graph			
graph	82/90	4-Takt motor demo	Dierckx
graph	82/82	Beer	Druijff
graph	82/272	Bird	Wittendal
graph	82/123	Change 16 colour mode in 4 colour mode	de Winter
graph	82/193	Colours 16-23 : demo	v.d. Hoeven
graph	82/272	Dessin; drawing 16 colours	Wittendal
graph	82/107	Discharging electric energy	Smit
graph	82/81	Drawing with paddles	Rens
graph	82/35	Driehoeksvormen	de Bont
graph	82/211	Easy background colour in mode 0	—
graph	82/331	FGT : demo program	v. Rompaey
graph	82/247	Falling dot	Druijff
graph	82/262	Fascination : graphic demo	DAIclub France
graph	82/247	Filling screen with dots	Druijff
graph	82/273	Frites	Rens
graph	82/175	Graphics demo	Jaeckle/Druijff
graph	82/306	Graphics demo	Poels
graph	82/319	Graphics demo	Dierckx
graph	82/21	Graphics demo program	Sigg
graph	82/194	Holland in mode 2	Delvoye
graph	82/112	Kaleidoscope	Meystre
graph	82/82	Large size characters	Druijff
graph	82/267	Lines : demo	Druijff
graph	82/38	Luxaflex	Berkx
graph	82/110	Mandala	—
graph	82/85	Patchwork	Druijff
graph	82/247	Polygons	Druijff
graph	82/226	Rectangles	Druijff
graph	82/358	Rotating screen	Poels
graph	82/242	Screen driver demo	Looije
graph	82/122	Snoopy	de Bont
graph	82/85	Spot on text	Druijff
graph	82/103	Testbeeld	Doornenbal
graph	82/225	Tree of Pythagoras	Regtien
graph	82/85	Tumbling lines	Druijff
hardw			
hardw	82/111	16 Colours in mode 0	—
hardw	82/305	DAI schematics : errata	Boerrigter
hardw	82/9	DCE bus : Connections	Schepens
hardw	82/111	Picture stabilisation	—
hardw	82/148	RS232	Moeys
hardw	82/39	Schematics : Sound + noise	de Jong
hardw	82/360	Super noise generator	Rens/de Winter
hardw	82/77	Video hardware : modificaties	Doornenbal
libr			
libr	82/27	Catalog	—
libr	82/295	Catalog	—
libr	82/317	Catalog	—
libr	82/167	DAI schematics	—
libr	82/204	G8 - G9 - TK3	—
libr	82/101	Tape 1980-81	—
pasc			
pasc	82/146	Decode	Dierckx
pasc	82/140	Description	Nijland/v. Eck
pasc	82/146	Dragon curve	Dierckx

pasc	82/147	Library routines	Nijland/v. Eck
pasc	82/144	Prime numbers	Nijland/v. Eck
pasc	82/145	Towers of Hanoi	Nijland/v. Eck
perip			
perip	82/76	DCE-connection OKI microline printer	Pekkeriet
perip	82/344	EPROM-programmer	Knoops
perip	82/148	EPSON MX-82 : hidden features	Moeys
perip	82/10	Serial-parallel printer interface	De Dauw
progr			
progr	82/11	BASIC : FOR-NEXT loop delays	Hard/Rens
progr	82/13	BASIC : GETC use	Rens
progr	82/120	BASIC : INT/FPT variables	Druijff
progr	82/301	COLORG : programming techniques	Druijff
progr	82/91	Combination BASIC and MLP	Boerrigter
progr	82/289	Conversion Apple/Atari/DAI	Verstegen
progr	82/48	Crash : what to do after it	Sip
progr	82/268	Efficiency in circle drawing	Druijff
progr	82/74	Inzenden van programma's	Druijff
progr	82/224	Loop problems	Boerrigter
progr	82/89	ML routines : several useful ones	Sip
progr	82/40	ML programs : where to store it	Boerrigter
progr	82/281	Output RS232 only - UT Read	—
progr	82/121	Programming hints	Druijff
progr	82/173	Programming techniques	Druijff
progr	82/88	RS232 output only	—
progr	82/12	Video RAM : Use	Hard/Rens
read			
read	82/37	8080 Publications	Interface Age
read	82/46	BASIC commands : references	Verwimp
read	82/17	Magazines	Schepens
read	82/308	Microcomputers in het onderwijs	Berckx
rem			
rem	82/264	BASIC : How to extend it	Boerrigter
rem	82/113	Benchmarks : results	Coughlan
rem	82/336	Benchmarks : results	PCW
rem	82/114	DAI op de piste	Vollinga
rem	82/202	HCC DAI-gg	Nijland
rem	82/119	IPROM	Werbeck
rem	82/102	The story of Anna List	Hunt
sound			
sound	82/115	Birthday song	Hunt
sound	82/124	More about talk and music	—
teach			
teach	82/50	BASIC tutor	Hermans
teach	82/193	BASIC tutor : updates	Hermans
teach	82/75	DIDAISOFT : doelstelling	v. Rompaey
teach	82/236	Demo logical functions	Hermans
teach	82/341	Demo logical functions : modification	Stokhorst
teach	82/233	Map of Belgium	Moens
teach	82/116	Travelling & standing waves	Roelants
tool			
tool	82/266	Audio cassette marker	Hermans
tool	82/160	Cassette list	Arts
tool	82/48	Cassette list	Sip
tool	82/292	Cassette/MDCR array saver/loader	Broekman
tool	82/210	DAI as terminal	Gruiters

read	81/138	8080A/8085 assembly language programming	Leventhal/Broekman
read	81/17	Bench test DAI	PCW
read	81/61	Colour graphics	Educational Computing 3/83
read	81/27	Magazine news	Schepens
read	81/185	Patterns	PCW/Sutcliffe
rem			
rem	81/139	Computers and education	v.d. Hijden
rem	81/192	Course «Micro-electronica in bedrijf»	Teleac
rem	81/122	DIDACOM : Project «motorieke training»	Broekman
rem	81/157	List DAInamic members	—
rem	81/244	Towards the firmware manual	Hermans
sound			
sound	81/12	Blue moon	—
sound	81/118	Fasing keyboard music	—
sound	81/170	Funny sirene	—
sound	81/119	Random noise	—
sound	81/22	Tubular bells	Sip
sound	81/151	Wilhelmus	—
teach			
teach	81/180	8080 simulator	Hermans
teach	81/242	Complements & supplements	Spica
teach	81/121	DIDACOM : Datastatements generator	Broekman
teach	81/214	Results table for teachers	v. Dijk
tool			
tool	81/134	Accidently «reset» : What to do	Assink
tool	81/244	Accidently «reset» : comments	Boerrigter
tool	81/176	An attractive cursor	—
tool	81/219	Basicode I	v. Lieshout
tool	81/236	Clear screen in software	Pommerell
tool	81/36	Coloured backgrounds in mode 0	—
tool	81/42	Display routine for all characters	—
tool	81/170	Endless continuation lines	—
tool	81/135	FGT : Loading object FGT in BASIC program	Assink
tool	81/46	Flashing in 4 colour mode	—
tool	81/154	Inverse colour background	—
tool	81/240	MLP : use of paddle routine	—
tool	81/101	Screen copy for Epson MX-80 (mode 1-4)	—
tool	81/234	Screen copy via DCE-bus on Epson MX-80 II	Hoffmann
tool	81/97	Start/stop via Event and Interrupt 7	Schepens
tool	81/211	Talk editor	Sip/Hermans
tool	81/232	Telex in Basic	de Dauw
tool	81/110	Time counter #01BE : Users routines	—
tool	81/117	Title for program collection on tape	—
tool	81/30	Variable atlas	—
tool	81/155	Video RAM : Mode 0 memory map	Kop
tool	81/224	Video RAM : Mode 4 memory map	Hermans
use			
use	81/173	Backgammon/Barricade/Morass/Snake/Poisoned cake	—
use	81/218	Basicode I	v. Lieshout
use	81/169	Cannon game/Othello/Star trek	—
use	81/75	FGT Table creator	—
use	81/133	FGT : BASIC call of FGT	Assink
use	81/67	FGT : Users guide	—
use	81/175	Invaders/Robots/Traffic test/Life	—
use	81/172	Mastermind/Goat game/Space invaders/Hannibal/Tower	—
use	81/170	Reaction test/Lunar landing	—

use	81/177	Secundary Education 1	—
use	81/171	Surround/Break-out/Amazing/Kim cache	—
use	81/168	Yathzee/Awari/Submarine/Cannon game	—
ut			
ut	81/223	Digital clock	v. Lieshout
ut	81/29	Digital/analog clock - variable table	v. Cooten
ut	81/80	Film title simulation	—
ut	81/239	Letter	Marchand

DAInamic 1982

CODE	PAGE	SUBJECT	AUTHOR
calc			
calc	82/258	Calender algorithmes	Lelie
calc	82/33	Conversions degrees/money/numbers	de Lombaert
calc	82/159	Faculteit	Dierckx
calc	82/112	Prime numbers	Meystre
disk			
disk	82/341	FGT : peek-poke	Gesp
firmw			
firmw	82/209	BASIC V1.0 - V1.1 : Modifications	Schepens
firmw	82/202	BASIC commands IMP INT-STR	Couplet
firmw	82/41	BASIC : SAVEA/LOADA stringarrays	Boerrigter
firmw	82/196	BASIC : internal representation	Raedt/Bellekens
firmw	82/105	Conversion ASCII/binary/ASCII	Jongen
firmw	82/203	EDIT-buffer : How it works	Boerrigter
firmw	82/93	Entry-points : useful ones	de Raedt
firmw	82/14	Entry-points : the most useful ones	de Raedt
firmw	82/129	FPT numbers : representation	v. Rompaey
firmw	82/40	Firmware locations	Boerrigter
firmw	82/97	Math. package & ROM interface	Jongen/de Raedt
firmw	82/7	Math. package : entry points	Jongen
firmw	82/180	Memory map. V4.4	Boerrigter
firmw	82/248	Video RAM : Set-up	Looije
firmw	82/255	Video RAM : screen constants	Boerrigter
firmw	82/326	Video RAM : screen control bytes	Looije
firmw	82/256	Video RAM : screen memory management	Boerrigter
game			
game	82/136	Bingo	Groeneveld
game	82/47	Carpenter mystery : solution	Hermans
game	82/238	Ellen	Smet
game	82/138	Grue + chrono	Moens
game	82/38	Lichtpen spel	Berkx
game	82/338	Multi-puzzle	Poels
game	82/88	Space invaders : modification	—
game	82/278	TV-tennis	Maes

DAInamic 1981

CODE	PAGE	SUBJECT	AUTHOR
calc			
calc	81/156	Graphics of 5-grade polynomes	v. Dijk
calc	81/47	Prime numbers	div.
calc	81/116	Statistics / random	Meystre
firmw			
firmw	81/206	BASIC command IMP STR : faulty	Krebs
firmw	81/10	BASIC commands SAVEA / LOADA	—
firmw	81/69	BASIC : Textbuffer and symbol table	—
firmw	81/190	INT number representation	v. Rompaey
firmw	81/141	Memory map V3.8	Boerrigter
firmw	81/244	Memory map : updates	Boerrigter
firmw	81/123	Power-on initialisation	Boerrigter
firmw	81/120	ROM entry points : SDOT/SDRAW/SFILL/SSCRN	—
firmw	81/45	ROM entry points : SMODE	—
firmw	81/43	ROM entry points : introduction	—
firmw	81/243	Restart routines	Boerrigter
firmw	81/188	Use of the heap	Boerrigter
firmw	81/16	Video RAM : 3 not used locations on left side	—
game			
game	81/35	Barricade	Druijff
game	81/217	Carpenters mystery	Hermans
game	81/78	FGT Word game	—
game	81/109	FGT : Drawing with paddles	—
game	81/152	Towers of Hanoi	Meystre
sound			
sound	80/1/27	Frequency range music instruments	v. Bussel
sound	80/0/8	Music : Frequencies	—
sound	80/2/6	Partiture on printer	—
sound	80/1/26	Several useful tones	—
tool			
tool	80/2/16	BASIC : Chaining of programs	—
tool	80/1/19	Cassette interface : test	—
tool	80/1/21	Cassette interface : writing from CAS 1 to CAS 2	v. Moerkom
tool	80/2/17	Format listing	—
tool	80/2/28	Format listing	Dessart
tool	80/0/11	Video RAM : Finding locations	—
use			
use	80/2/25	Disassembler : Users guide	De Raedt
use	80/2/8	Graphic Tablet : Users guide	—
ut			
ut	80/2/14	Digital / analog clock	v. Cooten
ut	80/1/22	Number conversions to other base	Verdonk
ut	80/2/13	Real time clock	—

graph			
graph	81/205	16 Colours in 4 colour mode	Mol
graph	81/232	Crane	Dierckx
graph	81/25	Demo video text	v. Cooten
graph	81/241	Graphic demo	—
graph	81/42	Lace cloth	Delvoye
graph	81/237	Poke action	Hermans
graph	81/24	Radar simulation	—
graph	81/66	Rotating ellipses	Beumers
graph	81/239	Sientje	Verwimp
graph	81/236	The hat	Corswandt
graph	81/23	Torch	—
hardw			
hardw	81/102	AMD9511 : Data sheets	AMD
hardw	81/89	AMD9511 : Evaluation of timing	Schepens
hardw	81/240	Acoustic signal at end LOAD/SAVE	Wanders
hardw	81/128	Cassette interface : Clippers	div.
hardw	81/228	Cassette interface : modifications	de Jong
hardw	81/231	Cassette interface : modifications	v. Lieshout
hardw	81/83	DAI connected to video monitor	Bakker
hardw	81/208	Hardware solution for CHR$(12)	Dessart
hardw	81/127	Schematics : RS232 interface	—
hardw	81/45	Schematics : Cassette interface	—
hardw	81/207	Schematics : Lay-out main board	Dessart
hardw	81/230	Schematics : Noise/random generator	de Jong
hardw	81/230	Schematics : Paddle interface	de Jong
hardw	81/41	TV : Modification if no colour	Verberkt
hardw	81/229	TV : Sound from cassette to TV	de Jong
hardw	81/115	X-bus : Connector lay-out	—
libr			
libr	81/38	Catalog	—
libr	81/88	Catalog	—
libr	81/166	Catalog	—
libr	81/227	English-German grammar teacher	—
libr	81/167	Game paddles	—
libr	81/176	Games collection 4	—
libr	81/114	Toolkit 1	—
periph			
periph	81/112	Cassette recorder Philips 2N2235	—
periph	81/66	Converter RS232 to 20 mA	—
periph	81/212	Epson MX-80 : Specifications/DIP-switch settings	—
periph	81/41	Epson TX-80 : Ribbon replacement	—
periph	81/233	Interface DAI/serial to Epson/parallel	de Dauw
periph	81/237	MDCR : Comments on use	v. Cooten
periph	81/114	MDCR : Info	Memocom
periph	81/32	Printer : Centronics 702/703	Cattaert/de Dauw
periph	81/233	RTTY convertor	de Dauw
periph	81/194	Speech synthesizers : Info + data sheets	Schepens
periph	81/241	TV : Video interface for Teletekst TV	Wanders
progr			
progr	81/206	BASIC : Where to place subroutines	—
progr	81/8	MLP : Programming in machine language	—
progr	81/31	Symbol table : Cleaning up	—
progr	81/11	Variables : Standardisation in names	—
read			
read	81/205	8080/8085 software design	Titus e.a./Broekman

BAUDRATE IN UTILITY

Do you use a printer with another baudrate than the 9600 baud the DAI defaults to? Did you ever try to make a print out of a 'LOOK' sequence?
It did not work although you are sure you did change the TIC baudrate register #FFF5 to the correct value?
This article explains why it did not work, and how to overcome this problem.

Before you do a 'GO' or a 'LOOK' under control of the Utility monitor, it is obliged to give a 'Z2' or 'Z3' command in advance.
Without this Z-command, the result may be a big mess on your screen.

Amongst other functions, the Z2/Z3 command initialises the Timer Interrupt Controller (TIC - 5501) and the Parallel interface chip 8255 (GIC).
See ROM-bank 3, address #ECBA, for more details.

By this Z-command, the baudrate register of the TIC is set to 9600 baud by storing the relevant value in RAM address #0060.
As soon as LOOK or a GO-command is given, this value is taken from address #0060, and stored in the TIC control register #FFF5. (see #3ED9C).

If you want another baud-rate, the only thing you have to do is changing the contents of address #0060. Each time a LOOK or a GO is performed, this value is taken and stored in the baudrate register #FFF5.

The following possibilities exist:

9600 baud: #FC	300 baud: #BC
4800 baud: #EC	150 baud: #AC
2400 baud: #DC	110 baud: #9C
1200 baud: #CC	

To change the default baudrate, just perform once after power-on or after a reset:

```
*UT
>Z3
>S0060 FC-**
```

in which '**' is one of the values given above.

(C) - Jan Boerrigter - Nov. 1983.

DAInamic 1980

```
1     REM ------------------------------------------------
2     REM *      CALCUL D'INTERET COMPOSE          *
5     REM *      Auteur: Ch. COLLET                *
6     REM ------------------------------------------------
8     PRINT CHR$(12):PRINT :PRINT :PRINT :PRINT :PRINT :PRINT :PRINT :PRINT
9     PRINT "             CALCUL D'INTERET COMPOSE":WAIT TIME 150:PRINT CHR$(12)
10    PRINT CHR$(12):PRINT "                  - MENU -":PRINT :PRINT
15    PRINT " Desirez-vous calculer        "
20    PRINT "                - la valeur acquise de votre capital ? :(1)"
25    PRINT "                - le taux d'interet compose          ? :(2)"
26    PRINT " Desirez-vous   - un complement d'explications       ? :(3)"
27    PRINT "                - arreter                            ? :(4) "
30    INPUT R!
35    ON R! GOTO 99,990,1600,9999
50    GOTO 9900
99    PRINT CHR$(12)
100   INPUT "Capital initial  ";C!
101   Z!=1.0
150   PRINT
200   INPUT "Duree en MOIS de la periode de capitalisation ";D!
250   PRINT
300   INPUT "Nombre de periodes de capitalisation ";N!
350   PRINT
400   INPUT "Taux d'interet annuel exprime en pourcent ";T!
450   PRINT :PRINT
500   V=C!*(1.0+(1E-2*T!*D!)/12.0)^N!
600   PRINT :PRINT " Valeur acquise  : ";V
630   PRINT :PRINT
640   PRINT "             Pour continuer, taper C"
641   PRINT "             Pour retourner au MENU,taper M"
642   PRINT "             Taper autre chose pour arreter"
645   A=GETC:IF A=0 THEN 645:REM BOUCLE D'ATTENTE
650   IF A=67 THEN 1800
655   IF A=77 THEN 10
660   GOTO 9999
990   PRINT CHR$(12)
991   Z!=2.0
1000  INPUT "Capital final : ";V!
1001  PRINT
1100  INPUT "Capital initial :";C!
1101  PRINT
1200  INPUT "Duree en mois de la periode de capitalisation :";D!
1201  PRINT
1300  INPUT " Nombre de periodes de capitalisation :";N!
1301  PRINT
1350  T!=(((V!/C!)^(1.0/N!))*12.0-12.0)/(1E-2*D!)
1400  PRINT :PRINT "       Taux d'interet compose annuel :";T!;"  pourcent"
1500  PRINT :PRINT :GOTO 640
1600  PRINT CHR$(12):PRINT :PRINT
1605  PRINT "Le programme permet:":PRINT :PRINT
1610  PRINT "         - (1) de calculer la valeur acquise V d'un"
1620  PRINT "               capital initial C, place a un taux"
1630  PRINT "               annuel T d'interet compose, le place-"
1640  PRINT "               ment se faisant pendant N periodes"
1650  PRINT "               de capitalisation, chacune de ces"
1660  PRINT "               periodes etant de D mois."
1670  PRINT
1680  PRINT "         - (2) de determiner le taux annuel T de"
1690  PRINT "               placement, connaissant le capital"
1700  PRINT "               initial C, le capital final ou valeur"
1710  PRINT "               acquise V, le nombre N et la duree"
1720  PRINT "               D des periodes de capitalisation."
1730  PRINT :PRINT :PRINT :PRINT "                          Pour continuer, presser une touche"
1740  B=GETC:IF B=0 THEN 1740:REM BOUCLE D'ATTENTE
```


CONT on p.22

PROBLEMEN MET PRINTING ? ? deel II

Heeft U ook al vaak gehad dat U wilde beginnen met iets op de printer en omdat U nog niet klaar was de printer afgeschakeld had met POKE #131,1 ? Zo begon vorig blad een van mijn bijdragen. Velen zullen gereageerd hebben met de verzuchting: "Nou vroeger niet zo veel, maar sinds dat artikel vrijwel constant." Hoe ik het voor elkaar gekregen heb is mij nog steeds niet duidelijk , maar ik denk voorlopig het record ' grootste aantal fouten in zo min mogelijk regels ' stevig in handen te hebben. Ik schreef dat de routine tien bytes lang is en ik gaf U slechts negen bytes. Heeft U enige ervaring in machinetaal is het U misschien opgevallen dat er wel een F5 maar niet een F1 bij stond . Er wordt dus wel op de stack ge'pushed' maar de bijbehorende 'pop' ontbrak. Hierdoor zal de stack dus snel volraken en krijgt U 'STACK OVERFLOW'. Deze fout verbeteren zal nog steeds niet de gewenste routine opleveren daar aan het eind van de routine gesprongen wordt naar adres #6CFD en dat zal kenners al evenzeer verbazen daar dit adres in de RAM's ligt. Wat er dan ook zal gebeuren is niet te voorspellen en normaal in geen geval hetgene zijn dat we wilden. In de laatste byte van de routine zijn namelijk de nibbles verwisseld . En of dit nog niet erg genoeg is gaf ik U in III de raad na V5 de spacebar in te drukken zodat U niet vector 5 maar vector 6 ging veranderen. De juiste routine staat nu hieronder.

```
I       UT [RETURN]
II      S260 [SPACE] F5 3E 01 32 31 01 F1 C3 FD C6 [CURSOR LEFT]
III     V5 C6FD-260 [CURSOR LEFT]
           ^^^^ (dit geeft de DAI !)
```

Pas op !! De routine zoals hierboven vermeld kan alleen gebruikt worden als U niet de EDIT of de floppy gebruikt. Bij EDIT zal #131 namelijk 2 bevatten en dat zal door deze routine verhinderd worden. Er is gelukkig een oplossing voor maar de routine is dan wel groter en dus trager . Elke interrupt-afhandeling , die wij onderbreken zal de snelheid van het programma beinvloeden ; dus dienen wij deze onderbrekingen zo kort mogelijk te houden. Vandaar onder advies de nu volgende routine alleen te gebruiken bij programma's, die snelheid niet zo erg nodig hebben en het wel nodig hebben deze routine erbij te gebruiken.

```
I       UT [RETURN]
II      S257 [SPACE] F5 21 31 01 7E FE 02 F2 63 02 36 01 F1 C3 FD C6
            [CURSOR LEFT]
III     V5 C6FD-257 [CURSOR LEFT]
           ^^^^ (dit geeft de DAI !)
```

Bovenstaande routine is niet mijn oorspronkelijke versie. Vlak voor publicatie kwam Nico Looije langs en die wist een nog iets slimmere oplossing (kleiner en sneller) dan ik gevonden had. En ere wie ere toekomt. U ziet toch wel hoop ik, dat de routine nu op een ander adres start ? Tevens is de routine nu niet meer zonder meer op een andere plaats te zetten. De aanroep is wel hetzelfde gebleven; dus POKE #262,0 of POKE #262,1 . Weer eindig ik met de wens velen hiermee een plezier te hebben gedaan.

Frank H. Druijff

DELETE /2

MONS,LE 02/01/83

AUX AMIS DU DAI,

PERMETTEZ-MOI D'APPORTER UN MOT D'EXPLICATION AU SUJET DE L'ARTICLE DE JOS VANDEBERGH,PARU DANS LE DAINAMIC 19 (P. 413-414).

EN FAIT,L'EXPLICATION DU 'PHENOMENE' EST ASSEZ SIMPLE.
EN GROS,NOUS POUVONS DIRE QUE L'EXECUTION DU SECOND 'EDIT' 'COURT-CIRCUITE' LA FIN NORMALE DE L'EXECUTION DU PREMIER 'EDIT'.

PRENONS UN EXEMPLE SIMPLE,SUPPOSONS LE PROGRAMME SUIVANT :

```
10 REM MYSTERE DU 'DELETE'
20 REM -- LIGNES
30 REM -- A
40 REM -- SUPPRIMER
50 REM FIN DU MYSTERE
60 REM ---------------
```

ENTRONS LA COMMANDE SUIVANTE : EDIT20-40:EDIT10
QUE SE PASSE-T-IL ? POUR CE,REPORTONS-NOUS A L'ARTICLE 'EDITOR STORY' (DAINAMIC 17 - P. 228-231).

1. EXECUTION DU PREMIER 'EDIT' (EDIT20-40) :

L'EXECUTION SE DEROULE SELON LE SCHEMA DECRIT AUX PAGES 228 A 230
AU RETOUR DE L'EXECUTION DE LA COMMANDE (#E284),NOUS AVONS EN MEMOIRE

	Editbuffer	Textbuffer	Symtab	Free space
LIGNES:	20 - 30 - 40	10 - 50 - 60		

SANS NOUVELLE INSTRUCTION 'EDIT',ON EXECUTE CE QUI EST DECRIT EN P.231 DU DAINAMIC 17 (LE CONTENU DE L'EDITBUFFER EST REPLACE DANS LA BONNE SEQUENCE DANS LE TEXTBUFFER).
MAIS COMME IL EXISTE UNE NOUVELLE COMMANDE 'EDIT',LE MONITEUR EN #C882 NE BRANCHE PAS EN #C8E5 MAIS CONTINUE EN SEQUENCE (INSTRUCTIONS #C885 ET SUIVANTES),EN FAIT IL EXECUTE LA SECONDE COMMANDE ('EDIT10') (EN #C88C).

2. EXECUTION DU SECOND 'EDIT' (EDIT10) :

EXECUTION NORMALE,A PART LE FAIT QUE LE TEXTBUFFER A CE MOMENT NE CONTIENT PLUS QUE LES LIGNES 10,50 ET 60 !
AU RETOUR DE L'EXECUTION (#E24B) DE CETTE COMMANDE,NOUS AVONS EN MEMOIRE :

	Editbuffer	Textbuffer	Symtab	Free space
LIGNES:	10	50 - 60		

LES LIGNES 20 A 40 SONT DONC DEFINITIVEMENT PERDUES !
A CE MOMENT,COMME IL N'Y A PLUS D'AUTRE INSTRUCTION,ON BRANCHE EN #C823 (FIN NORMALE) ET ON EXECUTE CE QUI EST DECRIT EN PAGE 231 DU DAINAMIC 17.

VOILA,EN EFFET,UNE COMMANDE 'DELETE' INFAILLIBLE !!

JOEL MENIER.

* * * Wedstrijd * * *

We willen een programmadeel dat in elk programma ingevoegd kan worden en dat hetzelfde doet als onderstaand programmadeel.

```
300 H=GETC
310 IF H=0 GOTO 400
320 IF H=16 THEN Y=Y+1:GOTO 400
330 IF H=17 THEN Y=Y-1:GOTO 400
 :
 :
 :
390 IF H=23 THEN X=X+8:GOTO 400
400 REM normaal vervolg van programma
```

We willen dat dit programmadeel geen GOTO's bevat, omdat die de verwerkingstijd afhankelijk maken van de plaats in het programma. Verkapte verplaatsinstructies zoals IF...THEN linenr en GOSUB en ON... zijn dus ook niet toegestaan. We willen geen uitstapjes naar machinetaal, maar weet U een leuke oplossing met gebruikmaking van de ROM's willen we die graag (buiten mededinging !!) ontvangen.

Het moet na doorgang X of Y veranderd hebben als een der cursortoetsen (eventueel met SHIFT) werd ingedrukt. Bij niet indrukken of indrukken van een andere toets moeten X en Y gelijk blijven. De verandering moet met alleen indrukken van de cursortoets 1 zijn en met tegelijk indrukken van cursortoets en SHIFT een door de gebruiker te bepalen integerconstante.

De inzendingen worden beoordeeld naar originaliteit en ook kijken we naar de volgende criteria: Het programmadeel dient zo min mogelijk regels te beslaan, liefst maar een regel. Het programmadeel dient zo zuinig mogelijk om te springen met geheugenruimte. (heap, symbol-table en initialisatie voor de juiste werking worden dan ook bekeken.)

De beste inzendingen krijgen een vermelding in DAInamic en een beloning in de vorm van software naar keuze.

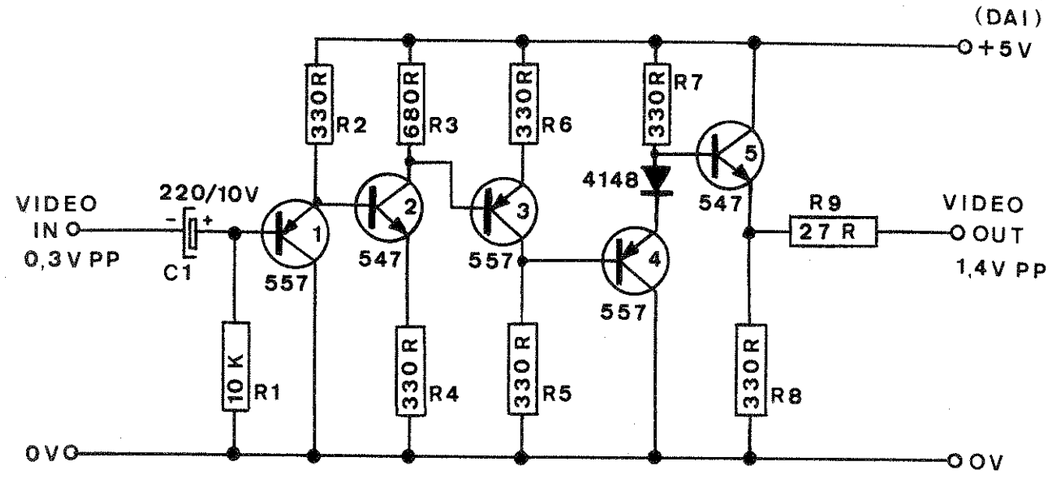
Inzendingen hiervoor worden verwacht bij

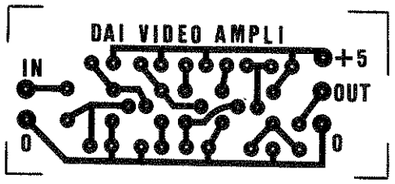
Nico P. Looije
Paludanushof 22
3151 CM Hoek van Holland
Nederland

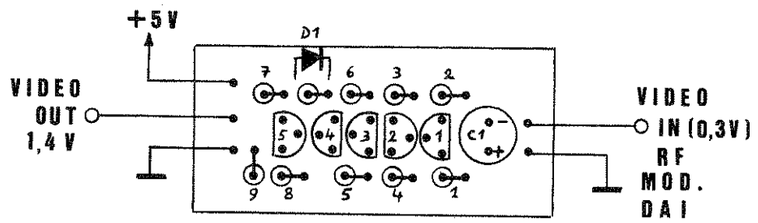
*** Contest ***

We are looking for a routine, as fast and as short as possible, performing the same action as the program above. Should NOT be used : GOTO , IF ... THEN , ON ... GOSUB.(If you can use routines from ROM, please send them although out of competition).
When the routine has been executed, variables X and Y should contain a new value, following the CURSOR-key being pressed.If no key has been pressed or another key,X and Y should remain the same.
CURSOR-key or CURSOR+SHIFT-key should give different results!
The best entries will be published and rewarded with software.
Send your solutions to Nico Looije.(see address above)

DAI VIDEO AMPLIFIER







CONT from p. 26

```
1750  GOTO 10
1800  ON Z! GOTO 99,990
9900  PRINT CHR$(12):PRINT "Vous avez fait une erreur de frappe":PRINT :GOTO 15
9999  END
```



```
*****************************************************************
*  Een andere mogelijkheid bij IF...THEN... in DAI BASIC V1.0 *
*  door F.J.A.Prop te Utrecht                                   *
*****************************************************************
```

In DAInamic 14 pag. 15 vertelt Jan Boerrigter over de mogelijkheden om een IF...THEN...ELSE statement te realiseren bij de DAI BASIC V1.0 versie die dit statement niet kent.

Na het commando:

```
10  IF A=1  GOTO  30:GOTO  40
```

gaat de DAI wanneer A<>1, naar regel 40.

Evenwel als we na het IF...THEN... commando iets anders toevoegen dan GOTO..., reageert de DAI op een andere manier. Boerrigter waarschuwt dan ook om deze constructie alleen te gebruiken in combinatie met GOTO. We kunnen echter ook nuttig gebruik maken van de andere mogelijkheid bij het programmeren. Neem bijv. de volgende regel:

```
40  IF  M=0  THEN  GOSUB 500:GOTO  100
```

hier wordt, wanneer de GOSUB is uitgevoerd, na

```
550 RETURN
```

netjes naar 100 gesprongen. Als echter M<>0 dan wordt niet naar 100 gesprongen, maar naar de volgende regel van het programma; dit is dus in feite een ELSE... wijzend naar die volgende regel. Hiervan gebruik makend, is het mogelijk om een hele reeks commando's achter een IF te zetten die alleen worden uitgevoerd als aan de IF-voorwaarde wordt voldaan.
Het volgende "nonsens"-programma illustreert dit: eerst het programma; daarna een aantal voorbeelden van uitvoering.

```
10   REM Nonsense programma ELSE in BASIC V1.0
20   INPUT "A=";A:PRINT"",
30   INPUT "B=";B:PRINT"",
40   INPUT "C=";C:PRINT"",
50   INPUT "D=";D:PRINT
60 IF A=1 THEN PRINT "Aa=";A,:IF B=1 THEN PRINT "Ba=";B,:
IF C=1 THEN PRINT "Ca=";C,:IF D=1 THEN PRINT "Da=";D;
80 PRINT:PRINT
90 PRINT"NIETS MEER! druk SPACETOETS":CALLM #D6DA
100 GOTO 10
*RUN
A=?2          B=?1          C=?1          D=?1

 NIETS MEER! druk SPACETOETS
A=?1          B=?2          C=?1          D=?1
Aa= 1

 NIETS MEER! druk SPACETOETS
A=?1          B=?1          C=?2          D=?1
Aa= 1         Ba= 1

 NIETS MEER! druk SPACETOETS
A=?1          B=?1          C=?1          D=?2
Aa= 1         Ba= 1         Ca= 1

 NIETS MEER! druk SPACETOETS
A=?1          B=?1          C=?1          D=?1
Aa= 1         Ba= 1         Ca= 1         Da= 1

 NIETS MEER! druk SPACETOETS
```

If we load this program somewhere in the memory there would be a good chance of a crash. If for example we return to Basic, the H & L registers could contain a different value from that which they held before the routine. To avoid such mishaps we start the routine with PUSH H and end it with POP H, in machine code E5 and E1. Do not take this too literally; the POP H should actually be before the RET not after it. Another risk is that the routine could set a flag without you making use of it. The flags can be protected with PUSH PSW and POP PSW. A safe rule for unsure programmers is to PUSH all (put on the stack) at the start of the program and at the end, to POP all (bring off the stack). The machine code now becomes:-

```
E5 21 00 00 22 B1 02 22 B3 02 22 B5 02 22 B7 02 E1 C9
```

and we can place it practically anywhere. Let us run through the possibilities.

- Placing the routine in page zero (0 to 2EB). This is a good solution but we must take care that the chosen address will not be used by our program. An examination of page zero shows only enough space for a short machine language program: the bytes from 1F5 to 274 where envelopes would be saved. If your program uses a small or no ENVELOPE then this space is a possibility.

- Before the Heap: Many MLPs (FGT included) make use of this possibility. Before we put in our program we set the START OF HEAP (29B 29C) to an address above our machine program and give, in Basic, a NEW or a CLEAR. Then we should have no problems in combining Basic and MLP.

- In the Heap: We define a string or array and put our program in it. An advantage is that larger programs are read in as arrays and there is no need to adapt the pointers. A disadvantage is that Basic programs are no longer free; the array or string used may not be shifted or replaced with another and the program must always start with the placing of the MLP.

- In the Basic program: Without getting too involved we have two methods, in a REM or in a PRINT statement. Advantages are: the program can be fully edited, listed, saved on tape and there are no pointers to be adjusted. Disadvantages: it is difficult to place and call the program. It is only suitable for short MLPs, preferably containing no jump addresses.

- Between the Basic program and the Symbol Table: Difficult to place, but safe. Could possibly be put here for saving (with Basic) but elsewhere for working.

- In the Symbol Table: As with the last one but easier to place at the end. Take care when moving as that location is sought via the end-of-symbol-table pointer (2A3 2A4) and that could indicate something else.

- In the free RAM between the end of the symbol table and the bottom of the screen. Take care that while running the program will not be wiped out by a change to a different MODE or be ruined by editing.

- In the ROM: Two possibilities, in EPROM such as is done with Memocom, or in the stack. But the latter is dangerous; you must really know what you are up to here. DBL for example works in this location. A program in the stack does work faster.

There are still more pros and cons in choosing a specific location but I do not want to dig too deeply now. However I still want to introduce a final change to the program. As it stands one must call the routine but it could be preferable for that to be done automatically. To do this I use the interrupt system. Every 20ms an interrupt is given on Vector 7, or in other words, there is an automatic jump to Interrupt Vector 7 every 20ms. This vector puts registers H & L on the stack, loads H & L with the contents of address 70 & 71 and then puts the contents of H & L in the program counter. Thus the program continues with the instruction which would be found at the address stored in location 70/71 while the old contents of H & L are saved on the stack. A

complicated but well thought out method.

We can intercept this interrupt routine by replacing the address in 70/71 with the address of our routine. Then we do not need to call the routine as it will be called automatically every 20ms (50 times a second). We must of course ensure that the normal business of Interrupt 7 can still proceed so instead of ending the MLP with a RET (C9) we end it with a jump to the normal Interrupt 7 routine, address D9A9. Thus instead of C9 we put C3 A9 D9 (bytes reversed - remember?) But there are still some problems. The interrupt handling must not be disrupted by another interrupt so we start our routine by DI, Disable Interrupts (F3) and at the end we restore them with EI, Enable Interrupts (FB). Our program now becomes, in machine code:

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

Our only worry now is that the interrupt should route us to the routine. It seems simple: change the address at 70 and 71 to the address where the routine resides by, for example, POKEs. But if we POKE address 70 precisely at the moment when an interrupt occurs on Interrupt Vector 7, the address in 70-71 will be neither the old one nor the new. There are two ways of overcoming this:

1) Make the routine's address be in the form xxA9 so that only one byte needs to be changed.
2) Write a short additional MLP that puts the desired address in 70-71 after the interrupts.

Another possibility is to change it, not in the program, but directly in UT, then V7 gives D9A9; change it to the required address. A major disadvantage is that as it is not within the program and restoring the old value is difficult. If for example you type UT what comes on the screen is something like UUUUUUUTTTTT and that gives SYNTAX ERROR. However we found a neat solution; the interrupt routine is at 38 to 3F but the last two bytes (3E-3F) are unused. The routine loads by means of the address 0070 which will be mentioned in 3B-3C. More accurately, 3B contains 70 and 3C contains 00. We now put the address of our routine in 3E-3F (reversed !) and can start our proceedings with POKE #3B,#3E and terminate with POKE #3B,#70. And because it puts registers H & L on the stack we can leave out the PUSH and POP. Below are Basic programs to demonstrate both.

With thanks to Nico and 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$="mpl": FOR I=1 TO 20: READ A: A$=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
```

From the tourist resort of Emperor Charles came a small wind of change in the form of a user group which has since achieved honour. From the earliest stencilled newsletters to the well planned and sophisticated magazines which now come through the post there has been much progress in a short time. With Dainamic strength programs have been written, projects, meetings and exhibitions planned. This unselfish enthusiasm spread contagiously far beyond the borders of our small country. The brain child of some of our northern neighbours, the DCR extension was born; from Germany and France came sophisticated programming aids, games and utility routines in machine language. All these goodies were available to the members at fair club prices and that made the already cheap DAI the best system, price and quality-wise.

The quality of the publications, already high at the start, seems to have been maintained so without a break. I look forward with great faith to the future expansion of our DAI. I thank you for all this and for the pleasure I have obtained from my DAI.

Best wishes from an enthusiastic DAI-user,

Staf Van Hoecke

PROGRAMMING TECHNIQUES

(from DAInamic 15, page 102)

A question we are often asked is how to write a program in machine language. Let me first give a few facts. In the DAI is a microprocessor, the 8080A from the firm of Intel, or more likely, one made under licence by NEC. This 8080 receives instructions of 1, 2 or 3 bytes. We can give it instructions ourself by first typing in UT followed by Return and then by, let us say, S3000, filling the bytes from #3000 onwards with our instructions. You must always put in only one byte, two Hex characters from 0 to F. If you put in more, the monitor program will only take the final pair. Then press the space bar for the next byte. Take note that the CHAR DEL does not work in this situation.
Let us try it. Switch on the computer and give UT [Return], S3000 [space] and the monitor displays the present contents of #3000. Now type C9 (the code for Return), then the cursor-left key and B which switches you back to Basic. Try now a CALLM #3000 and see the asterisk appear again indicating that your own machine language program has been executed and returned you to Basic again. That will do for now but we will have a longer program later.

Long programs are difficult to write in this fashion so there is a program which translates simple language into machine code. That simple language is Assembler. It uses mnemonics or abbreviations as instructions. Some examples are:

Machine language	Assembler	Meaning
3E 05	MVI A,05	Move the value 05 into register A
81	ADD C	Add the contents of C to A and put the answer in A
CA 12 34	JZ LABEL	Jump to the address "Label" if the zero flag is set (i.e. if the outcome of the last instruction which could set the flag is zero)

In assembly language one can assign a label which can be used as an address, like a line number is used in Basic. Take care, in the above example the address we go to for LABEL is #3412 and not #1234. If our program is not too long we can input it with the Substitute in UTility, or by POKEs in Basic, but if it contains Jump instructions we first have to calculate all their destination addresses. Also if alterations are subsequently made to the program or its location

in RAM is changed, all those addresses will need to be changed too. It is therefore strongly recommended that anyone programming in machine code/assembler should purchase an assembler program (DNA or SPL).

I can understand that some find the distinction between machine and assembly languages somewhat vague so I will give some clarification. We cannot combine a basic program with an assembly program since the translator only understands Basic and an assembler only assembly language. However we can first put our basic section into machine code with a compiler, then convert the assembly section into machine code and finally combine them. This is an involved method which gives rise to many problems when combining; and who owns a basic compiler for the DAI? It is therefore best to write a Basic program and in it call, with a CALLM, a machine language routine that temporarily takes over from Basic. That machine language routine can be put in by one of the previously discussed methods.

Now we will make up a clever program in machine language. I have chosen an idea from the previous issue: to set to zero the addresses #2B1 to #2B8 so that we always receive a response from GETC and so get a reaction from the machine as long as the key stays pressed. Example 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
```

Comments :-
1) All in integer.
2) In line 20, back to 20 and not to 10 which is good but slower and we do not want that.
3) The program stops after 4 lines.
4) With repeat it works slower.

To write this in machine code we study the list of 8080 mnemonics. We have to write more of the routines in Assembly than we would in Basic. There is, for example, no FOR - NEXT to give a simple loop.We choose the instruction "SHLD address" which stores the contents of the H and L registers in the stated address. That is, SHLD 3000 puts the contents of L in 3000 and the contents of H in 3001, although as we are putting zeros in both H & L the sequence is of no consequence in this case. There are several ways of making H & L zero but we will use the customary one, LXI H,0. This instruction fills H & L with the address, or value, that follows it. Here also the first byte goes in L and the second in H, but again the order has no significance when both are zero. As there are only 8 bytes to fill we will not create a loop but program each directly. This gives the following advantages:-
- No jump addresses so the program can be moved without change.
- The program runs faster.
- There is less chance of error, especially without an assembler.

We write the program in assembly language thus:-

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

In machine code this becomes:-

```
    21 00 00 22 B1 02 22 B3 02 22 B5 02 22 B7 02 C9
```

Step 6 gave rise to much thinking. How to find the next point and work around the figure. We go round as follows:- see if you can go upwards; if you cannot, try to the right; still not possible, try down and finally to the left. If you cannot go left you have found a point from which to draw a line to the right. Well programmed all this can be done in 4 lines (20,30,40 & 50). Line 60 checks readiness and then 70 draws the line.

There is another interesting point in the Basic program: lines 100 to 170 give a neat but simple method for moving a spot on the screen. The cursor keys move it one step at a time but the cursor keys with shift key held down move it 10 at a time (alterable).
This program section also caters for the possibility of you drawing figures yourself or of choosing a standard one (from DATA lines).

A few final remarks on the two machine language routines: the first version works in all modes but checks nothing. The second is faster, checks correct use of colour and gives OFF SCREEN if the edge of the picture is reached. Its disadvantage is that it only works in 4-colour modes, although this is not much of a limitation because conflict of colours is a problem with the 16-colour modes; try the Basic version in 16-colour mode to see this.

All solutions have snags: if there is a figure within a figure it casts a shadow to the right. This is not solvable realistically but here are a couple of possibilities:-
1) Draw a line between the outer edge and the inner figure; all will then be filled correctly and then the connecting line can be removed by redrawing it in the fill-colour.
2) Draw the outer figure and colour it; draw the inner figure and colour that with a different colour.
3) Choose a new starting point in the still unfilled section. The inner figure then becomes the border.

Below is the Basic program and as you type it in, do not forget to type first IMP INT.

```
10   CLEAR 256: GOTO 400: REM  PAINT  /  F.H.  DRUIJFF  12/82
20   IF SCRN(X,Y+1)<>G THEN Y=Y+1: GOTO 50: REM   Up
30   IF SCRN(X+1,Y)<>G THEN X=X+1: GOTO 20: REM   Right
40   IF SCRN(X,Y-1)<>G THEN Y=Y-1: GOTO 30: REM   Down
50   IF SCRN(X-1,Y)<>G THEN X=X-1: GOTO 40: REM   Left
60   K=X-1: IF X=BX THEN IF Y=BY THEN IF F=1 GOTO 80:F=1
70   K=K+1: IF SCRN(K,Y)<>G GOTO 70: DRAW X,Y  K-1,Y  W: GOTO 20
80   END
100  C=SCRN(X,Y): DOT X,Y  23: H=GETC: DOT X,Y  C: IF H=0 GOTO 100
110  IF H<16 GOTO 170: IF H>23 GOTO 170: V=H/20*9+1
120  ON H MOD 4 GOTO 140,150,160
130  Y=Y+V: IF Y<YMAX GOTO 170: Y=YMAX: GOTO 170
140  Y=Y-V: IF Y>0 GOTO 170: Y=0: GOTO 170
150  X=X-V: IF X>0 GOTO 170: X=0: GOTO 170
160  X=X+V: IF X<XMAX GOTO 170: X=XMAX
170  RETURN
300  PRINT " Move blinking dot. If inside field to colour press space."
310  GOSUB 100: IF H<>32 GOTO 310: X=X+1
320  X=X-1: IF SCRN(X,Y)<>G GOTO 320: X=X+1: BX=X: BY=Y: F=0: GOTO 60
400  G=14: W=1: COLORG 0 W G 15: MODE 4A: REM  Initialization
410  INPUT " Standard /Own drawing ";VE$: PRINT
420  X=33: Y=33: IF VE$="O" GOTO 500
430  READ P,Q: IF P+Q=0 GOTO 300: DRAW P,Q  X,Y  22: X=P: Y=Q: GOTO 430
500  GOSUB 100: IF H=13 GOTO300: IF H<>32 GOTO 500
510  IF F=0 THEN DOT X,Y  22: F=1: P=X: Q=Y
620  DRAW X,Y  P,Q  22: P=X: Q=Y: GOTO 500
900  DATA  147,12,147,45,90,45,80,31,80,60,120,70
910  DATA  85,70,75,60,66,45,45,55,33,33,0,0
```


The Defect Relating to the +5v Supply

(from DAInamic 15, page 86)

ATTENTION ! The printed circuit design published in DAITA is not correct.

It appears that some DAI users have recently had their +5 Volt supply go high, with the possibility of destroying various TTL chips, among them, some PROMs (74LS288), with somewhat large financial consequences.

Something can be done about it with the above circuit design [in DAInamic 15]. The design is the so-called CROWBAR circuit which detects voltages above a set level. When a defective component in the supply causes a too high voltage (22 volt) to be presented to the circuit, the over-voltage is intercepted and instantly short-circuited. Thus the high voltage is prevented from harming the circuit it feeds. The effect on the power supply circuit is that apart from the component that caused the over-voltage, some other components could be damaged. I am thinking for example of the TIP 34, BD 140 or the switching IC but this is little compared to the possible damage to the rest of the computer. The ELCO may also be destroyed. A possible voltage fault could also be indicated by the disappearance of the picture from your TV screen.

The connections are made with the + to the top of the fuse-holder, as seen from the keyboard. The - can be soldered to the screening of the supply; use a goodly sized wire for this. Judging from the component list it looks as if use is being made of not readily available components but together they guarantee a good level of protection.

Should you have problems in purchasing the components or in assembling the circuit contact the undersigned.

T Verberkt

Letter from Staf Van Hoecke.

(from DAInamic 15, page 87)

Dear DAI Users,

Now that the Users Club has been in existence for about three years I think it is appropriate to offer a few words of thanks to the management and editorial team. During that time it has become clear to every user that the popularity of the DAI Personal Computer has considerably increased. This is thanks to the enormous efforts of a hard core who must take comfort from the fact that a comparatively unknown computer has taken its rightful place among the most powerful systems available at hobbyists' prices. From its birth the DAI had all the hardware needed to put it among the great ones, but .... as with any system, the success or failure of a personal computer relies on the software and whether documentation and supplementary information is provided so that the system can be further extended.

I was one of the first to buy a DAI and I lived with the emptiness and silence which prevailed after the introduction of this genial computer. Nowhere was the existence of the DAI reported and neither did programs appear in the various magazines. On the subject of hardware extensions a similar enthusiastic silence was maintained. In brief, it was a period we can now look back on with a grim smile.

which affects the speed can be as near the front as possible. However, not a GOTO instead of a GOSUB because in a subroutine there must be no CLEAR command.
Make the cursor visible or invisible as desired. etc, etc ...

P.S. I want to draw attention to a circle drawing routine by Fred van Amerongen which was sent to me after an earlier article.

Frank H Druijff

```
10  REM INCREMENTAL CIRCLE GENERATION.
20  COLORG 0 5 10 15: MODE 6: MODE 6
30  K1!=3.0: R!=140.0: XC=XMAX/2: YC=YMAX/2: REM .... CENTRE
40  R!=R!-R!/10.0: REM .......... RADIUS
50  X1!=R!: Y1!=0.0: REM ......... STARTING POINT
60  K!=K1!/X1!: REM ............. INCREMENT
70  FOR I=0 TO (2*PI)/K!: X2!=X1!+K!*Y1!: Y2!=Y1!-K!*X2!
80  P=X1!: Q=Y1!: M=X2!: N=Y2!: DRAW P+XC,Q+YC  M+XC,N+YC  21
90  X1!=X2!: Y1!=Y2!: NEXT: IF R!>2.0 GOTO 40
99  END
```

LETTER
(from DAInamic 14, page 45)

Dear Sirs,

I read with interest the FGT-DISK-PEEK-POKE article in issue 13. The method used by Mr Gesp was, I thought, over-complicated and made no use of the possibilities that the DOS offers us.
Whenever we let the FGT program work with the DOS it will be loaded in at the same time and remain there as long as we wish. The division of program sections is also incorporated. It is only the table that is changed when we want another letter-type. By making D-files from the tables a program has available more than one letter-type.
Before going too far a spot of juggling is needed but no tricks are involved.
The new DOS consists of:-

1. The original DOS from 02E3 to 19CB
2. The FGT program from 19D0 to 1C05
3. The FGT table from 1C10 to TBLEND
4. Free space from TBLEND to 27E0

TBLEND is the end of the FGT table.
The free space is needed for the longest table to fit in the DOS. TBLEND of the longest table is thus 27E0. Whenever the new DOS is loaded the HEAP will start at 27E1. Therefore there is still room for a goodly sized program. Besides the DOS the various tables are on the disk for programs that can make use of them.

Now, how do we work it ?
After a HARD-RESET we load the FGT with the longest table and go to UTILITY to see where it ends. Then we load the FGT with standard table and put the whole of DOS and FGT on a system

-TRANSLATIONS-TRANSLATIONS-TRANSLATIONS-

disk from which $MSTRDOS has been deleted.
Type in : DSAVE $MSTRDOS:0 2E3 27E0 13EC [RET] and our new DOS is ready. The address 27E0 may, in your case, be some other; it depends on the length of the longest table you are considering using and of the location of your FGT. We now load successively the programs of the various tables, DSAVEing each of them on the disk with an appropriate name.

How does a program load its table ?
For this we use the command DLOAD but place it between the customary pokes; example :-

```
100  POKE #131,3: PRINT "DLOAD MATH: 0": POKE #131,1
```

This program line needs to be executed once only and can if desired be repeated for another letter-type. It is no longer necessary to load the whole program in array and POKE to its place, thus saving time and space.

I suggest you try it out by running the adapted demo-programs which are on the enclosed disk.

With best regards, Frans Couwberghs.

What is diDAIsoft ?
(from DAInamic 14, page 55)

diDAIsoft is a software group which works within DAInamic and consists of teachers from all levels of Belgian education (lower, secondary and college). The group aims to develop instructional software for all branches of (mainly secondary) education. At the time of writing the group has 27 members who meet 2-monthly in Haasrode (Belgium).

PAINT by Frank Druijff
(from DAInamic 14, page 70)

The PAINT instruction is a graphics command, available on some machines but lacking on the DAI. After some philosophising about how it could be achieved along came Hans Peters with a bright idea. When the Basic version was working the next step was to write it in machine language. It was faster but still not fast enough, possibly because it was using the SCRN routine of the ROM. That meant that on every call to the routine the DAI checked X and Y values and calculated the position of the required point. But that was only needed once, at the beginning. I asked Nico Looije to help and he improved the speed by a factor of 4 or 5. The DAI too can now have a PAINT command! It will appear in a new TOOLKIT collection.

How does PAINT work ?

1) We choose a point in the figure we want to colour.
2) Give the following data:- the X & Y of the point; the border colour of the figure to be filled; and the desired fill-colour.
3) The routine starts by seeking the left edge.
4) From here it seeks the right hand edge.
5) It draws a line from left edge to right edge using the fill-colour.
6) The next point on the edge is sought from which a line (possibly of length 1) can be drawn to the right.
7) If the exit point has not been reached, repeat 5 & 6.

PROGRAMMING TECHNIQUES

(From DAInamic 14, page 16)

In this article I will delve into the programming of a program, which at first sight is a strange topic. Everybody with a DAI regularly writes programs, but the actual programming is something quite distinct. Regretfully I have to say that on the basis of what is submitted to us, the design of the program and the implementation of program ideas leaves something to be desired. There is no standard arrangement which should be used in the design of all programs, but we should at least first ask ourselves what we want to do with the program. When we have decided that, we can start thinking of a suitable construction.

Some possibilities :-

1) A program of our own invention;
   - we must do it ourself
   + we can do it ourself
2) A program seen working on another computer;
   - we perhaps try something for which the DAI is less suitable
   + we can use the original material
3) We have the program listing but for a different machine;
   - we try something for which the DAI is unsuitable
   - we stick to the style of another programmer which inevitably means a worse program because the author was familiar with his machine and not with ours
   - programs in Hobbyscope's Basicode have the limitations, not of one machine but of many
   + nobody can accuse you of originality

If we add homemade programs to the list there is then the possibility that the DAI's capabilities are used to the full. For the next phase in our programming it is worthwhile thinking about the following points :-

1) For whom is the program intended ?
2) Will I still want to use the program in the future ?
3) Will I later use the program or part of it in another program ?
4) Is it necessary for others to understand how my program works ?
5) Is it necessary or desirable for my programs to bear my own signature ?
6) Is an explanation necessary for others to use my program ?
7) Are variables to be in integer or floating point ?
8) How quickly must the program be ready ?
9) At what speed must the program run ?

I will expound further on the respective points:

1: If a program is to be used only by you and just once, explanations are superfluous.
2: If you will want it again minimal explanation is essential. Put the date at the start too.
3: If yes, programs must be in modular form. More about this later.
4: Essential for teachers of computing, like me, but also useful if there are some programming tricks which others could assimilate.
5: Only if you wish or if there is a commercial purpose to your programming, but keep it simple.
6: Brief remarks are normally desirable. The more people who will use the program the greater the need for clear explanations. Try them out on an inexperienced user first to ensure adequacy.
7: Never index arrays in floating point. Cursor and Draw commands need integers. Many FOR-NEXT loops can also be in integers.

-TRANSLATIONS-TRANSLATIONS-TRANSLATIONS-

8: If you are in a hurry do not bother too much about the program's appearance; make use of standard routines. It may be useful to maintain a library of such routines.
9: If running speed is important, here are a few tips: Work as much as possible in integers. Try various programming possibilities. Make the speed affecting section the most significant part of the program and put it near the front. Write parts of the program in machine language.

I will quote you an example from my own experience. As a mathematics teacher I needed a list of whole numbers in groups of three, representing the three sides of right-angled triangles. The numbers were to be below 100 and were to be used in exercises on Pythagoras' Theorem. The program was not intended for use by others and was wanted quickly.

Now, for programs which others will be using, the following example of an introductory section could be something to aim for:-

```
10  REM JOURNEY AROUND THE WORLD / COPYWRIGHT Demaeker  Jan-83
20  GOSUB 8000: REM Announcement and Reading-in Data-1
30  GOSUB 7000: REM Explanation and Reading-in Data-2
40  GOSUB 6000: REM Initialising and Reading-in Data-3
50  GOSUB 1000: REM Program
60  PRINT "SPACE-BAR AGAIN"
70  H=GET: IF H=0 GOTO 70: IF H=32 GOTO 40
80  END
```

In subroutines 8000, 7000 and 6000 the subroutines for reading data is called. It is a pity that more programmers do not do this, as often a user sits reading an explanation on the screen while the computer does nothing. Then when the user signifies he is ready, data reading commences and the user sits doing nothing. It is obviously better if reading data and constructing files is done while the comments and explanations are on display. Because the speed of the GOTOs and GOSUBs could be influenced by the line's position in the program it is worthwhile using the same sequence as in the example. The number of the line is not as important as its position in the listing. This is because the DAI reacts to a GOTO 1000 with: 1 - get the first line number from the listing, and 2 - is it 1000 ? If yes, carry on, but if not find the length of this line and determine the position of the next line. Go back to 2.

The above is also an example of modular programming. It is in clearly recognisable sections, in this case in subroutine form. The style is good. The sections could be easily replaced by alternatives or extracted for use in other qrograms. Should the speed be too low it would be a simple matter to substitute a CALLM for a Basic subroutine. The program is readable and logically arranged. But there are snags: it will never achieve a really high speed and in the extreme case it could fail to fit in the memory because it is somewhat longer.

Finally some notes on the best way to start a program. On past occasions I recommended beginning with 10 MODE 0: ?CHR$(12) because many programs started without a clear screen and without switching to character mode. But later, snags became apparent when very long programs running in Mode 1 or 2 gave OUT OF MEMORY after the MODE 0.
The ?CHR$(12) has two disadvantages: if the whole screen is filled after this command the last line will annoyingly move up. The remedy is simple: ?CHR$(12); . We can also get OUT OF STRING SPACE if a CLEAR 4 had been given. You may well ask why give such a command: for example, to obtain the size of the program. Here too there is a simple remedy: give the CLEAR first, before the PRINT.

We would also wish to see a COLORT before MODE 0. Put a REM in the first line for the Program title and the author's name, but use a GOSUB for it so that the section of program

FWP

DCR-commands b1 (buffer 1) b2 (buffer 2) b1 or b2 (STEP) Array (load string) Load file Save file Print file Move (06-07) to (01) Kill (06-07) Replace word(s) Eliminate spaces Trim lines Width equal (fill spaces) Clear b1 or b2 (STEP) Default menu Headings in b2 Update/store headings Xfer b1 (06-07) to b2 Get b2 in b1 (01) Init. byte to printer Find word Basic (Z..UT)	Line (chars/line) Page (lines/page) Formfeed lines Margin (spaces) Step size Baudrate (1..7) End width (60..90%) Convert chars Tab menu Wait after page Nrs (page nrs) Auto-trim end line Hyphen (trim) end line Install colours

EXTRA FEATURES WITH FWP....

The final version of FWP offers the possibility to be linked with a BASIC program for exchange of information. One can make certain calculations on values, or string manipulations and transfer the results to FWP , to be incorporated in a text file. A BASIC demo program is included in the FWP-package.

Price : audio : 2000 Bfr DCR : 2150 Bfr

Update : It is possible to update from WP or DAINATEXT to FWP.
(FWP can read the old files).
This should be done before 1/5/84 with the enclosed order card. Price of update is 1000 Bfr for audio,
1150 Bfr for DCR.
(original cover of WP or DAINATEXT should be enclosed with your order !)

PUZZLY

price : audio/DCR : 2100 Bfr

U houdt van kunst , U houdt van breinkrakers...
speel PUZZLY , de eerste puzzel op computer.
PUZZLY biedt ook verbazende grafieken, verschillende spelniveaus's en chronometrage.
PUZZLY bevat 4 puzzels, nieuwe plaatjes zijn mogelijk.
U like art, you like intellectual games...
play PUZZLY, the first puzzle on computer.
PUZZLY offers amazing graphics, different levels and chronometer.
PUZZLY contains 4 puzzels, new pictures are possible.

DAYLAXIANS

audio/DCR : 2100 Bfr

U bent de bestuurder van een ruimtetuig.
U wordt aangevallen door een vloot XORS, onder leiding van twee "AMIRALS". De aanvallen worden steeds grimmiger, wees op uw hoede !
You are the captain of a space craft.
Fight against the agressor ships, under command of 2 "AMIRALS". Take care, the ennemy ships get more and more agressive !

DUEL

price : audio/DCR : 2100 Bfr

DUEL is geen wandeling door een vreedzame weide, maar een lijf-aan-lijf gevecht in een versterkt kasteel.
Reuzeschildpadden,ratten,honden en giftige spinnen maken uw strijd nog moeilijker.Pas op voor de bewakers van het kasteel : " de KALE met de knots " en " de woestijnridder met zijn zwaard".
DUEL is no walk in the meadow but a one-to-one fight in a middle-age castle.
Giantturtles,rats,dogs and poisened spins make surviving very difficult. Beware of the guards : "the BALD with the knot" and "the prince of desert with his sword".

C.L.I.O.

price : audio/DCR : 3000 Bfr

Maak uw eigen tekeningen met uw DAI en C.L.I.O. .
Vele grafische functies zijn mogelijk :
ZOOM : vergroot 4X of 16X een willekeurig gedeelte van het scherm.
Verander kleur , speel met ingebouwde geometrische vormen, verplaats of copieer een deel van het scherm,enz.
Create your own pictures with you DAI and C.L.I.O. .
Many build-in functions :
ZOOM : enlarge 4X or 16X a part of the screen.
Change color, play with the build-in geometric shapes, move or copy parts of the screen etç...

Noot: we hebben van deze pakketten slechts een beperkte voorraad, houdt rekening met een leveringstermijn van 2 à 3 weken.
Note: We have only a limited stock of these packages, please allow 2 or 3 weeks for delivery.

FWP

Introduction

FWP is a new 100% machine-code wordprocessor program designed especia-lly for the DAI. This enables very fast operation and economical use of memory, leaving the maximum possible available for text processing.

If you already posess one of the older wordprocessors which uses string-arrays for saving/loading text-files, note that these still are compatible with FWP and can be converted to object files (type 1), as are used by FWP.
This illustrates one of the advantages in using less memory space during loading/saving actions compared to working with string-arrays which can occupy twice as much memory space.

A very important aspect of text-processing is that the program offers maximum text protection against user errors. For instance even inad-vertant operation of the reset-button will not destroy program or text.
This program has been subjected to many months of daily practical use and is now proven to be very reliable.

FWP has 3 buffers, a main-buffer (b1), where the actual text is stored, an auxillary-buffer (b2), which serves as a temporary store of text and additionally a third buffer (b3) in which headers or cha-racter-strings can be stored and recalled.
A block of text in b1, once set between markers can be copied in b2, or conversely the whole of b2 can be copied in b1, as indicated by a marker.

Text handling in b2, using commands in edit-mode and main-menu closely follows b1 practices. However, when b2 is selected no use can be made of the commands "Load file" and "Save file", because these commands refer to the main-buffer (b1) only.

Headings or for instance a series of printer-commands can be prepared in b2 and then via the command "U" copied to b3. These can subsequent-ly be recalled to b1 for insertion into the text.
The content of b3 can be saved together with FWP, as one object file (see para 4.0).

THE ULTIMATE WORDPROCESSOR

It is now possible to select one of the main-commands by typing-in the relevant letter. This menu is also used to select secondary menu's, for instance "D" will display the "Set default-menu" which amongst other niceties sets text formatting/parameters.

DCR users have the possibility of giving commands such as REW1:LOOK, without leaving FWP.

FWP uses the standard edit-buffer, which as everybody knows, slows down as the text grows. This can lead to typing errors after a full screen of text has been typed in.
FWP minimises this problem by giving the possibility of working in STEP's. This actually means that the starting point of the edit-buffer is adjusted (according STEP-setting), towards end of the text, thereby compensating for the loss off speed by reducing the amount of text held in the edit-buffer.
A minor disadvantage of this method is that if a large text has already been typed-in and subsequently further editing needs to be done at the begining (STEP 0), the usual slow editing function will re-appear.
The solution for this is to take the relevant portion of text (between 2 markers) and copy to b2, using the command "X".
Once in b2, the copied section can be quickly edited and inserted in b1, after having deleted the "old" text.
This sounds complicated, but in practice works efficiently.

The standard cursor-functions are extended with 10 extra , some in-corporating "auto-repeat". For example fast re-positioning of the cur-sor by word or complete line is possible.

Another important feature of FWP is "auto-trim", which automatically gives a line-return at the nearest space to the set line-length, while ensuring that the maximum length is not exceeded.

The usual wordprocessor commands, such as replacing "R" or finding "F" words or complete strings are available from the main-menu.

There are no restrictions in using all the available key's for text input. In addition ASCII characters $\leq$ 80H (hex) can be selected and used by keying SHIFT and hold while pressing CHARDEL, followed by the 2 digit hex code for the required character.
If the value is $>$7FH, the MSB will be stripped e.g. 83H becomes 03H.
The value 00H or 80H will not be accepted.

The same procedure is used to generate and set the FWP special markers 01,02,06,07,18, and 19 (all hex).

An additional useful routine is a conversion routine, which gives the possibility of changing the value of ASCII characters in b1 or b2 during printing, to any value between 0-255 (decimal).

FWP ensures that al output to the printer is disabled, except when the "I" or "P" commands are in effect. This avoids displayed error-messages being printed.

West-Souburg, feb 1984

Geacht medelid,

Even mezelf voorstellen : Vorig jaar "kreeg" ik mijn DAI-computer. Dit feit bracht mij in contact met DAInamic en de bestuursleden. Daarbij werd mij de vraag gesteld om eens wat van mijn ervaringen als beginneling door te geven. Het boek van Bruno van Rompaey "GESTRUCTUREERD PROGRAMMEREN met DAI-BASIC", deel -1- is de aanleiding om deze belofte in te losse. Maar eerst verder met het verhaal. Ik met de computer en het handboek naar huis. Nu heb ik een nogal drukke werkkring (leraar wiskunde) en uit hoofde daarvan moest ik tussendoor een cursus programmeren in PASCAL volgen. Reden om niet al te veel tijd over te houden om me met de DAI bezig te houden. De weinige tijd die overbleef, gebruikte ik om het handboek door te werken en DAInamic door te lezen. Beide bezigheden brachten toch wat teleurstellingen. Het handboek begint eenvoudig, maar na korte tijd was ik de draad volledig kwijt. Nog steeds kan ik de logische opbouw voor de leek, wat daar is het boek toch voor, op z'n zachtst uitgedrukt niet bewonderen. Ook het lezen van DAInamic zonder enige voorbereiding is geen sinecure. Je kunt wel enige programma's intikken, de werking ervan bewonderen en proberen te doorgronden, maar het grootste deel van de inhoud is voor de onervaren computeraar niet te volgen. Misschien een idee om met een teken aan te geven voor welk niveau het artikel bedoeld is ?? Maar nu dan het kloek uitgevoerde en goed leesbare boek van Bruno van Rompaey. Laat ik klassiek beginnen met de complimenten: maak dit boek maar de echte DAI-handleiding. De manuel mag erbij geleverd blijven. Misschien dat een gevorderd DAI-gebruiker er een goed naslagwerk aan heeft. Algemene BASIC-boeken zijn vaak onhandig vanwege de typische DAI-kneepjes. Verder moet iets goed werken als je het zonder SYNTAX-fouten intoetst, anders raakt zeker de beginner snel gedemotiveerd. Om dit bezwaar te ondervangen is er de TELEAC-cursus, maar die is duur en vaak niet leverbaar . Het boek van Bruno van Rompaey biedt uitkomst. Alle moeite wordt gedaan om de gebruiker te laten begrijpen wat hij doet, zodat de opgedane kennis kan functioneren. Door zelf programma's in te tikken en daarin veranderingen aan te brengen kan via de resultaten veel geleerd worden. Prettig hierbij is dat de EDITOR snel gebruikt wordt. Het is geen boek met truucs : het bevat bvb een minimum aan POKES. Voor iemand die denkt dat hij DAI-BASIC onder de knie heeft, is het boek toch nuttig, daar naast BASIC-kennis ook een bredere kijk op de structuur van een programma wordt aangeleerd. Mocht het de schrijver slechts gedeeltelijk gelukt zijn programma's te structureren, dan is dit vooral te wijten aan het ontbreken van IF...THEN...ELSE, WHILE...DO ,REPEAT...UNTIL en Procedures in DAI-BASIC. Er is gekozen voor Nassi-Schneiderman diagrammen (structogrammen) om de loop van een programma te verduidelijken. Iemand die wel eens een stroomdiagram heeft gebruikt , kijkt hier in eerste instantie misschien even onwennig tegen aan. Hoewel de DAI-BASIC een chaotische programmeer-trant toelaat, wordt men door de structogrammen gedwongen tot overzichtelijk werken. Met name bij repetities is dit erg belamgrijk. De inhoud is klassiek van opzet : toetsenbord, SAVE , ... Prettig is dat EDIT snel behandeld wordt. Een zakenregister ontbreekt helaas (na het tweede deel ?). Na het motiverend woord vooraf met teksten als DAI-Picasso en DAI-Beethoven, is niet helemaal duidelijk waarom niet begonnen is met wat grafics. Ik weet niet of de opbouw van een groot programma voor administratie nu zo motiverend werkt voor hobbyisten en onderwijs, want dat is toch vooral de DAI-markt. Samenvattend wil ik zeggen dat het boek aanbevolen kan worden. Men kan ermee leren zindelijk te programmeren ondanks BASIC. Met spanning kijk ik uit naar deel 2.

Wim Schout

GESTRUKTUREERD PROGRAMMEREN

MET DAI - BASIC : 1100 Bfr

D-BASIC

soon available

D-BASIC is a new high-level-language for the DAI pc.
D-BASIC is being implemented on the DAI-pc at the moment and will be available soon.

D-BASIC is 100% compatible with the ROM-resident BASIC V1.0 and V1.1.
D-BASIC extends the existing instructionset with some statements intended for structured programming. Exemples of these structures are :

IF <expression> THEN statement's ELSE statement's ENDIF (selection)
WHILE <expression> DO statement's WEND (iteration)
REPEAT statement's UNTIL <expression> (iteration)

Other statements allow ERROR-trapping and ERROR-simulation, BREAK-trapping etc...

Besides an extension of the BASIC instructionset, D-BASIC also adds some totaly new elements to the ROM-resident BASIC.

These elements are :

1. labels

A line can be identified and refered to by a label.
Labels can be specified only in a program but they can be refered to also in command-mode.

2. functions

You can define frequently used expressions as a function. The definition can only be done in a program, however a function can also be invoked in command mode.
Any place where you can use an expression you can use also a function.
Functions can always be recursive i.e. they can always invoke themselves.
Functions can have 4 kinds of parameters : value-,variable-,array- and function-parameters. Value-parameters are parameters by value. The other 3 kinds are parameters by reference.

3. procedures

An oftenly used sequence of statements can be grouped into a procedure.
Procedure definition is done in a program, however procedure calls are also recognised in command mode.
Procedures in fact can be used to extend the BASIC vocabulary.
Also procedures can be used in a recursive manner.
The parameters of procedures have the same characteristics as the parameters of functions.

4. user defined commands

There is an easy way to link new statements code to D-BASIC. Besides the run-time machine-language code only a syntax description has to be entered in a table. This is ment for future extensions as :
-a dos-interface for a disk-oriented D-BASIC (ex. KEN-DOS)
-a dcr-interface (already exists with error-trapping etc ...)
-special graphical routines (ex. FGT-package ...)
-etc...

Willy Coremans

INHOUD — CONTENTS

NEWSLETTER 20

DAInamic subscription rates :

Benelux : 1000 Bfr
Europe : 1100 Bfr
Outside Europe 1500 Bfr
(Air Mail)

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

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

DAIPACMAN by MICHEL MEYER

Description of the game :

The aim of the game is to crunch the vitamines (little points) all around the maze without getting caught by one of the four ghosts. The large pink points are energizers: when you bite one of them, the ghosts are for a certain time vulnerable and you eat them (they look blue and get quite afraid or mad).

The first ghost eater awards 200 points, the second 400 pts, the third 800 and the fourth 1600 pts. So if you succeed ineating all of the four ghosts 3000 BONUS points. (all the points are 3410 pts worth)

Sometimes, there appear also bonus fruits at the center of the maze (the number of times they appear, the delay after which they appear and the time they keep in sight are all random (but in a certain measure only) for each new board.

Each ghost is quite different from the others in his reactions :

* the RED ghost is well called STICKY and is theoretically the most agressive.
* TRICKY, the green ghost is in fact also well named because of his quite smart reactions.(sometimes, he will wait you at the other end of the crossing).
* PINCKY is so called because of his colour, but he is not very agressive.
* POKEY does almost everything he wants but beware: he's often where you don't expect him!...

At the beginning of the game, you have 3 PACMAN's and the speed is rather slow.
But later in the game the speed is much greater and the ghosts more agressive.
The time of vulnerability (when YOU can eat the ghosts) is also decreasing in the long run ....!

You can get an extra PACMAN at 10000,20000,50000, 100000,200000,500000 pts and so on ...(if you reach 500000 points congratulations ! my actual high-score is 229010 pts)

The value of the fruit bonus are as follows:

BOARD #	FRUIT	VALUE (pts)
1	Cherry	100
2	Grapes	200
3	Apricot	300
4	Apple	500
5	Pear	1000
6	Orange	2000
7	Strawberry	3000
8	Banana	5000
9	Melon	10000
10	random type	random value between
11	" "	5000,10000,20000,
12	" "	30000,50000)

price : (including a competition card - see p.67)
audio : 1100 Bfr DCR : 1250 Bfr

COLOFON

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

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.
Redactie en software bibliotheek

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

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

Lidgelden / Subscriptions

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

Voor Nederland :

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

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

	audio	DCR
Games 12	☐ 750	☐ 900
Math'Fun	☐ 1000	☐ 1150
Bits & Bytes	☐ 750	☐ 900
Character Generator	☐ 1750	☐ 1900
PAC-MAN	☐ 1100	☐ 1250 (including competition-card)
FWP	☐ 2000	☐ 2150

☐ FWP : nederlandse handleiding
☐ FWP : english manual

FWP-update	☐ 1000	☐ 1150 (I include original cover of WP/DAINATEXT)
DAYLAXIANS	☐ 2100	☐ 2100
PUZZLY	☐ 2100	☐ 2100
DUEL	☐ 2100	☐ 2100
C.L.I.O.	☐ 3000	☐ 3000

☐ GESTRUCTUREERD PROGRAMMEREN MET DAI BASIC : 1100

TOTAL : .....
all prices in Bfr.

signature : ..... date : .....

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	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	VS	/	?	O	←	o	DEL

Herselt, februari '84

Beste Leden,

Op de omslag van dit nummer kan U merken dat er weer heel wat nieuwe software ter beschikking is. We kondigen twee eigen producties aan en 4 pakketten uit Frankrijk, verdeeld door DIALOGUE-informatique. De eigen producties :1/ FWP door G.Gruiters : een sublieme tekst-verwerker die niet moet onderdoen voor vele professionele pakketten. Tot 1/5/84 vorzien wij de mogelijkheid om tegen voordeelprijs van WP of DAINATEXT over te stappen op FWP, zie pagina 10. 2/ PAC-MAN door M.Meyer : Door de uitstekende grafieken (MODE 5) en klank-toepassingen een spel dat kan zorgen voor vele uren spanning en competitie in de huiskamer. Tot en met 21 april '84 wordt iedere PAC-MAN cassette geleverd met competitie-kaart. Hiermee kan U op onze bijeenkomst een van de prijzen in de wacht slepen : een RGB-monitor (aangeboden door INDATA), een OKI matrix-printer (aangeboden door MIKROSHOP HAGELAND) en verschillende software-pakketten. Train uzelf, drijf de highscore de hoogte in... De Franse pakketten : PUZZLY,DAYLAXIANS,DUEL en C.L.I.O. . : De prijzen liggen een stuk hoger dan we gewoon zijn, maar de kwaliteit is uitstekend. Deze pakketten worden geleverd met franse,engelse en nederlandse handleiding. Op de gekleurde bladzijden vindt U de complete inhoud van de DAInamic jaargangen 1980 tot 1983. Met de gekende precisie heeft Jan Boerrigter al die informatie in rubrieken gecatalogeerd zodat opzoeken in DAInamic een stuk makkelijker wordt. Knap werk Jan ! Vanuit Zwitserland vertelt A.Meystre hoe hij het floppy-probleem heeft opgelost : de benadering vindt U op bladzijden 41 - 59. Vanwege de uitstekend toegelichte SOURCE-code vinden we hierin waarschijnlijk ook tips en oplossingen voor andere mlp-projecten. Wegens tijdsgebrek hebben we het testrapport over KEN-DOS naar een volgend nummer moeten verschuiven. Wie kan er zorgen voor een bespreking van de INDATA-floppy systemen ? Mogelijk kunnen we dan de prestaties op een rijtje zetten en vergelijken. We ronden af met de aankondiging van een nederlands handboek-cursus "GESTRUCTUREERD PROGRAMMEREN MET DAI BASIC " door Bruno van Rompaey (zie p.6), waarschijnlijk kunnen we op 21 april speciale verzamelkaften voor de DAInamic-jaargangen aanbieden.

veel leesgenot, tot 21 april

Dear members,

As you can see on the colourful cover of this issue, there is a lot of new software available. Two own productions and four from DIALOGUE-informatique. Own productions :1/ FWP by G.Gruiters : A fantastic text-processor package, with all the features of a true professional program. Till 1/5/84 you can update from WP or DAINATEXT to FWP at a special reduction-price, see p.10. 2/ PAC-MAN by M.Meyer : the well kown game on your DAI with splendid color and sound-effects : hours of fun and competition for the whole family. Till 21 april '84 you get a competition card with each PAC-MAN cassette . Train yourself and win one of the fantastic prices on our meeting: - a RGB-monitor (offered by INDATA), a OKI-matrix printer (offered by MIKROSHOP HAGELAND) and many software packages. The 'french' programs : PUZZLY , DAYLAXIANS , DUEL and C.L.I.O. : Following our club-standards, the prices are rather high, but the qaulity of the programs is extremely good ! These programs are supplied with french, dutch and english manual . On the colored pages you find the complete contents of DAInamic 1980-1983. Jan Boerrigter did the job : finding an article in one of the old issues will be a lot more easier now ! From Suisse, A.Meystre tells us about his floppy-solution : see p. 41 - 59. Due to limit of time, we cannot bring the KEN-DOS report in this issue. If somebody can take care of a report on INDATA floppy systems, we could compare both systems in the next issue. We are preparing special binders for the DAInamic issues , maybe they will be available on our meeting.

see you on 21 april ?

W.Hermans

COLOFON

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

DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.
Redactie en software bibliotheek

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

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

Lidgelden / Subscriptions

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

Voor Nederland :

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

LSD	MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	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

DIALOG INFORMATIQUE

DAYLAXIANS

DAYLAXIANS

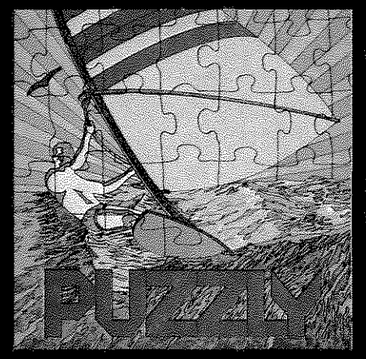
Commandeur DAI à pilote vaisseau DAYLAXIANS

Bien reçu votre message signalant votre présence dans la nébuleuse du crabe face à une escadrille de XORS commandée par deux vaisseaux «AMIRAL».
Autorisons l'action de vos rayons désintégrateurs pour parer leur attaque.
Tenez bon ! d'autres vaisseaux arrivent

BONNE CHANCE... NB : Le talon d'Achille des XORS est son vaisseau «AMIRAL» en attaque. Visez le en priorité !

DIALOG INFORMATIQUE

PUZZLY



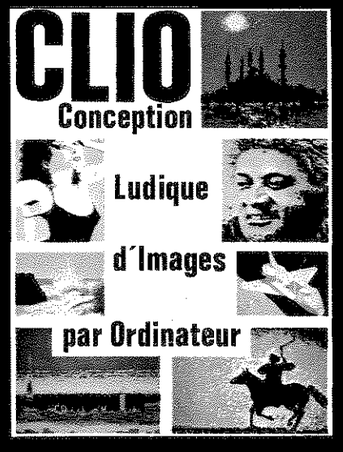
PUZZLY

Vous aimez l'art...
Vour aimez les casses têtes... ...jouez PUZZLY

PUZZLY, le premier puzzle sur micro-ordinateur.
Mais PUZZLY c'est aussi un graphisme étonnant, plusieurs niveaux de jeux, un chronométrage.
Quant aux images, sa vidéothèque ne cesse de s'agrandir.

Ce coffret contient 4 puzzles.

C.L.I.O.



C.L.I.O.

Le but de ce programme est de vous aider à réaliser vos propres images sur un écran de télévision avec un micro-ordinateur DAI.
Pour cela C.L.I.O dispose de plusieurs fonctions :
- «ZOOM» vous permettra d'agrandir de 4 ou 16 fois une partie de l'écran pour mieux définir votre dessin.

C.L.I.O. vous permet également :
- De modifier une couleur parmi les 4 ou 16 sélectionnées sur une partie ou la totalité de l'écran
- D'introduire des formes géométriques (carré, rectangle, rond, cercle, droite).
- De déplacer ou dupliquer une partie de l'écran.
- De colorier une partie de dessin, etc...

Toutes ces commandes ont une saisie de paramètre très aisée.
Elles s'effectuent par l'intermédiaire d'une manette de jeu (devenue votre pinceau) et par des touches spéciales sur le clavier.
Vous créerez ainsi vos propres dessins, qui pourront être mis sur bande magnétique et relus selon vos désirs.

DIALOG INFORMATIQUE

DUEL

DUEL

Erreur dans la programmation du bus spacio-temporel : Ce n'est pas une promenade dans les champs de fleurs qui vous attend mais un château fort fantastique.
Des tortues géantes, des rats, des araignées d'une taille démesurée, ainsi que d'autres bêtes terrifiantes le hantent.
Mais attention aux gardiens du château : L'homme à la massue cloutée dit «le chauve» et son compagnon «le prince du désert» avec son arbre tranchant sont là pour vous «raccourcir» la vie.

Une seule solution : COMBATTRE ces monstres venus d'une autre mythologie.