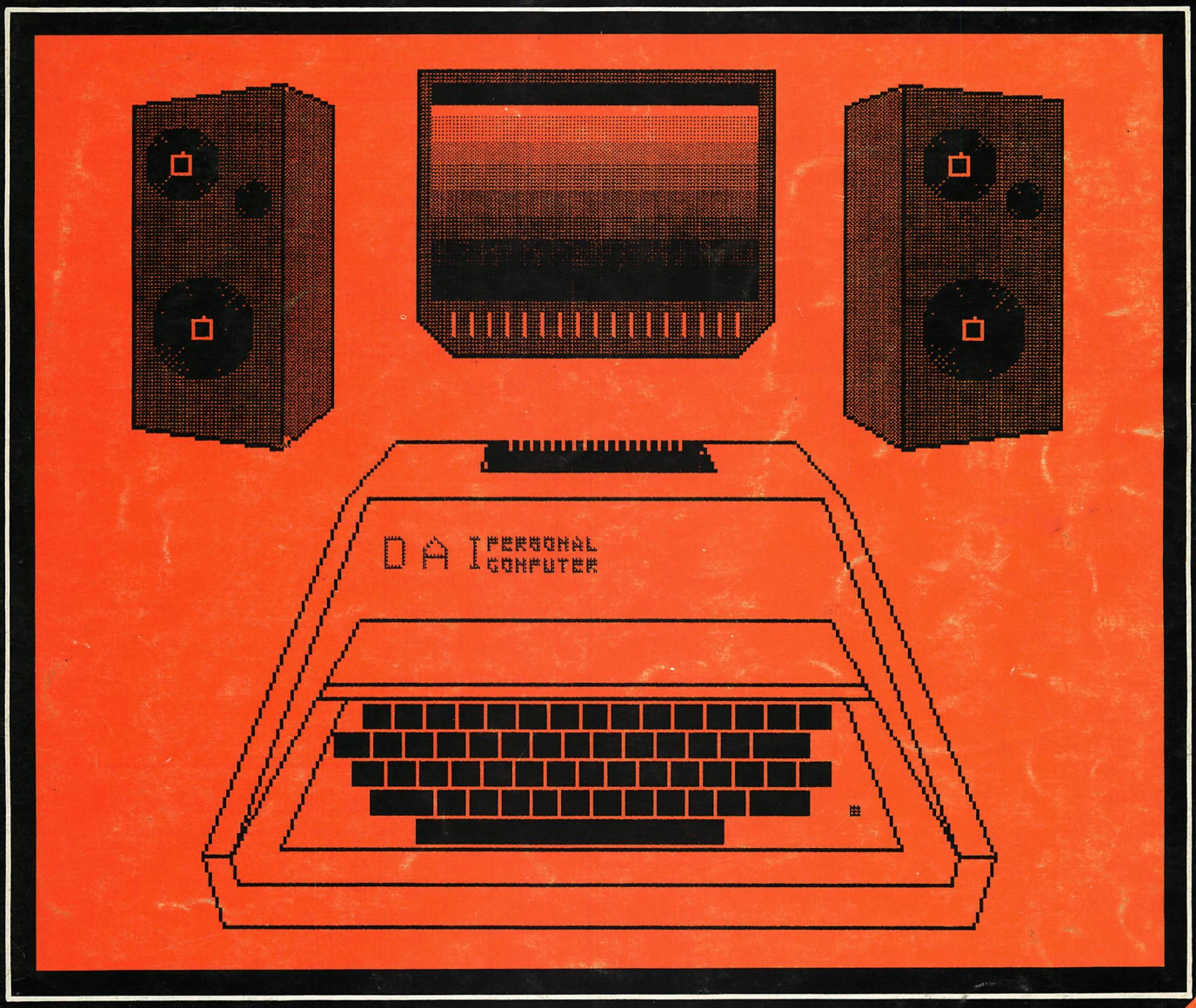
DAI NAMIC

17

tweemaandelijks tijdschrift juli - augustus 1983



personal computer users club

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

International

COLOFON

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

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

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

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

Lidgelden

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

DAINAMIC

PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

NEWSLETTER 17

Westmeerbeek, augustus 83

Beste leden,

Het zal zowat drie jaar geleden zijn dat de eerste publicatie van DAInamic de deur uitging. Een sober gestencild papiertje, verspreid via de firma DAI, kondigde aan dat er een gebruikersclub gesticht was rond de DAI personal computer. Het gebrek aan informatie omtrent dit revolutionaire toestel was mede de aanleiding geweest om contact te zoeken met medegebruikers. Het was de bedoeling mensen uit de regio te vinden om samen onze zo beruchte computer te leren gebruiken. De enorme bijval uit het buitenland, vooral Nederland tijdens de eerste maanden, was voor onze kern dan ook een enorme en prettige verrassing. De gebruikers van het eerste uur zullen zich zeker nog herinneren welke moeilijkheden er moes- ten overwonnen worden om een DAI computer in zijn bezit te krijgen. Velen van ons moesten het dan nog stellen met een 8K-versie zwart-wit, zonder sound. Rond die tijd was het voor DAI een enorme tegenvaller dat de toestellen voor de TELEAC- cursus niet tijdig konden geleverd worden, een enorme kans op ruime promotie ging hierdoor verloren. We mogen stellen dat alle toestellen die toen in productie genomen werden reeds lang op voorhand verkocht waren en de TELEAC-affaire was vlug vergeten. Helaas was de armslag van DAI te klein en kon de productie niet opgevoerd worden om aan de enorme aanvraag te voldoen. In Frankrijk , England , Duitsland en Italie was er zeer veel belangstelling en DAInamic werd overspoeld met aanvragen in de verschillende landstalen. Het lag dan ook voor de hand om af te stappen van het uni-Nederlands en zodoende zijn de verschillende vertaaldiensten op gang gekomen. Erg spijtig en onzeker waren de gebeurtenissen omtrent het faillisement van de toenmalige firma. We zijn dan ook tevreden dat er uiteindelijk een gezonde overname gekomen is : INDATA was de naam. Nieuwe mensen, nieuwe politiek : het is ondertussen duidelijk geworden dat er meer belangstelling was voor de thuismarkt. Deze opstelling heeft er voor gezorgd dat het aantal Belgische leden reeds meer dan 450 bedraagt ! In deze korte historiek mogen we een paar namen zeker niet vergeten : JC Camby, die als een ware diplomaat maar steeds weer de ongeduldige kopers en wachtenden moest te woord staan, Frank Druijff, die spoedig de Belgische kern kwam vervoegen, Freddy De Raedt, die zorgde voor programma's als FGT en Assembler en die ook de mensen die meer wilde weten over machinetaal te woord kon staan, Hans Wegman zette zijn mijlpaal in het DAIgebeuren met de ontwikkeling van MDCR, Jan Boerrigter die met zijn collega's er voor zorgde dat de hardware-geheimen van DAI voor ieder ontbloot werden en dan verder ging met zijn Firmware-manual. Bruno Van Rompaey nam de onderwijsmensen op sleep en schudde diDAIsoft uit zijn mouw... Vele naaste en verre medewerkers en correspondenten moeten we ongenoemd laten wegens plaatsgebrek. Allen samen hebben zij er voor gezorgd dat DAIpc zijn plaats op de turbulente computermarkt heeft weten te behouden en nog een gezonde toekomst voor zich heeft. We danken U voor de vele plezierige contacten ,

tot volgende keer,

Dear members,

DAInamic is 3 years young, this short historical review is too hard (and too long) to translate, you will find a translation in one of the next issues. To celebrate our club-birthday, we have a special software-offer, see the card in this issue.

yours sincerely,

Wilfried Hermans

NEWSLETTER 17

INHOUD — CONTENTS

215	Remark	Redactie
216	Inhoudstafel-contents	
217	Software contributions	F.Druijff
218	Videotex in Belgium	PUB
219	16 couleurs charcateres	A.Mariatte
220	cassette tape lister	D.Assink
224	Programmeer technieken	F.druijff
228	The EDITOR story	J.Boerrigter
231	corrections firmware manual	J.Boerrigter
232	8080 cassette routines SDK-85	J.van Ool
234	bootstrap for screen files	W.Dewinter
244	Programmes mathematiques	F.Duluins
247	Programgenerator	N.Looije
248	Shadeshape	H.di Ciris
249	INDATA news	INDATA
250	AZERTY user	J.Schepens
252	comparison between BASIC & mlp	G.Uliana
253	RUN linenumber BASIC V1.0 (t)	C.Dufour
254	new PADDLE-check routine	C.Dufour
256	screen-buffer buffer screen demo	C.De Bont
257	ON ERROR GOTO demo	C.De Bont
258	DAI VIDEO SCREEN RAM (translation)	UK
264	programming techniques (t)	UK
267	CONVERSION APPLE-ATARI-DAI (t)	UK
269	TV-tennis REM's (t)	UK
270	cursus microprocessoren part 1	A.Beuckelaers
276	Shape-design	W.Hermans
278	Copieervariaties	B.Vingerling

DAInamic subscription rates :

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

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

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

No part of this book may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the publisher.
Niets uit deze uitgave mag worden vervielvoudigd en/of openbaar gemaakt door middel van druk, fotocopie, microfilm of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

Software contributions

S O F T W A R E -< CONTRIBUTIONS - INZENDINGEN - BEITRAGEN >- S O F T W A R E

Its holyday-time. We want to get some rest. Please have some patience if you are waiting for some answer or reply.

Some questions asked by more than one will be answered right now.

1- Q: Is it possible to protect a program against unautherised copying ?
A: No. It is always possible to copy a program but we can make it rather difficult to do so. If you have a program of which you think it's good enough to be used by the club but would need some protection you can send it to us with this remark. If we agree, we will take care for the protection. The methods of protection we do use cannot be explained because someone who wants to break the protection will then be provided by just the facts he wants to know for this breaking.
However there are some protective measures you can do yourself; these are rather simple but gives the person who wants to change or copy the program more work to do. The measures will be discussed in one of the following issue's of DAInamic.

2- Q: Can I send my program for just comment. So I don't want give permission to use it in any way.
A: Yes. But if the program is good we will try to change your mind.

3- Q: Can I see a program before buying it
A: Yes. On the meetings where DAInamic is present with software you can see the new software and ask questions about it

4- Q: Can I have a program sometime before deciding to buy.
A: No

5- Q: Do I get a garantee if I send you software that you don't give it away?
A: The software that we receive is only distributed among those members of which you can expect answer. We automaticly assume that we can use the software for our library. If you tell explicity it is not to be used we obey. You can even ask for no distribution at all; in that case your software will stay with the person you send it to.

6- Q: Can I order software on floppy?
A: On this moment no. The floppy is still so uncommon that it would cost much to do so. Maybe in near future as many people have a floppydrive.

7- Q: How long will it take to get an answer?
A: Normaly (no holydays) you will have a letter notifying you that we did receive your software within a week. In that letter will be information on the person answering you. I try the completely handle it within two month, but that didn't work out always.

****

Videotex in Belgium

De staatssekretaris van PTT, mevrouw Paula d'Hondt heeft half april de modem vrijgegeven om zich niet tegen de technische evolutie op dat gebied te verzetten. Daarmee heeft zij dan ook een einde gemaakt aan een toestand, die door de gebruikers van het videotex-systeem als een klucht werd ervaren. De modem van de RTT huurde iedereen wel, maar gebruiken was een tweede. De grootste rem op de ontwikkeling van viewdata in België is daarmee in ieder geval weggenomen. Blijft alleen nog het relatief kleine probleem van de telefoonkosten.

"Wat is interaktieve videotex?" vraagt men zich af in een publikatie van de RTT. *"Videotex is een kommunikatiesysteem dat o.a. toelaat informatie, opgeslagen in een komputergeheugen te verspreiden en op te vragen. Daartoe heeft de gebruiker een telefoon nodig evenals een terminal bestaande uit een beeldscherm en een klavier. (...) Interaktieve videotex is een systeem met individuele tweewegtransmissie dat aldus een dialoog tussen de gebruiker en komputer toelaat. De opslagmogelijkheden van het hele systeem zijn slechts beperkt door de geïnstalleerde stockeringskapaciteit. 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 interaktie zijn ook bewerkingen en transakties mogelijk."* En: *"Voor het bedrijfsleven. Naast een informatiebron is videotex een goed kommunikatiesysteem om relaties met geografisch gespreide filialen, verdelers, kantoren of personeelskernen te onderhouden. Een snelle verspreiding van veelgebruikte en vaak gewijzigde gegevens naar vele medewerkers is dan mogelijk. Ook intern voor een groot bedrijf met vele afdelingen en diensten kan videotex de kommunikatieproblemen 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 komputers. Daarenboven kan op termijn een internationaal schakelnet tot stand komen.*

"Videotex is een ideaal publicitair medium voor grote verspreiding met attraktieve grafische mogelijkheden. De informatieleverancier kan op eenvoudige en efficiënte manier zijn informatie te koop aanbieden."

Tot zover de RTT. Nu de praktijk. Want het lijkt erop als je het RTT-proza leest dat er hier een nieuw, veelzijdig en uitgebreid medium, ter van iedereen gepaard te beschikking staat. Zonder gaan met problemen. Niets is minder waar. Er waren problemen in de vorm van het zich door de RTT toegeëigende monopolie op de modem, het apparaat dat gegevens van digitale in analoge vorm omzet, zodat het via de telefoonlijnen naar elders gestuurd kan worden, waar ook een modem staat, die het proces in omgekeerde volgorde doet. Het monopolie werd al in het 'Journal des Tribunaux' aangevallen (12-2-1983). De wetteksten hebben het enkel over een monopolie van de RTT op de telefoonlijnen, en niet op de apparaten die op die lijnen aangesloten kunnen worden.

Helemaal onoverkomelijk zou dat monopolie niet zijn geweest, als er maar goede modems geleverd werden, en juist daar zat het probleem. Terwijl een bedrijf als Barco schermen en terminals levert met een ingebouwde modem (Barco leverde een duizendtal viewdata-terminals aan de London Stock Exchange, en 500 aan het privénet van Jet-Air) ter grootte van een kwart velletje A4, leverde de RTT een modem, die door een van de door ons gekontakteerde personen, (die liever niet met naam en toenaam genoemd worden omdat het vrij delikate zaken zijn, en de viewdata-wereld een zeer kleine is) als een mastodont omschreven werd. Grootte bij benadering twee velletjes A4, hoogte een half velletje A4. Daarbij komt nog dat de door de RTT geleverde modem geen specifieke viewdata modem is, maar een voor komputers op lage snelheid. Op zich werkt hij wel goed, zeggen degenen die hem gebruiken, anderen weten het niet of nauwelijks. De modem staat er, en dat is alles.

Wat het nog erger maakte, was de exorbitante prijs voor de huur van de modem. Per 2 maand moesten de gebruikers 2.750 Bfr. neertellen, voor iets dat bij wijze van spreken in de kast stond. De kosten voor een in een Barco-terminal ingebouwde modem bedragen tussen de 1.000 en 2.000 frank. In het buitenland zijn de kosten voor de gebruiker van viewdata aanzienlijk lager. In Frankrijk levert de PTT voor 1.500 Bfr. huur per maand, de terminal inklusief de modem. De huur van de modem in Duitsland bedraagt 5 mark per maand, in franken 100. De vraag was ook wat men op het kabinet van de eerste minister zou doen, nu dat uitgerust wordt met 100 videotex-aansluitingen. Ingebouwde of externe, lees RTT-modems? Aanvankelijk wisten de andere gebruikers niet beter dan dat er RTT-modems op de Barco-terminals geschakeld zouden worden. Helaas, zei men toen, omdat zij anders als er moeilijkheden zouden komen naar de handelwijze van de eerste minister konden verwijzen. Onbekend is of het rekensommetje voor de huur alleen van de 100 modems op het kabinet van Martens de RTT ertoe aangezet heeft het apparaat vrij te geven. De dienst van de eerste minister zou dan 3.300.000 bfr. modemhuur moeten opbrengen.

Het gebruik van de RTT-modem was ook redelijk omslachtig. Men neme de telefoonhoorn in de handen, drukke op een knop, wachte tot men het signaal van de komputer hore. Dan de knop van de terminal indrukken, en de hoorn wegleggen. Dan moet nog het password ingegeven worden. Makkelijk is anders. De apparaten met ingebouwde modem hebben autodialing en geven automatisch het pass-word. Natuurlijk wist de RTT dat het met een ingebouwde modem makkelijker ging. Op de mediadag van de CVP, 4 december vorig jaar, had Barco volledig volgens de regels op de stand externe modems. Iets verderop stond de RTT mèt een Barco-terminal, maar zonder externe modem! Toen een van de andere gebruikers van videotex op een show stond (opnieuw geen namen) was de modem besteld. Men had daar twee mogelijkheden, of de externe RTT-modem, of de ingebouwde. De ingebouwde was wel veel sneller aangesloten, maar een technikus van de RTT kwam daar om de externe modem aan te sluiten. De technikus belde naar de RTT om te horen wat hij moest doen. Onze kontaktpersoon hoorde hem zeggen: *"Als hij betaald is, is er toch geen enkel probleem?"* Voor de duur van de tentoonstelling heeft de RTT-modem ongebruikt in de kast gestaan. Maar hij was betaald.

De verplichting de RTT-modem te gebruiken was ook een rem op een plan van Test-aankoop. Test-aankoop wilde op enkele plaatsen, bijvoorbeeld City 2, een corner-operator zetten, aangesloten op de data-

bank van Test-aankoop met konsumenteninformatie. Naar analogie ook van de viewdata automaten die in een groot aantal Nederlandse postkantoren te vinden zijn. Test-aankoop kon het echter niet doen, want de modem trok een streep door de rekening. Overigens, viewdata-automaten zijn ook niet te vinden in Belgische postkantoren.

Blijft het in vergelijking met de modem relatief kleine probleem van de telefoonkosten. In de zones 02 en 03 (Brussel en Antwerpen) geldt het lokale tarief, omdat daar de databanken geplaatst zijn (onder andere die van Editel in Brussel, en die van Bell-Telephone in Antwerpen), daarbuiten geldt het tarief van 5 fr. per 48 sekonden. Het komt erop neer dat de gebruikers meer aan de RTT in de vorm van telefoonkosten betalen dan aan de bedrijven waarvan zij informatie vragen (Editel bijvoorbeeld vraagt 3 fr. per minuut komputergebruik). In Engeland heeft 60 tot 70 % van de viewdata-gebruikers lokaal tarief.
Het hoeft dan ook geen betoog dat de markt van viewdata tot nu toe in België erg klein is. Partikulieren hebben er voor gepast 2.750 frank alleen voor de huur van de modem te betalen, daarbovenop komen dan nog de kosten voor de telefoon en het gebruik van de komputer. Viewdata wordt nu praktisch alleen in het bedrijfsleven benut, maar ook daar zijn de gebruikers op de vingers van twee handen te tellen: Editel (VUM), Bell Telephone, 3 Suisses, Test-aankoop, Jet-Air (echter buiten het videotex-systeem van de RTT), Telemedia, GBM, het Gemeentekrediet, Mediatel, en Sony (experimenteel). Geschat wordt dat er zo'n 200 tot 250 terminals geïnstalleerd zijn (Jet-Air niet meegerekend). Het gebruik van viewdata is voor de Regie TT nauwelijks kostenverzwarend. De telefoonlijnen zijn er, de ingebouwde modems zijn er ook.

Feitelijk hoeft de RTT alleen maar te inkasseren. Maar het vermoeden bestaat bij een enkeling dat dat nog te maken heeft met de nasleep van de Baudrin-affaire. Sinds die affaire zou niemand bij de RTT zijn nek nog uit durven steken. Pas als er iets zwart op wit van hogerhand komt, gaat het apparaat draaien.
Aan de viewdata-dienst bij de RTT ligt het niet. Een door ons geïnterviewde zei dat de dienst de dingen goed ziet zitten, en als ze eenmaal het groene licht zullen krijgen, dat alles dan goed in elkaar zit, omdat ze geleerd hebben van de ervaringen in Engeland, Nederland en Duitsland.
Dat is dan een magere troost.

AVP

bron: Pub 1/6/83

16 couleurs char

```
1     REM
3     REM
10    REM ***********************************************
20    REM *** S/P 16 COULEURS CARACTERES. LIGNES     ****
30    REM *** ADRESSEES PAR LA FN CURSOR(X,Y)       ****
40    REM *** SANS CALCULS COMPLIQUES !             ****
50    REM ***     (c) ALAIN MARIATTE 1983           ****
60    REM ***********************************************
70    REM
80    REM
85    MODE 0:PRINT CHR$(12):CF!=0.0:COLORT CF! 10 CF! CF!
87    PRINT :PRINT
100   INPUT "LIGNE TEXTE A ECRIRE:";A$:PRINT
110   INPUT "CURSOR (X,Y):";X!,Y!:PRINT :CURSOR X!,Y!:GOSUB 1000
111   REM
115   REM MODE 16 COULEURS (de CA a FA selon la taille)
120   POKE CB,#FA
121   REM
122   PRINT A$
125   REM
127   REM ON CHOISIT DES COULEURS  ALEATOIRES
130   FOR I!=1.0 TO LEN(A$)
140   POKE AC-3,(RND(16.0) SHL 4.0)+CF!
150   AC=AC-2
160   NEXT
170   REM
900   GOTO 100
990   REM
992   REM DETERMINER L'ADRESSE DU CONTROL BYTE & DU CURSEUR
994   REM
1000  CB=PEEK(#79)*256+PEEK(#78)
1020  AC=PEEK(#73)*256+PEEK(#72)
1030  RETURN
```

cassette tape lister

Als U, net als ik met een 'poor man's outfit' - bestaande uit: DAI pc, TV, Audio Cassette recorder (en ASR Teletype) - moet werken, dan is bovenstaand programma mogelijk ook voor U interessant.
Zeker, als U evenals ik de neiging hebt om vrij veel programma's op EEN bandje te zetten, waarna het punktueel noteren van WAT WAAR staat nogal eens te wensen over laat in de hitte van de strijd...

Genoemd programma is gebaseerd op een eerder in DAInamic verschenen List Programma, maar nu uitgebreid met een AUTOMATISCHE TELLERSTAND Berekening.
Vooral dit laatste is m.i. de grote kracht van deze versie, omdat alleen daarmee een werkelijk bruikbare Catalogus van een Bandje wordt verkregen.

HOE WERKT 'T?

Om de inleestijd niet al te groot te laten worden, heb ik uitvoerige tekst en uitleg in het programma zelf achterwege gelaten.
Daarom hier een toelichting op het principe:
Laat de Computer de TITELS van de programma's e.d. lezen, bereken de tijd tussen het lezen van twee opeenvolgende Titels, en leidt hieruit de bereikte Tellerstand af.
Omdat de Timer in de DAI niet loopt tijdens het eigenlijke inlezen, moeten we hiervoor corrigeren door ca 2.5 s bij te tellen voor elke ingelezen titel (ca 2.3 s synchronisatietoon + de tijd nodig voor 't inlezen van de Titel. Zie regel 310, waar de verstreken tijd in 20ms eenheden wordt berekend).
Daar de Afwikkelspoel geleidelijk aan sneller gaat draaien, moet de gemeten tijd omgerekend worden met de (benaderings-) formule voor de Tellerstand 'N' zoals deze voorkomt in regel 320, resp. 490.
Hierin is 'R' de straal van de VOLLE Afwikkelspoel; 'D' de Band-dikte voor een C60 cassette; (voor een C90 is dit ca 13 µm); en 'L' de TOTALE Lengte van de tape, gerekend vanaf het Begin van de Band (Tellerstand 000!) tot aan het begin van het Programma waarvan we de Tellerstand 'N' willen berekenen (zie ook regels 200 t/m 220).
De waarde van 'L' volgt uit: L! = T * V! * 2E-2 * F!
Daarbij is 'T' de in regel 310 berekende verstreken tijd in eenheden van 20 ms; 'V' is de bandsnelheid (in m/s), en 'F' een correctie factor, welke afhangt van het betreffende cassettebandje - zie hierna (aanvankelijk is de waarde van 'F' 1.0, reden waarom hij werd weggelaten in regel 320).
De faktor '0.6' is de overbrengingsverhouding tussen de Afwikkelspoel en de Teller (mogelijk is deze waarde afwijkend voor een ander merk dan de Philips N2225!)

Als we het programma starten zal dit eerst naar de Datum vragen, vervolgens naar de Naam van het Bandje, en tenslotte of het al een 'EOF' bevat.
Als we n.l. achter het laatste programma op de band een z.g. "End-Of-File Record" hebben staan dan kan daar op getest worden om te stoppen. (Het is anders niet (eenvoudig) mogelijk erachter te komen dat we het laatste programma op de band 'gezien' hebben..)
Ontbreekt het EOF record, dan biedt de computer aan er een voor ons te schrijven. Wensen we dit niet, of zijn we slechts ge-interesseerd in een beperkt aantal titels, dan kunnen we in plaats daarvan het Aantal Programma's opgeven.

Nadat we het bandje geheel teruggespoeld hebben en de TELLER op 000 hebben gezet, moeten we de Recorder starttoets en de Spatiebalk gelijktijdig indrukken. Het programma leest nu alle Titels, toont ze op het scherm, en

registreert zowel de verlopen Tijd (T), de berekende Tellerstand (N) als de gelezen Titel (A$) in de daartoe bestemde Arrays ("T(I)", "N(I)", "T$(I)")).

Als het gevraagde aantal Titels, resp. 'EOF' is gelezen, wordt een overzicht van alle geregistreerde titels getoond met hun Berekende Tellerstanden voor F!=1.0.
Vervolgens wordt gevraagd wat de WERKELIJKE stand behorend bij het als laatste getoonde Programma (resp. 'EOF') is. Hiermee wordt dan een iteratieve herberekening gemaakt (aanpassing van 'F!') om de zaak kloppend te maken (soms lukt dit niet precies voor alle tellerstanden, maar meer dan 1 à 2 nummers scheelt het zelden!).
Na de berekening wordt ter controle het gekorrigeerde overzicht weer getoond, en kan men desgewenst opnieuw een 'werkelijke waarde' ingeven, enz.

Tot slot moet 'O' worden gegeven om uit deze loop te raken en komt de vraag: "Print-out op de ASR (Y/N)?"
I.g.v. "Y" volgt de waarschuwing om de ASR On-Line te zetten, en wordt er na het indrukken van de Spatiebalk een Print-out gemaakt (zie voorbeeld).

N.B. Mocht men moeite hebben met de Lineariteit (nummers in middengebied te hoog of te laag), dan moet mogelijk de waarde van 'R!' (en/of 'D!') iets aangepast worden! (regel 240). Ook kan het zijn dat de bandsnelheid 'V!' niet precies 4.75 cm/s is.
Dit 'tunen' kan men heel eenvoudig doen door tijdens een 'BREAK' een of meer waarden te wijzigen, en na 'CONT' opnieuw de juiste 'werkelijke' waarde in te geven! (Zonodig herhalen tot de gewenste nauwkeurigheid is bereikt..)
Overigens kan men het uitprinten natuurlijk op elke willekeurige Printer laten doen. In dat geval moet alleen de Baudrate worden aangepast, en is het wachten na een Carriage Return (Lege loop met 'NUMBER' - geen WAITTIME, daar anders de timing verloren gaat!) meestal niet meer nodig.

cassette tape lister


```
10    REM CASSETTE-TAPE LISTER   V2.4
20    REM DØØR D. ASSINK;  83-05-20
30    CLEAR 15000:DIM X(0),N(100),T(100),T$(100)
35    EL$="                                ":EL$=EL$+EL$
40    CURS=20:AANT=100:I=1:BAUD=#C0:PØKE #FF05,BAUD
50    PØKE #131,0:PRINT CHR$(12);:AD$="":X9$=CHR$(9)
60    RESTØRE:FØR A=1 TØ 21:READ B:AD$=AD$+CHR$(B):NEXT
70    AD=PEEK(VARPTR(AD$)) IØR (PEEK(VARPTR(AD$)+1) SHL 8)+1
80    CURSØR 15,20:PPINT "CASSETTE TAPE LIST PRØGPAMMA"
90    PRINT TAB(15);"============================":PRINT :PRINT
100   PRINT "DIT PRØGRAMMA STELT DE INHØUDSØPGAVE VAN
      EEN CASSETTEBANDJE"
110   PRINT "SAMEN; WAARNA DIT ØP EEN ASR TELETYPE GEPRINT
      KAN WØRDEN.":PRINT
120   PRINT "EEN EVT. AFWIJKING TUSSEN DE BEREKENDE EN DE WERKELIJKE"
130   PRINT "TELLERSTAND KAN NØG VØØR HET UITPRINTEN WØRDEN
      GECØRRIGEERD."
140   PRINT :PRINT :IF T=0 THEN INPUT "DATUM";D$:PRINT
150   PRINT :INPUT "NAAM ØF NUMMER VAN HET BANDJE";N$:PRINT
160   PPINT :INPUT "IS HET BANDJE AL VØØRZIEN VAN 'EØF' RECØRD (Y/N)";
      A$:PRINT
170   IF A$<>"Y" THEN GØSUB 700
180   T=0:PRINT CHR$(12):PØKE #75,#20
190   PEM
200   PEM NU VØLGT DEFINITIE VAN: 'R' (RADIUS VØLLE SPØEL [MM]);
210   REM 'D' (DIKTE BAND [MICRØN] [C60]); 'V' (BANDSNELHEID [CM/S])
220   REM EN: 'F' (EXPERIM. CØRRECTIEFAKTØR VØØR GEM. BANDJE):
230   R!=24.7:D!=16.2:V!=4.75:F!=1.0:FF!=F!
240   R!=R!*1E-3:D!=D!*1E-6:V!=V!*F!*1E-2
250   CURSØR 0,23:PRINT "START RECØRDER, EN DRUK TEGELIJK
      ØP SPATIEBALK!";:GØSUB 600
260   B=#BFE7+239:PØKE #1BE,250:FØKE #1BF,255:CURSØR 0,21
270   PRINT " NR: TYPE:   FILE NAAM:"
280   CURSØR 0,23:PRINT EL$;
290   CURSØR 0,23:CALLM AD
300   TL=PEEK(#1BE):TH=PEEK(#1BF):PØKE #1BE,255:PØKE #1BF,255
310   T=T+255-TL+113+0.42*(#BFE7-B)+(255-TH)*256:REM TIMING
320   T(I)=T:L!=T*V!*2E-2:N=(R!-SQR(R!*R!-D!*L!/PI))/D!*0.6
330   N(I)=N:CURS=CURS-1:IF CURS=0 THEN CURS=19
340   CURSØR 0,CURS:PØKE #40,#30
350   PRINT EL$;:CURSØR 0,CURY:PRINT N;TAB(6);
360   A=#BFE7:A$=CHR$(PEEK(A))+"  "
370   A=A-2:IF PEEK(A)=#20 GØTØ 370
380   FØR B=#BF85 TØ A STEP 2:IF PEEK(B)=#20 THEN NEXT
390   IF A>B THEN FØR A=A TØ B STEP -2:A$=A$+CHR$(PEEK(A)):NEXT
400   T$(I)=A$:PRINT A$:I=I+1:IF I<=AANT AND RIGHT$(A$,3)<>"EØF"
      GØTØ 280:GØTØ 410
410   PØKE #FF05,BAUD:IF BAUD=1 THEN WAIT TIME 50
420   PRINT CHR$(12):PRINT "BAND: ";N$;TAB(45);"DATUM: ";D$:PRINT
430   PRINT " NR: TYPE:   FILE NAAM:":PRINT :IF BAUD=1
      THEN WAIT TIME 20
440   FLG=0:FØR K=1 TØ I-1:T=T(K):L!=T*V!*2E-2*F!
450   N=(R!-SQR(R!*R!-D!*L!/PI))/D!*0.6
460   PRINT N;TAB(6);T$(K):FØP J=1 TØ NUMBER:NEXT:NEXT K
470   PRINT :IF BAUD=1 GØTØ 560:INPUT "WERKELIJKE WAARDE VAN
      LAATSTE NUMMER (0 = EXIT)";N:PRINT
480   IF N=0 GØTØ 530:M=N:K=M:T=T(I-1)
490   F!=FF!*K/N(I-1):L!=T*V!*2E-2*F!:N=(R!-SQR(R!*R!-D!*L!/PI))/D!*0.6
500   IF N<M THEN K=K+1:FLG=1:GØTØ 490
510   IF N>M THEN IF FLG=0 THEN K=K-1:GØTØ 490
520   GØTØ 410
530   PRINT :INPUT "PRINT-ØUT ØP DE ASR (Y/N)";A$:PRINT
```

```
540   IF A$="Y" THEN GØSUB 800:GØTØ 410
550   GØTØ 570
560   PØKE #131,1:PRINT "ZET ASR ØFF-LINE!!":GØSUB 810
570   NUMBER=0:BAUD=#C0:PØKE #131,0:GØTØ 40
600   IF GETC<>0 GØTØ 600
610   IF GETC=0 GØTØ 610:RETURN
700   PRINT :INPUT "'EØF' RECØRD SCHRIJVEN";A$:PRINT
710   IF A$<>"Y" GØTØ 770
720   PRINT :PRINT "SPØEL CASSETTE NAAR GEWENSTE PØSITIE"
730   PRINT :PRINT "SET RECØRD,START TAPE,TYPE SPACE"
740   GØSUB 600:SAVEA X "EØF":PRINT
750   PRINT "SPØEL CASSETTE TERUG NAAR 000, EN TYP EEN SPATIE"
760   GØSUB 600:GØTØ 780
770   PRINT :INPUT "HØEVEEL PRØGRAMMA'S STAAN ER DAN (TENMINSTE) ØP";
      AANT:PRINT
780   RETURN
800   BAUD=1:NUMBER=1500:PRINT :PRINT "ZET ASR ØN-LINE!!"
810   ENVELØPE 0 15,15;0,12;:SØUND 0 0 15 0 FREQ(2000)
820   GØSUB 600:SØUND ØFF :RETURN
10000 DATA #F5,#C5,#D5,#E5,#01,#40,#00,#11,#B1,#80,#21,#9E
10010 DATA #E6,#CD,#CE,#02,#E1,#D1,#C1,#F1,#C9
```

Onderstaand een Voorbeeld van de Output zoals die op het scherm, en uiteindelijk op de Printer verschijnt:
(Bandje bevat een 'EOF' record op tellerstand 100)

```
BAND: DEMØNSTRATIE CASS.TAPE LISTER          DATUM: 83-05-20

 NR: TYPE:    FILE NAAM:

 5     0   CASSETTE TAPE LISTER  V2.4
 9     0   TEST PRØGPAMMA 1
 14    0   TEST PRØGPAMMA 2
 19    2   ARRAY XYZ
 24    2   ARRAY PQR
 29    0   BUDGET
 38    0   PLANNING  V1.1
 48    2
 57    0   MAIL  V2.3
 86    0   DØØLHØF  V2.2
 96    2   EØF

WERKELIJKE WAARDE VAN LAATSTE NUMMER (0 = EXIT)?100

BAND: DEMØNSTRATIE CASS.TAPE LISTER          DATUM: 83-05-20

 NR: TYPE:    FILE NAAM:

 5     0   CASSETTE TAPE LISTER  V2.4
 10    0   TEST PPØGPAMMA 1
 15    0   TEST PRØGRAMMA 2
 20    2   ARRAY XYZ
 25    2   APRAY PQR
 30    0   BUDGET
 40    0   PLANNING  V1.1
 50    2
 60    0   MAIL  V2.3
 90    0   DØØLHØF  V2.2
 100   2   EØF
```

← De Output zoals die de eerste maal op het SCHERM verschijnt.

← De HERBEREKENDE Output zoals die na ingave van de juiste tellerstand (hier dus: 100) voor het 'EOF' record ontstaat. Door hierna als 'werkelijke' waarde '0' te geven, wordt deze zelfde tekst uitgeprint.

>>>>>>Programmeer technieken<<<<<<<

Het probleem dat ik deze keer wilde bespreken is het toekennen van een waarde aan een variabele als de toegekende waarde weer op zijn beurt afhankelijk is van de waarde van een andere variabele. Dit komt in vele programma's voor en wordt vaak opgelost op de manier die ik hieronder laat zien :

```
150    IF A=5 THEN P=3:GOTO 200
160    IF A=6 THEN P=5:GOTO 200
170    IF A=7 THEN P=7:GOTO 200
180    IF A=8 THEN P=9:GOTO 200
190    P=0
200    .........
```

Dit is vanzelfsprekend een duidelijke methode, die zelfs in sommige gevallen de beste is. Maar als er veel mogelijkheden voor A en P zijn of als de toewijzing met een berekening uit te voeren is verdient dat toch de voorkeur, mits die berekening niet te ingewikkeld is.

Ik zal een aantal uitgewerkte voorbeelden geven hoe het in de meest voorkomende gevallen beter opgelost kan worden. Het betere van die oplossing zal een korter programma dat vaak sneller zal zijn; al hoeft dit laatste niet.
In tegendeel het programma zal zelfs trager kunnen worden; als normaal in bijna alle gevallen reeds aan de eerste of tweede conditie wordt voldaan zal mijn aanpak trager kunnen zijn.
Kiest u voor de 'oude' oplossing doe dit dan wel op een overdachte manier.
Het volgende voorbeeld zal dit duidelijk maken. We willen de getallen van 100 tot en met 300 onderzoeken op deelbaarheid door de priemgetallen onder de 20. (Hiermee vinden we de priemgetallen tussen deze grenzen.(19 is overbodig))
We gebruiken hiervoor het volgende programma: (intikken na IMPFPT)

```
5      WAIT TIME 1:POKE #1BE,#FF:POKE #1BF,#FF
10     FOR I=100.0 TO 300.0
20     IF I/2.0=INT(I/2.0) GOTO 110
30     IF I/3.0=INT(I/3.0) GOTO 110
40     IF I/5.0=INT(I/5.0) GOTO 110
50     IF I/7.0=INT(I/7.0) GOTO 110
60     IF I/11.0=INT(I/11.0) GOTO 110
70     IF I/13.0=INT(I/13.0) GOTO 110
80     IF I/17.0=INT(I/17.0) GOTO 110
90     IF I/19.0=INT(I/19.0) GOTO 110
100    PRINT I
110    NEXT
195    A=PEEK(#1BE):B=PEEK(#1BF):?(#FFFF-A-B*256.0)/50.0;" SEC"
```

Zoals u hopelijk weet zijn regel 5 en 195 bedoeld om de looptijd van dit programma te meten. Op mijn machine deed dit programma er 12.44 (6.46) seconden over. De tijd tussen haakjes is de tijd met math.chip aan. We zien simpel in dat bij de helft van de getallen reeds bij regel 20 naar de next gesprongen wordt. De hier gekozen testvolgorde is dus juist en logisch. Als we nu voor de demonstratie de regels 20 t/m 90 eens in omgekeerde volgorde nummeren zullen we zien dat de tijd enorm toeneemt nl 21.74 (11.12) seconden.

Men kan dit programma natuurlijk ook wel beter opzetten: (intikken na IMPINT)

```
5     WAIT TIME 1:POKE #1BE,#FF:POKE #1BF,#FF
10    FOR I=100 TO 300
20    IF I/19*19=I GOTO 110
30    IF I/17*17=I GOTO 110
40    IF I/13*13=I GOTO 110
50    IF I/11*11=I GOTO 110
60    IF I/7*7=I GOTO 110
70    IF I/5*5=I GOTO 110
80    IF I/3*3=I GOTO 110
90    IF I/2*2=I GOTO 110
100   PRINT I
110   NEXT
195   A=PEEK(#1BE):B=PEEK(#1BF):PRINT (#FFFF-A-B*256)/50.0
```

Werkt dit dan ook zullen sommigen zich misschien verwonderd afvragen ? Inderdaad dit werkt ook: als we een getal I eerst integer delen door bv 17 en dan weer vermenigvuldigen met 17 zal het getal afgerond worden op het grootste veelvoud van 17 dat kleiner of gelijk is aan het getal I. Als I dus deelbaar is door 17 zal aan de conditie I/17*17=I voldaan worden. Dit programma duurt 9.58 (6.56) seconden. De winst komt mijns inziens voornamelijk doordat er integer wordt gerekend, maar dit is niet te controleren omdat de 'truc' in floating point niet werkt. We zien echter wel een vreemde volgorde; we zetten dus de regels 20 t/m 90 weer in omgekeerde volgorde en krijgen 5.88 (4.4) seconden. Gaan we nu regel 30 bij regel 20 zetten en evenzo 50 bij 40 etc. komen we op 5.82 (3.74) seconden. Maar het kan nog steeds beter :

```
20    IF I/2*2=I THEN NEXT:GOTO 195
30    IF I/3*3=I THEN NEXT
40    IF I/5*5=I THEN NEXT
50    IF I/7*7=I THEN NEXT
60    IF I/11*11=I THEN NEXT
70    IF I/13*13=I THEN NEXT
80    IF I/17*17=I THEN NEXT
90    IF I/19*19=I THEN NEXT
100   PRINT I:NEXT
```

Regel 5,10 en 195 identiek aan vorige keer. Minder fraai doordat er bij een FOR maar liefst 9 NEXT'n staan en tevens reeds van te voren bepaald is dat het laatste getal door 2 deelbaar is. Om het programma algemeen te houden zou achter elke NEXT een GOTO 195 moeten staan. Maar dit zou alleen de intiktijd doen toenemen en niet de looptijd 5.7 (3.62) seconden. Er is echter nog winst te behalen door regel 100 te veranderen in :

```
100   PRINT I;:NEXT
```

Een kleine verandering maar de puntkomma achter de PRINT scheelt veel, we krijgen nu 4.96 (2.88) seconden.
U ziet een math chip versnelt wel (30 tot 50 %) maar goed programmeren loont vaak meer (75 %).

We zijn nog steeds bezig met de 'oude' versie en het wordt toch wel eens tijd om een echt andere aanpak te laten zien. Kijk nog eens naar het uitgangspunt.

Eerste probleem: A kan zijn 2 , 3 , 4 , 5 , 6 , 7 of 8 en in dezelfde volgorde moet P worden 12 , 15 , 18 , 21 , 24 , 27 of 30.
We zien vrij snel dat er een wiskundig verband bestaat tussen de waarde van A en de waarde van P. A loopt met 1 op als P met 3 oploopt; om dit verschil tot uitdrukking te laten komen zullen we A met 3 moeten vermenigvuldigen en om het dan geheel met elkaar in overeenstemming te brengen er nog 6 bij doen. We krijgen dan :

```
          OUD
150   IF A=2 THEN P=12
160   IF A=3 THEN P=15
170   IF A=4 THEN P=18
180   IF A=5 THEN P=21
190   IF A=6 THEN P=24
200   IF A=7 THEN P=27
210   IF A=8 THEN P=30
```

```
          NIEUW
150   P=A*3+6
```

Het oude programmadeel kan nog versneld worden door GOTO's aan het eind van de regels. Er is wel een verschil maar in de praktijk zal dat meestal niet een probleem zijn, 'oud' verandert P niet als aan geen der voorwaarden voldaan wordt

en 'nieuw' verandert P atijd.

Tweede probleem: iets moeilijker nu. Haast het zelfde maar nu kan A elke waarde hebben en P moet 9 worden als A kleiner dan 2 is en 33 als A groter dan 8 is.

```
          OUD
150 IF A<2 THEN P=9
160 IF A=2 THEN P=12
170 IF A=3 THEN P=15
::: :: ::: :::: ::::
220 IF A=8 THEN P=30
230 IF A>8 THEN P=33
```

```
          NIEUW
150   P=A*3+6
160   IF A<2 THEN P=9
170   IF A>8 THEN P=33
```

Nadeel van de nieuwe methode is het mogelijkerwijs voor niets uitvoeren van regel 150. Als in vele gevallen A kleiner is dan 2 is het dan ook beter regel 150 en regel 160 te verwisselen waarbij er dan natuurlijk wel een GOTO 180 moet komen achter de P=9. En zo zijn er nog wel wat varianten te verzinnen in geval A meestal een bepaalde waarde veel vaker dan andere aanneemt.

[
Derde probleem: we gaan nu het geval bekijken waar A regelmatig toemeent maar waar er geen eenvoudig wiskundig verband is tussen A en P, een berekening is dus lastig of onmogelijk.

```
          OUD
150   IF A=3 THEN P=7
160   IF A=4 THEN P=4
170   IF A=5 THEN P=15

180   IF A=6 THEN P=31
190   IF A=7 THEN P=76
200   IF A=8 THEN P=45
210   IF A=9 THEN P=29
```

```
          NIEUW
10    DIM P(9)
20    FOR I=3 TO 9:READ P(I):NEXT

150   P=P(A)

900   DATA 7,4,15,31,76,45,29
```

De voorgestelde nieuwe methode vertraagt het programma enigzins bij aanvang maar

compenseert dit later ruimschoots. Tel het aantal regels niet; het is een voorbeeld, als ik 'oud' uitbreid tot bv twintig regels zal 'nieuw' gelijk blijven of

met hoogstens een toenemen.

Vierde probleem: het geval waar P regelmatig toeneemt en A zich juist zeer onregelmatig gedraagt. Ook hier zullen we in de versie 'nieuw' gebruik maken van een

array.

```
          OUD
150   IF A=3 THEN P=2
160   IF A=5 THEN P=3
170   IF A=9 THEN P=4
180   IF A=12 THEN P=5
190   IF A=33 THEN P=6
   XT
200   IF A=42 THEN P=7
210   IF A=57 THEN P=8
```

```
          NIEUW
10   DIM A(8)
20   FOR I=2 TO 8:READ A(I):NEXT

150   FOR I=2 T 8:IF A(I)=A GOTO 160:NE

160   P=I
```

Vijfde probleem: er is totaal geen logica te ontdekken in de onderlinge verhoudingen van A en P nog in de waarde die zij hebben of krijgen. Dit is bijvoorbeeld het geval als A de ASCII-code is van een bepaalde toets en P de actiecode voor het programma. We gaan in de 'nieuwe' oplossing weer met een array werken. Er kan gekozen worden voor een twee-dimensionale array waar de A's e P's in paartjes bij elkaar staan of voor twee aparte array's; een voor de A's en een voor de P's.
De laatste mogelijkheid is misschien iets minder mooi maar werkt sneller.

```
          OUD
10   IF A=16 THEN P=7
20   IF A=17 THEN P=8

30   IF A=18 THEN P=15
40   IF A=19 THEN P=16
50   IF A=65 THEN P=0
60   IF A=66 THEN P=2
70   IF A=74 THEN P=-1
80   IF A=78 THEN P=99
90   IF A=83 THEN P=100
100  IF A=9  THEN P=5
```

```
          NIEUW
10   DIM A(10),P(10)
20   FOR I=1 TO 10:READ A(I),P(I):NEXT

50   FOR I=1 TO 10
60   IF A=A(I) THEN P=P(I):GOTO 80
70   NEXT
80   .........

900  DATA 16,7,17,8,18,15,19,16,65,0
910  DATA 66,2,74,-1,78,99,83,100,9,5
```

Nogmaals tel niet het aantal regels het gaat om het principe. En wat 'oud' gedaan wordt door de regels 10 t/m 100 wordt 'nieuw' gedaan door de regels 50,60 en 70. Indien gewenst kan dit ook nog in een regel gezet worden:

```
50   FOR I=1 TO 10:P=P(I):IF A=A(I) GOTO 60:NEXT
```

Toch kleven er ook wat nadelen aan de voorgestelde nieuwe methodes, waar we op moeten letten willen we er geen last van krijgen. De nieuwe regel 6 eindigt met GOTO 80; dit zal de snelheid ten goede komen maar de FOR-NEXT loop is dan vaak nog niet ten einde. Komt er dan een NEXT van een buitenloop zal die verkeerd worden geinterpreteerd. Dit kan voorkomen worden door NEXT I ipv alleen NEXT te gebruiken. De een-regel methode zal traag zijn: alle controles met behulp van array's en indien nog niet gevonden toch maar vast een toewijzing.
Daarnaast zal juist de ongeoefende programmeur het minder duidelijk zien wat de regel doet. Voor mij is het een voordeel bij het bekijken van inzendingen: Ik kan het programma korter maken en ik weet gelijk met wat voor soort programmeur ik hier te doen heb.
Ik zie echter in de inzendingen een duidelijke stijgende lijn zitten en ik vlei me met de gedachte dat dit mede door deze artikelenreeks komt

Frank H. Druijff

P.S. Het problem program is zowel door Nico P. Looije als door mij zelf nog aanzienlijk versneld.

The EDITOR story

This article describes the functioning of the BASIC-command 'EDIT'.
For details, the 'DAI pC FIRMWARE MANUAL' should be used.

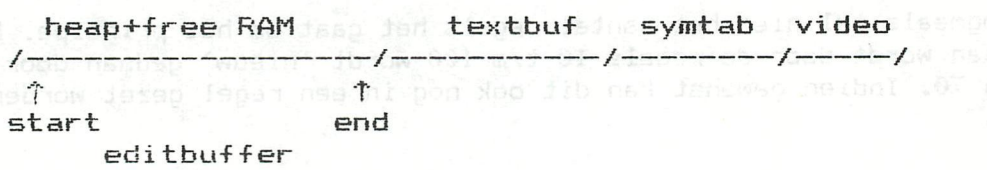
When the Basic command 'EDIT' is given (only in direct mode), the program jumps to address #0E1F5.

- Set up the editbuffer:

E1F5 E265 The EDIT buffer is set up. The screen mode 0 is selected (the screen is cleared). Via #D86D, an error message is given if insufficient memory for mode 0 is available.
If sufficient memory available, the heap and the symbol table are cleared (all variables are set

E26C to zero). Then the free RAM space is calculated, and the amount of heap-space is added to it. This gives the maximum available free RAM space for the editbuffer.

E275 Now the textbuffer and the symboltable are moved to the end of the free RAM. This leaves a large free RAM area between 'start of heap' and 'start textbuffer':



E27D The startaddress of the heap +2 is the startaddress of the editbuffer; the startaddress of the text-buffer is set as the end of the editbuffer.

E28E The output switch #0131 is set to the editbuffer.

E1F8 E19F Now the program is listed into the editbuffer. This is done in the same way as it is listed to the screen. So the contents of the editbuffer is a complete listing of the program in 'plain Basic text'.

E1FC Now the output switch #0131 is set back to the screen. The keyboard is enabled completely, and a '0' byte is added at the end of the editbuffer contents to indicate its end (all other bytes are <> 0).
The tab-table is disabled by clearing its pointer #00B4. Therefore, TAB's are normally not useable in the editbuffer.

E20B Here the screen-editor is initialised via RST5;

DATA #2A:

2EBF4 Again a mode 0 screen is set up and cleared, and all pointers for handling the cursor and the window (the visible part of the textbuffer on the screen) are cleared.

2EC17 On this cleared screen, one page (24 textlines) beginning at the start of the editbuffer, are printed.

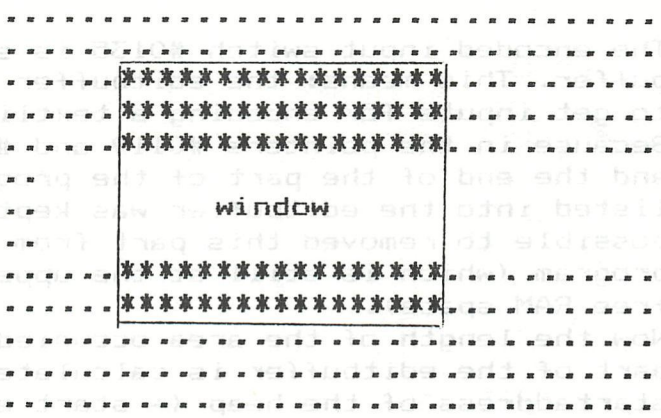
- Inputs into the editbuffer:

E20D Now the program enters a loop, in which inputs can be made. Depending on the key pressed, different actions are taken by the screen editor. Inputs can be stopped by pressing the BREAK-key.

E216 The inputs made are evaluated via RST5; DATA #2D:

2EC1E Via the cursor control keys, the cursor can be moved over the screen. The screen scrolls automatic-ly if the boundaries of the screen are reached. This means, that then the window is moved over the text in the editbuffer.

This 'window' can be seen as a rectangular looking glass, moved over a piece of paper. Only the part seen through this looking glass is visible:



The window can also be moved by depressing the cursor control keys together with the SHIFT-key. Now the cursor remains on the same text line (as long as the screen boundaries are not reached).

2EF4B Any other character typed in, is inserted in the editbuffer on the position of the cursor. The rest of the text is moved to make space for the new character.

2EFCC If the input character is 'CHAR DEL', the character at the current cursor position is removed from the buffer, and the rest of the text is moved one position.

E218 If an input character has been evaluated and eventually stored in the editbuffer, the next input character is awaited and obeyed. This process

continues until the 'BREAK' key is pressed.

- Return from EDIT:

E21B After a 'break', the screen is cleared. Then the next key input is awaited.

E222 If the second key depressed is again 'BREAK',
E24D all corrections made on the text in the editbuffer is ignored. In other words: the original program is restored.
The pointers for the heap, the textbuffer and the symbol table are set back to their original values, and the program is moved to its original position.
The editbuffer pointers and the contents of the editbuffer are not cleared. But because the edit-buffer was loaded from the lower end of the free RAM (incl. heap) onwards, it is overwritten by the original program.

E225 If after the first 'break',the second key is any other key except the BREAK-key, the modifications implemented in the editbuffer are now inserted in the original program.

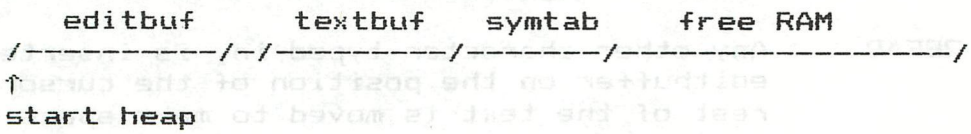
- Insert edited program lines:

E22A The encoded input switch #0135 is set to the edit-buffer. This means: the editbuffer is the source to get inputs for encoding a textline.

E231-E238 Because in the pointers #0119 and #011B the start and the end of the part of the program which was listed into the editbuffer was kept, it is now possible to removed this part from the original program (which is still at the upper end of the free RAM space).

E23B-E247 Now the length of the area occupied by the used part of the editbuffer is calculated: from the startaddress of the heap (= start editbuffer) to the inputpointer #00A4 of the editbuffer).

E248 Now the original program (minus the lines deleted) is moved to the end of the editbuffer:



E24B Now the Basic subroutine 'EDIT' is finished, and the program returns to the normal Basic monitor routine on #C818.

C84B Because the encoded input switch #0135 is set to
D879 the editbuffer, textlines are readed from this buffer.

- Inputs from the editbuffer:

E291 The startaddress of the editbuffer is stored in the encoding input pointer #0132. This pointer indicates the start of the textline for encoding.

E298 One character is taken from the editbuffer. If this character is not '0' (at the end of the buffer)

E2A0 the rest of the textline is read until a 'CR', which indicates the end of the textline.

E2A3 After a 'CR', the startaddress of the editbuffer is moved to the beginning of the next textline (making the editbuffer smaller every time), and the program returns to the Basic monitor.

C84E Now the textline read from the editbuffer is
C867 encoded (translated to the 'textbuffer code' and inserted in this buffer on the correct place). Via #C818, a next line from the editbuffer is fetched. This continues, until the end of the editbuffer (a '0') is found.

E2AA If all textlines of the editbuffer are encoded and inserted in the textbuffer, the encoding input switch #0135 is set back to the keyboard.

E2AD Now the editbuffer disappears: the textbuffer and the symboltable are moved to just after the heap and all pointers are updated.

Now further inputs can be made again from the keyboard: the program returns to the Basic monitor on #C818 - #C956.

(C) - Jan Boerrigter, June 1983

---

corrections firmware manual

The following corrections can be performed in the 'DAI pC FIRMWARE MANUAL':

- C84E JNC :C867 Encode textline if editbuf not empty

- C959 CALL :DDD2 Get first char from line ...

- Comments on subroutine CA57:

On entry is the pointer to the start of the name in the input not in register C, but in register B.

Jan Boerrigter, June 1983

Almelo, 16-6-1983

Geachte Heer,

Ondergetekende zendt U bij deze een complete sourcelisting van het beloofde programma, dat het mogelijk maakt dat alle microcomputers, welke werken op basis van de 8080,8085 of Z80 microprocessor kunnen communiceren met de DAIpc in machinetaal via de audio cassette-recorder.
Het complete programma bevat de schrijf- en leesroutines, die via de originele DAIpc-interface een machinetaalprogramma op cassette kunnen opnemen (CASSrc) en van cassette kunnen lezen (CASSrd), indien het micro-systeem van dezelfde interface wordt voorzien.
Wanneer de adressen waar het programma zich bevindt in een ander systeem voor andere doeleinden gebruikt worden, is het uiteraard mogelijk het geheel te verplaatsen. Dit zal voor de meeste DAIpc-gebruiker die de DNA-assembler kunnen hanteren niet moeilijk zijn.

Het nut van dit programma komt goed tot uiting in onderwijs situaties, waar de DAIpc draait als hulpmiddel voor het ontwikkelen van machine-taalprogramma`s door middel van de DNA-assembler (of de SPL-macro-ass.)
De door de DNA-assembler gegenereerde objectfiles m.b.v.#P. kunnen opgenomen worden op cassette en daarna in het micro-systeem via de leesroutine worden ingelezen. Dit bespaart de gebruiker het saaie intoetsen van de hex-codes.
Ondergetekende denkt hiermee vooral de DAIpc-gebruikers met een technische inslag een plezier te kunnen doen.

Mocht er in de toekomst belangstelling zijn voor een CHECK programma voor het testen van een opgenomen programma, dan zou ik dit gaarne van U horen.
Reeds eerder hebt U het schrijfprogramma toegezonden gekregen, daar het leesprogramma nog niet beschikbaar was. Dat schrijfprogramma kan worden vernietigd, daar het nu geleverde programma iets gewijzigd is.

Met genoegen ziet ondergetekende plaatsing in DAInamic tegemoet.

Met vriendelijke groeten van een noorderbuur,

HOOGACHTEND,

J.J.H. van OoL
Schierstins 19
7608 XZ Almelo. NL
Leraar electronica aan
de CHR. M.T.S te Almelo.

8080 cassette routines SDK-85

CASSrc V1.0 and CASSrd V1.0 on the SDK 85 microcomputerkit.

Before you can make use of the routines CASSrc and CASSrd in the SDK 85 microcomputer, it is necessary to program the routines in an EPROM (8755A) and install it at position A15 of the SDK 85 kit.

Further on the microcomputerkit has to be provided with the same cassette-interface as the DAIpc wich has to be connected to port 23H at b0 and b1.

The CASSrc routine makes it possible to record a Machine Language Program (MLP) on an audio cassette recorder in the same way DAIpc does. A MLP wich has been recorded from the DAIpc on audio-cassette, can be read into the SDK 85 microcomputerkit with the CASSrd-routine.
The way of recording and reading programs you can study in DAInamics Newsletter no. 14 and 15. (Part 1 and 2 from Mr. Jan Boerrigter)
The schematics of the cassette-interface you can find in DAInamic no.13 page 372.

The CASSrc and CASSrd routines are very usefull for anyone who wants to create objectprograms for microcomputersystems and therefore makes use of the DAInamic Assembler or perhaps Sphinx Macro Assembler on the DAIpc(eg. in technical tuition)
The routines can be easily made suitable for other microcomputersystems, based on 8085 compatible processors like 8080 and Z80.

How to use the CASSrc-routine ?

First you have to insert some data in the SDK 85 with the SUBSI.MEM.-function on addresses you can find in the heading of the sourcelisting.
The following data have to be inserted;

NAME : max. 16 hex.dec ASCII characters
Namelength (NMLG)
Lowest address of the MLP wich has to be recorded (LADR)
Destination address of the MLP (DADR). The destination address can be the same as the LADR, but it is also possible to load a program on other locations in other microsystems. So the DADR has to be different from the LADR.
Highest address of the MLP (HADR).

Program starting;

- audio-recorder on 'record' and start recorder.
- Type in : (GO) 0800 (EXEC)

Then the name of the routine (CASSrc) is shown in the SDK 85 display during recording of the MLP.
When recording is ready the display shows -80 85.
When the NAMElength is longer than 16 there is an error-reporting (-Err rc) in the display and no recording takes place.
At this moment there isn`t a CHECK-routine available, but of course it is possible to check your recording on the DAIpc with the CHECK-command in BASIC.

How to use the CASSrd-routine ?

Before starting the CASSrd-routine it is always necessary to insert the OFFSET with the SUBST.MEM.-function. When OFFS = 0000H, then the MLP is loaded on the address wich is equal to DADR on tape. When OFFS has a higher value the loading address becomes DADR + OFFS. So it can be adjusted to the microsystem wich is in use.

Program starting;

- Type in : (GO) 0900 (EXEC)
- Start recorder at the beginning of the MLP that you want to load.

After starting the CASSrd-routine it displays its own functionname during reading (loading) the MLP and when there are no reading-errors (-Err rd) the reading is ready if -80 85 appears in the display of the SDK 85.
When there is an error-reporting in the display (-Err rd) the loading is interrupted from the moment the fault has appeared. Check the volume-level on your recorder !
When there is a -Err rd error-reporting during reading your MLP, you must be sure, that the name of the program wasn`t too long at the moment you recorded your MLP on DAIpc. Otherwise it is not possible to load the program.

After loading the MLP without errors it is possible to examine all the data with the SUBSI.MEM.-function like NAME, NMLG, LADR, DADR, HADR, OFFS on the addresses you can find in the heading of the CASSrd-sourcelisting.

J.J.H. van OoL
Schierstins 19
7608 XZ Almelo.
Nederland

---

bootstrap for screen files

```
200   MODE 5A:PRINT CHR$(12):COLORT 8 0 0 8:COLORG 0 5 10 15
210   POKE #75,#5F:POKE #74,2:CLEAR #100
220   PRINT "*** DAINAMIC BOOTSTRAP LOADER V1.0 ex.***":PRINT
230   READ BGNADDR%,NMBR%
240   FOR ADDR%=BGNADDR% TO BGNADDR%+NMBR%-1
250   READ BYTE%:POKE ADDR%,BYTE%
260   NEXT:READ C%:IF C%<>#FFFF THEN PRINT "DATA READ ERROR":PRINT :END
270   PRINT "[ LOADING UTILITY FILE ]"
280   CALLM BGNADDR%
310   DATA #F800,95
320   DATA #31,#00,#F9,#21, #00,#00,#22,#00, #01,#01,#FF,#31, #CD,#CE,#02,#21
330   DATA #5D,#F8,#11,#5F, #F8,#CD,#D1,#02, #21,#00,#C0,#EB, #2A,#5D,#F8,#DC
340   DATA #D1,#02,#CD,#D4, #02,#D2,#A8,#D2, #21,#00,#01,#22, #9D,#02,#21,#EC
350   DATA #02,#22,#9B,#02, #CD,#B8,#DE,#CD, #5E,#DD,#01,#40, #F8,#C3,#92,#C8
360   DATA #AD,#01,#20,#18, #11,#5B,#20,#4C, #4F,#41,#44,#49, #4E,#47,#20,#42
370   DATA #41,#53,#49,#43, #20,#5D,#FF,#8B, #19,#00,#AD,#00, #87,#00,#00,#FFFF
```

```
10    COLORG PEEK(#BFFE) IAND #F PEEK(#BFFA) IAND #F PEEK(#BFF6) IAND #F
PEEK(#BFF2) IAND #F
20    IF PEEK(#BFF1)=#FF THEN MODE 5:GOTO 30
25    POKE #9D,#B:MODE 6
30    GOTO 30
```

8080 cassette routines SDK-85

```
**L
PAGE 01    SCHRIJFPROGRAMMA "CASSrc V1.0"

002             ********************************************************
003             *BRON       :8080 DNA-ASSEMBLER CHR.MTS. ALMELO.
004             *BESTEMMING :SDK 85   6.144 MHz CLOCK (T=326 nSEC)
005             *NAAM       :van OOL
006             *DATUM      :07-06-1983
007             *BUFFERDIMENSIE ASSEMBLER : 12
008             *TELLERSTAND BANDOPNAME VAN  004    TOT  020
009             ********************************************************
010             *TOELICHTING:
011             * HET PROGRAMMA SCHRIJFT DATA NAAR AUDIO-TAPE
012             * VOLGENS DE DAIpc-METHODE,ALS VOLGT:
013             *
014             *-VOORSIGNAAL (LEADER) CA. 1780 Hz (10%)
015             *-FLAGBYTE (55H) NAAR TAPE
016             *-FILE TYPE BYTE (31H)=MLP NAAR TAPE
017             *
018             *-LENGTE NAAM(2 BYTES) NAAR TAPE + UPD.CHKS.LENGTE
019             *-UPDATED CHKS LENGTEBYTES NAAR TAPE
020             *-NAAM NAAR TAPE + UPD.CHKS NAAM      (1340 Hz)
021             *-UPDATED CHKS VAN DE NAAM NAAR TAPE   (670 BAUD)
022             *
023             *-LENGTE DADR(2 BYTES) NAAR TAPE + UPD.CHKS.LENGTE
024             *-UPDATED CHKS VAN DADR-LENGTE NAAR TAPE
025             *-DADR NAAR TAPE + UPD.CHKS DADR
026             *-UPDATED CHKS VAN DADR NAAR TAPE
027             *
028             *-LENGTE DATABLOK(2 BYTES) NAAR TAPE+UPD.CHKS.LENGTE
029             *-UPDATED CHKS DATABLOKLENGTE NAAR TAPE
030             *-DATABLOK NAAR TAPE + UPD.CHKS.DATABLOK
031             *-UPDATED CHKS VAN DATABLOK NAAR TAPE
032             *
033             *-AFSLUITSIGNAAL(TRAILER) CA. 2140 Hz.
034             *
035             *INVOEREN m.b.v. SUBST.MEM-TOETS IN SDK 85
036             * :20B0    :NAAM V/H PROGRAMMA                )
037             *   T/M     IN HEXDEC ASCII-CODE               )
038             * :20BF                                        )NAME
039             * :20C0 : LOW-ORDER BYTE LENGTE VAN DE NAAM    )
040             * :20C1 :HIGH-ORDER BYTE LENGTE VAN DE NAAM    )NMLG
041             * :20C2 : LOW-ORDER BYTE LAAGSTE ADRES DATA    )
042             * :20C3 :HIGH-ORDER BYTE LAAGSTE ADRES DATA    )LADR
043             * :20C4 : LOW-ORDER BYTE BESTEMMINGSADRES DATA )
044             * :20C5 :HIGH-ORDER BYTE BESTEMMINGSADRES DATA )DADR
045             * :20C6 : LOW-ORDER BYTE HOOGSTE ADRES DATA    )
046             * :20C7 :HIGH-ORDER BYTE HOOGSTE ADRES DATA    )HADR
047             *
048             * OUTPUT: B0 VAN POORT 23H VIA INTERFACE
049             *         NAAR DINPLUG.
050             * PROGRAMMA-AANROEP:
051             * ZET TAPERECORDER OP OPNAME !
052             *   (GO):0800 (EXEC)
053             *
054             ********************************************************
055             NAME    EQU   :20B0
056             NMLG    EQU   :20C0
057             LADR    EQU   :20C2
058             DADR    EQU   :20C4
059             HADR    EQU   :20C6
060             TAPSL   EQU   :3F3F     TAPE SPEED LEADER   1:1
```

```
061                TAPSD   EQU   :693F     TAPE SPEED DATA     5:3
062                TAPST   EQU   :2A3F     TAPE SPEED TRAILER  2:3
063                FUNCT1  EQU   :0921     ROUTINE IN CASSrd
064                FUNCT2  EQU   :0939     ROUTINE IN CASSrd
065                        ORG   :0800     BEGINADRES IN EPROM SDK 85
066 0800 F3        CASSrc  DI              BLOKKEER ALLE INTERRUPTS
067 0801 31B020            LXI   SP,:20B0  INIT STACKPNT
068 0804 CD2608            CALL  WROPEN    OPENINGSROUTINE
069 0807 110200            LXI   D,:0002   DADR IS 2 BYTES LANG
070 080A 21C420            LXI   H,DADR
071 080D CD4A08            CALL  WRBLOK    DADR NAAR TAPE
072 0810 2AC620            LHLD  HADR      HADR IN (HL)
073 0813 EB                XCHG            HADR IN (DE)
074 0814 2AC220            LHLD  LADR      LADR IN (HL)
075 0817 13                INX   D         HADR+1
076 0818 7B                MOV   A,E       DATABLOK=(HADR+1)-LADR
077 0819 95                SUB   L         )
078 081A 5F                MOV   E,A       )
079 081B 7A                MOV   A,D       )
080 081C 9C                SBB   H         )
081 081D 57                MOV   D,A       ) (DE)=(DE)-(HL)
082 081E CD4A08            CALL  WRBLOK    DATABLOK NAAR TAPE
083 0821 CDE908            CALL  TRAILR    AFSLUITROUTINE
084 0824 FB                EI              INTERRUPTS TOEGESTAAN
085 0825 C7                RST   0         KLAAR: DISPL - 80 85
086 0826 3E0F      WROPEN  MVI   A,:0F     INITCODE POORTEN
087 0828 D320              OUT   :20       NAAR CSR:POORT 23H IS OUTP.
088 082A 32FF20            STA   :20FF     INITCODE NAAR STACK
089 082D CD2109            CALL  FUNCT1    CASS IN ADRESVELD
090 0830 CD3909            CALL  FUNCT2    rc IN DATAVELD
091 0833 CD9008            CALL  LEADER    VOORSIGNAAL
092 0836 3E55              MVI   A,:55     FLAGBYTE
093 0838 CDA608            CALL  WRBYTE    NAAR TAPE
094 083B 3E31              MVI   A,:31     FILE TYPE BYTE (ASCII=1)
095 083D CDA608            CALL  WRBYTE    NAAR TAPE
096 0840 2AC020            LHLD  NMLG      NMLG-ADRES
097 0843 EB                XCHG            NMLG IN (DE)
098 0844 CD7808            CALL  LGTST     TEST LENGTE V/D NAAM
099 0847 21B020            LXI   H,NAME    PNT NAAR BEGINADRES NAAM
100 084A CD5D08    WRBLOK  CALL  WRPNT     POINTERS NAAR TAPE
101 084D 0656              MVI   B,:56     INIT CHECKSUM
102 084F 7A        WBLOK   MOV   A,D       )
103 0850 B3                ORA   E         )BLOKTELLER
104 0851 CA7308            JZ    WRCHKS    BLOK=0? CHKS NAAR TAPE
105 0854 1B                DCX   D         VERLAAG BLOKTELLER
106 0855 7E                MOV   A,M       DATA NAAR ACCU
107 0856 23                INX   H         VERHOOG RAM-POINTER
108 0857 CD6C08            CALL  WDUPDC    DATA NAAR TAPE + UPDATE CHKS
109 085A C34F08            JMP   WBLOK     VOLGENDE DATABYTE
110 085D 0656      WRPNT   MVI   B,:56     INIT CHECKSUM
111 085F 7A                MOV   A,D       HIGH-ORDER-LENGTEBYTE
112 0860 CD6C08            CALL  WDUPDC    NAAR TAPE + UPD CHKS
113 0863 7B                MOV   A,E       LOW-ORDER-LENGTEBYTE
114 0864 CD6C08            CALL  WDUPDC    NAAR TAPE + UPD CHKS
115 0867 78                MOV   A,B       CHECKSUM BLOKLENGTE
116 0868 CDA608            CALL  WRBYTE    NAAR TAPE
117 086B C9                RET
118 086C CDA608    WDUPDC  CALL  WRBYTE    DATABYTE NAAR TAPE
119 086F A8                XRA   B         BEPAAL CHKS DATABYTE
120 0870 07                RLC             ROTEER CHKS
```

```
121 0871 47              MOV   B,A       CHKS IN B
122 0872 C9              RET
123 0873 78       WRCHKS MOV   A,B       CHKS IN A
124 0874 CDA608          CALL  WRBYTE    NAAR TAPE
125 0877 C9              RET
126 0878 7A       LGTST  MOV   A,D
127 0879 B7              ORA   A
128 087A C28408          JNZ   ERROR     NMLG TE GROOT
129 087D 7B              MOV   A,E
130 087E FE11            CPI   17        MAX.16 KARAKTERS
131 0880 D28408          JNC   ERROR     NMLG TE GROOT
132 0883 C9              RET
133 0884 AF       ERROR  XRA   A         00H=ADRESVELD
134 0885 47              MOV   B,A       00H=GEEN dp.
135 0886 219E03          LXI   H,:039E   ERRORTABELPNT
136 0889 CDB702          CALL  :02B7     Err IN DISPLAY
137 088C FB              EI              INTERRUPTS TOEGESTAAN
138 088D C36600          JMP   :0066     HAAL NIEUW COMMANDO OP
139 0890 213F3F   LEADER LXI   H,TAPSL   SIGNAALVERHOUDING 1:1
140 0893 01E807          LXI   B,:07E8   LEADERTIJD CA.2 SEC
141 0896 CDC308   TONE   CALL  WRBIT     LEADER NAAR TAPE
142 0899 0B              DCX   B         )LDRTIJDTELLER
143 089A 78              MOV   A,B       )
144 089B B1              ORA   C         )
145 089C C29608          JNZ   TONE
146 089F 213F69          LXI   H,TAPSD   SIGNAALVERHOUDING 5:3
147 08A2 CDC308          CALL  WRBIT     TAPSD NAAR TAPE
148 08A5 C9              RET
149 08A6 F5       WRBYTE PUSH  PSW
150 08A7 C5              PUSH  B
151 08A8 D5              PUSH  D
152 08A9 E5              PUSH  H
153 08AA 213F69          LXI   H,TAPSD
154 08AD 5C              MOV   E,H       H=E=3C KORT
155 08AE 55              MOV   D,L       L=D=64 LANG
156 08AF 0608            MVI   B,:08     BITTELLER=8
157 08B1 17       WBYTE  RAL             ROTEER DATABYTE
158 08B2 DCC308          CC    WRBIT     "1"=LANG/KORT
159 08B5 EB              XCHG            VERWISSEL LANG/KORT
160 08B6 D4C308          CNC   WRBIT     "0"=KORT/LANG
161 08B9 EB              XCHG            ZET K/L-PULS TERUG
162 08BA 05              DCR   B         VERLAAG BITTELLER
163 08BB C2B108          JNZ   WBYTE     VOLGENDE BIT
164 08BE E1              POP   H
165 08BF D1              POP   D
166 08C0 C1              POP   B
167 08C1 F1              POP   PSW
168 08C2 C9              RET
169 08C3 F5       WRBIT  PUSH  PSW
170 08C4 E5              PUSH  H
171 08C5 6C              MOV   L,H       L=H=KORT OF LANG
172 08C6 CDD608          CALL  OUTPUT    SYMM.PULS K OF L
173 08C9 E1              POP   H         HERSTEL TAPSD
174 08CA E5              PUSH  H         BEWAAR TAPSD
175 08CB 65              MOV   H,L       H=L=LANG OF KORT
176 08CC 7D              MOV   A,L
177 08CD D608            SUI   :08       (L)-8 (SYMM)
178 08CF 6F              MOV   L,A
179 08D0 CDD608          CALL  OUTPUT    SYMM. PULS
180 08D3 E1              POP   H
```

```
181 08D4 F1                 POP   PSW
182 08D5 C9                 RET
183 08D6 3E01       OUTPUT  MVI   A,:01      B0="1"
184 08D8 D323               OUT   :23        START POS.PERIODENTIJD
185 08DA 25         DEL1    DCR   H          VERLAAG TAPSL/D/T
186 08DB C2DA08             JNZ   DEL1       WACHT HALVE PERIODETIJD
187 08DE 2D                 DCR   L
188 08DF 2D                 DCR   L
189 08E0 2D                 DCR   L          (SYMMETRIE)
190 08E1 3D                 DCR   A          B0="0"
191 08E2 D323               OUT   :23        START NEG.PERIODENTIJD
192 08E4 2D         DEL2    DCR   L          VERLAAG TAPSL/D/T
193 08E5 C2E408             JNZ   DEL2       WACHT HALVE PERIODENTIJD
194 08E8 C9                 RET
195 08E9 213F2A     TRAILR  LXI   H,TAPST    SIGNAALVERHOUDING 2:3
196 08EC 010001             LXI   B,:0100    TIJDSDUUR TRLR
197 08EF C39608             JMP   TONE       GENEREER TRAILER
198 08F2            END     END
```

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

CASSrc	0800	DADR	20C4	DEL1	08DA	DEL2	08E4
END	08F2	ERROR	0884	FUNCT1	0921	FUNCT2	0939
HADR	20C6	LADR	20C2	LEADER	0890	LGTST	0878
NAME	20B0	NMLG	20C0	OUTPUT	08D6	TAPSD	693F
TAPSL	3F3F	TAPST	2A3F	TONE	0896	TRAILR	08E9
WBLOK	084F	WBYTE	08B1	WDUPDC	086C	WRBIT	08C3
WRBLOK	084A	WRBYTE	08A6	WRCHKS	0873	WROPEN	0826
WRPNT	085D						

```
0800 F3 31 B0 20 CD 26 08 11 02 00 21 C4 20 CD 4A 08
0810 2A C6 20 EB 2A C2 20 13 7B 95 5F 7A 9C 57 CD 4A
0820 08 CD E9 08 FB C7 3E 0F D3 20 32 FF 20 CD 21 09
0830 CD 39 09 CD 90 08 3E 55 CD A6 08 3E 31 CD A6 08
0840 2A C0 20 EB CD 78 08 21 B0 20 CD 5D 08 06 56 7A
0850 B3 CA 73 08 1B 7E 23 CD 6C 08 C3 4F 08 06 56 7A
0860 CD 6C 08 7B CD 6C 08 78 CD A6 08 C9 CD A6 08 A8
0870 07 47 C9 78 CD A6 08 C9 7A B7 C2 84 08 7B FE 11
0880 D2 84 08 C9 AF 47 21 9E 03 CD B7 02 FB C3 66 00
0890 21 3F 3F 01 E8 07 CD C3 08 0B 78 B1 C2 96 08 21
08A0 3F 69 CD C3 08 C9 F5 C5 D5 E5 21 3F 69 5C 55 06
08B0 08 17 DC C3 08 EB D4 C3 08 EB 05 C2 B1 08 E1 D1
08C0 C1 F1 C9 F5 E5 6C CD D6 08 E1 E5 65 7D D6 08 6F
08D0 CD D6 08 E1 F1 C9 3E 01 D3 23 25 C2 DA 08 2D 2D
08E0 2D 3D D3 23 2D C2 E4 08 C9 21 3F 2A 01 00 01 C3
08F0 96 08 20
```

```
LEESPROGRAMMA "CASSrd V1.0"

                ******************************************************
                *BRON       : 8080 DNA-ASSEMBLER CHR.MTS ALMELO
                *BESTEMMING : SDK85  6.144 MHz CLOCK(T=326 nSEC)
                *NAAM       : van OOL
                *DATUM      : 07-06-1983
                *TELLERSTAND BANDOPNAME VAN  050     TOT  075
                *BUFFERDIMENSIE ASSEMBLER : 12
                ******************************************************
                *TOELICHTING:
                * HET PROGRAMMA LEEST DATA VAN AUDIO-TAPE
                * VOLGENS DE DAIpc-METHODE. VERDER WORDEN
                * ONDERSTAANDE POINTERS IN HET GEHEUGEN
                * GEPLAATST, ZODAT EEN GELEZEN PROGRAMMA
                * ZONDER VEEL VOORBEREIDING OOK WEER KAN
                * WORDEN OPGENOMEN M.B.V. CASSrc V1.0 .
                * HET PROGRAMMA TEST NIET OF HET TE LADEN
                * PROGRAMMA PAST IN DE BESCHIKBARE
                * GEHEUGENRUIMTE VAN DE SDK 85 !
                * N.B.: DE OFFSET MOET VOOR HET STARTEN
                *       VAN HET CASSrd-PROGRAMMA WORDEN
                *       INGEVOERD. HET IN TE LEZEN PROGRAMMA
                *       WORDT OP EEN PLAATS IN HET
                *       GEHEUGEN(LADR) GEZET, WELKE GELIJK IS
                *       AAN (DADR+OFFSET)
                * DE OFFSET MAAKT HET MOGELIJK EEN PROGRAMMA,
                * INDIEN NOODZAKELIJK, OP EEN ANDER ADRES
                * IN TE LEZEN, DAN OP DE CASSETTE-
                * BAND STAAT. (LADR=DADR+OFFS)
                * INDIEN OFFS=0000,DAN IS (LADR=DADR)
                *
                * :20B0    :NAAM V/H PROGRAMMA              )
                *   T/M    IN HEXDEC ASCII-CODE             )
                * :20BF                                     )NAME
                * :20C0 : LOW-ORDER BYTE LENGTE V/D NAAM    )
                * :20C1 :HIGH-ORDER BYTE LENGTE V/D NAAM    )NMLG
                * :20C2 : LOW-ORDER BYTE LAAGSTE ADRES DATA )
                * :20C3 :HIGH-ORDER BYTE LAAGSTE ADRES DATA )LADR
                * :20C4 : LOW-ORDER BYTE BESTEMMINGSADRES   )
                * :20C5 :HIGH-ORDER BYTE BESTEMMINGSADRES   )DADR
                * :20C6 : LOW-ORDER BYTE HOOGSTE ADRES DATA )
                * :20C7 :HIGH-ORDER BYTE HOOGSTE ADRES DATA )HADR
                * :20C8 : LOW-ORDER BYTE OFFSET             )
                * :20C9 :HIGH-ORDER BYTE OFFSET             )OFFS
                *
                * INPUT: B1 VAN POORT 23H VIA INTERFACE
                *        NAAR DINPLUG
                * PROGRAMMA-AANROEP:
                *   (GO): 0900 (EXEC)
                * START DAARNA DE CASSETTERECORDER AAN HET
                * BEGIN VAN HET TE LADEN PROGRAMMA.
                ******************************************************
                NAME    EQU   :20B0     NAAMADRES
                NMLG    EQU   :20C0
                LADR    EQU   :20C2
                DADR    EQU   :20C4
                HADR    EQU   :20C6
                OFFS    EQU   :20C8
                ERROR   EQU   :0884     ROUTINE IN CASSrc
                LGTST   EQU   :0878     IDEM
```

```
061                     ORG   :0900     BEGINADRES IN EPROM SDK 85
062 0900 F3       CASSrd DI             GEEN INTERRUPTS TOEGESTAAN
063 0901 31B020         LXI   SP,:20B0  INIT STACKPNT
064 0904 CD5909         CALL  RDOPEN    OPENINGSROUTINE
065 0907 21C420         LXI   H,DADR    (HL)=DADR-ADRES
066 090A CD8D09         CALL  RDBLOK    LEES DADR VAN TAPE
067 090D 2AC420         LHLD  DADR      (HL)=DADR
068 0910 EB             XCHG            DADR IN (DE)
069 0911 2AC820         LHLD  OFFS      OFFSET IN (HL)
070 0914 19             DAD   D         LADR=DADR+OFFSET
071 0915 22C220         SHLD  LADR      LADR-ADRES IN SDK 85 RAM
072 0918 CD8D09         CALL  RDBLOK    LEES DATABLOK VAN TAPE
073 091B 2B             DCX   H         (HL)=HADR
074 091C 22C620         SHLD  HADR      HADR IN RAM
075 091F FB             EI              INTERRUPTS TOEGESTAAN
076 0920 C7             RST   0         KLAAR: DISPL - 80 85
077 0921 3E90     FUNCT1 MVI    A,:90     ADRESVELDCODE
078 0923 320019         STA   :1900     NAAR COMMANDREG.
079 0926 3E6C           MVI   A,:6C     C
080 0928 320018         STA   :1800     NAAR DISPLAY
081 092B 3E88           MVI   A,:88     A
082 092D 320118         STA   :1801     IDEM
083 0930 3E29           MVI   A,:29     S
084 0932 320218         STA   :1802
085 0935 320318         STA   :1803     S
086 0938 C9             RET
087 0939 3E94     FUNCT2 MVI    A,:94     DATAVELDCODE
088 093B 320419         STA   :1904     NAAR COMMANDREG.
089 093E 3EFA           MVI   A,:FA     r
090 0940 320418         STA   :1804     NAAR DISPLAY
091 0943 3E7A           MVI   A,:7A     c
092 0945 320518         STA   :1805     NAAR DISPLAY
093 0948 C9             RET
094 0949 3E94     FUNCT3 MVI    A,:94     DATAVELDCODE
095 094B 320419         STA   :1904     NAAR COMMANDREG.
096 094E 3EFA           MVI   A,:FA     r
097 0950 320418         STA   :1804     NAAR DISPLAY
098 0953 3E1A           MVI   A,:1A     d
099 0955 320518         STA   :1805
100 0958 C9             RET
101 0959 CD2109   RDOPEN CALL  FUNCT1    CASS IN DISPLAY
102 095C CD4909         CALL  FUNCT3    rd IN DISPLAY
103 095F 3E03           MVI   A,:03     INIT-CODE VOOR POORTEN
104 0961 D320           OUT   :20       POORT 23H IS INPUT
105 0963 32FF20         STA   :20FF     INITCODE NAAR STACK
106 0966 CDCF09   START  CALL  RDLEAD    LEES VOORSIGNAAL VAN TAPE
107 0969 CD230A         CALL  RDBYTE    LEES FLAGBYTE 55H VAN TAPE
108 096C DA6609         JC    START     CY=LEESFOUT
109 096F FE55           CPI   :55       TEST FLAGBYTE
110 0971 C26609         JNZ   START     TEST NIET OK?DAN OPNIEUW
111 0974 CD230A         CALL  RDBYTE    LEES FILE-TYPE BYTE VAN TAPE
112 0977 DA6609         JC    START     CY=LEESFOUT
113 097A FE31           CPI   :31       TEST FILE-TYPE
114 097C C26609         JNZ   START     31H=ASCII "1" = MLP
115 097F 21B020         LXI   H,NAME    ADRES PRGR-NAAM
116 0982 CD8D09         CALL  RDBLOK    LEES NAAM VAN TAPE
117 0985 CD7808         CALL  LGTST     TEST DE LENGTE V/D NAAM
118 0988 EB             XCHG            LENGTE NAAM IN (HL)
119 0989 22C020         SHLD  NMLG      NMLG IN RAM
120 098C C9             RET
```

```
121 098D CDAB09     RDBLOK  CALL  RDPNT     LEES LENGTEPNT VAN TAPE
122 0990 D5                 PUSH  D         BEWAAR LENGTEPOINTER
123 0991 DA8408             JC    ERROR     CY = LEESFOUT
124 0994 B7                 ORA   A         TEST CHKS
125 0995 C28408             JNZ   ERROR     CHKS FOUT
126 0998 0656               MVI   B,:56     INIT CHKS
127 099A 7A         RBLOK   MOV   A,D       )
128 099B B3                 ORA   E         )DATABLOKTELLER
129 099C CAC309             JZ    RDCHKS    DATABL.=0?LEES CHKS
130 099F 1B                 DCX   D         VERLAAG BLOKTELLER
131 09A0 CDBA09             CALL  RDUPDC    LEES DATA+UPD CHKS
132 09A3 DA8408             JC    ERROR     CY=LEESFOUT
133 09A6 77                 MOV   M,A       ZET DATA IN RAM
134 09A7 23                 INX   H         VERHOOG RAMPNT
135 09A8 C39A09             JMP   RBLOK     LEES VOLG. DATABYTE
136 09AB 0656       RDPNT   MVI   B,:56     INIT CHKS
137 09AD CDBA09             CALL  RDUPDC    LEES DATA+UPD CHKS
138 09B0 57                 MOV   D,A       HIGH-ORDER BYTE LENGTE
139 09B1 D4BA09             CNC   RDUPDC    LEES DATA+UPD CHKS
140 09B4 5F                 MOV   E,A       LOW-ORDER BYTE LENGTE
141 09B5 D4230A             CNC   RDBYTE    LEES CHKS LENGTE
142 09B8 90                 SUB   B         VERGELIJK CHKS EN UPD-CHKS
143 09B9 C9                 RET
144 09BA CD230A     RDUPDC  CALL  RDBYTE    LEES DATA VAN TAPE
145 09BD F5                 PUSH  PSW       BEWAAR DATABYTE
146 09BE A8                 XRA   B         BEPAAL CHKS
147 09BF 07                 RLC             ROTEER CHKS
148 09C0 47                 MOV   B,A       UPDATED CHKS IN B
149 09C1 F1                 POP   PSW       HERSTEL DATABYTE
150 09C2 C9                 RET
151 09C3 CD230A     RDCHKS  CALL  RDBYTE    LEES CHKS VAN TAPE
152 09C6 DA8408             JC    ERROR     CY=LEESFOUT
153 09C9 B8                 CMP   B         VERGELIJK CHKS EN UPDCHKS
154 09CA C28408             JNZ   ERROR     CHKS FOUT,DAN FOUTMELDING
155 09CD D1                 POP   D         HERSTEL LENGTEPOINTER
156 09CE C9                 RET
157 09CF 0628       RDLEAD  MVI   B,40      GESCHATTE PULSDUUR
158 09D1 DB23       NOSIGN  IN    :23       LEES POORT 23H
159 09D3 E602               ANI   :02       SELECTEER B1
160 09D5 C2D109             JNZ   NOSIGN    WACHT OP LDRSIGN
161 09D8 48                 MOV   C,B       GESCHATTE/WERK.PULSDUUR
162 09D9 1614               MVI   D,20      MINIMAAL AANTAL LDRPULSEN
163 09DB 1E00       CYCLE   MVI   E,0       LEESCYCLUS
164 09DD 1D         DELAY2  DCR   E         PULSDUURTELLER
165 09DE CAD109             JZ    NOSIGN    GEEN SIGN.?DAN OPNIEUW
166 09E1 DB23               IN    :23       LEES POORT 23H
167 09E3 E602               ANI   :02       SELECTEER B1
168 09E5 CADD09             JZ    DELAY2    B1="0",DAN WACHT
169 09E8 0600               MVI   B,0       INIT LDRCHECK
170 09EA 04         LDRCHK  INR   B         (B)=GEMETEN PULSDUUR
171 09EB CAD109             JZ    NOSIGN    GEEN SIGN.?DAN OPNIEUW
172 09EE DB23               IN    :23       LEES INPUTPOORT
173 09F0 E602               ANI   :02       SELECTEER B1
174 09F2 00                 NOP
175 09F3 C2EA09             JNZ   LDRCHK    WACHT OP LAAG SIGNAAL
176 09F6 78                 MOV   A,B       PULSDUUR IN (A)
177 09F7 91                 SUB   C         BEREKEN VERSCHIL PULSDUUR
178 09F8 F2FD09             JP    TEST%     PULSDUUR>STANDAARD?
179 09FB 2F                 CMA             PULSDUUR<STANDAARD? DAN
180 09FC 3C                 INR   A         BEPAAL POS.WAARDE
```

```
181 09FD 5F        TEST%   MOV   E,A      POS.WAARDE VERSCHIL IN (E)
182 09FE 79                MOV   A,C      STANDAARD PULSDUUR IN (A)
183 09FF E6F0              ANI   :F0      )
184 0A01 1F                RAR            )BEREKENING VAN TOEGESTANE
185 0A02 1F                RAR            )AFWIJKING (10%)
186 0A03 1F                RAR            ) (A)=04 IS 10% VAN 40 (C)
187 0A04 BB                CMP   E        VERSCHIL =TOEGESTANE ?
188 0A05 DA100A            JC    NOSYNC   VERSCHIL>10%,DAN NOSYNC
189 0A08 15                DCR   D        LDRPULSTELLER
190 0A09 C2DB09            JNZ   CYCLE    HERHAAL CYCLUS TOTDAT (D)=0
191 0A0C 14                INR   D        BIJ SYNC: (D)=01
192 0A0D C3DB09            JMP   CYCLE    GA DOOR TOT ASYNCHRONE PULS
193 0A10 15        NOSYNC  DCR   D        BIJ SYNC: (D)=00
194 0A11 C2D109            JNZ   NOSIGN   GEEN SYNC:DAN OPNIEUW
195 0A14 DB23      LOW     IN    :23      LEES ASYNCHRONE LDRPULS
196 0A16 E602              ANI   :02      SELECTEER B1
197 0A18 CA140A            JZ    LOW      WACHT OP HOOG SIGNAAL
198 0A1B DB23      HIGH    IN    :23      LEES ASYNCHRONE LDRPULS
199 0A1D E602              ANI   :02      SELECTEER B1
200 0A1F C21B0A            JNZ   HIGH     WACHT OP LAAG SIGNAAL
201 0A22 C9                RET
202 0A23 C5        RDBYTE  PUSH  B
203 0A24 D5                PUSH  D
204 0A25 E5                PUSH  H
205 0A26 1EFE              MVI   E,:FE    BYTETELLER
206 0A28 CD390A    RBYTE   CALL  RDBIT    LEES BIT VAN TAPE
207 0A2B DA350A            JC    POPRET   CY=LEESFOUT
208 0A2E 17                RAL            ROTEER BITWAARDE(B7) IN CY
209 0A2F 7B                MOV   A,E      BYTETELLER IN ACCU
210 0A30 17                RAL            BYTETELLER+BIT(S) IN (A)
211 0A31 5F                MOV   E,A      BEWAAR BYTETELLER +BIT(S)
212 0A32 DA280A            JC    RBYTE    VOLGENDE BIT.8 BITS=BYTE
213 0A35 E1        POPRET  POP   H
214 0A36 D1                POP   D
215 0A37 C1                POP   B
216 0A38 C9                RET
217 0A39 AF        RDBIT   XRA   A        RESET A,B,C,D EN CY
218 0A3A 57                MOV   D,A      INIT PERIODE-
219 0A3B 47                MOV   B,A      TIJDENDELAYS EN
220 0A3C 4F                MOV   C,A      BITWAARDETELLER IN (D)
221 0A3D 05        NEG1    DCR   B        BEGIN NEG.PERIODENTIJD
222 0A3E CA700A            JZ    SETCY    SIGNAALONDERBREKING?=CY
223 0A41 DB23              IN    :23      LEES POORT 23H
224 0A43 E602              ANI   :02      SELECTEER B1
225 0A45 CA3D0A            JZ    NEG1     NOG STEEDS NEG.?
226 0A48 0D        POS1    DCR   C        BEGIN POS.PERIODENTIJD
227 0A49 CA700A            JZ    SETCY    SIGNAALONDERBREKING?=CY
228 0A4C 15                DCR   D        VERLAAG BITWAARDETELLER(BW1)
229 0A4D DB23              IN    :23      LEES POORT 23H
230 0A4F E602              ANI   :02      SELECTEER B1
231 0A51 C2480A            JNZ   POS1     NOG STEEDS POS1?
232 0A54 010000            LXI   B,:0000  RESET (BC)
233 0A57 05        NEG2    DCR   B        BEGIN NEG.PERIODENTIJD
234 0A58 CA700A            JZ    SETCY    SIGNAALONDRBREKING?=CY
235 0A5B DB23              IN    :23      LEES POORT 23H
236 0A5D E602              ANI   :02      SELECTEER B1
237 0A5F CA570A            JZ    NEG2     NOG STEEDS NEGATIEF?
238 0A62 0D        POS2    DCR   C        BEGIN POS.PERIODENTIJD
239 0A63 CA700A            JZ    SETCY    SIGNAALONDERBREKING?=CY
240 0A66 14                INR   D        VERHOOG BITWAARDETELLER(BW2)
```

```
241 0A67 DB23           IN    :23     LEES POORT 23H
242 0A69 E602           ANI   :02     SELECTEER B1
243 0A6B C2620A         JNZ   POS2     NOG STEEDS POS2?
244 0A6E 7A             MOV   A,D      BW1>BW2:BITWAARDE="1" (B7)
245 0A6F C9             RET            BW1<BW2:BITWAARDE="0" (B7)
246 0A70 37       SETCY STC            ZET CY VOOR FOUTMELDING
247 0A71 C9             RET
248 0A72          END   END

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

CASSrd 0900   CYCLE  09DB   DADR   20C4   DELAY2 09DD
END    0A72   ERROR  0884   FUNCT1 0921   FUNCT2 0939
FUNCT3 0949   HADR   20C6   HIGH   0A1B   LADR   20C2
LDRCHK 09EA   LGTST  0878   LOW    0A14   NAME   20B0
NEG1   0A3D   NEG2   0A57   NMLG   20C0   NOSIGN 09D1
NOSYNC 0A10   OFFS   20C8   POPRET 0A35   POS1   0A48
POS2   0A62   RBLOK  099A   RBYTE  0A28   RDBIT  0A39
RDBLOK 098D   RDBYTE 0A23   RDCHKS 09C3   RDLEAD 09CF
RDOPEN 0959   RDPNT  09AB   RDUPDC 09BA   SETCY  0A70
START  0966   TEST%  09FD
```

```
0900 F3 31 B0 20 CD 59 09 21 C4 20 CD 8D 09 2A C4 20
0910 EB 2A C8 20 19 22 C2 20 CD 8D 09 2B 22 C6 20 FB
0920 C7 3E 90 32 00 19 3E 6C 32 00 18 3E 88 32 01 18
0930 3E 29 32 02 18 32 03 18 C9 3E 94 32 04 19 3E FA
0940 32 04 18 3E 7A 32 05 18 C9 3E 94 32 04 19 3E FA
0950 32 04 18 3E 1A 32 05 18 C9 CD 21 09 CD 49 09 3E
0960 03 D3 20 32 FF 20 CD CF 09 CD 23 0A DA 66 09 FE
0970 55 C2 66 09 CD 23 0A DA 66 09 FE 31 C2 66 09 21
0980 B0 20 CD 8D 09 CD 78 08 EB 22 C0 20 C9 CD AB 09
0990 D5 DA 84 08 B7 C2 84 08 06 56 7A B3 CA C3 09 1B
09A0 CD BA 09 DA 84 08 77 23 C3 9A 09 06 56 CD BA 09
09B0 57 D4 BA 09 5F D4 23 0A 90 C9 CD 23 0A F5 A8 07
09C0 47 F1 C9 CD 23 0A DA 84 08 B8 C2 84 08 D1 C9 06
09D0 28 DB 23 E6 02 C2 D1 09 48 16 14 1E 00 1D CA D1
09E0 09 DB 23 E6 02 CA DD 09 06 00 04 CA D1 09 DB 23
09F0 E6 02 00 C2 EA 09 78 91 F2 FD 09 2F 3C 5F 79 E6
0A00 F0 1F 1F 1F BB DA 10 0A 15 C2 DB 09 14 C3 DB 09
0A10 15 C2 D1 09 DB 23 E6 02 CA 14 0A DB 23 E6 02 C2
0A20 1B 0A C9 C5 D5 E5 1E FE CD 39 0A DA 35 0A 17 7B
0A30 17 5F DA 28 0A E1 D1 C1 C9 AF 57 47 4F 05 CA 70
0A40 0A DB 23 E6 02 CA 3D 0A 0D CA 70 0A 15 DB 23 E6
0A50 02 C2 48 0A 01 00 00 05 CA 70 0A DB 23 E6 02 CA
0A60 57 0A 0D CA 70 0A 14 DB 23 E6 02 C2 62 0A 7A C9
0A70 37 C9 00
```

```
10    PRINT CHR$(12):LIST 20-240:CALLM #D6DA
20    REM ################################################
30    REM ######### PROGRAMME ECRIT ET REALISE #########
40    REM #########            PAR             #########
50    REM #########                            #########
60    REM #########      Fabrice DULUINS       #########
70    REM #########                            #########
80    REM ################################################
90    REM ######### N.B :                      #########
100   REM #########                            #########
110   REM ######### JE CHERCHE CORRESPONDANTS  #########
120   REM #########   DE TOUT PAYS EN VUE      #########
130   REM #########  D'ECHANGES DE PROGRAMMES  #########
140   REM #########         INEDITS            #########
150   REM #########                            #########
160   REM ######### ECRIRE A :                 #########
170   REM #########                            #########
180   REM #########     Fabrice DULUINS        #########
190   REM ######### 4 allee de la Tour Renard  #########
200   REM #########    B - 1400 NIVELLES       #########
210   REM #########       BELGIQUE             #########
220   REM #########                            #########
230   REM #########                SPACEBAR    #########
240   REM ################################################
250   MODE 0:PRINT CHR$(12.0):PRINT "PRODUITS REMARQUABLES
 1"
260   PRINT "FORMULES DE SIMPSON                        2"
270   PRINT "MOYENNE ARITHMETIQUE                       3"
280   PRINT "CERCLE DETERMINE PAR TROIS POINTS          4"
290   PRINT "HARMONIC NUMBER                            5"
300   PRINT "LOGARITHME SUR N'IMPORTE QUELLE BASE       6"
310   PRINT "EQUATION DU SECOND DEGRE                   7"
320   PRINT "SYSTEME D'EQUATIONS A 2 INCONNUES          8"
330   PRINT "NOMBRE PREMIER ET PLUS PETIT FACTEUR       9"
340   PRINT "ALL GRAPHES                               10"
350   PRINT :PRINT "STOP                                   HALT":INPUT "VOTRE
CHOIX ";A$
360   PRINT CHR$(12):IF A$="1" GOTO 480
370   IF A$="2" GOTO 580
380   IF A$="3" GOTO 710
390   IF A$="4" GOTO 830
400   IF A$="5" GOTO 1020
410   IF A$="6" GOTO 1180
420   IF A$="7" GOTO 1300
430   IF A$="8" GOTO 1560
440   IF A$="9" GOTO 1740
450   IF A$="10" GOTO 1960
460   IF A$="HALT" THEN 470
470   END
480   PRINT CHR$(12):PRINT "          PRODUITS REMARQUABLES"
490   PRINT :PRINT "(A+B)^2 = A^2 + 2*A*B + B^2"
500   FOR I=1.0 TO NB
510   PRINT :PRINT "(A-B)^2 = A^2 - 2*A*B +B^2"
520   PRINT :PRINT "(A+B)*(A-B) = A^2 -B^2"
530   PRINT :PRINT "(A+B)^3 = A^3 + 3*A^2*B + 3*A*B^2 + B^3"
540   PRINT :PRINT "(A-B)^3 = A^3 - 3*A^2*B + 3*A*B^2 - B^3"
550   PRINT :PRINT "(A-B)*(A^2 + A*B + B^2) = A^3 - B^3"
560   PRINT :PRINT "(A+B)*(A^2 - A*B + B^2) = A^3 + B^3"
570   WAIT TIME 150:GOTO 250
580   PRINT "          FORMULES DE SIMPSON "
590   PRINT :PRINT "COS P + COS Q = 2 * COS (P+Q)/2 * COS (P-Q)/2"
600   PRINT :PRINT "COS P - COS Q = 2 * COS (P+Q)/2 * COS (P-Q)/2"
610   PRINT :PRINT "SIN P - SIN Q = 2 * SIN (P+Q)/2 * COS (P-Q)/2"
620   PRINT :PRINT "SIN P - SIN Q = 2 * COS (P+Q)/2 * SIN (P-Q)/2"
630   PRINT :PRINT "COS (A-B) = COS A * COS B + SIN A * SIN B"
```

```
640   PRINT :PRINT "COS (A+B) = COS A * COS B - SIN A * SIN B"
650   PRINT :PRINT "SIN (A+B) = SIN A * COS B + COS A * SIN B"
660   PRINT :PRINT "SIN (A-B) = SIN A * COS B - COS A * SIN B"
670   PRINT :PRINT "TG (A+-B) = (TG A +- TG B)/(1 -+ TG A * TG B)"
680   PRINT :PRINT "SIN 2*A = 2 * SIN A * COS A"
690   PRINT :PRINT "COS 2*A = COS^2 A - SIN^2 A = 1 - 2 *SIN^2 A = 2 * COS^
2 - 1"
700   WAIT TIME 150:GOTO 250
710   PRINT "           MOYENNE ARITHMETIQUE "
720   LET X=0.0
730   LET N=0.0
740   PRINT :PRINT "ENTREZ LE NOMBRE ";
750   INPUT W
760   IF W=0.0 GOTO 250
770   LET N=N+1.0
780   LET X=X+W
790   LET A=X/N
800   PRINT :PRINT " N= ";N;"  LE NOMBRE EST ";W
810   PRINT "LA MOYENNE EST DE ";A
820   GOTO 740
830   PRINT "           CERCLE DETERMINE PAR TROIS POINTS"
840   PRINT :INPUT "COORD DU 1e PT =";X1,Y1
850   PRINT :INPUT "COORD DU 2e PT =";X2,Y2
860   PRINT :INPUT "COORD DU 3e PT =";X3,Y3
870   LET A=(Y2-Y1)/(X2-X1)
880   LET B=(Y3-Y1)/(X3-X1)
890   LET C=((X2-X1)*(X2+X1))+((Y2-Y1)*(Y2+Y1))
900   LET D=C/(2.0*(X2-X1))
910   LET E=((X3-X1)*(X3+X1))+((Y3-Y1)*(Y3+Y1))
920   LET F=E/(2.0*(X3-X1))
930   LET Y0=(F-D)/(B-A)
940   LET X0=F-(B*Y0)
950   LET R=SQR((ABS(X3-X0))^2.0+(ABS(Y3-Y0))^2.0)
960   PRINT :PRINT "COORD DU CENTRE  X0,Y0 = ";X0;",";Y0
970   PRINT "RAYON = ";R
980   PRINT :PRINT "POUR CONTINUER TAPEZ SUR 1,SINON SUR 0"
990   INPUT L
1000  IF L=1.0 THEN GOTO 840
1010  GOTO 250
1020  PRINT "           HARMONIC NUMBERS"
1030  PRINT :PRINT "NOMBRE MAXIMUM DE TERMES"
1040  INPUT N
1050  LET K=0.0
1060  LET D=0.0
1070  PRINT :PRINT "NOMBRE DE TERMES ,VALEUR"
1080  FOR I=0.0 TO N
1090  LET K=I+1.0
1100  LET C=1.0/K
1110  LET D=D+C
1120  PRINT K,D
1130  NEXT I:PRINT
1140  PRINT "POUR CONTINUER TAPEZ SUR 1,SINON  SUR 0"
1150  INPUT L
1160  IF L=1.0 THEN GOTO 1020
1170  GOTO 250
1180  PRINT :PRINT "           LOG OF ANY BASE"
1190  REM BASE Y
1200  PRINT :INPUT " BASE = ";Y
1210  PRINT :INPUT " X = ";X
1220  J=LOG(X)/LOG(Y)
1230  PRINT :PRINT " LOG ";X;" EN BASE ";Y;" = ";J
1240  PRINT :PRINT "POUR CONTINUER TAPEZ SUR 1,SINON SUR 0"
1250  INPUT L
1260  IF L=1.0 THEN 1280
1270  GOTO 250
1280  PRINT
```

```
1290  GOTO 1200
1300  PRINT "        EQUATION DU SECOND DEGRE"
1310  PRINT :PRINT " AX^2 + BX + C =  0 "
1320  PRINT :INPUT "ENTREZ LES VALEURS DE A, B, ET C ";A,B,C
1330  D=((ABS(B)^2.0)-(4.0*A*C))/(4.0*ABS(A)^2.0):PRINT :PRINT " D= ";D
1340  IF D>=0.0 THEN 1410
1350  X=B/(2.0*A)
1360  Y=(SQR((4.0*A*C)-ABS(B)^2.0))/(2.0*A)
1370  PRINT :PRINT "Reponses complexes"
1380  PRINT "Part reelle     = ";X
1390  PRINT "Part imaginaire = ";Y
1400  GOTO 1500
1410  E=-B/(2.0*A)
1420  IF E>=0.0 THEN 1450
1430  Z=E-SQR(D)
1440  GOTO 1460
1450  Z=E+SQR(D)
1460  W=C/(Z*A)
1470  PRINT :PRINT "Reponses reelles"
1480  PRINT "1e reponse = ";Z
1490  PRINT "2e reponse = ";W
1500  PRINT "************************"
1510  INPUT "POUR CONTINUER SUR 1, SINON SUR 0 ";L
1520  IF L=1.0 THEN 1540
1530  GOTO 250
1540  PRINT
1550  GOTO 1320
1560  PRINT "          SYSTEME D'EQUATIONS A 2 INCONNUES"
1570  PRINT :PRINT " AX + BY = E "
1580  PRINT " CX + DY = F"
1590  PRINT :INPUT "ENTREZ LES PARAMETRES A,B,C,D,E,F";A,B,C,D,E,F
1600  M=(A*D)-(B*C)
1610  IF M=0.0 THEN 1670
1620  X=((E*D)-(B*F))/M
1630  Y=((A*F)-(E*C))/M
1640  PRINT :PRINT "SOLUTION "," X = ";X," Y = ";Y
1650  PRINT "****************************"
1660  GOTO 1690
1670  PRINT :PRINT "IL N'EXISTE PAS DE SOLUTION, OU PAS DE SOLUTION UNIQUE"
1680  PRINT "************************************************************"
1690  PRINT :INPUT "POUR CONTINUER TAPEZ SUR 1, SINON SUR 0 ";L
1700  IF L=1.0 THEN 1720
1710  GOTO 250
1720  PRINT
1730  GOTO 1590
1740  PRINT "            NOMBRE PREMIER "
1750  REM this program tests if a  number is prime
1760  REM it continues to cycle until zero is entered
1770  PRINT :PRINT "ENTREZ LE NOMBRE A TESTER, ZERO  POUR STOPPER"
1780  INPUT N
1790  IF N=0.0 THEN 1950
1800  IF N<4.0 THEN 1910
1810  I=0.0
1820  T=2.0
1830  J=INT(N/T)
1840  K=J*T
1850  IF N=K THEN 1930
1860  I=I+1.0
1870  L=T*T
1880  IF L>N THEN 1910
1890  T=(I*2.0)+1.0
1900  GOTO 1830
1910  PRINT :PRINT "  ";N;" EST PREMIER"
1920  GOTO 1770
1930  PRINT :PRINT N;" N'EST PAS PREMIER ";T;" EST LE PLUS PETIT FACTEUR"
1940  GOTO 1770
```

```
1950  GOTO 250
1960  PRINT "            ALL GRAPHES"
1970  PRINT "Entrez la fonction y=f(x) en 540"
1980  PRINT "Exemple:  EDIT 540  y=sin(x)"
1990  PRINT "Entrer ensuite RUN 505"
2000  INPUT "LA FONCTION EST-ELLE INTRODUITE ";O$:IF O$<>"O" THEN STOP
2010  CLEAR 15000:NB=100.0:DIM XV(NB),YV(NB):PRINT
2020  INPUT "ENTRER LES LIMITES DE VARIATION DE X";XL,XH
2030  PRINT :DR=15.0:BK=1.0
2040  XS=(XH-XL)/NB
2050  FOR I=0.0 TO NB-1.0
2060  X=XL+I*XS
2070  GOSUB 2200
2080  XV(I+1.0)=X
2090  NEXT
2100  MODE 5:FILL 0,0 XMAX,YMAX BK
2110  FOR I=1.0 TO NB
2120  YV(I)=YMAX*(YV(I)-YL)/(YH-YL)
2130  XV(I)=XMAX*(XV(I)-XL)/(XH-XL)
2140  X0=X1:X1=XV(I)
2150  Y0=Y1:Y1=YV(I)
2160  DRAW X0,Y0 X1,Y1 DR
2170  NEXT
2180  WAIT TIME 500
2190  GOTO 2250
2200  Y=EXP(-X)
2210  YV(I+1.0)=Y
2220  IF YH<Y THEN YH=Y
2230  IF YL>Y THEN YL=Y
2240  RETURN
2250  PRINT "POUR CONTINUER 1,POUR ARRETER 0"
2260  INPUT L
2270  IF L=1.0 THEN 2010
2280  GOTO 250
```

```
1     REM PROGRAMGENERATOR By N.P. LOOIJE 11/82
2     REM BASICLINE MUST BE HIGHER THAN THE GENERATOR !!!
3     REM useful for input of complex functions into BASIC while running a
4     REM program [e.g. 1000 Y=X^2+3*X+10]. You can generate a program in a
5     REM program by putting BASIClines in DATA statements READ them and
6     REM delete the DATAlines afterwards,and by using INPUT statements.
8     REM Expensive software such as.CORP,THE LAST ONE and AUTOCODE work this
9     REM way. This program makes all existing DATAstatementgenerators
10    REM oldfashioned because every BASIC statement can be generated.
11    REM Do not use this program within a FOR NEXT LOOP because FOR NEXT
12    REM loops use the symboltable which will be moved.
14    REM STUDY THE LISTING BEFORE AND AFTER THIS PROGRAM HAS RUN.
15    MODE 0:CLEAR 256:PRINT CHR$(12):GOSUB 200:REM INSTALL MLP
20    REM ----------------------GENERATE A PROGRAM-------------------------------
30    LIST 300-:READ LINE$:GOSUB 100:READ LINE$:GOSUB 100:REM GEN LINE 1000,1010
40    LINE$="300":GOSUB 100:LINE$="310":GOSUB 100:LIST 300-:REM DEL LINE 300,310
50    REM ---------------------- INPUT YOUR OWN LINES ----------------------------
60    INPUT "BASICLINE NO >310";LINE$:GOSUB 100
70    PRINT CHR$(12):LIST 310-:GOTO 60
99    REM ------------------CONVERT LINE$ TO BASIC LINE --------------------------
100   LINE$=LINE$+CHR$(13)+CHR$(0):A=VARPTR(LINE$):REM + dummy end (EDITbuffer)
110   A=PEEK(A)+PEEK(A+1)*256+1:B=A+LEN(LINE$):REM LINE$=begin & start EDITarea
120   POKE #A2,A MOD 256:POKE #A3,A SHR 8
130   POKE #A4,B MOD 256:POKE #A5,B SHR 8
140   POKE #135,2:CALLM #F800:RETURN:REM input from dummy EDITbuffer
199   REM -------------------MACHINELANGUAGE ---------------------------------
200   FOR D=0 TO 8:READ C:POKE #F800+D,C:NEXT:RETURN
210   DATA #C5,#CD,#79,#D8,#CD,#18,#C9,#C1,#C9: REM only 9 bytes!
299   REM -------------------DATA FOR BASICLINE -------------------------------
300   DATA 1000 REM THIS IS A PROGRAMGENERATOR EXAMPLE
310   DATA 1010 MODE 0:RETURN
```

program identification **Shadeshape**

title : SHADESHAPE

author : Hermano di Ciris

purpose : Generates impossible construction

comment :

```
5     REM SHADESHAPE by HERMANO DI CIRIS
10    MODE 6:COLORG 0 5 10 15:D%=300:R%=100
20    READ S%,T%:IF S%=0 THEN 40
30    READ X%,Y%:IF X%=0 THEN 20:DRAW S%,T% X%,Y% 23:S%=X%:T%=Y%:GOTO 30
40    WAIT TIME D%:COLORG 8 0 0 0:WAIT TIME D%
50    READ X1%,Y1%,X2%,Y2%,W%,C%:IF X1%=0 THEN 100:FOR I%=0 TO W%:DRAW X1%,Y1%-I
% X2%,Y2%-I% C%:NEXT:GOTO 50
100   WAIT TIME D%
110   COLORG 8 5 10 15:WAIT TIME D%:COLORG 8 1 0 0:WAIT TIME R%
120   COLORG 8 0 2 0:WAIT TIME R%:COLORG 8 0 0 3:WAIT TIME R%
140   GOTO 110
1000  DATA 83,127,52,145,52,162,220,255,235,246,235,211,265,228
1010  DATA 280,219,280,35,265,25,235,42,235,9,220,0,52,93,52,110
1020  DATA 83,127,0,0
1030  DATA 235,246,83,162,114,145,205,195,205,212,0,0,220,186,220,220,99,153,0,0
1040  DATA 52,110,205,25,205,59,114,110,99,102,205,42,0,0
1050  DATA 265,26,265,194,235,177,235,77,250,68,250,185,0,0
1060  DATA 205,212,205,195,114,145,0,0,205,195,220,186,0,0
1070  DATA 99,102,114,110,205,59,0,0,114,110,114,127,0,0
1080  DATA 235,211,235,195,280,219,0,0,83,127,99,136,52,162,0,0
1090  DATA 235,42,220,51,220,0,220,51,0,0,205,77,205,161,0,0
1100  DATA 250,68,250,51,220,68,220,169,145,127,205,94,0,0
1110  DATA 205,77,114,127,83,110,99,102,0,0,220,68,235,77,0,0
1120  DATA 220,186,129,136,145,127,0,0,0,0
2000  DATA 52,162,67,170,18,22,52,110,67,119,18,22
2002  DATA 220,86,250,68,18,22
2004  DATA 220,51,235,59,51,23,83,162,99,170,17,23
2006  DATA 83,110,145,145,17,23
2007  DATA 52,110,205,25,17,21,205,161,220,169,18,21
2009  DATA 205,161,220,152,152,21
2010  DATA 220,68,265,43,18,21,114,127,205,77,18,21
2015  DATA 250,185,265,194,18,21
2020  DATA 250,185,265,176,134,21,52,162,83,145,18,21
2025  DATA 83,145,99,136,16,21,205,212,220,220,18,21
2030  DATA 205,212,220,203,18,21
2035  DATA 220,255,235,246,18,22,67,171,220,255,18,22
2040  DATA 67,170,83,162,16,22,83,163,205,94,17,22
2050  DATA 83,111,204,43,18,22
2060  DATA 235,211,265,228,18,22,265,228,280,219,18,22
2070  DATA 67,119,205,195,17,22
2075  DATA 205,195,220,186,18,22,67,119,83,110,17,22
2080  DATA 145,144,265,211,17,23,265,211,280,219,185,23
2090  DATA 99,170,220,237,16,23,220,237,235,246,53,23
2100  DATA 220,169,235,177,101,23
5000  DATA 0,0,0,0,0,0
```

INDATA News

Nieuwe versie Dai Masterdos voor huidige floppy drives.

Er is een toevoeging aan Dai Masterdos aangebracht, dat het mogelijk maakt sectoren en tracks direkt te lezen en te schrijven.

Deze toevoeging werd gemaakt op algemeen verzoek en geeft de mogelijkheid tot het maken van een echte database.

de syntax is als volgt:

RREC filename.ext sect memplace(hex)
WREC filename.ext sect memplace(hex)

of als voorbeeld:

RREC TEST.BAS 1 5000 :

leest de eerste sektor van de file "test.bas" en zet deze informatie op hex 5000.
Alle namen en waarden kunnen ook variabelen zijn, zodat programmamatig deze waarden kunnen worden aangepast.

Prijs: Bf 500,-- voor een disk met uitleg.

SERVICE MANUAL

Voor de DAI Personal computer is er onlangs een service manual uitgegeven.

Dit manual geeft een zeer uitgebreide beschrijving van de hardware funkties, alsmede timing diagrammen, memory map, processor-, ram-, rom- en video beschrijvingen en verder alles wat een professioneel gebruiker moet weten om het interne van de machine goed te leren kennen.

En natuurlijk ook 16 bladen met het volledig schema van de computer.

Prijs: Bfr. 1500,--

GEBRUIK VAN AZERTY USER

1- Steek de cassette in de DCR en schakel de computer aan.
Nadat de DCR is gestopt kan U gewoon beginnen werken met het AZERTY toetsenbord !!!

2- Duwt U op reset zonder dat de cassette in de DCR zit,dan wordt Uw toetsenbord opnieuw als een QWERTY toetsenbord aanzien.

3- U kan echter zonder probleem terug een AZERTY toetsenbord bekomen indien U voordien gebruik hebt gemaakt van de USER cassette door vanuit BASIC in directe mode

```
CALLM #2F0
```

in te tijpen. Let echter op! De M bekomt U door op de vijfde toets van links op de onderste rij te drukken.
Dit is de plaats waar de QWERTY M zit.
Kijkt U altijd goed na wat op het scherm verschijnt.

4- Doe nooit een

```
G2F0
```

vanuit de utility-mode. Uw computer loopt dan gegarandeerd vast. Indien U in BASIC-mode een AZERTY toetsenbord heeft dan hebt U automatisch ook een AZERTY toetsenbord voor alle programma's in BASIC en in utility.

5- Het programma is niet als dusdanig te gebruiken met andere programma's die onder de HEAP zitten zoals bvb. het FGT programma. Indien U toch een aanpassing wenst om ook met dergelijke programma's te kunnen werken met een AZERTY toetsenbord, dan neemt U best contact op met mij. Vermeld steeds duidelijk om welk programma het gaat en geef ook de belangrijke gegevens zoals: begin & eindadres, entrypoint, versienummer van het programma, enz.
Vermeld ook een telefoonnummer indien U dit wenst. Hou er wel rekening mee dat ik U alleen tijdens het weekend kan terugbellen.(evt. prive-telefoon vermelden)

6- U kan een backup maken van het programma op de volgende manier:

```
*  REW              <ret>     terugspoelen nieuwe cassette
*  REW              <ret>     terugspoelen van de USER-cassette
>  UT               <ret>     naar uitility-mode gaan
>  R                <ret>     inlezen van de USER-file
>                             nieuwe cassette in de DCR steken
>  W2F0 37E USER    <ret>     copieren van de USER-file
>  B                          terug naar BASIC
*
```

??? Voor vragen,problemen en suggesties kan U terecht op
??? volgend adres:

Jos Schepens

Sint Jorisgilde 53
B-9330 DENDERMONDE (BELGIE)

??? Telefonisch ben ik te bereiken op het nummer 052/21 67 43
??? Alleen op zaterdag & zondag tussen 14.00 & 20.00

```
]#L.
    PAGE 01    USER-AZERTY 13-02-83 SCHEPENS V0.1

    002                 ************************************************
    003                 *           USER  AZERTY                       *
    004                 *                                              *
    005                 * THIS IS A PROGRAM TO RECONFIGURE              *
    006                 * A QWERTY KEYBD. TO AN AZERTY ONE.             *
    007                 * !!! USER-FILE ON DCR !!!                      *
    008                 * CREATED           12-02-83                   *
    009                 * LAST REVISION     13-02-83                   *
    010                 * COPYRIGHT BY      Jos Schepens DENDERMONDE   *
    011                 ************************************************
    012                 HEAP    EQU   :029B
    013                 HSIZE   EQU   :029D
    014                 RNEW    EQU   :DEB5
    015                         ORG   :2F0
    016 02F0 218003     ENTRY   LXI   H,AZEND+1
    017 02F3 229B02             SHLD  HEAP
    018 02F6 210001             LXI   H,:100
    019 02F9 229D02             SHLD  HSIZE
    020 02FC CDB5DE             CALL  RNEW
    021 02FF 3E10               MVI   A,:10
    022 0301 323D01             STA   :013D
    023 0304 210D03             LXI   H,AZTAB
    024 0307 2200F8             SHLD  :F800
    025 030A C383C7             JMP   :C783
    026 030D 303132     AZTAB   ASC   '012'
    027 0310 333435             ASC   '345'
    028 0313 363738             ASC   '678'
    029 0316 39                 ASC   '9'
    030 0317 2F4D3B             ASC   '/M;'
    031 031A 5E3A2D             ASC   '^:-'
    032 031D 0D                 DATA  :0D
    033 031E 514243             ASC   'QBC'
    034 0321 444546             ASC   'DEF'
    035 0324 474849             ASC   'GHI'
    036 0327 4A4B4C             ASC   'JKL'
    037 032A 2C                 DATA  :2C
    038 032B 4E4F50             ASC   'NOP'
    039 032E 415253             ASC   'ARS'
    040 0331 545556             ASC   'TUV'
    041 0334 5A5859             ASC   'ZXY'
    042 0337 572E5B             ASC   'W.['
    043 033A 20                 DATA  :20
    044 033B 000B10             DATA  :00,:08,:10
    045 033E 111213             DATA  :11,:12,:13
    046 0341 098000             DATA  :09,:80,:00
    047 0344 00                 DATA  :00
    048 0345 302122             ASC   '0!"'
    049 0348 232425             ASC   '#$%'
    050 034B 26                 ASC   '&'
    051 034C 27                 DATA  :27
    052 034D 28293F             ASC   '()?'
    053 0350 6D2B7E             ASC   'm+~'
    054 0353 2A3D               ASC   '*='
```

```
PAGE 02    USER-AZERTY 13-02-83 SCHEPENS V0.1

055 0355 0D                 DATA  :0D
056 0356 716263             ASC   'qbc'
057 0359 646566             ASC   'def'
058 035C 676869             ASC   'ghi'
059 035F 6A6B6C             ASC   'jkl'
060 0362 3C6E6F             ASC   '<no'
061 0365 706172             ASC   'par'
062 0368 737475             ASC   'stu'
063 036B 767A78             ASC   'vzx'
064 036E 79773E             ASC   'yw>'
065 0371 5D20               ASC   '] '
066 0373 000814             DATA  :00,:08,:14
067 0376 151617             DATA  :15,:16,:17
068 0379 0C8000             DATA  :0C,:80,:00
069 037C 004A53             DATA  :00,:4A,:53
070 037F           AZEND    END

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

AZEND  037F   AZTAB  030D   ENTRY  02F0   HEAP   029B
HSIZE  029D   RNEW   DEB5
```

```
100   REM
110   REM
120   REM ... A SPEED COMPARISON BETWEEN BASIC AND PLM
130   REM
140   REM
150   REM Gianni Uliana
160   REM Via Zuccoli 40    00137 ROMA      ITALY
170   REM
300   CLEAR 2000:DIM A(38.0)
310   FOR I%=0.0 TO 38.0:READ B%:POKE (#2F1+I%),B%:NEXT
320   REM ~~~~~~~~~~~~~~~~~~~~ BASIC ~~~~~~~~~~~~~~~~~~~~
330   PRINT CHR$(12):CURSOR 18,12:PRINT " BASIC  "
340   WAIT TIME 100:PRINT CHR$(12)
350   I=#B3E7
360   SOUND 1 1 1 0 FREQ(138.0)
370   FOR X=I TO I+131.0 STEP 2.0
380   POKE X,N
390   N=N+1.0:IF N>255.0 THEN N=0.0
400   NEXT
410   IF I<#BF69 THEN I=I+134.0:GOTO 370
420   SOUND OFF :WAIT TIME 100
430   REM ~~~~~~~~~~~~~~~~~~~~ PLM  ~~~~~~~~~~~~~~~~~~~~
440   PRINT CHR$(12):CURSOR 24,12:PRINT " PLM "
450   WAIT TIME 100:PRINT CHR$(12)
460   SOUND 1 1 1 1 FREQ(1975.0)
470   CALLM #2F1
480   SOUND OFF :WAIT TIME 100
490   GOTO 320
500   DATA #C5,#D5,#E5,#F5,#16,#3E,#3E,#00,#1E,#17,#01
510   DATA #EB,#BF,#02,#3C,#0B,#0B,#15,#C2,#FE,#02,#16
520   DATA #0A,#0B,#15,#C2,#08,#03,#16,#3E,#1D,#C2,#FE
530   DATA #02,#F1,#E1,#D1,#C1,#C9
```

RUN (Numero de ligne) AVEC LE BASIC V1.0

Avec le BASIC V1.0 , le redémarrage d'un programme à partir d'un numero de ligne donné peut etre a l'origine d'une erreur non recuperable à l'execution , la table des symboles et la zone de stockage des variables ayant été vidées.

La solution suivante est proposée par G.GRUITERS :

1) En utilisant la procedure S (substitute) en UT,tapez en code objet le court programme en MLP appelé RUNLIN :

```
23 23 56 23 5E EB CD F6 CA 44 4D CD 01 E4 21 00
00 22 15 01 AF 32 26 01 31 00 F9 B7 C3 8F C8
```

2) Ajustez le pointeur de pile 29B-29C,mettre le pointeur après la dernière adresse du code objet.

3) Tapez NEW

4) Chargez votre programme BASIC et tapez RUN

5) Pour redemarrer à un certain numero de ligne,definir d'abord une variable entiere a laquelle on donne la valeur du numero de ligne desire , suivi par un appel au MLP.

EXEMPLE : Supposons que l'adresse de depart du MLP soit #300. Redemmarage a la ligne 100

Tapez les commandes suivantes en mode direct :

```
LINE%=100:CALLM #300,LINE%
```

NOTA :

- On peut charger automatiquement le MLP et le programme BASIC en utilisant le DAINAMIC BOOTSTRAP LOADER. Dans ce cas on peut omettre les etapes 2 et 3.
- Si on redemarre souvent au meme numero de ligne, incorporer la variable au programme BASIC , par exemple:

```
10 LINE%=100
```

et redemarrer en faisant seulement CALLM #300,LINE%

CATALOGUE (Prix indicatifs en FF)

	AUDIO	DCR
GAMES COLLECTION 1	60 Frs	80 Frs
GAMES COLLECTION 2	60 Frs	80 Frs
GAMES COLLECTION 3	60 Frs	80 Frs
GAMES COLLECTION 4	120 Frs	140 Frs
GAMES COLLECTION 5	60 Frs	80 Frs
GAMES COLLECTION 6	110 Frs	130 Frs
GAMES COLLECTION 7	110 Frs	130 Frs
GAMES COLLECTION 8	110 Frs	130 Frs
GAMES COLLECTION 9	110 Frs	130 Frs
GAMES COLLECTION 10	110 Frs	130 Frs
GAMES COLLECTION 11	110 Frs	130 Frs
NEWSLETTER 10	75 Frs	95 Frs
NEWSLETTER 11-12	95 Frs	115 Frs
NEWSLETTER 13-14-15	95 Frs	115 Frs
TOOLKIT 1	150 Frs	170 Frs
TOOLKIT 2	150 Frs	170 Frs
TOOLKIT 3	150 Frs	170 Frs
TOOLKIT 4	150 Frs	170 Frs
DRIVER	90 Frs	110 Frs
SUPER INVADER	90 Frs	110 Frs
CENTIPEDE	90 Frs	110 Frs
ACROBATES	90 Frs	110 Frs

DU NOUVEAU POUR LES PADDLES !!!

Par cet article je vais tenter de vous decrire l'organisation materielle et logicielle des PADDLES . Elle debouche sur une routine de conversion Analogique/ Digitale plus perfomante que celle habituellement utulisée.

1) Principe de base (Simplifié)

Chaque paddle est branchée sur un MVM [1] (1/2 556). Quand une impulsion est envoyée sur la broche Trigger [2] de ce circuit, par l'intermediaire de l'adresse 0FD01H, sa sortie change d'etat pendant une durée "d" précisement déterminée par un reseau RC (Resistance-Condensateur). Or la resistance dependant de la position du potentiometre du paddle la durée "d" est donc elle aussi fonction de cette position. Le changement d'etat est transmis à un circuit (8253) qui se met à decompter. Au bout de la periode "d" le MVM revient a son etat initial ce qui a pour effet de stopper le decomptage sur une valeur precise. Il ne reste plus qu'a lire et a mettre cette valeur dans un format convenant à l'utilisateur.

2) Rapport de Temps

Il donne une idée des differentes durée qui peuvent etre en raport avec le temps de la conversion.

a) Le microprocesseur

L'horloge du processeur tourne a 2 MHZ (On trouve un quartz de 18 MHZ dont la frequence est divisée par 9 par un circuit specialisé : le 8224). Nous supposerons donc que la durée d'un cycle [3] est de 0,5 µs (En fait cette durée peut varier en fonction du temps d'accés aux memoires). Une instruction telle que JMP,par exemple, qui prend 10 Cycles mettrait donc 5 µs a etre executée.

b) Les paddles

La durée de l'impulsion du MVM est donné par la relation :

$$1{,}1 \times R \times C$$

Sur le DAI le reseau RC est composé d'un condensateur de 1,2 nF et d'une resistance de 15 Kohms en série avec un potentiometre de 100 Kohms. Après calcul on arrive aproximativement aux resultats suivants :

Durée minimum (R= 15 Kohms)	20 µs
durée maximum (R= 115 Kohms)	150 µs

La durée maximum de la conversion est donc proche de 150 µs ce qui represente 300 cycles d'horloge.

3) La routine de conversion du DAI

Le programme se presente ainsi :

```
0000            ;Le numero du paddle est selectionne grace aux
0000            ;trois premier bits de l'adresse 0FD06H.
0000            ;Le quatrieme bits doit etre a 1 pour autoriser
0000            ;la conversion . Attention car cette adresse
0000            ;contient aussi le commutateur des banques
0000                    ORG     0EBD3H          ;Banque0
EBD3  3E30              MVI A   30H             ;Initialisation du circuit 8253 en
EBD5  0106FC            LXI B   0FC06H          ;mode 'decompteur'
EBD8  02                STAX B
EBD9  21FFFF            LXI H   0FFFFH          ;Le decompteur est charge avec la
EBDC  2200FC            SHLD    0FC00H          ;plus grande valeur possible
EBDF  3A01FD            LDA     0FD01H          ;'lance' le MVM (Trigger)
EBE2  EB        PDL10   XCHG
EBE3  3E00              MVI A   0H
EBE5  02                STAX B
```

```
EBE6  2A00FC        LHLD    0FC00H      ;lit le contenu du compteur et le
EBE9  CD14DE        CALL    0DE14H      ;compare avec sa valeur precedente
EBEC  DAE2EB        JC      PDL10       ;tant qu'elle est inferieur continue
EBEF  CD26DE        CALL    0DE26H      ;quand elle est fixe le decompteur est
EBF2  11CEFF        LXI D   0FFCEH      ;alors arrete
EBF5  19            DAD D               ;Apres ce calcul H contient la valeur
EBF6                END                 ;du PDL . . . ect
```

Nota :Le programme contient une boucle d'attente. Il suffit de lire l'adresse 0FD01H pour envoyer l'impulsion de depart (Trigger).

4) Modification proposée

Dans des programmes ou les durées seront critique, on pourra reduire le temps d'éxecution de la routine de facon appreciable. On aura :

```
0000                ORG     5000H
5000  3E30           MVI A   30H
5002  0106FC         LXI B   0FC06H
5005  02             STAX B
5006  21FFFF         LXI H   0FFFFH
5009  2200FC         SHLD    0FC00H
500C  3A01FD         LDA     0FD01H
500F          ; A la place de la boucle d'attente il suffit
500F          ; de placer ici une partie de votre programme
500F          ; comprenant au moins une soixantaine d'instructions
500F          ; pour laisser au decompteur le temps de s'arreter
500F          ; Il n'y a pas de registres a sauver, il faut
500F          ; seulement veiller a ne pas utiliser le premier
500F          ; oscillateur du 8253 (0FC00H/0FC01H)
500F          ;       XRA A
500F  3206FC          STA     0FC06H
5012  2A00FC          LHLD    0FC00H
5015  CD26DE          CALL    0DE26H
5018  11CEFF          LXI D   0FFCEH
501B  19              DAD D
501C  7C              MOV A,H             ;HL contient la valeur du PDL
501D          ; ... ect
501D                  END
```

Nota :Il n'y a pas de limite maximale à respecter quant au nombres d'intructions que l'on peut mettre à la place de la boucle. En MOYENNE une intrution prend 6,5 cycles ce qui justifie le choix de 60 et respecte une bonne marge de 'securité'

Bonnes Conversions !
Cedric DUFOUR

Ouvrages de references :

Le FIRMWARE Manual par B.J. Boerrigter
DAI p.c. Schematics idem

1 Un MVM (MultiVibrateur Monostable) est un montage qui fournit à chaque impulsion d'entrée une impulsion de sortie d'une durée précise qui ne depend pas de celle d'entrée.

2 Trigger est la broche d'entrée du MVM (Voir 1)

3 On peut dire qu'un cycle represente UNE opération interne dans le processeur. Toute instrucrion (Operation programmable par l'utilisateur) necessite un certains nombre de ces operations.

```
10    REM *** SCREEN TO BUFFER / BUFFER TO SCREEN **********
20    REM *** GESCHREVEN DOOR DE BONT C.(NAAR HET PROGRAM **
30    REM *** VAN W.HERMANS UIT NEWSLETTER 16,PAGINA 168) **
40    REM ************************************************
50    CLEAR 4100:POKE #29B,0:POKE #29C,#10
60    FOR X=#D00 TO #D3F:CURSOR 20,3:PRINT #D3F-X;" ";
70    READ A:POKE X,A:NEXT
80    MODE 0:PRINT CHR$(12);:COLORG 8 0 3 14
90    PRINT "*************************************************************"
100   PRINT "*              SCREEN  BUFFER : BUFFER  SCREEN              *"
110   PRINT "*               ================================            *"
120   PRINT "*  DIT PROGRAMMA LAADT VAN UIT DATA EEN MLP IN RAM,WAAR-     *"
130   PRINT "*  DOOR HET MOGELIJK WORDT TEKENINGEN VANAF HET SCHERM       *"
140   PRINT "*  NAAR EEN BUFFER TE VERPLAATSEN EN VICE VERSA.ZOWEL IN     *"
150   PRINT "*  DE LAGE ALS IN DE MEDIUM RESOLUTIES IS DEZE TRANSACTIE    *"
160   PRINT "*  MOGELIJK.HELAAS IS BIJ HET GEBRUIK VAN MODE 5 OF 6 DE     *"
170   PRINT "*  GEHEUGENCAPACITEIT NAUWELIJKS TOEREIKEND OM ZOWEL EEN     *"
180   PRINT "*  PROGRAMMA ALS EEN BUFFER TE BEVATTEN.DAAROM BLIJFT DIT    *"
190   PRINT "*  PROGRAMMA BEPERKT TOT HET GEBRUIK IN MODE 1-2-3-4.        *"
200   PRINT "*     (BIJ HET GEBRUIK VAN SPLIT-SCREEN MODE ZAL OOK DE       *"
210   PRINT "*      ONDERSTAANDE TEKST IN HET BUFFER OPGENOMEN WORDEN)     *"
220   PRINT "*                                                            *"
230   PRINT "*         WELKE GRAFISCHE MODE WILT U (1 2 3 4)              *"
240   PRINT "*************************************************************"
250   G=GETC:IF G<49 OR G>52 THEN WAIT TIME 3:GOTO 250
260   PRINT CHR$(12);:IF G=49 THEN MODE 1A:GOTO 300
270   IF G=50 THEN MODE 2A:GOTO 300
280   IF G=51 THEN MODE 3A:GOTO 300
290   MODE 4A
300   PRINT TAB(10);"- RANDOM TEKENING IN MODE ";CHR$(G);"A -"
310   FOR X=50 TO 1 STEP -1:CURSOR 20,2:PRINT X;" ";
320   A=INT((XMAX-30.0)*RND(1.0)):B=INT((YMAX-30.0)*RND(1.0))
330   C=INT(30.0*RND(1.0))+A:D=INT(30.0*RND(1.0))+B
340   IF G MOD 2.0=1.0 THEN E=INT(15.0*RND(1.0)):GOTO 360
350   E=INT(4.0*RND(1.0))+20.0
360   FILL A,B C,D E:NEXT:PRINT
370   CURSOR 10,0:PRINT "DRUK OP DE SPATIEBALK ";
380   CALLM #D6DA:PRINT CHR$(12);
400   PRINT TAB(10);"---*** SCREEN TO BUFFER ***---"
410   WAIT TIME 40:PRINT CHR$(12);:CALLM #D00
420   CURSOR 10,0:PRINT "DRUK OP DE SPATIEBALK ";
430   CALLM #D6DA:PRINT CHR$(12);
500   PRINT TAB(10);"- IK WIS NU DE TEKENING UIT! -"
510   FILL 0,0 XMAX,YMAX 8
520   CURSOR 10,0:PRINT "DRUK OP DE SPATIEBALK ";
530   CALLM #D6DA:PRINT CHR$(12);
600   CALLM #D1C
610   PRINT TAB(10);"---*** BUFFER TO SCREEN ***---"
620   PRINT TAB(7);"EN DAAR IS DE TEKENING WEER TERUG !!"
700   CURSOR 10,0:PRINT "NOGEENS PROBEREN (J/N) ?";
710   G=GETC:G=GETC:G=GETC
720   G=GETC:IF G=0 THEN WAIT TIME 3:GOTO 720
730   IF G=74 OR G=106 THEN PRINT CHR$(12);:GOTO 80
740   IF G=78 OR G=110 THEN PRINT CHR$(12);:GOTO 760
750   GOTO 720
760   CURSOR 20,2:PRINT "E I N D E":GOTO 990
800   REM *** MODE -CORRECTIE
810   MK=PEEK(#9D)
820   IF MK<4 THEN AM=2143:GOTO 860:REM ---------- MODE 1,2
830   IF MK>3 AND MK<8 THEN AM=6563:GOTO 860:REM - MODE 3,4
840   IF MK>7 AND MK<11 THEN PRINT "MODE 5 EN 6:ONMOGELIJK"
850   PRINT "FOUTIEVE MODE ":END
860   BS=#BFEF-(AM*2)
865   POKE #D05,AM MOD 256:POKE #D06,AM SHR 8
```

```
870   POKE #D0B,BS MOD 256:POKE #D0C,BS SHR 8
880   POKE #D21,AM MOD 256:POKE #D22,AM SHR 8
890   POKE #D24,BS MOD 256:POKE #D25,BS SHR 8:RETURN
900   REM *** CALLM #D00 : SCREEN TO BUFFER
910   REM *** CALLM #D1C : BUFFER TO SCREEN
920   DATA #E5,#C5,#D5,#F5,#11,#2E,#17,#21
930   DATA #EF,#BF,#01,#00,#90,#7E,#02,#2B
940   DATA #0B,#1B,#7A,#B3,#C2,#0D,#0D,#F1
950   DATA #D1,#C1,#E1,#C9,#E5,#C5,#D5,#F5
960   DATA #11,#2E,#17,#21,#00,#90,#01,#EF
970   DATA #BF,#7E,#02,#2B,#0B,#1B,#7A,#B3
980   DATA #C2,#29,#0D,#F1,#D1,#C1,#E1,#C9
990   DATA #00,#00,#00,#00,#00,#00,#00,#00
```

ON ERROR GOTO demo

```
100   REM *** ON ERROR GOTO DEMO **************************
110   REM *** GESCHREVEN DOOR : DE BONT CORNEEL ************
120   REM *** GEBASEERD OP HET PROGRAMMA VAN N.P. LOOIJE **
130   REM *** ( NEWSLETTER 16 : PAGINA 172-173 ) ***********
140   REM *** SUBROUTINE 300-330 SCHAKELT DE 'ON ERROR' ***
150   REM *** AAN OF UIT NAARGELANG HUIDIGE STATUS ********
160   REM *** INLEZING MLP-ROUTINE VANUIT DATA
170   CLEAR 4000:POKE #29B,#60:POKE #29C,#3:PRINT CHR$(12);
180   FOR X=#300 TO #35F:CURSOR 20,20:PRINT #35F-X;" ";
190   READ PQ:POKE X,PQ:NEXT
200   REM *** DEMO PROGRAM
210   MODE 0:PRINT CHR$(12);:LNR=230:LIST 200-290
220   GOSUB 300:REM --------------SWITCH 'ON ERROR GOTO' ON
230   A=A+1:IF A<10 THEN PRINT " ";
240   PRINT A;"e MAAL ON ERROR"
250   IF A>10.0 THEN 270:REM -----------------10 x ON ERROR
260   DOT A,500 500:GOTO 230:REM -----NUMBERS OUT OF RANGE
270   GOSUB 300:REM -------------SWITCH 'ON ERROR GOTO' OFF
280   GOTO 260:REM ----NOW AN ERROR MESSAGE WILL BE PRINTED
290   END
300   REM *** 'ON ERROR GOTO' - ON EN OFF (LNR%=LIJNNUMMER)
310   IF PEEK(#6C)=#FD THEN POKE #6C,0:POKE #6D,#3:GOTO 330
320   IF PEEK(#6C)=0 THEN POKE #6C,#FD:POKE #6D,#C6
330   POKE 6,LNR IAND #FF:POKE 7,LNR SHR 8:RETURN
400   REM *** MLP 'ON ERROR GOTO' 300-35F IN DATA
410   DATA #F5,#2A,#06,#00,#7C,#B5,#CA,#21,#03,#21,#0A,#00
420   DATA #39,#7E,#FE,#53,#C2,#21,#03,#23,#7E,#FE,#DA,#C2
430   DATA #21,#03,#23,#7E,#FE,#40,#CA,#25,#03,#F1,#C3,#FD
440   DATA #C6,#2A,#06,#00,#CD,#F6,#CA,#D2,#4D,#03,#44,#4D
450   DATA #21,#00,#01,#3E,#15,#36,#00,#23,#3D,#C2,#35,#03
460   DATA #F3,#32,#31,#01,#32,#40,#00,#32,#06,#FD,#FB,#31
470   DATA #00,#F9,#C3,#92,#C8,#21,#00,#00,#22,#06,#00,#3E
480   DATA #04,#C3,#F5,#D9,#00,#00,#00,#00,#00,#00,#00,#00
```

DAI VIDEO SCREEN RAM

(from DAInamic 12, page 248)

DEAR DAI FRIENDS,

Here are some more programs for you. Two of them complement, with demonstrations, the article in DAInamic 10 on 240*528 resolution, ie, MODE 7 and MODE 8. There is also an explanation of the pointers. After reading the aforementioned article I was keen to try it out. The programme was typed in but alas, in spite of the screen being initialised nothing else would happen. I wanted at once to find out why and set about investigating what the DAI manual had to say on screen pictures and control words. The investigations produced the following:-
SCREEN STRUCTURE MODE 2A after COLORT 0 1 2 3, COLORG 0 1 2 3
N.B. Everything is referenced from the TOP of the screen.

```
#BFFF
* SCREEN ------------------------------------------------------------------------------------|
| 36 80--00 00   UNITCOLORMODE / COLORG 0      ==========                                    |
| 36 91--00 00       '     '   / COLORG 1      = HEADER =                                    |
| 36 A2--00 00       '     '   / COLORG 2      ==========                                    |
| 36 B3--00 00       '     '   / COLORG 3                                                    |
* SCTOP -----------------------------------------------------------------------------------|
| 03 40--00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| total 53 [GRL-GAL] lines of GRAPHICS with #18 [GXB] bytes/line                           |
* GRE -------------------------------------------------------------------------------------|
| 30 80--00 00   UNITCOLORMODE / COLORT 0                                                  |
| 30 91--00 00       '     '   / COLORT 1      ===============                           |
| 30 A2--00 00       '     '   / COLORT 2      = INTERMEDIATE =                          |
| 30 B3--00 00       '     '   / COLORT 3      ===============                           |
* CHS -------------------------------------------------------------------------------------|
| 7A 40--20 00 (66*[20 00])                                                                |
| total 4 [TXL] lines of TEXT with #86 [TXB] bytes/line                                    |
* CHE -------------------------------------------------------------------------------------|
| 3F 80--00 00   UNITCOLORMODE / COLORT 0                                                  |
| 3F 91--00 00       '     '   / COLORT 1      ===========                               |
| 3F A2--00 00       '     '   / COLORT 2      = TRAILER =                               |
| 3F B3--00 00       '     '   / COLORT 3      ===========                               |
* SCE/GTS ---------------------------------------------------------------------------------|
| 03 40--00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00 00 00 |
| total 12 [GAL] lines scrolled (from the upper part of screen)                      *FFB  |
* GTE -------------------------------------------------------------------------------------|
#B8B7
```

#BFFF-#BFF0 this line contains the HEADER. For mode 2A it is:
#BFFF-#BFFD 36 80 00 00 where ....
#36 is the MODE BYTE 'HIGH address'
bit 7,6 [00] display-mode control : 4 colour graphics
bit 5,4 [11] resolution control : 528 dots per line
bit 3,2,1,0 [0110] line repeat count : 6 repartitions
#80 is the COLOUR TYPE BYTE 'LOW address'

bit 7 [1] makes possible a colour change
bit 6 [0] puts this line in 'unit colour mode' so that the two data bytes can be repeated as many times as required by the resolution control bits.
bit 5,4 [00] defines colour register 0 from the COLORG in bits 3-0. Each time the HIGH and LOW data bits are 0 and 0 the colour from bits 3-0 are used. This continues until colour 0 is changed in another colour type byte by means of bits 7, 5 and 4.
bit 3,2,1,0 [0000] selects colour 0 from the 16 colours.
#00 is the HIGH BYTE 'HIGH address'
bit 7-0 [00000000] in conjunction with the low byte selects colour 0 from COLORG for all dots
#00 is the LOW BYTE 'LOW address'
bit 7-0 [00000000] see HIGH address. These two bytes will be repeated 65 times by the 'unit colour mode'

The subsequent 12 bytes will be dealt with in a similar way in the 'unit colour mode'. This makes a total of 28 empty lines. Bits 3-0 of #BFFA, #BFF6 and #BFF2 contain respectively colours 1, 2 and 3 from the COLORG registers, bits 5 and 4 stipulating which ones. These 28 empty lines appear in this way in all 4 colour modes. In 16 colour modes something rather similar happens, only here the bits with the 4 colour mode data are replaced by the 16 colour data. This means that among other things, colour changes do not occur at once after COLORG. The intermediate and trailer (line repeat count 0 and #F respectively) operate the same way but with data from COLORT. COLORG applies to the HEADER. COLORT applies to the INTERMEDIATE and TRAILER. The line mode bytes for graphics and text contain data for display mode control, resolution control, line repeat count and the length of the lines. The colour byte remains neutral, bit 7=0 (no colour change) and bit 6=1 (no unit colour mode). Information on the above is detailed in the manual in the section entitled PROGRAMMABLE GRAPHICS GENERATOR.

MAXIMUM RESOLUTION - 512*244 MODE 7 & 8
We can read in the manual that a max of 32K byte of RAM, #4000 - #BFFF, is available for the screen. Calculation shows that with 528 dots per line there are #86 (134) bytes per line and so 32768/134 = 244 lines max are possible; thus more than 240. The control words must be:-

MODE BYTE
bit 7,6 [00] for 4 colour graphics
[10] for 16 colour graphics
bit 5,4 [11] for 528 dots per line
bit 3,2,1,0 [0000] for 1 line, so no repeats
COLOUR TYPE BYTE
bit 7 [0] bits 5 - 0 ignored
bit 6 [1] no unit colour mode
bit 5-0 [XXXXXX] no effect
Thus we find for MODE 7 #B0 #40 and for MODE 8 #30 #40. The start-up for MODE 7 is via MODE 1, 3, or 5 and MODE 8 via MODE 2, 4, or 6.

From #BFEF with a step of #86 these bytes will be filled in 244 times. Afterwards must come a trailer (line repeat count #F) identical to the header and which serves to fill the screen with empty lines. If we do not do this we get a muddle lower down the screen. If this happens the screen is initialised but we can do nothing else with it. The pointers for the graphical functions must then be set with CURRENT STATE OF SCREEN, CHARACTER MODE and START OF SCREEN. After a short period of experimentally setting the various pointers it seems that DOT, DRAW, FILL and SCRN work excellently. Only error messages like COLOR NOT AVAILABLE and OFF SCREEN were given; XMAX apparently is hardware restricted to 511 and YMAX to 243.

The new pointers are:-

80/81	SCREEN	#BFFF	90/91	GTE	XXXX
82/83	SCTOP	#BFEF	92/93	GAS/GTS	XXXX
84/85	FFB	#4027	94/95	GRC	#0200
86/87	GRR	XXXX	96	GRL	# F4
88/89	GRE	#4037	97	GAL	# XX
8A/8B	CHS	#4027	98	GXB	# 86
8C/8D	GAE/CHE	#4037			
8E/8F	SCE	#4027			

HOW TO CALCULATE THE POINTERS

Without having had to study the ROM I have managed to discover the following values for the pointers. With some help from the memory map V4.4 I have tested the screen functions; there may be defects but at least it works in all modes. You can see the locations of the pointers in the sketches. The important ones for designing in a single mode are given below with a brief description and the method of calculating them. They are:-

A. Text (MODE 0)
B. Graphics (MODE 1-8)
C. Split mode, text below (MODE 1A-8A)
D. Split mode, text above (MODE 1B-8B)

Some useful values:-

HDRL = header length (normally #10)
INTL = length of intermediate (normally #10)
TRAL = trailer length (normally #10)
TXL = number of text lines
TXB = number of bytes per text line (#86) may also be #5A, #2E or #18 but the screen driver can only work with #86

#72/#73 CURSOR.

Position of the cursor on the screen. The setting for the cursor in a new text or split-mode must be 8 bytes after the line mode byte. In a graphics mode the cursor position will be set to 0000.

A. + C. + D. >> CHS - 8
B. >> 0000

#78/#79 LNSTR - LiNe STaRt.

Line mode byte for the cursor line.

A. + C. + D. >> CHS
B. >> don't care

#7A LNEND - LiNe END.

Low byte of the end of the cursor line. Used to check for end of line.

A. + C. + D. >> (CHS+#80) IAND #FF
B. >> don't care

#80/#81 SCREEN.

Top of the screen. Used for the cursor position and memory check during mode changing.

A. + B. + C. + D. >> #BFFF

#82/#83 SCTOP - SCreen TOP.
Pointer to the first byte (line mode byte) of the graphics, immediately after the intermediate or header. The name SCTOP is an unfortunate choice in this instance; GRS (GRaphics Start) would have been better because the pointer can be to anywhere in the screen memory. Used for COLORG and drawing commands.

A. >> don't care
B. + C. >> #BFFF-HDRL
D. >> #BFFF-HDRL-TXL*TXB-INTL

#84/#85 FFB - First Free Byte.
Pointer to the first byte after the graphics screen.

A. >> #BFFF-HDRL-TXL*TXB-TRAL
B. >> #BFFF-HDRL-GRL*GXB-TRAL
C. + D. >> #BFFF-HDRL-TXL*TXB-INTL-(GRL-GAL)*GXB-TRAL

#86/#87 GRR - GRaphics Rolled.
Pointer used in split modes. It indicates from where the screen is scrolled up to make room for the text. The portion above GRR to SCTOP is shifted to under the visible screen. Since scrolling is not possible in single (non-split) modes the pointer has little value.

A. >> don't care
B. + C. + D. >> #BFFF-HDRL-GAL*GXB or don't care

#88/#89 GRE - GRaphics End.
Points 1 byte after the end of the graphics section in use in the screen memory. It is used for the drawing functions.

A. >> don't care
B. >> #BFFF-HDRL-GRL*GXB
C. >> #BFFF-(GRL-GAL)*GXB
D. >> #BFFF-HDRL-TXL*TXB-INTL-(GRL-GAL)*GXB

#8A/#8B CHS - CHaracter Start.
Points to the first byte (line mode byte) of the text. Its use, amoung other things, is in PRINT and COLORT commands.

A. + D. >> #BFFF-HDRL
B. >> #BFFF-HDRL-GRL*GXB or don't care
C. >> #BFFF-HDRL-(GRL-GAL)*GXB-INTL

#8C/#8D GAE/CHE - Graphics Archive End/CHaracter End.
In either text or split-mode the pointer is to 1 byte after the text portion. In an unsplit mode it points 1 byte after the trailer. It is used in PRINT commands.

A. >> #BFFF-HDRL-TXL*TXB
B. >> #BFFF-HDRL-GRL*GXB-INTL
C. >> #BFFF-HDRL-(GRL-GAL)*GXB--INTL-TXL*TXB
D. >> #BFFF-HDRL-TXL*TXB

#8E/#8F SCE - SCreen End.
Points 1 byte after the end of the screen, thus immediately after the trailer. Uses : COLORT and COLORG, amoung others.

A. >> #BFFF-HDRL-TXL*TXB-TRAL
B. >> #BFFF-HDRL-GRL*GXB-TRAL
C. + D. >> #BFFF-HDRL-GRL*GXB-INTL-TXB*TXB-TRAL

#90/#91 GTE - Graphics Temporary save area End.
In a graphics mode the pointers indicate the end of the area where the scrolled section of graphics is located.

A. >> don't care
B. + C. + D. >> #BFFF-HDRL-(GRL+GAL)*GXB or don't care

#92/#93 GAS/GTS - Graphics Archive Start/Graphics Temporary save Start
In a graphics mode the pointer indicates the start of the area where the scrolled section of graphics is located.

A. >> don't care
B. >> #BFFF-HDRL-TXL*TXB-GRL*GXB or don't care
C. + D. >> #BFFF-HDRL-GRL*GXB

94/#95 GRC - GRaphics Columns.
This pointer gives the number of dots in the X direction; the value must be a minimum of 1 and a maximum of as many as stipulated in the line mode byte. For example with very high resolution at #86 bytes per line the maximum is 8*(#86/2-3)=#200=512 and for low resolution at #18 bytes per line it is 8*(#86/2)=#48=72. Used for drawing commands

A. >> don't care
B. + C. + D. >> GRC or XMAX+1

#96 GRL - GRaphics Lines.
This pointer gives the number of dots and hence the number of lines in the Y direction. Its value must be a minimum of 16 and a maximum of 256 lines. Used with drawing commands.

A. >> don't care
B. + C. + D. >> GRL = YMAX+1

#97 GAL - Graphics Archive Lines.
This pointer gives the number of lines scrolled or to be scrolled. You can use the value 0 when planning your own modes.

A. + B. + C. + D. >> don't care

#98 GXB - Graphics X Bytes.
The number of bytes per line in the mode used, graphics or split. Used with drawing commands.

A. >> don't care
B. + C. + D. >> as stipulated in the line mode bytes graphics, #86, #5A, #2E, #18 for MODE 7/8, 5/6, 3/4, 2/1 respectively

#99/#9A GREQ - GRaphics End Old.
The previous end of the graphics. Need not be filled in when planning a mode.

A. + B. + C. + D. >> don't care

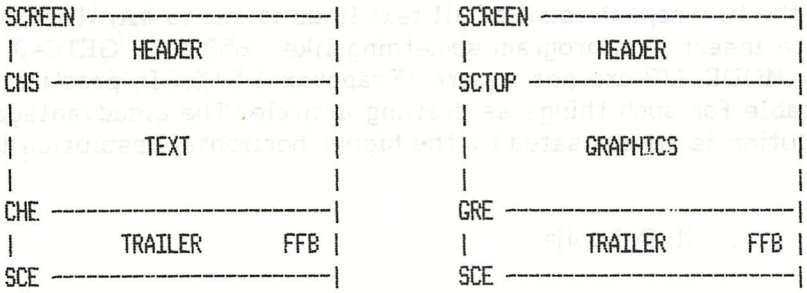
#9B/#9C CHSO - CHaracter Start Old.
The previous end of characters. Need not be filled when planning a mode.

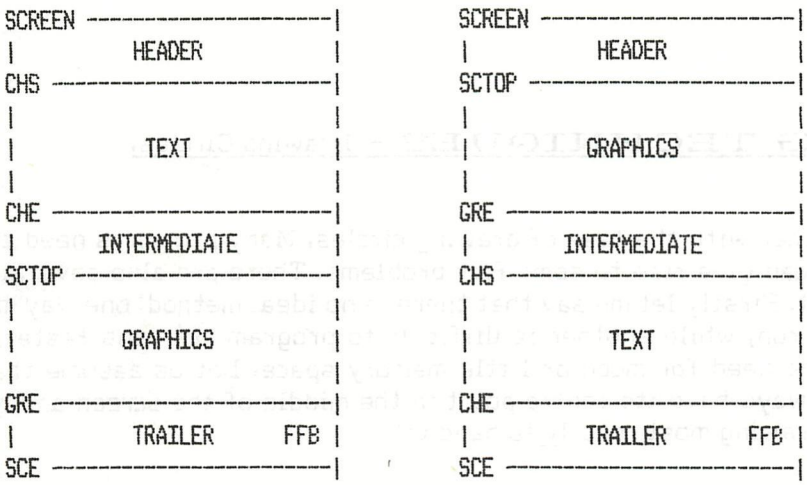
A. + B. + C. + D. >> don't care

#2A5/#2A6 SCBOT - SCReen BOTtom.
This is where the first byte of screen memory is. Amoung other uses it is for testing for OUT OF SPACE FOR MODE.

A. >> #BFFF-HDRL-TXL*TXB-TRAL+1
B. >> #BFFF-HDRL-GRL*GXB-TRAL+1
C. >> #BFFF-HDRL-GRL*GXB-INTL-TXL*TXB-TRAL+1

Maps of the various modes with the positions of the pointers.





Now follow the values for the HEADER, INTERMEDIATE & TRAILER after a COLORT w x y z and/or COLORG w x y z. The HEADER and the INTERMEDIATE contain the colour of the section above which they stand (graphics COLORG/ text COLORT). The TRAILER contains the colour for the area below it.

```
HEADER        4 COLOURS
36 8w 00 00 36 9x 00 00 36 Ay 00 00 36 Bz 00 00
INTERMEDIATE  4 COLOURS
30 8w 00 00 30 9x 00 00 30 Ay 00 00 30 Bz 00 00
TRAILER       4 COLOURS
3F 8w 00 00 3F 9x 00 00 3F Ay 00 00 3F Bz 00 00
HEADER       16 COLOURS
B6 8w w0 FF B6 9x w0 FF B6 Ay w0 FF B6 Bz w0 FF
INTERMEDIATE 16 COLOURS
B0 8w w0 FF B0 9x w0 FF B0 Ay w0 FF B0 Bz w0 FF
TRAILER      16 COLOURS
BF 8w w0 FF BF 9x w0 FF BF Ay w0 FF BF Bz w0 FF
```

Remarks.
In a character mode the text lines will be restored to normal spacing by LIST, ?CHR$(12) and UT. In 16 colour text these commands will also reset the screen to 4 colours and when scrolling the picture the bottom-most line will be printed in 4 colour mode. The lowest line of text will always be forced to normal line spacing. From experiments it would appear that MODES 8A & 7A

will not be affected by a BREAK because the screen driver does not recognise these modes. This must be accomplished by some other artifaçe, namely by taking 240 line graphics and 4 line text with a line repeat count of zero. With a BREAK the screen driver then forces 1 line to a repeat count of #A. On a ?CHR$ (12) the line repeat count of all text lines is set to #A. With full 244 line graphics it is advisable to insert in a program something like '65335 IF GETC=0 THEN 65335 : MODE 0' The dots in MODE 7/8 are not square (Y=approx 1.5*X). In practice these proportions appear to be suitable for such things as drawing a circle. The disadvantage of a somewhat lower vertical resolution is compensated by the higher horizontal resolution in, for example, graphics.

With friendly greetings, N. P. Looije.

PROGRAMMING TECHNIQUES - Drawing Circles.

(from DAInamic 12, page 268)

In this article I will deal further with the task of drawing circles. Many programs need circles or ellipses and drawing them can give rise to some fair problems. There are also several ways of achieving the desired result. Firstly let me say that there is no ideal method; one way may be quick to program but slow to run, while another is difficult to program but runs faster. Still further variations exist on the need for much or little memory space. Let us assume that the circle we wish to draw will always have its centre point in the middle of the screen and have a radius of 100. The method of drawing most readily to hand is:

```
10  MODE 6
20  FOR I=0.0 TO 2.0*PI STEP PI/180.0
30  DOT XMAX/2+100*SIN(I),YMAX/2+100*COS(I) 22:NEXT
```

As we know that there are 180 degrees in PI radians we take STEPs of PI/180, or 1 degree. When the RUN command is given the resultant circle looks as if it has gaps in it. We can overcome this by making the steps smaller but at the cost of speed. If the radius is 100 then the circumference of the circle is 200*PI, which is about 628. Therefore the steps must be such that each of the dots appears only once. The following uses a step size of 2*PI/200*PI which is the same as PI/315 or 0.01, or put even more simply, 1/radius. Eventually, after a bit of rounding off, we settled for 0.9/radius. Working with a variable is faster than with a constant. Therefore:

```
10  MODE 6:FOR I=0.0 TO 2.0*PI STEP PI/315.0
20  R=100.0:DOT XMAX/2+R*SIN(I),YMAX/2+R*COS(I) 22:NEXT
```

... works faster, but each time R is made equal to 100 I lose the trifling gain I had made. More to the point, I have made the DAI call XMAX & YMAX about 600 times and each time have made it divide them by 2. The next program improves that:

```
10  MODE 6:MX=XMAX/2.0:MY=YMAX/2.0:R=100.0
20  FOR I=0 TO 2.0*PI STEP 1.0/R:DOT MX+R*SIN(I),MY+R*COS(I) 22:NEXT
```

We notice that XMAX/2.0 is shown instead of XMAX/2 as hitherto. This is because the DOT

instruction expects integers and therefore constants like 2 were not turned into 2.0. While running the XMAX/2 is changed to floating point for the calculation of R*SIN(I) and then changed back again to integer to satisfy the DOT instruction. So by putting everything in integers we can speed up the program. The first attempt at this misfired because SIN(I) and COS(I) always became zero and consequently the loop variable I also remained zero. After more thought we came up with the following program:

```
IMP INT
10  MODE 6:MX=XMAX/2:MY=YMAX/2:R!=100.0
20  FOR I!=0.0 TO PI+PI STEP 1.0/R!:X=R!*SIN(I!):Y=R!*COS(I!)
30  DOT MX+X,MY+Y 22:NEXT
```

... and this works faster. Have you spotted why? Simply that PI+PI is faster than 2.0*PI; but that is not the whole story, the combination of line 20 and line 30 also contributes. However our programme is still slow. We let sines and cosines be calculated 628 times. We can change that when we realise that the left and right parts are mirror images, and arrive at:

```
10  MODE 6:MX=XMAX/2:MY=YMAX/2:R!=100.0
20  FOR I!=0.0 TO PI STEP 1E-2:X=R!*SIN(I!):Y=R!*COS(I!)
30  DOT MX+X,MY+Y 22:DOT MX-X,MY+Y 22:NEXT
```

This seems good, it works smoothly and it is obvious that we can achieve further improvements in speed by exploiting other symmetries. In line 20's FOR statement we will only go to PI/2 instead of to PI, and line 30 becomes :- DOT MX+X,MY+Y 22:DOT MX+X,MY-Y 22:DOT MX-X,MY+Y 22:DOT MX-X,MY-Y 22:NEXT. Becoming enthusiastic now we bring into use the symmetrical diagonals too:

```
10  MODE 6:MX=XMAX/2:MY=YMAX/2:R!=100.0
20  FOR I!=0.0 TO PI/4.0 STEP 1/R!:X=R!*SIN(I!):Y=R!*COS(I!)
30  DOT MX+Y,MY+Y 22:DOT MX+X,MY-Y 22:DOT MX-X,MY+Y 22:DOT MX-X,MY-Y 22
40  DOT MX+Y,MY+X 22:DOT MX+Y,MY-X 22:DOT MX-Y,MY+X 22:DOT MX-Y,MY-X 22:
NEXT
```

... and although the program runs reasonably quickly the programming itself does involve a fair amount of effort. Now we consider if the delays associated with the sines and cosines are really necessary. We can establish that one of the two can be overcome but that again does not lead to simple programming. Let us try another way, this time with Pythagoras' theorem. Reckoning from the centre there is a point on the circle's circumference with co-ordinates x and y such that x^2 + y^2 must be equal to the radius^2. Recognising that for each x value there must be two y values we start with:

```
10  MODE 6:XM=XMAX/2:YM=YMAX/2:R=100
20  FOR X=-R TO R:Y=SQR(R*R-X*X)
30  DOT MX+X,MY+Y 22:DOT MX+X,MY-Y 22:NEXT
```

This apparently does not work correctly. On the left and right many points are missed because the x values are larger versions of the y values. Could we make use of the upper/lower symmetry as well as the left/right? No, because although it would be quite feasible we would still find the left and right points failing. We saw that the horizontal was drawn correctly, thus we could use the above method to obtain that half and achieve the other half by exchanging the x and y co-ordinates, as follows: Line 10 remains the same.

```
20  FOR I=-R/2 TO R/2:J=SQR(R*R-I*I)
30  DOT MX+I,MY+J 22:DOT MX+I,MY-J 22
40  DOT MX+J,MY+I 22:DOT MX-J,MY+I 22:NEXT
```

But this also does not work satisfactorily. Drawing half a circle does not mean from -R/2 to R/2 but from -45 to +45 degrees. So we must put the boundaries at +/-70 and not 50. The figure of 70 can be obtained (it will be needed for other circles) by dividing the radius by root 2. Again using the symmetries we now arrive at:

```
10  MODE 6: MX=XMAX/2:MY=YMAX/2:R=100
20  FOR I=0 TO R/SQR(2):J=SQR(R*R-I*I)
30  DOT MX+I,MY+J 22:DOT MX+I,MY-J 22:DOT MX-I,MY+J 22:DOT MX-I,MY-J 22
40  DOT MX+J,MY+I 22:DOT MX+J,MY-I 22:DOT MX-J,MY+I 22:DOT MX-J,MY-I 22:NEXT
```

This works fast, especially on machines without the maths chip. It is clearly an improvement but again with a ponderous program. For small circles the previous methods are just as good. Did you notice that I used R*R and not R^2? It is faster and also works for negative numbers.

If we have to draw numerous circles it would be worthwhile putting our sine and cosine values in an array so that they can be used time and time again. The advantage is that the value is available quickly for rapid multiplication with the radius and conversion into integer. When the radii are different, one smaller than 100 will give us a small time advantage and with one less than half the size we have been using we can miss some array elements and save more time. There are snags too - we have used up more memory (more than half of it is filled with zeroes), and we have to wait until the table is ready. However let us try it:

```
10  MODE 6A:CLEAR 8000:DIM X(3,159),Y(3,159)
20  MX=XMAX/2:MY=YMAX/2 R!=100.0
30  FOR A!=0.0 TO PI+PI STEP PI/316.0
40  X=R!*SIN(A!):X=MX+X:Y=R!*COS(A!):Y=Y+MY
50  X(P,Q)=X:Y(P,Q)=Y:Q=Q+1
60  IF Q=159 THEN Q=0:P=P+1:PRINT P
70  NEXT:MODE 6
80  FOR P=0 TO 3:FOR Q=0 TO 158:DOT X(P,Q),Y(P,Q) 22:NEXT:NEXT
```

Note the PRINT P, it is unnecessary and serves only as an indicator. We are using a 2-dimensional array for X and for Y. This is certainly much quicker than a 3-dimensional array. Always use integers for the indices of the arrays; look-up times will be 4 times as long with floating point. Improvements can still be made such as cutting to a quarter the calculations by mirroring. The program below does this and also works out the +/-. Study it and check the working:

```
10  MODE 6A:CLEAR 8000:DIM X(159),Y(159)
20  MX=XMAX/2:MY=YMAX/2:R!=100.0
30  FOR A!=0.0 TO PI/2.0 STEP PI/316.0
40  X(Q)=R!*SIN(A!):Y(Q)=R!*COS(A!):Q=Q+1:NEXT
50  MODE 6
60  FOR Q=0 TO 159:FOR M=-1 TO 1 STEP 2:FOR N=-1 TO 1 STEP 2
70  DOT MX+M*X(Q),MY+N*Y(Q) 22:NEXT:NEXT:NEXT
```

The next step is to build up the array by Pythagoras but that can be left to the reader. We can now think about the use of sine and cosine, or rather an approximation of them. For this refer to the article by T.Berkx in issue 10 but take note that this method is only faster for machines

without the maths chip. For that reason I have not worked it out. I also have not worked on the following idea but it should certainly be useable for small circles. Put the required values in a number of DATAs, or read them in from an array. The latter will only interest owners of MDCRs and floppy discs. Now I will draw the circle really fast, no longer doing it point by point but with little lines. In joining up the points we can make use of one of the previously discussed methods.

```
10  MODE 6:MX=XMAX/2:MY=YMAX/2:R!=100.0:XO=MX:YO=MY+100
20  FOR I!=0.0 TO PI+PI STEP PI/15.0:X=R!*SIN(I!):Y=R!*COS(I!)
30  DRAW XO,YO MX+X,MY+Y 22:XO=MX+X:YO=MY+Y:NEXT
```

If you find that the circle is too angular the STEP needs to be altered. It may be necessary to go rather more than right round otherwise the final dash may not be drawn in. This method can be combined with the one which reads from an array and thus we get a fast running but awkward to write program.

```
10  MODE 6:CLEAR 2000:MX=XMAX/2:MY=YMAX/2:R!=100.0:DIM X(30),Y(30)
20  FOR A!=0.0 TO 2.06 STEP PI/15.0
30  X=R!*SIN(A!):X(P)=MX+X:Y=R!*COS(A!):Y(P)=MY+Y:P=P+1:NEXT
40  FOR I=0 TO 29:DRAW X(I),Y(I) X(I+1),Y(I+1) 22:NEXT
```

Even in this program some minor items can be trimmed. The 2 variables X and Y can be replaced by one. In place of the new variable I we can use P - it worsens the readability and does nothing for the speed but we gain a little in the memory (8 bytes). Combining lines 20 and 30 will also change the speed. We have only been discussing circles having a radius of 100 but small circles can sometimes be handled better with one of our earlier methods. I must also mention that if we want the circle sectioned we should avoid drawing a bundle of radii from the centre to the rim because through rounding off, some of the points will be lost, at least with larger circles. An obvious remedy is to reduce the size of the steps or even better, to draw diameters instead of radii. Such lines should be drawn horizontally, not vertically because of the DAI's screen handling system. I will leave the reader to have a go at programming this.

Frank H. Druijff

CONVERSION APPLE - ATARI - DAI

(from DAInamic 12, p289)

The conversion of Apple and Atari programs for DAI gives rise to various problems, among which is that counting in their graphics modes starts from the top left corner of the screen. Thus conversion, especially for long programs, involves extensive calculations with consequently more chance of error. An example :-

```
    Apple                  1st solution          2nd solution          3rd solution
                                                                       5  POKE #2FF,179: POKE #6C,0: POKE #6D,3
10  HGR                    MODE 6                MODE 6                10 MODE 6
20  Y=0:COLOR=15           Y=179:C=15            Y=0:C=15              20 Y=0:C=15
30  PLOT 100,Y             DOT 100,Y C           DOT 100,179-Y C       30 DOT 100,Y C
40  Y=Y+1                  Y=Y-1                 Y=Y+1                 40 Y=Y+1
50  IF Y>179 THEN END      IF Y<0 THEN END       IF Y>179 THEN END     50 IF Y>179 THEN END
60  GOTO 30                GOTO 30               GOTO 30               60 GOTO 30
```

The advantage of the third solution is that after the initial line 5, the Apple or Atari program can be input almost literally. The working is as follows. A machine language program does for solution 3 what solution 2 achieves in BASIC, namely the subtraction of each Y value from a previously specified value (179 in this example). This is possible because each BASIC command associated with the screen RAM makes use of RST + byte combination. RST 5 stands in for a CALL #28, and at #28 is the interrupt routine for vector 5 (see also DAInamic 11, pages 180-181). Interrupt routine 5 accomplishes a jump in accordance with the contents of locations #6C and #6D. If locations #6C and #6D contain #300 then on every screen command there will be a jump to #300, with the required parameters being in the registers. (see also DAInamic 5, page 120). From #300 onwards a part of the ROM is copied so that when a screen function has been executed there will be a jump back to the ROM. #313 looks for a DOT, DRAW, FILL or SCRN function. If there is none then it jumps back to the ROM, but if there is one then Y will be deducted from the contents of #2FF. Then follows the jump back to ROM and BASIC is none the wiser.

```
#300  E1       POP H
#301  F3       DI
#302  224300   SHLD #43     ;SAVE HL
#305  F5       PUSH PSW
#306  3E80     MVI A,#80    ;BANK SELECT BITS
#308  E1       POP H
#309  224100   SHLD #41     ;SAVE PSW
#30C  67       MOV H,A
#30D  AF       XRA A        ;CLEAR A
#30E  E3       XTHL
#30F  86       ADD M        ;ENTRY NUMBER IN ACCU
#310  23       INX H
#311  E3       XTHL
#312  6F       MOV L,A      ;COMPLETE ENTRYPOINT ADDRESS
#313  FE1E     CPI #1E      ;SEE IF ENTRY NUMBER
#315  DAD4C6   JC #C6D4     ;IS EQUAL OR BETWEEN
#318  FE2A     CPI #2A      ;#1E AND #27
#31A  D2D4C6   JNC #C6D4
#31D  3AFF02   LDA #2FF     ;Y1:=(#2FF)-Y1
#320  90       SUB B
#321  47       MOV B,A
#322  3AFF02   LDA #2FF     ;Y2:=(#2FF)-Y2
#325  91       SUB C
#326  4F       MOV C,A
#327  C3D4C6   JMP #C6D4

#300  E1 F3 22 43 00 F5 3E 80 E1 22 41 00 67 AF E3 86
#310  23 E3 6F FE 1E DA D4 C6 FE 2A D2 D4 C6 3A FF 02
#320  90 47 3A FF 02 91 4F C3 D4 C6
```

The clever ones who have followed this up to now will have noticed that for DOT and SCRN functions only one Y value is given in the BASIC although in the machine language there are always two subtractions. How does that happen? Indeed I do not know, but it appears to work all right in BASIC. When one has input the ML program the BASIC pointers have to be adapted: (UT S29B EC-30 02-3) then go back to BASIC and give a NEW command or clear 256. The ML program can be switched in with: POKE #6C,0: POKE #6D,3 and switched out again with: POKE #6C,#FD: POKE #6D,#C6. The previously dimensioned Y value is poked into #2FF. For completeness, the YMAXs of the Apple in the different graphic modes are: GR = 39 & HGR = 179.

In conclusion, if one can fathom out the workings a lot of pleasing effects can be achieved, with minor extensions such as moving drawings, reflections (mirroring) etc. One can also write a similar program for the CURSOR statement.

With best wishes, Frans Verstegen.

TV TENNIS REMs

(from DAInamic 12, page 278)

```
2       REM See the DAInamic original for rest of program.
10      REM initialise
49      REM Drawing the bats and erasing the earlier ones
99      REM Drawing the ball and erasing the old one
125     IF BX1<5.0 THEN GOSUB 1000:REM Ball at height of red bat
126     IF BX1>XMAX-5.0 THEN GOSUB 1200:REM Ball at height of blue bat
129     REM Ball at end of court
139     REM Ball at the edge of the court
150     BX1=BX1+SNX:BY1=BY1+SNY:REM Calculation of the new position for the ball
1021    IF BEW=BAT1 THEN RETURN:REM Effective hit or not
1400    SOUND 0 0 15 0 FREQ(35,0):WAIT TIME 10:SOUND OFF :BER$="RED missed..."
1600    SOUND 0 0 15 0 FREQ(50,0):WAIT TIME 10:SOUND OFF :BER$="BLUE missed..."
2002    REM Initialise
2015    REM START PAGE
2110    CURSOR 30,3:PRINT "by Luc Maes"
2398    REM PAGE 1:Choice of the level of difficulty
2410    CURSOR 10,20:PRINT "Which level of difficulty do you want ?"
2420    CURSOR 20,14:PRINT "1: easy (large racquets)"
2430    CURSOR 20,13:PRINT "2: hard (small racquets)"
2435    CURSOR 2,6:PRINT "On court: 'EVENT' to serve":CURSOR 12,5:PRINT "'M' for the
alternative level of difficulty"
2436    CURSOR 12,4:PRINT "'CHAR DEL' to stop play"
2500    REM Reckoning the score
2570    IF GR=40 AND J$="advantage RED" THEN J$="":GOTO 2800
2580    IF GR=40 AND J$="" THEN J$="advantage BLUE"
2710    IF GB=40.0 AND J$="advantage BLUE" THEN J$="":GOTO 2800
2720    IF GB=40.0 AND J$="" THEN J$="advantage RED":GOTO 2800
2800    REM PAGE 3:court + scoreboard
2840    CURSOR 8,3:PRINT "Red's score"
2860    CURSOR 39,3:PRINT "Blue's score"
2950    IF (SB+SR) MOD 2.0=0.0 THEN BX1=7.0:SNX=3.0:CURSOR 0,0:PRINT "RED
serves.":;GOTO 2990
2960    BX1=XMAX-7.0:SNX=-3.0:CURSOR 40,0:PRINT "BLUE serves.";
3005    IF (SB+SR) MOD 2=0 THEN BY1=3.0+PDL(5)*Q:GOTO 3050
4999    REM End of the game
5020    CURSOR 10,20:PRINT "'RETURN' to play again.":CURSOR 10,17:PRINT "'CHAR DEL' to
finish."
5030    CURSOR 10,14:PRINT "'L' to load the next program"
```

A. ALGEMEENHEDEN

HOOFDSTUK I : INLEIDENDE BEGRIPPEN

1.1. Situering van microprocessoren in de digitale techniek

Met de opkomst van de microprocessoren ontstond, naast de klassieke combinatorische en sequentiële logica, een belangrijke tak die micro-elektronica wordt genoemd.
Behalve het feit dat het aantal componenten per chip en dus de gecompliceerdheid per IC sterk is toegenomen, dienen we de schakelingen te benaderen met een nieuwe filosophie.
Met de klassieke logica was alles erop gericht om, vertrekkende van het 'lastenboek' van de toepassing een schema te ontwikkelen, waarbij volgende stadia worden doorlopen :

- Opstellen van de vergelijking(en) van de gewenste toepassing.
- Vereenvoudiging van deze vergelijking(en) met behulp van klassieke methoden zoals Karnaugh, Quine en Mc Cluskey.
- Opstellen van het schema aan de hand van de vereenvoudigde vergelijking(en).

Deze denkwijze dient op het gebied van de micro-elektronica grondig herzien te worden. Inderdaad, het komt er niet meer op aan het schema samen te stellen, maar vertrekkend van een standaardschema, of beter gezegd van standaard materiaal, (een microcomputer) de gewenste toepassing te 'programmeren'.
We dienen dus te spreken van programmeerbare logica of programmeerbare systemen. Het standaard materiaal is in feite een microcomputer waarvan de microprocessor zelf meestal slechts de centrale verwerkingseenheid vormt. We kunnen dus de microprocessor definiëren als een programmeerbare logische LSI bouwsteen.
In plaats van te spreken van logische schakelingen spreken we van logische systemen ; hiermee bedoelen we dat we niet één schakeling gebruiken maar dat meerdere schakelingen samengevoegd worden tot een systeem als fundamenteel schema.

1.2. Architectuur van een microprocessorsysteem.

Het principiële schema van een microprocessorsysteem herleidt zich tot het basisschema van een computer of beter gezegd een microcomputer (fig. 1.1).

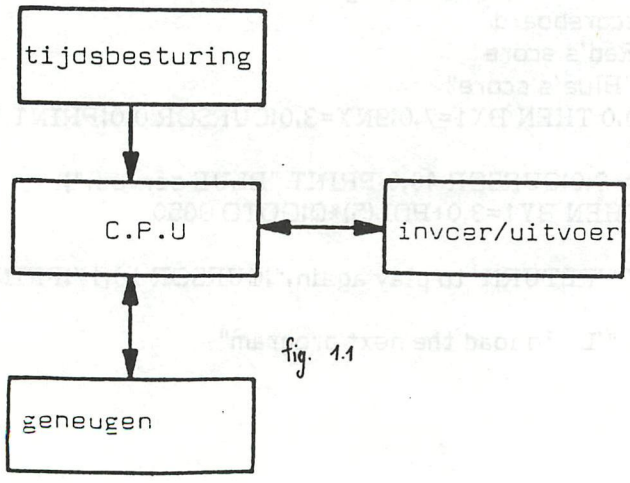


fig. 1.1

De samenstellende delen van dit schema zijn :

- de centrale verwerkingseenheid (microprocessor) (CPU : *central processing unit*).
  Deze schakeling voert alle logische en rekenkundige bewerkingen uit en is tevens verantwoordelijk voor de besturing en de controle van operaties in het ganse circuit.
  De centrale verwerkingseenheid wordt zelf gesynchroniseerd vanuit de klok of tijdsbesturing.
- De invoer- uitvoer schakelingen maken het mogelijk informatie te ontvangen of te sturen naar de randapparaten.
- Het geheugen laat toe programma's en of gegevens op te slaan voor verdere verwerking.

Opdat de centrale verwerkingseenheid de aangegeven functies naar behoren zou kunnen vervullen, moet de microprocessor geheugen of invoer/uitvoer poorten kunnen adresseren en informatie (gegevens, data) kunnen ontvangen, verwerken of doorzenden.
De verbinding van de microprocessor met de andere schakelingen gebeurt derhalve meestal met zogenaamde bussen.
Een bus is een meeraderige verbinding met een aantal draden bepaald door de soort informatie dat overgedragen wordt.
De klassiek gebruikte bussen zijn :

- Een data of gegevensbus waarmede informatie (data) van- en naar de microprocessor gebracht wordt.
  Bij een 8 bits microprocessor (die met een woordlengte van 8 bits of 1 byte werkt) is de breedte van de gegevensbus 8 bits; ze bestaat dus uit 8 parallelle geleiders.
- Een adresbus via dewelke de microprocessor het geheugen of de invoer/uitvoer poorten kan adresseren.
  De breedte van de adresbus is meestal 16 bits, dit in verband met het aantal te adresseren geheugencellen. Met 16 bits is het inderdaad mogelijk 2^{16} of 65536 (64K) geheugencellen te adresseren.
- Een controlebus of stuurbus.
  Via de signalen op de controlebus kan de microprocessor het systeem besturen, d.w.z. de gewenste doorschakelingen tussen de verschillende samenstellende schakelingen zo regelen dat de uitvoering van een bepaalde programma opdracht of van een gedeelte ervan mogelijk gemaakt wordt.

De uitbouw van het elementair microcomputer blokschema van fig. 1.1 tot een blokschema waar gebruik gemaakt wordt van busstructuur, brengt ons tot fig. 1.2.

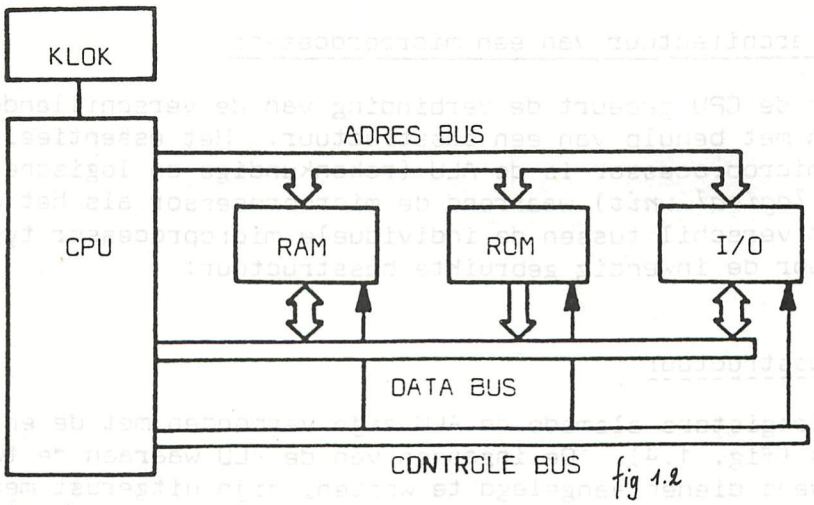


fig 1.2

In dit schema is het geheugen van de microcomputer tevens onderverdeeld in twee types :

- RAM geheugen (*random access memory*)
  Het is een geheugen waarvan elke cel willekeurig toegankelijk is en dat gebruikt wordt om veranderlijke gegevens en tussenresultaten in te schrijven of uit te lezen.
- ROM geheugen (*read only memory*).
  Dit is een geheugen waaruit enkel kan gelezen worden en dat meestal het programmageheugen is, d.w.z. het geheugen waarin het uit te voeren programma van een microprocessortoepassing is ondergebracht. Elk van deze geheugenblokken kan daarenboven nog uit verschillende modulen bestaan.

Als we tenslotte nog rekening houden met de speciale bouwstenen voor de in- en uitvoer, die zowel in serie als in parallel kan gebeuren, kan het blokschema uitgebreid worden tot dit van fig. 3.

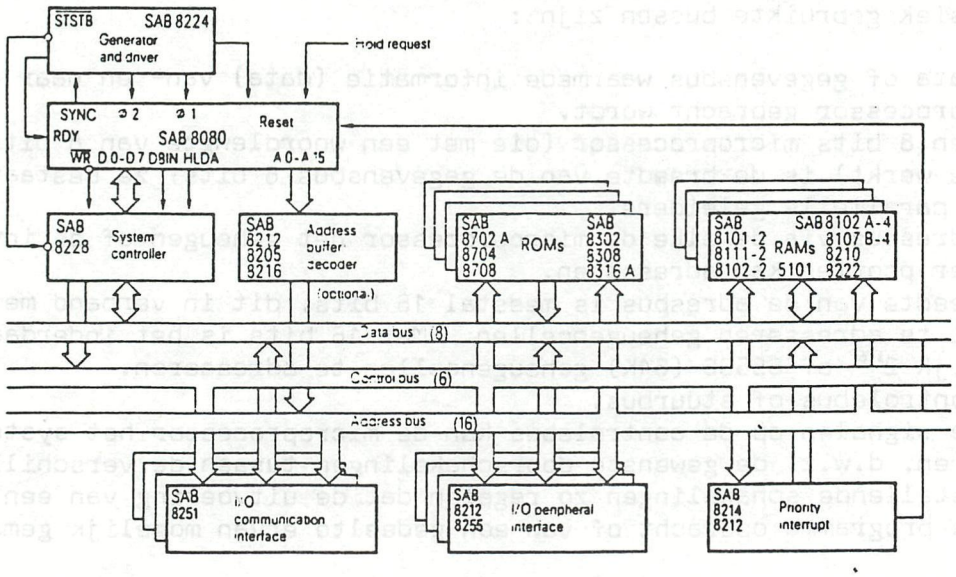


fig. 1.3

1.3. Inwendige architectuur van een microprocessor

Ook inwendig in de CPU gebeurt de verbinding van de verschillende samenstellende delen met behulp van een busstructuur. Het essentieelste element in de microprocessor is de ALU (rekenkundige en logische eenheid : *arithmetic and logical unit*) waarrond de microprocessor als het ware is opgebouwd. Het verschil tussen de individuele microprocessor types wordt o.a. bepaald door de inwendig gebruikte busstructuur.

1.3.1. Enkel busstructuur

Alle inwendige registers alsmede de ALU zijn verbonden met de enige inwendige databus (fig. 1.4). De ingangen van de ALU waaraan de twee te verwerken gegevens dienen aangelegd te worden, zijn uitgerust met een bufferregister.

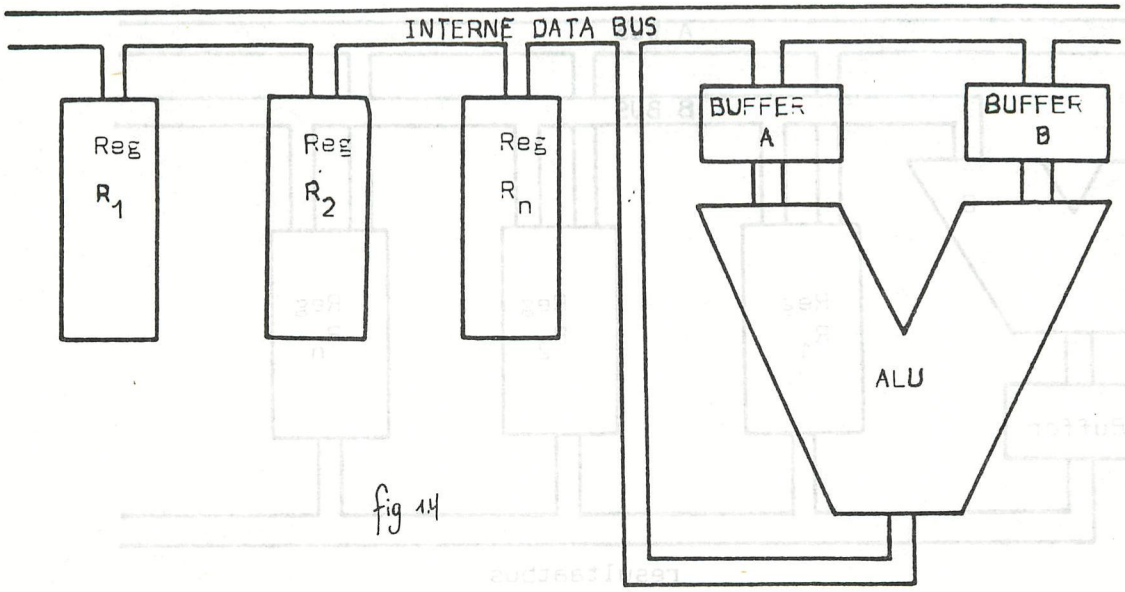


fig 14

Om een inzicht te verkrijgen in de werking van zulk systeem veronderstellen we dat de inhoud van R_1 en R_2 dient opgeteld te worden en dat het resultaat in R_1 dient ingeschreven te worden. De volgorde van de bewerkingen is :
Inhoud reg 1 in buffer A
Inhoud reg 2 in buffer B
Optellen van inhoud van buffer A en buffer B : resultaat naar reg 1.
Het register waarin het resultaat wordt ingeschreven (reg 1) en dat ook één van de op te tellen grootheden bevatte, noemt men de accumulator.
Voor alle operaties die dienen uitgevoerd te worden met de ALU dient een van de operanden ingeschreven te worden in de accumulator. Het resultaat wordt ook steeds en dit automatisch in de accumulator ingeschreven. Het volstaat dus in fig. 1.4 één van de registers R_1,R_n te voorzien voor deze functie en dus de naam accumulator te geven.
Bij al deze bewerkingen verloopt het transfert van de informatie via de inwendige bus die de naam draagt van inwendige databus.
Het voordeel van deze enkele busstructuur ligt in de plaatswinst op de geïntegreerde schakeling (er moet slechts één bus geïntegreerd worden).
Het is ook daarom dat de meeste microprocessoren volgens deze architectuur samengesteld zijn. Het nadeel ligt in de snelheid van uitvoering van de bewerkingen. Inderdaad dienden drie transferten plaats te vinden om een optelling te kunnen uitvoeren.

1.3.2. Multibus architectuur

Fig. 1.5 geeft een voorbeeld van een 3 bus architectuur voor een microprocessor. Elk van de ingangen van de ALU alsmede de uitgang zijn aangesloten op één bus.
Bij dezelfde optelling als hoger, gebeuren de bewerkingen nu als volgt.
De inhoud van register 1 wordt op ingang A gebracht via bus A.
Tegelijkertijd wordt de inhoud van register 2 op ingang B gebracht via bus B.
Het resultaat van de optelling wordt ingeschreven in register R_1 via de resultaatbus, vanuit de uitgangsbuffer van de ALU.
Het is duidelijk dat hierdoor de verwerkingssnelheid hoger is.
Deze architectuur wordt slechts zeer uitzonderlijk gebruikt bij 8 bits microprocessoren maar meer bij de 16 bits types.

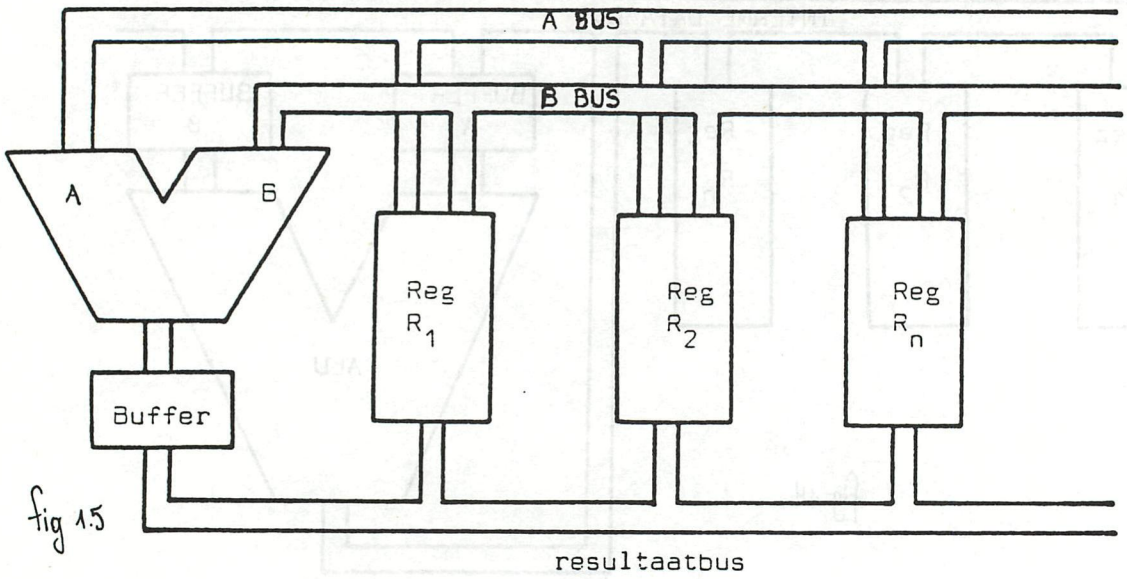


fig 1.5

1.4. Indelingscriteria voor microprocessoren

De indeling van microprocessoren kan gebeuren volgens verschillende criteria zoals :

- de familie bepaald door de constructeur waartoe de microprocessor behoort.
- de lengte of het aantal bits waaruit een datawoord bestaat.
- het aantal IC's dat dient gebruikt te worden om een microcomputer samen te stellen.

1.4.1. Indeling volgens de constructeur

Alhoewel de microprocessoren nog vrij jong zijn (de eerste werd geïntroduceerd in 1971) kennen we reeds een zeer grote verscheidenheid aan constructeurs en dus aan types.
In annex 1 wordt een tebel gegeven met een overzicht van de constructeurs en de door hen gefabriceerde microprocessoren. Er zijn echter slechts een viertal types die voldoende universeel gebruikt worden om als industriële standaard in aanmerking te komen. Het betreft :

- de familie van de 8080, 8085, 8086 van de firma INTEL met belangrijke second source producenten, waaronder SIEMENS.
- de familie van de 6800, 68000 van de firma MOTOROLA
- de familie van de Z80, Z8000 van de firma ZILOG
- de familie 650X van de firma MOSTEK

In dit boek bespreken we deze vier types, maar wat betreft programmatie en schakelschema's zal de nadruk gelegd worden op de familie van INTEL-SIEMENS.

1.4.2. Indeling volgens woordlengten

Volgens woordlengte kennen we de meest uiteenlopende cijfers, gaande van de 1 bit processor van MOTOROLA tot 32 bits processors zoals IAPX32 van INTEL met als tussenstadia 4, 8 en 16 bits.
De 8 bits en 16 bits processoren zijn tot op heden de meest gebruikte en net zijn deze die we in het bijzonder bestuderen.

Merken we op dat de woordlengte niet mag verward worden met de breedte van de adresbus. Inderdaad, de meeste 8 bits processoren (die dus met 8 bits data werken) maken gebruik van een 16 bits adresbus om voldoende adresseringscapaciteit te hebben. Met een 16 bits adresbus is het inderdaad mogelijk 65536 geheugencellen te adresseren.
De 16 bits processoren hebben bv. een adresbus van 20 bits (INTEL 8088), of 23 bits (ZILOG Z8000). Hiermede kunnen, gebruik makend van de zogenaamde segmenteringsadressering, respectievelijk 1MByte of 8 MByte geadresseerd worden. (zie verder).

1.4.3. Het aantal IC's dat dient gebruikt te worden

Bij de verdere evolutie van de integratietechnieken is het mogelijk geworden microprocessoren, geheugens en I/O poorten onder te brengen op één chip. Derhalve is men gaan spreken van multi-chip en één-chip microcomputers. In annex 2 is een overzicht gegeven van de verschillende types van elke constructeur met vermelding van de functies die ze vervullen.

1.5. Adresseringsmethodes

Vermits een microprocessor een programmeerbare LSI bouwsteen is, zal zijn taak er altijd in bestaan een programma uit te voeren.
Een programma bestaat, zoals we verder in detail bestuderen, uit een reeks programma opdrachten of instructies die in niet vluchtige geheugens ingeschreven zijn. Deze instructies dienen stuk voor stuk opgeroepen te worden om ze te kunnen uitvoeren.
Om een instructie op te roepen wordt een geheugencel geadresseerd waarna de inhoud van deze cel (instructie) naar de microprocessor gebracht wordt om daar behandeld te worden.
Het adresseren kan op verschillende wijzen gebeuren.
Gegevens zowel als adressen worden voorgesteld onder hexadecimale vorm.
Vermits een adres bestaat uit 16 bits kan het voorgesteld worden door een hexadecimaal getal van 4 digits. (elk hexadecimaal digit stelt immers 4 bits voor).
De totale adresomvang gaat bij een 16 bits adres van 0000 Hex tot FFFF Hex (fig. 1.6).

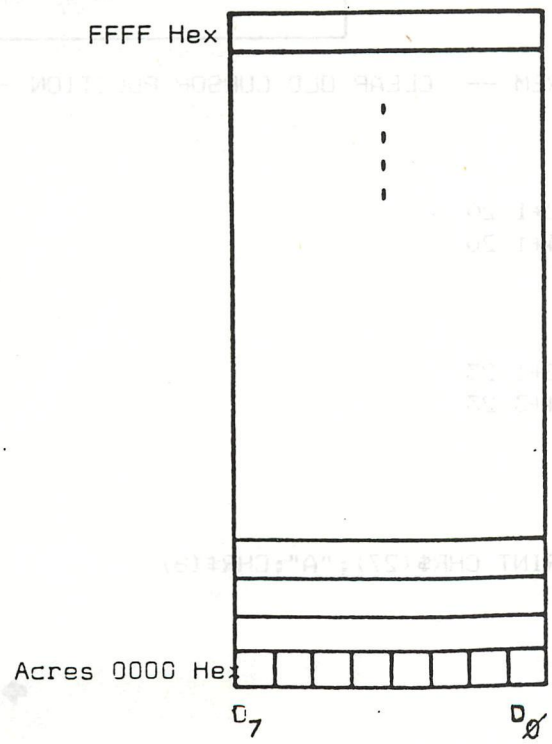


fig. 1.6

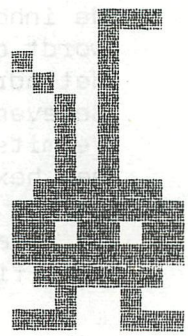
program identification

title	:	SHAPE-DESIGN (for FFGT & machine language prgs)
author	:	W.Hermans
purpose	:	to create objects in a very easy way
comment	:	we will extend the program in next issue ...

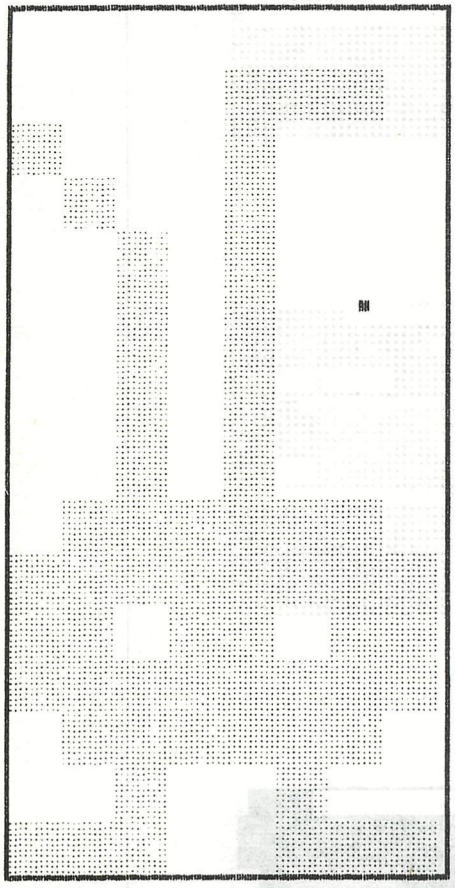
```
1     POKE #131,1:REM --- SCREEN ONLY ---
5     CLEAR 3000
10    MODE 6:COLORG 0 5 10 15
20    DIM A(8,16):REM --- ARRAY FOR PIXEL INFORMATION ---
30    FILL 9,9 91,171 21:FILL 10,10 90,170 20
35    FILL 200,100 217,133 20:FILL 120,100 153,165 20
37    FILL 250,100 259,117 20
40    X=1:Y=1:REM --- CURSOR START ---
100   IF GETC<>0 THEN 100
110   G=GETC:IF G=0 THEN 110
120   IF G=ASC("C") THEN 20:REM --- CLEAR ALL ---
125   IF G=13 THEN GOSUB 1000:REM --- CLEAR PIXEL ---
130   IF G=32 THEN GOSUB 2000:REM --- SET PIXEL ---
132   IF G=ASC("F") THEN GOSUB 6000:REM --- FILL ALL ---
135   IF G=9 THEN GOSUB 3000:REM --- PRINTOUT ---
140   IF G>=16 AND G<=19 THEN GOSUB 900
150   FILL X*10+5,Y*10+5 X*10+6,Y*10+6 23
160   GOTO 100:REM NEXT ACTION
200   IF Y=16 THEN RETURN
210   Y=Y+1:RETURN
300   IF Y=1 THEN RETURN
310   Y=Y-1:RETURN
400   IF X=1 THEN RETURN
410   X=X-1:RETURN
500   IF X=8 THEN RETURN
510   X=X+1:RETURN
900   COL=20:IF A(X,Y)=1 THEN COL=22
905   FILL X*10+5,Y*10+5 X*10+6,Y*10+6 COL:REM --- CLEAR OLD CURSOR POSITION ---
910   ON G-15 GOSUB 200,300,400,500
920   RETURN
1000  FILL X*10,Y*10 X*10+9,Y*10+9 20
1010  FILL 200+X*2,100+Y*2 200+X*2+1,100+Y*2+1 20
1012  FILL 120+X*4,100+Y*4 120+X*4+3,100+Y*4+1 20
1015  DOT 250+X,100+Y 20
1020  A(X,Y)=0:RETURN
2000  FILL X*10,Y*10 X*10+9,Y*10+9 22
2010  FILL 200+X*2,100+Y*2 200+X*2+1,100+Y*2+1 23
2012  FILL 120+X*4,100+Y*4 120+X*4+3,100+Y*4+3 23
2015  DOT 250+X,100+Y 23
2020  A(X,Y)=1:RETURN
3000  REM HEX-CONVERSION
3002  MODE 6A:INPUT " NAME ";N$
3005  POKE #131,0:PRINT TAB(20);N$:PRINT :PRINT CHR$(27);"A";CHR$(8)
3010  FOR Y=16 TO 1 STEP -1
3020  V=1:T=0
3030  FOR X=8 TO 1 STEP -1
3040  IF A(X,Y)=1 THEN T=T+V
3050  V=V SHL 1
3060  NEXT
```

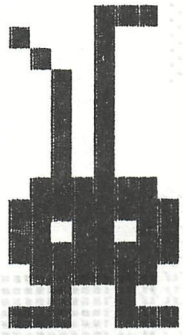
SCOOBY

```
# 0
# E
# 0
#80
#40
#20
#20
#20
#20
#7E
#FF
#D8
#FF
#7E
#24
#E7
```



```
3070  PRINT "#";:IF LEN(HEX$(T))=1 THEN PRINT " ";
3071  PRINT HEX$(T);TAB(10);
3072  D$=""
3075  FOR P=1 TO 8
3077  IF A(P,Y)=1 THEN GOSUB 4000:REM --- PIXEL ON ---
3078  IF A(P,Y)=0 THEN GOSUB 5000:REM --- PIXEL OFF ---
3080  NEXT:PRINT CHR$(27);"K";CHR$(64);CHR$(0);D$:REM --- BIT GRAPHICS ON EPSON ---
3085  NEXT:PRINT :PRINT
3090  POKE #131,1
3100  IF GETC<>0 THEN 3100
3110  IF GETC=0 THEN 3110
3120  X=1:Y=1:MODE 6:RETURN
4000  FOR D=1 TO 8:D$=D$+CHR$(#FF):NEXT
4020  RETURN
5000  FOR D=1 TO 8:D$=D$+CHR$(0):NEXT
5020  RETURN
6000  FILL 10,10 90,170 22:FILL 200,100 216,132 22
6010  FILL 250,100 258,116 22
6020  FOR M=1 TO 8:FOR N=1 TO 16:A(M,N)=1:NEXT:NEXT:REM --- ALL PIXELS ON ---
6030  RETURN
```





LOW-RES

MED.-RES

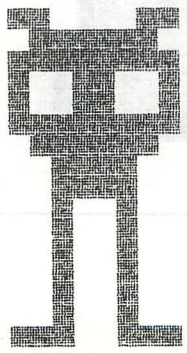
HIGH-RES

THE COMMANDS :

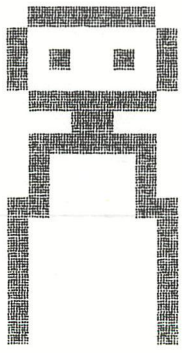
---

* CURSOR-KEYS : MOVE CURSOR
* SPACE-BAR : SET PIXEL
* RETURN : CLEAR PIXEL
* TAB : PRINT
* C : CLEAR ALL PIXELS
* F : SET ALL PIXELS

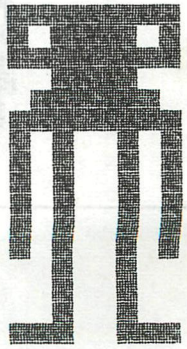
```
#C3
#7E
#FF
#99
#99
#FF
#7E
#3C
#3C
#24
#24
#24
#24
#24
#24
#E7
```



```
#7E
#81
#A5
#81
#7E
#18
#7E
#42
#42
#C3
#81
#81
#81
#81
#81
```

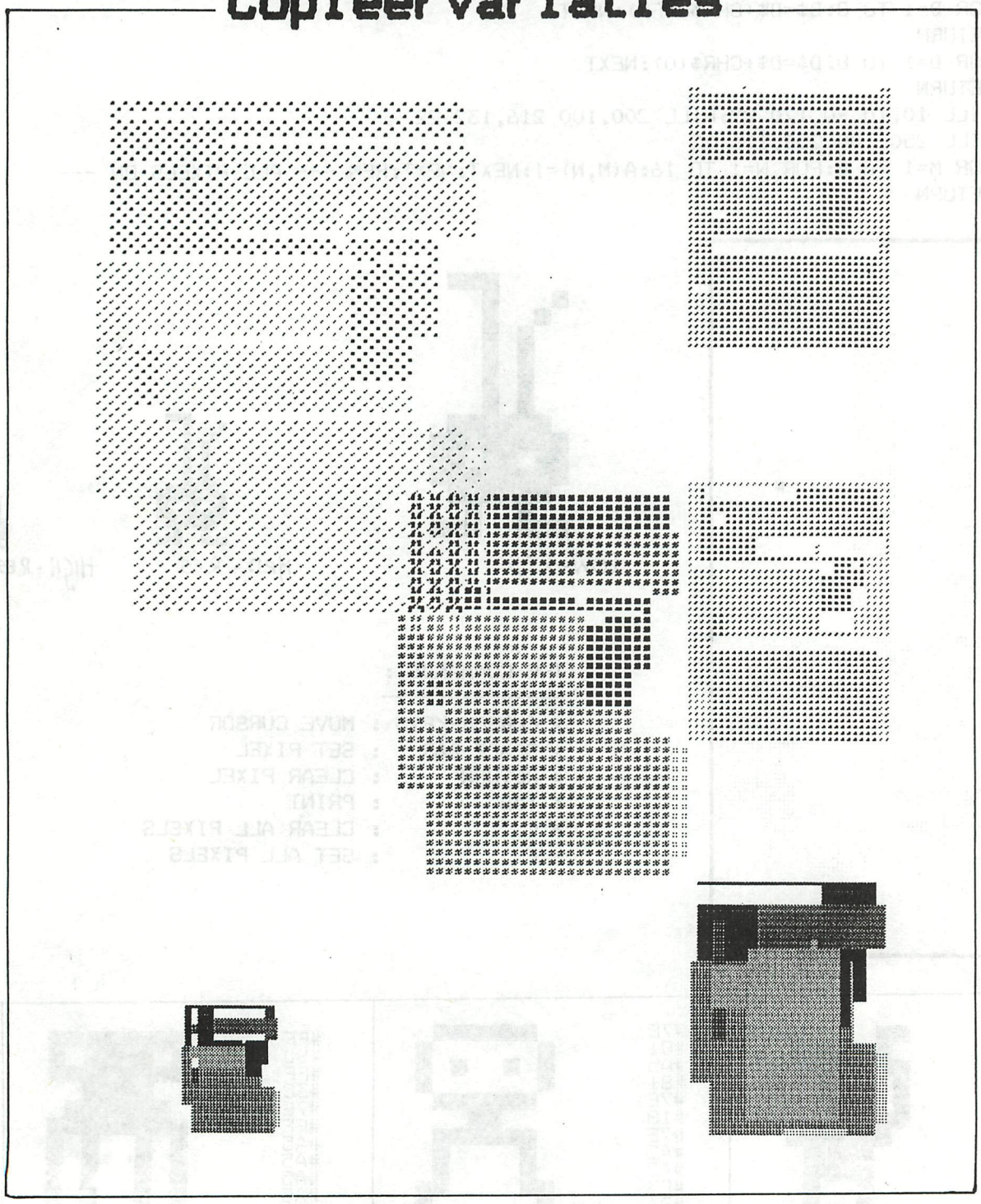


```
#FF
#BD
#FF
#3C
#7E
#FF
#A5
#A5
#A5
#A5
#A5
#A5
#24
#24
#24
#E7
```



```
10    REM BRAM VINGERLING / COPIEERVARIATIES
100   MODE 0:COLORG 0 5 10 15:MODE 5
110   XM2=XMAX/2:YM2=YMAX/2:XM4=XMAX/4:YM4=YMAX/4
120   FOR I=1 TO 20:FILL RND(XM4),RND(YM4) RND(XM4),RND(YM4) RND(16):NEXT
130   FOR X=0 TO XM4 STEP 2:X2=2*X:X3=X*3/2
140   FOR Y=0 TO YM4 STEP 2:Y2=2*Y:Y3=Y*3/2:C=SCRN(X,Y)
150   DOT X/2,YMAX-YM4+Y/2 C:C2=C/2:C3=C*3/4:DOT XMAX-XM4+X,Y C2
160   FILL XM4+X3,YM4+Y3 XM4+X3+1,YM4+Y3+1 C:DOT XM2+X2,YM2+Y2 C
170   DOT XM2-XM4/2+X,Y C3:DOT XM2+X2+2,YM2+Y2+2 C:NEXT:NEXT
180   GOTO 120
```

Copieervariaties



alle adressen in HEXvorm!

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

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

75	cursor symbol
74	cursor mode
72-73	cursor position

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

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

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

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

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

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

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

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

CTRL&COLOR BYTES IN A-MODE

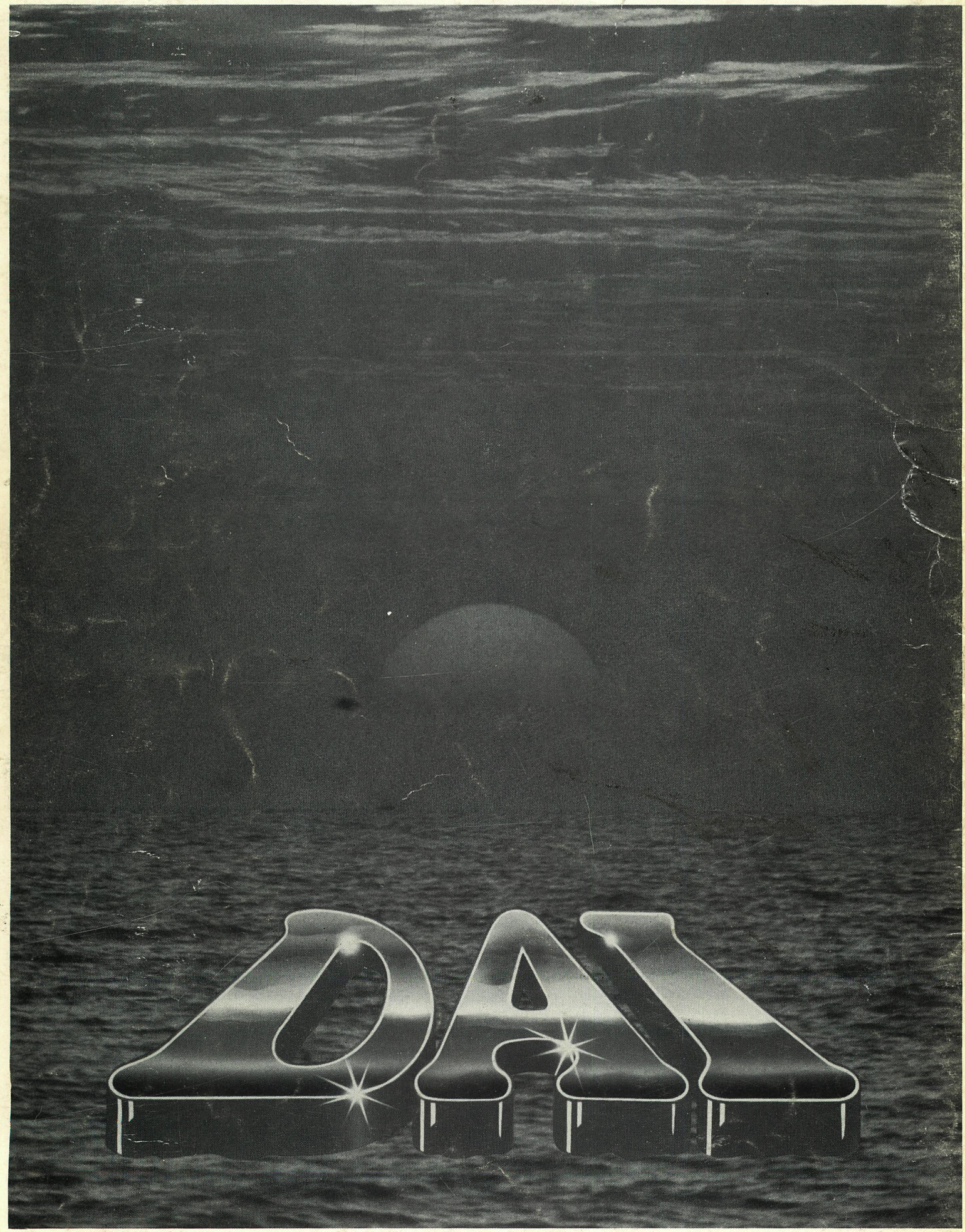
MODE	CTRL	COLOR	LIJN
1A/2A	BAE7	BAE6	3
	BA61	BA60	2
	B9DB	B9DA	1
	B955	B954	0
3A/4A	ACD3	ACD2	3
	AC4D	AC4C	2
	ABC7	ABC6	1
	AB41	AB40	0
5A/6A	7557	7556	3
	74D1	74D0	2
	744B	744A	1
	73C5	73C4	0

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

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

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

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


QA