DAI NAMIC

22

tweemaandelijks tijdschrift mei - juni 1984

personal computer users club

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

International

COLOFON

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

DAInamic redactie :

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

DAI NAMIC

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

22

mei - juni 1984

Beste leden,

Met de vacantie voor de boeg sturen we U nummer 22. DAInamic, met een ledenbestand van meer dan 1000 leden is uitgegroeid tot een omvangrijke organisatie. Vele medewerkers dragen hun steentje bij om alles zo vlot mogelijk te laten verlopen. We willen toch nog eens melden dat al deze activiteiten gebeuren op vrijwilligers-basis en dat er ook vele uurtjes vrije tijd verloren gaan aan minder prettige bezigheden zoals administratie, boekhouding, BTW enz. Wij doen ons best om uw bestellingen, abonnementen en vragen snel en nauwkeurig af te werken. U kan ons hierbij helpen door uw problemen door te spelen aan de juiste personen : Bruno Van Rompaey verzorgt de ledenadministratie, hier kan U ook terecht voor oude nummers van DAInamic. Frank Druijff ontvangt graag uw spelprogramma's of programma's ivm programmeer-technieken of problemen. Hij wordt hierin bijgestaan door Nico Looije en nog een paar andere medewerkers. De rest van uw problemen en creaties mag U loslaten op de redactie in Herselt. U bespaart ons ook veel tijd door uw bijdragen of programma's te bezorgen op velletjes A4 (formaat van het tijdschrift) , een redelijke marge links en rechts maakt het layout-werk makkelijker. Zorg aub voor een goede "zwarting" van uw bijdragen op papier, een nieuw lint op de printer of schijfmachine kan wonderen ver-richten ! Het is begrijpelijk dat niet alle bijdragen en artikels meteen hun weg vinden naar de bibliotheek of het tijdschrift. Mogelijk heeft een medelid uw idee vlugger,beter,eerder of duidelijker in een programma verwerkt. Een eenvoudige en klare documentatie maakt het ons mogelijk om uw werk vlot te apprecieren en verder te orienteren. Aarzel niet om nog eens contact te nemen of een verbeterde versie op te sturen indien uw inzending zonder gevolg gebleven is. Overigens is het aanbod van nieuwe en kwalitatieve software momenteel erg hoog : Verder in dit nummer vindt U tal van nieuwe pakketten. We zijn bijzonder fier op de creatie van Willy Coremans die met D-BASIC onze DAIcomputer nog meer allure en programmeergenot bezorgt. We kennen momenteel een belangrjke ledenaangroei in Frankrijk. Cedric Dufour heeft een paar medewerkers gevonden zodat onze afdeling in Frankrijk spoedig opnieuw aan de slag zal gaan. wij wensen U amusement en lering met nummer 22 en verder een prettige vacantie!.

Dear members,

We hope you can spend some interesting time with DAInamic 22 during your summer holidays. We welcome the many new French members. Cedric Dufour has good news for them : he found a few friends willing to assist him in managing DAInamic-division in France. They will announce their activities in next issue. For the moment, please contact Bruno Van Rompaey for subscriptions, game-programs can be send to Frank Druijff, other programs, articles for our newsletter and software-orders are welcome on the redaction in Herselt. In this (and next) issue you will find many new software packages, we are very proud with the realisation of D-BASIC by Willy Coremans,a very active member of DAInamic redaction.

We wish you relaxing summer holidays , dai dai ...

Wilfried Hermans

NEWSLETTER 22

INHOUD — CONTENTS

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

DAInamic subscription rates :

Benelux : 1000 Bfr
Europe : 1100 Bfr
Outside Europe 1500 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

1. Introduction

After a period of extensive testing, DAInamic is proud to present you the DBASIC V2.1 package.
The specifications given in in newsletter 20 page 7 are still valid for a 100%. However during testing some extra commands and statements have been added to extend the power of DBASIC.
Now up to 95 commands and/or statements are to the disposal of the programmer.

2. Brief description of the instruction-set.

NOTE : -only the statements and commands added by DBASIC are described here.
-for every instruction a very small description and eventualy an exemple is given.

+ definition of <label>
-program only
-a label has to be defined in the beginning of the line (cfr. DATA statement)
-restriction : a label in the first program-line will not be recognised in run-time.

ex.
```
...
1000 "SUBROUTINE1 FOR I=1 TO 10 ...
```

+ RESTORE <line number>
-program/command
-sets the datapointer to <line number>

+ RESTORE <label>
-program/command
-sets the datapointer to labeled line

ex.
```
...
10   RESTORE "BEGINDATA:...
...
10000 "BEGINDATA REM DATA FOR SUBROUTINE CALC
10010 DATA 10,20,...
...
```

+ RUN <label>
-command only
-starts program execution at labeled line

ex.
```
*RUN "START
*RUN "HELP
```

+ GOTO <label>
-program only
-proceeds program execution at labeled line

```
+ GOSUB <label>
    -program only
    -starts execution of the subroutine at labeled
     line
  ex.     ...
          2000  GOSUB "CALC
          ...
          30000 "CALC A%=F1%(B%):...:RETURN

+ IF <expression> GOTO <label>
    -program only
    -if the logical expression is true then proceeds
     program execution at labeled line, else continues
     program execution at the next statement
  ex.     ...
          100   IF A=1 GOTO "EXIT:A=1:...
          ...
          10000 "EXIT END

+ IF <expression> THEN ...:ELSE ...:END IF
    -program only
    -if the logical expression is true then executes
     statements between THEN and ELSE, else executes
     statements between ELSE and ENDIF
    -note : the ELSE statement is optional, END IF is
     required
  ex.     ...
          1200  IF HELPFLAG$="Y" THEN GOSUB "HELP:...
          ...
          1250  ELSE STOP
          1260  END IF
          ...

+ ON ERROR GOTO <line number>
    -program only
    -when a runtime-error occurs, program execution
     will proceed at <line number>
+ ON ERROR GOTO <label>
    -program only
    -when a runtime-error occurs, program execution
     will proceed at labeled line
  ex.     ...
          10    ON ERROR GOTO "TRAPERROR
          ...
          50000 "TRAPERROR IF ERR$="" GOTO "DBASICERROR
          50010 ELSE...
          ...

+ RESUME
    -program only
    -ends the error recovery routine and resumes pro-
     gram execution at the statement wich caused the
     error
```

```
+ RESUME <line number>
    -program only
    -ends the error recovery routine and resumes pro-
     gram execution at <line number>
+ RESUME <label>
    -program only
    -ends the error recovery routine and resumes pro-
     gram execution at <label>
+ RESUME NEXT
    -program only
    -ends the error recovery routine and resumes pro-
     gram execution at the statement immediately
     following the one wich caused the error
  ex.     ...
          50000 "TRAPERROR IF ERR=10 THEN RESUME 10
          50010 ELSE IF ERL=1000 RESUME "EXIT
          50020 ELSE RESUME NEXT:END IF :END IF
          ...

+ ON <expression> GOTO <label1>,<label2>,...
    -program only
    -when the value of <expression> is 1 then execu-
     tion proceeds at <label1>, when the value of
     <expression> is 2 then execution proceeds at
     <label2> ...etc...

+ ON <expression> GOSUB <label1>,<label2>,...
    -program only
    -when the value of <expression> is 1 then execu-
     tion of the subroutine at <label1> is started ...
  ex.     ...
          200   ON GETC-#30 GOSUB "INIT,"PROCESS,"EXIT

+ WHILE <expression> DO ...:WEND
    -program only
    -while <expression> is true executes the
     statements between DO and WEND
  ex.     ...
          1000  WHILE GETC=0 DO WEND:REM WAIT FOR KEY

+ REPEAT ...:UNTIL <expression>
    -program only
    -repeats statements between REPEAT and UNTIL,
     until <expression> is true
  ex.     ...
          2000  REPEAT A=A+1:UNTIL A=1000

+ ERROR <expression>
    -program/command
    -provokes the error with number <expression>
  ex.     *ERROR 50
          END PROCEDURE WITHOUT PROCEDURE
          *
```

```
+ ON BREAK GOTO <line number>
     -program only
     -suspends  program execution and starts  execution
      of the interrupt service routine at <line number>
      when the BREAK-key is pressed
+ ON BREAK GOTO <label>
     -program only
     -suspends  program execution and starts  execution
      of  the  interrupt  service  routine  at  <label>
      when the BREAK-key is pressed
  ex.      ...
           10   ON BREAK GOTO "INTERRUPT
           ...
           65000 "INTERRUPT PRINT I:I=I+1:CONTINUE

+ CONTINUE
     -program/command
     -continues program execution at the point where it
      was interrupted

+ DOKE <expression1>,<expression2>
     -program/command
     -puts the 2-byte value of <expression2> in address
      location <expression1>, low byte first
  ex.      ...
           2000  HEAP%=#29B:DOKE HEAP%,#4000

+ BREAK ON/BREAK OFF
     -program only
     -enables/disables interruption of a DBASIC program

+ COMPILE
     -command only
     -prepares the program for execution and checks for
      structural errors.
-note  :  -when  giving  the RUN command  the  compiler
          will  be called automaticaly if  the  program
          has not been compiled yet.
          -compilation  will  execute very  fast  since
          (D)BASIC is already semi-compiled
          -a compiled program will be auto-start

+ DEF PROC <name> <p1>,..VAR <v1>,..ARR <a1>,..FN <f1>.
     -program only
     -defines a procedure
      -<p1>,..<pi>  are value-parameters,  they will be
       assigned  a value equal to the actual  value  of
       the expressions in the caller.
      -<v1>,..<vj>   are   variable-parameters,   these
       variables   will   refer   to   the   corresponding
       variables in the caller.
      -<a1>,..<ak>  are  array-parameters,  they  will
       refer to the corresponding arrays in the caller.
```

-<f1>,..<fl> are function-parameters, the symbols will refer to the corresponding expressions in the caller.

ex.
```
...
10    DEF PROC PLOT MIN%,MAX% VAR X% FN Z
20    FOR X%=MIN% TO MAX%
30    DOT X%,Z 5
40    NEXT
50    END PROC
...
```
-this procedure can be called by
```
...
1000  PLOT 0,YMAX,X%,(X%*X%)/YMAX
...
```
-all parameters are optional

+ PROCEDURE ...:END PROC
-see DEF PROC ...

ex.
```
...
10    PROCEDURE HOME:PRINT CHR$(12);:END PROC
```

+ DEF FN ...:END FN/FUNCTION ...:END FN
-program only
-defines a function
-the parameters are of the same type as procedure-parameters.

ex.
```
...
20    FUNCTION FAC%(N%)
30    IF N%>1 THEN FN = N%*FAC%(N%-1)
40    ELSE FN = 1
50    END IF :END FN
...
```
-note : -a function has a certain type (fpt/int/$)
-for rather simple functions a simplified syntax is possible

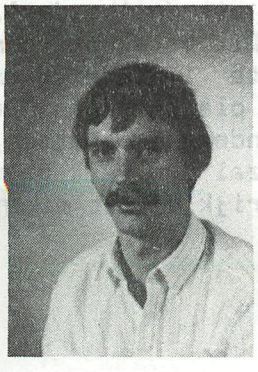
ex.
```
...
10    DEF FN LOWCASE$=CHR$(GETC IOR #20)
```

+ FN = <expression>
-program only
-ends the execution of a function

+ LOCAL <v1>,<v2>,...
-program only
-declares variables <v1>,<v2>,.. as local to the procedure or function

I hope this bit of information will give you an idea of the possibilities of DBASIC. In the next issue I will discuss in more detail procedures, functions, new implicit defined functions and DBASIC extensions.

Willy Coremans



Willy Coremans, o 20/5/58 , electronics engineer, joined DAInamic redaction in oct 82 . Software analyst by profession, he worked on D-BASIC from oct 83 until now .

P R O G R A M M E E R T E C H N I E K E N

Regelmatig krijg ik vragen over het een of ander gesteld . Soms, als het een specifiek probleem voor een persoon is, geef ik die direct antwoord (vermeldt aub telefoonnummer!), maar andere keren schrijf ik het probleem op en laat het dan tijdelijk rusten. Later zal het dan in een artikel aan bod komen, maar dan niet alleen, maar in een context van allerlei soortgelijke en er mee samenhangende problemen. Deze keer wilde ik een aantal problemen behandelen, die alle iets met MODE 0 van doen hebben . Dit onderwerp is echter zo omvangrijk gebleken, dat ik het over twee artikelen wil verdelen . Sommige onderwerpen behoren zijdelings bij een MODE 0 artikel, maar zijn ook betrokken bij een artikel dat over PRINT gaat. Daarom zal ik vervolgens daar een artikel over schrijven.

Eerst de algemene theorie nog maar eens doornemen hoe de standaard tekstmode MODE 0 is opgebouwd. Het beeldgeheugen hiervoor start op adres #BFFF en wordt dan naar beneden toe uitgelezen. Dat wil zeggen dat de eerste byte waar de DAI naar kijkt om te weten hoe het beeld is opgebouwd #BFFF is en de tweede #BFFE, de derde #BFFD enz.. Als U dan ook een bepaalde plaats in het beeld wilt veranderen door middel van een POKE op adres XX en U wilt daarna een plaats rechts of onder de vorige plaats veranderen, dan zult U moeten POKE'n in een adres dat kleiner is dan de XX van daarvoor. De eerste 16 (#10) bytes zijn onderverdeeld in 4 groepjes van 4. Zij vertellen de DAI dat we in MODE 0 werken en welke 4 kleuren we willen gebruiken. Zo'n groepje van 4 bytes is eigenlijk niets anders dan een regel in de zgn UNIT COLOUR MODE.
De kenners zullen echter al een fout in de voorafgaande regels hebben gezien. Ik ben mij daar wel bewust van, maar ik wil liever de beginner in deze stof, eerst een simpele uitleg, die dan later wordt verbeterd, geven dan ineens mensen afschrikken, die als zij hadden doorgelezen het best hadden begrepen.

```
* * * * * i n t e r m e z z o * * * * * * * i n t e r m e z z o * * * * * *
*                                                                         *
* #BFFF bevat #36 = 0011 0110 d.w.z.: 00 vier kleuren grafisch !!!!!      *
*                                     11 hoogste resolutie (tekst/mode 7/8) *
*                                     0110 = 6 dus 7 regels hoogte        *
* #BFFE bevat #88 = 1000 1000 d.w.z.: 1 kleurverandering mogelijk         *
*                                     0 maak unit colour mode             *
*                                     00 verander kleurregister 0         *
*                                        eerste van de COLORT = kleur 20  *
*                                     1000 = 8 achtergrond is grijs       *
* #BFFD en #BFFC bevatten standaard 0 dus geen punten zichtbaar.          *
*                                                                         *
* Bovenstaande informatie zorgt er voor dat bovenaan het beeld zeven grijze *
* regels komen. De andere groepjes van vier bevatten dezelfde eerste, derde *
* en vierde byte, maar de tweede byte zal een van de andere kleuren doen  *
* veranderen. #BFFA = #9x, #BFF6 = #Ay en #BFF2 = #Bz, na COLORT a x y z.  *
*                                                                         *
* * * * * i n t e r m e z z o * * * * * * * i n t e r m e z z o * * * * * *
```

Het MODE 0 - beeld wordt dan in het geheugen opgebouwd door 24 regels van elk 134 bytes. De eerste twee bytes zijn de zgn controlebytes voor de gehele regel Zij zijn opgebouwd op dezelfde wijze als de twee eerste bytes uit de 'header'. Na een MODE 0 opdracht zijn zij respectievelijk #7A en #40. Dit betekent voor de DAI tekstregels met vier kleuren mogelijkheid en regelhoogte 11 terwijl er geen unit colour mode is en er ook geen kleurverandering moet plaatsvinden. De laatste zes bits , die de informatie voor eventuele kleurverandering bevatten, worden genegeerd als het eerste bit 0 is . 'Spelen' met POKE zal dus vaak geen gevolg hebben. We kunnen echter wel hiermee een aardig kleurrijk effect maken

zonder allerlei ingrijpende veranderingen aan te hoeven brengen. We veranderen de kleurcontrolebyte(s) zodat bv tekst of achtergrond van kleur veranderen. Om dit te doen moeten we het volgende doen. Bepaal het adres van de byte die veranderd moet worden; bv vanaf regel 10 willen we de achtergrondkleur veranderen van de standaard 8 naar bv 11. Het adres is nu #BFEF-100*#86-1, te weten startadres van de eerste regel min aantal regels (10) maal de regellengte (#86=134) min een om de kleurcontrolebyte te krijgen . Om kleur te kunnen veranderen zal bit 7 (het meest linker) moeten aanstaan (een zijn), bit 6 moet aanstaan om de unit colour mode te voorkomen, bit 4,5 moeten 0 zijn om juist kleur 20 (achtergrondkleur) te veranderen en tot slot moet 0,1,2,3 gevuld worden met de nieuwe kleur hier 11 dus #B. We doen dus een POKE #BFFE-10*#86-1,#CB en we hebben een nieuwe achtergrondkleur na de bovenste 10 regels. Wilt U deze kleur alleen laten gelden voor die regel en de volgende regel de oorspronkelijke achtergrondkleur laten heben, zal U de kleurcontrolebyte van die volgende regel moeten gaan veranderen. U doet dit mbv POKE #BFEF-11*#86-1,#C8.

*** : Het is mogelijk het gehele beeld op en neer te laten gaan door ***
TIP enkel de regelhoogte(s) in de 'header' te veranderen. TIP
*** Vervang de regelhoogte 7 bv door 16 in #BFFF. ***
Mark Hooykaas gebruikte dit in spelonk (GC 10).
tik maar in:

```
10    B=#BFFF:FOR I=0 TO 10:POKE B,#3F:WAIT TIME 2:POKE B,#36:WAIT TIME 2:NEXT
```

Ik heb zoeven echter een belangrijke bron van vergissingen nog niet vermeld. De beeldopbouw en de uitlezing daarvan doet vermoeden dat de bovenste regel in het beeld regelnummer een of eventueel nul heeft, maar dit is jammer genoeg in tegenspraak met de CURSOR instructie die juist de onderste regel als regelnummer nul neemt en de bovenste als nummer 23.

We gaan wat dieper in op de opbouw van de regels in MODE 0. We zagen al dat de regel 134 (=#86) bytes lang is. Elke regel begint met twee controlebytes die aangeven dat dit een tekstregel is met regelhoogte 11 (#A ! inderdaad niet #B) en dat er geen unit colour mode regel volgt en er geen kleurveranderingen plaatsvinden. De regel zelf bestaat uit 60 karakters die elk een byte nodig hebben en daarnaast een kleurcontrolebyte. Een eenvoudig rekensommetje leert ons echter dat dit nog geen 134 bytes zijn. (60*2+2=122) Wat zijn de laatste 12 bytes dan ? Nu die zijn eerlijk verdeeld over begin en einde van de regel. Elke regel begint met drie karakters en eindigt met drie karakters, die met de 'normale' BASIC instructies echter niet gemaakt kunnen worden. De PRINT, LIST, en EDIT bv beginnen allemaal met pas het vierde karakter. Dit is gedaan om de afstelling van uw scherm op de computer te vergemakkelijken. Aan de randen van het beeld krijgen we anders snel, dat bepaalde tekens niet meer leesbaar zijn. Maar met POKE kunnen we die bytes echter toch vullen . Het volgende programma zal U duidelijk maken hoe dat gedaan kan worden.

```
000     10      REM FIRST & LAST COLUMNS / F.H. DRUIJFF - 6/84      -<
001     20      MODE 0:COLORT 8 0 9 14:A=48                         -<
002     30      FOR I=#BFEF TO I-23*#86 STEP -#86                   -<
003     40      POKE I-2,48:POKE I-4,A:POKE I-6,48+B                -<
004     50      POKE I-5,#FF:POKE I-7,#FF:POKE I-9,#FF              -<
005     60      POKE I-128,ASC("-"):POKE I-130,ASC("<")             -<
007     70      POKE I-131,#FF:POKE I-133,#FF                       -<
008     80      B=(B+1) MOD 10:A=A+1-SGN(B):NEXT                    -<
009     90      LIST-:END
```

Maar, zult U zeggen, het waren toch drie karakters aan het eind van de regel ? Juist , maar voordat we dat kunnen begrijpen moet ik eerst iets vertellen over het gebruik van de vier kleuren in tekst. De kleuren zijn in te stellen met de

COLORT A B C D met A,B,C en D getallen van 0 t/m 15, COLORT 8 0 0 8 is standaard. Pas op met de getallen; er wordt bij intikken geen controle op uitgevoerd ! COLORT 99 999 9999 99999 geeft geen SYNTAX ERROR maar kan niet uitgevoerd worden. Overigens wordt integer verondersteld maar floating point is wel toegestaan. De kleur van de letters en de bij die letter horende achtergrond is afhankelijk van de kleurenset van die plek en de inhoud van de bij die plek horende colourbyte. Anders dan in de vier kleuren grafische modes kunnen we dus niet alle naast elkaar liggende punten een van de vier kleuren geven. We nemen even COLORT A B C D na een MODE 0 en geen extra veranderingen en we bekijken een bepaalde plaats op het beeldscherm. Hiervoor nemen we bv de plaats

*** : Om een schone bladzijde in MODE 0 te krijgen moet U niet, zoals vaak ***
TIP gedacht wordt een PRINT CHR$(12) geven, maar PRINT CHR$(12); . TIP
*** Zonder de puntkomma wordt de bovenste regel gelijk overgeslagen . ***
Bij een eventuele scroll van een regel krijgen we een opschuivend beeld, dat nieuwe gebruikers doet vermoeden dat ze iets missen en gebruikers, die het programma kennen, irriteert.

linksboven in het beeld. We plaatsen hier een "O" dmv CURSOR 0,23:PRINT "O" en zien dat de achtergrond kleur van de 'O' kleur A is en de letter zelf kleur B heeft. Als we nu de colourbyte vullen met #FF zal de achtergrond kleur C worden en de letter zelf kleur D. Als we de colour byte met #0F (=00001111) vullen zullen de eerste vier kolommen van de letter (bits zijn 0) de normale voor en achtergrondkleur hebben (dus A en B) en de laatste vier kolommen zullen uit de tweede set kleuren (dus C en D) opgebouwd zijn.
De colourbyte is NIET de byte naast de byte , waar de karaktercode in staat maar drie bytes verder; dwz adres-3. Dit betekent echter , dat de colourbyte , die behoort bij het laatste karakter van een regel tevens de regelcontrolebyte is van de volgende regel. U begrijpt dat deze 'fout' moeilijk te verwerken is zodanig dat we er geen last van hebben.

*** : We kunnen in vier kleuren mode wel degelijk alle kleuren op een ***
TIP beeld gebruiken. Hiervoor moeten we dan bij elke regel opnieuw een TIP
*** van de vier kleuren veranderen. Het recept hiervoor staat in het ***
artikel en ook in een der programma's.

Het veranderen van de kleurcontrolebytes is dus mogelijk door de adressen uit te rekenen en die te vullen met #FF. Zie mijn programma 'SPOT ON TEXT' maar eenvoudiger is het meestal gebruik te maken van de cursor. Als de cursor op de juiste plaats staat, staat de betreffende kleurbyte in #76; als we deze informatie veranderen in #FF (of iets anders indien gewenst) zal de DAI zelf het wel op zijn plaats zetten. Het programma hierna maakt het hopelijk duidelijk.

```
10      REM 4 COLOUR CHARACTERS / F.H. DRUIJFF - 5/84
20      MODE 0:PRINT CHR$(12);:POKE #74,0:POKE #75,255:COLORT 8 0 8 15
30      S$="12345678901234567890123456789012345678901234567890123456789012345678901234567890"
40      T$="colour demo"
50      FOR I=0 TO 22:PRINT S$:NEXT:PRINT S$;
60      LC=LC+1:L=RND(50.0):CURSOR L,24-LC
70      FOR I=0 TO 10:POKE #76,#FF:PRINT MID$(T$,I,1);:NEXT
80      IF LC<24 GOTO 60:GOTO 80
```

U zult als U de moeite neemt om dit programma in te tikken en het ook te laten werken, merken dat de kleurgegevens niet mee doen in de scroll (dat is het omhoog schuiven van het beeld om plaats te maken voor een nieuwe regel). Ik kom daar op terug. Nu eerst het programma met 16 kleuren in een 4 kleurenopbouw. We kunnen daarvoor hetzelfde programma als hiervoor gebruiken en er hoeft maar een regel aan toegevoegd te worden. Regel 10 veranderen we voor de netheid natuurlijk ook even.

```
  10      REM 4/16 COLOUR CHARACTERS / F.H. DRUIJFF - 5/84
```
De nieuwe regel geven we nummer 65, zodat hij er zo tussen kan.
```
  65      POKE #C074-LC*#86,#F1+RND(15.0)
```

N.B. #C074 = #BFEE + #86 nodig omdat LC begint met 1 en niet 0.
We komen nu toe aan de 16 kleurentekstmode. Ook in tekst is het mogelijk om met zestien kleuren te werken. De methode van codering lijkt veel op die in de 16-kleuren grafische modes. De kleurcontrolebyte bevat dan de twee kleuren die wij per letter als achtergrondkleur en voorgrondkleur wensen. Maar hoe kan de DAI dan weten, dat wij in zestienkleurenmode wensen te werken en niet in de kleurcontrolebyte alleen een 4-kleurencode hebben gegeven. Welnu dit doen we met de regelcontrolebyte. Pas op de eerste en niet de tweede !! Dit kan verwarrend zijn; daar we zoeven wel in de tweede byte iets moesten veranderen om iets aan de kleur te doen. De eerste regelcontrolebyte (#BFEF - .. x #86) moet nu beginnen met 1 (bit 7). Aangezien we nog wel de hoogste resolutie wensen en de regelhoogte normaal willen laten moeten we POKE'n met #FA. Oefen wat met deze stof om het goed door te krijgen. In het begin willen we nog wel eens de voorgrondkleur en achtergrondkleur in de verkeerde nibble (halve byte) plaatsen. Hieronder een demonstratieprogramma, waarbij weer gebruik gemaakt wordt van de cursor. Voor onze rekenaars zie DAInamic 17 blz. 219.

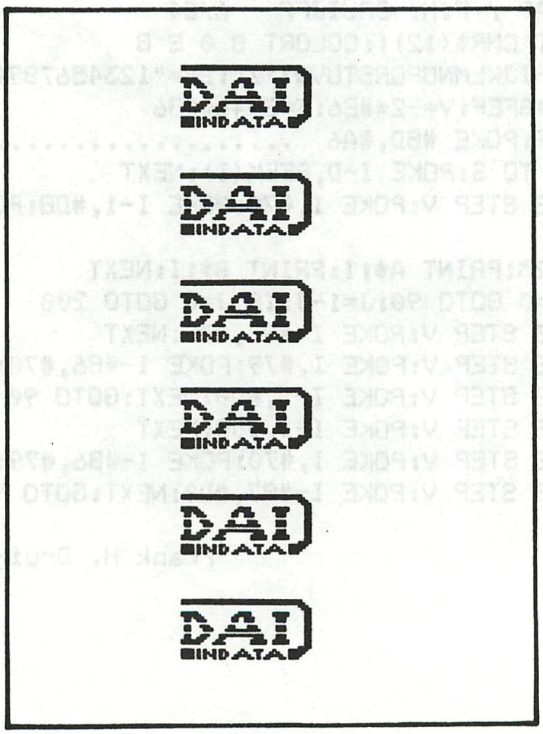
```
  10      REM 16 COLOUR CHARACTERS / F.H. DRUIJFF - 5/84
  20      MODE 0:PRINT CHR$(12);:COLORT 8 0 0 8
  30      S$="12345678901234567890123456789012345678901234567890"
  40      FOR I=1 TO 23:PRINT S$:NEXT:PRINT S$;
  50      LC=LC+1:LCB=#C075-LC*#86:POKE LCB,#FA
  60      FOR I=1 TO 60:C=(C+1) MOD 16:POKE LCB-9-I-I,C*16+8:NEXT
  70      IF LC<24 GOTO 50
```

Nu tot slot iets over de regelhoogtes. In elke eerste regelcontrolebyte staat standaard #7A, waarmee de linerepeatcount op 10 gezet wordt. De regelhoogte is dan 1+10=11, wat overeen komt met 22 scans op uw beeldbuis. Veranderen wij nu de regelhoogtes kunnen we daar grappige effecten mee maken. In de aankondiging van een Games Collection wordt van een aantal van de bovenste regels de regelhoogte op maximum gezet. Hierdoor zijn de onderste regels waar tekst in staat niet zichtbaar. Nu worden de regelhoogtes kleiner gemaakt. De tekst schuift daardoor langzaam van onder af het beeld in. Een ander aardig effect laat ik hieronder zien. We breiden MODE 0 uit tot 48 regels, die om en om regelhoogte 1 of 10 krijgen en tekstkleur identiek achtergrondkleur of verschillend. We wachten op een toetsaanslag en dan worden alleen regelcontrolebytes gewijzigd.

```
  10      REM PAGE SWAP / F.H. DRUIJFF - 6/84
  20      MODE 0:PRINT CHR$(12);:COLORT 8 0 8 8
  30      A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":B$="1234567890"
  40      D=24*#86:S=#BFEF:V=-2*#86:E=S-46*#86
  50      POKE #8C,#CF:POKE #8D,#A6  ..................uitleg volgend nummer !
  60      FOR I=#B350 TO S:POKE I-D,PEEK(I):NEXT
  70      FOR I=S TO E STEP V:POKE I,#70:POKE I-1,#D8:POKE I-#86,#79:
                                                        POKE I-#87,#D0:NEXT
  80      FOR I=1 TO 23:PRINT A$;I:PRINT B$;I:NEXT
  90      H=GETC:IF H=0 GOTO 90:J=1-J:IF J=0 GOTO 200
 100      FOR I=S TO E STEP V:POKE I-#87,#D8:NEXT
 110      FOR I=S TO E STEP V:POKE I,#79:POKE I-#86,#70:NEXT
 120      FOR I=S TO E STEP V:POKE I-1,#D0:NEXT:GOTO 90
 200      FOR I=S TO E STEP V:POKE I-1,#D8:NEXT
 210      FOR I=S TO E STEP V:POKE I,#70:POKE I-#86,#79:NEXT
 220      FOR I=S TO E STEP V:POKE I-#87,#D0:NEXT:GOTO 90
```

Frank H. Druijff

```
1      REM ****************************
2      REM * DAI LOGO on  EPSON RX-80 *
3      REM * DUAL DENSITY BIT GRAPHIC *
4      REM *  Daniel   R O S S I O N  *
5      REM *   13  W MSL -  W / OPS   *
6      REM *   B-4090  POST 12  FBA   *
7      REM ****************************
10     RESTORE:GOTO 100
20     PRINT CHR$(X%);
30     RETURN
100    FOR A%=1 TO 4
110    PRINT CHR$(27);"L";
120    X%=98:GOSUB 20:X%=0:GOSUB 20
130    FOR I%=1 TO 49:READ X%:GOSUB 20:GOSUB 20:NEXT I%
140    PRINT CHR$(27);"3";CHR$(20);CHR$(9)
150    NEXT A%
160    PRINT :PRINT :GOTO 10
200    END
1000   DATA 0,0,0,0,24,24,24,24,24,24,24,24,24,24
1010   DATA 24,24,24,24,24,24,24,24,24,24,24,24,24,24
1020   DATA 24,24,24,24,24,24,24,24,24,24,24,24,24,28
1030   DATA 14,7,3,0,0,0,0
1040   DATA 0,0,0,0,96,123,123,96,112,56,26,11,3,1
1050   DATA 0,0,0,0,0,3,27,56,96,96,120,123,27,3
1060   DATA 0,0,0,0,0,96,123,123,123,96,0,0,0,0
1070   DATA 0,255,255,0,0,0,0
1080   DATA 0,0,0,0,6,222,222,6,6,14,28,216,208,192
1090   DATA 2,6,30,26,208,208,208,208,208,208,208,208,214,222
1100   DATA 222,30,6,0,0,6,222,222,222,6,0,0,0,0
1110   DATA 0,255,255,0,0,0,0
1120   DATA 0,0,0,0,60,60,60,60,0,60,0,60,16,8
1130   DATA 60,0,60,36,24,0,4,12,24,40,24,12,4,32
1140   DATA 60,32,4,12,24,40,24,12,4,0,60,60,60,120
1150   DATA 240,224,192,0,0,0,0
```



DAI LOGO

```
002                 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
003                 x                                    Berlin, 8/12/1983  x
004                 x                                                       x
005                 x  Use this program to write memory-contents to,        x
006                 x  or read them from tape under use of the Basic        x
007                 x  or with ML-programs.                                 x
008                 x                                                       x
009                 x Writing:                                              x
010                 x In Basic first do a ) INPUT NAME§:POKE§13F,LEN        x
011                 x (NAME§) ( in order to put a name for the tape-        x
012                 x sequenze into the input-buffer.Don't worry about      x
013                 x the stringname; it will not be used.                  x
014                 x Using this program from a ML-program, you have        x
015                 x to put the name directly into the input-buffer        x
016                 x which has to be bilt up in the following form:        x
017                 x §13E: §19 , §13F: laength of name ,                   x
018                 x §140 - §1BD : name                                    x
019                 x Afterwards poke the beginn-address of the memory-     x
020                 x area, which has to be written on tape, into           x
021                 x memorylocation ADR1 (low byte first).In the same      x
022                 x way you put the endadress into ADR2. Then the         x
023                 x program is started with CALLM WRITE. (You find        x
024                 x the values of ADR1,ADR2 and WRITE in the Symbol-      x
025                 x Table.)                                               x
026                 x                                                       x
027                 x Reading:                                              x
028                 x Put the tape-sequenze-name into the input-buffer      x
029                 x like discribed above. Then poke the offset for        x
030                 x reading from tape (same meaning like R XXXX in        x
031                 x utility) into OFFSET like discribed above.            x
032                 x The program is started with CALLM READ. (You          x
033                 x find READ in the Symbol-Table.)                       x
034                 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
035
036                 xxx WRITING A MEMORY-AREA ON TAPE: xxx
037                 x Deklarations:
038                 INBUF   EQU   :013F    startaddress input-buffer
039                 BSW1    EQU   :0040    bankswitchaddress 1
040                 BSW2    EQU   :FD06    bankswitchaddress 2
041                 BANK0   EQU   :30      code for bank 0
042                 BANK3   EQU   :F0      code for bank 3
043
044                         ORG   :0400    start of WRITE
045 0400 F5         WRITE   PUSH  PSW
046 0401 E5                 PUSH  H
047 0402 C5                 PUSH  B
048 0403 D5                 PUSH  D
049 0404 3EF0               MVI   A,BANK3  switch bank 3
050 0406 324000             STA   BSW1
051 0409 3206FD             STA   BSW2
052 040C 211F04             LXI   H,FORT1  store address for
053 040F E5                 PUSH  H        continuing after writing
054 0410 2A2E04             LHLD  ADR2     load endaddress
```

```
PAGE 02    WRITE & READ    R.Kietzmann

055 0413 EB                XCHG            into DE,
056 0414 2A2C04            LHLD  ADR1      beginnaddress into HL
057 0417 E5                PUSH  H         store them
058 0418 D5                PUSH  D
059 0419 213F01            LXI   H,INBUF   start input-buffer
060 041C C3F0EE            JMP   :EEF0     call WRITE in ROM
061 041F 3E30       FORT1  MVI   A,BANK0   switch bank 0
062 0421 324000            STA   BSW1
063 0424 3206FD            STA   BSW2
064 0427 D1                POP   D
065 0428 C1                POP   B
066 0429 E1                POP   H
067 042A F1                POP   PSW
068 042B C9                RET
069 042C            ADR1   RES   2         store beginnaddress here
070 042E            ADR2   RES   2         store endaddress here
071
072                 xxx READING A MEMORY-AREA FROM TAPE: xxx
073
074 0430 F5         READ   PUSH  PSW       start of READ
075 0431 E5                PUSH  H
076 0432 C5                PUSH  B
077 0433 D5                PUSH  D
078 0434 3EF0              MVI   A,BANK3   switch bank 3
079 0436 324000            STA   BSW1
080 0439 3206FD            STA   BSW2
081 043C 214A04            LXI   H,FORT2   store address for
082 043F E5                PUSH  H         continuing after reading
083 0440 2A5704            LHLD  OFFSET    load offset into HL
084 0443 E5                PUSH  H         store it
085 0444 213F01            LXI   H,INBUF   startaddress input-buffer
086 0447 C317EF            JMP   :EF17     call READ in ROM
087 044A 3E30       FORT2  MVI   A,BANK0   switch bank 0
088 044C 324000            STA   BSW1
089 044F 3206FD            STA   BSW2
090 0452 D1                POP   D
091 0453 C1                POP   B
092 0454 E1                POP   H
093 0455 F1                POP   PSW
094 0456 C9                RET
095 0457            OFFSET RES   2         store Offset here
096 0459                   END

xxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx Symbol  Tabelle xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx

ADR1   042C   ADR2   042E   BANK0  0030   BANK3  00F0
BSW1   0040   BSW2   FD06   FORT1  041F   FORT2  044A
INBUF  013F   OFFSET 0457   READ   0430   WRITE  0400
```

```
0400 F5 E5 C5 D5 3E F0 32 40 00 32 06 FD 21 1F 04 E5
0410 2A 2E 04 EB 2A 2C 04 E5 D5 21 3F 01 C3 F0 EE 3E
0420 30 32 40 00 32 06 FD D1 C1 E1 F1 C9

0430 F5 E5 C5 D5 3E F0 32 40 00 32 06 FD 21 4A 04 E5
0440 2A 57 04 E5 21 3F 01 C3 17 EF 3E 30 32 40 00 32
0450 06 FD D1 C1 E1 F1 C9
```

TIMING PROBLEMS

(from DAInamic 16, page 153)

While developing BASICODE it struck me that there were inconsistencies in the instruction times of the 8080 microprocessor. It seemed that the cause of this was as follows. The hardware is so constructed that the microprocessor and the video processor make use of the RAM in turn. However at the moment of frame fly-back, (that is when the electron beam changes from bottom right to top left of the screen), the video processor is not using the RAM and the microprocessor is running at its full capacity. Possibly this also happens with the line synch (line fly-back). The microprocessor works most effectively at an average frequency of 1.17 MHz, so normally one would not notice any effect. However when the periodicity is short or the frequency is high, such as during the reception of BASICODE, telex or morse and certainly while receiving satellite transmissions, then the resulting frequency will not be what was expected. Moreover there will be 50Hz interference from the screen fly-back (this is not hum from mains lighting etc). The interference could possibly be reduced with the help of sub-routines in ROM, (delays etc). But the ROM is not involved in the above effect because the video processor does not make use of it. With a periodicity of more than 50 to 100 milliseconds there will be little problem because the screen fly-back will not cause difficulty. I hope these notes will spare some folks from sleepless nights.

With friendly greetings,

Th. van Lieshout.

PROGRAMMING TECHNIQUES

(from DAInamic 16, page 154)

On this occasion I want to put the running speed of programs under the magnifying glass. Of course the first problem is how one measures the speed of a particular set of instructions but here the DAI can be put to good use. It is as well to write a short timer program which can form a sub section of the program that is to be checked. Here is such a program:-

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

These lines can be put in the section to be tested. The variables T1BE and T1BF are of course integers. We make use of the 20mS clock to reduce the count on #1BE and #1BF by 1 every 20 milli-seconds until the count is zero. The POKEs in line 5 fill these two bytes with #FFFF. The WAIT TIME ensures a clean start. Also, since the WAIT TIME instruction makes use of these addresses, it should not appear in the part of program being tested. Reading the addresses is best done in the order shown otherwise the timing could go wrong; an error could be caused by a count-down pulse arriving between the two PEEKs, resulting in a 20mS shortfall. There still remains another chance of error: at the 256th turn #1BE will be at zero and a pulse then would reduce it by 1, resulting in a time not of 20mS but of 256*20, about 5 seconds. However, an error of this magnitude is large enough to be noticeable. Note that the division in line 95 must be by 50.0 and not by 50 because a floating point result is needed.

We will experiment with this timer and no doubt some will find the results surprising. A single instruction is executed so quickly that it cannot be timed with this method, so we will put a

FOR-NEXT loop around it. The timing of the loop itself:

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

and do it after an IMP FPT and after an IMP INT. Then try:

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

and this too is a useless loop which is run through 5000 times?

Now let us put something in between, such as A=3 or A=B. The time mounts up so we can reduce the loop to 1000 or 500, or even 100 when functions like SQR, SIN, COS or TAN are used. It is useful to try changing the loop from say 1000 to 10000 to see if in fact it does take 10 times as long. For practical reasons a time of between 10 and 30 seconds is recommended, although the maximum possible measurement is #FFFF*20 mS, about 20 minutes.

Now experiment further by putting inside the loop A=8 or A=8.0 or A=B or A=B+4 or A=4+B, after an IMP INT and also after an IMP FPT. Integer is not invariably quicker! If you have doubts try A%=SQR(P%) and A!=SQR(P!).

To give A the value of twice B plus 4, compare the following possibilities:

```
A=2*B+4     convenient isn't it?
A=B*2+4     different?
A=B+B+4     this is quicker !!!
A=B+4+B  or  A=4+B+B
A=(B+B)+4   does this make any difference ?
A=B+(B+4)   must be wrong.
A=(((((B)))))+((((B))))+(((4)))    pleases me.
```

But to continue: A=B SHL 1 is faster than A=B*B but beware, A=B SHL 5 is slower than A=B*32. Try the difference between having the whole of a FOR-NEXT loop on a single line and spreading it over 2 or 3 lines.

In the second part I wish to deal with the instructions GOTO, GOSUB, IF xx THEN, and ON W GOSUB. I never use IF xx THEN followed by a line number. I consider a jump command of sufficient importance for it to be clearly indicated by a GOTO and therefore use IF xx GOTO. The GOTO instruction is the most damnable instruction in BASIC, more suited to use in an Italian kitchen, being ideal for the production of spaghetti. Here is the kind of thing I often come across in programs submitted to us:

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

Naturally it works, otherwise its author may have had second thoughts. If A equals three, P becomes two, else P is unchanged. But it can be done better:

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

or if we want to keep it short:

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

or if we want to avoid GOTO:

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

Even the IF can be avoided but I do not recommend this as a convenient method:

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

Try to find your own way of avoiding IF and then try this line.

In an earlier article there was a recommendation to jump higher rather than lower in the program for the sake of speed, but did you know that ON W GOTO xx is faster than IF xx GOTO? For example, IF PEEK(I)=1 GOTO 200 is slower than ON PEEK(I) GOTO 200.

Testing variables by means of ON W GOTO will not always be appropriate. An extra calculation can cancel out the gain achieved. Check at the end if there are too many GOTOs in the program. I have actually seen in one example, in Line 50 GOTO 170 and then in 170, GOTO 220 and then in 220 GOTO 30. That was the result of making changes to an earlier version of the program.

Now, regarding the GOSUB, while the clarity of a program benefits from a single entry and exit from a subroutine, sometimes the speed can be improved by additional entry and exit points for the same subroutine. Example:

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

In such cases I choose to use: 2000 IF A=5 GOTO 2020, thus letting clarity prevail.
I come across the command ON xx GOSUB so rarely that I think it can be little known.
Elsewhere in this issue you will find an article by Just van Dunne' about the GOTO. What he describes can be very useful on some occasions but it does take longer than the normal GOTO. There is also a short program of mine under the heading PROBLEM PROGRAM.

Some reactions to the 'circle' program in issue 14 have been arriving. Mostly they report fast drawing of ellipses instead of true circles. Thank you for the comments; we cannot publish them all but maybe a typical one or a combination. If your contribution is not explicitly quoted please do not refrain from sending further offerings as we are pleased to receive them and certainly take good note of the contents.

Good luck with the timer and if you make any startling discoveries don't keep them to yourself.

Frank H Druijff

ROTATING BUNGALOW

(from DAInamic 16, page 158-160)

```
1       REM ************************************
2       REM ****   3D PLOT VERSION 19.6.82   ****
3       REM ***  A. VINGERLING, ROTTERDAM  ***
4       REM ************************************
100     CLEAR 2750: PRINT CHR$(12);: GOSUB 7000: GOTO 103
101     COLORT 5 15 0 0: MODE 0: CURSOR 13,12
102     PRINT "INITIALISATION, PLEASE WAIT"
103     SF=1.0: GOSUB 2000: REM READ DATA
104     GOSUB 6000: REM   DISPLAY POSSIBILITIES
120     COLORG 15 0 15 0: MODE 6: MODE 6
130     GOSUB 600: REM   DRAW NEW PLAN
140     GOSUB 810: REM   CHANGE COLOUR
150     GOSUB 700: REM   DELETE OLD PLAN
160     Q%=1-Q%: KS=ABS(KS)
170     IF FL%=1.0 THEN 210
180     A%=GETC: GOSUB 880: IF A%<16 THEN 180
190     IF A%=ASC("H") THEN GOSUB 4000: GOTO 130
200     GOSUB 5000: ON P210% GOTO 210
205     IF A%=ASC("/") AND AA%=0 THEN 180
206     GOTO 230
210     P210%=0: IF AA%>0.0 THEN A%=AAA%(AA%): AA%=AA%-1: GOTO 230
220     AA%=0: FL%=0: GOTO 180
230     IF A%=ASC(",") THEN VF=1.0/0.8: GOTO 250: REM  >
240     VF=0.8: REM  <
250     IF A%=ASC("1") THEN A%=1: GOTO 280
260     IF A%=ASC("2") THEN A%=2: GOTO 280
270     IF A%<=19 THEN A%=A%-13
280     FOR P=1.0 TO NP: XX(P)=X(P): YY(P)=Y(P): NEXT
290     IF A%=ASC(",") OR A%=ASC(",") THEN GOSUB 900: GOTO 130
300     IF A%>0 THEN ON A% GOTO 320,330,400,410,500,510
305     IF A%=ASC("C") THEN GOSUB 800: CF%=1-CF%: GOTO 130
310     GOTO 180
320     KS=-KS
330     FOR P%=1 TO NP%: X=X(P%): Y=Y(P%): X(P%)=X*KC+Y*KS: Y(P%)=Y*KC-X*KS: NEXT
340     GOTO 130
400     KS=-KS
410     FOR P%=1 TO NP%: Y=Y(P%): Z=Z(P%): Y(P%)=Y*KC+Z*KS: Z(P%)=Z*KC-Y*KS: NEXT
420     GOTO 130
500     KS=-KS
510     FOR P%=1 TO NP%: Z=Z(P%): X=X(P%): Z(P%)=Z*KC+X*KS: X(P%)=X*KC-Z*KS: NEXT
520     GOTO 130
600     REM   DRAW NEW PLAN
610     FOR L%=1 TO NL%: PA%=LA%(L%): PB%=LB%(L%)
620     DRAW X(PA%)+XC%,Y(PA%)+YC%  X(PB%)+XC%,Y(PB%)+YC%  17+Q%*2
630     NEXT:RETURN
700     REM DELETE OLD PLAN
710     FILL 0,0 XMAX,YMAX 18-2*Q%: RETURN
800     IF CF%=0 THEN FILL 306,0 300,6 15: GOTO 802
801     FILL 306,0 300,6 0
802     RETURN
```

```
810    IF CF%=0 THEN COLORG 0 15*(1-Q%) 15*Q% 15: GOTO 812
811    COLORG 15 15*Q% 15-Q%*15 0
812    RETURN
880    QT%=QT%+1: IF QT%<15 THEN DRAW 305,1 300,6 15: DRAW 305,6 300,1 15: RETURN
881    DRAW 305,1 300,6 0: DRAW 305,6 300,1 0: IF QT%<30 THEN RETURN
882    QT%=0: RETURN
900    FOR QQ%=1 TO NP%: X(QQ%)=VF*X(QQ%): Y(QQ%)=VF*Y(QQ%): Z(QQ%)=VF*Z(QQ%): NEXT:
RETURN
2000    REM READ DATA
2005    POKE #75,32: CURSOR 0,5: PRINT "Reading in data";
2006    CURSOR 15,5: DPI=12.0: GOSUB 5200
2010    XC%=165: YC%=127: Q%=1
2020    READ NP%,NL%: PRINT 1.0
2030    DIM X(NP%),Y(NP%),Z(NP%)
2040    DIM XX(NP%),YY(NP%)
2050    DIM LA%(NL%),LB%(NL%),AAA%(100,0)
2060    FOR P%=1 TO NP%: READ X(P%),Y(P%),Z(P%)
2070    X(P%)=X(P%)*SF
2080    Y(P%)=(Y(P%)-20.0)*SF
2090    Z(P%)=Z(P%)*SF
2100    CURSOR 20,5: PRINT NP%-P%;: NEXT
2101    CURSOR 15,5: PRINT 0.0
2110    FOR L%=1 TO NL%: READ LA%(L%),LB%(L%)
2111    CURSOR 20,5: PRINT NL%-L%;: NEXT
2120    POKE #75,95: RETURN
4000    GOSUB 6000: MODE 6: FF=1.0: GOSUB 900: RETURN
5000    IF A%=47.0 AND AA%>0.0 THEN FL%=1: GOSUB 800: P210%=1: RETURN
5010    IF A%=47 AND AA%=0 THEN RETURN
5015    AA%=AA%+1
5020    IF A%=49 THEN AAA%(AA%)=50: GOSUB 800: RETURN
5030    IF A%=50 THEN AAA%(AA%)=49: GOSUB 800: RETURN
5040    IF A%=17 THEN AAA%(AA%)=16: GOSUB 800: RETURN
5050    IF A%=16 THEN AAA%(AA%)=17: GOSUB 800: RETURN
5060    IF A%=19 THEN AAA%(AA%)=18: GOSUB 800: RETURN
5065    IF A%=18 THEN AAA%(AA%)=19: GOSUB 800: RETURN
5070    IF A%=46 THEN AAA%(AA%)=44: GOSUB 800: RETURN
5080    IF A%=44 THEN AAA%(AA%)=46: GOSUB 800: RETURN
5081    IF A%=ASC("H") THEN MODE 0: GOSUB 6000: MODE 6: GOSUB 800: RETURN
5090    IF A%<>ASC("S") THEN RETURN
5091    DP=180.0/DPI:PRINT "Turning angle is now";: CURSOR 20,3: PRINT 180.0/DPI;"  (start
ing angle =";180.0/DPI;"  degrees)"
5092     PRINT "Larger angle : upper cursor key + repeat key": PRINT "Smaller angle : lower
cursor key + repeat key"
5093    A%=GETC: IF A%=0 THEN 5093
5094    IF A%=94 THEN MODE 6: A%=0: RETURN
5095    IF A%=16 THEN DP=DP+1.0: CURSOR 20,3: PRINT "          ";: CURSOR 20,3: PRINT DP;:
GOTO 5093
5096    IF A%=17.0 THEN DP=DP-1.0: CURSOR 20,3: PRINT "          ";: CURSOR 20,3: PRINT DP;:
GOTO 5093
5097    IF A%<>13.0 OR DP=0.0 THEN 5093: DPI=180.0/DP
5098    MODE 6: GOSUB 5200: GOSUB 800: A%=0: RETURN
5200    PHI=PI/DPI: KS=SIN(PHI): KC=COS(PHI): RETURN
6000    COLORT 5 5 5 5: MODE 0: PRINT CHR$(12);: IF A%=ASC("H") THEN CURSOR 0,20
```

```
6010   PRINT "============================================================"
6011   PRINT "This program draws an outline plan and allows it to be "
6020   PRINT "rotated, enlarged or reduced, with the help of the cursor  "
6030   PRINT "keys and with 1,2,<,> and /":PRINT
6040   PRINT CHR$(136);" : rotate to the left          +XZ"
6050   PRINT CHR$(137);" : Rotate to the right         -XZ"
6060   PRINT CHR$(94);" : Tilt backwards               +YZ"
6070   PRINT CHR$(140);" : Tilt forwards               -YZ"
6080   PRINT "1 : Turn clockwise              +XY"
6090   PRINT "2 : Turn anticlockwise          -XY"
6100   PRINT "> : Enlarge (don't use Shift)    +Y"
6110   PRINT "< : Reduce  (don't use Shift)    -Y"
6120   PRINT "/ : Step back to the original figure in reverse order"
6125   PRINT "H : HELP routine, displays the various commands"
6126   PRINT "S : Magnitude of turning angle. Input with cursor keys. RET."
6127   PRINT "C : Change; invert the colours"
6128   IF A%=ASC("H") THEN 6170
6129   PRINT "============================================================"
6130   PRINT "IN THE DATA SET : NP = Total number of nodes"
6140   PRINT SPC(18);"NL = Number of lines"
6150   PRINT SPC(18);"x1,y1,z1,x2,y2,z2,= Node cordinates"
6160   PRINT SPC(18);"b1,e1,b2,e2,= begin/end of line 1,2 etc"
6170   PRINT "============================================================"
6180   PRINT "The program begins when a key is pressed ";
6185   COLORT 5 15 0 0
6190   IF GETC=0 THEN 6190
6200   MODE 6: RETURN
7000   REM  STATEMENT
7001   COLORT 5 5 5 5: MODE 0: PRINT CHR$(12);: CURSOR 0,15
7002   PRINT "00000  000000      000000   0     00000   0000000"
7003   PRINT "      0  0    0      0    0  0    0   0      0   "
7004   PRINT "      0  0    0      0    0  0    0   0      0   "
7005   PRINT " 0000   0    0  000 00000   0    0   0      0   "
7006   PRINT "      0  0    0      0      0    0   0      0   "
7007   PRINT "      0  0    0      0      0    0   0      0   "
7008   PRINT "00000  000000      000     0000000 00000    000  "
7009   COLORT 5 15 0 0: RETURN
7010   DATA 43,49
7020   REM x1,y1,z1,x2,y2,z2 ... node coordinates
7030   DATA -60,0,-50,20,0,-50,-60,0,50,-60,40,-50
7040   DATA 20,0,-10,20,40,-50,60,0,50,-60,40,50
7050   DATA -20,70,-40,60,0,-10,20,40,-10,20,40,50
7060   DATA 60,30,50,-20,70,40,60,30,-10
7070   DATA -50,0,-50,-50,25,-50,-35,25,-50,-35,0,-50
7080   DATA -20,10,-50,-20,25,-50,10,25,-50,10,10,-50
7090   DATA 20,10,-40,20,25,-40,20,25,-20,20,10,-20
7100   DATA 30,0,-10,30,25,-10,50,25,-10,50,0,-10
7110   DATA 60,10,10,60,25,10,60,25,30,60,10,30,0,0,0
7111   DATA -46,3,-50,     -46,5,-50, -39,5,-50, -39,3,-50
7112   DATA -40,12,-52, -37,12,-52, -37,12,-50
7120   REM b1,e1,b2 ... beginnings and ends of the lines
7130   DATA 1,2,1,3,1,4,2,5,2,6,3,7,3,8,4,6
7140   DATA 4,8,4,9,5,10,5,11,6,9,6,12,7,10,7,13
```

```
7150    DATA 8,12,8,14,9,14,10,15,11,15,12,13,12,14,13,15
7160    DATA 16,17,17,18,18,19,20,21,21,22,22,23,23,20,24,25
7170    DATA 25,26,26,27,27,24,28,29,29,30,30,31,32,33
7180    DATA 33,34,34,35,35,32,36,37,37,38
7190    DATA 38,39,39,40,40,37,41,42,42,43
8000    REM NP,NL     Number of nodes/lines
8010    DATA 43,49
8020    REM x1,y1,z1,x2,y2,z2 ... node coordinates
8030    DATA -60,0,-50,20,0,-50,-60,0,50,-60,40,-50
8040    DATA 20,0,-10,20,40,-50,60,0,50,-60,40,50
8050    DATA -20,70,-40,60,0,-10,20,40,-10,20,40,50
8060    DATA 60,30,50,-20,70,40,60,30,-10
8070    DATA -50,0,-50,-50,25,-50,-35,25,-50,-35,0,-50
8080    DATA -20,10,-50,-20,25,-50,10,25,-50,10p10,-50
8090    DATA 20,10,-40,20,25,-40,20,25,-20,20,10,-20
8100    DATA 30,0,-10,30,25,-10,50,25,-10,50,0,-10
8110    DATA 60,10,10,60,25,10,60,25,30,60,10,30,0,0,0
8111    DATA -46,3,-50,   -46,5,-50, -39,5,-50, -39,3,-50
8112    DATA -40,12,-52, -37,12,-52, -37,12,-50
8120    REM b1,e1,b2 ... beginnings and ends of the lines
8130    DATA 1,2,1,3,1,4,2,5,2,6,3,7,3,8,4,6
8140    DATA 4,8,4,9,5,10,5,11,6,9,6,12,7,10,7,13
8150    DATA 8,12,8,14,9,14,10,15,11,15,12,13,12,14,13,15
8160    DATA 16,17,17,18,18,19,20,21,21,22,22,23,23,20,24,25
8170    DATA 25,26,26,27,27,24,28,29,29,30,30,31,32,33
8180    DATA 33,34,34,35,35,32,36,36,37,38
8190    DATA 38,39,39,40,40,37,41,42,42,43
9999    END
10000   REM  STATEMENT
10001   COLORT 5 5 5 5: MODE 0: PRINT CHR$(12);: CURSOR 0,15
10002   PRINT "00000  000000      000000    0     00000    0000000"
10003   PRINT "     0  0    0       0    0  0       0    0     0   "
10004   PRINT "     0  0    0       0    0  0       0    0     0   "
10005   PRINT " 0000   0    0  000  00000   0       0    0     0   "
10006   PRINT "     0  0    0       0      0       0    0     0   "
10007   PRINT "     0  0    0       0      0       0    0     0   "
10008   PRINT "00000   000000      000     0000000  00000   000   "
10009   COLORT 5 15 0 0: RETURN
```

SORTING DEMO.

(from DAInamic 16, page 162-163)

```
1       CLEAR 1000: DIM S$(10,0),A$(10,0)
10      REM SORTING DEMO
15      COLORT 8 0 8 8: RF$="N"
19      POKE #75,32
20      GOSUB 1500
30      PRINT "         # VISISORT #"
31      PRINT TAB(20);"Bert Maertens":FOR SX!=1,0 TO 37,0:A$=A$+CHR$(64):NEXT:PRINT A$
32      A$=""
40      PRINT :PRINT "    1) BUBBLE SORT":PRINT "    2) DELAYED SORTING"
50      PRINT "    3) SHELL-METZNER SORT":PRINT "    4) END PROGRAM"
80      PRINT :PRINT "Your choice :"
```

```
81    CH!=GETC: IF CH!=0 THEN 82
82    CH!=CH!-48
90    IF CH!<1 OR CH!>4 OR CH!<>INT(CH!) THEN 81
100   IF CH!=4.0 THEN PRINT CHR$(12): END
110   GOSUB 1500: GOSUB 160
120   Y!=22.0:FOR K=1 TO NS!:A$(K)=S$(K):CURSOR 2,Y!:PRINT A$(K):Y!=Y!-1.0:NEXT
130    NE=0: NC=0: CURSOR 20,22: PRINT "# ITEMS =";NS: CURSOR 20,21: PRINT "# COMPARE
=";NC: CURSOR 20,20: PRINT "# EXCHANGE =";NE
140   ON CH! GOSUB 310,400,540
150   CURSOR 0,12: PRINT "SORT LIST AGAIN <Y/N>"
151   RF!=GETC: IF RF!=0.0 THEN 151
152   RF$=CHR$(RF!): GOTO 20
160   IF LEFT$(RF$,1)="Y" THEN 250
170   NS$="10": PRINT "How many items to be sorted (MAX 9)";
171   NS!=GETC: IF NS!=0.0 THEN 171
172   NS=NS!-48.0: IF NS<2.0 OR NS>10.0 OR NS<>INT(NS) THEN 171
173   PRINT NS
180   F1$="C": PRINT "Provided by (U)ser or (C)omputer"
181   R1!=GETC: IF R1!=0.0 THEN 181
182   R1$=CHR$(R1!)
190   IF LEFT$(R1$,1)="U" THEN 240
200   R$="N": PRINT "(N)umbers or (L)etters"
201   R!=GETC: IF R!=0.0 THEN 201
202   R$=CHR$(R!)
203   IF R$="L" OR R$="N" THEN 210
204   GOTO 201
210   FOR K=1 TO NS
220   IF LEFT$(R$,1)="L" THEN S$(K)=CHR$(RND(26.0)+65.0)
222   IF LEFT$(R$,1)="N" THEN S$(K)=CHR$(RND(10.0)+48.0)
230   NEXT: GOTO 250
240    GOSUB 1500: PRINT "Maximum 1 character per item": FOR K=1 TO NS: PRINT "ITEM #";
K;" = ";
241   S!=GETC: IF S!=0.0 THEN 241
242   S$(K)=CHR$(S!): PRINT S$(K): NEXT
250   RF$="N": GOSUB 1500
260   RETURN
270   CURSOR 0,12: PRINT "Press a key to proceed"
271   IF GETC=0 THEN 271
290   CURSOR 0,12: PRINT SPC(31)
300   RETURN
310   CURSOR 9,23: PRINT "-BUBBLE SORT-": GOSUB 270
320   FOR I=1 TO NS-1
330   FOR J=I+1 TO NS
340   X=I:Y=J: GOSUB 710: IF A$(I)<=A$(J) THEN 360
350   GOSUB 750
360   NEXT J
370   NEXT I
380   CURSOR 0,13: PRINT "SORTING ENDED....":GOSUB 270
390   RETURN
400   CURSOR 3,23: PRINT "  -DELAYED SORTING-":GOSUB 270
410   J=0: R=0: I=0
420   I=I+1
430   IF I=NS THEN 520
```

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

DAI VIDEO HARDWARE - Sharper pictures in Black & White

(from DAInamic 16, page 164)

Working with the DAI in combination with a normal TV seems a very fatiguing task so I have been trying to achieve some improvement. Working with text is worst as 64 characters per line are too many for a normal TV and in addition the characters do not remain perfectly still. The letters flutter and appear ragged, particularly with some colour combinations.If the screen is filled with repetitions of the same letter a diagonal striped pattern can be seen; it gives the impression of a curtain of flowing water. It is possible to achieve economically a satisfactory picture with a black and white monitor and some simple modifications to the DAI printed circuit, but before I describe that I will report my unsuccessful attempts.

- With the colour TV: My Blaupunkt FM 121 has an IC, TDA 3300, which has extra RGB connections (pins 24, 25 & 26) and pin 23 which is presumably for switching, but I was unable to find anyone who knew how to make use of the facility.

- Improving the DAI video card: Using the article by Anton Doornenbal (DAInamic March/June 1982) I first changed the connections of the ZN 134 (interlace) which gave a small improvement but I decided that, not being an electronics man, I lacked the skill to accomplish the other changes recommended for bandwidth alterations.

- Use of a black and white monitor: I bought a second-hand Hong Kong set and sought advice from Anton Doornenbal who provided the monitor connections described below. Both DAI and monitor now work well together. It matters little whether or not the interlacing modifications have been done. The diagonal pattern would seem to be due to the colour pulse being switched in the receiver for colour recovery, but with a black and white monitor this pulse is superfluous.

BLACK & WHITE MONITOR CONNECTIONS

The diagram on page 165 shows the PAL video card with some of its main components.

1 Simplest connection:

Fit the socket for the monitor lead at the position shown shaded on the diagram. Connect the outer sleeve of the socket to the modulator casing (=GND). Connect the the centre conductor of the socket to the VIDEO signal pin. The offending colour pulse is killed by short-circuiting the 4,4336 MHz crystal with a small switch (SW). Make a small hole in the cover of the DAI so that the switch can be operated by a pencil through the hole, thus allowing a colour TV or a monitor to be used.

2 Deluxe connection:

With this method the crystal is not impeded. The monitor socket is connected by a short screened cable to a transistor stage (see sketches on page 166). This transistor, a BFY 55 or 2N2219, and the two resistors with it are mounted on the solder side, approximately underneath Tr1. The cable screen is soldered to an earth point as near as possible to the transistor. We now have a tapping from the video signal, raised to the required level of 1 Volt.

Those wishing to read more about the working of a similar video card, using the LM 1889, can refer to an article in ELEKTUUR No 231 of Jan 1983. [Note: Elektuur is a Dutch language version of the magazine which is on sale in the UK under the name ELEKTOR]

Geert Hospers (with thanks to A. Doornenbal)

NEGATIVE CURSOR

(from DAInamic 16, page 178)

```
100   REM ******** NEGATIVE CURSOR *********
110   REM ****** Witten by Corneel de Bont ******
140   GOTO 400
200   REM *** OUTSIDE THE MARGIN SUBROUTINE
210   IC=#BFEE-((23-YP)*#86): POKE IC,208+ZP
215   IF YP>0 THEN POKE IC-#86,208
220   FOR IX=0 TO LEN(G$)-1: T$=MID$(G$,IX,1): GX=ASC(T$)
230   IP=#BFE7-((23-YP)*#86)-(XP*2): POKE IP,GX
240   XP=XP+1: NEXT
250   RETURN
300   REM *** GETC ROUTINE
310   CURSOR 0,0: PRINT SPC(58);: POKE #75,95
320   CURSOR 10,0: PRINT "PRESS A KEY TO CONTINUE";
330   CC=INT(15*RND(1)): IF CC=8 THEN 330
340   POKE #B3E4,CC+208: SOUND 1 0 15 0 FREQ((CC*50)+50)
350   WAIT TIME 3: SOUND OFF: G=GETC: G=GETC: G=GETC
360   G=GETC: IF G=0 THEN WAIT TIME 3: I=I+1: IF I<30 THEN 360
370   I=0: IF G=0 THEN 330
380   POKE #75,32: RETURN
400   REM *** TITLE
410   MODE 0: COLORT 8 0 8 8: PRINT CHR$(12);: POKE #75,32
420   CURSOR 55,7: ZP=-1: FOR I=0 TO 23: XP=-3+I: YP=23-I
430   G$="DEMONSTRATION OF NEGATIVE CURSOR"
440   ZP=ZP+1: IF ZP=16 THEN ZP=0
450   IF ZP=8 THEN ZP=ZP+1
460   GOSUB 200: NEXT: PRINT
470   GOSUB 300: RESTORE
500   PRINT CHR$(12);: COLORT 8 0 8 8: CURSOR 55,7
510   FOR YP=23 TO 1 STEP -1: READ G$: ZP=0
520   XP=INT(3*RND(1)): IF RND(10)>5 THEN XP=XP*(-1)
530   GOSUB 200: NEXT
540   GOSUB 300: GOTO 400
600   REM
610   DATA "DEMONSTRATION OF A NEGATIVE CURSOR"
620   DATA "************************************"
630   DATA "THIS PROGRAM DEMONSTRATES A SUB-    "
640   DATA "ROUTINE IN WHICH A NEGATIVE CURSOR  "
650   DATA "WILL BE USED. A PARTICULAR STRING   "
660   DATA "IS TO BE PUT ON THE SCREEN AT THE   "
670   DATA "REQUIRED PLACE, JUST AS ONE DOES    "
680   DATA "WITH A CURSOR STATEMENT, THE BIG    "
690   DATA "DIFFERENCE IS THAT X-VALUES OF TO   "
700   DATA "-3 CAN BE USED. THERE IS ALSO THE   "
710   DATA "POSSIBILITY OF COLOURING THE LINES  "
720   DATA "BY CALLING THE SUBROUTINE AND       "
730   DATA "GIVING VARIABLE ZG A VALUE BETWEEN  "
740   DATA "0 AND 15. (0=BLACK).                "
750   DATA "THIS SUBROUTINE CAN BE CALLED IN THE "
```

```
760   DATA "FOLLOWING MANNER:- FIRST DEFINE:        "
770   DATA "XG: (X AXIS) A NUMBER BETWEEN -3 AND 58"
780   DATA "YG: (Y AXIS) A NUMBER BETWEEN 0 AND 23  "
790   DATA "ZG: (COLOUR) A NUMBER BETWEEN 0 AND 15 "
800   DATA "G$: THE STRING (MAXIMUM 62 LETTERS)     "
810   DATA "AND THEN GIVE A GOSUB 200               "
820   DATA "****************************************"
830   DATA "(XG=-3:YG=9: ZG=7:G$='DEMO': GOSUB 200)  "
```

Power Supply & Overvoltage Protection

(from DAInamic 16, page 207)

This circuit (see diagrams on page 207) and the fuse F1 prevent damage to the computer from the unregulated 22 volt DC in the event of a power supply fault. It will also protect the, until now unfused, power supply itself from overload.

Should the supply voltage exceed its normal value by more than 0.9V the thyristor strikes and the fuse will blow. Construction can be on a small Vero board which will fit in front of the electrolytic capacitor C55 inside the power supply enclosure. The leads for the frame earth and the 22 volt input are connected to the main circuit board via two small drilled holes. On the component side of the main board runs a wide conductive strip and from this conductor the +5V can be lead to the protection circuit. At the same time these connecting leads hold the protection circuit board in place. The fuse is fitted next to the rectifier block BR1 by drilling four small holes for its holder. The U+ conductor strip from the rectifier BR1 is cut and the fuse connected between the cut ends.

R. Corswandt.

SOCIALE GEOGRAFIE

ALGEMEEN OPZET VAN DE VERZAMELTAPE "SOCIALE GEOGRAFIE"

De verzameltape bevat één kant met demonstratie-programma's en één met interactieve programma's. Naargelang de mogelijke didactische werkvormen kunnen volgende types onderscheiden worden:

- demo's met een inleidend karakter (vb;sfeerscheppend,...)
- consultatieprogramma's die gekoppeld zijn aan een demo of/en simulatieprogramma
- oefenprogramma's, meestal onder spelvorm.

Hoewel alle programma's klassikaal gebruikt kunnen worden, zijn de oefenprogramma's specifiek voor individueel gebruik bedoeld. De meeste programma's bevatten een summiere herhaling van de theorie, evenals de noodzakelijke uitleg voor het begrijpen ervan en voor hun gebruik.

De behandelde leerinhouden zijn eerder sporadisch genomen uit het geheel van de leerstof Sociale Geografie. De volgorde van de programma's op de tape is evenmin gestructureerd. Alle programma's moeten dus op zichzelf staande gezien worden, ook al zijn sommige verwant met elkaar. Ze geven ook geen banden met een of ander leerplan. De belangrijkste motivatie voor hun keuze berust bijhun didactische gebruiksmogelijkheden.

Inhoudelijk ligt het accent op de bevolkingsgeografie (groei,nataliteit,mortaliteit, groeimodellen,...), sociale aspecten die hiermee verwant zijn, spreidingsproblemen en enkele speciefieke problemen (tijdzones, territoriale wateren,..). Verschillende grafische voorstellingswijzen evenals wiskundige modellen en statistische technieken worden aanschouwelijk toegepast en gedemonstreerd.

Het niveau is vooral gericht op de 3de graad van het Secundair Onderwijs (determinatie) Een aantal oefenprogramma's in spelvorm worden daarentegen reeds succesvol gebruikt vanaf de leeftijd van de 1ste graad SO (observatie).

UURGORDELS

AUTEUR : Marian De Jaeger-1983
VERSIE : oefenprogramma onder spelvorm

DOEL

Inoefenen van de begrippen plaatselijke tijd, uurgordels en datumlijn.
Gebruik van de uurgordels om de plaatselijke tijd te bepalen.

BESCHRIJVING

Na een introductie begint het programma met een korte herhaling van de theorie. Vervolgens worden de instructies gegeven voor het gebruik en spelen. Dan verschijnt de wereldkaart met de uurgordels en de datumlijn. De uurgordels zijn begrensd door rechte lijnen en hebben dus alleen betrekking op de plaatselijke zonnetijd en niet op de burgerlijke tijd. Door toeval bepaalt de computer een plaats van vertrek (aangeduid door V) en een plaats van aankomst (A op de kaart), evenals een vertrekdag, -uur en vluchtduur. Men dient dan de dag en het uur (plaatselijke tijd) van aankomst op te geven. Bij een foutief antwoord krijgt men enkele nieuwe kansen. Bij een juist antwoord krijgt men een nieuwe opgave tot wanneer het vooraf opgegeven oefeningen is bereikt. Tot slot geeft het programma een evalutie van het resultaat.

SPREIDING VAN PUNT-OBJECTEN

AUTEUR: Marc Antrop - 1983
VERSIE: demo

DOEL

De werkwijze voor het toepassen van een CHI-kwadraat-toets stap voor stap demonstreren.

BESCHRIJVING

De pc genereert eerst een (toevallig) 2-dimensioneel puntenpatroon.De probleemstelling is of die ruimtelijke spreiding nu toevallig is of niet.Hiertoe wordt een CHI-kwadraat toets gebruikt.De verschillende berekeningsstappen worden getoond, evenals de te gebruiken formules.Na de berekening wordt een overzicht gegeven van de resultaten en van de hypothesen.De besluiten dienen getrokken worden na consultatie van de statische tabellen.

LORENZ-CURVE

AUTEUR: Marc Antrop - 1983
VERSIE: demo

DOEL

Demonstreert de constructie en het gebruik van een Lorenz-curve

BESCHRIJVING

Het programma geeft een menu voor een keuze van de demo-versie of de invoer van eigen gegevens.
Deze versie springt steeds naar de demo die gebruikt maakt van de inkomens- en bevolkingsverdeling tussen de rassen in Zuid Afrika:
De probleemstelling is die van een gelijke verdeling van de inkomens tussen de rassen. In tabel worden de cijfergevens getoond evenals de noodzakelijke bewerking ervan om de Lorenz-curve te construeren. Die grafiek wordt dan getoond. Het behelst eerst de cijfers van 1967. Dan wordt een nieuwe probleemstelling aangebracht, nl. of het vroeger slechter was dan nu. In een nieuwe tabel worden de cijfers van 1967 en 1936 gegeven. Hierna worden de 2 Lorenz-curven op elkaar getekend.

NATALITEIT-MORTALITEIT

AUTEUR: Marc Antrop - 1983
VERSIE: demo + consultatie

DOEL

Cijfergegevens voorstellen in een 2 dimensioneel strooiingsdiagram.
Dit toepassen op de bevolkingscijfers NATALTEIT en MORTLITEIT per land.

BESCHRIJVING

Het menu geeft 3 mogelijkheden:
(1) een demo met de opbouw van het strooiingsdiagram : hiervoor worden een 20-tal landen voorgesteld, geselecteerd uit de verschillende hoofdtypes;
(2) een voorstelling van alle landen; de verschillende wereldelen of types worden door verschillende kleuren voorgesteld;
(3) een voorstelling van landen naar keuze.
De landen worden eerst (alfabetisch en genummerd) in tabel voorgesteld: naam, geboorte- en sterftecijfer. De keuze wordt gemaakt door het volgnummer in te voeren: typ nummer <RETURN>. De computer controleert dan of hij dat land in file heeft en of er cijfergegevens beschikbaar zijn. Een foutmelding volgt eventueel met en nieuwe keuzemogelijkheid. Ook wordt gecontroleerd of geen tweemaal hetzelfde land gekozen wordt.

GROEIMODELLEN

AUTEUR: Marc Antrop - 1983
VERSIE: demo met de bevolking van Engeland en Wales

DOEL

Vergelijking van verschillende groeimodellen met de werkelijke bevolkingsgroei.

BESCHRIJVING

Het programma begint met een theoretisch overzicht van 3 groeimodellen : het model van Euler, het exponentieel groeimodel en het logistisch groeimodel. De parameters en de gebruikte symbolen worden toegelicht. Die modellen worden dan toegepast op de bevolkingsevolutie van Engeland en Wales, waarvoor er vanaf 1751 gegevens voorhanden zijn. In de grafische voorstelling wordt de beginpopulatie aan 100 gelijk gesteld. Tenslotte wordt de werkelijke bevolkingsgroei getekend en kunnen de verschillende groeimodellen geevalueerd worden.

BEVOLKINGSBOM

AUTEUR: Marc Antrop - 1983
VERSIE: demo

DOEL

Figuratief voorstellen van de groei van de wereldbevolking in de geindustrialiseerde en ontwikkellingslanden, met als open vraag ... de toekomst.

BESCHRIJVING

Voorstelling van de evolutie van de bevolking in de 2 grote wereldblokken sinds de Industriele Revolutie tot het jaar 2100. UN-schattingen. De voorstelling gebeurt in een dubbele staafdiagram. Na dat de bevolkingsaandelen voorgesteld zijn, worden ook de aandelen van de werkende en niet-werkende bevolking gegevens. Het programma eindigt met ????

HOOFDPLAATSEN

AUTEUR: Marc Antrop - 1982
VERSIE: oefenprogamma (drill-and-practice)

DOEL

Inoefenen en trainen van feitenkennis met betrekking tot administratief-bestuurlijke indelingen en hun hoofdplaatsen of zetels.

BESCHRIJVING

Het menu biedt achtereenvolgens de keuze uit de moeilijkheidsgraad, het aantal gewenste oefeningen, het aantal herkansingen bij een foutief antwoord, de richting van de vraagstelling, en de snelheid. Met een aantal hulptoetsen wordt het gehele programma gestuurd. Na start <S> wordt het werkblad voorgesteld. Hierop worden nog eens de menuselecties getoond. Naargelang de gekozen optie wordt een bestuurlijk gebied of hoofdplaats/zetel op gegeven. Een lopende cursor laat toe het juiste antwoord te selecteren (door op SPATIE te drukken) uit een opgegeven lijst. Foutieve antwoorden worden meteen gemeld en een nieuwe keuze wordt aangeboden. Wanneer alle kansen verbruikt zijn, geeft de computer zelf het juiste antwoord en vraagt dit te herhalen. Wanneer het aantal gevraagde oefeningen beantwoord werden, wordt een evaluatie gegeven : score + foutenanalyse. Een tussenevaluatie kan gevraagd worden met de hulptoets <E>, de oefening wordt verder gezet met hulptoets <V>.

OPSPORINGSSPEL

AUTEUR : Marian De Jaeger-1982
VERSIE : oefenprogramma onder spelvorm

DOEL

Inoefenen van de geografische coordinaten NB, ZB, OL, WL
onder spelvorm.

BESCHRIJVING

Het programma begint met een herhaling van de theoretische begrippen van de geografische plaatsbepaling op aarde: de halfronden en de afkortingen van de coordinaten. Dan volgt de uitleg voor het spel. Na het intypen van het eerste coordinatenpaar verschijnt de wereldkaart met de evenaar en de nulmeridiaan. De ingevoerde positie van de speler wordt aangeduid met een vierkantje. De computer berekent nu het verschil in lengte- en breedteligging tussen die positie en de onbekende, toevallig bepaalde, positie van de 'chef' die moet gevonden worden. Die resultaten verschijnen onderaan het beeld. Wanneer de opgegeven positie niet juist is, worden nieuwe coordinaten gevraagd. Men krijgt 3 kansen.

BRUTO NATIONAAL PRODUKT v2

AUTEUR: Marc Antrop - 1983
VERSIE: consultatie

DOEL

Vergelijken van het Bruto Nationaal Produkt per inwoner van een aantal geselecteerde landen.

BESCHRIJVING

Het programma bevat het BNP/inw. van ca.120 landen in data-arrays. De namen van de landen evenals het BNP/inw. wordt geladen van data-files die onmiddelijk na het hoofdprogramma op tape gesaved zijn.
De landen worden in tabelvorm voorgesteld en de gebruiker kan zijn keuze maken. Maximaal 20 landen kunnen geselecteerd worden. Na de keuze wordt nog een overzicht van de keuze in tabelvorm gegeven.
Die gegevens worden weergegeven in BF en voorgesteld in staafdiagrammen.
De kleurkeuze hiervan gebeurt random

BEVOLKINGSTELLER v1

AUTEUR : Marc Antrop - 1983
VERSIE : demo

DOEL

Concreet voorstellen van de aangroeisnelheid van de wereldbevolking

BESCHRIJVING

Het programma maakt gebruikt van de UN-schattngen van de gemiddelde geboorte- en sterftecijfers van de wereldbevolking. Het aantal geboorten, sterften en de natuurlijke aangroei per seconde wordt hieruit berekend. De computer toont die groei per seconde vanaf de start van het programma. Na 60 sec wordt de teller onderbroken en worden alle resultaten nog eens in tabelvorm gegeven.

TERRITORIALE WATEREN

AUTEUR: Marc Antrop - 1982 , naar een idee van Eric De Jaeger
en een uitwerking voor Apple door F.Goethals.
VERSIE: demo

DOEL

Invloed nagaan van de kustlijnvorm op de begrenzing van territoriale wateren m.b.v.d mediaanlijn, volgens de methode van Boggs.

BESCHRIJVING

Na de introductie geeft het programma de theoretische grondslagen en de probleemstelling.
In 5 fazen wordt de grenszee tussen de verschillende landen verdeeld. De kustlijn van het onderste land blijft constant, de kustlijn van het bovenliggend continent verloopt volgens een sinusoide, waarvan het amplitude met iedere fase toeneemt. De grenslijn wordt bepaald door de mediaanlijn en de oppervlakteverhoudingen van de verschillende zee-territoria wordt eveneens telkens gegeven. Als besluit worden de resultaten nog eens in tabelvorm gesynthetiseerd.
Dit programma berekent geen mediaanlijn omdat dit te tijdrovend is. De resultaten van deze berekeningen worden echter in deze demo-versie gebezigd.

DAI-gebruiker ... ?

Leraar in het secundair of hoger onderwijs ... ?

Voldoende ervaring met de BASIC-programmeertaal .... ?

Nog geen lid van de werkgroep diDAIsoft...., dan mis je regelmatige contacten met collega's die jij wat te bieden hebt, maar waarvan jij vast en zeker ook wat kan opsteken.

Geïnteresseerd... neem conctact met Bruno Van Rompaey (016/461085)

3.3. Behandeling van een instructie door de microprocessor 8080

3.3.1. Machinecycli in een instructie

Een instructie wordt samengesteld uit minimum 1, maximum 5 machinecycli (Mn) uit de volgende tien.

- De FETCH of haalcyclus. Hierbij wordt een instructie gelezen of opgehaald uit het programmageheugen en in de centrale verwerkingseenheid gebracht voor behandeling. Het is evident dat alle instructies beginnen met een haal cyclus, hij krijgt derhalve de naam van machinecyclus nr 1 (M1).
- De MEMORY READ cyclus is de cyclus die doorlopen wordt als de instructie gebruikt wordt voor een lezing uit het geheugen.
- De MEMORY WRITE cyclcus is de cyclus die doorlopen wordt als er dient geschreven te worden in het geheugen.
- De STACK READ cyclus is eveneens een cyclus die leest uit het geheugen maar wel in een speciaal gedeelte van het geheugen, nl. het stapelgeheugen.
- De STACK WRITE gaat gepaard met een schrijven van gegevens in het stapelgeheugen.
- De INPUT cyclus betreft het lezen van informatie aangebracht door een periferie apparaat aan de ingangspoort van de microprocessor.
- de OUTPUT cyclus betreft een schrijven in de uitgangspoort van een microprocessor om informatie over te brengen naar een periferie apparaat.
- De INTERRUPT cyclus is de cyclus die de onderbrekingsaanvraag van bijvoorbeeld een periferie apparaat behandelt.
- De HALT cyclus doet de microprocessor wachten op een traag periferie apparaat of een traag geheugen of op een sein dat een andere processor geeft als deze zijn werkzaamheden beeindigd heeft (*multiprocessing*). Onder traag geheugen of traag periferie apparaat verstaan we die elementen die een datawoord kunnen verwerken of aanbieden in één periode van de klok.
  Een andere processor is bijvoorbeeld een schakeling die direct toegang heeft tot het geheugen (DMA : *direct memory access*) om met grote snelheid informatie in of uit het geheugen te lezen.
- De HALT+INTERRUPT cyclus stopt de microprocessor met het oog op het behandelen van een interrupt aanvraag.

Merken we op dat het begin van elk van deze machinecycli gesynchroniseerd wordt door het signaal SYNC dat de microprocessor uitgeeft.
Door middel van een 8 bits statuswoord op de databus geeft de microprocessor te kennen in welke machinecyclus hij zich bevindt. Deze statusinformatie is beschikbaar in het begin van elke machinecyclus tijdens de SYNC impuls.
In fig. 3.6 zien we dat gedurende de eerste klokperiode tijdens $\emptyset_2$xSYNC de status beschikbaar is op de databus en gedurende de tweede klokperiode tijdens $\emptyset_1$ SYNC kan vergrendeld worden en gedecodeerd in een afzonderlijke bouwsteen.
Fig. 3.7 geeft een overzicht van de 10 statussignalen overeenkomstig de 10 machinecycli met aanduiding van de 5 controlesignalen die hieruit kunnen verkregen worden door decodering. Zo bijvoorbeeld is de status bij een machinecyclus 2; d.i. een leescyclus uit het geheugen, gelijk aan 10000010 of 82H. Dit kan gedecodeerd worden in een MEMR controlesignaal.

De controlesignalen zijn aktief laag, waardoor een direkte verbinding mogelijk is met ROM, RAM en I/O bouwstenen.

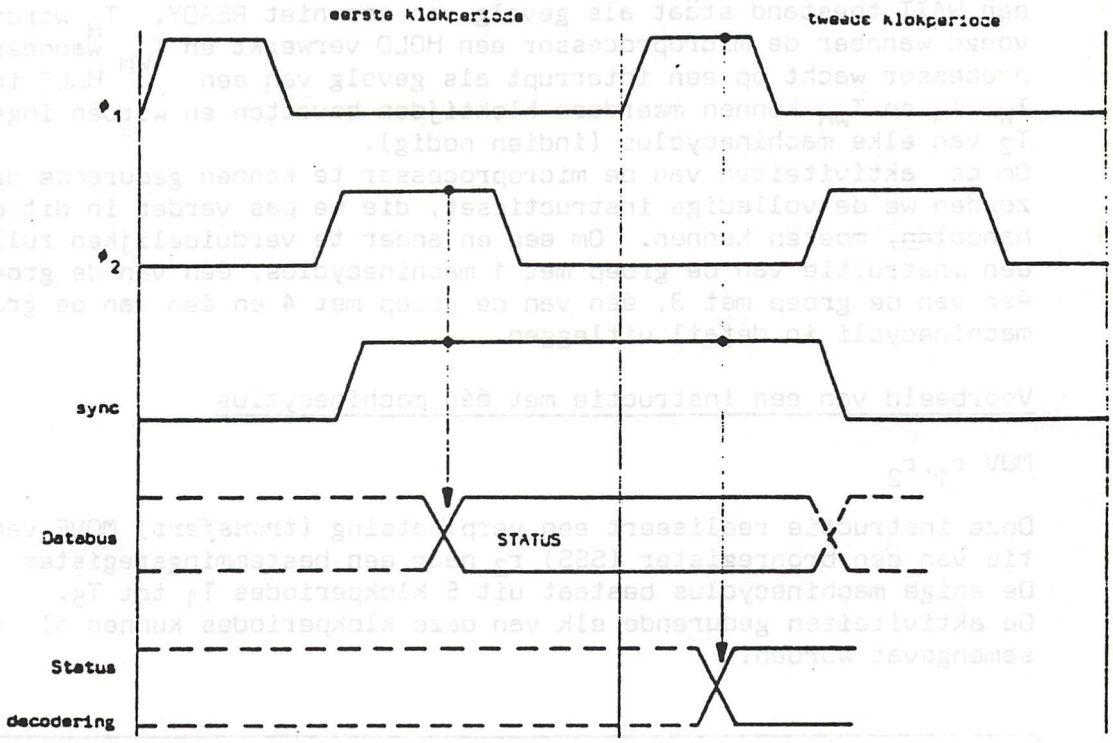


Fig. 3.6

Type of machine cycle

Data bus bit	Status information	Instruction fetch	Memory read	Memory write	Stack read	Stack write	Input read	Output write	Interrupt acknowledge	Halt acknowledge	Interrupt acknowledge while halt	
		①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	Ⓝ Status word
D_0	INTA	0	0	0	0	0	0	0	1	0	1	INTA
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1	(NONE)
D_2	STACK	0	0	0	1	1	0	0	0	0	0	INTA
D_3	HLTA	0	0	0	0	0	0	0	0	1	1	I/O W
D_4	OUT	0	0	0	0	0	0	1	0	0	0	I/O R
D_5	M_1	1	0	0	0	0	0	0	1	0	1	MEM W
D_6	INP	0	0	0	0	0	1	0	0	0	0	MEM R
D_7	MEMR	1	1	0	1	0	0	0	0	1	0	MEM W
												MEM R

Control signals

Fig. 3.7

3.3.2. Onderverdeling van de machine cycli

Een machinecyclus wordt verdeeld in minimum 3 en maximum 5 periodes van de besturingsklok, naargelang de instructie waar hij deel van uitmaakt (T_n). Hieraan kan T_W worden toegevoegd wanneer de microprocessor in een WAIT toestand staat als gevolg van een niet READY. T_H wordt toegevoegd wanneer de microprocessor een HOLD verwerkt en T_{WH} wanneer de microprocessor wacht op een interrupt als gevolg van een HALT instructie. T_W, T_H en T_{WH} kunnen meerdere kloktijden bevatten en worden ingelast na T_2 van elke machinecyclus (indien nodig).
Om de aktiviteiten van de microprocessor te kennen gedurende deze cycli, zouden we de volledige instructieset, die we pas verder in dit deel behandelen, moeten kennen. Om een en ander te verduidelijken zullen we een instructie van de groep met 1 machinecyclus, één van de groep met 2, één van de groep met 3, één van de groep met 4 en één van de groep met 5 machinecycli in detail uitleggen.

Voorbeeld van een instructie met één machinecyclus

MOV r_1, r_2

Deze instructie realiseert een verplaatsing (*transfert*) MOVE van informatie van een bronregister (SSS) r_2 naar een bestemmingsregister (DDD) r_1.
De enige machinecyclus bestaat uit 5 klokperiodes T_1 tot T_5.
De aktiviteiten gedurende elk van deze klokperiodes kunnen als volgt samengevat worden.

M_1				
PC out, Status	PC=PC+1	INST → TMP/IR	(SSS) → IMP	(TMP) → DDD
T_1	T_2	T_3	T_4	T_5

De betekenis van de in deze samenvatting gebruikte afkortingen zijn :

T_1 Gedurende deze klokperiode wordt het adres aanwezig in de programmateller op de adresbus gezet (PC OUT) via de adresbuffer. Tegelijkertijd wordt de STATUS uitgegeven op de databus. Door decodering van deze statussignalen (tijdens T_2) zal het mogelijk zijn de signalen van de controlebus te vormen die nodig zijn om de doorverbindingen tussen de verschillende schakelingen te maken, zodat de betreffende machinecyclus naar behoren kan uitgevoerd worden. De status van een eerste machinecyclus (M_1) van een instructie is ten allen tijde een FETCH, d.w.z. een cyclus die de instructies zal lezen uit het programmageheugen en ter behandeling inbrengen in de microprocessor.

T_2 Gedurende deze periode wordt de programmateller losgekoppeld van de adresbuffer en vervolgens met één verhoogd ($PC=PC_1$). Hierdoor staat de teller klaar om de volgende instructie of de volgende byte van een meerbyte instructie te lezen. Tevens wordt de status gedecodeerd om de signalen van de controlebus te vormen. Om deze decodering mogelijk te maken, dient er een decoder geschakeld te worden op de databus. Deze decoder bestaat onder de vorm van een speciale schakeling (*system controller/bus driver* 8228).

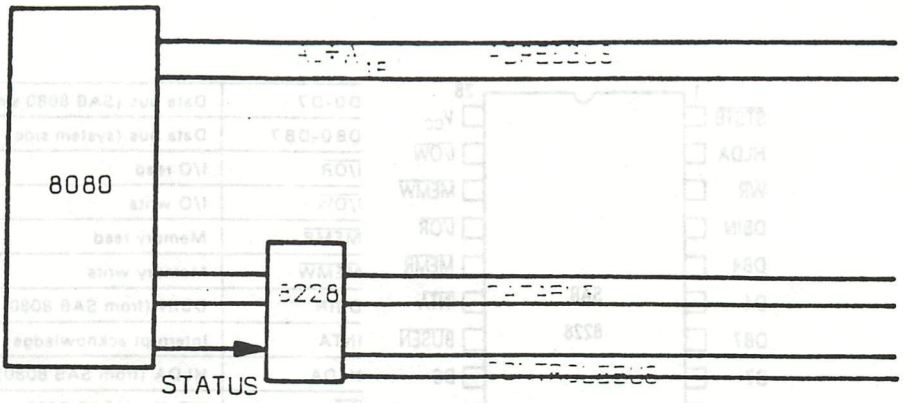


fig. 3.8

Zoals uit fig. 3.8 blijkt, ontvangt deze schakeling de databusuitgangen van de 8080 die gemultiplexeerd zijn tussen gegevens en status. Met behulp van een signaal STSTB (*status strobe*), afgeleid van $\emptyset_1$ SYNC kan de status in de 8228 vergrendeld worden en verder gedecodeerd om de 5 controlesignalen te vormen.
Een functioneel schema van de 8228 met aansluitgegevens geeft fig. 3.9a en fig. 3.9b.
Het STSTB signaal wordt opgewekt door een afzonderlijke bouwsteen *clock driver* (8224) en wordt laag gedurende $\emptyset_1$ van elke T_2.
De controlesignalen MEMR, I/OR en INTA worden gegenereerd, gebruik makend van de desbetreffende statusbits aanwezig in de vergrendelaar (*status latch*) en het DBIN signaal afkomstig van de processor.
De controlesignalen MEMW, I/OW maken gebruik van het WR signaal.
De 8228 bevat eveneens een bidirectionele *busdriver* zodat een groot aantal geheugens en I/O bouwstenen aangeschakeld kunnen worden (typisch 15mA i.p.v. 1,9mA). Deze *busdriver* wordt gestuurd door de *status decoder* voor input of output bewerkingen (DBIN,WR) voor isolatie van de systeem databus bij het verwerken van de status, alsook bij een HOLD aanvraag (*databus 3-state*).
Het BUSEN ingangssignaal (indien hoog) zet de ganse bouwsteen in 3-state.
Het INTA signaal kan gebruikt worden om een input poort te selecteren bij een interruptverwerking. Hierdoor kan een 8-bitswoord ingelezen worden dat als voortzettingsadres geïnterpreteerd wordt door de microprocessor.

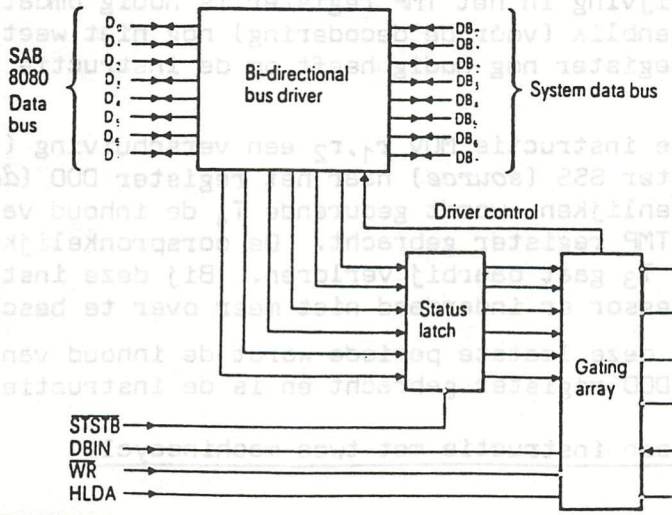
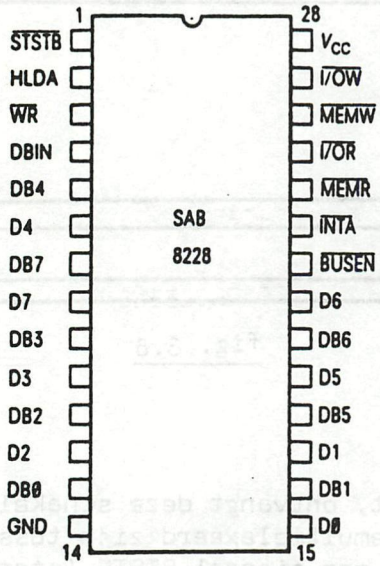


fig. 3.9a



D0-D7	Data bus (SAB 8080 side)
DB0-DB7	Data bus (system side)
$\overline{I/OR}$	I/O read
$\overline{I/OW}$	I/O write
$\overline{MEMR}$	Memory read
$\overline{MEMW}$	Memory write
DBIN	DBIN (from SAB 8080)
INTA	Interrupt acknowledge
HLDA	HLDA (from SAB 8080)
$\overline{WR}$	WR (from SAB 8080)
$\overline{BUS\ EN}$	Bus enable input
$\overline{STSTB}$	Status strobe (from SAB 8224)
V_{cc}	+5 V supply voltage
GND	Ground (0)

fig. 3.9b

Bij de 8080 is de algemene vorm van dit voortzettingsadres 11...111 zodat 8 mogelijkheden voorhanden zijn, RST (*restart*) 0 tot 7 genaamd met respectievelijk adressen 0, 8, 10, 18, 20, 28, 30, 38H (vectoren). Indien slechts één vector nodig is, volstaat het de $\overline{INTA}$ uitgang via een 1KΩ weerstand te verbinden met +12V. Hierbij wordt automatisch een RST7 instructie gegenereerd op de databus bij een DBIN=1 op het ogenblik van de interrupt aanvraag.
Tijdens T_2 wordt nagegaan of er een READY is, een HOLD of een HALT.
Na T_2 volgt : T_W : indien READY=0
T_H : indien een HOLD aangevraagd werd
T_{WH}: indien een HALT instructie gegeven werd.
Zoniet volgt :

T_3 : Gedurende de periode T_3 wordt de instructie ingelezen en ingeschreven in het instructieregister (IR) en tegelijkertijd in het voorlopig register (IMP : *temporary* = voorlopig) dat toegang verleent tot de ALU.
Inschrijving in het instructieregister (IR) is nodig om de instructie te kunnen doorgeven naar de instructiedecoder die de instructie zal decoderen en zal bepalen welke de verdere operaties zijn.
De inschrijving in het TMP register is nodig omdat de microprocessor op dit ogenblik (vóór de decodering) nog niet weet of hij de inhoud van dit register nog nodig heeft om de instructie verder te kunnen afwerken.

T_4 : Vermits de instructie MOV r_1,r_2 een verschuiving (*transfert*) van het register SSS (*source*) naar het register DDD (*destination*) dient te verwezenlijken, wordt gedurende T_4 de inhoud van het bronregister naar het TMP register gebracht. De oorspronkelijke inhoud van TMP gedurende T_3 gaat daarbij verloren. Bij deze instructie dient de microprocessor er inderdaad niet meer over te beschikken.

T_5 : Gedurende deze laatste periode wordt de inhoud van het TMP register naar het DDD register gebracht en is de instructie afgewerkt.

Voorbeeld van een instructie met twee machinecycli

MOV r,M

Deze instructie realiseert het transfert van de inhoud van een geheugencel M, waarvan het adres zich in het HL register bevindt, naar een register r.
De opdracht bestaat uit 2 machinecycli, onderverdeeld in 4, respectievelijk 3 periodes. De eerste noemen we de haalcyclus (*fetch*) en de tweede de uitvoeringscyclus (*execute*).

M_1			
PC OUT, STATUS	PC=PC+1	INST→IMP/IR	
T_1	T_2	T_3	T_4

M_2		
HL OUT, STATUS	DATA	DDD
T_1	T_2	T_3

De machinecyclus M_1 verloopt zoals bij de vorige instructie voor wat betreft de periodes T_1 T_2 T_3.
De periode T_4 is een periode die wel bestaat maar alleen inwendig door de microprocessor gebruikt wordt voor de decodering van de instructie bijvoorbeeld en de bepaling van het aantal bytes. Uitwendig is hiervan niets waarneembaar.
Gedurende machinecyclus M_2 wordt de in T_4 van voorgaande cyclus bepaalde status uitgegeven naar de decoder van de controller 8228 (*memory read*).
De programmateller blijft ongewijzigd.
Het adres van de geheugencel M dat zich in het HL registerpaar bevindt, wordt op de adresbus gezet.
T_2 en T_3 van M_2 vormen de geheugen subcycli; gedurende T_2 wordt de geheugencel M gelezen en tijdens T_3 wordt de inhoud naar het bestemmingsregister DDD gebracht.

Voorbeeld van een instructie met drie machinecycli

LXI rp,data

Deze instructie laat toe een 16 bits datawoord of een 16 bits adres in te schrijven in een registerpaar. Deze instructie kan gebruikt worden om een adres bijvoorbeeld in het registerpaar HL in te schrijven.
De machinecyclus M_1 verloopt volledig identiek aan die van voorbeeld 2.
Gedurende de machinecycli M_2 en M_3 worden uit het programmageheugen 2 bijkomende bytes B_2 en B_3 gelezen die respectievelijk de 8 minst beduidende bits en de 8 meest beduidende bits uitmaken van de informatie die dient ingeschreven te worden in het registerpaar. De minst beduidende byte van een informatie moet in r_e geschreven worden en de meest beduidende byte in r_h.

M_2		
PC OUT, STATUS	PC=PC+1	$B_2 \rightarrow r_1$
T_1	T_2	T_3

M_3		
PC OUT, STATUS	PC=PC+1	$B_3 \rightarrow r_h$
T_1	T_2	T_3

Voorbeeld van een instructie met vier machinecycli

LDA adr

Deze instructie laadt de accumulator met het datawoord dat zich op het adres adr bevindt.
Machinecyclus M_1 is weer dezelfde als hoger, evenzo M_2 en M_3 met dat verschil dat de bytes B_2 en B_3 die het adres vormen, nu ingeschreven worden in de interne registers Z en W die alleen ter beschikking zijn van de microprocessor en die niet toegankelijk zijn voor de programmator.
De vierde machinecyclus wordt :

M_4		
WZ OUT, STATUS	DATA →	A
T_1	T_2	T_3

Voorbeeld van een instructie met vijf machinecycli

LHLD adr

Deze instructie laadt het registerpaar HL met de inhoud die zich bevindt op de plaatsen aangegeven door adr en adr+1.
Opnieuw is M_1 zoals voor alle instructies identiek. Gedurende M_2 en M_3 worden de bytes B_2 en B_3 geladen in de inwendige registers Z, respectievelijk W.
M_4 en M_5 tenslotte verlopen zoals hieronder aangegeven.

M_4		
WZ OUT, STATUS	DATA WZ=WZ+1	L
T_1	T_2	T_3

M_5		
WZ OUT, STATUS	DATA	H
T_1	T_2	T_3

Annex 3 geeft tenslotte een overzicht van de samenstelling in functie van de tijd van de instructies van de 8080. Het is de lezer aangeraden eerst de volledige instructieset (§ 4.2) door te nemen vooraleer deze tabel te bestuderen.

LISTAGE DE CASSETTE

DAInamic 17 p. 220-221

Si, comme moi, vous devez travailler avec la panoplie du pauvre - composée d'un DAI pc, d'une TV, d'un magnétocassette (et d'un télétype ASR) - alors le programme ci-dessous peut vous intéresser.

Surtout si, comme moi, vous avez tendance à enregistrer pas mal de programmes sur une seule bande et si, au plus fort de la bataille, vous négligez de noter ponctuellement " où " se trouve " quoi "...

Le programme en question est basé sur un programme de liste déjà paru dans DAInamic avec une extension supplémentaire de calcul automatique de l'indication du compteur..

C'est surtout ce calcul qui, à mon avis, représente la grande force de cette version, car il n'y a que par ce moyen que l'on peut obtenir un catalogue vraiment utilisable d'une cassette.

COMMENT CELA FONCTIONNE-T-IL ?

Afin de ne pas allonger le temps de lecture, je n'ai pas fait figurer dans le programme lui-même le texte détaillé et les explications du principe que l'on trouvera ci-après :

Laisser l'ordinateur lire le TITRE des programmes et des fichiers, calculer le temps entre la lecture de deux titres successifs et en déduire le niveau atteint par le compteur.

Compte tenu que le compteur interne du DAI ne tourne pas pendant la lecture du titre, il y a lieu d'apporter une correction en ajoutant environ 2,5 secondes pour chaque titre lu (environ 2,3 s pour le signal de synchronisation + le temps nécessaire à la lecture du titre. Voir ligne 310 où le temps écoulé est calculé en unités de 20 ms).

Etant donné que la bobine de déroulement va tourner progressivement plus vite, le niveau " N " atteint par le compteur au bout du temps T est calculé au moyen de la formule (approchée) qui figure dans les lignes 320 et 490.

Dans cette formule " R " est le rayon de la bobine de déroulement pleine; " D " est l'épaisseur de la bande pour une cassette C 60; ($\simeq 13\,\mu$ pour une C 90); et " L " est la longueur TOTALE du ruban magnétique, calculée depuis le début de la bande (compteur à 000 !) jusqu'au commencement du programme pour lequel nous voulons calculer le niveau " N " atteint par le compteur. (voir aussi les lignes 200 à 200 inclus).

La valeur de " L " est donnée par : L! = T * V ! * 2 E - 2 * F!

où " T " est le temps écoulé en unités de 20 ms calculé à la ligne 310; " V " est la vitesse de la bande (en m/s) et " F " un facteur de correction qui dépend de la cassette concernée - voir plus loin (au départ, la valeur de " F " est de 1.0, c'est pourquoi " F " ne figure pas dans la ligne 320). Le facteur" 0.6 " est le rapport de transmission entre la bobine de déroulement et le compteur (il est possible que cette valeur soit différente pour une marque autre que le Philips N 2225 !).

Au lancement du programme, celui-ci demande d'abord la date, puis le nom de la bande et enfin s'il y a déjà un " EOF " (End of File : indicateur de fin de fichier). En effet, si après le dernier programme enregistré sur la bande, il y a un "EOF", on peut tester dessus pour arrêter. (,sinon il n'est pas (facilement) possible de découvrir si l'on a " vu " le dernier programme de la bande ...).

S'il n'y a pas d'enregistrement " EOF ", l'ordinateur proposera d'en écrire un. Si on n'en veut pas ou si seulement un nombre limité de titres est intéressant, on indique à la place le nombre de programmes.

Après avoir réenrouler la bande et mis le COMPTEUR à 000, on doit appuyer en même temps sur la touche " START " du magnetocassette et la barre-espace. Le programme lit alors les titres, les affiche à l'écran et enregistre le temps écoulé (T), le niveau calculé pour le compteur (N) et le titre lu (A $) dans les tableaux prévus à cet effet (" T (i), " N (i) ", " T$ (i) ").

Lorsque le nombre de titres demandé ou l' "EOF" a été lu, apparaît une vue d'ensemble de tous les titres enregistrés avec les niveaux du compteur correspondants calculés pour F! = 1.0.

Le programme demande alors le niveau REEL correspondant au dernier programme (ou à l'EOF"). Avec cette donnée, il est fait un nouveau calcul itératif (adaptation de " F ! ") pour faire coïncider les calculs avec la réalité (parfois, cela ne réussit pas de façon précise pour tous les niveaux du compteur, mais l'écart dépasse rarement 1 à 2 chiffres !).

Après ce calcul, la vue d'ensemble corrigée apparaît pour contrôle et on peut, si on veut, à nouveau, introduire une " valeur réelle ", etc ...

Finalement la valeur " 0 " doit être donnée pour sortir de cette boucle et la question " Impression (O/N) ? est posée.

Si c'est " O " apparaît l'avertissement de mettre l'imprimante (ou le télétype) en service, puis l'impression est faite après appui sur la barre-espace.

--------ooOoo---------

NB.- Si on a des difficultés avec la linéarité (nombres dans la zone du milieu trop grands ou trop petits), il faudra probablement adapter un peu la valeur de " R " (et / ou de " D ") (ligne 240). Il se peut également que la vitesse " V " ne soit pas exactement 4.75

Cette " adaptation " peut se faire tout simplement en modifiant pendant un " BREAK " une ou plusieurs valeurs, et en introduisant à nouveau après " CONT " la " valeur réelle"exacte ! (Répéter si nécessaire jusqu'a obtention de la précision désirée ...).

L' impression peut évidemment être réalisée sur n'importe quelle imprimante. Dans ce cas, seule la "baudrate " doit être adaptée et l'attente après une retour chariot (CR) (marche à vide avec " NUMBER " - pas de WAITTIME sinon le chronométrage est perdu !) n'est souvent plus nécessaire.

DCE-bus INTERFACING

Pinconfiguration DCE-bus seen to the back of the DAIpc.

```
  pin 33  31  29  27  25  23  21  19  17  15  13  11   9   7   5   3   1

     INTR B1  C3  C1  B3  B5  B7  C5  C7  A7  A6  A5  A4 EXRS IN7 -5V  +5V
      IN7 B2  B0  C2  C0  B4  B6  C4  C6  A0  A1  A2  A3  NC  XINT GND +12V

  pin 34  32  30  28  26  24  22  20  18  16  14  12  10   8   6   4   2
```

Pinconfiguration IC 8255A GIC.

```
 pin  20  19  18  17  16  15  14  13  12  11  10   9   8   7   6   5   4   3
       2   1

  B2  B1  B0  C3  C2  C1  C0  C4  C5  C6  C7  A0  A1 GND nCS nRD ,A0  A1  A2
       A3
  B3  B4  B5  B6  B7 Vcc  D7  D6  D5  D4  D3  D2  D1  D0 RST nWR  A7  A6  A5
       A4

 pin  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
      39  40
```

The schemes that follow represent the possibilities of the
ic 8255 in its 0-mode.
You will have to POKE the hexadecimal value as command word
in the adress #FE03 of the GIC-controller.
Warning: do not PEEK #FE03.

######## ic-8255a-PINS ########

######## m o d e 0 ########

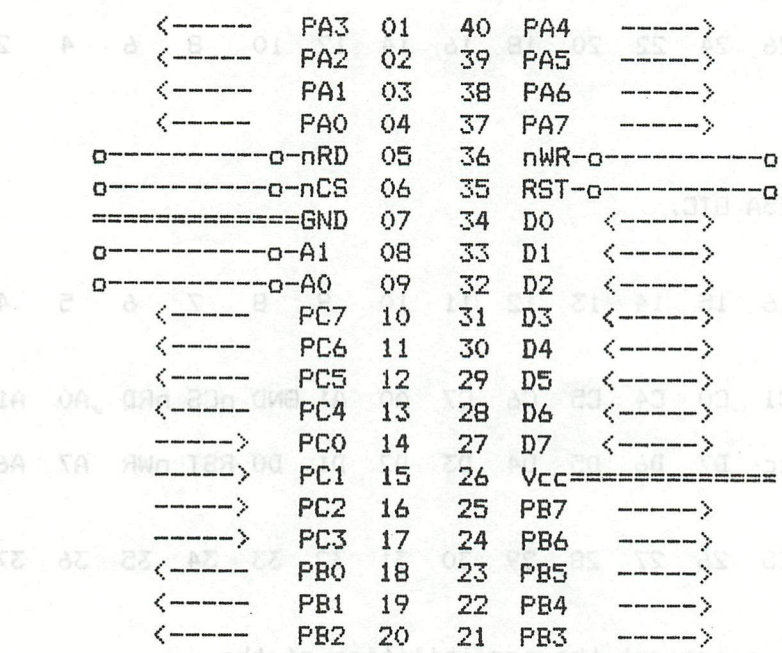
control word #0

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    0  0  0  0
hexadecimal :              #80
```

```
          <-----   PA3  01   40  PA4   ----->
          <-----   PA2  02   39  PA5   ----->
          <-----   PA1  03   38  PA6   ----->
          <-----   PA0  04   37  PA7   ----->
      o----------o-nRD  05   36  nWR-o----------o
      o----------o-nCS  06   35  RST-o----------o
      ============GND  07   34  D0   <----->
      o----------o-A1   08   33  D1   <----->
      o----------o-A0   09   32  D2   <----->
          <-----   PC7  10   31  D3   <----->
          <-----   PC6  11   30  D4   <----->
          <-----   PC5  12   29  D5   <----->
          <-----   PC4  13   28  D6   <----->
          <-----   PC0  14   27  D7   <----->
          <-----   PC1  15   26  Vcc============
          <-----   PC2  16   25  PB7   ----->
          <-----   PC3  17   24  PB6   ----->
          <-----   PB0  18   23  PB5   ----->
          <-----   PB1  19   22  PB4   ----->
          <-----   PB2  20   21  PB3   ----->
```

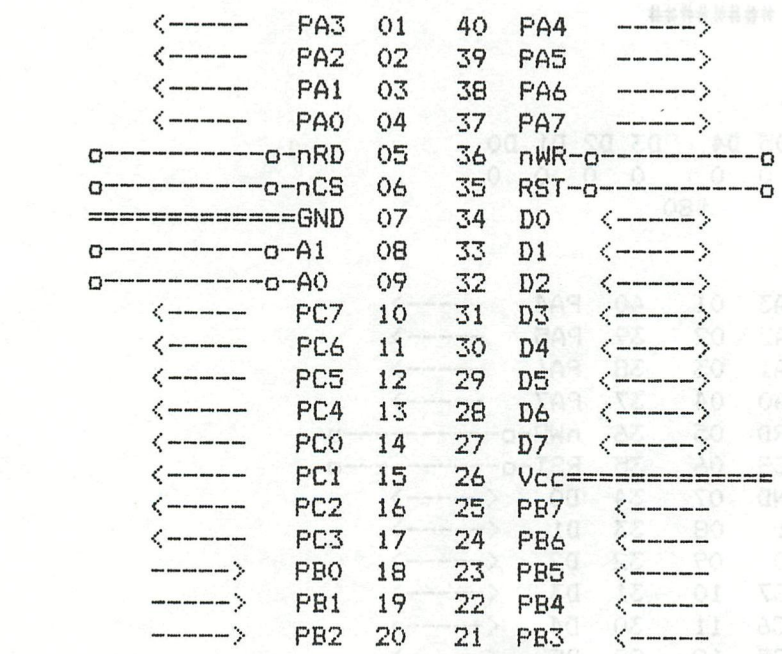
control word #1

```
              D7 D6 D5 D4   D3 D2 D1 D0
binary:        1  0  0  0    0  0  0  1
hexadecimal :            #81
```



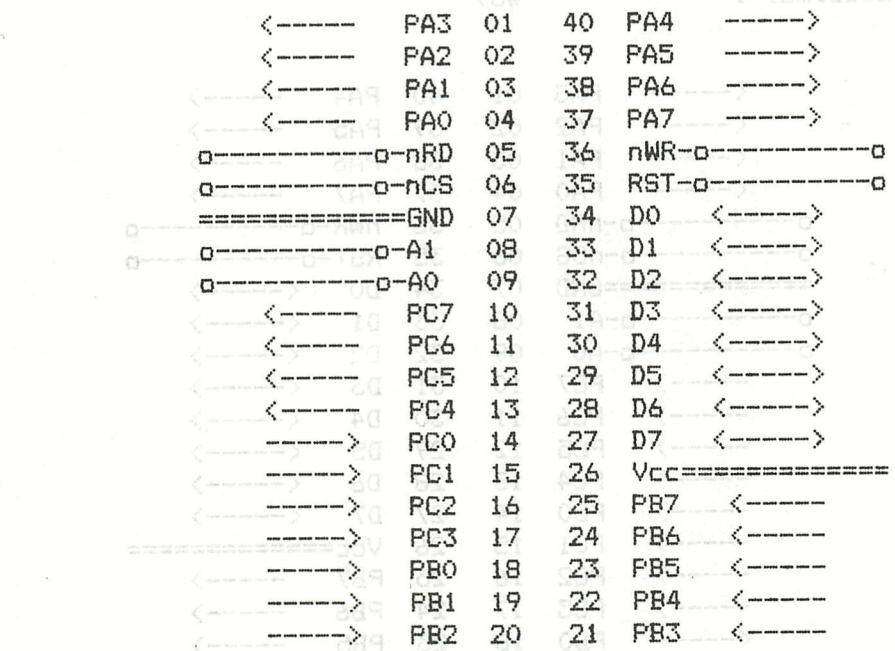
control word #2

```
              D7 D6 D5 D4   D3 D2 D1 D0
binary:        1  0  0  0    0  0  1  0
hexadecimal :            #82
```



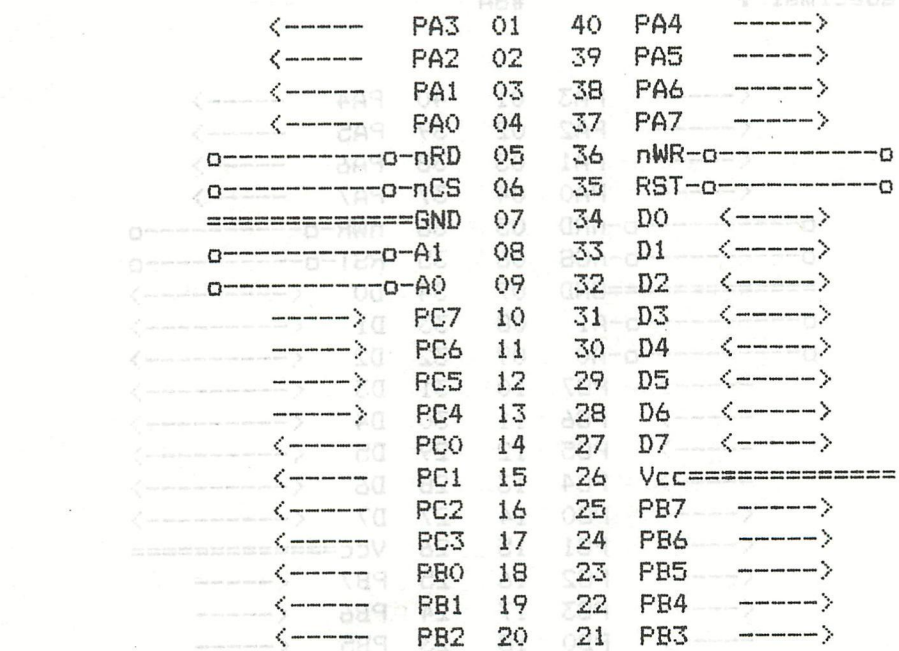
control word #3

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    0  0  1  1
hexadecimal :          #83
```



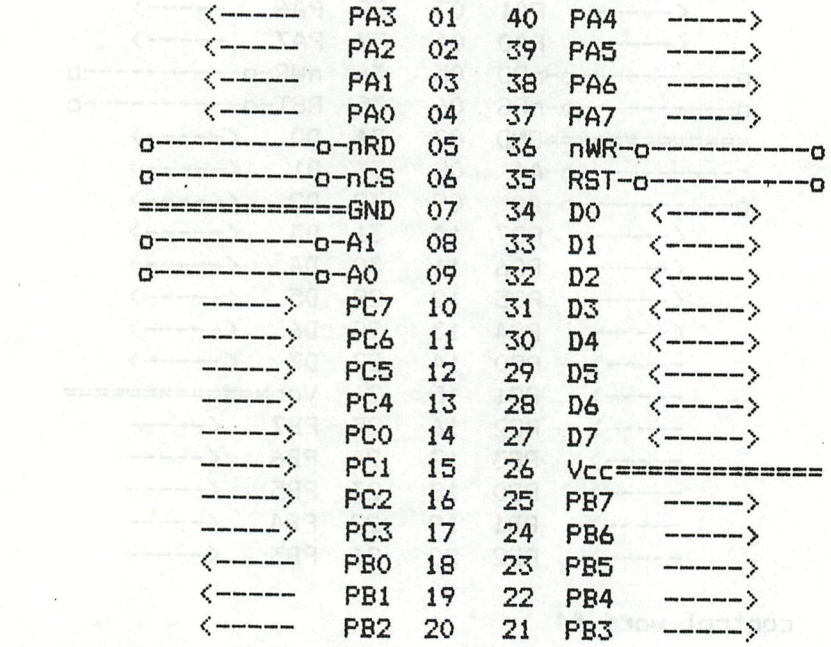
control word #4

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    1  0  0  0
hexadecimal :          #88
```



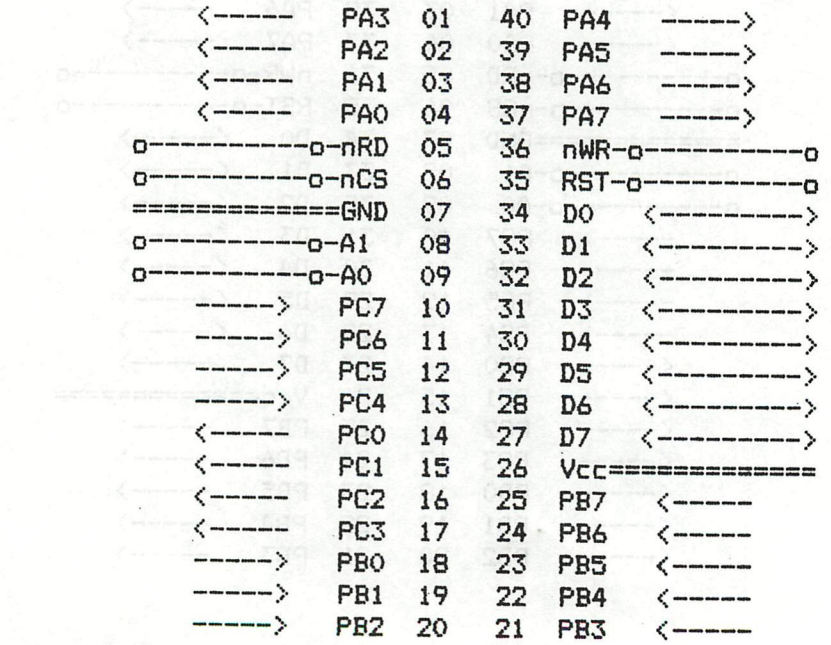
control word #5

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    1  0  0  1
hexadecimal :              #89
```



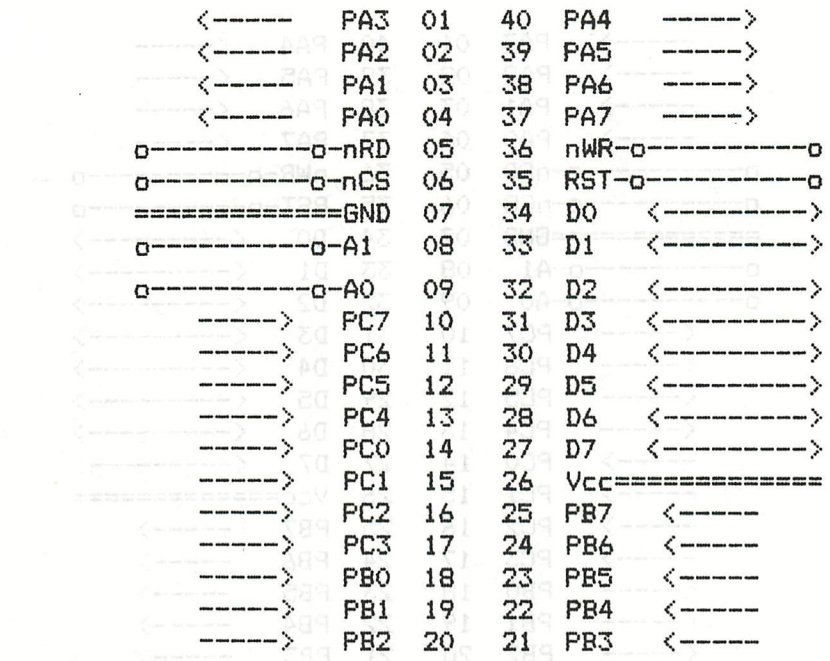
control word #6

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    1  0  1  0
hexadecimal :              #8A
```



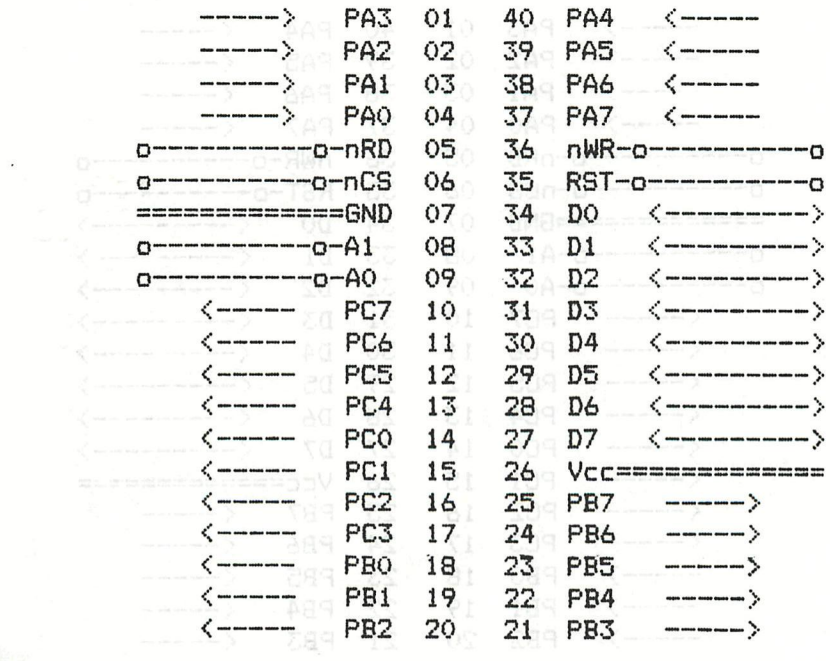
control word #7

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  0    1  0  1  1
hexadecimal :              #8B
```



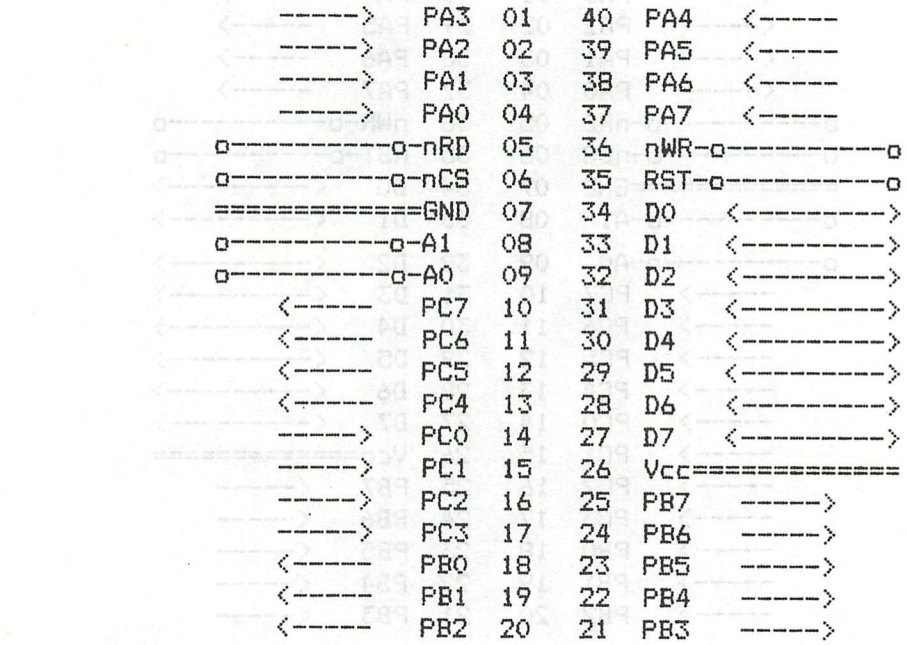
control word #8

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    0  0  0  0
hexadecimal :              #90
```



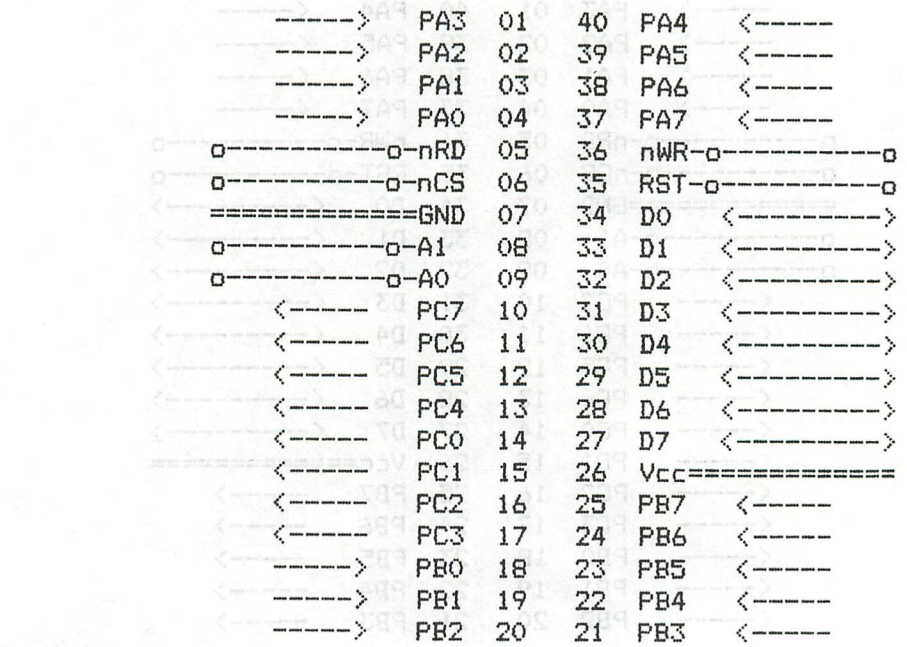
control word #9

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    0  0  0  1
hexadecimal :              #91
```



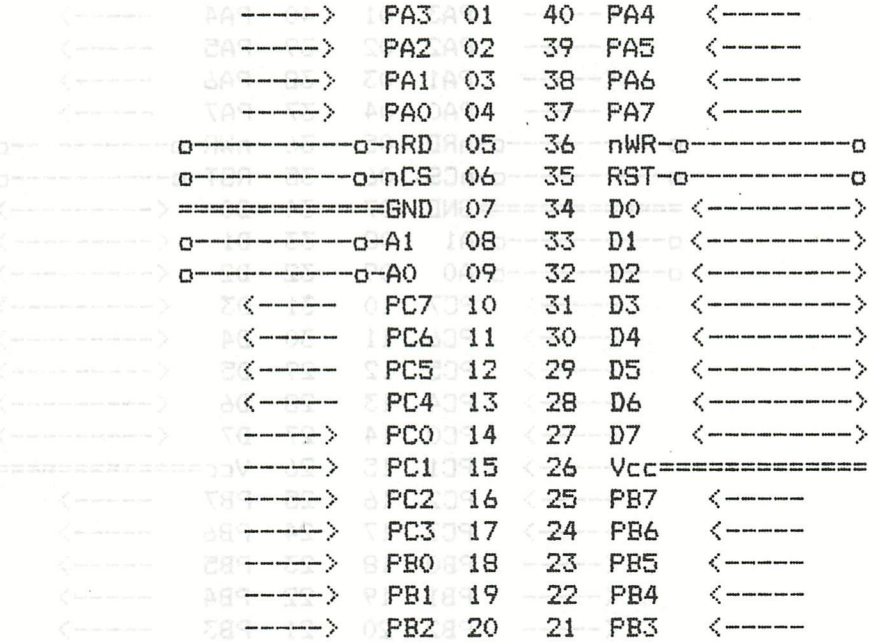
control word #10

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    0  0  1  0
hexadecimal :              #92
```



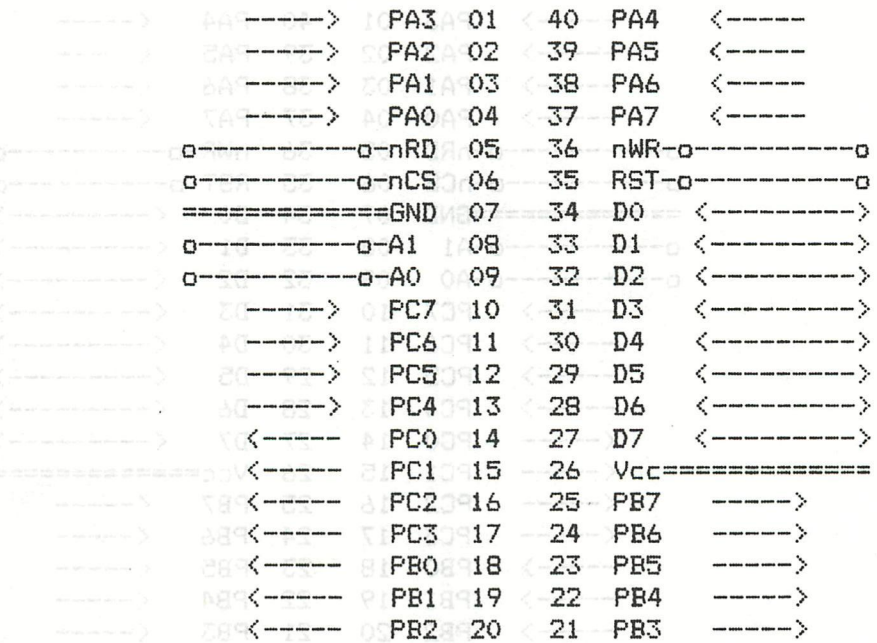
control word #11

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    0  0  1  1
hexadecimal :          #93
```



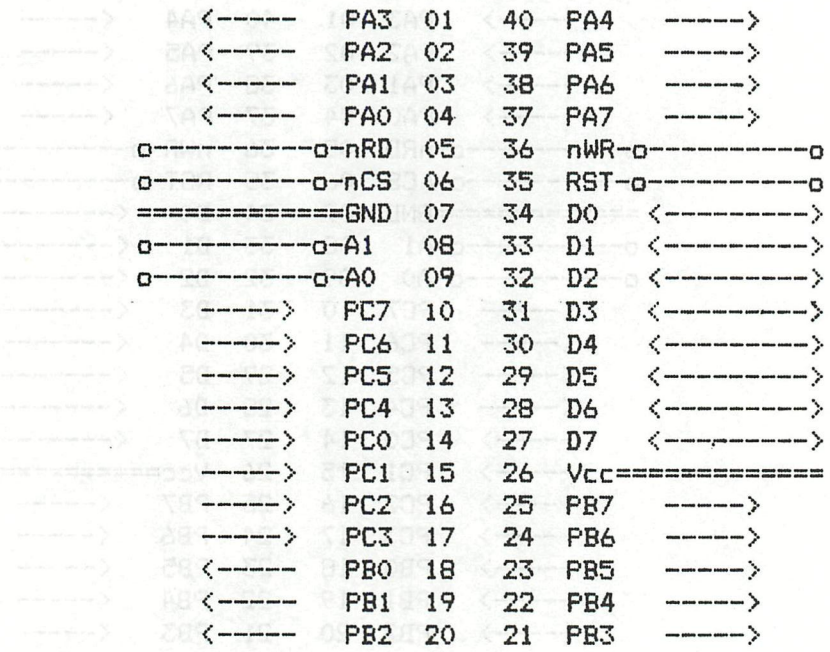
control word #12

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    1  0  0  0
hexadecimal :          #98
```



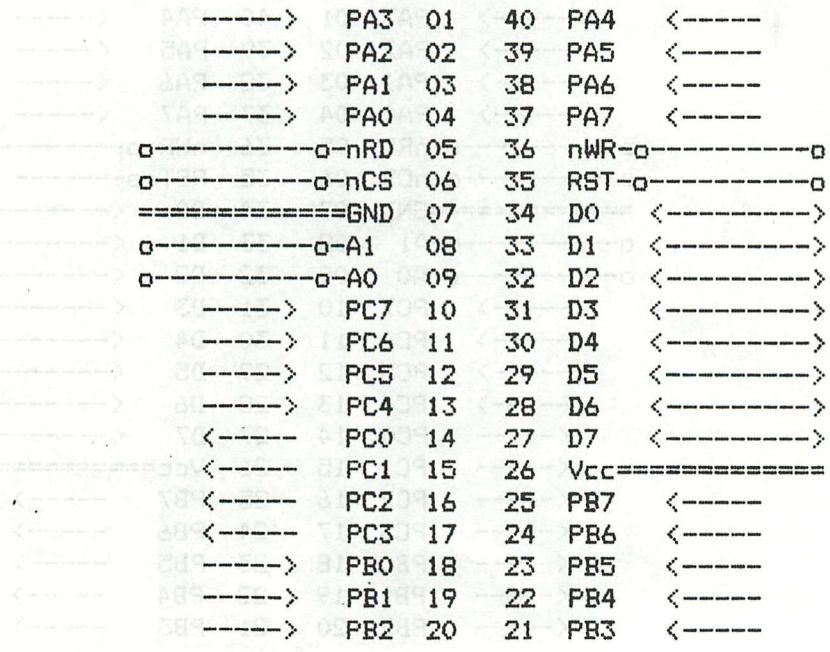
control word #13

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    1  0  0  1
hexadecimal :              #99
```



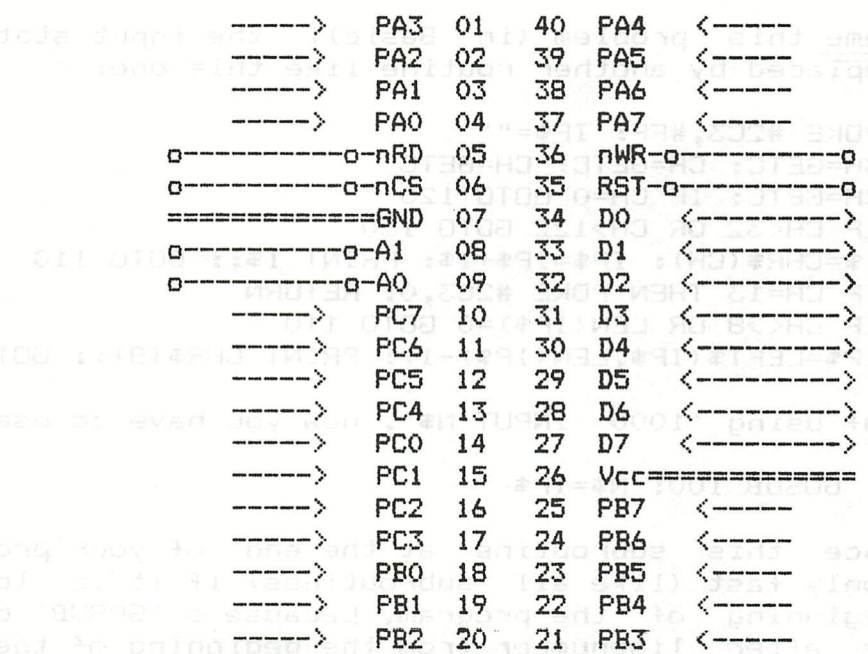
control word #14

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    1  0  1  0
hexadecimal :              #9A
```



control word #15

```
                D7 D6 D5 D4   D3 D2 D1 D0
binary:          1  0  0  1    1  0  1  1
hexadecimal :          #9B
```



The pins nRD, nCS, A1, A0, nWR and RST are not connected to the DCE-connector.

Please send your comments to Inno Broekman
Avenbeek 98
2182 RZ Hillegom
The Netherlands

LOWER CASE CHARACTERS IN INPUT STATEMENT.

In the Basic version V1.1, all keyboard pointers are set to default when an INPUT statement has to be performed. Therefore, it is not possible to obtain lower case characters in the input statement, except by using the SHIFT- or the CTRL-key.
Also the statement 'POKE #2C3,#FF' before the input statement remains without action.

To overcome this problem (in Basic), the input statement must be replaced by another routine like this one:

```
100 POKE #2C3,#FF: IP$=""
110 CH=GETC: CH=GETC: CH=GETC
120 CH=GETC: IF CH=0 GOTO 120
130 IF CH<32 OR CH>122 GOTO 150
140 I$=CHR$(CH): IP$=IP$+I$: PRINT I$;: GOTO 110
150 IF CH=13 THEN POKE #2C3,0: RETURN
160 IF CH<>8 OR LEN(IP$)=0 GOTO 110
170 IP$=LEFT$(IP$,LEN(IP$)-1): PRINT CHR$(8);: GOTO 110
```

In stead of using '1000 INPUT N$', now you have to use:

```
1000 GOSUB 100: N$=IP$
```

Do not place this subroutine at the end of your program. It runs only fast (like all subroutines) if it is located at the beginning of the program, because a 'GOSUB' checks linenumber after linenumber from the beginning of the program until the correct linenumber has been found.

(c) - Jan Boerrigter, Febr. 1984

Use less code for fast 8080 multiply

Larry D Iffrig
Western Electric Co, Ballwin, MO

An alternative to Ira Baxter's unrolled-loop method for 8080-μP multiplication (EDN, February 17, pg 261) requires much less code and results in only a small increase in execution time. The new method (**figure**) retains a program loop, thus eliminating seven replications of add-and-shift operations. However, it avoids the usual loop-overhead functions of initializing and decrementing a counter.

The new routine still performs a conditional branch back to the loop's origin, though. It initially sets the HL register pair's bit 9 (register H's bit 1) to ONE, and each loop iteration shifts HL's contents to the left, thus setting the Carry flag and terminating the loop after seven iterations.

With an 800-nsec clock cycle, the new routine achieves an average execution time of 251 μsec and a worst-case time of 283 μsec. Although these figures represent a 25% increase over the original routine, the new routine requires only 60% as much code. **EDN**

```
        LXI  H,?00H    Clear product accumulator, set "counter" bit
        MOV  A,D       Copy multiplier to A register
        MVI  D,00H     Make D/E register pair into 16-bit multiplicand
LOOP    ADD  A         Shift multiplier bit into carry
        JNC  SHIFT     Branch if multiplier bit is zero
        DAD  D         Add multiplicand to partial product
SHIFT   DAD  H         Left shift partial product
        JNC  LOOP      Branch if "counter" bit not shifted to carry
        ADD  A         Eighth shift of multiplier bit into carry
        JNC  END       Branch if eighth multiplier bit is zero
        DAD  D         Add multiplicand to partial product
END
```

Using only 20 bytes of code, *this 8080-μP multiplication routine executes in an average time of 251 μsec. It accepts an 8-bit multiplier in register D and an 8-bit multiplicand in register E, producing a 16-bit product in register pair HL.*

9. TIMERS

De vijf timers kunnen elk, afhankelijk van de waarde waarmee ze worden gevuld, een tijdvertraging tot stand brengen van minimaal 64 µsec en maximaal 16.320 µsec.
Elke timer is in feite een 8-bits down counter, die een klok-signaal krijgt toegevoerd met een periodetijd van 64 µsec.

Wanneer de inhoud van een timer 00000000_2 wordt, geeft hij een interrupt request die een bepaalde interrupt service routine tot gevolg heeft.

Deze timers worden gebruikt wanneer we b.v. een proces hebben, waarbij om de zoveel tijd een bepaalde test uitgevoerd moet worden.
Steeds wanneer een timer de waarde 0 heeft bereikt, wordt naar een interrupt service routine gesprongen, die deze test uitvoert en daarna de timer weer met een bepaalde waarde vult.
Hierna kan met het hoofdprogramma worden doorgegaan, totdat de timer weer een interrupt request geeft, enz.

10. INTERRUPT CONTROLLER

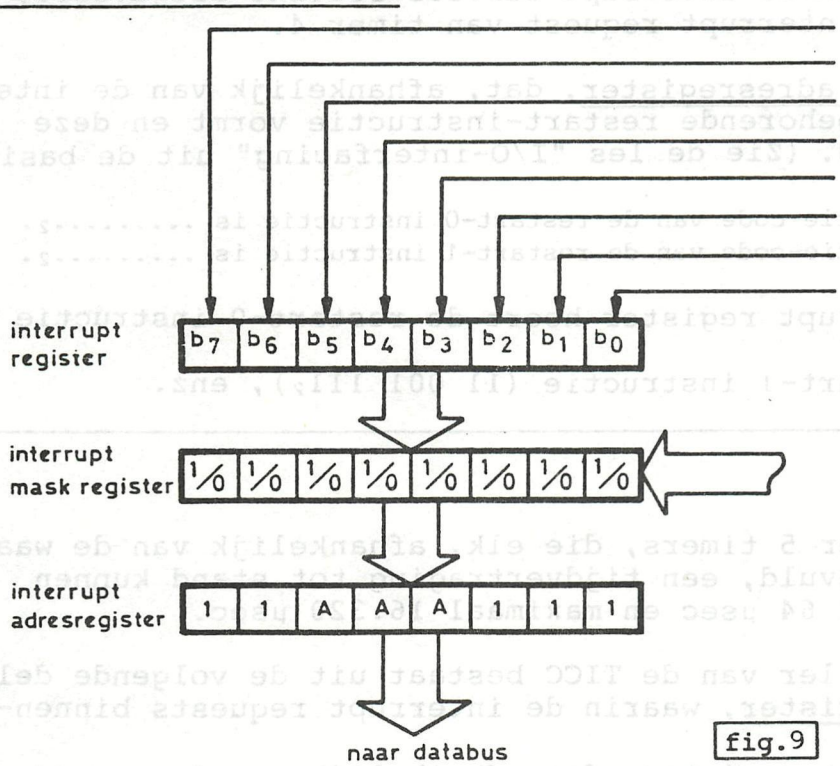


fig.9

De interrupt controller die in de TICC is aangebracht, bestaat uit de volgende delen (fig. 9).
1. Een 8 bits interrupt register, waarin de interrupt requests worden opgeslagen.

Vraag 10: Wanneer b_3 van het interrupt register 1 is, betekent dit, dat er een interrupt request is van timer 2/het transmitter register/timer 3.

Is b_3 b.v. 1, dan betekent dit, dat timer 3 de waarde 00000000_2 heeft bereikt en een interrupt request heeft gegeven.

Wanneer een bepaalde interrupt service routine is uitgevoerd, wordt de betreffende bit in het interrupt register automatisch gereset.

Antw. 10: timer 3.

2. Een 8 bits interrupt mask register, waarmee we kunnen bepalen welke interrupt request aan de CPU wordt doorgegeven. Dit register heeft adres 9808_{16}. Willen we b.v. alleen een interrupt request van timer 3 aan de CPU doorgeven, dan vullen we het interrupt mask register met de waarde 00001000_2. We kunnen interrupt requests dus maskeren (= aan het oog onttrekken), door de overeenkomstige bit in het interrupt mask register 0 te maken.

   Wanneer we het interrupt mask register vullen met een zodanige waarde, dat meer dan één interrupt wordt doorgegeven, dan controleert dit register tevens de prioriteiten.
   Hierbij geldt, dat b_7 van het interrupt register de laagste prioriteit heeft.

   Voorbeeld:
   Wanneer we het interrupt mask register gevuld hebben met 01010000_2 en timer 4 en het receiver register plegen tegelijkertijd een interrupt request, dan wordt eerst de interrupt service routine uitgevoerd die hoort bij de interrupt request van het receiver register. Daarna wordt de interrupt service routine uitgevoerd, die behoort bij een interrupt request van timer 4.

3. Een 8 bits interrupt adresregister, dat, afhankelijk van de interrupt request, de bijbehorende restart-instructie vormt en deze doorzendt naar de CPU. (Zie de les "I/O-interfacing" uit de basiscursus.)

   Vraag 11: De operatie-code van de restart-0 instructie is .....$_2$.
   De operatie-code van de restart-1 instructie is .....$_2$.

   Bij b_0 van het interrupt register hoort de restart-0 instructie ($11\ 000\ 111_2$).
   Bij b_1 hoort de restart-1 instructie ($11\ 001\ 111_2$), enz.

SAMENVATTING 3

7. De TICC beschikt over 5 timers, die elk, afhankelijk van de waarde waarmee ze worden gevuld, een tijdvertraging tot stand kunnen brengen van minimaal 64 µsec en maximaal 16.320 µsec.

8. De interrupt controller van de TICC bestaat uit de volgende delen.
   a. Een interrupt register, waarin de interrupt requests binnenkomen.
   b. Een interrupt mask register, dat afhankelijk van de waarde waarmee het wordt gevuld, bepaalde interrupts kan maskeren of doorlaten. Het interrupt mask register controleert tevens de prioriteiten.
   c. Een interrupt adresregister, waarin de restart-instructies gevormd worden.

COPYRIGHT © 1978 ELEKTRONICA OPLEIDINGEN DIRKSEN ARNHEM, NEDERLAND

Testbericht KEN-DOS

Vor einem Jahr wurde von dem niederländischen DAI-Benutzer Kenneth Gooswit ein selbstentwickeltes DOS für den DAI angekündigt, das angeblich wesentlich schneller und leistungsfähiger sein sollte, als das DOS von dem DAI-Hersteller INDATA. Nach langem Warten ist das KEN-DOS System jetzt endlich erhältlich. Ich arbeite seit Mitte Januar damit, und wollte hier meine ersten Erfahrungen schildern.

Mein System hatte ich im Oktober bestellt und Anfang November bezahlt. Daß ich es erst im Januar erhielt, liegt zum Teil daran, daß die für mich gekauften einseitigen Laufwerke defekt waren (die Epromkarte war schon Mitte Dezember fertig); als ich mich Anfang Januar entschloß, gegen Aufpreis vorrätige doppelseitige Laufwerke zu nehmen, wurde das System prompt verschickt und ich konnte es nach einer weiteren Woche Transportzeit beim Zoll abholen (es mußte zwar keine Zollgebühr bezahlt werden, aber 14% Mehrwertsteuer.) Meins ist sicher eins der ersten ausgelieferten Systeme,so daß diese Lieferzeit nicht unbedingt typisch ist. Kenneth Gooswit gibt zur Zeit 4 Wochen als Lieferzeit an--hinzu kommen 1-2 Wochen für die Überweisung des Kaufpreises und 1 Woche Versandzeit.

Was sich mir darbot, als ich die Verpackung öffnete, machte einerseits einen sehr soliden Eindruck, verriet aber auch, daß es nicht von einem professionellen Computerbetrieb stammte. In einem sehr schweren und stabilen Gehäuse (so wie man sie in Elektronikläden kaufen kann) waren die beiden Laufwerke eingebaut (in meinem Fall, doppelseitige 80-Spur Laufwerke Marke MPI) zusammen mit dem Kontroller und dem Netzteil für die Laufwerke--auf der Rückseite des Gehäuses waren zwei riesige Kühlbleche, eine Flachbandkabelverbindung zum DCE-Bus des DAI, mit aufgesetzter Buchse zum Anschluß eines DCR (ob man problemlos hier auch andere Real World Karten anschließen kann, weiß ich nicht), und ein etwas klein dimensionierter Ein/Ausschalter für die Diskettenstation; wohlversteckt unter dem Flachbandkabel ein weiterer Schalter zum Umschalten zwischen reinen Diskettenbetrieb und MDCR Betrieb (natürlich kann man auch dann gleichzeitig die Disketten benutzen! aber mit dem Schalter auf "Disk" kann man die DCR nicht benutzen.) Ein amateurhaftes Detail: die Einfassung der Laufwerke in das Gehäuse war mit einem schwarzen Klebeband verziert. Für die Laufwerke lag ein dickes Stromkabel bei.

Zum Lieferumfang gehörte natürlich auch die Epromkarte mit dem eigentlichen DOS (etwa 16 KB Maschinenprogramm, wie mir scheint). Diese Karte wird auf den X-Bus im DAI aufgesteckt und nimmt dann die volle Höhe und etwa 2/3 der Breite des DAI ein. Das DOS befand sich in zwei gesockelten Eproms, aber es sind sechs Sockel auf der Karte vorhanden, so daß man leicht eigene wichtige Programme in Eproms auf dieser Karte halten kann (die neueste DOS-Version belegt aber schon drei Sockel). Das DOS ist geteilt in Module, die jeweils den Adressraum #F000 bis #F7FF belegen und über ein KEN-DOS-eigenes Bankswitching selektiert werden. Die Karte enthält auch 512 Byte RAM im Adressenbereich #F900 bis #FAFF, das vom DOS als Speicher für DOS-interne Daten verwendet wird; auch das Programm zum Umschalten der Bänke wird in dieses RAM kopiert. Dieses System hat große Vorteile für den Benutzer: zum einen belegt das DOS trotz seines Umfangs nur wenig von dem normalen RAM-Bereich des DAI (1,5 KB für Zeiger und Directory). Zum andern ist der DAI so getaktet, daß Datenzugriffe auf das RAM mit Zugriffen des Videokontrollers alternieren, wodurch Programme im RAM nur halb so schnell laufen wie theoretisch möglich; bei Programmen in ROM oder in statischem RAM im Adressbereich oberhalb #C000 gibt es hingegen keine Verzögerungen, sie laufen mit vollen 2 MHz. Insofern ist KEN-DOS etwa doppelt so schnell in der Ausführungszeit, als wenn es im RAM läge. (Ein Tip: wenn Sie Programme haben, die sehr schnell oder mit sehr genauem Zeitablauf laufen müssen, legen Sie sie auf den Stapel! oder in das KEN-DOS RAM. Auch hierzu ist das KEN-DOS RAM nützlich, wenn es sich, wie ich vermute, um ein statisches RAM handelt. Da die Beschriftung von den IC's abgekratzt wurde, kann ich das nicht genau feststellen.)

Natürlich lag auch eine Bedienungsanleitung bei, mit einer Kurzbeschreibung der KEN-DOS Befehle, einer Beschreibung von der Speicherbelegung, und einer sehr gründlichen und klaren Anleitung zur Inbetriebnahme des Systems. Die Kurzbeschreibung der Befehle war etwas dürftig, aber eine bessere Bedienungsanleitung soll nachgeliefert werden. Die Anleitung zum Anschluß des Systems ließ aber keine Fragen offen. Ein etwas merkwürdiges Detail: die Bedienungsanleitung war ein von einem Matrixdrucker gedrucktes Original!

Zur Inbetriebnahme war erst einmal nötig, eine Drahtverbindung auf

die DAI-Platine einzulöten zwischen Pin 49 des X-Bus (Hold Request) und Pin 8 des DCE-Bus; dieser Umbau war genau beschrieben und ging völlig problemlos vonstatten--der Draht dazu und sogar Lötzinn wurden mitgeliefert! Da an den Lötstellen keine IC's sitzen besteht fast keine Gefahr, den DAI zu beschädigen. Das Formatieren der ersten Diskette und das Austesten der Laufwerke lief auch einwandfrei und ich konnte bald beginnen, mit dem System zu arbeiten.

Als erstes wollte ich natürlich wissen: wie schnell und wie leistungsfähig ist KEN-DOS? Hier wurde ich nicht enttäuscht! Das System kann in der Speicherkapazität und in der Schnelligkeit mit den besten kommerziellen Produkten konkurrieren: auf eine Seite von einer 80-Spur Diskette passen 400 KB, wovon 385 KB für den Benutzer frei bleiben (der Rest ist für das DOS reserviert), und das Abspeichern von 60 KB auf der Diskette, einschließlich Erstellen einer Datei, Anlaufen des Motors usw., braucht etwa 8 Sekunden!! (Das ist nicht ganz so schnell wie die von Gooswit behauptete Zeit von 32 Sekunden für 400 KB, ist aber auch nicht nur die reine Datentransferzeit. Es ist immerhin zehnmal schneller als die DCR und 100 mal schneller als Kassetten, und zum Vergleich etwa viermal schneller als die Datentransferrate der **professionellen** CBM Laufwerke [ich meine damit nicht das langsame CBM64 Laufwerk]) Man kann also ohne weiteres auch große Blöcke oft auf Diskette sichern, oder ein großes Programm für eine schnelle Benutzung kurz einmal laden, ohne daß es lästig wird. Zum Teil verdankt das System seine Effizienz der von KEN-DOS benutzten ungewöhnlichen Sektorenlänge von 1 KB, aber diese Länge ist bei der üblichen Dateigröße von über 4 KB insgesamt wesentlich sparsamer als die üblichen 128 oder 256 Byte-Sektoren, weil weniger Platz auf der Diskette für die Markierung der Sektoren gebraucht wird.

Die Freude über die Leistungsfähigkeit des Systems wurde aber bald getrübt durch das Auftreten der ersten Mängel, sowohl in der Software wie in der Hardware. Wichtigstes Hardwareproblem ist, daß der DAI mit KEN-DOS außerordentlich empfindlich wird gegen Schwankungen in der Netzspannung und statischer Aufladung; vielleicht liegt das zum Teil daran, daß die Epromplatine von dem normalen DAI Netzteil versorgt wird, das nicht dafür ausgelegt ist. Die Auswirkungen: spontane Hardbreaks oder Stack Overflow, oder die Laufwerksmotoren fangen von alleine an zu laufen. (Bei mir fingen die Motoren auch an zu laufen, wenn ich vom UT aus ein Maschinenprogramm nach Z3 mit dem G Befehl starten wollte). Diese Fehler treten auch oft auf, wenn man z.B. einen Drucker einschaltet oder ein anderes Gerät. Komischerweise hatte ich solche Fehler meistens bei Maschinenprogrammen, wo das nur ein Ärgernis war und nicht zu Datenverlust führte, aber die Folgen können manchmal schon schlimmer sein, zumal man sich auch nicht immer mit einem Reset aus einem Aufhängen des Rechners retten kann: wegen der Bankumschaltung wird KEN-DOS bei Reset nur richtig initialisiert, wenn Bank 0 selektiert ist--sonst muß man den DAI ganz ausschalten!!

Herr Gooswit behauptet, durch Einlöten eines Widerstands und eines Kondensators auf die DAI-Platine kann die Empfindlichkeit gegen Störung reduziert werden, und er hat diesen Umbau auch (kostenlos) bei meinem Rechner durchgeführt. In der Tat scheinen die Ausstiege auch seltener geworden zu sein, aber ganz behoben ist das Problem damit noch nicht. Auf der Gegenseite kann man sich gegen die Konsequenzen solcher Ausstiege schützen durch häufiges Absichern der Programme und Daten auf Diskette, was wegen der Schnelligkeit des Systems keinen Zeitverlust bedeutet. Es gibt sogar einen DOS-Befehl, der erlaubt, mit nur einer Taste sein Programm oder einen Speicherbereich abzuspeichern.

Das Schreiben und Lesen auf der Diskette scheint sehr zuverlässig zu sein, und ich habe fast nie Schreib- oder Lesefehler gehabt. Allerdings habe ich schon einige Schwierigkeiten erlebt bei Verwendung der innersten Spuren der 80-Spur Disketten, aber ich weiß nicht inwiefern man diese Probleme dem DOS zuschreiben soll oder eher den Laufwerken. KEN-DOS kann so ziemlich mit beliebigen Laufwerken geliefert werden, oder man kann die Laufwerke auch selber kaufen.

Was die Software betrifft, so war die Version, die mir im Januar geliefert wurde, zwar lauffähig (mit einigen kleineren Fehlern) aber unvollständig (einige Befehle waren noch nicht implementiert, aber eine vollständigere Version war für Ende Januar versprochen). Größter Nachteil war die Inkompatibilität zum DAI-Kassettenbetriebssystem. Zwar funktionierten die normalen Befehle SAVE,LOAD,SAVEA,LOADA,R und W auch mit DOS, aber der Aufbau einer Kassettendatei wurde im DOS nicht nachgemacht, so daß andere Programme, die die Kassettenschreib- und Leseprogramme für Nichtstandarddateien verwendeten, mit DOS nicht richtig funk-

tionierten. Das Problem lag darin, daß auf Kassette zwei Datenblöcke geschrieben werden, von denen der erste außer bei Basicprogrammen nur ein oder zwei Byte lang ist (z.B. bei AHT Assemblerquelltextdateien enthält dieser Block die Anzahl der Zeilen). Da es sich nicht lohnt, nur zwei Byte auf die Diskette zu schreiben, haben die frühen DOS Versionen diese Information einfach weggeschmissen!

Wegen der Unvollständigkeit des ersten DOS hatte ich ohnehin Gelegenheit, Herrn Gooswit einige Änderungen und Verbesserungen vorzuschlagen, und es hat sich zwischen uns eine sehr fruchtbare Korrespondenz entwickelt, bei der er sich sehr bereitwillig gezeigt hat, auf meine Vorschläge einzugehen. Ich habe inzwischen schon die dritte wesentlich verbesserte DOS-Version, immer noch nicht ganz vollständig, aber meine Verärgerung darüber, daß ich immer noch nicht das DOS habe, für das ich im November bezahlt habe, wird erheblich gemildert durch die Tatsache, daß ich auch nicht gezwungen bin, mich mit den eventuellen Unzulänglichkeiten des DOS abzufinden--was mir nicht gefällt, wird ohne Murren ausgebessert! So ist die oben erwähnte Unverträglichkeit mit Kassettenprogrammen weitgehend behoben--die kurzen 2-Byte Blöcke, die wichtige Information enthalten können, werden jetzt in die Directory gepackt, wo ohnehin Platz vorhanden war, und stehen beim Lesen wieder zu Verfügung. Auch andere ähnliche Probleme sind inzwischen behoben. Zwar sind noch einige zum Teil auch schwerwiegende *Programmierfehler* im DOS vorhanden, aber diese werden sicher in der nächsten DOS-Version korrigiert sein, und ich erwarte, daß das DOS demnächst in einer vollständigen und gut funktionierenden Version vorliegen wird. Ich will deshalb auch nicht weiter hier auf gegenwärtige DOS-Fehler eingehen, sondern lieber versprechen, diesen Bericht später in der Clubzeitung zu aktualisieren, und hier nur noch eine allgemeine Beschreibung des DOS anfügen. Übrigens, alle bisherigen Ausbesserungen des DOS habe ich völlig kostenlos erhalten, gegen Austausch der Eproms.

Das DOS ist (oder wird in seiner Endversion) recht bequem sein. Es erlaubt die Erstellung von sequentiellen oder Random-Dateien--bei letzteren ist der Zugriff auf einzelne Sektoren möglich, so daß man die Diskette als einen großen virtuellen Speicher einsetzen kann (z.B. für Textverarbeitung oder Dateiverwaltung). Random-Dateien können bis zu 255 KByte lang sein. Bei Double Density Betrieb können bis zu 128 Dateien angelegt werden. Einmal geschriebene Dateien können überschrieben werden (also aktualisiert), aber sie können auch gegen Überschreiben geschützt werden. Sie können auch gestrichen werden ("DELETE"), erscheinen aber noch in der Directory, und können später wieder zugänglich gemacht werden. Sie können auch unwiederbringbar gelöscht werden (zur Sicherheit aber nur nach vorherigem DELETE); dann kann der freigewordene Diskettenplatz für andere Dateien neu genutzt werden (und wird auch automatisch genutzt wenn neue Dateien geschrieben werden). Die Diskettenverwaltung verschwendet also keinen Platz auf der Diskette, auch wenn Dateien mehrmals geschrieben und gelöscht werden.

Die Diskettenverwaltung benötigt 1,5 KB vom DAI RAM für eine "File Allocation Map" ("DateiBelegungsKarte") und einen Teil des Directory. Dieser Block sitzt normalerweise direkt unterm Bildschirm (und wird bei Modeänderungen verschoben), kollidiert somit nicht mit den meisten Maschinenprogrammen. Durch Ändern eines Zeigers kann man diesen Block an eine beliebige Stelle im Speicher legen. Versehentliches Überschreiben dieses Blocks scheint keine schlimmen Konsequenzen zu haben, da er dann wieder (nach einmaliger Fehlermeldung) von der Diskette gelesen wird.

Dateien können durch Zuteilung einer Codenummer gegen unerlaubten Zugriff geschützt werden.

Außer den bekannten Basic und Utility Befehlen gibt es im DOS Befehle DSAVE und DLOAD um von Basic aus Utility-Dateien zu schreiben und lesen. Maschinenprogramme können über den üblichen Sprungvektor in #2C5 bis #2E5 die Diskette ansprechen; dies funktioniert jetzt fast auf gleiche Weise, wie bei Kassettenbetrieb. Ein verbleibender interessanter Unterschied--das DOS gibt den Dateityp mit einem Namen an (BAS, UTY, ARY usw.) statt mit einer Zahl (0, 1 oder 2), und es stehen zusätzlich zu Verfügung die Datentypen 3 (SRC für Assemblerquelltext), 4 (RND), 5 (TXT für Textdateien) und 6 (DBS=DataBaSe??); es empfiehlt sich auch, Programme notfalls umzuschreiben, damit sie diese Typen benutzen (dazu muß meist nur ein oder zwei Byte geändert werden), zumal nichtstandard Anwendungen der normalen Typen 0,1,2 mit DOS Probleme ergeben kann. Andere Datentypen (etwa "*" oder "%") werden zwar angenommen aber in den Bereich "0" bis "7" umgewandelt, was auch Probleme verursachen könnte.

Der Befehl DIR (Directory), der das Inhaltsverzeichnis der Diskette

abruft, ist sehr bequem. Zuerst wird ein Kopf ausgegeben mit Name der Diskette, Anzahl der freien Sektoren und Directoryeintragungen, Anzahl der schlechten Sektoren, Formatierungsdatum und Status (z.B. ob gegen Neuformatierung geschützt oder nicht). Dann werden die Dateien einzeln aufgelistet, mit Dateityp, Status, und Name, und man blättert mit der Leertaste Zeile für Zeile weiter. Benutzt man dabei die Repeattaste, so werden die Angaben sehr schnell aufgelistet, so daß man zügig zu jeder beliebigen Eintragung kommen kann. Nach der letzten Eintragung wird die Liste wiederholt so oft man will, oder man kann mit Cursor-auf oder -ab sofort zur ersten Zeile wiederkehren. Ist die zuletzt angezeigte Datei vom Typ BAS oder UTY, so kann sie mit Cursor-links gleich geladen werden, und bei Basic-Dateien mit Cursor-rechts geladen und gestartet werden. Leider ist es nicht möglich, vom Directory aus Dateien zu löschen usw., aber das ist zum Teil auch eine Schutzmaßnahme.

In der Directory sind für jede Datei noch weiter Merkmale abgespeichert, die man auch ausgeben lassen kann: das sind die Dateilänge, der oben erwähnte 2-Byte Informationsblock (normalerweise die Ladeadresse), die Anzahl der belegten Sektoren, und das Datum der Erstellung der Datei und der letzten Änderung. Auch kann man in Kurzform nur die Eintragungen eines bestimmten Dateityps ausgeben lassen.

Natürlich gibt es auch Befehle, um Disketten zu formatieren (einfache und doppelte Dichte) und um Disketten zu kopieren. Zur Zeit können nur ganze Disketten Spur für Spur kopiert werden, (BACKUP) aber später soll es möglich sein, einzelne Dateien zu kopieren oder eine Diskette unter Zusammenrücken der Dateien zu kopieren (kompaktieren!) Übrigens sind auch diese Operationen sehr schnell--das Formatieren von einer Seite einer 80-Spur Diskette braucht 1 Minute 10 Sekunden und ein Backup einer fast vollen 385 KB-Diskette (nur 9 KB frei) braucht 1 Min 32 Sekunden (eigentlich wollte ich eine ganz volle Diskette kopieren, aber das funktionierte bei zwei Versuchen nicht!)

Diskettenbetrieb ist auch in Kombination mit Kassetten oder MDCR möglich, und es gibt DOS Befehle DISK, CAS, RCAS, WCAS und DCR, die zwischen diesen Möglichkeiten hin- und herschalten. Das DOS enthält eine Kopie des DCR TOS, so daß die mit den DCR's gelieferte Karte für den X-Bus nicht mehr benötigt wird.

Zusätzlich zu einigen weiteren Diskettenbefehlen (z.B. zur Umbenennung von Dateien usw.) enthält das DOS noch einige kleine Hilfsbefehle, die allgemein nützlich sind, z.B. BAS zur Veränderung der Basic-Zeiger, LPRINT zur Druckausgabe auch über den DCE-Bus (oder wahlweise Bildschirm, RS232C, usw.), und FIND zum auffinden von Zeichenketten im Speicher. Schließlich gibt es noch nicht implementierte Befehle als Schnittstellen zu späteren Erweiterungen wie CPM oder eine Echtzeituhr.

Die DOS-Befehlstabelle ist ähnlich aufgebaut wie beim DCR TOS. Insbesondere können DOS Befehle direkt in Basic Kommandozeilen auftreten, und in Basic Programmzeilen können sie benutzt werden in der Form "CALLM #F000: REM Befehl...". Der Zeiger zur Befehlstabelle sitzt im RAM, so daß die Befehlsliste leicht erweitert werden kann. Nur der Aufruf von DOS-Befehlen aus Maschinenprogrammen ist noch etwas umständlich, weil die unumgänglichen Dateinamen zuerst in das KEN-DOS RAM abgelagert werden müssen; dies soll aber in Zukunft geändert werden und ein festes Protokoll zur Übergabe von Zeigern in den CPU-Registern vereinbart werden.

Um zusammenzufassen: KEN-DOS ist ein sehr schnelles und sehr leistungsfähiges, aber noch nicht ganz ausgereiftes Diskettenbetriebssystem für den DAI. Es ist sehr empfindlich gegen Störungen aber dies kann zum Teil (nicht ganz) durch Hardware-Änderungen am DAI behoben werden. Auch die Software funktioniert noch nicht ganz einwandfrei (es gibt auch sicher viele Fehler, die ich einfach noch nicht entdeckt habe!), aber die Softwarefehler werden nach und nach ausgebessert, und dies bisher kostenlos--sonst wäre das System auch nicht annehmbar gewesen. Schon jetzt finde ich es trotz allem Ärger, den ich mit Fehlern hatte, wesentlich angenehmer mit KEN-DOS zu arbeiten als mit Kassetten, und ich hoffe nach Beendigung der Entwicklungsarbeit, die noch im Gange ist, recht zufrieden mit dem System zu sein.

Bochum, den 6. März 1984 Gordon Wassermann

CURSORTOETSEN

In het eerste nummer van DAINAMIC dit jaar stond een prijsvraag waarin gevraagd werd om een routine te programmeren die het volgende doet:

VOORWAARDEN:

1. Alleen de waarden van X en Y veranderd als er een cursortoets ingedrukt is.
2. X of Y vermeerderd of verminderd met 1 als er een cursortoets is ingedrukt.
3. Als er ook een shifttoets is ingedrukt dan wordt X of Y met een door de gebruiker in te stellen integerwaarde verhoogd.

BEOORDELINGSCRITERIA:

1. De correcte werking.
2. De grootte van de routine zowel na een als twee keer gebruiken in bytes.
3. De originaliteit van de routine.
4. De bruikbaarheid en programmeerbaarheid.

ad 1. Van de 45 ontvangen oplossingen voldeden er vier niet aan de gestelde criteria. Aan een aantal andere moest iets worden toegevoegd of weggelaten. Zoals bij mensen die voor het gemak de waarde met shift niet hadden geinitialiseerd of zelfs geen lijnnummmers hadden gegeven.

ad 2. De bij de prijsvraag vermelde routine nam 221 bytes respectievelijk 200 bytes in beslag. Er werd gevraagd naar een oplossing die zo kort mogelijk en het liefst op een regel paste. Er waren toch mensen die met gemak grotere routines schreven, die tot 1300 bytes geheugen gebruikten.

ad 3. De originaliteit van de routines was een van de moeilijkste punten om te beoordelen er zat in veel routines wel een originele vinding uiteindelijk is voor de origineelste methode gekozen.

ad 4. De meest bruikbare en makkelijk te programmeren routines zijn die die met behulp van een array werken. Helaas is dit niet altijd de kortste en zeker niet de origineelste manier. Andere manieren zijn korter maar moeilijker te programmeren.

DE INZENDINGEN:

Hier volgt een korte beschrijving van de gebruikte methodes:

1. Het bepalen of een cursortoets is ingedrukt:

Aangezien het toegestaan was om IF ... THEN .... te gebruiken hebben zeer veel inzenders daar gebruik van gemaakt. De meest gebruikte routine was

```
IF H>15 AND H<24 THEN ..... (19 bytes)
```

Dit is eigenlijk een verkapte verplaatsingsroutine omdat geldt:
IF NOT(H>15 AND H<24) THEN next line ELSE .....
Eleganter waren de oplossingen die zonder IF .. THEN .. werkten. Dat hield dan wel in dat een term 0 of 1 moet worden gemaakt en met het uiteindelijke resultaat moet worden vermenigvuldigd. Voorbeelden van zo'n factor die 1 geven als aan de voorwaarde voldaan wordt:

```
H IXOR #E8/#F8                14 bytes
1-SGN(H IAND #E8)             18 bytes (ook 0- 7 !)
1-SGN(H SHR 3 IXOR 2)         25 bytes (ook 24-31 !)
SGN(SGN(H-16)-SGN(H-23))      29 bytes
```

Het ging bij de bitmanipulatie om bit 7 tot en met 3 die respectievelijk 00010XXX moesten zijn.
Deze methodes zijn allen goedgekeurd voor zover zij vanaf het toetsenbord ook goed werkten. De korste manier was de eerste, deze gebruikte 2 operatoren (IXOR,/ 2 bytes), twee constanten (#E8,#F8 10 bytes) en een variabele (H 2 bytes). Let wel, de SGN functie in combinatie met integers gebruikt conversiebytes INT -> FPT en FPT -> INT!

2. Het bepalen of een SHIFT toets is ingedrukt:

Ook deze informatie kan uit de bits worden afgeleid als bit 2 1 is dan is SHIFT ingedrukt. De eenvoudige manier is testen of H groter dan of gelijk is aan 20.
Zo onstonden de volgende termen bijv:

```
1+H/20*7                          19 bytes
1+(H SHR 2 IAND 1)*7              27 bytes
-(INOT (H/20*7)) zonder mathchip  18 bytes
INOT (-H/20*7) met mathchip       17 bytes
X(H) (array)                       6 bytes
ASC(MID$("JHHJQAAQ",H-16,1))-73   33 bytes
1+(Q SHR (H-16)*4 IAND #F)        31 bytes
```

Hierbij werd dan gelijk de waarde met shift meeberekend. Let op het verschil tussen INOT met mathchip en zonder nu zal dit niet zovaak tot problemen leiden om dat alleen ondergetekende van deze methode gebruik had gemaakt.
De drie laatste methodes zijn gebaseerd op een offset in array of string of numerieke variable. De twee laatste methodes hebben het nadeel dat de stapvariatie zeer klein is.

3. Het bepalen van de richting X of Y.

Deze bewerking leverde bijna niemand problemen op de correcte stand van bit 1 (0 voor Y,1 voor X) werd zeer makkelijk uitgewerkt. Als voorbeelden:

```
Z=(H SHR 1 IAND 1)*V:X=X+Z:Y=Y+V-Z
Z=H/2 MOD 2*V:X=X+Z:Y=Y+V-Z
X=X+X(H):Y=Y+Y(H)                    array
```

De waarden van de toetsdruk (V) vermenigvuldigd met de waarde van bit 1 en gecorrigeerd voor X of Y. Of gewoon opgehaald uit een array.

4. Het bepalen van de richting + of -.

Dit kostte meer inspanning om dit netjes en kort te programmeren. Aangezien 16, 19 ,20 en 23 positief en 17, 18, 21 en 23 negatief resultaat geven, moesten er meerdere bits getest worden. De mensen met een array hadden het niet zo moeilijk. Maar voor anderen was dit de grootste term. Als bit 0 en 1 gelijk zijn is de waarde positief anders negatief. Voorbeelden:

```
INOT(-H IAND 2) met mathchip      11 bytes
(-H IAND 2)-1                     14 bytes
```

De meeste andere manieren waren complexer en inverteerden optellen en aftrekken.

De meest gemaakte fouten

Een veelgemaakte fout bij het werken met bitoperatoren was het feit dat te veel () werden gebruikt. Dit omdat men blijkbaar de prioriteiten niet goed kent. Zo is bijv. (H IAND 2) SHR 1 verkeerd, beter is: H SHR 1 IAND 1. Het artikel van Frank Druijff uit voorgaand nummer nog maar eens goed doorlezen! Ook het gebruik van te veel variabelen of lange namen voor variabelen kost geheugen. Dit is niet bevorderlijk voor een zo kort mogelijke routine.
Er waren programma's bij waarbij de stap zo beperkt kon worden geregeld dat een minimum van tussen 1 en 50 niet mogelijk was.
Er waren velen die termen dubbel gebruikten en twee maal de volledige term in plaats van deze in een variable op te slaan voorwaar verspilling van bytes en computertijd.
Het feit dat toch nog enige routines met INTEGER operatoren geheel in FPT waren geschreven geeft te denken! Dit gaat goed maar het blijft oppassen voor afwijkingen!
Als je na IMP INT nog INT(A/4.0) ziet staan, is er toch nog iemand die de DAI niet vertrouwd.
Als er om een zo kort mogelijke routine wordt gevraagd is het nuttig om de gehele regel (126 gecompileerde bytes) ten volle te benutten!

DE UITSLAG:

De telling geschiedde als volgt: als initialisatie werd de lengte van de SYMBOLTABLE en HEAP-4 (de minimale CLEAR) en het eenmalig inlezen van array's en variabelen geteld. Het deel dat overblijft kan dan iedere keer weer gebruikt worden.
De kortste routines en hun gebruik van bytes voor eenmalig en iedere volgende keer:

```
     124 / 96 bytes met mathchip!!!!!!!                N.P.LOOIJE
10   H=GETC:V=INOT (-H IAND 2)*(H IXOR #E8)/#F8*( INOT (-H/
C     20*7)):H=H/2 MOD 2*V:Y=Y+V-H:X=X+H

     129 / 101 bytes                                   N.P.LOOIJE
10   H=GETC:V=(H IXOR #E8)/#F8*( INOT (H/20*7))*((-H IAND 2
C    )-1):H=H/2 MOD 2*V:Y=Y+V-H:X=X+H
```

```
     130 / 99 bytes                                    J.G.LOOIJE
10   H=GETC:IF H IAND #F8=16 THEN A=ASC(MID$("JHHJQAAQ",H-1
C     6,1))-73:H=H/2 MOD 2*A:X=X+H:Y=Y+A-H

     144 / 105 bytes                                   M.SIGG
10   H=GETC:IF 16<=H AND H<=23 THEN H$=MID$("IJIHHIJIIQIAAI
C     QI",H+H-32,2):X=X+ASC(H$)-73:Y=Y+ASC(RIGHT$(H$,1))-73

     149 / 121 bytes                                   H.PETERS
10   A=GETC:FOR I=1 TO 8 STEP 7:A=A-I/2:Y=Y+(ABS(SGN(A-17))
C     -ABS(SGN(A-16))):X=X+I*(ABS(SGN(A-18))-ABS(SGN(A-19))
C     ):NEXT

     158 / 116 bytes                                   L.MAES
10   G=GETC:E=1:F=(SGN(G SHR PI IXOR 2)-E)*(SGN(G IAND 4)*7
C     +E)*(E-((G IAND E) SHL E)):W=G SHR E IAND E:Y=Y+W*F-F
C     :X=X+W*F

     158 / 127 bytes                                   H.LAMBRECHT
10   H=GETC-16:IF H>=0 AND H<9 THEN S=1-(H+1 IAND 2):X=X+S*
C     VAL(MID$("0108",H/2,1)):Y=Y+S*VAL(MID$("1080",H/2,1))
```

De eerste drie oplossingen zijn buiten mededinging dus de prijs voor de korste routine gaat naar MARCUS SIGG. Dit zijn allemaal oneliners! De oplossing met een "string" zijn regelbaar tussen 1 en 58 bij gebruik van het TAB-pijltje en "^". De andere zijn stappenloos regelbaar.

De origineelstes routine werden ingezonden door:

```
20   H=GETC:H!=H MOD 4                          JOS VANDEBERGH
30   H=H/10*SGN(SGN(H-16)-SGN(H-23))
40   H=H*(3*H-2)
50    D=INT(H!*(1.83333+H!*(0.666667*H!-2.5))+0.1):X=X+D*H
60   D=INT(1.1+H!*(-4.83333+H!*(3.5-0.666667*H!))):Y=Y+D*H
```

De routine van Jos Vandebergh valt op door de originele benadering van het probleem en wel het vervatten in een functie. Het aantal regels is er drie teveel (regel 20, 30, en 40 samen en regel 50 en 60 samen). De stap is ook niet makkelijk te wijzigen, maar Jos wint de prijs voor originaliteit.

```
10   SCH=8                                          MARC GOFFART
20   H=GETC
30   STP=1+(PEEK(#2B0) IAND 64)/64*(SCH-1)
40   Y=Y+((PEEK(#2A9) IAND 64)-(PEEK(#2AA) IAND 64))*STP/64
50   X=X+((PEEK(#2AC) IAND 64)-(PEEK(#2AB) IAND 64))*STP/64
```

Ook de aanpak van Marc Goffart was uniek en het programma heeft als grootste voordeel ook nog een ingebouwde autorepeat! Hij bekijkt de laatste toetsenbordscan en haalt hier de informatie uit. Echt kort geprogrammeerd is het niet. Het vervangen van 64 door een variabele, het inkrimpen tot een regel en het vervangen van (SCH-1) door een constante levert bytes en een bruikbaarder routine op:

```
10   Z=64:H=GETC:STP=1+(PEEK(#2B0) IAND Z)/Z*7:Y=Y+((PEEK(#
C     2A9) IAND Z)-(PEEK(#2AA) IAND Z))*STP/Z:X=X+((PEEK(#2
C     AC) IAND Z)-(PEEK(#2AB) IAND Z))*STP/Z
```

Hieronder volgen nog een aantal routines die ook kort, origineel of in een andere manier opvielen, probeert U zelf de werking eens na te gaan:

```
     142 / 56 bytes !korste routine bij 2 maal te traag!
     N. P. Looije
10   RESTORE:H=GETC:FOR A=16 TO 24:READ P,Q:IF A<>H THEN NE
C     XT
20   X=X+P:Y=Y+Q
40   DATA 0,1,0,-1,-1,0,1,0,0,8,0,-8,-8,0,8,0,0,0

     183 / 125 bytes F. van Amerongen
10   SHFT=10
20   H=GETC:IF H IAND 248=16 THEN H1=(1+(SHFT-1)*((H IAND 4
C     ) SHR 2))*(1-((H IAND 1) SHL 1)):H2=H1*((H IAND 2) SH
C     R 1):Y=Y+H1-H2:X=X-H2

     167 / 83 bytes W. Didier
10   A$="PPOQPPHXQOPPXHPP"
20   H=GETC:IF H IOR #F=31 THEN X=X+ASC(MID$(A$,H-16,1))-80
C     :Y=Y+ASC(MID$(A$,H-8,1))-80

     195 / 149 bytes J. Vandebergh
10   Q=21125
20   H=GETC
30   A=SGN(SGN(H-16)-SGN(H-23))*(1+H/20*7)
40   H=(H IAND 3) SHL 1:X=X+A*((Q SHR H IAND 3)-1)
50   Y=Y+A*((Q SHR 8 SHR H IAND 3)-1)

     179 / 305 bytes F. Verstegen
10   N=1
20   H=GETC
30   H=H-16:IF H IAND #FFFFFFF8=0 THEN B=(H IAND 2) SHR N:H
C     =B IXOR H:Z=(H/4*7+N)*(((H IAND N IXOR N) SHL N)-N):Y
C     =Y+Z*(B IXOR N):X=X+Z*B

     121 / 86 bytes F.H. Druijff !alleen stap ^10.0!
20   H=GETC IXOR 16:IF H<8 THEN C!=COS(H*PI)*ALOG(H SHR 2):
C     A!=(H IAND PI)/2*C!:X!=X!-A!:Y!=Y!+C!-A!

     127 / 99 bytes P. Gobert !alleen stap 8!
10   K=GETC:E=1:E=E-((K IAND E) SHL E) SHL (-((K SHR 2 IXOR
C     5)-E)*PI):K=E*(K SHR 1 IAND 1):Y=Y-K+E:X=X-K

     178 / 61 bytes F.H. Druijff
10   DIM V(7):FOR H=0 TO 5:READ V(H):NEXT
20   DATA 1,-1,0,0,10,-10
30   H=GETC IXOR 16:IF H<8 THEN X=X-V((H+6) MOD 8):Y=Y+V(H)
```

Hierbij feliciteren wij Marcus Sigg, Jos Vandebergh en Marc Goffart met hun overwinning en danken bovengenoemden en verder:
Christian Lequesne, Marc de Droog, Ph. Wanet, E. Ysebaert, Robert Bruijninckx, Mark Vanbaelen, Jan Boerrigter, G. van de Ven, Dirk de Boeck, Dieter Rumpenhorst, Lieven Heuninck, G. Doumont, Guy Topff, Pieter van der Hijden, J-P Mallien, Marc Nemegeer, Theo van Ooik, E Waeytens en Stefan Gogaert.

De winnaars krijgen een softwarepakket naar keuze en verder krijgen alle inzenders een reduktiebon voor DAINAMIC software.

Nico Looije.

DOS Toolkit

DAInamic software

Contains :

1. Extension for DOS 1.0 featuring random access whith OPEN, CLOSE, GET and PUT commands.

2. One-key commands. You are able to enter BASIC keywords and DOS commands whith one keystroke in command mode as well as in EDIT mode.

3. A Disk Operating System combining the features of previous programs.

4. ZAP. This program ables you to examine and edit the contents of a floppy disk. The display gives you the ASCII and Hexadecimal representation of the contents of one sector. A special directory is included to locate easily files on diskette.

5. SPLIMPLM. This is a source for SPL. After assembling this sourcefile SPL reads and writes his sourcefiles whith the extension SPL to avoid confusion whith BIN files.

NOTE : This toolkit is exclusively for 2 X 80k drives.
---- Delivered on diskette.

Dataflex

DAInamic software

Contains :

1. DATAFLEX. Random access databaseprogram exploring the maximum capacity of your 2 X 80k floppy-drive.
Includes editor, searcher, relative report generator, relative label generator, quick-sort, compactor, deletor and MARK-generator.

2. CREATOR. Dataflex file-creator.

3. REDACTIE.RND. Sample data-file.

4. SHAPE.INT. Sample label-shape.

NOTE : This programs and files are exclusively for 2 X 80k floppy-drives

MIX 1

Een nieuw soort collectie van DAInamic.

Bij de vele inzendingen, die wij binnenkrijgen, zitten regelmatig programma's, die eigenlijk nergens bij thuishoren. Het zijn geen echte spelletjes en ook geen echte TOOLKIT-programma's . Vaak zijn het best leuke, aardige, nuttige programma's. Zij zijn echter wel erg smaakgebonden. De een zal bv nummer drie al de prijs van de hele collectie waard vinden, terwijl een ander nummer drie, direct na ontvangst,gaat uitwisen om zo meer band tot zijn beschikking te hebben Soms is ook niet meer bekend wie de oorspronkelijke auteur van het betreffende programma is. Het is dan ook beter dit in het programma zelf te vermelden bij inzending. Programma's zijn kritiekloos in de collectie opgenomen, per slot van rekening hoeft onze smaak niet de uwe te zijn. Later zullen nog meer van deze collecties volgen.

c o n t e n t s

A mixture of all kinds of programms.
See list above. We do hope you do enjoy them.

Quest

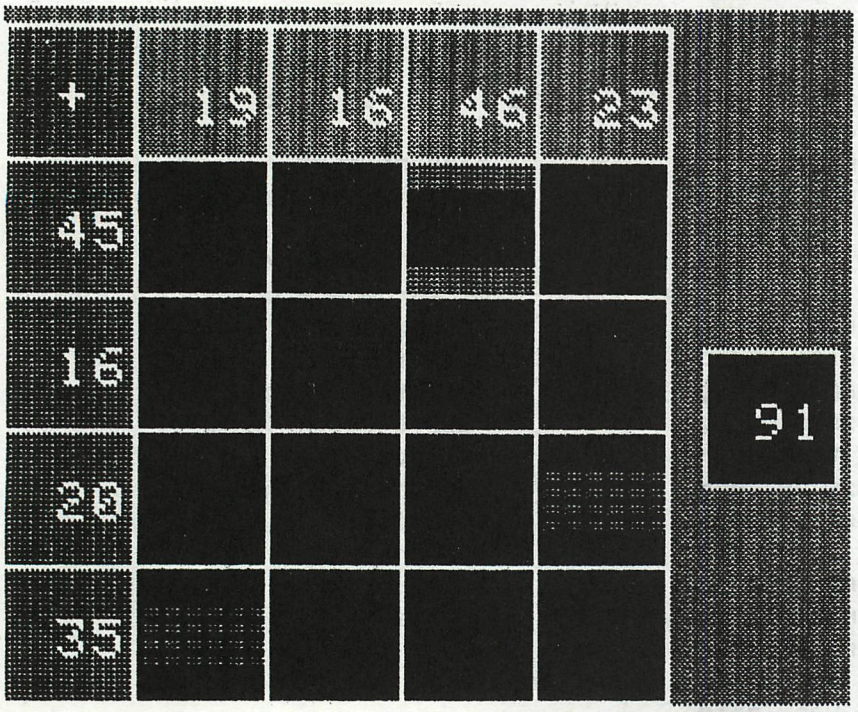
Een tekstaventurenspel. De avonturier moet proberen via instructies , de problemen, die hij op zijn reis tegenkomt op te lossen. Daarbij moet hij wel zoveel mogelijk schatten zien te verzamelen. U kunt het spel afbreken en op een later tijdstip doorgaan met de positie waar U de vorige keer in verkeerde. De gegevens kunnen nl op band bewaard blijven. Volgens de auteur zijn de problemen relatief simpel op te lossen. Engelse tekst

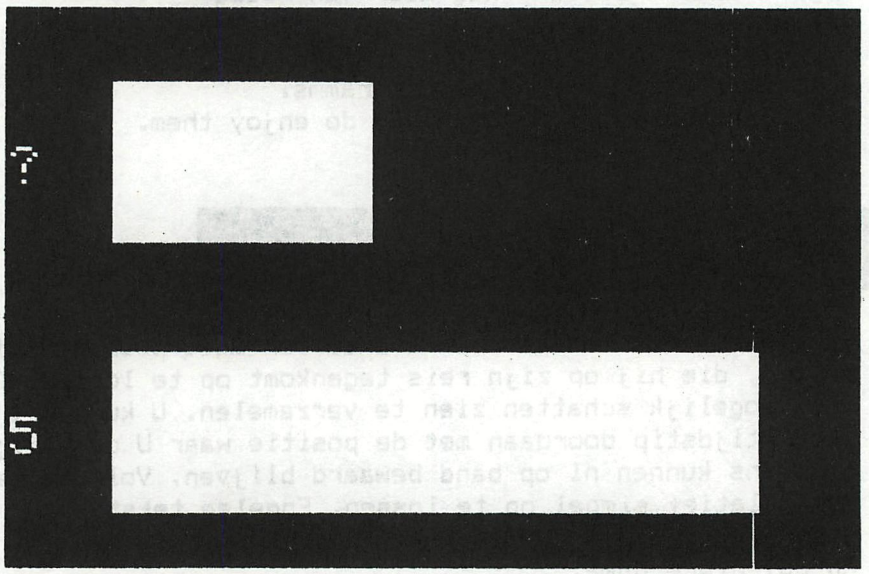
An adventure-game.
Accordingly to the author are all problems relatively simple to solve.
Instructions are given in english.

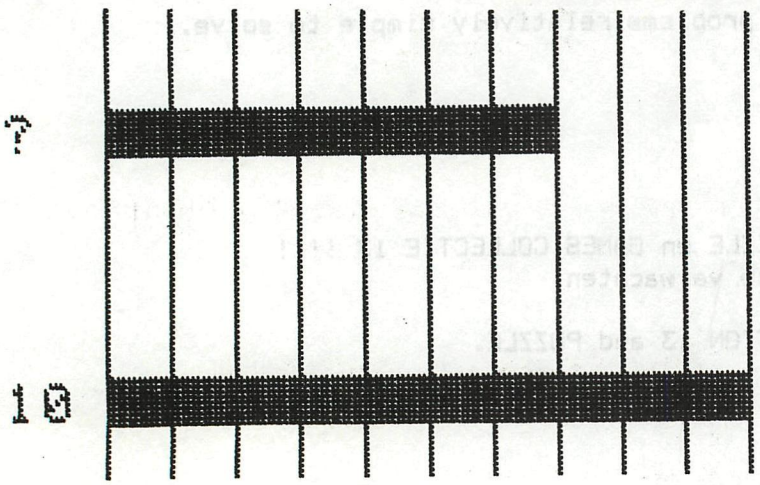
Haast klaar zijn de collecties PUZZLE en GAMES COLLECTIE 13 !!!!
Het volgend tijdschrift kunt U deze verwachten.

Almost ready the new GAMES COLLECTION 13 and PUZZLE.
You can expect them next issue.

Math' fun 2







Another collection of fun-to-play & easy-to-learn math programs for children from 7-12 years old. Including : rekenmatrix (math-matrix) , schatten (estimate) and H.T.E. (hundred-ten-one).

CATALOG

DAInamic software

title	audio	dcr

RECENT TITLES

Games 12	750	900
Math'fun 1	1000	1150
Bits & Bytes	750	900
Character Generator	1750	1900
FWP	2000	2150
DAYLAXIANS	2100	2100
PUZZLY	2100	2100
DUEL	2100	2100
C.L.I.O.	3000	3000

Gestructureerd programmeren met DAI-BASIC 1100

NEW TITLES

MIX 1	500	650
QUEST adventure	600	750
Sociale Geografie	1000	1150
D-BASIC	2000	2150
Math'fun 2	1000	1150
DOS-TOOLKIT	2 x 80K : 1250	
DATAFLEX	2 x 80K : 1500	

SOON AVAILABLE

Frogger

The long-awaited ARCADE-classic also on you DAI-computer.
Lead the froggs across the road and the dangerous river...

Eagle

A very original 100% machine language space game with many levels and beautiful features...

Toolkit 6

Another collection of very useful routines for programmers..

Turtle-Basic

All the commands build-in in BASIC to learn and play with turtle-graphics on your screen.Turtle-BASIC has also some new extensions for BASIC : LLIST,LPRINT,HOME,REPEAT UNTIL, WHILE-WEND,DOKE,DEEK,SWAP .....

DAI graphics

SEND YOUR DRAWINGS TO DAINAMIC EDITOR

SOFTWARE ORDER CARD 1-6-84

NAME : .....

ADRES : .....

.....

☐ find cheque enclosed

☐ I pay with international postal money order

☐ I pay with transfer on

no 401-1009701-46 of Kredietbank Herselt

Send this card no : DAInamic software & library
c/o W. Hermans Mottaart 20
B-3170 HERSELT

Gelieve volgende pakketten op te sturen :
Please send following packages :

title	audio	DCR
Games 12	☐ 750	☐ 900
Math'fun 1	☐ 1000	☐ 1150
Bits & Bytes	☐ 750	☐ 900
Character Generator	☐ 1750	☐ 1900
FWP	☐ 2000	☐ 2150
DAYLAXIANS	☐ 2100	☐ 2100
PUZZLY	☐ 2100	☐ 2100
DUEL	☐ 2100	☐ 2100
C.L.I.O.	☐ 3000	☐ 3000

title	audio	DCR
MIX 1	☐ 500	☐ 650
QUEST adventure	☐ 600	☐ 750
Sociale Geografie	☐ 1000	☐ 1150
D-BASIC	☐ 2000	☐ 2150
Math'fun 2	☐ 1000	☐ 1150
DOS-TOOLKIT	☐ 2 x 80K : 1250 (on diskette)	
DATAFLEX	☐ 2 x 80K : 1500 (on diskette)	

☐ GESTRUCTUREERD PROGRAMMEREN MET DAI-BASIC : 1100

date : .....

TOTAL : .....

all prices in Bfr.

signature : .....