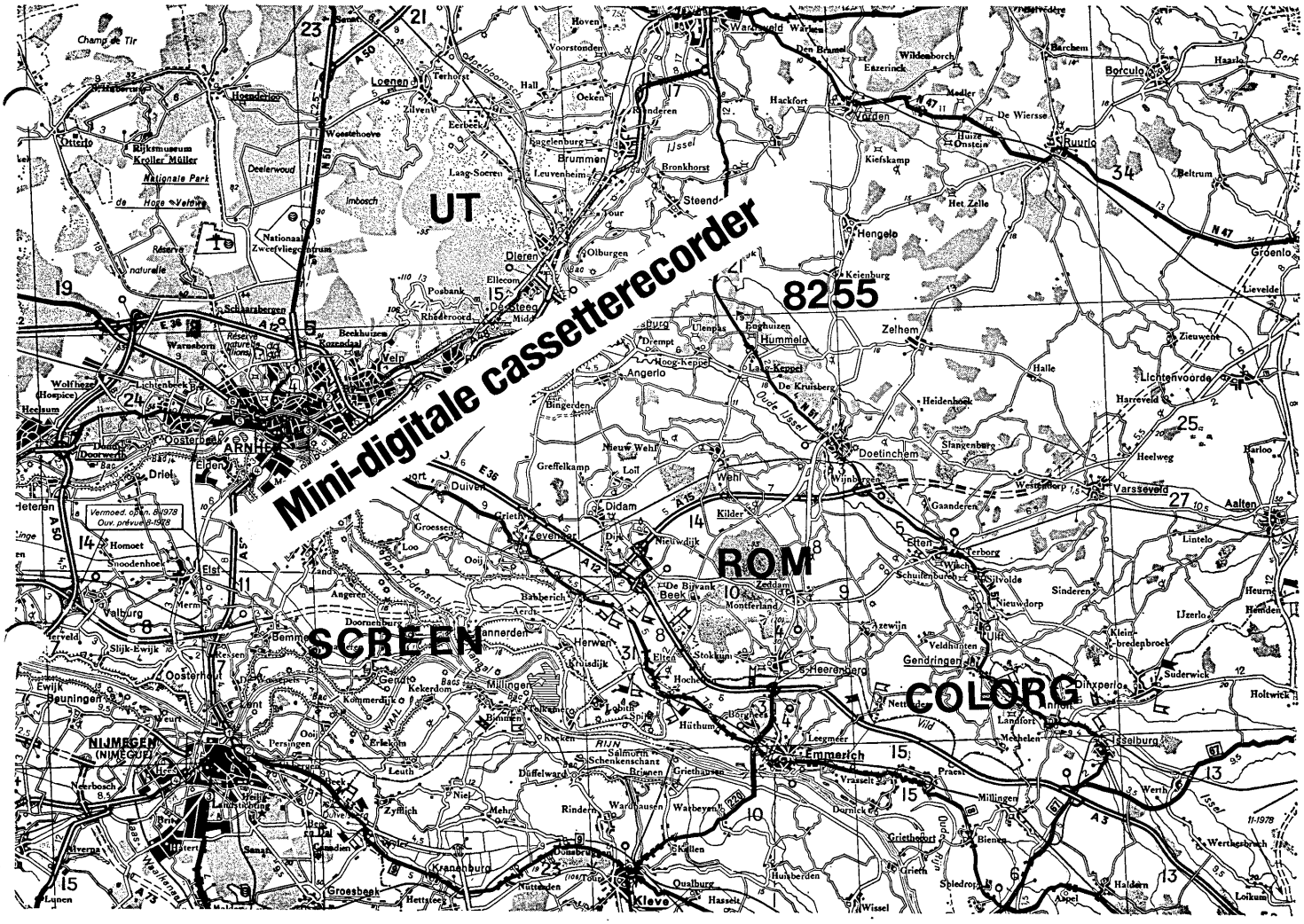
DAI NAMIC

r 6 JULI—AUGUSTUS 1981



GEDRUKTE PERIODIEK verschijnt tweemaandelijks

Verantw. Uitgever : W. HERMANS HEIDE 98 3171 WESTMEERBEEK

COLOFON

DAInamic verschijnt tweemaandelijks.
abonnementsprijs is inbegrepen in de jaarlijkse contributie:
750 Bfr 50 Gld 50 Dm
Bij toetreding worden de verschenen nummers van de jaargang toegezonden.
DAInamic redactie:
Dirk Bonné
Freddy De Raedt
Wilfried Hermans
Jules Meulenbergs
Jos Schepens
Roger Theeuws
Bruno Van Rompaey
Jef Verwimp
vormgeving :Ludo van Mechelen

U wordt lid door storting van de contributie op nr406-3016141-33 van KREDIETBANK WESTMEERBEEK,via bank-instelling of POSTGIRO.
Abonnement loopt van januari tot december.

U kan telefonisch contact nemen op nr 016/698623.

---

correspondentieadres:
DAInamic
Heide 98
3171 WESTMEERBEEK BELGIE

DAInamic verschijnt de eerste week van de pare maanden.
Bijdragen zijn steeds welkom.

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

DAInamic
DAI USERS CLUB

Westmeerbeek augustus 81

Beste DAInamic-leden,

De verscheidenheid aan lettertypes in dit vacantienummer toont aan dat vele leden zich de moeite getroosten om eens een leuk idee of programma op papier te zetten. Dit is een zeer gezonde evolutie , hartelijk dank aan alle nabije of verre medewerkers. Door het steeds stijgende aantal nederlandsonkundige leden doen we ons best om o.a. ook engelse en duitse bijdragen te publiceren. Dat dit gepaard gaat met de onvermijdelijke spel-& taalfouten neemt U ons hopelijk niet kwalijk. Dank zij het stijgende ledenaantal en de programmaverkoop kon de club zich een aantal investeringen veroorloven die in de toekomst de kwaliteit van ons blad zeker zullen beïnvloeden: een copiëermachine, een reprocamera (vergroten, verkleinen. rasteren dus fotografisch werk in de toekomst) en een mastermachine voor offset. Ondertussen begint ook de huiskamer te klein te worden zodat er plannen zijn voorgesteld om ergens een lokaal te plaatsen waar de DAInamic werkzaamheden kunnen plaatsvinden. Dit zal ook de mogelijkheid bieden om op bepaalde data open vergaderingen, studiedagen en cursussen in te richten. We hebben de gelegenheid gehad om gedurende 2 weken intensief de digitale cassette te testen , besluit : 100 % voldoening. We hopen spoedig hetzelfde te kunnen melden ivm de discdrive van DAI. Op 28 novemben gaat de jaarlijkse HCCdag door, deze keer te UTRECHT. Omdat de afstand nogal redelijk is willen we de verplaatsing maken met een autocar, Belgische leden die willen meereizen gelieve contact op te nemen met de redactie. Vermoedelijke opstapplaatsen: Aarschot, Westmeerbeek, Herentals en Antwerpen. In dit nummer vindt U de beloofde ledenlijst, een reuze-memorymap, besprekingen van de software-bibliotheek en meer interessante artikelen. Volgende keer zullen we het uitvoerig hebben over spraaksynthese , we publiceren een 8080-simulator programma, een TALK-editor en nog meer moois.

veel leesgenot, we verwachten uw vacantiecreaties...

de redactie

BLADWIJZER

131	REMARK	Redaktiepraatje
132	BLADWIJZER	
133	LOOK	Standaard Subroutine : BASIC CALL OF FGT
134	LOOK	"
135	LOOK	Automatisch laden van Object Fgt in een BASIC programma
136	LOOK	BASIC plus FGT automatisch laden, maar dan nog sneller ...
137	LIST	FGT-Object MOVE ROUTINE
138	READ	Boekbespreking "8080A/8085 Assembly Language Programming"
139	TALK	Notities over de IFIP konferentie in Lausanne
140	TALK	"Computers in het Onderwijs"
141	LOOK	DAIpc MEMORY MAP
142	LOOK	"
143	LOOK	"
144	LOOK	"
145	LOOK	"
146	LOOK	"
147	LOOK	"
148	LOOK	"
149	LOOK	"
150	LOOK	"
151	LIST	Wilhelmus
152	READ	Duitse bijdrage
153	LIST	"TUERME VONHANOI met Problemlösung"
154	LIST	"
155	LOOK	DAI 24 x 60 Character display
156	LIST	Grafieken van 5de graads polynomen
157	LOOK	De beloofde ledenlijst "DAINAMIC-LEDEN"
158	LOOK	"
159	LOOK	"
160	LOOK	"
161	LOOK	"
162	LOOK	"
163	LOOK	"
164	LOOK	"
165	LOOK	"
166	CATALOG	De software bibliotheek
167	CATALOG	Game paddles
168	CATALOG	Games collection 1
169	CATALOG	"
170	CATALOG	"
171	CATALOG	Games collection 2
172	CATALOG	"
173	CATALOG	Games collection 3
174	CATALOG	"
175	CATALOG	"
176	CATALOG	Games collection 4
177	CATALOG	Didactics in mathematics
178	CATALOG	"

****

'BASIC Call of FGT'

De Subroutine 'BASIC Call of FGT' beoogt een vereenvoudigde en gestandaardiseerde aanroep van de Assembly routine FAST GRAPHICS TEXT. De gebruikte namen zijn beter in overeenstemming met de funktionele betekenis van de parameters, en daardoor gemakkelijker te onthouden. Uit snelheids overwegingen zijn uitsluitend Integer variabelen toegepast:

XG, YG } positie v.h. linker beneden hoekpunt v.h. eerste Tekst karakter van de te tekenen String in Scherm-coördinaten volgens de gekozen MODE. (Links onder komt overeen met: 0,0);

G$ Bevat de te tekenen Tekst String;

TC Text Color (Tekst Kleur 0...23);

TH Text Height (Tekst Hoogte 0..15); abs. grootte wordt bepaald door de MODE. Indien > 0 , wordt dubbele lijndikte genomen.

HS Horizontal Space (Afstand tussen de voorzijden van 2 opeenvolgende letters);

VS Vertical Space (Afstand tussen de onderzijden van 2 boven elkaar staande Tekststrings);

VF Vertical Flag (0 = Horizontaal; 1 = Vertikaal);

DF Direction Flag (0 = pos. X-ri ('v.l.n.r.') c.q. pos. Y-ri i.g.v. VF = 1);
(1 = neg. X-ri ('v.r.n.l.') c.q. neg. Y-ri i.g.v. VF = 1);

PF Position Flag (0 = start op pos. gegeven door XG,YG; 1 = plaats de Tekst direkt achter de voorafgaande (als ';' in PRINT));

BC Background Color Number (Volg nummer (0..3), overeenkomend met de 4 door COLORG gedefinïeerde kleuren. Definieert achtergrond kleur bij vooraf wissen v. Tekstveld i.g. BF = 1);

BF Background Flag (1 = Wis vooraf de achtergrond v.d. Tekst string met Kleur bepaald door BC, en grootte gegeven door de waarden van HS en VS);

N.B. Als een Tekst te lang is voor één regel, wordt automatisch (op een afstand VS) op de volgende regel doorgegaan, en wel vanaf de zelfde X-positie (resp. Y-pos., i.g.v. VF = 1) als waarop het eerste Tekst karakter werd geplaatst. (Door HS = XMAX te kiezen, kunnen zo teksten met letters vertikaal boven elkaar worden geschreven!)

De - op bovenstaande Variabelen namen gebaseerde - Standaard Subroutine heeft als titel: 'BASIC Call of FGT', en heeft de volgende gedaante:

```
49999   REM  Init. and Call of the Assembly-Subr. 'FGT'
50000   CQQ = BC x = 40 + TC: FQQ = BF x = 80 + PF x = 40 + DF x # 20 + VF x = 10 + TH
50010   POKE = 2F0,CQQ: POKE = 2F1,FQQ: POKE = 2F2,XG MOD 256
50020   POKE # 2F3,XG SHR 8: POKE # 2F4,YG: POKE # 2F5,HS
50030   POKE # 2F6,VS: CALLM # 300,G$: RETURN
```

Deze routine zal in het algemeen, noch wat Variabele namen, noch t.a.v. Lijnnummers, een konflikt met bestaande BASIC programma's vormen, zodat ongewijzigde overname op de bekende wijze kan worden bewerkstelligd:

- Zorg voor voldoende ruimte in de HEAP buffer, b.v. door het commando CLEAR 1000 te geven, en plaats de Subr. tijdelijk in de EDIT buffer:

```
* CLEAR 1000
* LOAD            (lees 'BASIC Call of FGT' in)
* EDIT 49999-
  BREAK  BREAK
* NEW
* LOAD            (lees BASIC hoofdprogramma in, of typ het in)
* POKE # 135,2    (kombineer met 'BASIC Call of FGT')
* SAVE " ..."     (SAVE kombinatie onder gewenste naam)
```

Nadat Obj. FGT + Pointers is ingelezen, kan nu in iedere MODE ≠ 0 van FGT gebruik gemaakt worden d.m.v. een GOSUB 50000; uiteraard nadat de juiste waarden voor de van belang zijnde parameters zijn ingevuld.
(Hoe Obj. FGT op eenvoudige wijze samen met het BASIC programma kan worden ge-SAVED en weer ingelezen, vindt U in het verhaal op de volgende bladzijden.)

- - - - - - - - - - - -

... IF YOU ACCIDENTLY HIT 'RESET'...

Onder dit kopje wordt in het DAI Manual beschreven hoe men een BASIC prog. kan 'terug vinden' nadat per abuis op RESET is gedrukt.
Dit werkt echter alleen als er geen veranderingen zijn geweest nadat men de inhoud van de pointers # 29B tot # 2A4 heeft genoteerd... Bij het uittesten van een programma zal dit vrijwel steeds het geval zijn; vandaar de volgende suggestie:
Neem direkt a/h begin van ieder programma - maar nà een evt. CLEAR - de volgende Statement op:

```
10  FOR I = 0 TO 9: POKE # 45+I,PEEK(# 29B+I): NEXT
```

Na een 'Crash' volstaat dan: >M45 4E 29B (in UT mode) om het BASIC programma terug te vinden.

Automatisch laden van Object FGT in een BASIC programma.

Het is goed mogelijk om FGT Object samen met het Hoofdprogramma waarin het moet worden gebruikt, te SAVEN, zodanig, dat het daarna steeds automatisch mee komt bij het laden van het BASIC Programma.
Hiertoe behoeven aan het BASIC Hoofdprogramma slechts enkele PEEK en POKE Statements toegevoegd te worden, en moet Object FGT eenmalig in de SYMBOL-table worden ondergebracht. De procedure hiervoor is als volgt:

1. Hard RESET
2. Laad BASIC Hoofdprogramma. Wijzig dit zonodig, maar voeg in elk geval direkt aan het begin de volgende Statements toe:

```
10  IF PEEK(#29C)=9 THEN CLEAR 1000: GOTO 50
20  CLEAR #700: POKE #29C,9: POKE #29E,1: CLEAR 1000
30  ES = PEEK(#2A3) + PEEK(#2A4)*256 - #65B - #2A5
40  FOR I = #2A5 TO #8FF: POKE I,PEEK(ES + I): NEXT
50  ...
```

Verklaring:

10: Skip verplaatsing, etc. als Obj FGT al aanwezig is.
20: Schuif het BASIC Programma op, update HEAP Start en Size; zorg voor een echte CLEAR v. 1000 (mag ook een andere waarde zijn...).
30: Bepaal eerste te verhuizen Byte v. Obj FGT in SYMBOL table.
40: Verplaats Obj FGT naar werkgebied.

3. SAVE het aldus gewijzigde BASIC Programma.
4. Hard RESET
5. Ga naar Utility mode, en
   lees Object FGT in ('R' commando).
6. Verplaats Object FGT naar een hoog vrij adres, b.v.:
   >M 2A5 8FF 92A5
7. Hard RESET
8. Laad BASIC Hoofdprogramma opnieuw.
9. Verplaats in Utility mode Object FGT als volgt:
   >M 92A5 98FF xxyy (xxyy = inhoud van End-of-SYMBOL table ptr: #2A4,2A3)
10. Update End-of-SYMBOL table ptr:
    [#2A4,2A3] := [#2A4,2A3] + #65B (lengte Object FGT)
11. SAVE BASIC Programma op normale wijze m.b.v. 'SAVE' commando vòòrdat 'RUN' is gegeven!

Vanaf nu kan het betreffende programma normaal als BASIC programma worden ingelezen en gestart. Object FGT komt automatisch mee..!

D. Assink
81-06-24

BASIC plus FGT automatisch laden, maar dan nòg sneller...

Daar het verplaatsen van Object FGT m.b.v. PEEK's en POKE's nog altijd zo'n 8 sekonden vergt, volgt hier een alternatief met een in Assembly geschreven MOVE Routine. De procedure verandert daardoor niet wezenlijk, alleen moeten uiteraard enkele adressen aangepast worden, en moet ook behalve Object FGT, de MOVE Routine in de SYMBOL table worden ondergebracht:

1. Hard RESET
2. Laad het BASIC Hoofdprogramma. Wijzig dit zonodig, maar voeg in elk geval direkt aan het begin de volgende Statements toe:

```
10  IF PEEK(#29C)=9 THEN CLEAR 1000: GOTO 50
20  CLEAR #700: POKE #29C,9: POKE #29E,1: CLEAR 1000
30  ES = PEEK(#2A3) + PEEK(#2A4)*256 - #922
40  FOR I = #900 TO #921: POKE I, PEEK(ES + I): NEXT
50  ...
```

Verklaring:

10: Skip verplaatsing, etc. als Obj FGT al aanwezig is.
20: Schuif het BASIC Programma op, update HEAP Start en Size; zorg voor een echte CLEAR v. 1000 (mag ook een andere waarde zijn..).
30: Bepaal eerste te verhuizen Byte v. 'Object FGT MOVE Routine' (hierna te noemen: MOVE) in de SYMBOL table.
40: Verplaats MOVE naar werkgebied (#900 .. #921)

3. SAVE het aldus gewijzigde BASIC Programma.
4. Hard RESET
5. Ga naar Utility mode, en
   lees Object FGT in ('R' commando);
   lees MOVE in.
6. Verplaats Object FGT + MOVE naar een hoog vrij adres, b.v.:
   >M 2A5 921 92A5
7. Hard RESET
8. Laad BASIC Hoofdprogramma opnieuw.
9. Verplaats in Utility mode de kombinatie Object FGT + MOVE als volgt:
   >M 92A5 9921 xxyy (xxyy = inhoud van End-of-SYMBOL table ptr: #2A4,2A3)
10. Update End-of-SYMBOL table ptr:
    [#2A4,2A3] := [#2A4,2A3] + #67D (lengte Obj FGT + MOVE)
11. SAVE BASIC Programma op normale wijze m.b.v. commando 'SAVE' vòòrdat 'RUN' is gegeven!

Vanaf nu kan het betreffende programma normaal als BASIC programma worden ingelezen en gestart. Object FGT lijkt dan onmiddellijk aanwezig...

D. Assink
81-06-25

FGT-Object MOVE ROUTINE

```
*
* USE: CALLM #900
*
* Routine to be called from a BASIC
* Program to move the FGT Obj. Code
* from the SYMBOL TABLE to its Work
* Area (#2A5 - #8FF).
*
          ORG   :900
        *
START     PUSH  B           SAVE REGS
          PUSH  D
BEGIN     EQU   :2A5        START OF FGT
END       EQU   :922        END OF MOVE
LNGFGT    EQU   START-BEGIN  LENGTH OF FGT
LNGTOT    EQU   END-BEGIN+1  TOTAL OBJ LENGTH
        *
          LXI   B,LNGTOT    B,C := TOTAL LENGTH
        *
          LDA   :2A3        GET END-OF-SYMBOL
          SUB   C           TABLE; SUBTRACT
          MOV   L,A         TOTAL LENGTH OF
          LDA   :2A4        OBJECT. STORE
          SBB   B           INTO H,L AS A
          MOV   H,A         FETCH POINTER.
        *
          LXI   B,LNGFGT    LENGTH OF FGT
          LXI   D,BEGIN     D,E := STORE PTR
        *
LOOP      MOV   A,M         GET A BYTE
          STAX  D           STORE IT
          INX   D           TALLY STORE PTR
          INX   H           TALLY FETCH PTR
          DCX   B           ALL DONE?
          MOV   A,C
          ORA   B
          JNZ   LOOP        NO
        *
          POP   D           RESTORE REGS
          POP   B
          RET
          END
```

D. Assink
81-06-25

boekbespreking

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%   " 8080A / 8085  Assembly  Language  Programming  "     %
%                                                          %
%      door  Lance A. Leventhal                            %
%      uitgeverij Osborne & Associates, Inc. 1978          %
%      P.O. box 2036, Berkeley, California 94702           %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

In de D.A.I. zit een 8080A microprocessor, welke op verschillende manieren rechtstreeks aanspreekbaar is.

We zien even voorbij aan de halfcompilerende Basic, die standaard in de machine wordt meegeleverd, maar we bekijken via het boek van Leventhal de mogelijkheden die ons resten.

Leventhal geeft in zijn boek een groot aantal standaardroutines in machinetaal, zowel als in assembly, welke laatste met behulp van een assembler-editor-loader zonder problemen ingetypt kunnen worden.

Voor zo een assembler-programma kunt u bij DAInamics terecht, maar het werk wordt interessanter als u direct in machinetaal gaat werken.

De verdienste van Leventhals boek is, dat het je stap voor stap door deze ingewikkelde materie leidt, zonder dat je het spoor bijster raakt, maar ook zonder dat hij de leerstof versimpelder voorstelt, dan striktgenomen noodzakelijk is.

Soms is het dan wel noodzakelijk, om de Engelse tekst meerdere keren over te lezen, maar om het je gemakkelijk te maken zijn de belangrijke teksten dik gedrukt.

Er wordt zeer uitgebreid ingegaan op de 8080instructieset en het geheel wordt geillustreerd met duidelijke diagrammen.

De programmavoorbeelden worden steeds begeleid door stroomdiagrammen, het bronprogramma in mnemonics en het objectprogramma in hexadecimalen.

De bijbehorende uitleg laat verder niets te wensen over.

De D.A.I.-gebruiker van dit boek, moet echter de hexadecimale adressen aanpassen: elk objectprogramma begint hier op #00, terwijl je die programma's het beste kunt intypen in de D.A.I. op adressen vanaf #300.

Vanzelfsprekend is dit probleem op te lossen als u voor elk 'memory adress' steeds een 3 of hoger typt.

Elk programma wordt afgesloten met 'JUMP HERE' welke

instructie problemen oplevert, als niet is aangegeven waar 'HERE' is.

Denk dus niet dat de software fout is, als de D.A.I. op tilt gaat.

Een assembler-loader zal dat probleem ongetwijfeld vanzelf oplossen maar een beginner raakt gauw teleurgesteld, als hij voor de eerste keer in UTility meteen al niks voor elkaar kan krijgen.

Als u een en ander een beetje aardig organiseert, kunt u op deze manier een subroutinecollectie in machinetaal aanleggen, welke u via CALLM in samenwerking met de Basic kunt gebruiken.

Mocht het onverhoopt dan niet werken, dan geeft de auteur aanwijzingen, hoe men de fouten in de software kan opsporen.

"8080A / 8085 Assembly Language Programming" is voor een D.A.I.-gebruiker een uitstekend naslagwerk, waarmee ook de beginner zich niet reddeloos verloren hoeft te voelen.

Bij deze aangeraden dus.

inno broekman

TALK

o

DAINAMIC 558-81

WCCE 81 lausanne

COMPUTERS IN HET ONDERWIJS
notities over de IFIP konferentie in Lausanne
27-31 juli 1981

Ruim 1200 deelnemers, zeker 90% van het mannelijk geslacht, bezochten de WCCE'81 konferentie. De UNESCO en het Intergouvernementaal Bureau voor Informatica hadden het mogelijk gemaakt dat ook een aantal mensen uit de zogenaamde derde wereld landen konden deelnemen. Niettemin lag de verhouding 'noord-zuid', alsook die tussen 'oost' en 'west' behoorlijk scheef en in het voordeel van 'noord-west'.

De konferentie bestond uit een honderdtal lezingen en ruim twintig panel diskussies. Telkens vonden er zo'n zes bijeenkomsten tegelijertijd plaats. Bovendien waren de wandelgangen vol aktiviteit en kon zeker ook de tentoonstelling van apparaatuur niet overgeslagen worden.

De hele konferentie was een kaleidoskoop van wat er zo allemaal gaande was. Van geklungel tot professionele aanpak, van enthousiast teamwerk dat tot degelijke en fraaie resultaten leidde, tot dure volksverlakkerij. Moeilijk om enkele flitsen uit dit alles te lichten.

Computers in het onderwijs werd op verschillende wijzen opgevat. - computers bij het plannen en beheren van het nationale onderwijssysteem, bij het produceren van leermiddelen. - de opleiding in de computer vakken binnen de diverse onderwijs typen. - het gebruik van de computer binnen de verschillende vakken. Ook het onderwijs aan gehandicapten kwam aan bod. De nieuwe audio-visuele hulpmiddelen (videodisk, videolongplay en microfiches) kregen aandacht. Een aantal sessies was bovendien gewijd aan de ontwikkelingslanden. Waarom dat laatste zo was was niet geheel duideijk en ook een aantal mensen uit die landen zelf wilden dat wel eens uitgelegd zien. Een Braziliaan gaf aan dat het onjuist is dat er vanuit gegaan wordt dat in de ontwikkelingslanden alleen de lagere funkties op computergebied vervuld moeten worden. Een Senegalese klaagde dat de Amerikaanse firma's aan Afrika tegen veel hogere prijzen leveren dan aan Europa. De aap kwam aardig uit de mouw van Uncle Sam die krokodille tranen begon te huilen over het feit dat er ontwikkkelingslanden waren die meenden een eigen mening te kunnen hebben en dat dat IBM in de ogen stak.

Ook DAI en DAInamic hadden een stand op de tentoonstelling. Een prototype van de diskette eenheid was er te zien. Helaas niet in werking. In septembeer is dit apparaat leverbaar.

De zwitserse PTT is ook met viewdata begonnen. Dit gaat nogal ingewikkeld ii verband met de drie talen van het land. Vele takken van de 'zoekboom' zijn nog leeg.

In Engeland hebben enkele teams veel interaktieve programma's voor het gebruik (middels microcomputers) op middelbare scholen gemaakt. Zo'n 70 programma's heeft men al. Pakketten met handleiding voor de leraar en uitleg en werkbladen voor de leerlingen zijn leverbaar voor het onderwijs in de wis-, natuur- en scheikunde, geografie en ekonomie. Een lokkertje was het 'windmill location game'. Fraaie kleuren graphics; dat moet niet moeilijk op de DAI te programmeren zijn. Helaas was van dit programma de software nog niet vrij gegeven. De software die wel te koop is, is overigens voor het merendeel nog gebaseerd op een teletype als terminal. De interaktie is daardoor nogal schamel. Maar het denkwerk dat er achter zit, is zeker bruikbaar.

Vanaf januari 1982 start de BBC met een introduktie kursus over computers. De kijkers kunnen voor een paar honderd engelse pond al een aardige microcomputer in huis krijgen. Het is een speciale versie van de ACORN computer. De kleuren en grafische en geluidsmogelijkheden doen onder voor de DAI. BASIC kan echter wat meer, zoals 'while' ... 'do' konstrukties en een 'eval' funktie. Met deze laatste kan men een expressie die in een string staat opgeslagen laten evalueren. Een aardige uitbreidingsmogelijkheid is een teletekst converter. Een apart tv toestel voor de ontvangst van teletekst heeft men dan niet meer nodig. De BBC gaat trouwens haar software via teletekst distribueren.

De Apple III was nauwelijks te zien, de Apple II echter volop. Van de tentoongestelde merken kon er geen in prijs/prestatie verhouding de DAI verslaan. Ook Atari niet en evenmin de Volkscomputer, waarvan de karakters en de kleurren belabberd slecht waren.

De teksten van de lezingen zijn inmiddels al gepubliceerd: R.Lewis and E.D.Tagg (editors). Computers and Education. North Holland, Amsterdam, 1981. De twee delen bevatten samen 876 bladzijden.

Pieter van der Hijden.

DAI pc MEMORY MAP

version 3.8

INTERRUPT VECTOR ROUTINES: 0000 - 003F

0000 - 0007: Interrupt vector routine 0
0008 - 000F: Interrupt vector routine 1
0010 - 0017: Interrupt vector routine 2
0018 - 001F: Interrupt vector routine 3
0020 - 0027: Interrupt vector routine 4
0028 - 002F: Interrupt vector routine 5
0030 - 0037: Interrupt vector routine 6
0038 - 003F: Interrupt vector routine 7

Interrupt vector routines:

```
00  NOP
E5  PUSH H
2A  LHLD:
..  ) vector address location
..  )
E9  PCHL
00  NOP
00  NOP
```

UTILITY WORK AREA: 0040 - 0061

0040 POROM: Bank select port status.
Duplicate of (#FD06)
0041/42 RSWK1: Save (psw) during ROM bank switching
0043/44 RSWK2: Save (H,L) during ROM bank switching

005F TICIM: Current interrupt mask.
Duplicate of (#FFF8)

INTERRUPT VECTOR ADDRESSES: 0062 - 0071

0062/63 I0USA: Vector address RST 0
0064/65 I1USA: Vector address RST 1: utility/encode.
0066/67 I2USA: Vector address RST 2: stack interrupt.
0068/69 I3USA: Vector address RST 3: sound interrupt.
006A/6B I4USA: Vector address RST 4: math. restart.
006C/6D I5USA: Vector address RST 5: screen restart.
006E/6F I6USA: Vector address RST 6: keyb. int. serv.
0070/71 I7USA: Vector address RST 7: clock interrupt.

SCREEN VARIABLES: 0072 - 00CF

0072/73 Cursor position address.
0074 Cursor type:
#00: cursor flashes in colour.
#01: cursor alternates between actual character and contents #0075.
0075 Cursor information:
If type = 0: Mask which is EXOR'ed with the colour byte for that character to flash it.
If type = 1: Cursor alterates between actual character and this information.
0076/77 Contents screen RAM location indicated by cursor. #0077 contains the data; #0076 the colour byte.
0078/79 Line mode byte of currently used line of the screen RAM.
007A Free character space on currently used line of screen RAM.
007B #00 is stored during screen RAM routines.
007C) colours for colour X3
007D) registers COLOR1 X2
007E) X1
007F) X0

Variables set to describe the current state of the screen.

0080/81 SCREEN: Points to first byte of screen RAM.
0082/83 SCTOP: Points after header.
0084/85 FFB: First free byte in this mode.
0086/87 GRR: Points to top of rolled area.
0088/89 GRE: Points after end of graphics area.
008A/8B CHS: Points to start of character area.
008C/8D GAE: End archive area (unsplit).
CHE: After end of character area (if char).
008E/8F SCE: End of screen (after trailer).
0090/91 GTE: End area used splitting mode.
0092/93 GAS: Unsplit: start archive save area.
GTS: Split: start temporary save area.
0094/95 GRC: Number of blobs horizontally in mode.
0096 GRL: Number of lines of graphics in mode.
0097 GAL: Number saved lines of graphics.
0098 GXB: Number of bytes/line this mode.
0099/9A GREO: Previous end of graphics.
009B/9C CHSO: Previous start characters.
009D SMODE: Current screen mode (updated after mode changed):

#00 mode 1	#08 mode 5
#01 mode 1A	#09 mode 5A
#02 mode 2	#0A mode 6
#03 mode 2A	#0B mode 6A
#04 mode 3	
#05 mode 3A	#10 ?
#06 mode 4	
#07 mode 4A	#FF mode 0

009E) colours for colour X3
009F) registers COLOR6 X2
00A0) X1
00A1) X0
00A2/A3 EBUFR: Start EDIT buffer.
00A4/A5 EBUFN: Input pointer EDIT buffer.
00A6/A7 EBUFS: End available space.

```
00AC      Duplicate of (#FE03), Control word 8255.

00B4      TABTP:  Null tab table.

00C4/C5   init. #CA01: Used to jump to address #CA01.
00C6/C7   init. #CA25: Used to jump to address #CA25.


MATH. WORKING AREA: 00D0 - 00FF
===============================

00D0/D1   init.#C7F2.
00D2/D3   init.#DDE0.

00D4      MVECA:  Offset of start HW/SW vector:
                  Read from #FB02 if math.chip installed:
                     #00  No math. chip.
                     #7B  math. chip present.

00D5 - 00FF:  Used as scratch pad memory for math.
              package. Used in single and double byte
              configurations.


BASIC VARIABLES: 0100 - 02EB
============================

          Following are saved by soft break:

0100/01   CURRNT: Start of current line.
0102/03   BRKPT:  Start of current command.
0104/05   LOPVAR: Points to current loop variable.
                  0 if no running loop.
0106      LSTPF:  Flag for integer/fpt loop and
                  implicit/explicit loop.
0107-0A   LSTEP:  Step value if explicit.
010B-0E   LCOUNT: Loop iteration count.
010F-10   LOPPT:  Pointer to start loop.
0111/12   LOPLN:  Pointer to start loop line.
0113/14   STKGOS: Stack level at last GOSUB.
                  0 if no active call.
0115      TRAFL:  Trace flag.
0116      STEPF:  Step flag.
          (STRFL: Trace/step flag together)
0117      RDIPF:  Flag set while running input.
0118      RUNF:   Flag set while running program.
                  (Previous 2 bytes must be consecutive)

          Runtime scratch area:

          GSNWK:  Scratch area for GOSUB/NEXT.
          LISW1:  Start of listed area.
0119/1A   COLWK:  Scratch area for SCOLG, SCOLT.
011B/1C   LISW2:  End listed area.

          Save area for restart on error.

011D/1E   ERSSP:  Stack pointer.
011F-21
0122      ERSFL:  Set if encoding a stored line.
```

Data/read variables:

0123 DATAC: Offset of next character to encode.
0124/25 DATAP: Pointer to current data line.
0126 CONFL: Set if there is a suspended program.
0127/28 STACK: Current base stack level.

Scratch location for expression evaluation.

0129-2C WORKE: Scratch area.

Random number kernel:

012D-30 RNUM: Random number kernel.
!RNDLY: Random number delay count (1 byte).

Output switching:

0131 OTSW: #00 output to screen + RS232.
#01 output to screen only.
#02 output to edit buffer.
#03 output to disk (DCE-bus).

Input switching:

!INSW: #00 from keyboard.
#01 from disk (DCE-bus).

Encoding input source switching:

0132/33 EFEPT: Pointer.
0134 EFECT: Count.
0135 EFSW: #00 Input from keyboard/screen.
#01 Input from string.
#02 From edit buffer to program area.

Variables used during expression encoding.
(could overlap with runtime variables).

0136 TYPE: Type of latest expression or item.
0137 RGTOP: Latest priority operator.
0138 OLDOP: Old priority operator.
0139/3A HOPPT: Pointer to place for operator.
013B/3C RGTPT: Pointer to RGT operand latest operator.
(order of last 7 bytes important)

Mask to select cassette 1 or 2:

013D CASSL: #10 Cassette 1 activated.
#20 Cassette 2 activated.

Encoded input buffer:

013E-BD EBUF: 128 bytes buffer. Also used by utility.

Interrupt handler variables:

01BE/BF TIMER: Timer location.
01C0 CTIMR: Cursor clock. Used for cursor flashing.
CTIMV: #15: Flash time in 20 ms units.
01C1 KBXCT: Extend keyboard scan time counter. When 0, keyboard scan will be performed.
KBXCK: #02: Keyboard scan time (16 ms units). RAND routine needs this even.

Sound control block storage:

01C2-CF SCB0: Sound control block 0.
01D0-DB SCB1: Sound control block 1.
01DE-EB SCB2: Sound control block 2.
01EC-F4 NCB: Noise control block.
SCBL: Length of a sound block (14 bytes).
NCBL: Length of noise block (9 bytes).

Envelope storage:

01F5 - 0274: ENVST: Envelope storage (128 bytes).
ENVLL: #40: Number of bytes/envelope
NUMENV: #02: Number of envelopes.
0275 - 028E: IMPTAB: Implicit type table.

028F IMPTYP: Default number type.
0290 REQTYP: Required number type.

Space variable space:

0291/92 DATAQ:
0293 RNDLY:
0294 POROM: Duplicate of (#FD04).
0295 POR1M: Duplicate of (#FD05).
0296 INSW: Checked in each GETC routine:
If 0, input from keyboard.
If <>0, input from RS232.

Heap/text buffer/symtab pointers:

029B/9C HEAP: Start address of HEAP.
029D/9E HSIZE: Size of HEAP.
HSIZD: #100: Default size.
029F/A0 TXTBGN: Start address of text buffer.
02A1/A2 TXTUSE: End text buffer.
STBBGN: Start symbol table.
02A3/A4 STBUSE: End of symbol table.
02A5/A6 SCRBOT: Bottom screen RAM area:
mode 0: #B350
mode 1A/2A: #B7A0
mode 3A/4A: #A65C
mode 5A/6A: #63B8

Keyboard variables + constants:

```
02A7/A8  KBTPT:   Pointer to table with ASCII-codes.
02A9-B0: MAP1:    Latest scan of keys (key-codes).
                  (row 0 in #02A9, row 7 in #02B0)
02B1-B8: MAP2:    Previous scanning of keyboard.
02B9     KNSCAN:  Set to scan for BREAK only. When (#02B9)
                  <> 0, scanning for BREAK only.
02BA-BD: KLIND:   4 byte circular buffer to store the ASCII
                  values for keys pressed.
                  KBLEN/KEYL: #04: length rollover buffer.
02BE/BF  KLIIN:   Next position for input to KLIND.
02C0/C1  KLIOU:   Next position for output from KLIND.
02C2     RPCNT:   Count for REPT. #01 when REPT is not
                  pressed. Else it is used as timer for the
                  repeat function.
02C3     SHLK:    Set to #FF when CTRL is pressed to
                  invert SHIFT. Else #00. Used to
                  calculate the offset for the ASCII code
                  table.
02C4     KBRFL:   #FF indicates BREAK pressed.
```

Disc/cassette switching vectors:

02C5-EB: Copy of ROM (#D7A4 - #D7CA).

```
02C5  WOPEN:   C3 B8 D2  JMP: D2B8
02C8  WBLK:    C3 F1 D2  JMP: D2F1
02CB  WCLOSE:  C3 27 D4  JMP: D427
02CE  ROPEN:   C3 25 D3  JMP: D325
02D1  RBLK:    C3 40 D3  JMP: D340
02D4  RCLO:    C3 45 D4  JMP: D445
02D7  MBLK:    C3 A2 D3  JMP: D3A2
02DA  RESET:   C9        RET
02DB           00        NOP
02DC           00        NOP
02DD  DOUTC:   C9        RET
02DE           00        NOP
02DF           00        NOP
02E0  DINC:    C3 B4 DD  JMP: DDB4
02E3           C9        RET
02E4           00        NOP
02E5           00        NOP
02E6  TAPSL:   24 24     Tape speed leader.
02E8  TAPSD:   24 3C     Tape speed data.
02EA  TAPST:   24 18     Tape speed trailer.
```

HEAP, PROGRAM AREA, SCREEN RAM: 02EC - BFFF

```
HEAP (Strings + arrays).
Program (compiled Basic).
Symbol table.
Not used RAM.
Screen RAM.
```

ROM AND CPU AREA: C000 - F8FF

C000 - EFFF: 24K ROM:
C000-DFFF: 8K non-switched ROM.
E000-EFFF: 4 banks of each 4K ROM.
(switchable).

F000 - F7FF: Used with external interrupts.
Can be used for ROM extension (reading only).

F800 - F8FF: Microcomputer stack.
Incl. vector for MDS jump instructions.
F800 SRBOT Bottom of stack RAM.
F900 STTOP Top of stack RAM.

I/O DEVICE ADDRESSES: F900 - FFFF

F900 - FAFF: Spare I/O device addresses.

MATH. CHIP AMD 9511: FB00 - FBFF

FB00 MTHAD: Data math.chip.
FB02 Command + status.

PROGRAMMABLE INTERVAL TIMER 8253: FC00 - FCFF

Used for sound generator. 3 independent 16 bits down counters with programmable counter modes.

FC00/01 SND0: Counter 0 (oscillator channel 0).
(PDLCH: used as counter for paddle operations)
FC02/03 SND1: Counter 1 (oscillator channel 1).
FC04/05 SND2: Counter 2 (oscillator channel 2).
(16 bit data; LSB first)
FC06 SNDC: Command 8253. To be loaded prior to freq. selection with resp. #36, #76 and #B6.
Command word format:
bit 0 : 0 binairy counter 16 digits.
1 BCD counter (4 decades).
3,2,1: 000 mode 0: Int. on end count.
001 mode 1: Progr. one shot.
x10 mode 2: Rate generator.
x11 mode 3: Sq.wave rate gen.
100 mode 4: SW trig. strobe.
101 mode 5: HW trig. strobe.
5,4 : 00 Counter latch operation.
01 Read/load MSB only.
10 Read/load LSB only.
11 Read/load LSB first, then MSB.
7,6 : 00 Select counter 0.
01 Select counter 1.
10 Select counter 2.
11 Illegal.

```
Several control words:
COFIX: #00  Fix count on channel 0.
COM0:  #30  Chan.0, mode 0, 2 byte op.
COM1:  #32  Chan.0, mode 1, 2 byte op.
COM3:  #36  Chan.0, mode 3, 2 byte op.
C1M3:  #76  Chan.1, mode 3, 2 byte op.
C2M3:  #B6  Chan.2, mode 3, 2 byte op.
```

DISCRETE I/O DEVICE ADDRESSES: FD00 - FDFF

```
FD00    PORI:   IN (1)  bit 0: -
                        1: -
                        2: PIPGE: Page signal
                        3: PIDTR: Serial output ready
                        4: PIBU1: Button on paddle 1
                                  (1 = closed)
                        5: PIBU2: Button on paddle 2
                                  (1 = closed)
                        6: PIRPI: Random data
                        7: PICAI: Cassette input data

FD01    PDLST:  IN (3)  Single pulse used to trigger
                        paddle timer circuit.

FD04    PORO:   OUT (2) bit 0 - 3: volume osc. channel 0
                            4 - 7: volume osc. channel 1
FD05    POR1:   OUT (2) bit 0 - 3: volume osc. channel 2
                            4 - 7: volume random noise
                                   generator.
FD06    PORO:   OUT (3) bit 0: POCAS: Cassette data output
                          1,2: PDLMSK: Paddle select
                            3: POPNA:  Paddle enable
                            4: POCM1:  Cassette 1 motor
                                       control.(0 = run)
                            5: POCM2:  Cassette 2 motor
                                       control.(0 = run)
                          7,6:         ROM bank switching:
                                          00  bank 0
                                          01  bank 1
                                          10  bank 2
                                          11  bank 3
```

PROGR. PERIPHERAL INTERFACE 8255 : FE00 - FEFF

Used for DCE-bus (GIC Controller).

```
FE00    (1) GIC: I/O port A
FE01    (1)      I/O port B
FE02    (1)      I/O port C
```

FE03 (6) Command word 8255:

Contr.	PA	PCH	PCL	PB	
#80	out	out	out	out	(mode 0) RWMOP → basic out
#81	out	out	in	out	
#82	out	out	out	in	
#83	out	out	in	in	
#88	out	in	out	out	
#89	out	in	in	out	
#8A	out	in	out	in	
#8B	out	in	in	in	
#90	in	out	out	out	RWMIP → basic in.
#91	in	out	in	out	
#92	in	out	out	in	
#93	in	out	in	in	
#98	in	in	out	out	
#99	in	in	in	out	
#9A	in	in	out	in	
#9B	in	in	in	in	

TICC: TIMER + INTERRUPT CONTROLLER 5501: FF00-FFFF

FFF0 (4) Serial input buffer. Contains the last character received on the RS232 interface.

FFF1 (4) Keyboard input port. Bottom 7 bits are data input from the keyboard. Bit 7 is the IN7 line from the DCE-bus and is attached to the page-blanking signal for the TV. Every 20 ms. an impulse is present.

FFF2 (5) Interrupt address register:
bits 5,4,3: Number of pending interrupt.
7,6 :)
2,1,0:) always '1'

FFF3 (4) Status register:
bit 0: Frame error. Set by a BREAK on the RS232 input.
1: Overrun error. Set if a character has been received but not taken by the CPU.
2: Serial input. Set if no data is received.
3: Receive buffer loaded. Set if a character has been received.
4: Transmit buffer empty. Set if RS232 output is ready to accept another character.
5: Interrupt pending. Set if one or more of the enabled interrupts has occured.
6: Full bit detected. Set if the first data bit of an incoming character has been detected.
7: Start bit detected. Set if the start bit of an incoming character has been detected.

```
FFF4     (5) Command register:
             bit 0: TICC reset.
                 1: Send Break. If set, the serial output
                    is high impedance.
                 2: Interrupt 7 select. A '1' selects IN7
                    of the DCE-bus, a '0' selects Timer 5.
                 3: Interrupt acknowledge enable.
                    A '1' enables TICC to accept a INTA
                    signal from the CPU.
                 4 - 7: Always 0.
FFF5     (6) Communications rate register:
             bit 0:  110 baud
                 1:  150 baud
                 2:  300 baud
                 3: 1200 baud
                 4: 2400 baud
                 5: 4800 baud
                 6: 9600 baud
                 7: 1 - one stop bit.
                    0 - two stop bits
FFF6     (6) Serial output buffer. Write byte to this
             location to send it on the RS232 output.
             Use only when #FFF3-bit 4 is high.
FFF7     (7) Keyboard output port. Data output to scan
             keyboard.
FFF8     (5) Interrupt mask register:
             bit 0: timer 1 has expired.
                 1: timer 2 has expired.
                 2: External interrupt.
                 3: Timer 3 has expired.
                 4: Serial receiver loaded.
                 5: Serial transmitter empty.
                 6: Timer 4 has expired.
                 7: Timer 5 has expired or IN7.
             (react only on low-high transition)
FFF9     (5)          Timer 1 address
FFFA     (5)          Timer 2 address
FFFB     (5) SNDIAD:  Timer 3 address
FFFC     (5) KBIAD:   Timer 4 address
FFFD     (5)          Timer 5 address
FFFE         not used.
FFFF         not used.


NOTES:   (1) Read and write allowed.
         (2) Reading allowed. Writing too, but may be
             overwritten by BASIC program.
         (3) No writing allowed.
         (4) Reading allowed, writing not.
         (5) Should not be accessed.
         (6) Writing allowed, reading not.
         (7) Reading not allowed, writing is harmless but
             useless; keyboard scanner will overwrite it.
```

REMARKS:

ADDRESSES FB00 - FFFF:

The 2 highest bytes of the address are used for the chip select signal CS of the peripheral equipment 8253, 8255, 5501 etc. The lowest byte is used to address the several registers of the peripheral. The 2nd LSB does not have any value. So addresses in this range can be read as FBxO - FFxF, in which x is a don't care.

For additional information and comments, please contact:

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

LIST

*** WILHELMUS *** PAGE 01

```
1       REM wilhelmus ***
10        CLEAR 1000
20        DIM TOON(8)
30        DIM FR(68),WT(68)
40        FOR I=0 TO 8:READ TOON(I):NEXT
50        DATA 392,440,494,523,587,659,698,784,880
60        FOR I=1 TO 18
70    1     READ FR(I),WT(I)
80    1     FR(I+18)=FR(I):WT(I+18)=WT(I)
90    1     NEXT
100       FOR I=37 TO 68
110   1     READ FR(I),WT(I)
120   1     NEXT
130       T=3
140       FOR I=1 TO 68
150   1     IF FR(I)=9 THEN SOUND OFF :GOTO 180
160   1     SOUND 0 0 15 0 FREQ(TOON(FR(I))*1.5)
170   1     SOUND 1 0 12 0 FREQ(TOON(FR(I))*1.5-9)
180   1     WAIT TIME WT(I)*T
190   1     NEXT
200       SOUND OFF :PRINT
210       INPUT "NOG EEN KEER ";JOFN$
220       IF JOFN$="NEE" GOTO 300
230       IF JOFN$="JA" GOTO 140
240       PRINT :PRINT "BEGRIJP IK NIET ! MOET HET ";
250       GOTO 210
300       PRINT :END
510       DATA 0,10,3,8,9,2,3,10,4,5,5,5,6,5,4,5,5,10,4,5,5,5,6,10,5,10,
C         4,5,3,5,4,10,3,20,9,10
520       DATA 5,5,6,5,7,20,8,10,7,20,6,10,5,10,4,5,5,5,6,10,5,10,4,10,3
C         ,10,4,20,9,10
530       DATA 0,10,3,5,2,5,3,5,4,5,5,10,4,20,3,10,2,10,0,10,1,5,2,5,3,8
C         ,9,2,3,10,2,10,3,40
```

T U E R M E V O N H A N O I mit Problemlösung

A. MEYSTRE / MAI 81

Hatten Sie schon die nötige Geduld um die 511 Züge zur richtigen Umschichtung auszuführen?
Falls ja -- herzliche Gratulation, falls nein -- lassen Sie es doch durch den Computer tun!
Das modifizierte HANOI-Programm löst das Problem mit wählbarer Geschwindigkeit.

Erklärung der rekursiven Lösung in Pseudo-BASIC:

Hauptprogramm:
```
10  A=1 : B=2 : C=3 : M=8
20  GOSUB MOVTOWER (M,A,B,C)
30  END
```

Unterprogramm:
```
500 DEFF SUB MOVTOWER (M,A,B,C)
510 IF M=1 THEN PRINT A,C: RETURN
520 GOSUB MOVTOWER (M-1,A,C,B)
530 PRINT A,C
540 GOSUB MOVTOWER (M-1,B,A,C)
550 RETURN
```

Das Unterprogramm MOVETOWER ruft sich selbst immer wieder auf, bis das ganze Problem gelöst ist.
Jeder Aufruf speichert alle 4 Parameter in einen Stack fortlaufend ab. M ist die Turmhöhe,
A, B und C enthalten die momentanen Turmpositionen 1, 2 und 3. Um die 9 Scheiben Umzuschichten,

werden $2^9 - 1 = 511$ Züge benötigt

Für eine Turmhöhe von 3 Scheiben sieht der Vorgang für die 7 Züge wie folgt aus:

CALL MOVETOWER in Zeile	PARAMETERWERTE M	A	B	C	Ausgabe des Zuges Zeile	PRINT A , C
20	3	1	2	3		
520	2	1	3	2		
520	1	1	2	3		
	1	1	2	3	510	1 3
RETURN	2	1	3	2		
	2	1	3	2	530	1 2
540	1	3	1	2		
	1	3	1	2	510	3 2
RETURN	2	1	3	2		
RETURN	3	1	2	3		
	3	1	2	3	530	1 3
540	2	2	1	3		
520	1	2	3	1		
	1	2	3	1	510	2 1
RETURN	2	2	1	3		
	2	2	1	3	530	2 3
540	1	1	2	3		
	1	1	2	3	510	1 3
RETURN	2	2	1	3		
RETURN	3	1	2	3		
RETURN	3	1	2	3		

Um die obige Prozedur in BASIC zu schreiben, ist der Rekursionsvorgang sowie die Parameterübergabe zu simulieren. Diese Simulation findet in den Zeilen 1000 - 1220 statt.
Im Originalprogramm wurde lediglich eine zusätzliche Abfrage für den automatischen Ablauf eingebaut. Die Zeilen 111 - 114 wurden ersetzt durch

```
111 IF A$<>"JA" THEN GOSUB 350                    Handeingabe wie früher
112 IF A$= "JA" THEN GOSUB 1000: WAIT TIME WT     Automatische Lösung
```

Um die automatischen Züge zu beobachten, kann die Verweilzeit pro Zug WT eingegeben werden.

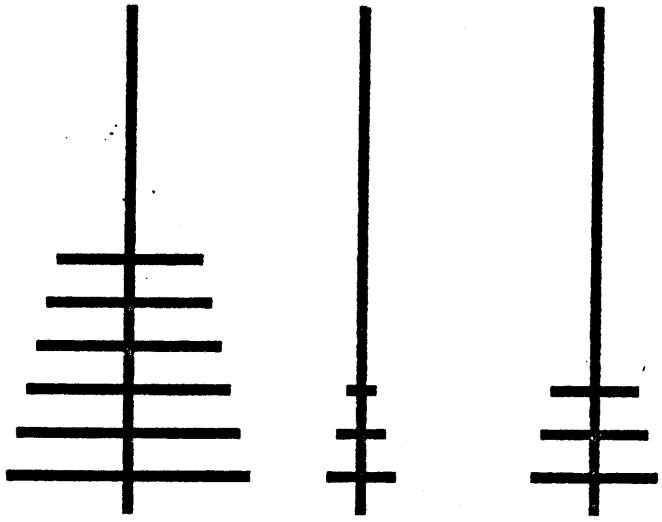
neue Variablen:

ST(TH,2)	Stack mit 2 mal TH Elemente und Stackpointer SP
TH	Turmhöhe (Anzahl Scheiben + 1)
FA,FB,FC	Turmpositionen 1,2 und 3
RET	Rücksprungposition
RESTART	Wiedereintrittsposition

Das Unterprogramm lässt sich für eine beliebige Turmhöhe anwenden.

Literatur: THE COMPUTER JOURNAL volume 20 number 3 1977

```
1     REM HANOI  ERWEITERTE DAI-VERSION   A. MEYSTRE  6/81
5     CLEAR 2000:DIM Z(200.0)
8     MODE 0:PRINT CHR$(12):PRINT :PRINT
9     PRINT "     ..............TUERME VON HANOI................."
10    PRINT :PRINT
12    PRINT "     SIE MUESSEN ALLE BALKEN VOM TURM 1 ZU TURM 3"
13    PRINT "     UMSCHICHTEN, SO DASS NIE EIN LAENGERER BALKEN"
14    PRINT "         AUF EINEN KUERZEREN ZU LIEGEN KOMMT."
15    PRINT "   UM DIE BALKEN ZU BEWEGEN, GEBEN SIE DIE NUMMER DES"
16    PRINT " TURMES AN VON WELCHEM DER BALKEN GENOMMEN WERDEN SOLL,"
17    PRINT "        SOWIE DIE NUMMER DES EMPFAENGERS AN"
18    PRINT :PRINT "            GEBEN SIE DIE TURMHOEHE ALS"
19    INPUT "           ZAHL ZWISCHEN 0 UND 12 EIN ";N:PRINT :IF N<1 OR N>12 THEN 19
22    PRINT :PRINT "        WOLLEN SIE DEN AUTOMATISCHEN ABLAUF?"
24    PRINT "        ANTWORT JA ODER NEIN ";:INPUT A$:PRINT
25    IF A$<>"JA" THEN 28
26    NZ=INT(2^N-0.5):PRINT :PRINT "        ES WERDEN ";NZ;" ZUEGE BENOETIGT !!!"
27    PRINT :PRINT "        VERWEILZEIT PRO ZUG IN 1/50-SEKUNDEN = ";:INPUT WT
28    PRINT CHR$(12):COLORT 7 0 0 0:COLORG 7 4 5 1:MODE 2A
30    JC1=0:Y9=60.0:C1=4.0:C2=5.0:C3=1.0:C0=7.0
33    DRAW 0,0 71,0 C1
36    FOR I=1.0 TO 3.0
38    DRAW I*24-12,0 I*24-12,Y9 C2
40    Z(1.0)=0.0:Z(I*20.0)=10.0:NEXT
50    M=1.0:C=C3
60    FOR I=1.0 TO N
70    Z(1.0)=I:Z(20.0+I)=11.0-I
80    GOSUB 900:NEXT
90    GOTO 110
100   PRINT "     UNGUELTIGER ZUG"
110   JC1=JC1+1:PRINT "     IHR ZUG VON  <1,2 OR 3>  ";
111   IF A$<>"JA" THEN GOSUB 350
112   IF A$="JA" THEN GOSUB 1000:WAIT TIME WT
120   IF M1<>INT(M1) OR M1<1.0 OR M1>3.0 GOTO 100
130   IF M2<>INT(M2) OR M2<1.0 OR M2>3.0 GOTO 100
140   IF M1=M2 OR Z(M1)=0.0 GOTO 100
150   P1=Z(M1)+20.0*M1
160   P2=Z(M2)+20.0*M2
170   IF Z(P1)>Z(P2) GOTO 100
190   M=M1:C=C0:GOSUB 900
210   Z(M2)=Z(M2)+1.0:Z(P2+1.0)=Z(P1)
220   Z(M1)=Z(M1)-1.0
230   M=M2:C=C3:GOSUB 900
240   G=G+1.0
250   IF Z(3.0)<N GOTO 110
300   PRINT "     SIE HABEN  ",JC1,"ZUEGE BENOETIGT";
310   INPUT A$:GOTO 1
350   P=GETC:WAIT TIME 5:IF P=0.0 GOTO 350
360   M1=P-48.0:PRINT M1;:PRINT "  NACH  ";
370   P=GETC:WAIT TIME 5:IF P=0.0 GOTO 370
380   M2=P-48.0:PRINT M2;:PRINT "   ";:PRINT JC1;:PRINT " ZUEGE"
390   RETURN
900   X=M*24.0-12.0
910   Y=4.0*Z(M)
920   X1=Z(Z(M)+20.0*M)+2.0
930   DRAW X-X1,Y X-1,Y C
935   XX1=X1+X:IF XX1>XMAX THEN XX1=XMAX
940   DRAW X+1,Y XX1,Y C
950   RETURN
960   END
```



```
1000  REM UNTERPROGRAMM ZUR REKURSIVEN LOESUNG DES PROBLEMS
1010  REM BEIM AUTOMATISCHEN ABLAUF      A. MEYSTRE  MAI/81
1020  ON RESTART GOTO 1100,1080
1030  DIM ST(N,2.0):SP=0:TH=N:FA=1:FB=2:FC=3:RET=1
1040  REM ENTRY IN MOVETOWER
1050  SP=SP+1:ST(SP,1.0)=TH:ST(SP,2.0)=RET
1060  IF TH=1 THEN RESTART=1:M1=FA:M2=FC:GOTO 1200
1070  TH=ST(SP,1.0)-1.0:HELP=FB:FB=FC:FC=HELP:RET=2:GOTO 1050
1080  TH=ST(SP,1.0)-1.0:HELP=FB:FB=FA:FA=HELP:RET=3:GOTO 1050
1100  REM RETURN FROM MOVETOWER
1110  ON RET GOTO 1130,1150,1140
1130  RETURN
1140  SP=SP-1:TH=ST(SP,1.0)+1.0:HELP=FA:FA=FB:FB=HELP:RET=ST(SP,2.0):GOTO 1100
1150  SP=SP-1:TH=ST(SP,1.0)+1.0:HELP=FB:FB=FC:FC=HELP:RET=ST(SP,2.0)
1160  RESTART=2:M1=FA:M2=FC
1200  PRINT M1;:PRINT "  NACH  ";
1210  PRINT M2;:PRINT "   ";:PRINT JC1;:PRINT " ZUEGE"
1220  RETURN
```

dringende vraag ..

Wie heeft de bit-grafieken op MX-80/2 al bestudeerd ?

```
10    MODE 0:PRINT CHR$(12):COLORT 8 0 0 8
14    REM
15    REM Tekst die met CURSOR op 'n bepaalde plaats
16    REM wordt gezet,moet met 'n spatie beginnen.
17    REM
20    CURSOR 30,20:INPUT " Tekst kleur";TK
30    CURSOR 20,15:INPUT " Achtergrond kleur";AK
40    INV=30.0:CURSOR 13,10:GOSUB 1000:INPUT " INVERSE Achtergrond kleur";INVAK
50    INV=25.0:CURSOR 3,5:GOSUB 1000:INPUT " INVERSE Tekst kleur";INVTK
70    COLORT AK TK INVAK INVTK:PRINT :PRINT :END
998   REM
999   REM ↕ ↕ ↕ ↕ ↕   De inverse routine   ↕ ↕ ↕ ↕ ↕
1000  FOR I=2.0*CURX+8.0 TO 2.0*CURX+8.0+2.0*INV STEP 2.0
1010  POKE #B3E2+CURY*86-I,#FF:REM ADAPT #B3E2 FOR RAM SIZE
1020  NEXT:RETURN
```

DAI 24 × 60 CHARACTER DISPLAY

DS →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
23	BFE7	E5	E3	E1	DF	DD	DB	D9	D7	D5	D3	D1	CF	CD	CB	C9	C7	C5	C3	C1	BF	BD	BB	B9	B7	B5	B3	B1	AF	AD	AB	A9	A7	A5	A3	A1	9F	9D	9B	99	97	95	93	91	8F	8D	8B	89	87	85	83	81	7F	7D	7B	79	77	75	73	71
22	BF61	5D	59	55	51	4D	49	45	41	3D	37	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BEFD	F9	F5	F1	ED																																
21	BEDB	D7	D3	CF	CB	C7	C3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67																																
20	BE55	51	4D	49	45	41	3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BDFD	F9	F5	F1	ED	E9	E5	E1																																
19	BDCF	CB	C7	C3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B																																
18	BD49	45	41	3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BCFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5																																
17	BCC3	BF	BB	B7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F																																
16	BC3D	39	35	31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BBFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9																																
15	BBB7	B3	AF	AB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43																																
14	BB31	2D	29	25	21	1D	19	15	11	0D	09	05	01	BAFD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD																																
13	BAAB	A7	A3	9F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37																																
12	BA25	21	1D	19	15	11	0D	09	05	01	B9FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1																																
11	B99F	9B	97	93	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B																																
10	B919	15	11	1D	09	05	01	B8FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5																																
9	B893	8F	8B	87	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F																																
8	B80D	09	05	01	B7FD	F9	F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99																																
7	B787	83	7F	7B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13																																
6	B701	B6FD	F9	F5	F1	ED	E9	E5	E1	ED	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D																																
5	B67B	77	73	6F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07																																
4	B5F5	F1	ED	E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81																																
3	B56F	6B	67	63	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B4FF	FB																																
2	B4E9	E5	E1	DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75																																
1	B463	5F	5B	57	53	4F	4B	47	43	3F	3B	37	33	2F	2B	27	23	1F	1B	17	13	0F	0B	07	03	B3FF	FB	F7	F3	EF																																
0	B3DD	D9	D5	D1	CD	C9	C5	C1	BD	B9	B5	B1	AD	A9	A5	A1	9D	99	95	91	8D	89	85	81	7D	79	75	71	6D	69																																

↑ x

high address byte x + 8 (MODE BYTE) BIT 7 = 0/1 BIT 6 = 1 BIT 5-4 = 10/11 BIT 3-0 line repeat count (BASIC)

CATALOG

DE SOFTWARE BIBLIOTHEEK

Voor onze bibliotheek gelden volgende afspraken:
Wie een goed programma instuurt mag een drietal programma's in ruil kiezen.Als je meer programma's instuurt kan je een verzameltape kiezen.
Als uw programma later opgenomen wordt in een collectie krijg je nog eens de hele collectie gratis.
Voorlopig zijn alleen de listings beschikbaar bij G1,G2 en G3.
We proberen wel zoveel mogelijk documentatie te verstrekken bij de programma's.

GAMES COLLECTION 1 (G1) : 500 Bfr/ 35 Gld
Yathzee,Awari,Submarine,Kanonspel,Othello,Reactietest,Lunar landing,Vier op een rij

GAMES COLLECTION 2 (G2) : 500 Bfr/ 35 Gld
Surround,Breakout,Amazing,Kim cache,Mastermind,Geitenspel, Space invaders,Hannibal 2000,Towers of Hanoi

GAMES COLLECTION 3 (G3) : 500 Bfr/ 35 Gld
Backgammon,Barricade,Moeras,Slang,Hap maar,Invasion,Robots Traffic test,Life

GAMES COLLECTION 4 (G4) :1000 Bfr/ 66 Gld(machine lang.Prog.)
Football,Break-out,Gompy,Space Invasion,The car

TOOLKIT 1 1250 Bfr/ 84 Gld
Renumber,New format listing,Data statements generator, Basic utilities (labeljump...)

ASSEMBLY PACKAGE 2250 Bfr/ 150 Gld
Assembler,Loader,Disassembler

FGT PACKAGE 1250 Bfr/ 84 Gld
standard FGT,Table creator,Word game,pictures+minuscules, shadow characters,DAI graphic character set,greece alfabet, trigisch alfabet,math symbols,russian alfabet,morse alfabet, fat characters,computer characters

PRIMARY EDUCATION 1250 Bfr/ 84 Gld
Universal math trainer,clock reading,math competition, technical reading,visual discrimination,missing character, pictographic reading

SECUNDARY EDUCATION 1250 Bfr/ 84 Gld
Product of matrices,3X3 determinant,triangle algorithm, volume of box,area computing,Y=A.SIN(B.X+C),examination sine function,volume of cylinder,horner algorithm,Darboux sums

DE SOFTWARE BIBLIOTHEEK (2)

FGT APPLICATIONS 1 : 1250 Bfr/ 84 Gld
Horner Algorithm,555 design,Math competition,TV-TENNIS,
Superwurm,Mastermind,Clock training,FGT-PADDLES
GRAPHIC TABLET : 1250 Bfr/ 84 Gld
WORD PROCESSOR : 1250 Bfr/ 84 Gld
MUSIC COLLECTION 1(M1) 300 Bfr/ 20 Gld
Baby Elephant walk,Music tutor, The sting,Examinus,Motet,
Menuet from beethoven
MUSIC COLLECTION 2(M2) 300 Bfr/ 20 Gld
Promenade,Jesus joy..,40th Mozart,Traumerei Schumann,
La Cathedral Barrios
MAILING LIST/DATA BASE 1250 Bfr/ 84 Gld

GAMES COLLECTION 5 : 500 Bfr/ 35 Gld
Atomic attack,Zich-zagger,Rat Maze (3-dim maze ***),
Dom-dam,boter kaas & eieren,scherpschutter,life in graphic
mode,katz und maus

GAME PADDLES

Gedurende de vakantiedagen hebben we een voorraad game-paddles geconstrueerd.De paddle heeft 3 potmeters met gekleurd dopje, event-knop en controle-led.De bovenkant van het stevige kastje is bedrukt met zeeftechniek.
prijs van de geassembleerde paddles:1500 Bfr,100 Gld per paar.
niet geassembleerd(wel voorgeboord en bedrukt): 1200 Bfr/80 Gld per paar.(inclusief alle onderdelen)
Daar het solderen erg tijdrovend is willen we de geassembleerde paddles graag voorbehouden voor leden die echt niet met de bout kunnen omgaan.
De kwaliteit van de gebruikte onderdelen staat borg voor een lange levensduur.
Mogelijk kunnen we volgende keer ook eenheden aanbieden met joy-stick.Knutselaars kunnen alvast eens binnenstappen in een TANDY-shop, daar is momenteel een joy-stick unit verkrijgbaar aan een redelijke prijs.(100 K weerstand)
Frank Druijff meldde ons dat de uiterste waarden van de potmeter van deze exemplaren niet altijd correct zijn.

GAMES

1. YATHZEE - YANN

Mode Ø : 1 to max. 4 players.
This play will be played with 5 dies and 12 games.
A very well lay-out from the screen will display you : at the top the player's name(put in on the keyboard), which game you play and the score.At your left hand at the bottom, 5 dies are shown and can be changed for the highest possible combination.
Each game you have 3 attempts on highest score. After each game the score will be updated and displayed.
When the games are over, the highest score is the winner.

2. AWARI

Mode 4A : 1 player to the computer.
You are playing to the computer in this game.
Each player has 18 stones to build up the largest house.
The screen is divided into 14 fields; right and left larger fields to build the house, and in the middle, each player has 6 fields to store 3 stones. Each attempt, all the stones are moved to the right and looking for an empty place. If the last stone arrives in your larger field, you have a new attempt.
But, don't forget, each game the computer looses, he will restart the game playing a little bit better.

3. SUBMARINE

Mode 2A : 1 player.
You should try to prevent enemy ships from reaching the other side.
You are the submarine and you can move with the cursor keys, right( ) or left( ), and stop with the space bar, on a little higher speed than the enemies. To fire, push the cursor-up key( ).
Take care, because the enemies drop bombs. The game is over when you are hit and you drop down to the bottom of the sea.
Missed and hit shots are displayed.
The highest score is the winner.

4. KANONSPEL

Mode 1A : 1 player.
In this game you try to shoot somebody in a safety net.
At the beginning you choose : the beginning speed, the fire-angle and the weight of the person.
There is one handicap : the wind.
Windforces are selected at random (between 0 and 12), into the right direction as well as the inverse direction.
You have 5 attempts, with some helpful comments.

5. OTHELLO with algorithme

Mode 2A : 1 player to computer.
It is a well known game and you are playing to the computer.
Starting the game you select : - which color
- who is beginning.

You'll see a checkerboard with 8 rows and 8 columns, 2 black and 2 white squares. With the cursor keys, you can move to all the free fields, to drop your selected color into the box, and to change the enclosed fields in your own color.
We hope you have the most selected colors at your side, at the end. If none, you have more chance the next time.

6. STAR TREK - ENTERPRISE

Mode 4A
You are the commander of the spaceship "Enterprise".
Your mission is to destroy a strange spaceshuttle with a limited quantity of energy.
Your weapons are : - a laser
- a laser canon
- a torpedo

You can move the enterprise into 3 different directions :
- to the strange bases
- to the strange space-shuttle
- back to the own bases.

The boardcomputer can give you the distances in km. and the energy stock.
You can play the game with 3 different levels.

7. REACTIE-TEST

Mode 0

This game is testing your reaction speed.
You have to watch for a counter.
As soon as possible you have seen the first display, push on the space bar.
You can play it with a lot of players to force the highest score and to be the quickest.
Don't touch the space bar before you seen the first figure. Your record will be destroyed.

8. LUNAR LANDING

Mode 4A : 1 to max. 4 players.
You should try to make a soft landing with your spaceship on the moon.
The screen displays a moon-picture with a landing area.
To decrease the speed down, use the cursor-up key( ).To move left and right use the other cursor keys.
At the bottom of the screen your energy stock and speed will be displayed. 3 levels are foreseen to learn.
The highest energy stock at the end is the winner.

```
5     REM FUNNY SIRENE
10    O=#FC00
20    C=#FC06
30    POKE C,#36
100   FOR X=0.0 TO 50.0:POKE O,X:POKE O,X:NEXT
110   FOR X=50.0 TO 0.0 STEP -1.0:POKE O,X:POKE O,X:NEXT
120   GOTO 100
```

```
5     REM ENDLESS CONTINUATION LINES......
6     REM WITH ILLUSION OF A RIGHT MOVING BLOCK OF TEXT
7     REM ALSO TRY : POKE #BF68,#C0, FOR RUNNING MESSAGE !
8     REM #7B HOLDS COUNT OF CONTINUATION LINES
10    PRINT "TEST  ";
20    POKE #7B,0
30    GOTO 10
```

INPAKKEN - SURROUND
==========

Bij dit spel trekken jij en de computer elk een lijn.
Het is de bedoeling om jouw lijn zo te trekken dat de computer zich met zijn lijn vastloopt.
De lijn die jij trekt zal door blijven gaan in de richting die hij heeft tot dat je met de grijze toetsen een nieuwe richting opgeeft.
Een lijn die alsmaar naar boven doorgetrokken wordt zal aan de onderkant weer te voorschijn komen.
Evenzo voor lijnen die links of rechts van het beeld af lopen.
Om te winnen moet je vijf opeenvolgende partijtjes in jouw voordeel beslechten.
Het tempo en de aard van het speelveld kan je zelf kiezen.

BREAKOUT
========

Een soort squash op het TV-scherm.
In een veld, afgebakend door drie lijnen, zijn zes horizontale lijnen getekend.
Met een balletje moet je proberen zoveel mogelijk stukjes van deze lijnen af te breken.
Een zwart balkje (bedoeld als pallet) kun je over het scherm bewegen met behulp van een paddle.
Tien maal mag je opnieuw proberen.
Je bent gewonnen wanneer alle lijnen afgebroken zijn en alle balletjes nog niet verspeeld zijn.

AMAZING
=======

De computer berekent voor jou een willekerige doolhof. Jij bepaalt hoogte en breedte.
Met behulp van de cursor-toetsen moet je nu vanuit de enige ingang van de doolhof zo vlug mogelijk de uitgang proberen te bereiken.

KIM CACHE.
==========

Dit spel kan met maximum vier spelers gespeeld worden.
Achter 28 witte vierkanten zijn 14 paar gekleurde vierkanten verborgen.
Ditmaal gebruik je je eigen geheugen om uit te zoeken waar deze paren zich bevinden.
Per beurt mag je twee maal raden.
De speler die 1 paar kleuren heeft gevonden mag opnieuw spelen.
De winnaar van het spel is diegene die het meeste paren heeft gevonden.

MASTERMIND (Basic Graf Text inbegrepen)
==========

De computer kiest een kombinatie van vier verschillende kleuren uit een reeks van zes : zwart, blauw, rood, groen, geel, wit.
Jij moet deze kombinatie zoeken door de kleuren te kiezen met behulp van de cijfers 1 tot en met 6
Na het indrukken van de return-toets kan de volgende kleur gekozen worden (Voor het indrukken van deze toets kan men zich nog steeds bedenken.)
Na keuze van de vier kleuren geeft de komputer weer, hoeveel kleuren goed zijn en hoeveel er op de goede plaats staan.
Je hebt 15 kansen om de kleurenkombinatie te raden.

GEITENSPEL
==========

Drie witte en drie zwarte geitjes willen een rivier oversteken.
Het is een bergrivier bezaaid met grote rotsen.
Op elke rots is slechts plaats voor 1 geitje.
Elk van hen kan slechts over één enkel ander geitje springen.
Het doel van het spel is, de positie van de drie witte en de drie zwarte geitjes om te wisselen.

SPACE-INVADERS
==============

De planeet aarde wordt aangevallen door een hele boel buiten-aardse ruimtetuigen.(UFO's).
Het is de bedoeling zoveel mogelijk punten te halen door die UFO's met jou aardse tank te beschieten.
Door beschutting te zoeken moet je proberen dat jou tank niet beschoten wordt.

HANNIBAL 2000
=============

Een spel voor soldaten en kinderen.
Je zit met je tank in de Alpen tussen bossen, hoge bergen en diepe dalen.
Hier en daar zijn mijnen verborgen.
Met jou tank kun je obstakels vernielen.
Maar opgelet, wanneer jou tank een verborgen mijn raakt kun je ze niet meer gebruiken.
Het doel van het spel is die hoge Alpen-bergen over te steken, zo dikwijls als je maar kan.

Towers of

TOWERS OF HANOI
===============

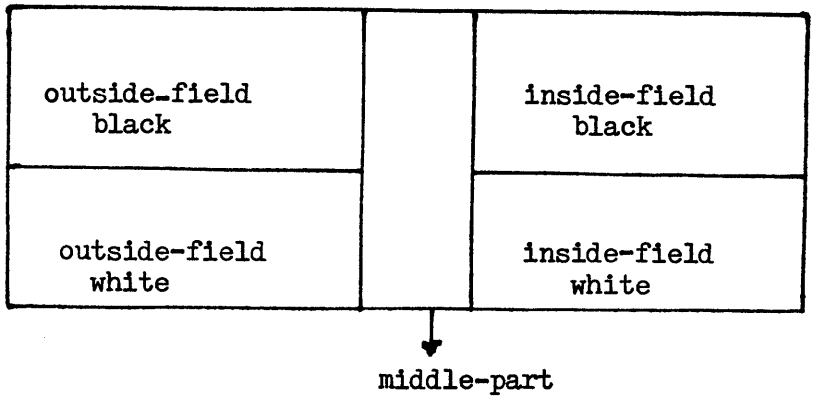
Kies een tempo en een aantal schijven.
De computer tekent drie staven, één staaf voorzien van het aantal schijven door jou gekozen.
Nu probeert de computer, in zoweinig mogelijk stappen, de schijven in de goede orde van grootte op de derde staal te rangschikken.
Dit doet hij door tijdelijk sommige schijven op de tweede staaf te zetten

1. Backgammon

Mode 4A.

After RUN the program gives a short explanation. Some more information about the rules is still necessary. Pushing any key gives the game-board with the startposition of the black and white game-pieces.

You're playing with the white ones against the computer. The game-board is divided into 5 parts (see figure)



Every field contains 6 triangles indicated with the letters A until X.

The objective of the game is to bring all your pieces into your own inside-field. This can be done by throwing with dice. Pushing a key (A-X) makes a game-piece out of that triangle move foreward (up to your inside field) as many places as is indicated by the first (second) die. If all your 15 game-pieces are in your inside-field the second objective of the game is to bring them out of the board.

Winning means all of your pieces out of the game-board.

Throwing the dice is done by spacebar. When you throw a dubble (e.g. 2 times a 3) you can move foreward 4 (all or not different) pieces as many places as is shown by one die.

You can only move your pieces up to a triangle with pieces of your own, or with only one piece of the opponent. In this second case the opponent-piece disappears and is put on the middle-part of the game-board. In such a case the opponent has to bring back in the game the lost piece with his next try. He can't move other pieces. To restart with a lost piece push key Y. If this isn't possible with the number of eyes on the first die, pushing Z changes the order of the dice. Pushing Z twice means you can't play this try.

Two or more pieces on a same triangle means the triangle is occupated. No piece of such a triangle can be taken. One has to jump over such a triangle.

2. Barricade

Mode 2.
Inside a square a dot moves fast up and down, left and right. By pushing the space-bar you put barricades on the way of the moving dot. The objective of the game is to surround the dot.
In one game you have to surround 5 dots. After you surrounded the 5th dot the colours on the screen change. A few moments later the original colours appear again. In the mean time the computer calculates the number of barricades you used to surround the 5 dots. The number can be asked for by pushing any key. A new game starts by pushing S.

3. Moeras -- Morass

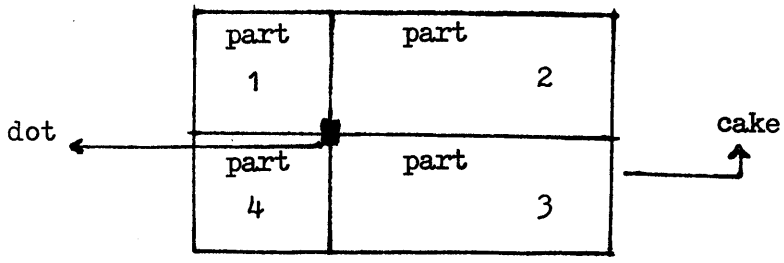
Mode 2A
A game for two players. On the screen a square (morass) is drawn. In that square there are dots (firm ground). The objective is for player one to join the upside with the bottom, and for player two the left-side with the right-side of the square. This can be done by building a bridge between two places of firm ground (dots).
One restriction: you can't cross a connection of the opponent. Moving up to the right places for the connections is done by using the 4 cursor-keys.

4. Slang -- Snake

Mode 0.
The objective is to make a snake (composed of ↑) as long as possible. Two restrictions: the snake may not touch a block drawn on the screen or cross itself.
The movement of the snake is controlled by the left- and right-cursor-key.
The speed can change from 0 to 15 (0 is fast; 15 is slow). A counter displays the length of the snake.
The game can be played by 1 to 10 persons. Each player has 5 tries. The computer displays after each try the highest and the mean score.
After the complete game highest and mean score for all the players can be displayed.

5. Hap maar -- The poisoned cake

Mode 2A
Two players have to eat by turns a piece of poisoned cake, drawn on the screen. You're a dead man (woman) if you eat the poisoned piece.
To eat some cake you just move up the dot to the right place and push key 1, key2, key 3 or key 4 to eat the desired part (see figure)



6. Invaders

Mode 2
25 UFO's land. With one cannon you have to shoot them all with at most 40 balls. Moving the cannon is done by the left- and right-cursor-keys. To fire use the space-bar.

7. Robots

Mode 0
You're inside the walls of a high tension cabine. Five robots try to kill you. Your only chance to survive is to move in such a way that the robots by following you touch a high tension object. Movements are done by PDL(1) and PDL(2).

8. Traffic-test

Mode 5
On the screen a traffic-light is drawn. This light changes at random. On a green light no reaction of the player is expected. On a red light he has to touch a key as fast as possible. A comment on your reaction is given and the evolution of your reaction-speed is shown graphically.

9. Life

This program is written in mode 0 and simulates the evolution of living cells. The simulation is slow or fast. The cells can be placed on the screen by means of the 4 cursor-keys. The C-key is used to generate a cell; the X-key to kill one. S for start.

GAMES COLLECTION 4 66 Gld / 1000 Bfr

A collection of 5 superb machine language programs:

FOOTBALL

A high speed football game in MODE 4.
Try to score 20 points with realistic sound effects.
2 player game, 2 paddles+event. Each player has control over 2 bats.

THE CAR

Try to drive as many circuits as possible, while your car is going faster and faster.
Mileage and rounds are updated all the time.
1 player game, 1 paddle + event.

BREAK-OUT

This is a super-quality arcade game in high resolution.
Try to hit as many bricks as possible, the speed of the ball depends upon the colour of the brick being hit.
Real fun and competition with high-score.
1 player game, 1 paddle.

COMPY

Test your reaction and try to hit the COMPY while he is passing windows. The COMPY moves with random speed, very difficult to finish the game in a reasonable time.
1 player game, event.

SPACE INVASION

Are you fast enough to prevent the invaders to land ?
Each game gives you 9 invasions , try to finish your mission with the highest score.
1 player game, paddle + event.

NOTE: these programs need 48K RAM !

```
10    REM A VERY ATTRACTIVE CURSOR
20    POKE #74,0:POKE #75,#FF:COLORT 8 0 15 0
```

DAInamic software : didactics in mathematics.

1. The matrixproduct

Mode 6 + FGT

This program generates a semi-random (3x4)-matrix to be multiplied by a semi-random (4x3)-matrix.

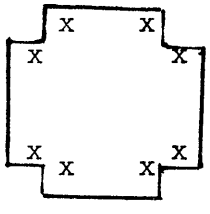
The algorithm of the matrixproduct is shown graphically by flashing numbers and changing colours.

semi-random means that the matrixelements only at random are choosen out of the decimal digits; this allows a didactic screen-layout.

2. Triangle-algorithm for (3x3)-determinants

This program draws a semi-random (3x3)-determinant on the screen. The triangle-algorithm to calculate the value of this determinant is shown graphically.

3. Maximal contents



Take a square sheet of paper; if you cut the corners as shown in the figure, you can make a box. For another x-value you get another box with a different contents. For which x-value is this contents maximal ?

The program allows the user to try out different x-values. Every time the computer calculates the right contents and draws the corresponding box. It is possible not to clear the drawn boxes, so the user can compare the results for different x-values. The maximal value can also be asked for. The didactic objective of this program is to motivate students while they are studying geometric applications on the theory of maxima and minima problems.

4. Maximal surface

This is a program analogous to the previous, with the same didactic objective. Here's asked for the maximal rectangular surface that can be made with a given perimeter. The program allows also to show the surface-evolution for changing values of the length.

5. Study of the function y=a.sin(bx + c)

Mode 6A

The objective of this program is to show the influence of the parameters a,b and c

on the graphic of the function.

$0 < a < 6$ $0 < b < 4$ $-\pi < c < \pi$

The user can choose the different parameter-values.
For each choosen a,b and c the computer draws the corresponding graphic.
For each new graphic the old ones stay on the screen,so they can be compared.

6. Examination of the function y=a.sin(bx+c)

The objective of this program is to evaluate the users' knowledge of the influence of the parameters a,b and c on the graphic of y=a.sin(bx+c).
Pushing key 1 until key 9 generates the computer to draw in black the graphic of a function of the form y=a.sin(bx+c). The functions stored on those keys can be changed easily.
When the graphic is displayed the computer asks for the values of a,b and c.
After the user has given his answer the computer draws in red the graphic corresponding to the users'answer. If the given values for a,b and c are all correct the colour of the black graphic changes slowly from left to right into red.
If not all the values were correct a second (red) graphic appears on the screen and the user can start to think about the error in his answer.

7. Contents of a lying cylinder + contentsfunction

This program asks for the value (in cm) of the length and the radius of the cylinder and displays (mode 0) the contents (in litre) of the cylinder by steps of 5 cm.
Pushing spacebar puts the screen in mode 6 and starts the computer to draw the corresponding cylinder and to fill it up slowly. At the same time the computer draws the graphic of the contentsfunction.

8. Algorithm of Horner

Mode 6
This program uses a graphical way to explain by means of a given example the algorithm of Horner. This algorithm can be used to calculate the remainder and the quotient of the division of a polynomial by x+ a.

9. Definition of the definite integral - Darboux-sums

Mode 6
A graphic way shows how the limit of the Darboux-undersums goes up to the surface underneath the graphic of a fixed function. In a same way the limit of the Darboux-uppersums is treated. In both cases the integration-interval is divided into 6,12,24 and 48 subintervals. For each partition the sum is calculated.

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

alle adressen in HEXvorm!

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

75	cursor symbol
74	cursor mode
72-73	cursor position

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

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

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

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

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

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

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

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

CTRL&COLOR BYTES IN A-MODE

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

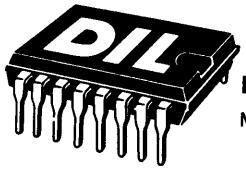
```
FD00 b2 page signal
     b3 serial out rdy
     b4 right paddle
     b5 left  paddle
     b6 random data
     b7 cass. input
FD01    Trigger paddle
FD04 0-3 volume ch.1(0)
     4-7 volume ch.2(1)
FD05 0-3 volume ch.3(2)
     4-7 volume noise
FD06 b0 cass.out
     b1/2 paddle select
     b3 paddle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

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

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

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

DAI-DEALERS *** COMPUTERVERENIGINGEN



D.I.L.-ELEKTRONIKA
Mijnsherenlaan 108, 3081 CH Rotterdam

ALLE DOE-HET-ZELF ELEKTRONIKA - TECHN. TIJDSCHRIFTEN EN -BOEKEN

LEGOTRONICS
Middenstraat 8
8800 ROESELARE BELGIE
tel. 051/207878

ORDIMAX
Rue de la Bonnefemme 11
4030 GRIVEGNEE BELGIE

MULTISOFT
Rue Bargue 25
75015 PARIS FRANCE
7838837

TELEC
Steenstraat 40
9711 GP GRONINGEN NEDERLAND

MSB R.NEDELA
MARTKSTRASSE 3
POSTFACH 1420
D7778 MARKDORF GERMANY

COMPAC
Plaats 25
2513 AD DEN HAAG NEDERLAND

HCC NEDERLAND hobby computer club
Prinsenhof 11
2641 RN PIJNACKER
NEDERLAND

DAI BRUSSEL
Raketstraat 60
1130 BRUSSEL BELGIE
02/2166010

HCC BELGIE
Borkelstraat 51
2120 SCHOTEN BELGIE
031/589674

DAI NEDERLAND
Van Vollenhovenstraat 15A
3016 BE ROTTERDAM NEDERLAND
010/361288

Stichting BASICned
Tolakkerweg 81
3739 JJ HOLL.RADING NEDERLAND

DIDACOM computers&onderwijs
p/a I.BROEKMAN AVENBEECK 98
2182 RZ HILLEGOM
2520/18032 NEDERLAND