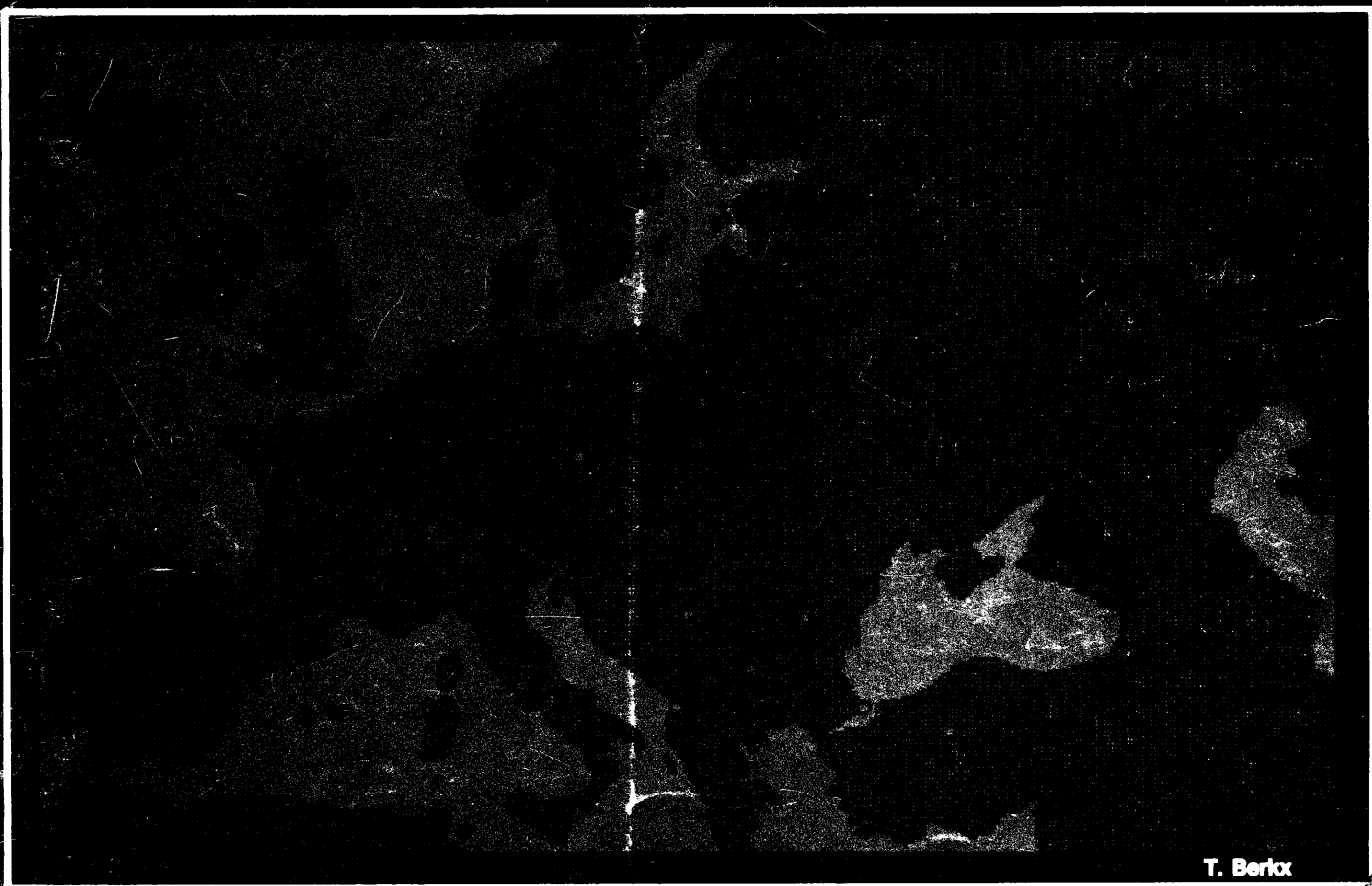


tweemaandelijks tijdschrift september - oktober 1983



T. Berckx

7

national computer users club

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

Internationaal

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

Vanaf 1 nov. '83 :

Mottaart 20

3170 Herselt

Tel. 014/54 59 74

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

Voor Nederland :

GIRO : 4083817

t.n.v. J.F. van Dunne'

Hoflaan 70

3062 JJ ROTTERDAM

Tel. : (010) 144802

Generale Bankmaatschappij Leuven

nr. 230-0045353-74

Inzendingen : Games & Strategy

Frank Druijff

's Gravendijkwal 5A

NL 3021 EA Rotterdam

Nederland

tel. : 010/25.42.75

**DAI NAMIC**

PERSONAL COMPUTER USERS CLUB

4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	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

	MSD	0	1	2	3	4	5	6	7
	LSD	000	001	010	011	100	101	110	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

Beste leden,

Als alles verloopt zoals gepland, is dit het laatste nummer dat in Westmeerbeek samengesteld wordt. De redactie, in casu ondergetekende, wordt momenteel namelijk belast met verbouwing en verhuizing. Waarschijnlijk hebben sommigen dit reeds tot ergernis bemerkt aan de onbeantwoorde telefoonoproepen, onze excuses, wij hopen dat binnen een aantal weken de toestand alweer gestabiliseerd is. Vanaf +/- 1 november bevindt DAINamic zich op volgend adres: Mottaart 20, 3170 Herselt, en heeft de PTT ons bedacht met dit telefoonnummer: 014/545974. Een aantal feiten hebben onze clubactiviteiten de laatste weken nog bemoeilijkt: - de poststaking in België, - het enorme succes van onze verjaardagsaanbieding (er moesten dagelijks 10 to 30 pakketten de deur uit!), - het chronische gebrek aan DCR-cassettes. Bovendien schijnt PHILIPS een partij cassettes op de markt gebracht te hebben die het label "betrouwbaar" beslist niet verdienen. Deze partij was voorzien van oranje wieltjes, wat beslist niet wil zeggen dat we alle cassettes met oranje wieltjes moeten wantrouwen... Heeft U van ons DCR-cassettes ontvangen die moeilijkheden geven, dan kunnen deze steeds geruild worden. Geef ons echter enige tijd, want onze voorraad is weer totaal uitgeput! Van het floppy-front is er voorlopig nog geen nieuws te melden: noch INDATA, noch KEN-DOS hebben ons een compleet werkend systeem ter test kunnen aanbieden. Met een 200-tal DCR-cassettes in gebruik wordt het onder-tussen op de redactie erg moeilijk om in het software-bos de programma-boompjes terug te vinden: de nood aan een degelijk floppy-systeem is erg groot! Wij wensen de heren ontwerpers moed en succes (op korte termijn graag!). Als dit nummer verschijnt bevinden we ons weer in de gezellige drukte van de HCC-dagen in Utrecht, daar wordt weer bewezen dat onze hobby maatschappelijk in het licht van de schijnwerpers staat, wie had een paar jaar terug kunnen denken dat het zo'n vaart zou lopen? Met de aankondiging van GAMES COLLECTION 12 sluiten we dit voorwoord, tot een volgende keer,

#### GAMES COLLECTION 12

-----  
(Chase, shoot and hunt ...)

Duck shooting by Hans Peters  
Slangenjacht by Hendrik-Jan van Randen  
Deer Hunt by Luc Maes  
Cobra by Fred van Amerongen  
Star-hunt by Koert van Espen

price : audio : 750 Bfr , DCR : 900 Bfr  
available : 1/11/83

Dear members,

Dainamic will be on a new address from 1st of november : Mottaart 20, 3170 Herselt, the new phone number will be : 014/545974. It seems that PHILIPS has produced a lot of DCR-cassetts that are not 100 % reliable, the cassettes have orange wheels. If you have troubles whit tapes you got from us, please return for exchange ! INDATA-floppy system is not ready, KEN-DOS is not ready : will there never come an end to our growing collection of DCR-cassettes ? Above you can find the contents of GAMES COLLECTION 12 .

W. Hermans

283	Remark	Redactie
284	Bladwijzer - contents	
285	Programmeertechnieken	F.Druijff
290	DAInamic info France	C.Dufour
294	Printers : STAR 510	
295	EPSON RX-80	
296	EPSON FX-80	
297	CENTRONICS 739	
298	THE BASIC MONITOR	J.Boerrigter
303	Translations UK	UK - D.Atherton
* 311	ROM INDEX	C.Hards
* 317	SFGT-NOTE-POKES	M.Hermans
* 318	Advertentie	Mikroshop Hageland
319	Real circles	G.Doumont
320	Erratum schematics	J.Boerrigter
321	Convert to IMP INT	N.Looije
322	Cursus Microprocessoren part 2	A.Beuckelaers
327	HELP hints & tips	UK - D.Atherton
330	High Speed Data Loader	H.Kop / H.Rison
333	Catalogus voor Nederland	J. van Dunne
334	Unidata	T.Groeneveld
335	Cursus DAI-DCE Dirksen	Dirksen Opleidingen
343	Micro Fast Graphics	R.Tjoews
346	DAI Video & Graphics	T.Mikulic

## DAInamic subscription rates :

Benelux	:	900 Bfr
Europe	:	1000 Bfr
Outside Europe	:	1400 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

# PROGRAMMEERTECHNIKEN

Half augustus kreeg ik een band binnen van Koert van Espen die mij zeer interesseerde. De band was keurig voorzien van de testarray's om inlezen te vergemakkelijken en het programma zelf stond er ook meermalen op, keurig !

In de begeleidende brief vertelde Koert de inhoud van de band en verklaarde dat hij STARBUILDER volledig zelf had verzonnen. Dit wekte vanzelfsprekend mijn belangstelling en ik laadde het programma dan ook snel in. Het bleek een eenvoudig maar aardig actiespel te zijn. Een programma dat, hoewel het vrij klein is, toch heel wat te bieden heeft. De bestudering van de listing overtuigde mij ervan dat dit inderdaad origineel werk van Koert is. Er zaten weliswaar wat 'foutjes' in, maar het geheel en bepaalde details toonden inzicht. Mogelijke problemen werden onderkend en de oplossing daarvan meestal gevonden. Aan de andere kant waren de foutjes en aanpak soms zo illustratief voor de gedachtengang van vele programmeurs (en echt niet alleen beginners!) dat ik besloot Koert op te bellen om hem te vragen of ik zijn programma mocht gaan gebruiken als basis voor een van mijn artikelen. Koert stemde hiermee in wat ik niet alleen zeer waardeer maar ook bewonder. Hij weet dat ik in zijn programma van alles ga bekritisieren en dat dan nog eens ga laten publiceren ook. Kritiek zeker indien die opbouwend bedoeld is kan prettig zijn maar om ten overstaan van alle DAInamic-lezers (ik hoop tenminste dat ze het lezen) je te laten bekritisieren getuigt van moed.

Ik hoop dan ook dat anderen zich de moeite willen getroosten om Koerts versie te bestuderen en dan zelf eens een lijstje te maken met foutjes die zij in dit programma zien of menen te zien en dan pas verder te lezen.

Daarnaast zijn er natuurlijk ook nog smaakgevoelige punten en mogelijk zelfs punten die ik over het hoofd heb gezien. Zou ik inderdaad iets belangrijks over het hoofd hebben gezien; laat mij het dan weten, ik kan er in de toekomst rekening mee houden. En als ik in een bui even moedig als Koert ben geef ik die onvolkomenheid nog toe ook.

Het oorspronkelijke programma van Koert :

```
10 REM Program by Koert Van Espen Date:1983 08
20 GOSUB 10000
30 PRINT "Do you want to play - without red bricks (type '1')?"
40 PRINT " - with red bricks (type '2')?"
50 G! =GETC:IF G!<>49.0 AND G!<>50.0 GOTO 50
100 FOR B=1 TO 3
110 PRINT CHR$(12):MODE 2A
120 CURSOR 0,2:PRINT "SCORE: ";S:CURSOR 30,2:FOR W=1 TO 3-B:IF B<>
3 THEN PRINT CHR$(255);" ";
130 NEXT
140 DRAW 0,0 XMAX,0 3:DRAW 0,YMAX-14 XMAX,YMAX 3:DRAW 0,0 0,YMAX-14
3:DRAW XMAX,0 XMAX,YMAX-14 3
150 X=XMAX/2:Y=YMAX/2
160 DOT X,Y 5
170 E=30:R=15:FOR K=1 TO G!-48
180 FOR O=1 TO E:XX=INT(RND(2.0*X-2.0)+1.0):YY=INT(RND(2.0*Y-16.0)+1
.0):IF SCRN(XX,YY)=15 OR SCRN(XX,YY)=3 THEN O=O-1:NEXT
190 DOT XX,YY R:NEXT:E=15:R=3:NEXT
200 ENVELOPE 0 15,200;0,200;:IF T=0 GOTO 1000
300 H=SCRN(X,Y):IF H=3 GOTO 500
310 IF H=15 THEN S=S+100:T=T+1:SOUND 0 0 15 0 4545
320 IF T=30 GOTO 900:DOT X,Y 3:CURSOR 8,2:PRINT S;" ":SOUND 0 0 15 1 36363
330 GG=GETC:IF GG=0 THEN 1100:G=GG:ON GG-15 GOTO 2000,3000,4000,5000
500 SOUND 0 1 15 0 64516:ENVELOPE 1 15,20;10,20;5,10;:WAIT TIME 100:SOUND OFF
510 G=0:GG=0:T=0:NEXT
520 PRINT "Again?(Y/N)"
```

```

530  A=GETC:IF A<>89 AND A<>78 THEN 530:IF A=78 THEN END
800  G=0:GG=0:T=0:S=0:GOTO 100
900  G=0:GG=0:T=0:S=S+5000:GOTO 110
1000 G=GETC:IF G=0 GOTO 1000
1100 S=S-1:ON G-15 GOTO 2000,3000,4000,5000
2000 Y=Y+1:GOTO 300
3000 Y=Y-1:GOTO 300
4000 X=X-1:GOTO 300
5000 X=X+1:GOTO 300
10000 PRINT CHR$(12):COLORT 7 0 0 0:COLORG 0 3 5 15:MODE 0
10010 CURSOR 19,18:PRINT "S T A R B U I L D E R "
10020 CURSOR 19,17:PRINT "- - - - -"
10030 CURSOR 19,15:PRINT " KOERT VAN ESPEN"
10040 PRINT :PRINT :PRINT :PRINT "DO YOU WANT INSTRUCTIONS? (Y/N)"
10050 G!=GETC:IF G!<>89.0 AND G!<>78.0 THEN 10050:IF G!=78.0 THEN RETURN
10060 PRINT CHR$(12):PRINT "Try to catch the white stars as fast as possible,"
10070 PRINT "Use the cursorkeys to move your ship.":RETURN

```

Zoals U kunt zien is het een vrij klein programma (40 regels) en als U de moeite neemt om het in te tikken kunt U zien dat het bijna altijd goed werkt.

De enige fout in de werking (HEEFT U DIE GEVONDEN ????) komt slechts zelden aan het licht. Bijna altijd zult U normaal kunnen spelen.

Bij het bestuderen van de listing zal U zijn opgevallen dat als regelnummers alleen tientallen worden gebruikt; een uitstekende gewoonte, die mij vertelt dat niet alleen bij het programmeren zelf is nagedacht, maar ook nadien het programma nog eens bekeken werd voor het werd ingestuurd. Dit laatste ontbreekt bij vele inzendingen. Velen zijn tevreden zodra het programma werkt. Maar ik vind dat na het programmeren zelf altijd nog een of twee stappen horen. De beroepsprogrammeur zal na de laatste regel nog een handleiding / bedieningsaanwijzing of iets dergelijks moeten maken OOK als het programma zelf uitleg geeft. Ik zie bijvoorbeeld mensen aan de slag willen gaan met DAInatext. Op de eerste uitleg-pagina staat dat je met [SHIFT]+[RETURN] terug kunt komen in het menu, maar dat is niet leesbaar als je met de tekst bezig bent. Ook de aanwijzing dat je niet eerst [SHIFT] en dan [RETURN] moet drukken maar tegelijkertijd met eerst de [SHIFT] en dan pas de [RETURN] erbij ontbreekt. De amateurprogrammeur zal zijn programma in een vorm moeten vermaken dat hij er mee voor de dag kan komen. Het is toch uw liefhebberij en dan is het toch fijn iets moois af te leveren.

Het eerste is ook voor amateurs aan te bevelen, zeker als hun programma's voor anderen bedoeld zijn. Het tweede zullen beroepsmensen zich ook moeten aantrekken al kunnen die dit om economische redenen (een volgende opdracht wacht) achterwege laten. Overigens zou moeten gelden dat juist bij beroepsmensen de structurele opzet al zo groot is dat er weinig meer verbeterd kan worden.

Zo, en nu komen we dan bij de kritiek aan.

Zoals reeds eerder vermeld ben ik een tegenstander van het vragen of de gebruiker instructies nodig heeft als de uitleg zo kort is als hier. De intro / uitleg staat correct aan het eind van de listing met logische regelnummers. Maar deze subroutine heeft twee RETURNS en dat vind ik voor de structuur minder geslaagd. Het geheel kan trouwens ook goed met GOTOs gedaan worden. We gaan alleen vanaf regel 20 naar regel 10000 en keren dan altijd terug naar regel 100. Willen we dan later nog een CLEAR opnemen kan dat dan probleemloos. Met CLEAR is het programma trouwens beter: nu loopt het stuk na een CLEAR4 voor de RUN. CLEAR 256 is bv goed. De regels 30,40 & 50 vind ik bij de uitleg horen, daar de keuze maar eenmaal in het programma wordt gemaakt.

Over regel 50 nog het volgende : beseft U dat

```
50 G=GETC:IF G=49 OR G=50 GOTO 100:GOTO 50    hetzelfde doet ?
```

Maar pas op, Koerts oplossing laat U vrij een regel tussen regel 50 en regel 100 in te voegen en de tweede methode niet direct, al is dat aan te passen. In beide mogelijkheden wordt echter door de AND respectievelijk OR extra tijd gebruikt. Hier misschien niet zo belangrijk maar het toetsenbord zal niet zo fijn reageren.

Vandaar mijn voorstel :

```
50 G=GETC:IF G<49 GOTO 50:IF G>50 GOTO 50
```

Hetzelfde recept kan ook worden toegepast op regel 530.

Koert wist vermoedelijk niet en daarin is hij zeker niet de enige dat bij een IF AND of een IF OR combinatie de DAI altijd beide mogelijkheden bekijkt en dan combineert alvorens actie te ondernemen. Beter zou zijn te stoppen als het toch al fout is. Dit is zo bij een OR als al aan de eerste voorwaarde wordt voldaan en bij een AND als de eerste al niet voldoet.

Structureel is de AND en OR te prefereren omdat anderen dan sneller kunnen begrijpen hoe en waarop getest wordt maar uit oogpunt van snelheid (bij DAI) niet.

Toch maak ik de ontwerpers van de DAI hier geen verwijt. De tijdwinst die soms geboekt kan worden gaat weer verloren door structuuronderzoek, terwijl nu de programmeur het kan beïnvloeden.

Is de snelheid belangrijk kunnen we meerdere IFs gebruiken en de volgorde juist zo maken dat er zo min mogelijk getest hoeft te worden.

De score wordt bijgehouden op regel 320; achter S wordt nog een tweetal spaties afgedrukt dit is juist, daar anders soms de eindcijfers van een vorig maal kunnen blijven staan. Bedenkt U zelf eens waarom twee spaties nodig zijn en in welk geval een spatie niet voldoende is. Achter deze " " zou ik echter nog graag een ';' willen zien om het zenuwachtige cursorgedrag te beperken. Of beter nog de cursor onzichtbaar maken door middel van POKE #75,32.

Over regel 180 valt het volgende op te merken X+X is sneller dan 2.0\*X. Er is juist gebruik gemaakt van floating point getallen 1.0 en 2.0 omdat RND daarmee werkt en nu dus sneller is dan met integers. De INT is volledig overbodig en vertraagt alleen maar omdat er toch een toewijzing aan een integervariabele is.

En nu de fout. Koert voorzag terecht dat het mogelijk zou zijn dat een 'ster' op een reeds geplaatste ster terecht zou komen en dan zouden er maar 29 (of nog minder) in plaats van 30 sterren zijn en kunt U nooit 30 sterren 'opeten' om een nieuw veld te krijgen.

Koert loste dit schijnbaar goed op door te testen of de gekozen plaats leeg is. Is dit niet zo dan verlaagt hij de loopcounter O met 1. Dit is logisch gezien een uitstekende oplossing alleen werkt het op de DAI niet !! Aan het begin van elke FOR-NEXT loop wordt het aantal malen berekend dat de loop doorlopen moet worden. Het tussentijds veranderen van de loopvariabele O (een dummy) heeft dan ook geen invloed meer op het aantal malen dat de loop doorlopen wordt.

Ook heb ik structurele bezwaren tegen een FOR waar twee NEXT'n bijhoren.

Ook had de test zelf beter gekund, niet alleen de constructie met OR die hiervoor al besproken is maar ook het tweemaal laten berekenen van de SCRN(XX,YY) kost tijd.

En het is niet nodig: beter is HULP=SCRN(XX,YY):IF HULP=15 OR HULP=3 THEN.....

Nog beter IF SCRN(XX,YY)<>0 THEN ....maar dit is water naar de zee dragen, de fout wordt nu mooier en sneller gemaakt. Om de loopvariabele te verminderen is hier zinloos. Hoe lossen we het op ?

Er zijn meerdere methoden voorhanden; ik noem er twee :

- I ) we houden bij hoeveel sterren we plaatsen. Zijn het er bijvoorbeeld maar 28 dan behoeven we ook maar 28 sterren te pakken voor een nieuw veld. In plaats van letterlijk te testen of  $SCRN(XX,YY)=0$  is kunnen we de teller bijhouden door  $TELLER=TELLER+1-SGN(SCRN(XX,YY))$
- II) we kiezen een ander punt als het gekozen punt al bezet was en om tijd te winnen kiezen we alleen een nieuwe YY-waarde.
- ```

200 FOR O=1 TO 30:XX=RND(X)
210 YY=RND(Y):IF SCR(N(XX,YY)<>0 GOTO 210
220 NEXT
```

Ook de methode van Koert om zowel de witte als de rode punten met dezelfde loop neer te zetten overtuigd mij van zijn programmeercapaciteiten. Zonder meer origineel maar naar mijn smaak hier de leesbaarheid van het programma zo beïnvloedend, dat ik liever twee losse FOR-NEXT'n zie om de punten te plaatsen. Het is nauwelijks meer werk maar wel duidelijker voor anderen. Maar weet dat Koerts oplossing juist is en alleen mijn smaak zegt gekunsteld en niet kunstig. Een ander argument tegen Koerts oplossing is dat nu bij de eerste doorgang van de loop altijd nodeloos op kleur 3 wordt getest die dan nog niet aanwezig is. Maar genoeg over regel 180, het is weliswaar de interressantste regel van het programma maar zo vullen we tien DAInamics met alleen dit programma.

Bij regel 800 worden G,GG en T op 0 gezet, waarna naar het begin van het programma wordt gesprongen. Het was beter geweest deze initialisering dan daar aan het begin te doen. De nulmaking (wat een woord) kan dan ook op regel 510 en regel 900 achterwege blijven.

Na deze verandering verhoogt regel 900 alleen nog maar de score S met 5000 en gaat dan naar 110. We kunnen echter alleen vanuit regel 320 in regel 900 komen en dus is het beter daar (regel 302) de verhoging te doen en regel 900 te verwijderen. Regel 800 kan na een kleine verandering ook weggelaten worden.

De ENVELOPE's veranderen nooit en kunnen dus beter bij de initialisatie staan.

De opbouw van het veld voor het overzicht over het programma in een apart blok aan het eind gezet. Dit blok laten beginnen met een aantal GETC's om keybounce tegen te gaan. In de oorspronkelijk versie was je wel eens al dood nog voor je iets gedaan had. Beter gezegd je had wel iets gedaan maar dat was bij de vorige beurt.

De GOTO in regel 200 beviel mij ook niet. De richtingsbepaling is essentieel en dus wil ik die voor in het programma. Het moet in feite het eerste zijn dat we zien afgezien van initialiseringszaken.

De methode die Koert gebruikt om de blokjes in beeld te krijgen die aangeven hoeveel ronden er nog zijn is goed en origineel. (zie de regels 120 en 130)

Toch vond ik het gekunsteld. 'Vreemde' methodes hebben dat nu eenmaal. Als het als volgt gedaan zou zijn opgelost :

```

40 IF B=1 THEN PRINT CHR$(255)+" "+CHR$(255)
50 IF B=2 THEN PRINT CHR$(255)+" "
60 IF B=3 THEN PRINT " "
```

zou, iedereen het zeer snel duidelijk zijn wat er gebeurt. Regel 60 kan zelfs na een CHR\$(12) weggelaten worden. Hij heeft echter een eigen methode gezocht en dat is zeker te waarderen. Zelf ben ik toen ook een aparte oplossing gaan zoeken.

```

40 PRINT MID$(CHR$(255)+" "+CHR$(255)+" ",B+B,3) en de B dan van 0 tot 2
en niet van 1 tot 3. Ik wilde echter graag de CHR$(255) vermijden. Ik heb toen het volgende gedaan. De desbetreffende regel regelnummer 1 geven en de string die de MID$ gebruikt vervangen door "AAAAAAA". Toen naar utility gegaan en op #29F, #2A0 gekeken waar de textbuffer begint. Zonder CLEAR zal dit #3EC zijn. Het stuk na #3EC bekeken en de AAAAAA's opgezocht. (code 41) Vervolgens deze 41's vervangen door FF 20 FF 20 20 20. Voor degenen die niet zo thuis zijn in ASCII-codes FF is volle blok en 20 is spatie. Terug naar BASIC en toen in EDIT het regelnummer weer aangepast. Moet U ook eens doen ! 'Vreemde' dingen als U LIST en / of EDIT doet met deze regel.
```



Regel 330 stond mij ook niet aan. Het is logisch om de GETC in een andere variabele te zetten om zo de oorspronkelijke richting te bewaren, maar waarom moest bij richtingverandering geen strafpunt gegeven worden ? De ON GG=15 GOTO .... dus vervangen door een GOTO 1100.

En ondanks dat het spel snel is toch ook principieel kijken naar de gevolgde logica. In regel 320 staat een test die onzinnig tenzij het laatste punt een wit punt was. De gehele regel hoort dus thuis achteraan in regel 310.

In regel 330 nu nog IF GG=0 veranderd in IF GG<16 omdat dan 'TAB' en 'RETURN' geen problemen meer kunnen geven.

Door verder het programma iets andere volgorde te geven was het mogelijk regel 330 en regel 1100 te combineren. Ik vermoed dat Koert er aan heeft zitten denken maar er niet uitgekomen is.

Verder heb ik nog een paar schoonheidsfoutjes verbeterd. Het omzetten van floating point variabelen in integers (regel 50 en regel 10050) is daar een voorbeeld van.

Ik heb programma tot slot gere-numberd en een naamswijziging doen ondergaan.

Dit laatste vind ik zelf een grote ingreep daarom nu de argumenten die mij er toe brachten het toch te doen. Games collection nummer 12 is een jachtcollectie geworden en nu leek mij een naam als STAR-HUNTER beter. Ten eerste omdat dat mijns inziens ook beter de bedoeling van het spel aangeeft en ten tweede omdat die naam duidelijk thuishoort in een collectie als nummer 12.

De gewijzigde versie laat ik nu expres niet zien. U heeft de oorspronkelijke versie en met het doornemen van dit artikel kunt U de 'nieuwe versie verkrijgen.

Ook het aanschaffen van Games Collection 12 verschaft U de nieuwe versie.

Nogmaals Koert hartelijk dank zonder jou was dit artikel niet ontstaan.

Frank Druijff

## ERRATUM DAInamic Debugging Tool

```
10 REM ERRATUM DAInamic Debugging Tool (TK 5)
20 REM Please adapt lines 1350/1360/1370 as follows :
30 REM -----
1350 GOSUB 51000:REM --- REGISTERS TO VARIABLES---
1360 IF PC=POK(8) OR PC=POK(8)+1 THEN PC=PCNEXT
1370 IF RSTCALLFLAG=1 AND SPOLD-SP=2 THEN POKE MLPSTACKPTR+12,PCNEXT
MOD 256:POKE MLPSTACKPTR+13,PCNEXT SHR 8:GOSUB 51000
```

# DAInamic INFO

## LIGNES D'EXTENSION

Le moniteur du DAI permet l'utilisation de lignes d'extensions. Ceci est possible jusqu'à 3 lignes supplémentaires. Chaque ligne supplémentaire commence par un C et contient 6 espaces à son début.

Mr Markus SIGG a conçu un programme en langage machine (mlp) pour manipuler ces lignes. Il permet l'utilisation de plus de 3 lignes supplémentaires, efface le C et permet de mettre plus ou moins de blancs en début de ligne.

```
INIT    PUSH H
        LXI H    START    ;Ad. du programme
        SHLD   6CH      ;charge le vecteur de RST 5
        POP H
        RET

;
START   PUSH H
        PUSH PSW      ;A=dernier car. entre
        CPI     0DH    ;est-ce un CR ?
        JZ      READY  ;si oui -Fin
        LDA     72H    ;Pos. Curseur (Poids faible)
        LHLD   7AH    ;Pos. dans ligne
        CMP L      ;Fin de ligne atteint ?
        JNZ     READY  ;Si non -Fin
        CALL   0DD5EH  ;Imprime un CR
SPACE   MVI A     20H  ;espace en ASCII
        MVI H     6H   ;nombre d'espaces
OUTC    RST 5      ;sortie sur ecran
        DB      3H
        DCR H
        JNZ     OUTC   ;espace suivant
READY   POP PSW
        POP H
        JMP     0C6FDH ;RST 5 normal

END
```

La routine INIT doit être utilisée une fois au début du programme. Elle remplace l'adresse du vecteur d'interruption RST 5 en #006C/D par l'adresse de ce programme. (Ceci peut être fait sous utilitaire en faisant UT , V5=START)

Le reste du programme sera utilisé par le moniteur à chaque interruption RST 5 (Gestion de l'écran). Si vous ne voulez pas d'espace en début de ligne, sautez la partie SPACE.

On peut modifier le nombre d'espaces en changeant le contenu de H dans la routine SPACE.

À la fin du programme on retourne à l'adresse normale de RST5:#C6FD. Ce programme n'a pas d'influence sur la sortie RS 232.

## DAI RESTART ROUTINES

Le basic du DAI et le système d'exploitation interne sont conçus de telle façon qu'il est difficile de les modifier. L'utilisateur ne peut définir que les Input/Output (Cassette, Clavier, RS 232C, bus DCE, Gestion de la mémoire).

Vous pouvez donc écrire vos propres routines (comme le TOS (Tape Operating System) du MEMOCOM MDCR qui vérifie les entrées au clavier. Dès que (RETURN) est pressée, il regarde s'il y a des commandes DCR.)

Cette méthode fonctionne tant que le basic ne reprend pas la main et dans un programme, vous devez utiliser l'instruction CALLM pour travailler avec le MDCR. Mais il y a d'autres moyens d'étendre le basic et le système d'exploitation interne. Un de ceux-ci par exemple, est la méthode que le DAI utilise pour "switcher" les banques de ROM. Dans les ROM du DAI, les adresses E000-EFFF existent 4 fois (Dans les banques 0, 1, 2, 3). Le numéro de la banque sélectionnée est déterminé par les deux bits les plus significatifs des adresses #FD06 et #40. En basic, la routine ressemblerait à ceci:  
 POKE #FD06, (PEEK(#FD06) IAND #3F) IOR (NUMEROBANQUE SHL 6)  
 La même chose pour l'adresse #40. N'essayez pas ceci car le basic utilise la banque 0!

De façon interne, le DAI utilise la même méthode pour "switcher" les banques chaque fois qu'un appel est effectué aux banques 1, 2 ou 3. Ceci ne peut se faire d'une façon directe par l'instruction CALL car un CALL ne switche pas les banques. Il existe une instruction spéciale dans le 8080: RST X (X=0-7) qui est suivie d'un byte-donnée. RST X est une instruction occupant un byte et similaire à CALL.

On utilise les RST X suivants Pour connecter les banques:

|       |                     |                      |
|-------|---------------------|----------------------|
| RST 4 | switche la banque 1 | Math & routines Son  |
| RST 5 | switche la banque 2 | Gestion d'écran/EDIT |
| RST 1 | switche la banque 3 | Encodage/UTILITAIRE  |

TOUS LES NOMBRES SONT EN NOTATION HEXADECIMALE!

Par exemple, la routine pour imprimer un caractère sur l'écran est RST 5, DATA 3 (EF 03). C'est un appel à l'adresse #E003 de la banque 2. Le registre A doit contenir le code ASCII du caractère. RST 5 est un appel à l'adresse #28 (= 8\*5 = 40D).

A #28, les instructions suivantes préparent l'accès à la banque:

|    |           |          |
|----|-----------|----------|
| 28 | NOP       | 00       |
| 29 | PUSH H    | E1       |
| 2A | LHLD 006C | 2A 6C 00 |
| 2D | PCHL      | 59       |
| 2E | NOP       | 00       |
| 2F | NOP       | 00       |

A l'adresse #6C/#6D se trouve FD C6, un saut sera donc effectué à #C6FD avec le contenu initial de HL placé au dessus de la pile. Comme tout ceci se trouve en RAM, vous pouvez modifier le contenu des adresses #6C/#6D ou #2B. Ces adresses peuvent être modifiées par la commande V (après avoir tapé UT). Ex.:  
 V5 C6FD-300 (flèche gauche). Ceci force le RST 5 à continuer à l'adresse #300 ou à l'endroit où se trouve votre programme. Un bon exemple est le programme de conversion APPLE/ATARI qui fut publié dans une précédente newsletter.

En utilisant cette méthode, vous pouvez manipuler toutes les fonctions des banques 1 à 3 et même de la banque 0. RST 5, DATA 3 par exemple est très utile. Cette routine est utilisée pour tout ce qui doit être affiché sur l'écran.

Cette routine est utilisée par PRINT, LIST et tous les messages du DAI. Ainsi,

nous pouvons changer toutes ces routines en modifiant:

- 1) les contenus des registres A-L du 8080.
- 2) le pointeur de pile.
- 3) l'adresse de retour.
- 4) le byte-donnée après RST X.

Pour ceci nous devons connaître les contenus des registres, l'objet du RST X (la valeur du byte-donnée) ou la valeur du pointeur de pile au moment du RST X. (Ceux qui ne connaissent pas le firmware du DAI peuvent se référer aux routines publiées dans des précédentes newsletters ou au "Firmware Manual" de J. Boerrigter.) Le principal est de vérifier le byte-donnée parce qu'il détermine quelle routine est appelée, et le contenu des registres.

Comme exemple, nous prendrons RST 5, DATA 3 et supposons que notre routine commence à #300. Comment trouver le byte-donnée? Lorsque le RST X est exécuté, l'adresse de retour est placée dans la pile et cette adresse est celle du byte-donnée car elle pointe directement après l'instruction RST X. Avant que le saut ne soit fait à votre propre routine, le registre double HL sera placé dans la pile (V. l'exemple sur la première page).

Dans notre exemple, le registre A est vérifié après que nous ayons trouvé que le byte-donnée est 3. Si le caractère est une majuscule, il sera converti en minuscule.

L'interpréteur basic ne reconnaît pas les commandes écrites en minuscules. Pour cela, nous devons changer de nouveau les commandes écrites en minuscules, en majuscules. Ceci peut être fait. L'interpréteur utilise une routine pour lire des caractères sur l'écran (RAM). Cette routine est RST 5, DATA 15.

Nous devons changer l'adresse de retour après cette routine et placer l'adresse de notre propre routine et ensuite, vérifier si le caractère reçu est une minuscule (et dans ce cas, le convertir en majuscule). De cette façon, l'interpréteur accepte les commandes en minuscules. La routine complète sera la suivante:

```

COMMANDES EN MINUSCULES
300      POP H          ;retire HL de la pile
301      XTHL          ;échange HL et le sommet de la pile
           ;l'adresse de retour est dans HL
302      PUSH PSW      ;sauve PSW
303      MOV A,M       ;A reçoit le contenu de l'adresse de retour
304      CPI 3H        ;Est-ce le byte-donnée pour imprimer
           ;un caractère à l'écran?
306      JZ MAJmin
309      CPI 15H       ;Est-ce le byte-donnée pour lire
           ;un caractère à l'écran?
30B      JZ minMAJ
30E out0 POP PSW      ;retire HL de la pile
30F out  XTHL          ;RST 5 si ce n'est pas la bonne routine
310      PUSH H
311      JMP C6FDH     ;adresse normale du RST 5

320 MAJmin LDA 118H
323      CPI FFH
325      JNZ out0
328      POP PSW      ;A reçoit de nouveau la valeur ASCII
329      CPI 'A'      ;si valeur inf. à 'A', pas d'action
32B      JC out
32E      CPI 91       ;si valeur sup. à 'Z', pas d'action
330      JNC out

```

```

333      ORI 20H      ;converti en minuscule
335      JMP out

338 minMAJ POP PSW    ;retire le PSW
339      INX H        ;l'adresse de retour est celle du byte-donnée
33A      XTHL         ;HL=HL initial
                        ;sommet de la pile=ancienne adresse de retour+1
33B      PUSH H       ;sauve HL
33C      LXI H,ret    ;adresse de retour à notre routine
33F      XTHL         ;HL=HL initial
                        ;sommet de pile=nouvelle adresse de retour
340      PUSH H       ;sommet de pile=HL initial
341      JMP C6FDH    ;continue RST 5 et retour à notre routine
344 ret   DB 15H      ;byte-donnée après l'adresse de retour
345      PUSH PSW     ;sauve les status
346      CPI 'a'      ;si inf. à 'a', pas d'action
348      JC out2
34B      CPI 123      ;si sup. à 'z', pas d'action
34D      JNC out2
350      SUI 20H      ;converti en majuscule
352 out2  XTHL        ;status dans L
353      MOV H,A      ;caractère dans H
354      XTHL
355      POP PSW
356      RET          ;retour à l'ancienne adresse de retour
                        ;après le byte-donnée
END

```

Dans cette routine la conversion n'est pas opérationnelle pendant l'entrée de commandes basic. Vous pouvez donc mixer majuscules et minuscules. Tous les listings même en UTILITY ou avec l'assembleur DNA seront en minuscules.

Parfois il n'est pas suffisant de connaître le byte-donnée ou l'adresse de retour si nous voulons changer une routine parce que la routine qui utilise d'autres routines différentes avant une instruction RST X est utilisée. Comme exemple, prenons les routines d'erreurs du DAI. Si nous voulons créer un ON ERROR GOTO routine, il y a un moment où nous pouvons appeler cette routine avant que l'exécution ne soit stoppée. Ce moment est juste avant que le message ne s'imprime. Ce message est affiché par (encore!) RST 5, DATA 3. Pour savoir si un message d'erreur est imprimé, nous devons regarder la pile et le pointeur de pile. Ceci parce que chaque fois qu'une routine différente est appelée, elle laisse une adresse de retour au sommet de la pile puis le pointeur de pile est décrémenté de deux.

Cela veut dire que nous pouvons retrouver l'adresse de retour de l'erreur quelque part dans la pile. Comme nous pouvons examiner le sommet de la pile au moyen de l'instruction DAD SP et que nous connaissons combien de niveau de sous-routines se trouvant dans la pile, nous savons toujours si l'appel initial est une routine d'erreur. Avec ces connaissances, nous allons traiter la pile comme une mémoire normale. Nous ajoutons au pointeur de pile deux fois le nombre de niveaux de sous-routines et nous obtenons l'adresse de retour au moyen de l'instruction MOV A,M. Si nous trouvons une erreur, le programme basic continue, avant qu'un message d'erreur ne s'imprime, à une ligne de programme spécifiée par l'utilisateur. Si cette ligne n'existe pas, comme les routines d'erreurs sont à ce moment rendues opérationnelles, un message "UNDEFINED LINENUMBER" sera imprimé.

En utilisant la méthode décrite ci-dessus, il est possible d'étendre le basic du DAI et le système d'exploitation. Quelques fois très facilement, quelques fois en recopiant un peu et en modifiant les routines déjà existantes en ROM. J'ai écrit un programme d'environ 500 bytes utilisant cette méthode et qui rend possible l'extension du basic à 65 nouvelles commandes entièrement compilées au format standard du basic du DAI. Ceci pour illustrer la puissance de cette méthode. J'espère que cet article vous donnera une idée de la façon d'interagir avec le basic du DAI et de faire plus avec votre DAI.

# NEW PRINTERS

## CHARACTERISTICS

- Printing speed: 100 cps (Normal character)
- Super Subscript printing
- Bit Image
- Mixture printing of various size characters in one line

## SPECIFICATIONS

|                            |                                                                                                                                                                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Printing system            | Serial impact dot matrix system                                                                                                                                                                                                                    |
| Interface                  | Parallel interface (Centronics compatible) -standard-<br>Serial interface (RS-232C, Current loop) -option-                                                                                                                                         |
| Printing composition       | Standard characters: 9 x 9 dot matrix<br>Block graphic: 6 x 6 dot matrix<br>Bit image: (7 or 8) x 480 dot matrix (single density)<br>(7 or 8) x 960 dot matrix (double density)                                                                    |
| Direction of printing      | Standard characters: Bi-directional printing (with logical seeking)<br>Bit image: Uni-directional printing                                                                                                                                         |
| Number of print characters | 80, 96, 132 characters (in case of expanded characters: 40, 48, 66 characters)                                                                                                                                                                     |
| Printing speed             | 100 characters/sec                                                                                                                                                                                                                                 |
| Character types            | 96: Standard ASCII character type<br>96: Italic ASCII character type<br>64: Special character type<br>32: Block graphic character type<br>96: Proportional ASCII character type (*)<br>32: Proportional special character type (*)<br>(*) = option |
| Character size             | 2.4 (H) x 2.0 (W) mm (in the case of 80 characters/line printing)                                                                                                                                                                                  |
| Character pitch            | 10, 12, 17 characters/inch (in the case of expanded characters; 5, 6, 8.5 characters/inch)                                                                                                                                                         |
| Line pitch                 | 1/6, 1/8, n/72, n/144 inch line feed                                                                                                                                                                                                               |

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| Paper feed system | Sprocket feed and friction feed                                           |
| Paper feed speed  | 10 lines/sec (in case of 1/6 inch line feed)                              |
| Dimensions        | 392 (W) x 315 (D) x 136 (H) mm (15.2 x 12.4 x 5.3 inches)                 |
| Weight            | Approx. 7kg (15.4 pounds)                                                 |
| Power requirement | 100V, 120V, 220V, 240V AC ±10%, 50/60Hz, approx. 120W (power consumption) |

### FORM SPECIFICATIONS

|                |                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Single sheet   | Width: 8-10 inches<br>Thickness: 0.07 - 0.1mm                                                                                                   |
| Roll paper     | Width: 8.5 inches<br>Thickness: 0.07 - 0.1mm<br>Diameter of roll: max. 5 inches                                                                 |
| Sprocket paper | Copy: 3 sheets (original + 2 sheets) no carbon paper.<br>Width: 3 - 10 inches<br>Thickness: 0.07 - 0.1mm (single paper)<br>0.28mm (3 part form) |

### INK RIBBON SPECIFICATIONS

|                      |                                                                                      |
|----------------------|--------------------------------------------------------------------------------------|
| Color                | Black single color                                                                   |
| Material of ribbon   | Nylon # 40                                                                           |
| Dimensions of ribbon | 13 (W) x 10.5 (L) mm (1/2 inch x 11.5 yards)<br>Eyelets are provided near both ends. |
| Spool standard       | Underwood's plastic type. 13mm x 50mm dia.                                           |
| Recommended product  | Type SF-02B made by Fuji Kagakushi Kogyo, Co.                                        |

## CONTROL CODE

|        |                                                                                                                  |
|--------|------------------------------------------------------------------------------------------------------------------|
| CR     | CR code causes print out of buffer.                                                                              |
| LF     | One line paper feed is done by an input of LF code after buffer data is printed.                                 |
| VT     | Feed paper to the next VT position where programmed by "ESC Pnnnn NUL" after buffer data is printed.             |
| FF     | Paper is fed to the print start line in next page by pre-set program (Header line) after buffer data is printed. |
| HT     | Moves the print head to the next TAB set position.                                                               |
| SO     | Turns on Double width character print mode.                                                                      |
| SI     | Turns on Compressed mode.                                                                                        |
| DC2    | Cancell SI-mode.                                                                                                 |
| DC4    | Cancell SO-mode.                                                                                                 |
| BEL    | Buzzer sounds for 1/4 sec. when this code is input.                                                              |
| DC1    | This code causes printer to be selected.                                                                         |
| DC3    | This code causes printer to be deselected.                                                                       |
| ESC 0  | This code changes line feed length to 1/8 inches.                                                                |
| ESC 1  | This code changes line feed length to 7/72 inches.                                                               |
| ESC 2  | This code changes line feed length to 1/6 inches.                                                                |
| ESC 3n | This code changes line feed length to n/144 inches.                                                              |
| ESC An | This code changes line feed length to n/72 inches.                                                               |
| ESC Jn | One time line feed of n/144 inches.                                                                              |
| ESC B  | ESC B1 Sets 10 CPI. 80 columns.<br>ESC B2 Sets 12 CPI. 96 columns.<br>ESC B3 Sets 17 CPI. 132 columns.           |

|               |                                                                   |
|---------------|-------------------------------------------------------------------|
| ESC W         | Double width character print mode is selected.                    |
| ESC Cn        | Sets form length to n lines.                                      |
| ESC COn       | Sets form length to n inches.                                     |
| ESC Mn        | Sets LH-Margin at n columns from left hand side in printing area. |
| ESC Qn        | Sets RH-Margin at n columns from left hand side in printing area. |
| ESC Dnn-n NUL | Sets new HT positions (Horizontal Tab).                           |
| ESC Pnn-n NUL | Sets new VT positions (Vertical Tab).                             |
| ESC -         | All data after this code is printed with under line.              |
| ESC E         | Selects EMPHASIZE print mode.                                     |
| ESC F         | Cancels EMPHASIZE print mode.                                     |
| ESC G         | Selects DOUBLE STRIKE print mode.                                 |
| ESC H         | Cancels DOUBLE STRIKE print mode.                                 |
| ESC S         | Super/Subscript mode is selected.                                 |
| ESC T         | Resets "ESC S" and "ESC U"                                        |
| ESC U         | Selects uni-direction print mode.                                 |
| ESC Rn        | Sets Header line position at n line.                              |
| ESC Nn        | Sets skip over perforation to n lines.                            |
| ESC O         | Resets skip over perforation to zero lines.                       |
| ESC 4         | This code selects character ROM set of ITALICS.                   |
| ESC 5         | This code resets standard character ROM set of ITALICS.           |

# NEW PRINTERS

## Personal Printer **RX-80**

### SPECIFICATIONS

|                            |                                                                                                             |
|----------------------------|-------------------------------------------------------------------------------------------------------------|
| Print Method .....         | Serial impact dot matrix                                                                                    |
| Print Rate .....           | 100 characters per second                                                                                   |
| Print Direction .....      | Bidirectional with logic-seeking<br>in the text mode<br>Unidirectional in the bit image<br>and graphic mode |
| Number of Pins in Head ... | 9                                                                                                           |
| Line Spacing .....         | 1/6, 1/8, 7/72 inch or<br>programmable                                                                      |

### PRINTING CHARACTERISTICS

|                           |                                                                                             |
|---------------------------|---------------------------------------------------------------------------------------------|
| Character Set .....       | Full 96-characters ASCII with<br>descenders plus 11 international<br>character sets         |
| Character Size .....      | 2.1(W) × 3.1(H)mm (Normal)                                                                  |
| Character Structure ..... | Text mode: 9 × 9<br>Bit image: 480 × 8, 640 × 8,<br>720 × 8, 960 × 8,<br>1920 × 8 dots/line |

### PRINTING SIZE

|                          | Max. Characters/Line |
|--------------------------|----------------------|
| Normal .....             | 80                   |
| Enlarged .....           | 40                   |
| Condensed .....          | 137                  |
| Condensed-Enlarged ..... | 68                   |
| Elite .....              | 96                   |
| Elite-Enlarged .....     | 48                   |

### FORMS HANDLING

|                 |                                  |
|-----------------|----------------------------------|
| Form Feed ..... | Programmable length to 127 lines |
|-----------------|----------------------------------|

### MEDIA HANDLING

|                         |                                                                                 |
|-------------------------|---------------------------------------------------------------------------------|
| Paper Feed .....        | Sprocket pin feed                                                               |
| Paper Width Range ..... | Fanfold; 4" to 10"                                                              |
| Copies .....            | One original plus two carbon<br>copies (total thickness not to<br>exceed 0.3mm) |

### INTERFACE

|                |                                 |
|----------------|---------------------------------|
| Standard ..... | Centronics-style 8 bit parallel |
| Optional ..... | IEEE 488, RS-232C, etc.         |

### INKED RIBBON

|                       |                      |
|-----------------------|----------------------|
| Color .....           | Black                |
| Type .....            | Exclusive cartridge  |
| Life Expectancy ..... | 3 million characters |

### ENVIRONMENTAL CONDITIONS

|                           |                          |
|---------------------------|--------------------------|
| Operating Temperature ... | 5 to 35° centigrade      |
| Operating Humidity .....  | 10 to 80% non-condensing |

### POWER REQUIREMENT

|                         |                                            |
|-------------------------|--------------------------------------------|
| Voltage .....           | AC 120V, 220/240V ± 10%,<br>49.5 to 60.5Hz |
| Power Consumption ..... | 70 VA max.                                 |

### PHYSICAL CHARACTERISTICS

|              |       |
|--------------|-------|
| Height ..... | 107mm |
| Width .....  | 389mm |
| Depth .....  | 303mm |
| Weight ..... | 5.1kg |

### MAJOR SOFTWARE CONTROL CODES

| Control Code | Function                                         |
|--------------|--------------------------------------------------|
| CR           | starts printing                                  |
| LF           | advances paper one line                          |
| FF           | advances paper to top of next form               |
| VT           | vertical tabulation                              |
| HT           | horizontal tabulation                            |
| SO           | sets enlarged printing mode                      |
| SI           | sets condensed printing mode                     |
| BEL          | sounds the buzzer                                |
| DC2          | Cancels condensed printing mode                  |
| DC4          | Cancels enlarged printing mode set by<br>SO code |
| DEL          | Cancels last printable code                      |
| BS           | prints and backs by one character                |
| ESC *        | selects 8 dot bit image mode                     |
| ESC 0        | sets 1/8" line spacing                           |
| ESC 2        | sets 1/6" line spacing                           |
| ESC 3 + n    | sets n/216" line spacing                         |
| ESC 4        | selects alternate character set                  |
| ESC 5        | Cancels alternate character set                  |
| ESC 8        | ignores paper end signal                         |
| ESC 9        | enables paper end signal                         |
| ESC ←        | sets/cancels underline printing mode             |
| ESC <        | prints from leftmost to right for one line       |
| ESC @        | initializes the printer                          |
| ESC A        | sets amount of one line paper feed               |
| ESC C        | sets form length                                 |
| ESC E        | sets emphasized printing mode                    |
| ESC F        | Cancels emphasized printing mode                 |
| ESC G        | sets double-strike printing mode                 |
| ESC H        | Cancels double-strike printing mode              |
| ESC I        | selects control code or printable character      |
| ESC K        | sets normal-density bit image mode               |
| ESC L        | sets dual-density bit image mode                 |
| ESC M        | sets elite-sized character                       |
| ESC N        | sets skip-over perforation                       |
| ESC O        | Cancels skip-over perforation                    |
| ESC P        | sets pica-sized character                        |
| ESC Q        | sets column end                                  |
| ESC R        | selects international character set              |
| ESC S        | sets super/subscript printing mode               |
| ESC T        | Cancels super/subscript printing mode            |
| ESC U        | sets/cancels unidirectional printing             |
| ESC W        | sets/cancels enlarged printing mode              |
| ESC Y        | sets double speed dual-density bit image<br>mode |
| ESC Z        | sets quadruple-density bit image mode            |
| ESC e        | relative horizontal/vertical tabulation          |
| ESC l        | sets column head                                 |
| ESC s        | sets/cancels half speed printing                 |

Specifications are subject to change without notice.

# NEW PRINTERS

## Versatile Printer **FX-80**

| SPECIFICATIONS             |                                                                                                                                                                  |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Print Method .....         | Serial impact dot matrix                                                                                                                                         |
| Print Rate .....           | 160 characters per second                                                                                                                                        |
| Print Direction .....      | Bidirectional with logic-seeking in the text mode<br>Unidirectional in the graphic mode                                                                          |
| Number of Pins in Head ... | 9                                                                                                                                                                |
| Line Spacing .....         | 1/6, 1/8 inch selectable by DIP switch or programmable from 1/216 to 255/216 inch                                                                                |
| PRINTING CHARACTERISTICS   |                                                                                                                                                                  |
| Character Set .....        | Full 96-character ASCII with descenders plus 9 international character sets                                                                                      |
| Character Size .....       | 3.1(H) × 2.1(W)mm (Normal)                                                                                                                                       |
| Character Structure .....  | Text mode; 9 × 9<br>Bit image; 480 × 8, 960 × 8, 576 × 8, 640 × 8, 720 × 8, 1920 × 8, 480 × 9, 960 × 8 dots/line                                                 |
| PRINTING SIZE              |                                                                                                                                                                  |
|                            | Max. Characters/Line                                                                                                                                             |
| Normal .....               | 80                                                                                                                                                               |
| Enlarged .....             | 40                                                                                                                                                               |
| Condensed .....            | 137                                                                                                                                                              |
| Condensed-Enlarged .....   | 68                                                                                                                                                               |
| Elite .....                | 96                                                                                                                                                               |
| Elite-Enlarged .....       | 48                                                                                                                                                               |
| FORMS HANDLING             |                                                                                                                                                                  |
| Form Feed .....            | Programmable length to 255 lines                                                                                                                                 |
| Horizontal Tab .....       | To 32 positions                                                                                                                                                  |
| Vertical Tab .....         | To 16 positions at 8 channels each with<br>Electronical Vertical Feed Unit                                                                                       |
| MEDIA HANDLING             |                                                                                                                                                                  |
| Paper Feed .....           | Adjustable sprocket pin feed plus friction feed                                                                                                                  |
| Paper Width Range .....    | Fanfold; 9.5" to 10"<br>(with an optional tractor unit;<br>4" to 9")<br>Roll Paper; 8.5" ± 0.12" (with an optional roll paper holder)<br>Cut Sheet; 8.3" to 8.5" |
| Copies .....               | One original plus two carbon copies (total thickness not to exceed 0.3mm)                                                                                        |
| INTERFACE                  |                                                                                                                                                                  |
| Standard .....             | Centronics-style 8 bit Parallel                                                                                                                                  |
| Optional .....             | IEEE 488, RS-232C, etc.                                                                                                                                          |
| INKED RIBBON               |                                                                                                                                                                  |
| Color .....                | Black                                                                                                                                                            |
| Type .....                 | Exclusive cartridge                                                                                                                                              |
| Life Expectancy .....      | 3 million characters                                                                                                                                             |
| Environmental Conditions   |                                                                                                                                                                  |
| Operating Temperature ...  | 5 to 35° centigrade                                                                                                                                              |
| Operating Humidity .....   | 10 to 80% non-condensing                                                                                                                                         |
| POWER REQUIREMENT          |                                                                                                                                                                  |
| Voltage .....              | 120, 220, 240V, 50/60Hz                                                                                                                                          |
| Current .....              | 1 Amp max.                                                                                                                                                       |
| Power Consumption .....    | 70 VA max.                                                                                                                                                       |
| PHYSICAL CHARACTERISTICS   |                                                                                                                                                                  |
| Height .....               | 100mm                                                                                                                                                            |
| Width .....                | 420mm                                                                                                                                                            |
| Depth .....                | 347mm                                                                                                                                                            |
| Weight .....               | 7.5kg                                                                                                                                                            |

| MAJOR SOFTWARE CONTROL CODES |                                               |
|------------------------------|-----------------------------------------------|
| Control Code                 | Function                                      |
| CR                           | starts printing                               |
| LF                           | advances paper one line                       |
| FF                           | advances paper to top of next form            |
| VT                           | vertical tabulation                           |
| HT                           | horizontal tabulation                         |
| SO                           | sets enlarged printing mode                   |
| SI                           | sets condensed printing mode                  |
| BEL                          | sounds the buzzer                             |
| DC2                          | cancels condensed printing mode               |
| DC4                          | cancels enlarged printing mode set by SO code |
| DEL                          | cancels last printable code                   |
| BS                           | prints and backs by one character             |
| ESC !                        | selects print mode                            |
| ESC %1                       | selects download character set                |
| ESC :                        | copies fonts from ROM CG to download CG       |
| ESC &                        | defines downloadable character font           |
| ESC *                        | selects 8 dot bit image mode                  |
| ESC /                        | selects vertical format unit channel          |
| ESC 0                        | sets 1/8" line spacing                        |
| ESC 2                        | sets 1/6" line spacing                        |
| ESC 3                        | sets n/216" line spacing                      |
| ESC 4                        | selects alternate character set               |
| ESC 5                        | cancels alternate character set               |
| ESC 8                        | ignores paper end signal                      |
| ESC 9                        | enables paper end signal                      |
| ESC -                        | sets/cancels underline printing mode          |
| ESC <                        | prints from leftmost to right for one line    |
| ESC >                        | sets MSB to logical 1                         |
| ESC =                        | sets MSB to logical 0                         |
| ESC @                        | initializes the printer                       |
| ESC A                        | sets amount of one line paper feed            |
| ESC B                        | sets vertical TAB stop position               |
| ESC C                        | sets form length                              |
| ESC D                        | sets horizontal TAB stop position             |
| ESC E                        | sets emphasized printing mode                 |
| ESC F                        | cancels emphasized printing mode              |
| ESC G                        | sets double-strike printing mode              |
| ESC H                        | cancels double-strike printing mode           |
| ESC I                        | selects control code or printable character   |
| ESC K                        | sets normal-density bit image mode            |
| ESC L                        | sets dual-density bit image mode              |
| ESC M                        | sets elite-sized character                    |
| ESC N                        | sets skip-over perforation                    |
| ESC O                        | cancels skip-over perforation                 |
| ESC P                        | sets pica-sized character                     |
| ESC Q                        | sets column end                               |
| ESC R                        | selects international character set           |
| ESC S                        | sets super/subscript printing mode            |
| ESC T                        | cancels super/subscript printing mode         |
| ESC U                        | sets/cancels unidirectional printing          |
| ESC W                        | sets/cancels enlarged printing mode           |
| ESC Y                        | sets double speed dual-density bit image mode |
| ESC Z                        | sets quadruple-density bit image mode         |
| ESC b                        | sets VFU stop position                        |
| ESC j                        | reverses paper feed                           |
| ESC l                        | sets column head                              |
| ESC m                        | selects special CG                            |
| ESC p                        | sets/cancels proportional spacing mode        |
| ESC s                        | sets/cancels half speed printing              |

Specifications are subject to change without notice.



## IMPRESSIONS of the CENTRONICS 739

Looking through the back issues of DAInamic and from contacts with other DAI users it would seem that most have chosen the EPSON MX.. range printers, and having made a different choice myself I thought a review of my printer might be of interest. A year or so ago when I started investigating printers the advertised price of the Centronics 739 was more or less the same as the MX 80, and as my main requirement (after program listing) was for a document printer, I decided to go for the Centronics.

The printer can produce two entirely different character sets, one used for the letter quality proportional space characters and the other for standard print. The standard set (10 CPI Normal) produces 80 characters/line, and the same set can be compacted for 132 characters/line (16.7 CPI Condensed). The printer has a nine wire/pin head which produces a 7x8 matrix for Normal and Condensed print, whilst proportional characters are formed by up to 9 vertical and 6 - 18 horizontal dots. In graphics mode only the top 6 pins are used, with 594 dot positions per line of 8 inches. Each character type can be elongated, and there are optional foreign characters selectable by an internal DIL switch. In addition, two other DIL switches select auto line feed on/off and a self test function. There are three external controls at the front of the printer, on/off, online/local, and rev/forward paper feed. Strangely, the on/off indicator LED was mounted inside the case on the main circuit board making it invisible except when looking straight down on the top. I have since moved it to the obvious position next to the switch. The printer can handle three paper types, 11 x 9.5 inch perforated fan-fold, 8.5 inch continuous roll, and single sheets, and runs at 100 cps in the standard and 80 cps in proportional mode. The printer can produce super and subscript by shifting the character base line half a line up or down, and an underline can be switched on or off anywhere in a line. Within any line the print head can be backspaced or moved on any number of dot positions under program control, and this can be used to advantage for right hand margin justification, as the print on this page shows.

Since having the printer I have suffered two breakdowns, the first time a miniature fuse being the only casualty, however in the second case I had to replace four diodes, which was less easy to trace. Nevertheless, I am well satisfied with the printers overall robustness and the high quality print output. Here are some examples . . .

```

%%
PROPORTIONAL          !"#$%&'()*+,-./0123456789;:<=>?@ABCDEFGHIJKLMN
OQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
10 CPI NORMAL         !"#$%&'()*+,-./0123456789;:<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
16.7 CPI CONDENSED   !"#$%&'()*+,-./0123456789;:<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
PROPORTIONAL ELONGATED !"#$%&'()*+,-./012
3456789;:<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
10 CPI ELONGATED    !"#$%&'()*+,-./012
3456789;:<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
16.7 CPI ELONGATED   !"#$%&'()*+,-./0123456789;:<=>?@ABCD
EFGHIJKLMN
OPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
%%

```

# THE BASIC MONITOR (#C80C-#C956)

---

## PART 1: GENERAL

This article describes the principles of the BASIC monitor routine of the DAI in general. In further articles, the details of certain parts of the monitor will be explained. As reference for this article, the 'DAI pC FIRMWARE MANUAL' is used. The annexed (simplified) flow chart will be the guide through this article.

The BASIC monitor might be seen as the 'heart' of the operating system of the DAI-BASIC. Under control of this monitor, program lines can be entered into the textbuffer, direct commands will be executed, the run of a program is controlled, a 'break' is managed, etc.

The monitor is entered by the DAI after a reset or power-on at address #C818. Other entries (on #C80C, #C814 and #C823) are also used after finishing certain routines. This will be explained in part 2.

### INITIALISATION (#C818-#C843):

---

The monitor must be initialised. That means, a defined start situation has to be created. The stackpointer is set, all flags indicating the running of inputs, programs, subroutine calls and of encoding stored lines are cleared. Result: the DAI is 'at rest'. Keyboard interrupts and clock interrupts are enabled, otherwise no inputs can be made and no output to the screen would be possible.

### INPUT FROM THE EDIT BUFFER (#C846-#C84E):

---

The encoded input switch #0135 is checked. This switch defines the source for inputs which have to be encoded. Encoding means: Translation of BASIC lines (direct commands or program lines) from the readable BASIC into the code which is understandable by the DAI semi-compiler in order to execute these lines. In this code a program is also stored in the textbuffer. Details about this pseudo-code can be found in Newsletter 11, page 196 and further.

If the input source is the editbuffer, then handling of the inputs from this buffer is prepared via #D879, and encoding takes place via #C867. When one textline is encoded, the monitor starts again at the beginning, until the editbuffer is empty.

See also a previous Newsletter for the 'editor story'.

Remark: Encoding of 'undefined rubbish' (namely the momentary contents of the encoded input buffer) occurs also if the switch #0135 is >2 !! This firmware failure is based on the assumption that #0135 is always <=2.

**INPUT FROM KEYBOARD (#C851-#C864):**  
-----

If the input source is not the editbuffer, but the keyboard, the prompt ('\*') is displayed on the screen and the input of a BASIC textline is awaited via #DD1A. In plain text: the DAI waits for a direct command line or a program line to be stored in the textbuffer, which must be entered via the keyboard. This input sequence is only finished by typing a CR ('RETURN') or by a break. All characters typed in are displayed on the screen. The screen memory is at the same time the onliest place where the inputs are stored.

When CR is typed, the monitor starts working again. It takes the first character from the input line in the screen memory for investigation.

The difference between direct command lines and textlines to be stored in the textbuffer is the existance of a line-number at the beginning of program lines to be stored. If the first character of an input line is a number, then the line must be stored in the textbuffer after encoding. Otherwise it is a direct command line, and execution of this line must take place directly after encoding.

If the first character in the input line is a CR, then no input line is available, and the monitor waits for a new input line.

**ENCODING PROGRAM LINES (#C867-#C86A):**  
-----

If a program line has been typed in, this line is encoded and stored in the textbuffer on the correct location; that means in sequence of linenumber. The monitor is restarted to get new input lines.

The details on how encoding is processed will be explained in a next article.

**ENCODING OF DIRECT COMMAND LINES (#C86D-#C87D):**  
-----

The direct command lines are encoded via the routine for encoding a BASIC command via RST1/00. The result of this process - the semi-compiled code - is stored in the encoded input buffer #013E-#01BD. This buffer is 128 bytes large, so the maximum length of an encoded program line is therefore 128 bytes, otherwise an error message 'LINE TOO COMPLEX' will be the result.

The flag #0117 is set (#FF), simulating a 'RUN' of a program line; the registers BC - used to indicated the position in a program line where the monitor is busy executing it - point to the beginning of this encoded direct command line in the encoded input buffer EBUF, and the direct command line will be executed.

### RUNNING OF DIRECT COMMANDS OR PROGRAM LINES (#C87F-#C88F):

---

The next part is valid both for the execution of direct command lines as for the execution of program lines stored in the textbuffer.

The onliest difference is the contents of the register pair BC. For direct command execution, it points to the encoded input buffer EBUF. For the execution of a program stored in the textbuffer, it points to the start of a textline in this buffer.

The first character of a direct command or a stored line is taken. If it is a token - an abbreviation for a BASIC command - the startaddress of the execution routine of the BASIC command belonging to this token is found in the table #CF02, and this routine is executed via #C8A9.

If the first byte is not a token (#C8E5), then it could be a '0' or the length byte at the beginning of a stored program line. A '0' indicates the end of the textbuffer contents (where a '0' is inserted after the last statement), or the end of a direct command line (see #C876).

When encountering a '0', the monitor is restarted.

If the first byte read by the monitor is the length byte of a stored program line, then the trace/step flags are tested (#0115/#0116). If one of these flags is set, the new program line will be listed on the screen. In case of 'STEP', pressing the spacebar is awaited.

If no 'break' is pressed, the monitor continues on #C87F.

### SPECIAL END OF ACTIONS (#C8AA-#C8B5):

---

If a token has been found, and the particular execution routine has been performed, the monitor returns to #C88F. Here is checked if the execution routine for the particular BASIC routine has eventually a special ending. This special end is indicated by a code in accumulator A and the carry-flag set after running the BASIC execution routine. See for example #DF03: 'STOP': CY=1, A=3.

Four special actions are possible:

- 0: #C908-#C915: After 'LOAD': If 'LOAD' is not a direct command, the loaded program will be runned immediately.
- 1: #C818: Can't continu: After commands which do change program pointers or which alter the contents of the textbuffer, the monitor is restarted.
- 2: #C8C0-#C8C8: After a 'soft break': The 'break' must be handled.
- 3: #C8B8-#C8BD: After a 'STOP' command in a program line. The place where the program execution is stopped must be remembered and 'STOPPED IN LINE ....' has to be printed. Further handling will be as a 'soft break'.

**'SOFT BREAK' HANDLING (#C8CB-#C8E2):**

-----

If a 'break' has been given during the run of a program, 'BREAK IN LINE ....' will be printed. All pointers which are relevant to the program status - the so-called FRAME #0100-#0115 - are saved on stack, and flag #0126, indicating the existence of a suspended (= interrupted) program is set before the monitor is restarted.

If a 'break' is given during execution of a direct command, then only a CR is printed. No program status have to be saved now.

**SUMMARY OF THE MAIN LOOPS IN THE MONITOR:**

-----

Once the monitor is initialised, a program is stored in the textbuffer, and the direct command 'RUN' is given, the following happens:

- 'RUN' is a direct command, which is encoded into the encoded input buffer EBUF. Via its token #87, the execution routine on #DF9E is found. There - amongst others - the register pair BC is set to the startaddress of the textbuffer, and the monitor is re-entered on point (E).
- Via main loop 1, the first character of the textbuffer is taken. This is the length byte of the first program line. If no TRACE or STEP flags are set, the monitor returns at point (B) with incremented linepointer BC.
- Via main loop 2 the monitor continues. If a token is found, after the linenumber, the appropriate BASIC execution routine is executed, and the monitor continues at point (B).
- This goes on until the end of the textbuffer - a '0' - is found. Then the monitor restarts itself.

For a direct command line, the same explication applies.

**REMARKS:**

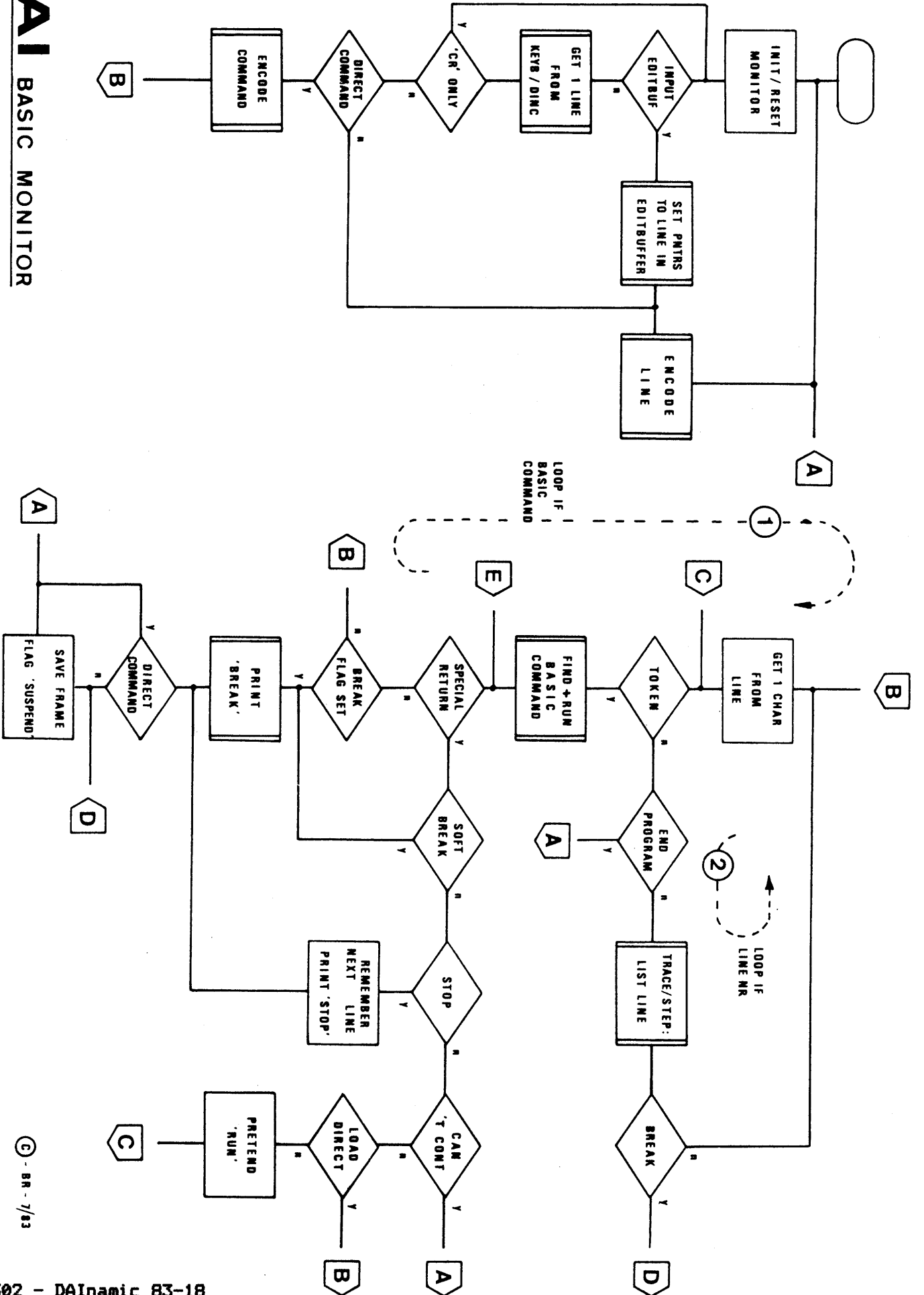
-----

In further articles, some of the details of certain parts of the monitor will be explained extendedly.

For those who are not yet in the possession of the 'DAI pC FIRMWARE MANUAL': It can still be obtained by sending an Eurocheque or an international postal money order of the amount of Hfl. 68.- to 'Micro-Service', Fabritiusstr.15, 6174 RG Sweikhuizen, The Netherlands.

(C) - Jan Boerrigter, July 1983

# DAI BASIC MONITOR



## PROGRAMMING TECHNIQUES

(from DAInamic 13, page 301)

I will start with my apologies for some errors which crept into my last article. I hope that the article was clear enough for you to correct them.

The main subject on this occasion is colours 16 to 19, with colours 20 to 23 as a subsidiary topic. Beginning with the latter, it would be a good thing if all our contributors would employ colours 20 to 23 when drawing in the 4-colour mode. It is so simple! at the start of the program we give a COLORG A B C D, with ABC & D being the 4 wanted colours. Then we give a DRAW instruction in the program using 20 if we want colour A, 21 for B, 22 for C and 23 for D. It is good practice first to give the COLORG instruction and then the MODE instruction. This avoids starting with, say, a bright yellow screen from the previous COLORG and then changing back to the wanted colour. In the 16-colour mode the first colour in COLORG is used for the background and the rim too takes on the same colour. This cannot be cleared by a FILL 0,0 XMAX,YMAX C, but only by a new COLORG followed by a new MODE 1,3 or 5. The advantage of using colours 20 to 23 lies in the fact that beginners can easily change a poor colour combination. I can make excellent use of COLORG 0 1 2 3 with my RGB monitor but I am sure viewers in black and white would be unhappy with my efforts if I changed over to 4 5 8 13.

Before we get on to colours 16 to 19 I must explain the screen arrangement in the RAM. If my explanation is inadequate refer to the articles by N J Looije or J Boerrigter in previous issues.

The screen RAM can be seen as two blocks of memory which together determine the ultimate picture. The memory banks are separated hardwarewise - the B bank is for even numbered bytes and the C bank for the odd numbered ones. This is important for the programmer.

What follows is the picture construction for the 4-colour modes. The 16-colour modes have a different structure. The colour which will be adopted by a dot on the screen is defined as follows :- two bits appropriate to each dot indicate from which of the four colour registers that dot will take its colour. Firstly, the standard case : after a reset we give a COLORG 0 3 9 14 command and then a MODE 4. The whole screen will become black and then change to MODE 4A. By going to Utility with UT we can establish with a Display that all the line control bytes show 00. (DB00 B500 for example). What happens now if we set these bytes to a different value? Let us try, POKE #B000,#AA and see 4 dots appear on the screen coloured red. For those with little experience of bit information #AA=10101010. We now follow with POKE #B002,#66 (#66=01100110). Two small red lines appear next to the dots. POKE #B001,#F0 : Of the four red dots the leftmost two change to yellow and two blue ones come up. With these examples and a little thought we can reason out how the colouring is effected.

| Even byte bit | Odd byte bit | Normal colour | Colour in example |
|---------------|--------------|---------------|-------------------|
| 0             | 0            | 20            | 0 = black         |
| 1             | 0            | 21            | 3 = red           |
| 0             | 1            | 22            | 9 = blue          |
| 1             | 1            | 23            | 14 = yellow       |

It is even possible to have more than four colours in 4-colour modes. The article by N J Looije about the video screen RAM gives much information on that topic so I will not discuss it again here. I will limit myself to establishing that in any line we want we can choose the colour control byte so that the colour registers can be changed one by one. One of the colours can be changed per line. We can take care of this with a bit of clever programming whereby we give each dot the precise colour we want while still arranging all the colours over the whole screen.

**TRANSLATIONS-TRANSLATIONS-TRANSLATIONS-**

But to return to the subject, I wanted to deal with colours 16 to 19. Firstly, the colours above 16 are not real colours but special applications: colours 20 - 23 refer to the colours of the COLORG instruction and the colours 16 - 19 put bytes in the screen RAM on or off. Here we again come back to the introduction: the colour of each dot in a picture is stipulated by two bits. These two bits are in two different bytes, one having an even numbered address and the other an odd number, the odd one being higher than the even one. By now giving a DRAW instruction with colour 17 the bits belonging to the even byte are set to one. If after a reset and MODE 4, the command FILL 0,0 44,44 C is given it makes no difference whether C is 5,17 or 21. If without a reset but after another program, we use 21 it could happen that we do not get green but some other colour. However 21 is better than 5 because using 5 after another program brings with it the chance of an error message, COLOUR NOT AVAILABLE. There is no risk of that when drawing in colour 17. We will demonstrate the difference between drawings in colours 17 and 5. Key in MODE 4

```
FILL 22,22 66,66 22
FILL 0,0 44,44 5
```

and see that the green area has overwritten a part of the orange (22=10 standard) square. Tap in the instructions again but this time use colour 17 instead of colour 5 and the green area is seen to be no longer a square but a kind of L.

The part where the squares overlap is now a new square coloured white. Let us analyse that: Through the MODE 4 command we set (unless we came from MODE 4A) all the bits which designate the colour to zero. This is also valid if we had not made colour 0 the first colour of a COLORG although the screen is then another colour. The first FILL produces an orange square and sets the appropriate memory bits, the even bytes to zero and the odd bytes to one. The second FILL, using colour 5 sets the even byte bits to one and the odd byte bits which became one on the first FILL, now become zero again. The resultant colour is therefore that which was last given, in this case green (5). Because with the second FILL command we gave 17, only the bits of the even bytes will go to one and the bits of the odd bytes will NOT be changed. Thus the area where the two squares overlap, on the first FILL receives ones in the odd bytes, and with the second FILL using 17, ones in the even bytes. So this overlap square takes colour 23 (normally white). We can therefore set at one, only the bits of the even bytes, with a 17. However, with a 16 we can zero them again. Similarly with colour 19 we can set to one the bits of the odd bytes and with colour 18 switch them to zero again.

What is the significance of this to the programmer? According to the Manual we can achieve an effect of animation, but we must not expect too much from this. The effect is all right in theory but the drawing is too slow to give the appearance of film.

But we will show how this can be done. We do a drawing in colour 17 with a COLORG 0 5 x x and then see it on the screen in colour 5 (green). If we choose zero for x in the COLORG we can afterwards draw what we want in colour 19 without it being visible. Colour 19 puts ones in the odd bytes and therefore the points are either in colour 22 if the other bit is not one, or colour 23 if the other bit is one. But then the point becomes black. Also green points become black if they are drawn over. We can overcome this by letting colour 23 be green. That is, by using COLORG 0 5 0 5. When the new drawing in colour 19 is ready we give a COLORG 0 0 5 5.

All points with two bits set (colour 23) are green and all points with only the bit from the odd byte set (colour 22) are green. The points where both bits are zero (colour 20) are black and all points where only the bits from the even bytes are at one (colour 21) are black. We thus see nothing of the old drawing but all of the new. Then we erase drawing 1. Give all instructions



**TRANSLATIONS-TRANSLATIONS-TRANSLATIONS-**

which had colour 17 again, but this time with colour 16 (the bits of the even bytes at zero), or if it is quicker, with a FILL 0,0 XMAX,YMAX 16. Then do a new drawing (3) in colour 17. COLORG 0 5 0 5 and drawing 3 is visible and drawing 2 is not. Erase drawing 2 with colour 18, do drawing 4 in colour 19 and then again COLORG 0 0 5 5 to make drawing 4 visible and drawing 3 disappear. Erase drawing 3 with colour 16, do drawing 5 in colour 17 and then again a COLORG 0 5 0 5, etc.

In this way we can, for example, draw a car and move it across the screen. If the car takes up a lot of lines its speed is too low for the motion to be classed as animation. Even a drawing of 5 or 6 lines each with about 20 points can not reasonably be called dashing. Some people may be less critical and could find a larger number of lines acceptable. I am thinking of a large scale static drawing (in colour 23) with only a few moving parts, such as a clock. Remember though the snag which limits these animations to the 2 colours (foreground and background).

However there are applications where the facilities are suitable in DAI-basic. For example J Visser uses them very effectively in his 3-dimensional maze. The situation seen from the player's standpoint is shown on the screen. The player chooses a move, the program draws the fresh situation invisibly and then a COLORG rapidly changes the picture.

I myself have also written a maze program which by the use of colours 16-19 makes a specified part of the screen visible. One thus sees only a small part of the maze. It is done like this: The maze is drawn in colour 17, then we put a block in colour 19 (FILL A,B C,D 19). After a COLORG 0 0 14 9 the maze is invisible because colour 17 signifies even bytes and with odd bytes being zero, colour 21, thus 0=black. However where the block is, the background is yellow and where both maze and block are the colour is 23, thus 9=blue. We can let the block move by drawing a line on one side of it in colour 19 and erasing a line on the other side of it with colour 18. Another happy incidental is the possibility of giving an inverted colour to a particular part of the picture. Think about it with me: Make a drawing in colour 17 and colour in the required part (FILL) with colour 19. Then give a COLORG 0 12 12 0 and the drawing goes blue with a black background. But without the FILL this is reversed. So we can point out or make conspicuous a particular part of a picture without disturbing the remainder.

With this article are a number of programs which show how we can craftily make use of colours 16-19. [see pages 304 & 305 of DAInamic 13].

Frank H Druijff

**A CHEAP & VERSATILE EPROM-PROGRAMMER**  
(DAInamic 13, p344/9)

The DAI personal computer has many extended input/output possibilities. With these many applications can be realised. This article deals with one such application.

**WHAT IS AN EPROM ?**

An Erasable Programmable Read Only Memory is a memory element which, as its name suggests, can be only read by the computer. The characteristic of Read Only Memories is that they do not lose their contents when their power supply is switched off. They are thus especially suitable for a microprocessor system which must work as soon as it is switched on. ROMs have the disadvantage that their contents must be set in the factory, by a process which is only

economically attractive for large quantities. An EPROM on the other hand has the ability of being repeatedly programmed by the user. After programming the contents can, if needs be, be erased with Ultra Violet light. Therefore it is an ideal building block for prototyping, for short runs and for the hobbyist.

### PROGRAMMING AN EPROM

Nowadays most EPROMs are constructed as a matrix of 1024x8, 2048x8 or 4096x8 bits, ie, the 1K, 2K and 4K EPROM. They can be easily incorporated in a system with words of 8 bits. Fig 1 shows the EPROMs with type numbers and pin identities. In normal circumstances the EPROM is part of a microprocessor system. The microprocessor gives an address, sends a chip select and a read signal; then a memory element will be selected and its contents written, via its output lines, to the microprocessor. This data can be, for example, the next instruction for the system. The EPROM must of course have been programmed. The starting point for this is an erased IC, in which all the bits are "1". Programming is the changing of selected bits to "0" by connecting a +25V programming voltage to the Vpp pin, then giving an address and the required data. A 50 msec TTL pulse on the PROG input then programmes the EPROM. In effect the data is "burned" into the addressed memory location. By repeating this process the whole EPROM can be programmed. This is usually done by incrementing the address by 1 each time and writing a new data word. Table 1 gives the operating specification for the EPROMs. As programming takes 50 msec for each location, fully programming a 2785 takes 50 sec, a 2716 or 2516 takes 100 sec and a 2732 or 2752 takes 200 sec. A programmed EPROM can be erased but professional equipment is expensive. Erasure can however be done using a special UV-lamp, the Philips TUV 6W-E which costs about 50 Dutch guilders. If the EPROM is fastened to the lamp with an elastic band it will be wiped clean in 15 to 20 minutes. (Warning; Ultra Violet light damages the eyes, so wear red goggles or operate the lamp in a closed box).

### BUILDING A PROGRAMMER

We have seen that to program an EPROM we must provide an address, a data word and the required control signals to the IC. These can all be generated by the DAI and sent via the DCE bus to the programming equipment. It is also necessary to provide the various voltages which the IC needs. The supply voltage, +5V is available from the DAI; the +25V programming voltage must be separately generated.

### THE HARDWARE

The EPROM programmer needs connection to the DAI and the 25V supply (figs 2 and 3). The programming voltage is provided from a 24V AC supply, rectified, smoothed and then stabilised by a 7812 (+12V) voltage regulator which is adjusted to the correct operating level by the potentiometer connected to its base lead. The potentiometer should be set so that the output level reaches precisely +25V. Fig 2 shows which DCE bus leads are to be connected to the EPROM pins. The numbers printed nearest the IC are its pin numbers; the others are for the DCE bus. Table 2 shows how the DCE bus is used.

The schematic (fig 2) shows two relays which are switched by the DAI, via the DCE bus, each being buffered by a transistor. 12V relays were chosen as they can use the DAI's power supply. There is a 12V, 1A feed in the DAI and here only about 0.5A of it is used. It is also possible to use 24V relays, feeding them from the programming voltage, but then a larger 24V transformer would be needed to cope with the increased load. The diodes connected across the relay coils are to dissipate the voltage surge induced when the relays are switched, a surge which would destroy the transistors. One relay selects the 2732 or the other EPROMs as the 2732's connections differ from the rest. Three connections have to be switched and most programmers accomplish this with a hand operated switch which can cause mistakes. The second relay connects the programming voltage to the IC. All the components can be mounted in a small case. The base for the EPROM can be a normal IC socket but as so much care is needed to insert and



DNA and DAI software presents some nice possibilities, especially for developing an 8080/8085 system. Use of the programmer need not be restricted to the 5 types of EPROM mentioned here. Altering it, for example for a 2708 is relatively simple, while with a suitable printed board it is possible to program a one-chip microprocessor like the Intel 8748. There will be more about this later if there is sufficient interest.

COMPONENT LIST

Most of the items in the component list, Table 5, need no translation. Here are the less obvious ones:- B1 Bridge rectifier B40/C1000, L1 Green LED, Z1 100mA Fuse. Extras: 2 x 12V Relays, fuse-holder, 24V 100mA mains transformer, ribbon cable with 34 pin connector, zero insertion force IC socket, case, mains lead, piece of circuit board.

G. Knoops

UNIT COLOUR MODES AND TEXT MODES

(from DAInamic 13, page 326)

Dear DAI Friends

Since the last issue of DAInamic I have had a number of reactions from members and others about the unit colour and text modes. The Handbook would appear to be neither clear nor sufficiently comprehensive. This letter and the accompanying demonstration program are therefore intended to supplement the earlier article.

According to the Handbook the DAI recognises eight resolutions, namely:

- > Unit colour mode      4 resolutions      = 4 bytes
- > Low resolution        88 blobs per line    = 24 bytes
- > Medium resolution    176 blobs per line   = 46 bytes
- > High resolution      352 blobs per line   = 90 bytes
- > Super resolution     528 blobs per line   = 134 bytes

The first omission is that the book says nothing about the number of bytes per line, so the numbers shown above are the values normally assumed. Each is calculated from the number of bytes for the characters times two, with two added for the LINE MODE BYTE and the COLOUR TYPE BYTE. There will be more later about the exceptions. The arrangement of the screen hardware shows, in simplified form, the following: Firstly the highest RAM address is loaded and two instructions are read (the LINE MODE and COLOUR TYPE BYTES). In association a counter is loaded with the number of bytes still to be read before the following instructions. At the same time readout speed, the number of repeats, the colour mode and character or graphics mode are established. When the counter has run down (from top of RAM - length of line) the next RAM address holds the following instructions which are then read: LINE MODE BYTE and COLOUR TYPE BYTE.

If for example, in MODE 0 (Super resolution), you put a line in High resolution by replacing the LINE MODE BYTE with #6A the video hardware will read 90 of the original 134 bytes, the 91st byte will be read as LINE MODE BYTE and the 92nd as COLOUR TYPE BYTE. Suppose the 92nd is a colour byte set at #00; the hardware puts the line in unit colour mode and the resolution then depends on whatever character is in the 91st byte. The result would be a repeating pattern of stripes or bits of characters for a minimum of 11 lines (44 bytes divided up into 11 unit colour mode lines) until the hardware encounters the LINE MODE BYTE of the following line. These



-TRANSLATIONS-TRANSLATIONS-TRANSLATIONS-

The table below summarizes the effect of CONTROL BYTES.

| Control bytes                                                                                              | Display  | Colours | Resolution | Bytes to new control byte | Data and col. bytes | Visible fields and blobs |
|------------------------------------------------------------------------------------------------------------|----------|---------|------------|---------------------------|---------------------|--------------------------|
| 0x 40                                                                                                      | GRAPHICS | 4       | LOW        | 24                        | 11                  | 11 - 88                  |
| 1x 40                                                                                                      | GRAPHICS | 4       | MEDIUM     | 46                        | 22                  | 22 - 176                 |
| 2x 40                                                                                                      | GRAPHICS | 4       | HIGH       | 90                        | 44                  | 44 - 352                 |
| 3x 40                                                                                                      | GRAPHICS | 4       | SUPER      | 134                       | 66                  | 66 - 528                 |
| 4x 40                                                                                                      | CHAR     | 4       | LOW        | 26                        | 12                  | 10 - 87                  |
| 5x 40                                                                                                      | CHAR     | 4       | MEDIUM     | 48                        | 23                  | 21 - 173                 |
| 6x 40                                                                                                      | CHAR     | 4       | HIGH       | 90                        | 44                  | 43 - 345                 |
| 7x 40                                                                                                      | CHAR     | 4       | SUPER      | 134                       | 66                  | 64 - 519                 |
| 8x 40                                                                                                      | GRAPHICS | 16      | LOW        | 24                        | 11                  | 11 - 88                  |
| 9x 40                                                                                                      | GRAPHICS | 16      | MEDIUM     | 46                        | 22                  | 22 - 176                 |
| Ax 40                                                                                                      | GRAPHICS | 16      | HIGH       | 90                        | 44                  | 44 - 352                 |
| Bx 40                                                                                                      | GRAPHICS | 16      | SUPER      | 134                       | 66                  | 66 - 528                 |
| Cx 40                                                                                                      | CHAR     | 16      | LOW        | 26                        | 12                  | 10 - 87                  |
| Dx 40                                                                                                      | CHAR     | 16      | MEDIUM     | 48                        | 23                  | 21 - 173                 |
| Ex 40                                                                                                      | CHAR     | 16      | HIGH       | 90                        | 44                  | 43 - 345                 |
| Fx 40                                                                                                      | CHAR     | 16      | SUPER      | 134                       | 66                  | 64 - 519                 |
| XXXXXXXXXXXXXXXXXXXXXXXXX UNIT COLOUR MODE XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX |          |         |            |                           |                     |                          |
| 0x 00                                                                                                      | GRAPHICS | 4       | LOW        | 2                         | 1                   | 11 - 88                  |
| 1x 00                                                                                                      | GRAPHICS | 4       | MEDIUM     | 2                         | 1                   | 22 - 176                 |
| 2x 00                                                                                                      | GRAPHICS | 4       | HIGH       | 4                         | 1                   | 44 - 352                 |
| 3x 00                                                                                                      | GRAPHICS | 4       | SUPER      | 4                         | 1                   | 66 - 528                 |
| 4x 00                                                                                                      | CHAR     | 4       | LOW        | 2                         | 1                   | 11 - 88                  |
| 5x 00                                                                                                      | CHAR     | 4       | MEDIUM     | 2                         | 1                   | 22 - 176                 |
| 6x 00                                                                                                      | CHAR     | 4       | HIGH       | 4                         | 1                   | 44 - 352                 |
| 7x 00                                                                                                      | CHAR     | 4       | SUPER      | 4                         | 1                   | 66 - 528                 |
| 8x 00                                                                                                      | GRAPHICS | 16      | LOW        | 2                         | 1                   | 11 - 88                  |
| 9x 00                                                                                                      | GRAPHICS | 16      | MEDIUM     | 2                         | 1                   | 22 - 176                 |
| Ax 00                                                                                                      | GRAPHICS | 16      | HIGH       | 4                         | 1                   | 44 - 352                 |
| Bx 00                                                                                                      | GRAPHICS | 16      | SUPER      | 4                         | 1                   | 66 - 528                 |
| Cx 00                                                                                                      | CHAR     | 16      | LOW        | 2                         | 1                   | 11 - 88                  |
| Dx 00                                                                                                      | CHAR     | 16      | MEDIUM     | 2                         | 1                   | 22 - 176                 |
| Ex 00                                                                                                      | CHAR     | 16      | HIGH       | 4                         | 1                   | 44 - 352                 |
| Fx 00                                                                                                      | CHAR     | 16      | SUPER      | 4                         | 1                   | 66 - 528                 |

These are the right values for correct screen construction. Here too is a demonstration program showing all the possibilities.

With friendly greetings      N. P. LOOIJE

BANK 0:#C000 to #EFFF

C000 Entrypoints  
C035 Package Initialisation  
C04B Load (OOD0/1) on stack and continue on this address  
C05E Continue on (OOD2/3)  
C06C Continue on (OOD6/7)  
C073 Continue on (OOD2/3)  
C079 Floating point compare  
C0AC Integer compare  
C0BB Increment integer number in memory  
C0D5 Decrement integer number in memory  
C0F3 Increment floating point number in memory  
C1FB Decrement floating point number in memory  
C21E Save FPAC on stack  
C234 Retrieve FPAC from stack  
C249 Input floating point number to FPAC  
C361 Convert a floating point number for output  
C486 Pretties up FPT or INT number  
C51A Move 11 bytes 1 memory location higher  
C531 Move bytes one memory location down  
C573 Input integer number to FPAC  
C5B2 Convert integer number for output  
C614 Input hex number to IAC  
C653 Convert IAC to hex for output  
C6C0 Math. restart (RST 4)  
C6CF ROM bank switching  
C6F2 Switch to ROM bank  
C6FD Screen restart (RST 5)  
C70E Utility/encode (RST 1)  
C719 Reset  
C7AB Initialisation screen data  
C7E0 Initialisation parameters  
C7FB Check for highest RAM addr  
C80C Start from scratch  
C918 Encoding when line number is given  
C957 Error while encoding a stored line  
C9D1 Initialise program buffers  
CA01 Memory management routine  
CA25 Emergency stop routine (Graphics modes)  
CA34 Find string BASIC instruction in table  
CB23 Empty HEAP  
CB9E Load 0 into 4 consecutive memory locations  
CBBF Strings - BASIC commands  
CD64 basiccmd TALK  
CD8B Pointers to strings of BASIC commands  
CE70 Print ', '  
CE75 Print a string between spaces  
CEB5 Change screen MODE  
CEC6 Set HEAP size to default value  
CECF Print DAI PERSONAL COMPUTER in medium resolution  
CEE4 Select ROM bank 0 , print message  
CEF9 Print 'COMPUTER' under 'DAI PERSONAL'  
CF02 Pointers to routines BASIC-COMMANDS  
CF86 Strings arithmetic and logical operators

D101 Reset pointer 0000H  
D106 Add string data after end other string  
D121 Compare two strings  
D16D Transfer of string data  
D195 Organise program buffers  
D1DB Keyboard scanning: Check source and input  
D236 Set MSB of byte  
D23D basiccmd SAVE  
D270 basiccmd LOAD  
D2A8 Loading error  
D288 Open tape file  
D2C3 basiccmd CHECK  
D2F1 Write block on tape  
D30F Write byte,update checksum  
D316 Write block length, update checksum  
D325 Start file reading  
D340 Read block  
D384 Read byte, calculate checksum  
D38D Read name length  
D3A2 Read + check program name and file type  
D3F4 Read file header  
D40C Write file leader  
D422 Write file trailer  
D42E Start cassette motors  
D445 Stop cassette motors  
D453 Read bit  
D480 Read leader  
D4D4 Read byte  
D4ED Write leader  
D509 Write byte  
D524 Write bit  
D53C Write cycle  
D550 Write trailer bits  
D560 Initiate keyboard pointers  
D578 Keyboard interrupt service (RST 6)  
D59A Scan keyboard, store result  
D606 Break  
D620 Complete keyboard scan  
D632 Get ASCII value of key pressed  
D63F Get key-ASCII and store it  
D642 Check if output to RS232  
D668 Write 2 blocks + trailer on tape  
D68D Cursor handling  
D695 Print character  
D69C Keyboard scanning: Check if ASCII-buffer full  
D6A5 Keyboard scanning: Check if new inputs  
D6BB Keyboard scanning  
D6DA Wait for spacebar  
D71A Stop loading programs  
D720 Initiate writing file leader  
D72D Init SOUND gen,GIC,start HEAP,transfer CASSETTE data  
D750 Check break pointer  
D755 SOUND interrupt (RST 3)  
D783 Failure during ropen  
D78A Check if LOAD during run program  
D790 Check free RAM space



D795 Transfer CASSETTE vectors  
D7A4 DISC/CASSETTE switching vectors  
D7D8 Write block + trailer on tape  
D7DE Failure during ropen  
D7EB Write byte on CURSOR pos addr and update CURSOR pos  
D7FB SAVE : Write name length  
D7FF Initialise loading from tape  
D81D basiccmd SAVEA  
D85E basiccmd LOADA  
D879 Input from edit buffer  
D8A6 Initiate SOUND generator  
D8C8 Output to DCE-BUS  
D8E0 Input from DCE-BUS  
D8FB Initialise interrupt system  
D949 Set up interupt vector area  
D96B Interrupt vector routine  
D973 Update TICC interrupt mask  
D9A9 Clock interrupt (RST 7)  
D9CD General interrupt return  
D9DB Enable clock interrupt  
D9E2 Stack interrupt (RST 2)  
D9F5 Error handling  
DA0B Pointers to error messages  
DA3D Error while running program/inputs  
DA50 Print error message  
DA64 Error when not running program  
DA75 Print line number in which error occurred  
DA94 Pointers to error messages  
DAD4 Print message  
DB32 Print string  
DB44 Print string message  
DB6F Strings for maching messages  
DD1A Scan keyboard and print characters  
DD55 Get X-coordinate CURSOR position  
DD60 Output routine  
DD94 Output to RS232  
DDB4 Input from RS232  
DDBA RS232 frame error  
DDD1 Get character from line, neglect TAB + SPACE.  
DDE0 Get character from line  
DE02 Check if upper case character  
DE09 Check if character is number or upper case  
DE14 Compare (HL) and (DE)  
DE1A Calculate length of block  
DE26 Two complement of 16-bits data  
DE30 Calculate off-set address  
DE39 Calculate address after sting  
DE41 Delay routine  
DE4F Data block transfer  
DE7C Load bank with identical data  
DEB5 basiccmd NEW  
DED5 basiccmd CONT  
DEFE basiccmd STEP  
DF03 basiccmd STOP  
DF0C basiccmd END  
DF15 basiccmd IF

|      |          |                                             |
|------|----------|---------------------------------------------|
| DF20 | basiccmd | IF                                          |
| DF2A | basiccmd | GOSUB                                       |
| DF4C | basiccmd | RETURN                                      |
| DF63 | basiccmd | GOTO                                        |
| DF6A | basiccmd | ON                                          |
| DF71 | basiccmd | ON                                          |
| DF9E | basiccmd | RUN                                         |
| DFBA | basiccmd | RUN                                         |
| DFC0 | basiccmd | POKE                                        |
| DFC9 | basiccmd | OUT                                         |
| DFD5 | basiccmd | WAIT                                        |
| DFF7 | basiccmd | WAIT mem                                    |
| E016 | basiccmd | WAIT TIME                                   |
| E02B | basiccmd | FOR                                         |
| E0C5 | basiccmd | NEXT                                        |
| E0E5 | basiccmd | NEXT                                        |
| E13C |          | Save on soft break ??                       |
| E167 |          | Restore pointers to continue                |
| E18F | basiccmd | DATA - REM                                  |
| E197 | basiccmd | LIST                                        |
| E1AA | basiccmd | LIST                                        |
| E1B6 | basiccmd | LIST                                        |
| E1F5 |          | Run edit commamd                            |
| E253 |          | List edit                                   |
| E25C |          | List edit                                   |
| E265 |          | Initialise EDIT buffer                      |
| E2B3 | basiccmd | PRINT                                       |
| E2FC | basiccmd | INPUT                                       |
| E323 | basiccmd | READ                                        |
| E401 | basiccmd | RESTORE                                     |
| E45A | basiccmd | LET                                         |
| E4BC | basiccmd | SOUND                                       |
| E50C | basiccmd | NOISE                                       |
| E570 | basiccmd | ENVELOPE                                    |
| E5B2 | basiccmd | CURSOR                                      |
| E5BB | basiccmd | MODE                                        |
| E5C1 | basiccmd | DOT                                         |
| E5CE | basiccmd | DRAW                                        |
| E5D7 | basiccmd | FILL                                        |
| E60E | basiccmd | COLORT                                      |
| E615 | basiccmd | COLORG                                      |
| E62F | basiccmd | DIM                                         |
| E69E | basiccmd | UT                                          |
| E6A4 | basiccmd | CALLM                                       |
| E6B5 | basiccmd | CLEAR                                       |
| E6CE | basiccmd | TRON                                        |
| E6D5 | basiccmd | TROFF                                       |
| E8EE |          | Table of jumps to INT/FPT operator routines |
| E9D9 |          | Calculate start address routines            |
| E9F0 |          | Pointers to start address routines          |
| EB51 |          | Calculate free RAM space                    |
| EBC1 |          | Paddle operation (?)                        |
| ECCC |          | Print instruction from token                |
| ECF8 |          | Pointers to start address routines          |
| EE8D |          | Set input direction                         |
| EFED |          | Print string                                |