen de letters B en b bits (0 of 1) voor. Het aantal B's is afhankelijk van het vast aantal bits dat gebruikt wordt om de fractie voor te stellen. Verder in deze tekst is dit aantal 24. Het aantal b's bepaalt de maximale waarde van de exponent van het grondtal 2. Dit aantal is 7.

Daar de fractie uit 24 bits bestaat zal een nauwkeurigheid van $\frac{1}{2^{24}} \approx 0.000000059$ bekomen worden. Dit komt neer op een nauwkeurigheid van ongeveer 7 decimale cijfers.

De exponent van het grondtal wordt met 7 bits weergegeven in 2-complement notatie, d.w.z. hij varieert van:

$$0111111 = 2^{6} - 1 = 63$$

$$\text{tot} \quad 1000000 = -64$$

De grootste macht waarmee de fractie bijgevolg kan vermenigvuldigd worden is

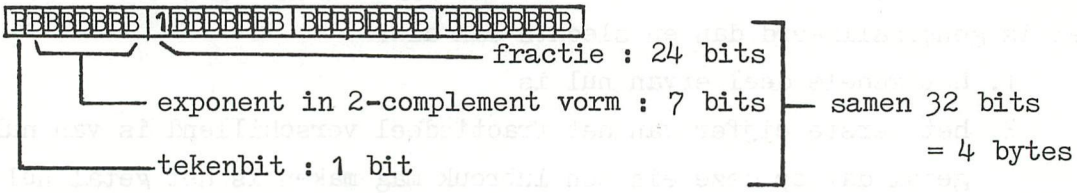
$$2^{63} \approx 9.2 \times 10^{18}$$

en de kleinste:

$$2^{-64} \approx 5.4 \times 10^{-20}$$

zodat kan gesteld worden dat de strikt positieve floating point getallen die expliciet kunnen worden voorgesteld gelegen zijn tussen 10^{-19} en 10^{19}.

De 7 exponentbits worden door de tekenbit tot één byte vervolledigd. Deze tekenbit is 0 voor positieve floating point getallen en 1 voor negatieve. De grenzen die hoger werden gegeven voor de strikt positieve floating point getallen (10^{-19} en 10^{19}) kunnen nu aangepast worden voor de negatieve: -10^{19} en -10^{-19}.
Deze teken-exponent-byte tesamen met de 3 fractie-bytes vormen de 4 bytes die gebruikt worden om floating point getallen voor te stellen. Ze zijn als volgt in het geheugen opgeslagen:



Daar deze voorstellingswijze genormaliseerd is, is de meest beduidende bit van de fractie - met uitzondering van het floating point getal 0.0 - steeds 1. Dit werd dan ook zo weergegeven in de schematische voorstelling. Daar het gehele deel nul is, zou het slechts een verlies aan geheugenruimte zijn indien ook dit gehele deel in de voorstellingswijze zou worden opgenomen. De punt (scheiding tussen geheel en fractie deel) is bijgevolg ook overbodig. Impliciet staat ze voor de tweede byte.

3.5 Binaire genormaliseerde wetenschappelijke notatievorm: enkele voorbeelden

Door middel van voorbeelden zullen de papier-en-potlood-algoritmen voor de verschillende gevallen behandeld worden.

Voorbeeld 1 Binaire representatie van het floating point getal 20.23

Deze voorstelling wordt in 3 fasen berekend:

1. binaire voorstelling van het gehele deel 20

   $$20 = 10100$$

2. binaire voorstelling van het fractiedeel: 0.23
   Hiertoe maken we volgende bedenking:

   $$0.23 = 2 \times 10^{-1} + 3 \times 10^{-2}$$

   We onderstellen nu dat de binaire equivalenten van de machten van 10 (tien) met negatieve exponenten gekend zijn met een nauwkeurigheid van 32 cijferbits en in een niet-genormaliseerde vorm. Deze binaire equivalenten staan in tabel 1 en werden berekend met programma 2. Gezien slechts 7 decimale cijfers nauwkeurig kunnen worden weergegeven, volstaat het deze tabel op te bouwen tot 10^{-7}.

Programma 2

```
5     CLEAR 1000
10    D%=10
20    FOR K=1.0 TO 7.0
25    DT%=2:B$=""
30    FOR N%=1 TO 32
40    Q%=DT%/D%
50    IF Q%=1 THEN DT%=DT%-D%
60    DT%=DT%*2
70    C$=MID$(STR$(Q%),1,1)
80    B$=B$+C$
90    NEXT
100   PRINT "BINAIR EQUIVALENT VAN";1.0/D%
110   FOR I=0.0 TO 3.0
120   PRINT MID$(B$,I*8,8);"  ";
130   NEXT
140   PRINT :PRINT :D%=D%*10
150   NEXT
```

Tabel 1 Tabel van de binaire equivalenten van 10^{-1} tot en met 10^{-7}

```
BINAIR EQUIVALENT VAN 0.1
00011001  10011001  10011001  10011001

BINAIR EQUIVALENT VAN 1E-2
00000010  10001111  01011100  00101000

BINAIR EQUIVALENT VAN 1E-3
00000000  01000001  10001001  00110111

BINAIR EQUIVALENT VAN 1E-4
00000000  00000110  10001101  10111000

BINAIR EQUIVALENT VAN 1E-5
00000000  00000000  10100111  11000101

BINAIR EQUIVALENT VAN 1E-6
00000000  00000000  00010000  11000110

BINAIR EQUIVALENT VAN 1E-7
00000000  00000000  00000001  10101101
```

De berekening van de binaire voorstelling van het fractiedeel gebeurt nu als volgt:

```
2 x 0.1 = 2 x 00011001 10011001 10011001 10011001
        =     00110011 00110011 00110011 00110010

3 x 0.01= 3 x 00000010 10001111 01011100 00101000
        =     00000111 10101110 00010100 01111000
```

Opgeteld geeft dit:

```
0.23 = 00110011 00110011 00110011 00110010
     + 00000111 10101110 00010100 01111000
       00111010 11100001 01000111 10101010
```

opmerking: gezien in de binaire representatie de notatie 0. niet gebruikt wordt, staan in het rechterlid uitsluitend cijfers van het fractiedeel.

De berekeningen van fase 1 en fase 2 kunnen nu als volgt schematisch worden samengebracht:

gehele deel	fractiedeel	teken en exponent
00010100	00111010 11100001 01000111 10101010	00000000

3. tijdens fase 3 wordt deze binaire wetenschappelijke notatie genormaliseerd. Dit gebeurt door alle bits van het gehele en het fractie deel zoveel plaatsen naar rechts op te schuiven tot het gehele deel nul wordt. Telkens 1 bit wordt opgeschoven, moet de exponent met 1 worden vermeerderd. Na deze fase bekomt men volgende genormaliseerde vorm:

gehele deel	fractiedeel	teken en exponent
00000000	10100001 11010111 00001010 00111101	00000101

Door de fractie te beperken tot 3 bytes - eventueel afronden naar boven vanuit de vierde byte - en de exponent vooraan te plaatsen, wordt de gewenste binaire notatie van het floating point getal 20.23 bekomen.

exponent	fractie
00000101	10100001 11010111 00001010

Voorbeeld 2: Binaire representatie van het floating point getal -20.23

Het enige verschil met het floating point getal uit het vorige voorbeeld is het minteken. Dit wordt gecodeerd in bit 7 van de teken-exponentbyte, zodat de binaire representatie van -20.23 is:

teken en exponent	fractiedeel
10000101	10100001 11010111 00001010

Voorbeeld 3: Binaire representatie van het floating point getal 0.012

fase 1: Vermits het gehele deel 0 is, is deze fase hier overbodig

fase 2:

$$0.012 = 1 \times 10^{-2} + 2 \times 10^{-3}$$

1×10^{-2}	=	00000010 10001111 01011100 00101000
2×10^{-3}	=	00000000 10000011 00010010 01101110
0.012	=	00000011 00010010 01101110 10010110

We bekomen alzo de voorstelling:

geheel deel	fractiedeel	teken en exponent
00000000	00000011 00010010 01101110 10010110	00000000

fase 3: om deze vorm te normaliseren moeten de bits van de fractie 6 plaatsen naar links worden verschoven. De exponent vermindert hierbij telkens met 1, zodat na de verschuiving deze de waarde -6 heeft. In 2-complement notatie met 7 bits wordt dit: 1111010.
Na verschuiving krijgen we volgende representatie:

geheel deel	fractiedeel	teken en exponent
00000000	11000100 10011011 10100101 10000000	01111010

Als we het gehele deel in de notatie weglaten,de fractie beperken tot 3 bytes - met afronding vanuit de vierde byte - en de exponent voor de fractie plaatsen, bekomen we:

teken en exponent	fractiedeel
01111010	11000100 10011011 10100110

Voorbeeld 4: Binaire representatie van het floating point getal - 0.012

Door codering van het minteken in de notatie van vorig voorbeeld bekomen we:
11111010 11000100 10011011 10100110 xxxxx

bingo

```
1     REM BINGOSPEL       PROGR: T.GROENEVELD LEEUWARDEN NEDERLAND
2     REM GEBRUIKSAANWIJZING
3     REM DIT SPEL MAAKT GEBRUIK VAN EEN DRUKKNOP, MET DAARAAN
4     REM PARRALLEL EEN WEERSTAND VAN 47K EN IS AANGESLOTEN OP PDL(2)
5     REM NA GEREEDMELDING OP DRUKKNOP DRUKKEN VOOR EERSTE GETAL.
6     REM VOOR ELK VOLGEND GETAL KORTSTONDIG OP DRUKKNOP DRUKKEN.
7     REM ALS ER BINGO GEROEPEN WORDT, KRUKKNOP NET ZOLANG INDRUKKEN
8     REM TOT HET BINGOVELD VERSCHIJNT MET DE GEWEESTE GETALLEN.
10    MODE 0:COLORT 8 0 8 0:CLEAR 1000:DIM G(76.0):L=0.0:PRINT CHR$(12):POKE #75
,32:COLORT 8 0 8 0
11    CURSOR 10,20:PRINT "##########################################"
12    CURSOR 10,19:PRINT "#                                        #"
13    CURSOR 10,18:PRINT "#          B   I   N   G   O             #"
14    CURSOR 10,17:PRINT "#          ================              #"
15    CURSOR 10,16:PRINT "#                                        #"
16    CURSOR 10,15:PRINT "##########################################"
17    CURSOR 10,14:PRINT "uitgevoerd met een == D A I computer =="
18    CURSOR 10,10:PRINT "DE COMPUTER IS NU BEZIG EEN WILLEKEURIG"
19    CURSOR 10,9:PRINT "-------------------------------------"
20    CURSOR 12,7:PRINT "BINGO GETALLEN BESTAND OP TE BOUWEN"
22    CURSOR 12,6:PRINT "-----------------------------------"
23    REM OPBOUWEN GETALLENBESTAND
25    R=RND(75.0)+1.0
26    ENVELOPE 1 15,9;0:SOUND 1 1 15 0 FREQ(RND(269.0)+31.0)
27    NOISE 1 15
30    FOR A=1.0 TO 75.0:IF G(A)<>R THEN NEXT A
40    IF A<76.0 AND G(A)=R THEN 25
50    IF G(I)=0.0 THEN G(I)=R
65    I=I+1.0:IF I=76.0 THEN 100
70    GOTO 25
80    REM GEREEDMELDING VAN GETALLENBESTAND
100   PRINT CHR$(12):COLORT 5 15 5 5:POKE #75,32:POKE #BA2D,95
110   SOUND OFF :CURSOR 0,12:PRINT "BINGO IS GEREED"
115   REM WACHTEN OP TOETSENBORD
120   REM R%=GETC:IF R%=0 THEN 120
130   REM WACHTEN OP PADDLE
140   P!=PDL(2):IF PDL(2)>10 THEN 140
160   L=0:COLORT 8 0 8 0:POKE #BA2D,122
162   REM GETALLENBESTAND EEN VOOR EEN LEEGLEZEN
163   MODE 1:COLORG 1 10 0 0:PRINT CHR$(12)
165   L=L+1.0:IF L=76 THEN 185
180   PRINT "            Getal is ==";G(L);" == Is er BINGO (J/N) ?"
181   FILL 5,5 65,55 2:GOSUB 300
182   SOUND 0 0 15 0 FREQ(800.0):WAIT TIME 5:SOUND OFF
184   REM WACHTEN MET GETC OP TOETSENBORD
185   REM G=GETC:IF G=0.0 THEN 185
186   REM IF G=74.0 THEN 194
188   REM WACHTEN MET GETC OP PADDLE
189   IF PDL(2)>10 THEN 189
190   WAIT TIME 100
191   IF PDL(2)<10 THEN 194
193   GOTO 165
194   REM GEGEVEN BINGOGETALLEN WEERGEVEN
195   MODE 0:PRINT CHR$(12)
196   IF L=76.0 THEN L=75
```

```
197   PRINT "     ==================================================="
198   PRINT "     # # # #   B   #   I   #   N   #   G   #   O  # # # #"
199   PRINT "     ==================================================="
200   FOR T=1 TO L
205   P=INT(G(T)-1.0)/15.0
210   CURSOR P*8.0+12,20-G(T)+P*15.0
230   PRINT G(T):NEXT T
240   CURSOR 5,4:PRINT "==================================================="
250   CURSOR 5,2:PRINT "GOEDE BINGO (J/N) ?"
254   CURSOR 23,2
255   R=GETC:IF R=0 THEN 255
260   IF R=74 THEN 600
262   FOR W!=500.0 TO 40.0 STEP -1.0
264   SOUND 0 0 15 2 FREQ(W!):NEXT:SOUND OFF
265   CURSOR 5,1:PRINT "VOOR VOLGENDE GETALLEN OP SPATIEBALK DRUKKEN !!!"
266   G!=GETC:IF G!=0.0 THEN 266
267   CURSOR 5,1:PRINT "                                                ":GOTO 1
63
300   N=30:X=INT(G(L)/10.0)
310   IF X=0.0 THEN 330
320   GOTO 470
325   REM BINGOGETALLEN GROOT WEERGEVEN
330   A=G(L)
340   ON A+1 GOTO 350,360,370,380,390,400,410,420,430,440
350   GOSUB 500:GOSUB 510:GOSUB 520:GOSUB 530:GOSUB 540:GOSUB 550:GOTO 450
360   GOSUB 540:GOSUB 550:GOTO 450
370   GOSUB 530:GOSUB 540:GOSUB 560:GOSUB 510:GOSUB 500:GOTO 450
380   GOSUB 530:GOSUB 540:GOSUB 550:GOSUB 500:GOSUB 560:GOTO 450
390   GOSUB 520:GOSUB 560:GOSUB 540:GOSUB 550:GOTO 450
400   GOSUB 530:GOSUB 520:GOSUB 560:GOSUB 550:GOSUB 500:GOTO 450
410   GOSUB 530:GOSUB 520:GOSUB 510:GOSUB 500:GOSUB 550:GOSUB 560:GOTO 450
420   GOSUB 530:GOSUB 540:GOSUB 550:GOTO 450
430   GOSUB 530:GOSUB 520:GOSUB 510:GOSUB 500:GOSUB 550:GOSUB 540:GOSUB 560:GOTO
 450
440   GOSUB 530:GOSUB 520:GOSUB 560:GOSUB 540:GOSUB 550:GOSUB 500:GOTO 450
450   IF X=0.0 THEN RETURN
460   A=FRAC(G(L)/10.0)*10.0+0.5:X=0:N=30:GOTO 340
470   A=X:N=0:GOTO 340
500   DRAW 10+N,10 30+N,10 8:RETURN
510   DRAW 10+N,10 10+N,30 8:RETURN
520   DRAW 10+N,30 10+N,50 8:RETURN
530   DRAW 10+N,50 30+N,50 8:RETURN
540   DRAW 30+N,50 30+N,30 8:RETURN
550   DRAW 30+N,30 30+N,10 8:RETURN
560   DRAW 30+N,30 10+N,30 8:RETURN
600   ENVELOPE 0 15
602   FOR J!=1.0 TO 15.0
605   SOUND 1 0 15 2 FREQ(800.0):WAIT TIME 5
610   SOUND 1 0 15 2 FREQ(600.0):WAIT TIME 5
615   NEXT:SOUND OFF
620   CURSOR 5,1:PRINT "WILT U NOG EEN KEER SPELEN (J/N) ?"
625   CURSOR 38,1
630   G!=GETC:WAIT TIME 3:IF G!=0.0 THEN 630
640   IF G!=78.0 THEN 660
650   GOTO 10
660   POKE #75,95:PRINT :END
```

grue + chrono

```
1     REM
2     REM GRUE + CHRONO - D'APRES DAINAMIC NO 8 .
3     REM ----------------------------------------
4     REM ECRIT PAR MOENS JACQUES - NOVEMBRE 1981
5     REM ----------------------------------------
10    MODE 0:PRINT CHR$(12):COLORT 12 0 12 12:PRINT :PRINT :PRINT
11    PRINT TAB(20);"GRUE + CHRONO":PRINT TAB(20);"-------------":PRINT
12    PRINT TAB(10);"LE BUT DU JEU EST DE RAMENER LE COLIS ":PRINT TAB(10);"SUR
LE QUAI SITUE A GAUCHE DE L'ECRAN"
13    PRINT TAB(9);"EN UN MINIMUM DE MOUVEMENTS ET DE TEMPS.":PRINT :PRINT TAB(1
5);"LES MOUVEMENTS SONT :"
14    PRINT TAB(15);CHR$(94);"  ";CHR$(140);"  ";CHR$(136);"  ";CHR$(137)
15    PRINT TAB(15);"1 = SERRAGE DES PINCES"
16    PRINT TAB(15);"2 = ECARTEMENT DES PINCES"
17    PRINT TAB(7);"POUR ACCELERER APPUYEZ SUR LA TOUCHE 'REPT'"
18    PRINT :PRINT :PRINT TAB(10);"POUR COMMENCER APPUYEZ SUR UNE TOUCHE"
19    IF GETC=0.0 THEN 19
20    GOTO 140
30    YM1%=YMAX-1:YP1%=YMAX-Y1%+1:DRAW X1%,YM1% X1%,YP1% C%
35    XM1%=X1%-D1%:YM1%=YMAX-Y1%-1:YM5%=YMAX-Y1%-5
40    DRAW XM1%,YM1% XM1%,YM5% C%
50    XP1%=X1%+D1%:DRAW XP1%,YM1% XP1%,YM5% C%
60    YM1%=YMAX-Y1%:DRAW X1%-10,YM1% X1%+10,YM1% C%
61    IF AC=0 THEN RETURN
62    FILL XG1,YG1 XD1,YD1 D%
70    RETURN
140   MODE 2:COLORG 0 1 10 5
150   DRAW 10,YMAX XMAX-10,YMAX 21
151   FILL 50,0 60,10 10
152   DRAW 0,30 29,30 5:DRAW 29,30 29,0 5
160   C%=21:X1%=11:Y1%=5:D1%=1:AC=0.0:PO=0.0
161   XG1=50.0:YG1=0.0:XD1=60.0:YD1=10.0:GOSUB 30
170   A%=GETC:S%=S%+1:IF A%=0 THEN 170
180   S%=S%+10:M%=M%+1:X2%=X1%:Y2%=Y1%:D2%=D1%
181   XG2=XG1:XD2=XD1:YG2=YG1:YD2=YD1
182   IF AC=1.0 THEN 190
186   IF SCRN(XG1-1,YD1)=0 THEN 190
187   IF SCRN(XD1+1,YD1)=0 THEN 190
188   IF SCRN(XG1+6,YD1+2)=0 THEN 190
189   AC=1.0
190   IF SCRN(10,31)<>10 AND SCRN(19,31)<>10 THEN 200
195   PO=1.0:GOTO 290
200   IF A%=50.0 THEN D2%=D1%+1:IF D2%<11 THEN 270
205   IF A%=49 THEN D2%=D1%-1:IF D2%>0 THEN 270
210   IF A%<>16 THEN 220
211   Y2%=Y1%-1
212   IF AC=1 THEN YG2=YG1+1.0:YD2=YD1+1.0
213   IF Y2%>1.0 THEN 260
220   IF A%<>17.0 THEN 230
221   Y2%=Y1%+1:Y5%=YMAX-Y2%-5
222   IF AC=0 THEN 226
225   YG2=YG1-1.0:YD2=YD1-1.0:IF YG2<=0.0 THEN 250
226   IF SCRN(XD2,YD2+1)=1 THEN 250
228   IF (Y2%<59.0) AND (SCRN(X1%+D1%,Y5%)=0) AND (SCRN(X1%-D1%,Y5%)=0) THEN 260
230   IF A%<>18.0 THEN 240
231   X2%=X1%-1
232   IF AC=1 THEN XG2=XG1-1.0:XD2=XD1-1.0
233   IF X2%>10.0 THEN 260
240   IF A%<>19.0 THEN 250
241   X2%=X1%+1
242   IF AC=1 THEN XG2=XG1+1.0:XD2=XD1+1.0
243   IF X2%<XMAX-10.0 THEN 260
250   SOUND 1 0 15 0 FREQ(100.0):WAIT TIME 10:SOUND OFF :GOTO 170
260   IF X2%<31.0 AND Y2%>28.0 THEN 250
261   IF X2%<40.0 AND Y2%>33.0 THEN 250
262   IF AC=0.0 THEN 265
263   IF SCRN(XG2-1,YG2)=5 THEN 250
```

240 × 528 resolution

```
PAGE 01

001                 ****************************************************
002                 * CE PROGRAMME PERMET DE METTRE L'ECRAN EN MODE 8
003                 * C'EST A DIRE EN 240 X 528 4 COULEURS.
004                 * IL EST DONNE AVEC SA REPLIQUE 'BASIC'
005                 * POUR CEUX QUI NE SONT PAS FAMILIARISE
006                 * AVEC LE LANGUAGE MACHINE.
007                 * POUR LANCER LE PROGRAMME IL SUFFIT DE FAIRE:
008                 * MODE 6:CALLM#300
009                 ****************************************************
010                 START   ORG   :300
011                 * LANGAGE MACHINE        LE MEME EN BASIC
012                 * --------------         ----------------
013                 * MODE6:CALLM#300       * 1   MODE6
014 0300 F5               PUSH  PSW         * 10  PPSW%=PSW%
015 0301 C5               PUSH  B           * 20  PB%=BC%
016 0302 D5               PUSH  D           * 30  PD%=DE%
017 0303 E5               PUSH  H           * 40  PH%=HL%
018 0304 21EFBF           LXI   H,:BFEF     * 50  HL%=#BFEF
019 0307 1EF0             MVI   E,240       * 60  E%=240
020 0309 1684       LOOP0 MVI   D,:84       * 70  D%=#84
021 030B 3630             MVI   M,:30       * 80  POKE HL%,#30
022 030D 2B               DCX   H           * 90  HL%=HL%-1
023 030E 3640             MVI   M,:40       * 100 POKE HL%,#40
024 0310 2B         LOOP1 DCX   H           * 110 HL%=HL%-1
025 0311 3600             MVI   M,:00       * 120 POKE HL%,#00
026 0313 15               DCR   D           * 130 D%=D%-1
027 0314 C21003           JNZ   LOOP1       * 140 IF D%<>0 THEN 110
028 0317 2B               DCX   H           * 150 HL%=HL%-1
029 0318 1D               DCR   E           * 160 E%=E%-1
030 0319 C20903           JNZ   LOOP0       * 170 IF E%<>0 THEN 70
031 031C E1               POP   H           * 180 HL%=PH%
032 031D D1               POP   D           * 190 DE%=PD%
033 031E C1               POP   B           * 200 BC%=PB%
034 031F F1               POP   PSW         * 210 PSW%=PPSW%
035 0320 C9               RET               * 220 RETURN
036 0321            FIN   END

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

FIN    0321   LOOP0  0309   LOOP1  0310   START  0000
```

grue + chrono

```
265   C%=20:D%=20:GOSUB 30:C%=21:D%=10:X1%=X2%:Y1%=Y2%:D1%=D2%
267   IF AC=1 THEN XG1=XG2:XD1=XD2:YG1=YG2:YD1=YD2
268   GOSUB 30:IF P=1.0 THEN 290
269   GOTO 170
270   IF (SCRN(D2%+X1%,YM5%)=0) AND (SCRN(X1%-D2%,YM5%)=0) THEN 262
271   GOTO 250
290   PRINT "TERMINE EN";M%;" MOUVEMENTS ET";S%/50.0;" SEC"
291   INPUT "UN NOUVEL ESSAI (O/N) ";R$
292   IF R$="O" THEN MODE O:PRINT CHR$(12):M%=O:S%=O:GOTO 20
293   IF R$="N" THEN 999
294   GOTO 291
999   PRINT :PRINT "AU REVOIR ET A BIENTOT J'ESPERE":PRINT :END
1000  REM PROGRAMME ECRIT PAR JACQUES MOENS
1010  REM ..................... CLOS FONTAINE DES DUCS,6
1020  REM ..................... B - 1310  LA HULPE
1030  REM ..................... TEL. 02/657.95.60.
```

```
##############################
#DAI TINY PASCAL#
#     COMPILER     #
##############################
```

PASCAL

1. INTRODUCTION

On tape following software modules are at your disposal:

A. PASCAL SYSTEM V3.1 consisting of:
- * an editor to generate "source code"
- * the Tiny Pascal compiler which translates the sourcecode into "P-code"
- * the 8080 translator which converts P-code into 8080 statements
- * the runtime modules

B. The Pascal Library which contains a set of procedures that can be merged with the PASCAL programs, enabling to make use of DAI's grafic features. If desired you can enlarge the contents of this liberary by yourself.

C. Convertor, which is a program that enables Pascal V1.X users to transform programs ,made with DAI's BASIC editor int the V3.1 compatible format.

D. A set of examples

It is important to remark that the "runtime modules" are not saved as a separate file on the tape. However, such a file is important when the object code of a Pascal program is directly loaded into memory.
To get the separate "runtime module" perform:

```
* UT
> R cr
> Z3 cr
> W2EC 9C4 RUNTIME-MODULES cr
```

Object files normally always start at address #9C5. To start an object file perform:

```
* UT
> Z3
> R RUNTIME-MODULES
> R OBJECTFILE
> G9C5
```

2. MEMORY MAP

```
adddress  2EC .. 9C4   RUNTIME MODULES
          9C5 .. 1FFF  OBJECT CODE
         2000 .. 2FFF  EDIT BUFFER (SOURCE CODE)
         3000 .. 612A  PASCAL SYSTEM V3.1
         612B .. 6FFF  RUNTIME STACK
         7000 .. 7FFF  P-CODE BUFFER
         8000 .. B34F  FREE
```

3. HOW GETTING STARTED.

Following actions have to be peformed:

```
*UT
>Z3
>R PASCAL SYSTEM V3.1
>G3000
```

Now a menu appearson the screen with follwing options

- <E> :enables you to create the PASCAL Source File. The normal DAI editing features are applicable. To return from edit mode, just type <break>.
- <D> :Deletes the contents of the editbuffer. To avoid undesired loose of a sourcefile, before typing 'D', '/' has to be entered first.
- <L> :Loads the editbuffer with a source file from tape
- <M> :Adds a "source file" to the contents of the edit buffer. This option must be selected when the already mentioned PASCAL LIBRARY has to be linked with a created source file. The position of the cursor determines the insertion of the file.
- <S> :Saves the contents of the editbuffer on tape.
- <C> :Compilation - the source file in the editbuffer is converted into p-code. When a syntax error has beeen detected, compilation is halted. On hitting the spacebar a return to the primery menu is accomplished. Possible errors can now be corrected. The compiler provides an option that when selected displays the mnemonics of the generated P-code statements.
- <T> :Translation of P-code into 8080 code. In principle, the translator polls for a set addresses. To avoid problems only respond by <return>. Note begin and end addresses mentioned by the translator
- <R> :loads object file from tape
- <W> :writes bject file to tape
- <G> :start execution of the object code. When no specific address is filled in , execution will start from #9C55 When the prgram finishes a returnal to utility mode is performed. Just accomplish G<return> to return to to the primery menu.
- <U> :return to utility.
- <B> :return to basic.
- <P> :-not displayed on the primery option menu. The listing of the source file is displayed on the screen and printed on hardcopy equipment.

4. TINY PASCAL STATEMENTS

Following keywords will be understood by the compiler

AND	END	PROC (:procedure)
ARRAY	FOR	READ
BEGIN	FUNC(:function)	REPEAT
CALL	IF	SHR (:shift right)
CASE	INTEGER	SHL (:shift left)
CONST	MEM	THEN
DIV	MOD	TO
DO	NOT	UNTIL
DOWNTO	OF	VAR
ELSE	OR	WHILE
		WRITE

Before starting creating Pascal programs, first read both appendixes.
.APPENDIX I gives the Tiny Pascal Syntax diagrams
.APPENDIX II gives a lot of examples on all thes keywords

PASCAL

```
   ***************
!  *DECLARATIONS*
   ***************

   CONSTANTS (SINGLE BYTE) ARE PERMITTED
!
  CONST LF=13;FF=12;


!
   TINY PASCAL ONLY UNDERSTANDS INTEGERS
   BY THIS  A VARIABLE CALLED "TEST" ALWAYS
   MUST BE DECLARED AS
!
   VAR TEST1,TEST2 : INTEGER;

! WHILE AN ARRAY OF ELEMENTS "VECTOR" AS
!
  VECTOR :ARRAY[300] OF INTEGER;
! REMARK THAT
  . THIS COMPILER ONLY UNDERSTANDS
    SINGLE DIMENTION DECLARATIONS
  . DECLARATIONS MUST BE PERFORMED IN FOLLOWING
    SEQUENCE
     a. CONSTANTS
     b. VARIABLES
     c. ARRAYS

 !
   ***************
   *INPUT^OUTPUT*
   ***************

   A VARIABLE MAY BE PRESENTED IN
    -INTEGER  FORMAT      :  <VARIABLE>%
    -HEX INTEGER FORMAT   :  <VARIABLE>#
    -ASCI                 :  <VARIABLE>


! VAR A IS SET TO 13,
  THIS PROGRAM RESPECTIVELY WRITES
   . HEX VALUE OF A (4 DIGITS)
   . DEC VALUE OF A
   . ASCII VALUE OF A (CR^LF)
!
   VAR A,B :INTEGER;
   BEGIN
    A:=13;
    WRITE('   ',A#,A,'   ',A%,A);
    WRITE('   ','that is all folks')
   END. ! '.' MARKS END OF PROOGRAM
```

```
! THIS PROGRAM DEMONSTATES
  THE USAGE OF THE 'READ COMMAND'
  THE PROGRAM ASKS FOR FOUR DECIMAL
  INPUTS AND DISPLAYS THE HEX REPRE-
  SENTATION
!
VAR I,A :INTEGER;
BEGIN
 I:= 0;
 REPEAT
    WRITE ('      ');
    READ (A%);! INTEGER INPUT!
    WRITE('        ',A#,13);!WRITE IN HEX FOMAT!
    I:=I+1
 UNTIL I=4
END.



 ! FOR ... DO EXAMPLE
   100 FUNNY A'S ARE GOING TO BE
   DISPLAYED ON THE SCREEN
   .. IN STEAT OF A SINGLE STATEMENT, ALSO
      A COMPOUND STATEMENT MAY BE USED
 !
 CONST FUNNYA=#40;
 VAR I :INTEGER;
 BEGIN
   FOR I:=0 TO 99 DO  !<-- NO  SEMI-COLOMN !
       WRITE(FUNNYA);
 END.
!
 ************************
 * REPETITIVE STATEMENTS*
 ************************

 FOR ... DO
 REPEAT .. UNTIL
 WHILE ..  DO

 FOR ... DO EXAMPLE
   100 FUNNY A'S ARE GOING TO BE
   DISPLAYED ON THE SCREEN
   .. IN STEAT OF A SINGLE STATEMENT, ALSO
      A COMPOUND STATEMENT MAY BE USED
 !
 CONST FUNNYA=#40;
 VAR I :INTEGER;
 BEGIN
   FOR I:=0 TO 99 DO  !<-- NO  SEMI-COLOMN !
       WRITE(FUNNYA);
 END.
!
 SAME EXAMPLE WITH THE DO..WHILE STATEMENT
!
 CONST FUNNYA=#40;
 VAR I :INTEGER;
 BEGIN
   I:=0;
   WHILE I<>100 DO!AS LONG AS I DIFFERS FROM 100!
   BEGIN
       WRITE(FUNNYA);
       I:=I+1; !INCREMENT I!
   END;
 END.
!
```

```
  SAME EXAMPLE WITH THE REPEAT UNTIL STATEMENT
!
  CONST FUNNYA=$40;
  VAR I :INTEGER;
  BEGIN
    I:=0;
    REPEAT !NO ";"!
        WRITE(FUNNYA);
        I:=I+1; !INCREMENT I!
    UNTIL I=99
  END.


!
  ************************
  * CONDITIONAL STATEMENTS *
  ************************

  THERE EXISTS :
     IF ...THEN (..ELSE)
     THE CASE STATEMENT


  THE IF STATEMENT
!
  VAR A,B :INTEGER;
  BEGIN
   B:=1;
   WHILE B DO  !ALSO MAY BE B=1!
    BEGIN
     READ(A);
     IF A='A' THEN WRITE('   HELLO BUDDY',13)
       ELSE IF A='B' THEN WRITE('   NICE WHETHER',13)
           ELSE IF A='S' THEN
              BEGIN
                 B:=0;
                 WRITE('   THIS IS ALL FOLKS',13)
              END
           ELSE WRITE('   YOU FOOL',13)
  END
  END.


!
  THE SAME FUNCTION PERFORMED BY THE
  CASE STATEMENT
!
  VAR A,B :INTEGER;
  BEGIN
   B:=1;
   WHILE B DO  !ALSO MAY BE B=1!
    BEGIN
     READ(A);
     CASE A OF
     'A' : WRITE('   HELLO BUDDY',13);
     'B' : WRITE('   NICE WHETHER',13);
     'S' : BEGIN
             B:=0;
             WRITE('   THIS IS ALL FOLKS',13)
           END
    ELSE WRITE('   YOU FOOL',13)
  END END
  END.
```



```
!
  ****************************
  * PROCEDURES AND FUNCTIONS *
  ****************************
```

PASCAL

```
  A. PROCEDURES

  FORMAT: PROC <NAME>(a1,a2,..an);
    WHERE a1.. an represent parameters: I/O TOWARDS
    THE PROCEDURE BODY.
    PARAMETERS NEED NOT TO BE DECLARED.
    A PROCEDURE IS CALLED BY ITS NAME.
    PROCEDURES MAY HAVE LOCAL VARIABLES

  FOLLOWING PROGRAM READS A SET OF INTEGERS AND DISPLAYS
  IT IN REVERSED ORDER
!
  VAR I,J :INTEGER;
      A: ARRAY[60] OF INTEGER;
  !PROCEDURE READSTRING!
  PROC READSTRING;
    VAR I :INTEGER;
    BEGIN
     I:=0;J:=0;READ(I%);
     WHILE I<>9999 DO
      BEGIN
        A[J]:=I;J:=J+1;READ(I%)
      END
    END;
  !PROCEDURE REVERSE STRING!
  PROC REVERSESTRING(X);
  VAR I:INTEGER;
  BEGIN
   FOR I:=X-1 DOWNTO 0
    DO WRITE(A[I]%,' ');
  END;
  ! PROCEDURE NEWLINE!
  PROC NEWLINE;
  BEGIN
   WRITE(13,'          ')
  END;
   ! MAIN!
  BEGIN
   NEWLINE;READSTRING;NEWLINE;REVERSESTRING(J)
  END.
!


   B. FUNCTONS

  FORMAT: FUNC <NAME>(a1,a2,..an);
    WHERE a1.. an represent parameters: I/O TOWARDS
    THE FUNCTION BODY.
    PARAMETERS NEED NOT TO BE DECLARED.
    A PROCEDURE IS ALWAYS CALLED IN AN EXPRESSION I.E.
      A:=<NAME>(a1,...an)
    FUNCTIONS MAY HAVE LOCAL VARIABLES

  FOLLOWING PROGRAM READS A SET OF 4 INTEGERS AND DISPLAYS
  THE GREATEST VALUE
!
```

```
VAR A1,A2,A3,A4,A,GREATEST :INTEGER;
FUNC MAX4(X1,X2,X3,X4);
 FUNC MAX2(X1,X2);
   BEGIN
    IF X1>X2 THEN MAX2:=X1
            ELSE MAX2:=X2
   END;
  BEGIN
  MAX4:=MAX2(MAX2(X1,X2),MAX2(X3,X4))
  END;
  BEGIN
   A:='Y';
   WHILE A='Y' DO
    BEGIN
     WRITE(13,'  A NEW GAME ?  [Y/N]');READ(A);
     WRITE(13,'  =======> input   ');
     IF A='Y' THEN
       BEGIN
        READ(A1%,A2%,A3%,A4%);
        GREATEST:=MAX4(A1,A2,A3,A4);
        WRITE('       ',GREATEST%);
       END
    END
   END.
```

```
**********************
* BOOLEAN EXPRESSIONS*
**********************
```

BOOLEAN EXPRESSIONS MAY BE USED IN
. THE "IF" STATEMENT
. THE "WHILE" STATEMENT
. THE "REPEAT UNTIL" STATEMENT

```
EX. IF A*(B+C)/D > E THEN ...
                 ELSE ...
```

FOLLOWING EXPRESSIONS ARE VALID
- `<` LESS THEN
- `>` GREATER THEN
- `<=` LESS OR EQUAL
- `>=` GREATER OR EQUAL
- `<>` DIFFERENT

```
************************
* ARITHMETIC OPERATORS *
************************
```

. + ADD	EX. A:=B+C;
. - SUBSTRACT	EX. A:=B-D;
. * MULTIPLY	EX. A:=A*B;
. DIV DIVIDE (INTEGER)	EX. A:=B DIV C
. MOD MODULUS	EX. A:=B MOD 8

FOLLOWING EXAMPLE DEMONSTATES THESE OPERATORS

```
VAR A,B,C :INTEGER;
BEGIN
 WRITE(13,'INPUT B,C :  ');READ(B%,C%);
 WRITE(13,'      B + C = ',(B+C)%,13,
       '      B - C = ',(B-C)%,13,
       '      B x C = ',(B*C)%,13,
       '      B : C = ',(B DIV C)%,13,
       '      B mod C=',(B MOD C)%,13)
END.
```

```
**********************
* BOOLEAN OPERATORS *
**********************
```

THESE ARE:

. OR	EX. A:=B OR C
. AND	EX. A:=B AND C
. NOT	EX. A:= NOT B
. SHL SHIFT LEFT	
. SHR SHIFT RIGHT	EX. A:=B SHR C

```
VAR I,J :INTEGER;
BEGIN
 WRITE(13,'INPUT 2 HEX VARIABLES   ');READ(I#,J#);
 WRITE(13,'    I OR J    = ',(I OR J)#,13,
       '    I AND J   = ',(I AND J)#,13,
       '    NOT I     = ',(NOT I)#,13,
       '    I SHL 2   = ',(I SHL 2)#,13,
       '    J SHR 2   = ',(J SHR 2)#,13)
END.
```

```
**************************
* PEEK AND POOK AND CALLM*
**************************
```

```
. PEEK :  A:=MEM #131;
. POKE :  MEM #131:=2;
. CALLM:  CALL(#D6BE)
```

PASCAL

```
! Eratosthenes Sieve Prime Number Program in PASCAL !

VAR  FLAGS : ARRAY[1000] OF INTEGER;
     SIZE,I,PRIME,K,COUNT : INTEGER;

BEGIN
WRITE(12,' FROM 1 TILL  '); READ(SIZE%);
SIZE := SIZE DIV 2; WRITE(13,13,'  2     '); COUNT := 1;
FOR I := 0 TO SIZE DO FLAGS[I] := 1;
FOR I := 0 TO SIZE DO
     IF FLAGS[I] THEN BEGIN
                PRIME := I+I+3;  K := I + PRIME;
                WHILE K <= SIZE DO
                   BEGIN FLAGS[K]:=0; K:=K + PRIME END;
                COUNT := COUNT + 1; WRITE(PRIME%,' ');
                IF PRIME <   10 THEN WRITE(' ');
                IF PRIME <  100 THEN WRITE(' ');
                IF PRIME < 1000 THEN WRITE(' ');
                IF COUNT MOD 11 = 0 THEN WRITE(13,'  ')
                END;
WRITE(13,13,' ',COUNT%,' PRIME NUMBERS GENERATED')
END.


! PROGRAM TOWERS OF HANOI !

VAR I, X, Y, N, NDISKS, TIME : INTEGER;
    D : ARRAY[33]      OF      INTEGER;
    Z : ARRAY[ 2]      OF      INTEGER;

PROC MODE(M);
   BEGIN
   MEM[#7D0]:=M; CALL(#7FD)
   END;

PROC COLORG(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#82B)
   END;

PROC DRAW(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#886)
   END;

FUNC XMAX;
   BEGIN
   MEM[#7D2]:=0; MEM[#7D5]:=0; MEM[#7D6]:=0;
   CALL(#912); XMAX:=256*MEM[#7D4]+MEM[#7D3]
   END;

PROC WAITTIME(T);
   BEGIN
   MEM[#1BE]:=T AND 255; MEM[#1BF]:=T SHR 8;
   REPEAT UNTIL (MEM[#1BE]=0) AND (MEM[#1BF]=0)
   END;
```

PASCAL

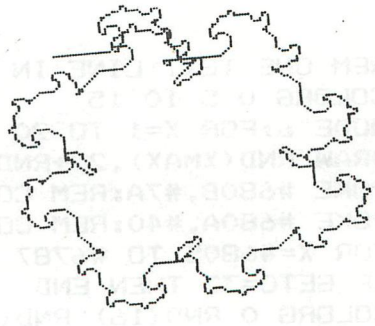
```
PROC HANOI(N,F,T,C);

   PROC MOVEDISK(F,T) ! LOCAL declared within hanoi !;
      VAR X1, X2, Y, H1, H2 : INTEGER;
      BEGIN ! procedure body of movedisk !
      WAITTIME(TIME);
      X1:=D[Z[F]+F*NDISKS]+F*24;
      X2:=11+F*24; Y:=Z[F]*4;
      DRAW(X1,Y,X2,Y,7);              ! whipe out    !
      DRAW(X2+2,Y,48*F+24-X1,Y,7);    ! "from" disk  !
      X1:=D[Z[F]+F*NDISKS]+T*24;
      X2:=11+T*24; Y:=Z[T]*4+4;
      DRAW(X1,Y,X2,Y,1);              ! draw         !
      DRAW(X2+2,Y,48*T+24-X1,Y,1);    ! "to" disk    !
      Z[T]:=Z[T]+1;
      H1:=Z[T]+T*NDISKS; H2:=Z[F]+F*NDISKS;
      D[H1]:=D[H2]; Z[F]:=Z[F]-1
      END ! movedisk !;

   BEGIN ! procedure body of hanoi !
   IF N=1 THEN MOVEDISK(F,T)
          ELSE BEGIN
               HANOI(N-1,F,C,T);      ! RECURSIVE def. !
               MOVEDISK(F,T);
               HANOI(N-1,C,T,F)       !        "       !
               END ! else !
   END ! hanoi !;

BEGIN ! *** main program *** !
MODE(3); COLORG(7,4,5,1);
REPEAT
   WRITE(12,'   ENTER NUMBER OF DISKS (1 - 11) : ');
   READ(NDISKS%)
   UNTIL (NDISKS>0) AND (NDISKS<12);
DRAW(0,0,XMAX,0,4);
FOR I:=1 TO 3 DO DRAW(I*24-12,0,I*24-12,60,5);
FOR X:=1 TO NDISKS DO
   BEGIN
   Y:=4*X; D[X]:=X;
   DRAW(X,Y,11,Y,1); DRAW(13,Y,24-X,Y,1)
   END ! x-loop !;
Z[0]:=NDISKS; Z[1]:=0; Z[2]:=0;
WRITE(13,'   ENTER TIME BETWEEN MOVES ');
WRITE('(UNITS OF 1/50 SEC) : '); READ(TIME%);
WRITE(13,'   PRESS ANY KEY TO START');
READ(X); WRITE(12);
HANOI(NDISKS,0,2,1);
N:=1; FOR I:=1 TO NDISKS DO N:=2*N; N:=N-1;
WRITE(13,13,'   TOTAL NUMBER OF MOVES MADE = ',N%)
END ! main !.
```

```
10    REM ONE TEXT LINE IN MODE 5/6
20    COLORG 0 5 10 15
30    MODE 6:FOR X=1 TO 20
35    DRAW RND(XMAX),20+RND(200) RND(XMAX),20+RND(200) 21+RND(3):NEXT
40    POKE #680B,#7A:REM CONTROL BYTE
50    POKE #680A,#40:REM COLOR   BYTE
60    FOR X=#6805 TO #6787 STEP -2:POKE X,#40+RND(26):POKE X-3,0:NEXT
70    IF GETC=32 THEN END
80    COLORG 0 RND(16) RND(16) RND(16):GOTO 60
```

PASCAL

```
!PROGRAM DECODE!
CONST STOP=#FF; EXIT=#3000;
VAR STADR, INDX, NUM: INTEGER;
PROC CRLF;
  BEGIN WRITE (13) END;
PROC FMT (VAL, LEN);
  VAR I, J: INTEGER;
  BEGIN !FMT!
    J:=VAL;
    FOR I:=2 + (J=0) TO LEN DO
      IF J=0 THEN WRITE (32) ELSE J:=J DIV 10;
    IF J>9 THEN WRITE ('&');
    WRITE (VAL%)
  END !FMT!;
BEGIN! MAIN!
  WRITE (12,'START DECODING AT ');READ (STADR#);CRLF;
  CRLF;INDX:=0;
  WHILE (STADR < #7FFF ) AND (MEM[STADR] <> STOP ) DO
    BEGIN FMT(INDX, 10);WRITE ('   ');INDX:=INDX+1;
      NUM:=MEM[STADR];
      CASE NUM OF
      0      : WRITE ('LIT');
      1      : WRITE ('OPR');
      2,#12  : WRITE ('LOD');
      3,#13  : WRITE ('STO');
      4      : WRITE ('CAL');
      5      : WRITE ('INT');
      6      : WRITE ('JMP');
      7      : WRITE ('JPC');
      8      : WRITE ('CSP')
      ELSE BEGIN WRITE ('ILL'); MEM[STADR]:= STOP END
      END;  !CASE!
      IF (NUM=#12) OR (NUM=#13) THEN WRITE ('X')
                                ELSE WRITE (32);WRITE (32);
      WRITE (MEM[STADR+1]%,',');
      NUM:=MEM[STADR+3] SHL 8 + MEM[STADR+2];
      WRITE (NUM%);CRLF;
      IF INDX MOD 15 = 0 THEN
        BEGIN READ (NUM);IF NUM='S' THEN CALL(EXIT) END;
      IF MEM[STADR] <> STOP THEN STADR:= STADR + 4;
    END; !WHILE!
END.!MAIN!
```

```
10    REM DRAGON KURVE
15    REM NAAR DRAGON CURVE
16    REM H4 RECURSION AND ITTERATION  BOEK : LISP DOOR P.H  WINSTON
20    REM RECURSIEF GEDEFINIEERDE FIGUUR
25    CLEAR 3000
30    DIM RETST(20.0),ST(60.0):SP=0.0:RETSP=0.0
40    ANGLE=0.0
45    MODE 6:COLORG 8 10 13 15:MIDX=XMAX/2.0:MIDY=YMAX/2.0
50    LENGTH=12.0
60    P=1.0:GOTO 1000:REM CALL FUNCTION
70    IF LENGTH>1.0 THEN X=MIDX+LENGTH*COS(ANGLE):Y=MIDY+LENGTH*SIN(ANGLE):DRAW
X,Y MIDX,MIDY 21:MIDX=X:MIDY=Y:GOTO 90
80    GOTO 2000:REM FUNCTION RETURN
90    LENGTH=LENGTH/SQR(2.0):ANGLE=ANGLE+SIGN*PI/4.0:SIGN=1.0
100   P=2.0:GOTO 1000:REM CALL FUNCTION
110   LENGTH=LENGTH/SQR(2.0):ANGLE=ANGLE-SIGN*PI/4.0:SIGN=-1.0
120   P=3.0:GOTO 1000:REM CALL FUNCTION
130   GOTO 2000:REM FUNCTION RETURN
1000  REM ======= CALL FUNCTION ======
1005  ST(SP)=SIGN:SP=SP+1.0
1010  ST(SP)=LENGTH:SP=SP+1.0
1020  ST(SP)=ANGLE:SP=SP+1.0
1030  RETST(RETSP)=P:RETSP=RETSP+1.0
1040  GOTO 70
2000  REM ======= RETURN FUNCTION  =====
2010  IF RETSP=0.0 GOTO 50
2030  SP=SP-1.0:ANGLE=ST(SP)
2040  SP=SP-1.0:LENGTH=ST(SP)
2045  SP=SP-1.0:SIGN=ST(SP)
2050  RETSP=RETSP-1.0
2060  ON RETST(RETSP) GOTO 70,110,130
```



PASCAL

```
PROC COLORT(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#7DE)
   END;

PROC MODE(M);
   BEGIN
   MEM[#7D0]:=M; CALL(#7FD)
   END;

PROC COLORG(C1,C2,C3,C4);
   BEGIN
   MEM[#7D0]:=C1; MEM[#7D1]:=C2; MEM[#7D2]:=C3;
   MEM[#7D3]:=C4; CALL(#82B)
   END;

PROC DOT(X,Y,C);
   BEGIN
   MEM[#7D5]:=X AND 255; MEM[#7D6]:=X SHR 8;
   MEM[#7D2]:=Y; MEM[#7D0]:=C; CALL(#84A)
   END;

PROC DRAW(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#886)
   END;

PROC FILL(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D3]:=X1 AND 255; MEM[#7D4]:=X1 SHR 8;
   MEM[#7D5]:=X2 AND 255; MEM[#7D6]:=X2 SHR 8;
   MEM[#7D1]:=Y1; MEM[#7D2]:=Y2; MEM[#7D0]:=C;
   CALL(#8CC)
   END;

FUNC SCRN(X,Y);
   BEGIN
   MEM[#7D5]:=X AND 255; MEM[#7D6]:=X SHR 8;
   MEM[#7D2]:=Y; CALL(#912); SCRN:=MEM[#7D0]
   END;

FUNC XMAX;
   BEGIN
   MEM [#7D2]:=0;MEM [#7D5]:=0;MEM[#7D6]:=0;
   CALL(#912); XMAX:=256*MEM[#7D4]+MEM[#7D3]
   END;

FUNC YMAX;
   BEGIN
   MEM [#7D2]:=0;MEM [#7D5]:=0;MEM[#7D6]:=0;
   CALL(#912); YMAX:=MEM[#7D1]
   END;

FUNC CURX;
   BEGIN
   CALL(#938); CURX:=MEM[#7D5]
   END;

FUNC CURY;
   BEGIN
   CALL(#938); CURY:=MEM[#7D6]
   END;

PROC CURSOR(X,Y);
   BEGIN
   MEM[#7D5]:=X; MEM[#7D6]:=Y; CALL(#93E)
   END;

FUNC PDL(I);
   BEGIN
   MEM[#7D0]:=I; CALL(#94F); PDL:=MEM[#D8]
   END;

PROC WAITTIME(T);
   BEGIN
   MEM[#1BE]:=T AND 255; MEM[#1BF]:=T SHR 8;
   REPEAT UNTIL (MEM[#1BE]=0) AND (MEM[#1BF]=0)
   END;
```

DAI-RS232

Volgende ervaringen had ik (als DAI-user-beginner) bij het aansluiten van een printer...

DAI RS-232 (rev.5)

Bij het aansluiten van een EPSON MX-82 op mijn 2 maand jonge DAI (via RS-232-parallel interface), bleek een en ander niet naar behoren te funktioneren: het listen op de DAI liep rustig verder zonder op de printer te wachten.
Vermits printer en interface bij aankoop uitgetest waren, belandde de bewuste DAI spoedig op de operatie-tafel voor een inwendig onderzoek.
Bijgaande figuren tonen print-layout en schema van de RS-232 DTR ingang (main-board rev.5).

INTERRUPT: hangt de 56k weerstand bij vorige revisies niet aan massa ipv aan +5V ?

Uit metingen bleek dat bij een 'NOT READY' signaal (0.13 V), de spanning op ingangspin 10 van ic 74LS373 (latch) 0.95 V bedroeg wat te hoog is (volgens de ic-specs moet V(IL) < 0.8 V, low-level input voltage).
De spanningsval over de 3k3 weerstand is veroorzaakt door de stroom via de 56k weerstand (0.072 mA) en de 'low-level input current' van het ic (0.176 mA) die nog ruim binnen de ic-specs valt (I(IL) < -0.4 mA) .
Door de 3k3 R te verlagen tot 2k (met parallelweerstand van 5k6) was de patient weer in uitstekende gezondheid.

EPSON MX-82 met VERBORGEN TALENTEN

80 karakters ipv de in de manual aangekondigde 96 spoorden mij aan tot een tweede operatie: bij inwendig nazicht stond DIP-switch 1-5 in de OFF-stand.
Over de funktie hiervan zwijgt de manual in alle talen en vermeldt alleen:
'Never set this pin to the OFF position, always leave it in the ON position'.
En inderdaad, in de ON-stand krijg je de volle 96 karakterbreedte, in de OFF-stand slechts 80 (+ 16 marge achteraan).

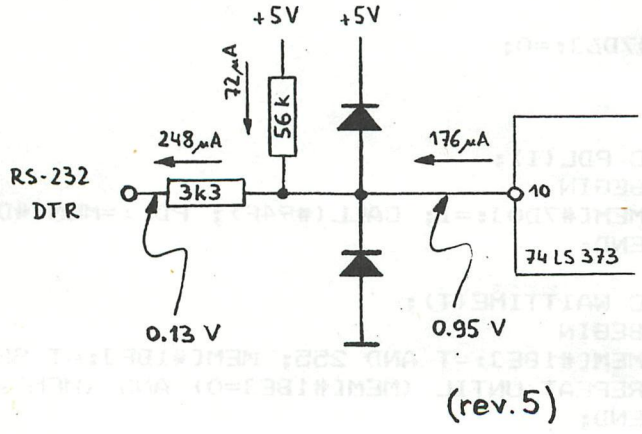
Tot slot nog een happy end !
Bij een vergelijking met de control-codes van de MX-100 waren 3 hiervan niet terug te vinden in de MX-82 manual: uitproberen dus en... met positief resultaat !

ESC G	double print (advance paper 1/216'' and repeat line)
ESC H	cancels ESC G
ESC M	elite print (letterbreedte tussen normal en condensed)

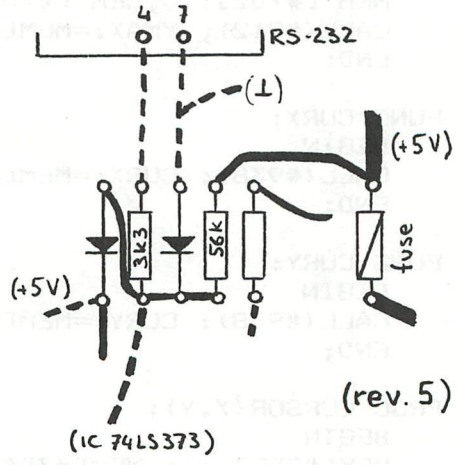
Deze codes staan dus vanaf nu ook in mijn MX-82 manual.

Herman MOEYS

3200 KESSEL-LO (LEUVEN)



(rev.5)



(rev.5)

machinetaal in een rem-statement

(Mijn eerste "eigen" machinetaal programma)

In MC 4 (Die Mikrocomputer Zeitschrift nov/dec 1981) wordt op de mogelijkheid gewezen om (korte) stukje machinetaal onder te brengen in een basic REM regel
Voordeel: Alles wordt in een keer geladen op de gewone manier van een basic programma.

Nadeel: Die stonden er niet bij maar ....

Mogelijkheden: Die komen in het navolgende aan de orde.

① Hoe weet je waar de gebruikte REM regel in het geheugen zit?

b.v. 10 REM *****

Op de adressen $2BF, $2A0 vinden we EC, $3 dat wil zeggen dat het begin van de basic text te vinden is op adres $3EC. We vinden daar:

```
                                                    <-- +5 ..
$3E0  FF FF FF FF FF FF FF FF FF FF 7F FF $09 $00 $0A $8F

      ...-->
$3F0  $05 2A 2A 2A 2A 2A $00 $00 FF FF FF FF FF FF FF FF
```

7F FF : einde van de heap (zie DATAMANIC nr 7 blz 188)
$09 : de komende basic regel bevat 9 ~~tekens~~ bytes van $00 t/m 2A
$00 $0A : regelnummer 10
8F : code voor REM
$05 : na de REM komen 5 tekens
2A....2A : 5 maal asterix

② In plaats van 5 maal 2A kunnen we ook 5 andere bytes kiezen die een machinetaalprogramma vormen. Dit programma kan dan aangeroepen worden d.m.v. CALLM #3F1 mits aan twee voorwaarden is voldaan:

- het programma begint met 10 REM ....
- de text begint op adres $3EC; dit is zeker zo na power on of hard reset.
  voorafgaande aan de eerste basic regel mag geen CLEAR gegeven worden

In de listing bevat de eerste regel een reeks willekeurige karakters. Soms is de regel niet meer kompleet. In het stukje machinetaal zit dan b.v. $C; bij het listen wordt dan het scherm schoongemaakt.

③ Een voorbeeld

Het volgende programma zet alle karakters op het scherm van CHR$(#FF) aftellend tot CHR$(#0C) = CHR$(12) ⇐ scherm schoon;

```
1   03F1   C5       PUSH B        } inhoud v.
2   03F2   D5       PUSH D        } registers bewaren op stack
3   03F3   E5       PUSH H        }
4   03F4   F5       PUSH PSW      }
5   03F5   06 0C    MVI  B,0C     CHR$(12) in B
6   03F7   3E FF    MVI  A,FF     #FF in A
7   03F9   CD 60 DD CALL : BD60   CHR$(<A>) → scherm ←...
8   03FC   3D       DCR  A        <A> ← <A> - 1
9   03FD   B8       CMP  B        } als <A> - <B> ≠ 0 dan
A   03FE   C2 F9 03 JNZ           } naar 03F9 ------
B   04 01  F1       POP  PSW      }
C   04 02  E1       POP  H        } register van stack halen
D   04 03  D1       POP  D        } inhoud
E   04 04  C1       POP  B        }
F   04 05  C9       RET
```

Het programma bevat 21 bytes. Daarom begint het basic programma met
```
10 REM *** 21 x asterix ***
20 CALLM #3F1
30 END
```

Met het Substitute kommando worden in UT op adres 03F1 t/m 0405 alle 2A bytes vervangen door de programma bytes.

Het geheel kan nu als basic gesaved en geladen worden.

Een REM statement kan tot 122 tekens bevatten; het machinetaal programma kan dus een overeenkomstige maximale lengte hebben (zonder al te grote kunstgrepen)

(4) De tweede voorwaarde genoemd onder punt (2) (start of text at 03EC) kan problemen geven.
Als door een CLEAR de heap ruimte groter is gemaakt dan #100 bytes, dan zal het begin van de text buffer zijn opgeschoven naar een hoger adres.
Dit adres is weer te vinden op #29F en #3A0 deze bevatten resp het lage en hoge aangepaste adres byte van het begin van de text buffer.

Het voorbeeld zou nu als volgt aangepast kunnen worden voor eventuele heap vergroting:

```
10 REM ----------
20 ADRES = PEEK(#3A0)*256+PEEK(#29F)+5
30 CALLM ADRES
40 END
```

We kijken dan eerst op #29F en #3A0 waar de text buffer begint. Door hij het begin adres 5 op te tellen (zie (1)) hebben we het begin adres van het machinetaal programma!

⑤ Helaas, als we een en ander proberen met ons voorbeeld programma dan loopt de zaak vast, zodat alleen het bekende witte knopje (de rode mag ook) uitkomst biedt.

In het machinetaal programma zit een sprong terug van regel A naar regel 7.
In de sprong instruktie wordt het adres $03FC vermeldt.
Maar dat klopt niet meer als de text buffer en daarmee het machinetaalprogramma naar hogere adressen wordt verplaatst.

Een eenvoudige oplossing hiervoor zou zijn een instruktie in de vorm: springterug, aantal adressen

Zo'n relatieve sprong instruktie kent de 8080A helaas niet!

En dan zit je als beginner met machinetaal goed vast
Na een paar telefoontjes krijg ik de volgende tip van Freddy de Raat:

- bereken het adres waarnaar je terug springt en zet dat in reg's H, L
- zet daarna de inhoud van H,L op stack: PUSH H
- komt nu de sprong voorwaarde bv een test op nul, dan kun je springen d.mv RETURN NON ZERO
- of verder gaan met POP H waardoor de stack weer op zijn oude hoogte terug is.

De truc is dat de berekening van het sprong adres uitgaat van het adres dat op #29F en #2A$ als wijzer naar de text buffer staat.
Omdat deze wijzer wordt aangepast bij een CLEAR, wordt het sprong adres nu ook aangepast.

⑥ Voorbeeld

1	$3F1	C5	PUSH B	
2	$3F2	D5	PUSH D	
3	$3F3	E5	PUSH H	
4	$3F4	F5	PUSH PSW	
5	$3F5	2A 9F $2	LHLD $29F	} berekening sprong adres
6	$3F8	11 14 $$	LXI D $$14	
7	$3FB	19	DAD D	
8	$3FC	$6 $C	MVI B, $C	
9	$3FE	3E FF	MVI A, FF	
A	$4$$	CD 6$ DD	CALL DD$6	
B	$4$3	3D	DCR A	
C	$4$4	B8	CMP B	
D	$4$5	E5	PUSH H	} relatieve sprong
E	$4$6	C$	RNZ	
F	$4$7	E1	POP H	
10	$4$8	F1 E1 D1 C1	POP ALL	
11	4C	C9	RET	

Toelichting

Regel nr.

5 Startadres van text buffer wordt gelezen op #2gF. #2Aϕ en geplaatst in regs H, L

6 Het adres waar naar gesprongen wordt ligt #4ϕϕ - #3EC = #14 adressen verder dan het start adres van de text buffer

daarom ϕϕ #, 14 in de regs D, E

7 na optellen staat het resultaat (dus het sprongadres) in de regs H, L

D Het sprong adres wordt vanuit H, L op stack gezet.

E Is <A> - <B> ≠ϕ dan return naar adres dat bovenaan op stack staat

Wordt nu voor het laden een CLEAR gegeven of is de HEAP door een voorafgaand programma vergroot, dan loopt het prospamma nu wel.

En dan ben je zo blij dat het werkt, dat je alles nog eens op-schrijft zoals het voor al niet te vergeten.

Mogelijk heeft een ander er dan ook nog wat aan.

TOEGIFT :

Groetjes Thijs Berke
Vlissingen

```
1ϕ   MODE 6: COLOR 4 ϕ 5 ϕ ϕ
15   FOR SH = ϕ TO 5ϕ STEP 2
2ϕ   V = ϕ : Y = 25 - SH
25   FOR X = ϕ TO XMAX - SH
3ϕ     A = -1ϕ * Y
4ϕ     V = V + A * ϕ.ϕ2          } "snelle sinus"   1ϕ regelt de periode
5ϕ     Y = Y + V * ϕ.ϕ2
6ϕ     DOT X+SH, 75+Y+2*SH  5
7ϕ   NEXT X: NEXT SH
1ϕϕ  GOTϕ 1ϕϕ
```

Probeer ter vergelijking dit programma eens met SIN !

TEKENS NAAR SCHERM

REM + MLP + REL. JMP

HEX DUMP

```
03 0E 00  FF FF FF FF FF FF FF FF FF FF 7F FF 20 00 00 0A A9

03 0F 00  1C C5 D5 E5 F5 2A 9F 02 11 14 00 19 06 0C 3E FF

04 00 0C  CD 60 DD 3D B8 E5 C0 E1 F1 E1 D1 C1 C9 21 00 14

04 01 00  A5 40 06 BF A0 A0 A3 20 17 15 00 00 02 A0 14 00

04 20 00  00 01 00 20 17 15 00 00 02 9F 14 00 00 00 05 07

04 30 00  00 1E B3 9F 40 06 FF 03 00 28 84 00 05 41 44 52

04 40 00  45 53 04 0A FC 40 00 05 4C 49 49 53 54 04 00 00

04 50 00  00 00 01 54 04 00 00 00 00 00 00 00 00 00 00 00
```

```
09F2  03
0AF2  EC
```
} pointer naar start of text

```
10 REM ..... oorspronkelijk 28 x asterix ....
20 ADRES = PEEK(#2A0)*256 + PEEK(#29F)+5
30 CALL ADRES
40 END
```

INTERACTIEVE VIDEOTEX

BEKNOPTE BESCHRIJVING

1. WAT IS INTERACTIEVE VIDEOTEX?

VIDEOTEX is een communicatiesysteem dat o. a. toelaat informatie, opgeslagen in een computergeheugen, te verspreiden en op te vragen. Daartoe heeft de gebruiker een telefoon nodig evenals een terminal bestaande uit een beeldscherm en een klavier. Het kan hier gaan om een specifieke terminal, een aangepast kleurentelevisietoestel, een persoonlijke microcomputer enz...

Terloops weze opgemerkt dat het coderingsschema van de interactieve videotex eveneens gebruikt wordt voor de omgeroepen videotex, genoemd Teletekst. Het betreft hier een éénwegsysteem voor het verspreiden van informatie, die verzonden wordt bij middel van het televisiesignaal.

Interactieve videotex is een systeem met individuele tweeweg-transmissie dat aldus een dialoog tussen gebruiker en computer toelaat. De opslagmogelijkheden van het ganse systeem zijn slechts beperkt door de geïnstalleerde stockeringscapaciteit. De bediening is zeer eenvoudig en door enkele toetsen in te drukken krijgt de gebruiker alle gekozen informatie bijna onmiddellijk op het scherm. Dank zij de interactie zijn ook bewerkingen en transacties mogelijk.

REGIE VAN TELEGRAFIE
EN TELEFONIE

Departement Informatieverwerking

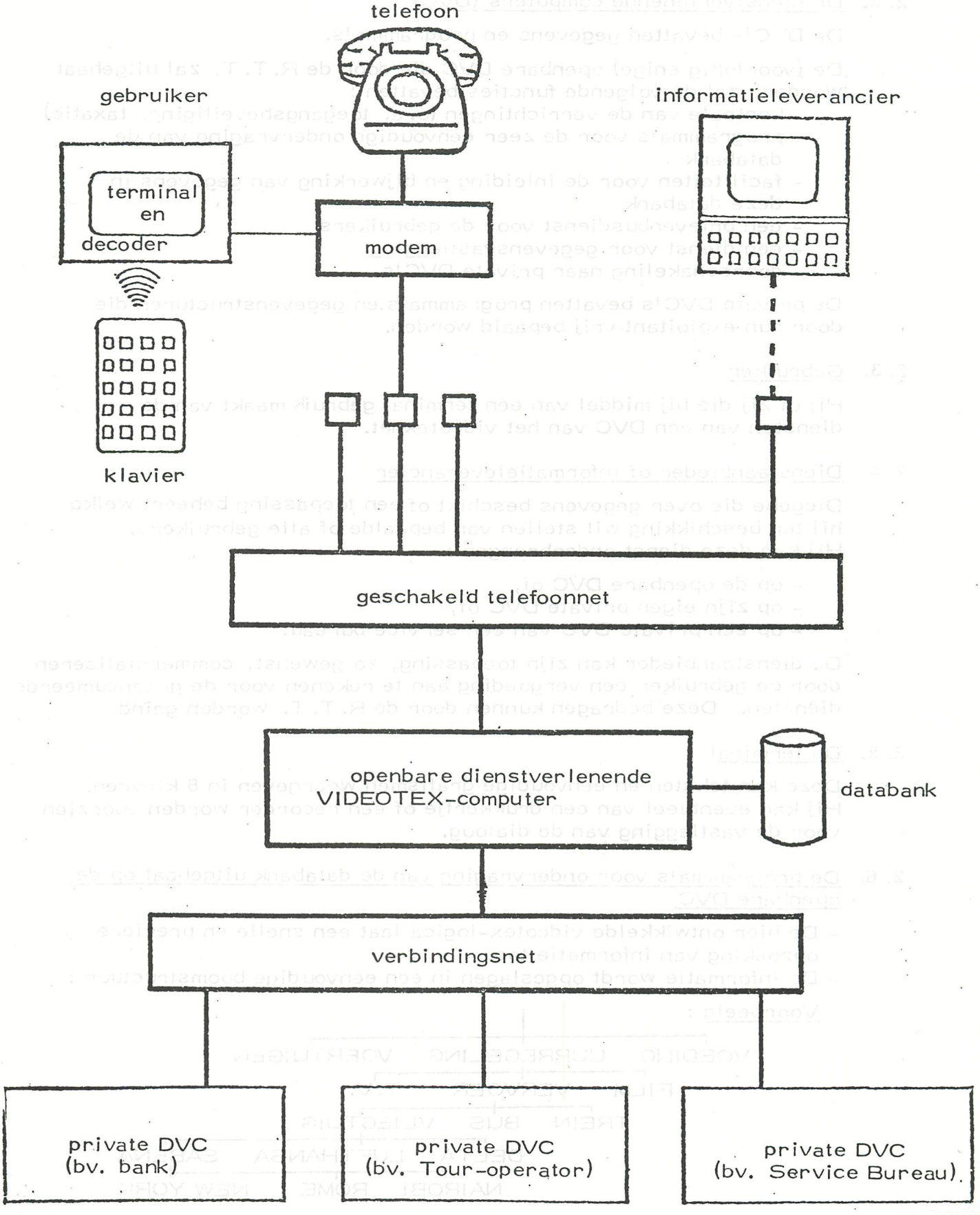
Projectgroep Videotex
Secretariaat : 02/216 38 27

April 1982

Projectgroep Videotex R. T. T.

2. DE SAMENSTELLENDE ONDERDELEN

2. 1. Algemeen schema



Alle onderdelen die deel uitmaken van de R. T. T. -infrastructuur zijn in een dikkere lijnsoort weergegeven.

2. 2. De dienstverlenende computers (DVC)

De DVC's bevatten gegevens en programma's.

De (voorlopig enige) openbare DVC die door de R. T. T. zal uitgebaat worden, zal de volgende functies bevatten :

- kontrole van de verrichtingen (o. a. toegangsbeveiliging, taxatie)
- programma's voor de zeer eenvoudige ondervraging van de databank
- faciliteiten voor de inleiding en bijwerking van gegevens in deze databank
- een brievenbusdienst voor de gebruikers
- een dienst voor gegevensvastlegging
- doorschakeling naar private DVC's.

De private DVC's bevatten programma's en gegevenstructuren die door hun exploitant vrij bepaald worden.

2. 3. Gebruiker

Hij of zij die bij middel van een terminal gebruik maakt van de diensten van een DVC van het videotexnet.

2. 4. Dienstaanbieder of informatieleverancier

Diegene die over gegevens beschikt of een toepassing beheert welke hij ter beschikking wil stellen van bepaalde of alle gebruikers.
Hij kan deze dienst onderbrengen :

- op de openbare DVC of,
- op zijn eigen private DVC of,
- op een private DVC van een service bureau.

De dienstaanbieder kan zijn toepassing, zo gewenst, commercialiseren door de gebruiker een vergoeding aan te rekenen voor de geconsumeerde diensten. Deze bedragen kunnen door de R. T. T. worden geïnd.

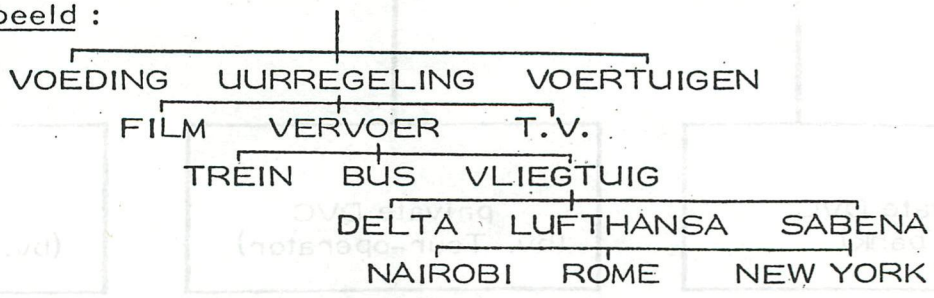
2. 5. De terminal

Deze kan teksten en eenvoudige grafismen weergeven in 8 kleuren.
Hij kan eventueel van een drukkertje of een recorder worden voorzien voor de vastlegging van de dialoog.

2. 6. De programma's voor ondervraging van de databank uitgebaat op de openbare DVC

- De hier ontwikkelde videotex-logica laat een snelle en precieze opzoeking van informatie toe.
- De informatie wordt opgeslagen in een eenvoudige boomstructuur :

Voorbeeld :



- De opgeslagen informatie kan ook langs een bepaald "trefwoord" worden opgezocht of via de naam van de dienstaanbieder.

3. WAT IS KARAKTERISTIEK AAN INTERACTIEVE VIDEOTEX?

3.1. In het algemeen

Videotex is een snel en eenvoudig informatiesysteem dat iedereen kan gebruiken, zonder speciale opleiding.
Informatie kan op een zeer eenvoudige manier worden opgezocht, dank zij een aangepaste zoekstructuur.
De informatie kan zeer gemakkelijk "up to date" gehouden worden.
In principe is de opslagcapaciteit onbeperkt.
Bij videotex kan een goedkope terminal (TV scherm) gebruikt worden.
Met videotex zijn heel wat grafische mogelijkheden voorzien zodat een aantrekkelijk beeld opgebouwd kan worden (kleuren, knipperen, ...).
Twee richtingscommunicatie tussen gebruiker en videotex-systeem laten alle types van interactie toe.
De gebruiker kan ook boodschappen sturen naar het videotex-centrum of andere gebruikers.
De "besloten gebruikersgroepen" kunnen de informatie beperken tot hun leden.
De informatie is beveiligd met persoonlijke paswoorden.
Beperkte dataverwerking is mogelijk op de openbare DVC o. a. gegevensvastlegging.
Overdrachtsnelheid :

- van abonnee naar videotex-systeem : 7,5 tekens per seconde
- van videotex-systeem naar abonnee : 120 tekens per seconde

3.2. Voor het bedrijfsleven

Naast een informatiebron is videotex een goed communicatiesysteem, om relaties met geografisch gespreide filialen, verdelers, kantoren of personeelskernen te onderhouden.
Een snelle verspreiding van veelgebruikte en vaak wijzigende gegevens naar vele medewerkers is dan mogelijk.
Ook intern voor een groot bedrijf met vele afdelingen en diensten kan videotex de communicatieproblemen oplossen.
Door het opzetten van "besloten gebruikersgroepen" blijft de vaak vertrouwelijke informatie beperkt tot de leden.
De informatieleverancier bepaalt zelf wie toegang heeft tot zijn informatie en aan welke prijs.
Het is mogelijk met de videotex-dienst toegang te krijgen of te verlenen tot uw eigen of andere computers. Daarenboven kan op termijn een internationaal schakelnet tot stand komen.
Videotex is een ideaal publicitair medium voor grote verspreiding met attractieve grafische mogelijkheden.
De informatieleverancier kan op een eenvoudige en efficiënte manier zijn informatie "te koop" aanbieden.

4. VOORBEELDEN VAN TOEPASSING.

Financieel : Agentschappen krijgen diverse informatie ter beschikking, bv. beursberichten, stand der rekeningen, financiële, juridische, fiscale informatie, boodschappendienst. Home-banking.

Verzekeringen : Opvolgen van contracten van de klanten. Speciale reglementeringen, berichten, tarieven.

Toerisme, transport : - Informatie aan de reisbureau's, catalogus, actuele tarieven, beschikbare zitplaatsen of kamers, onmiddellijke reservaties, boekingen.
- Uurregelingen allerlei : treinen, vliegtuigen, ...

Automobielsector : Videotex-net bij dealers, invoerders, garages, voor berichten, catalogen, prijslijst, wisselstukken, voorraad wagens, ...

Openbare dienst : Het omvangrijke pakket aan reglementen, wetteksten, procedures.

Uitgeverijen : Verspreiden van vakinformatie ...

Bedrijfsnieuws : Groepering van diverse gegevens over de bedrijfswereld.

Distributiesector : Communicatie met filialen, franchises. Winkelketens.

Immobiliën : Informatie voor en tussen makelaars. Aanbod aan het kliënteel.

Wetgeving : Gestructureerd bijhouden van wetten, reglementen, arresten, staatsblad, ...

Verenigingen : Informatie voor de leden van de beroepsvereniging. Aankondigingen, vergaderingen, brievenbus, ...

faculteit

```
2     CLEAR 3000:PRINT CHR$(12)
5     INPUT "  FACULTEIT";F%:PRINT
10    DIM N%(255)
15    S%=-1:V%=1:N%(0)=1:REM INIT. 1!
19    PRINT :PRINT CHR$(#E);F%;" !":POKE #131,1
20    S%=S%+1:T%=0:W%=0
25    CURSOR 10,20:PRINT V%
30    B%=N%(T%)/10000
40    A%=N%(T%)-10000*B%
50    A%=A%*V%
60    B%=B%*V%
70    A%=A%+W%
80    A2%=A%/10000
90    A1%=A%-A2%*10000
100   B%=B%+A2%
110   W%=B%/10000
120   B1%=B%-W%*10000
130   N%(T%)=A1%+B1%*10000
140   T%=T%+1
150   IF T%-1<>S% GOTO 30
200   IF N%(S%)=0 THEN S%=S%-1
203   V%=V%+1:IF F%<>V%-1 THEN 20
210   POKE #131,0:T%=S%:P$=""
220   B%=N%(T%)/10000:A%=N%(T%)-B%*10000
230   S$=STR$(B%):S$=RIGHT$(S$,LEN(S$)-1):S$=LEFT$(S$,LEN(S$)-2)
235   IF S$="0" THEN S$=""
240   T$=STR$(A%):T$=RIGHT$(T$,LEN(T$)-1):T$=LEFT$(T$,LEN(T$)-2)
250   IF B%<>0 THEN IF LEN(T$)<4 THEN T$="0"+T$:GOTO 250
260   P$=S$+T$:T%=T%-1:IF -T%=1 GOTO 350
270   FOR T%=T% TO 0 STEP -1
280   B%=N%(T%)/10000:A%=N%(T%)-B%*10000
290   S$=STR$(B%):S$=RIGHT$(S$,LEN(S$)-1):S$=LEFT$(S$,LEN(S$)-2)
300   IF LEN(S$)<4.0 THEN S$="0"+S$:GOTO 300
310   T$=STR$(A%):T$=RIGHT$(T$,LEN(T$)-1):T$=LEFT$(T$,LEN(T$)-2)
320   IF LEN(T$)<4 THEN T$="0"+T$:GOTO 320
330   P$=P$+S$+T$:IF LEN(P$)>65 THEN GOSUB 1000
340   NEXT T%
350   IF LEN(P$)<5 GOTO 358
351   FOR U%=5 TO LEN(P$) STEP 5
352   D$=D$+LEFT$(P$,5)+" "
354   P$=RIGHT$(P$,LEN(P$)-5)
356   NEXT U%
358   PRINT D$;:PRINT P$:P$="":D$=""
999   END
1000  O$=RIGHT$(P$,LEN(P$)-65)
1010  P$=LEFT$(P$,65)
1020  FOR U%=5 TO 65 STEP 5
1030  D$=D$+LEFT$(P$,5)+" "
1035  P$=RIGHT$(P$,65-U%)
1040  NEXT U%
1050  PRINT D$:P$=O$:D$=""
1060  RETURN
```

```
FACULTEIT?100

  100  !
93326 21544 39441 52681 69923 88562 66700 49071 59682 64381 62146 85929 63895
21759 99932 29915 60894 14639 76156 51828 62536 97920 82722 37582 51185 21091
68640 00000 00000 00000 00000 000
END PROGRAM
```

cassette list

```
5     REM CASSETTE LIST rev 5. ARTS
6     CURS=20.0
10    MODE 0:PRINT CHR$(12);:POKE #131,1:AD$=""
30    FOR A%=1 TO 21:READ B%:AD$=AD$+CHR$(B%):NEXT
40    AD%=PEEK(VARPTR(AD$)) IOR (PEEK(VARPTR(AD$)+1) SHL 8)+1
50    PRINT "DIT PROGRAMMA"
51    PRINT "GEEFT EEN INHOUDSOPGAVE VAN EEN CASSETTE"
52    PRINT "A    OP   110 BAUD VOOR DE MATRIX-PRINTER"
53    PRINT "B    OP   300 BAUD VOOR EEN MODEM VERBINDING"
54    PRINT "C    OP  1200 BAUD VOOR DE P880 VERBINDING"
55    PRINT "D    OP  2400 BAUD VOOR HET INTEL-SYSTEEM"
56    PRINT "E    OP  9600 BAUD VOOR HET DISPLAY"
60    PRINT " WAT WORDT GEWENST ? MAAK UW KEUZE !  ";
70    K=GETC:IF K<#41 OR K>#45 THEN GOTO 70
73    REM WAIT TIME 20
75    PRINT CHR$(12)
80    ON K-#40 GOSUB 91,92,93,94,95
81    POKE #FF05,BAU:GOTO 100
91    BAU=1.0:RETURN
92    BAU=#84:RETURN
93    BAU=#88:RETURN
94    BAU=#90:RETURN
95    BAU=#C0:RETURN
100   INPUT " CODE NUMMER VAN DE CASSETTE IS :";CN$:PRINT :PRINT
111   PRINT "WAT IS DE DATUM VAN VANDAAG   YYMMDD"
112   INPUT "VB 14 MAART 1982 IS  820314  ";DAT$:PRINT
113   IF CN$="" OR LEN(DAT$)<>6.0 THEN GOTO 1998
115   PRINT CHR$(12)
120   POKE #131,0:PRINT "DATUM : "+DAT$:PRINT CN$:FOR A%=0 TO LEN(CN$)-1:PRINT "
-";:NEXT:PRINT CHR$(#D):CPT%=1
1000  CURSOR 0,23:POKE #FF05,#C0:POKE #131,1
1001  PRINT "                                                              "
1010  CURSOR 0,23:CALLM AD%:CURS=CURS-1.0
1011  IF CURS=0.0 THEN CURS=20.0
1012  CURSOR 1,CURS
1013  POKE #FF05,BAU:POKE #131,0:POKE #40,#30:PRINT CPT%;" - ";
1030  FOR A%=#BFE7 TO #BF85 STEP -2:PRINT CHR$(PEEK(A%));:NEXT:PRINT CHR$(#D)
1035  IF GETC<>0 THEN 2000
1040  CPT%=CPT%+1:GOTO 1000
1998  PRINT :PRINT "DIT PROGRAMMA WERKT NIET ZONDER GOEDE GEGEVENS"
1999  PRINT "START HET PROGRAMMA OPNIEUW MET 'RUN'":PRINT :END
2000  POKE #131,1:PRINT :STOP
2010  GOTO 1040
10000 DATA #F5,#C5,#D5,#E5,#01,#40,#00,#11,#B1,#80,#21,#9E,#E6,#CD,#CE,#02,#E1,#
D1,#C1,#F1,#C9
```

```
100   REM /** DNE 241 MODIFIER 21MRT82 **/
200   REM /* FILE TYPE (RD/WR) : #23 (#) OR #31 (1)
210   POKE #2BE1,#23:POKE #2BDC,#23
300   REM /* #L PAGE FORMAT
310   POKE #1F40,#42:REM PAGE 1
320   POKE #1F3A,#42:REM FOLLOWING PAGES
400   REM /* HARDCOPY : WAITSPACE (#L AND #P)
410   REM POKE #1F36,#CD:POKE #1F37,#CE:POKE #1F38,#2C:REM YES
420   POKE #1F36,0:POKE #1F37,0:POKE #1F38,0:REM NO
900   REM /*** LAB USE ONLY ***/
910   POKE #2F01,1:REM EPSON AUTO-LF
1000  FOR I=0 TO 2500:IF GETC=#20 GOTO ^1100:NEXT
1100  POKE #100,0:POKE #101,0:CALLM 12000
```

weersatelliet foto's op uw tv-scherm

Sinds kort gebruik ik mijn DAI computer voor de weergave van foto's , afkomstig van weersatellieten. Het blijkt, dat de grafische mode 6 geschikt is voor het zichtbaar maken van de foto's van wolkenformaties, welke dagelijks door een aantal weersatellieten naar de aarde worden gestuurd.

Op de plaatjes die tot nu toe via de DAI computer zijn geproduceerd zijn deze wolken min of meer duidelijk te onderscheiden, maar ik denk dat zowel de software als de hardware nog verder verbeterd kunnen worden.

Wanneer er meer DAInamic leden zijn die hun computer willen gaan gebruiken voor het weergeven van satellietfoto's, of die geinteresseerd zijn in het geheel van ontvangen en weergeven van foto's , kunnen zij kontakt opnemen met mij.
Als veel mensen meer willen weten over deze fascinerende hobby, dan kan ik wat informatie toesturen aan de redaktie van DAInamic voor plaatsing in de nieuwsbrief.

Voor geinteresseerden: de kosten voor het ontvangen en weergeven van weersatellietfoto's bedragen ongeveer f 800,- (BFR 11500) en daarvoor heeft u dan een ontvanger, antenne en interface voor de DAI.

Voor meer informatie:

H. Bakker
Rijnstraat 28
6811 EW Arnhem
tel. 085-451394

INTERFACE HARDWARE

Op het blokschema in Fig. 2 komt linksboven het 2400 Herz signaal binnen van de tuner. Dit signaal wordt eerst gefilterd en versterkt (A1 resp. A2), daarna gelijkgericht en de condensator die hierbij wordt opgeladen, wordt met een constante stroom ontladen. De Comparator A3 zorgt ervoor dat het zaagtand-achtige signaal over de condensator C wordt omgezet in een blokspanning met variable pulsbreedte.
Verder zorgt het timer IC NE 555 ervoor dat de puls na ca. 300 micro seconden in ieder geval weer laag wordt, ook al is het ingangssignaal te hoog in volume.
Bij weergave van het opgenomen signaal zet OP AMP A4 de misvormde videopulsen weer om in een nette blokspanning, die via buffers N4 en N5 naar de DAI gaan.
Tevens triggert de positieve flank van deze pulsen een oscillator gevormt door N2 en N3. De blokspanning van deze oscillator heeft dus dezelfde frequentie als de videopulsen en tevens vallen van beide signalen de positieve flank samen.

Deze positieve flank zal later in het machinetaal programma worden gebruikt als referentiestart voor het nemen van samples.

Voor we de interface kunnen gebruiken, moeten we eerst met potmeter P3 de frequentie van de oscillator instellen op iets minder dan 2400 Herz. Wie geen frequentiemeter heeft, kan op het gehoor afgaan door eerst te luisteren naar de 2400 Herz van een weersatelliet, en de oscillatorfrequentie iets lager dan deze frequentie in te stellen (ca. 2360 Herz).

Als we nu een ontvanger aansluiten op de interface en we wachten tot we het geluid van een weersatelliet horen, dan moeten we eerst P1 en P2 zo instellen, dat de pulsbreedte varieert tussen ca. 100 en 300 microseconden. Ook dit is eventueel op het gehoor af te regelen door P1 *zo in te stel*len dat het signaal naar de Tape rec. nog net zuiver klinkt. Maar een oscilloscoop is hierbij toch wel handig ...
P1 bepaalt de versterking en P2 het contrast.

WEERSATELLIETEN FOTO'S OP DE DAI COMPUTER.

De DAI Computer kan voor veel dingen gebruikt worden, maar bovengenoemde toepassing zal velen van u vreemd voorkomen. Toch is het met vrij eenvoudige middelen mogelijk om foto's, afkomstig van weersatellieten, zichtbaar te maken op een TV scherm. Wat hiervoor nodig is, zal in dit verhaal aan de orde komen.

Eerst iets over weersatellieten. Zij kunnen verdeeld worden in globaal twee groepen. De eerste soort draait met de aarde mee en staat daardoor stil ten opzichte van de aarde (dit heet Geostationair). Deze satellieten vinden we op ca. 36000 Km hoogte loodrecht boven de evenaar.
Doordat ze ten opzichte van de aarde stil staan, fotograferen ze steeds hetzelfde gedeelte van deze aarde.

De tweede soort draait rond de aarde via noord- en zuidpool (ongeveer) en staan daardoor niet stil t.o.v. de aarde. Bovendien is hun vlieghoogte 'slechts' 800 - 1500 Km. Met deze tweede soort wil ik me verder bezighouden.

Een gevolg van deze noord-zuid baan is dat dit soort satellieten slechts dan te ontvangen is, wanneer ze over West Europa 'vliegen' en dan nog maar gedurende maximaal 20 minuten. Bij iedere omwenteling verschuift de baan van de satelliet 15 graden, zodat na een halve dag weer opnieuw ontvangst mogelijk is. De satelliet vliegt dan echter in omgekeerde richting.

Om weersatellieten te kunnen ontvangen hebben we (u raadt het al) een ontvanger nodig. Wanneer u handig bent is zo'n ontvanger zelf te maken. Later zal een leverancier genoemd worden waar bouwpakketten en kant-en-klaar ontvangers te koop zijn.

Het signaal wat we ontvangen bestaat uit een toon van 2400 Herz en de eigenlijke video informatie zit in de amplitude van dit signaal (Amplitude Modulatie AM).
Wanneer we dit signaal gelijkrichten houden we sinusvormige pulsen over waarvan de hoogte een maat is voor een licht- of donker niveau .
De frequentie van deze toon is echter niet continu. Twee maal per seconde wordt deze onderbroken door een toon van een andere frequentie. Deze korte toon geeft het begin aan van een nieuwe beeldlijn aan, want weersatellieten maken eigenlijk geen foto van de aarde, maar zij 'scannen' deze lijn voor lijn af, net zoals een TV scherm wordt opgebouwd.
Per seconde zendt een satelliet dus twee beeldlijnen uit, maar dat geldt alleen voor de Russische satellieten. Amerikaanse satellieten zenden per seconde vier beeldlijnen uit, nl twee beeldlijnen van het zichtbare gedeelte van de aarde +wolken, en twee lijnen van het Infra Rode gedeelte . Hierdoor kunnen tevens foto's welke 'snachts ontvangen worden zichtbaar gemaakt worden.
De mooiste foto's komen echter van de Russische satellieten Meteor 120 en - 240, zodat we ons eerst met de weergave van deze foto's zullen bezighouden.

Het begin van iedere beeldlijn wordt door de satelliet aangegeven door een sink-puls. Gedurende een zekere tijd wordt het video signaal onderdrukt (vergelijk het TV signaal) en hierop zal een later te behandelen machinetaal programma reageren door te beginnen op een nieuwe regel.

Goed, wat we dus ontvangen is een toon van 2400 herz welke in amplitude (sterkte) varieert. Om later de foto meerdere malen te kunnen weergeven nemen we deze toon tijdens de ontvangst op een cassette recorder of (beter) bandrecorder op.
Als we de toon rechtstreeks op de band zetten, zullen we echter later bij de weergave last kunnen krijgen van zg. Drop-outs.
Dit is het kort weg vallen van het geluid tijdens weergave door stof of slechte plekken op de band. Als het geluid wegvalt zien we dat meteen in een kleurverandering op het TV-scherm.
Om dit te voorkomen vorm ik het signaal eerst om tot pulsen die in lengte afhankelijk zijn van de amplitude van het ontvangen signaal. Met andere woorden: als ik na het gelijkrichten van het 2400 Herz signaal een 'halve sinusvormige puls' krijg met een HOGE amplitude, dan zet ik een puls met een LANGE pulslengte op de band.

De hard-ware die ervoor zorgt dat de video informatie wordt omgezet naar pulsen van variabele lengte, is niet erg ingewikkeld. Ik zal hier later verder op ingaan.
Als we een geslaagde opname hebben gemaakt, staan op de band pulsen, die bij weergave naar de computer gaan. De computer heeft behalve deze pulsen nog een tweede signaal nodig, nl een klok signaal. Dit tweede signaal wordt in de weergave hardware opgewekt door een oscillator die precies dezelfde frequentie heeft als de video puls-frequentie.
Wat de computer nu tijdens weergave van de foto moet doen is het volgende:

1 - Wacht tot de video-informatie gedurende enige tijd laag blijft (trigger sink puls),

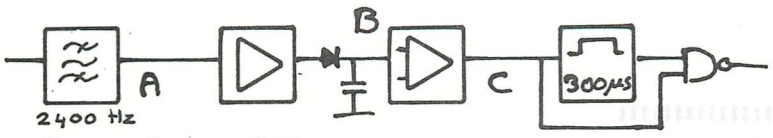
2 - Kijk hierna steeds na het hoog worden van de klok hoe lang de video puls duurt,

3 - Maak afhankelijk van de puls-lengte een witte, donker-grijze, licht-grijze of zwarte punt als dit kan (zie verder..).

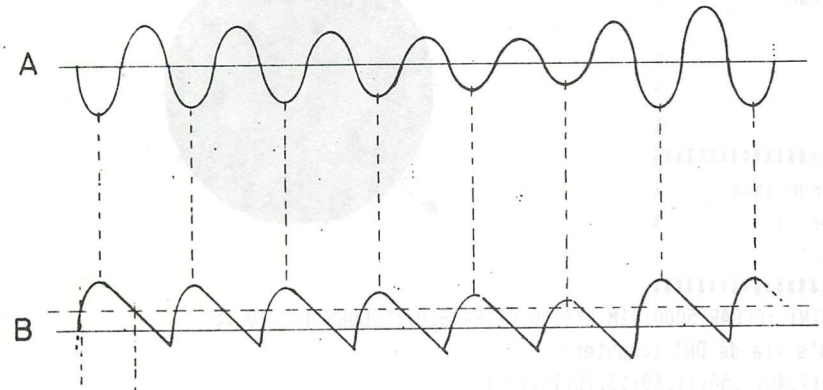
Verder moet de computer natuurlijk de aldus gemaakte punten (dot's) op de juiste plaats op het beeldscherm plaatsen, en stoppen als het scherm vol is.
Het aantal pulsen (halve sinusvormige video-pulsen) per beeld lijn is echter groter dan het aantal dot's wat de DAI op een lijn kan plaatsen.
Twee beeldlijnen p/sec met een pulsfrequentie van 2400 Herz betekent per beeldlijn een kleine 1200 pulsen (2400/2 - sink puls). De DAI komt niet verder dan 336 dots.
Daarom plaatsen we ten eerste een gedeelte van de beeldlijn niet op het scherm ,en ten tweede plaatsen we van het wel zichtbare gedeelte slechts de helft (1 video puls WEL, de volgende NIET enz.) . Met een INPUT statement is de grootte van het linker gedeelte van het totale beeld wat overgeslagen moet worden, in te voeren.
De computer zal na de sink-puls eerst dit gedeelte overslaan, vervolgens de helft van het vervolg van de beeldlijn displayen tot de regel op het scherm vol is, en tenslotte weer wachten op de volgende sink-puls.
Na 256 lijnen keert het machine taal programma terug naar BASIC en wacht tot een toets wordt ingedrukt.

wordt vervolgd...

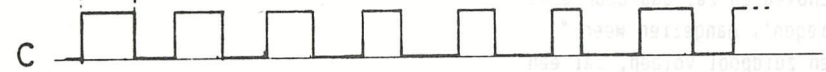
weersatelliet foto's op uw tv-scherm



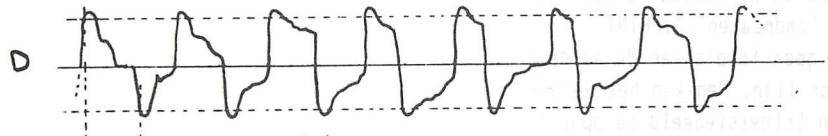
Signaal van A1:



Signaal van de Comparator:

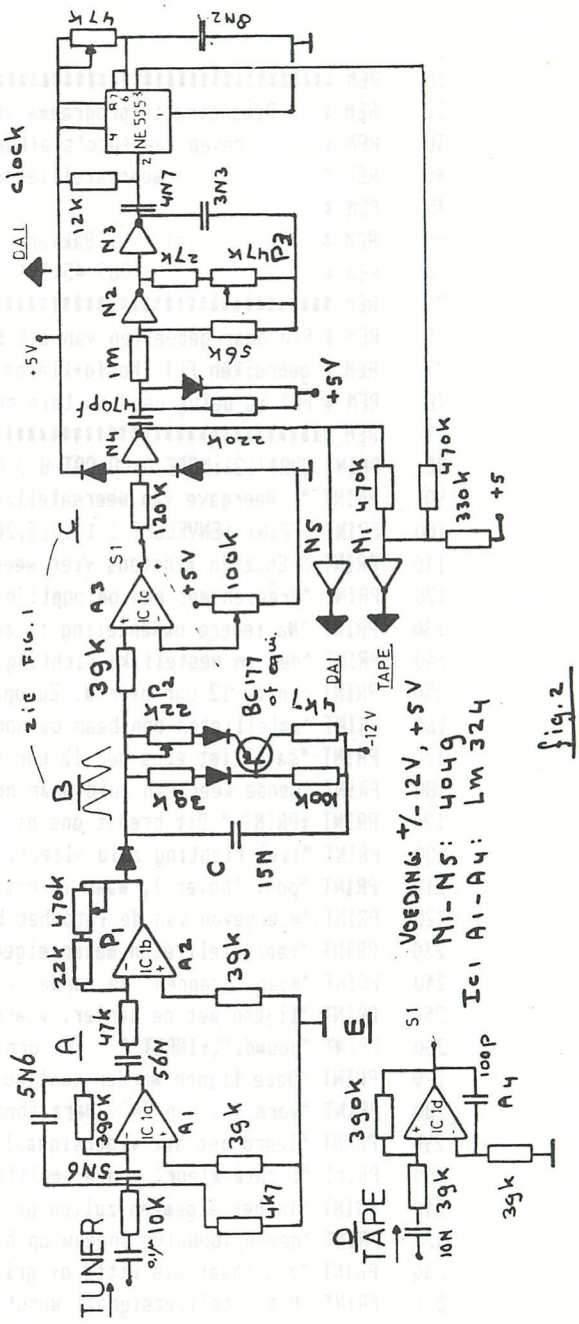


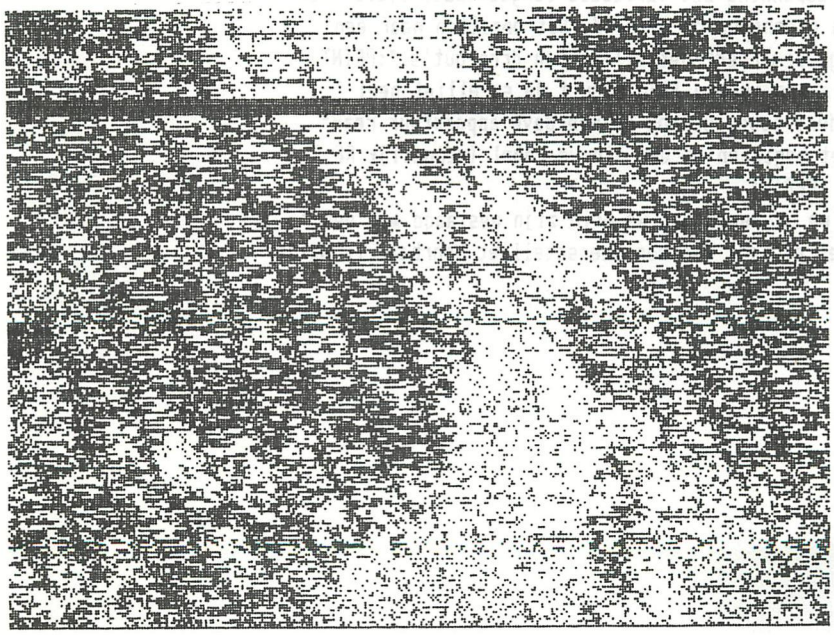
Signaal afkomstig van Taperec:



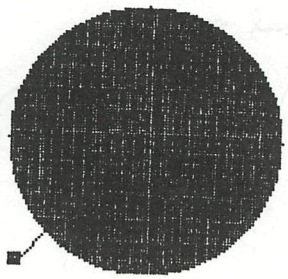
Signaal van A4:

fig. 1





```
T
10    REM ************************************************
20    REM *    Demonstratie programma voor het weer-      *
30    REM *       geven van foto's afkomstig van          *
40    REM *             weersatellieten                   *
45    REM *                                               *
50    REM *          (c)   H. Bakker   1982               *
60    REM *                085-451394                     *
70    REM ************************************************
75    REM * Een paar gedeelten van dit DEMO*programma     *
77    REM * gebruiken FGT (hoofd+kleine letters )         *
78    REM * FGT is being used in this program !!          *
79    REM ************************************************
80    PRINT CHR$(12):MODE 0:COLORT 8 0 8 0:PRINT :CLEAR 5000:DIM A%(100.0),F(16.0):COLORG 1 5 10 15
90    PRINT "  Weergave van weersatellietfoto's via de DAI computer"
100   PRINT :PRINT :ENVELOPE 1 1,10;3,20;5,30;7,40;9,50;11,60;13,70;15,100;
110   PRINT " Er zijn minstens vier weersatellieten, die rond de aarde"
120   PRINT "draaien met een omlooptijd van ongeveer 100 minuten."
130   PRINT "Na iedere omwenteling is zo'n weersatelliet ongeveer 25 gra-"
140   PRINT "den in westelijke richting verschoven en zal daardoor eens"
150   PRINT "in de 12 uur over W. Europa 'vliegen'. Aangezien weer-"
160   PRINT "satellieten een baan om noord- en zuidpool volgen, zal een"
170   PRINT "satelliet eens per 12 uur van noord naar zuid, en de vol-"
180   PRINT "gende keer van zuid naar noord overvliegen."
190   PRINT :PRINT " Dit brengt ons bij het eerste probleem : wanneer een satel-"
200   PRINT "liet richting zuid vliegt, is het beeld in orde (noord-"
210   PRINT "pool 'boven'), maar in het andere geval moeten we bij het"
220   PRINT "weergeven van de foto het beeld 'omdraaien'.":PRINT
230   PRINT "Weersatellieten maken eigenlijk geen foto's van de aarde,"
240   PRINT "maar 'scannen' de aarde lijn voor lijn. Men kan het verge-"
250   PRINT "lijken met de manier, waarop een televisiebeeld is opge-"
260   PRINT "bouwd.";:INPUT "   (return)";D$:PRINT CHR$(12)
270   PRINT "Deze lijnen worden continu naar de aarde gezonden in de"
280   PRINT "vorm van een 2400 Herz toon, welke in amplitude is gemodu-"
290   PRINT "leerd met het videosignaal. Kleine amplitude betekent"
300   PRINT "lichte kleur, grote amplitude geeft de kleur zwart weer.":PRINT
310   PRINT "In het algemeen zullen de zee en het land een donkere kleur"
320   PRINT "geven (behalve sneeuw op b.v. bergen), en zijn wolken"
330   PRINT "zichtbaar als witte of grijze vlekken.":PRINT
331   PRINT "Het satellietsignaal wordt tijdens ontvangst eerst omgezet"
332   PRINT "naar een blokspanning met variabele lengte (duty cycle), en"
333   PRINT "wordt daarna opgenomen op de band. Dit omzetten naar een"
334   PRINT "blokspanning vermindert het effekt van drop-out's.":PRINT
340   PRINT "Aan het begin van iedere lijn stuurt de satelliet heel "
350   PRINT "even een andere toon, waarop de weergave apparatuur rea-"
360   PRINT "geert met het beginnen op een nieuwe regel (vergelijk RE-"
370   PRINT "TURN & LINE FEED)."
380   PRINT "Goed, laten we aannemen dat we erin zijn geslaagd met een "
390   PRINT "ontvanger het signaal van een weersatelliet hoorbaar te ma-"
395   PRINT "ken."
```



```
400  PRINT " Dit geluid klinkt ongeveer zo....: (return)";
410  FOR X%=1 TO 100:A%(X%)=SQR((10000.0-(100.0-X%)*(100.0-X%)-X%*X%)/2.0):NEXT X%:INPUT D$
420  MODE 6A:Q%=XMAX/2.0:PRINT CHR$(12):COLORT 1 8 8 0
430  PRINT :PRINT "              ** GELUID OP KANAAL 1 & 2 **"
440  FOR X%=1 TO 50:DOT RND(XMAX),RND(YMAX) 15:NEXT X%
450  FOR X%=1.0 TO 100.0:SOUND 1 1 10 3 FREQ(50.0*X%)
460  DRAW Q%-A%(X%),75+X% Q%+A%(X%),75+X% 10:NEXT X%
470  FOR X%=2 TO 100:Z%=15:IF X%=75 THEN GOSUB 580
480  SOUND 1 1 10 1 FREQ(2400.0):FILL Q%-A%(X%)-30.0,X%+42+X%/3 Q%-A%(X%)-5.0,X%+73+X%/4 1
490  FILL Q%-A%(X%)-10.0,X%+X%/3+58 Q%-A%(X%)-14.0,X%+X%/3+62 5
500  IF X%=25.0 THEN GOSUB 550
510  DRAW Q%-A%(X%)-10.0,X%+X%/3+60 Q%-A%(X%)-1.0,X%+75 Z%
520  IF Z%=15.0 THEN WAIT TIME 8:Z%=1:GOTO 510
530  SOUND 1 1 8 0 FREQ(1300.0):NEXT X%:SOUND OFF
540  WAIT TIME 100:GOTO 620
550  PRINT :PRINT " Dit is de 2400 Herz toon waarvan de amplitude veranderd"
560  PRINT "overeenkomstig de beeldinformatie (Amplitude geModuleerd)."
570  WAIT TIME 250:RETURN
580  PRINT CHR$(12):PRINT " Dit is een toon van een andere frequentie, welke het begin"
590  PRINT "van iedere video lijn aangeeft (trigger burst)."
600  PRINT "Deze toon is kort hoorbaar aan het begin van iedere lijn.";
610  WAIT TIME 250:RETURN
620  REM
621  FOR X%=1.0 TO 16.0:F(X%)=SIN((X%*PI)/8.0):SOUND 1 1 15 3 FREQ((17.0-X%)*100.0)
622  SOUND 2 1 15 3 FREQ(X%*100.0):NEXT X%:Q%=XMAX/2.0:SOUND OFF
625  Q=XMAX:SP=6.0:CC=15.0:X=1.0
626  MODE 5:ENVELOPE 0 15
630  GOSUB 2000
640  C=GETC:WAIT TIME 10
650  C=GETC:IF C=0.0 THEN 650
660  PRINT CHR$(12):COLORT 8 0 8 0:MODE 0
670  PRINT "De computer ontvangt dus bij weergave van de band +/- 2400"
680  PRINT "maal p/sec. een puls van wisselende lengte.  De taak van de"
690  PRINT "computer is te herkennen wanneer de puls begint (herkennen"
700  PRINT "van de positieve flank) en dan te kijken hoelang de puls"
710  PRINT "hoog blijft. Afhankelijk van de pulslengte vormt de computer"
720  PRINT "een zwarte, donkergrijze, lichtgrijze of witte punt."
730  PRINT :PRINT "Het zal duidelijk zijn dat dit vormen van de punten niet"
740  PRINT "te veel tijd in beslag mag nemen omdat de computer klaar"
750  PRINT "moet zijn als de volgende puls begint. Daarom zal het niet"
760  PRINT "eenvoudig zijn een machinetaal programma te schrijven wat"
770  PRINT "zo snel is dat de computer gebruik kan maken van de 16 kleu-"
780  PRINT "ren mode. Maar zelfs met de vier kleuren die wel mogelijk"
790  PRINT "zijn, zien de resultaten er best acceptabel uit.":PRINT
800  PRINT "Bij het machinetaal programma hoort een kort BASIC program-"
810  PRINT "ma. In dit programma wordt u gevraagt hoeveel punten van"
820  PRINT "iedere beeldlijn u wilt overslaan (SKIP LEFT PART) en ver-"
830  PRINT "volgens dient u in te voeren van welk type (Amerikaanse of"
840  PRINT "Russische satelliet) u een foto wilt weergeven."
850  PRINT :PRINT :INPUT "    (return)";D$:PRINT CHR$(12)
855  PRINT "Hierna worden door dit BASIC programma diverse va-"
860  PRINT "riabelen ten behoeve van het machinetaalprogramma gepoked."
```

```
870  PRINT "Dan gaat de computer naar MODE 6 en begint het machinetaal-"
880  PRINT "programma met het starten van de bandrecorder  en wacht"
890  PRINT "even tot de bandrecorder op snelheid is gekomen. Na ongeveer"
900  PRINT "1 seconde begint het weergeven nadat de computer een juist"
910  PRINT "triggersignaal heeft herkent."
920  PRINT "Als het beeld wat verschijnt te licht of te donker is, dan"
930  PRINT "kunt u tijdens het weergeven met de toets CURSOR LEFT het"
940  PRINT "beeld donkerder, en met CURSOR RIGHT het beeld lichter"
950  PRINT "krijgen."
960  PRINT "Het indrukken van de TAB toets stopt het machinetaal pro-"
970  PRINT "gramma. Druk niet op de BREAK toets, want dit stopt wel"
980  PRINT "de bandrecorder, maar niet het programma."
990  PRINT "Als het beeld vol is, stopt de band en springt het program-"
991  PRINT "ma terug naar BASIC."
992  PRINT "De klokpuls moet u aansluiten op BIT 0 van PORT 0 (pin 14"
993  PRINT "van de DCE bus) en het video signaal komt binnen op pin"
994  PRINT "16 van de DCE bus. Verbindt de andere 6 bits van port 0 met"
995  PRINT "nul via een weerstand van 1000 ohm.":PRINT :INPUT "    (return)";D$
1000 MODE 6:FOR X=1.0 TO 100.0:SOUND 1 1 13 3 FREQ(50.0*X):NEXT X:COLORG 13 9 6 2
1010 A$="* WEERSATELLIET ONTVANGST *":X=1.0:Y=200.0:SP=6.0:CC=2.0:DF=1.1:FF=1.0:FC=2.0:GOSUB 5000
1020 A$="Voor informatie over ontvangers :":X=10.0:Y=150.0:CC=9.0:DF=0.9:FF=1.0:GOSUB 5000
1030 X=20.0:CC=2.0:A$="fa.  MECOM":Y=130.0:GOSUB 5000
1040 A$="     Coendersstraat   24":Y=110.0:GOSUB 5000
1050 A$="     9780 AA  Bedum":Y=90.0:GOSUB 5000
1060 A$="Tel: 05900 - 14390":Y=70.0:GOSUB 5000:FF=0.0
1070 X=10.0:CC=6.0:A$=" of : H. Bakker  Rijnstraat 28  6811 EW  Arnhem":Y=40.0:GOSUB 5000
1080 A$="      Tel: 085 - 451394":Y=30.0:GOSUB 5000
1090 A$="=":FOR X=1.0 TO XMAX-6.0 STEP 10.0:Y=1.0:GOSUB 5000:Y=YMAX-10.0:GOSUB 5000:NEXT
1091 A$="Bedankt voor uw aandacht - Tot ziens":X=70.0:Y=10.0:CC=6.0:FF=1.0:GOSUB 5000
1095 C=GETC:WAIT TIME 20:C=GETC
1098 C=GETC:IF C=0.0 THEN 1098
1099 END
2000 COLORT 0 8 8 0
2001 Y=245.0:A$="Kleine amplitude geeft witte kleur ,":GOSUB 5000
2002 Y=230.0:A$="Grote amplitude geeft zwarte kleur :":GOSUB 5000
2003 Y=145.0:A$="Klok voor de synchronisatie :":GOSUB 5000
2004 Y=95.0:A$="Signaal afkomstig van Tape recorder :":GOSUB 5000
2005 Y=50.0:A$="Overeenkomstige kleur van de punt op het scherm :":GOSUB 5000
2008 Y=2.0:A$="( DRUK OP 'RETURN' VOOR VERVOLG )":X=50.0:CC=8.0:GOSUB 5000
2020 DRAW 0,195 XMAX,195 10:R%=19:GOSUB 5000
2030 FOR X%=-9.0 TO 0:P=ABS(2.0*X%)-9.0
2035 FILL R%+15,20 R%+43,40 X%+13.0:SOUND 1 0 ABS(X%) 1 FREQ(2400.0)
2040 DRAW R%+2,115 R%+16,115 15:DRAW R%+16,115 R%+16,135 15:DRAW R%+16,135 R%+32,135 15
2042 DRAW R%+32,135 R%+32,115 15:DRAW R%+4+P,65 R%+16,65 15:SOUND OFF
2044 DRAW R%+16,65 R%+16,85 15:DRAW R%+16,85 R%+32+P,85 15
2045 DRAW R%+32+P,85 R%+32+P,65 15
2050 FOR Y%=1.0 TO 16.0:R%=15*X%*2+2*Y%+Q%+120
2070 DOT R%+X%+5,F(Y%)*X%*3.0+195.0 15
2080 NEXT:NEXT
2999 WAIT TIME 200:RETURN
5000 SOUND 2 1 10 3 FREQ(10.0*Y+100.0):C=FC*#40+CC:F=FF*#80+PF*#40+ZF*#20+VF*#10+VF*#10+DF
5002 POKE #2F0,C:POKE #2F1,F
5004 POKE #2F2,X MOD 256:POKE #2F3,X/256:POKE #2F4,Y
5006 POKE #2F5,SP:POKE #2F6,ID
5008 CALLM #300,A$:SOUND OFF :RETURN
```

DAI PERSONAL COMPUTER SCHEMATICS

Sheet 1 CPU
Sheet 2 RAM DRIVE
Sheet 3 RAM
Sheet 4 TIMING
Sheet 5 ROM + I/O
Sheet 6 MAIN BOARD GENERAL LAY-OUT
Sheet 7 SOUND
Sheet 8 PADDLE
Sheet 9 B/W TV CARD
Sheet 10 PSU
Sheet 11 TIMING
Sheet 12 PAL COLOUR CARD LAY-OUT
Sheet 13 PAL COLOUR CARD
Sheet 14 RGB CARD LAY-OUT
Sheet 15 RGB CARD

Now available from your club : the complete DAIpc schematics,
drawn by Mr Boerrigter and co during many,many sleepeless nights...
Most sheets measure about 40cm x 60 cm,in professional styling.
order from :

DAInamic
Heide 4
3171 WESTMEERBEEK
BELGIUM

price of the complete package : 850 Bfr including mailing and VAT.
This hardware information is available for personal use only !

How to order DAInamic software :
send cheque in belgian francs or cash together with your order to
DAInamic Heide 4 3171 WESTMEERBEEK BELGIUM.

Contact address for contribution,membership,backnumbers,the best of
DAInamic 80/81 :
Bruno Van Rompaey Bovenbosstraat 4 3044 HAASRODE BELGIUM

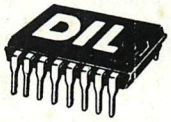
"The BEST of DAInamic" will be available in august 82. You can order
now all programs that have been published in these issues :
"T80/81" : price : 850 Bfr. (57 programs on cassette/DCR) see contents p. 101

Also available on tape : the 23 programs published in this issue:
save yourself hours of typing for 500 Bfr.
name of the package : "N10".

p.v.b.a. A.C.S.

Meensesteenweg 49
8800 ROESELARE
Tel. 051/21 30 89

Beenhouwersstraat 87
8000 BRUGGE
050/33 08 01



D.I.L.-ELEKTRONIKA,

MIJNSHERENLAAN 108,
3081 CH ROTTERDAM
Nederland
TELEFOON: 010 - 854213

Guibernau Electronica, s.a.

Sepulveda 104
BARCELONA-15 (SPAIN)
Tel. 243-34-32

IDS 2000

Rue de la Bonne Femme 11
4030 GRIVEGNEE
Tel. 041/41 32 20

LEGOTRONICS

Kon. Albert I laan 97
8800 ROESELARE
Tel. 051/22 01 03

MEMOCOM

Mini-digitale cassetterecorder

Postbus 2924
3000 CX ROTTERDAM - Nederland
Tel. 010-148284

MICRO SELECT

Toutes applications micro-électroniques
Vente de systèmes et composants micro-processeur

3, rue Delcloche
4020 LIÈGE
Tel. 041/41 28 10

Bennenbergweg 1
3221 NIEUWRODE (bij AARSCHOT)
Tel. 016/56 87 70

DAI - Epson Printers - Memocom digitale cassette recorder - Barco kleurenmonitor - Software - Microlectuur - Service

Publishing House J. VAN IN
att. : L. CAMPS

Educational Software
primary - secondary schools

Grote Markt 39
2500 LIER
Tel. 031/80 55 11

TEVETRONIC

Avenue Milcamps 57
1040 BRUSSEL
Tel. 02/736 61 24