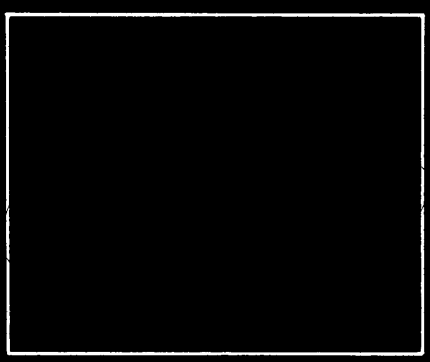
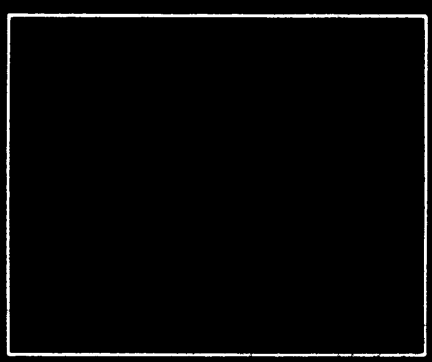
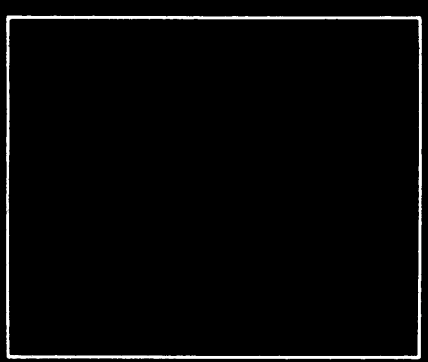
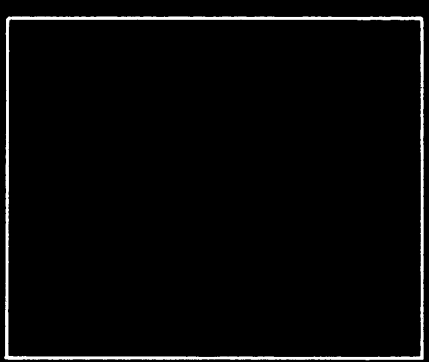
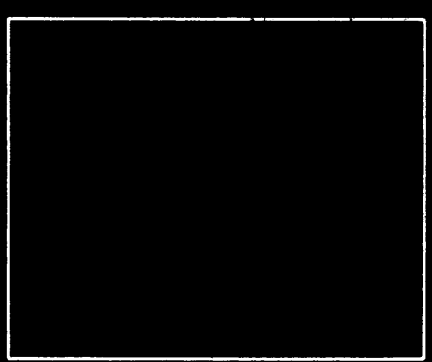
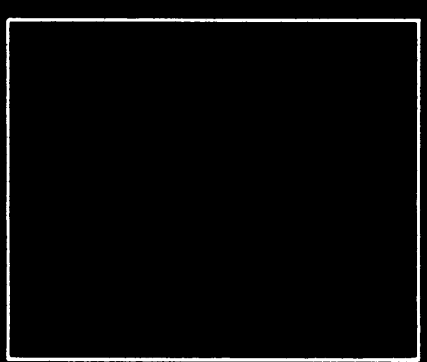
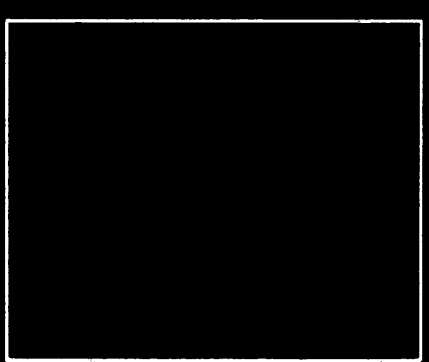
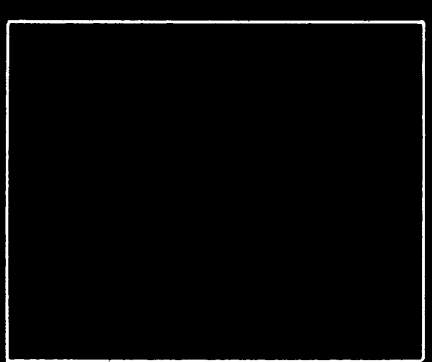
DAI NAMIC

tweemaandelijks tijdschrift maart - april 1983

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

COLOFON

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

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.
Redactie en software bibliotheek

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

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

Lidgelden

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

DAINAMIC

PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

Westmeerbeek, april 83

Beste leden,

Even zag het er naar uit dat we onder tijdsdruk weer naar de noodrem van het dubbel-nummer zouden moeten grijpen. Gelukkig kregen we de laatste dagen nog zo veel kant-en-klare copij dat nummer 15 nog net in april de deur uit kan. Terwijl de persen draaien zijn we dan weer bezig aan nummer 16 ... verveling is er niet bij met deze hobby !
Met veel genoegen kunnen we weer terugblikken op onze succesvolle bijeenkomst van 9 april jongstleden. Meer dan 600 bezoekers, veel programma's, veel projecten en een fijne barbecue. Blijkbaar was er zoveel te zien dat weinigen de tijd vonden om eens gaan te snuffelen in onze tijdschriften-bibliotheek. We danken allen die hun bijdrage aan deze bijeenkomst hebben geleverd. Speciale dank en bewondering voor de vele buitenlandse bezoekers (sommigen hebben hun nachtrust moeten opofferen om tijdig Tongelsbos te bereiken!).
Eindelijk schijnen de mensen van DAI-INDATA begrepen te hebben dat er wat promotie nodig is rond de DAI-computer. De stand van INDATA op de eerste HCC-dagen in België was zeer gelukt, de belangstelling van het publiek was navenant. We hopen dat we U met nummer 16 de kleurenfolder kunnen meesturen die onlangs gedrukt is.
In dit nummer merkt U dat de vertaaldienst aardig op gang is gekomen. Nederlandstalige lezers mogen deze bladzijden overslaan, de vele buitenlandse leden zullen deze bijdragen zeker op prijs stellen.
Mogelijk komen we vanaf nummer 16 tot een integratie met de Duitse gebruikersgroep, we zijn er van overtuigd dat dit alleen maar kan bijdragen tot de verhoging van de kwaliteit van ons blad.

Dear members,

The meeting of 9th april was very succesfull : there were more than 600 visitors. Some of them were very tired after the long trip they made to reach Tongelsbos! On page 143 you will find some photos of the meeting.
INDATA is making good promotion for our computer : they had a beautiful stand on HCC-meeting Belgium (first edition : over 8000 visitors).
We hope to send you the color-brochure with newsletter 16.
In this issue you will find more results of our translation service: C.Dufour and D.atherton and collaborators have done a very good job!
It is possible that there will be an integration with the German club from newsletter 16 on: I'm sure this will lead to an even better magazine! This will mean that there will be one big DAI-users-family in Europe, this will result in a lot of information, a lot of software and a better support for our DAIpc. For the moment, there is only one weak point on the map of Europe : Paris, with Multisoft and DAI club france. Both organisations (same address!!) continue to distribute software that belongs to DAInamic, DAInamic Germany or other clubs. For the moment they even offer very expensive and illegal copies of the famous firmwarebook of Jan Boerrigter! We strongly advise to order the firmwarebook from Mr Boerrigter: legal copies are more reliable and cheaper ... the same applies to our software. If someone has ideas or suggestions how to stop this piracy, please contact us, we will not hesitate to start legal procedures...

W.Hermans

INHOUD — CONTENTS

HOW TO ORDER DAInamic SOFTWARE

1/ for Belgium : banc order, cheque , cash
2/ foreign countries : due to the very high charges with other modes of payment, we will only accept :
- eurocheque in Belgian francs
- international postal money order (in your local post office).Note your order and address in CAPITALS on the postal card please !

both payments to : DAInamic V.Z.W Heide 4 3171 WESTMEERBEEK BELGIUM.

DAInamic subscription rates :

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

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

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

games 10

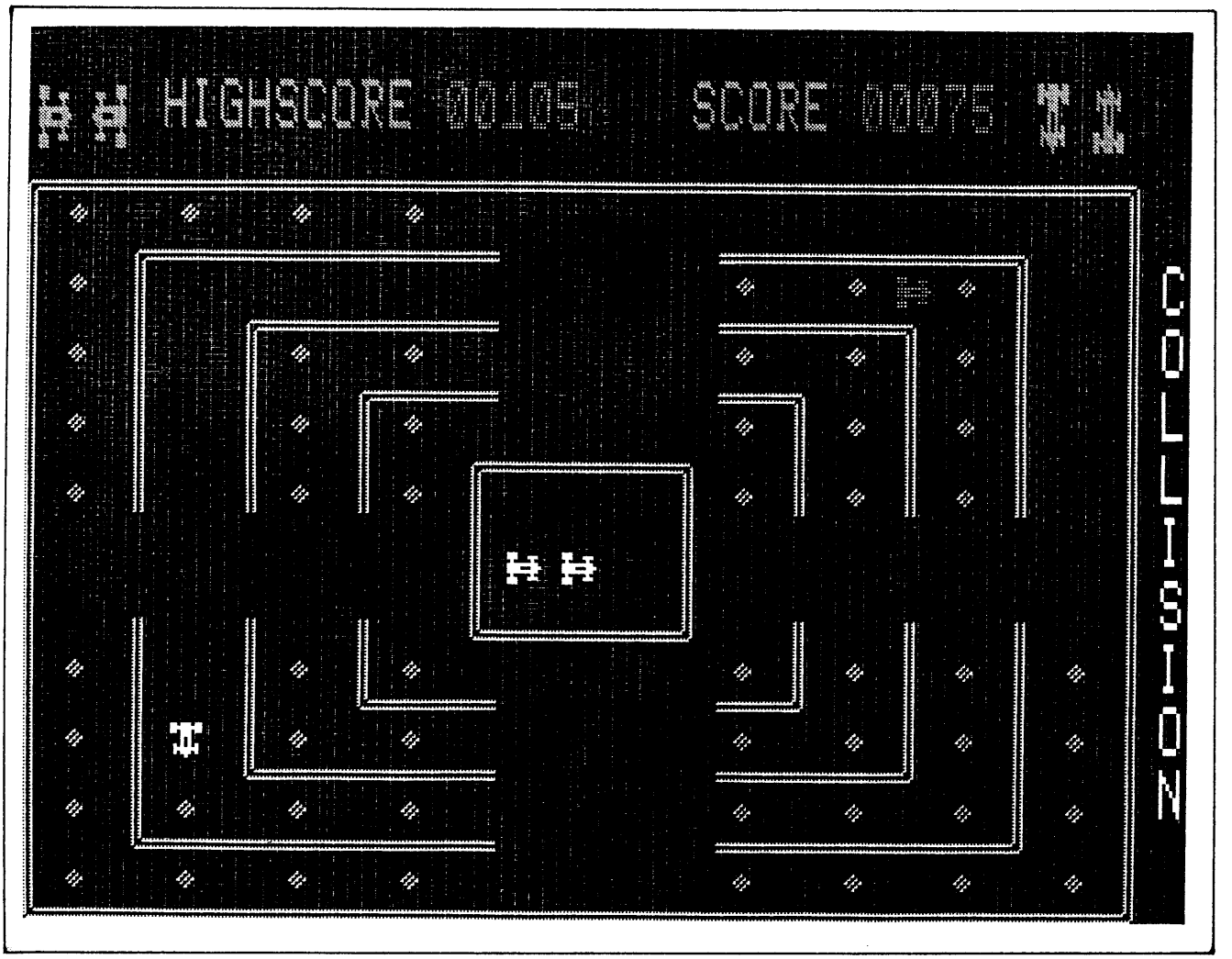
CATALOG

GAMES COLLECTION 10

1/ COLLISION — N.P.LOOIJE

A beautiful PACMAN-alike cargame.You control your car with the cursorkeys, spacebar to accelerate/slowdown.
Try to drive as many miles as possible but watch out for the collisioncar!

keyboard



2/ MINIGOLF — R.SIP

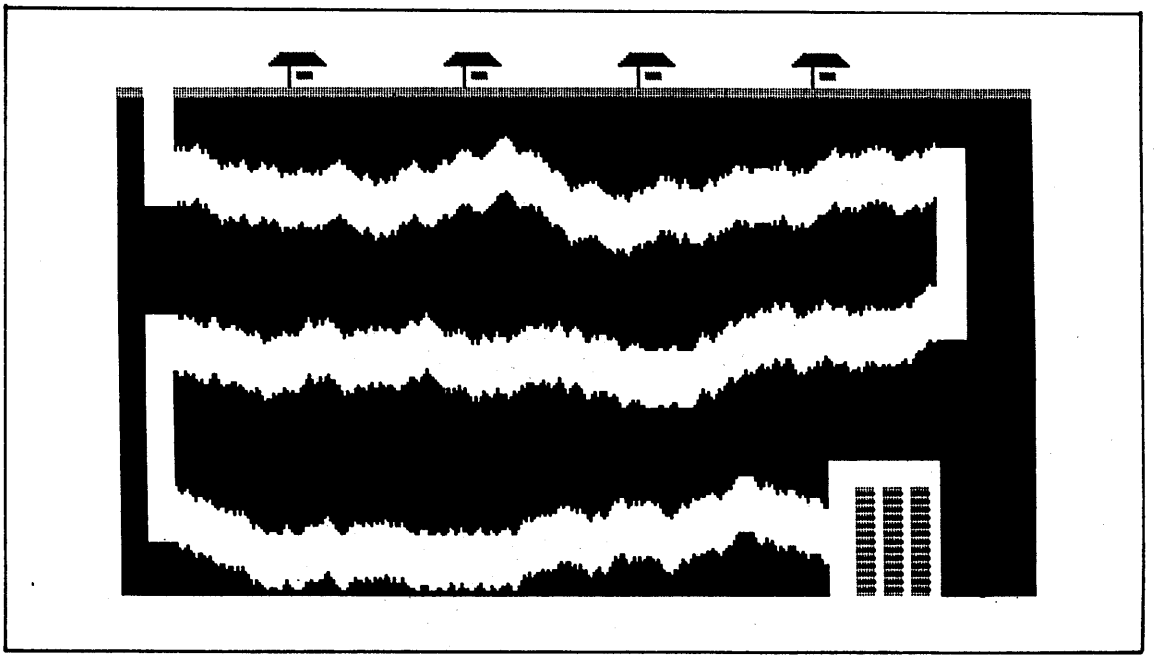
Up to 8 players can participate this realistic minigolf-game. The game contains 14 different fields!

paddle+event

3/ SPELONK M.Hooykaas

Try to reach the treasure inside the cave, but avoid crashing against the wall!

keyboard



4/ SOLITAIRE-PATIENCE J.Rowbotham

The classical card game for bachelors in splendid high resolution graphics. keyboard

5/ LEFT-RIGHT F.Van Amerongen

The object of the game is to see if you know your left from your right. The computer will draw a colored box and, at the bottom of the screen, two colored bars. You must determine whether the left of the right hand bar matches the box's color. However, if the background is black or if instead of the box a cross is drawn, you must choose the other bar. keyboard or event

6/ FLIEGENPATSCHE M. van Meegen

Try to shoot the flies, which are controlled by the computer. The best time is the highscore.

paddle + event

7/ PUNT W.Hendrix

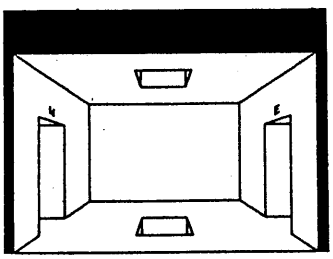
2-player game : try to stop the moving object of your opponent. keyboard

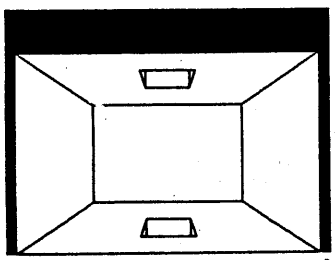
games 11

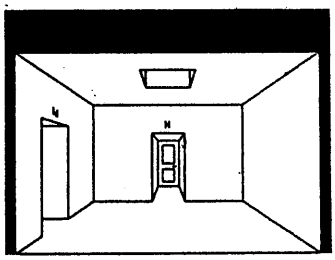
GAMES COLLECTION 11

1/ QUINTIMAZE — F. van Amerongen

Try to find your way out of this 5x5x5 cubic maze.Very fast representation of the maze-rooms! keyboard

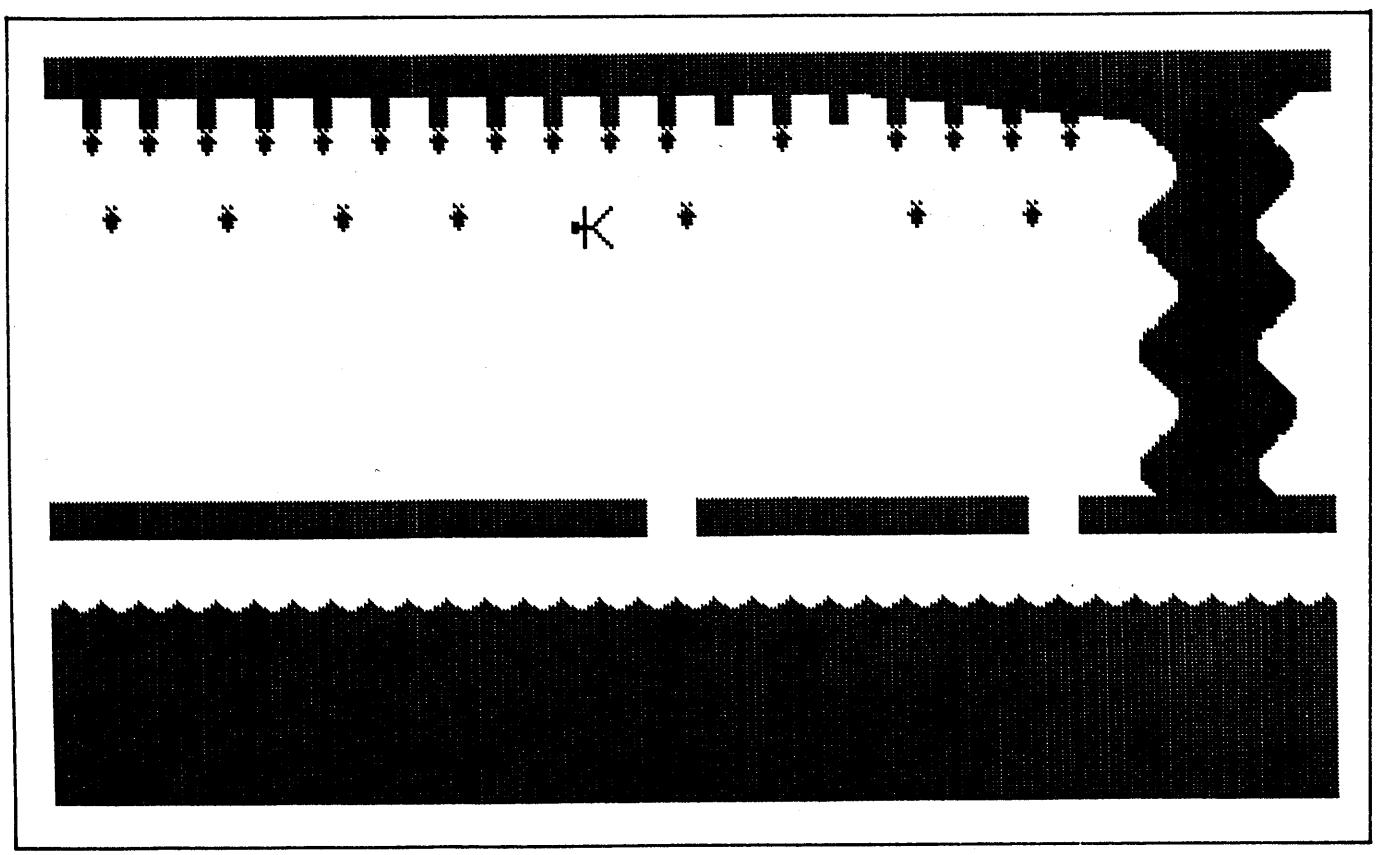






2/ MANGROVE — M.Hooykaas

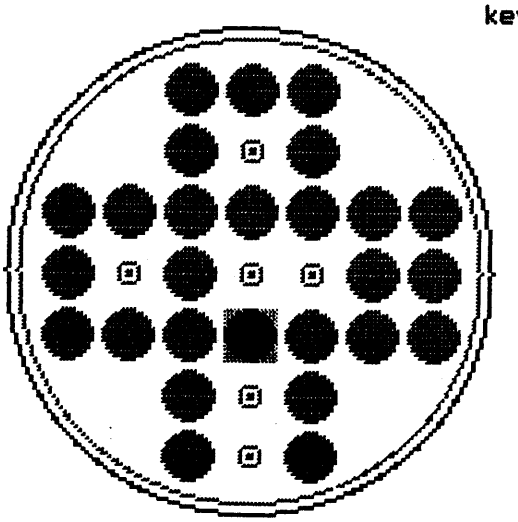
3 castaways, living on an island, their only food being the mangrove-fruits. Look out for the crocodiles while jumping for fruit!! keyboard



3/ SOLITAIRE — F.Druijff

The object of this game is to loose all pegs except one. You loose a peg by moving another peg from one side of the peg to the other (empty) side.

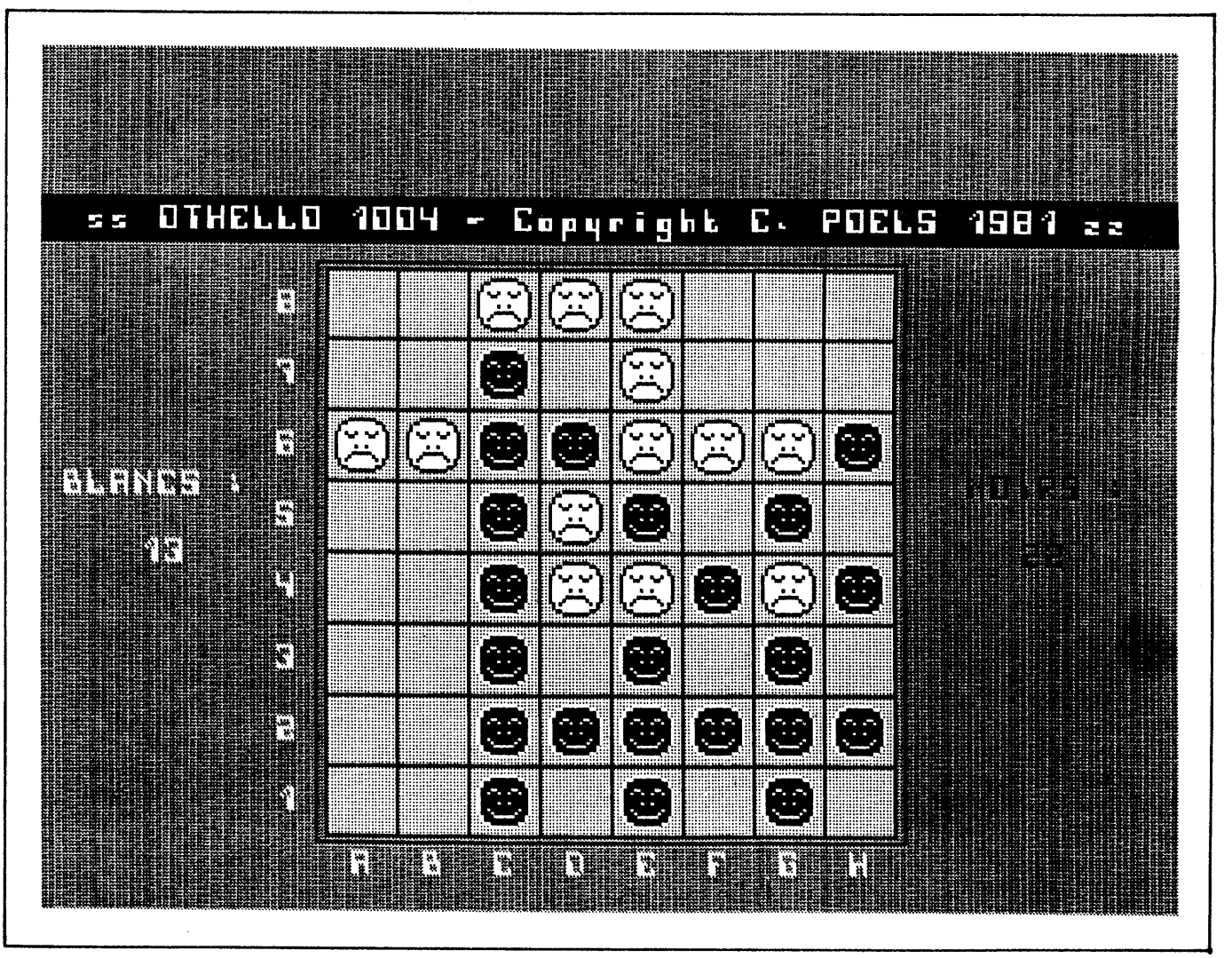
keyboard



4/ FUNNY OTHELLO — C.Poels

The classical othello-game, but this time in a very funny graphical presentation.

keyboard



5/ BRUMM — M.van Meegen

You control a car with the paddle and must drive one lap with it as your opponent tries to do his.
Driving the car is difficult and amusing !

paddle

6/ APPLE — G. van Dongen

Try to catch the apples falling from the tree...

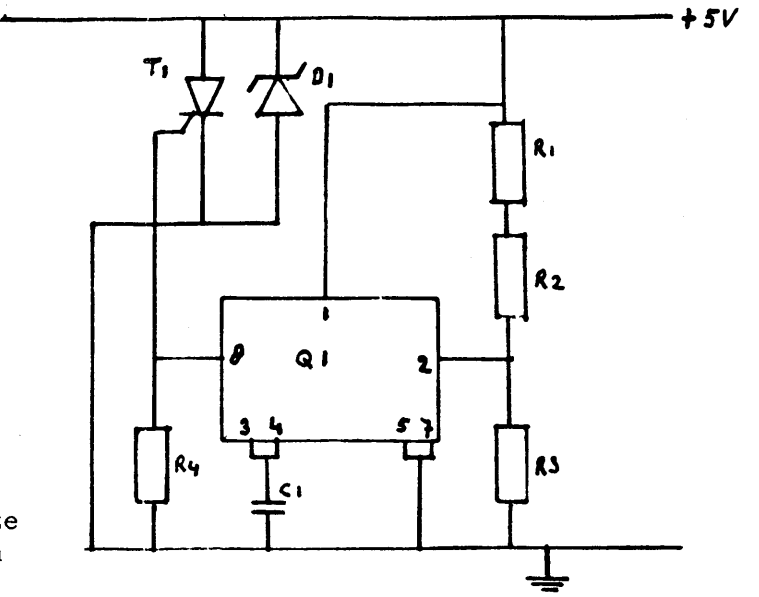
7/ REACTION TEST — Xennt

This program gives you an idea of your action delay.

keyboard

G10 & G11 collectioned and edited by F.DRUIJFF

HET DEFECT RAKEN VAN DE +5 VOLT VOEDING



Onderdelen:
r1= 390 OHM
r2= 3K3
r3= 2k7
r4= 47 OHM
c1= 47 nF
tr1= 2n6504
d1= ESDA 5 (M.C.A.TRONIX
in Rijswijk)
Q1= MC3423 (motorola)

ATTENTIE !

Het printontwerp bij deze schakeling, afgedrukt in DAITA,is niet correct !

Recentelijk is gebleken dat bij enige DAI-bezitters de +5 volt voeding is "opgeblazen",waardoor de mogelijkheid bestaat, dat er diverse TTL's worden vernield.Dit blijkt een nogal behoorlijke financiele consequentie te zijn,daar er ook div. PROMS (74LS288) worden vernield.

Om hier nu wat aan te doen is bovenstaand schema ontworpen. Dit ontwerp is een zogeheten "CROWBAR" schakeling,die spanningen boven een ingesteld niveau detecteerd.Wanneer door het defect raken van een component in de voeding een te hoge spa-ning (22 volt) dreigt door te dringen naar het circuit,dan wordt deze ondervangen door het "overvoltage circuit" en ogenblikkellijk kortgesloten. Op deze manier is het dan uitgesloten dat hogere spanningen naar het circuit kunnen doordringen.Het gevolg voor de voeding is,dat buiten het component dat een eventuele overvoltage veroorzaakte,er nog enige andere componenten kunnen sneuvelen.

Ik denk hierbij aan b.v. TIP 34,BD 140 of het schakel IC ,maar dit weegt niet op tegen de eventuele schade die aangericht zou kunnen worden aan de rest van de computer.Het is ook mogelijk dat de ELCO wordt vernield.

Een eventuele spannings-storing is ook merkbaar als het beeld van uw monitor of tv wegvalt.
De schakeling wordt aangesloten met de + aan de bovenzijde van de zekeringhouder,gezien vanaf het keybord.de - kunt u aan de voedingskooi vastsolderen.Gebruik hiervoor een behoorlijke diameter draad.
Uit bovenstaand onderdelenlijstje blijkt,dat er gebruik is gemaakt van niet alledaagse onderdelen,doch de samenhang garandeert een degelijke beveiliging.

Mocht u problemen hebben met de aanschaf van de onderdelen of het monteren van de schakeling wendt u zich dan tot onderstaand persoon.

T.Verberkt
v.Buerenstraat 13
5256 KL Oud-Heusden
tel.04162-2667

Hamme,11 februari 1983.

vanwege :Staf Van Hoecke
Kon. Albertplein,13
9160 HAMME.

Aan DAINAMIC : Bestuur.

Beste Dai-gebruikers,

Na een zowat drie-jarig bestaan van de users-club,vind ik het passend om even een woord van dank tot de leden van het bestuur en de redactie te richten.
Het zal voor iedere Dai-gebruiker ondertussen wel duidelijk geworden zijn dat de populariteit van de DAI personal computer sterk is gestegen.Dit dank zij de enorme inspanningen die een harde kern van gebruikers zich heeft getroost ,om deze nogal stiefmoederlijk behandelde CP de plaats te geven die hem toekomt,namelijk tussen de krachtigste systemen die voor een hobbyisten-prijs te koop zijn.
Van huis uit heeft de DAI alle hardware-eigenschappen meegekregen om tot die grootsten te behoren.Maar...zoals bij elk systeem ,valt of staat het succes van een PC met de software die ter beschikking komt,alsmede of er al dan niet documentatie en supplementaire informatie gepubliceerd wordt om het systeem verder uit te diepen.

Persoonlijk behoor ik tot een van de eersten die zich een DAI heeft aangeschaft,en kon ik toen "meegenieten" van de enorme leegte en stilte,die na de introductie van deze geniale CP heerste. Nergens werd over het bestaan van de DAI gerept,laat staan dat in een of ander tijdschrift een programma zou te vinden zijn geweest . Over hardware-uitbreidingen kon men toen al even enthousiast stilte bewaren, kortom het was een periode waar we nu nog met de glimlach naar terugblikken.

Vanuit het toeristisch verblijfsoord van Keizer Karel is dan opeens een frisse wind beginnen waaien,onder de vorm van een gebruikersgroep die sindsdien haar naam alle eer aandoet.
Van een eerste schuchtere gestencileerde publicatie,tot de gesofistikeerde verzorgde tijdschriften die nu op de bus gaan,is een lange weg afgelegd,echter in een korte tydspanne.Met Dainamische kracht zijn er programma's geschreven en verzameld,projecten gecoordineerd,vergaderingen en demonstraties ingericht.
Dit onbaatzuchtig enthousiasme werkte aanstekelijk,tot ver over de grenzen van ons klein land.Ondermeer is aan het brein van enkele onzer noorderburen de belangrijke DCR-extensie ontsproten, en uit Duitsland en Frankrijk kwamen gesofisticeerde programma-hulpen, games en utilitaire routines in machine-taal aanspoelen.
Al dit moois kwam ter beschikking van de leden aan meer dan democratische prijzen, en maakte de reeds goedkope Dai tot het best gesitueerde apparaat wat PRIJS/KWALITEIT verhouding betreft.

De kwaliteit van de publicaties,die reeds van bij het begin ernst en degelijkheid uitstraalden,schijnt onafgebroken het zelfde gehalte te bewaren.Ik zie met veel vertrouwen de verdere komende ontwikkelingen van onze Dai tegemoet.

Om al het voorgaande, en om het plezier dat ik tot hiertoe aan mijn Dai beleefd heb, wil ik U danken, en aansporen om in dezelfde richting verder te gaan.

Beste groeten vanwege een enthousiast Dai-user,

DAInamic INFO

Chers membres,

Voici de nouvelles traductions qui concernent cette fois le N0 14 de la revue. On participe à la réalisation de cet encart :
Pierre Holuigue Georges Cochin Christian Poels
Ne vous inquietez pas si votre contribution ne se trouve pas dans le present numéro, Il y en a bien d'autre à venir !.
Encore merci à toute l'equipe .

C. Dufour

CONVERSION Analogique/Digitale (p20)

Afin de pouvoir réaliser des mesures de basse fréquence, j'ai construit une interface avec l' ADW-IC ZN 427 de Ferranti sur plaque Veroboard. Cette interface est abrité dans un boitier de 150x80x50 mm et est raccordé par un cable plat au bus DCE.
Le programme de démonstration accompagnant permet environ 13900 impulsions par seconde. Peut etre y a-t-il un membre du club qui désirerait également construire un convertisseur A/D 8 Bits rapide et qui cherche une solution bon marché. C'est pour cette raison que je veux mettre mes documents à leur disposition. La combinaison possible des bruits "ADW" et de l'analyse de la parole permet de faciliter la programmation de la fonction TALK.
Durant la réalisation de l'interface et les essais, j'ai réalisé la liaison logique avec le DAI comme indiqué fig.3. Ainsi les ports B et C sont programmés en sortie et le port A en entrée. A une des sorties on retrouve à partir du BASIC une mesure lente qui produit l'impulsion du quartz conduisant le convertisseur. (Voir schémas p21..)
Bon amusement pour la construction,Avec mes salutations amicales.

INSTRUCTION "IF...THEN...ELSE..."

Dans les langages de programmation évolués, l'instruction 'IF-THEN-ELSE-' est possible. Et sur le DAI ?? Egalement, avec cependant quelques restrictions, en utilisant judicieusement la sequence 'IF-GOTO'.
Essayez par exemple le programme suivant :

```
10 IF A=1 GOTO 30
20 PRINT "20",
30 PRINT "30",
40 PRINT "40"
```

RUN affiche : '30 40' si A=1 et '40' si A<>1.
La ligne 20 n'est donc jamais executée. Essayez maintenant :

```
10 IF A=1 GOTO 30 : PRINT "TEST"
20 PRINT "20",
30 PRINT "30",
40 PRINT "40"
```

RUN affiche : '30 40' si A=1 et 'TEST 20 30 40' si A<>1
Vous voyez ainsi comment simuler l'instruction 'IF-THEN-ELSE' avec 'IF-GOTO-:...' . Mais attention la première partie DOIT etre de la forme 'IF-GOTO- ou IF-THEN-'!!!
L'explication de cet astuce tient dans la manière dont est executé l'instruction. La routine en MEM (ROM) se trouve en #DF15.
Utilisez prudemment cette instruction et seulement sous la forme sus-citée afin de n'avoir aucun ennuis dans vos programmes...!

DUPLICATION (P33)

Pour dupliquer une ligne vous pourrez utiliser ce truc qui nous vient d'Italie !

* EDIT N0 ligne a dupliquer
* Changer son numéro
* BREAK (2 fois)
* POKE 309,9
* LIST la ligne est copiée!

La fonction BASIC " TAB " (P28)

Comme vous le savez surement, la fonction TAB peut etre utilisée pour disposer des données en colonnes. Le curseur va directement à la colonne fixée par TAB en placant des espaces. Vous avez du remarquer un mauvais fonctionnement de cette fonction en voici l'explication

Pour le BASIC V1.0

Quand le DAI execute la fonction TAB, il controle l'état du drapeau de sortie DOUTC (#131). Lorsque DOUTC=0 (Ecran+RS 232) la fonction TAB fonctionne normalement. Pour toutes autres valeurs de DOUTC seulemen UN espace est généré...!

Pour le BASIC V1.1

Veinards !,votre fonction TAB fonctionne correctement pour DOUTC=0 ET DOUTC=1 (Ecran seul).

TECHNIQUES DE PROGRAMMATION (p16/19)

Tous les possesseurs de DAI ont programmé au moins déja une fois dans leur vie. Cependant, la mise au point et le développement des programmes envoyés laissent encore souvent à desirer. Il n'existe pas de standard de programmation, mais quelques conseils judicieux peuvent grandement aider.

Tout d'abord il faut se demander ce que l'on veut faire avec le programme. Ainsi quelques questions peuvent venir à l'esprit :

- A qui est déstiné le programme ?
- Des REMarques seront elles necessaires ?
- Comment seront les variables: entieres ou en virgule flottante ?
- Le programme doit il etre rapide ?
- Le programme est il urgent ?

A toutes ces questions il y a bien sur une réponse rapide à donner :

- Si le programme doit etre utilisé par d'autres personnes, Très souvent des explications seront nécessaires.
- Oui, si le programme est complexe ou s'il utilise des 'astuces' de programmation.
- Suivant le programme, mais certaines variables peuvent toujours etre entieres (Boucle FOR-NEXT;COLORT;...).
- Si oui alors travaillez avec des entiers, testez differentes possibilités de programmation ou ecrivez des portions de programme en langage machine.
- Dans ce cas, l'apparence du programme n'est pas primordiale, on remarque alors l'importance d'avoir une bibliotheque de programmes.

Le programme donné en page 18 (N0 14) donne l'exemple d'un programme de base 'idéal', car il utilise des modules de programmes. Ce systeme permet de pouvoir réutiliser ces modules dans d'autres programmes. Ce qui augmente nettement la lisibilité. On peut facilement remplacer une partie par son équivalent en langage machine pour en augmenter la vitesse. Dans le cas ou cette methode est utilisée, le programme sera beaucoup plus long et pourrait ne plus tenir dans la mémoire (Cas extreme !)

Bien qu'initialement recommandé, l'initialisation du programme tel que :

```
MODE 0 : PRINT CHR$(12)
```

Présente quelques désavantages : Un programme éxtremement grand qui tourne en mode 1 ou 2 peut donner un "OUT OF MEMORY" apres un MODE 0.

P.S. Vous trouverez ci-apres une nouvelle methode pour le tracé d'un cercle.

TIPS--TIPS--TIPS--TIPS

(from DAInamic 10, page 121)

1. Start most programs with IMP INT.
2. Put the commentary or explanation for the user where it will appear at the beginning of a RUN, not at the start of a listing, e.g. 10 CLEAR 3000: GOSUB 10000 etc., with the foreword starting at line 10000. The program will thereby run faster and can be examined more readily.
3. Avoid the American habit of starting with the question "What is your name?" and never making use of the information received.
4. Do not ask "Do you wish to read the foreword?" There are two better ways.
   (1) Display it, if it is brief, or
   (2) Make an auxiliary program for it, if large.
   The latter has these advantages:
   (1) If the operator does not wish to see the instructions etc., they need not be called or read in.
   (2) The program may run faster.
   (3) Very large programs make better use of the memory by being split into sections.
   (4) The main program can be read in while the instructions are on display.
5. Have a leader or header on more important programs; by this is meant a frame containing the title of the program and the name of its author, preferably in a simple, straightforward style.
6. Do not have announcements and explanations on the screen for what you consider a suitable period, but give the user freedom to choose how long he needs. He can press the space bar to continue the program.
7. When asking a question about the degree of difficulty or level of skill, give some idea of what is difficult and what is easy.

DAI - RS232

(from DAInamic 10, page 148)

As a DAI user-beginner I had the following experiences when I connected a printer .....

DAI RS-232 (rev 5)
On connecting an EPSON MX-80 to my 2 months old DAI, via the RS-232 parallel (surely he means serial) interface, one thing and another did not seem to work as it should. The listing on the DAI went happily on without waiting for the printer. As the printer and interface were tested before buying the suspect DAI was speedily placed on the operating table for an internal examination. The accompanying figures show the print layout and schematic of the RS-232 DTR input (main board, rev 5).
INTERRUPT : Wasn't the 56K resistor on earlier revisions connected to the frame instead +5v? It appears from measurements that with a NOT READY SIGNAL (0.13v) the voltage on input pin 10 of IC 74LS373 (latch) is 0.95v, and is thus somewhat too high. According to the IC specification V(IL) must, for the low level input voltage, be less than 0.8v. The voltage drop across the 3K3 resistor is caused by the current through the 56K resistor (0.072mA) and the low level input current of the IC (0.176mA), which falls easily within the IC spec - I(IL) less than -0.4mA. Lowering the 3K3 resistance to 2K (with a parallel resistor of 5K6) restored the patient to excellent health.

EPSON MX-82 WITH HIDDEN TALENTS.
Getting 80 characters instead of the 96 quoted in the manual drove me to a second operation. An

internal inspection showed DIP switch 1 - 5 in the OFF position. Concerning the function of this switch the manual is to all intents silent, reporting only "Never set this pin to the OFF position, always leave it in the ON position". And indeed in the ON position I received the whole 96 character width, in the OFF position only 80 (plus 16 for margin at the end).

FINALLY, A HAPPY END !
By comparing the control codes with those of the MX 100, three of the latter were not to be found in the MX 82 manual. They were therefore tried out and the results were positive!

ESC G double print (advance paper 1/216" and repeat line)
ESC H cancels ESC G
ESC M elite print (width of letters between normal and condensed)

So, from now on, these codes are also in my MX 82 manual.

Herman Moeys

MACHINE LANGUAGE IN A REM STATEMENT.

(from DAInamic 10, page 149)

(my first very own machine language program)

In MC4 (Microcomputer Journal Nov/Dec 1981) the difficulties were discussed of putting a short section of machine language in a BASIC REM line. Advantage! all could be loaded in one go in the normal way as with a BASIC program. Disadvantages! They were not mentioned, but ... Snags! These will come up for discussion in the following.

1. How do you know where the required line with the REM is in the memory? For example, 10 REM***** In address #29F, #2A0 we find #EC, #03 which means that the start of the BASIC text is to be found at address #3EC. We find there:-

```
03E0   FF FF FF FF FF FF FF FF FF FF 7F FF 09 00 0A A9
03F0   05 2A 2A 2A 2A 2A 00 00 FF FF FF FF FF FF FF FF
```

7F FF end of HEAP (see DAInamic 7, page 188)
09 the coming Basic line contains 9 bytes from 00 to 2A
00 0A line number 10
A9 code for MEM
05 5 items come after the REM
2A ... 2A five asterisks

2. In place of five times #2A we can choose 5 other bytes which form a machine language program. This program can then be called up with CALLM #3F1, provided that two conditions are satisfied:-
- the program begins with 10 REM
- the text begins at address #3EC; this is certain to be so after power-up or a hard reset. No CLEAR instruction may be given prior to the first Basic line. In the listing the first line contains a series of arbitary characters. Sometimes there is nothing more in the line. If for example in the little bit of machine language is the byte #0C, then when listed the screen would be cleared.

3. An Example. The following program puts all characters on the screen, from CHR$(#FF),up to CHR$(#0C),clear screen.

```
1   03F1   C5       PUSH B       contents of
2   03F2   D5       PUSH D       registers
3   03F3   E5       PUSH H       saved on
4   03F4   F5       PUSH PSW     stack
5   03F5   060C     MVI  B,:0C   CHR$(12) in B
6   03F7   3EFF     MVI  A,:FF   #FF in A
7   03F9   CD60DD   CALL :DD60   CHR$(in A) to screen
8   03FC   3D       DCR  A       (A)=(A)-1
9   03FD   B8       CMP  B       if (A)-(B)#0 then
A   03FE   C2F903   JNZ          jump to #3F9
B   0401   F1       POP  PSW     fetch
C   0402   E1       POP  H       back
D   0403   D1       POP  D       registers
E   0404   C1       POP  B       from stack
F   0405   C9       RET
```

The program contains 21 bytes, so the basic program begins with:

```
10  REM **** 21 asterrisks ****
20  CALLM #3F1
30  END
```

By using the substitute instruction in Utility at addresses #3F1 to #405 all #2A bytes can be replaced by the 21 bytes of the program. The whole can now be saved and loaded as Basic. REM statement can contain up to 122 items, so the machine language can be a fair length.

4. The second proviso in section 2 (start text at #3EC) can cause problems. If the heap space is made larger than #100 bytes with a CLEAR then the start of the text buffer will be moved to a higher address. This address can again be found in #29F and #2A0, where the contents are now the new low and high address bytes for the start of the text buffer. Th example will now be changed, as follows, for eventual enlargement of the heap:

```
10  REM ....
20  ADRES= PEEK (#2A0)*256+PEEK (#29F)+5
30  CALLM ADRES
40  END
```

We first look at #29F and #2A0 where the text buffer begins. By counting up 5 from the start address (see section 1) we find the start address of the machine language program.

5. Alas, as we try one thing or another with our sample program things go wrong so that only that well known reset button brings a solution. In the machine language program there is a jump back from line A to line 7. In the jump instruction address #3F9 is mentioned, but that is no longer valid as the text buffer and therefore the machine language program were moved to higher addresses. A simple solution for this would be an instruction to jump back so many addresses, but unfortunately such a relative jump instruction is not available on the 8080A. And so as a beginner you come to a full stop with your machine language program. After a couple of telephone calls I got the following tips from Freddy de Roat:
- calculate the address to which you want to jump and put it in registers HL
- then put the contents of HL on the stack (PUSH H)
- now comes a conditional jump, e.g. a test for zero. You can then jump with a Return Non Zero

- or carry on with POP H by means of which the stack is returned again to its former size.
The fact is that calculating the jump address from the addresses resident in #29F and #2A0 (pointers to the text buffer), is no longer possible because the pointers are changed by a CLEAR and therefore the jump address will also be changed.

6. Example

```
 1   03F1   C5        PUSH B
 2   03F2   D5        PUSH D
 3   03F3   E5        PUSH H
 4   03F4   F5        PUSH PSW
 5   03F5   2A9F02    LHLD :029F      calculate
 6   03F8   111400    LXI  D,:0014    jump
 7   03FB   19        DAD  D          address
 8   03FC   060C      MVI  B,:0C
 9   03FE   3EFF      MVI  A,:FF
 A   0400   CD60DD    CALL :DD60
 B   0403   3D        DCR  A
 C   0404   B8        CMP  B
 D   0405   E5        PUSH H
 E   0406   C0        RNZ             relative jump
 F   0407   E1        POP  H
10  0408   F1E1D1C1  POP  ALL
11  040C   C9        RET
```

Explanation:
Line 5 - Start address of text buffer is read at #029F, #2A0 and put into registers HL
Line 6 - The destination address for the jump lies #400-#3EC = #14 addresses further on from the start address of the text buffer. Therefore #0014 goes into registers D,E.
Line 7 - after counting up , the result (i.e. the jump address) is in registers HL
Line D - The jump address is taken from HL and put on the stack
Line E - If (A)-(B)≠0 then return to the above address which is on the stack.

Now, should the HEAP be enlarged by an earlier program, or a CLEAR be given before loading, your program will still run O.K. You are so pleased that it works that you write it all down before you forget! It could be that someone else will benefit.

Greetings Thijs Berckx

contactaddress U.K. : Dave Atherton
16 Douglas Street
ATHERTON MANCHESTER M29 9FB
U.K. tel : 44-942 876210

Stand Ø1.März 1983. Alle genannten Preise sind einschließlich Mehrwertsteuer.Der Versand erfolgt ausschließlich per Nachnahme zuzüglich Versandkosten.Es gelten die allgemeinen Geschäftsbedingen nach BGB. Bei Versand ins Ausland wird die Mehrwertsteuer abgezogen.Lieferzeiten bis 7 Wochen vorbehalten.

Real World Karten Adapter mit je 4 Buchsen-u. 4Stiftleisten (31pol,4 Steckpl.), 34 pol.Stiftstecker zum Anschluß an den DCE-Bus,Karte gelötet und getestet
DSP-RWC1 75.ØØ DM

In Vorbereitung:
RWC-A/D-Wandler
RWC-Relaiskarte
RWC-Druckerinterface mit 3K Buffer

Epromprogrammier programmiert 27Ø8,2716,2732 2516 und 2532,arbeitet ohne zusätzliche Spannung,bei KENDOS durch Benutzung eines Epromplatzes, normal Epromhalter erforderlich, komfort. Software,(Prüfen,Check,Vergleichen),m.Bedienungsanl.
DSP-EPP 17Ø.ØØ DM

RWC-Netzteilkarte, +5V/3A, +12V/2A,-5V/1A, 5Ø Hz und 1ØØ Hz Taktausgang,Karte gelötet und getestet, erf.Trafo 8V/3A, 8V/1A, 16V/2A
DSP-NT1 85.ØØ DM

RWC-Adapteranschlußkabel zum Anschluß von RWC1 an den DAI, wird nicht benötigt wenn am DCR-Kabel eine Buchse frei
DSP- RWCKAB 3Ø.ØØ DM

Epromhalter extra
DSP-EPH1 35.ØØ DM
In Vorbereitung:
Epromhalter für DCR und Epromprogrammier, ermögl.Betrieb vom DSP-EPP m.DCR,CAS u. DAI-Floppy

Eprom 2716 DSP-2716 6.9Ø DM
Eprom 2732 DSP-2732 14.ØØ DM
Eprom 2732A DSP-2732A 15.2Ø DM

RWC-Real-Time-Clock Quarzst.,Sec,Min,Std,Tage Wochentage,Monate,Jahr autom.Schaltjahrkorrektur, Start/Stop-Funktion p.Software,12/24 Std. selektierbar Notstromversorgung,1 Sekunden-Ausgang, Interrupt p.Softw. selektierb., alle Sek/Min/Sdt/ 1/1Ø24 Sek.,2 Treiber f.Schaltausgang,Kartenadresse wählbar, umfangreiche Dokumentation
DSP-RTC 185.ØØ DM

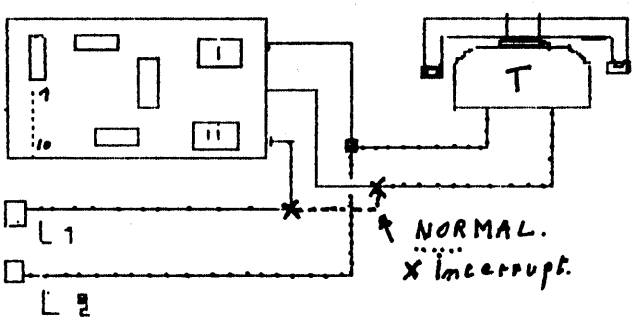
D 8255 AC-5 DSP-8255 8.7Ø DM
AMD 9511 DSP-AMD 33Ø.ØØ DM

RGB-Monitor,15 MHz Auflösung, weißes Geh. DSP-CMB2 129Ø.ØØ DM

Diskette, 5.25", 1.Wahl, solange Vorrat reicht DSP-DIS1 6.ØØ DM

Tastatur DSP-KBS 197.ØØ DM

E.NEVE 25 Hoog straat DWORP 1512.T.02/380 35 81.

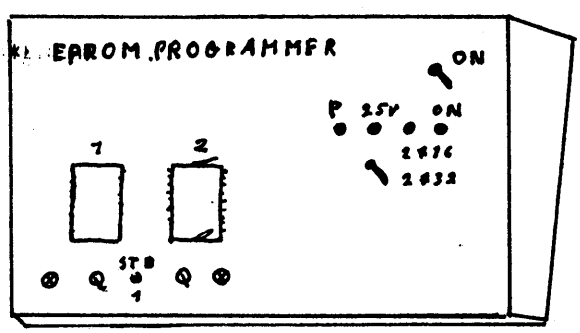


CIRCUIT TELEPHONIQUE

CE CIRCUIT Permet de former par COMPUTER des numeros de Telephone grace a un integre specialise pour ce travail.Le DAI ou un AUTRE peut Commander le systeme;il doit pour cela avoir u interf.8 SORTIES . Le programm met des NIVEAUX B C D aux entrees du circuit.

N'importe quel No de TEL.peut entrer.La MEMOIRE INTERNE est de 16 CHIFFRES.

##



* EPROM PROGRAMMER *

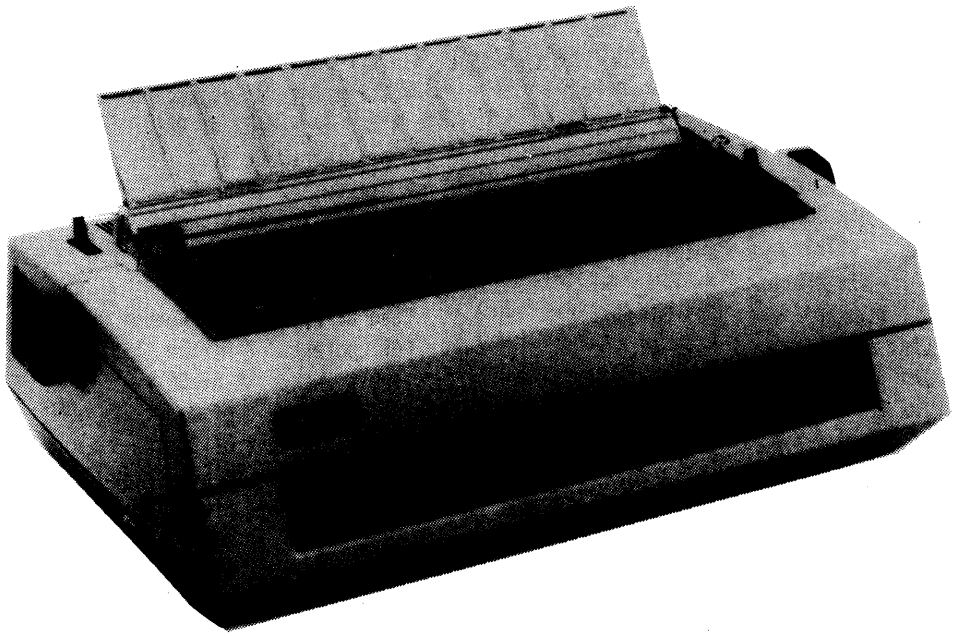
SYSTEM for CHECK READ PROGRAMM an COPY OF 2716 2732 MEMORIES.

CARACTERISTICS.

220 AC. POWERED 5 WATTS.
MEMORIES CAN BE SET IN STAND BY INDIVIDUALLY WITH MANUAL SWITCHES AND WITH THE PROGRAMM.

---

daisy wheel printer HR-1



47500,- BFr. EXCL. BTW

INCLUSIEF 2 JAAR GARANTIE EN SERVICECONTRACT

- RS-232 of Centronics parallel interface
- Instelbare pitch (10/12 of 15 Kar/inch)
- Print Bidirectioneel met Logic seeking
- Snelheid 40 karakters/seconde nominaal
- Standaard geleverd met Friction feed
- Als optie een Tractorfeed of Sheetfeeder
- Wagenbreedte 420 mm. (195 pos.maximaal)
- Werkt met standaard IBM lintcassette's
- Ruime keus in letterwielen (ca. 20 stuks)

NEDERLAND
HERENGRACHT 317
1016 AV AMSTERDAM
TEL : (020) 22 41 33

BELGIE
FRANKRIJKLEI 70
2000 ANTWERPEN
TEL : (03) 2334088

8080 ASSEMBLY LANGUAGE PROGRAMMING FOR BEGINNERS - OP CODES

HEX	MNEMONIC	FLAGS	BASIC EQUIVALENT	DESCRIPTION
00	NOP	0	-	Does nothing
01	LXI B,dddd	0	LET BC=dddd	Load BC with 16-bit number dddd
02	STAX B	0	POKE BC,A	Put contents of A into 16-bit memory address in BC
03	INX B	0	LET BC=BC+1	16-bit increment of BC
04	INR B	X	LET B=B+1	8-bit increment of B
05	DCR B	X	LET B=B-1	8-bit decrement of B
06	MVI B,dd	0	LET B=dd	Load B with 8-bit number dd
07	RLC	C	LET A=(A*2) MOD 256	Move all bits of A one to the left. Carry=Bit 7;Bit 0=Bit 7
08	-	0	-	-
09	DAD B	0	LET HL=(HL+BC) MOD 65536	16 bit-addition of HL and BC:result in HL
0A	LDAX B	0	LET A=PEEK(BC)	Put contents of memory address in BC into A
0B	DCX B	0	LET BC=BC-1	16-bit decrement of BC
0C	INR C	X	LET C=C+1	8-bit increment of C
0D	DCR C	X	LET C=C-1	8-bit decrement of C
0E	MVI C,dd	0	LET C=dd	Load C with 8-bit number dd
0F	RRC	C	A=INT(A/2)+128*Bit 0 of A	Move all bits of A one to the right. Carry=Bit 0;Bit 7=Bit 0
10	-	0	-	-
11	LXI D,dddd	0	LET DE=dddd	Load DE with 16-bit number dddd
12	LDAX D	0	POKE DE,A	Put contents of A into 16-bit memory address in DE
13	INX D	0	LET DE=DE+1	16-bit increment of DE
14	INR D	X	LET D=D+1	8-bit increment of D
15	DCR D	X	LET D=D-1	8-bit decrement of D
16	MVI D,dd	0	LET D=dd	Load D with 8-bit number dd
17	RAL	C	LET A=(A*2) MOD 256	As RLC (07) but Bit 0=Carry
18	-	0	-	-
19	DAD D	C	LET HL=(HL+DE) MOD 65536	16-bit addition of HL and BC:result in HL
1A	LDAX D	0	LET A=PEEK(DE)	Put contents of memory address in DE into A
1B	DCX D	0	LET DE=DE-1	16-bit decrement of DE
1C	INR E	X	LET E=E+1	8-bit increment of E
1D	DCR E	X	LET E=E-1	8-bit decrement of E
1E	MVI E,dd	0	LET E=dd	Load E with 8-bit number
1F	RAR	C	A=INT(A/2)+128*Carry	As RRC (0F) but Bit 7=Carry
20	-	0	-	-
21	LXI H,dddd	0	LET HL=dddd	Load HL with 16-bit number dddd
22	SHLD dddd	0	POKE dddd,L:POKE dddd+1,H	Put 16-bit number in HL into memory location dddd and dddd+1
23	INX H	0	LET HL=HL+1	16-bit increment of HL
24	INR H	X	LET H=H+1	8-bit increment of H
25	DCR H	X	LET H=H-1	8-bit decrement of H
26	MVI H,dd	0	LET H=dd	Load H with 8-bit number dd
27	DAA	A	-	Switch the 8080 to BCD mode for next add or subtract
28	-	0	-	-
29	DAD H	C	LET HL=HL*2 MOD 65536	16 bit addition of HL and HL:result in HL (therefore double HL)
2A	LHLD dddd	0	H=PEEK dddd:L=PEEK dddd+1	Put 16-bit number contained in dddd and dddd+1 into HL
2B	DCX H	0	LET HL=HL-1	16-bit decrement of HL
2C	INR L	X	LET L=L+1	8-bit increment of L
2D	DCR L	X	LET L=L-1	8-bit decrement of L
2E	MVI L,dd	0	LET L=dd	Load L with 8-bit number dd
2F	CMA	0	LET A=255-A	Complement A. Turn 0 bits to 1, 1 bits to 0
30	-	0	-	-
31	LXI SP,dddd	0	LET SP=dddd	Load the stack pointer with the 16-bit number dddd
32	STA dddd	0	POKE dddd,A	Load memory address dddd with the contents of A
33	INX SP	0	LET SP=SP+1	16-bit increment of the stack pointer
34	INR M	X	POKE HL,PEEK(HL)+1	8-bit increment of the memory location in HL (HL itself unaffected)
35	DCR M	X	POKE HL,PEEK(HL)-1	8-bit decrement of the memory location in HL (HL itself unaffected)
36	MVI M,dd	0	POKE HL,dd	Load memory location HL with 8-bit number dd
37	STC	C	Carry=1	Set carry flag
38	-	0	-	-

dd=any 8 bit number (0-FF). dddd=any 16-bit number (0-FFFF). ss=number of bytes to jump between -128 and 127

8080 ASSEMBLY LANGUAGE PROGRAMMING FOR BEGINNERS - OP CODES

HEX	MNEMONIC	FLAGS	BASIC EQUIVALENT	DESCRIPTION
39	DAD SP	C	LET HL=HL+SP	16-bit addition of HL and stack pointer; result in HL
3A	LDA dddd	0	LET A=PEEK(dddd)	Load A with the contents of memory address dddd
3B	DCX SP	0	LET SP=SP-1	16-bit decrement of the stack pointer
3C	INR A	X	LET A=A+1	8-bit increment of A
3D	DCR A	X	LET A=A-1	8-bit decrement of A
3E	MVI A,dd	0	LET A=dd	Load A with the 8-bit number dd
3F	CMC	C	LET Carry=1-Carry	Flip the carry flag over to its other state
40	MOV B,B	0	LET B=B	Copy the contents of the B register into the B register
41	MOV B,C	0	LET B=C	Copy the contents of the C register into the B register
42	MOV B,D	0	LET B=D	Copy the contents of the D register into the B register
43	MOV B,E	0	LET B=E	Copy the contents of the E register into the B register
44	MOV B,H	0	LET B=H	Copy the contents of the H register into the B register
45	MOV B,L	0	LET B=L	Copy the contents of the L register into the B register
46	MOV B,M	0	LET B=PEEK(HL)	Copy the contents of the memory addressed by HL into the B register
47	MOV B,A	0	LET B=A	Copy the contents of the A register into the B register
48	MOV C,B	0	LET C=B	Copy the contents of the B register into the C register
49	MOV C,C	0	LET C=C	Copy the contents of the C register into the C register
4A	MOV C,D	0	LET C=D	Copy the contents of the D register into the C register
4B	MOV C,E	0	LET C=E	Copy the contents of the E register into the C register
4C	MOV C,H	0	LET C=H	Copy the contents of the H register into the C register
4D	MOV C,L	0	LET C=L	Copy the contents of the L register into the C register
4E	MOV C,M	0	LET C=PEEK(HL)	Copy the contents of the memory addressed by HL into the C register
4F	MOV C,A	0	LET C=A	Copy the contents of the A register into the C register
50	MOV D,B	0	LET D=B	Copy the contents of the B register into the D register
51	MOV D,C	0	LET D=C	Copy the contents of the C register into the D register
52	MOV D,D	0	LET D=D	Copy the contents of the D register into the D register
53	MOV D,E	0	LET D=E	Copy the contents of the E register into the D register
54	MOV D,H	0	LET D=H	Copy the contents of the H register into the D register
55	MOV D,L	0	LET D=L	Copy the contents of the L register into the D register
56	MOV D,M	0	LET D=PEEK(HL)	Copy the contents of the memory addressed by HL into the D register
57	MOV D,A	0	LET D=A	Copy the contents of the A register into the D register
58	MOV E,B	0	LET E=B	Copy the contents of the B register into the E register
59	MOV E,C	0	LET E=C	Copy the contents of the C register into the E register
5A	MOV E,D	0	LET E=D	Copy the contents of the D register into the E register
5B	MOV E,E	0	LET E=E	Copy the contents of the E register into the E register
5C	MOV E,H	0	LET E=H	Copy the contents of the H register into the E register
5D	MOV E,L	0	LET E=L	Copy the contents of the L register into the E register
5E	MOV E,M	0	LET E=PEEK(HL)	Copy the contents of the memory addressed by HL into the E register
5F	MOV E,A	0	LET E=A	Copy the contents of the A register into the E register
60	MOV H,B	0	LET H=B	Copy the contents of the B register into the H register
61	MOV H,C	0	LET H=C	Copy the contents of the C register into the H register
62	MOV H,D	0	LET H=D	Copy the contents of the D register into the H register
63	MOV H,E	0	LET H=E	Copy the contents of the E register into the H register
64	MOV H,H	0	LET H=H	Copy the contents of the H register into the H register
65	MOV H,L	0	LET H=L	Copy the contents of the L register into the H register
66	MOV H,M	0	LET H=PEEK(HL)	Copy the contents of the memory addressed by HL into the H register
67	MOV H,A	0	LET H=A	Copy the contents of the A register into the H register
68	MOV L,B	0	LET L=B	Copy the contents of the B register into the L register
69	MOV L,C	0	LET L=C	Copy the contents of the C register into the L register
6A	MOV L,D	0	LET L=D	Copy the contents of the D register into the L register
6B	MOV L,E	0	LET L=E	Copy the contents of the E register into the L register
6C	MOV L,H	0	LET L=H	Copy the contents of the H register into the L register
6D	MOV L,L	0	LET L=L	Copy the contents of the L register into the L register
6E	MOV L,M	0	LET L=PEEK(HL)	Copy the contents of the memory addressed by HL into the L register
6F	MOV L,A	0	LET L=A	Copy the contents of the A register into the L register

dd=any 8 bit number (0-FF). dddd=any 16-bit number (0-FFFF)

8080 ASSEMBLY LANGUAGE PROGRAMMING FOR BEGINNERS - OP CODES

HEX	MNEMONIC	FLAGS	BASIC EQUIVALENT	DESCRIPTION
70	MOV M,B	0	POKE HL,B	Copy contents of B register into the memory location addressed by HL
71	MOV M,C	0	POKE HL,C	Copy contents of C register into the memory location addressed by HL
72	MOV M,D	0	POKE HL,D	Copy contents of D register into the memory location addressed by HL
73	MOV M,E	0	POKE HL,E	Copy contents of E register into the memory location addressed by HL
74	MOV M,H	0	POKE HL,H	Copy contents of H register into the memory location addressed by HL
75	MOV M,L	0	POKE HL,L	Copy contents of L register into the memory location addressed by HL
76	HLT	0	STOP	Stop the processor. Can only be restarted by interrupts
77	MOV M,A	0	POKE HL,A	Copy the contents of A register into memory location addressed by HL
78	MOV A,B	0	LET A=B	Copy the contents of the B register into the A register
79	MOV A,C	0	LET A=C	Copy the contents of the C register into the A register
7A	MOV A,D	0	LET A=D	Copy the contents of the D register into the A register
7B	MOV A,E	0	LET A=E	Copy the contents of the E register into the A register
7C	MOV A,H	0	LET A=H	Copy the contents of the H register into the A register
7D	MOV A,L	0	LET A=L	Copy the contents of the L register into the A register
7E	MOV A,M	0	LET A=PEEK(HL)	Copy contents of memory location addressed by HL into the A register
7F	MOV A,A	0	LET A=A	Copy the contents of the A register into the A register
80	ADD B	A	LET A=A+B	8-bit addition of A and B:result in A
81	ADD C	A	LET A=A+C	8-bit addition of A and C:result in A
82	ADD D	A	LET A=A+D	8-bit addition of A and D:result in A
83	ADD E	A	LET A=A+E	8-bit addition of A and E:result in A
84	ADD H	A	LET A=A+H	8-bit addition of A and H:result in A
85	ADD L	A	LET A=A+L	8-bit addition of A and L:result in A
86	ADD M	A	LET A=A+PEEK(HL)	8-bit addition of A and contents of memory addressed by HL:result in A
87	ADD A	A	LET A=A+A	8-bit addition of A to itself i.e. A=A*2:result in A
88	ADC B	A	LET A=A+B+Carry	As 80 but plus 1 if carry set
89	ADC C	A	LET A=A+C+Carry	As 81 but plus 1 if carry set
8A	ADC D	A	LET A=A+D+Carry	As 82 but plus 1 if carry set
8B	ADC E	A	LET A=A+E+Carry	As 83 but plus 1 if carry set
8C	ADC H	A	LET A=A+E+Carry	As 84 but plus 1 if carry set
8D	ADC L	A	LET A=A+L+Carry	As 85 but plus 1 if carry set
8E	ADC M	A	LET A=A+PEEK(HL)+Carry	As 86 but plus 1 if carry set
8F	ADC A	A	LET A=A+A+Carry	As 87 but plus 1 if carry set
90	SUB B	A	LET A=A-B	8-bit subtraction of B from A:result in A
91	SUB C	A	LET A=A-C	As 90 but with C
92	SUB D	A	LET A=A-D	As 90 but with D
93	SUB E	A	LET A=A-E	As 90 but with E
94	SUB H	A	LET A=A-H	As 90 but with H
95	SUB L	A	LET A=A-L	As 90 but with L
96	SUB M	A	LET A=A-PEEK(HL)	8-bit subtraction of contents of memory addressed by HL from A:res.in A
97	SUB A	A	LET A=A-A	Subtract A from itself i.e. A=0
98	SBB B	A	LET A=A-B-Carry	As 90 but less 1 if carry set
99	SBB C	A	LET A=A-C-Carry	As 91 but less 1 if carry set
9A	SBB D	A	LET A=A-D-Carry	As 92 but less 1 if carry set
9B	SBB E	A	LET A=A-E-Carry	As 93 but less 1 if carry set
9C	SBB H	A	LET A=A-H-Carry	As 94 but less 1 if carry set
9D	SBB L	A	LET A=A-L-Carry	As 95 but less 1 if carry set
9E	SBB M	A	LET A=A-PEEK(HL)-Carry	As 96 but less 1 if carry set
9F	SBB A	A	LET A=A-A-Carry	As 97 but less 1 (ie -1) if carry set
A0	ANA B	A	LET A=A AND B	Logical AND A with B:result in A
A1	ANA C	A	LET A=A AND C	As A0 but with C
A2	ANA D	A	LET A=A AND D	As A0 but with D
A3	ANA E	A	LET A=A AND E	As A0 but with E
A4	ANA H	A	LET A=A AND H	As A0 but with H
A5	ANA L	A	LET A=A AND L	As A0 but with L
A6	ANA M	A	LET A=A AND PEEK(HL)	Logical AND A with the contents of memory addressed by HL
A7	ANA A	A	LET A=A AND A	Logical AND A with itself:A will remain unchanged but flags will alter

8080 ASSEMBLY LANGUAGE PROGRAMMING FOR BEGINNERS - OP CODES

HEX	MNEMONIC	FLAGS	BASIC EQUIVALENT	DESCRIPTION
A8	XRA B	A	LET A=A XOR B	Logical XOR (Exclusive-OR) A with B:result in A
A9	XRA C	A	LET A=A XOR C	As A8 but with C
AA	XRA D	A	LET A=A XOR D	As A8 but with D
AB	XRA E	A	LET A=A XOR E	As A8 but with E
AC	XRA H	A	LET A=A XOR H	As A8 but with H
AD	XRA L	A	LET A=A XOR H	As A8 but with L
AE	XRA M	A	LET A=A XOR PEEK(HL)	As A8 but with contents of memory addressed by HL
AF	XRA A	A	LET A=A XOR A	As A8 but with itself ie.LET A=0
B0	ORA B	A	LET A=A OR B	Logical OR A with B
B1	ORA C	A	LET A=A OR C	As B0 but with C
B2	ORA D	A	LET A=A OR D	As B0 but with D
B3	ORA E	A	LET A=A OR E	As B0 but with E
B4	ORA H	A	LET A=A OR H	As B0 but with H
B5	ORA L	A	LET A=A OR L	As B0 but with L
B6	ORA M	A	LET A=A OR PEEK(HL)	As B0 but with contents of memory addressed by HL
B7	ORA A	A	LET A=A OR A	As B0 but with itself:A will remain unchanged but flags will alter
B8	CMP B	A	A-B	Tests A-B and sets flags accordingly. No registers altered
B9	CMP C	A	A-C	As B8 but for A-C
BA	CMP D	A	A-D	As B8 but for A-D
BB	CMP E	A	A-E	As B8 but for A-E
BC	CMP H	A	A-H	As B8 but for A-H
BD	CMP L	A	A-L	As B8 but for A-L
BE	CMP M	A	A-PEEK(HL)	As B8 but for A-memory location addressed by HL
BF	CMP A	A	A-A	As B8 but with itself:Will always set zero flag
C0	RNZ	0	IF Zero=0 THEN RETURN	If zero flag not set return from subroutine
C1	POP B	A	B=PEEK(SP+1):C=PEEK(SP)	Take top two bytes off stack and put into BC:SP=SP+2
C2	JNZ dddd	0	IF Zero=0 GOTO dddd	If zero flag not set jump to address dddd
C3	JMP dddd	0	GOTO dddd	Jump to address dddd
C4	CNZ,dddd	0	IF Zero=0 GOSUB dddd	If zero flag not set jump to subroutine:put PC+3 on stack:SP=SP-2
C5	PUSH B	0	POKE SP+1,B:POKE SP,C	Put contents of BC onto stack:SP=SP-2
C6	ADI dd	A	LET A=A+dd	8-bit addition of A and 8-bit number dd:result in A
C7	RST 0	0	GOSUB 0	Jump to subroutine at address 0000:put PC+1 on stack:SP=SP-2
C8	RZ	0	IF Zero=1 THEN RETURN	If zero flag set return from subroutine
C9	RET	0	RETURN	Return from subroutine (by popping return address off stack)
CA	JZ dddd	0	IF Zero=1 GOTO dddd	If zero flag set jump to address dddd
CB	-	0	-	-
CC	CZ dddd	0	IF Zero=1 GOSUB dddd	If zero flag set jump to subroutine:put PC+3 on stack:SP=SP-2
CD	CALL dddd	0	GOSUB dddd	Jump to subroutine at address dddd:put PC+3 on stack:SP=SP-2
CE	ACI dd	A	LET A=A+dd+Carry	As C6 but plus 1 if carry flag set:result in A
CF	RST 1	0	GOSUB 8	As C7 but address is 0008 (hex)
D0	RNC	0	IF Carry=0 THEN RETURN	As C0 but carry flag not zero flag is tested
D1	POP D	A	D=PEEK(SP+1):E=PEEK(SP)	As C1 but register-pair involved is DE
D2	JNC dddd	0	IF Carry=0 GOTO dddd	As C2 but carry flag not zero flag is tested
D3	OUT dd	0	OUT dd,A	Sends number in A to output port dd (up to 256 allowed)
D4	CNC dddd	0	IF Carry=0 GOSUB dddd	As C4 but carry flag not zero flag is tested
D5	PUSH D	0	POKE SP+1,D:POKE SP,E	As C5 but register-pair involved is DE
D6	SUI dd	A	LET A=A-dd	8-bit subtraction of 8-bit number dd from A:result in A
D7	RST 2	0	GOSUB 16	As C7 but address is 0010 (hex; 16 in decimal)
D8	RC	0	IF Carry=1 THEN RETURN	As C8 but carry flag not zero flag is tested
D9	-	0	-	-
DA	JC dddd	0	IF Carry=1 GOTO dddd	As CA but carry flag not zero flag is tested
DB	IN dd	0	LET A=INP(dd)	Read 8-bit number on input port dd into A
DC	CC dddd	0	IF Carry=1 GOSUB dddd	As CC but carry flag not zero flag is tested
DD	-	0	-	-
DE	SBI dd	A	LET A=A-dd-Carry	As D6 but less 1 if carry flag set:result in A
DF	RST 3	0	GOSUB 24	As C7 but address is 0018 (hex; 24 in decimal)

dd=any 8-bit number (0-FF). dddd=any 16-bit number (0-FFFF)

8080 ASSEMBLY LANGUAGE PROGRAMMING FOR BEGINNERS - OP CODES

HEX	MNEMONIC	FLAGS	BASIC EQUIVALENT	DESCRIPTION
E0	RPO	0	IF Parity=0 THEN RETURN	As C0 but parity flag not zero flag is tested
E1	POP H	A	H=PEEK(SP+1):L=PEEK(SP)	As C1 but register-pair involved is HL
E2	JPO dddd	0	IF Parity=0 GOTO dddd	As C2 but parity flag not zero flag ia tested
E3	XTHL	0	As POP HL(E1)+PUSH HL(E5)	Contents of HL+top 2 bytes on stack exchanged:no change in SP
E4	CPO dddd	0	IF Parity=0 GOSUB dddd	As C4 but parity flag not zero flag is tested
E5	PUSH H	0	POKE SP+1,D:POKE SP,E	As C5 but register-pair involved is HL
E6	AND dd	A	LET A=A AND dd	Logical AND A with 8-bit number dd:result in A
E7	RST 32	0	GOTO 32	As C7 but address is 0020 (hex; 32 in decimal)
E8	RPE	0	IF Parity=1 THEN RETURN	As C8 but parity flag not zero flag is tested
E9	PCHL	0	GOTO HL	Jump to the memory location addressed by HL
EA	JPE dddd	0	IF Parity=1 GOTO dddd	As CA but parity flag not zero flag is tested
EB	XCHG	0	HL=DE:DE=HL	Exchange 16-bit numbers in DE and HL (D=H:E=L:H=D:L=E)
EC	CPE dddd	0	IF Parity=1 GOSUB dddd	As CC but parity flag not zero flag is tested
ED	-	-	-	-
EE	XRI dd	A	LET A=A XOR dd	Logical XOR (Exclusive-OR) A with 8-bit number dd
EF	RST 5	0	GOSUB 40	As C7 but address is 0028(hex; 40 in decimal)
F0	RP	0	IF Sign=0 THEN RETURN	As C0 but sign flag not zero flag is tested
F1	POP PSW	A	A=PEEK SP+1:Flags=PEEK SP	As C1 but register-pair involved is A+Flags. A is the high-order byte
F2	JP dddd	0	IF Sign=0 GOTO dddd	As C2 but sign flag not zero flag is tested
F3	DI	0	-	Disable interrupts
F4	CP dddd	0	IF Sign=0 GOSUB dddd	As C4 but sign flag not zero flag is tested
F5	PUSH PSW	0	POKE SP+1,A:POKE SP,Flags	As C5 but register-pair involved is A+Flags. A is the high-order byte
F6	ORI dd	A	LET A=A OR dd	Logical OR A with 8-bit number dd:result in A
F7	RST 6	0	GOSUB 48	As C7 but address is 0030(hex; 48 decimal)
F8	RM	0	IF Sign=1 THEN RETURN	As C8 but sign flag not zero flag is tested
F9	SPHL	0	LET SP=HL	Load the stack pointer with the 16-bit contents of HL
FA	JM dddd	0	IF Sign=1 GOTO dddd	As CA but sign flag not zero flag is tested
FB	EI	0	-	Enable interrupts
FC	CM dddd	0	IF Sign=1 GOSUB dddd	As CC but sign flag not zero flag is tested
FD	-	-	-	-
FE	CPI dd	A	A-dd	Tests A minus dd and sets flags accordingly. No registers altered
FF	RST 7	0	GOSUB 56	As C7 but address is 0038(hex ; 56 decimal)

Notes:
dd is any 8-bit number (0-FF)
dddd is any 16-bit number (0-FFFF). All memory addresses are 16-bit.
FLAGS:A=This operation affects all flags
C=This operation affects the carry flag only
X=This operation affects all flags except the carry
0=This operation has no effect on the flags
Sign : Set if A<0 (in two's complement i.e. 80 - FF)
Zero : Set if A=0
Parity: Set if there is an even number of 1's in the binary representation of A
Carry : Set if the last operation caused A to go below 0 or above 255

All instructions that have 'dd' in them are followed by a one-byte number in the machine code program. Therefore the next instruction is one memory address after that. Instructions containing 'dddd' have a 16-bit number following them in the program and this of course needs two byte. Therefore the next instruction will be taken as that in the 'next memory location but two'. Also note that where 16-bit numbers are included in the program, they should be entered with the low-order byte first. i.e. to load HL with 9ABC, the code is 21 BC 9A not 21 9A BC.

n.b. The BASIC used to describe the operations is not always exactly accurate, but it is as close as it is possible to get to simulate the instruction. Note that on the DAI pc, the operators IXOR, IAND and IOR should be read for the BASIC equivalents of the XRA/I, ANA/I and ORA/I instructions.

Dave Atherton
10 April 1983

```
5     REM CHARACTER INVASION W.Hermans
10    CLEAR 2000:GOSUB 10000:GOTO 30
15    POS=#B3DD-KPOS*2:POKE POS,32:POKE POS+134,32:RETURN
20    POS=#B3DD-KPOS*2:POKE POS,255:POKE POS+134,11:RETURN
30    V=RND(26)+65:NOISE 0 15
35    POINT=POINT+1
40    R=1+RND(60):COL=TOP-R-R:SOUND OFF
50    IF A(R)=0 THEN C(R)=V:A(R)=A(R)+134:POKE COL-A(R),V:GOTO 60
55    IF A(R)<>0 THEN POKE COL-A(R),32:POKE COL-A(R)-3,#FF:A(R)=A(R)+134:PO
KE TOP-R*2-A(R),C(R)
56    IF A(R)>2680 THEN 2000:REM END
60    G=GETC
65    IF G<>18 THEN IF G<>19 THEN IF G<>C(KPOS) THEN WAIT TIME 2:GOTO 30
70    IF G=18 THEN IF KPOS>1 THEN GOSUB 15:KPOS=KPOS-1:GOSUB 20
80    IF G=19 THEN IF KPOS<60 THEN GOSUB 15:KPOS=KPOS+1:GOSUB 20
85    IF G=C(KPOS) THEN GOSUB 100
90    GOTO 30
100   NOISE 1 15:FOR X=A(KPOS) TO 0 STEP -134:SOUND 1 0 15 0 FREQ(50.0+X)
110   POKE TOP-KPOS*2-X,32:POKE TOP-KPOS*2-X-3,0
120   NEXT:SOUND OFF :A(KPOS)=0
125   NOISE OFF
130   RETURN
2000  PRINT CHR$(12);
2010  CURSOR 21,10:PRINT POINT;" POINTS"
2015  CURSOR 15,8:PRINT "space-bar to play again"
2020  G=GETC:IF G=0 THEN 2020
2030  IF G=32 THEN 10
2040  END
10000 MODE 0:PRINT CHR$(12)
10010 COLORT 1 15 3 0
10015 FOR X=0 TO 15:POKE #BFEF-X*134,#70+X:NEXT
10020 TOP=#BFE7
10025 POKE #BFEE-23*134,#C0
10030 POKE #75,32
10040 DIM A(60.0)
10047 DIM C(60.0)
10050 KPOS=30:GOSUB 20
10052 POINT=0
10055 ENVELOPE 0 15
10060 RETURN
```

move base with cursor left/right,
shoot invader by typing the character.

Datum :5/4/83 **Programm :** CHARACTER INVASION W.Hermans

A % ()	array, holding offset of characters (invaders) from upper line	50,55,56,100,120,10040,
C % ()	array, holding the characters from each column	50,55,65,85,10047,
COL %	actual column to act upon	40,50,55,
G %	GETC-variable	60,65,70,80,85,2020,2030,
KPOS %	position of defender (1-60)	15,20,65,70,80,85,100,110,120,10050,
POINT %	score	35,2010,10052,
POS %	real address of defender	15,20,
R %	random value for column-choice	40,50,55,56,
TOP %	top-left character address (= H BFE7)	40,55,110,10020,
V %	random character (ASCII-value)	30,50,
X %	loop-variable	100,110,10015,

Een vraag die ons vaak gesteld wordt is :Hoe schrijf je nu een programma in machinetaal? Veelal met opmerkingen erbij als "Ik heb nu DNA gekocht maar weet niet goed wat ik er mee aanmoet." of "die nieuwe assembler, die SPL is beter he, moet ik die nu maar kopen ?"
Eerst even voor de goede orde de zaken op een rijtje zetten. In de DAI zit een microprocessor, de 8080A van de firma Intel al kan in uw DAI er best een zitten van NEC (licentie). Deze 8080 nu krijgt instructies die een,twee of drie bytes groot zijn. We kunnen de 8080 zelf direct instructies geven door eerst UT in te tikken en [return] natuurlijk en dan met bv S3000 de bytes vanaf #3000 te vullen met door ons gewenste instructies. U vult altijd maar een byte tikt U meer dan twee tekens (0 t/m F) dan neemt het monitor programma de laatste twee met de spatiebalk gaat U naar de volgende byte. Let er wel op dat de char.del. niet werkt ! Om het programma te laten uitvoeren kunt U terug in basic een CALLM#3000 geven zowel direct als in een programma.
Laten we het eens gaan proberen: machine aan en UT [return]
S3000 [spatie] monitor geeft huidige inhoud van 3000, U tikt C9 (de code voor return. U drukt cursor-left in en dan B en U bent weer terug in basic. Even proberen : CALLM#3000 en ja hoor met het sterretje van basic ziet U dat U uw eigen machine language program hebt laten uitvoeren en zonder ongelukken terug bent gekomen. Het nut van dit programma zal vrijwel een ieder ontgaan. Maar dat doet nu niet terzake. We komen straks op een nuttig programmaatje wat dan wel wat groter zal zijn.

Grotere programma's zijn op deze manier erg moeilijk te schrijven en men heeft daar wat op gevonden. Er is een programma gemaakt dat een voor mensen eenvoudige taal omzet in machine-codes. Dit is vooral voor referenties veel handiger. De taal die we gebruiken is echter een compromis. Enerzijds voor mensen gemakkelijker te begrijpen dan machinetaal-codes maar nog wel lastig omdat de machine het ook gemakkelijk wil hebben. De taal waar ik het hier over heb is assembler. Deze taal is het gemakkelijkst te beschouwen als machinetaal met normale engelse woorden of afkortingen daarvan als instructies. Een paar voorbeelden:

machine-taal	assembler	betekenis
3E 05	MVI A,05	zet de waarde 05 direct in register A
81	ADD C	tel de inhoud van C bij A en zet antwoord in A
CA 12 34	JZ LABEL	ga naar adres als de zeroflag aanstaat dwz als de uitkomst van de laatste instructie die de vlag kan zetten nul is.

Voor de instructie kan men in assembler een label zetten, zodat men daar naar toe kan gaan zoals bij basic het regelnummer.
PAS OP in het voorbeeld gaan we naar LABEL dit staat op #3412 en niet op #1234

Assembler is voor de machine vrij gemakkelijk te begrijpen dwz te vertalen in machine-code maar voor de mens nogal lastig. Doordat we zo dicht bij de machinecode staan zal meestal alles veel sneller gaan.
Om een programma te schrijven zullen we eerst precies moeten weten wat we willen maken. Is het programma niet te groot kunnen we het invoeren mbv Substitute of vanuit een basicprogramma mbv POKE. Als er echter sprongen in het programma zitten zullen ten eerste alle adressen zelf moeten uitrekenen en ten tweede zullen we veel adressen moeten veranderen als er achteraf wijzigingen in het programma komen. Ook de plaats waar het programma in het geheugen komt ligt op straffe van verandering van alle sprongadressen vast.

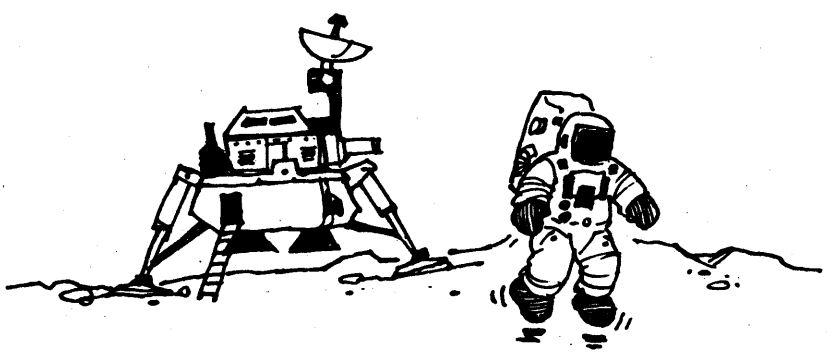
Als U dus wil gaan programmeren in machinecode/assembler is het sterk aan te raden een assembleerprogramma aan te schaffen. (DNA of SPL)

Basic is een taal die in structuur erg dicht bij assembler ligt. Iedereen weet dat het veel dichter bij de mens dan bij de machine staat maar de principieele opzet van de taal is toch vrij indentiek. Dit is een groot voordeel als we een combinatie van basic met machinetaal willen maken.
Ik kan me voorstellen dat sommige mensen het verschil tussen assembler en machinetaal weer onduidelijk gaat worden, daarom nog enige toelichting. We kunnen niet een basicprogramma met een assembler programma combineren omdat de vertaler alleen basic aankan en assembleerprogramma's alleen assembler. Wel zouden we eerst het basicdeel om kunnen zetten in machinecode dmv een compiler en vervolgens het assemblerdeel omzetten in machinecode en die dan samenvoegen. Dit is echter een omslachtige methode die erg veel problemen zal geven bij het samenvoegen. Daarbij wie bezit er een basiccompiler voor DAI ?
We zullen dus het liefst een basicprogramma schrijven en in dat basicprogramma indien dit voor de snelheid gewenst is dmv CALLM een machinetaalroutine aanroepen die het dan tijdelijk van basic overneemt. Deze machinetaalroutine kunnen we op een van de eerder besproken methodes op zijn plaats krijgen.

We gaan nu een klein handig programmaatje maken in machinetaal.
Ik heb hier gekozen voor een idee uit het vorige nummer; namelijk het op nul zetten van de adressen #2B1 t/m #2B8 zodat we altijd respons verkrijgen op een GETC en zo dan reactie van de machine krijgen zolang de toets ingedrukt blijft.
Voorbeeld in basic:

```
10      FOR I%=#2B1 TO #2B8:POKE I%,0:NEXT
20      H%=GETC:IF H%=0 GOTO 20
30      PRINT CHR$(H%);:GOTO 10
```

Opmerkingen bij dit programmaatje 1) alles in integer !!!!
2) in regel 20 naar 20 terug en niet naar 10 is wel goed maar trager en dat is juist in dit geval ongewenst.
3) programma stopt na 4 regels.
4) met rept werkt het trager !



First steps in machine language ...

Het op nul zetten kost tijd, zeker in basic, en we willen graag optimale snelheid. Als er op meerdere plaatsen in het programma een GETC gedaan wordt waar we deze 'rept' willen gebruiken ligt een subroutine voor de hand maar die vertraagt nog meer. We besluiten hiervoor een machinetaalroutinetje te schrijven.

We nemen de lijst van 8080 mnemonics voor ons en bestuderen die grondig. We stellen vast dat er aan programmeren in assembler heel wat meer vast zit dan in basic. Een eenvoudige loop moeten we zelf schrijven er bestaat niet zoiets als FOR - NEXT, we kunnen alleen maar optellen en aftrekken; vermenigvuldigen,delen machtsverheffen en vele andere zaken zullen we zelf moeten doen.
Maar niet te veel geklaagd, we behoeven maar 8 bytes op nul te zetten en dat kan toch niet al te veel problemen opleveren.

We selecteren na wat denkwerk de volgende instructie 'SHLD addr' die de inhoud van het H en L register op het aangegeven adres plaatst.
Dwz SHDL 3000 zet de inhoud van L op 3000 en de inhoud van H op 3001;omdat we H en L toch beide nul maken doet de volgorde er niet toe. Om de registers H e L op nul te zetten zijn er vele methodes, we kiezen voor een vrij gebruikelijke LXI H,0. Deze instructie vult de registers H en L met het adres (maar dat kan ook een gewone waarde zijn) dat er achter gegeven wordt. Ook hier weer de eerste byte in L en de tweede in H, voor ons niet van belang omdat beide met nul gevuld worden. Omdat we maar acht bytes te vullen hebben willen we geen loop maken maar rechttoe rechtaan programmeren. Dit biedt ons de volgende voordelen
- geen sprongadressen dus programma overal te plaatsen en zonder veranderingen te verplaatsen
- programma werkt nog sneller
- minder kans op fouten zeker zonder assembleerprogramma

We schrijven het programma:
In assembler :

```
    LXI     H,0
    SHLD    2B1
    SHLD    2B3
    SHLD    2B5
    SHLD    2B7
    RET
```

In machinecode wordt dit 21 00 00 22 B1 02 22 B3 02 22 B5 02 22 B7 02 C9

Als we dit programma ergens in het geheugen neerzetten is er echter toch een grote kans op een crash. Als we namelijk terugkeren naar basic staan de registers H en L op een andere waarde dan zij aan het begin van de routine en om alle eventuele problemen te voorkomen beginnen we onze routine met PUSH H en eindigen we hem met POP H. In machinecode E5 en E1. Pas op ik heb meegemaakt dat iemand dit zo letterlijk nam dat hij de POP H na de RET zette en dan wordt hij natuurlijk niet meer uitgevoerd. En tweede risico is dat U in een routine een mnemonic gebruik die een vlag zet zonder dat U daar gebruik van maakt, er wordt dan nogal snel vergeten ook een PUSH PSW resp POP PSW te doen.
Een verstandige vuistregel voor programmeurs die niet zeker weten wat ze doen of gaan doen is aan het begin van elke routine alles te pushen (op stack zetten) en aan het eind van die routine alles te poppen (van stack halen)

De machinecode wordt nu: E5 21 00 00 22 B1 02 22 B3 02 22 B5 02 22 B7 02 E1 C9
De routine is vrijwel overal te plaatsten, we gaan de mogelijkheden langs:

- We plaatsen de routine in de zeropage (0 t/m 2EB) Dit is een goede oplossing maar we moeten er wel zeker van zijn dat de gebruikte adressen niet door ons programma gebruikt worden. Bestudering van de zeropage levert eigenlijk maar een bruikare plaats op voor kleine machinetaalprogramma's nl de bytes van 1F5 t/m 274 waar de envelopes in worden bewaard. Gebruikt U een kleine of geen ENVELOPE in uw programma is hier een mogelijkheid.

- Voor de heap: Vele machinetaalroutines (FGT oa) hebben deze mogelijkheid benut. Voordat we ons programma plaatsen zetten we de start-heap (29B-29C) op een adres boven ons machineprogramma en geven dan in basic een NEW of CLEAR. We heben dan nooit problemen met basic en mlp gecombineerd.

- In de heap. We definieren een array of string en zetten ons programma dan daarin.Voordeel is dat grotere programma's als array in te lezen zijn en dat geen pointers aangepast behoeven te worden. Nadeel is dat basicprogramma niet meer helemaal vrij is; de array of string die gebruikt wordt mag niet verzet worden dus geen andere er later voor zetten en programma moet altijd beginnen met het plaatsen van het mlp.

- In het basicprogramma: De mogelijkheden hiervoor als we het niet te ingewikkeld willen maken zijn in een REM of PRINT statement.
  Voordeel-programma volledig te editten en te listen, geen pointers aanpassen, en alles in een keer weg te schrijven. Nadeel-programma moeilijk te plaatsen en aan te roepen. Alleen toepasbaar voor kleine mlp's die liefst geen sprongadressen bevatten.

- tussen basicprogramma en symboltable:lastig te plaatsen maar wel veilig. eventueel hier plaatsen voor 'meesaven' maar elders plaatsen voor werken.

- In symboltable: als vorig punt maar gemakkelijker te plaatsen aan het eind Echter pas op met verplaatsen als de plaats opgezocht wordt via de end symboltablepointer (2A3-2A4) omdat die misschien iets verkeerds aanwijst.

- In de vrije RAM dwz tussen end of symboltable en bottomscreen. Pas op dat tijdens run het programma niet gewist wordt door een grotere MODE, of dat U zelf met edit alles verruineert.

- In de ROM: twee mogelijkheden in eprom zoals memocom doet of in de stack. De laatste mogelijkheid is erg gevaarlijk.Je moet goed weten wat je doet in dat geval, DBL werkt bv op deze plaats.
  Extraatje:programma in de stack werkt sneller.

Er zijn nog andere argumenten dan de genoemde om voor of tegen een bepaalde plaats te kiezen, maar ik wil niet al te diep graven.
Ik wil echter nog wel een laatste verandering aanbrengen in het programma. Ik wil voorkomen dat men de routine steeds opnieuw moet aanroepen maar dat dit automatisch geschiedt. Hiervoor kies ik de methode van de interrupt op tijdsbasis Elke 20 ms wordt er een tijdsinterrupt gegeven en die interrupt wordt gegeven op vector 7. Dat wil zeggen elke 20 ms komt er automatisch een sprong naar interruptvector 7. Deze interruptvector zet registers H en L op de stack, laad de registers H en L met de inhoud van de adressen 70 en 71 en zet dan de inhoud van de registers H en L in de programcounter. Dit betekent dat het programma verder gaat met de instructies die gevonden worden op het adres dat staat op adres 70/71 en de oude inhoud van de registers H en L op stack staan.
Een ingewikkelde maar toch wel doordachte methode.

We kunnen nu deze interrupt routine ondervangen door de adressen op 70/71 te wijzigen in het adres waar onze routine staat. We behoeven de routine dan niet meer aan te roepen want dit zal automatisch elke 20 ms (50 maal per seconde) gebeuren. Wel moeten we er voor zorgen dat de normale interrupt 7 afhandeling

kan plaatsvinden, we zullen ons programma dan ook niet mogen eindigen met een RET (C9) maar met een sprong naar de normale afhandeling van interrupt 7. Dat is adres D9A9 dus ipv C9 zetten we nu C3 A9 D9 (weet U nog, omgekeerd ?)

Maar er zijn nog een paar problemen. De interruptbehandeling zal zelf niet door een andere interrupt onderbroken mogen worden dus beginnen we de routine met DI disable interrupts (F3). Om reden van programmeringsfatsoen zetten we de interrupts aan het eind van onze routine weer aan met EI-enable interrupts (FB).

Ons programma wordt nu in machinecode:

F3 E5 21 00 00 22 B1 02 22 B3 02 22 B5 02 22 B7 02 E1 FB C3 A9 D9

Nu alleen er nog voor zorgen dat de interrupt afhandeling naar de routine gaat. Het lijkt simpel; verander het adres op 70 en 71 in het adres waar de routine staat, bv dmv POKE's. Maar als we adres 70 gePOKEd hebben en precies op dat moment komt er een interrupt op vector 7 is het adres wat er dan op 70-71 staat noch het oude noch het nieuwe adres. Er zijn twee oplossingen:

1) Zorg er voor dat de routine staat op adres xxA9 zodat slechts een byte veranderd hoeft te worden.
2) schrijf een extra klein mlp dat 70-71 met het gewenste adres vult na de interrupts uitgezet te hebben.

Een volgende mogelijkheid is niet in een programma maar in direct mode UT en dan V7 dit geeft dan D9A9 en verander dit dan in het gewenste adres. Groot nadelen hiervan zijn - niet in programma en terugzetten op oude waarde lastig want als U bv UT tikt staat er op beeld gelijk UUUUUUUTTTT en dat geeft SYNTAX ERROR. We vonden echter een zeer fraaie oplossing. De interruptroutine staat van 38 t/m 3F.De laatste twee bytes hiervan (3E-3F) zijn ongbruikt. De routine laadt dmv het adres 0070, dit wordt genoemd op 3B-3C. Nauwkeuriger 3B bevat 70 en 3C bevat 00. We zetten nu het adres van onze routine op 3E-3F (omgekeerd!) en kunnen onze rept aanzetten met POKE #3B,#3E en uitzetten met POKE #3B,#70. En omdat de registers H en L toch op stack gezet worden kunnen we de PUSH en POP weglaten. Hieronder een basicprogramma dat een en ander demonstreert.

met dank aan Nico en Just

Frank H. Druijff

```
10    A$="machine language program"
20    P=VARPTR(A$):P1=PEEK(P):P2=PEEK(P+1):P=P1+P2*256
30    FOR I=1 TO 22:READ A:POKE P+I,A:NEXT:P=P+1
40    POKE #3E,P MOD 256:POKE #3F,P SHR 8:POKE #3B,#3E
50    H=GETC:IF H=0 GOTO 50:IF H=9 THEN POKE #3B,#70:END
60    PRINT CHR$(H);:GOTO 50
99    DATA #F3,#E5,#21,0,0,#22,#B1,2,#22,#B3,2,#22,#B5,2,#22,#B7,2,#E1,#FB,#C3,#A9,#D9
```

```
10    A$="mlp":FOR I=1 TO 20:READ A:A$=A$+CHR$(A):NEXT
20    P=VARPTR(A$):P=PEEK(P)+PEEK(P+1)*256+4
30    POKE #3E,P MOD 256:POKE #3F,P SHR 8:POKE #3B,#3E
40    H=GETC:IF H=0 GOTO 40:IF H=9 THEN POKE #3B,#70:END
50    PRINT CHR$(H);:GOTO 40
99    DATA #F3,#21,0,0,#22,#B1,2,#22,#B3,2,#22,#B5,2,#22,#B7,2,#FB,#C3,#A9,#D9
```

DAI RESTART ROUTINES

DAI BASIC and the internal operating system are written in a way that they are hard to extend. Only the Input/Output (Cassette, Keyboard, RS 232C,DCE bus, Memorymanagement and the EmergencyStop) can be defined by the user.

With the above mentioned routines you can define your own I/O routines (like the MEMOCOM MDCR the Tape operating system checks the input from keyboard and if a <RETURN> is given then it checks for DCR commands.)

This method works before the internal BASIC 'takes over' and in a program you have to use a CALLM statement to work with the MDCR. But there are other ways to extend BASIC and te internal operating system. One for instance is the method the DAI uses to switch ROMbanks. In the DAI-ROMS the addresses E000-EFFF exist 4 times. They are referred to as bank 0,1,2,3. Which of the four banks is selected is determined by the two highest (6,7) bits of addres #FD06 and a duplicate is held in addres #40. In BASIC the routine looks like this:
POKE #FD06,(PEEK(#FD06) IAND #3F) IOR (BANKNUMBER SHL 6)
and the same for addres #40. Don't try this because BASIC uses Bank 0 !

Internally the DAI uses the same method to switch banks Whenever a CALL is made to bank 1,2,3 this can't be done in a direct way by a CALL instruction because a CALL doesn't switch banks. For this purpose the special 8080 CALL's the RST X (X=0-7) and a databyte are used. These RST X are one byte CALLinstructions and the databyte is fetched in actual RST routine and then the return addres is incremented by one to return after the databyte. This databyte is an offset from E000 in the selected bank.

To switch banks the following RST X are used;

RST 4	switches to bank 1	Math & Sound package
RST 5	switches to bank 2	Screendriver/Edit
RST 1	switches to bank 3	Encoding/Utility

ALL NUMBERS IN HEXADECIMAL NOTATION !

For instance the routine to print one character on the screen is RST 5,DATA 3 (EF 03) this is a CALL to BANK 2 address #E003. The A register contains the ASCII value of the character. The RST 5 instruction is a CALL to addres 28 (=8* 5 =40D). At 28 the following instuctions prepare for bankswitch;

28	NOP	00
29	PUSH H	E1
2A	LHLD 006C	2A 6C 00
2D	PCHL	E9
2E	NOP	00
2F	NOP	00

At addresses 6C/6D there is FD C6, so a jump will be performed to C6FDH with the original contents of the HL registerpair on top of the stack. Because all of this is in RAM you can change the contents of addresses 6C/6D or 2B. In Utility by the V-command e.g. >V5 C6FD-300 <cursor left>. This forces RST 5 to continue at addres 300 or where your own program is located. A good example is the APPLE/ATARI conversion program which was published in an earlier magazine.

Using this method you can manipulate all the functions in bank 1 to 3 as shown above and even in bank 0.
Very useful in this method is RST 5,DATA 3 this routine is used for everything (except messages during LOAD/LOOK and VER) that has to be printed on the screen.

This routine is used by PRINT,LIST and all DAI messages
thus we can change all these routines by checking or changing;

1) the contents of the 8080 registers A-L
2) the stackpointer
3) the returnaddres
4) the databyte after RST X

To do this we need to know the contents of the registers,
the purpose of the RST X (value of databyte) or the stackpointer
value at the moment the RST X is started or the caller of the
RST X. (People who have no knowledge of the firmware can study
the routines published in earlier magazines or the Firmware
manual of J. Boerrigter.) The most important thing is to check
the databyte because this determines which routine is called,
and the contents of the registers.

As an example we will take RST 5, DATA 3 and assume that our
own routine starts at #300. How to find the databyte ? After
the RST X is executed there will be a returnaddres on stack this
addres is the addres of the databyte because it points directly
after the RST X instruction. Then before the jump is made to
your own routine the HL registerpair is PUSHed on stack (see
example first page).
In our example the A register is checked after we've found
that the databyte is 3. If the character is uppercase it will
be converted to lowercase.
The Basic interpreter won't recognize lowercase commands. To change
this we have to take action. We have to change lowercase commands
into uppercase again. This can be done . The interpreter uses a
routine to get characters from the screen(ram). This routine
is RST 5,DATA 15.
We have to change the returnaddres after this routine to our
own routine and then check if the character just fetched is
lowercase and convert it to uppercase. This way the interpreter
accepts the commands. The total routine will look as follows;

LOWERCASE COMMANDS

```
300          POP H                ;retrieve HL from stack
301          XTHL                 ;exchange HL/top of stack
                                  ;returnaddres is in HL
302          PUSH PSW             ;save PSW
303          MOV A,M              ;get contents of returnaddres
304          CPI        3H        ;is it databyte to print one
                                  ;character on the screen ?
306          JZ         UPlow
309          CPI        15H       ;is it databyte to get one
                                  ;character from screen ?
30B          JZ         lowUP
30E out0     POP PSW              ;restore stack and continu
30F out      XTHL                 ;RST 5 if not the proper routine
310          PUSH H
311          JMP        0C6FDH    ;normal RST 5 addres

320 UPlow    LDA        118H      ;is it during input then
323          CPI        0FFH      ;ignore charactertype
325          JNZ        out0
328          POP PSW              ;Get ASCIIvalue in A again
329          CPI        'A'       ;<'A' then no action
32B          JC         out
32E          CPI        'Z'       ;>'Z' then no action
330          JNC        out
333          ORI        20H       ;convert it to lowercase
335          JMP        out
338 lowUP    POP PSW              ;retrieve PSW
```

```
339             INX H                   ;set the returnaddr after the
                                        ;data byte
33A             XTHL                    ;HL=original HL/top of stack=
                                        ;old returnaddres+1
33B             PUSH H                  ;save HL
33C             LXI H           return  ;returnaddr to our own routine
33F             XTHL                    ;HL=original HL/top of stack=
                                        ;new returnaddres
340             PUSH H                  ;top of stack=orig. HL
341             JMP             0C6FDH  ;continu RST 5 and return to
                                        ;our own routine

344 return      DB              15H     ;databyte after the returnaddres
345             PUSH PSW                ;preserve flags
346             CPI             'a'     ;<'a' then no action
348             JC              out2
34B             CPI             '{'     ;>'z' then no action
34D             JNC             out2
350             SUI             20H     ;convert it to uppercase
352 out2        XTHL                    ;get flags in L
353             MOV H,A                 ;character in H
354             XTHL                    ;restore HL
355             POP PSW                 ;restore PSW
356             RET                     ;go back to the old returnaddres
                END                     ;after the databyte
```

In this routine the conversion is disabled during input of BASIC commands so you can mix upper- and lowercase. All listings even in utility or with the DNA assembler will be in lowercase.

Sometimes it will not be enough to know the databyte or the returnaddres if we want to change a routine because the routine which uses different other routines before a RST X instruction is used. As an example we will take the ERRORroutines in the DAI. If we want to make an ON ERROR GOTO routine there is a point where we can trap this routine before the execution is stopped. This point is just before the message is printed. This message is printed by (again) RST 5,DATA 3. To find out if an ERRORmessage is printed we have to check the stack and the stackpointer. Because everytime a different routine is CALLed it leaves a returnaddres at the top of the stack and the stack pointer is decremented by two. This means we can find the ERROR returnaddres somewhere on te stack. Because we can check the stackpointer by means of the DAD SP instruction and we know how many subroutinelevels are at the stack we always know if the original CALLer is the ERRORroutine. With this knowledge we are going to treat the stack as normal memory. We add the stackpointer to two times the subroutinelevel and get the returnaddres by means of the MOV A,M instruction. If we have found an ERROR than the BASICprogram is continued ,before a message is printed, at a user specified line. If the line specified does not exist errors are enabled and a UNDEFINED LINENUMBER message will be printed. See the example on the next page.

Using the methods described above it is possible to extend the DAI BASIC & operating system. Sometimes very easily sometimes with a little copying and altering ROMroutines. I wrote a program using this method of about 500 bytes which makes it possible to extend BASIC with 65 new BASICcommands fully compiled to standard DAI BASIC format. This just as an illustration of the power of this method. I hope this article will give you an idea how to interact with DAI BASIC and to do more with your DAI.

N.P. Looije

FOR SALE

H I L F E ich möchte meinen Computerplatz ausbauen!

Das Geld liegt bei mir in nicht gebrauchten Geräten fest. Erbitte kollegiale Hilfe, Vielleicht haben Sie im Geschäft oder privat Bedarf für:

Dosierpumpe mit Schrittmotor, max. 100 ml/min, mit digitalem Multiplikator 1, Stuersignal 4-20 mAdc. Neupreis Fr4000. ungebraucht.
Thermodrucker 17 Zeichen & autom. Balken/Strichdiagramm aus BCD oder binär parallel Anschluss.In Gehäuse,220V.Von Ziegler. Neu Fr.1700.-

Lötkolben Butagas 100 W, flammenfrei mit Katalysator Brennkammer, heizt 2 Std. Restbestand aus Grosseinkauf.Pistolenform.Fr 44.-

Geregelte Stromquelle 4-20 ma, für Versuche, im Gehäuse, passt zur Pumpe !

Stab. Stromversorgung 9/15 v, 5A, kurzschlusssicher 90.-
do 5V 1 A do 130.-

Fairchild Kassettenfilmprojektor für Werbung, gebraucht 250.- net

Anlass-Strom Begrenzer 1 Ph 3KW 220 V für Motoren 100.-

Wolf Stichsäge Nr 16 40.-

Lichtregler 400 W Schuko, UFO Modell 40.-

Dachgepäckträger zu Peugeot 204, Montage mit Dachschrauben (Original), 100 Kg zugelassen.Nicht Regenrinne! gebraucht. 50.-

Nordmende Globetrotter 13 KW,UKW,MW,LW,Schiffswelle, mit Autohalterung. OK, nur UKW rev. bedürftig 100.- net

Engl. Leichtfahrad,<10kg, 3Gang, faltbar in Tasche verstaut als Handgepäck transportierbar, dann sehr klein 600.-

Olympus Pearlcorder SD3 mit LCD Zähler/Uhr, UKW -108MHz Sprachsteuerung,Tel-pickup, Etuis, viele Mikrokassetten 850.-

Schalenkoffer 60x45x15 cm innen, für Demogerät 150.-

DAI DCE Designers Handbook 60.-

Folgendes nur für Abnehmer in der Schweiz

Hängemappenkoffer für Mobilmappen 9o.-
Hängemappen Vetro Lateral v. Fürrer Patent 2.-
Ablegeschachteln, billigster Fachschriftenordner 1.-

Zu den Preisen: Wo nicht anders vermerkt (gebraucht,net) sind Neupreise genannt, und die Sachen sind neu oder nur wenige Std. in Betrieb. Ich erwarte deshalb ein angemessenes Angebot, andererseits hoffe ich, Insertionskosten zu sparen, was dem jetzigen Anbieter zum Vorteil gereicht.

21.3.83
E. Zahner, CH 8910 Affoltern, Tel 01-7617872