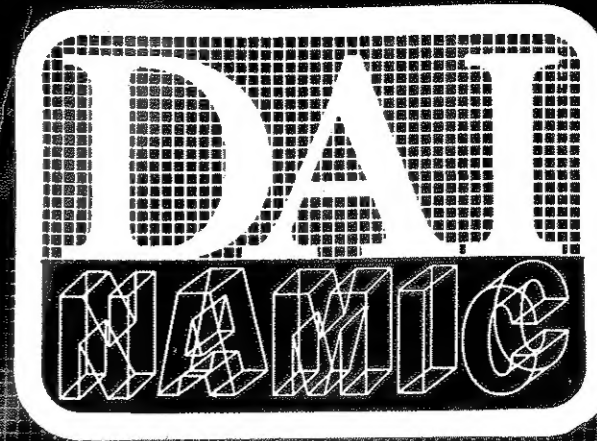
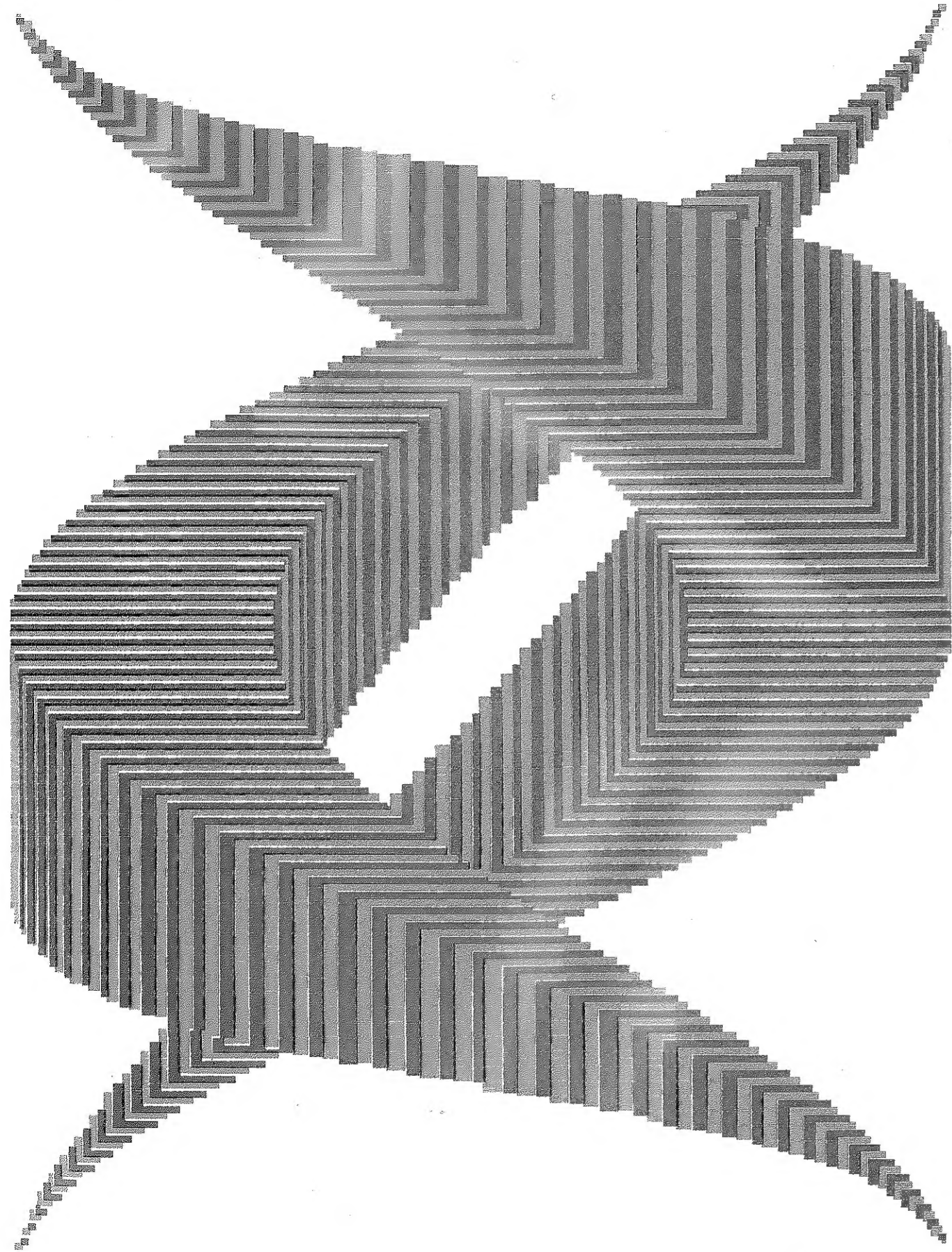
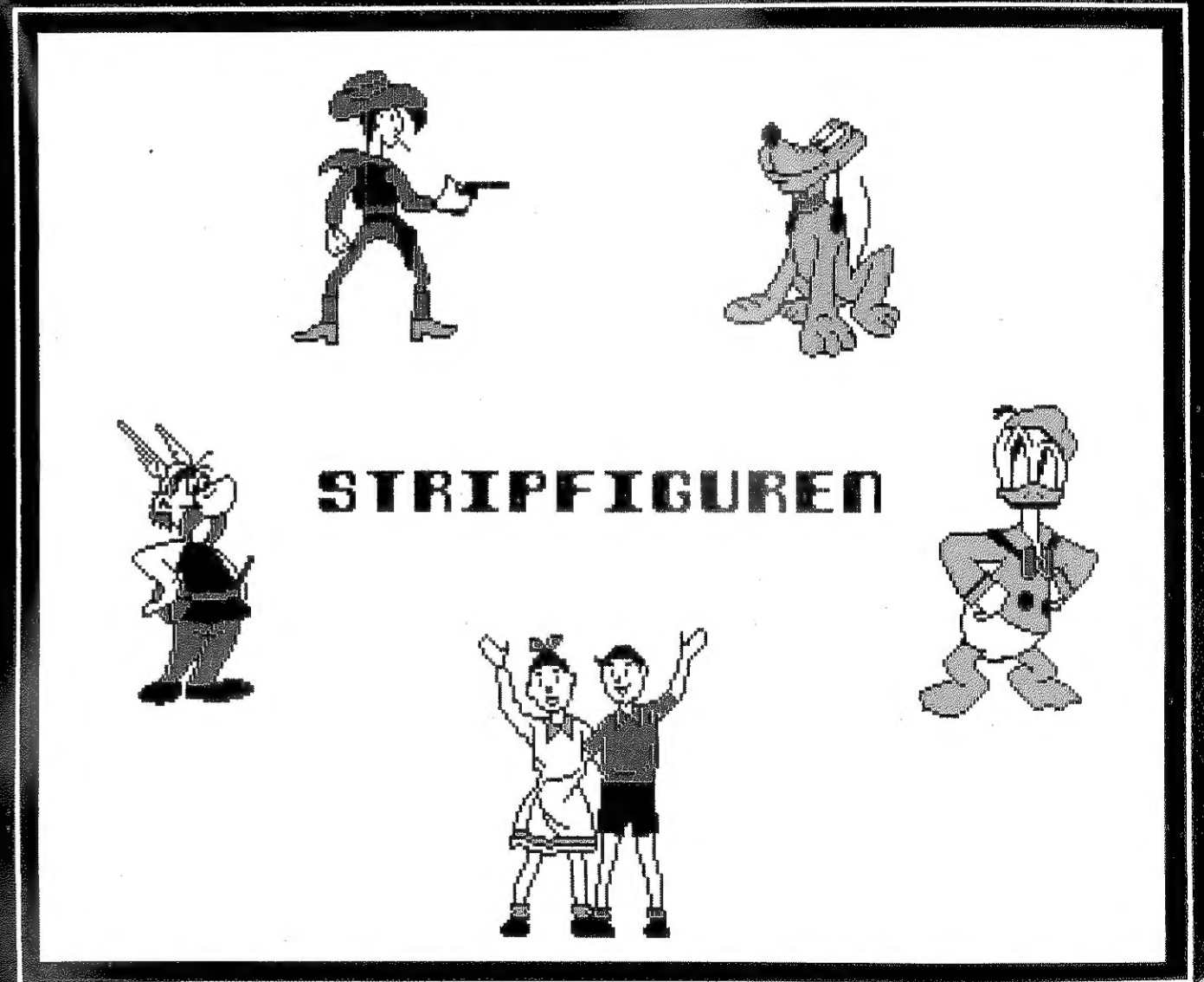


COLORB 15 15 15 0  
\*COLORB 15 0516 15  
\*CALLM#300



31

tweemaandelijks tijdschrift november - december 1985



personal computer users club

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

*International*

**COLOFON**

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

DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.  
 Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans Kredietbank Herselt  
 Mottaart 20 nr. 401-1009701-46  
 3170 Herselt BTW : 420.840.834  
 Tel. 014/54 59 74

Lidgelden / Subscriptions

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

Voor Nederland : Pour la France :

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

Inzendingen : Games & Strategy

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

**DAINAMIC**

PERSONAL COMPUTER USERS CLUB

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

**belangrijke ASCII-waarden in DAInpc**

functie/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	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	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	EOF	DC4	\$	4	D	T	d	t
5	0101	ESC	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	REL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	VS	/	?	O	_	o	DEL

**Remark**

Herselt, dec '85

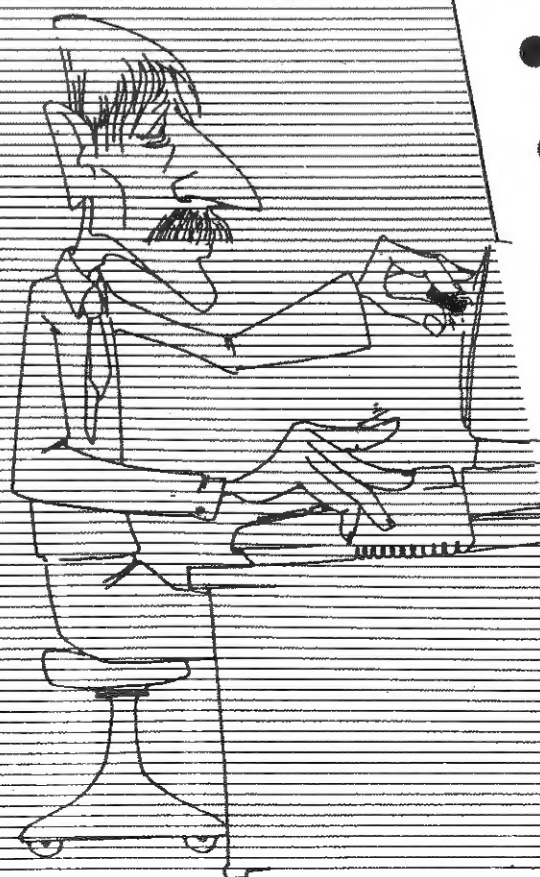
Beste Leden,

Mogelijk is 1986 al een paar dagen oud als U dit  
 voorwoord leest. We hebben de uitgave van dit  
 nummer uitgesteld tot de onderhandelingen met onze  
 collega's van de waalse clubs rond waren. Dat is  
 ondertussen gebeurd en we kunnen U bevestigen dat  
 U voortaan een gezamenlijke uitgave van DAInamic  
 en CLIC mag verwachten. In een vorig nummer hebben  
 we al geschetst wat dit betreffende de inhoud van  
 ons tijdschrift zal betekenen: meer informatie,  
 meer variatie. Vele leden hebben al gereageerd op  
 onze vorige oproep om nog eens wat mooie  
 sturen. Met dank hebben we heel wat mooie  
 inzendingen kunnen ontvangen. Uiteraard moeten we  
 bij de jaarwisseling aan uw deur kloppen met  
 het verzoek om uw lidmaatschap zo spoedig mogelijk  
 te hernieuwen. Het lidgeld is niet gewijzigd, de  
 rekeningnummers waarop U dit bedrag kwijt kan ook  
 niet. Onze verjaardagswedstrijd was blijkbaar te  
 gemakkelijk. We mochten slechts een paar foute  
 antwoorden ontvangen:  
 computerfreund, computerfreeks, computerfanbal. Het  
 lot was gunstig voor Koen Van de Perre: een  
 onschuldige kinderhand bepaalde dat hij eigenaar  
 wordt van een mooie RGB-monitor. We zullen de  
 volgende wedstrijd beslist moeilijker maken...  
 Rest mij nog U de beste wensen voor 1986 over te  
 maken vanwege het ganse DAInamic team.

dear members,

As announced in a previous issue, DAInamic will  
 start a close cooperation with the colleagues of  
 DAIn CLIC. This can only mean that you will find  
 more information in our magazine. We received a  
 lot of articles last month: many thanks to all  
 these creative spirits... As 1986 is already a few  
 days old, we have to knock on your door with our  
 best wishes for the new year ... and our  
 invitation to renew your subscription. The price  
 remains the same, the banknumbers also. The first  
 price of our anniversary contest goes to Koen van  
 de Perre who thought that COMPUTERFREAKS was the  
 solution. The contest was too easy, next time we  
 will create a real mindbreaker!

Best wishes from the whole DAInamic team,  
 keep on DAIn-ying! (but stay alive!)



**DAInamic redaction!**

319	REMARK	REDACTION
320	CONTENTS	REDACTION
321	PROGRAMMEERTECHNIEKEN	F.DRUIJFF
325	ADC-CONVERTOR	J.VAN OOL
330	DAI RESET	H.RISON
331	RGB MONITOR	H.RISON
334	CENTRONICS PRINTERS	G.CATHCART
339	EMULATE & STORE	G.CATHCART
342	ACK LATCHED	G.CATHCART
344	KLEURCONFLICTEN	H.BELLEKENS
349	DRAW OR POKE	H.BELLEKENS
351	FGT EXTENSION	J.VAN OOL
352	EXOR	J.VAN OOL
353	STATISTISCHE ANALYSE	F.VERBERCKMOES
361	GOSSIM	F.VAN AMERONGEN
362	ASSEMBLY PROGRAMMING	C.READ
369	KOMPAKT TEKENING	M.STOUT
371	DIDACOM SUBROUTINES	I.BROEKMAN
373	A TASTE OF SUPERFONT	REDACTION
374	TRANSLATIONS	DAInamic UK
380	DAISTAR	K.PETER

Niets uit deze uitgave mag worden veelevoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de uitgever.

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.

Ondanks alle aan de samenstelling van de tekst bestede zorg, kan noch de redactie noch de uitgever aansprakelijkheid aanvaarden voor eventuele schade, die zou kunnen voortvloeien uit enige fout, die in deze uitgave zou kunnen voorkomen.

Ik vervolg de verhandeling over de goniometrische functies maar zal trachten nu wat minder theoretisch te zijn en meer praktisch gerichte toepassingen te laten zien. Wel moet er eerst nog een klein beetje theorie doorgenomen worden. Ik kreeg een aantal reacties dat met name nog niet duidelijk was hoe je zowel de sinus van een hoek kon nemen als de sinus van een getal en hoe dat dan samenhangt.

**Theorie**

Zoals ik in vorig artikel reeds beschreef kunnen we de sinus als volgt zien: Neem een cirkel met straal 1 en laat een punt gelijkmatig langs de cirkel bewegen. De afstand van dat punt tot de horizontale lijn door het middelpunt van de cirkel is dan de sinus van de hoek alpha. Deze hoek alpha wordt gevormd door de horizontale lijn door het middelpunt van de cirkel en de verbindingslijn van het middelpunt van de cirkel en het langs de cirkel bewegende punt.

**Moeilijk ?**

Teken voor uzelf een cirkel met een straal van b.v. 5 cm. Maakt U de tekening op ruitjespapier met hokjes van 5mm x 5mm is de straal precies 10 hokjes lang. Noem het middelpunt van de cirkel M en teken vervolgens een horizontale lijn door het middelpunt. Het punt waar deze middellijn door de cirkel gaat (aan de rechterkant !) noemen we S. Kies op de cirkel een punt P en teken de lijn van M door P. De hoek die wordt gevormd door de twee lijnen die er nu staan noemen we alpha. Bedenk wel dat we altijd de hoek bedoelen gemeten vanaf de horizontale lijn tegen de wijzers van de klok in naar het andere been.

De sinus van de hoek alpha is nu de afstand van punt P tot de horizontale lijn. Ligt punt P boven de lijn en is de hoek dus tussen 0 en 180 graden hebben we direct de sinus ligt punt P echter onder de horizontale lijn dan zetten we er een min (-) voor. Al de afstanden worden genormaliseerd d.w.z. in verhouding tot de cirkel

gegeven. Ligt punt P in ons geval 7 hokjes boven de horizontale lijn is de sinus van de bijbehorende hoek niet 7 (van de hokjes) of 35 (van de millimeters) of 3,5 (van de centimeters) maar 0,7. Om tot deze waarde te komen nemen we de verhouding van de lengte van de afstand van P tot de horizontale lijn en de lengte van de straal van de cirkel. Hier dus 7 hokjes / 10 hokjes = 0,7. Het is met de tekening ook simpel in te zien dat er twee hoeken tussen 0 en 180 graden zijn met een sinus van 0,7. Verder kunnen we ook simpel zien dat een sinus nooit groter dan 1 kan zijn. En dat de sinus van nul graden (punt P ligt dan op S) nul bedraagt.

**golvend**

Laten we punt P langs de cirkel lopen zien we dat de sinus begint bij 0 dan steeds groter wordt en bij een hoek van 90 graden 1 geworden is. Dan neemt de waarde weer af tot hij bij 180 graden 0 is geworden, verdergaand langs de cirkelonderkant wordt de sinuswaarde kleiner en is bij 270 graden -1, vanaf dat punt neemt hij weer toe (maar blijft negatief want aan de onderkant) om bij 360 graden weer op 0 uit te komen. Draaien we door (je zou dan kunnen spreken van een hoek van 450 graden) herhaalt de hele cyclus zich. We krijgen dan een golvende beweging tussen -1 en +1. Deze beweging is in vele processen die niets met hoeken te maken hebben terug te vinden. Denk hierbij maar aan de rok lengte. Men ging dus zoeken naar een vorm van definitie voor de sinus zonder het hoekbegrip nodig te hebben.

**Getal ipv hoek**

De waarde van de sinus wordt ook nu weer gerelateerd aan de cirkel. We beginnen bij punt S dus het snijpunt (rechts) op de horizontale lijn door het middelpunt van de cirkel met de cirkel zelf. Dan gaan we tegen de wijzers van de klok in (linksom) de cirkel volgen. We houden de afstand bij die we hebben afgelegd met punt P. De sinus die bij de afstand hoort is identiek aan de sinus van de hoek gedefinieerd, dus de afstand van punt P tot de horizontale lijn.

Daar de definitie identiek is is het simpel in te zien dat er ook nu een golvende beweging tussen -1 en 1 zal ontstaan. Na welke afstand zal de cirkel echter een maal gerond zijn? Of anders gezegd bij welke afstand begint de cyclus opnieuw?

### Verhouding

Bij een poging tot schatten van dit getal komen we al snel tot een waarde tussen 6 en 7. Nauwkeuriger benaderen leert ons de waarde 6,28.. Maar de puntjes staan er niet voor niets de waarde is niet precies 6,28 maar iets meer. Vroeger ging men niet uit van de straal van de cirkel maar van zijn middellijn.

### PI

Het getal dat de verhouding van middellijn en omtrek van een cirkel aangeeft wordt aangeduid met  $\pi$ . Op de DAI zit deze waarde (in benadering) onder de naam PI.

Vragen we aan de DAI PI krijgen we :

```
*?PI
3.14159
```

Om het enigzins nauwkeuriger te krijgen kunnen we om PI-3 vragen :

```
*?PI-3
.141593
```

We krijgen zo nog een betrouwbare 3 achter de 9. Het proces kunnen we theoretisch herhalen tot we velen cijfers van PI achter het decimaal teken kennen. De DAI laat ons echter al snel in de steek :

```
*?PI-3.1
4.15928E-2
```

De 2 (achter de 9) is correct maar de 8 is beslist fout.

Menig lezer zal opgevallen zijn dat ik net een 3 achter de 9 als goed beoordeelde terwijl bij de volgende benadering een 2 werd gevonden. Beide zijn hier goed ! Maar lees het a.u.b. goed het zesde cijfer achter het decimaal teken is een 2, daar echter het zevende cijfer meer dan 5 is dient PI afgerond op zes cijfers achter het decimaal teken als laatste cijfer een 3 te hebben.

Wat PI precies is kan niemand U in decimale notatie laten zien daar het een waarde is die niet zo te schrijven is. Wel kunnen we PI net zo nauwkeurig als we wensen decimaal benaderen. En bedenk als we het op b.v. 10 cijfers achter het decimaal teken afronden die afgeronde waarde minder dan een twintig miljardste scheelt van de werkelijke waarde.

Voor diegenen die het nauwkeuriger willen weten volgt hier PI met wat meer cijfers dan gebruikelijk is.

```
3.141592 653589 793238 462643 383279
502884 197169 399375 ..... ..
```

### Fout

De DAI gaat zoals we zien bij het zevende cijfer achter de decimaal in de fout en zelfs met een verschil van meer dan 2. Dit betekent dat als we PI vermenigvuldigen met 5 000000 (vijf miljoen) de fout meer dan een bedraagt. Dit soort verschillen leidt soms tot rampzalige fouten.

We tonen de fout eerst aan :

```
100 REM TEST NAUWKEURIGHEID PI
110 MODE 0:LIST:PRINT
120 T%=T%+1
130 PRINT "*** TEST No ";T%:" ***"
140 INPUT "WAARDE VOOR X ";X
150 PRINT
160 PRINT "SIN X =";
170 PRINT SIN(X)
180 PRINT "SIN X+2*PI =";
190 PRINT SIN(X+PI*2.0)
200 PRINT "SIN X+5000000*PI =";
210 PRINT SIN(X+PI*5E6)
220 PRINT :GOTO 120
```

De variabele X moet vanzelfsprekend een floating point variabele zijn. Tik het gegeven programma in en laat het lopen. Probeer meerdere waarden voor X en verbaas U over een tweetal opmerkelijke zaken.

- 1) de waarde van de sinus van X en van X + 2\*PI verschillen soms in de laatste decimaal bij kleine waarden van X.

Nemen we de X groot zal het verschil ook groter worden al is dit niet altijd zo.

- 2) De waarde van de sinus van X + 5000000\*PI lijkt vrijwel onafhankelijk van de ingevoerde waarde voor X.

De gegeven waarde voor de sinus zal vrijwel altijd -1.0, 0.0 of 1.0 zijn!

Nemen we de X echter zodanig dat hij weer aanzienlijk de totale waarde van X + 5000000\*PI in de buurt van nul krijgt krijgen we ook weer 'normale' sinuswaarden. De aldus gevonden waarden zijn echter totaal onbetrouwbaar en mogen dus niet als basis voor beslissingen worden genomen.

### Is het erg ?

Hoe komt deze fout tot stand en kunnen we hem voorkomen ? Ik heb de vorige keer reeds uitgelegd dat de waarden van sinus e.d. worden berekend door de DAI met behulp van een zogenaamde reeks.

Een van de bekendste reeksen waarmee de sinus berekend kan worden is de volgende :

$$\text{SIN}(X) = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{X^9}{9!} - \dots$$

De eerste term in deze reeks is ook te schrijven als  $X$  gedeeld door 1!. De reeks zit er dan nog mooier uit maar aangezien X tot de macht 1 hetzelfde is als X en 1! gelijk is aan 1 kunnen we ook alleen X zetten.

Ter herinnering :  $X = X \times X \times X$  en  $4! = 1 \times 2 \times 3 \times 4$

N.B. : Hier is x als maal bedoeld.

Bekijken we eerst de noemers van de termen :  $3! = 6$   $5! = 120$   
 $7! = 5040$   $9! = 362880$

We zien het werkelijk enorm snel oplopen en op het eerste gezicht worden de termen dus snel kleiner omdat ze door deze getallen worden gedeeld. Bekijken we echter de teller zien we dat daar X tot de derde, vijfde, zevende enz macht wordt

genomen en ook dat neemt voor X'n boven de 1 snel toe.

### kleine X

Is de X klein b.v. kleiner dan een tiende is de derde macht hiervan al kleiner dan een duizendste en dat wordt dan nog gedeeld door zes. De derde term in de reeks is kleiner dan een honderdduizendste gedeeld door 120 dus kleiner dan een tien miljoenste deze term beïnvloed dan het zevende cijfer achter het decimaal teken. De reeks behoeft in dit geval slechts tot de derde term uitgerekend te worden om al binnen zes cijfers nauwkeurig te zijn.

### rond 1

Bekijken we nu het geval dat X rond de een ligt. Willen we nu een nauwkeurigheid tot het zesde cijfer achter het decimaal teken zal de noemer groter dan tien miljoen moeten zijn. Dit is bereikt bij de zesde term in de reeks (1!) en ook hier behoeven we dus echt niet zoveel termen uit te rekenen om de sinus in de gewenste nauwkeurigheid te krijgen.

### grotere X

Is de X wat groter zo rond de tien bijvoorbeeld is het simpel in te zien dat we dan problemen kunnen krijgen met het uitrekenen van de teller als we te ver doorgaan in de reeks. De tiende term heeft teller X tot de macht 21 en dat kan de DAI al niet meer aan. Is de berekeningsvolgorde efficiënt dus bv om en om vermenigvuldigen en delen met de factoren uit teller en noemer kan misschien nog de term uitgerekend worden maar het probleem wordt alleen iets verschoven en niet echt opgelost.

N.B. De termen gaan trouwens altijd verderop in de reeks kleiner worden.

We zien dus dat het om meerdere redenen niet erg zinvol is om teveel termen van zo'n reeks te ontwikkelen. Ook voor andere reeksen die op de waarde van de sinus uitkomen gelden soortgelijke argumenten. Maar als de reeks altijd na een

beperkt aantal termen wordt afgebroken zal in sommige gevallen nog lang niet de juiste waarde bereikt zijn. Deze juiste waarde kan echter gezien de nauwkeurigheid waarin de DAI zelf werkt ook niet worden bereikt door veel meer termen te berekenen. Het dilemma van de ontwerpers van de firmware is hopelijk duidelijk. Men heeft bij DAI duidelijk en terecht gekozen voor vrijwel perfecte benaderingen voor de sinuswaarden als X tussen -10 en +10 zit.

Heeft U berekeningen waarbij U buiten deze grenzen zit en U heeft wel zo nauwkeurig mogelijke sinuswaarden nodig, kunt U het best deze sinuswaarden zelf berekenen met een geschikt algoritme (zoals een reeksontwikkeling) die wel geschikt is voor de waarden van X die U juist nodig heeft.

Voor erg kleine hoeken (minder dan 0.02) is de sinuswaarde van X vrijwel gelijk aan X zelf. Zie de gebruikte reeksontwikkeling waarbij de tweede term al hooguit het laatste cijfer nog iets laat variëren.

Gaan we dit gebruiken voor grafische toepassingen kunnen onze normen nog wat ruimer gesteld worden.

Gaan we b.v. een cirkel tekenen zal de gebruikte sinus met hoogstens 100 vermenigvuldigd worden. Dit betekent weer op zijn beurt omdat bij het plaatsen van een punt toch wordt afgerond op hele punten dat de waarde van de gebruikte sinus al in de derde decimaal fout mag zijn omdat dit na vermenigvuldiging met 100 toch achter het decimaal teken komt en die grafisch niet significant is. Net niet meer toegestane fout in dit geval dus 1/100. De X waarbij dit gebeurt is zo dat X tot de derde gedeeld door 6 kleiner is dan 1/100. We berekenen dus de derdemachtswortel uit 0,06 en komen zo tot ongeveer 0,39.

Een andere manier om uit te vinden waar deze grens ligt, vooral nuttig voor die lezers die de wiskunde van het voorafgaande te lastig vonden, is volgend programma :

```
10 INPUT X:PRINT
20 PRINT INT(100*SIN(X)),INT(100*X)
30 PRINT :GOTO 10
```

En met dit programma spelend zijn we er snel achter dat bij X is 0,39 nog dezelfde waarden voor INT(100.0\*X) en INT(100.0\*SIN(X)) gegeven worden. Deze waarde is overigens 0,38 en niet zoals verwacht 0,39.

De hoek die bij deze waarde hoort is 0.39 x 180 / PI graden en dat is ongeveer twee en twintig en een halve graad. Met deze wetenschap zou een cirkel met straal 100 al voor de helft zonder grote ingrepen te tekenen zijn met gebruik van X in plaats van de sinus van X.

Om de hele cirkel te kunnen tekenen moeten we of speciale trucs uithalen of de straal beperken.

We beredeneren als volgt :

De hoek moet 45 graden kunnen zijn. Dat betekent dat de waarde van X een kwart PI dus 0,785.. moet zijn.

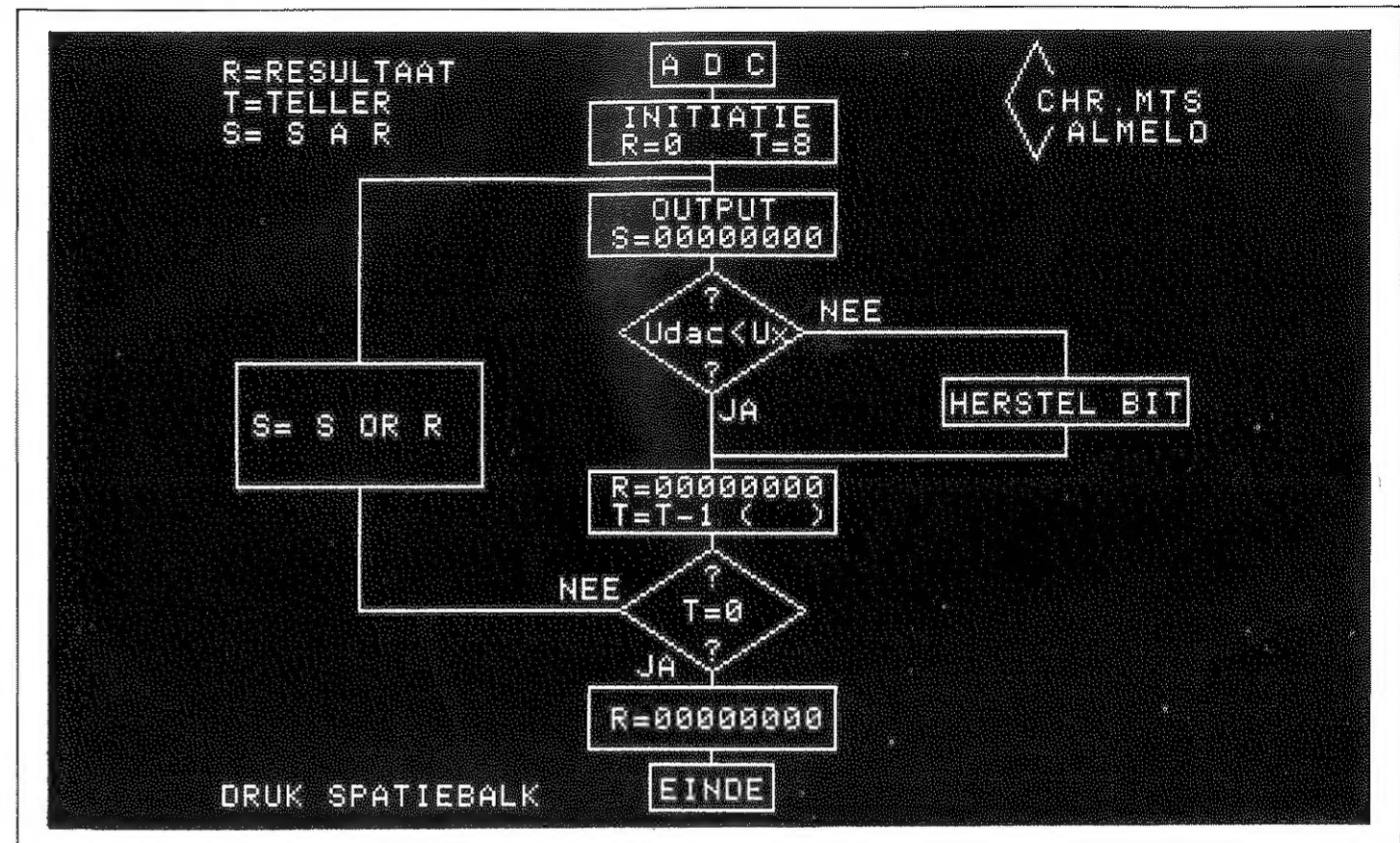
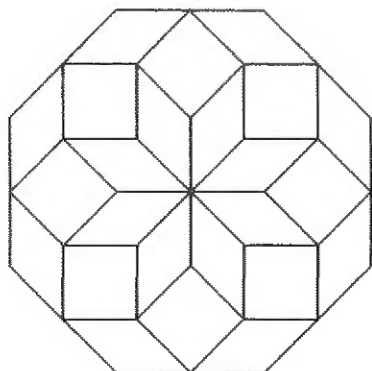
Wat is het verschil tussen de sinus van X en X zelf ? En bij welke straal is dit verschil groter dan 1?

Dit blijkt bij berekening iets meer dan zes te zijn maar onderstaand programma laat zien dat tot en met straal 10 er nog geen problemen zijn omdat we niet echt afronden.

```
10 INPUT R:PRINT
20 PRINT INT(SIN(0.785)*R),
30 PRINT INT(0.785*R)
40 PRINT :GOTO 10
```

Aan U de eer om met deze kennis nu een slim cirkelalgoritme te maken dat bruikbaar is voor het tekenen van kleine (straal<=10) cirkels.

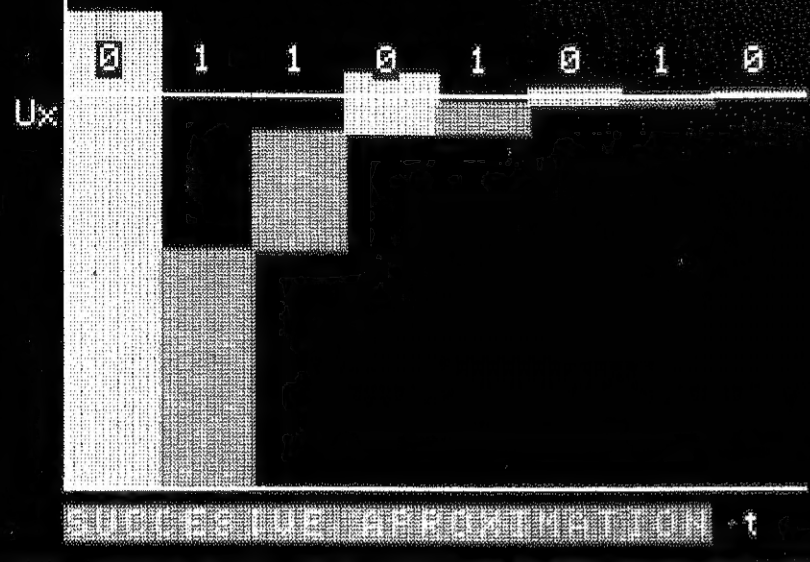
Frank H. Druijff



```
10 CLEAR 512: DIM BW(7.0), VH(8.0),
   B$(7.0): KL=21
20 ID=10: W=0: H=0: SP=7: Y1=0
30 GOTO 600
50 GOTO 1000
200 COLORT 0 14 0 0: H1=1
210 PRINT CHR$(12):: CURSOR 0,3: PRINT
   "INVOER VAN DE BINAIRE
   GEGEVENS...."
220 CURSOR 0,2: PRINT "VOER DE WAARDE
   '0' of '1' IN VAN BIT"
225 W=256: DW=0: X$=""
230 FOR TELLER=7 TO 0 STEP -1
240 1 POKE #75,32: CURSOR 38,2: PRINT
   1 SPC(10)
250 1 CURSOR 37,2: PRINT TELLER;" : ";
   1 POKE #75,95: INPUT B$(TELLER)
260 1 IF B$(TELLER)="" THEN 240
265 1 IF LEFT$(B$(TELLER),1)="-" THEN
   1 240
270 1 IF ASC(B$(TELLER))>49.0 THEN 240
275 1 BW(TELLER)=VAL(B$(TELLER))
280 1 IF BW(TELLER)<0.0 OR
   1 BW(TELLER)>1.0 THEN 240
285 1 IF BW(TELLER)=1.0 THEN DW=DW+W/2
290 1 W=W/2: X$=X$+B$(TELLER)
295 NEXT TELLER
340 PRINT CHR$(12): "DE BINAIRE WAARDE
   BEDRAAGT: "; X$; " = #"; HEX$(DW)
345 PRINT "DE DECIMALE WAARDE BEDRAAGT:
   "; DW; " mV."
```

```
350 GOSUB 9500: RETURN
400 CURSOR 0,0: PRINT "NOGMAALS
   HERHALEN <J/N> ";
410 IF H1=0.0 THEN PRINT : PRINT "ER IS
   NOG GEEN BINAIRE OF DECIMALE
   INVOER GEWEEST."
420 G=GETC: IF G=0 THEN 420
430 IF G=74.0 AND H1=0.0 THEN GOSUB
   9500
440 IF G=74.0 AND H1=1.0 THEN PRINT
   CHR$(12):: H1=0: GOSUB 200
450 IF G=74.0 AND H1=2 THEN PRINT
   CHR$(12):: H1=0: GOSUB 20000
460 IF G=74.0 AND H1=3.0 THEN PRINT
   CHR$(12):: H1=0: GOSUB 30000
480 IF G=78 THEN W=1: RETURN
500 GOTO 420
600 MODE 0: COLORT 9 15 0 0: PRINT
   CHR$(12)
605 CURSOR 10,21: PRINT "INTRODUCTIE"
610 CURSOR 0,19: PRINT "DIT PROGRAMMA
   GEEFT DE MOGELIJKHEID TOT HET
   BESTUDEREN"
620 PRINT "VAN DE ANALOOG NAAR
   DIGITAAL OMZETTER VOLGENS DE
   METHODE"
630 PRINT "VAN 'SUCCESSIVE
   APPROXIMATION'."
640 PRINT "DEZE METHODE VOLGT EEN
```

CHR. MTS  
ALNELO



```

PROCEDURE, DIE DE ANALOGE SPANNING
Ux"
650 PRINT "BENADERT DOOR EEN SPANNING
AAN EEN DIGITAAL NAAR ANALOOG-"
660 PRINT "OMZETTER AAN TE BIEDEN EN
DAARNA TE KIJKEN OF DE AANGEBODEN"
670 PRINT "SPANNING TE HOOG DAN WEL TE
LAAG IS. DIT GEBEURT ACHTEREEN-"
680 PRINT "VOLGENS, TOTDAT DE SPANNING
Ux IS BENADERD."
690 PRINT :PRINT "VOOR HET WERKELIJK
LATEN FUNKTIONEREN VAN DE OMZETTER"
700 PRINT "VOLGENS PUNT 7 VAN HET MENU,
DIENT MEN HET DAI-RWC-REK EN"
710 PRINT "DE BETREFFENDE INTERFACE
AAN TE SLUITEN OP DE DCE-BUS."
720 PRINT :PRINT "VEEL SUCCES."
730 CURSOR 36,CURY:PRINT "(C) 85
J.J.H. van Ool."
750 GOSUB 12000
1000 MODE 0:PRINT CHR$(12):COLORT 9 15
9 3:COLORG 0 10 1 14
1010 FOR CU=25 TO 31 STEP 2:POKE
#B3DA-2*CU+20*#86,#FF:NEXT
1020 CURSOR 25,20:PRINT "M E N U":PRINT
1030 PRINT "Algemeen symbool
A/D-omzetter <1>"
1040 PRINT "Schema van de
A/D-omzetter... <2>"
1050 PRINT

```

```

" SAR-algoritme..... <3>"
1060 PRINT "Binaire invoer van
gegevens.. <4>"
1070 PRINT "Decimale invoer van
gegevens. <5>"
1080 PRINT "Tekenen van de
grafiek..... <6>"
1090 PRINT "DAI-RWC met
A/D-Interface.... <7>"
1095 PRINT "Stoppen met het
programma.... <8>"
1100 PRINT :PRINT "Maak Uw keuze <1 t/m
8>..... ";
1200 INPUT K$:PRINT
1210 IF K$="" THEN 1100
1220 IF LEFT$(K$,1)="-" THEN 1100
1225 IF ASC(K$)>57.0 THEN 1100
1230 IF VAL(K$)<1 OR VAL(K$)>8 THEN 1100
1240 A=VAL(K$)
1245 PRINT CHR$(12);
1250 ON A GOSUB 9000,25000,2000,200,
20000,9450,30000,1300
1260 GOTO 1000
1300 MODE 0:PRINT CHR$(12):POKE #75,95:
END
2000 REM SAR-ALGORITME
2010 MODE 6:COLORG 0 13 3 14:Y1=50:
GOSUB 15000
2020 KL=21:GOSUB 6500:DRAW 165,140 165,

```

```

115 KL:GOSUB 6550
2030 GOSUB 6770
2040 SP=7:DF=0:CC=23
2050 A$="A D C":X=148:Y=242:GOSUB 10000
2060 A$="INITIATIE":X=135:Y=225:GOSUB
10000
2065 A$="R=0 T=8":Y=216:GOSUB 10000
2070 A$="OUTPUT":X=145:Y=195:GOSUB 10000
2080 A$="S=00000000":GOSUB 7115
2085 A$="?":X=162:Y=166:GOSUB 10000:Y=
142:GOSUB 10000
2090 A$="Udac<Ux":X=143:Y=155:GOSUB
10000
2095 GOSUB 7125:GOSUB 7175
2100 A$="HERSTEL BIT":X=242:Y=132:GOSUB
10000
2110 A$="R=00000000":GOSUB 7225
2120 A$="T=T-1 ( )":X=132:Y=96:GOSUB
10000
2130 A$="?":X=162:Y=76:GOSUB 10000:Y=52:
GOSUB 10000
2140 A$="T=0":X=155:Y=65:GOSUB 10000
2145 GOSUB 7325:GOSUB 7375
2150 A$="R=00000000":GOSUB 7425
2160 A$="EINDE":X=148:Y=7:GOSUB 10000
2170 A$="S= S OR R":X=15:Y=125:GOSUB
10000
2200 A$="R=RESULTAAT":X=5:Y=240:GOSUB
10000
2210 A$="T=TELLER":Y=230:GOSUB 10000
2220 A$="S= S A R":Y=220:GOSUB 10000
2240 GOSUB 13000
2300 DW=RND(254.0)+1.0:GOSUB 21000:DW$=
X$:L3=LEN(STR$(DW))
2310 UX$=MID$(STR$(DW),1,L3-3):X=190:Y=
241:CC=22
2320 A$="Ux="+UX$+" mV":GOSUB 10000
2330 SAR=128:SOR1=0:DW=SAR:GOSUB 21000:
SOR1$=X$
2340 KL=22:GOSUB 7000:WAIT TIME 40:
GOSUB 7050
2400 T4=8:FF=1
2410 IF T4=8 THEN SOR=SAR
2420 DW=SOR:GOSUB 21000:SOR$=X$
2430 XB$=MID$(DW$,8-T4,1)
2440 KL=22:GOSUB 7100:CC=22:A$="S="+
SOR$:GOSUB 7115:UDAC$=STR$(SOR)
2450 L2=LEN(UDAC$):DAC$=MID$(UDAC$,1,
L2-3)
2500 X=210:Y=186:CC=22:A$="Udac="+DAC$+
" mV ":GOSUB 10000:GOSUB 13000
2510 KL=22:GOSUB 7150:IF XB$="1" THEN
CC=22:GOSUB 7175
2520 IF XB$="0" THEN CC=22:GOSUB 7125:
GOSUB 7300
2530 GOSUB 13000
2540 GOSUB 7250:XC$=MID$(SOR$,8-T4,1)
2550 IF XC$=XB$ THEN A$="R="+SOR$:CC=22:
GOSUB 7225:SOR1=SOR
2560 IF XC$<>XB$ THEN A$="R="+SOR1$:CC=

```

```

22:GOSUB 7225:SOR$=SOR1$:SOR=SOR1
2580 T4=T4-1:A$=MID$(STR$(T4),1,1):X=
183:Y=96:CC=22:GOSUB 10000
2590 IF T4=0 THEN 2900
2600 KL=22:CC=22:GOSUB 7350:GOSUB 7325
2610 GOSUB 13000
2620 GOSUB 7200
2630 SAR=SAR/2:DW=SAR:GOSUB 21000:SOR$=
X$
2700 X=15:Y=135:CC=22:A$="R="+SOR$:
GOSUB 10000
2710 Y=125:A$="S="+SOR$:GOSUB 10000
2720 DRAW 15,124 85,124 23
2730 SOR1$=SOR$:SOR=SAR IOR SOR:DW=SOR:
GOSUB 21000:SOR$=X$
2740 Y=112:CC=23:A$="S="+SOR$:GOSUB
10000:GOSUB 6770:GOSUB 13000
2850 KL=21:CC=23:GOSUB 7125:GOSUB 7175:
GOSUB 7325
2860 KL=21:GOSUB 6770:GOSUB 6540:GOSUB
6550
2870 GOTO 2410
2900 REM
2950 KL=22:CC=22:GOSUB 7350:GOSUB 7375:
GOSUB 7400
2970 A$="R="+DW$:GOSUB 7425:GOSUB 13000
2980 GOSUB 7450
2990 GOSUB 13000
3000 FF=0:RETURN
6500 REM STROOMSCHEMA
6510 GOSUB 7000:REM BLOK 1
6520 GOSUB 7050:REM BLOK 2
6530 GOSUB 7100:REM BLOK 3
6540 GOSUB 7150:RETURN:REM RUIT 1
6550 GOSUB 7200:REM BLOK 4A
6560 GOSUB 7250:REM BLOK 4B
6570 GOSUB 7300:REM BLOK 4C
6580 GOSUB 7350:REM RUIT 2
6590 GOSUB 7400:REM BLOK 5
6600 GOSUB 7450:RETURN:REM BLOK 6
6770 DRAW 50,150 50,210 KL:DRAW 50,210
165,210 KL
6900 RETURN
7000 XM=165:Y2=240:L=40:B=14:GOSUB 8000:
DRAW XM,240 XM,235 KL:RETURN
7050 XM=165:Y2=215:L=80:B=20:GOSUB 8000:
DRAW XM,215 XM,205 KL:RETURN
7100 XM=165:Y2=185:L=80:B=20:GOSUB 8000:
RETURN
7115 X=132:Y=186:GOSUB 10000:RETURN
7125 A$="NEE":X=200:Y=162:GOSUB 10000:
DRAW 195,160 280,160 KL
7130 DRAW 280,160 280,145 KL:RETURN
7150 DRAW 165,185 165,180 KL:XM=165:Y2=
160:L=60:GOSUB 8100:RETURN
7175 A$="JA":X=167:Y=130:GOSUB 10000:
DRAW 165,140 165,115 KL:RETURN
7200 XM=50:Y2=110:L=80:B=40:GOSUB 8000:

```

```

RETURN
7225 X=132:Y=105:GOSUB 10000:RETURN
7250 XM=165:Y2=95:L=80:B=20:GOSUB 8000:
RETURN
7300 XM=280:Y2=130:L=80:B=15:GOSUB 8000:
DRAW 280,130 280,120 KL
7310 DRAW 165,120 280,120 KL:DRAW 165,
120 165,115 KL:RETURN
7325 A$="NEE":X=115:Y=72:GOSUB 10000:
DRAW 50,70 135,70 KL
7330 DRAW 50,70 50,110 KL:RETURN
7350 DRAW 165,95 165,90 KL:XM=165:Y2=70:
L=60:GOSUB 8100:RETURN
7375 A$="JA":X=140:Y=47:GOSUB 10000:
DRAW 165,45 165,50 KL:RETURN
7400 XM=165:Y2=25:L=80:B=20:GOSUB 8000:
RETURN
7425 X=132:Y=30:GOSUB 10000:RETURN
7450 DRAW 165,20 165,25 KL:XM=165:Y2=5:
L=40:B=15:GOSUB 8000:RETURN

8000 REM RECHTHOEK
8010 X2=XM-L/2:REM L%=LENGTE B%=BREEDTE
8020 DRAW X2,Y2 X2+L,Y2 KL:DRAW X2,Y2
X2,Y2+B KL
8030 DRAW X2,Y2+B X2+L,Y2+B KL:DRAW X2+
L,Y2+B X2+L,Y2 KL
8040 RETURN

8100 REM RUIT:L%=LENGTE HOOGTE=2/3
LENGTE
8110 X2=XM-L/2:X3=X2+L/2:X4=X2+L:Y3=Y2+
L/3:Y4=Y2-L/3
8120 DRAW X2,Y2 X3,Y3 KL:DRAW X2,Y2 X3,
Y4 KL
8130 DRAW X3,Y3 X4,Y2 KL:DRAW X3,Y4 X4,
Y2 KL
8150 RETURN
8500 END
9000 MODE 0:PRINT CHR$(12):COLORT 0 14
0 0
9010 Y0=20:X0=200:KL=21
9220 MODE 6A:DF=5:X=120:Y=80:CC=23
9230 A$=CHR$(23):GOSUB 10000:DF=0:Y=160:
X=255:X$="01101101"
9240 FOR T=0 TO 7
9250 1 DRAW 211,Y 250,Y 21:A$=MID$(X$,T,
1 1):Y=Y-5:GOSUB 10000:Y=Y+5
9260 Y=Y-10:NEXT
9270 DRAW 20,150 119,150 23:DRAW 20,100
119,100 22
9280 DRAW 30,100 30,80 22:FILL 25,80 35,
85 22
9290 A$="ANALOGG/DIGITAAL":DF=1:X=30:Y=
180:CC=21:SP=7
9300 FOR T=1 TO 3
9310 1 GOSUB 10000:X=X+1:Y=Y+1:CC=CC+1
9320 NEXT
9325 GOSUB 15000
9330 A$="OMZETTER":DF=1:X=110:Y=20:ID=

```

```

10:FF=1:FC=1:CC=22:GOSUB 10000:FF=0
9340 A$="Ux":X=30:Y=155:DF=0:CC=23:
GOSUB 10000
9390 GOSUB 12000
9400 RETURN
9450 COLORT 0 14 0 0:PRINT "HET
PROGRAMMA VERVOLGT MET DE LAATSTE
INVOER"
9500 MODE 6A:GOSUB 15000
9510 X0=30:Y0=20:T=25
9520 DRAW X0-1,Y0-1 X0+8*T,Y0-1 23:DRAW
X0-1,Y0-1 X0-1,Y0+200 23
9530 X=X0+7*T:Y=Y0-15:CC=23:A$=" t":
GOSUB 10000
9540 VH(8.0)=Y0
9550 VH(7.0)=128
9560 IF BW(7.0)=1.0 AND BW(6.0)=1.0
THEN VH(7.0)=96
9570 X1=X0:KL=21
9580 FOR TELLER=7 TO 0 STEP -1
9590 1 IF BW(TELLER)=0.0 THEN KL=22:FILL X1,
1 VH(8.0) X1+T,VH(8.0)+VH(TELLER) KL:
1 VH(8.0)=VH(8.0):KL=21
9600 1 IF BW(TELLER)=1.0 THEN FILL X1,
1 VH(8.0) X1+T,VH(8.0)+VH(TELLER) KL:
1 VH(8.0)=VH(8.0)+VH(TELLER)
9610 1 IF TELLER>0.0 THEN VH(TELLER-1.0)=
1 VH(TELLER)/2.0
9620 1 X1=X1+T
9630 NEXT
9640 SP=7:X=X0-15:Y=VH(8.0)-10.0:CC=23:
A$="Ux":GOSUB 10000
9650 DRAW X0,VH(8.0) X0+8*T,VH(8.0) 23:
COLORG 0 10 1 14
9700 A$="SUCCESSIVE APROXIMATION"
9710 DF=0:SP=8:X=X0-1:Y=5:CC=22:FF=1:FC=
1:GOSUB 10000:FF=0
9720 X=X0+T/3:Y=VH(8.0)+5.0:CC=23:FC=0:
FF=1:SP=7
9730 IF VH(8.0)>180.0 THEN Y=
VH(8.0)-15.0
9740 FOR TELLER=7 TO 0 STEP -1
9750 1 A$=MID$(STR$(BW(TELLER)),1,1):
1 GOSUB 10000
9760 1 X=X+T
9770 NEXT
9780 FF=0:PRINT
9790 GOSUB 400
9890 GOSUB 12000
9900 RETURN

10000 REM SUBROUTINE FGT
10010 POKE #2F2,X MOD 256:POKE #2F3,
X/256:POKE #2F4,Y
10020 POKE #2F5,SP:POKE #2F6,ID
10030 C=FC*#40+CC:F=FF*#80+PF*#40+ZF*#20+
VF*#10+DF:POKE #2F0,C:POKE #2F1,F
10040 CALLM #300,A$
10050 RETURN
12000 IF W=1.0 THEN 12040

```

```

12020 CURSOR 0,0:PRINT "DRUK OP DE
SPATIEBALK OM VERDER TE GAAN..";
12030 B=BETC:IF B<>32 THEN 12030
12040 W=0:PRINT CHR$(12);
12050 RETURN
13000 A$="DRUK SPATIEBALK":X=5:Y=5:CC=23:
FC=0:FF=1:GOSUB 10000
13010 B=BETC:IF B<>32 THEN 13010
13020 A$=" " :X=5:Y=5:GOSUB
10000
13030 RETURN
15000 DRAW 260,185+Y1 270,204+Y1 23:DRAW
260,185+Y1 270,166+Y1 23
15010 DRAW 270,204+Y1 275,194+Y1 23:DRAW
270,166+Y1 275,176+Y1 23
15020 A$="CHR.MTS":X=270:Y=180+Y1:SP=8:
CC=22:DF=0:GOSUB 10000
15030 A$="ALMELO":X=280:Y=170+Y1:GOSUB
10000
15100 Y1=0:RETURN
20000 COLORT 0 14 0 0:H1=2
20010 PRINT CHR$(12);:CURSOR 0,3:PRINT
"DECIMALE INVOER VAN DE
INGANGSSPANNING Ux...."
20050 CURSOR 0,2:PRINT "Ux MOET EEN
POSITIEF GEHEEL GETAL ZIJN.
0<Ux<255 mV.";
20110 INPUT G$
20120 IF G$="" THEN 20110
20140 IF LEFT$(G$,1)="-" THEN 20110
20150 IF ASC(G$)>57.0 THEN 20110
20160 IF VAL(G$)<0.0 OR VAL(G$)>255.0
THEN 20110
20180 PRINT " mV.":DW=VAL(G$)
20190 GOSUB 21000
20350 PRINT CHR$(12);"DE DECIMALE WAARDE
BEDRAAGT: ";DW;" mV."
20360 PRINT "DE BINAIRE WAARDE VAN ";DW;
" mV = ";X$;" = #";HEX$(DW)
20380 FOR TELLER=0 TO 7
20400 1 BW(TELLER)=VAL(MID$(X$,7-TELLER,
1 1))
20420 NEXT
20450 GOSUB 9500
20600 RETURN
21000 M=VARPTR(DW)
21200 P=PEEK(M+3):X$=""
21240 FOR T=7 TO 0 STEP -1
21260 1 B=P IAND (2^(T+1))
21280 1 K=B SHR T:ST$="0":IF K=1.0 THEN
1 ST$="1"
21300 1 X$=X$+ST$
21340 NEXT
21350 RETURN

25000 REM SCHEMA A/D-OMZETTER
25010 MODE 6A:COLORT 0 14 0 0
25020 A$="SUCCESSIVE APPROXIMATION
METHODE":X=20:Y=180:DF=0:CC=23:SP=
7:GOSUB 10000:GOSUB 15000

```

```

25030 A$=CHR$(24):X=100:Y=100:DF=3:CC=21:
GOSUB 10000
25040 FILL 10,20 60,160 22:A$="S A R":FF=
1:FC=0:VF=1:X=50:Y=30:DF=2:CC=23:
GOSUB 10000
25050 FF=0:VF=0
25060 X=100:Y=50:DF=1:CC=22:A$=CHR$(13):
GOSUB 10000:DRAW 61,65 99,65 23
25090 A$=CHR$(20):X=200:Y=60:DF=3:ZF=1:
CC=22:GOSUB 10000:ZF=0:DF=0
25100 DRAW 61,32 160,32 23
25110 Y=145
25120 FOR T=1 TO 8
25130 1 DRAW 61,Y 99,Y 23:Y=Y-5
25140 NEXT
25150 DRAW 160,130 220,130 23
25160 DRAW 220,130 220,52 23
25170 DRAW 200,52 220,52 23
25180 DRAW 200,12 250,12 23
25190 A$="Ux":X=260:Y=10:CC=21:GOSUB
10000
25200 A$="COMPARATOR":SP=7:X=150:Y=65:CC=
22:GOSUB 10000:A$="+":X=185:Y=45:
CC=21:GOSUB 10000
25300 PRINT CHR$(12);:PRINT "SCHEMA
A/D-CONVERTOR."
25310 PRINT "SAR = Succesive
Approximation Register."
25320 PRINT "De SAR-FUNCTIE kan ook
software-matig worden
gerealiseerd."
25330 GOSUB 12000
29000 RETURN
30000 SPI=0:COLORT 0 14 0 0:REM
SUBROUTINE DAI-RWC-I/O
30010 OUT #A3,#91:REM INIT POORT #A0=
INPUT EN #A1=OUTPUT
30020 A=0:B=#80:C=#0:D=8:H1=3
30080 A=A IOR B
30090 C=A
30100 OUT #A1,A
30110 I=INP(#A0) IAND #80
30120 IF I=0 THEN 30200
30130 A=B
30140 A=(-A):A1=VARPTR(A):A=PEEK(A1+3)-1
30150 A=A IAND C
30160 C=A
30200 A=B
30210 A=A SHR 1
30220 B=A
30230 A=C
30240 D=D-1
30250 IF D<>0.0 THEN 30080
30255 A=A/100.0
30260 IF SPI=1.0 THEN POKE #75,32:GOTO
30285
30265 CURSOR 0,3:PRINT "INVOER VIA
DAI-RWC MET A/D-INTERFACE."
30270 PRINT "DE SPANNING Ux BEDRAAGT: ";
A!;" V."

```

cont. on p. 379

zie tekening p.333

## DAI RESET

Reeds eerder is er geschreven over de storgevoeligheid van het reset circuit van de DAI.

De ene DAI reageert sneller op storingen van buitenaf dan de andere.

Mij is dan ook al verschillende malen de vraag gesteld: "als mijn koelkast aankomt slaat mijn DAI op tilt, wat moet ik daar aan doen".

Bij gebruikers van de Commodore 1541 Floppy (inmiddels een aardig aantal) komt deze klacht wat vaker voor. Dit zit echter niet in het door ons ontwikkelde systeem maar vermoedelijk in de totale lengte van de gebruikte kabels.

Het reset signaal gaat n.l. vanaf de DCE Bus via de flatkabel naar de interface en vandaar via de afgeschermd floppy kabel naar de drive.

Hoewel de 1541 voorzien is van een netfilter en kan worden aangesloten op een kontaktdoos voorzien van een randaarde is een verhoogde storgevoeligheid met alle nare gevolgen vanden niet uit te sluiten.

Een oplossing is misschien kortere kabels, gebruik maken van een andere geaarde kontaktdoos en als dat niet helpt misschien een kleine verandering in Uw DAI.

Deze verandering bestaat uit het aanbrengen van een condensator van 1 uF/35V over de "RESET" naar aarde. De condensator is zo'n kleine rode of blauwe tantaalcondensator zoals er zovelen in de DAI zitten.

Waar moet hij komen, nu dat is eenvoudig.

Open Uw DAI (alle kabels eraf natuurlijk en het kabeltje van het metaal rondom het toetsenbord los gemaakt).

U ziet dan recht van het toetsenbord vlak bij de BREAK toets een printbaantje van onder het toetsenbord met een boog omhoog gaan en stoppen bij een boorgat.

Hierin komt de + zijde van de condensator. Links van dit boorgat komt een dikkere printbaan van een blokje (bij mij is het blauw) dit is een weerstandsblokje boven het toetsenbord en stopt ook in een boorgat links van de eerste. Hierin komt de - zijde van de condensator. U moet NIET het boorgat hebben wat erboven zit of de printbaan aan de rand van de DAI want dat is +5volt. De tekening helpt U verder.

Deze wijziging is ook uitgevoerd vanaf rev. 7.1 door INDATA.

Ik hoop dat hiermede het probleem is opgelost.

H. Rison

## RGB MONITOR

zie tekening p.333

Werkt U ook al jaren met een TV aan Uw DAI en bent U dan ook wel eens jaloers op die mensen die zo'n mooi plaatje op hun RGB monitor hebben.

Nu de RGB monitoren met een medium resolutie goedkoper worden is de aanschaf het overwegen waard.

Maar het is wel oppassen met het geen U koopt.

Wat meestal niet bekend is is dat er twee soorten RGB monitors bestaan n.l. met een analoge en een digitale RGB ingang. De laatste worden vaak IBM compatible genoemd.

Het verschil tussen beide is het volgende:

1. Een analoge RGB monitor ontvangt een drietal analoge signalen welke variëren in spanning. Hiermede zijn een groot aantal kleuren variaties mogelijk afhankelijk van de amplitude van de aangeboden signalen.

2. Een digitale RGB monitor ontvangt een drietal digitale signalen van TTL niveau wat wil zeggen of +5 volt of 0 volt.

Hierdoor is het aantal kleuren beperkt tot het aantal data bits waarover de processor beschikt bij de DAI maximaal 8. De DAI kan normaal 16 kleuren leveren en een digitale monitor is dan ook niet direct geschikt voor de DAI.

Tevens moeten de meeste digitale RGB monitoren 2 sync. signalen ontvangen n.l. horizontale en verticale sync. De analoge monitor heeft genoeg aan composite sync. dit is een combinatie van beiden.

De DAI RGB interface kaart is van het analoge type en levert behalve analoge RGB een composite sync. signaal en een audio signaal.

Wil men dus een RGB monitor aansluiten dan moet men dus een DAI RGB interface kaart kopen en deze op de plaats van de normale Video kaart monteren tenzij men een soldeerbout pakt en zelf een RGB kaart bouwt.

Voor de zelfbouwers volgt dan ook een schema met ROM inhoud voor een RGB kaart.

De gegevens in de ROM zijn niet gelijk aan de originele ROM maar geven een betere kleuren weergave.

De werking is als volgt:

De signalen K0, K1, K2, en K3 worden gebruikt voor de kleuren generatie. Met deze 4 bits zijn 16 adres combinaties mogelijk.

Deze 16 combinaties zijn geprogrammeerd in een ROM van het type N82S123 en staan als 8 bits aan de uitgang ter beschikking als het signaal VLB welke als een enable werkt laag gaat.

Deze 8 bits worden dan in z.g. 'D' Flipflops geklokt met een het signaal CKL.

De Q uitgangen worden in groepen van 3 via verschillende weerstanden aan elkaar geknoopt in een z.g. summing network.



Deze wordt dan aan de basis van een transistor toegevoerd die dan een spanning afgeeft tussen +5 volt en -5 volt afhankelijk van de spanning aan de basis. Deze drie groepen vormen dan de Rood, Groen en Blauw uitgang naar de monitor. Bezit men echter een digitale RGB monitor dan is deze oplossing niet mogelijk.

Inhoud Prom type 82S123

	D7	D6	D5	D4	D3	D2	D1	D0	HEX	ADR
Black	0	0	0	0	0	0	0	0	00	00
Dark Blue	1	1	0	0	0	0	0	0	C1	01
Purple Red	1	0	0	0	0	1	1	1	87	02
Red	0	0	0	0	0	1	1	1	07	03
Purple Red	1	0	1	0	0	1	1	1	A7	04
Emerald Green	0	0	1	1	1	0	0	0	38	05
Khaki Brown	0	1	1	0	1	1	1	1	6F	06
Mustard Brown	0	0	1	0	0	1	0	1	25	07
Gray	1	0	1	0	1	1	0	1	AD	08
Middle Blue	1	1	1	0	0	1	0	0	E4	09
Orange	0	0	1	0	0	1	1	1	27	0A
Pink	1	1	1	0	1	1	1	1	EF	0B
Light Blue	1	1	1	1	0	0	0	0	F0	0C
Light Green	1	0	1	1	1	0	0	0	BB	0D
Light Yellow	0	0	1	1	1	1	1	1	3F	0E
White	1	1	1	1	1	1	1	1	FF	0F

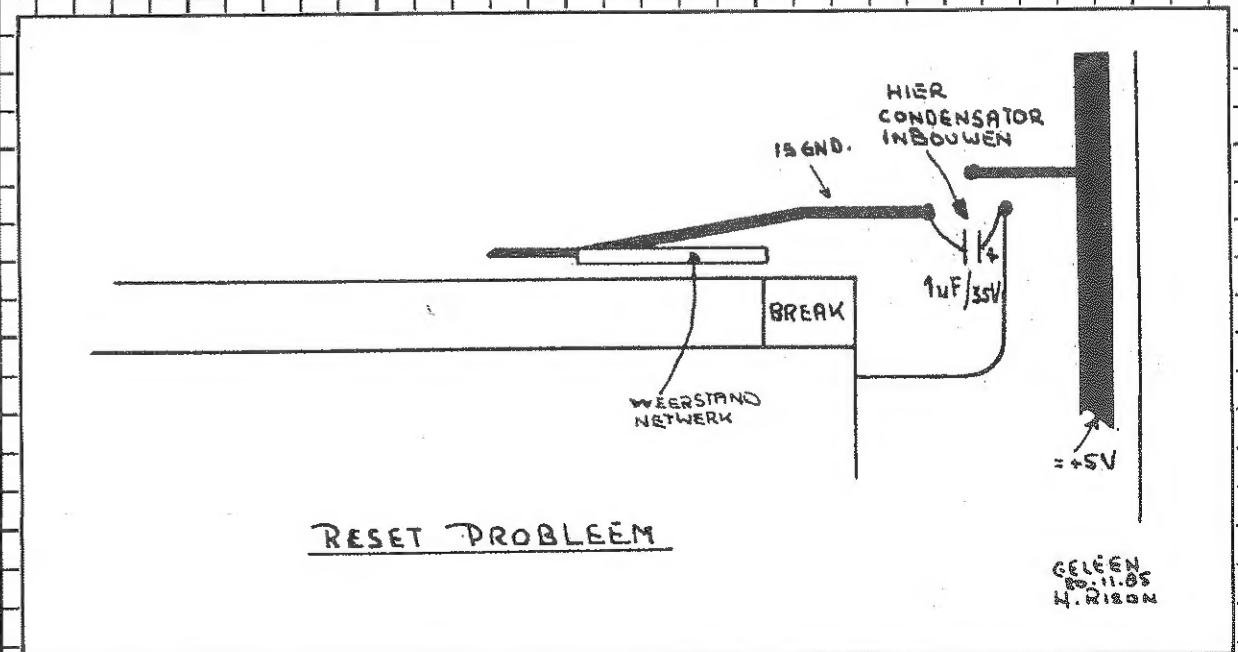
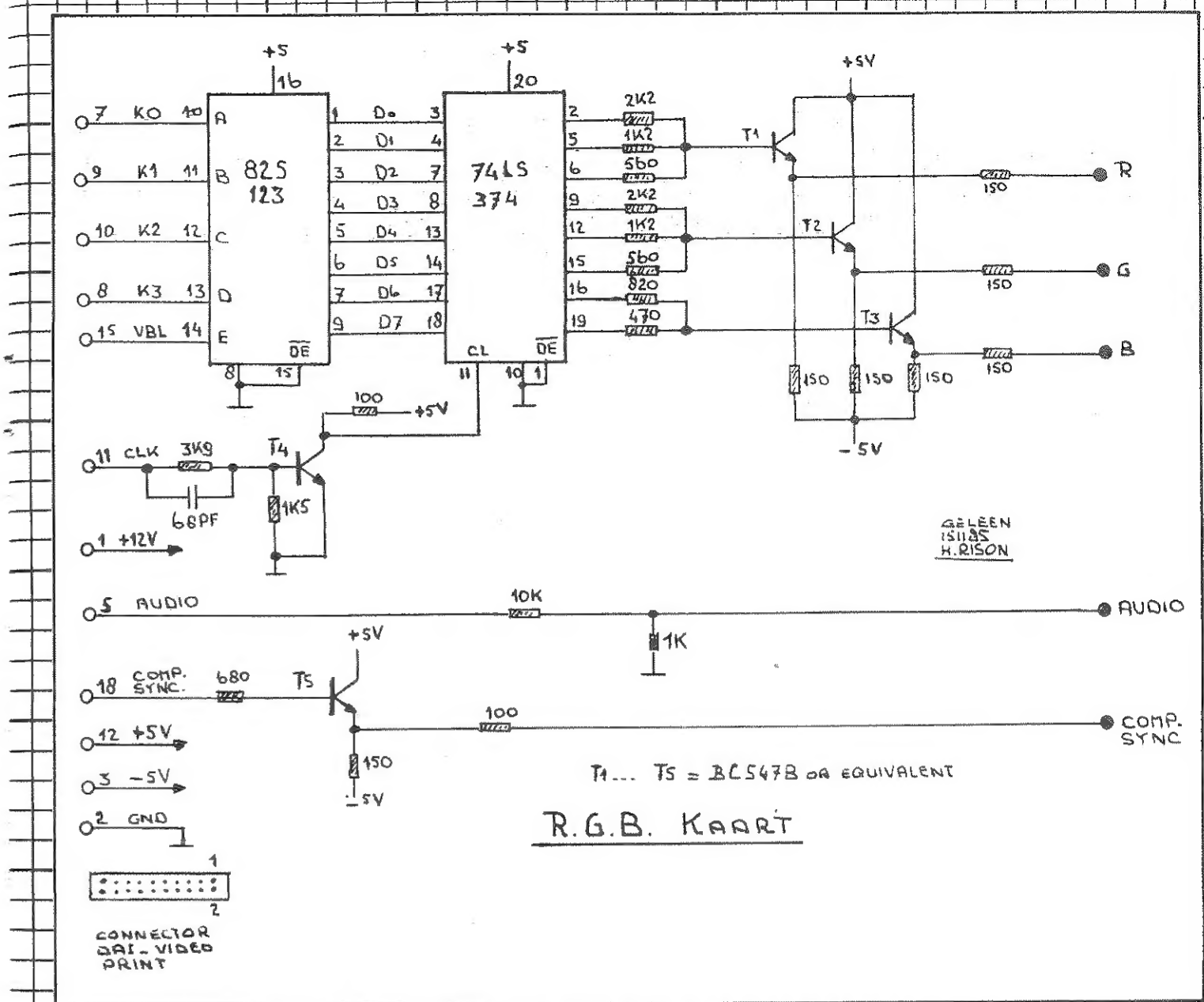
Als een 82S123 eenmaal geprogrammeerd is en de kleuren staan U niet aan dan kan hij helaas niet meer veranderd worden en moet U een nieuwe nemen. Een alternatief is om een 2716 EPROM te monteren en hiermee te experimenteren totdat U de kleuren die U wenst heeft gevonden. Deze kan dan inplaats van de 82S123 gebruikt worden. Het audio signaal kunt U direkt afnemen van de connector. Het composite sync. signaal wordt via een ermittervolger aan de uitgang toegevoerd. De signalen zijn terug te vinden op de volgende pin nrs. van de connector naar de video print:

- Pin 2 = Gnd.
- Pin 3 = -5V.
- Pin 5 = Audio.
- Pin 7 = K0.
- Pin 8 = K3.
- Pin 9 = K1.
- Pin 10 = K2.
- Pin 11 = CLK.
- Pin 12 = +5V.
- Pin 15 = VLB.
- Pin 18 = Comp.Sync.

De pin nummering begint rechts boven (gezien vanaf het keyboard) met de oneven nrs. aan de bovenzijde en de even nrs. aan de ondezijde.

De mogelijkheid om een digitale monitor aan te sluiten op de DAI wil ik in een volgende uitgave behandelen. Tevens zal ik dan een vervangingschema voor een 2716 EPROM hierin op nemen.

H. Rison



## CENTRONICS PRINTERS

PRINTERS WITH PARALLEL INTERFACES ARE BECOMING MORE COMMON AND ARE NOW OFTEN CHEAPER THAN THOSE WITH SERIAL (RS232) INTERFACES. IN MANY PRINTERS THE SERIAL INTERFACE IS AN (EXPENSIVE) ADD-ON BOARD. AS MOST, IF NOT ALL, PROGRAMS AVAILABLE FROM DAINAMIC ARE WRITTEN FOR SERIAL (RS232) PRINTERS IT IS NECESSARY TO MODIFY THEM FOR USE WITH A PARALLEL PRINTER. THIS ARTICLE OUTLINES METHODS THAT I HAVE USED.

MOST OF THE PARALLEL INTERFACES CONFORM EITHER TO THE FULL CENTRONICS OR A REDUCED CENTRONICS STANDARD. IT IS THEREFORE NECESSARY TO UNDERSTAND HOW A CENTRONICS INTERFACE WORKS. THE USUAL REDUCED INTERFACE CONSISTS OF CONNECTIONS FOR 8 DATA LINES, AND SOME 'HAND-SHAKING' LINES. THESE 'HAND-SHAKING' LINES ARE: A STROBE (STB), AN ACKNOWLEDGE (ACK), AND SOMETIMES A BUSY SIGNAL. ON A FULL CENTRONICS INTERFACE OTHER LINES, FOR EXAMPLE: PAPER OUT, PRINTER SELECT AND FAULT, ARE AVAILABLE.

TO PRINT DATA THE SEQUENCE IS AS FOLLOWS: - THE COMPUTER PLACES A BYTE OF ASCII DATA ON THE 8 DATA LINES AND KEEPS IT THERE. A LOW PULSE (STROBE) IS THEN SENT TO THE PRINTER TO TELL IT THERE IS VALID DATA AWAITING ATTENTION. THE PRINTER ACCEPTS THE DATA AND RESPONDS WITH A SHORT (LOW) ACKNOWLEDGE PULSE. THE PRINTER IS NOW READY FOR THE NEXT BYTE OF DATA. IF THE PRINTER IS PROVIDED WITH A BUSY LINE, THIS IS KEPT LOW WHILE THE PRINTER CAN RECEIVE DATA AND HIGH WHILE THE PRINTER IS BUSY. FOR EXAMPLE THE PRINTER IS BUSY DURING A CARRIAGE RETURN OR WHEN THE PAPER RUNS OUT. IF THERE IS A BUSY LINE AVAILABLE IT IS USUALLY EASIER TO USE THIS RATHER THAN THE ACKNOWLEDGE LINE AS THE SHORT ACK PULSE CAN EASILY BE MISSED BY THE PROGRAM. ONE WAY OF ALWAYS CATCHING ACK IS TO INCLUDE A LATCH IN THE INTERFACE. A CIRCUIT FOR THIS IS INCLUDED LATER. USING THE BUSY SIGNAL MAY NOT BE WITHOUT PROBLEMS BECAUSE IT IS NOT GENERATED AFTER EACH CHARACTER RECEIVED. THE COMPUTER MUST THEREFORE NOT SEND CCHARACTERS TOO FAST FOR THE PRINTER. THIS IS NOT NORMALLY A PROBLEM.

TO SUMMARISE: - THE SEQUENCE OF EVENTS IS AS FOLLOWS

USING THE BUSY SIGNAL.	USING A LATCHED ACK SIGNAL
CHECK IF PRINTER BUSY,	RESET LATCH
YES - CHECK AGAIN	SEND DATA AND STROBE SIGNAL
NO - SEND DATA	CHECK IF LATCH SET
SEND STROBE SIGNAL	NO - CHECK AGAIN
START AGAIN	YES - START AGAIN

IN PRACTICE IT IS USUALLY NECESSARY TO CHECK THE DATA GOING TO THE PRINTER SO THAT AFTER A CARRIAGE RETURN, A LINE FEED CAN BE INSERTED. ALSO IF THE PRINTER DOESN'T RECOGNISE THE FORMFEED COMMAND (CHR#12) IT IS NECESSARY TO KEEP A COUNT OF THE LINES AS THEY ARE PRINTED SO THAT CHR\$(12) CAN BE CONVERTED TO THE CORRECT NUMBER OF LINEFEEDS. IN MY CASE, AS MY PRINTER ONLY RECOGNISES CAPITAL LETTERS, I ALSO HAVE TO CATCH SMALL LETTERS AND CONVERT THEM TO CAPITALS.

THE EASIEST WAY TO CONNECT THE DAI TO A PARALLEL PRINTER IS VIA THE DCE BUS. THE DCE BUS CONNECTIONS USED ARE GIVEN AT THE START OF BOTH

OF THE SAMPLE PROGRAMS AT THE END OF THIS ARTICLE. THE ACTUAL CONNECTIONS IN THE CENTRONICS V6.5 PROGRAM WERE USED IN ORDER TO BE COMPATABLE WITH THE PARALLEL PRINTER ROUTINE GIVEN IN IMPLM (SPL SOURCE CODE) IN LINES 320 TO 353. THE CENTRONICS V6.5 PROGRAM IS STORED IN SOME BATTERY BACKED UP RAM (£F000 - £F7FF) WHICH I HAVE ADDED TO THE DAI AND STARTS AT ADDRESS £F000. AS PRESENTED, THE OTHER PROGRAM IS STORED IN THE ENVELOPE BUFFER FROM £01F0 TO £0260. NOTE THAT THIS AREA IS OVERWRITTEN FROM BASIC BY THE ENVELOPE STATEMENT. WHEN USING BASIC, MACHINE CODE PROGRAMS STORED AT THE START OF USER RAM BELOW THE HEAP (WHICH NORMALLY STARTS AT £02EC) ARE ONLY SAFE IF THE HEAP POINTER WHICH IS STORED AT ADDRESSES £029B AND £029C IS CHANGED TO POINT AFTER THE PROGRAM.

NOTE: - EXAMPLES BELOW USE THE CENTRONICS V6.5 PROGRAM LISTED AT THE END OF THIS ARTICLE. ALSO NOTE THAT THE PROGRAMS ARE ENTERED BY WAY OF POINTERS. THIS SAVES TROUBLE EACH TIME THE PRINT PROGRAM IS UPDATED. THESE PROGRAMS ARE WRITTEN AS SMALL FUNCTIONAL ROUTINES ALLOWING CHANGES TO BE MADE EASILY AND, HOPEFULLY, ALLOWING THEM TO BE UNDERSTOOD EASILY.

## BASIC

THE EASIEST WAY TO USE THE PRINTER FROM BASIC IS VIA THE DOUTC VECTOR. FIRST A JUMP INSTRUCTION (C3) AND THE ADDRESS OF THE PRINT PROGRAM ARE STORED IN DOUTC (DOUTC STARTS AT £02DD AND IS 3 BYTES LONG). NOTE THE ADDRESS IS STORED LOW BYTE FIRST, THUS C3 90 F0 MEANS JUMP F090. ONCE THIS IS DONE POKE £131,3 ENABLES THE PRINTER AND POKE £131,1 STOPS IT. I PREFER TO USE A SHORT MACHINE CODE ROUTINE TO SET UP DOUTC AND DO OTHER INITIALISATIONS FOR ME (CALLM £F000). IN BOTH THE SAMPLE PROGRAMS BELOW THIS ROUTINE IS CALLED PRINT.

## PRINT EDITOR BUFFER

INCLUDED IN THE PRINT PROGRAM IS A ROUTINE (START £F004) USED TO PRINT THE EDIT BUFFER. THIS IS USEFUL FOR PRINTING SHORT NOTES WITHOUT LOADING THE WORDPROCESSOR. TO USE FROM BASIC: - TYPE EDIT AND WRITE THE TEXT IN THE EDIT BUFFER. EXIT BY HITTING BREAK TWICE. PRINT IT USING CALLM £4000. RE-ENTER THE EDITOR USING CALLM £E1FB.

## SPL

THE EXAMPLE BELOW IS JUST ONE WAY OF INTERFACING A CENTRONICS PRINTER WITH SPL. CHANGE IMPLM AS FOLLOWS (EXAMPLE FOR CENTRONICS V6.5)

```
79  PRINTR SET    PARALL      ;PRINTER TYPE
323  IF          PRINTR=PARALL
324  ORG          SPL+20E0H
325  DW          PRINIT
326  ORG          SPL+248FH
327  DW          PRINT
328  ORG          PRTFNC
329  PRINT JMP    OF008H      ;JMP OWN PRINT PROG
330  ORG          SPL+2EA3H    ;OLD PRINT ORG
```

PASCAL

```

341 PRINT: LXI H PRIMER ; READY ?
342 CALL STROUT ; OUTPUT TO SCREEN
343 CALL WAITEM ; WAIT FOR [ ] OR
344 ; [BREAK]
345 CALL OF000H ; OWN INIT PROG
346 PUSH F-W
347 MVI A 1H ; TURN OFF DOUTC
348 STA 131H
349 POP PSW
340 RET
341 ENDIF
342 IF PRINTER=SERIAL
343

```

FWP

ON PAGE 8 OF THE ENGLISH FWP MANUAL IT SAYS THAT THE SPACE FROM 03B6H TO 03E9H HAS BEEN KEPT FREE FOR A PARALLEL PRINTER PROGRAM. THERE CAN BE SOME PROBLEMS USING THIS BECAUSE FWP SETS THE 8255 PPI CONTROL REGISTER IN A WAY THAT MAY NOT BE COMPATIBLE WITH SOME PRINT PROGRAMS. ALSO RECENT ISSUES OF DYNAMIC GIVE PATCHES WHICH USE THIS AREA. I HAVE FOUND THAT SENDING CONTROL DATA TO THE 8255 PPI BEFORE EACH BYTE TO BE PRINTED CAUSES MY PRINTER PROBLEMS (I DO NOT UNDERSTAND WHY) AND THIS METHOD CANNOT INITIALISE LINE COUNTS ETC WHERE THESE ARE NECESSARY. ONE SOLUTION IS TO BREAK INTO FWP BEFORE IT SENDS THE "PRINTER READY ? ETC." MESSAGE TO THE SCREEN, AND DIVERT THE PROGRAM TO AN INITIALISATION PROGRAM (SEE FWPI) WHICH ENDS BY PRINTING THE MESSAGE AND RETURNING TO FWP.

THE RELEVANT PARTS OF FWP ARE LISTED BELOW.

STANDARD FWP

```

071B 21722A LXI H 2A72H ; START ADDRESS OF
; PRINTER MESSAGE
071E CDD4DA CALL ODAD4H ; DISPLAY IT
;
03B6 CD94DD CALL ODD94H ; RS232 OUT ROUTINE
03B9 C9 RET ; RETURN

```

NEW FWP

```

071B 21722A LXI H 2A72H ; START ADDRESS OF
; PRINTER MESSAGE
071E CD0CF0 CALL F00C ; CALL OWN INIT.
; ROUTINE (FWPI)
;
03B6 CD08F0 CALL OF008H ; OWN PRINT ROUTINE
03B9 C9 RET ; RETURN

```

THE PASCAL SOURCE CODE IS STORED IN THE EDIT BUFFER (STARTING AT) 2000H. THE STANDARD PRINT ROUTINE IS AS FOLLOWS :-

```

3B4D AF XRA A ; ZERO IN A
3B4E 323101 STA 131H ; RS232 ON
3B51 CDFFDA CALL OD AFFH ; PRINT MESSAGE POINTED TO
3B54 ; BY NEXT TWO BYTES
3B54 0020 DB 0H, 20H ; ADDR START EDIT BUFFER
3B56 3E01 MVI A 1H ; PRINTER OFF
3B58 323101 STA 131H
3B5B CDDAD6 CALL OD6DAH ; WAIT FOR SPACE BAR
3B5E C36530 JMP 3065H ; CONTINUE PROGRAM
3B61 00 NOP
3B62 00 NOP
3B63 00 NOP
3B64 00 NOP
3B65 END

```

BELOW ARE TWO SIMPLE WAYS TO ALTER THIS ROUTINE TO DRIVE A CENTRONICS PRINTER. THE FIRST WAY IS TO REPLACE THE XRA A AND STA 131H AT THE START OF THE ROUTINE AS FOLLOWS :-

```

3B4D CD00F0 CALL OF000H ; CALL OWN INIT. ROUTINE
3B50 00 NOP ; WITH OUTPUT VIA DOUTC

```

THE SECOND REPLACES THE ENTIRE ROUTINE AS FOLLOWS :-

```

3B4D CD04F0 CALL OF004H ; PRINT EDIT BUFFER
3B50 C36530 JMP 3065H ; CONTINUE PROGRAM

```

IN THE SECOND WAY THERE IS NO "WAIT FOR SPACE BAR" BEFORE DISPLAYING THE MENU. THIS CAN BE ADDED BEFORE JMP 3065H (OR REMOVED FROM THE FIRST WAY) IF REQUIRED.

FOR THOSE WITH A PRINTER WITH AN RS232 INTERFACE BUT USING A BAUD RATE SLOWER THAN 9600 THE FOLLOWING PATCH CAN BE USED. -

```

3B4D 3E XX 32
3B50 05 FF AF 32 31 01 CD FF DA 00 20 3E 01 32 31 01
3B60 C3 65 30 00 00

```

WHERE XX IS THE BAUD RATE (88 IS 300 BAUD)

CENTRONICS CAPITALS V6.5.

HOPEFULLY THIS PROGRAM IS UNDERSTANDABLE. FROM THE REMARKS ONE POINT WORTH NOTE IS THE WAY THAT THE LINE COUNT WORKS. THIS IS STORED AFTER RESET VECTOR 0 IN ADDRESSES 06 AND 07 (THIS DOES WORK WITH DBASIC). THE LINE COUNT IS DECREMENTED BY THE BUSY SIGNAL. THIS AUTOMATICALLY



```

1      TITL      '--- DAI CENTRONICS EMULATE & STORE ---'
2      ;
3      ; WRITTEN BY  GEORGE CATHCART      SEPTEMBER 1985
4      ;              12 EVORA PARK
5      ;              HOWTH, CO.DUBLIN
6      ;              IRELAND
7      ;              TEL. 01-324030
8      ;
9      ; PURPOSE   TO TRANSFER DATA IN ASCII FROM ANY MICRO
10     ;           WITH A CENTRONICS INTERFACE
11     ;
12     ; SIZE      72H BYTES (FROM 300H TO 371H)
13     ;
14     ; DCE BUS
15     ; CONNECTIONS DATA . . . . . PORT A BITS 0-7
16     ;              STROBE (FROM OTHER MICRO). PORT B BIT 7
17     ;              ACKNOWLEDGE (FROM DAI) . . PORT C BITS 4-7
18     ;              GROUND (0V). . . . . PIN 4
19     ;
20     ; TO START - 0300
21     ; TO EXIT  - SEND CHR$(4); TO DAI (ASCII 4 WITH NO CR.)
22     ; ON EXIT  - RECEIVED ASCII CODE IN EDIT BUFFER STARTING
23     ;           AT 1000H (MAX. SIZE 4000H). START OF HEAP
24     ;           AT 5000H
25     ;
26     START EQU 300H
27     ORG   START
28     ;
29     PUSH PSW
30     PUSH H
31     PUSH B
32     PUSH D
33     LXI H 1000H      ;START TEXT STORE AT #1000
34     SHLD 0A2H      ;SET POINTER TO START EDIT BUFFER
35     XCHG
36     LXI H 5000H
37     SHLD 29BH      ;SET START OF HEAP TO #5000
38     DCX H
39     SHLD 0A6H      ;SET POINTER TO END EDIT BUFFER
40     MVI A 0CH
41     RST 5
42     DB    3H
43     ;
44     PUT   "b"       ;8255 PPI
45     MVI A 10010010B ;MODE 0 A-IN,B-IN,CH-OUT,
46     PUT   "h"       ;CL-IN
47     STA  0FE03H
48     MVI A 0FFH
49     STA  0FE02H      ;SET ACK HIGH
50     LXI H 0FE01H
51     DI
52     CALL SCAN        ;SCAN INPUT
53     MVI A 0H
54     XCHG
55     MOV M,A          ;STORE 0H AS EOT IN
56     SHLD 0A4H        ;EDIT BUFFER AND STORE
57     ;               POINTER
58     CALL ACK

```

```

59     POP D
60     POP B
61     POP H
62     POP PSW
63     EI
64     RET
65     ;
66     ; SCAN
67     MOV A,M
68     ANI 80H
69     JNZ SCAN
70     LDA 0FE00H
71     ANI 7FH
72     CPI 2H
73     JZ   IGNORE
74     CPI 0AH
75     JZ   IGNORE
76     CPI 7FH
77     CZ   DEL
78     CPI 4H
79     RST 5
80     DB    3H
81     DI
82     XCHG
83     MOV M,A
84     INX H
85     XCHG
86     IGNORE CALL ACK
87     JMP   SCAN
88     ;
89     ; DEL
90     MVI A 1DH
91     RET
92     ;
93     ; ACK
94     MVI A 0H
95     STA 0FE02H
96     MVI A 0FFH
97     STA 0FE02H
98     RET
99     END
100

```

--- DAI CENTRONICS EMULATE & STORE ---

;CHECK STROBE  
 ;SET ?  
 ;NO ! CHECK AGAIN  
 ;YES ! READ DATA  
 ;DELETE PARITY IF PRESENT  
 ;IGNORE ASCII 2  
 ;(SENT BY ORIC BEFORE +NUMBERS.)  
 ;IGNORE LINEFEED  
  
 ;CHANGE 'DELETE' TO  
 ;6 HORIZONTAL LINES  
 ;IS IT END OF TRANSMISSION ?  
 ;YES ! THEN RETURN  
 ;NO ! THEN SEND TO SCREEN

;AND STORE

;REMOVE DELETE AND  
 ;HIGHLIGHT FOR SCREEN

;SEND ACK TO OTHER MICRO  
 ;AND RESET IT



```

0000          TITL      '*** CENTRONICS ACK LATCHED V1.6 ***'
0000          ;
0000          ;
0000          ; WRITTEN BY GEORGE CATHCART 16/10/85
0000          ; TO RUN AN ORIC MCP 40 PRINTER/PLOTTER
0000          ;
0000          ; THIS OUTPUTS DATA TO THE SCREEN AND A CENTRONICS
0000          ; PRINTER VIA THE DCE BUS AND USES DOUTC.
0000          ;
0000          ; IT IS STORED IN THE ENVELOPE BUFFER, STARTING AT 1F0H.
0000          ; DATA GOES TO PORT A, STROBE (HIGH) TO PORT B BIT 7
0000          ; RESET LATCH (HIGH) TO PORT B BIT 6 AND THE ACK SIGNAL
0000          ; IS RECEIVED ON PORT C BIT 4.
0000          ;
0000          ; TO INITILISE CALLM #1F0. TO STOP PRINTING POKE #131,1
0000          ;
0000          ; FROM FWP CALL 1FCH AND THEN CALL 1F8H
0000          ;
0000          ;----- SET UP POINTERS -----
0000          ;
0000 @=01F0 START EQU 1F0H
0000          ORG START ;ALLOWS PROG TO BE MOVED
01F0 C3FF01 JMP PRINT ;WITHOUT CHANGING ENTRY POINT
01F3          ORG START+8H
01F8 C33402 JMP PRINT1 ;VECTOR TO PRINTER ONLY PROG
01FB          ORG START+0CH
01FC C30902 JMP FWPIT ;VECTOR TO FWP PRINT INIT
01FF          ;
01FF          ;----- INITILISE GENERAL PRINT PROGRAM -----
01FF          ;
01FF CD2202 PRINT CALL PPI ;INITILISE 8255 PPI (DCE BUS)
0202 CD1202 CALL DOUTC ;SET UP DOUTC POINTERS
0205 CD2A02 CALL OUTPUT ;OUTPUT VIA DOUTC
0208 C9 RET
0209          ;
0209          ;----- INITILISE FWP (ETC) PRINT PROGRAM -----
0209          ;
0209 E5 FWPIT PUSH H ;STORE START ADDR OF MESSAGE
020A CD2202 CALL PPI ;INITILISE 8255 PPI
020D E1 POP H ;GET START MESSAGE ADDRESS
020E CDD4DA CALL 0DAD4H ;PRINT MESSAGE
0211 C9 RET
0212          ;
0212          ;----- INITILISE SUBROUTINES -----
0212          ;
0212          ;
0212 E5 DOUTC PUSH H
0213 F5 PUSH PSW
0214 3EC3 MVI A 0C3H ;STORE JMP
0216 32DD02 STA 2DDH ;TO
0219 213202 LXI H PRINT ;PRINT PROG START ADDRESS
021C 22DE02 SHLD 2DEH ;IN DOUTC
021F F1 POP PSW
0220 E1 POP H
0221 C9 RET
0222          ;
0222 F5 PPI PUSH PSW
0223 3E88 MVI A 98H ;INITILISE 8255 - MODE 0
0225 3203FE STA 0FE03H ; A,B & CL = OUT. CH = IN

```

```

0228 F1 POP PSW
0229 C9 RET
022A          ;-----
022A F5 OUTPUT PUSH PSW
022B 3E03 MVI A 3H
022D 323101 STA 131H ;OUTPUT VIA DOUTC
0230 F1 POP PSW
0231 C9 RET
0232          ;
0232          ;----- PRINT PROGRAM -----
0232          ;
0232          ; PRINT - SENDS DATA TO SCREEN AND PRINTER
0232          ; PRINT1 - SENDS DATA TO PRINTER ONLY
0232          ;
0232          ; ENTRY DATA TO BE SENT IN A
0232          ;
0232 EF PRINT RST 5
0233 03 DB 3H ;DISPLAY ON SCREEN
0234 FE1F PRINT1 CPI 1FH ;IS IT A CONTROL?
0236 DC3D02 CC CNTROL ;YES - CHECK IT
0239 CD4902 CALL SEND ;SEND TO PRINTER
023C C9 RET
023D          ;-----
023D FE0D CNTROL CPI 0DH ;IS IT CARRIAGE RETURN?
023F CC4302 CZ CR ;YES - SEND LF TOO
0242 C9 RET ;NO - THEN RETURN & SEND
0243          ;
0243 CD4902 CR CALL SEND ;SEND CR AND THEN SEND
0246 3E0A MVI A 0AH ;LINEFEED TOO
0248 C9 RET
0249          ;-----
0249 3200FE SEND STA 0FE00H ;SEND DATA
024C E5 PUSH H
024D F5 PUSH PSW
024E 2101FE LXI H 0FE01H
0251 PUT "b"
0251 3640 MVI M 01000000B ;RESET ACK LATCH
0253 3680 MVI M 10000000B ;STROBE PRINTER-DCE BUS OFF!
0255 3600 MVI M 00000000B ;RESET STROBE
0257 PUT "h"
0257 23 INX H
0258 7E ACK MOV A,M ;PORT C BIT 4
0259 E610 ANI 10H ;IS LATCH SET ?
025B CA5802 JZ ACK ;NO TRY AGAIN
025E F1 POP PSW
025F E1 POP H
0260 C9 RET
0261          END

```

KLEURKONFLIKTEN VERMIJDEN  
bij tekenen in 16 kleuren

1. Inleiding

Om in de 16-kleurenschermen (mode 1,3, 5 of 1A,3A,5A) de volledige kleuraange- lijkheden van de DAI te benutten en kleurkonflikten te vermijden, doe je er best aan het scherm in de breedte er best aan het scherm in de breedte denkbeeldig in te delen in velden van acht dots (0-7,8-15,...).

We werken enkel met horizontale lijnen omdat er geen kleurkonflikten optreden in verticale zin. (Je kan zonder problemen 16 dotjes van een verschillende kleur onder elkaar zetten.)

We werken in onze voorbeelden met de volgende kleuren :

- 0 (#0) zwart
- 1 (#1) donkerblauw
- 2 (#2) purperrose
- 3 (#3) rood
- 4 (#4) groenbruin
- 5 (#5) groen
- 6 (#6) bruin
- 7 (#7) purper
- 8 (#8) grijs
- 9 (#9) lavendelblauw
- 10 (#A) oranje
- 11 (#B) rose
- 12 (#C) hemelsblauw
- 13 (#D) lichtgroen
- 14 (#E) geel
- 15 (#F) wit

De 8 dotjes (8 bits) van een zelfde veld kunnen in principe slechts 2 kleuren aannemen : 'foreground' (FG) en 'background' (BG).

Per veld van 8 bits (8 dots) voorziet de computer een veldkleurbyte (VKB) en een velddatabyte (VDB).

A. Veldkleurbyte (VKB)

De VKB geeft de 2 kleuren aan die in zijn veld van 8 dots gebruikt worden.

Die 2 kleuren noemen we in het vervolg de 'foregroundcolour' (FG) en de 'backgroundcolour' (BG).

Stel dat je in een veld van 8 dots de kleuren wit en donkerblauw gebruikt dan ziet de VKB eruit als volgt :

```

/ 1 1 1 1 / 0 0 0 1 /
  15          1
  #F          #1
    
```

of

```

/ 0 0 0 1 / 1 1 1 1 /
  1          15
  #1          #F
    
```

In het vervolg hanteren we de notatie #F1 of #1F.

Het linkse hexadecimaal cijfer van de VKB duidt FG aan, het rechtse duidt BG aan.

B. Velddatabyte (VDB)

De VDB geeft aan waar je FG en BG gebruikt in het veld.

Stel dat je wit en rood (de veldkleurbyte is #F3) op volgende manier gebruikt

```

/ W W R W / R R W R /
    
```

dan ziet de VDB eruit als volgt : #D2

Bij FG staat bit op 1 en bij BG staat bit op 0.

```

VDB / 1 1 0 1 / 0 0 1 0 /
     13          2
     #D          #2
           #D2
    
```

2. Testprogramma

Om dit artikel te volgen en de volgende voorbeelden zelf te kunnen verwerken is het nodig het testprogramma in te tikken.

(listing programma na artikel)

We werken in mode 1A, de schermkleur is grijs en het scherm is opgedeeld in velden van 8 dots (de zwarte lijnen duiden het begin van elk veld aan : 0, 8, 16, ...).

Bij de start van het programma kan je kiezen tussen 'DRAWEN' en 'POKEN'. Al de inputs kunnen zowel decimaal als hexadecimaal ingevoerd worden.

Bij keuze 'DRAWEN' worden de volgende inputs gevraagd :

- LIJN : Hier geef je een getal van 1 tot en met 15. De LIJN-waarde bepaalt de Y-coördinaat, met 1 als hoogste en 15 als laagste positie op het scherm.
- XW-1 : Hier geef je de eerste X-coördinaat. (0 t/m 71)
- XW-2 : Hier geef je de tweede X-coördinaat. (0 t/m 71)
- KLEUR : Hier geef je de kleurwaarde aan. (0 t/m 15)

Bij keuze 'POKEN' worden de volgende inputs gevraagd :

- LIJN : Zie onder 'DRAWEN'.
- VELD : Hier geef je een getal van 1 tot en met 9. Veld nr.1 is het veld met de laagste X-coördinaat.
- VKB : Hier geef je de veldkleurbyte.
- VDB : Hier geef je de velddatabyte. (#0 t/m #FF)

3. Basisprincipes in het grafisch werken

A. Hou er steeds rekening mee dat de schermkleur waarin je werkt (grijs) meetelt als gebruikte kleur.

Als je de veldkleurbytes opvraagt van lijn 1 dan geeft de computer voor al de velden op lijn 1 #80 aan. Hierbij duidt '8' (grijs) op FG en '0' (zwart) op BG.

Bij het 'DRAWEN' betekent die '0' in de beginsituatie dat je gelijk welke kleur kan gebruiken als achtergrondkleur.

Bij het 'POKEN' daarentegen wordt de kleur 'zwart' in het geheugen opgeslagen als BG, wat wel kan gewijzigd worden. (zie punt B)

Wanneer je de velddatabytes opvraagt van lijn 1 dan wordt voor elk veld

#F aangegeven, wat onderdaaf overeenkomt met het volledig gebruik over al de velden van de voorgrondkleur (FG).

Al de bits in elk veld staan op 1 zodat #FF gegenereerd wordt.

Voorbeeld 1

```

Draw inputs : LIJN    1
              XW-1    8
              XW-2   11
              KLEUR   #E
    
```

De veldkleurbyte (VKB) van dit veld is #E8, waarbij #E voorgrondkleur (FG) en #8 achtergrondkleur (BG) is. De velddatabyte (VDB) van dit veld is #F0.

Bij FG staan de bits op 1 en bij BG staan de bits op 0.

```

Dus : / 1 1 1 1 / 0 0 0 0 /
      #F          #0
    
```

Wanneer de VKB en de VDB gekend zijn is het dan ook eenvoudig om hetzelfde effect te verkrijgen via de POKE inputs.

We doen dit op de lijn eronder, zodat dit overduidelijk wordt.

```

Poke inputs : LIJN    2
              VELD    2
              VKB     #E8
              VDB     #F0
    
```

Voorbeeld 2

Hier tekenen we een lijn van acht dots lang, de eerste helft geel de tweede helft rood.

```

Draw inputs : LIJN    3
              XW-1    8
              XW-2   11
              KLEUR   #E
    
```

```

LIJN    3
XW-1   12
XW-2   15
KLEUR  #3
    
```

```
Poke inputs : LIJN 4
              VELD 2
              VKB #E3
              VDB #F0
```

Voorbeeld 3

In het volgende veld wil je purper en wit afwisselend gebruiken.

```
Draw inputs : LIJN 3
              XW-1 16
              XW-2 16
              KLEUR #7
```

```
LIJN 3
XW-1 17
XW-2 17
KLEUR #F
```

Zoals je merkt wordt er geen witte dot geplaatst, omdat je in veld 3 #7 als FG en #8 als BG bepaald hebt zodat wit de derde kleur zou zijn wat niet kan.

Wanneer je 2 kleuren afwisselend wil gebruiken in een veld, moet je er voor zorgen dat je eerst het hele veld kleurt in een van de 2 gewenste kleuren (zo sluit je de schermkleur #8 uit).

```
LIJN 3
XW-1 16
XW-2 23
KLEUR #7
```

```
LIJN 3
XW-1 17
XW-2 17
KLEUR #F
```

```
LIJN 3
XW-1 19
XW-2 19
KLEUR #F
```

enz...

```
Poke inputs : LIJN 4
              VELD 3
              VKB #7F
              VDB #AA
```

B. In elk veld van 8 dots (8 bits) kan je toch drie kleuren gebruiken op voorwaarde dat de eerste dot(s) in het nieuwe veld gekleurd word(en) in de achtergrondkleur van het vorige veld.

Het is dus uitermate belangrijk dat de gebruiker weet welke kleur nu FG (voorgrondkleur) en BG (achtergrondkleur) is.

In principe is de eerste kleur die gebruikt wordt in een veld FG en de tweede BG.

Zie vb.2 : geel = FG en rood = BG

Indien echter de eerste kleur in het nieuwe veld BG is in het vorige veld, dan wordt deze kleur als 'vorige achtergrondkleur (PBG)' beschouwd (bits staan op 0) in het nieuwe veld, indien in dit veld nog twee andere kleuren gebruikt worden.

Voorbeeld 4

We gaan de achtergrondkleur van veld 2 verder gebruiken in veld 3, zodat we in dit veld kunnen beschikken over drie kleuren.

```
Draw inputs : LIJN 5
              XW-1 8
              XW-2 11
              KLEUR #E
```



```
LIJN 5
XW-1 12
XW-2 17
KLEUR #3
```

```
LIJN 5
XW-1 18
XW-2 20
KLEUR #1
```

```
LIJN 5
XW-1 21
XW-2 23
KLEUR #F
```

```
Poke inputs : LIJN 6
              VELD 2
              VKB #E3
              VDB #F0
```

```
LIJN 6
VELD 3
VKB #1F
VDB #38
```

Met de gegevens waarover we nu beschikken bekijken we nogmaals voorbeeld 3.

Omdat de eerste dot in kleur #7 OP DIE MANIER VAN DRAWEN als FG bepaald werd en niet als PBG (previous backgroundcolor), kan je hier onmogelijk nog twee nieuwe kleuren gebruiken.

Om problemen zoals in vb. 3 te vermijden en om, rekening houdend met de voorwaarden, drie kleuren in een veld te gebruiken is het altijd best te DRAWEN VAN RECHTS NAAR LINKS. Op die manier laat je steeds de kans open de derde kleur als PBG te gebruiken.

Voorbeeld 5 (verbeterde versie vb.3)

```
Draw inputs : LIJN 7
              XW-1 17
              XW-2 17
              KLEUR #F
```

```
LIJN 7
XW-1 16
XW-2 16
KLEUR #7
```

```
LIJN 7
XW-1 12
XW-2 15
KLEUR #3
```

```
LIJN 7
XW-1 8
XW-2 11
KLEUR #E
```

Op die manier kan je zelfs gans het derde veld afwisselend vullen met de kleuren #7 en #F.

Voorbeeld 6

```
Draw inputs : LIJN 8
              XW-1 21
              XW-2 23
              KLEUR #E
```

```
LIJN 9
XW-1 18
XW-2 20
KLEUR #1
```

```
LIJN 8
XW-1 16
XW-2 17
KLEUR #5
```

```
LIJN 8
XW-1 12
XW-2 15
KLEUR #3
```

```
LIJN 8
XW-1 8
XW-2 11
KLEUR #A
```

Wanneer je dit probeert te DRAWEN van links naar rechts zal je merken dat dit niet kan.

Bij het POKEN daarentegen moet je steeds van LINKS NAAR RECHTS werken omdat je op voorhand moet bepalen welke kleuren je gebruikt en welke kleur nu FG of BG is.

Hou er dus terdege rekening mee dat wanneer je in een veld drie kleuren wil gebruiken, je de eerste kleur in het nieuwe veld PBG maakt, wat dus wil zeggen dat je die kleur BG maakt in het vorige veld.

Dok in veld 1 kan je drie kleuren gebruiken. Men moet dan de PBG poken in veld 0.

Voorbeeld 7 (voorbeeld 6 POKEN)

```
Poke inputs : LIJN 9
              VELD 1
              VKB #8A
              VDB #FF
```

```
LIJN 9
VELD 2
VKB #35
VDB #F
```

```
LIJN 9
VELD 3
VKB #1E
VDB #38
```



Zoals je merkt, ookten we in veld 1 reeds de BG #A, omdat we in veld 2 drie kleuren willen gebruiken, en dus de eerste kleur in veld 3 BG moet zijn in veld 2, wat impliceert dat de eerste kleur van veld 2 (waarin we twee andere kleuren gebruiken) BG moet zijn in veld 1.

C. Opmerking

Zoals je gemerkt hebt in onze voorbeelden kan je alles zowel POKEN en DRAWEN.  
Er is echter 1 uitzondering waarin je alleen via POKEN het gewenste resultaat verkrijgt.

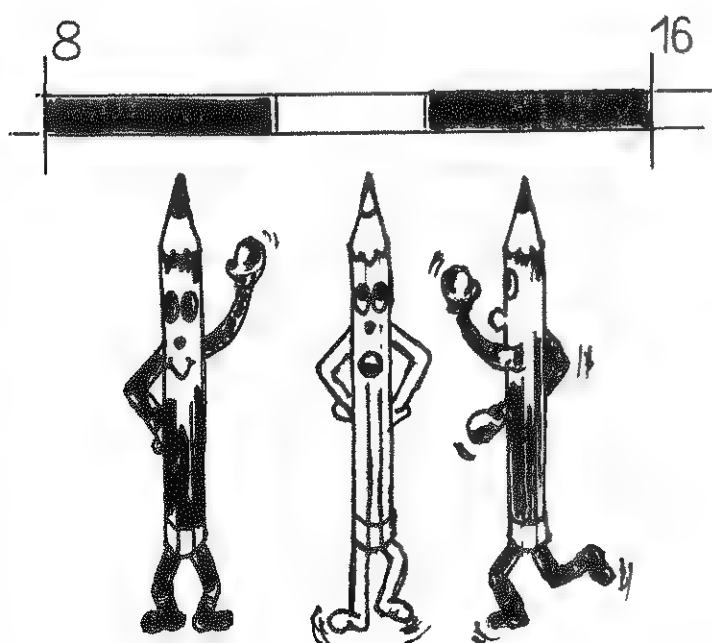
Voorbeeld 8

```

Poke inputs : LIJN  10
              VELD   1
              VKB   #8E
              VDB   #FF

              LIJN  10
              VELD   2
              VKB   #5E
              VDB   # 1

              LIJN  10
              VELD   3
              VKB   #1F
              VDB   #18
  
```



Deze situatie kan je onmogelijk verkrijgen via DRAWEN.  
De stelregel die hier geldt is de volgende : Het is onmogelijk om uitsluitend de laatste dot in een veld FG te maken.

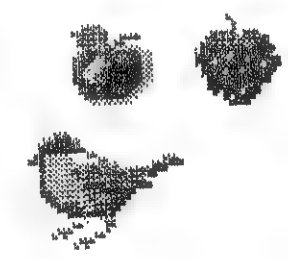
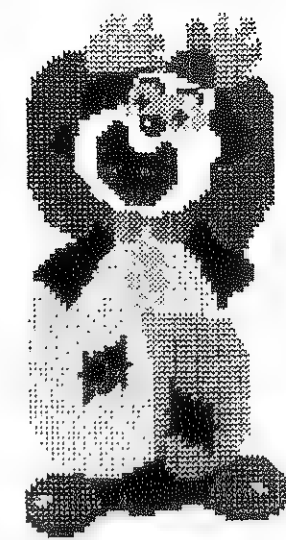
D. Besluit

Wanneer je een bepaalde veelkleurige lijn op het scherm wil tekenen (dikwijls gebruik makend van de mogelijkheid om drie kleuren in een veld te plaatsen) is het toch wel goed vooraf de bitnotatie te noteren. (Bij FG staat bit op 1, bij BG op 0)  
Hieruit volgt volgende stelregel :  
- zolang nullen niet onderbroken worden door een 1 kan de kleurwaarde van de nul niet gewijzigd worden.  
- dus kan men in een veld na een 1 slechts een kleur bij gebruiken.

Het FILL- en het DOT- kommando kan je eveneens gebruiken, als je maar rekening wil houden met deze uitleg.

```

10 REM *****
15 REM *
20 REM * KLEURENKONFLIKTEN *
25 REM *
30 REM *****
40 POKE #75,32
45 PRINT CHR$(12):MODE 0:COLORT 0 14 0 0
50 CURSOR 10,20:INPUT "POKEN OF DRAWEN (P/D) ";PD$
51 CURSOR 10,18:PRINT "POKEN : inputs zijn rose gekleurd"
52 CURSOR 10,17:PRINT "DRAWEN : inputs zijn lichtgroen gekleurd"
53 CURSOR 10,15:PRINT "Om tijdens het werken over te gaan van"
54 CURSOR 10,14:PRINT "POKEN naar DRAWEN of omgekeerd geef"
55 CURSOR 10,13:PRINT "je voor LIJN een waarde die kleiner"
56 CURSOR 10,12:PRINT "is dan 1 of groter is dan 15"
57 CURSOR 10,10:PRINT "Om tijdens het DRAWEN de veldkleurbyte"
58 CURSOR 10,9:PRINT "(VKB) en velddatabyte (VDB) op te vragen"
59 CURSOR 10,8:PRINT "geef je voor XW-1 een waarde hoger dan"
60 CURSOR 10,7:PRINT "100."
61 CURSOR 10,5:PRINT "Om tijdens het POKEN de veldkleurbyte"
62 CURSOR 10,4:PRINT "en velddatabyte op te vragen geef je"
63 CURSOR 10,3:PRINT "voor VELD een waarde die kleiner is"
64 CURSOR 10,2:PRINT "dan 0 of groter dan 9."
75 CURSOR 15,0:PRINT " >>>> druk spatie <<<<";
76 G%=GETC:G%=GETC:G%=GETC
77 G%=GETC:IF G%<>#20 THEN GOTO 77
81 IF PD$="P" THEN GOSUB 100:GOSUB 400:GOTO 1000
82 IF PD$="D" THEN GOSUB 100:GOSUB 400:GOTO 500
85 GOTO 45
100 MODE 2:COLORG 8 0 0 0:COLORT 0 14 0 0:MODE 1A
110 FILL 0,0 XMAX,YMAX 8
200 FOR LZ=0 TO XMAX STEP 8
210 DRAW LZ,0 LZ,17 0
220 DRAW LZ,33 LZ,YMAX 0
230 NEXT
250 RETURN
400 REM *****
401 REM *** SCHERM CLEAREN ***
402 REM *****
405 COLORT 0 0 0 0
410 FOR SCZ=3 TO 0 STEP -1
420 CURSOR 0,SCZ:FOR APOZ=0 TO 59:PRINT " ";:NEXT
430 NEXT
440 CURSOR 0,3
449 RETURN
500 REM *****
501 REM *** D R A W E N ***
502 REM *****
503 COLORT 0 13 0 0
504 INPUT "LIJN ";LZ
505 IF LZ<1 OR LZ>15 THEN GOSUB 400:GOTO 1000
510 PRINT " ";:INPUT "XW-1 ";XW1Z
515 IF XW1Z>=100.0 THEN GOSUB 400:GOTO 500
520 PRINT " ";:INPUT "XW-2 ";XW2Z
530 PRINT " ";:INPUT "KLEUR ";KLZ
550 IF XW1Z>XMAX OR XW1Z<0 OR XW2Z<0 OR XW2Z>XMAX OR KLZ>15 OR KLZ<0 THEN FMZ=1:GOTO 575
560 DRAW XW1Z,33-LZ XW2Z,33-LZ KLZ
575 IF FMZ=1 THEN GOSUB 2000
590 PRINT
595 GOTO 500
  
```



```

600 REM *****
601 REM ***** VELDKLEURBYTE PEEKEN *****
602 REM ***** VELDDATA BYTE PEEKEN *****
603 REM *****
620 GOSUB 400:COLORT 0 12 0 0
625 CURSOR 0,3:PRINT " VKB ";
630 J%=45-L%
635 FOR I%=1 TO 9
640 VKB%=HEX$(PEEK(#BFEF-(YMAX-J%)*24-I%*2-3))
641 PRINT SPC(2-LEN(VKB%));VKB%;" ";
642 NEXT
645 CURSOR 0,1:PRINT " VDB ";
646 FOR I%=1 TO 9
647 VDB%=HEX$(PEEK(#BFEF-(YMAX-J%)*24-I%*2-2))
648 PRINT SPC(2-LEN(VDB%));VDB%;" ";
649 NEXT
650 CURSOR 35,0:PRINT ">>>> druk spatie <<<<";
651 G%=GETC:G%=GETC:G%=GETC
655 G%=GETC:IF G%<>#20 THEN GOTO 655
660 GOSUB 400
699 RETURN
1000 REM *****
1001 REM **** P O K E N ****
1002 REM *****
1010 COLORT 0 11 0 0
1050 INPUT "LIJN ";L%
1055 IF L%<1 OR L%>15 THEN GOSUB 400:GOTO 500
1060 PRINT " ";:INPUT "VELD ";V%
1065 IF V%<0.0 OR V%>9.0 THEN GOSUB 400:GOTO 1000
1070 PRINT " ";:INPUT "KLEURBYTE ";KB%
1080 PRINT " ";:INPUT "DATABYTE ";DB%
1085 IF KB%<0.0 OR KB%>#FF OR DB%<0.0 OR DB%>#FF THEN FM%=1:GOTO 1250
1090 J%=45-L%
1100 A%=#BFEF-(YMAX-J%)*24-V%*2-2
1110 B%=A%-1
1200 POKE A%,DB%
1210 POKE B%,KB%
1250 IF FM%=1 THEN GOSUB 2000
1255 PRINT
1260 GOTO 1050
2000 REM *****
2001 REM * FOUTMELDING INPUT *
2002 REM *****
2003 FM%=0
2004 PRINT
2005 PRINT " **** FOUTIEVE INPUT ****"
2006 RETURN
9999 GOTO 9999

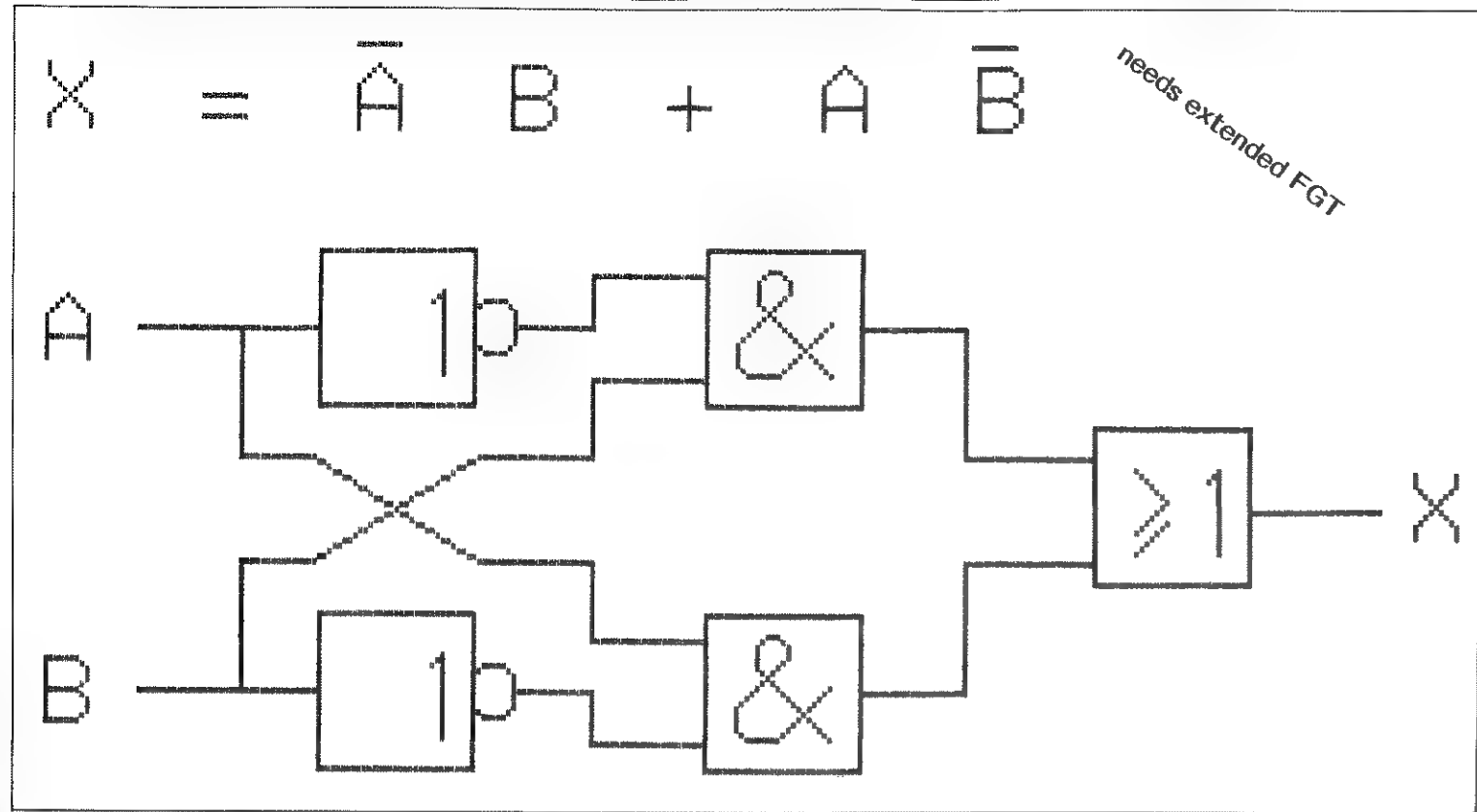
```

This is the set of new symbols for FGT to be used with the program ADC-converter. Other programs using this symbols will be published in following issues. You can enter the data in DAI-monitor with the SUBSTITUE-command.

0	1	2	3	4	5	6	7	8	
13	14	15	16	17	18	19	20	21	
9	10							22	23
11	12							24	

0996	00	0F	00	F0	F0	FF	00	0F	0F	FF						
09A0	43	73	34	36	36	8B	7D	8C	5C	6D	5A	5C	5A	C3	73	C8
09B0	34	43	8B	8C	6D	7D	01	09	00	F0	F0	FF	00	0F	0F	FF
09C0	43	76	45	78	4B	78	C3	CB	BA	CB	02	08	00	0F	0F	FF
09D0	00	F0	F0	FF	C3	CB	BA	CB	29	69	27	67	03	07	06	15
09E0	09	1A	15	35	1A	3A	3A	49	35	46	46	49	04	11	00	F0
09F0	00	0F	0F	FF	F0	FF	28	58	28	29	29	5C	3D	4D	2C	3D
0A00	88	8D	8A	8D	8B	88	44	64	53	55	92	96	85	96	4D	5C
0A10	05	08	40	B0	40	4E	4E	BE	B0	BE	47	B7	58	58	56	56
0A20	A2	A3	06	0A	00	0E	0E	BE	00	B0	BE	BB	B0	B3	B3	F3
0A30	BB	FB	14	44	35	44	33	44	07	06	00	0E	0E	FE	00	F0
0A40	F0	FE	70	7E	80	8E	08	06	00	0E	00	F0	F0	FE	0E	FE
0A50	70	7E	80	8E	09	08	00	0F	00	F0	0F	FF	F0	FF	45	95
0A60	6A	BA	65	6A	95	9A	0A	0A	00	F0	00	0F	0F	FF	F0	FF
0A70	00	FF	39	6C	3C	69	A7	C5	C5	E7	C2	C5	0B	06	00	F0
0A80	00	0F	0F	FF	F0	FF	C3	CB	BA	CB	0C	06	40	4E	40	B0
0A90	B0	BE	4E	BE	88	98	96	98	0D	0D	00	F0	00	0F	0F	FF
0AA0	F0	FF	26	46	46	48	48	68	66	68	66	86	86	88	88	A8
0AB0	A6	A8	A6	C6	0E	06	70	73	53	93	53	5C	5C	9C	93	9C
0AC0	7C	7F	0F	06	07	37	35	39	35	C5	C5	C9	39	C9	C7	F7
0AD0	10	0A	70	75	25	C5	25	26	26	C6	C5	C6	29	C9	C9	CA
0AE0	2A	CA	29	2A	7A	7F	11	0A	07	57	52	5C	52	62	62	6C
0AF0	5C	6C	92	9C	92	A2	A2	AC	9C	AC	A7	F7	12	0C	07	57
0B00	54	5A	5A	6A	64	6A	54	64	68	AC	66	A2	84	94	83	84
0B10	83	94	A0	A2	AC	AF	13	0D	07	47	44	4A	44	54	54	5A
0B20	4A	5A	58	9C	9C	9F	56	92	56	92	90	92	73	83	83	84
0B30	73	84	14	06	20	2E	2E	97	20	97	97	F7	02	22	0C	2C
0B40	15	0F	07	27	27	29	29	3A	3A	4A	4A	59	57	59	59	6A
0B50	6A	7A	7A	89	87	89	89	9A	9A	AA	AA	B9	B7	B9	B7	D7
0B60	16	0F	70	72	72	92	92	A3	A3	A4	95	A4	75	95	95	A6
0B70	A6	A7	98	A7	78	98	98	A9	A9	AA	9B	AA	78	98	78	7D
0B80	17	11	00	F0	F0	FF	00	0F	0F	FF	00	FF	28	2C	2C	3D
0B90	3D	4D	58	5C	4D	5C	2A	5A	A2	C2	A2	A7	A7	C7	C2	D3
0BA0	D3	D6	C7	D6	18	11	00	F0	00	0F	0F	FF	F0	FF	00	FF
0BB0	28	2D	2D	4D	4D	5C	28	48	48	59	59	5C	D2	D6	C7	D6
0BC0	A2	A6	A6	B7	B7	C7	A4	D4	5A	FF	00	00	00	00	00	00



PAGE 01 -- EXOR

```

10  MODE 0:COLORT 9 14 0 0:MODE
    6A:COLORG 9 10 14 15
20  A#=CHR$(1):SP=15:DF=1:CC=22:
    X=245:Y=81:GOSUB 10000
30  A#=CHR$(0):X=170.0:Y=45.0:
    GOSUB 10000
40  A#=CHR$(0):Y=115.0:GOSUB
    10000
50  A#=CHR$(11)+CHR$(3):X=95:Y=
    45:GOSUB 10000
60  Y=115:GOSUB 10000
70  A#="X=AB+AB":X=40:Y=180:
    GOSUB 10000
75  A#="A":X=40:Y=120:GOSUB
    10000
76  A#="B":X=40:Y=50:GOSUB 10000
77  A#="X":X=305:Y=86:GOSUB
    10000
80  A#="-":X=100:Y=192:GOSUB
    10000
90  A#="-":X=219:Y=192:GOSUB
    10000
100 DRAW 134,60 148,60 21:DRAW
    148,60 148,50 21:DRAW 148,
    50 169,50 21
105 DRAW 94,60 60,60 21:DRAW 94,
    130 60,130 21
110 DRAW 134,130 148,130 21:
    DRAW 148,130 148,140 21:
    DRAW 148,140 169,140 21
120 DRAW 201,130 220,130 21
130 DRAW 201,60 220,60 21

```

```

140 DRAW 220,130 220,105 21
150 DRAW 220,60 220,85 21
160 DRAW 220,105 244,105 21
170 DRAW 220,85 244,85 21
180 DRAW 148,70 169,70 21:DRAW
    148,85 148,70 21
190 DRAW 148,120 169,120 21:
    DRAW 148,105 148,120 21
200 DRAW 148,85 126,85 21:DRAW
    126,85 94,105 21:DRAW 94,
    105 80,105 21:DRAW 80,105
    80,130 21
210 DRAW 148,105 126,105 21:
    DRAW 126,105 94,85 21:DRAW
    94,85 80,85 21:DRAW 80,85
    80,60 21
220 DRAW 276,95 300,95 21
225 PRINT CHR$(12);
230 PRINT " EXCLUSI
    EVE OF - S C H A K E L
    I N G"
1000 PRINT :END
10000 REM SUBROUTINE FGT
10010 POKE #2F2,X MOD 256:POKE
    #2F3,X/256:POKE #2F4,Y
10020 POKE #2F5,SP:POKE #2F6,ID
10030 C=FC**40+CC:F=FF**80+PF**40+
    ZF**20+VF**10+DF:POKE #2F0,
    C:POKE #2F1,F
10040 CALLM #300,A#
10050 RETURN

```

Het programma STATISTISCHE ANALYSE bestaat uit 4 delen. De eerste 3 berekenen het gemiddelde van een aantal gegevens met de standaardafwijking (of middelbare fout) hierop. Er wordt gebruik gemaakt van formules uit de statistiek.

De middelbare fout (MF) wordt automatisch afgerond op 1 beduidend cijfer als ze kleiner is dan 0.1, anders tot op 2 cijfers. Ook het gemiddelde wordt afgerond tot op dezelfde eenheid als de MF.

Vb: 0.0254 wordt 0.02  
 0.4312 " 0.43  
 0.00456 " 0.005

1° Statistisch gemiddelde

Toepassen als alle metingen verricht werden met dezelfde precisie  
 bv. men meet 3 keer de lengte van een voetbalterrein en vindt 100.14 m , 100.13 m , 100.16 m waarbij de fout op de laatste decimaal zit.

Het gemiddelde  $\langle x \rangle = \frac{\sum x_i}{N} = \frac{100.14 + 100.13 + 100.16}{3} = 100.14333...$

$$M.F. = \sqrt{\frac{\sum (x_i - \langle x \rangle)^2}{N(N-1)}} = \sqrt{\frac{(100.14 - 100.14333...)^2 + (100.13 - \langle x \rangle)^2 + (100.16 - \langle x \rangle)^2}{3(3-1)}}$$

= 0.0088192...  
 → x = 100.143 +/- 0.009 m

2° Gewogen gemiddelde 1

Toepassen op metingen die verricht werden met een verschillende precisie. De precisie kan veranderen omdat bv. andere meetapparatuur werd gebruikt of een andere proefopstelling.

Vb. Er worden 3 verschillende experimenten gedaan om het alcoholgehalte te bepalen en men vindt 8.7 +/- 0.1  
 8.65 +/- 0.05 en 8.8 +/- 0.3 .

Het is duidelijk dat de nauwkeurigste meting het dichtst bij het echte gemiddelde zal liggen.

Gemiddelde  $\langle x \rangle = \frac{\sum g_i x_i}{\sum g_i} = 8.66305 \approx 8.66$

met  $g_i = \left(\frac{1}{MF_i}\right)^2$  MF<sub>i</sub> = de fout op elke afzonderlijke meting

$$RF = \frac{1}{\sqrt{\sum g_i}} = 4.42326E-2 \approx 0.04$$

### 3° Gewogen gemiddelde 2

Gemiddelde en standaardafwijking van een populatie.

Vb. Men telt het aantal larven van vlinders die per dag uitkomen.

$x_i$	dagen	30	31	32	33	34
$g_i$	aantal	10	52	17	5	1

Na hoeveel dagen gemiddeld zal er een larve uitkomen met de standaardafwijking hierop.

$$\langle x \rangle = \frac{\sum g_i x_i}{\sum g_i} = 31.077 \approx 31.31$$

$$\text{Standaardafwijking} = \sqrt{\frac{\sum g_i \sum g_i x_i^2 - (\sum g_i x_i)^2}{\sum g_i (\sum g_i - 1)}} = 0.88$$

### 4° Lineaire regressie

Dit is het 4° deel van het programma waarmee de best passende rechte door een aantal punten berekend wordt.

Gebruikte formules voor  $y = a + bx$

$$b = \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2}$$

$$a = \frac{\sum y_i - b \sum x_i}{n}$$

$$r = \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{\sqrt{[n \sum x_i^2 - (\sum x_i)^2][n \sum y_i^2 - (\sum y_i)^2]}} \quad (\text{correlatiecoefficient})$$

Voor meer uitleg hierover verwijs ik naar het artikel "Lineaire regressie" in Dainamic 30 van Aad de Bruijn.

+ + + + + + + + +

Verberckmoes Filip

Meidoornlaan.16

2750 BEVEREN

```

PAGE      1  STATISTISCHE ANALYSE
          20  TITLE "STATISCHE ANALYSE"
          30  REM Verberckmoes Filip  11/1985
          40  GOTO "MAIN"
          50  PROCEDURE KOLOM
          60  1  IF FLAG=0 THEN
          70  2  A=A+1:CRY=CRY-1:CURSOR 0,CRY
          80  1  PRINT A;:CURSOR CRX,CRY:
          90  2  END IF
          100 1  IF FLAG=1 THEN
          110 2  KOLM=KOLM+1
          120 2  IF KOLM=2 THEN
          130 3  KOLM=0:CRX=CRX-STAP:CRY=CRY-1
          140 3  A=A+1:CURSOR 0,CRY:PRINT A;:CURSOR CRX,CRY
          150 2  ELSE
          160 3  CRX=CRX+STAP:CURSOR CRX,CRY
          170 2  END IF :
          180 1  END IF
          190 END PROC
          200 PROCEDURE CORRECTIE CX,CY
          210 1  K=N:CURSOR CX,CY:POKE #75,9
          220 1  REPEAT
          230 2  H=GETC
          240 2  ON ABS(H-15) GOSUB "UP","DOWN","LEFT","RIGHT
          250 3  GOSUB "CORR:
          260 2  END IF
          270 1  UNTIL H=69:
          280 1  CURSOR CRX,CRY:POKE #75,95
          290 END PROC
          300 "UP
          310 IF CY<=17 THEN
          320 1  CY=CY+1:K=K-1-FLAG
          330 1  CURSOR CX,CY:
          340 1  END IF :
          350 1  RETURN
          360 "DOWN
          370 IF CY>=2 THEN
          380 1  K=K+1+FLAG:CY=CY-1
          390 1  CURSOR CX,CY:
          400 1  END IF :
          410 1  RETURN
          420 "LEFT
          430 IF CX=5+STAP THEN
          440 1  K=K-1:CX=CX-STAP
          450 1  CURSOR CX,CY:
          460 1  END IF :
          470 1  RETURN
          480 "RIGHT
          490 IF CX=5 AND FLAG=1 THEN
          500 1  K=K+1:CX=CX+STAP
          510 1  CURSOR CX,CY:
          520 1  END IF :
          530 1  RETURN
          540 "CORR
          550 CURSOR CX,CY:PRINT SPC(STAP-1);:CURSOR CX,CY
          560 INPUT MT!(K):CURSOR CX,CY:RETURN
          570 PROCEDURE WIS
          580 1  IF FLAG=0 THEN
          590 2  FOR I=18 TO 1 STEP -1
          600 3  CURSOR 0,I:PRINT SPC(4);:CURSOR 5,I:PRINT SPC(12)
          610 2  NEXT I:

```



```

DBASIC V2.2
PAGE 4 STATISTISCHE ANALYSE

1360 FOP 1-4 TO 43 STEP 13
1370 1 FOR J=20 TO 1 STEP -1
1380 2 CURSOR I,J:PRINT T$;;
1390 1 NEXT
1400 REPEAT
1410 1 N=N+1
1420 1 INVOER
1420 2 WIS :
1430 1 END IF
1430 UNTIL QUIT=1:
1440 N=N-1
1440 FOR I=1 TO N:
1450 1 SOMX!=SOMX!+MT!(I)
1450 NEXT I:
1460 GEMX!=SOMX!/N
1470 SCROLL :PRINT CHR$(12);
1470 FOR I=1 TO N:
1480 1 DI!=MT!(I)-GEMX!:DD!=DI!*DI!
1480 1 RES!(0)=MT!(I):RES!(1)=DI!:RES!(2)=DD!:AR=3
1490 1 UITVOER
1500 1 SOMDD!=SOMDD!+DD!
1510 NEXT I
1520 MF!=SQR(SOMDD!/(N*(N-1))):MF$=STR$(MF!):MFROUND
1530 UN!=SEARCHUNIT!(MF!)
1540 IF UN!>=0.1 THEN
1540 1 UN!=UN!/10:
1550 END IF
1550 GEM!=ROUND!(GEMX!,UN!)
1560 PRINT :PRINT "X = ";GEM!;" +/- ";MFTOT$:PRINT
1570 GOTO "EINDE
1600 "GEWG1
1610 FLAG=1:CRX=18:CRY=19:KOLM=1:STAP=13
1620 PRINT CHR$(12);TAB(15);"GEWOGEN GEMIDDELDE 1"
1630 PRINT TAB(15);"= = = = = = = = = = ="
1640 PRINT " # x MF g=(1/MF)^2 g*x"
1650 FOR I=0 TO 55:
1650 1 PRINT CHR$(11);:
1660 NEXT I
1660 FOR I=4 TO 57 STEP 13
1670 1 FOR J=20 TO 1 STEP -1
1680 2 CURSOR I,J:PRINT T$;;
1690 1 NEXT
1700 NEXT
1710 REPEAT
1720 1 N=N+1
1730 1 INVOER
1740 1 IF CRY=1 AND KOLM=1 THEN
1750 2 WIS :
1760 1 END IF
1770 UNTIL QUIT=1:
1780 N=N-1
1790 SEPERATE
1800 SCROLL :PRINT CHR$(12);
1810 FOR I=1 TO NN
1820 1 GI!=(1/MT2!(I))^2:GXI!=GI!*MT1!(I)
1830 1 RES!(0)=MT1!(I):RES!(1)=MT2!(I):RES!(2)=GI!
1840 1 RES!(3)=GXI!:AR=4
1850 1 UITVOER

```

```

DBASIC V2.2
PAGE 5 STATISTISCHE ANALYSE

1810 1 SOMG!=SOMG!+GI!:SOMGX!=SOMGX!+GXI!
1820 NEXT I
1830 MF!=SQR(1/SOMG!):MF$=STR$(MF!):MFROUND
1840 GEMX!=SOMGX!/SOMG!:UN!=SEARCHUNIT!(MF!)
1850 IF UN!>=0.1 THEN
1850 1 UN!=UN!/10:
1860 END IF
1860 GEM!=ROUND!(GEMX!,UN!)
1870 PRINT :PRINT
1880 PRINT "X = ";GEM!;" +/- ";MFTOT$
1890 PRINT :GOTO "EINDE
2000 "GEWG2
2010 FLAG=1:CRX=18:CRY=19:KOLM=1:STAP=13
2020 PRINT CHR$(12);TAB(15);"GEWOGEN GEMIDDELDE 2"
2030 PRINT TAB(15);"= = = = = = = = = = ="
2040 PRINT " # x g g*x g*x^2"
2050 FOR I=0 TO 55:
2050 1 PRINT CHR$(11);:
2060 NEXT I
2060 FOR I=4 TO 57 STEP 13
2070 1 FOR J=20 TO 1 STEP -1
2080 2 CURSOR I,J:PRINT T$;;
2090 1 NEXT
2100 NEXT
2110 REPEAT
2120 1 N=N+1
2130 1 INVOER
2140 1 IF CRY=1 AND KOLM=1 THEN
2150 2 WIS :
2160 1 END IF
2170 UNTIL QUIT=1:
2180 N=N-1
2190 SEPERATE
2200 SCROLL :PRINT CHR$(12);
2210 FOR I=1 TO NN
2220 1 GX!=MT1!(I)*MT2!(I):GXX!=MT2!(I)*(MT1!(I)^2)
2230 1 RES!(0)=MT1!(I):RES!(1)=MT2!(I):RES!(2)=GX!
2240 1 RES!(3)=GXX!:AR=4
2250 1 UITVOER
2260 1 SOMGXX!=SOMGXX!+GXX!:SOMGX!=SOMGX!+GX!
2270 1 SOMG!=SOMG!+MT2!(I)
2280 NEXT I
2290 TT!=SOMGXX!-(SOMGX!^2)/SOMG!
2300 MF!=SQR(TT!/(SOMG!-1)):MF$=STR$(MF!):MFROUND
2310 GEMX!=SOMGX!/SOMG!:UN!=SEARCHUNIT!(MF!)
2320 UN!=UN!/10:GEM!=ROUND!(GEMX!,UN!)
2330 PRINT :PRINT "X = ";GEM!;" +/- ";MFTOT$
2340 PRINT :GOTO "EINDE
3000 "LINRES
3010 FLAG=1:CRX=16:CRY=19:KOLM=1:STAP=11
3020 PRINT CHR$(12);TAB(15);"LINEAIRE REGRESSIE"
3030 PRINT TAB(15);"= = = = = = = = = = ="
3040 PRINT " # x y y*x x^2 y^2"
3050 FOR I=0 TO 59:
3050 1 PRINT CHR$(11);:
3060 NEXT I
3070 FOR I=4 TO 59 STEP 11
3080 1 FOR J=20 TO 1 STEP -1
3090 2 CURSOR I,J:PRINT T$;;

```

```

1 NEXT
3090 NEXT
4000 REPEAT
1 N=N+1
4010 1 INVOER
4020 1 IF CRY=1 AND KOLM=1 THEN
2 WIS :
1 END IF
4030 UNTIL QUIT=1:
N=N-1
4040 SEPERATE
4050 SCROLL :PRINT CHR$(12);
4060 FOR I=1 TO NN
4070 1 XY!=MT1!(I)*MT2!(I)
4080 1 XX!=MT1!(I)^2:YY!=MT2!(I)^2
4090 1 RES!(0)=MT1!(I):RES!(1)=MT2!(I):RES!(2)=XY!
4100 1 RES!(3)=XX!:RES!(4)=YY!:AR=5
4110 1 UITVOER
4120 1 SOMX!=SOMX!+MT1!(I):SOMY!=SOMY!+MT2!(I)
4130 1 SOMXY!=SOMXY!+XY!:SOMXX!=SOMXX!+XX!
4140 1 SOMYY!=SOMYY!+YY!
4150 NEXT I
4160 PRINT
4170 RES!(0)=SOMX!:RES!(1)=SOMY!:RES!(2)=SOMXY!
4180 RES!(3)=SOMXX!:RES!(4)=SOMYY!:AR=5:I=0
4190 UITVOER
4200 TELLER!=NN*SOMXY!-SOMX!*SOMY!
4210 NOEMER!=NN*SOMXX!-SOMX!^2
4220 RICO!=TELLER!/NOEMER!
4230 AX!=(SOMY!-RICO!*SOMX!)/NN
4240 NOEMER1!=NN*SOMXX!-SOMX!^2
4250 NOEMER2!=NN*SOMYY!-SOMY!^2
4260 CORR!=TELLER!/SQR(NOEMER1!*NOEMER2!)
4270 PRINT :PRINT "Y = ";AX!;" + ";RICO!;" * X"
4280 PRINT :PRINT "CORRELATIECOEFFICIENT = ";CORR!
4290 PRINT
4300 INPUT "HOEVEEL BEDUIDENDE CIJFERS WENST U!";BD
4310 PRINT :K!=10^(BD-1)
4320 UN!=SEARCHUNIT!(AX!):UN!=UN!/K!:AX!=ROUND!(AX!,UN!)
4330 UN!=SEARCHUNIT!(RICO!):UN!=UN!/K!
4340 RICO!=ROUND!(RICO!,UN!)
4350 UN!=SEARCHUNIT!(CORR!):UN!=UN!/K!
4360 CORR!=ROUND!(CORR!,UN!)
4370 PRINT :PRINT "Y = ";AX!;" + ";RICO!;" * X"
4380 PRINT :PRINT "CORRELATIECOEFFICIENT = ";CORR!
4390 GOTO "EINDE
5000 "UITLEG
PRINT CHR$(12);
5010 PRINT TAB(15);"UITLEG":PRINT TAB(14);"= = = = ="
5020 PRINT "STATISTISCHE ANALYSE is een programma dat gemiddelden met"
5030 PRINT "hun standaardafwijking berekent."
5040 PRINT "Daarnaast kan de kleinste kwadraten rechte berekend
worden."
5050 PRINT :PRINT "1 Statistisch gemiddelde : gemiddelde van een
aantal"
5060 PRINT TAB(27);"metingen (x) die allen dezelfde"
5070 PRINT TAB(27);"middelbare fout (MF) hebben."
5080 PRINT TAB(27);"vb. x=7.01 m; 7.02 m; 7.01 ..."
5090 PRINT :PRINT "2 Gewogen gemiddelde 1 : gemiddelde van een
aantal"
5100 PRINT TAB(27);"metingen (x) die verschillende "

```

```

5110 PRINT TAB(27);"middelbare fouten (MF) hebben."
5120 PRINT TAB(27);"vb. 7.02 +/- 0.01; 7.01 +/- 0.05"
5130 PRINT :PRINT "3 Gewogen gemiddelde 2 : gemiddelde van een
aantal"
5140 PRINT TAB(27);"metingen (x) die een aantal"
5150 PRINT TAB(27);"keer (g) voorkomen"
5160 PRINT TAB(27);"vb. staartlengte muizen "
5170 PRINT TAB(30);"16.1 cm komt 10 keer voor"
5180 PRINT TAB(30);"16.2 cm komt 20 keer voor"
5190 PRINT TAB(30);"16.3 cm komt 8 keer voor"
5200 INPUT " DRUK RETURN -->";RET$
5210 PRINT CHR$(12):PRINT "4 Lineaire regressie : opstellen van de
best"
5220 PRINT TAB(23);"passende rechte tussen de meetpunten"
5230 PRINT TAB(23);"X en Y via de methode van de kleinste"
5240 PRINT TAB(23);"kwadraten + de correlatiecoefficient."
5250 PRINT :PRINT :PRINT TAB(20);"WERKING":PRINT TAB(19);"= = = = ="
5260 PRINT :PRINT "Corrigeren van een voorgaande meting : tik C
[RETURN]"
5270 PRINT :PRINT "Besturing cursor via de cursortoetsen."
5280 PRINT :PRINT "[SHIFT] + ";CHR$(137);" : deze meting verbeteren."
5290 PRINT :PRINT "E : (zonder RETURN !) einde corrigeren"
5300 PRINT :PRINT "Als alle metingen zijn ingetikt : E [RETURN]."
5310 PRINT :PRINT :INPUT "DRUK RETURN -->";RET$
5320 GOTO "MAIN
10000 "EINDE
PRINT
10010 INPUT "Wilt U meer berekeningen maken (J/N)";AN$
10020 IF AN$="J" THEN
1 MODE 0:GOTO "MAIN:
END IF
10030 MODE 0:PRINT CHR$(12)
10040 CURSOR 0,20:PRINT "TOT DE VOLGENDE KEER.":END

```

PAGE 01 -- GOSSIM

```

1 REM *****
2 REM *** COSSIN vAm '84 ***
3 REM *****
4 REM
9 CLEAR 256:MODE 0:PRINT CHR$(12);:
MODE 6:COLORG 0 5 7 0
300 YM=YMAX/2.0:PI2!=2.0*PI/(XMAX+20.0):
CC=21:E=XMAX-1.0
310 FOR X=E TO 1.0 STEP -2.0:CX!=COS((X+
10.0)*PI2!)
320 1 Y1=35+92*CX!:Y2=127*CX!:X1=X+Y2-Y1
330 1 FILL X,YM-Y1 X1,YM-Y2 CC
340 1 FILL E-X,YM+Y1 E-X1,YM+Y2 CC
350 CC=22+21-CC:NEXT
999 GOTO 999

```

see picture of GOSSIM on backside of this issue

PROGRAMMING THE DAI IN MACHINE AND ASSEMBLY LANGUAGES  
by C W Read

Part 1 - GETTING STARTED

This series of articles is intended for DAI users who wish to venture into the realms of machine code programming. The advantages of machine code over BASIC really become obvious with long programmes: much quicker loading from audio cassette and faster run times. There are also some jobs that can only be done with a machine code programme: for example, a table later in this article is a copy of the screen display from the DAI's Utility LOOK facility. A short machine code programme was written to transfer that table from screen to the text buffer of the word processor used for typing this. BASIC could not be used because the action of switching from Utility to Basic would clear the screen, deleting the table before it could be copied.

The starting point is the DAI's versatile Machine Language Utility Monitor (see DAI Manual pages 106-117) which is used, instead of BASIC, for entering and displaying machine code instructions. For the present, only six of the Utility commands are needed. They are: Z, initialize; F, fill; S, substitute; D, display; G, go; and L, look. Short programmes, say up to about 30 lines, can be entered with the S command. Longer programmes need an assembler. In this first article we will manage without an assembler by keeping the programme examples short. The examples introduce a few instructions but are chiefly aimed at demonstrating movement of information within the microprocessor. Subsequent articles will deal with more machine code instructions and will show how DAI ROM routines can be used to shorten our machine code programmes. Later examples will include subroutines that can be used to build up a simple machine language word processor or file writer programme.

It may be helpful to start with a few definitions:

- Assembler ... a programme which checks the validity of assembly language language statements and converts them into machine code.
- Assembly language ... a programming language that allows the use of mnemonics in place of machine code instructions and labels instead of addresses.
- Source Code/Programme ... statements/programme written in a higher level language than machine code; eg BASIC or assembly language.
- Object Code/Programme ... machine language code/programme produced by an assembler or compiler from source code.
- Machine Code/Machine Language Programme (MLP) ... instructions/programme expressed in the form they are used by the computer.
- Register .... a device for storing data temporarily.

For programming in either assembly or machine language it is necessary to know a little about the microprocessor in the computer. The DAI uses the 8080A which has 10 registers available to the user: 2 have capacities of 2 bytes (16 bits); the others are of 1 byte (8 bits) capacity but they can be paired to make 2-byte registers. The ten registers are:

Register B	General purpose.	B and C can be paired.
Register C	ditto	
Register D	ditto	D and E can be paired.
Register E	ditto	
Register H	ditto	H and L can be paired.
Register L	ditto	

The 2-byte capacity registers:

PC register	Programme Counter. It stores the 2-byte address where the next instruction is to be found.
SP register	Stack Pointer. It stores the 2-byte address where the last item placed on the stack resides.

In assembly language, paired registers adopt the identity of the first register of the pair, eg: B identifies both register B and register-pair BC; the type of instruction used determines whether one register or a pair is needed. Although registers H and L are classed as general purpose registers their main use is as a register pair to hold 2-byte memory addresses, with H holding the high byte of the address and L the low byte. Data in any register or register-pair can be incremented or decremented by one, but that is the only arithmetic that can be done in general purpose registers; the accumulator is used for anything more complicated, and it is in the accumulator that the result of the arithmetic operation will be found; the result replacing any previous content.

The Flag register holds 8 bits, but only 5 are flags. The flags are Carry, Parity, Zero, Sign and Auxiliary carry. They signal conditions resulting from arithmetical or logical operations within the microprocessor, eg: if the result of a calculation was a negative value the Sign flag would be set to 1, otherwise it would be 0. The bit number of each flag is given in the DAI Manual (page 113). Use of flags will be covered in Part 2.

Microprocessor operating instructions, often referred to as op codes, are 8-bit binary codes. The DAI displays and accepts 8 binary digits as 2 hexadecimal characters, and it is these that make up the machine code. Hex characters may be simple to key into a computer but they are hard to remember. Therefore they are replaced by mnemonics to ease the task of writing source code. Some machine code instructions are single byte and are self-sufficient, others need single or double-byte immediate operands. Here are two examples of each:-

Hex machine code	Mnemonic code	Comments
(no operands)		
50	MOV D,B	Move data to register D from register B.
0C	INR C	Increment register C (add 1 to the data already in C).
(1-byte operands)		
3E 41 (operand=41)	MVI A,41	Put #41 into the accumulator (A).
C6 17 (operand=17)	ADI 17	Add #17 to value in accumulator.
(2-byte operands)		
21 34 12 (operand=3412)	LXI H,1234	Load register pair HL with the memory address #1234.
C3 00 12 (operand=0012)	JMP 1200	Jump to (Go to) address #1200.

The 1-byte capacity registers:

Register A	Accumulator	A and Flag paired are called
Flag register	Holds 5 flags.	the Programme Status Word (PSW).



Note that operands are in hex and that the bytes in the operands of the last 2 examples are reversed when translated from mnemonic form to machine code; the microprocessor wants the low address byte of the operand first, then the high byte.

The full 8080A instruction set is printed at the end of DAINamic 24. For a comparison of the instructions with their BASIC equivalents see Dave Atherton's list in DAINamic 15, and at the same time correct a misprint there: hex code 12 is shown as LDAX D; it should read STAX D.

Now try a simple machine code programme to put the letters DAI on the screen. Assembly language mnemonics will be used for writing the source code. The mnemonics will then be translated into machine code by referring to the 8080A instruction set. Place the letters near the bottom of the screen, at say position #B4A1 (character position 43, 3 lines up). This will leave the top of the screen free for demonstrating the working of the programme. #B4A1 is a 2-byte address and therefore has to be stored in a register pair; HL is the most suitable pair so we will load the address into that, using an LXI instruction. LXI is the mnemonic for Load extended Immediate (extended signifies a register pair; immediate refers to the data or address that immediately follows the register identity. See line 1 of Programme 1). Next we will store the 3 letters D, A and I; they will be stored singly so 1-byte registers will be suitable, using MVI (Move Immediate) instructions to load registers A,B and C. We have to load the hex ASCII code of the required letter, which for D is 44; if we loaded D the micro-processor would accept it as #D which is decimal 13. (See lines 2,3 and 4).

When screen address and the letters are in registers the programme has to move the letters to the screen: a move instruction is again wanted but this time not a Move Immediate because the data is already waiting in registers; the mnemonic is MOV but destination and source of the data to be moved must be added. The destination is M which means the Memory at the address held in register pair HL. The source is register A. Note that the mnemonic has the destination before the source: MOV M,A. Line 5 puts D on the screen. The next step is to change the address in HL to indicate the next character position on the screen; lines 6 and 7 do that by subtracting 2 from HL. Moving letters A and I to the screen could be done by MOV M,B and MOV M,C instructions but a longer method, moving them to the screen via the accumulator, is used for this demonstration. Lines 8 and 9 deal with letter A, lines 10 to 13 with letter I.

#### Programme 1.

Source Code	Object Code (-bytes-)	Comments
	(1 2 3)	
0 NOP	00	No operation. Micropr steps over this.
1 LXI H,B4A1	21 A1 B4	Load reg pair HL with address #B4A1. (B4 goes into reg H, A1 into reg L)
2 MVI A,44	3E 44	Move to reg A the ASCII code for 'D'
3 MVI B,41	06 41	Move to reg B the ASCII code for 'A'
4 MVI C,49	0E 49	Move to reg C the ASCII code for 'I'
5 MOV M,A	77	Move the contents of reg A into the memory addressed by reg pair HL. In this example the instruction takes #44 from A and puts it in #B4A1.
6 DCX H	2B	Decrements the HL register pair. It
7 DCX H	2B	is done twice to step over the colour byte and on to the next character

8 MOV A,B	78	position of the screen, #B49F.
9 MOV M,A	77	Move to A the #41 stored in B
		Put into memory location #B49F the #41 that is now in reg A.
10 DCX H	2B	Step over colour byte to the next character position.
11 DCX H	2B	Move to A the #49 stored in C
12 MOV A,C	79	Put #49 into memory #B49D
13 MOV M,A	77	No operation.
14 NOP	00	Stops the microprocessor.
15 HLT	76	

Byte number 1 in the object code column is the op code, bytes 2 and 3 are the immediate operands. The line numbers shown are not needed by the computer; they are only for our convenience.

Now the machine code has to be loaded into memory (RAM) so that it can be run. We will place it at address #300 onwards; it could of course be placed anywhere in the available RAM space. Switch the DAI to its Machine Language Utility mode (page 106 of the DAI Manual). It is worthwhile filling an area of memory with noughts before entering a short programme; there is no technical need for this but it does make for easier viewing with the Utility Display facility if bytes left from earlier programmes are cleared away. Use the Utility command FILL (page 111 of the Manual) as follows:-

```
F300 3FF 0
```

Now type in the object code programme manually using the Utility command SUBSTITUTE (page 112 of the Manual). Start with S300 and enter the object code a byte at a time thus:- S300 00 21 A1 B4 3E etc, ending with 76. Then leave the substitute routine (with cursor left key) and use the DISPLAY command (p 110 of the Manual) to check the code entered:

```
D300 31F
```

The screen display should be the same as this:

```
0300 00 21 A1 B4 3E 44 06 41 0E 49 77 2B 2B 78 77 2B
0310 2B 79 77 00 76 00 00 00 00 00 00 00 00 00 00
```

Initialise with the Utility command Z3 then run the programme Utility command G300. The letters DAI should appear on the screen and then the cursor symbol will stop flashing because the HLT instruction at the end of the programme has stopped the microprocessor. Some means of terminating the run is needed, like the END statement in BASIC. HLT is rather drastic inasmuch as it is necessary to use the RESET button to restore the DAI to normal, but it is a convenient, single byte instruction. Now reset the DAI, using the button, switch back to Utility and Display with D300 31F again. The machine code programme will still be there, even after the reset. That is an advantage which BASIC cannot match!.

Use the Utility LOOK command to see what is happening in the registers during a run, but first clear the screen and initialise registers and vectors. Initialisation (with Z3) is always necessary before the G or L Utility commands. Switching to BASIC and back again to Utility is an easy way to clear the screen. The sequence is as follows:-

```
B
UT
Z3
L300 300 313
```

A table will be displayed (there is a copy on next page). The first entry 'I' on each line is the address of the instruction just executed;

P, the programme counter, gives the address of the next instruction; A to L are the registers and the bytes beside them are their contents. The Flag (F) and Stack pointer (S) registers can be ignored as they are not used in this short programme. The first line shows all the registers at zero after the first instruction. They were so set by the Utility Z3 command and stayed like that because the first machine code was a 'no operation' (NOP) which did nothing more than increment the Programme counter register. The second line shows that the screen address has been loaded into the HL pair by LXI H,B4A1. As that was a 3-byte instruction the Programme counter stepped on 3 to point to the next instruction at #304. A glance down columns H and L shows when the address in HL is decremented. The ASCII codes for the letters go into the appropriate register one after the other. Note that the various 'move' instructions only copy data to the destination; the original data remains in the source until it is over-written by a later move or load instruction. Thus 41 is moved to the accumulator from the B register by the instruction MOV A,B at #30D, yet still remains in B for the rest of the programme.

The RESET button has to be used again to restart the DAI.

```

I=0300 A=00 F=02 B=00 C=00 D=00 E=00 H=00 L=00 S=F8FE P=0301
I=0301 A=00 F=02 B=00 C=00 D=00 E=00 H=B4 L=A1 S=F8FE P=0304
I=0304 A=44 F=02 B=00 C=00 D=00 E=00 H=B4 L=A1 S=F8FE P=0306
I=0306 A=44 F=02 B=41 C=00 D=00 E=00 H=B4 L=A1 S=F8FE P=0308
I=0308 A=44 F=02 B=41 C=49 D=00 E=00 H=B4 L=A1 S=F8FE P=030A
I=030A A=44 F=02 B=41 C=49 D=00 E=00 H=B4 L=A1 S=F8FE P=030B
I=030B A=44 F=02 B=41 C=49 D=00 E=00 H=B4 L=A0 S=F8FE P=030C
I=030C A=44 F=02 B=41 C=49 D=00 E=00 H=B4 L=9F S=F8FE P=030D
I=030D A=41 F=02 B=41 C=49 D=00 E=00 H=B4 L=9F S=F8FE P=030E
I=030E A=41 F=02 B=41 C=49 D=00 E=00 H=B4 L=9F S=F8FE P=030F
I=030F A=41 F=02 B=41 C=49 D=00 E=00 H=B4 L=9E S=F8FE P=0310
I=0310 A=41 F=02 B=41 C=49 D=00 E=00 H=B4 L=9D S=F8FE P=0311
I=0311 A=49 F=02 B=41 C=49 D=00 E=00 H=B4 L=9D S=F8FE P=0312
I=0312 A=49 F=02 B=41 C=49 D=00 E=00 H=B4 L=9D S=F8FE P=0313
I=0313 A=49 F=02 B=41 C=49 D=00 E=00 H=B4 L=9D S=F8FE P=0314

```

The programme can now be altered to demonstrate some more machine code instructions. This time we will add or subtract values to change the accumulator from its original value (44) instead of moving in new values. Instructions are available to change the value in the accumulator by adding or subtracting either immediate data or values already in registers; they are respectively: ADI, SUI, ADD r and SUB r, where the 'r' represents the identity of the register. Here are the original and modified programmes:-

Programme 1	Programme 2 code	Programme 2 comment
Source code	Source	Object
NOP	NOP	00
LXI H,B4A1	LXI H,B4A1	21 A1 B4
MVI A,44	MVI A,44	3E 44
MVI B,41	MVI B,8	06 08
MVI C,49	MOV M,A	77
MOV M,A	DCX H	2B
DCX H	DCX H	2B
DCX H	SUI 3	D6 03
		Subtract 3 from accumulator, result 41 in accumulator
MOV A,B	NOP	00
		Easy way to delete an unwanted instruction from the program.
MOV M,A	MOV M,A	77
DCX H	DCX H	2B
		Letter A to screen

DCX H	DCX H	2B	
MOV A,C	ADD B	80	Add value in reg B to value in A, leave result in A.
MOV M,A	MOV M,A	77	Letter I to screen
NOP	NOP	00	
HLT	HLT	76	

Change the machine code in memory by using the Utility Substitute command, and the new programme should look like this:

```

0300 00 21 A1 B4 3E 44 06 08 77 2B 2B D6 03 00 77 2B
0310 2B 80 77 00 76 00 00 00 00 00 00 00 00 00 00

```

Initialize again and use Utility L300 300 313 to observe the results of the SUI 3 and ADD B instructions on the A register.

Now rewrite the programme to put a space between each letter. Printing the spaces can be combined with the move to the next character position, in a subroutine. That will introduce some more instructions: CALL addr, for which the machine code is CD plus a 2-byte address, in low byte high byte order. CALL is like BASIC's GOSUB and it too must find a Return (mnemonic RET, machine code C9) at the end of the subroutine. The label 'SPACE' will identify the subroutine. The other new instructions are PUSH and POP. PUSH puts the contents of a register pair on to the DAI's Stack. POP takes those same contents from the stack and restores them to the register pair. The Stack is a block of memory (#F800-#F8FF in the DAI) used by the microprocessor as a temporary store. Data is always added to the 'top' of the stack, thus the last item PUSHed on is the first to be POPped off. In this programme register A will insert the space, but we need to save the ASCII value it already contains; therefore we PUSH PSW at the start of the SPACE subroutine and POP PSW at the end, before returning to the main programme. The machine codes are F5 and F1 respectively. We push PSW, not A, because it is a register pair that is being preserved.

#### Programme 3

Source code	Object code	Comment
NOP	00	
LXI H,B4A1	21 A1 B4	
MVI A,44	3E 44	
MVI B,08	06 08	
CALL SPACE	CD 16 03	Call subroutine 'SPACE'
SUI 3	D6 03	Value in reg A becomes 44-3
CALL SPACE	CD 16 03	Prints letter A+space
ADD B	80	Value in A becomes 41+8
CALL SPACE	CD 16 03	Prints letter I+space
NOP	00	
HLT	76	End of main programme
Subroutine to print character in reg A, print a space, adjusting reg pair HL to next character position after each print, then return to main programme:		
SPACE PUSH PSW	F5	Save contents of A on Stack
MOV M,A	77	Send letter in reg A to screen
DCX H	2B	Step to next
DCX H	2B	character position
MVI A,20	3E 20	Load A with ASCII space (#20)
MOV M,A	77	and send it to screen
DCX H	2B	
DCX H	2B	
POP PSW	F1	Restore A register, from Stack
RET	C9	Return to main programme

Here it is in memory:

```
0300 00 21 A1 B4 3E 44 06 08 CD 16 03 D6 03 CD 16 03
0310 80 CD 16 03 00 76 F5 77 2B 2B 3E 20 77 2B 2B F1
0320 C9 00 00 00 etc.
```

The address #316, of the subroutine, had to be found by counting from #300 the number of programme bytes already allocated prior to reaching the label SPACE. It is the address of the first instruction (F5) of the subroutine. That address was then put in the object code wherever there was a CALL SPACE.

Use Z3 then L300 300 314 to examine the main programme and Z3 then L300 316 320 to examine the subroutine. Note how the Programme counter changes on CALLs to and RETURNS from the subroutine, also how the Stack pointer alters with the PUSH and POP.

An assembler can calculate the address of all subroutines and put them in the object code at the correct places. It also accepts instructions in the form MVI 'D', obviating the need to look up ASCII codes. It allows the source programme to be edited, saved on cassette and converted to object code. Then it enters the object code into RAM at the correct location. Can you afford to be without such a useful tool? Two assemblers are listed in the DAInamic software library catalogue: the original DNA and the more recent SPL.

In conclusion, a few more notes on the instructions used:

Arithmetic instructions - ADD r, ADI d, SUB r and SUI d, where r and d are registers or data respectively. Obviously there could be a 'carry' from the addition or a 'borrow' with the subtraction; such possibilities will be explored later, with more arithmetic instructions.

Instructions - DCX H (decrement value in HL) and LXI H, address (load HL with an address, or a 2-byte value).

Similar instructions are available for the other register-pairs and there is also an 'increment' instruction. Here they are, with their op codes in brackets:

DCX B (0B)	INX B (03)	LXI B,addr/value	(01 xx xx)
DCX D (1B)	INX D (13)	LXI D,addr/value	(11 xx xx)
DCX H (2B)	INX H (23)	LXI H,addr/value	(21 xx xx)
DCX SP (3B)	INX SP (33)	LXI SP,addr. *	(31 xx xx)

\* The stack pointer points to an address in the Stack. There is no reason to load it with except another address.

#### KOMPAKT TEKENING

De bedoeling van het programma is een 2 kleuren tekening op een kompakte manier in het geheugen te zetten, zodat het als het weggeschreven wordt op kassette weinig plaats inneemt.

gebruik :

- 1) tekening wegschrijven ( De tekening staat op het scherm.)
  - COLORG 8 X X X (De achtergrondkleur moet grijs zijn !)
  - NEW (Er mag zich geen programma in het geheugen bevinden).
  - POKE #29B,0:POKE #29C,#40:NEW (Verplaats de startpositie van de Basic tot #4000)
  - Laad het basicprogramma "KOMPAKT" en RUN het. Op het scherm verschijnt een teller die gaat tot XMAX : de X waarde van de lijn die nu in het geheugen gezet wordt. Als dit klaar is wordt de tekening bij wijze van test op het scherm gezet. Het begin- en eindadres worden afgedrukt.
  - De tekening kan nu in UT met die adressen weggeschreven worden.
- 2) tekening laden
  - verplaats de startpositie van de Basic tot voorbij de tekening die je gaat laden (meestal is verplaatsen tot #2000 voldoende)
  - MODE X (zet de juiste mode klaar)
  - laad het machinetaalprogramma en de tekening
  - CALLM #300 (roep het programma aan)

```
10 REM : KOMPAKT-TEKENING 85 - 11 - 08 door Mark Stout
20 REM : gebruik : zie bijgevoegd artikel, opm.: na IMP INT
30 REM :
90 REM routine om de tekening in het geheugen te zetten
100 A=#350:POKE A,0:AL=0:FOR I=0 TO XMAX:FOR J=0 TO YMAX
110 D=SCRN(I,J):IF D<>8 GOTO 150
120 IF LIJN=0 GOTO 160
130 POKE A+(AL*2)+1,X:POKE A+(AL*2)+2,J-1
140 AL=AL+1:POKE A,AL:LIJN=0:GOTO 160
150 IF LIJN=0 THEN X=J:LIJN=1
160 NEXT J:A=A+(AL*2)+1:LIJN=0:AL=0:POKE A,0:PRINT I
170 NEXT I:POKE A+1,#FF
200 REM * Basic routine om de tekening op scherm te zetten
210 COLORG 0 8 15 0:A=#350:I=0
220 B=PEEK(A):IF B=#FF THEN 260
230 A=A+1:IF B=0 GOTO 250
240 FOR J=1 TO B:DRAW I,PEEK(A) I,PEEK(A+1) 22:A=A+2:NEXT J
250 I=I+1:GOTO 220
260 PRINT "De tekening zit van #350 tot #";HEX$(A+1)
```

```

0000 @=0350 START EQU 350H
0000 ORG 300H
0300 C5 PUSH B
0301 D5 PUSH D
0302 E5 PUSH H
0303 F5 PUSH PSW
0304 215003 LXI H START
0307 110000 LXI D OH
030A 7E FOR MOV A,M
030B FEFF CPI OFFH
030D CA3603 JZ END
0310 23 INX H
0311 FE00 CPI OH
0313 CA3203 JZ NEXT
0316 47 MOV B,A
0317 97 SUB A
0318 4F ZLIJN MOV C,A
0319 C5 PUSH B
031A 46 MOV B,M
031B 23 INX H
031C 4E MOV C,M
031D E5 PUSH H
031E 62 MOV H,D
031F 6B MOV L,E
0320 D5 PUSH D
0321 3E0F MVI A OFH
0323 EF RST 5
0324 21 DB 21H
0325 D1 POP D
0326 E1 POP H
0327 23 INX H
0328 C1 POP B
0329 79 MOV A,C
032A 3C INR A
032B B8 CMP B
032C CA3203 JZ NEXT
032F C31803 JMP ZLIJN
0332 13 NEXT INX D
0333 C30A03 JMP FOR
0336 F1 END POP PSW
0337 E1 POP H
0338 D1 POP D
0339 C1 POP B
033A C9 RET
033B END

```

```

0300 C5 D5 E5 F5 21 50 03 11 00 00 7E FE FF CA 36 03
0310 23 FE 00 CA 32 03 47 97 4F C5 46 23 4E E5 62 6B
0320 D5 3E 0F EF 21 D1 E1 23 C1 79 3C B8 CA 32 03 C3
0330 18 03 13 C3 0A 03 F1 E1 D1 C1 C9

```

PAGE 01 -- ALFANUMERIEK TOETSEN

```

1000 REM +++ DEMO 0001/0002-05-00 ~ V1.3/1.1
1010 REM +++ alfanumeriek toetsenbord
1015 REM +++ numeriek toetsenbord
1020 REM +++ DAI
1100 PRINT :PRINT "Typ iets in en sluit af met <RETURN> of"
1150 PRINT "'escape' (<---).":PRINT
1200 X=5:REM maximaal aantal in te toetsen tekens
1300 GOSUB 31013:REM alfanumeriek toetsenbord
1400 PRINT :PRINT "XE=";XE;:IF XE=0 THEN PRINT " Afgesloten met return"
1450 IF XE=(-1) THEN PRINT " Afgesloten met 'escape'"
1470 IF X<0 GOTO 2700
1500 PRINT :PRINT "De string is: ";X$
1600 FOR W=1 TO 200:NEXT W:REM wachtlus
1700 X=5
1800 PRINT :PRINT "Voer een getal in en sluit af met"
1850 PRINT "<RETURN> of 'escape' (<---).".
1900 GOSUB 31012:REM numeriek toetsenbord
2000 PRINT :PRINT "XE=";XE;:IF XE=0 THEN PRINT " (numeriek + <RETURN>)"
2020 IF XE=(-1) THEN PRINT " (afgesloten met 'escape')".
2040 IF XE=1 THEN PRINT " (niet-numeriek + <RETURN>)"
2100 PRINT :PRINT "De numerieke invoer was: ";X$
2200 FOR W=1 TO 200:NEXT W
2300 X=-6
2400 PRINT :PRINT "Voer iets in en sluit af met <RETURN>.".
2500 FOR W=1 TO 200:NEXT W
2600 GOTO 1300
2700 PRINT :PRINT "De alfanumerieke invoer werd onderdrukt en was: ";X$
3600 GOTO 1100
17500 END

20000 REM %%%% numeriek toetsenbord %%%%
20006 GOSUB 31013:X=0
20007 IF XL=0 GOTO 20012
20008 FOR XI=1 TO XL
20009 1 IF MID$(X$,XI-1,1)<"0" GOTO 20014
20010 1 IF MID$(X$,XI-1,1)>"9" GOTO 20014
20011 NEXT XI
20012 RETURN
20013 REM niet num
20014 IF X<>44 GOTO 20017:IF MID$(X$,XI-1,1)<>"," GOTO 20017
20015 X$=MID$(X$,0,XI-1)+". "+MID$(X$,XI,LEN(X$)-XI)
20016 X=1:GOTO 20011
20017 XE=1
20018 RETURN

20025 REM %%%% alfanumeriek toetsenbord %%%%
20031 X$="":XE=0:XL=0
20032 XI=GETC:XI=GETC:XI=GETC
20033 XI=GETC:XI$=CHR$(XI):IF XI=0 THEN GOTO 20033:REM get key
20034 IF XI=9 THEN GOSUB 31607:REM tab=screenprint
20035 IF XI=13 AND XL>0 GOTO 20058:REM return
20036 IF XI=18 GOTO 20054:REM escape (<---)
20037 IF XI=19 GOTO 20051:REM -->
20038 IF XI=8 GOTO 20047:REM char del
20039 IF XI<32 OR XI>126 THEN 20032:REM alfanumeriek karakter
20040 IF XL=>ABS(X) THEN GOSUB 31701:GOTO 20032:REM bel
20041 IF XL=LEN(X$) THEN X$=X$+XI$:GOTO 20043
20042 X$=MID$(X$,0,XL)+XI$+MID$(X$,XL+1,LEN(X$)-XL-1)
20043 XL=XL+1

```

# A TASTE OF SUPERFONT...

PAGE 02 -- ALFANUMERIEK TOETSEN

```

20044 IF X>0 THEN PRINT XI$;
20045 GOTO 20032

20046 REM char del
20047 IF XL=0 THEN GOSUB 31701:GOTO 20032:REM bel
20048 XL=XL-1:IF X>0 THEN PRINT XI$;
20049 GOTO 20032

20050 REM -->
20051 IF XL>=LEN(X$) THEN GOSUB 31701:GOTO 20032:REM bel
20052 XL=XL+1:IF X>0 THEN PRINT MID$(X$,XL-1,1);
20053 GOTO 20032

20054 REM 'escape'
20055 X$="":XL=0:XE=-1
20056 RETURN

20057 REM 'return'
20058 IF X>0 THEN PRINT SPC(X-XL);
20059 X$=MID$(X$,0,XL)
20060 RETURN

27005 REM %%%% attentiesignaal leerkracht %%%%
27006 ENVELOPE 0 15:SOUND 0 0 15 0 FREQ(400):WAIT TIME 25:SOUND OFF
27007 RETURN
31012 GOTO 20006:REM numeriek toetsenbord
31013 GOTO 20031:REM alfanumeriek toetsenbord
31607 RETURN:REM scherm naar printer
31701 GOTO 27006:REM attentiesignaal leerkracht
    
```

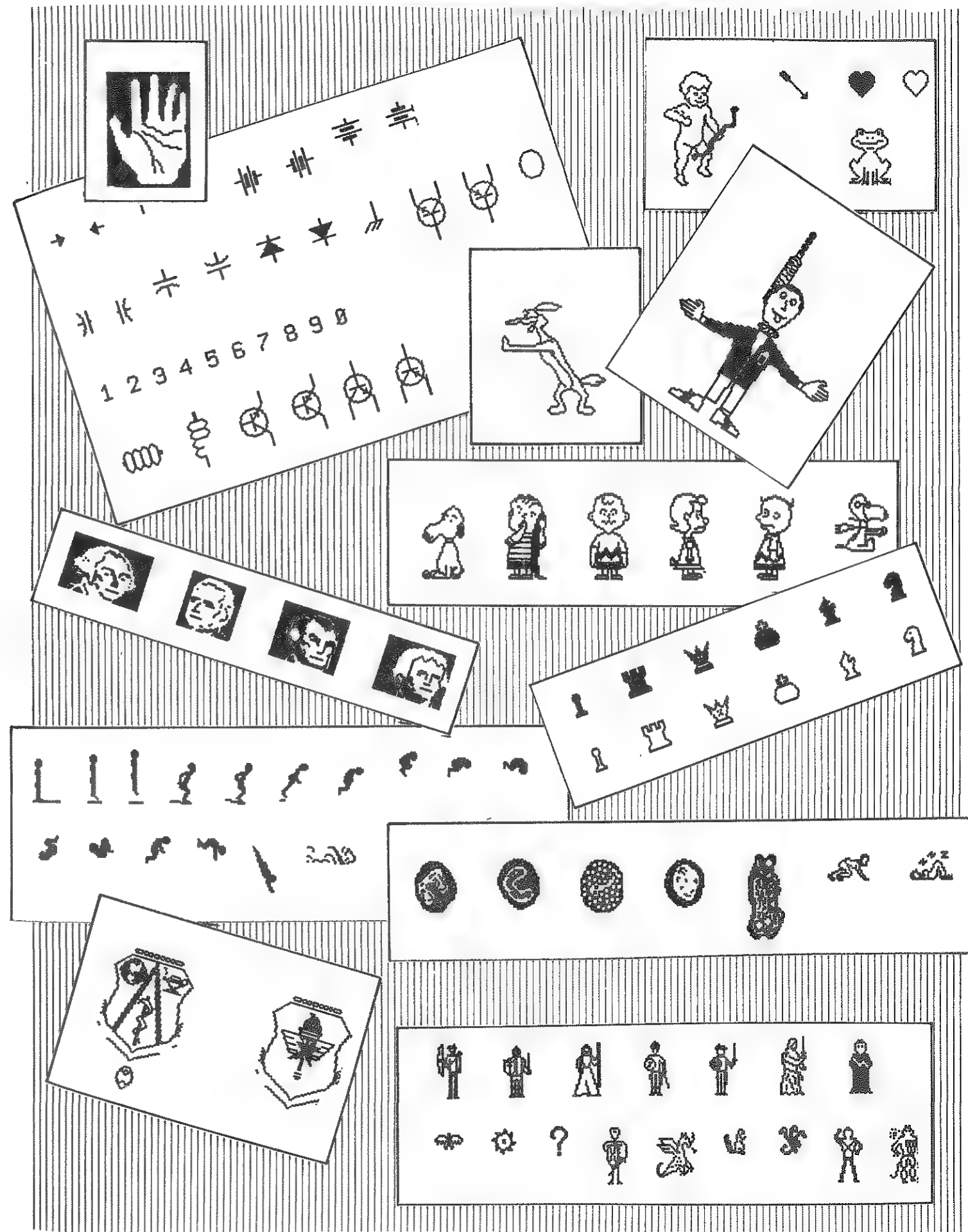
PAGE 01 -- PAUZEROUTINE

```

1000 REM +++ DEMO 0009-05-00 / V1.1
1010 REM +++ pauze routine
1020 REM +++ DAI
1100 INPUT "Typ wat in en druk daarna op <RETURN> ";A$
1200 GOSUB 31601:REM scherm schoon
1300 INPUT "Hoeveel seconden wachttijd ";X:PRINT :REM pauzetijd in seconden
1400 INPUT "Type weer wat, en druk op <RETURN> ";B$:PRINT
1500 GOSUB 31902:REM pauze
1600 GOSUB 31601:REM scherm schoon
1700 PRINT "KLAAR MET WACHTEN."
17500 END

26005 REM %%%% SCHERM SCHOON %%%%
26006 PRINT CHR$(12):RETURN

29030 REM %%%% PAUZE ROUTINE %%%%
29031 IF X<=0 OR X>=30 THEN X=1
29032 XT=X*50:WAIT TIME XT:RETURN
31601 GOTO 26006:REM scherm schoon
31902 GOTO 29031:REM pauze
    
```







\*NEW  
\*POKE 309,2

This method is especially suitable for separating part of a program so that it can be used in another program. It can only be used when the part to be saved is a continuous block of lines.

3. The "Monte Carlo method": If after pressing RETURN following the EDIT 1000-2000 you very quickly press BREAK twice it is possible for a section of the program between 1000 and 2000 to disappear. The rest can then be removed by method 1. This method has support among those who like trying to be faster than the computer, that is, folk in the "Games & Strategy" rather than the "Serious Programming" category.

4. To eliminate for example lines 70 - 120 give (in the command mode):

\*EDIT 70-120; EDIT n (where n is an arbitrary number)

Lines 70 to 120 appear on the screen in Edit mode. Then press <BREAK> <SPACE>. Line n appears on the screen. Press <BREAK> <BREAK>.

The method seems to work without fail but I cannot explain how. I have tried to include this trick in a Basic program but it then does not appear to work in all cases. (with help from Mr Dufour's program in DAInamic 16, p200).

When you have removed a large part of a program it is advisable to tidy up the symbol table, as it will still contain the variables from the deleted section of program:

\*CLEAR xxxxx (sufficient to hold the whole program)

\*EDIT  
<BREAK> <BREAK>

\*NEW  
\*POKE 309,2

Jos Vandenberg

SORTING by simple comparison in machine language (from DAInamic 19, p370) Copyright Christian POELS - 2/6/83

The program below sorts a series of items from a set or a whole table. The sorting can be performed on the complete table or only on part of it. The number of items to be sorted must be stored at the addresses #300 and #301 and the call to the routine is made in the following way:

CALLM #306,NAME OF TABLE (INDEX OF FIRST ITEM OF THE SERIES)

If a table has to be sorted, we need to know that all its items are stored in the memory, line by line. Thus, the items too will be sorted, line by line.

To try out the example, type in the machine code and the basic, and record them on tape. RESET. Alter the pointers too!

POKE #29B,0; POKE #29C,4; CLEAR 1000. Load the machine code, NEW, Load the basic, RUN. In this example, the sorting is done for a set of 256 items whose values are all between -500 and 500. The sorting in itself takes about 6 seconds. Here are some examples of time taken:

No. of items:	Time (sec.):
100	1
200	4
300	8
400	15
500	23
600	32
1000	89
1500	209

The time taken obviously corresponds to the number of items but in practice this delay is generally acceptable. Of course, to sort thousands of numbers, another method, suited to large series, is preferable.

```

10 REM CHRISTIAN POELS - 26/6/83
20 REM EXAMPLE OF MAIN PROGRAM
30 CLEAR 2000; DIM A(255)
40 PRINT CHR$(12);"ITEMS TO BE SORTED!"; PRINT
50 FOR I=0 TO 255: A(I)=RND(1000)-500: PRINT A(I);; NEXT
60 PRINT: PRINT: PRINT: PRINT "SORTING IS TAKING PLACE!"
70 N=256; REM N=NUMBER OF ITEMS TO BE SORTED
80 E=0; REM SORTING N ITEMS, STARTING FROM A(E)
90 GOSUB 60000; REM SORTING
100 PRINT: PRINT: PRINT "ITEMS SORTED!"; PRINT
110 FOR I=0 TO 255: PRINT A(I);; NEXT: PRINT
120 END
60000 REM CALLING MACHINE CODE ROUTINE
60010 POKE #300,N MOD 256; POKE #301,N/256
60020 CALLM #306,A(E)
60030 RETURN
    
```

300	00	00	00	00	00	00	00	C5	D5	E5	F5	EB	2A	00	03	44	4D
310	0B	EB	54	5D	C5	E5	CD	6B	03	23	23	23	23	E5	23	23	
320	23	3A	05	03	96	2B	3A	04	03	9E	2B	3A	03	03	9E	2B	
330	3A	02	03	9E	FA	3C	03	54	5D	CD	6B	03	0B	E1	78	B1	
340	C2	19	03	E1	1A	4F	7E	71	12	23	13	1A	4F	7E	71	12	
350	23	13	1A	4F	7E	71	12	23	13	1A	4F	7E	71	12	23	C1	
360	0B	78	B1	C2	12	03	F1	E1	D1	C1	C9	7E	32	02	03	23	
370	7E	32	03	03	23	7E	32	04	03	23	7E	32	05	03	2B	2B	
380	2B	C9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

PAGE 09 -- ADC CONVERTOR

```

30280 PRINT "DRUK OP DE SPATIEBALK OM
VERDER TE GAAN.."
30285 IF SP1=1.0 THEN CURSOR 25,2:PRINT
SPC(10):CURSOR 25,2:PRINT A!;" V.":
CURSOR 25,2
30290 SP1=1
30300 G=GETC: IF G<>32 GOTO 30020
30350 POKE #75,95:DW=A:GOSUB 20190
30500 RETURN
    
```

cont. from p. 329



Did you ever thought to program your DAI in PASCAL, FORTRAN, C or COBOL ?

What's about having a fast floppy system for your DAI PC with 2,4 MB of storage ?

# DAIstar is your solution!

*DAIstar is an expansion system that makes your DAI a professional PC. You simply connect it to your DAI - and just you have two computers in one !*

## THE FLOPPY MODE

You work with your DAI as usual. A small disc-operating-system (DOS) is loaded into the memory at power on. It extends the DAI-BASIC by 32 new instructions that are really programmable and allow you a comfortable filehandling. The DOS doesn't only support loading and saving of all kinds of programs - it offers you the possibility to work with pictures, text files, ML-programs and random access files with variable record length.

Since the disk format is the same as in the CP/M mode, you can exchange pictures, textfiles and ML-programs between the two modes.

To be as compatible as possible with the existing DAI software, we have made the DOS very small ( 1,8 KB ). And an even smaller and moveable TINYDOS allows you to run nearly all existing DAI software without modifications.

The DOS supports a cassette recorder, a MDCR and allows the use of DCE-bus peripherals.

To meet the needs of assembly language programmers we integrated an assembly language interface that allows the use of DOS commands from assembly language. Sector read and write commands allow the direct disc access.

Last not least you won't have storage problems any more with your disk storage of up to 2,4 MBytes formatted.

## THE CP/M OPERATING MODE

Now you have full access to *TURBO PASCAL, FORTRAN, C, COBOL, MBASIC, FORTH, ADA* and many other programming languages. *WORDSTAR, MAILMERGE, SPELLSTAR, DBASE, MULTIPLAN* and all other CP/M programs will run on the DAIstar system without any restrictions....

All these programs are running up to 5 times faster than on the DAI PC.

This is done by the DAIstar's integrated computerboard with a fast Z80 CPU, 64KB of RAM and a lot of other features.

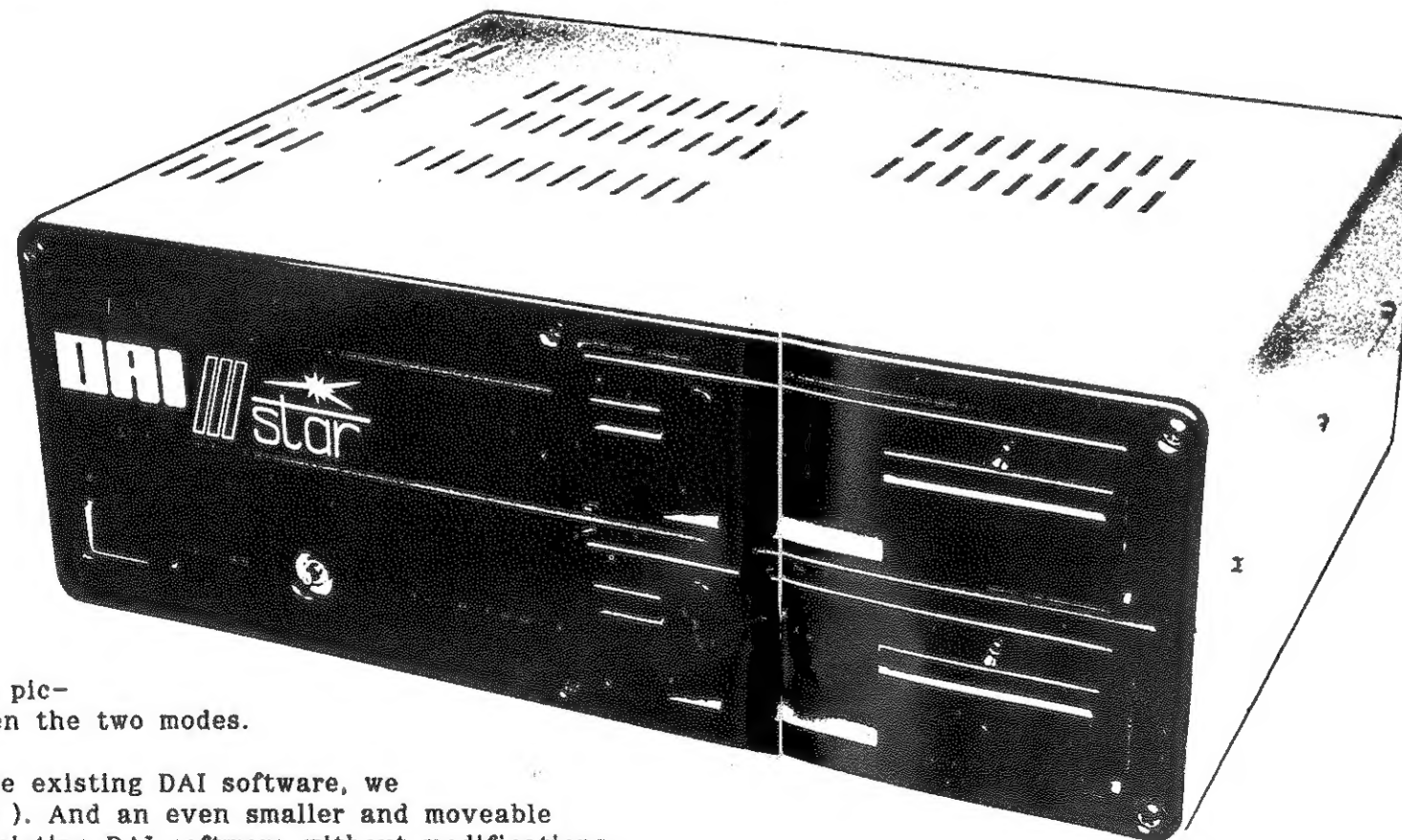
At the same time your DAI PC is working as a colour graphics terminal that even supports sound capabilities. In consequence of the parallel processing drawing complex pictures is much faster than on a single DAI.

This parallel processing even allows to load a ML-program into the DAI and to run it in the background.

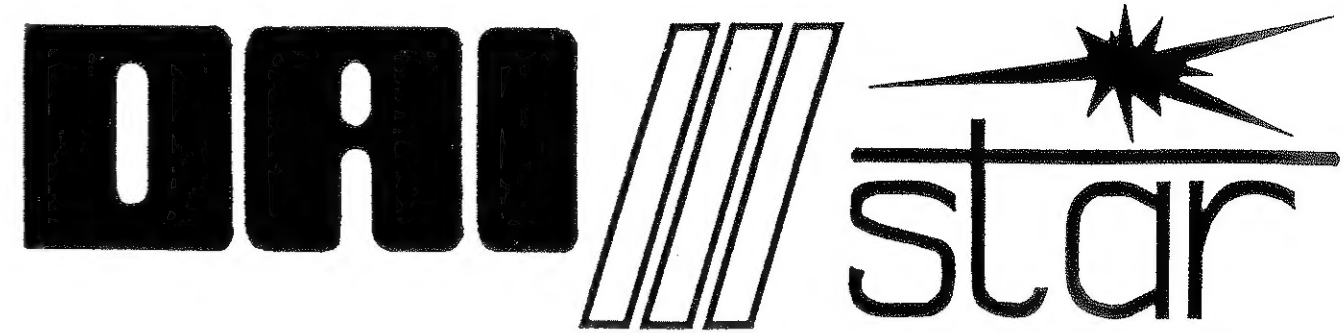
New instructions allow you to draw lines, circles, circle sections, polygons and to fill whole areas. It's easy to draw complex graphics and complete them with text in different fonts, scales and colours. That's even possible at a resolution of 512\*240 pixels. And when your picture is finished you can store it onto disc with a single command.

The text mode is supported by a virtual screen that can be configured to a logical size of up to 120\*120 characters. This method allows you to use every screen size you like. The visible window moves across it.

Getting CP/M software for the system is quite easy since DAIstar is able to read the wide spread *OSBORNE I* disc format.



# Instructions



## THE CP/M MODE:

### graphics instructions    graphic-text instructions

MODE 0-8	GCUR x,y
CGORC f1 f2 f3 f4	GCOL f
COLORT f1 f2 f3 f4	GWIN x1,x2
SCREEN x,y	GSPACE dx,dy
DOT x,y c	GSCALE sx,sy
DRAW x,y x,y c	GSTAND
FILL x,y x,y c	GTEXT string
POLY x,y x,y .. c	GOUT 1
CIRCLE x,y r a,e c	GOUT 0
AREA x,y r f	GTLOAD t,p filename
CLEAR	GDIST v
PSAVE addr	
PSAVE filename	and many others...
FLOAD addr	
FLOAD filename	

- virtual screen between 1<sup>st</sup> and 120<sup>th</sup>120 characters
- virtual screen administrates second colour set
- autostart capability
- instructions useable from all programming languages
- all standard CP/M 80 programs are running
- fast disc access and large storage capacity

## THE FLOPPY MODE:

### LOAD & SAVE    general commands

LOAD "Name"	DIR
SAVE "Name"	DIRA
LOADA Arr "Name"	DIRB
SAVEA Arr "Name"	CASS
R offset Name	FLOPPY
W begin length Name	NOCR
DLOAD "Name"	A:
DSAVE "Name"	B:
FLOAD "Name"	ERA "Name"
PSAVE "Name"	REN "Name1", "Name2"
/Name	COPY "Name1", "Name2"
	BACKUP
	READS addr blocknr
	WRITES addr blocknr

### textfiles    random-access-files

CREATE idt,D "Name"	CREATE idr,recien "Name"
OPEN idt intvar	OPEN idr intvar
CLOSE idt intvar	CLOSE idr intvar
EOF idt intvar	EOF idr intvar
PRINT arglist	PUT idr;varlist
INPUT varlist	GET idr;varlist
CIN channel	SEEK idr position
COUT channel	POS idr intvar
	SIZE idr intvar

- 4 open random-files at the same time
- 4 open textfiles at the same time
- autostart capability
- assembly language interface
- DOS length ca. 1,8 KBytes
- TINYDOS length ca. 300 Bytes

## HARDWARE:

- dimensions (w)35cm \* (d)29cm \* (h)13cm
- power requirements 230V +/-15%, 50/60 Hz
- power supply 5V 5A, 12V 3A, -5V, -12V
- DCE-Bus interface board with transfer rates up to 80 KBytes/s
- computerboard with 4 (6) MHz Z80 CPU, 64 KB of RAM, floppy controller with data separator for higher reliability
- up to two drives with each 800 KByte of formatted storage, optionally switchable to 1280 KBytes each

DAIstar, K.Peter & R.Schillingner  
Neubriach 13, 7982 Balenfurt, GERMANY

Intrested ? Then please request our free informationsheet with a detailed description, prices and options.

This document was written with WORDSTAR on a DAI with DAIstar

brings your DAI up to date ...

