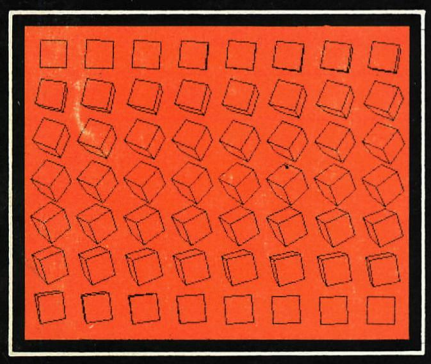
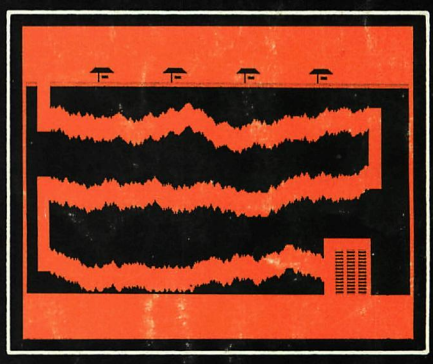
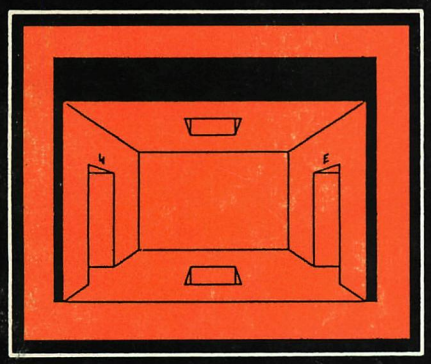
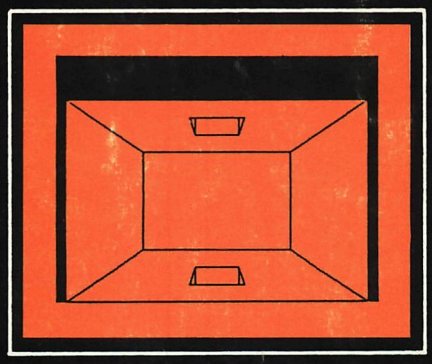
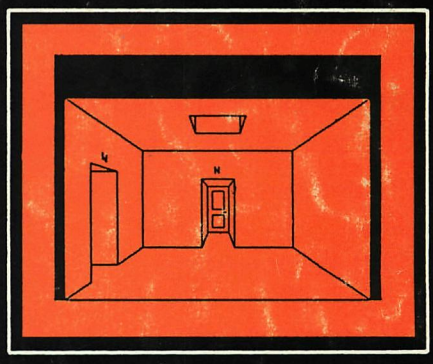
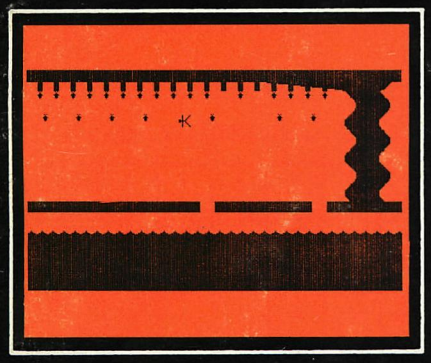
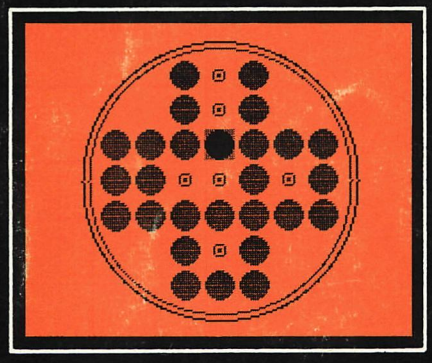
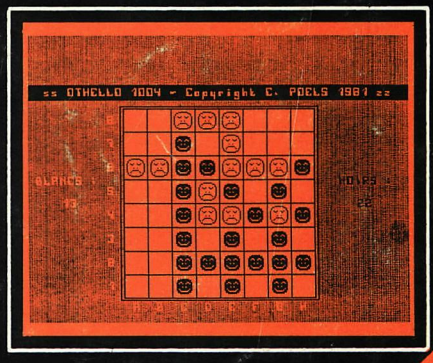
DAI NAMIC

15

tweemaandelijks tijdschrift maart - april 1983



personal computer users club

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

International

COLOFON

DAInamic verschijnt tweemaandelijks.
Abonnementsprijs is inbegrepen in de jaarlijkse contributie .
Bij toetreding worden de verschenen nummers van de jaargang toegezonden.
DAInamic redactie :
Dirk Bonné
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
0	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

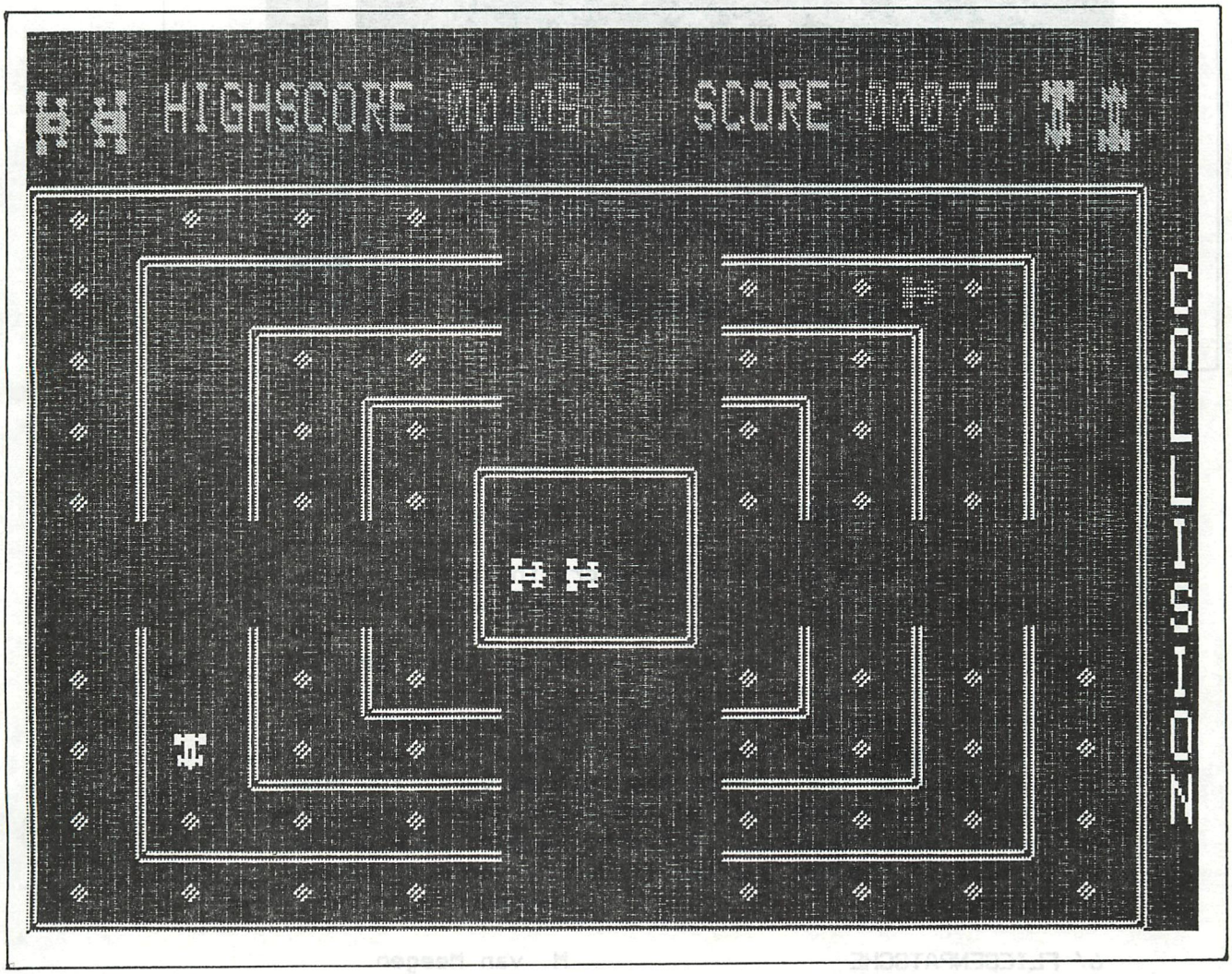
CATALOG

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

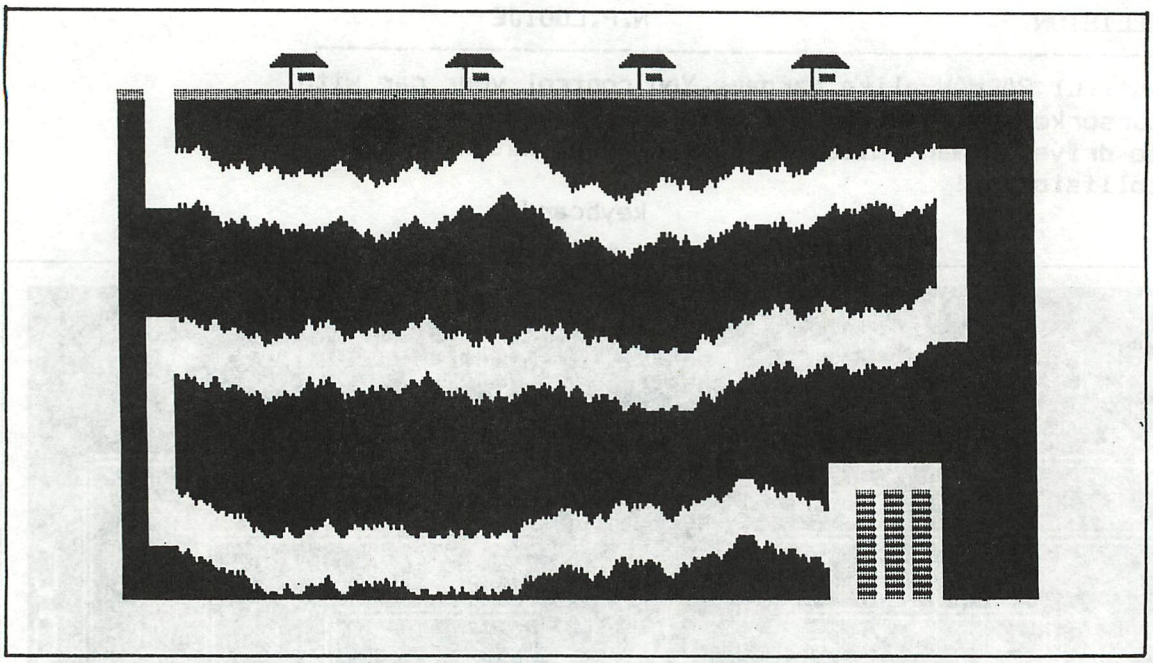
Up to 8 players can participate this realistic minigolf-game. The game contains 14 different fields!

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

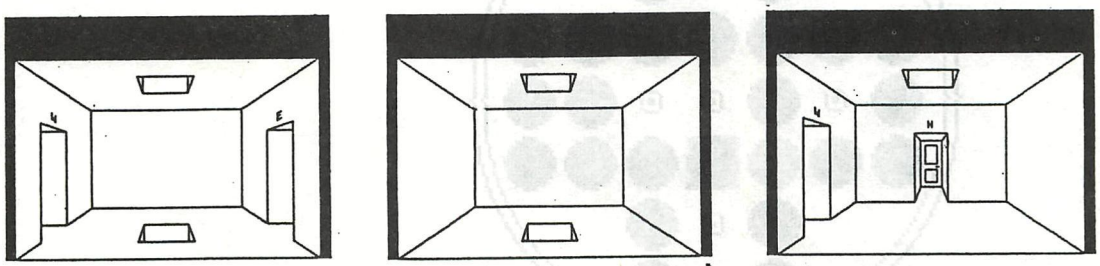
2-player game : try to stop the moving object of your opponent. keyboard

GAMES COLLECTION 11

1/ QUINTIMAZE

F. van Amerongen

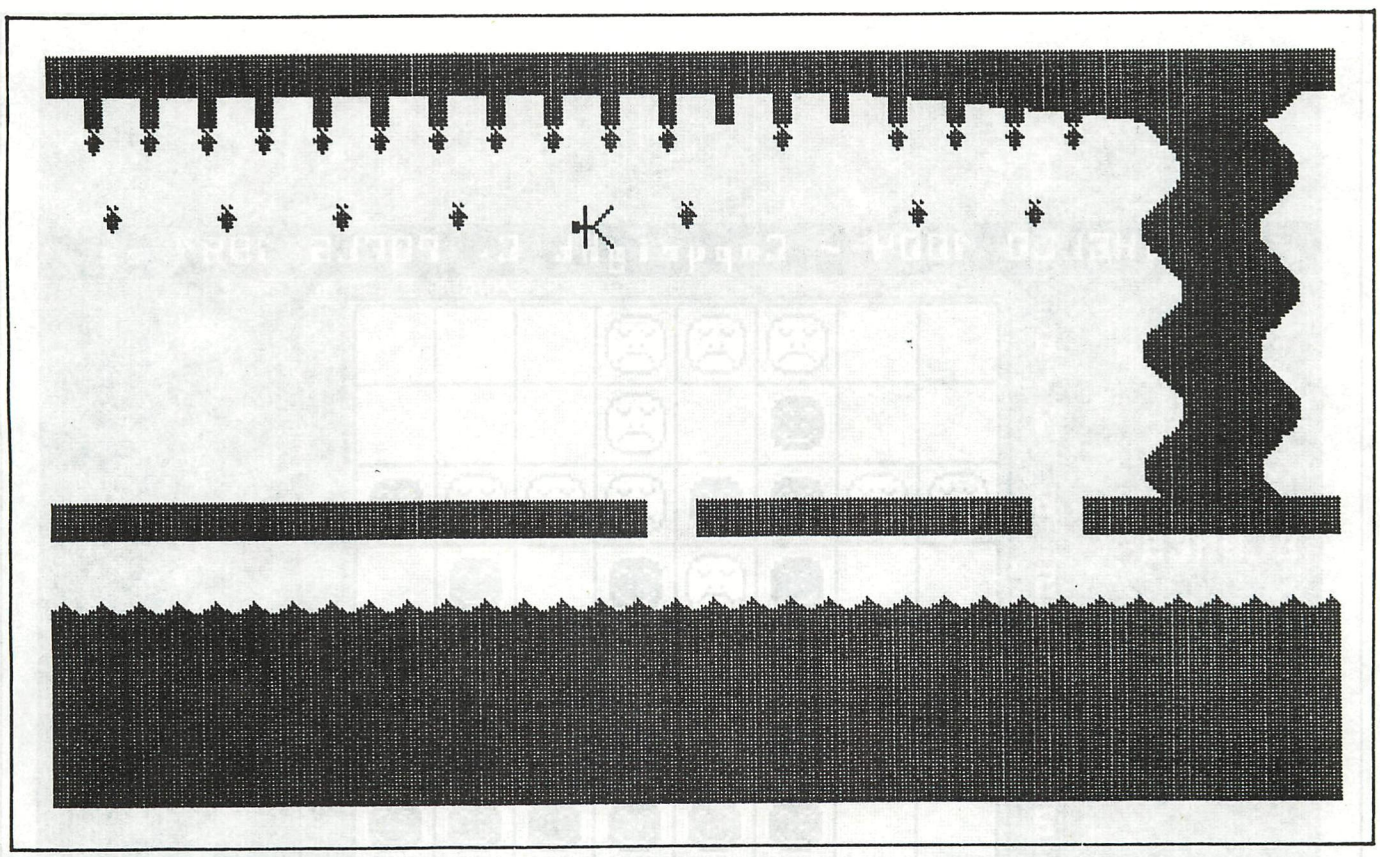
Try to find your way out of this 5x5x5 cubic maze.Very fast representation of the maze-rooms! keyboard



2/ MANGROVE

M.Hooykaas

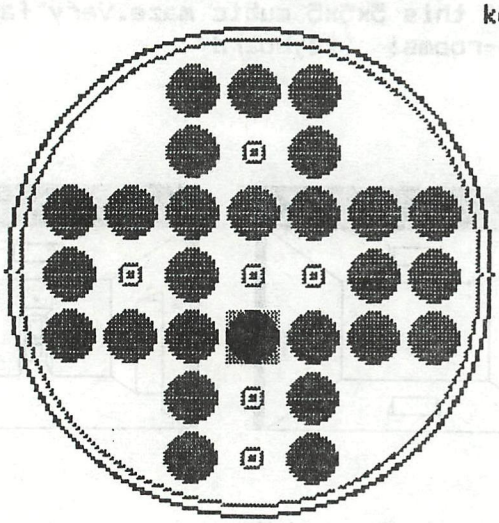
3 castaways, living on an island, their only food being the mangrove-fruits. Look out for the crocodiles while jumping for fruit!! keyboard



3/ SOLITAIRE F.Druijff

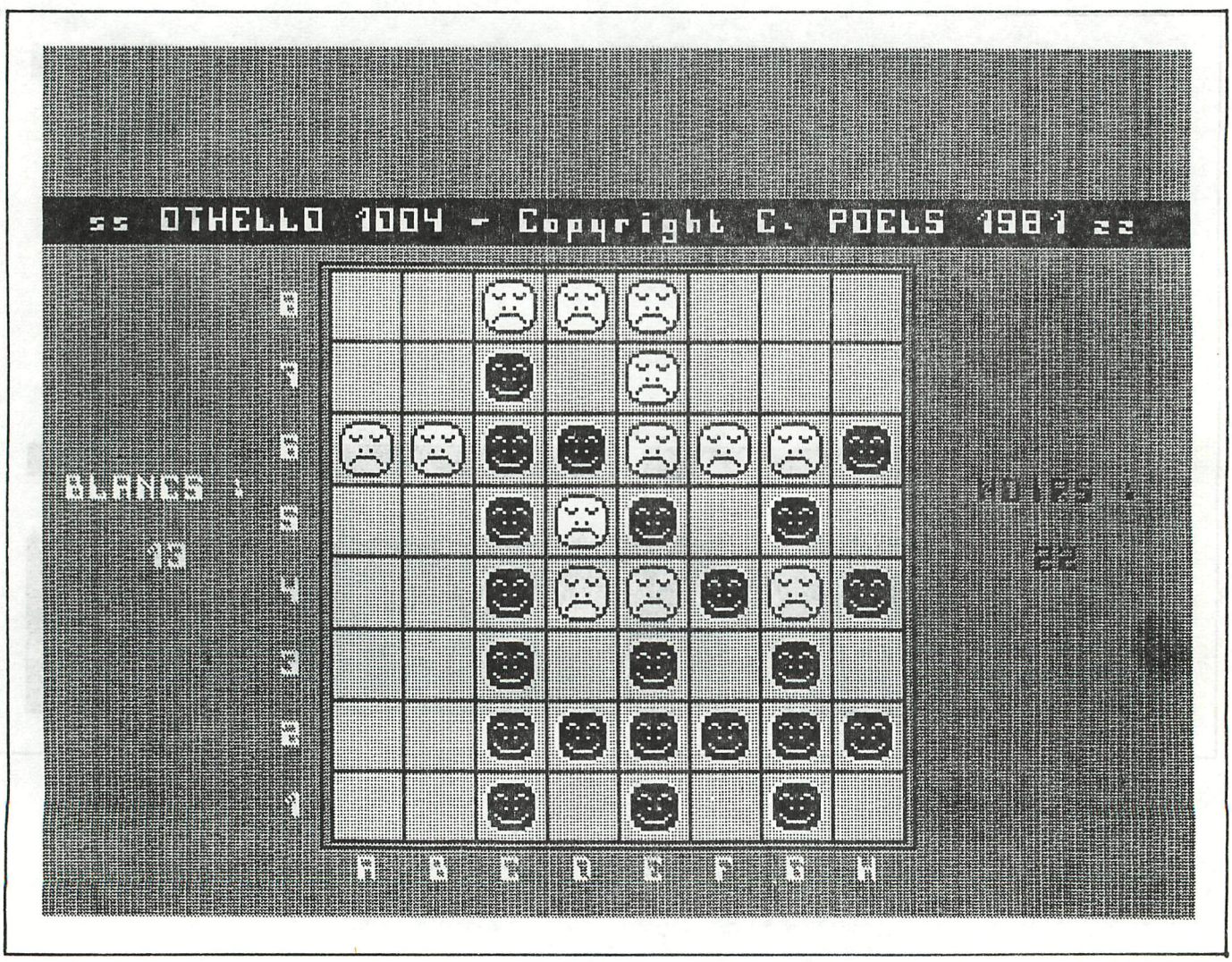
The object of this game is to loose all pegs except one.
You loose a peg by moving another peg from one side of
the peg to the other (empty) side.

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...

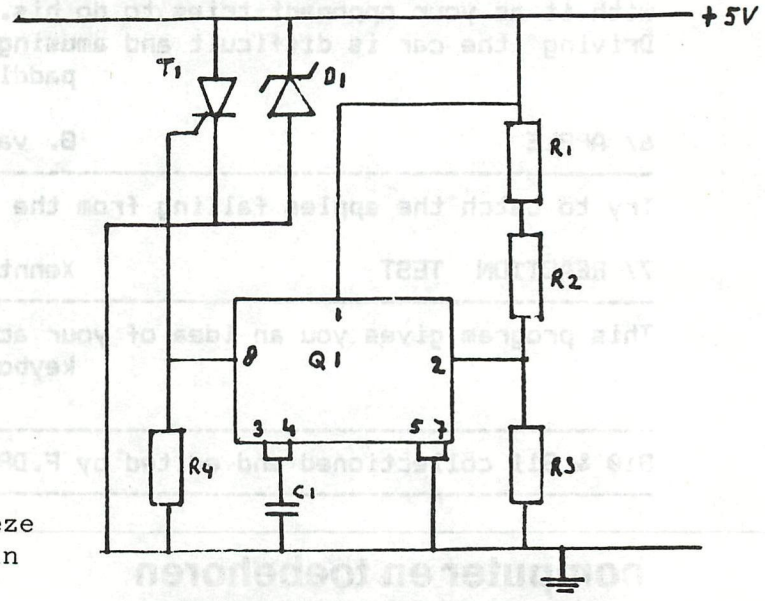
7/ REACTION TEST — Xennt

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 01.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 75.00 DM

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

Epromprogrammier programmiert 2708,2716,2732 2516 und 2532,arbeitet ohne zusätzliche Spannung,bei KEN-DOS durch Benutzung eines Epromplatzes, normal Epromhalter erforderlich, komfort. Software,(Prüfen,Check,Vergleichen),m.Bedienungsanl.
DSP-EPP 170.00 DM

RWC-Netzteilkarte, +5V/3A, +12V/2A,-5V/1A, 50 Hz und 100 Hz Taktausgang,Karte gelötet und getestet, erf.Trafo 8V/3A, 8V/1A, 16V/2A
DSP-NT1 85.00 DM

RWC-Adapteranschlußkabel zum Anschluß von RWC1 an den DAI, wird nicht benötigt wenn am DCR-Kabel eine Buchse frei
DSP- RWCKAB 30.00 DM

Epromhalter extra
DSP-EPH1 35.00 DM
In Vorbereitung:
Epromhalter für DCR und Epromprogrammier, ermögl.Betrieb vom DSP-EPP m.DCR,CAS u. DAI-Floppy

Eprom 2716	DSP-2716	6.90 DM
Eprom 2732	DSP-2732	14.00 DM
Eprom 2732A	DSP-2732A	15.20 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/ 1/1024 Sek.,2 Treiber f.Schaltausgang,Kartenadresse wählbar, umfangreiche Dokumentation
DSP-RTC 185.00 DM

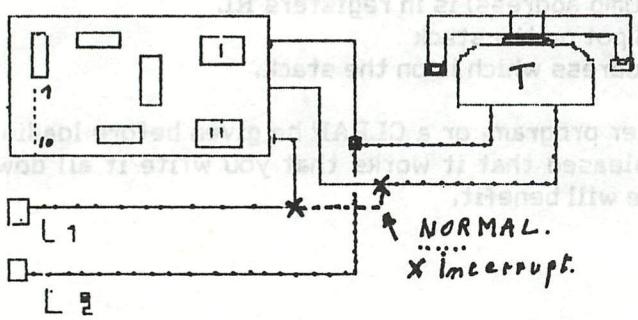
D 8255 AC-5	DSP-8255	8.70 DM
AMD 9511	DSP-AMD	330.00 DM

RGB-Monitor,15 MHz Auflösung, weißes Geh. DSP-CMB2 1290.00 DM

Diskette, 5.25", 1.Wahl, solange Vorrat reicht DSP-DIS1 6.00 DM

Tastatur DSP-KBS 197.00 DM

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

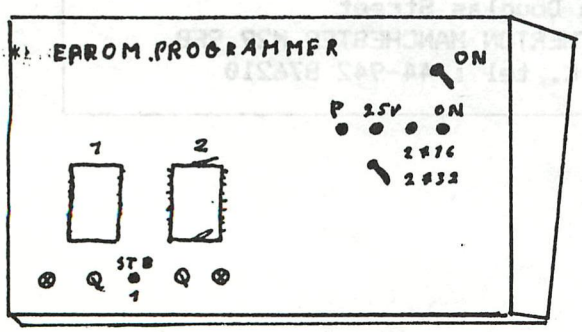


CIRCUIT TELEPHONIQUE

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,

>>>>>>>>PROGRAMMEER-TECHNIEKEN<<<<<<<<

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.

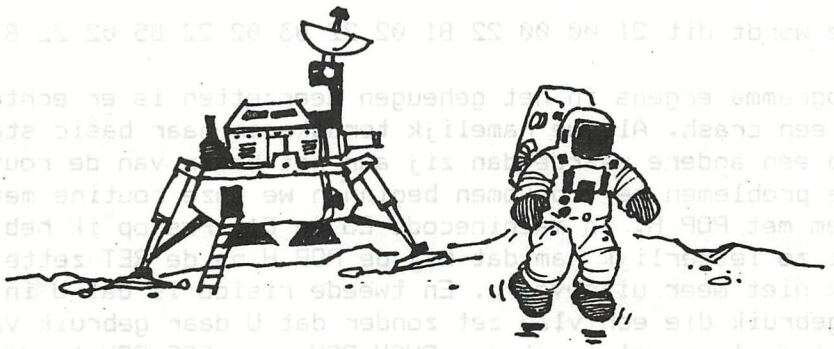
Als U dus wil gaan programmeren in machinecode/assembler is het sterk aan te raden een assembleerprogramma aan te schaffen. (DNA of SPL)

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

BASIC V1.0 - V1.1

(from DAInamic 11, page 209)

For some time now there have been two versions of DAI-BASIC, V1.0 and the newer V1.1. As far as possible DAI have corrected the bugs, at least the ones they knew about, in the V1.0 ROMs. By dint of experiment with the two ROM sets I have tried to locate the differences - some of this information I obtained directly from DAI. Here are my provisional findings.

1 V1.1 reacts differently on a RUN line-number. In V1.0 all the variables would be cleared but that does not happen with the V1.1 and is a major benefit of the new ROM; now debugging BASIC programs is simpler. For example, while running a program it is now possible to break, to list and even to change directly the value of variables before letting the rest of the program continue.

2 DIM (255,255) is permitted with V1.1

3 Sound channel 2 will be switched off by a hard reset; that was never guaranteed with V1.0 so there was always a possibility that the sound would return when switching on again

4 SAVEA and LOADA now work correctly (see previous DAInamic newsletters)

5 The bug has been removed from the SGN function

6 A minor bug has also been cleared from the EDIT mode. This fault did not have any serious consequences and was reported by only two users. (under some circumstances the cursor disappeared for a while)

7 The basic operator '-' is correctly recognised in V1.1 so that it is no longer necessary to put brackets around the negative expression.

8 There was a bug in the TALK command but I do not know what it was.

9 GETC is now debounced. The keyboard buffer is cleared first.

10 Returning from the EDIT mode it is now unnecessary to key immediately RUN

11 The TAB function now works correctly, ie, independently of the serial channel (printer) if that was not switched on.

12 There was also an incorrect error message given sometimes, such as ERROR IN LINE .... without a line number.

The foregoing differences between the two ROMS are those I know about. If anyone wants a list of those differences in hexadecimal format he can obtain it from Jos Schepens for the cost of copying and postage. Hopefully this short review will enable you to judge if the new ROMs are good value. I do not know what they will cost but I have an idea that the price will be fairly high.

DAI pc's communicate via public telephone lines.

This article will give you a detailed description how to communicate (TALK), directly via your screen, exchange electronic mail, program-listings or object files, from one DAI to another, by using a low cost (acoustical) modem.
To achieve this, the following steps have to be taken:

a) Load the machine language program, named DAICOM (DAI's COMmunicate) and start it with UT-Z3-6400.
   Not CALLM #400 !!
b) Connect your modem to the RS232 at the rear of the DAI.
c) Make your telephone-call with the other DAI-user,who should also have completed steps a and b.

Assuming you can hear each other loud and clear, as usually on a Dutch line, first make the decision who works in Half/Full-duplex (always in the opposit mode), before switching over to the modem.

The advantage of working in full-duplex is that you can see if something goes wrong during data transmission, because the typed character is first sent to the receiver, who echo's it, whereafter it comes on your own screen.
After this choise is made press both <T> and TALK via the key-board.
Typing cursor-up displays the MENU again and one of the other features can be selected.

The choise of the modem.

The modem (MOdulate DEmodulate) should comply the CITT V21 standard i.a.w.:

- Full duplex operation.
- 300 baud.
- 2 switch-able channels (each DAI works on a different channel).

The best choise is a modem which is directly connected to the telephone-line, but these types are generally expensive and must have (in Holland) PTT approval.
A good alternative is a cheaper acoustic modem.
We personally have good experience with a modem (kit), offered by the firm NENIJWA in EDE (Holland) telephone number 08380-10856.
With just a little understanding of basic electronics and a good soldering iron (small tip), you should be able to assemble the kit in a couple of hours.
A minor disadvantage of such a modem is that it easily picks-up background noise, but a piece of foam could help reduce this.

What are the possibilities of working with DAICOM

The most interesting feature is that you can either transmit and receive both ASCII and HEX (object) files.

Transmit files.

For ASCII, all that is (or moved) in the editbuffer can be transmitted.
You can for example type a message in the edit-mode <E> and afterwards send it away.

Sending a BASIC program

After DAICOM is loaded and started go to <B> BASIC and simply LOAD a program from tape or DCR (heap pointers are adjusted automatically).
Type : -CLEAR XXXX
-EDIT
-BREAK
-BREAK
The program is in the editbuffer and ready for transmitting.

Send DNA source listing

DNA source files (bufferdim. 16), can directly be sent by typing <S> in the menu.

Send HEX files

To transmit a hex-file first the start and end address of the data-block is required, before the transmission starts.

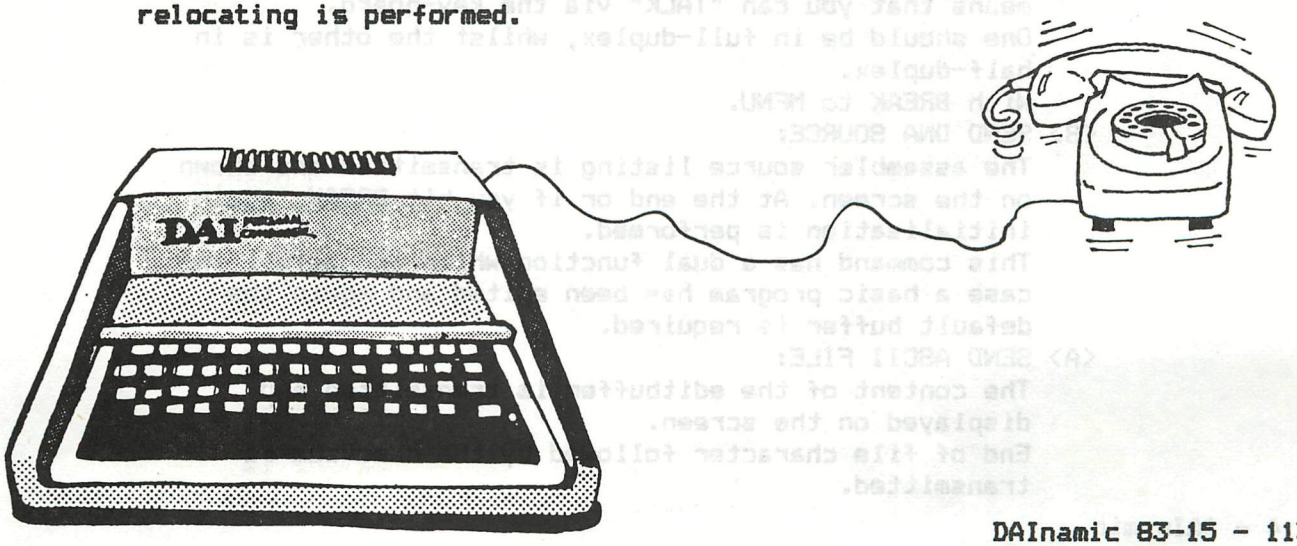
It is important to note, that you can always see on the screen what is coming in or going out.

Receive files

All incoming data is stored in the (default #3000-#6FFD) buffer.If more space is required ,go to basic and give CLEAR XXXX, EDIT, BREAK, BREAK.

Received basic programmes are ready for LIST or RUN after POKE #135,2 is performed.

If an object file is received, the start and end address of the received file are given and the relocate (start) address is asked.
After converting the ASCII characters in HEX bytes, relocating is performed.



DNA source files can be saved on tape or DCR by first finding the end adddress of the file.
Edit buffer pointer A4-A5 gives the end address (LLHH).
The start address is always #3000.
Go to UT and Display >DA2 A4

Example:the address 3277 is displayed as 77 32

Now typing W3000 HHLL and save the file on tape or DCR.
If DNA is loaded type R. and the file will be read as a normal source file.

How to work via the menu.

First load the program.

1) UT
2) Z3
3) R
4 G400

Now the menu is displayed and the following options can be selected.

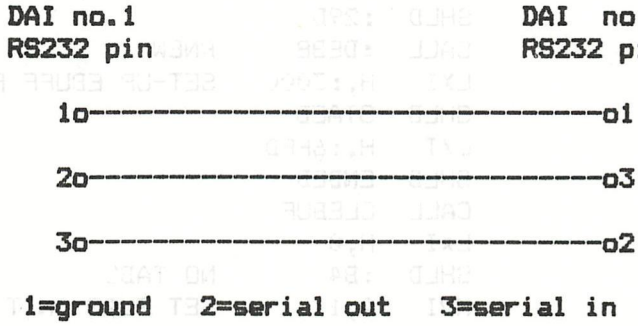
<E> ENTER EDIT MODE:
    This facility can be used to prepare a letter (mini-word processor).
    Also received ASCII files will come automatically in this mode and can easily be edited.
    Escape with BREAK
<H> HALF-DUPLEX (DCE data computer):
    The receiver of files (ASCII or HEX) comes automatically in this mode.
    The program accepts the received character stores it in the (edit)buffer, echo's it back to the transmitter and displays it on the screen.
    During "TALKING" via the key-board, the typed characters are both send to the screen as well to the RS232 (2).
<F> FULL-DUPLEX (DTE data terminal):
    Full-duplex is automaticaly selected during transmitting files.
    Characters are first sent to the receiver (other DAI in receive mode), who echo's the character before you get it back on the screen.
<T> TALK
    In this mode, both can see the typed characters, this means that you can "TALK" via the key-board.
    One should be in full-duplex, whilst the other is in half-duplex.
    With BREAK to MENU.
<S> SEND DNA SOURCE:
    The assembler source listing is transmitted and shown on the screen. At the end or if you hit BREAK, again initialisation is performed.
    This command has a dual function which is usefull in case a basic program has been edited and again the default buffer is required.
<A> SEND ASCII FILE:
    The content of the editbuffer is transmitted and displayed on the screen.
    End of file character followed by the checksum is transmitted.

<#> SEND HEX FILE:
The program asks the (hex) start and end address of the file and start transmitting immediately after the end address is given.
<R> RECEIVE ASCII FILE:
Received characters are displayed on the screen and stored in the editbuffer.The end of text character (3) followed by the checksum are not displayed.
If the checksums are different,in the cursor appears an "F" and the program stays in Receive mode.
Escape with BREAK, and enter the Edit-mode.
<H> RECEIVE HEX FILES:
Similar to <R>, but displays the start and end address of the rceived file and ask the new start address of the file (relocates the file).
<U> BACK TO UTILITY:
<B> BACK TO BASIC:

Some practical notes.

You have now read the whole article and typed or loaded the programme in the DAI
At this stage you can start playing with it even without modem.
Just take a piece of wire, strip it and connect 2 and 3 of the RS232 (see manual).Be not afraid, all in/outputs on this connector are buffered.
All the "send" commands can be exercised (full-duplex).

Another possibility is if there are 2 DAI's.
In this case,couple them by making the following connections.



The same rules apply as if working with a modem.

Finally I would like to thank Noud Rynaerts and Leo v.d.Laak for their valuable comments and suggestions and the extensive "field tests".

If you have problems or questions call me on extension 04780-84180 after 18.00 hrs.

Success,

Ger Gruiters
Laurahof 12
5801 JE Venray

DAICOM will be available on TOOLKIT 5, to be released soon.

```
PAGE 01

001                     *DAICOM V3.3 BY G.GRUITERS 16-1-1983
002                     *
003                     BDRATE  EQU   :84       300 BAUD 1 STOP-BIT
004                     STSREG  EQU   :FFF3     STS-REG IN TICC
005                     INCHAR  EQU   :FFF0     SER INP BUFF
006                     STAEB   EQU   :A2       ADDR START EDIT-BUFFER
007                     EBPTR   EQU   :A4       ADDR END OF TEXT IN EBUFF
008                     ENDEB   EQU   :A6       ADDR END EBUFF
009                     *
010                             ORG   :400      ENTRY PROGRAM
011 0400 CD2304                 CALL  START     INITIALISE
012 0403 C3E604                 JMP   SPECL     DISPLAY MENU
013                     *
014                     *       TALK
015                     *
016 0406 CD6B04         MAINLP  CALL  CHKSTS    CHECK IF CHAR RECEIVED
017 0409 C47104                 CNZ   INTR1     IF YES,HANDLE/DISPLAY
018 040C CDBED6                 CALL  :D6BE     KEY PRESSED?
019 040F DAE604                 JC    SPECL     IF BREAK MENU
020 0412 C4AD04                 CNZ   TRANS     YES,HANDLE/TRANSMIT
021 0415 C30604                 JMP   MAINLP    CONTINUE LOOP
022                     *
023 0418 3E40           TOUT    MVI   A,64      BAUDRATE 9600
024 041A 3205FF                 STA   :FF05
025 041D 3100F9                 LXI   SP,:F900  RESTORE STACKPNTR
026 0420 C309E0                 JMP   :E009     BACK TO UTILITY
027 0423 3A2C08         START   LDA   FLAG1     CHECK IF THIS IS
028 0426 FE01                   CPI   1         THE 1ST PASS,IF NOT
029 0428 CA5404                 JZ    ENDWST    SKIP INIT.
030 042B 219A0A                 LXI   H,ENDPRG  SET START HEAP AFTER
031 042E 229B02                 SHLD  :29B      END PROGRAM
032 0431 210001                 LXI   H,:100    ADJUST HEAP SIZE
033 0434 229D02                 SHLD  :29D
034 0437 CDB8DE                 CALL  :DEB8     RNEW
035 043A 210030                 LXI   H,:3000   SET-UP EBUFF PNTR'S
036 043D 22A200                 SHLD  STAEB
037 0440 21FD6F                 LXI   H,:6FFD
038 0443 22A600                 SHLD  ENDEB
039 0446 CD5F04                 CALL  CLEBUF
040 0449 210000                 LXI   H,0
041 044C 22B400                 SHLD  :B4       NO TABS
042 044F 3E01                   MVI   A,1       SET FLAG INIT DONE
043 0451 322C08                 STA   FLAG1
044 0454 3E84           ENDWST  MVI   A,BDRATE
045 0456 32F5FF                 STA   :FFF5
046 0459 3E01                   MVI   A,1       TO SCREEN ONLY
047 045B 323101                 STA   :131
048 045E C9                     RET
049 045F 2AA200         CLEBUF  LHLD  STAEB     CLEAR EBUFF
050 0462 EB                     XCHG
051 0463 21FF6F                 LXI   H,:6FFF
052 0466 AF                     ZAR
053 0467 CD7CDE                 CALL  :DE7C


PAGE 02

054 046A C9                     RET
055 046B 3AF3FF         CHKSTS  LDA   STSREG
056 046E E608                   ANI   8         CHAR RECEIVED ?
```

```
057 0470 C9                 RET
058                 *
059                 * HANDLE RECEIVED CHARACTERS
060                 *
061 0471 3AF0FF     INTR1   LDA   INCHAR    GET RECEIVED CHAR
062 0474 FE7F               CPI   127       NOT VALID IF MSB=1
063 0476 F0                 RP
064 0477 FE08       BACKSP  CPI   8         BACKSPACE
065 0479 CA8404             JZ    SCREEN
066 047C FE0D               CPI   13        CAR-RET
067 047E CA8404             JZ    SCREEN
068 0481 FE20               CPI   32        FILTER ALL CHAR'S BELOW 32
069 0483 D8                 RC              EXCEPT 8 AND 13
070 0484 CD95D6     SCREEN  CALL  :D695     DISPLAY CHAR
071 0487 CD8F04             CALL  ECHO
072 048A AF                 ZAR
073 048B 322F08             STA   FLAG4     CLEAR TALK FLAG
074 048E C9                 RET
075 048F 57         ECHO    MOV   D,A
076 0490 3A2E08             LDA   FLAG3
077 0493 FE01               CPI   1
078 0495 C0                 RNZ             ECHO ONLY IN HALF-DPL
079 0496 3A2F08             LDA   FLAG4
080 0499 FE01               CPI   1
081 049B C8                 RZ              NO ECHO IF TALK FLAG IS SET
082 049C CDA304     WAIT2   CALL  WAIT      READY FOR TRANSMIT?
083 049F 32F6FF             STA   :FFF6     SEND CHAR VIA RS232
084 04A2 C9                 RET
085 04A3 3AF3FF     WAIT    LDA   STSREG    MASK FOR "TRANSMIT
086 04A6 E610               ANI   16        BUFF EMPTY, IF NOT
087 04A8 CAA304             JZ    WAIT      TRY AGAIN
088 04AB 7A                 MOV   A,D
089 04AC C9                 RET
090                 *
091                 * HANDLE CHARACTERS TO TRANSMIT
092                 *
093 04AD F5         TRANS   PUSH  PSW
094 04AE 3E01               MVI   A,1
095 04B0 322F08             STA   FLAG4     SET TALK FLAG
096 04B3 F1                 POP   PSW
097 04B4 57         TRANSC  MOV   D,A
098 04B5 CDA304             CALL  WAIT
099 04B8 FE10               CPI   16        IF CURSOR-UP
100 04BA CAE604             JZ    SPECL     DISPLAY MENU
101 04BD FE08               CPI   8         (SEE LABEL BACKSPACE)
102 04BF CACD04             JZ    OUT
103 04C2 FE0D               CPI   13
104 04C4 CACD04             JZ    OUT
105 04C7 FE7F               CPI   127
106 04C9 F0                 RP
```

PAGE 03

```
107 04CA FE20               CPI   32
108 04CC D8                 RC
109 04CD 7A         OUT     MOV   A,D
110 04CE 32F6FF             STA   :FFF6     SEND VIA RS232
111 04D1 3A2E08             LDA   FLAG3
112 04D4 FE01               CPI   1         HALF-DPLX?
113 04D6 CAE204             JZ    HDUPLX
114 04D9 3A3108     SUMCHA  LDA   CHKSUM    LOAD OLD CHECKSUM
115 04DC AA                 XRA   D         UPDATE
116 04DD 07                 RLC
117 04DE 323108             STA   CHKSUM    STORE NEW CHECKSUM
118 04E1 C9                 RET
```

```
119 04E2 7A          HDUPLX  MOV   A,D
120 04E3 C37704              JMP   BACKSP
121                  *
122                  *       MENU
123                  *
124 04E6 21E0DD      SPECL   LXI   H,:DDE0    RESTORE INP SUBR
125 04E9 22D200              SHLD  :D2
126 04EC AF                  ZAR             ZERO FLAGS
127 04ED 322F08              STA   FLAG4
128 04F0 323008              STA   FLAG5
129 04F3 3E5F                MVI   A,:5F     NORMAL CURSOR
130 04F5 327500              STA   :75
131 04F8 216308              LXI   H,MSG2    DISPLAY MENU
132 04FB CDD4DA              CALL  :DAD4
133 04FE CDBED6      CHOISE  CALL  :D6BE     WAIT FOR CHOISE
134 0501 FE45                CPI   69        E
135 0503 CA2107              JZ    INITEB
136 0506 FE48                CPI   72        H
137 0508 CA4C05              JZ    HALFD
138 050B FE46                CPI   70        F
139 050D CA5405              JZ    FULLD
140 0510 FE54                CPI   84        T
141 0512 CA3B05              JZ    TALK
142 0515 FE41                CPI   65        A
143 0517 CA6305              JZ    SASCI
144 051A FE23                CPI   35        #
145 051C CA5B05              JZ    SHEX
146 051F FE53                CPI   83        S
147 0521 CAE005              JZ    SSTEB
148 0524 FE52                CPI   82        R
149 0526 CA6D05              JZ    RECASC
150 0529 FE58                CPI   88        X
151 052B CA7705              JZ    RECHEX
152 052E FE55                CPI   85        U
153 0530 CA1804              JZ    TOUT
154 0533 FE42                CPI   66        B
155 0535 CA4405              JZ    TOBAS
156 0538 C3FE04              JMP   CHOISE
157 053B 213C08      TALK    LXI   H,MSG1
158 053E CDD4DA              CALL  :DAD4
159 0541 C30604              JMP   MAINLP
```

PAGE 04

```
160 0544 3E40        TOBAS   MVI   A,64
161 0546 32F5FF              STA   :FFF5
162 0549 C3A0C7              JMP   :C7A0     BACK TO BASIC
163 054C 3E01        HALFD   MVI   A,1
164 054E 322E08              STA   FLAG3     SET HALF-DUPLX
165 0551 C3E604              JMP   SPECL
166 0554 AF          FULLD   ZAR
167 0555 322E08              STA   FLAG3     SET FULL-DUPLX
168 0558 C3E604              JMP   SPECL
169 055B 3E01        SHEX    MVI   A,1
170 055D 322D08              STA   FLAG2     SET HEX
171 0560 C35107              JMP   EDIT2     SEND HEX FILE
172 0563 AF          SASCI   ZAR
173 0564 322D08              STA   FLAG2     SET ASCII
174 0567 CD8205              CALL  SMSG
175 056A C3AF05              JMP   EDIT      SEND ASCII FILE
176 056D AF          RECASC  ZAR
177 056E 322D08              STA   FLAG2
178 0571 CD9805              CALL  RMSG
179 0574 C30106              JMP   READ      RECEIVE ASCII FILE
180 0577 3E01        RECHEX  MVI   A,1
```

```
181 0579 322D08          STA   FLAG2
182 057C CD9805          CALL  RMSG
183 057F C30106          JMP   READ       RECEIVE HEX FILE
184 0582 21CB09    SMSG  LXI   H,MSG3
185 0585 CDD4DA          CALL  :DAD4      MESSAGE"SEND"
186 0588 CDAA05          CALL  CLCHKS
187 058B AF              ZAR
188 058C 322E08          STA   FLAG3
189 058F 3AF0FF          LDA   INCHAR     CLEAR INP BUFF
190 0592 1682            MVI   D,:82
191 0594 CD9C04          CALL  WAIT2      SEND START CHAR
192 0597 C9              RET
193 0598 21EB09    RMSG  LXI   H,MSG4
194 059B CDD4DA          CALL  :DAD4      MESSAGE"RECEIVE"
195 059E CD5F04          CALL  CLEBUF
196 05A1 CDAA05          CALL  CLCHKS
197 05A4 3E01            MVI   A,1
198 05A6 322E08          STA   FLAG3
199 05A9 C9              RET
200 05AA AF        CLCHKS ZAR              ZERO CHECKSUM ADDR.
201 05AB 323108          STA   CHKSUM
202 05AE C9              RET
203                *
204                *  SEND EDIT-BUFFER
205                *
206 05AF 3A2D08    EDIT  LDA   FLAG2       IF HEX FLAG SET,JUMP
207 05B2 FE01            CPI   1           TO HEX SEND ROUTINE
208 05B4 CA5107          JZ    EDIT2
209 05B7 2AA200    SDNAS LHLD  STAEB       GET 1ST CHAR FROM EBUFF
210 05BA 22A400    NEXTCH SHLD EBPTR
211 05BD 7E              MOV   A,M
212 05BE FE00            CPI   0           END OF DATA?

PAGE 05

213 05C0 CAD205          JZ    CHKFL1
214 05C3 E67F            ANI   127         STRIP MSB (DNA SOURCE)
215 05C5 CD0E08          CALL  TRCDEL      TRANSMIT CHAR
216 05C8 23              INX   H           INCR. EBUFF PNTR
217 05C9 CDBED6          CALL  :D6BE       BREAK PRESSED?
218 05CC DAD205          JC    CHKFL1
219 05CF C3BA05          JMP   NEXTCH      NEXT CHAR
220 05D2 3A2C08    CHKFL1 LDA  FLAG1       IF FLAG IS SET THEN
221 05D5 FE00            CPI   0           AGAIN INITIALISATION
222 05D7 C24207          JNZ   STAY1       SEND END OF TEXT & CHECKSUM
223 05DA CD2304          CALL  START
224 05DD C34207          JMP   STAY1
225                *
226                *  SEND DNA (16)SOURCE VIA (EDIT)BUFF.
227                *
228 05E0 3A5912    SSTEB LDA   :1259
229 05E3 FE00            CPI   0           DNA SOURCE AVAILABLE?
230 05E5 CAEB05          JZ    SSOURC
231 05E8 C3D205          JMP   CHKFL1      IF NOT,TO MENU
232 05EB CD8205    SSOURC CALL SMSG
233 05EE AF              ZAR
234 05EF 322C08          STA   FLAG1       NEW INIT. REQUIRED
235 05F2 110070          LXI   D,:7000     DNA SOURCE START
236 05F5 2A5512          LHLD  :1255       PNTS TO DNA SYMB-TABLE
237 05F8 010030          LXI   B,:3000
238 05FB CD4FDE          CALL  :DE4F       MOVE SOURCE
239 05FE C3B705          JMP   SDNAS
240                *
241                *  TRANSFER TO (EDIT) BUFFER
242                *
```

```
243 0601 0E00       READ    MVI   C,0       ZERO CNTR
244 0603 2AA200             LHLD  STAEB     START BUFFER
245 0606 22A400             SHLD  EBPTR     POINTS TO START
246 0609 CD4E06             CALL  LOOK3
247 060C DA7C06             JC    STOPTR
248 060F FE82               CPI   :82       START TXT?
249 0611 C20106             JNZ   READ      ELSE WAIT
250 0614 22A400     LOOK    SHLD  EBPTR     UPDATE PNTR
251 0617 CD4E06     LOOK2   CALL  LOOK3
252 061A DA7C06             JC    STOPTR
253 061D FE7F               CPI   127       NOT VALID IF MSB=1
254 061F F21706             JP    LOOK2
255 0622 FE03               CPI   3         END OF TEXT CHAR RECEIVED?
256 0624 CA5D06             JZ    TEST1     YES, GO TO TEST CHECKSUM
257 0627 FE0D               CPI   13
258 0629 CA3106             JZ    DISPL
259 062C FE20               CPI   32        FILTER <32, NOT CAR-RET
260 062E DA1706             JC    LOOK2
261 0631 77         DISPL   MOV   M,A
262 0632 CD60DD             CALL  :DD60     DISPL ON SCRN
263 0635 56                 MOV   D,M
264 0636 CDD904             CALL  SUMCHA    UPDATE CHECKSUM
265 0639 2AA600             LHLD  ENDEB     END OF BUFF REACHED?


PAGE 06

266 063C EB                 XCHG
267 063D 2AA400             LHLD  EBPTR
268 0640 CD14DE             CALL  :DE14
269 0643 CA7C06             JZ    STOPTR
270 0646 7E                 MOV   A,M
271 0647 23                 INX   H         INCR PNTR
272 0648 CD8F04             CALL  ECHO      RECEIVER (DCE) ECHO'S
273 064B C31406             JMP   LOOK      NEXT CHAR
274 064E B7         LOOK3   ORA   A         CLEAR CARRY
275 064F CDBED6             CALL  :D6BE     IF BREAK STOP
276 0652 D8                 RC              RECEIVING
277 0653 CD6B04             CALL  CHKSTS    CHECK FOR INCOMING
278 0656 CA4E06             JZ    LOOK3     CHAR'S
279 0659 3AF0FF             LDA   INCHAR
280 065C C9                 RET
281 065D CDBED6     TEST1   CALL  :D6BE
282 0660 DA7C06             JC    STOPTR    ABORT IF BREAK
283 0663 CD6B04             CALL  CHKSTS    LOOK IF CHECKSUM ARRIVES
284 0666 CA5D06             JZ    TEST1     WAIT IF NOT
285 0669 3AF0FF             LDA   INCHAR
286 066C 57                 MOV   D,A       STORE DTE CHKSUM IN REG
287 066D 3A3108             LDA   CHKSUM    DCE CHKSUM IN ACCU
288 0670 BA                 CMP   D
289 0671 CA7C06             JZ    STOPTR    JUMP IF CHKSUM OK
290 0674 3E46               MVI   A,70      NOT OK "F" IN CURSOR
291 0676 327500             STA   :75       AND STAY IN RECEIVE MODE
292 0679 C31706             JMP   LOOK2     ESCAPE WITH BREAK
293 067C 3A2D08     STOPTR  LDA   FLAG2
294 067F FE01               CPI   1         IF HEX,CONVERT & RELOCATE
295 0681 C22107             JNZ   INITEB    IF ASCII, TO EDIT-MODE
296 0684 2AA200             LHLD  STAEB
297 0687 223A08             SHLD  BYTE      START ADDR BYTE-PNTR
298 068A 22A400     CVTBUF  SHLD  EBPTR     ADJUST BUFF-PNTR
299 068D 7E                 MOV   A,M
300 068E FE00               CPI   0         LAST CHAR IN BUFF REACHED?
301 0690 CAB706             JZ    PRADDR    TO RELOCATE ROUTINE
302 0693 FE20               CPI   32        SKIP SPACE & CAR-RET
303 0695 CAB306             JZ    INCREB    AND INCR BUFF-PNTR
304 0698 FE0D               CPI   13
```

```
305 069A CAB306            JZ    INCREB
306 069D CDF006            CALL  HEXPAS    ASCII TO HEX CONVERSION
307 06A0 23                INX   H         FIRST PASS DONE
308 06A1 22A400            SHLD  EBPTR     INCR & ADJUST BUFF-PNTR
309 06A4 7E                MOV   A,M
310 06A5 CDF006            CALL  HEXPAS    SECOND PASS
311 06A8 2A3A08            LHLD  BYTE      STORE FINAL BYTE IN ADDR
312 06AB 77                MOV   M,A       POINTED BY BYTE-PNTR
313 06AC 23                INX   H         INCR BYTE-PNTR
314 06AD 223A08            SHLD  BYTE
315 06B0 2AA400            LHLD  EBPTR
316 06B3 23       INCREB   INX   H         INCR BUFF-PNTR
317 06B4 C38A06            JMP   CVTBUF    NEXT CHAR
318 06B7 21500A   PRADDR   LXI   H,MSG7
```

PAGE 07

```
319 06BA CDD4DA            CALL  :DAD4
320 06BD 2AA200            LHLD  STAEB
321 06C0 CD18ED            CALL  :ED18     PRINT START FILE
322 06C3 EB                XCHG
323 06C4 216C0A            LXI   H,MSG8
324 06C7 CDD4DA            CALL  :DAD4
325 06CA 2A3A08            LHLD  BYTE
326 06CD 2B                DCX   H
327 06CE CD18ED            CALL  :ED18     PRINT END FILE
328 06D1 21740A            LXI   H,MSG9
329 06D4 CDD4DA            CALL  :DAD4
330 06D7 CDC607            CALL  HEXINP    MOVE FILE TO ?
331 06DA 3A3008            LDA   FLAG5
332 06DD FE01              CPI   1         RELOCATE ABORTED?
333 06DF C2E506            JNZ   NBRK
334 06E2 2AA200            LHLD  STAEB     NO MOVE
335 06E5 44       NBRK     MOV   B,H       SWAP
336 06E6 4D                MOV   C,L
337 06E7 2A3A08            LHLD  BYTE
338 06EA CD4FDE            CALL  :DE4F     MOVE ROUTINE
339 06ED C3E604            JMP   SPECL
340 06F0 323208   HEXPAS   STA   STORE1    STORE 1ST CHAR BYTE
341 06F3 79                MOV   A,C
342 06F4 FE01              CPI   1         2ND CHAR OF BYTE DONE?
343 06F6 CA0807            JZ    PASS2
344 06F9 CD1307            CALL  CONVRT    GO TO CONVERT
345 06FC 07                RLC
346 06FD 07                RLC
347 06FE 07                RLC
348 06FF 07                RLC
349 0700 E6F0              ANI   :F0       MASK
350 0702 323308            STA   STORE2    STORE RESULT
351 0705 0E01              MVI   C,1       FIRST CHAR DONE
352 0707 C9                RET             FETCH NEXT CHAR
353 0708 CD1307   PASS2    CALL  CONVRT
354 070B 47                MOV   B,A
355 070C 3A3308            LDA   STORE2    GET PREVIOUS RESULT
356 070F B0                ORA   B         AND MAKE FINAL BYTE
357 0710 0E00              MVI   C,0       ZERO CNTR
358 0712 C9                RET             STORE BYTE IN BUFF
359 0713 3A3208   CONVRT   LDA   STORE1    LOOK IF RECEIVED ASCII
360 0716 FE3A              CPI   58        CHAR IS NUMBER OR
361 0718 D21E07            JNC   LETTER    LETTER AND ADJUST
362 071B D630              SUI   48
363 071D C9                RET
364 071E D637     LETTER   SUI   55
365 0720 C9                RET
366 0721 2AA200   INITEB   LHLD  STAEB
```

```
367 0724 7E       LKETXT  MOV   A,M       SEARCH FOR END OF
368 0725 FE00             CPI   0         TEXT IN EBUFF
369 0727 CA2E07           JZ    ADJPTR
370 072A 23               INX   H
371 072B C32407           JMP   LKETXT

PAGE 08

372 072E 23       ADJPTR  INX   H
373 072F 22A400           SHLD  EBPTR
374 0732 EF               RST   5           INIT SCRN EDIT-MODE
375 0733 2A               DATA  :2A
376 0734 CDBED6   GETC     CALL  :D6BE       WAIT FOR INPUT
377 0737 DAE604           JC    SPECL
378 073A CA3407           JZ    GETC
379 073D EF               RST   5           EDIT ON SCRN
380 073E 2D               DATA  :2D
381 073F C33407           JMP   GETC
382 0742 1603     STAY1   MVI   D,3
383 0744 CD9C04           CALL  WAIT2       SEND END  OF TEXT
384 0747 3A3108           LDA   CHKSUM
385 074A 57               MOV   D,A
386 074B CD9C04           CALL  WAIT2       SEND CHECKSUM
387 074E C3E604           JMP   SPECL
388               *
389               * SEND HEX FILE
390               *
391 0751 210E0A   EDIT2   LXI   H,MSG5
392 0754 CDD4DA           CALL  :DAD4
393 0757 CDC607           CALL  HEXINP      INPUT START ADDR
394 075A 223408           SHLD  STAFIL
395 075D 21300A           LXI   H,MSG6
396 0760 CDD4DA           CALL  :DAD4
397 0763 CDC607           CALL  HEXINP      INPUT END ADDR
398 0766 23               INX   H
399 0767 223608           SHLD  ENDFIL      IF WRONG INPUT,TRY AGAIN
400 076A EB               XCHG
401 076B 2A3408           LHLD  STAFIL
402 076E CD14DE           CALL  :DE14
403 0771 D25107           JNC   EDIT2
404 0774 1682             MVI   D,:82
405 0776 CD8205           CALL  SMSG
406 0779 110000           LXI   D,0         ZERO CNTR'S
407 077C 2A3408           LHLD  STAFIL      ADDR FIRST BYTE
408 077F 223808   NXTCH   SHLD  PTRFIL      POINTS TO BYTE
409 0782 7E               MOV   A,M         BYTE IN ACCU
410 0783 0F       SHIFT   RRC
411 0784 0F               RRC
412 0785 0F               RRC
413 0786 0F               RRC
414 0787 E60F     MASK    ANI   :0F         MASK
415 0789 C630             ADI   48          MAKE ASCII 0-9
416 078B FE3A             CPI   58
417 078D DA9207           JC    STEP        IF NOT A NUMBER
418 0790 C607             ADI   7           MAKE ASCII A-F
419 0792 CD2608   STEP    CALL  VERZ        SEND CHAR
420 0795 1C               INR   E
421 0796 3E02             MVI   A,2         2ND CHAR OF BYTE
422 0798 BB               CMP   E           DONE?
423 0799 CAA007           JZ    NXTBYT      NEXT BYTE
424 079C 7E               MOV   A,M
```

```
425 079D C38707            JMP   MASK       GET 2ND CHAR OF BYTE
426 07A0 3E20      NXTBYT  MVI   A,32       SEND SPACE BETWEEN BYTES
427 07A2 CD2608            CALL  VERZ
428 07A5 1E00              MVI   E,0        CLEAR CHAR CNTR
429 07A7 14                INR   D          INCR BYTE CNTR
430 07A8 3E10              MVI   A,16       LINE FULL?
431 07AA BA                CMP   D          YES, THEN CAR-RET
432 07AB CC1E08            CZ    CARRET
433 07AE 23                INX   H          INCR PNTR
434 07AF E5                PUSH  H
435 07B0 D5                PUSH  D
436 07B1 EB                XCHG
437 07B2 2A3608            LHLD  ENDFIL     END BUFF ?
438 07B5 CD14DE            CALL  :DE14
439 07B8 D1                POP   D
440 07B9 E1                POP   H
441 07BA CA4207            JZ    STAY1      EXIT AFTER END FILE OR
442 07BD CDBED6            CALL  :D6BE      BREAK
443 07C0 DA4207            JC    STAY1
444 07C3 C37F07            JMP   NXTCH      ELSE NEXT BYTE
445 07C6 21DD07    HEXINP  LXI   H,INSBR    ADDR INP SUBR
446 07C9 22D200            SHLD  :D2
447 07CC 0E00              MVI   C,0        CLEAR CHAR CNTR
448 07CE CD2AC0            CALL  :C02A      HEX INPUT TO MACC
449 07D1 21960A            LXI   H,RESULT
450 07D4 E7                RST   4
451 07D5 0F                DATA  :0F
452 07D6 2A980A            LHLD  RESULT+2
453 07D9 7C                MOV   A,H        SWAP
454 07DA 65                MOV   H,L
455 07DB 6F                MOV   L,A
456 07DC C9                RET
457 07DD 79        INSBR   MOV   A,C        CHAR CNTR
458 07DE FE04              CPI   4          MAX 4 CHAR'S
459 07E0 C8                RZ
460 07E1 CDBED6    GHEX    CALL  :D6BE      WAIT FOR INP
461 07E4 CAE107            JZ    GHEX
462 07E7 DAF507            JC    SFL5       IF BREAK TO MENU
463 07EA FE0D              CPI   13
464 07EC C2FD07            JNZ   ALFN       IS CHAR CAR-RET?
465 07EF 47                MOV   B,A
466 07F0 AF                ZAR
467 07F1 B9                CMP   C          FIRST INPUT?
468 07F2 C2FB07            JNZ   EXIT
469 07F5 3E01      SFL5    MVI   A,1        YES,SET ABORTFL
470 07F7 323008            STA   FLAG5
471 07FA C9                RET
472 07FB 78        EXIT    MOV   A,B
473 07FC C9                RET
474 07FD CD09DE    ALFN    CALL  :DE09      TEST ALFANUM
475 0800 D2E107            JNC   GHEX
476 0803 3F                CMC
477 0804 FE47              CPI   71         TEST A-F
```



```
478 0806 D2E107            JNC   GHEX
479 0809 3F                CMC
480 080A CD95D6            CALL  :D695      DISPLAY ON SCRN
481 080D C9                RET
482 080E CDB404    TRCDEL  CALL  TRANSC     TRANSMIT CHAR
483 0811 06FF              MVI   B,:FF      DELAY CONSTANT
484 0813 CD6B04    TEST    CALL  CHKSTS     LOOK FOR RECEIVED
485 0816 C27104            JNZ   INTR1      CHAR'S DURING DELAY
486 0819 05                DCR   B
```

```
487 081A C21308          JNZ   TEST
488 081D C9              RET
489 081E 1600      CARRET MVI   D,0        CLEAR LINE BYTE CNTR.
490 0820 3E0D            MVI   A,13       START NEW LINE
491 0822 CD2608          CALL  VERZ       TRANSMIT
492 0825 C9              RET
493 0826 D5        VERZ  PUSH  D
494 0827 CD0E08          CALL  TRCDEL     TO SEND ROUTINE
495 082A D1              POP   D
496 082B C9              RET
497 082C 00        FLAG1 DATA  0          INIT. FLAG
498 082D 00        FLAG2 DATA  0          ASCII/HEX FLAG
499 082E 00        FLAG3 DATA  0          HALF/FULL DUPLEX FLAG
500 082F 00        FLAG4 DATA  0          TALK FLAG
501 0830 00        FLAG5 DATA  0          ABORT FLAG
502 0831 00        CHKSUM DATA 0
503 0832 00        STORE1 DATA 0
504 0833 00        STORE2 DATA 0
505 0834 0000      STAFIL DBL  0
506 0836 0000      ENDFIL DBL  0
507 0838 0000      PTRFIL DBL  0
508 083A 0000      BYTE   DBL  0
509 083C 0C0D      MSG1   DATA 12,13
510 083E 202020          ASC   '     TALK !!  (cursor-up for MENU)'
511 0860 0D0D00          DATA  13,13,0
512 0863 0C0D      MSG2   DATA 12,13
513 0865 202045          ASC   '  ENTER EDIT MODE     <E>'
514 087E 0D0D            DATA  13,13
515 0880 202048          ASC   '  HALF DUPLEX (DCE)   <H>'
516 0899 0D              DATA  13
517 089A 202028          ASC   '  (echo"s to DTE)'
518 08AB 0D              DATA  13
519 08AC 202046          ASC   '  FULL DUPLEX (DTE)   <F>'
520 08C5 0D0D            DATA  13,13
521 08C7 202054          ASC   '  TALK                <T>'
522 08E0 0D0D            DATA  13,13
523 08E2 202053          ASC   '  SEND DNA SOURCE     <S>'
524 08FB 0D              DATA  13
525 08FC 202028          ASC   '  (+warm start init.)'
526 0911 0D              DATA  13
527 0912 202053          ASC   '  SEND ASCII FILE     <A>'
528 092B 0D              DATA  13
529 092C 202053          ASC   '  SEND HEX  FILE      <#>'
530 0945 0D0D            DATA  13,13
```

PAGE 11

```
531 0947 202052          ASC   '  RECEIVE ASCII FILE  <R>'
532 0960 0D              DATA  13
533 0961 202052          ASC   '  RECEIVE HEX  FILE   <X>'
534 097A 0D0D            DATA  13,13
535 097C 202042          ASC   '  BACK TO UTILITY     <U>'
536 0995 0D0D            DATA  13,13
537 0997 202042          ASC   '  BACK TO BASIC       <B>'
538 09B0 0D0D0D          DATA  13,13,13
539 09B3 202054          ASC   '  TYPE INSTRUCTION !   '
540 09CA 00              DATA  0
541 09CB 0C0D      MSG3  DATA  12,13
542 09CD 202020          ASC   '                    SEND FILE'
543 09E8 0D0D00          DATA  13,13,0
544 09EB 0C0D      MSG4  DATA  12,13
545 09ED 202020          ASC   '                    RECEIVE FILE'
546 0A0B 0D0D00          DATA  13,13,0
547 0A0E 0C0D0D    MSG5  DATA  12,13,13
548 0A11 494E50          ASC   'INPUT START ADDRESS OF FILE  #'
```

```
549 0A2F 00                DATA  0
550 0A30 0D          MSG6  DATA  13
551 0A31 494E50            ASC   'INPUT LAST ADDRESS  OF FILE  #'
552 0A4F 00                DATA  0
553 0A50 0C0D0D      MSG7  DATA  12,13,13
554 0A53 484558            ASC   'HEX FILE LOCATED FROM  #'
555 0A6B 00                DATA  0
556 0A6C 202054      MSG8  ASC   '  TO  #'
557 0A73 00                DATA  0
558 0A74 0D0D        MSG9  DATA  13,13
559 0A76 52454C            ASC   'RELOCATE TO (GIVE STARTADDR.) #'
560 0A95 00                DATA  0
561 0A96             RESULT RES  4,0
562 0A9A             ENDPRG END

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

ADJPTR 072E   ALFN   07FD   BACKSP 0477   BDRATE 0084
BYTE   083A   CARRET 081E   CHKFL1 05D2   CHKSTS 046B
CHKSUM 0831   CHOISE 04FE   CLCHKS 05AA   CLEBUF 045F
CONVRT 0713   CVTBUF 068A   DISPL  0631   EBPTR  00A4
ECHO   048F   EDIT   05AF   EDIT2  0751   ENDEB  00A6
ENDFIL 0836   ENDPRG 0A9A   ENDWST 0454   EXIT   07FB
FLAG1  082C   FLAG2  082D   FLAG3  082E   FLAG4  082F
FLAG5  0830   FULLD  0554   GETC   0734   GHEX   07E1
HALFD  054C   HDUPLX 04E2   HEXINP 07C6   HEXPAS 06F0
INCHAR FFF0   INCREB 06B3   INITEB 0721   INSBR  07DD
INTR1  0471   LETTER 071E   LKETXT 0724   LOOK   0614
LOOK2  0617   LOOK3  064E   MAINLP 0406   MASK   0787
MSG1   083C   MSG2   0863   MSG3   09CB   MSG4   09EB
MSG5   0A0E   MSG6   0A30   MSG7   0A50   MSG8   0A6C
MSG9   0A74   NBRK   06E5   NEXTCH 05BA   NXTBYT 07A0
NXTCH  077F   OUT    04CD   PASS2  0708   PRADDR 06B7

PAGE 12

PTRFIL 0838   READ   0601   RECASC 056D   RECHEX 0577
RESULT 0A96   RMSG   0598   SASCI  0563   SCREEN 0484
SDNAS  05B7   SFL5   07F5   SHEX   055B   SHIFT  0783
SMSG   0582   SPECL  04E6   SSOURC 05EB   SSTEB  05E0
STAEB  00A2   STAFIL 0834   START  0423   STAY1  0742
STEP   0792   STOPTR 067C   STORE1 0832   STORE2 0833
STSREG FFF3   SUMCHA 04D9   TALK   053B   TEST   0813
TEST1  065D   TOBAS  0544   TOUT   0418   TRANS  04AD
TRANSC 04B4   TRCDEL 080E   VERZ   0826   WAIT   04A3
WAIT2  049C
```

* neu * nieuw * new * nouveau *

* Schreibschrift *
* writing-characters *

DAI # Itho 8510

****

Das neue 4k Eprom 'SSV1' ermöglicht die Darstellung von zwei verschiedenen Zeichensätzen im mixed Mode. Das Eprom enthält einen deutschen Schreibschrift- und einen deutschen Standartzeichensatz. Setzt man ein Eprom vom Typ SSV1-DAI in den DAI-Personal Computer und ein Eprom vom Typ SSV1-Itho in den Drucker Itho 8510, so lassen sich beide Zeichensätze sowohl mit dem DAI als auch mit dem Drucker über Steuerzeichen im mixed Mode darstellen.

The new 4k eprom 'SSV1' permits the display of two different character sets in mixed mode. The eprom contains a german writing and a german standart set. If you put for example one eprom type SSV1-DAI into the DAI-Personal Computer and one eprom type SSV1-Itho into the Itho 8510 printer, thus both character sets can be displayed in mixed mode on the DAI-Personal Computer as well as on the Itho printer by useing control signs.

#A0	#A1 !	#A2 "	#A3 #	#A4 $	#A5 %	#A6 &	#A7 '	#A8 (
#A9)	#AA *	#AB +	#AC ,	#AD -	#AE .	#AF /	#B0 0	#B1 1
#B2 2	#B3 3	#B4 4	#B5 5	#B6 6	#B7 7	#B8 8	#B9 9	#BA :
#BB ;	#BC <	#BD =	#BE >	#BF ?	#C0 §	#C1 A	#C2 B	#C3 C
#C4 D	#C5 E	#C6 F	#C7 G	#C8 H	#C9 I	#CA J	#CB K	#CC L
#CD M	#CE N	#CF O	#D0 P	#D1 Q	#D2 R	#D3 S	#D4 T	#D5 U
#D6 V	#D7 W	#D8 X	#D9 Y	#DA Z	#DB Ä	#DC Ö	#DD Ü	#DE ^
#DF _	#E0 `	#E1 a	#E2 b	#E3 c	#E4 d	#E5 e	#E6 f	#E7 g
#E8 h	#E9 i	#EA j	#EB k	#EC l	#ED m	#EE n	#EF o	#F0 p
#F1 q	#F2 r	#F3 s	#F4 t	#F5 u	#F6 v	#F7 w	#F8 x	#F9 y
#FA z	#FB ä	#FC ö	#FD ü	#FE ß				

1 Eprom 49,- DM
+ 7,50 DM post & packing.

address:

Frank Caßebaum
Grenzstraße 64
D-2800 Bremen 01
Tel.: 0421/3 96 23 53

eprom type:

SSV1-DAI for DAI-PC
SSV1-Itho for Itho printer 8510

UMLA-Epson for Epson printer LX80 with standart character set including german 'Umlaute' and kraftrax (Plotter)

your order by check or on Bancaccount nr: 550005301309
Verbraucherbank Bremen
Bankleitzahl: 29020300
West - Germany

* neu * nieuw * new * nouveau *

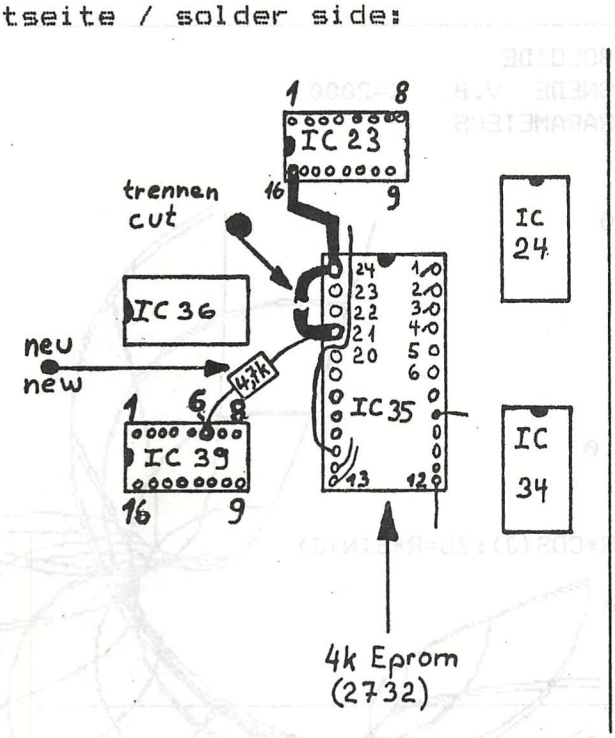
* 256 *
* Zeichen *
* characters *

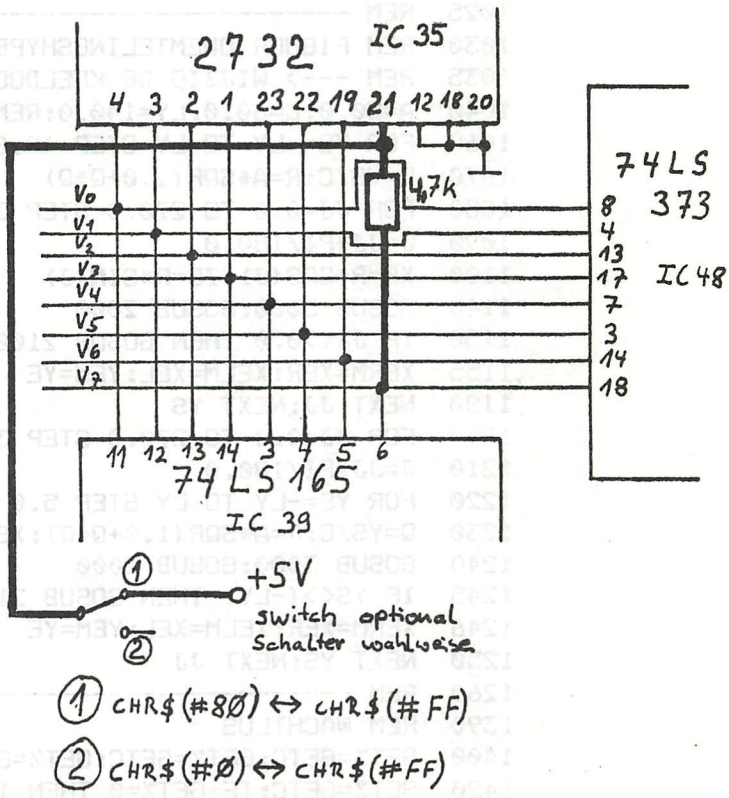
****

Eine kleine Änderung Ihrem DAI-PC ermöglicht die Darstellung von zusätzlichen 125 Zeichen. Es sind lediglich eine Leiterbahn zu unterbrechen, ein 4,7 kOhm Widerstand und ein 4k Eprom einzubauen. Nach diesem Umbau können Zeichen von #0 bis #FF mit dem DAI dargestellt werden.

Just a little change in your DAI-PC permits the display of additional 125 characters. You have to cut a track and to insert a resistor and a 4k eprom. After doing this, characters from #0 to #FF can be displayed with your DAI-PC.

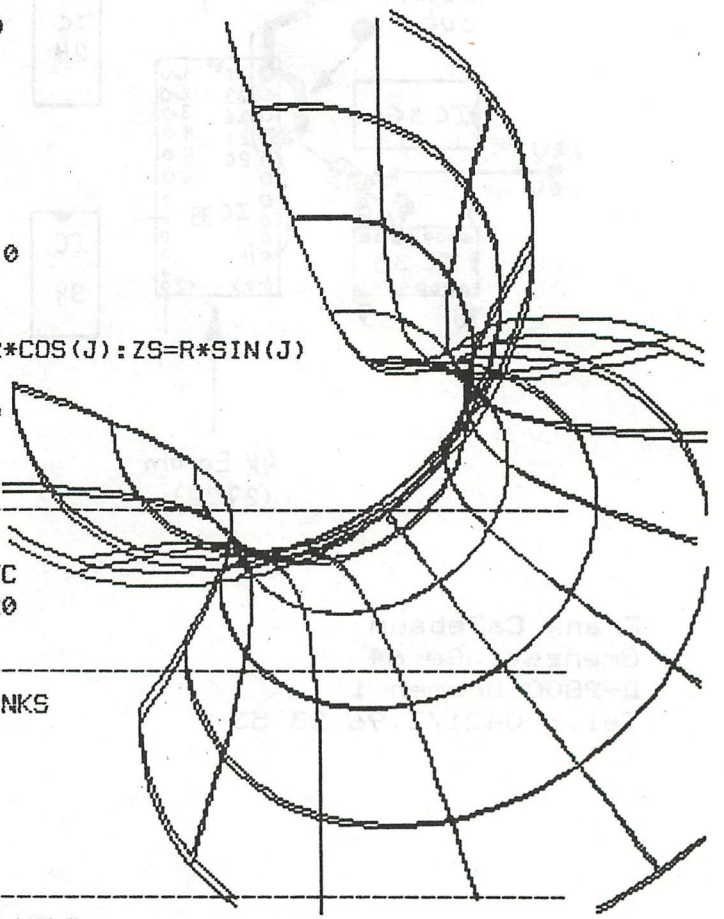
Lötseite / solder side:



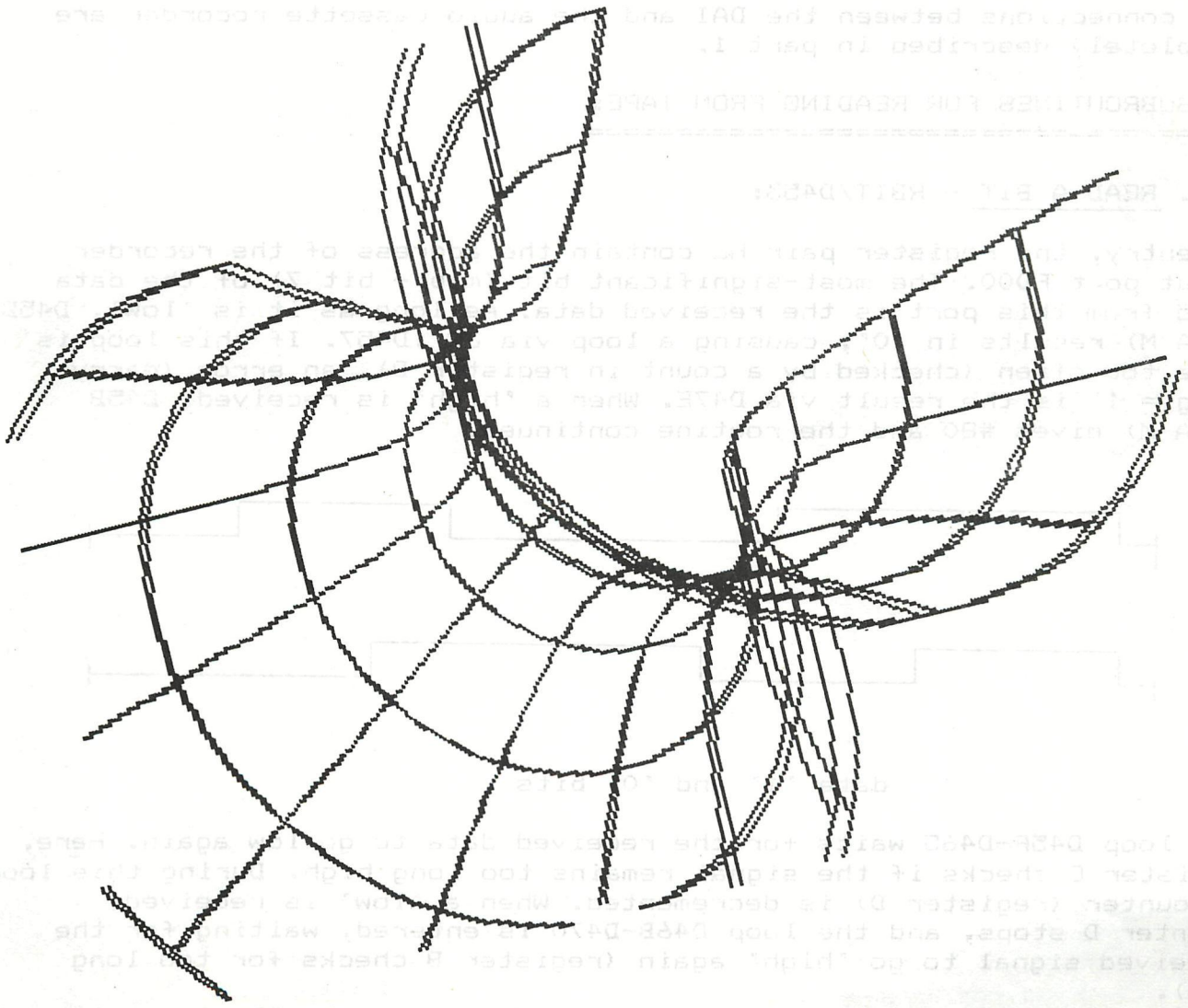


Frank Caßebaum
Grenzstraße 64
D-2800 Bremen 1
Tel.: 0421/3 96 23 53

```
50    MODE 0:PRINT CHR$(12):REM TEKST MODE - VAAG SCHERM
65    CURSOR 0,18
66    PRINT "  ************************************************************"
67    PRINT "  *                                                          *"
70    PRINT "  *  D R I E   D I M E N S I O N E E L   T E K E N E N   *"
72    PRINT "  *                                                          *"
73    PRINT "  *                                                          *"
74    PRINT "  *                    EENBLADIGE                            *"
75    PRINT "  *                                                          *"
76    PRINT "  *             OMWEMTELINGSHYPERBOLOIDE                     *"
77    PRINT "  *                                                          *"
80    PRINT "  *                                              J.ROELANTS  *"
82    PRINT "  ************************************************************"
84    PRINT :PRINT :PRINT
90    REM ------------------------------------------------------------
100   V=2000.0:L=35.0:REM WAARNEMINGSAFSTAND - AFSTAND TUSSEN W.PUNTEN
600   PRINT " Geef de verdraaiing rond X,Y,Z AS in graden (v.b.70,30,0)"
605   INPUT "----> ";A,B,C:PRINT
610   REM OMZETTING NAAR RADIALEN
620   A=A/57.296:B=B/57.296:C=C/57.296
640   GOSUB 4000
1000  MODE 6:MODE 6:REM GRA FISCHE MODE 336 x 256
1010  REM KLEURKEUZE ZWART ROOD GROEN GRIJS
1020  COLOR6 0 3 5 8
1025  REM ------------------------------------------------------------
1030  REM FIGUUR OWEMTELINGSHYPERBOLOIDE
1035  REM ---> WIJZIG DE KEELDOORSNEDE  V.B.  C=2000
1040  A=50.0:C=60.0:LY=140.0:REM RARAMETERS
1060  FOR YS=-LY TO LY STEP 40.0
1070  Q=YS/C:R=A*SQR(1.0+Q*Q)
1080  FOR JJ=0.0 TO 270.0 STEP 3.0
1090  J=JJ*PI/180.0
1100  XS=R*COS(J):ZS=R*SIN(J)
1140  GOSUB 3000:GOSUB 2000
1150  IF JJ<>0.0 THEN GOSUB 2100
1155  XERM=XER:XELM=XEL:YEM=YE
1180  NEXT JJ:NEXT YS
1200  FOR JJ=0.0 TO 270.0 STEP 30.0
1210  J=JJ*PI/180.0
1220  FOR YS=-LY TO LY STEP 5.0
1230  Q=YS/C:R=A*SQR(1.0+Q*Q):XS=R*COS(J):ZS=R*SIN(J)
1240  GOSUB 3000:GOSUB 2000
1245  IF YS<>(-LY) THEN GOSUB 2100
1246  XERM=XER:XELM=XEL:YEM=YE
1250  NEXT YS:NEXT JJ
1260  REM ------------------------------------------------------------
1390  REM WACHTLUS
1400  GET%=GETC:GET%=GETC:GET%=GETC
1420  GET%=GETC:IF GET%=0 THEN 1420
1500  GOTO 50
1990  REM ------------------------------------------------------------
2000  REM BEELDPUNTEN RECHTS EN LINKS
2020  K=V/(ZP+V)
2040  D=(1.0-K)*L:KX=K*XP+168.5
2060  XER=KX+D:YE=K*YP+128.5
2080  XEL=KX-D
2090  RETURN
2095  REM ------------------------------------------------------------
2100  REM TEST OF LIJN NIET BUITEN VALT
2110  IF YE<0.0 OR YE>YMAX OR YEM<0.0 OR YEM>YMAX THEN 2220
2120  IF XER<0.0 OR XER>XMAX OR XERM<0.0 OR XERM>XMAX THEN 2220
2130  IF XEL<0.0 OR XEL>XMAX OR XELM<0.0 OR XELM>XMAX THEN 2220
2140  REM TEKEN RODE EN GROENE LIJN
```



```
2150  DRAW XER,YE XERM,YEM 17:DRAW XEL,YE XELM,YEM 19
2220  RETURN
2230  REM ------------------------------------------------------------
3000  REM ROTATIE
3010  XP=AX*XS+BX*YS+CX*ZS
3020  YP=AY*XS+BY*YS+CY*ZS
3040  ZP=AZ*XS+BZ*YS+CZ*ZS
3200  RETURN
3990  REM ------------------------------------------------------------
4000  AX=COS(B)*COS(C)
4020  BX=SIN(A)*SIN(B)*COS(C)-COS(A)*SIN(C)
4040  CX=COS(A)*SIN(B)*COS(C)+SIN(A)*SIN(C)
4060  AY=COS(B)*SIN(C)
4070  BY=SIN(A)*SIN(B)*SIN(C)+COS(A)*COS(C)
4080  CY=SIN(C)*SIN(B)*COS(A)-COS(C)*SIN(A)
4090  AZ=(-1.0)*SIN(B)
4100  BZ=SIN(A)*COS(B)
4120  CZ=COS(A)*COS(B)
4200  RETURN
4220  REM ************************************************************
```



AUDIO-CASSETTE INTERFACE

PART 2: READING FROM TAPE

1. INTRODUCTION:

1.1. WRITING TO TAPE:

See a previous Newsletter for part 1 of this article about writing to tape.
Details on the software routines can be found in the DAI pc Firmware Manual.

1.2. SIGNAL FORMATS:

In part 1, the format of the signals on tape, the routines used for writing and the set-up of a tape file are explained.
Part 2 describes the way signals are retrieved from tape by the DAI resident software.

The principle of reading tape files is identical to the way they are written, but of course just the other way around.
The reading routines will be described in the same sequence as done with the writing routines.

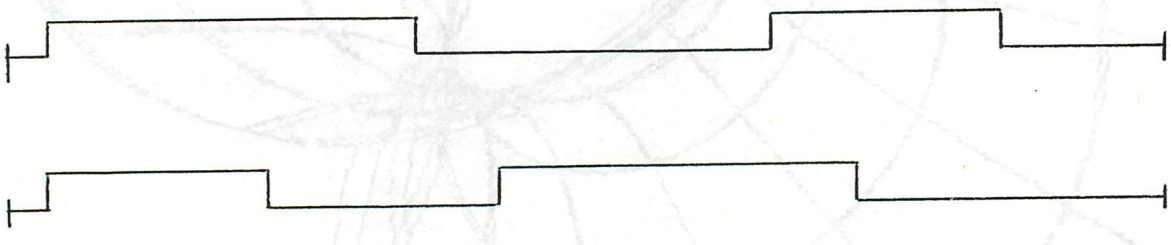
2. HARDWARE CONNECTIONS:

The connections between the DAI and the audio cassette recorder are completely described in part 1.

3. SUBROUTINES FOR READING FROM TAPE:

3.1. READ A BIT - RBIT/D453:

On entry, the register pair HL contain the address of the recorder input port FD00. The most-significant bit (msb - bit 7) of the data read from this port is the received data. As long as it is 'low', D45B (ORA M) results in '0', causing a loop via JP :D457. If this loop is done too often (checked by a count in register B), an error (carry-flag = 1) is the result via D47E. When a 'high' is received, D45B (ORA M) gives #80 and the routine continues.



data '1' and '0' bits

The loop D45F-D465 waits for the received data to go low again. Here, register C checks if the signal remains too long high. During this loop, a counter (register D) is decremented. When a 'low' is received, counter D stops, and the loop D46B-D470 is entered, waiting for the received signal to go 'high' again (register B checks for too long low).
If this happens, loop D473-D479 waits for a return to 'low' (register C checks too long high). During this loop, counter D is now incremented until the signal goes low.

Depending on which impulse is the longest, the value of counter D at the end will be positive or negative:

1st impulse longer: D<0	data = '1'
2nd impulse longer: D>0	data = '0'

The contents of counter D is moved into the accumulator (D47C). If D<0, the msb of A is '1'. If D>0, the msb is '0'. So the received bit is now bit 7 of the accumulator.

If a reading error occurs (the signal is too long high or low), the routine will exit with the carry-flag set (CY=1) via D47E.

Note that reading a bit is time-independent. No relation with a fixed impulse length is used, only a compare between both impulses is made.

3.2. READ A BYTE - RBYTE/D4D4:

Because a byte consists of 8 bits, the routine RBIT has to be performed 8 times.
The register E is loaded with #FE (1111 1110). A bit is read via RBIT into the msb of the accumulator. Via the carry-flag, this bit is shifted into the register E (D4E2-D4E5). As long as the bit which is shifted out of E is a '1', the routine is repeated (totally 8 times) via JC :D4DC.
Each time a new bit is read, it is shifted behind the previous one into register E via D4E2-D4E5.
After 8 times, the CY-flag is '0'. Then the received byte is available in the accumulator.

If during RBIT a reading error occurs, RBYTE is aborted via D4DF (JC :D4E9). Then on exit of RBYTE, the carry-flag will be set, indicating a loading error.

3.3. READ A LEADER - RLEAD/D480:

Is the absolute timing not important when reading data bits, for the recognition of the leader tone on tape it is very important.
Reading a leader tone consists of reading a continuous tone from tape and checking if its frequency is within a certain margin. If this is true long enough, the tone is recognised as a leader.

- D48A-D48E: As long as the received signal (via port FD00) is high, the loop waits for the signal to go low. Via EI/DI, the cursor remains flashing.
- D491-D49C: Now is waited for the signal to go high. If it lasts too long (register E counts), the leader reading routine is restarted (JZ :D488).
- D49F-D4A6: As soon as the signal goes high, register B is used as a counter. It is incremented as long as the signal remains high. A too long high level restarts the leader check routine.
- D4A9-D4AB: As soon as the signal goes low again, the duration count in register B is compared with an estimate (in register C: #28). The difference is stored in register E (D4B0).
- D4B0-D4B8: The tolerated margin is calculated (04) and compared with the difference in register E. If the difference is too large (that means: no leader impulse), JC :D4C3 occurs.
- D4BB-D4C0: If the received impulse has a length which is within the margin, the next impulse is checked via JNZ :D494. This is repeated 20 times (register D counts) to be sure the received signal is a real leader tone.

If 20 times (#14) a correct impulse is received (that means: leader tone found), the routine waits in a loop for other impulses than those of the leader. Because the leader consists of 2024 impulse pairs of the duration '24', followed by a data '1' bit (duration '3C/24'), the '3C' impulse of the latter will cause the routine to be aborted because it differs too much from the margin. Then D4B8 causes JC :D4C3.

- D4C3-D4C4: If 'out of margin' occurs during the first 20 runs of the routine D494-D4BC, the register D is not yet 0. That means synchronisation is not yet found, and the routine starts anew.
More than 20 runs means synchronisation is detected, and the program continues.
- D4C7-D4CD: After the first impulse of the data '1' bit after the leader tone (duration '3C': out of margin), here the second impulse of this bit is handled.

4. FILE HANDLING:

Read again carefully what is written in part 1 about the set-up of a file.
For audio-cassette recorder operation, the resident software has routines to open a file from tape (CROPEN), read data blocks from tape (CRBLK) and to close a file read from tape (CRCLOSE).
The startaddresses of these file handling routines are moved from ROM into RAM during reset. The default addresses for audio-cassette recorder operation are:

```
ROPEN    #02CE    JMP #D325
RBLK     #02D1    JMP #D340
RCLOSE   #02D4    JMP #D445
```

4.1. OPEN A TAPE FILE - CROPEN/D325:

- Get length of name expected (if given) in register pair DE; HL points to the name (MPT23/D7FF). Switch on cassette motors (CASST/D42E).
- Disable sound interrupts (SNDDI/D98F).
- Read the header (RHDR/D3F4):
  - Read leader, find synchronisation (RLEAD/D480).
  - Read flag byte #55 (RBYTE/D4D4).
  - Read file type byte (RBYTE/D4D4):
    - #30: A Basic program.
    - #31: A Hex file.
    - #32: An array.
- If not load during program run: Display file type byte on the screen (MPT24/D78A); This byte is directly POKE'd into the screen memory.
- Read the name of the program (CMBLK/D337).
  The length of the name (and its checksum) is read, then the name is read and POKE'd into the screen memory, and the checksum on the name is read.
  If a loading error occurs on one of these read routines, a loading error report is prepared via D3ED. If all checks are O.K., the CROPEN routine is ready now.

4.2. READ A BLOCK FROM TAPE - CRBLK/D340:

- Calculate the available memory space (MPT25/D790).
- Read the length of the block and the checksum on the length (INLNG/D38D).
  An error is reported if a reading error (via D37E) or checksum error occurs or if the available memory space is too small (via D380).
- Read the contents of the block (D364-D36C). The error exit via D37E is taken if a reading error occurs.
- Read the checksum on the block contents (RBYTE/D4D4). Here again the error exit via D380 if a checksum error has been found.

4.3. CLOSE A TAPE FILE - CRCLOSE/D445:

- Switch off the cassette motors (CASSP/D445).

Note: The trailer tone on the tape is not written !

5. RESIDENT FILE READING ROUTINES:

Three resident file reading routines are available:

'LOAD'	file type 0	Read Basic programs.
'UT/Read'	file type 1	Read Hex files.
'LOADA'	file type 2	Read Arrays.

The use of the routines ROPEN, RBLK and RCLOSE is not 100% identical for each of these methods.

5.1. BASIC COMMAND 'LOAD' - RLOAD/D270:

- D270: Clear all variables in the heap and in the symbol table. Evaluate the (eventually) expected file name. Check if load during program run.
- D28A: ROPEN (see 4.1).
- D28D: Get pointers to begin of textbuffer (TXTBGN) and bottom of screen memory (SCRNBOT) for calculation of available memory space.
- D294: RBLK: Read length of textbuffer + checksum. Idem for textbuffer contents.
- D297: Get pointer to start of symboltable for calculation of available memory space.
- D29A: RBLK: Read length of symboltable + checksum. Idem for symboltable contents.
- D29D: Store endaddress of symboltable. RCLOSE (see 4.3).
- D2A2: Eventually, run 'LOADING ERROR 0/1/2/3'.

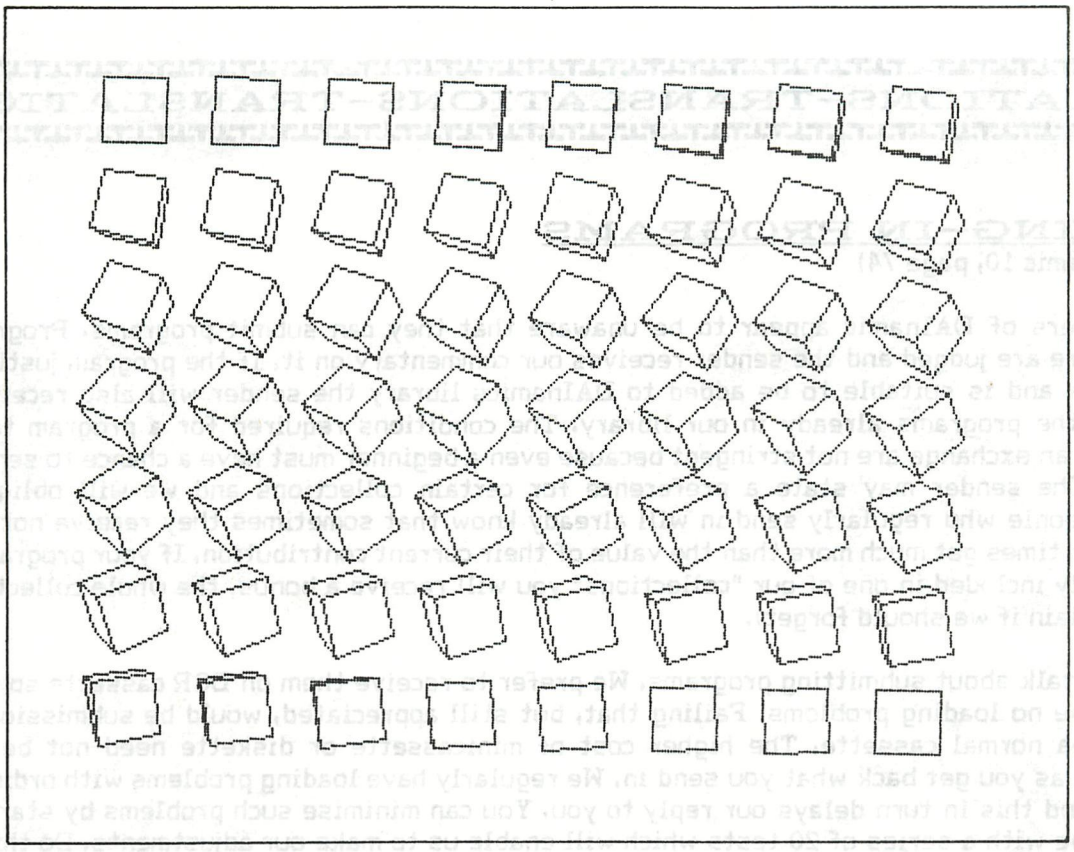
5.2. UTILITY COMMAND 'R(ead)' - RHEXK/3EF0F:

- EF0F: Get an evt. offset for startaddress on stack.
- EF14: Get an evt. name in the encoded input buffer.
- EF1E: ROPEN.
- EF21: Sets the maximum available RAM area to F900 (end of stack RAM). This is incorrect, because C000-EFFF is ROM and F000-F7FF is also not available.
- EF25: RBLK: Read startaddress from tape.
- EF28: Add an evt. offset given.
- EF29: RBLK: read the data block from tape. RCLOSE: Stop reading.
- EF2A: Run an evt. loading error (print '?').

5.3. BASIC COMMAND 'LOADA' - RLODA/D85E:

- D85E: Evaluate type of array to be loaded.
- D863: Evaluate an evt. program name; switch to ROM bank 1.
- 1EE0F: ROPEN; RBLK: read array type.
  If INT/FPT arrays: RBLK: read array contents into the array in the heap.
  If string arrays: RBLK: read the array contents into the free RAM space (see Newsletter 1981/p.10). Then move the data into the heap.
  Switch to ROM bank 0.
  RCLOSE.

```
10    CLEAR 256:MODE 0:CLEAR 1500
11    REM ** Cube rotating and translating in space **
20    SV1!=0.0:SV2!=0.0:SV3!=0.0:SQ1!=0.0:SQ2!=0.0:SQ3!=0.0
21    IPTS=0:IPTF=0:K1=0:K2=0:F!=5.9
22    X=-21:XT=42:XS=21:Y=240:YT=37:DF=23
30    NV=8:NE=12:NS=6:CO=3:AN=4
31    DIM M!(CO,CO),NORM!(CO,NS),VERT!(CO,NV)
32    DIM EDGV1(NE),EDGV2(NE),PEDG(AN*NS)
33    DIM EDG(NE),XP!(NE),YP!(NE)
50    GOSUB 11000:MODE 6:COLORG 8 0 10 15
3000  FOR L=1 TO 56
3002  FOR U=1 TO NE:EDG(U)=0:NEXT
3004  IF L=1 OR L=56 THEN RESTORE:GOSUB 10000:GOTO 3020
3010  FOR U=1 TO NV
3011  SV1!=VERT!(1.0,U):SV2!=VERT!(2.0,U):SV3!=VERT!(3.0,U)
3012  VERT!(1.0,U)=SV1!*M!(1.0,1.0)+SV2!*M!(1.0,2.0)+SV3!*M!(1.0,3.0)
3013  VERT!(2.0,U)=SV1!*M!(2.0,1.0)+SV2!*M!(2.0,2.0)+SV3!*M!(2.0,3.0)
3014  VERT!(3.0,U)=SV1!*M!(3.0,1.0)+SV2!*M!(3.0,2.0)+SV3!*M!(3.0,3.0)
3019  NEXT
3020  IPTS=1:FOR U=1 TO NS
3022  SQ1!=NORM!(1.0,U):SQ2!=NORM!(2.0,U):SQ3!=NORM!(3.0,U)
3023  NORM!(1.0,U)=SQ1!*M!(1.0,1.0)+SQ2!*M!(1.0,2.0)+SQ3!*M!(1.0,3.0)
3024  NORM!(2.0,U)=SQ1!*M!(2.0,1.0)+SQ2!*M!(2.0,2.0)+SQ3!*M!(2.0,3.0)
3025  NORM!(3.0,U)=SQ1!*M!(3.0,1.0)+SQ2!*M!(3.0,2.0)+SQ3!*M!(3.0,3.0)
3030  IPTF=IPTS+3:K1=PEDG(IPTS):K2=EDGV1(K1)
3031  XN2!=-VERT!(1.0,K2):YN2!=-VERT!(2.0,K2):ZN2!=-VERT!(3.0,K2)-F!
3032  DT!=NORM!(1.0,U)*XN2!+NORM!(2.0,U)*YN2!+NORM!(3.0,U)*ZN2!:IF DT!<0.0
GOTO 3041
3040  FOR V=IPTS TO IPTF:VV=PEDG(V):EDG(VV)=EDG(VV)+1.0:NEXT
3041  IPTS=IPTS+4
3049  NEXT
3050  FOR U=1 TO NV:ZPF!=VERT!(3.0,U)+F!
3051  XP!(U)=F!*VERT!(1.0,U)/ZPF!:YP!(U)=F!*VERT!(2.0,U)/ZPF!:NEXT
3055  X=X+XT:IF X>XMAX THEN X=XS:Y=Y-YT
3060  FOR U=1 TO NE:IF EDG(U)=0.0 GOTO 3069
3062  X1=XP!(EDGV1(U))*DF:Y1=YP!(EDGV1(U))*DF
3063  X2=XP!(EDGV2(U))*DF:Y2=YP!(EDGV2(U))*DF
3065  DRAW X+X1,Y+Y1 X+X2,Y+Y2 0
3069  NEXT
3099  NEXT
9999  END
10000 FOR I=1 TO CO:FOR J=1 TO NS:READ K:NORM!(I,J)=K:NEXT:NEXT
10002 DATA 0,-1,0,1,0,0,0,0,0,0,-1,1,-1,0,1,0,0,0
10004 FOR I=1 TO CO:FOR J=1 TO NV:READ K:VERT!(I,J)=K-0.5:NEXT:NEXT
10006 DATA 0,0,1,1,0,0,1,1,0,1,1,0,0,1,1,0,0,0,0,0,1,1,1,1
10008 FOR I=1 TO NE:READ EDGV1(I):NEXT
10010 DATA 1,2,3,4,5,6,7,8,1,2,3,4
10012 FOR I=1 TO NE:READ EDGV2(I):NEXT
10014 DATA 2,3,4,1,6,7,8,5,5,6,7,8
10016 FOR I=1 TO NS*AN:READ PEDG(I):NEXT
10018 DATA 1,2,3,4,1,9,5,10,8,7,6,5,3,11,7,12,4,12,8,9,2,10,6,11
10099 RETURN
11000 RAD!=PI/(180.0/1.25):SN!=SIN(RAD!):CS!=COS(RAD!)
11001 M!(1.0,1.0)=SN!*SN!*SN!+CS!*CS!:M!(1.0,2.0)=SN!*CS!:M!(1.0,3.0)=SN!*C
S!-SN!*SN!*CS!
11002 M!(2.0,1.0)=-SN!*CS!+SN!*SN!*CS!:M!(2.0,2.0)=CS!*CS!:M!(2.0,3.0)=-SN!
*SN!-SN!*CS!*CS!
11003 M!(3.0,1.0)=-SN!*CS!:M!(3.0,2.0)=SN!:M!(3.0,3.0)=CS!*CS!
11099 RETURN
```



LES MESSAGES D'ERREUR

Les 26 messages d'erreur du BASIC sont affiches grace a un programme qui se trouve a l'adresse

D9F5

L'acces a ce programme se fait de la maniere suivante:

```
3E nn     -   MVI  A,VAL
C3 F5 D9  -   JUMP :D9F5
```

Avec nn = VAL = numero du message. Ci dessous le tableau qui donne la valeur nn (VAL) pour chaque message.

00 : NEXT WITHOUT FOR	0E : LOADING ERROR 3
01 : RETURN WITHOUT GOSUB	0F : UNDEFINED ARRAY
02 : OUT OF DATA	10 : COLOR NOT AVAILABLE
03 : OVERFLOW	11 : OFF SCREEN
04 : UNDEFINED LINE NUMBER	12 : ERROR LINE RUN
05 : SUBSCRIPT ERROR	13 : OUT OF MEMORY
06 : DIVISION BY ZERO	14 : TYPE MISMATCH
07 : OUT OF STRING SPACE	15 : LINE NUMBER OUT OF RANGE
08 : STRING TOO LONG	16 : STACK OVERFLOW
09 : NUMBER OUT OF RANGE	17 : SYNTAX ERROR
0A : INVALID NUMBER	18 : COMMAND INVALID
0B : LOADING ERROR 0	19 : CAN'T CONT
0C : LOADING ERROR 1	1A : LINE TOO COMPLEX
0D : LOADING ERROR 2	

SENDING-IN PROGRAMS

(from DAInamic 10, page 74)

Many members of DAInamic appear to be unaware that they can submit programs. Programs received here are judged and the sender receives our commentary on it. If the program justifies an exchange and is suitable to be added to DAInamics library the sender will also receive a number of the programs already in our library. The conditions required for a program to be eligible for an exchange are not stringent because even a beginner must have a chance to send in his work. The sender may state a preference for certain collections and we will oblige if possible. People who regularly send in will already know that sometimes they receive nothing but at other times get much more than the value of their current contribution. If your program is subsequently included in one of our "collections" you will receive a bonus, the whole collection, free. (Complain if we should forget).

Now let us talk about submitting programs. We prefer to receive them on DCR cassette so that there will be no loading problems. Failing that, but still appreciated, would be submission on disc or on a normal cassette. The higher cost of minicassette or diskette need not be too burdensome as you get back what you send in. We regularly have loading problems with ordinary cassettes and this in turn delays our reply to you. You can minimise such problems by starting off your tape with a series of 20 tests which will enable us to make our adjustments. Do this in the following way:-

```
NEW return
DIM A(0) return
FOR I=1 TO 20 : SAVEA A "TEST" : NEXT return
and of course do have the recorder on Record !
```

Furthermore you can assist with an accompanying letter, telling us what to expect on the tape. That saves us running the tape right through to its end when there is nothing more on it. An actual written letter is needed because the same information on a tape would be useless if, because of loading problems, it could not be read. Now for the most important part, the program itself.

May I give a spot of advice to all who are about to send in programs, do let someone with no knowledge of computers try out your program first. I have had scores of programs which after a RUN left me waiting only to find that the control was in a closed loop, or sometimes expecting me to give a particular response, although at that time I would have had no knowledge of what was required. At other times, an apparently correct program takes a long time reading in data, and this can be annoying if there has been no prior indication, preferably by an announcement and a running counter which indicates progress. I have also had many programs which after a RUN report OUT OF MEMORY, COLOUR NOT AVAILABLE, NUMBER OUT OF RANGE, OUT OF STRING SPACE, UNDEFINED ARRAY, OUT OF DATA and almost every other message, while further on in the program OFF SCREEN is the winner. Check your programs thoroughly - often another can do it better. A program with a confirmed fault will not be included in a collection. You can avoid many faults by improving the structure of the program. Remember that everybody does not see the same picture equally well, so your chosen colour combination could be difficult for another to read. I find 4/5, 4/8, 4/12 unuseable, but 6/7 and 7/8 are nice and clear. There are obviously still many aspects remaining to be discussed but I would like to end this article with a plea to send in something. Remember that we all started as beginners, and in any case, nobody yet turns out perfect programs. Beginners especially often show originality which many old-timers cannot match.

Frank H. Druijff

```
100   GOTO 500:REM Gemaakt door Hendrik-Jan van Randen.
150   X1=XM-X1(T):Y1=YM-Y1(T):X2=XM-X2(T):Y2=YM-Y2(T)
160   DRAW X1(T),Y1(T) X2(T),Y2(T) Z:DRAW X1,Y1 X2,Y2 Z:DRAW X1,Y1(T) X2,Y2
(T) Z:DRAW X1(T),Y1 X2(T),Y2 Z:RETURN
200   FOR W=0 TO AL STEP A:K=21+RND(3):K(K-21)=1+RND(#F):COLORG 0 K(0) K(1)
 K(2):Z=20:T=W:GOSUB 150
210   X1(W)=RND(XM):Y1(W)=RND(YM):DX=(RND(XM)-X1(W))/A:DY=(RND(YM)-Y1(W))/A
:X2(W)=X1(O):Y2(W)=Y1(O):Z=K:GOSUB 150:V=W
250   FOR T=W+1 TO W+A-1:O=O+1:Z=20:GOSUB 150:X1(T)=X1(V)+DX:Y1(T)=Y1(V)+DY
:X2(T)=X1(O):Y2(T)=Y1(O):Z=K:GOSUB 150:V=T:NEXT
260   O=W:NEXT:W=0:GOTO 200
500   CLEAR 5000:MODE 6:MODE 6:AF=3:A=10:AL=AF*A-1:XM=XMAX:YM=YMAX:DIM X1(A
L),Y1(AL),X2(AL),Y2(AL),K(2):GOTO 200
```

```
10    GOTO 500:REM ZANDLOPER / F.H. DRUIJFF 2/82
20    Y=YR-1:DOT X,Y KZ:K=1-K:ON K GOTO 40
30    IF SCRN(X+1,Y-1)<>KA GOTO 50:DOT X,Y KA:X=X+1:Y=Y-1:DOT X,Y KZ:GOTO 3
0
40    IF SCRN(X-1,Y-1)=KA THEN DOT X,Y KA:X=X-1:Y=Y-1:DOT X,Y KZ:GOTO 40
50    IF YR>=44 GOTO 70:DOT X,110-Y KA
60    YR=YR+1-SGN(ABS(35-X)):X=35:GOTO 20
70    DRAW 35,YL 35,YL-2 KA:YL=YL-3:IF YL>42 GOTO 60
80    IF GETC=0 GOTO 80
500   CLEAR 5000:MODE 4:KA=0:KZ=10:KL=3:COLORG KA KR KZ KL
510   X=35:Y=65:FILL 24,79 46,103 KZ:READ A,B
520   FILL 25,104 45,105 KZ:DOT 25,105 KA:DOT 45,105 KA
530   READ C,D:IF C=999 GOTO 560:DRAW X+A,Y+B X+C,Y+D KL
540   DRAW X-A,Y-B X-C,Y-D KL:DRAW X+A,Y-B X+C,Y-D KL
550   DRAW X-A,Y+B X-C,Y+D KL:A=C:B=D:GOTO 530
560   Y=68:FOR I=1 TO 10:DRAW X-I,Y+I X+I,Y+I KZ:NEXT
570   IF SCRN(X,Y)=KA THEN DOT X,Y KZ:WAIT TIME 0:Y=Y-1:GOTO 570
580   YR=Y+2:YL=68:GOTO 20
800   DATA 1,0,1,3,12,14,12,60,11,60,11,61
810   DATA 10,61,10,62,0,62,999,0
```

```
5     REM BLOC IN REVERSE F.DRUIJFF
10    MODE 6:COLORG 0 10 10 0:XM=XMAX-1:YM=YMAX-1
15    FOR I=0 TO YMAX STEP 6:DRAW 0,0 XMAX,I 17:DRAW XMAX,0 0,I 17:NEXT
20    X=159:Y=183:S=20:XS=X+S:YS=Y+S:FILL X,Y XS,YS 19
30    H=GETC:IF H=0 GOTO 30
40    IF H<>16 GOTO 50
45    H=GETC:IF H<>0 GOTO 40:IF YS>=YM GOTO 30:DRAW X,YS+1 XS,YS+1 19:DRAW
X,Y XS,Y 18:YS=YS+1:Y=Y+1:GOTO 45
50    IF H<>17 GOTO 60
55    H=GETC:IF H<>0 GOTO 40:IF Y<=1 GOTO 30:DRAW X,Y-1 XS,Y-1 19:DRAW X,YS
 XS,YS 18:YS=YS-1:Y=Y-1:GOTO 55
60    IF H<>18 GOTO 70
65    H=GETC:IF H<>0 GOTO 40:IF X<=1 GOTO 30:DRAW X-1,Y X-1,YS 19:DRAW XS,Y
 XS,YS 18:X=X-1:XS=XS-1:GOTO 65
70    IF H<>19 GOTO 30
75    H=GETC:IF H<>0 GOTO 40:IF XS>=XM GOTO 30:DRAW XS+1,Y XS+1,YS 19:DRAW
X,Y X,YS 18:XS=XS+1:X=X+1:GOTO 75
80    GOTO 30
```

RUN (LINE NUMBER) WITH BASIC V1.0

****

Nearly all DAI-users who have still the V1.0 BASIC ROM's will have undoubtedly been frequently upset after a non recoverable error occurred during program execution.
A re-start will always empty the heap and symbol-table.
After study of the differences between V1.0 and V1.1 , I found the following solution.

1) Type the object code of the short MLP, called RUNLIN in RAM, by using the Substitute feature in UTility.
   Note: The object code is relocatable.
2) Adjust the heap pointer 29B-29C, set pointer after last address object code.
3) Type NEW
4) Load your BASIC program and type RUN.
5) To re-start at a certain line number, first define a variable (integer), which is given the value of the required line number, followed by a call to the MLP.

   EXAMPLE: Assume start address MLP is #300.
            Re-start at line 100

Type the following commands in direct mode:

```
LINE%=100:CALLM #300,LINE%
```

The program will now automatically start at line 100, without emptying the heap and symbol-table.

Finally a few suggestions:

-After the object code is saved on tape, it can together with the BASIC program, easily be loaded with the earlier in DAINAMIC published Bootstrap-loader. Step 2 and 3 can be ommitted in this case
-If frequently is re-started at the same line-number, make the variable part of the BASIC program i.e. 10 LINE%=100 and re-start with only CALLM #300,LINE%.

success

G.GRUITERS

```
*RUNLIN BY G.GRUITERS 16-1-83
*
*THIS MLP IS ONLY 31 BYTES LONG
*AND RELOCATABLE!
*USERS WITH BASIC V1.0 CAN RE-START
*THEIR BASIC PROGRAM  RUN (LINE-NR)
*WITHOUT EMPTY-ING HEAP AND SYMBOL
*TABLE.
*
*       ORG   :300      START ADDR MLP
        INX   H         SKIP 1ST 2 ADDRESSES
        INX   H         OF VARIABLE IN SYMB-TABLE.
        MOV   D,M       NEXT 2 ADDRESSES CONTAIN
        INX   H         INTEGER LINE-NUMBER AND ARE
        MOV   E,M       MOVED IN REGISTER DE.
        XCHG            EXCHANGE DE WITH HL
        CALL  :CAF6     SEARCH LINE-NR IN TEXTBUFF
        MOV   B,H       ADDRESS TEXT LINE IN HL IS
        MOV   C,L       MOVED IN REG. B
        CALL  :E401     RESTORE ROUTINE
        LXI   H,0
        SHLD  :115      RESET TRACE/STEP FLAG
        ZAR
        STA   :126      NO SUSPENDED PROGRAM
        LXI   SP,:F900  RESET STACK-PNTR
        ORA   A         CLEAR FLAGS
        JMP   :C88F     RUN BASIC PROGRAM
        END
```

RUNLIN OBJECT CODE

```
23 23 56 23 5E EB CD F6 CA 44 4D CD 01 E4 21 00
00 22 15 01 AF 32 26 01 31 00 F9 B7 C3 8F C8
```

DAI VIDEO-HARDWARE

(DAInamic 10, page 77-81)

1. MODIFYING THE DAI COMPUTER FROM INTERLACING TO NON-INTERLACING. (2)

A TV picture is made up of 625 lines. These lines are not traced on the screen all at once but in two goes. In other words, firstly the odd numbered lines are traced across the screen and then the even numbered ones. Of the 625 lines only about 550 are visible. Your DAI computer works in the same way but here the information on the odd and even lines is the same. Suppose that the next trace is an odd numbered line; the following trace, an even one, carries the same information as the previous one but will display it a little lower on your screen. The distance between the two traced lines is from 0.5 to 1.0 mm. A line trace lasts 20 mS (50 Hz). Summing up, the picture generated by your DAI flickers somewhat in an up-down fashion, with a periodicity of 2 x 20 mS, i.e. 40 mS, and that is the reason for a rather unsteady picture.

PROPOSED MODIFICATIONS. (3)

(In the diagram the Dutch word Aanbrengen means Provide and Verwijderen means Remove.) Since pin 12 of IC25 is connected to ground (GND) under the chip on the component side of the printed circuit board it is difficult to break the connection. In order to free this pin it is best to snip through it immediately above the print (component side) and bend it up so that a wire can be run from it to pin 14.

(4) The advantage of the foregoing modification will be clear for all to see but there is also a disadvantage. If one is using the 20 mS interrupt (vector 7) in, for example a real time clock, then the clock will run a shade too fast.

f(crystal (ZNA 134)) = 2,562,500 Hz
f(line) = f(crystal) / 164 = 15625 Hz
f(raster) = f(line) / 312.0 (was 312.5) = (50 + 25 / 312) Hz

Thus the 20 mS interrupt now comes every 19.968 mS. Taking it further, the real time clock will gain 138 seconds every 24 hours, assuming that the crystal frequency is as above. (See also section 3 of this letter).

2. MODIFYING DAI's PAL COLOUR BOARD WITH RESPECT TO VIDEO AND COLOUR BANDWIDTH. (5)

I noticed that the test card broadcast by the Dutch TV stations,had a better resolution than the pictures produced by my DAI. The reason for this is that the video bandwidth of the DAI-PAL colour board is only 3MHz (-6dB). Similarly the colour bandwidth is only 0.75 MHz. Why has the DAI company chosen such low bandwidths? Perhaps to overcome interference? Television bandwidths are 5MHz and 1.5MHz respectively. After the modifications given in the following pages the TV interface bandwidths are 7MHz and 2.5MHz respectively. This produces an enormous improvement. Since doing the modification I have had no trouble from interference or similar problems. With some TV sets however there could be. Mine is a Philips with the KT3 chassis, 43cm 90 degree picture tube. The use of a 90 degree tube makes this set ideal for a DAI monitor as the improved focussing possibilities of the tube are better than needed for the original 5MHz bandwidth.

(6) Proposed modification of the DAI-PAL colour board (brightness).
(7) Proposed modification of the DAI=PAL colour board (colour).

The Dutch words in the two diagrams, (6) and (7) can be translated as follows:- wordt = becomes, verwijderen = remove, spoel = coil, en doorverbinden = and connect through, instel = instal (or fit). (trim, adjust)

3. ADJUSTING THE TV LINE FREQUENCY (15625 Hz). (8)
The frequency generated by the DAI was about 5MHz too low. This is not really serious for normal computer use, unless one attaches much importance to the accuracy of the 20 mS interrupt (V7). If the modifications described in section 1 have been done then the error is more serious. As I am a radio and television service amateur I attach great value to the accuracy of the line frequency, for the regulation and checking of the colour demodulator. In order to adjust the line frequency in your DAI a small modification has to be made, namely the replacement of a small capacitor by an adjustable one. (See page 9). Anyone who does not use his DAI as a test pattern generator has no need to bother with the alterations described in this section.

MODIFICATION AND ADJUSTMENT OF TV LINE FREQUENCY. (9)
(The note beneath the diagram reads :-) Replace capacitor C by a 50pF trimmer capacitor. Using the trimmer the frequency can be adjusted to 1,281,250.0 Hz. Connect the frequency counter to pin 10 of IC25 (ZNA 134).

4. ADJUSTMENT AND MODIFICATION OF THE COLOUR CARRIER WAVE. (10)
Using the DAI as a test pattern generator soon taught me that the carrier frequency was so high that some TV sets could not tolerate it, resulting in colour reproduction being over-coupled. To lower the frequency and correct it a 22pF trimmer capacitor is fitted in parallel with the crystal on the modulator board. The frequency must be very precisely adjusted and because my frequency counter was not accurate enough I had to devise another method of measurement - which was to equate the frequency with that of a TV transmitter by means of Lissajous patterns on an oscilloscope. One channel (of the scope) is connected to the input of the DAI's HF modulator (the small tin box on the modulator board, pin 4) while the other channel is connected to the output of the colour reference generator of the TV set. Type on the DAI : 100 COLORG 1 0 0 0:MODE 5:GOTO 100 followed by RUN.

(11). Tune your TV to a transmitter and check that the test card is still displayed in colour. Let the computer and the TV warm up for half an hour. Then adjust the newly fitted trimmer until the Lissajous pattern on the scope is stationary.

5. ADJUSTMENT OF THE SOUND CARRIER. (12)
In the same way, the frequency of the sound carrier is important when the DAI is used as a test pattern generator. A sharpened matchstick is needed to do the adjustment. Connect the frequency counter to pin 1 of the HF modulator. In the lid of the modulator are two small holes. The one you want is that nearest the side holding the connector. Adjust the frequency to 5,500 KHz. During this have 'SOUND OFF'. Adjustment was not necessary on my DAI.

6. PROGRAM 'TEST CARD'. (13)
Because, as you know, I am a service amateur I had a need for my own test pattern generator. This was the chief reason for modifying various parts of my DAI computer (see the previous sections). An example of an inconvenient picture is the test card of the Dutch transmitters. The real problem is that the DAI Basic V1.1 in graphic modes could not cope with the unseen parts, the most important features of a test card. That is why there is so much POKEing to the screen in the program, listed on pages 103 and 104. The circle has turned out a bit smaller because this time I wanted to make use of the basic support. This program is not expected to be suitable for all sorts of tuners, but it is all right in my practice where I want the same picture every time. Here is a short description of the program:-
Lines 100-999 the short main program
Lines 1000-1060 initialise the screen memory so as to push the screen up a little and add a bit below it.
Lines 2000-2804 place the white blocks around the frame of the picture (most are only partially

visible)
Lines 3000-3070 divide the picture into small rectangles.
Lines 4000-4050 draw a circle with short dashes. It is not used in the main program. When needed it is called by a GOSUB 4000 in line 210; change GOSUB 5000 to GOSUB 4000.
Lines 5000-5960 fill the circle in the middle of the test card. The circle is organised from top to bottom as follows:-
-red/yellow bar to check the delay line.
-black reflection bar to check for reflections and side lobes in the intermediate frequency band.
-graded grey bar for linearity check of the final video amplifier.
-colour bars to check the colour reproduction.
-there is a cross in the centre of the circle for convergence.
-frequency bars for checking the frequency characteristics of the IF section and video final amplifier(s).
-graded grey bar.
-yellow/red bar to check the delay line.
At the end of the program is my own version of SGT (slow graphic text). It is for the most part taken from the DAI handbook. Some precautions had to be anticipated and modified so that the text string could run separately from the picture. In addition the program has been extended to include an option for drawing text at various angles, in contrast to other versions of graphic text where this is only possible with angles which are multiples of 90 degrees. SGT can be called with a GOSUB 40000. In the main program one must insert an extra CLEAR 1400 for this subroutine. Variable A$ is the text to be printed. X and Y are the starting point co-ordinates. Variable DEV is the angle at which the string is to be printed, radially anti-clockwise. C is the colour code and F the enlargement factor for drawing the text.

DAInamic CATALOG 1-4-83

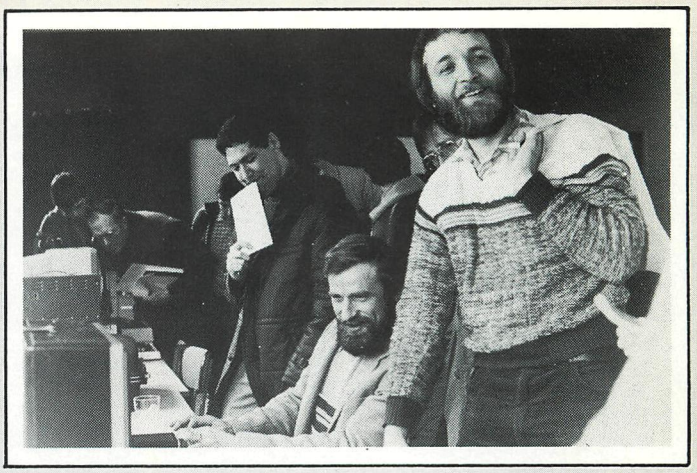
CODE	TITLE	AUDIO	DCR
G1	GAMES COLLECTION 1	400	550
G2	GAMES COLLECTION 2	400	550
G3	GAMES COLLECTION 3	400	550
G4	GAMES COLLECTION 4	800	950
G5	GAMES COLLECTION 5	400	550
G6	GAMES COLLECTION 6	750	900
G7	GAMES COLLECTION 7	750	900
G8	GAMES COLLECTION 8	750	900
G9	GAMES COLLECTION 9	750	900
G10	GAMES COLLECTION 10	750	900
G11	GAMES COLLECTION 11	750	900
DNA	DNA ASSEMBLY PACK	1100	1250
FGT	FAST GRAF TEXT	1000	1150
FGTA	FGT-APPLICATIONS	1000	1150
TK1	TOOLKIT 1	1000	1150
TK2	TOOLKIT 2	1000	1150
TK3	TOOLKIT 3	1000	1150
TK4	TOOLKIT 4	1000	1150
PE1	PRIMARY EDUCATION 1	1000	1150
SE1	SECUNDARY EDUCATION 1	1000	1150
SE2	SECUNDARY EDUCATION 2	1000	1150
WP	WORD PROCESSOR I	1000	1150
ML	MAILING LIST	1000	1150
GT	GRAPHIC TABLET	1000	1150
M1	MUSIC COLLECTION 1	300	450
M2	MUSIC COLLECTION 2	300	450
M3	MUSIC COLLECTION 3	300	450
DTP	DAI TINY PASCAL	1000	1150
EGT	ENGLISH-GERMAN TRAINER	1000	1150
DD	DAI DEMO + BASIC TUTOR	500	650
CH	SARGON CHESS	1500	1650
SI	SPACE INVADERS I	800	950
T80	TAPE 80-81	850	1000
N10	NEWSLETTER 10	500	650
N11	NEWSLETTER 11-12	650	800
N13	NEWSLETTER 13-14-15	650	800
CTP	CENTIPEDE	600	750
DRI	DRIVER	600	750
SUI	SUPER INVADER	600	750
DTX	DAINATEXT	2000	2150
DAPA	DAIPANIC	800	950
JR	MICRO'S-ONDERWIJS	990	1100
SPL	SPL MACRO-ASSEMBLER	1100	1250
W3	WISKUNDE 3	750	900
TT1	TAAL 1	750	900
F1	FYSICA 1	750	900
FB	FAMILIEBUDGET	500	650
GH	GRAFISCHE HULP	500	650
ACR	ACROBATES	600	750

HARDWARE & PUBLICATIONS

PCS	DAIpc SCHEMATICS	850
BOD	BEST of DAInamic (80-81)	500
SNG	SUPER NOISE GENERATOR	1500
NC	NEW CHARACTER GENERATOR	1000
DCE	DCE-INTERFACE-CARDS	2500
N8	NEWSLETTERS 8-13 (1982)	500

DAInamic meeting 9th april 1983

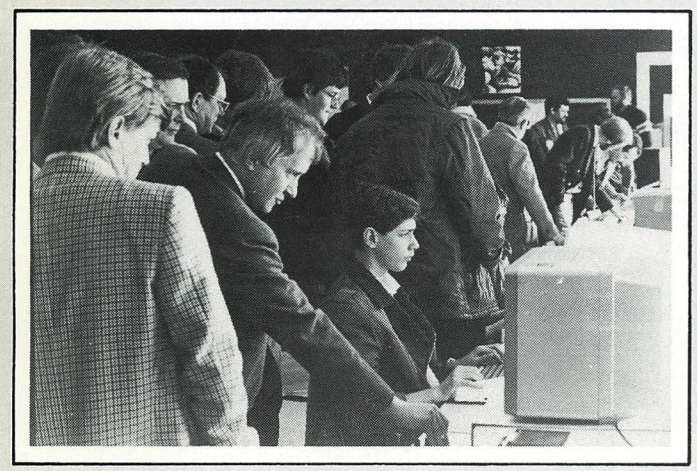
3-D illusion



mode 7 & 8, SFGT ...

DAInamic library

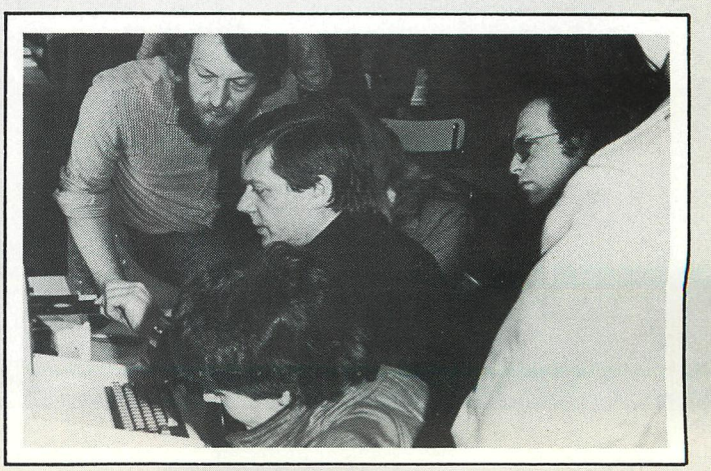
diDAIsoft



a lot of software ...

DAInamic software

looking for peripherals



INDATA : present !


QAI