DAI NAMIC

19

tweemaandelijks tijdschrift november - december 1983

Jaap Delvoye

personal computer users club

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

International

COLOFON

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

DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

Wilfried Hermans
Mottaart 20
3170 Herselt
Tel. 014/54 59 74

Kredietbank Herselt
nr. 401-1009701-46
BTW : 420.840.834

Lidgelden / Subscriptions

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

Voor Nederland :

GIRO : 4083817
t.n.v. J.F. van Dunne'
Hoflaan 70
3062 JJ ROTTERDAM
Tel. : (010) 144802

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

DAINAMIC

PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

Herselt, december '83

Beste leden,

Met nummer 19 sluiten we onze jaargang '83. Een terugblik op de voorbije activiteiten gebeurde naar aanleiding van ons 3-jarig bestaan, dat hoeft dus niet meer. Een woordje van dank is echter altijd op zijn plaats : dank aan al diegenen die hun bijdragen leveren aan het DAInamic-gebeuren : de vaste kern (die ondertussen al aardig uitgebreid is), de vaste correspondenten en al die leden die af en toe eens een artikeltje plegen ... Samen hebben we dit jaar weer gezorgd voor 414 pagina's informatie over en programma's voor DAIpc. We wensen iedereen een productief en creatief '84 !

Mededelingen :

Ondanks de stijgende onkosten blijft de contributie dezelfde : 900 fr voor Benelux, 1000 fr voor Europa en 1400 fr voor luchtpost. Dit zijn de prijzen voor hernieuwing vóór 1 feb 1984. Betalingen na deze datum worden beschouwd als nieuwe abonnementen, en bedragen dan respectievelijk 1000,1100 en 1500 Bfr. We hopen dat deze maatregel er zal toe bijdra-gen dat iedereen tijdig zijn abonnement vernieuwt,wat het werk van onze adminstratie zeker zal vereenvoudigen.

Betalingen voor software dienen voortaan te gebeuren op volgend nummer : 401-1009701-46 van Kredietbank Herselt, het banknummer voor de contributie is nog steeds : 230-0045353-74 van Generale Bankmaatschappij Leuven. De voordeligste manier van betalen (voor het buitenland) is echter per internationale postal order,dit gebeurt nl zonder extra onkosten voor U en voor ons...

We hebben al een paar weken de beschikking over een KEN-DOS systeem : het is onvoorstelbaar dat één man deze hard-& software heeft kunnen ontwikkelen in zijn vrije tijd, maar de prestaties zijn ronduit geweldig: de snelheid is fenomenaal (een mode 5 plaatje wordt geladen in minder dan 1,5 seconde),compatibiliteit met audio en DCR is compleet ... alle bestaande programma's kunnen zonder meer op schijf gezet worden en zijn onmiddelijk te gebruiken . Een uitvoerig testrapport volgt in de komende editie, we hopen dat we dan eveneens het INDATA floppy systeem kunnen bespreken.

we wensen iedereen prettige feestdagen, tot in '84...

dear members,

Thanks to your coöperation, we could bring 414 pages of information and programs in our 1983-magazines. We hope that creativity and efforts will be the same in 1984.
The contribution will be the same in 1984 if you pay before 1 feb 1984 : this to make the life of our membership administrator a little bit easier. Please note the new banc number for software orders : 401-1009701-46 of Kredietbank in Herselt, the number for contributions is still the same : 230-0045353-74 of Generale Bankmaatschappij Leuven. The cheapest way of transferring money however is by international postal money order!

During a few weeks we have been working with KEN-DOS system . A complete testreport will be in our next issue. We appreciate the new system very much, especially the enormous speed and the total compatibility with audio and DCR-cassettes !

we wish you and your family a happy 1984 !

Wilfried Hermans

NEWSLETTER 19

INHOUD — CONTENTS

DAInamic subscription rates :

Benelux : 1000 Bfr (renewal before 1 feb : 900 Bfr)
Europe : 1100 Bfr (renewal before 1 feb :1000 Bfr)
Outside Europe 1500 Bfr (" " " " 1400 Bfr)
(Air Mail)

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

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

BITS & BYTES

see p. 355

author : B.Van Rompaey

A collection of didactical and demonstration programs about computer hard- & software. As seen on Belgium television in the computer-course "CHIPS".

Programs : introduction - parts of a computer - from keyboard to central memory - parts of an instruction - AND-gate - manipulating 2 bytes - arithm. unit : decimal - arithm. unit : binary - addition - control unit - instruction - magnetic tape - developments - structures - advantages of higher languages.

All programs are in graphic modes, text (very little) is in dutch.

price : audio : 750 fr
DCR : 900 fr

CHARACTER GENERATOR

see p. 356-357

author : Tomislav Mikulic

A very special program for VIDEO and TITLE-freaks, supplying 3 fonts (Helvetica 12 , Helvetica 22 and PICFONT), they can be entered very easy from keyboard.Text is formatted proportional, with full graphics overlay.Various shadows and edges are possible. By manipulating COLORG and screen copy routines, many extra fonts can be achieved.CHARACTER GENERATOR has AUTO-mode for dynamic (and DAInamic) generation of titles.

price : audio : 1750 fr
DCR : 1900 fr

MATH' FUN

see p. 358

author : W.Hermans

Let your children learn while they play games ...

1/ math-robot : You define difficulty and type of exercices : robot will laugh when the answer is good, he will become angry (with red ears) if answer is bad.
for children from 7-12 (+ , - , X exercices).
2/ math-bingo : a competition game for 2 children, with bonus points on good answers. If childs fails 3 times to give the right answer, visual representations of the exercice are displayed to give a helping hand...
for children from 7-12
3/ color and shape with LOGI-blocks. A pre-math program to learn to combine different aspects of things. Various shapes, difficulty is changing all the time.
for children from 5-9

price : audio : 1000 fr
DCR : 1150 fr

available soon :

FWP : the ultimate wordprocessor for DAIpc (see next issue)
PACMAN : the real and only PACMAN-game with all the specifications of the original arcade-game.(see next issue)

diDAIsoft

Om onze leden sneller en nog betere programma's te kunnen aanbieden hebben we diDAIsoft-vakcoordinatoren aangesteld. Hun opdracht bestaat erin de hen toegestuurde programma's op een verzameltape te brengen en een uitgebreide handleiding samen te stellen.
Bezit U didactische programma's voor het secundair of hoger onderwijs stuur ze aan onze coordinatoren. Op deze wijze komen ze via de DAInamic-bibliotheek ter beschikking van alle DAInamic-leden.
Net zoals voor alle andere DAInamic-programmatuur geldt ook hier de regel dat U gratis de verzameltape krijgt waarop één van uw programma's werd opgenomen.
Zo bekom je in ruil voor één programma, er zeven, acht... andere.
Elke andere vorm van vergoeding moet met Bruno Van Rompaey worden besproken.

DiDAIsoft werd als volgt structureel uitgebouwd:

algemene coordinatie

Bruno Van Rompaey
Bovenbosstraat 4
3044 HAASRODE
016/ 461085

vakcoordinatoren

WISKUNDE
Jos De Moor
Elf-Novemberlaan 5
3500 HASSELT
011/ 229815
alleen programma's
op audiocassette

SCHEIKUNDE-BIOLOGIE-ECONOMIE
Jos Vandebergh
Goetsbloetsstraat 33
3500 HASSELT
011/ 253597
audio of
DCR-cassette

INFORMATICA
Bruno Van Rompaey
Bovenbosstraat 4
3044 HAASRODE
016/ 461085
audio of
DCR-casseette

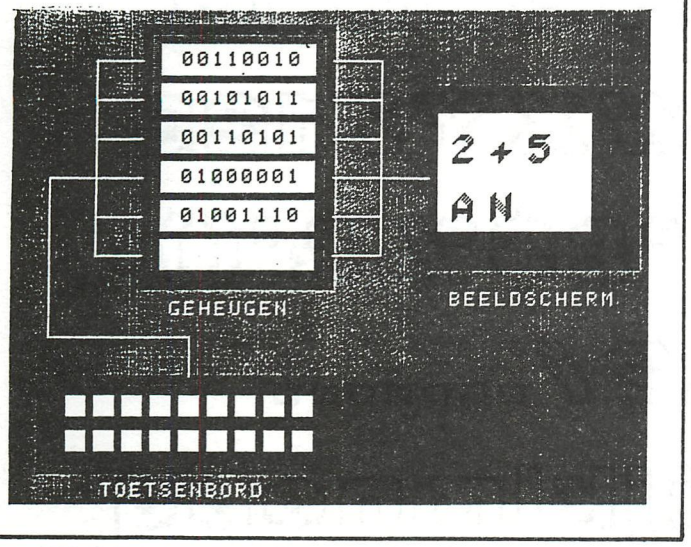
AARDRIJKSKUNDE
Marc Antrop
Beekstraat 29
9920 LOVENDEGEM
091/ 728561
audio/DCR

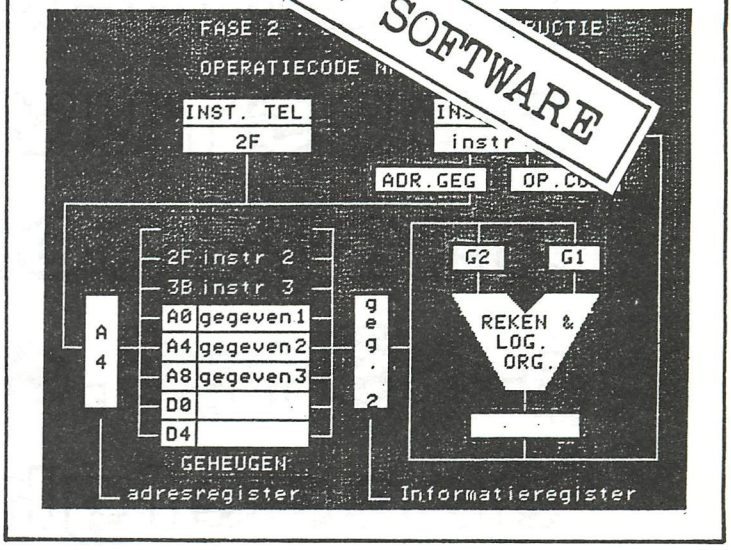
NATUURKUNDE
Danny Frère
Festraetsstraat 40 b 6
3800 ST-TRUIDEN
011/ 685826
audio/DCR

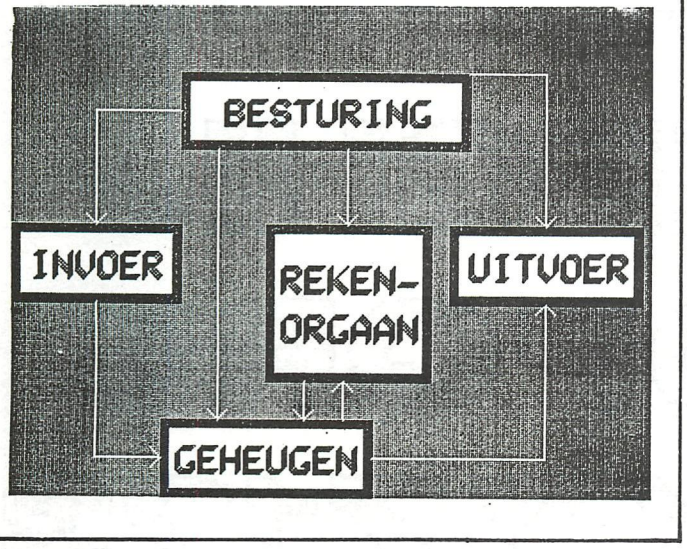
TALEN
Leo Vandijck
Mgr Koningsstraat 38 B
3680 MAASEIK
alleen programma's
audiocassette

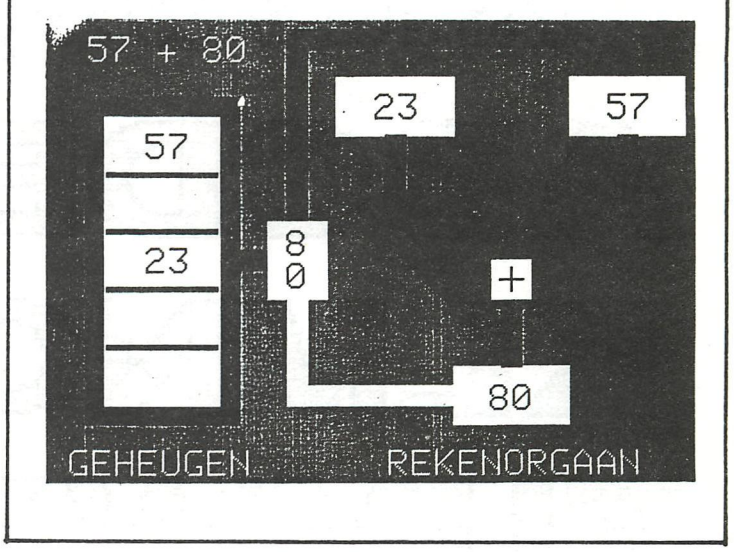
BITS & BYTES

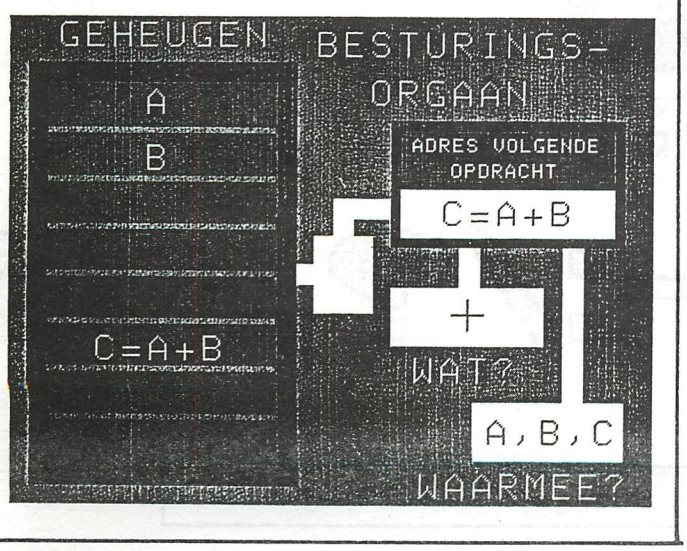
NEW SOFTWARE

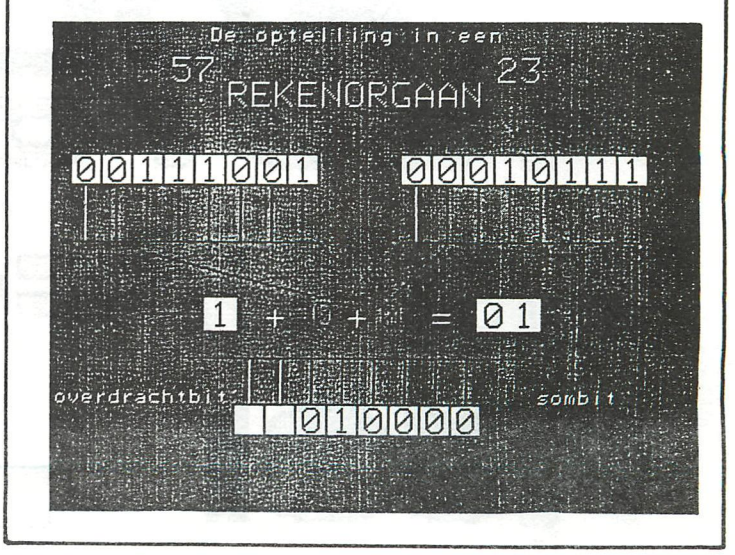












CHARACTER

HELVETICA 12 OUTLINE

BOLD

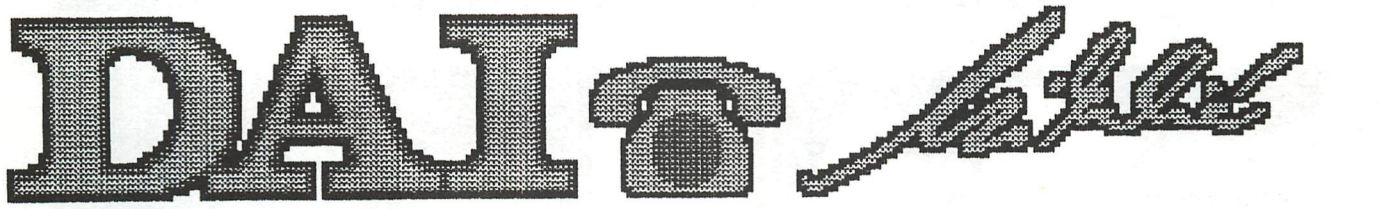
HELVETICA 22

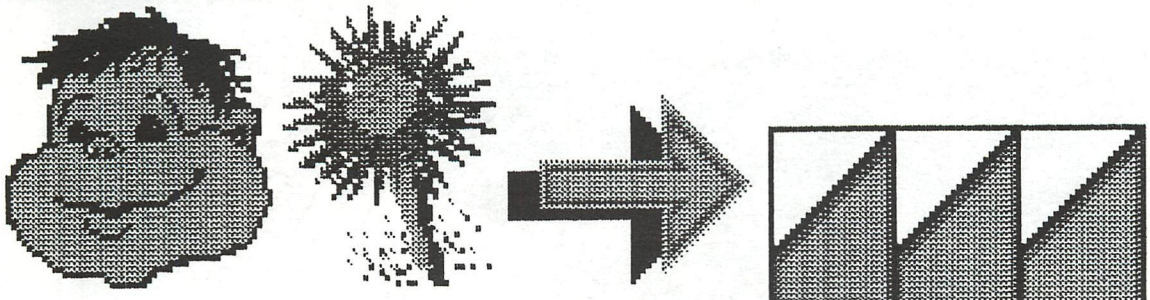
GENERATOR

NEW SOFTWARE

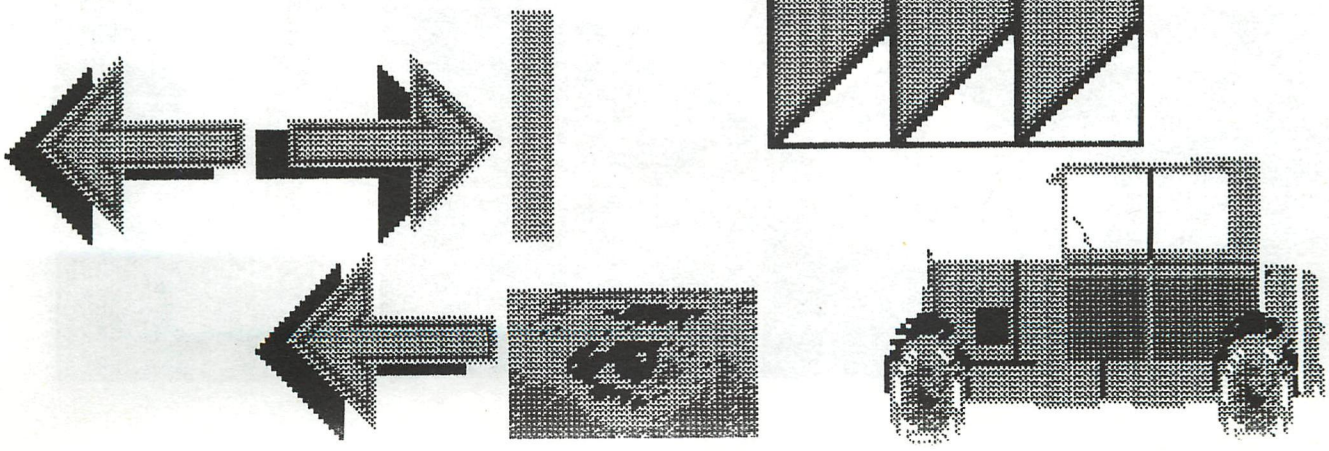
ABCDEFGHIJLMNO
PQRSTUVWXYZ123
4567890!"£$%&'():
:-sš+;ć,č.?/.....
abcdefghijklmnopq
rstuvwxyz

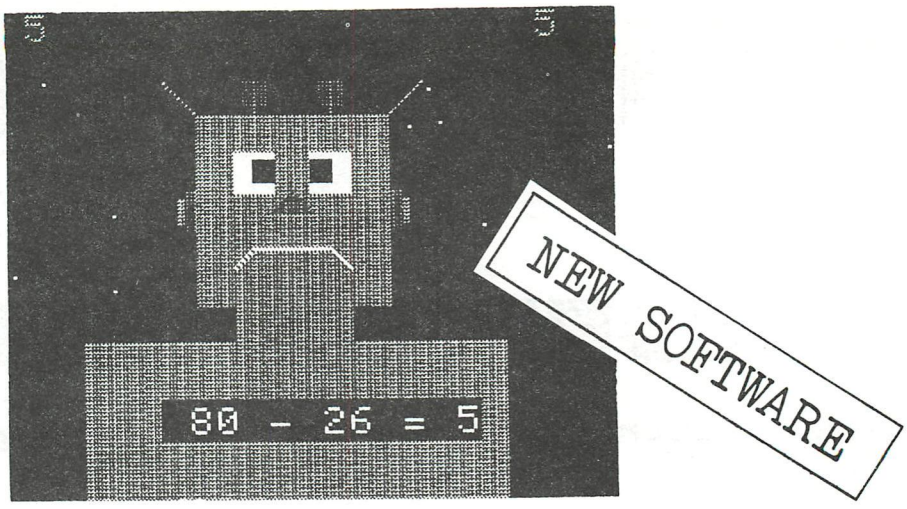
HELVETICA 12

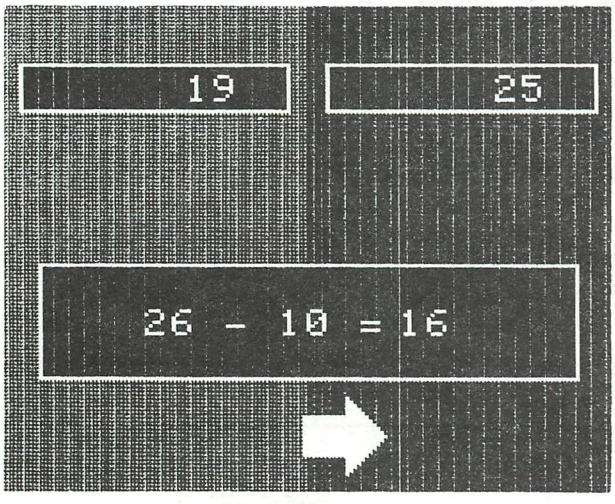


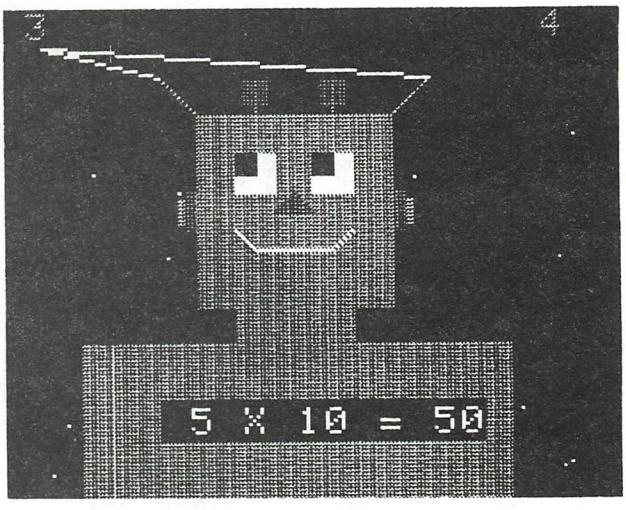


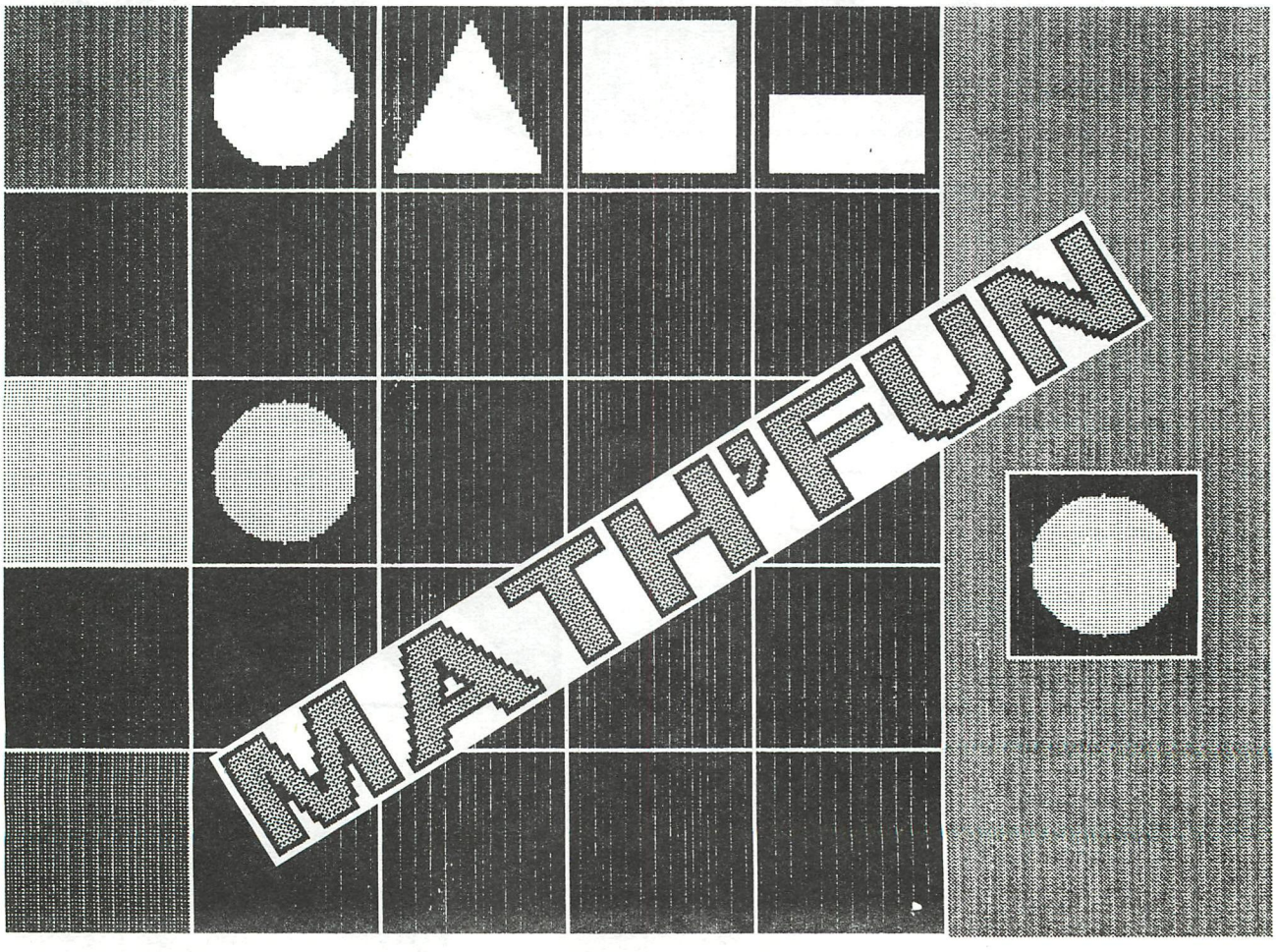
PICFONT




80 - 26 = 5
NEW SOFTWARE


19
25
26 - 10 = 16


5 X 10 = 50


MATHFUN

programmeertechnieken

Voordat ik begin aan het onderwerp van deze keer eerst nog een opmerking over het programma van Koert dat ik de vorige keer besproken heb. Koert had mij gevraagd hoe het kwam dat het programma aanzienlijk vertraagde na een keer het hele veld schoon te maken. De verklaring hiervoor is vrij simpel : De score is het enige dat verandert na het schoonmaken van een heel veld, dus moet de oorzaak van de vertraging daarin te vinden zijn. Het afdrukken van die score is er inderdaad de oorzaak van dat de speelsnelheid aanzienlijk vermindert als die score een groot getal is. Stop het programma maar eens met 'BREAK' en verander dan de score maar eens in een erg groot getal en geef dan 'CONT'.Het is moeilijk hiervoor een remedie te geven. De score verdelen in een normale score en een zgn bonusscore is misschien nog het best aan te bevelen. Het helpt niet om de score in twee delen af te drukken namelijk het deel van de duizenden en de rest, daar de winst die zit in de slechts zelden veranderende duizendtallen wordt tenietgedaan door de test hierop. Daarbij is het tegelijk op-en aflopen van de score een extra probleem waardoor ikzelf althans geen voor mij bevredigende oplossing kon vinden. Ik houd mij aanbevolen voor een goed idee.

Ik onderbreek mijzelf hier even om iets te zeggen over het ontstaan van dit artikel. Het onderwerp is steeds kleiner geworden. Ik begon met de functies en operatoren al schrijvend kwam ik er achter dat er zoveel te zeggen was dat ik mij wilde beperken tot de operatoren. Maar ook dit bleek zo uitgebreid te behandelen dat ik zelfs dit onderwerp in twee delen wil aanpakken. Onder meer de opmerkingen van Nico Looije brachten weer zoveel zaken, die het vermelden waard zijn, aan het licht dat een artikel te lang geworden zou zijn.

Nu het onderwerp van deze keer; ik wilde wat nader ingaan op verschillende rekenkundige en logische operatoren die de DAI-BASIC kent, later zullen de functies ook nog eens aan bod komen. Er zijn er die wel wat nadere toelichting behoeven. Weet U soms iets bijzonders te vertellen over een van de operatoren of functies meldt mij dit dan alstublieft. Ik zal het later weer doorgeven aan alle DAInamic-lezers.

Om te beginnen de MOD operator. Iemand met een mathematische achtergrond zal deze operator wel bekend zijn maar het is mij gebleken dat velen het gewoon onbekend is. En wat erger is: het handboek geeft vrijwel geen uitleg over de werking. Simpel gezegd komt het op eenvoudig lagere school rekenen neer. Weet U zich nog sommetjes als 12 gedeeld door 3 te herinneren ? En toen we dat voldoende geoefend hadden we problemen kregen voorgeschoteld als 13 gedeeld door 3 ? En dat we toen zeiden "Dat kan niet !!" ? De leerkracht was dat met ons eens en vertelde ons dat als we 13 appels onder 3 jongens moesten verdelen het duidelijk was dat ieder recht had op 4 appels en dat er dan nog 1 overbleef. Wiskundiger gezegd: 13 gedeeld door 3 is gelijk aan 4 rest 1 of korter gezegd 13:3=4 r 1 De MOD operator geeft U nu de rest bij deling van een getal m door een getal n. Ik geef wat voorbeelden dan wordt het vast wel duidelijk.

lagere school leerling	DAI programmeur nu
16:5=3 rest 1	16 MOD 5=1
22:4=5 rest 2	22 MOD 4=2
36:6=6 rest 0	36 MOD 6=0
87:10=8 rest 7	87 MOD 10=7

Goed zult U misschien zeggen het is me nu wel duidelijk wat de MOD operator doet maar wat doe ik ermee ? Of anders gezegd hoe kan ik hem zinvol gebruiken in mijn programma's ? Ik geef U een aantal voorbeelden waarin U de MOD handig kunt gebruiken.

I) U wilt de X-waarde van een punt aan een machinetaal programma doorgeven. Deze X-waarde kan vanaf MODE 5 een getal zijn dat groter is dan 255 en met POKE I,J kunt U maar een byte (maximale inhoud 255) wegzetten. De X-waarde zal dan ook gepoked moeten worden in twee bytes. De ene byte geeft dan aan of X groter is dan 255 (in dat geval wordt hij dus 1) en de andere byte geeft aan hoeveel groter dan 255 of 0. De eerste byte poken we met X/256, (inderdaad 256 en niet 255) omdat dat 0 is als X kleiner is dan 256 en 1 als X groter is dan 255 (en natuurlijk kleiner dan 512). De tweede byte poken we met X MOD 256. Hetzelfde resultaat kunnen we trouwens ook bereiken door bijvoorbeeld :

```
POKE I,X/256:POKE I-1,X-PEEK(I)*256
```

of

```
IF X>255 THEN POKE I,1:X=X-256 POKE I-1,X
```

II) U wilt zorgen dat een punt niet van het scherm afgaat. Met de MOD XMAX respectievelijk MOD YMAX zorgt u hier op een heel simpele manier voor. Tik het volgende programmaatje maar eens in.

```
10 MODE 2:XM=XMAX+1:YM=YMAX+1
20 DOT X,Y 21:X=(X+1) MOD XM:Y=(Y+1) MOD YM:GOTO 20
```

Heeft U gezien dat ik MOD XM doe terwijl XM=XMAX+1 ? Dit is logisch daar de coordinaten best XMAX resp. YMAX mogen zijn maar de MOD operator levert als grootst mogelijk resultaat altijd XM-1 dus hier precies XMAX.

III) U wilt op een simpele manier snel de richting laten bepalen door de cursortoetsen. De ASCII-codes van deze toetsen zijn 16,17,18,19. We nemen nu in ons programma de volgende regel op:

```
40 ON GETC MOD 4 GOTO 60,70,80
```

Is GETC dan 16 gaan we naar de volgende regel (50) bij 17 naar regel 60, bij 18 naar regel 70 en bij 19 naar regel 80. Het programma zal bij deze aanpak ook op andere toetsen reageren maar niet stuklopen of een foutmelding geven. Toch kleeft er een belangrijke fout aan boverstaand programmastukje Niet in de MOD maar in de GETC. Als er niets ingedrukt is levert GETC 0 op en zal naar de volgende instructie gesprongen worden. Beter is dan ook:

```
40 H=GETC:IF H=0 GOTO 40:ON H MOD 4 GOTO 60,70,80
```

of

```
40 ON GETC MOD 5 GOTO 60,70,80,90:GOTO 40
```

Nu de 'technische' werking van de MOD operator. Bij A MOD B wordt het getal A integer gedeeld door B. Dit integer delen houdt in dat de eventuele cijfers achter de komma worden weggelaten, we spreken van afkappen en het handboek van 'truncated'. Zeg dat de uitkomst van de deling Q is. Dan wordt vervolgens A verminderd met B * Q. We kunnen hieruit een aantal zaken leren. Als we MOD 0 doen krijgen we de foutmelding 'DIVISION BY ZERO'. We kunnen indien we dit wensen B negatief laten zijn. De resultaten zijn hetzelfde als bij positieve B daar Q negatief wordt en B * Q dus weer positief. Ook A kan negatief zijn en nu krijgen we wel andere antwoorden: het resultaat ligt nu tussen 0 en -B aangenomen dat B positief is. Ook dit is weer gemakkelijk in te zien als we de berekening van de DAI volgen. Een volgend probleem zult U krijgen als U, nieuwsgierig geworden naar het zojuist gelezene, nu gaat intikken PRINT 7 MOD -3 en dan tot uw verbazing SYNTAX ERROR krijgt. En dat terwijl ik geschreven had dat B negatief mocht zijn. We hebben dan beide gelijk. B mag negatief zijn maar niet een aftrekking ! Het '-' symbool heeft in de wiskunde twee mogelijke betekenissen.

De zogenaamde unaire min werkt op een waarde en geeft aan dat het tegengestel-

de van het er achterstaande bedoeld wordt. Simpeler gezegd deze min maakt normale getallen negatief en reeds negatieve waarden juist weer positief.

De zogenaamde binaire min werkt op twee waarden en is als zodanig een echte operator, dwz een bewerking tussen twee getallen. Deze operator kent iedereen als aftrekken.

De DAI BASIC V1.0 had hier wat probleempjes mee en ook in BASIC V1.1 zijn die nog niet geheel opgelost. De remedie is echter vrij eenvoudig. Levert -3 of -X problemen op dan vervangt U de -3 door (-3) en de -X door (-X) of U kent de waarde eerst toe aan een nieuwe variabele. Tikt U nu maar in: ?5 MOD (-2) en U bent uw probleem kwijt.

Ook de '+' kent een unaire en een binaire versie met de betekenissen positief respectievelijk optellen maar hier zijn minder vaak problemen mee omdat de unaire plus meestal weggelaten wordt.

Nog een opmerking voordat ik MOD verlaat. Het is een integer operator bij DAI en kan wel gebruikt worden bij floating point getallen maar zal wel verkeerde uitkomsten geven als we niet opletten. Ik had zelf eens opgemerkt dat functies als SIN trager worden naarmate het argument groter wordt. Ik dacht toen heel slim via een omweg het toch weer sneller te doen door het argument MOD TPI te nemen met TPI=PI+PI. Inderdaad werkte dit veel sneller maar jammer genoeg niet goed. TPI=6.28.... en werd dus bij de MOD veranderd in 6 zodat de uitkomsten niet meer juist waren. Toch kunnen we het idee wel gebruiken al praat ik niet meer over tijdwinst. In plaats van SIN(X) nemen we SIN(X/F MOD G*F) met F=2*PI/G en G een groot getal zoals bv tien miljoen. Een ieder die een beetje de sinus waarden kent en wel eens de sinus van een groot getal aan de DAI gevraagd heeft zal weten dat ik momenteel puur theoretisch praat en in het geheel niet praktisch maar daarover een ander keer.

Andere operators zoals '*' en '/' leveren over het algemeen weinig problemen op. Zeker zolang we in floating point werken; in integer behoeven sommigen bij de deling misschien nog wat uitleg daar de uitleg in het handboek wel erg summier (slechts een voorbeeld) is. Zolang de delingen uitkomen is er niets aan de hand.
Voorbeelden intikken na IMPINT

```
?15/5 , ?21/7 , ?74/2
```

maar U krijgt 'vreemde' resultaten als U tikt:

```
?13/3 , ?22/5 , ?75/7
```

Dit komt omdat de deling om het zo te zeggen wel goed uitgevoerd wordt, maar het deel achter de komma gewoon wordt weggelaten. In werkelijkheid wordt de deling anders uitgevoerd en krijgen we geeneens een deel achter de komma, de gebruikte methode kunnen we vergelijken met een deling op de basisschool waar er een rest blijft. Soms kunnen deze afrondingen ons echter ongewild toch parten spelen. Denk eens aan het volgende voorbeeld. Ik wil graag een aantal lijnen tekenen. Bijvoorbeeld van 0,0 naar 10,30 en vervolgens steeds 3.2 verder naar rechts schuiven. Om een en twintig lijnen te krijgen zou het volgende programma goed moeten zijn.

```
10 MODE4:REM NA IMPFPT
20 FOR X=0 TO 64 STEP 3.2
30 DRAW X,0 X+10,30 22
40 NEXT
```

Tikken we dit in en laten we het uitvoeren dan zien we,daar DRAW integers verwacht en dus de floating point getallen afkapt dat de ruimte tussen de lijnen soms 3 en soms 4 is. Louis Gidney had hier een knappe maar ingewikkelde oplossing voor, maar wil eventuele publicatie zelf verzorgen en dat mag.

Ik kan me ook voorstellen dat bij de vergelijking tussen A% en B! er fouten gemaakt kunnen worden, omdat in dit geval A% wordt omgewerkt tot een floating point getal voordat de vergelijking wordt gedaan en ik kan me voorstellen dat hoewel we er zeker van zijn dat een van de twee kleiner is het resultaat een gelijkheid is of omgekeerd. Enkel en alleen door afrondingen bij de conversie. Toch heb ik hier zelf nooit problemen mee gehad (Wie wel ?).Nico Looije kwam voor het uitprinten van dit artikel met het volgende aan:

```
10 IF INT(2^3)=8 THEN ?"HET IS INDERDAAD HETZELFDE".
```

Machtsverheffen levert soms problemen op. Het pijltje dat we daar voor gebruiken is een floating point verwachtende operator en eventuele integers worden geconverteerd. Daar bij A tot de macht B, B ook een breuk (beter een gebroken getal) of negatief mag zijn is de definitie (en dus ook de berekening) anders dan we destijds op school leerden toen we er voor het eerst kennis mee maakten Voorbeelden als 2^3 betekent 2*2*2 en 7^4 betekent 7*7*7*7 enz. zijn simpel te begrijpen als een andere schrijfwijze, die zeker als de exponent groot is erg handig kan zijn. We kunnen doorgaand in deze gedachtengang ook nog wel 8^1=8 en na het maken van een tabelletje van 2^4=16, 2^3=8, 2^2=4, 2^1=2 ook nog wel begrijpen dat het logisch is om 2^0=1, 2^(-1)=.5, 2^(-2)=.25 enz af te spreken Het is alweer een stap moeilijker om 2^.5 te snappen maar als we weten dat 2^3 * 2^5 = 2^8 dan moet 4^.5 * 4^.5 = 4^1 dus 4 zijn. En het getal dat met zichzelf vermenigvuldigd 4 oplevert is 2 dus is 4^.5 = 2 en als het wat minder mooi uitkomt 6^.5 = 6 de normale SQR(6). Nemen we 64^(1/3) krijgen we 4 de derdemachts wortel uit 64, maar pas op alleen na IMPFPT anders krijgen we 1.0 omdat na integerdeling 1/3 de exponent 0 is en 64^0 = 1 !

We zien dan ook dat iets als (-4)^2.3 onmogelijk is met reele getallen al kunnen we best (-4)^2 zelf uitrekenen door (-4)*(-4) te doen. De methode die DAI gebruikt is echter niet mogelijk met negatieve grondtallen, dus ook niet als de exponent een geheel getal is. Als U zich, zonder te diep wiskundig te gaan graven, een voorstelling wilt maken; doe dan het volgende : Neem een vel ruitjespapier en maak daar een normaal assenstelsel op. Kies als grondtal bijv 2. Bij de horizontale 1 zet U een punt op hoogte 2 (=2^1), bij horizontaal 2 een punt op hoogte 4 (=2^2), bij horizontaal 3 een punt op hoogte 8 =(2^3) enz. Verbind de punten die U gevonden heeft door een zo vloeiend mogelijk lopende lijn. U kunt nu in deze zelf gemaakte grafiek redelijk aflezen wat bv 2^2.61 moet zijn.Controleer dit met uw DAI. Vindt U het moeilijk om U voor te stellen hoe de grafiek naar links voor de negatieve waarden doorgaat nummer dan de vertikale as niet 1,2,3,4,.... maar met 1,2,4,8,16,32,64,128,.... dus steeds het dubbele. U zult zien dat het aflezen van de grafiek moeilijker wordt maar de grafiek zelf gemakkelijker. Pas echter op met het nummeren van de vertikale as naar beneden toe het is niet -1,-2,-4,-8,-16,... maar 1/2,1/4,1/8,1/16,.... met 1/2 bij het streepje onder de 1 (naar boven toe is maal twee, naar beneden toe is gedeeld door twee). Neem nu eens als grondtal 3 maar gebruik dezelfde nummering als daarnet. (verbaasd ?) Poog nu hetzelfde te doen met grondtal -2. De basispunten waarbij de exponent toch geheel is kunt U bepalen, maar kunt U nu ook nog gemakkelijk zien hoe de grafiek moet lopen ? Misschien voelen sommigen zich gekwetst door mijn aanname dat zij dit nog niet wisten, maar bedenk dan dat U misschien minder vaak dan ik te maken krijgt met DAI-bezitters die stomverbaasd zijn dat (-3)^2 niet kan. Aan de andere kant zou het wel prettig zijn als de DAI een tweede machtsverheffing kende die alleen op integers werkte zodat bv de stelling van Pythagoras probleemloos gebruikt kan worden. Maar zoals eerder reeds verteld A*A is echt sneller dan A^2 en het resultaat is betrouwbaarder en het werkt probleemloos met negatieve A.

Verder wil ik U nog wijzen op de prioriteitsregels voor de verschillende operatoren. Daar het in een totaal ander deel van het handboek staat (6.2.2.5) is het niet vreemd als U vaak veel te veel haakjes gebruikt om zeker van een juiste afhandeling te zijn. Nadeel van deze extra haakjes is een iets langere werktijd (zeer gering). Een voordeel kan soms zijn dat het sneller duidelijk

wordt hoe er gerekend wordt.Aan de andere kant moeten we toch ook zien dat een expressie waarin meer dan 10 haakjes voorkomen niet snel te begrijpen valt. Zoals in de wiskunde gebruikelijk kan elke prioriteit door de programmeur veranderd worden door het betreffende deel tussen haakjes te zetten als hij dat als eerste wil laten uitwerken.
Het hoogste niveau is machtsverheffen gevolgd door vermenigvuldigen , delen en modulo nemen (Wist U dat de uitspraak van MOD modulo was ?). Deze drie hebben alle dezelfde prioriteit en zullen (zonder haakjes) van links naar rechts worden afgewerkt.

222 MOD 30*2=24 en 111/3 MOD 10=7 enz.

Op het laagste niveau staan hier optellen en aftrekken. Er zijn nog lagere niveaus maar die zijn er voor SHL,SHR,IOR,IAND IXOR, > , < , = , <> , >= , <= , AND en OR. Deze zullen echter in het tweede deel van deze verhandeling aan bod komen. Ik heb het nu al haast klaar en zal wat dat betreft dus rustig van de komende feestdagen kunnen genieten.

Ik wil besluiten iedereen eveneens prettige feestdagen toe te wensen en alvast een DAInamisch 1984

PROBLEMEN MET PRINTING ? ? ?

Heeft U ook al vaak gehad dat U wilde beginnen met iets op de printer en omdat U nog niet klaar was de printer afgeschakeld had met POKE #131,1 ? U veranderde iets en maakte een klein foutje en kreeg dus 'SYNTAX ERROR' Dit hoeft geen probleem te zijn, maar omdat deze tekst ook op de printer komt is ons mooie lege vel papier ineens niet leeg meer. De printer OFF LINE zetten helpt niet, het uitzetten van de printer wel. Maar dan zien we de instellingen van de printer zoals tab en double printing weer verloren gaan. Dit alles alleen omdat de DAI op #131 weer een 0 zet als een foutmelding komt. Dit probleem is echter op te lossen. In DAInamic nr 18 schetste ik het al in het kort. Iedere keer dat iets afgedrukt gaat worden wordt interrupt 5 doorlopen. Als we er nu voor zorgen dat deze eerst via onze omweg op #131 zet wat we willen zal er alleen dan naar de printer tekst gestuurd worden als wij dat willen.
Onze routine die bij elke doorgang van interrupt vijf doorlopen zal worden is slechts tien bytes groot en zal normaal probleemloos in 't gebied van de envelopes opgeborgen kunnen worden. Als het startadres is gekozen voor #260. U kunt de routine met Substitute in Utility vanaf #260 plaatsen en vector vijf naar onze routine laten springen. En vanaf dit moment kunt U de printer laten printen als U dat wil.

Printer aan : POKE #262,0

Printer uit : POKE #262,1

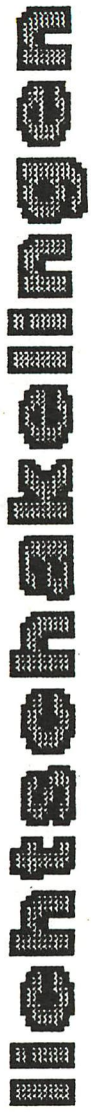
zoals U ziet dezelfde poke's als U gewend bent, alleen het poke-adres is tweemaal hetgeen U gewend was. Nu de routine en hoe U hem op zijn plaats krijgt.

```
I        UT
II        S260 [SPACE] F5 3E 01 32 31 01 C3 FD 6C [CURSOR LEFT]
III        V5 [SPACE] 260 [CURSOR LEFT]
```

Ik hoop velen hiermee een plezier te hebben gedaan

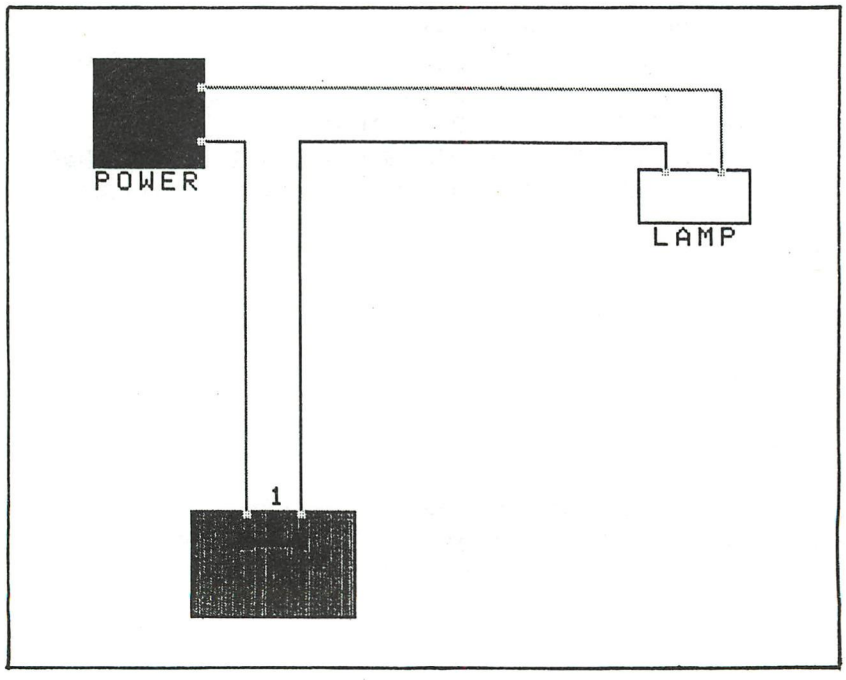
Frank H Druijff

```
100   REM *** ELECTRISCHE LICHT-SCHAKELINGEN *******
110   REM *** GESCHREVEN DOOR : DE BONT CORNEEL ****
120   REM ********************* 3 - 7 - 1983 *******
130   REM *** DIT PROGRAMMA TOONT DE WERKING VAN : *
140   REM *** 1) ENKELPOLIGE SCHAKELING ************
150   REM *** 2) DUBBELPOLIGE SCHAKELING ***********
160   REM *** 3) WISSEL-SCHAKELING *****************
170   REM *** 4) KRUIS-SCHAKELING ******************
180   REM ****************************************
190   CLEAR 500:HT=10:GOTO 1000
200   REM *** SCHAKELAARS (X,S)
210   XG=X+23:FILL X-5,0 X+55,40 S:FILL X-4,1 X+54,39 8
220   FILL X+13,26 X+17,30 0:G$="3":IF X=40 THEN G$="1"
230   IF S=3 THEN DRAW X+5,28 X+12,28 3:GOTO 250
240   DRAW X+15,31 X+15,40 3
250   FILL X+33,26 X+37,30 0:YG=41:IF X=120 THEN G$="2"
260   GOSUB 900:IF S=4 THEN DRAW X+38,28 X+45,28 3:GOTO 280
270   DRAW X+35,31 X+35,40 3:IF S=1 THEN 360
280   FILL X+13,10 X+17,14 0
290   IF S<5 THEN DRAW X+5,12 X+12,12 3
300   DRAW X+5,12 X+5,40 3:FILL X+33,10 X+37,14 0
310   IF S<5 THEN DRAW X+38,12 X+45,12 3
320   DRAW X+45,12 X+45,40 3:IF S<5 THEN 360
330   DRAW X+5,3 X+5,12 3:DRAW X+5,3 X+35,3 3
340   DRAW X+35,3 X+35,9 3:DRAW X+45,6 X+45,12 3
350   DRAW X+15,6 X+45,6 3:DRAW X+15,6 X+15,9 3
360   FOR Z=X TO X+50 STEP 5
370   IF SCRN(Z,39)=3 THEN FILL Z-1,38 Z+1,40 14
380   NEXT:IF S>2 THEN GOSUB 400
390   RETURN
400   REM *** SCHAKELEN (X)
410   SC=SCRN(X,0):SW=SCRN(X+20,28)
420   X1=X+14:X2=X+16:X3=X+18
430   X4=X+32:X5=X+34:X6=X+36
440   IF SC<>1 THEN 460
450   FILL X3,27 X4,29 0:IF SW=0 THEN FILL X3,27 X4,29 8
460   IF SC<>2 THEN 490
470   C=0:IF SW=0 THEN C=8
480   FILL X3,11 X4,13 C:FILL X3,27 X4,29 C
490   IF SC<3 OR SC>4 THEN 530
500   IF SW=8 THEN 520
510   FILL X3,11 X4,13 0:FILL X3,27 X4,29 8:GOTO 530
520   FILL X3,11 X4,13 8:FILL X3,27 X4,29 0
530   IF SC<>5 THEN 590
540   IF SW=0 THEN 570
550   FILL X1,15 X6,25 8:FILL X3,27 X4,29 0
560   FILL X3,11 X4,13 0:GOTO 590
570   FILL X3,11 X4,29 8:FILL X1,15 X2,25 0
580   FILL X5,15 X6,25 0
590   RETURN
600   REM *** LAMP (L)
610   CC=0:IF L=1 THEN CC=15
620   FILL 200,150 240,170 0:FILL 201,151 239,169 CC
630   FILL 209,168 211,170 14:FILL 229,168 231,170 14
640   XG=205:YG=140:G$="LAMP":GOSUB 900
650   RETURN
700   REM *** POWER SUPPLY
710   FILL 0,170 40,210 0:FILL 1,171 39,209 3
730   FILL 38,179 40,181 14:FILL 38,199 40,201 14
740   XG=0:YG=160:G$="POWER":GOSUB 900
750   RETURN
800   REM *** GETC
810   G=GETC:G=GETC:G=GETC:P=#73C4:W=0:POKE #75,95
820   CC=INT(15*RND(1))+1:POKE P,208+CC:IF CC=8 THEN 820
```

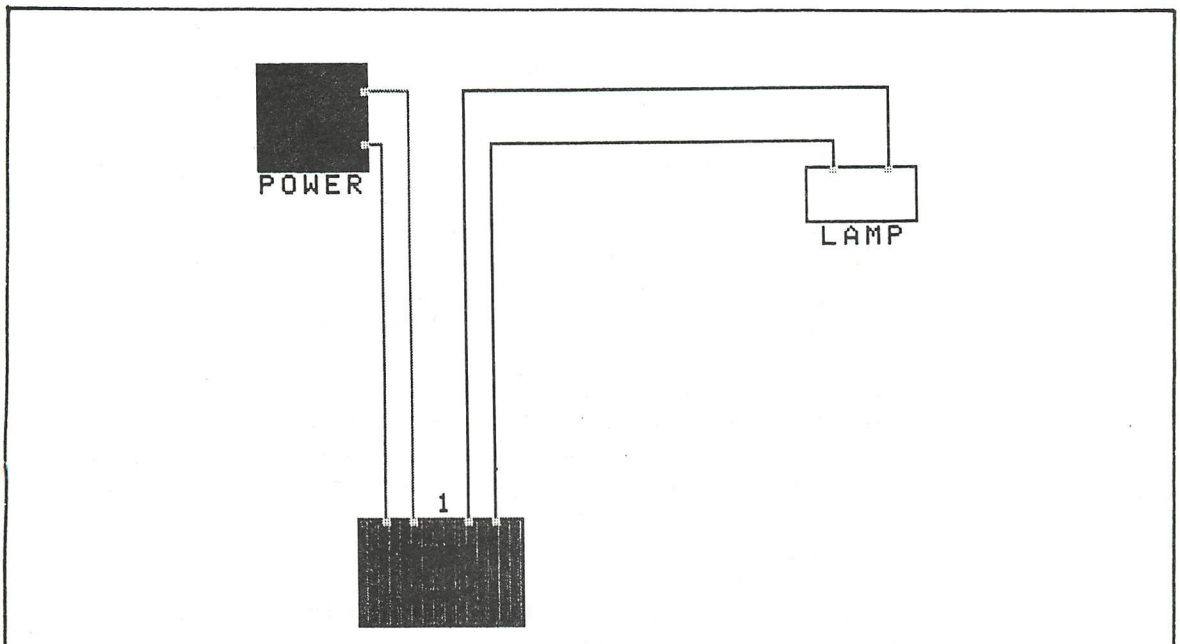


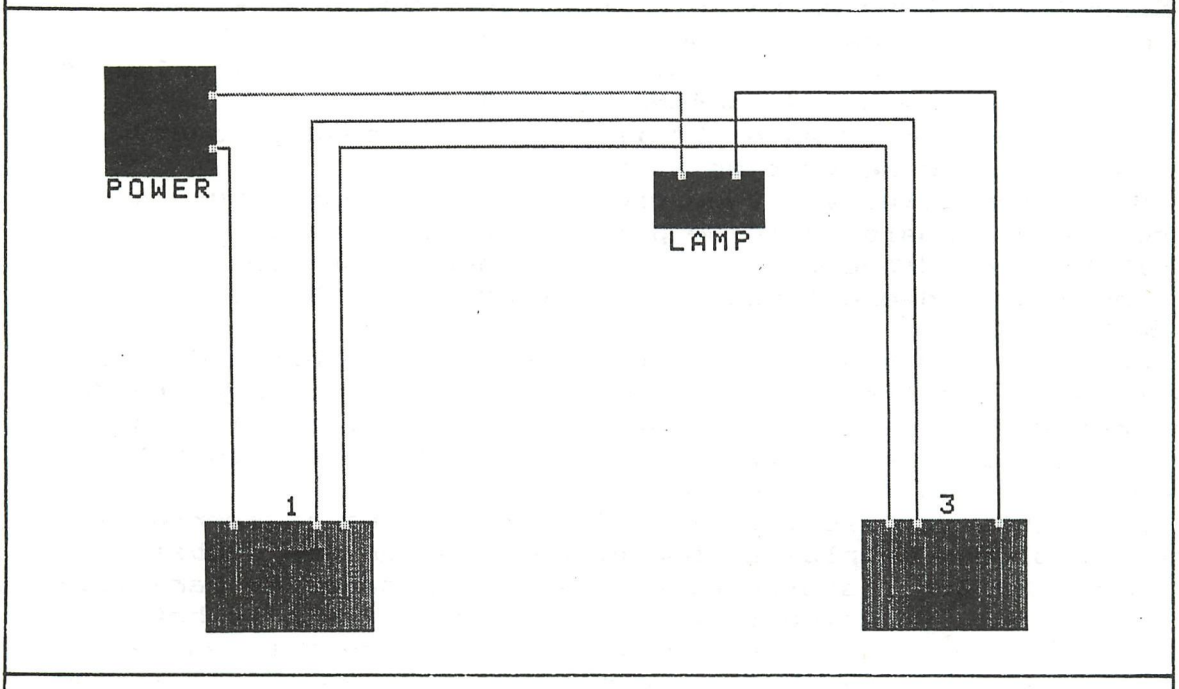
```
830   SOUND 1 0 9 0 FREQ((CC*50)+50):WAIT TIME 3:SOUND OFF
840   G=GETC:IF G=0 THEN W=W+1:WAIT TIME 3:IF W<30 THEN 840
850   IF G=0 THEN W=0:GOTO 820
860   POKE #75,32:RETURN
900   REM *** BASIC CALL OF FGT
910   SOUND 1 0 15 0 FREQ(XG+YG+50)
920   POKE #2F0,TC:POKE #2F1,TH:POKE #2F2,XG MOD 256
930   POKE #2F3,XG SHR 8:POKE #2F4,YG:POKE #2F5,8
940   POKE #2F6,10:CALLM #300,G$:SOUND OFF :RETURN
1000  REM *** MENU
1010  COLORG 8 5 3 14:MODE 6A:PRINT CHR$(12);:ENVELOPE 0 15
1020  COLORT 8 0 0 0:FOR I=0 TO HT:TC=22:IF I=HT THEN TC=23
1030  G$="ELECTRISCHE     LICHT-          SCHAKELINGEN."
1040  XG=80+I:YG=120+I:TH=1:GOSUB 900:NEXT:TH=0:TC=21
1050  XG=270:YG=40:G$=" door:  De Bont Corneel.":GOSUB 900
1060  TH=0:TC=0:POKE #75,32:CURSOR 3,1:HT=2
1070  PRINT "1=ENKELPOLIG 2=DUBBELPOLIG 3=WISSEL 4=KRUIS 5=STOP"
1080  PRINT TAB(15);"WELKE KIEST U (1-5) ";
1090  GOSUB 800:IF G<49 OR G>53 THEN 1060
1100  PRINT CHR$(12);TAB(15);:G=G-48
1110  IF G<5 THEN MODE 5A:MODE 5A
1120  ON G GOTO 2000,3000,4000,5000,6000
1130  GOTO 1060
2000  REM *** ENKELPOLIG
2010  PRINT "ENKELPOLIGE SCHAKELING "
2020  GOSUB 700:GOSUB 600:X=40:S=1:GOSUB 200
2030  DRAW 41,180 55,180 6:DRAW 55,41 55,180 6
2040  DRAW 75,41 75,180 0:DRAW 75,180 210,180 0
2050  DRAW 210,180 210,171 0:DRAW 41,200 230,200 9
2060  DRAW 230,200 230,171 9
2070  PRINT "DE SCHAKELAAR EN DE LAMP STAAN IN SERIE GESCHAKELD AAN DE"
2080  PRINT "VOEDINGSSPANNING.DE SCHAKELAAR ONDERBREEKT 1 DER 2 DRADEN"
2090  PRINT "U KUNT SCHAKELEN MET TOETS '1' (TAB=MENU) ";
2100  X=40:GOSUB 400:L=1:GOSUB 600
2110  GOSUB 800:L=0:IF G=9 THEN 1000
2120  IF G=49 THEN X=40:GOSUB 400:GOTO 2140
2130  GOTO 2110
2140  IF SCRN(60,28)=0 THEN L=1
2150  GOSUB 600:GOTO 2110
3000  REM *** DUBBELPOLIG
3010  PRINT "DUBBELPOLIGE SCHAKELING"
3020  GOSUB 700:GOSUB 600:X=40:S=2:GOSUB 200
3030  DRAW 41,180 45,180 6:DRAW 45,180 45,41 6
3040  DRAW 41,200 55,200 9:DRAW 55,200 55,41 9
3050  DRAW 75,41 75,200 0:DRAW 75,200 230,200 0
3060  DRAW 230,200 230,171 0:DRAW 85,41 85,180 0
3070  DRAW 85,180 210,180 0:DRAW 210,180 210,171 0
3080  PRINT "DE SCHAKELAAR STAAT NU IN SERIE MET ALLEBEI DE VOEDINGS-"
3090  PRINT "DRADEN.NU WORDEN BEIDE DRADEN ONDERBROKEN OF GESCHAKELD."
3100  PRINT "U KUNT SCHAKELEN MET TOETS '1' (TAB=MENU) ";
3110  X=40:GOSUB 400:L=1:GOSUB 600
3120  GOSUB 800:L=0:IF G=9 THEN 1000
3130  IF G=49 THEN X=40:GOSUB 400:GOTO 3150
3140  GOTO 3120
3150  IF SCRN(60,28)=0 THEN L=1
3160  GOSUB 600:GOTO 3120
4000  REM *** WISSEL-SCHAKELING
4010  PRINT "   WISSEL-SCHAKELING   "
4020  GOSUB 700:GOSUB 600:X=40:S=3:GOSUB 200
4030  X=280:S=4:GOSUB 200:DRAW 41,180 45,180 6
4040  DRAW 45,180 45,40 6:DRAW 41,200 210,200 9
4050  DRAW 210,200 210,171 9:DRAW 75,41 75,190 5
4060  DRAW 75,190 295,190 5:DRAW 295,190 295,41 5
4070  DRAW 85,41 85,180 5:DRAW 85,180 285,180 5
4080  DRAW 285,180 285,41 5:DRAW 325,41 325,200 0
4090  DRAW 325,200 230,200 0:DRAW 230,200 230,171 0
```

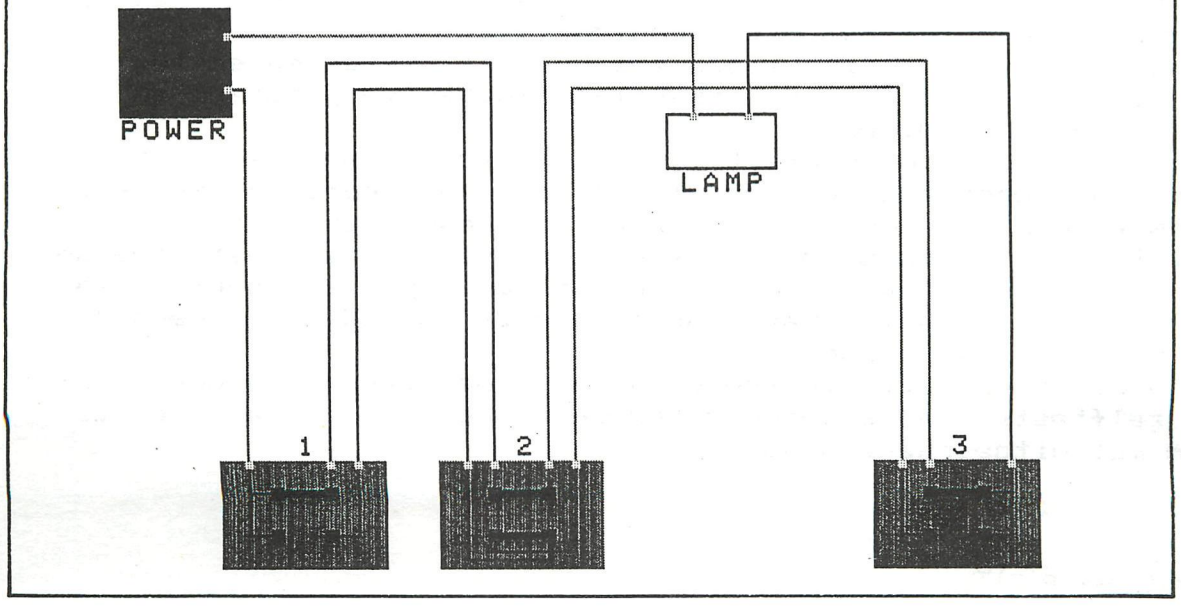
```
4100  PRINT "TWEE SCHAKELAARS BEDIENEN NU VIA TWEE WISSELDRADEN"
4110  PRINT "EEN EN DEZELFDE LAMP (OMSCHAKELING VAN 2 DRADEN)"
4120  PRINT "U KUNT SCHAKELEN MET TOETSEN '1' EN '3' (TAB=MENU) ";
4130  L=1:GOSUB 600
4140  GOSUB 800:L=0:IF G=9 THEN 1000
4150  IF G=49 THEN X=40:GOSUB 400:GOTO 4180
4160  IF G=51 THEN X=280:GOSUB 400:GOTO 4180
4170  GOTO 4140
4180  IF SCRN(60,28)=0 AND SCRN(300,28)=0 THEN L=1
4190  IF SCRN(60,12)=0 AND SCRN(300,12)=0 THEN L=1
4200  GOSUB 600:GOTO 4140
5000  REM *** KRUIS-SCHAKELING
5010  PRINT "   KRUIS-SCHAKELING    "
5020  GOSUB 700:GOSUB 600:X=40:S=3:GOSUB 200:X=120:S=5
5030  GOSUB 200:X=280:S=4:GOSUB 200:DRAW 41,180 45,180 6
5040  DRAW 45,180 45,40 6:DRAW 41,200 210,200 9
5050  DRAW 210,200 210,171 9:DRAW 75,41 75,190 4
5060  DRAW 75,190 135,190 4:DRAW 135,190 135,41 4
5070  DRAW 85,41 85,180 4:DRAW 85,180 125,180 4
5080  DRAW 125,180 125,41 4:DRAW 155,41 155,190 7
5090  DRAW 155,190 295,190 7:DRAW 295,190 295,41 7
5100  DRAW 165,41 165,180 7:DRAW 165,180 285,180 7
5110  DRAW 285,180 285,41 7:DRAW 325,41 325,200 0
5120  DRAW 325,200 230,200 0:DRAW 230,200 230,171 0
5130  PRINT "DRIE SCHAKELAARS BEDIENEN NU VIA TWEE WISSELDRADEN"
5140  PRINT "EEN EN DEZELFDE LAMP (OMSCHAKELING VAN 2 DRADEN)"
5150  PRINT "U KUNT SCHAKELEN MET TOETSEN '1','2' EN '3' (TAB=MENU) ";
5160  L=1:GOSUB 600
5170  GOSUB 800:L=0:IF G=9 THEN 1000
5180  IF G=49 THEN X=40:GOSUB 400:GOTO 5220
5190  IF G=50 THEN X=120:GOSUB 400:GOTO 5220
5200  IF G=51 THEN X=280:GOSUB 400:GOTO 5220
5210  GOTO 5170
5220  IF SCRN(60,28)=0 AND SCRN(140,28)=0 AND SCRN(300,28)=0 THEN L=1
5230  IF SCRN(60,12)=0 AND SCRN(140,12)=0 AND SCRN(300,12)=0 THEN L=1
5240  IF SCRN(60,12)=0 AND SCRN(136,15)=0 AND SCRN(300,28)=0 THEN L=1
5250  IF SCRN(60,28)=0 AND SCRN(136,15)=0 AND SCRN(300,12)=0 THEN L=1
5260  GOSUB 600:GOTO 5170
6000  REM *** EINDE
6010  PRINT CHR$(12);TAB(22);"E I N D E":POKE #75,95
```



lichtschakelingen







TECH-TIP

TECH-TIP TECH-TIP TECH-TIP TECH-TIP TECH-TIP TECH-TIP TECH-TIP

MX80-100 MET GRAFTRAX + TYPE III EPROMS

De MX80-100 gebruikers hebben nu de mogelijkheid alle functies te benutten die er zijn met zowel de GRAFTRAX als de type III printers. U dient wel over beide eprom-sets te beschikken.
v.b.: schuin schrift (italic) zowel als super klein schrift (superscript) blijven behouden.
Dit is mogelijk door beide eprom-sets boven op elkaar te solderen, behalve de voedingspennen. Met een uitwendige schakelaar kan U eenvoudig het gewenste eprom-dek kiezen met de bijhorende karakter-set.De plaats en de bevestiging voor de schakelaar is reeds voorzien!

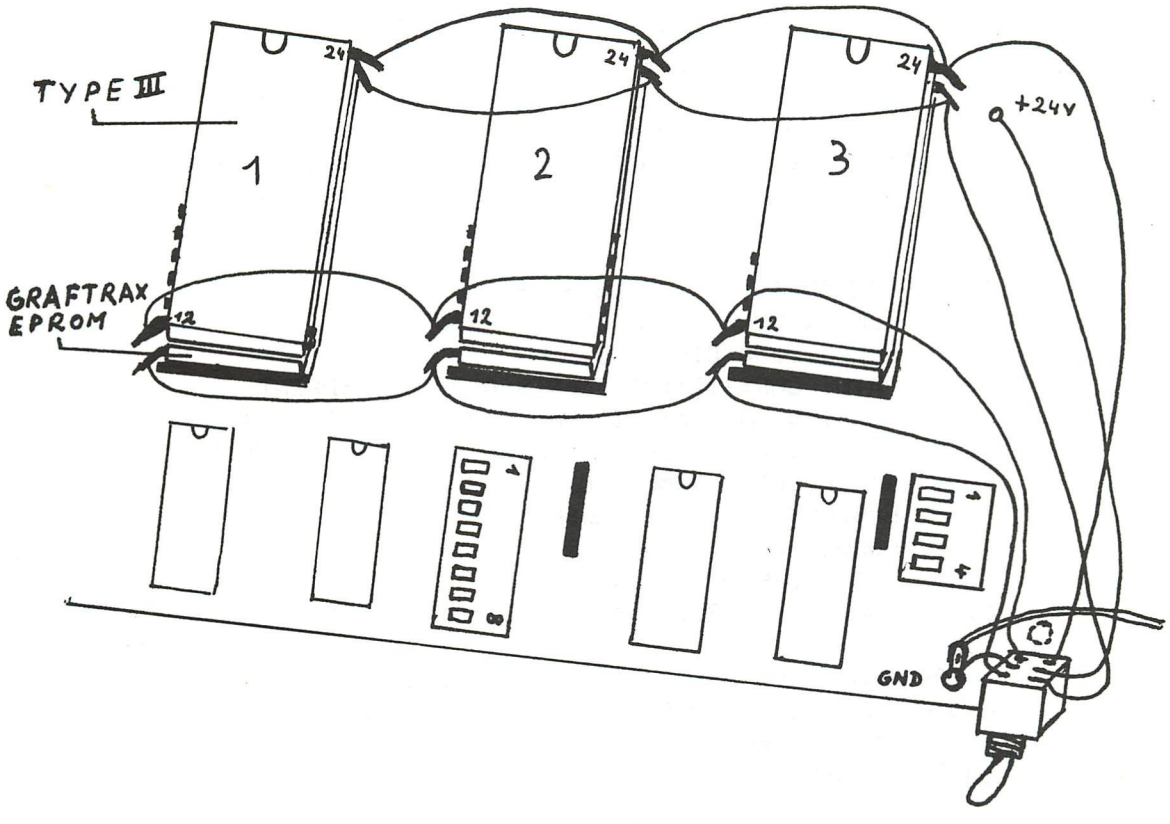
De juiste werkwijze is als volgt:

Na het demonteren van de bodem van de printer verwijdert U het bestaande eprom-set (let wel op de nummering).
Vervolgens worden de twee eprom-sets (per twee) boven op elkaar gelegd en gesoldeerd, behalve de pennen 12 en 24, die nog NIET gesoldeerd worden.Dus eprom GRAFTRAX nr1 onder en TYPE III nr1 boven en vervolgens hetzelfde met de nummers 2 en 3 .
Plooi nu alle pennen nr 12 en nr 24 lichtjes naar boven, zodat ze straks niet meer in de voetjes kunnen .
Zet nu de drie stapeltjes van twee eproms naast elkaar, ongeveer op dezelfde afstand zoals ze in de printer moeten komen.
Met heel fijn soepel draad worden van het onderste dek de penen 12 verbonden en gesoldeerd ; hetzelfde met de pennen 12 van het bovenste dek.
Draai de sets om en verbind de pennen nummers 24 van het onderste dek met elkaar ; hetzelfde met de pennen nr 24 van het bovenste dek.
Soldeer alvast vier stukjes draad van 8 cm aan resp. nr. 12 van onderdek, nr. 12 bovendek, nr.24 onder, en nr. 24 boven, en dit aan de rechterkant van eproms nr.3.
Steek de eprom-sets voorzichtig terug in hun voetjes (nummering!).
Localiseer vervolgens de plaats voor de montage van een dubbele miniatuur-omschakelaar : achter de printer, vlak naast de parallel-stekker ziet U een uitsparing in de kast met een vijsje in het aluminium chassis. Dit vijsje kan dienen voor het bevestigen van een rechthoekig plaatje van 2 X 1 cm met twee gaatjes : 1 (3mm) voor het vastzetten van het plaatje en 1 (6mm) voor het monteren van de schakelaar.(De tekening zal dit verduidelijken)
Bij het monteren van de schakelaar zal U mogelijks de massaklem met de groene draad vlakbij de schakelaar moeten verplaatsen en meer links moeten solderen.
Verbind het linkse middencontact van de schakelaar met massa, en het rechtse middencontact met de plus 5 volt (bij eprom 3 pin 24).
Vervolgens worden de 4 draden van de eproms gesoldeerd : het onderdek pin 12 aan linkse onderste contact schakelaar, het bovendek pin 12 aan bovenste linkse contact schakelaar ; pin 24 onderdek aan rechtse onderste contact schakelaar en pin 24 bovendek aan rechtse bovenste contact schakelaar.
Hiermee is de installatie voltooid en kan U met eenvoudig omschakelen de beide zelftests van de beide eprom-sets proberen. Schakel echter alleen om met uitgeschakelde printer.

Nog enkele voordelen:

- Het is zondermeer mogelijk om screencopy-routines van de EPSON MX82 te gebruiken. (lukte niet met GRAFTRAX !)
- Tijdwinst is enorm: screencopy 9 grijstinten, met GRAFTAX = 21 min ; met TYPE III slechts 8 minuten !!!
- Schuin schrift blijft behouden (Italics)
- Mogelijk labels maken met superscript
- Stand GRAFTRAX : standaard 80 karakters
- Stand TYPE III : standaard 132 karakters

A. De Dauw



= TRI PAR SELECTION SIMPLE EN LANGAGE MACHINE =

Le programme que je vous propose ci-dessous trie une série d'éléments se trouvant dans un vecteur ou un tableau entier. Le tri peut s'effectuer sur l'ensemble du tableau ou sur une partie de celui-ci. Le nombre d'éléments à trier doit etre stocké aux adresses #300 et #301 et l'appel à la routine se fait de la façon suivante:
CALLM#306,NOM DU TABLEAU(INDICE(S) DU PREMIER ELEMENT DE LA SERIE)

Si le tri doit s'effectuer dans un tableau, il faut savoir que tous les éléments de celui-ci sont rangés en mémoire ligne par ligne. Les éléments seront donc aussi triés ligne par ligne.

Pour essayer l'exemple, taper le machine et le basic et les enregistrer sur cassette. RESET. Adapter les pointeurs: POKE#29B,0:POKE#29C,4:CLEAR1000. Lire le machine. NEW. Lire le basic. RUN. Dans cet exemple, le tri se fait dans un vecteur de 256 éléments contenant des valeurs entières comprises entre -500 et 500. Le tri en lui-meme prend environ 6 secondes. Voici quelques exemples de temps d'exécution:

Nbre d'éléments:	Temps (sec.):
100	1
200	4
300	8
400	15
500	23
600	32
1000	89
1500	209

Le temps d'exécution croit évidemment beaucoup suivant le nombre d'éléments mais dans la plupart des cas pratiques, le délai est raisonable. Bien sur, pour trier plusieurs milliers de nombres, il serait préférable d'utiliser une autre méthode mieux adaptée aux grandes séries.

```
10    REM CHRISTIAN POELS - 26/6/83
20    REM EXEMPLE DE PROGRAMME PRINCIPAL
30    CLEAR 2000:DIM A(255)
40    PRINT CHR$(12);"ELEMENTS A TRIER:":PRINT
50    FOR I=0 TO 255:A(I)=RND(1000)-500:PRINT A(I),:NEXT
60    PRINT :PRINT :PRINT "LE TRI S'EFFECTUE!"
70    N=256:REM N=NOMBRE D'ELEMENTS A TRIER
80    E=0:REM TRI SUR N ELEMENTS A PARTIR DE A(E)
90    GOSUB 60000:REM TRI
100   PRINT :PRINT :PRINT "ELEMENTS TRIES:":PRINT
110   FOR I=0 TO 255:PRINT A(I),:NEXT:PRINT
120   END
60000 REM APPEL DE LA ROUTINE MACHINE
60010 POKE #300,N MOD 256:POKE #301,N/256
60020 CALLM #306,A(E)
60030 RETURN
```

```
0300 00 00 00 00 00 00 C5 D5 E5 F5 EB 2A 00 03 44 4D
0310 0B EB 54 5D C5 E5 CD 6B 03 23 23 23 23 E5 23 23
0320 23 3A 05 03 96 2B 3A 04 03 9E 2B 3A 03 03 9E 2B
0330 3A 02 03 9E FA 3C 03 54 5D CD 6B 03 0B E1 78 B1
0340 C2 19 03 E1 1A 4F 7E 71 12 23 13 1A 4F 7E 71 12
0350 23 13 1A 4F 7E 71 12 23 13 1A 4F 7E 71 12 23 C1
0360 0B 78 B1 C2 12 03 F1 E1 D1 C1 C9 7E 32 02 03 23
0370 7E 32 03 03 23 7E 32 04 03 23 7E 32 05 03 2B 2B
0380 2B C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

TRI par selection simple

```
001                             ORG   :306
002 0306 C5                     PUSH  B
003 0307 D5                     PUSH  D
004 0308 E5                     PUSH  H
005 0309 F5                     PUSH  PSW
006 030A EB                     XCHG
007 030B 2A0003                 LHLD  :300
008 030E 44                     MOV   B,H
009 030F 4D                     MOV   C,L
010 0310 0B                     DCX   B
011 0311 EB                     XCHG
012 0312 54             LOOP1   MOV   D,H
013 0313 5D                     MOV   E,L
014 0314 C5                     PUSH  B
015 0315 E5                     PUSH  H
016 0316 CD6B03                 CALL  PLPE
017 0319 23             LOOP2   INX   H
018 031A 23                     INX   H
019 031B 23                     INX   H
020 031C 23                     INX   H
021 031D E5                     PUSH  H
022 031E 23                     INX   H
023 031F 23                     INX   H
024 0320 23                     INX   H
025 0321 3A0503                 LDA   :305
026 0324 96                     SUB   M
027 0325 2B                     DCX   H
028 0326 3A0403                 LDA   :304
029 0329 9E                     SBB   M
030 032A 2B                     DCX   H
031 032B 3A0303                 LDA   :303
032 032E 9E                     SBB   M
033 032F 2B                     DCX   H
034 0330 3A0203                 LDA   :302
035 0333 9E                     SBB   M
036 0334 FA3C03                 JM    PPP
037 0337 54                     MOV   D,H
038 0338 5D                     MOV   E,L
039 0339 CD6B03                 CALL  PLPE
040 033C 0B             PPP     DCX   B
041 033D E1                     POP   H
042 033E 78                     MOV   A,B
043 033F B1                     ORA   C
044 0340 C21903                 JNZ   LOOP2
045 0343 E1                     POP   H
046 0344 1A                     LDAX  D
047 0345 4F                     MOV   C,A
048 0346 7E                     MOV   A,M
049 0347 71                     MOV   M,C
050 0348 12                     STAX  D
051 0349 23                     INX   H
052 034A 13                     INX   D
053 034B 1A                     LDAX  D
054 034C 4F                     MOV   C,A
055 034D 7E                     MOV   A,M
056 034E 71                     MOV   M,C
057 034F 12                     STAX  D
058 0350 23                     INX   H
059 0351 13                     INX   D
060 0352 1A                     LDAX  D
061 0353 4F                     MOV   C,A
062 0354 7E                     MOV   A,M
063 0355 71                     MOV   M,C
064 0356 12                     STAX  D
065 0357 23                     INX   H
066 0358 13                     INX   D
067 0359 1A                     LDAX  D
068 035A 4F                     MOV   C,A
069 035B 7E                     MOV   A,M
070 035C 71                     MOV   M,C
071 035D 12                     STAX  D
072 035E 23                     INX   H
073 035F C1                     POP   B
074 0360 0B                     DCX   B
075 0361 78                     MOV   A,B
076 0362 B1                     ORA   C
077 0363 C21203                 JNZ   LOOP1
078 0366 F1                     POP   PSW
079 0367 E1                     POP   H
080 0368 D1                     POP   D
081 0369 C1                     POP   B
082 036A C9                     RET
083 036B 7E             PLPE    MOV   A,M
084 036C 320203                 STA   :302
085 036F 23                     INX   H
086 0370 7E                     MOV   A,M
087 0371 320303                 STA   :303
088 0374 23                     INX   H
089 0375 7E                     MOV   A,M
090 0376 320403                 STA   :304
091 0379 23                     INX   H
092 037A 7E                     MOV   A,M
093 037B 320503                 STA   :305
094 037E 2B                     DCX   H
095 037F 2B                     DCX   H
096 0380 2B                     DCX   H
097 0381 C9                     RET
098 0382                        END
```

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

LOOP1  0312   LOOP2  0319   PLPE   036B   PPP    033C
```

DOOLHOFSPEL

```
100   REM *** DOOLHOFSPEL ****************************
110   REM *** GESCHREVEN DOOR : DE BONT CORNEEL *******
120   REM ********************* 30 - 6 - 1983 *********
130   REM *** DE DOOLHOFGENERATOR (LINES 1000-1499) ***
140   REM *** IS EEN GEWIJZIGDE VERSIE VAN DE SUB- ****
150   REM *** ROUTINE WELKE OP ZONDAG 26-6-1983 IN ****
160   REM *** DE NOS-UITZENDING 'HOBBYSCOOP' WERD *****
170   REM *** UITGEZONDEN. ****************************
180   REM *****************************************
190   GOTO 700
200   REM *** GETC-ROUTINE
210   G=GETC:G=GETC:G=GETC:W=0:POKE #75,95
220   SOUND 1 0 9 0 FREQ(RND(900)+50):WAIT TIME 3:SOUND OFF
230   G=GETC:IF G=0 THEN W=W+1:WAIT TIME 3:IF W<30 THEN 230
240   IF G=0 THEN W=0:GOTO 220
250   POKE #75,32:RETURN
700   REM *** TITELPAGINA
710   MODE 0:PRINT CHR$(12);:CLEAR 14000:POKE #75,32
720   COLORT 8 0 8 8:POKE #BF69,#4A:POKE #BF68,220
730   POKE #BE5D,#4A:POKE #BE5C,221
740   POKE #BD51,#4A:POKE #BD50,222
750   POKE #BC44,209:CURSOR 0,22:PRINT "DOOL"
760   CURSOR 0,20:PRINT " HOF-":CURSOR 0,18:PRINT "  SPEL"
770   CURSOR 10,16:PRINT "DRUK OP 1 TOETS OM TE BEGINNEN ";
780   GOSUB 200
800   REM *** INPUT'S
810   COLORG 8 0 10 14:MODE 2A
820   PRINT CHR$(12);TAB(15);"DOOLHOFSPEL.":PRINT
830   PRINT TAB(10);"BEWEEG MET CURSORTOETSEN":DIM CL(3)
840   PRINT " BREEDTE DOOLHOF * 5 (2-7) ";
850   GOSUB 200:IF G<50 OR G>55 THEN 850
860   H=(G-48)*5:PRINT CHR$(G);" * 5 =";H
870   PRINT "  HOOGTE DOOLHOF * 5 (2-5) ";
880   GOSUB 200:IF G<50 OR G>53 THEN 880
890   V=(G-48)*5:PRINT CHR$(G);" * 5 =";V
900   PRINT "WILT U DE DOOLHOF ZIEN GROEIEN (J/N) ";
910   GOSUB 200:IF G=74 OR G=106 THEN 1000
920   IF G=78 OR G=110 THEN COLORG 0 0 0 0:GOTO 1000
930   GOTO 910
1000  REM *** DOOLHOF-GENERATOR (GEWIJZIGDE VERSIE UIT
1010  REM *** NOS-HOBBYSCOOP UITZENDING VAN 26-6-1983)
1020  PRINT CHR$(G):PRINT TAB(10);"EVEN GEDULD A.U.B!";
1030  M=H*2-2:N=V*2-2:DIM D(M,N):X=INT(H/2)*2
1040  XP=INT((71-M)/2)-1:YP=INT((52-N)/2)-2
1050  Y=INT(V/2)*2:D(X,Y)=9:FILL XP,YP+1 XP+M+2,YP+N+3 21
1060  T=0:R1=0:R2=0:R3=0:R4=0
1070  IF X+2>M THEN 1100
1080  IF D(X+2,Y)<>0 THEN 1100
1090  DX=2:DY=0:T=T+1:R1=1
1100  IF Y+2>N THEN 1130
1110  IF D(X,Y+2)<>0 THEN 1130
1120  DX=0:DY=2:T=T+1:R2=1
1130  IF X<2 THEN 1160
1140  IF D(X-2,Y)<>0 THEN 1160
1150  DX=-2:DY=0:T=T+1:R3=1
1160  IF Y<2 THEN 1190
1170  IF D(X,Y-2)<>0 THEN 1190
1180  DX=0:DY=-2:T=T+1:R4=1
1190  IF T=0 THEN 1270
1200  IF T<>1 THEN 1340
1210  REM *** 1 AANSLUITEND HOKJE VRIJ
1220  D(X+0.5*DX,Y+0.5*DY)=1
1230  DOT XP+(X+0.5*DX+1),YP+(Y+0.5*DY+1)+1 20
1240  DOT XP+X+1,YP+Y+2 23:X=X+DX:Y=Y+DY
```

```
1260  DOT XP+X+1,YP+Y+2 22:D(X,Y)=-10*DX-DY:GOTO 1060
1270  REM *** GEEN AANSLUITEND HOKJE VRIJ DOE STAP TERUG
1280  DOT XP+X+1,YP+Y+2 20
1290  IF D(X,Y)=9 THEN 1420
1300  BK=D(X,Y):IF ABS(BK)=2 THEN 1320
1310  X=X+BK/10:GOTO 1330
1320  Y=Y+BK
1330  DOT XP+X+1,YP+Y+2 20:GOTO 1060
1340  REM *** >1 AANSLUITEND HOKJE VRIJ-KIES ER EEN UIT
1350  DR=INT(4*RND(1))+1
1360  ON (DR) GOTO 1370,1390,1400,1380
1370  IF R1=1 THEN DX=2:DY=0:GOTO 1220
1380  IF R4=1 THEN DX=0:DY=-2:GOTO 1220
1390  IF R2=1 THEN DX=0:DY=2:GOTO 1220
1400  IF R3=1 THEN DX=-2:DY=0:GOTO 1220
1410  GOTO 1350
1420  REM *** INGANG EN UITGANG
1430  COLORG 8 0 10 14:FOR VE=0 TO 2*V STEP 2*V
1440  HO=(INT(H*RND(1))+1)*2-1:DOT XP+HO,YP+VE+1 20:NEXT VE
1450  DRAW XP,YP XP,YP+N+4 21:DRAW XP+M+2,YP XP+M+2,YP+N+4 21
2000  REM *** UW WANDELING
2010  FOR I=0 TO 3:CL(I)=PEEK(#9E+I)-(#80+(I*#10)):NEXT
2020  PRINT CHR$(12);"WILT U EEN SPOOR ACHTERLATEN (J/N) ";
2030  GOSUB 200:IF G=74 OR G=106 THEN SK=22:GOTO 2070
2050  IF G=78 OR G=110 THEN SK=20:GOTO 2070
2060  GOTO 2030
2070  PRINT CHR$(G):X=10+XP:Y=YP:HX=9:HY=0:T=0
2080  SOUND 1 0 15 0 FREQ(X+Y+50):DOT X,Y 23:HX=X:HY=Y
2090  G=GETC:IF G<16 OR G>19 THEN WAIT TIME 3:SOUND OFF :GOTO 2090
2100  IF G=16 AND SCRN(X,Y+1)<>CL(1) THEN Y=Y+1:IF Y>YP+N+4 THEN Y=YP+N+4
2110  IF Y=0 THEN 2130
2120  IF G=17 AND SCRN(X,Y-1)<>CL(1) THEN Y=Y-1:IF Y<YP THEN Y=YP
2130  IF G=18 AND SCRN(X-1,Y)<>CL(1) THEN X=X-1:IF X<XP+1 THEN X=XP+1
2140  IF G=19 AND SCRN(X+1,Y)<>CL(1) THEN X=X+1:IF X>XP+M+3 THEN X=XP+M+3
2150  T=T+1:DOT HX,HY SK
2160  IF Y=N+4+YP THEN 3000
2170  GOTO 2080
3000  REM *** EIND-KEUZE
3010  DOT X,Y 23:PRINT CHR$(12);"U BENT GESLAAGD !!"
3020  PRINT "IN EEN TIJD VAN";T/10.0;" SECONDEN !!"
3030  PRINT "WILT U NOGMAALS PROBEREN (J/N) ";
3040  GOSUB 200:IF G=78 OR G=110 THEN PRINT CHR$(G):GOTO 4000
3050  IF G=74 OR G=106 THEN PRINT CHR$(G):GOTO 3070
3060  GOTO 3040
3070  PRINT "WILT U DEZELFDE DOOLHOF OF EEN ANDER (D/A) ";
3080  GOSUB 200:IF G=68 OR G=100 THEN PRINT CHR$(G):GOTO 3200
3090  IF G=65 OR G=97 THEN PRINT CHR$(G):GOTO 800
3100  GOTO 3080
3200  REM *** WIS SPOOR UIT
3210  DRAW XP+1,YP XP+M+1,YP 22:DRAW XP+1,YP+N+4 XP+M+1,YP+N+4 22
3220  FOR X=0 TO M+2:FOR Y=0 TO N+4
3230  IF SCRN(X+XP,Y+YP)=CL(2) THEN DOT XP+X,YP+Y 20
3240  NEXT:NEXT:GOTO 2000
4000  REM *** EINDE
4010  PRINT CHR$(12);TAB(22);"E I N D E":POKE #75,95
```

*** The continuing KEN-DOS story *** (see nr. 14)

First of all I have to make an apology to those of you who have written for more information or even wanted to order a system after the first article on KEN-DOS was published. I have not been able to answer all letters. I promise,however,that your letter will be answered in the forth-comming weeks. You might even have received the information you asked for in the past few days.

When writing the article published in DAInamic nr. 14 , I was confident that I would ship the forst units on short notice. I had a working prototype,the software performed all right,so what could go wrong ?
Well,in fact nothing went wrong,but as I discovered a little later I had underestimated the problems still to come. From working prototype to well designed through-plated pcb is a big step,bigger than I anti-cipated it to be at that time.
Further I decided to make some last minute changes,which caused me to redesign part of the pcb as well as rewrite part of the software. Anyway, I think the changes I was able to make were well worth the effort. Then there were all the administrative problems,specially those involved in exporting technical equipment, I had to deal with. It seemed best to found a company,which from now on will take over all activities connected with KEN-DOS. Name and adress of this company I will give at the bottom of this article. As you will surely understand all this took a lot of time and caused the relatively long period of silence after first publi-cation.

Below I will give a summary of what the system consists of;

1) Bankselectable EPROMcard. On this EPROMcard is room for 6!! 16K EPROMS. 2 EPROMS are used for the KEN-DOS opera-ting system,while another is reserved for the optional CP/M bios.This leaves room for 72K of user EPROMS.

2) Controller-card for 1-8 drives single-sided,double density or 1-4 double-sided,double density drives.
MAXIMUM STORAGE CAPACITY 3,2MB or ± 3.200.000 characters !
On request the controller-card can be modified to handle 8" or micro-drives

3) One or more diskdrives.

4) Powersupply with ringcore transformer. This powersupply is dimensioned to provide more than enough current for two double-sided drives.(storage capacity 1,6 MB)

5) Metal cabinet to house drive(s),powersupply and controller-card

6) Flatcable with connectors to connect drive(s) and control-ler to the DAIpc.

7) KEN-DOS operating system in two 8KB EPROMS ore one 16KB EPROM.
This operating system supports both 40 and 80 tracks drives in single or double density mode.
Available to the user are 43 ! commands which can be used either in direct mode or from basic programs.
Commands include : formatting,create and delete files, protect disk or file,reading of specific tracks or sectors,reading from DCR or ordinary cassette and writing to disk as well as the other way around, bankselect of EPROM banks.

8)Manual "HOW TO GET STARTED WITH KEN-DOS"
This manual contains a brief description of the possibillities of KEN-DOS and syntax with short example of all commands. (a comprehensive manual with programming examples will be available later this year)

Optional CP/M 2.2 bios in EPROM (soon available)

The whole system comes assembled and tested.You only have place the EPROM-card on the bus-connector inside your DAI pc.
A simple hardware modification has to be made which consists of soldering a wire between two points on the DAI pc-board. This is to allow both single and double density. Each unit comes with clear instructions how to perform this minor modification.

For more information,prices and conditions of sale,please write or call ;

For Belgium :
MIKROSHOP HAGELAND
HERSELTSESTEENWEG 103
3220 AARSCHOT
BELGIUM
(TEL : 016 / 56 87 70)

All other countries :
MIPI v.o.f.
P.O. BOX 40
1616 ZG HOOGKARSPEL
THE NETHERLANDS

Kind regards,

KENNETH GOOSWIT.

ken-DOS

BASIC MONITOR part 2

In a previous Newsletter, part 1 is published. It describes in general the working of the BASIC monitor routine in the DAI firmware. This second article will explain some functions of the monitor in detail.

INITIALISATION (#C80C-#C843):

The monitor routine can be entered on different entrypoints, The 'normal' entrypoint is #C818. Here the monitor is entered after a reset and here it returns after encoding of a program line and after an 'END'-statement in a runned program.

Other entrypoints before #C818 are #C80C and #C814. Both are addresses to which the monitor returns after finishing the execution of certain tasks:

#C80C: Return after a 'hard-break': '*** BREAK' is printed before the monitor is restarted.
#C814: Return after a run-time error: the keyboard has to be re-defined as input source.

#C818: Because it is a 'fresh' (re)start of the monitor, the stackpointer is reset to #F900 (the stack top level), and its value is preserved in pointer #0127. This pointer is e.g. important if a program has to be restarted with a 'CONT' after a soft-break. The flag for suspended programs - programs which are interrupted by a soft-break - is cleared: there are at this moment no suspended programs.

Another entrypoint is #C823. It is used by the monitor itself after finishing the execution of a direct command, after a soft-break, after 'STOP', etc.

#C823: The stackpointer is set to the value which is stored in the pointer #0127. On entry at #C818 or before this has no effect, but if a program run is interrupted by a soft-break, then this preserved stackpointer is of importance to enable the continuation of the suspended program.

#C827: All pointers indicating some activity are cleared:
(V1.0) #0100: Current line number. This is the linenumber in the BASIC program on which the monitor is busy executing the program.
#0104: Pointer to current loop variable. When running a FOR-NEXT loop, this pointer indicates the variable which is used in this loop.
#0113: Stack level at last GOSUB. Used to remember its origin when a subroutine is called.
#0122: Flag for encoding a stored line. It indicates

that the monitor is busy with encoding.
#0117: Flag indicating that an 'INPUT'-statement in a program is being executed.
#0118: Flag indicating that a BASIC prgram is being runned.

In BASIC V1.1, an update is made. Before resetting all the pointers mentioned above, the keyboard pointers are reset: the key input buffer is cleared to avoid keybounce via a CALL to #D563.

#C840: The keyboard interrupts are enabled as well as the clock interrupts in order to have an accessible keyboard and to have outputs to the screen.

Now the monitor is initialised and ready to handle direct commands or program lines to be stored in the textbuffer.

ENCODING OF DIRECT COMMANDS AND PROGRAM LINES:

Any direct command and any program line typed in has to be translated into the code which can be understood by the BASIC run-time execution module. In other words: A conversion must be made to the code which is required for storage in the textbuffer. See Newsletter 11, page 196 for more details on this pseudo-code.

This conversion is required both for direct commands and for textlines to be stored. In both cases, the conversion is done via the encoded input buffer EBUF (#013E-#01BD). In this buffer, from #013F onwards, the encoded line is put together in bits and pieces by the BASIC encoding routines. When done, the length of the command line is stored in #013E.

After encoding, a direct command line (that means a command or a line without a linenumber) remains in the EBUF and is executed immediately. A program line (a line with a line number) is stored in the textbuffer.

The source for encoding a textline can be different. It is determined by the encoding input switch #0135:

#00: Input source is the keyboard or a user defined input routine DINC.
#01: Inputs from a string.
#02: The editbuffer is the source to get input lines from.

Values for #0135 >2 are invalid and may cause problems (see part 1 of this series).

For a clear view on how encoding proceeds, let's examine the encoding of a program line. This routine can be found on #C918.

#C918: The HL register pair is loaded with the startaddress of the EBUF, the location where to store (parts of) the line when it is encoded.

#C91B: Via RST1/03 the line number is encoded:

#3E72A: #3E731: The first character is taken from the input source (depending on #0135). Via #C024, the linenumber - in ASCII-code - is read from the input line into the math.accumulator MACC in a binary form. If the line number is <>0 and not >#FFFF, then it is moved into the EBUF at the first 2 locations and the EBUF input pointer (HL) is updated.
If the line number is incorrect, the error message 'NUMBER OUT OF RANGE' is printed and the encoding is aborted: the monitor is re-started.

#C91D: If in the input line, the line number is followed by a 'CR' only, then the line with the same line number is deleted from the textbuffer via #C9A2 and the encoding is done.
If BASIC statements follow the line number, these statements are encoded as well.

#C929: A mask (#40) is loaded into the D-register. This mask indicates a 'stored command'. If this mask is #80, it is used for a direct command (see #C86D). The mask is used for testing if the BASIC-command is correctly used, because not all commands are valid as direct commands or as commands used in a program (see table #CBBF).

#C92B: #C93C: Now the program line will be encoded via a CALL to #C93C. At first, the current stackpointer is preserved in #011D. This is important to be able to return to the point where the encoding starts in case a failure might occur during the encoding of the input line. The flag #0122 is set to indicate that the monitor is busy encoding a line, and the encoding of the line is executed via RST1/00: a call to ROM-bank 3, to #3E024.

#3E024 #3E031 #3E035 After putting an address on stack for returning after an encoding routine is finished, this routine searches the table #CBBF to find a match with the BASIC-command in the input line. If it is found, the code for this particular command is verified with the mask in the D-register to see if it is a valid command (error 'COMMAND INVALID' if not).

#3E040 #3E046 Now the command code is transformed into the token, and preserved in the E-register. The address of the encoding routine belonging to this BASIC command is fetched from the table and put on stack.

#3E04A The token is stored in the encoded input buffer EBUF, the EBUF pointer is updated via #3E018, and via the RET-instruction the 8080 goes to the encoding routine, because its address is the last one on stack.

A lot of examples of encoding routines can be found from #3E05F onwards.

#3E04F When the encoding of a BASIC command is finished, the next character is fetched from the input line. This can only be a CR - whole input line encoded and ready - or a ';', else a 'SYNTAX ERROR' is the result. In case of ';', the next BASIC command will be encoded by returning to #3E024.

#C94F: When the whole input line is encoded, the saved input line pointer BC and the EBUF input pointer HL on stack are cancelled. They where only saved on stack in case of an error found in the input line during encoding.
#C952: The flag for encoding an input line is cleared.

#C92E: The length of the encoded input line in the encoded input buffer EBUF is stored at its beginning.
#C935: Now the old program line with the same line number - if present - is deleted from the textbuffer (#C9A2), and the new line is inserted (#C9BD).

Encoding of a direct command (#C86D-#C87D) is done following the same principles. But now the encoded line remains in the EBUF with '00' at the end - indicating the end of the 'program',and the line is runned immediately.

(C) - Jan Boerrigter, July 1983

CORRECTIONS FIRMWARE MANUAL - 2

The following updates can be made in your copy of the DAI pC firmware manual:

#C87F: The B-register is loaded with the first byte of the EBUF in case of a direct command line, or with the first byte of the text line in the textbuffer in case of a stored program line.

#CBBF: In the header, the type bytes are incorrectly indicated.
If bit 6=1, then the command is valid during a program run.
If bit 7=1, then the command can be used in a direct command line.

Jan Boerrigter
Fabritiusstraat 15
6174 RG Sweikhuizen
tel. 04493-2093

EPROM-PROGRAMMER

```
PAGE 01    EPROM programmer 2716/2732

002                    *
003                    *
004                    *       By  A . BEUCKELAERS
005                    *       Ellebroecken 1
006                    *       2510  Mortsel.
007                    *       Tel 03/4496301
008                    *
009                    RICIN   EQU   :D8E0     RIC inputroutine
010                    RICOUT  EQU   :D8C8     RIC outputroutine
011                    AIN     EQU   :2390     Kanaal A als input
012                    AOUT    EQU   :2380     Kanaal B als output
013                    MONIT   EQU   :EA42     terug naar UTility
014                            ORG   :A000
015 A000 2101A2        INIT    LXI   H,MES1    pointer message 1
016 A003 CDC3A1                CALL  PNT$      printen message 1
017 A006 2112A2                LXI   H,MES2    pointer message 2
018 A009 CDC3A1                CALL  PNT$      printen message 2
019 A00C CD06ED        TYPE    CALL  :ED06     inlezen karakter per klavier
020 A00F E60F                  ANI   :0F       op nul stellen b7 tot b4
021 A011 17                    RAL             x4
022 A012 17                    RAL
023 A013 FE08                  CPI   :8
024 A015 CA2EA0                JZ    TYPE16    indien 2716
025 A018 FE10                  CPI   :10       indien <> 2732
026 A01A C23FA0                JNZ   ERRTYP    dan fout
027 A01D 21FF0F        TYPE32  LXI   H,:0FFF   indien 2732
028 A020 22C4A2                SHLD  BUFLEN    laden bufferlengte
029 A023 32CAA2                STA   NMBLK     laden aantal blokken
030 A026 3E20                  MVI   A,32
031 A028 32CDA2                STA   ROMTYP    laden Romtype met 32
032 A02B C347A0                JMP   SELCRD    springen naar selektierout.
033 A02E 21FF07        TYPE16  LXI   H,:07FF   indien 2716
034 A031 22C4A2                SHLD  BUFLEN    zelfde cyclus doorlopen
035 A034 32CAA2                STA   NMBLK
036 A037 3E10                  MVI   A,16
037 A039 32CDA2                STA   ROMTYP
038 A03C C347A0                JMP   SELCRD
039 A03F 0E3F          ERRTYP  MVI   C,'?'     foutmeldingsroutine
040 A041 CDB4EE                CALL  :EEB4
041 A044 C30CA0                JMP   TYPE
042 A047 119023        SELCRD  LXI   D,AIN     selekteren kan A als input
043 A04A CDC8D8                CALL  RICOUT
044 A04D 115022                LXI   D,:2250   deselketeren CS
045 A050 CDC8D8                CALL  RICOUT
046 A053 213BA2        INSERT  LXI   H,MES3    pointer op message 3
047 A056 CDC3A1                CALL  PNT$      printen message
048 A059 CD06ED        WAITSP  CALL  :ED06     wachten op SPACE
049 A05C FE20                  CPI   :20
050 A05E C259A0                JNZ   WAITSP
051 A061 CD3AED        PROMPT  CALL  :ED3A     printen 'A'
052 A064 0E5B                  MVI   C,'A'
053 A066 CDB4EE                CALL  :EEB4
054 A069 CD06ED        GETCMD  CALL  :ED06     inlezen bevel
```

```
055 A06C 21A9A1          LXI   H,CMDTAB
056 A06F 23        CMD   INX   H          vergelijken met tabel
057 A070 BE              CMP   M
058 A071 DAB7A1          JC    ERROR
059 A074 23              INX   H
060 A075 5E              MOV   E,M        laden adres routine
061 A076 23              INX   H
062 A077 56              MOV   D,M
063 A078 C26FA0          JNZ   CMD
064 A07B EB              XCHG             starten routine
065 A07C E9              PCHL
066              *
067              * Leesbevel
068              *
069 A07D CD83A0    RDCMD CALL  READ
070 A080 C361A0          JMP   PROMPT
071 A083 110022    READ  LXI   D,:2200    selekteren RIC
072 A086 CDC8D8          CALL  RICOUT
073 A089 2100A3          LXI   H,BUFFER   pointer begin buffer
074 A08C 3ACAA2          LDA   NMBLK      laden aantal blokken
075 A08F 47              MOV   B,A
076 A090 0E00            MVI   C,:00      byteteller op 0
077 A092 110022          LXI   D,:2200    laden MSB adres
078 A095 D5        NXTBLK PUSH  D
079 A096 CDC8D8          CALL  RICOUT
080 A099 110021          LXI   D,:2100    laden LSB adres
081 A09C D5        NEXT  PUSH  D
082 A09D CDC8D8          CALL  RICOUT
083 A0A0 1620            MVI   D,:20      selekteren kanaal A
084 A0A2 CDE0D8          CALL  RICIN      lezen van 1 byte
085 A0A5 73              MOV   M,E        en naar buffer brengen
086 A0A6 23              INX   H          bufferpointer ophogen
087 A0A7 D1              POP   D
088 A0A8 13              INX   D          adresteller ophogen
089 A0A9 0D              DCR   C          byteteller-1
090 A0AA C29CA0          JNZ   NEXT       volgende byte lezen
091 A0AD D1              POP   D          zoniet
092 A0AE 13              INX   D          MSB adres +1
093 A0AF 05              DCR   B          blokkenteller -1
094 A0B0 C295A0          JNZ   NXTBLK     volgende blok lezen
095 A0B3 115022          LXI   D,:2250    zoniet deselekt EPROM
096 A0B6 CDC8D8          CALL  RICOUT
097 A0B9 C9              RET
098              *
099              * Testbevel
100              *
101 A0BA CDC0A0    TSTCMD CALL TEST
102 A0BD C361A0          JMP   PROMPT
103 A0C0 CD83A0    TEST  CALL  READ       EPROM lezen
104 A0C3 2AC4A2          LHLD  BUFLEN
105 A0C6 23              INX   H
106 A0C7 EB              XCHG
107 A0C8 2100A3          LXI   H,BUFFER   bufferpointer op 0
108 A0CB 7E        NXTTST MOV   A,M        byte in accu
109 A0CC FEFF            CPI   :FF        vergelijken met FF
110 A0CE C2E2A0          JNZ   NEMP       indien niet foutmelding
111 A0D1 23              INX   H          indien ja
112 A0D2 1B              DCX   D          lengte -1
113 A0D3 7B              MOV   A,E        testen op 0
114 A0D4 B2              ORA   D
115 A0D5 C2CBA0          JNZ   NXTTST     volgende byte
116 A0D8 CD3AED          CALL  :ED3A      print CR LF
117 A0DB 2185A2          LXI   H,MES7     pointer message 7
```

```
118 A0DE CDC3A1          CALL  PNT$      printenmessage
119 A0E1 C9              RET
120 A0E2 2157A2   NEMP   LXI   H,MES4    foutmelding
121 A0E5 CD3AED          CALL  :ED3A     indien EPROM niet
122 A0E8 CDC3A1          CALL  PNT$      volledig gewist
123 A0EB C361A0          JMP   PROMPT
124 A0EE C9              RET
125                *
126                * Programmeerbevel
127                * P beginadr eindadr beginadr eprom.
128                *
129 A0EF 0E03     PGMCMD MVI   C,3
130 A0F1 CDDEEA          CALL  :EADE     lezen 3 adressen
131 A0F4 0D              DCR   C
132 A0F5 F2B7A1          JP    ERROR
133 A0F8 C1              POP   B         beginadres EPROM
134 A0F9 D1              POP   D         eindadres EPROM
135 A0FA E1              POP   H         beginadres source
136 A0FB C5              PUSH  B
137 A0FC 22C8A2          SHLD  SRCBUF    naar sourcebuffer
138 A0FF 7B              MOV   A,E       berekening lengte
139 A100 95              SUB   L
140 A101 6F              MOV   L,A
141 A102 7A              MOV   A,D
142 A103 9C              SBB   H
143 A104 67              MOV   H,A
144 A105 23              INX   H
145 A106 22C6A2          SHLD  LENGTH    wegschrijven lengte
146 A109 22CBA2          SHLD  LENGTE
147 A10C 21B1A2          LXI   H,MES10   pointer message 10
148 A10F CD2FED          CALL  :ED2F     printen message 10
149 A112 118023          LXI   D,AOUT    kanaal A in output
150 A115 CDC8D8          CALL  RICOUT
151 A118 3ACDA2          LDA   ROMTYP
152 A11B FE10            CPI   16
153 A11D CA23A1          JZ    EPR16
154 A120 C228A1          JNZ   EPR32
155 A123 1EA0     EPR16  MVI   E,:A0      besturingswoord  2716
156 A125 C32AA1          JMP   GO
157 A128 1EB0     EPR32  MVI   E,:B0      besturingswoord 2732
158 A12A 2AC8A2   GO     LHLD  SRCBUF    laden HL bufferpointer
159 A12D C1              POP   B         beginadres uit stack
160 A12E 1622            MVI   D,:22     laden MSB adres
161 A130 78              MOV   A,B
162 A131 B3              ORA   E
163 A132 5F              MOV   E,A
164 A133 D5       NXTBLC PUSH  D
165 A134 CDC8D8          CALL  RICOUT    em naar RIC
166 A137 59              MOV   E,C
167 A138 1621            MVI   D,:21     laden LSB adres
168 A13A D5       NXTBY  PUSH  D
169 A13B CDC8D8          CALL  RICOUT    en naar RIC
170 A13E 1620            MVI   D,:20     selekteren kanaal A
171 A140 5E              MOV   E,M       programmeren 1 byte
172 A141 CDC8D8          CALL  RICOUT
173 A144 CDCDA1          CALL  IMPULS    programmeerimpuls
174 A147 23              INX   H
175 A148 E5              PUSH  H
176 A149 2AC6A2          LHLD  LENGTH    lengte -1
177 A14C 2B              DCX   H
178 A14D 7D              MOV   A,L
179 A14E B4              ORA   H
180 A14F 22C6A2          SHLD  LENGTH    terug wegschrijven
```

```
181 A152 E1              POP   H
182 A153 CA63A1          JZ    DONE      indien 0 dan einde
183 A156 D1              POP   D
184 A157 1C              INR   E         adresteller+1
185 A158 7B              MOV   A,E       laatste adres van blok?
186 A159 FE00            CPI   0
187 A15B C23AA1          JNZ   NXTBY      volgende byte
188 A15E D1              POP   D
189 A15F 1C              INR   E         blokkenteller +1
190 A160 C333A1          JMP   NXTBLC    volgende blok
191 A163 115022    DONE  LXI   D,:2250   deselekteren EPROM
192 A166 CDC8D8          CALL  RICOUT
193 A169 21BFA2          LXI   H,MES11   pointer message 11
194 A16C CD2FED          CALL  :ED2F     printen message 11
195 A16F CD83A0          CALL  READ      verifieren EPROM in-
196 A172 2ACBA2          LHLD  LENGTE    houd byte per byte
197 A175 EB              XCHG
198 A176 2AC8A2          LHLD  SRCBUF
199 A179 0100A3          LXI   B,BUFFER
200 A17C 0A        NXTVER LDAX  B
201 A17D BE              CMP   M
202 A17E C298A1          JNZ   ERR
203 A181 23              INX   H
204 A182 03              INX   B
205 A183 1B              DCX   D
206 A184 7B              MOV   A,E
207 A185 B2              ORA   D
208 A186 C27CA1          JNZ   NXTVER
209 A189 115022          LXI   D,:2250
210 A18C CDC8D8          CALL  RICOUT
211 A18F 217BA2          LXI   H,MES6
212 A192 CDC3A1          CALL  PNT$
213 A195 C361A0          JMP   PROMPT
214 A198 E5        ERR   PUSH  H
215 A199 219BA2          LXI   H,MES9    pointer message 9
216 A19C CD3AED          CALL  :ED3A
217 A19F CD2FED          CALL  :ED2F     printen message 9
218 A1A2 E1              POP   H
219 A1A3 CD18ED          CALL  :ED18     printen adres
220 A1A6 C361A0          JMP   PROMPT
221                *
222                * bevelentabel
223                *
224 A1A9 00        CMDTAB DATA  00
225 A1AA 50               DATA  'P'
226 A1AB EFA0             DBL   PGMCMD
227 A1AD 52               DATA  'R'
228 A1AE 7DA0             DBL   RDCMD
229 A1B0 54               DATA  'T'
230 A1B1 BAA0             DBL   TSTCMD
231 A1B3 55               DATA  'U'
232 A1B4 42EA             DBL   :EA42
233 A1B6 FF               DATA  :FF
234 A1B7 CD3AED    ERROR  CALL  :ED3A
235 A1BA 0E3F             MVI   C,'?'
236 A1BC 2A5B00           LHLD  :5B
237 A1BF F9               SPHL
238 A1C0 C361A0           JMP   PROMPT
239 A1C3 CD3AED    PNT$   CALL  :ED3A
240 A1C6 CD2FED           CALL  :ED2F
241 A1C9 CD3AED           CALL  :ED3A
242 A1CC C9               RET
243 A1CD 3ACDA2    IMPULS LDA   ROMTYP    programmeerimp  2716
```

```
244 A1D0 FE10            CPI   16
245 A1D2 C2E5A1          JNZ   IMP2
246 A1D5 110923          LXI   D,:2309
247 A1D8 CDC8D8          CALL  RICOUT
248 A1DB CDF5A1          CALL  DELAY
249 A1DE 110823          LXI   D,:2308
250 A1E1 CDC8D8          CALL  RICOUT
251 A1E4 C9              RET
252 A1E5 110823    IMP2  LXI   D,:2308   programmeerimp 2732
253 A1E8 CDC8D8          CALL  RICOUT
254 A1EB CDF5A1          CALL  DELAY
255 A1EE 110923          LXI   D,:2309
256 A1F1 CDC8D8          CALL  RICOUT
257 A1F4 C9              RET
258 A1F5 E5        DELAY PUSH  H         lengte impuls
259 A1F6 217008          LXI   H,:0870
260 A1F9 2B        LUS   DCX   H
261 A1FA 7D              MOV   A,L
262 A1FB B4              ORA   H
263 A1FC C2F9A1          JNZ   LUS
264 A1FF E1              POP   H
265 A200 C9              RET
266 A201 0C        MES1  DATA  :0C
267 A202 50524F          ASC   'PROM PROGRAMMER'
268 A211 00              DATA  00
269 A212 574849    MES2  ASC   'WHICH TYPE OF EPROM'
270 A225 0D              DATA  :0D
271 A226 282020          ASC   '(  2716=2  2732=4  )'
272 A23A 00              DATA  00
273 A23B 494E53    MES3  ASC   'INSERT EPROM AND TYPE SPACE'
274 A256 00              DATA  00
275 A257 455052    MES4  ASC   'EPROM NOT EMPTY'
276 A266 00              DATA  00
277 A267 4E4F20    MES5  ASC   'NO MATCH AT ADDRESS'
278 A27A 00              DATA  00
279 A27B 4E4F20    MES6  ASC   'NO ERRORS'
280 A284 00              DATA  00
281 A285 524541    MES7  ASC   'READY FOR PROGRAMMIMG'
282 A29A 00              DATA  00
283 A29B 202020    MES9  ASC   '   ERROR AT ADDRESS  '
284 A2B0 00              DATA  00
285 A2B1 50524F    MES10 ASC   'PROGRAMMING  '
286 A2BE 00              DATA  00
287 A2BF 444F4E45  MES11 ASC   'DONE'
288 A2C3 00              DATA  00
289 A2C4           BUFLEN RES  2
290 A2C6           LENGTH RES  2
291 A2C8           SRCBUF RES  2
292 A2CA           NMBLK  RES  1
293 A2CB           LENGTE RES  2
294 A2CD           ROMTYP RES  1
295 A2CE           CONST  RES  1
296                       ORG  :A300
297 A300           BUFFER RES  :1000,00
298 B300                  END
```

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

AIN     2390   AOUT   2380   BUFFER A300   BUFLEN A2C4
CMD     A06F   CMDTAB A1A9   CONST  A2CE   DELAY  A1F5
DONE    A163   EPR16  A123   EPR32  A128   ERR    A198
```

```
ERROR  A1B7   ERRTYP A03F   GETCMD A069   GO     A12A
IMP2   A1E5   IMPULS A1CD   INIT   A000   INSERT A053
LENGTE A2CB   LENGTH A2C6   LUS    A1F9   MES1   A201
MES10  A2B1   MES11  A2BF   MES2   A212   MES3   A23B
MES4   A257   MES5   A267   MES6   A27B   MES7   A285
MES9   A29B   MONIT  EA42   NEMP   A0E2   NEXT   A09C
NMBLK  A2CA   NXTBLC A133   NXTBLK A095   NXTBY  A13A
NXTTST A0CB   NXTVER A17C   PGMCMD A0EF   PNT$   A1C3
PROMPT A061   RDCMD  A07D   READ   A083   RICIN  D8E0
RICOUT D8C8   ROMTYP A2CD   SELCRD A047   SRCBUF A2C8
TEST   A0C0   TSTCMD A0BA   TYPE   A00C   TYPE16 A02E
TYPE32 A01D   WAITSP A059
```

EPROM programmeerapparaat voor 2716 en 2732

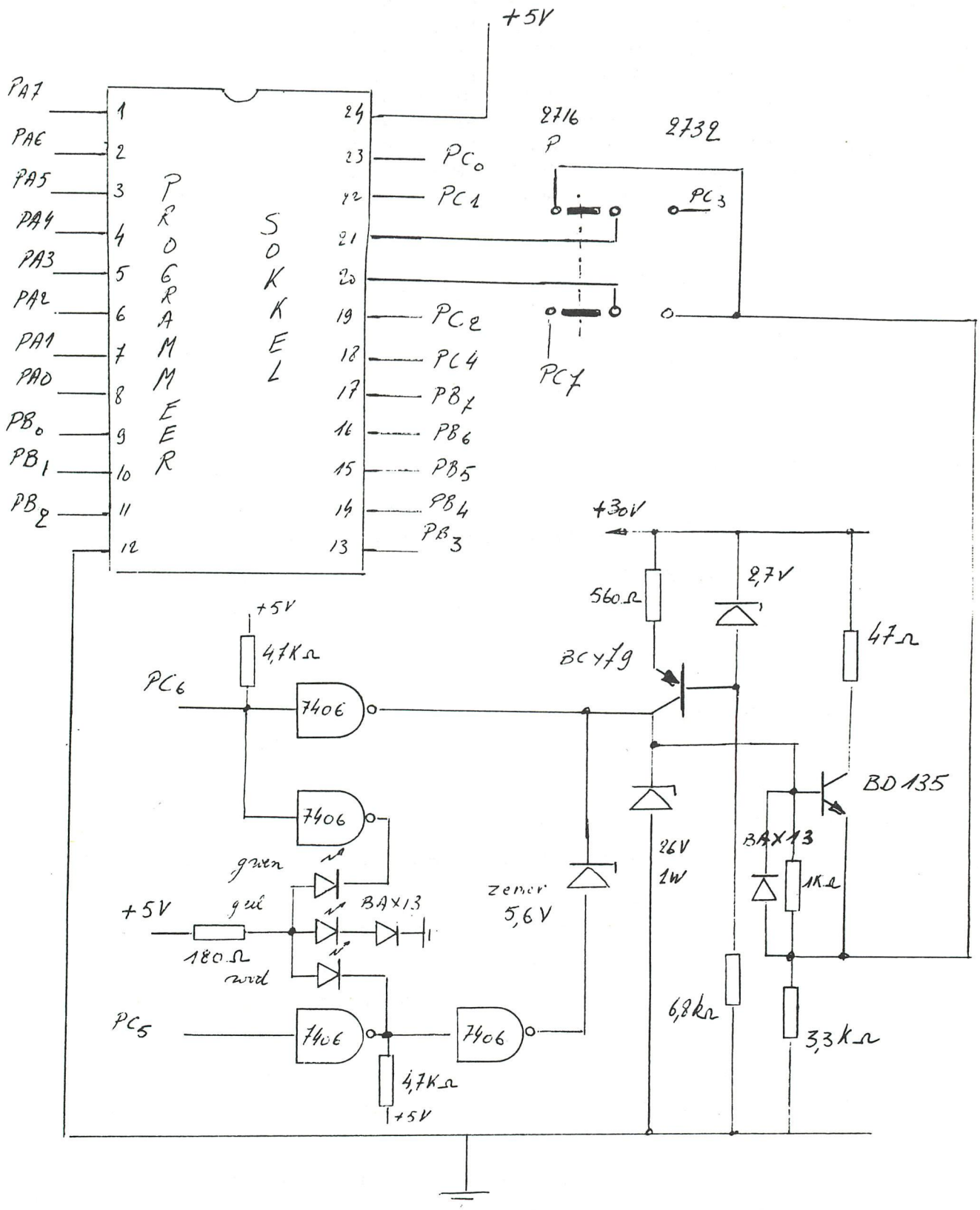
Het programmeerapparaat kan gemonteerd worden op het WIRE-WRAP veld van een universele PC DCE BUS INTERFACEKAART, en kan dus parallel met een snelle cassette en alle eventuele verdere DCE kaarten gebruikt worden. Er kunnen zowel 2K als 4K EPROMS gebruikt worden. Het programma herkent een aantal bevelen, die gegeven worden door het intikken van een karakter, al dan niet gevolgd door het aangeven van een of meerdere adressen.
Na het starten van het programma meldt het systeem zich met
PROM PROGRAMMER.

WHICH TYPE OF EPROM
(2716=2 2732=4).

Na het intikken van 2 of 4 alnaargelang men met een 2716 of 2732 wenst te werken, meldt het programma zich terug met
INSERT EPROM AND TYPE SPACE
Nu kan u zonder gevaar de EPROM in de programmeersokkel steken (groene LED brandt) en daarna een SPACE intikken.
Het systeem meldt zich met de prompt en wacht nu op het intikken van een bevel.
Het is normaal dat men eerst wenst te testen of de EPROM leeg is of behoorlijk gewist is. Dit kan gebeuren door het ingeven van het bevel (T).
Het programma leest (gele LED brandt) de ganse EPROM en verifieert of inderdaad in elke geheugenplaats FF staat. Is zulks het geval dan komt de melding
READY FOR PROGRAMMING
In het tegenovergestelde geval komt de melding
EPROM NOT EMPTY
Wenst men nu over te gaan tot de programmering van de EPROM dan kan dit gebeuren met het bevel (P). Dit bevel dient,om volledig te zijn,gevolgd te worden door drie hexadecimale ad-

cont. on p.411

SCHEMA EPROM PROGRAMMER



$PA_0 \div PA_7$, $PB_0 \div PB_7$, $PC_0 \div PC_7$ zijn de overeenkomstige klemmen van de 8255 op de DCE interfacekaart.

program identification

title : MAZE GAME

author : M. DIERCKX

purpose : try to move the object through the maze

comment :

```
1     MODE 0
2     REM M.DIERCKX
3     REM IMP INT
10    REM het maken van een doolhof
20    MODE 2:COLORG 0 8 8 0
80    FOR I=0 TO XMAX STEP 10:DRAW I,0 I,YMAX 21:NEXT
90    FOR I=0 TO YMAX STEP 10:DRAW 0,I XMAX,I 21:NEXT
130   EIND=RND(30)+60
135   X1=0:Y1=0
140   FOR I=1 TO EIND
150   IF INT(RND(2))=0 THEN FLAG=1:X=RND(XMAX):DRAW X1,Y1 X,Y1 0:X1=X
160   IF FLAG=0 THEN Y=RND(YMAX):DRAW X1,Y1 X1,Y 0:Y1=Y
165   FLAG=0
170   NEXT I
175   DRAW X1,Y1 X1,YMAX 0
176   DRAW X1,YMAX XMAX,YMAX 0
180   COLORG 0 8 12 14
190   X=0:Y=0
191   DOT X,Y 0
192   X2=RND(XMAX):Y2=RND(YMAX):IF SCRN(X2,Y2)=0 THEN DOT X2,Y2 14
195   IF X=XMAX AND Y=YMAX GOTO 2000
196   ON RND(4)+1 GOTO 220,240,260
200   XI=X+1:IF XI>XMAX GOTO 195
210   IF SCRN(XI,Y)=0 THEN DOT XI,Y 12:GOSUB 1000:DOT X,Y 0:X=XI:GOTO 200
215   GOTO 195
220   YI=Y+1:IF YI>YMAX GOTO 195
230   IF SCRN(X,YI)=0 THEN DOT X,YI 12:GOSUB 1000:DOT X,Y 0:Y=YI:GOTO 220
235   GOTO 195
240   XI=X-1:IF XI<0 GOTO 195
250   IF SCRN(XI,Y)=0 THEN DOT XI,Y 12:GOSUB 1000:DOT X,Y 0:X=XI:GOTO 240
255   GOTO 195
260   YI=Y-1:IF YI<0 GOTO 195
270   IF SCRN(X,YI)=0 THEN DOT X,YI 12:GOSUB 1000:DOT X,Y 0:Y=YI:GOTO 260
280   GOTO 195
1000  WAIT TIME 2:SEC=SEC+1:Q=GETC:IF Q=0 THEN RETURN
1010  IF Q=19 THEN X2I=X2+1:IF X2I<=XMAX THEN IF SCRN(X2I,Y2)=0 THEN DOT X2I,Y2
14:DOT X2,Y2 0:X2=X2I:RETURN
1020  IF Q=18 THEN X2I=X2-1:IF X2I>=0 THEN IF SCRN(X2I,Y2)=0 THEN DOT X2I,Y2 14:
DOT X2,Y2 0:X2=X2I:RETURN
1030  IF Q=16 THEN Y2I=Y2+1:IF Y2I<=YMAX THEN IF SCRN(X2,Y2I)=0 THEN DOT X2,Y2I
14:DOT X2,Y2 0:Y2=Y2I:RETURN
1040  IF Q=17 THEN Y2I=Y2-1:IF Y2I>=0 THEN IF SCRN(X2,Y2I)=0 THEN DOT X2,Y2I 14:
DOT X2,Y2 0:Y2=Y2I:RETURN
1050  RETURN
2000  MODE 0
2010  CURSOR 10,10:PRINT " u hebt er ";SEC/70;"  seconden over gedaan"
2020  Q=GETC:IF Q=0 GOTO 2020
2030  GOTO 10
```

VIDEO RAM TABLE


```
001                ******************************************
002                *                                        *
003                *        Video RAM table generator       *
004                *                                        *
005                *          by Salvatore PENNISI          *
006                * Via Mario Borsa 63 - 00159 ROMA (ITALY) *
007                ******************************************
008                *                                        *
009                *   This program generates a table with  *
010                * the addresses of the lines of the video *
011                * RAM in any MODE.                       *
012                *   It is particularly useful when the   *
013                * speed is very important   (games,      *
014                * graphics,...).                         *
015                *   In fact, having the y-coordinate,    *
016                * it is possible to know the address of  *
017                * line without time consuming            *
018                * calculations.                          *
019                *                                        *
020                ******************************************
021                *                                        *
022                *         How to use this program        *
023                *  1) Load this program                  *
024                *     UT                                 *
025                *     R                                  *
026                *     Z3                                 *
027                *     G300                               *
028                *  2) Write in HEX the start address of  *
029                *     video RAM table to merge after     *
030                *     in your ASSEMBLER program          *
031                *  3) Select the MODE                    *
032                *  OK. The table is created, save it     *
033                *      on tape by the W command          *
034                ******************************************
035                * N.B. For example the video RAM table   *
036                *      in MODE 5 (512 bytes) is :        *
037                *      Start address                     *
038                *      6649 66A3 66FD 6757 ... BF95 BFEF *
039                ******************************************
040                * INDIRIZZI ROM
041                *
042                RSTART  EQU   :C80C
043                USTART  EQU   :E009
044                GETFRC  EQU   :D6BE     INPUT DA TASTIERA
045                PMSG    EQU   :DAD4     PRINT DI UN MESSAGGIO
046                OUTC    EQU   :DD60     OUTPUT DI UN CARATTERE
047                ALFNUM  EQU   :DE09     CONTROLLO 0-9 A-Z
048                MODE0   EQU   :FF
049                *
050                * COSTANTI MODE 0
051                *
052                INCR0   EQU   :86
053                VB0     EQU   :B35F     :B3E5-:86
054                COUNT0  EQU   :18
055                *
056                * COSTANTI MODE 1/2
057                *
058                INCR1   EQU   :18
059                VB1     EQU   :B9D7     :B9EF-:18
060                COUNT1  EQU   :41
061                *
```

```
062                     * COSTANTI MODE 3/4

PAGE 02

063                     *
064                     INCR3   EQU   :2E
065                     VB3     EQU   :A893     :A8C1-:2E
066                     COUNT3  EQU   :82
067                     *
068                     * COSTANTI MODE 5/6
069                     *
070                     INCR5   EQU   :5A
071                     VB5     EQU   :65EF     :6649-:5A
072                     COUNT5  EQU   :FF       CONTATORE RIGHE
073                     *
074                     * COSTANTI ASCII
075                     *
076                     FF      EQU   :0C       FORMFEED
077                     CR      EQU   :0D       CARRIAGE RETURN
078                     DELC    EQU   :08       DEL CHAR
079                     CAR0    EQU   :30       ZERO
080                     CAR1    EQU   :31       UNO
081                     CAR3    EQU   :33       TRE
082                     CAR5    EQU   :35       CINQUE
083                     *
084                     CARG    EQU   :47       G
085                     *
086                             ORG   :300
087                     *
088                     * IMPOSTA MODE 0
089                     *
090 0300 3EFF           START   MVI   A,MODE0
091 0302 EF                     RST   5
092 0303 18                     DATA  :18
093                     *
094                     * PRINT CHR$(12)
095                     *
096 0304 217C04                 LXI   H,CHR12$
097 0307 CDD4DA                 CALL  PMSG
098                     *
099                     * PRINT MESS0
100                     *
101 030A 217E04                 LXI   H,MESS0
102 030D CDD4DA                 CALL  PMSG
103                     *
104                     * PRINT MESS1
105                     *
106 0310 21A104         ME1     LXI   H,MESS1
107 0313 CDD4DA                 CALL  PMSG
108 0316 CDBED6         GETC    CALL  GETFRC
109 0319 CA1603                 JZ    GETC
110 031C FE08                   CPI   DELC
111 031E CA1003                 JZ    ME1
112 0321 CD09DE                 CALL  ALFNUM
113 0324 D21003                 JNC   ME1
114 0327 FE47                   CPI   CARG
115 0329 DA2F03                 JC    OK
116 032C C31003                 JMP   ME1
117 032F CD60DD         OK      CALL  OUTC
118 0332 323805                 STA   IND
119 0335 CDBED6         GETC1   CALL  GETFRC
120 0338 CA3503                 JZ    GETC1
121 033B FE08                   CPI   DELC
122 033D CA1003                 JZ    ME1
123 0340 CD09DE                 CALL  ALFNUM
```

```
124 0343 D21003             JNC   ME1


PAGE 03

125 0346 FE47               CPI   CARG
126 0348 DA4E03             JC    OK1
127 034B C31003             JMP   ME1
128 034E CD60DD     OK1     CALL  OUTC
129 0351 323905             STA   IND+1
130 0354 CDBED6     GETC2   CALL  GETFRC
131 0357 CA5403             JZ    GETC2
132 035A FE08               CPI   DELC
133 035C CA1003             JZ    ME1
134 035F CD09DE             CALL  ALFNUM
135 0362 D21003             JNC   ME1
136 0365 FE47               CPI   CARG
137 0367 DA6D03             JC    OK2
138 036A C31003             JMP   ME1
139 036D CD60DD     OK2     CALL  OUTC
140 0370 323A05             STA   IND+2
141 0373 CDBED6     GETC3   CALL  GETFRC
142 0376 CA7303             JZ    GETC3
143 0379 FE08               CPI   DELC
144 037B CA1003             JZ    ME1
145 037E CD09DE             CALL  ALFNUM
146 0381 D21003             JNC   ME1
147 0384 FE47               CPI   CARG
148 0386 DA8C03             JC    OK3
149 0389 C31003             JMP   ME1
150 038C CD60DD     OK3     CALL  OUTC
151 038F 323B05             STA   IND+3
152                 *
153                 * TEST CARRIAGE RETURN
154                 *
155 0392 CDBED6     CRETUR  CALL  GETFRC
156 0395 CA9203             JZ    CRETUR
157 0398 FE08               CPI   DELC
158 039A CA1003             JZ    ME1
159 039D FE0D               CPI   CR
160 039F CAA503             JZ    CONV
161 03A2 C39203             JMP   CRETUR
162                 *
163                 * CONVERSIONE IND DA  ASCII IN HEX
164                 *
165                 *  PRIMO E SECONDO BYE
166                 *
167 03A5 213805     CONV    LXI   H,IND
168 03A8 7E                 MOV   A,M
169 03A9 FE40               CPI   :40
170 03AB DAB003             JC    ASCZ
171 03AE D607               SUI   :07
172 03B0 D630       ASCZ    SUI   CAR0
173 03B2 07                 RLC
174 03B3 07                 RLC
175 03B4 07                 RLC
176 03B5 07                 RLC
177 03B6 47                 MOV   B,A
178 03B7 23                 INX   H
179 03B8 7E                 MOV   A,M
180 03B9 FE40               CPI   :40
181 03BB DAC003             JC    ASCZ1
182 03BE D607               SUI   :07
183 03C0 D630       ASCZ1   SUI   CAR0
184 03C2 80                 ADD   B
185 03C3 323D05             STA   INDH+1
```

```
186                     *


PAGE 04

187                     * TERZO E QUARTO BYTE
188                     *
189 03C6 23                     INX   H
190 03C7 7E                     MOV   A,M
191 03C8 FE40                   CPI   :40
192 03CA DACF03                 JC    ASCZ2
193 03CD D607                   SUI   :07
194 03CF D630           ASCZ2   SUI   CAR0
195 03D1 07                     RLC
196 03D2 07                     RLC
197 03D3 07                     RLC
198 03D4 07                     RLC
199 03D5 47                     MOV   B,A
200 03D6 23                     INX   H
201 03D7 7E                     MOV   A,M
202 03D8 FE40                   CPI   :40
203 03DA DADF03                 JC    ASCZ3
204 03DD D607                   SUI   :07
205 03DF D630           ASCZ3   SUI   CAR0
206 03E1 80                     ADD   B
207 03E2 323C05                 STA   INDH
208                     *
209                     * CONTROLLO INDH MAGGIORE #055F
210                     *
211 03E5 3A3D05                 LDA   INDH+1
212 03E8 FE05                   CPI   :05
213 03EA DA1003                 JC    ME1
214 03ED C2F803                 JNZ   ME2
215 03F0 3A3C05                 LDA   INDH
216 03F3 FE5F                   CPI   :5F
217 03F5 DA1003                 JC    ME1
218                     *
219                     * SCELTA MODE
220                     *
221 03F8 21E304         ME2     LXI   H,MESS2
222 03FB CDD4DA                 CALL  PMSG
223 03FE CDBED6         GETC4   CALL  GETFRC
224 0401 CAFE03                 JZ    GETC4
225 0404 FE08                   CPI   DELC
226 0406 CAF803                 JZ    ME2
227 0409 FE30                   CPI   CAR0
228 040B CA2004                 JZ    M0
229 040E FE31                   CPI   CAR1
230 0410 CA2D04                 JZ    M1
231 0413 FE33                   CPI   CAR3
232 0415 CA3A04                 JZ    M3
233 0418 FE35                   CPI   CAR5
234 041A CA4704                 JZ    M5
235 041D C3FE03                 JMP   GETC4
236                     *
237                     * INIZIALIZZA MODE 0
238                     *
239 0420 CD60DD         M0      CALL  OUTC
240 0423 1618                   MVI   D,COUNT0
241 0425 1E86                   MVI   E,INCR0
242 0427 015FB3                 LXI   B,VB0
243 042A C35404                 JMP   INIZIO
244                     *
245                     * INIZIALIZZA MODE 1/2
246                     *
247 042D CD60DD         M1      CALL  OUTC
```

```
248 0430 1641              MVI   D,COUNT1


PAGE 05

249 0432 1E18              MVI   E,INCR1
250 0434 01D7B9            LXI   B,VB1
251 0437 C35404            JMP   INIZIO
252                  *
253                  * INIZIALIZZA MODE 3/4
254                  *
255                  *
256 043A CD60DD    M3      CALL  OUTC
257 043D 1682              MVI   D,COUNT3
258 043F 1E2E              MVI   E,INCR3
259 0441 0193A8            LXI   B,VB3
260 0444 C35404            JMP   INIZIO
261                  * INIZIALIZZA MODE 5/6
262                  *
263 0447 CD60DD    M5      CALL  OUTC
264 044A 16FF              MVI   D,COUNT5
265 044C 1E5A              MVI   E,INCR5
266 044E 01EF65            LXI   B,VB5
267 0451 C35404            JMP   INIZIO
268                  *
269                  * CREAZIONE TABELLA IND. VIDEO
270                  *
271 0454 2A3C05    INIZIO  LHLD  INDH
272                  *
273 0457 79        LOOP    MOV   A,C
274 0458 83                ADD   E
275 0459 4F                MOV   C,A
276 045A 78                MOV   A,B
277 045B CE00              ACI   :00
278 045D 47                MOV   B,A
279 045E 70                MOV   M,B
280 045F 23                INX   H
281 0460 71                MOV   M,C
282 0461 23                INX   H
283 0462 15                DCR   D
284 0463 C25704            JNZ   LOOP
285                  *
286                  * CALCOLO FUORI LOOP
287                  *
288 0466 79                MOV   A,C
289 0467 83                ADD   E
290 0468 4F                MOV   C,A
291 0469 78                MOV   A,B
292 046A CE00              ACI   :00
293 046C 47                MOV   B,A
294 046D 70                MOV   M,B
295 046E 23                INX   H
296 046F 71                MOV   M,C
297                  *
298                  * PRINT FINE
299                  *
300 0470 212B05    FINE    LXI   H,ENDMES
301 0473 CDD4DA            CALL  PMSG
302 0476 3100F9            LXI   SP,:F900
303 0479 C309E0            JMP   USTART
304                  *
305                  * MESSAGGIO CHR$(12)
306                  *
307 047C 0C        CHR12$  DATA  FF
308 047D 00                DATA  0
309                  *
```

```
310                     * MESSAGGI

PAGE 06

311                     *
312 047E 0D0D           MESS0   DATA  CR,CR
313 0480 202020                 ASC   '      Video RAM table generator'
314 049F 0D                     DATA  CR
315 04A0 00                     DATA  0
316 04A1 0D0D           MESS1   DATA  CR,CR
317 04A3 537461                 ASC   'Start addrress of table'
318 04BA 20696E                 ASC   ' in HEX'
319 04C1 0D                     DATA  CR
320 04C2 286164                 ASC   '(address greater then #055F)   #'
321 04E2 00                     DATA  0
322 04E3 0D0D           MESS2   DATA  CR,CR
323 04E5 30203D                 ASC   '0 = MODE 0'
324 04EF 0D                     DATA  CR
325 04F0 31203D                 ASC   '1 = MODE 1/2'
326 04FC 0D                     DATA  CR
327 04FD 33203D                 ASC   '3 = MODE 3/4'
328 0509 0D                     DATA  CR
329 050A 35203D                 ASC   '5 = MODE 5/6'
330 0516 0D0D                   DATA  CR,CR
331 0518 53656C                 ASC   'Select the MODE
332 052A 00                     DATA  0
333 052B 0D0D           ENDMES  DATA  CR,CR
334 052D 2A2A20                 ASC   '** DONE **'
335 0537 00                     DATA  0
336                     *
337                     * CAMPI DI LAVORO
338                     *
339 0538                IND     RES   4,0
340 053C                INDH    RES   2,0
341 053E                        END
```

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

ALFNUM DE09   ASCZ   03B0   ASCZ1  03C0   ASCZ2  03CF
ASCZ3  03DF   CAR0   0030   CAR1   0031   CAR3   0033
CAR5   0035   CARG   0047   CHR12$ 047C   CONV   03A5
COUNT0 0018   COUNT1 0041   COUNT3 0082   COUNT5 00FF
CR     000D   CRETUR 0392   DELC   0008   ENDMES 052B
FF     000C   FINE   0470   GETC   0316   GETC1  0335
GETC2  0354   GETC3  0373   GETC4  03FE   GETFRC D6BE
INCR0  0086   INCR1  0018   INCR3  002E   INCR5  005A
IND    0538   INDH   053C   INIZIO 0454   LOOP   0457
M0     0420   M1     042D   M3     043A   M5     0447
ME1    0310   ME2    03F8   MESS0  047E   MESS1  04A1
MESS2  04E3   MODE0  00FF   OK     032F   OK1    034E
OK2    036D   OK3    038C   OUTC   DD60   PMSG   DAD4
RSTART C80C   START  0300   USTART E009   VB0    B35F
VB1    B9D7   VB3    A893   VB5    65EF
```

NEW CHARS ON GP100

```
PAGE 01    MODERNISED CHARS FOR SEIKOSHA GP100

002                     *
003                             ORG     :2EC
004                     PPICMD  EQU     :FE03
005                     PPIP~A  EQU     :FE00
006                     PPIP~B  EQU     :FE01
007                     PPIP~C  EQU     :FE02
008 02EC 800000         CHRP$   DATA    :80,0,0,0,0,0 < CHAR. DATA WORD
009 02F2 0D             LSTCHR  DATA    :D
010 02F3 00             ACTCHR  DATA    :0
011 02F4 0F             PRMODE  DATA    :F              < ACTUAL PRINTER MODE
012 02F5 100000         PRPOS$  DATA    :10,0,0         < LEFT SPACE POSITION
013                     *
014                             ORG     :2FF
015 02FF 0E             LEFTSP  DATA    14              NUMBER OF SPACES LEFT-SIDE
016                     *
017 0300 E5             INIT    PUSH    H               INIT PARALLEL PRINT-SERV
018 0301 F5                     PUSH    PSW
019 0302 3EA0                   MVI     A,:A0
020 0304 3203FE                 STA     PPICMD          DCE 8255 COMMAND MODE 1
021 0307 3EFF                   MVI     A,:FF           .  Port A : DATA OUT
022 0309 3201FE                 STA     PPIP~B          .  Port B : DEV ADDR OUT
023 030C 21DD02                 LXI     H,:02DD         .  Port C : CONTROL I/O
024 030F 36C3                   MVI     M,:C3
025 0311 23                     INX     H
026 0312 3640                   MVI     M,:40
027 0314 23                     INX     H               ADJUST DOUTC
028 0315 3603                   MVI     M,:03           SYSTEM JUMP #340
029                     *
030 0317 3AFF02                 LDA     LEFTSP          INIT INSERT LEFT-SPACE
031 031A 3D                     DCR     A               NUMBER OF SPACES LEFT
032 031B 27                     DAA                     DECIMAL ADJUST
033 031C 6F                     MOV     L,A
034 031D E6F0                   ANI     :F0
035 031F 0F                     RRC
036 0320 0F                     RRC
037 0321 0F                     RRC
038 0322 0F                     RRC
039 0323 C630                   ADI     :30
040 0325 32F602                 STA     PRPOS$+1        TENTHS IN ASCII
041 0328 7D                     MOV     A,L
042 0329 E60F                   ANI     :0F
043 032B C630                   ADI     :30
044 032D 32F702                 STA     PRPOS$+2        SINGLES IN ASCII
045 0330 F1                     POP     PSW
046 0331 E1                     POP     H
047 0332 C9                     RET                     END OF INIT-ROUTINE
048                     *
049                     *
050                                                     ENTRY : CHR IN ACCU
051                             ORG     :340
052 0340 E5             START   PUSH    H               START OF DOUTC-PROCESSING
053 0341 D5                     PUSH    D
054 0342 C5                     PUSH    B
```

MODERNISED CHARS FOR SEIKOSHA GP100

```
055 0343 F5                PUSH  PSW
056 0344 32F302            STA   ACTCHR
057 0347 3AF202            LDA   LSTCHR
058 034A FE0D              CPI   :D
059 034C CAC803            JZ    CARRET
060 034F 3AF302      NOCRRT LDA   ACTCHR
061 0352 FE61              CPI   'a'          TEST CHR IN RANGE
062 0354 DA9D03            JC    UPCASE
063 0357 FE7B              CPI   :7B          'z'+1
064 0359 F29D03            JP    UPCASE
065 035C 3AF402            LDA   PRMODE
066 035F FE08              CPI   :8
067 0361 C4FA03            CNZ   PRMD=F
068                  *
069 0364 3AF302            LDA   ACTCHR
070 0367 210004            LXI   H,TABLE
071 036A D661              SUI   'a'          COMPUTE INDEX
072 036C 5F                MOV   E,A          IN NEW CHR TABLE
073 036D 010500            LXI   B,:0005
074 0370 CA7803      NEXTCH JZ    INXSET
075 0373 09                DAD   B
076 0374 1D                DCR   E
077 0375 C37003            JMP   NEXTCH       EXIT : HL=TABLE-CHAR-POINTER
078 0378 11ED02      INXSET LXI   D,CHRP$+1
079                  *
080 037B 7E          NEXTBT MOV   A,M
081 037C C680              ADI   :80
082 037E EB                XCHG               COPY TABLE-CHR$ TO CHRP$
083 037F 77                MOV   M,A
084 0380 EB                XCHG
085 0381 23                INX   H
086 0382 13                INX   D
087 0383 0D                DCR   C
088 0384 C27B03            JNZ   NEXTBT
089                  *
090 0387 21EC02            LXI   H,CHRP$
091 038A 0E06              MVI   C,6
092 038C CDBE03            CALL  PRPAP$
093                  *
094 038F 3AF302      RETURN LDA   ACTCHR
095 0392 CDAE03            CALL  PRSCRN
096 0395 32F202            STA   LSTCHR
097 0398 F1                POP   PSW
098 0399 C1                POP   B
099 039A D1                POP   D
100 039B E1                POP   H
101 039C C9                RET                END OF DOUTC-PROCESSING
102                  *
103                  *
104 039D 3AF402      UPCASE LDA   PRMODE
105 03A0 FE0F              CPI   :F
106 03A2 C4EC03            CNZ   PRMD=8
107 03A5 3AF302            LDA   ACTCHR
```

MODERNISED CHARS FOR SEIKOSHA GP100

```
108 03A8 CDB103      PRTUPC  CALL  PRPAPC
109 03AB C38F03              JMP   RETURN
110                  *
111 03AE EF          PRSCRN  RST   5          ENTRY : Accu = act char
112 03AF 03                  DATA  3
113 03B0 C9                  RET
114                  *
115 03B1 F5          PRPAPC  PUSH  PSW        ENTRY : Accu
116 03B2 3A02FE      WTACKP  LDA   PPIP~C
117 03B5 A7                  ANA   A
118 03B6 F2B203              JP    WTACKP
119 03B9 F1                  POP   PSW
120 03BA 3200FE              STA   PPIP~A
121 03BD C9                  RET              EXIT : Accu
122                  *
123 03BE 7E          PRPAP$  MOV   A,M        ENTRY : HL=strPntr,C=nrbytes
124 03BF CDB103              CALL  PRPAPC
125 03C2 0D                  DCR   C
126 03C3 C8                  RZ
127 03C4 23                  INX   H
128 03C5 C3BE03              JMP   PRPAP$
129                  *
130 03C8 3AF302      CARRET  LDA   ACTCHR
131 03CB FE0D                CPI   :D
132 03CD CAA803              JZ    PRTUPC
133 03D0 3AFF02              LDA   LEFTSP
134 03D3 A7                  ANA   A
135 03D4 CA4F03              JZ    NOCRRT     JUMP IF NO LEFTSPACE
136 03D7 FE01                CPI   :1
137 03D9 CAE403              JZ    ONESPC
138 03DC 21F502              LXI   H,PRPOS$
139 03DF 0E03                MVI   C,3
140 03E1 CDBE03              CALL  PRPAP$
141 03E4 3E20        ONESPC  MVI   A,:20
142 03E6 CDB103              CALL  PRPAPC
143 03E9 C34F03              JMP   NOCRRT
144                  *
145 03EC 3E80        PRMD=8  MVI   A,:80      HAMMER-SPC AT LOW-UP TRANS.
146 03EE CDB103              CALL  PRPAPC
147 03F1 3E0F                MVI   A,:F
148 03F3 32F402      SVPRMD  STA   PRMODE
149 03F6 CDB103              CALL  PRPAPC
150 03F9 C9                  RET
151                  *
152 03FA 3E08        PRMD=F  MVI   A,:8
153 03FC C3F303              JMP   SVPRMD
154                  *
155                                       TABLE WITH NEW CHARACTER-SET
156                                       normal width, normal hight
157                                       no descenders
158                  *
159                          ORG   :400
160 0400 205454      TABLE   DATA  :20,:54,:54,:54,:78 >  a
```

MODERNISED CHARS FOR SEIKOSHA GP100

```
161 0405 7F4444              DATA  :7F,:44,:44,:44,:38 >  b
162 040A 384444              DATA  :38,:44,:44,:44,:44 >  c
163 040F 384444              DATA  :38,:44,:44,:44,:7F >  d
164 0414 385454              DATA  :38,:54,:54,:54,:58 >  e
165 0419 00447E              DATA  :00,:44,:7E,:45,:01 >  f
166 041E 085454              DATA  :08,:54,:54,:54,:3C >  g
167 0423 7F0404              DATA  :7F,:04,:04,:04,:78 >  h
168 0428 00447D              DATA  :00,:44,:7D,:40,:00 >  i
169 042D 404044              DATA  :40,:40,:44,:3D,:00 >  j
170 0432 7F1028              DATA  :7F,:10,:28,:44,:44 >  k
171 0437 00417F              DATA  :00,:41,:7F,:40,:00 >  l
172 043C 7C047C              DATA  :7C,:04,:7C,:04,:78 >  m
173 0441 7C0404              DATA  :7C,:04,:04,:04,:78 >  n
174 0446 384444              DATA  :38,:44,:44,:44,:38 >  o
175 044B 7C1414              DATA  :7C,:14,:14,:14,:08 >  p
176 0450 081414              DATA  :08,:14,:14,:14,:7C >  q
177 0455 447C48              DATA  :44,:7C,:48,:04,:04 >  r
178 045A 485454              DATA  :48,:54,:54,:54,:24 >  s
179 045F 043F44              DATA  :04,:3F,:44,:40,:00 >  t
180 0464 3C4040              DATA  :3C,:40,:40,:40,:7C >  u
181 0469 1C2040              DATA  :1C,:20,:40,:20,:1C >  v
182 046E 3C4078              DATA  :3C,:40,:78,:40,:3C >  w
183 0473 442810              DATA  :44,:28,:10,:28,:44 >  x
184 0478 0C5050              DATA  :0C,:50,:50,:50,:7C >  y
185 047D 446454              DATA  :44,:64,:54,:4C,:44 >  z
186 0482 7F7F7F              DATA  :7F,:7F,:7F,:7F,:7F > res
187 0487                     END
```

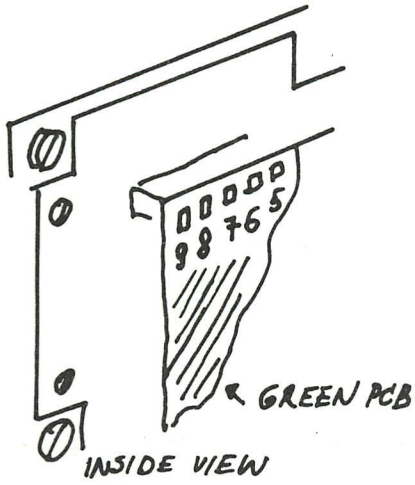
```
**************************
* S Y M B O L   T A B L E *
**************************
```

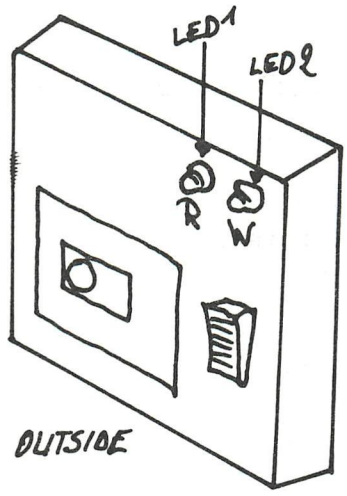
```
ACTCHR 02F3   CARRET 03C8   CHRP$  02EC   INIT   0300
INXSET 0378   LEFTSP 02FF   LSTCHR 02F2   NEXTBT 037B
NEXTCH 0370   NOCRRT 034F   ONESPC 03E4   PPICMD FE03
PPIP~A FE00   PPIP~B FE01   PPIP~C FE02   PRMD=8 03EC
PRMD=F 03FA   PRMODE 02F4   PRPAP$ 03BE   PRPAPC 03B1
PRPOS$ 02F5   PRSCRN 03AE   PRTUPC 03A8   RETURN 038F
START  0340   SVPRMD 03F3   TABLE  0400   UPCASE 039D
WTACKP 03B2
```

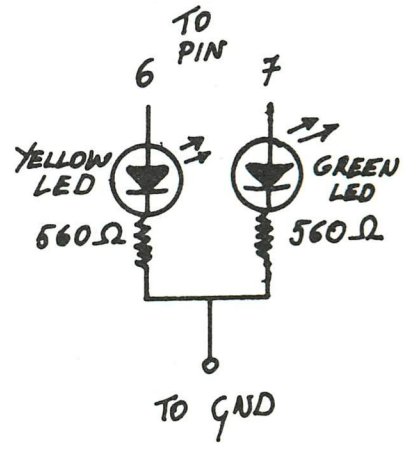
```
    ]#P.
02EC 00 00 00 00
02F0 00 00 0D 00 0F 10 00 00

02FF 0E
0300 E5 F5 3E A0 32 03 FE 3E FF 32 01 FE 21 DD 02 3C
0310 C3 23 36 40 23 36 03 3A FF 02 3D 27 6F E6 F0 0F
0320 0F 0F 0F C6 30 32 F6 02 7D E6 0F C6 30 32 F7 02
0330 F1 E1 C9

0340 E5 D5 C5 F5 32 F3 02 3A F2 02 FE 0D CA C8 03 3A
0350 F3 02 FE 61 DA 9D 03 FE 7B F2 9D 03 3A F4 02 FE
0360 08 C4 FA 03 3A F3 02 21 00 04 D6 61 5F 01 05 00
0370 CA 78 03 09 1D C3 70 03 11 ED 02 7E C6 80 EB 77
0380 EB 23 13 0D C2 7B 03 21 EC 02 0E 06 CD BE 03 3A
0390 F3 02 CD AE 03 32 F2 02 F1 C1 D1 E1 C9 3A F4 02
03A0 FE 0F C4 EC 03 3A F3 02 CD B1 03 C3 8F 03 EF 03
03B0 C9 F5 3A 02 FE A7 F2 B2 03 F1 32 00 FE C9 7E

03BF CD
03C0 B1 03 0D C8 23 C3 BE 03 3A F3 02 FE 0D CA A8 03
03D0 3A FF 02 A7 CA 4F 03 FE 01 CA E4 03 21 F5 02 0E
03E0 03 CD BE 03 3E 20 CD B1 03 C3 4F 03 3E 80 CD B1
03F0 03 3E 0F 32 F4 02 CD B1 03 C9 3E 08 C3 F3 03

0400 20 54 54 54 78 7F 44 44 44 38 38 44 44 44 44 38
0410 44 44 44 7F 38 54 54 54 58 00 44 7E 45 01 08 54
0420 54 54 3C 7F 04 04 04 78 00 44 7D 40 00 40 40 44
0430 3D 00 7F 10 28 44 44 00 41 7F 40 00 7C 04 7C 04
0440 78 7C 04 04 04 78 38 44 44 44 38 7C 14 14 14 08
0450 08 14 14 14 7C 44 7C 48 04 04 48 54 54 54 24 04
0460 3F 44 40 00 3C 40 40 40 7C 1C 20 40 20 1C 3C 40
0470 78 40 3C 44 28 10 28 44 0C 50 50 50 7C

047D 44 64 54
0480 4C 44 7F 7F 7F 7F 7F
```

DCR TAPE DIRECTION

Using the DCR after a CHECK command, it is sometimes difficult to guess if the tape is still running forward or if it is already being rewound... but there is a very simple way to abtain a tape direction monitor on the DCR, using only two small resistors, two LED's and a small knife...







Referring to the drawing, pins 6 & 7 of a connector inside the DCR bear +5V, when the engine is winding or rewinding the tape. An easy to solder ground connection can be found on the interface board. (the one with socketed IC's): the ground track is the largest one, near a bolt. Moreover, it is very easy to take out the black DCR-front panel, unscrewing four little screws (the smallest ones) from inside the DCR: so here the sharp knife becomes use-ful to make two little holes in the plastic panel, the tiny and cheap pressure LED mountings perfectly match the colour of the front panel... The two 560 Ω resistors tie the LED cathodes to ground; so three thin isolated wires are sufficient for the connections. Be careful not to interfere with the microcassette-detection microswitch inside the front panel.

Paolo Siccardo - Savona - Italy

HOOFDSTUK III : ARCHITECTUUR VAN 8-BITS MICROPROCESSOREN

We geven een korte samenvatting van de meest voorkomende microprocessoren en noemen dit de 'identiteitsfiche'.
Van de reeds eerder vernoemde industriële types geven we daarenboven nog een gedetailleerde beschrijving.

3.1. Identiteitsfiche van de microprocessor 8080 van INTEL

- TECHNOLOGIE : NMOS dynamisch

- KLOKFREQUENTIE : maximum 2MHz (8080) ; 3,13 MHz (A1 versie) ; 2,63 MHz (A2 versie).
  minimum 500 kHz

- KENMERKEN : voedingsspanningen : $\pm$ 5V en +12V
  verbruik : 1W
  klok : tweefasig verkregen met afzonderlijke schakeling (8224)
  databus : 8 bits
  adresbus : 16 bits
  controlesignalen : 5 bits afgeleid van de statussignalen die tijdens de eerste klokperiode beschikbaar zijn op de databus. Dit kan gebeuren door de 8228 (systeem en bus controller).
  interruptmogelijkheden : 1 niveau met 8 vaste sprongadressen.
  stapel (*stack*) : in het werkgeheugen (RAM) op een willekeurige plaats.
  instructieset : 78 basisinstructies
  logische niveaus : TTL compatibel
  uitvoeringstijden : van 2 tot 9µs (8080), van 1,3 tot 5,8µs (8080-A1) en van 1,5 tot 6,8µs (8060-A2)
  adresseringsmogelijkheden : impliciet, direct, indirect, immediate.

- INWENDIGE REGISTERS :

Accumulator	Flags
B	C
D	E
H	L
stapelwijzer (SP)	
programmateller (PC)	

- VEREENVOUDIGD TIJDSDIAGRAMMA (fig. 3.1)

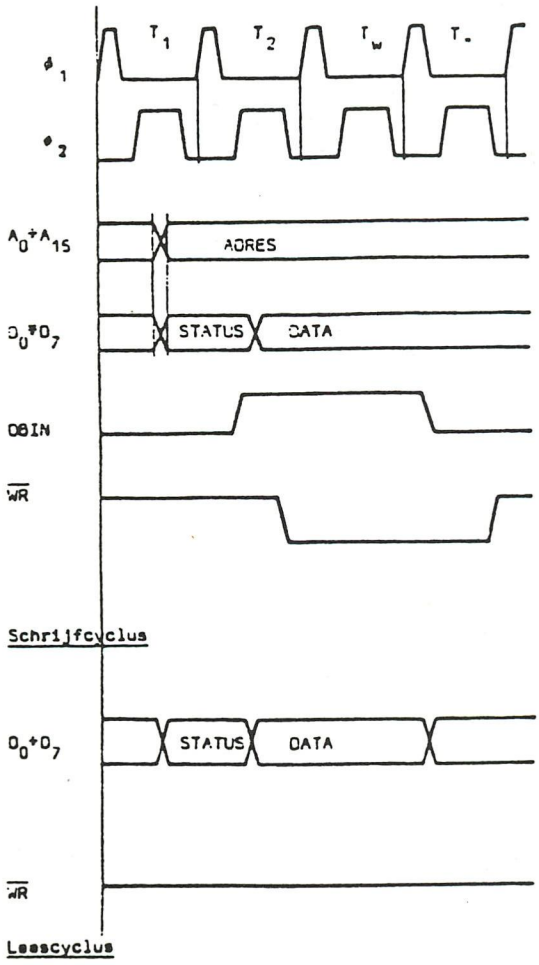


fig. 3.1

3.2. Architectuur van de microprocessor 8080 van INTEL

De microprocessor 8080 van INTEL is een dynamische 8-bits processor uitgevoerd in een N-kanaal MOS met één bus architectuur. Het inwendige schema wordt gegeven in fig. 3.2 en bestaat uit 5 delen :

- de 3-state buffers voor de uitwendige data- en adresbus
- de logische en rekenkundige eenheid met hulpregisters (ALU)
- het instructieregister (IR) met de instructiedecoder
- het registerveld met de adresseringslogica
- het controlegedeelte

3.2.1. De 3-state buffers voor data en adressen

Door deze buffers in de hoge impedantietoestand te plaatsen, kan de microprocessor zich isoleren zowel van de uitwendige gegevensbus als van de adresbus.

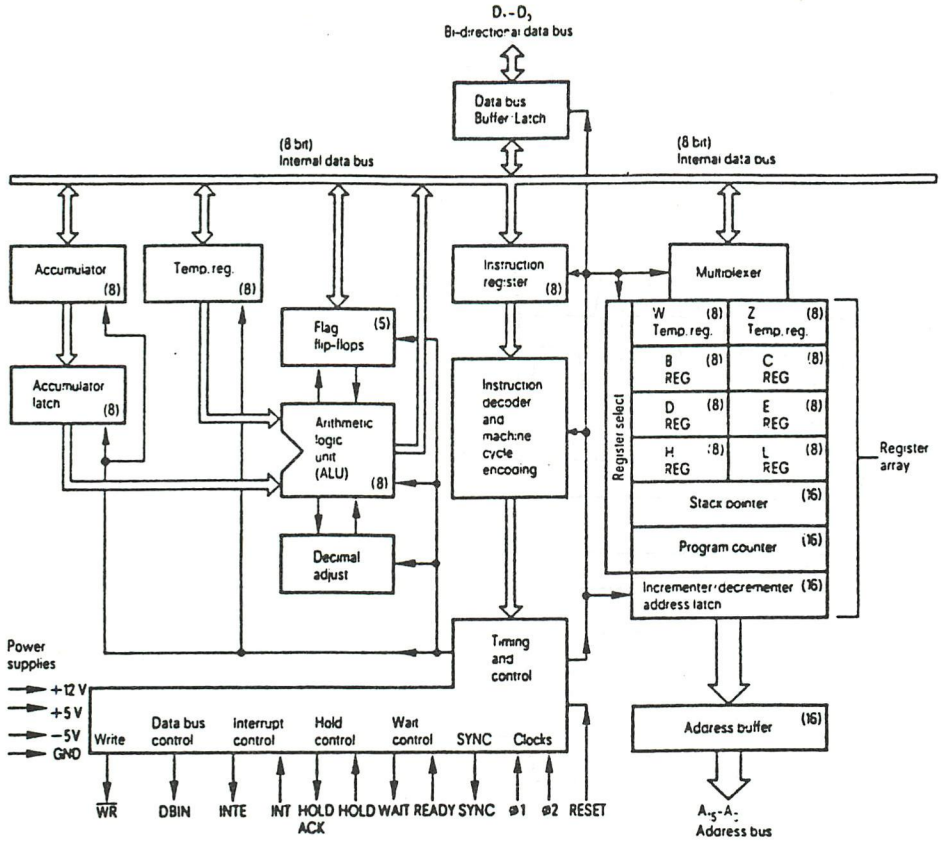


fig. 3.2

De bewerkingen die dan gebeuren in het inwendige van de microprocessor zijn onzichtbaar voor de omgeving. Het transfert van informatie in de microprocessor gebeurt via de interne databus waarop alle inwendige delen aangesloten zijn.

3.2.2. De logische en rekenkundige eenheid (ALU)

Dit gedeelte bevat :

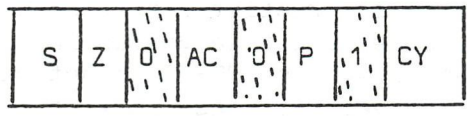
- een 8-bits accumulator (A)
- een 8-bits accumulator buffer/geheugen (*latch*)
- een 8-bits voorlopig register (TMP)
- de logische en rekenkundige eenheid zelf
- een 5 bits toestandsregister (vlag register)
- een register voor decimale correctie van de accumulatorinhoud bij het verwerken van BCD getallen.

Alle bewerkingen zoals rekenkundige bewerkingen en de logische operaties EN, OF, EXOF die door de ALU uitgevoerd worden, gebeuren met de inhoud van de accumulator en het TMP register.
Het resultaat van de bewerking wordt terug ingeschreven in de accumulator.
Om rondlopen van de informatie te vermijden is de accumulator voorzien van een latch die bij het begin van de operatie de inhoud van de accumulator ontvangt en onmiddellijk daarna nog slechts toegankelijk is voor uitlezing.

Het register voor decimale correctie kan de 8 bits binaire accumulator inhoud omzetten in 2 tetraden in de BCD code.
Het vlagregister, ook wel register van de toestandenbits genoemd, memoriseert bepaalde merkwaardige eigenschappen van het resultaat van een logische of rekenkundige operatie, uitgevoerd door de ALU. Deze eigenschappen worden gesymboliseerd door een bit in het toestandenregister op 1 te zetten of terug te zetten op 0.
De toestanden die in het register opgenomen worden zijn :

- het al (1) of niet (0) (*zero*=z) zijn van de accumulatorinhoud
- het negatief (1) of positief (0) zijn van het teken (*sign*=S) van het resultaat.
- het bestaan (1) of niet bestaan (0) van een overdrachtbit (*carry*=C of CY)
- het bestaan (1) of niet bestaan (0) van een hulpoverdracht (*auxiliary carry*=AC) tussen de eerste en tweede tetrade.
- een even (1) of oneven (0) pariteit (*parity*=P) van het resultaat.

Deze 5 bits hebben een vaste plaats in het 8-bits toestandenregister, wat meteen betekent dat 3 vaste bits in dit register ingeschreven zijn.



3.2.3. Het instructieregister met de decoder

Het instructieregister (IR) ontvangt de uit te voeren instructie uit het programmageheugen (ROM) via de databusbuffer.
Deze instructie wordt dan in de instructiedecoder gedecodeerd om uit te maken wat dient uitgevoerd te worden. Deze decodering gebeurt in een inwendige ROM. Het resultaat van deze decodering zet uiteindelijk een signaal op de controlebus. Dit signaal maakt de gewenste onderlinge verbinding van de verschillende schakelingen van het systeem.
De controlesignalen kunnen zijn :

MEMR *Memory read* wanneer de instructie gepaard gaat met een lezing uit het geheugen hetzij ROM, hetzij RAM.

MEMW *Memory write* wanneer de instructie een inschrijving in het geheugen RAM veroorzaakt.

I/OR *Input/output read* wanneer een van de ingangskanalen gelezen wordt.

I/OW *Input/output write* wanneer in een van de uitgangskanalen geschreven wordt.

INTA *Interrupt acknowledgement* (kennisgeving van ontvangst van een onderbrekingsaanvraag) wanneer het een onderbreking betreft van een interruptprogramma dat in uitvoering is (zie verder).

De controlesignalen zijn niet zonder meer op de klemmen van de microprocessor aanwezig maar als een 8 bits statuswoord op de databus, afwisselend met de gegevens. Dit veronderstelt dat de inhoud van de databus gemultiplexeerd is tussen de gegevens (data) en statussignalen waaruit de controlesignalen opgebouwd worden. Met behulp van een speciale bouwsteen (bv. 8228) wordt deze informatie op de databus omgezet in de 5 controlesignalen (zie verder).

3.2.4. Het registerveld met de adresseringslogica

Het registerveld (*scratch pad*, kladblok) is in feite een inwendig geheugen met beperkte afmetingen waarin de CPU rechtstreeks kan schrijven of lezen door aangepaste instructies. Het voordeel van dit inwendig geheugen is de snelheid van afwikkeling van de operaties. Alles gebeurt binnen de CPU en er moet niet steeds gelezen of geschreven worden in het uitwendig geheugen. Het registerveld bestaat uit 8 registers van 8 bits, die kunnen samengevoegd worden tot drie 16 bits registers. Het zijn de registers B, C, D, E, H en L die eveneens kunnen aangesproken worden als registerparen B, D en H.
Twee bijkomende registers W en Z worden alleen gebruikt voor de inwendige verwerking van sommige instructies door de microprocessor; toegang tot deze registers via een programma is onmogelijk.
We merken op dat de registers H en L een dubbele functie hebben. Enerzijds kunnen ze dienst doen als inwendig geheugen, zoals hoger aangegeven. Anderzijds wordt het HL registerpaar gebruikt als adresseringspointer en bevatten H en L respectievelijk de hoogste byte (*high*) en de laagste byte (*low*) van een geheugenadres bij indirecte adressering.
Naast deze registers van algemeen nut zijn er nog 2 registerparen die een vaste functie hebben in verband met adresseringslogica : het PC register (*program counter*) of de instructieteller. Dit registerpaar bevat steeds het adres van de volgende uit te voeren instructie. Het wordt door een RESET op 0 gezet, d.w.z. na RESET is het adres 0000H ingeschreven in het PC register. De eerstvolgende instructie die zal uitgevoerd worden bevindt zich dus op adres 0000H van het programmageheugen. Het registerpaar SP (*stack pointer*) of stapelwijzer bevat het adres van de laatste geheugencel die gebruikt is in het stapelgeheugen (*stack*).
Het beginpunt van dit stapelgeheugen wordt vastgelegd in het programma en begint gewoonlijk bij het hoogste physische RAM adres. Inschrijving of lezing in de *stack* gebeurt steeds per 2 bytes. Bij inschrijving in het stapelgeheugen telt de stapelwijzer af. Bij lezing uit het stapelgeheugen daarentegen telt hij op.

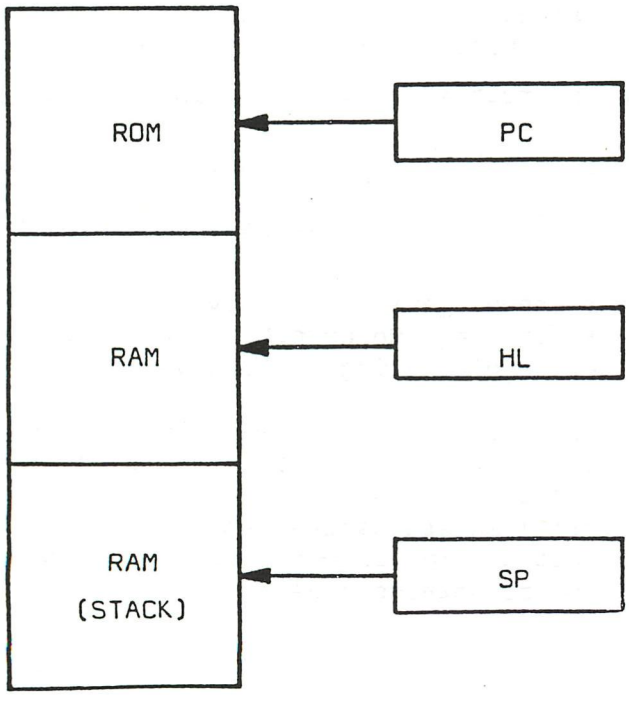


fig. 3.3

De *stack* wordt door de microprocessor gebruikt, o.a. om met terugkeeradres in te schrijven wanneer er gewerkt wordt met subroutines of om de inhoud van die registers veilig te stellen die nodig zijn om een hoofdprogramma te kunnen verderzetten na een onderbreking (*interrupt*). Resumerend kunnen we zeggen dat in het registerveld 3 adresseringswijzers aanwezig zijn (fig. 3.3):

- de programmateller (*program counter*) wijst het adres aan van een instructie in het ROM geheugen.
- het HL registerpaar wordt gebruikt als adreswijzer (*pointer*) in het RAM geheugen.
- de stapelwijzer (*stack pointer*) is de adreswijzer van het speciaal gedeelte in het RAM geheugen dat de stapel of *stack* wordt genoemd.

3.2.5. Het controle gedeelte

Het controlegedeelte is verantwoordelijk voor de besturing en de controle van de centrale verwerkingseenheid.
De ingangscontrolesignalen zijn :

RESET — Het signaal waarmee de microprocessor op nul gezet wordt zodat de uitvoering van een programma kan beginnen. Het RESET signaal zet de instructieteller (PC) op nul, terwijl de andere registers niet beïnvloed worden. RESET is aktief met niveau hoog en moet een duur hebben van minstens 3 klokperiodes.

$Ø_1$ en $Ø_2$ — Deze twee signalen vormen een tweefasenklok voor de tijdsbesturing van de CPU met een amplitude van 8V, opgewekt door een afzonderlijke bouwsteen 8224. Het spanningsverloop van deze kloksignalen wordt gegeven in fig. 3.4.

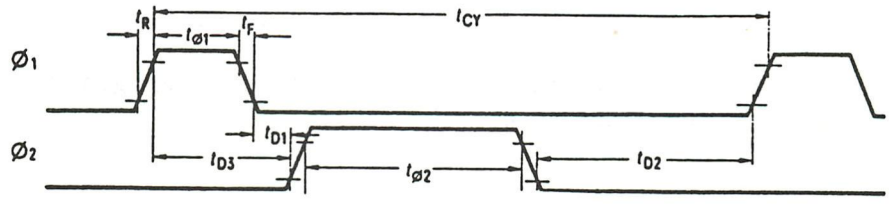


fig. 3.4

INT — Interrupt is het signaal dat aan de CPU meedeelt wanneer een van de periferie apparaten het hoofdprogramma wenst te onderbreken. Het is dus een aanvraag tot onderbreking.

READY — Is het signaal dat aan de CPU meedeelt dat het (traag) geheugen of het (trage) periferie apparaat nog niet klaar is met de uitvoering van de gevraagde cyclus. De CPU wacht en kan niet verder werken voor het READY signaal op hoog komt. READY is aktief met niveau laag.

HOLD — Is eveneens een vraag tot stoppen van de kativiteiten van de CPU met dit verschil dat nu zowel adres als gegevensbuffers in de hoge impedantietoestand worden gezet (3-state) en de CPU zich isoleert van dit bussysteem zodat een andere processor bv. direkte toegang tot het geheugen DMA) de besturing van de bussen van het systeem kan overnemen. HOLD is aktief met een hoog niveau.

De signalen die vertrekken van het controlegedeelte zijn :

$\overline{WR}$ Het signaal dat aangeeft op de operatie die volgt een schrijf- of een leesopdracht is.
Niveau laag geeft schrijven.
Niveau hoog geeft lezen.

DBIN (*data bus in*) Door dit signaal geeft de processor te kennen dat gegevens ingelezen kunnen worden via de bidrirectionele databus (niveau hoog) of uitgelezen worden (niveau laag). DBIN is eveneens laag bij het uitsturen op de databus van de status-signalen die dan verder gecodeerd dienen te worden om de controlebussignalen te vormen van de 5 bits controlebus.

HLDA (*hold acknowledgement*) is het signaal dat de ontvangst van een HOLD aanvraag bevestigt met een hoog niveau.

SYNC Dit synchronisatiesignaal geeft het begin aan van elke nieuwe machinecyclus en synchroniseert de afwikkeling van de verschillende onderdelen van een instructie, in functie van de tijd. Dit signaal wordt hoog na de opgaande flank van $Ø_2$ tijdens de eerste klokperiode tot de volgende opgaande flank van $Ø_2$. (zie fig. 3.4).

WAIT Met WAIT geeft de CPU te kennen dat hij wacht op het READY signaal dat kan uitgegeven worden door een traag geheugen of een traag periferie apparaat.
WAIT is aktief voor niveau hoog.

Figuur 3.5 geeft de bezetting van de pennen van het microprocessor-IC. Het is een 40 pennen IC waarop al de hoger vermelde signalen terug te vinden zijn, nl. :

- 16 adressignalen
- 8 gegevenssignalen
- 12 signalen van het besturings- en controlesysteem

Tevens zijn er 4 pennen beschikbaar voor de 3 voedingsspanningen waarvan de 8080 gebruik maakt en de massa.

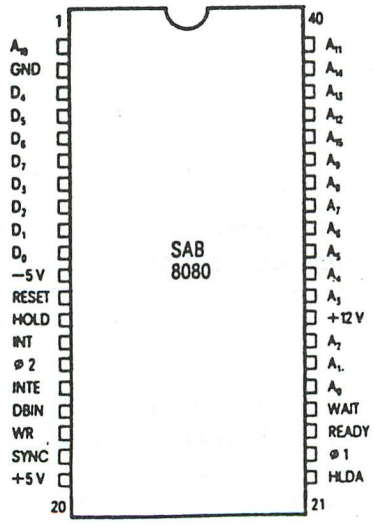


fig. 3.5

RGB-VIDEO

****
* ADAPTEUR DAI/VIDEO MONOCHROME *
****

D'apres Radio-Plans de juillet 1983 No 428 p 20

Alain Mariatte

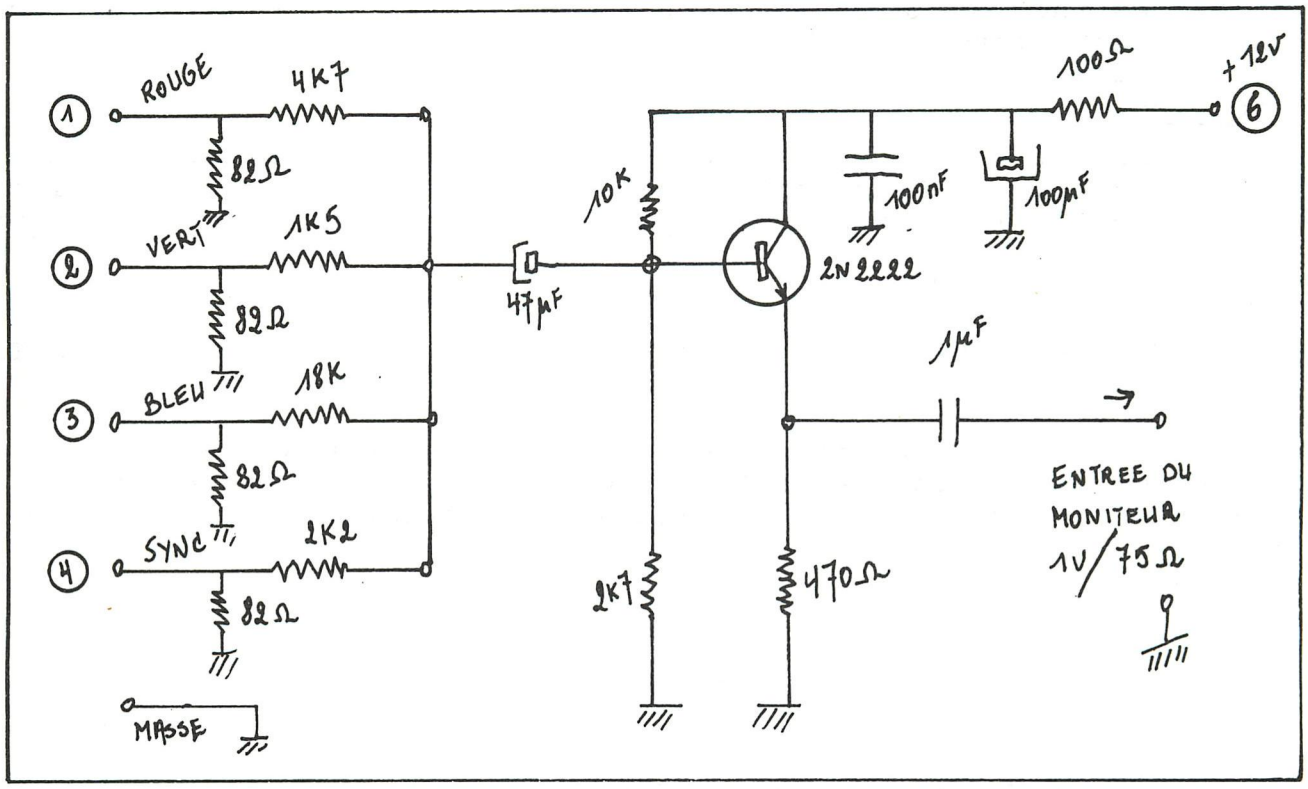
Pour utiliser un ordinateur a sorties Peritel (RGB) comme le DAI avec un moniteur monochrome necessitant un signal composite de 1V crete sur 75 Ohms (moniteur qui equipe la majorite des ordinateurs de table), il faut faire la somme des signaux que le DAI sort sur la prise "VIDEO".

Cette somme doit respecter la relation:

$$Y = 0.3\ R + 0.59\ V + 0.11\ B$$

Le circuit suivant, tres simple et tres largement inspire de l'article cite en reference, assure ce travail et permet de se passer momentanement de l'encombrant tele couleur. On y perd bien-entendu la couleur et le son, mais ce n'est pas grave pour bien des applications (et on gagne en facilite de transport: on fait actuellement des moniteurs vraiment legers).

Les numeros des broches sont ceux de la prise DIN sortie PERITEL (RGB) du DAI. Bien entendu, la sortie son '5' n'est pas utilise ici.



6. DE TICC

Vraag 5: TICC is een afkorting van .....

De tweede I/O-module waarover de DCE beschikt is de TICC (Timer Interrupt and Communications Controller).
De TICC wordt gevormd door de TMS 5501 van Texas Instruments.
De TICC bestaat uit de volgende delen.
a. Een 8-bits input-poort.
b. Een 8-bits output-poort.
c. Een serial input- en output.
d. Een aantal timers.
e. Een interrupt controller.

In fig. 6 is een gedetailleerd blokschema van de TICC weergegeven. Tussen haakjes is aangegeven welk adres elk register en elke timer heeft.
(Om de tekening overzichtelijk te houden zijn de selectie-lijnen van de adres decoder en de besturingsbus niet getekend.)

7. PARALLEL INPUT/OUTPUT

De TICC beschikt over een 8 bits input-poort en een 8-bits output-poort via welke de data van of naar de CPU wordt getransporteerd.
De modes van deze poorten kunnen, in tegenstelling tot de GIC-poorten, niet veranderd worden.

Vraag 6: Met de instructie STA 9807H brengen we de
..... naar .....

Willen we b.v. de inhoud van de accumulator naar de output-poort met adres 9807_{16} brengen, dan gebruiken we de instructie STA 9807H.
Met de instructie LDA 9801H halen we de informatie op de input-poort naar de accumulator.

Opmerking:

Een opgaande flank op b_7 van de input-poort genereert tevens een interrupt request. Deze komt binnen op b_7 van het interrupt register (zie paragraaf 10). Deze interrupt noemen we de auxiliary interrupt.

8. SERIAL INPUT/OUTPUT

Voor de communicatie met b.v. een TTY of een display beschikt de TICC over een serial input en een serial output.

Willen we een 8-bitswoord in serie (dus achter elkaar) overzenden, dan brengen we dit woord naar het transmitter register (to transmit = zenden), die ervoor zorgt dat de bits achter elkaar de lijn worden opgestuurd.

Antw.5: Timer, Interrupt and Communications Controller.
Antw.6: inhoud van de accumulator; output-poort.

Het transmitter register zorgt zelf voor het opwekken van de start- en stop-bits (fig. 7).

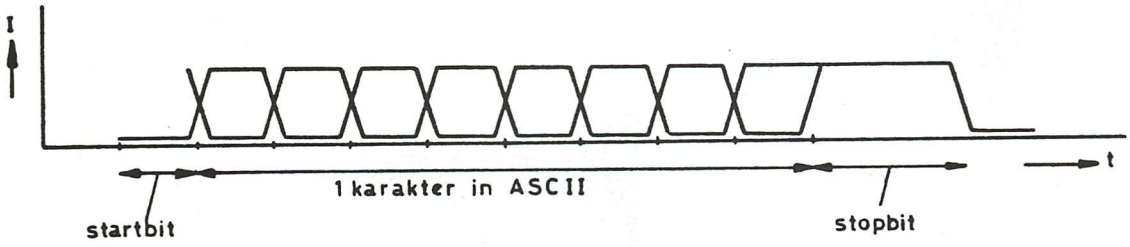


fig.7

Vraag 7: De baud rate is het aantal ..... per .....

De baud rate (= aantal bits/sec) waarmee dit overzenden gebeurt, moeten we van te voren aanpassen aan de baud-rate van het randapparaat.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
stop-bit select	9600 Baud	4800 Baud	2400 Baud	1200 Baud	300 Baud	150 Baud	110 Baud

fig.8

In fig. 8 is het baud-rate register weergegeven. Door één bepaalde bit 1 te maken, kunnen we de baud-rate waarmee de TICC data ontvangt en overzendt instellen.
Door b_7 1 of 0 te maken kunnen we bepalen of het transmitter-register bij het overzenden van een karakter 1 resp. 2 stopbits genereert.

Vraag 8 : Willen we de baud rate instellen op 110 Baud en het transmitter register 2 stopbits laten genereren, dan moeten we naar adres $.....{}_{16}$ de waarde $.....{}_{16}$ sturen.

Voor een baud-rate van b.v. 110 moeten we b_0 van het transmitter register 1 maken en de bits b_6 t/m b_1 0.
Vereist het randapparaat b.v. 2 stopbits, dan dient b_7 0 te zijn.
In dit geval moeten we het baud-rate register (adres 9805) vullen met $00000001_2 = 01_{16}$.

Wanneer het transmitter register alle 8 bits van een karakter en de start- en stopbits de lijn op heeft gezonden, geeft het een interrupt request om aan te geven, dat de CPU nieuwe data mag zenden.

Op dezelfde manier geeft het receiver register (to receive = ontvangen) een interrupt request wanneer een 8 bits woord en de start- en stop-bits zijn ontvangen. De CPU kan de data dan overnemen.

Vraag 9 : De CPU brengt data van het receiver register naar de accumulator m.b.v. de instructie .....

De CPU doet dit m.b.v. een instructie die data overbrengt van geheugenwoord met adres 9800_{16} (het receiver register) naar de accumulator. Dit is b.v. de instructie LDA 9800H.

Antw.7: bits per seconde. Antw.8: 9805_{16}; 01_{16}. Antw.9: LDA 9800. 20-01

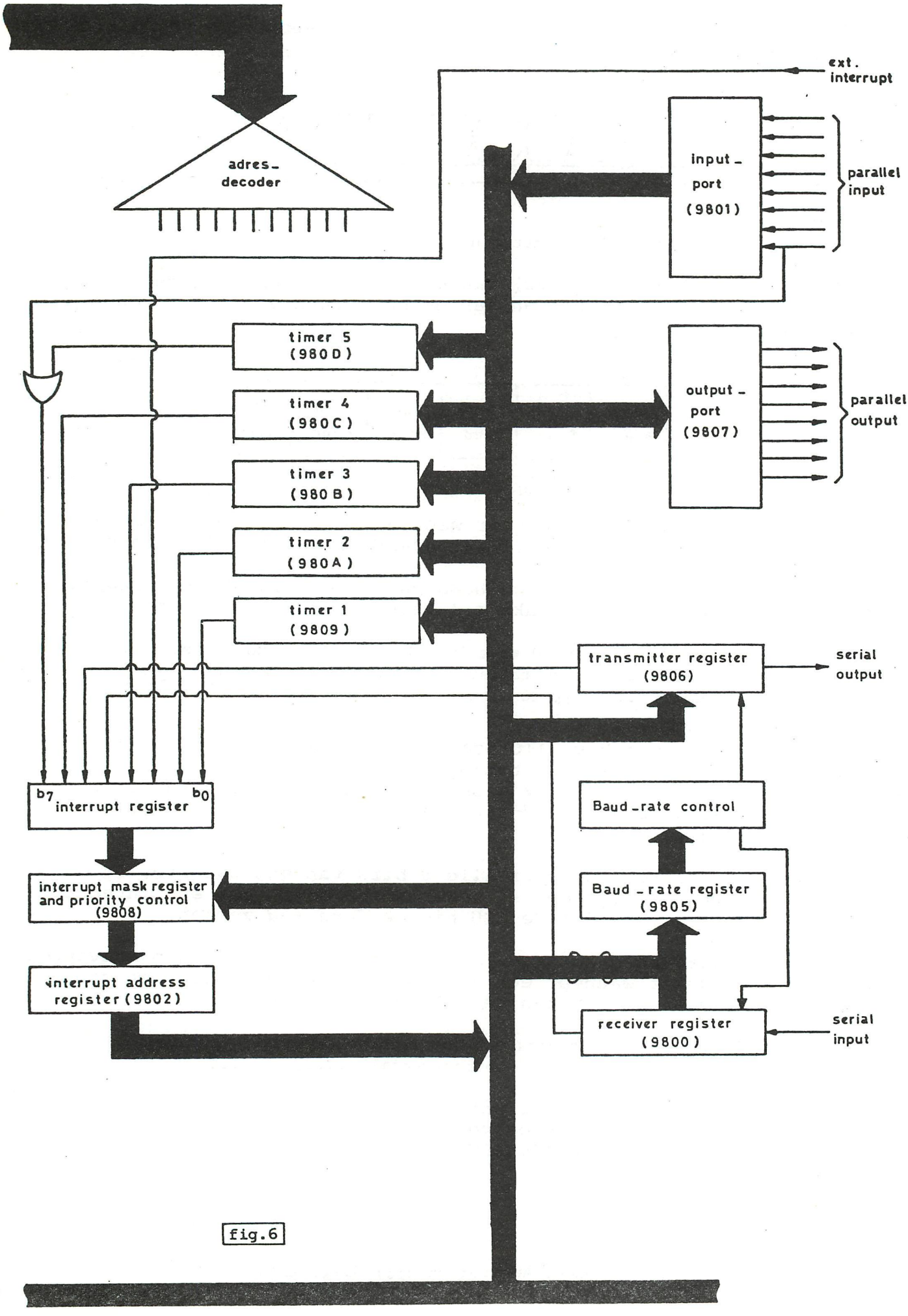

ext. interrupt
adres_ decoder
input_ port (9801)
parallel input
timer 5 (980D)
timer 4 (980C)
timer 3 (980B)
timer 2 (980A)
timer 1 (9809)
output_ port (9807)
parallel output
transmitter register (9806)
serial output
b7 interrupt register b0
Baud_rate control
interrupt mask register and priority control (9808)
Baud_rate register (9805)
interrupt address register (9802)
receiver register (9800)
serial input

fig.6

Opmerking: Het voordeel van deze "interrupt-methode" is, dat de CPU verder kan gaan met de uitvoering van het hoofdprogramma, terwijl het transmitter register of het receiver register bezig is de data de lijn op te zenden of te ontvangen.

SAMENVATTING 2

4. TICC betekent Timer, Interrupt and Communications Controller. De TICC beschikt over
   a. een 8-bits input-poort en een 8-bits output-poort,
   b. een serial in- en output,
   c. 5 timers,
   d. een interrupt controller.
5. Wanneer we gebruik maken van de serial in- en output van de TICC, moeten we van te voren de Baud-rate instellen, door één van de bits van het Baud-rate register 1 te maken. Tevens moeten we aangeven of een karakter gevolgd wordt door 1 of door 2 stopbits.
6. Het in serie overzenden en ontvangen van een karakter geschiedt op basis van interrupt I/O. Wanneer een karakter is overgezonden of ontvangen, zendt de TICC een interrupt request naar de CPU, zodat de CPU nieuwe data naar de TICC kan zenden of data vanuit de TICC kan ophalen.

cont. from p.386

ressen elk gescheiden door een space (cfr het Display bevel uit de utility). Het eerste adres is het beginadres van de geheugenzone die in EPROM dient gebracht te worden (begin sourcebuffer), het tweede adres is het eindadres van deze geheugenzone (eindadres sourcebuffer), het derde adres ten slotte is het beginadres waar de data dienen ondergebracht te worden in de EPROM.

Hiervoor dient altijd het beginadres van een blok van 256 byte (pagina) genomen te worden. Gedurende de programmatie brandt de rode LED. Na het voltooien van de programmatie, test het programma of alles foutloos is gebeurd en meldt zich met

PROGRAMMING DONE

gevolgd door de melding NO ERRORS of

ERROR AT ADDRESS xxxx .

xxxx staat voor het hexadecimale adres waar de eerste fout gelezen is.

Het is eveneens mogelijk de inhoud van een EPROM te lezen. Het bevel is (R) 'READ' en de geheugenruimte beginnende bij het hexadecimale adres A300 wordt dan gevuld met de inhoud van de EPROM.

Wenst men terug te keren naar UTILITY dan kan dit gebeuren door het bevel (U).

2 EPROM's mogen slechts in de sokkel gestoken of uit de sokkel gehaald worden, als de groene LED brandt.

```
10    REM ================================================
11    REM titel           KERSTNACHT
12    REM datum           82-12-20
13    REM (c)             Herman MOEYS - 1982
14    REM ================================================
20    REM INITIALIZATIE ----------------------------------
21    POKE #75,32:PRINT CHR$(12):COLORT 0 0 0 0:COLORG 0 0 0 0:MODE 5A
:POKE #744B,#6A:CLEAR 10000:RESTORE
22    DIM KL(5.0),XL(24.0),YL(24.0),XS(3.0,99.0),KS(15.0),T$(1.0)
23    FOR I=0 TO 5:READ T:KL(I)=T:NEXT:FOR I=0 TO 15:READ T:KS(I)=T:NE
XT
24    DATA 1,2,3,10,12,14,8,1,2,3,4,15,6,7,0,9,10,11,12,13,14,15
25    MS=30:XM=XMAX+1:YM=YMAX+1:XC=118+INT(RND(13.0))*8.0
26    T$(0.0)="          PRETTIGE  KERSTDAGEN          "
27    T$(1.0)="           VOORSPOEDIG   1984           "
30    REM KERSTBOOM HALEN --------------------------------
31    CURSOR 0,1:PRINT "     een kerstboompje halen  ....":COLORT 0 10
 0 0
32    YC=171:BB!=20.0:EB!=2.0:TB=24:GOSUB 100
33    YC=147:BB!=35.0:EB!=10.0:TB=26:GOSUB 100
34    YC=117:BB!=55.0:EB!=17.0:TB=34:GOSUB 100
35    YC=79:BB!=80.0:EB!=27.0:TB=44:GOSUB 100
36    YC=35:BB!=110.0:EB!=40.0:TB=52:GOSUB 100
37    FOR Y=45 TO 0 STEP -1:DRAW XC-10,Y XC+10,Y 6:DOT XC-11,Y 15:DOT
XC+11,Y 15:NEXT
40    REM LAMPJES HANGEN ---------------------------------
41    T=0
42    COLORT 0 0 0 0:CURSOR 0,1:PRINT "     enkele lampjes ophangen  .
..    ":WAIT TIME 30:COLORT 0 10 0 0
43    X=RND(XM):Y=RND(YM):IF SCRN(X,Y)<>5 GOTO 43
44    XL(T)=X:YL(T)=Y:L2=T MOD 6:GOSUB 200
45    T=T+1
46    IF T<25 GOTO 43
50    GOTO 1000
100   REM [S] KERSTBOOM TEKENEN --------------------------
101   TAU!=TB/LOG(BB!/EB!)
102   FOR T=TB TO 0 STEP -1
103   X=BB!*EXP(-T/TAU!)-2.0
104   DRAW XC-X,YC+T XC+X,YC+T 5
105   DOT XC-X,YC+T 15
106   DOT XC+X,YC+T 15
107   NEXT
108   RETURN
200   REM [S] LAMPJES KLEUREN ----------------------------
201   K=KL(L2)
202   FILL X-3,Y-3 X+3,Y+3 K
203   FILL X-1,Y-4 X+1,Y+4 K
204   K=INT(K/8.0)*15.0
205   DRAW X-4,Y X+4,Y K
206   DRAW X-4,Y-1 X+4,Y-1 15-K
207   DRAW X-4,Y+1 X+4,Y+1 15-K
208   RETURN
1000  REM KERSTNACHT -------------------------------------
1100  REM KNIPPERENDE STER -------------------------------
1110  FOR L1=0 TO 5
1120  COLORT 0 0 0 0:YC=200:K=KL(L1)
1130  FOR I=0 TO K STEP K
1131  DRAW XC-10,YC XC+10,YC I
1132  DRAW XC-7,YC-7 XC+7,YC+7 I
1133  DRAW XC-7,YC+7 XC+7,YC-7 I
1134  DRAW XC,YC+10 XC,YC-10 I
1135  DRAW XC-7,YC-3 XC+7,YC+3 I
1136  DRAW XC-3,YC-7 XC+3,YC+7 I
1137  DRAW XC-3,YC+7 XC+3,YC-7 I
```

```
1138  DRAW XC-7,YC+3 XC+7,YC-3 I
1139  NEXT
1140  CURSOR 0,1:PRINT T$(L1 MOD 2.0):COLORT 0 10 0 0
1200  REM KNIPPERENDE LAMPJES ------------------------
1210  FOR L2=0 TO 5
1220  T=RND(25.0):X=XL(T):Y=YL(T):GOSUB 200
1300  REM SNEEUWVLOKJES ------------------------------
1310  FOR L3=0 TO 5
1320  X=RND(XM):Y=RND(YM)
1321  K=KS(SCRN(X,Y))
1322  HS=XS(X/100.0,X MOD 100.0)
1323  IF HS<MS-30.0 THEN HS=MS-30
1330  DOT X,Y K
1331  DOT X,HS 15
1340  XS(X/100.0,X MOD 100.0)=HS+1
1350  IF HS<MS GOTO 2000
1360  FOR I=0 TO XM-1
1361  IF SCRN(I,HS)=8 THEN DOT I,HS 15
1362  NEXT
1370  DRAW 0,HS-30 XM-1,HS-30 15
1380  MS=MS+1
1390  IF MS=212 GOTO 10
2000  NEXT:NEXT:NEXT:GOTO 1000
```

DELETE

HOE WIS JE EEN GEDEELTE VAN EEN PROGRAMMA UIT?

Het komt nogal vaak voor dat men een gedeelte van een programma dat overbodig geworden is wil verwijderen, of dat men een onderdeel uit een groot programma wil afzonderen. Sommige BASIC-dialecten gebruiken daarvoor een speciaal bevel (bvb. DELETE in TRS 80-basic), maar in DAI-basic is dit niet voorzien.

Toch zijn er een aantal mogelijkheden om iets uit te wissen:

1. Tik gewoon de nummers van de te wissen regels opnieuw in. Deze methode is omslachtig als je grote gedeelten van een programma wil verwijderen.

2.-Zet het programmadeel dat je wil bewaren in de EDIT-buffer
   -Wis de tekstbuffer
   -Copieer de EDIT-buffer in de tekstbuffer.

We willen bijvoorbeeld een subroutine bewaren die op regel 1ØØØ tot 2ØØØ staat, en de rest van het programma vernietigen. Dit kan op volgende manier:

```
*CLEAR 3ØØØ
*EDIT 1ØØØ-2ØØØ
 ⟨BREAK⟩ ⟨BREAK⟩
*NEW
*POKE 3Ø9,2
```

Deze methode is vooral geschikt om programmaonderdelen uit een groot programma af te zonderen (bvb. een subroutine die men in een ander programma wil gebruiken). Ze kan enkel gebruikt worden als het te bewaren stuk een aaneensluitend geheel vormt.

3. De "Monte Carlo-methode": als je zeer snel na het indrukken van de RETURN- toets achter 'EDIT1ØØØ-2ØØØ" 2X⟨BREAK⟩ duwt is het mogelijk dat er een gedeelte van het programma tussen 1ØØØ en 2ØØØ verdwenen is. De rest kan je dan met methode 1 verwijderen. Deze methode vindt enkel aanhangers bij personen die het genoegen willen smaken sneller te zijn dan de computer en hoort dus eerder thuis in het hoofdstuk "Games&Strategy" dan in het hoofdstuk "Serieus programmeren" Af te raden!

4. Om bvb. regels 7Ø - 12Ø te verwijderen doe je (in command mode): *EDIT 7Ø-12Ø :EDIT n (n= willekeurig nummer)

Op het scherm verschijnen dan regels 7Ø-12Ø in EDIT-mode. Doe dan ⟨BREAK⟩ ⟨SPATIE⟩ . Regel n verschijnt op het scherm; doe ⟨BREAK⟩ ⟨BREAK⟩
Deze methode blijkt feilloos te werken, al kan ik niet verklaren hoe.

Ik heb ook getracht deze "truuk" in te bouwen in een basic-programma, maar dan blijkt het niet in alle gevallen te werken. (met behulp van het programma van de heer Dufour uit Dainamic 16, p.200)

Als je grote gedeelten uit een programma verwijderd hebt is het aan te raden de symbol table weer op te poetsen. Daar staan immers nog steeds de variabelen in die uit het programma verdwenen zijn.

```
*CLEAR xxxxx      (voldoende om het ganse programma te bevatten)
*EDIT
⟨BREAK⟩ ⟨BREAK⟩
*NEW
*POKE 3Ø9,2
```

J o s V a n d e b e r g h

---

p.s. don't forget your subscription

1984

JANUARY

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

FEBRUARY

SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

MARCH

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

APRIL

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY

SUN	MON	TUE	WED	THU	FRI	SAT
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

JUNE

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JULY

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUGUST

SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEPTEMBER

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

OCTOBER

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

NOVEMBER

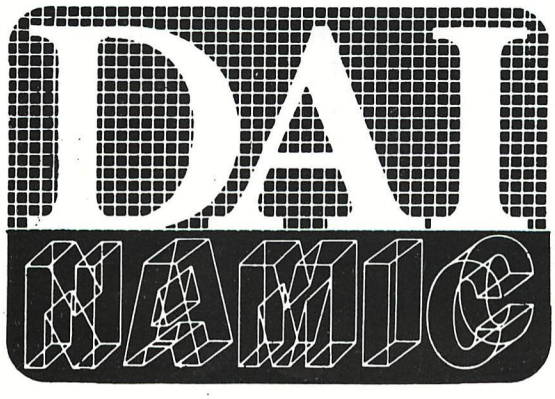
SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

DECEMBER

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Morrison

4th international DAInamic meeting on saturday 21 april in Tongelsbos, Westerlo.



DAI graphics

Jeroen Overvoorden

SEND YOUR DRAWINGS TO DAINAMIC EDITOR