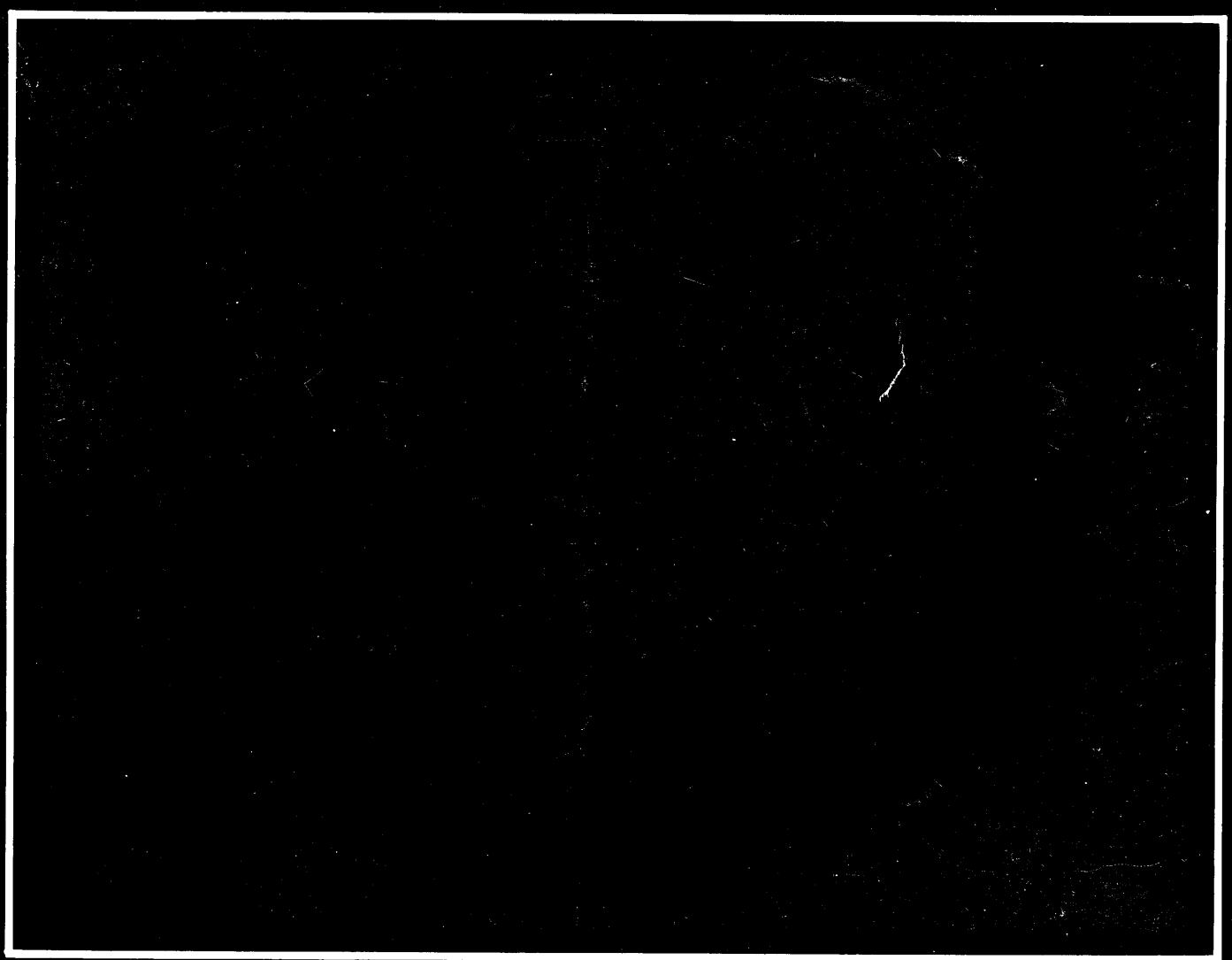
DAI NAMIC

tweemaandelijks tijdschrift september - oktober 1984



een uitgave van dainamic v.z.w.
verantw. uitgever w. hermans, mottaart 20 - 3170 herselt

COLOFON

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

DAInamic redactie :

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

Vormgeving : Ludo Van Mechelen.

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

DAInamic verschijnt de pare maanden.
Bijdragen zijn steeds welkom.

CORRESPONDENTIE ADRESSEN.

Redactie en software bibliotheek

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

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

Lidgelden / Subscriptions

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

Voor Nederland :

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

Generale Bankmaatschappij Leuven
nr. 230-0045353-74

Inzendingen : Games & Strategy

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

DAINAMIC
PERSONAL COMPUTER USERS CLUB

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

belangrijke ASCII-waarden in DAIpc

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

ASCII - HEX - ASCII CONVERSION TABLE

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

DAInamic 24

Herselt, oct '84

Beste Leden,

De kalender meldt weer tal van activiteiten gedurende de volgende maanden:

20 october : DAI-dag te Nijvel (B)
17 november : HCC-dag te Utrecht (NL)
24 november : onderwijs-symposium te Diepenbeek (B)
? april '85 : HCC-dag Belgie te Deurne (B)
? april '85 : DAInamic-dag te Westerlo (B)

We hopen dat dit lijstje nog uitgebreid wordt met de opening van ons nieuw lokaal (op het redactieadres), onze huidige werkplaats te Westmeerbeek is echt te eng aan het worden.We zullen dan ook de mogelijkheid hebben om regelmatig het lokaal open te stellen voor leden die eens willen langs komen voor informatie, software of zomaar een gesprekje... Door de dag in Nijvel is ook de verzending van dit nummer een week vervroegd , nog zo'n inhaalmanoeuver en onze publicatie verschijnt weer op tijd ! Op de voorpagina van dit nummer kijkt Nico Looije U ernstig aan : deze foto is een van de eerste resultaten met de Video Camera Interface. Spijtig genoeg zijn momenteel de experimenten gestopt door een panne aan het apparaatje. Voor alle duidelijkheid : we hebben de interface gekocht bij de firma ENTRAC bv te Diemen en Nico heeft op zeer korte tijd de nodige software geschreven.(bij het apparaat worden alleen programma's voor de BBC geleverd.)

Tot op een van de volgende manifestaties ...

Dear members,

Above you see the list of manifestations during the coming months. We also hope to announce soon the oficial opening of our new workshop. Did you notice that this issue is one week earlier in your mailbox? On the cover you find Nico Looije staring at you through our Video Camera Interface , for the moment there are some problems with the unit but you will sure find the whole story and software description in one of the coming issues. In this issue another 3 new software titles from our library : this should bring the total number of packages on 80 (take an average of 5 titles per collection : this means 400 programs available now !) the new titles and prices :

GAMES 13 : 750 Bfr (900 Bfr on DCR)
GAMES 14 (PUZZLE 1) : 750 Bfr (900 Bfr on DCR)
HELP (S.O.S. HELI) : 1750 Bfr (from DIALOGUE)

Through contact with the German club, the HCC-DAI users group and the local clubs in Liege,Namur and Brussels , we have a lot of articles available , but translation to English should be done : please contact us if you can translate from Dutch,French or German to English, your efforts will be rewarded with free software !

We hope to meet you on one of the coming shows ...

W.H.

REMARK

NEWSLETTER 24

INHOUD — CONTENTS

DAInamic subscription rates :

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

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

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

P R O G R A M M E E R T E C H N I E K E N

Na MODE 0 wil ik nu deze keer een aantal onderwerpen de revue laten passeren die daar nauw mee samenhangen. Als eerste wil ik de PRINT-instructie bekijken. Het woord 'PRINT' is voor veel computergebruikers eigenlijk al niet meer correct. Vroeger (15 a 20 jaar geleden ?) droomde men alleen maar van een beeldscherm dat door de computer zou worden aangestuurd. Nee, in de hobbysfeer, dwz betaalbaar met minder dan een jaarsalaris kon U de uitkomsten van uw computer alleen maar op papier krijgen. Vaak was de printer, die werd gebruikt dan ook nog alleen maar een soort telstrook met een matige drukkwaliteit. Maar om iets te zien te krijgen (op papier dus) was de instructie 'PRINT' logisch. Zou men BASIC nu schrijven zou het mogelijkerwijs iets als 'SHOW' of 'DISPLAY' geworden zijn. Later in de geschiedenis kreeg men betere en snellere printers en kwam ook het beeldscherm. Nu ontstond echter de behoefte om de eventueel ook aangesloten printer niet altijd te laten meewerken. Vele fabrikanten losten dit op door de printer altijd uit te zetten en alleen te laten werken bij opdrachten die daar zelf om vroegen.

N.B. Eigenlijk wordt niet de printer uitgezet, maar krijgt de printer geen signalen van de computer meer door via zijn aansluiting.

Zo kreeg men naast PRINT de LPRINT en naast LIST de LLIST. Deze methode heeft echter het nadeel dat het programma niet simpel met of zonder printer gebruikt kan worden. Veel gebruikers programmeerden dan ook vaak met gebruik van printer en zetten de printer zelf dan af om geen papier te verspillen. Bij de DAI heeft men gelukkig niet voor deze oplossing gekozen. Er is gekozen voor de volgende oplossing; de DAI heeft adres #131 waarin een waarde staat. Deze waarde geeft aan waar de 'OUTPUT' heen moet, zie de tabel hieronder.

```
#131 = 0 - - uitvoer naar beeld en RS 232
#131 = 1 - - uitvoer naar beeld
#131 = 2 - - uitvoer naar de EDIT-buffer
#131 = 3 - - uitvoer volgens een speciale routine
             deze routine kan eventueel zelf geschreven worden
             elke waarde tussen 3 en 256 heeft hetzelfde effect.
```

Uit bovenstaand overzicht blijkt al dat het in de meeste gevallen het handigst is de printer via de serieele RS 232 poort aan te sluiten; daar de printer dan met een eenvoudige POKE #131,1 kan worden losgekoppeld en met een POKE #131,0 weer aangekoppeld. Wilt U de printer zelf toch liever parallel aansturen is er altijd een goedkope interface te koop die het seriele signaal omzet in een parallel signaal. Daarbij blijft de DCE-bus die signalen parallel uitstuurt vrij voor b.v. floppy of DCR. Toch hebben meerdere DAI-gebruikers gekozen voor een aansluiting via de DCE-bus. De POKE #131,3 wordt gebruikt door de DAI-floppy, die met deze POKE de uitvoer door een routine in de DOS laat verzorgen. Dat wat uitgevoerd moet worden geeft de gebruiker zelf op in een PRINT tussen de dubbele quotes. ("uit te voeren opdracht")
Lastig is dit wel : eerst de uitvoer (van de PRINT's) de goede kant opzetten, dan de opdracht zelf in een PRINT geplaatst en dan meestal de uitvoer weer terugzetten naar beeld en/of RS 232. Voor mensen, die het leuk lijkt eens wat te experimenteren met een zelf geschreven outputroutine nog wat nadere gegevens. Als U een POKE #131,3 geeft dan zal de output verzorgt worden door een CALL naar de DOUTC op #2DD. Op dit adres staat initieel #C9 (= RETURN). We kunnen dit echter veranderen in een sprong naar een eigen routine. We zetten C3 00 03 (of iets soortgelijks) op #2DD,#2DE en #2DF en laten onze routine dan beginnen op #300. De routine moet dan wel eindigen met een RETURN (=C9).

Jammer genoeg is het niet direct mogelijk de output alleen naar RS 232 te sturen. Een handige toepassing zou ik daarvoor al kunnen geven. Als we een stuk

machinetaal willen testen kunnen we met L (van LOOK) een aantal gegevens (registers) bijhouden, daar de inhoud op het scherm gezet wordt. Daardoor kunnen we in splitmode komen te werken en kan het programma, alleen al daardoor, verkeerd werken. Sturen we nu de gegevens van L alleen naar de printer (dwz DCE-bus / RS 232) wordt ons beeld niet vernield door de L. Iets soortgelijks geldt in BASIC bij gebruik van de TRON - instructie. In BASIC wordt weliswaar alles correct uitgevoerd, maar in de noodgedwongen splitmode is niet altijd alles te zien. Als iemand zich geroepen voelt dit uit te werken; er is bij voorbaat een plaatsje op de volgende TOOLKIT en/of volgend tijdschrift ingeruimd.

*** Als u bij een listing duidelijk wilt laten zien van welk type elke ***
TIP variabele is moet U eens voordat U LIST geeft eerst de implicit type TIP
*** table vullen met #80 door in UTility in te tikken F275 28E 80[return]***

In de rest van mijn verhandeling ga ik uit van een standaard MODE 0 tenzij anders vermeld. Uitvoer start altijd op die positie , waar de cursor zich op dat moment bevindt. Wordt bij uitvoer naar scherm meer tekst aangeboden dan op de regel past waar we aan het printen zijn wordt automatisch naar de volgende regel gegaan. Het afdrukken wordt dan hervat na een C op de eerste positie van die volgende regel gevolgd door zes spaties. Deze automatische continuering op de volgende regel geschiedt slechts drie maal achtereen. Hieruit volgt dat het maximale aantal tekens dat met een PRINT naar het scherm kan worden uitgevoerd 4 x 60 - 3 x 7 = 240 - 21 = 219 bedraagt. Dit kan wel eens tot problemen leiden om dat we in de EDIT - buffer wel 256 tekens op een regel kunnen krijgen. Bij terugkomst uit EDIT krijgen we dan 'LINE TOO COMPLEX'. Overigens kunt U na 256 posities wel degelijk verder de EDIT - buffer vullen, maar er is dan niets van op het scherm te zien. Willen we meer dan drie C - regels moeten we op #7B de DAI misleiden door daar steeds weer nul (evt 1 of 2) in te plaatsen.

De cursorpositie is horizontaal van links naar rechts genummerd van 0 t/m 59, Verticaal van onder naar boven van 0 t/m 23. Dit alles voor de normale MODE 0, in de splitmodes horizontaal gelijk aan MODE 0, maar verticaal van onder naar boven van 0 t/m 3.

Hoe kunnen we nu de PRINT gebruiken ? Ik loop de verschillende mogelijkheden eens met U na.

PRINT A — met A is variabele. Deze variabele mag gevolgd worden door een typeaanduiding : ! (Floating PoinT) ,% (INTeger) of $ (STRing). Dit is echter niet nodig.

PRINT "tekst" — de tekst, die tussen de dubbele quotes (") staat wordt nu uitgevoerd.

, ; + — Dit zijn symbolen die bij PRINT gebruikt kunnen worden.

Het teken , (komma) kunnen we achter de output (tekst of variabele) zetten. De cursor zal in dat geval de volgende TAB - positie opzoeken. De standaard TAB-posities zijn 0,12,24,36 en 48. Voor zover mij bekend, zijn deze posities niet door de gebruiker te wijzigen. Er is echter nog iets bijzonders op te merken over deze cursorverplaatsing. Staat de cursor op x-positie 0 dan blijft hij daar op staan, staat hij op 12,24,36 of 48 gaat hij in dat geval naar 24,36,48 of 0. Inderdaad 0; de C wordt in dit geval niet gebruikt en het aantal regels is niet beperkt. In dit geval zit er dus altijd minstens een spatie tussen de output's. Om dit te controleren moet U de volgende twee programa's maar eens intikken.

```
10    FOR I=1 TO 100:PRINT "abc",:NEXT
```

```
10    FOR I=1 TO 100:PRINT "123456789012",:NEXT
```

U zult overigens zien dat als U in bovenstaande voorbeelden PRINT I, zou opgeven er schijnbaar niet op positie 0 wordt begonnen. Getallen worden echter afgedrukt met een min (-) ervoor als ze negatief zijn, de plus als ze positief zijn wordt echter weggelaten. Er mag overigens best output gegeven worden van meer dan 12 karakters. Na afdrukken wordt de cursor verplaatst naar de volgende TAB-positie, tenzij die toevallig op 0 stond.

Het teken ; (puntkomma) kunnen we achter de output (tekst / variabele) zetten. De cursor zal in dat geval blijven staan. De output, die erna komt, zal er dan direct achter worden gezet. Wordt in dit geval het einde van de regel bereikt zal er wel met een continueringsregel (dwz een C en zes spaties) worden doorgegaan. Zonder de POKE #7B,0 zullen er dan ook maar drie vervolgregels kunnen. De POKE #7B,0 moet wel steeds ververst worden.

```
xxx     POKE #7B,0:FOR I=1 TO 100:PRINT I;:NEXT      is  f o u t.

xxx     FOR I=1 TO 100:PRINT I;:POKE #7B,0:NEXT      is  g o e d.
```

Merk op dat de getallen schijnbaar gescheiden worden door een spatie . In werkelijkheid zijn het allemaal onderdrukte plussen.

Het teken + (plus) kan alleen gebruikt worden tussen twee outputs in die van hetzelfde type zijn. Getallen worden opgeteld en bij teksten heeft het schijnbaar dezelfde functie als de puntkomma. Als U de volgende voorbeelden inkijkt zult U mogelijk geen verschil ontdekken; dan maar intikken en RUN geven.

```
10   PRINT "AB";"123456"              &   20   PRINT "AB"+"123456"

30   A$="CBA":C=123456:PRINT A$;C     &   40   A$="CBA":C=123456:PRINT A$+C

50   PRINT "ABCDEF";"1234567"         &   60   PRINT "ABCDEF"+"1234567"
```

En weet U de verschillen ? Bij intikken blijkt dat regel 40 domweg niet kan en we krijgen een TYPE MISMATCH. Goed, nu geven we RUN. Regel 10 en 20 doen schijnbaar hetzelfde evenals 50 en 60. Bij regel 30 staat er ineens een spatie tussen tekst en getal. Na enig nadenken begrijpen we, dat bij de overige regels er twee teksten staan en bij regel 30 staat na de tekst een getal en dat begint met een onderdrukte plus, dus spatie. Maar ook tussen 10 en 20 is een verschil. Geef maar eens CLEAR 6 en dan RUN. U zult zien 10 wordt uitgevoerd maar 20 niet en U krijgt de melding OUT OF STRING SPACE. Pas na een CLEAR 12 of een groter getal zal ook regel 20 worden uitgevoerd. Is de CLEAR echter nog kleiner dan 22 zal nu regel 60 niet uitgevoerd worden. Het vermoeden is dus gerechtvaardigd dat we met de '+' de HEAP gebruiken en met de ';' niet. Laten we eens kijken in de HEAP. Op #29B en #29C vinden we standaard #EC en 02 waarmee aangegeven wordt dat de HEAP op #2EC begint. En inderdaad vinden we hier (vanaf #2EE) de tekst waar er een '+' bij werd gebruikt. PRINTen we dus een of andere output zonder '+' worden alle teksten netjes naar het beeld gebracht. Gebruiken we echter wel de '+' zal de output eerst in de HEAP worden bijeengebracht en pas daarna naar het scherm gaan. Is de HEAP te klein, beter : is er te weinig ruimte vrij in de HEAP, krijgen we een OUT OF STRING SPACE melding.

De POKE #131,3 voor een PRINT met een '+' zal het resultaat niet op beeld zetten, maar wel in de HEAP laten verschijnen. Getallen komen trouwens voor hun uitvoer eerst op #E3 t/m #F1 en U zou ze daar na een loze PRINT dwz PRINT na POKE #131,3 , kunnen ophalen voor verdere bewerking. Vreemd genoeg staan ook integers daar met een decimale punt genoteerd.

Er is nog een aardige toepassing van het verschil tussen '+' en ';'. U kent allen ongetwijfeld de INPUT-statement. Om de gebruiker van een programma om

een bepaalde waarde te vragen kunt U natuurlijk de volgende methode gebruiken.

```
10 PRINT "Wat is de beginwaarde";:INPUT BW:PRINT
```

Maar beter vind ik : (Let ook op de spatie aan eind van tekst !)

```
10 INPUT "Wat is de beginwaarde ";BW:PRINT
```

omdat na een BREAK en een CONT de tekst weer bij gegeven wordt. Ook als iemand keer op keer iets verkeerds intikt zoals een te groot getal, gevolg OVERFLOW RETYPE LINE. Dan kan zeker in splitmodes nogal snel uit beeld verdwijnen wat er gevraagd wordt. Maar hoe kunnen we de tekst van de INPUT nu veranderen ? Nu, dit is erg simpel.

```
10      INPUT "Wat is je"+LEFT$(STR$(P),LEN(STR$(P))-2)+"e poging ";INV
```

met INV het getal dat ingevoerd wordt en P het nummer van de poging.

Tot slot de behandeling van het afdrukken van getallen. Vele programmeurs hebben gemerkt dat, als men daar niets tegen onderneemt, de getallen die worden afgedrukt altijd op ..0 eindigen. Dit is natuurlijk niet zo. Als U de machine aanzet wordt automatisch een IMPFPT gegeven, dwz alle getalvariabelen worden geacht van het type floating point, dus met een decimaal deel, te zijn. Als U dit niet wilt kunt U zelf IMPINT geven.

```
*** Alle variabelen van het stringtype maken is ook mogelijk middels een ***
TIP IMPSTR A-Z.  Zonder volgletters dus alleen IMPSTR is niet mogelijk.  TIP
***                                                                      ***
```

Ik ben al vele mogelijkheden om nu de .0 weg te krijgen tegengekomen. Ik laat er enige de revue passeren.

```
10    PRINT MID$(STR$(W!),0,LEN(STR$(W!))-2)
20    W$=STR$(W!):PRINT LEFT$(W$,LEN(W$)-2)
30    PRINT W!+CHR$(8)+CHR$(8)
40    B$=CHR$(8):PRINT W!+B$+B$
50    B$=CHR$(8):B$=B$+B$:PRINT W!+B$
60    PRINT W!;:CURSOR CURX-2,CURY:PRINT "  "
```

En de natuurlijk enige echt redelijke methode.

```
90    W%=W!:PRINT W%
```

Nog beter is het natuurlijk, om als U met integerwaarden werkt, U dan ook integers gebruikt. Een nadeel van al de methoden voor regels 90 is dat een getal niet meer dan een cijfer achter de komma mag hebben en het niet zo mag zijn dat een E-notatie nodig is.

Een ander probleem waar velen mee zitten is het uiterlijk van de getallen. Men wil graag dat elk getal twee cijfers achter de komma heeft. Dus ook 00 en 04 en 20 moeten mogelijk zijn. Dit wordt dan nog vaak gecombineerd met de wens dat de getallen in een kolom met de komma's onder elkaar staan. Vooral bij financiele problemen zeer zinnige wensen. De beste oplossing die ik kon bedenken gaat er van uit dat U de bedragen in integers op centen nauwkeurig bijhoudt. Pas bij het afdrukken worden het dan guldens en centen.

Voordat ik de oplossingen, die ik bedacht heb, geef wil ik eerst een aantal problemen met u doornemen. Mijn eerste gedachte om de getallen zo onder elkaar te krijgen, dat de rechterkanten een mooie rechte lijn vormen was het gebruik van de LOGT-functie. De grondgedachte is zeer simpel : LOGT(x) is de waarde

waartoe men 10 (Tien vandaar de T in LOGT) moet verheffen om x te krijgen. Voor de wiskundig minder geschoolden geef ik het volgende tabelletje :

getal tussen	LOGT tussen	getal	LOGT
		1	0
1 en 10	0 en 1	10	1
10 en 100	1 en 2	100	2
100 en 1000	2 en 3	1000	3
1000 en 10000	3 en 4	10000	4
enz.	enz.	enz.	enz.

Theoretisch nemen we dus de LOGT van een getal, maken er een integer van en ronden hem daarmee eventueel naar beneden af en tellen daar dan nog een bij op. We hebben dan precies het aantal cijfers waaruit het getal bestaat. Als dit aantal b.v. A heet kunnen we de getallen keurig onder elkaar krijgen met CURSOR 30-A,CURY:PRINT GETAL of CURSOR 29-LOGT(GETAL),CURY:PRINT GETAL.

Maar dit blijkt bij controle niet altijd te werken. LOGT werkt met floating points. 10, 100, 1000 enz. worden dan als exponent van 2 opgeslagen; met een noodzakelijke afrondingsfout. De LOGT heeft dientengevolge ook een afrondingsfout. Deze fout is zeer klein, maar in ons geval fundamenteel tot fouten leidend. De CURSOR instructie werkt met integers en zal de uit LOGT voortkomende waarde transformeren naar dit type. En hiermee krijgen we dan onze fouten. Als U mij niet gelooft moet U maar eens intikken: PRINT LOGT(10000) U krijgt als antwoord 4.0, Nu intikken A%=LOGT(10000):PRINT A% en U krijgt 3 op het scherm. Een heel klein getal bij de LOGT tellen voor conversie is mogelijk maar kies dat getalletje dan wel zo klein, dat 9999 ed niet weer een positie verkeerd gaat. Een goede en snelle oplossing geef ik nu hieronder:

```
10      REM RIGHT ALIGNMENT / F.H. DRUIJFF - 9/84
20      INPUT A:W=1:FOR L=0 TO 10:W=W*10:IF A>=W THEN NEXT
30      CURSOR 30-L,CURY:PRINT A:GOTO 20
```

Maar we zijn nog niet tevreden en willen de getallen ook nog van een komma (of punt) voorzien en die netjes onder elkaar. U wordt op uw wenken bedient :

```
10      REM RIGHT ALIGNMENT % DECIMAL COLUMN / F.H. DRUIJFF - 9/84
20      INPUT A:G=A/100:D=A MOD 100:W=1
30      FOR L=0 TO 10:W=W*10:IF G>=W THEN NEXT:CURSOR 30-L,CURY:PRINT G;
40      IF D>=10 THEN PRINT D;:CURSOR CURX-3,CURY:PRINT ",":GOTO 20
50      PRINT ",";D;:CURSOR CURX-2,CURY:PRINT "0":GOTO 20
```

Maar ik wilde een nog gebruikers-vriendelijker oplossing en gebruikte daarvoor de eerder besproken adressen #E3 t/m #F1. Deze laatste oplossing is ook aan te passen om getallen zonder de E-notatie af te drukken. maar met meerdere nullen aan het begin of eind. Bestudeer de listing en U ziet wel de mogelijkheden.

```
10      REM PRINT USING / F.H. DRUIJFF - 9/84
20      USING$="311111110200":LU=LEN(USING$)   USING$ bevat de gewenste vorm
30      DIM T$(3):T$(0)="0":T$(1)=" ":T$(2)=",":T$(3)="$"          $ is optie
40      INPUT A:CURSOR 30,CURY:T$=""                              initialisatie
100     POKE #131,3:PRINT A:POKE #131,0          dummy getalopbouw in #E3-#F1
110     E3=#E3+PEEK(#E3)+1:I=LU-1                                 initialisatie
120     J=VAL(MID$(USING$,I,1))                                    bepaal code
130     IF J>1 THEN T$=T$(J)+T$:GOTO 160                       speciaal karakter
140     E3=E3-1:IF E3<=#E4 THEN T$=T$(J)+T$:GOTO 160      aanvullend karakter
150     T$=CHR$(PEEK(E3))+T$                                     echte karakter
160     I=I-1:IF I>=0 GOTO 120:PRINT T$:GOTO 40        eindcontrole en afdruk
```

Frank H. Druijff

NEW!

Games collection 13

Een ongelukkig nummer, maar wel een verzameling die veel te bieden heeft.

1	Domino J. Overvoorde	Het bekende spel nu op de DAI. U speelt tegen de DAI en U kunt dus niet meer vals spelen. Bekijk ook eens de waarlijk schitterende listing. Een echte topper !
2	Deflector N.P. Looije	Een leuk actie spel, waarbij duidelijk blijkt dat de DAI superieur is aan vele andere machines.
3	Boggle H.P. Legry	Geen letters meer op hun kop en de tijd wordt exact gemeten. De DAI maakt de score op.
4	Termite R. Cuypers	Laat de termieten door een boomstam knagen. Onthoud waar de oneetbare stukken zitten. Vier speelniveaus.
5	Jeu de Retourne H.P. Legry	Zoek de kaarten die bij elkaar horen . Grafisch zeer fraai. Ook bekend onder de naam 'memory'.
6	Maanlander M. van Toer	Naam spreekt voor zichzelf. Het is een programma dat weer eens demonstreert, dat je ook met weinig BASIC-regels veel speelgenoegen kan geven.

NEW!

Games collection 14

THE PUZZLE COLLECTION 1

Een nieuw type collectie, dat U een aantal breinkrakers voorschotelt. De programma's zijn er op uit U aan het denken te zetten. Alleen voor actieve leden.

1	Verschillen J. Overvoorde	Bekend van de T.V. U krijgt een aantal verschillende zaken (huizen, kastelen, auto's ed) getoond; twee ervan zijn hetzelfde aan U om uit te zoeken welke. Jeroen heeft hierbij zichzelf overtroffen. Dit programma is voor programmeurs een lichtend voorbeeld. De beste inzending in jaren.
2	Jig Saw N.P. Looije	Doe de legpuzzle nu eens op het beeldscherm. Weinig stukjes maar lastiger dan je vermoedt.
3	Black Box Ph. Demoulin	Zend stralen door een doos met onbekende inhoud. Aan U de taak te deduceren waar de doos gevuld is.
4	Teaser A. Bathe	Wat lijkt dit spel gemakkelijk en wat is het lastig ! X'n en O's veranderen om tot de winnende combinatie te komen. Tik tak tor in het kwadraat.
5	Even Wint E. Smet	Als U de winnende formule niet kent zal deze puzzle U veel hoofdbrekens kosten. Sterk zijn en niet LIST !
6	Wijn hevelen J. van Dunne	U blijft aan het overgieten. De opdracht luidt , meet precies 10 liter af of halveer de totale hoeveelheid.
7	African River B. Gortz	Zorg ervoor dat een aantal mensen veilig aan de overkant van de rivier komt. Moeilijke 'wolf,geit,kool'.

ADD r	10000sss	Z,S,P,C,AC	1	4

Add register to accumulator
De inhoud van een register wordt opgeteld bij de inhoud van de accumulator en het resultaat wordt in de accumulator geschreven.
(A) + (r) → (A)

ADD M	10000110	Z,S,P,C,AC	1	7

Add memory to accumulator
De inhoud van de geheugencel geadresseerd door het HL registerpaar wordt opgeteld bij de accumulator en wordt in de accumulator gezet.
(A) + (M) → A met M = (HL)

ADC r	10001sss	Z,S,P,C,AC	1	4

Add register to accumulator with carry
Deze instructie telt niet alleen de inhoud van een register bij de accumulator maar tevens wordt de carry bit opgeteld.
(A) + (r) + C → A

ADC M	10001110	Z,S,P,C,AC	1	7

Add memory to accumulator with carry
De accumulatorinhoud en de overdrachtbit en de inhoud van de geheugencel geadresseerd door het HL registerpaar worden opgeteld en het resultaat wordt in de accumulator gezet.
(A) + (M) + C → A met M = (HL)

DAD rp	00rp1001	- - -,C,-	1	10

Double add
De inhoud van een registerpaar B, D, H of SP wordt opgeteld bij de inhoud van het registerpaar HL en het resultaat komt in het registerpaar HL.
(HL) + (rp) → HL

SUB r	10010sss	Z,S,P,C,AC	1	4

Substract register from accumulator
De inhoud van het register wordt afgetrokken van de inhoud van de accumulator en het resultaat wordt in de accumulator geschreven.
Bij een overloopbit wordt echter wegens de complementmethode de carrybit teruggezet op 0.

SUB M	10010110	Z,S,P,C,AC	1	7

Substract memory from accumulator
De inhoud van de geheugencel geadresseerd door het HL registerpaar, wordt afgetrokken van de inhoud van de accumulator; het resultaat wordt ingeschreven in de accumulator.
(A) - (M) → A

SBB r	10011sss	Z,S,P,C,AC	1	4

Substract register from accumulator with borrow
De inhoud van het register en de carry bit wordt afgetrokken van de inhoud van de accumulator; het resultaat wordt in de accumulator ingeschreven
(A) - (r) - C → A

SBB M	10011110	Z,S,P,C,AC	1	7

Substract memory from accumulator with borrow
De inhoud van de geheugencel geadresseerd door het HL registerpaar en de carrybit worden afgetrokken van de inhoud van de accumulator; het resultaat wordt ingeschreven in de accumulator.
(A) - (M) - C → A

ADI const	11000110	Z,S,P,C,AC	2	7

Add immediate to accumulator
De constante (d_8) wordt opgeteld bij de accumulatorinhoud en het resultaat wordt ingeschreven in de accumulator.
(A) + const → A

ACI const	11001110	Z,S,P,C,AC	2	7

Add immediate to accumulator with carry
De constante (d_8) en de carry bit worden opgeteld bij de accumulator. Het resultaat wordt in de accumulator geschreven.
(A) + const + C → A

SUI const	11010110	Z,S,P,C,AC	2	7

Substract immediate from accumulator
De constante d_8 wordt afgetrokken van de accumulatorinhoud. Het resultaat wordt in de accumulator geschreven.
(A) - const → A

SBI const	11011110	Z,S,P,C,AC	2	7

Substract immediate from accumulator with borrow
De constante en de carry bit worden afgetrokken van de inhoud van de accumulator. Het resultaat wordt in de accumulator geschreven.
(A) - const - C → A

DAA	00100111	Z,S,P,C,AC	1	4

Decimal adjust accumulator
De inhoud van de accumulator wordt zo behandeld dat de twee tetrades de juiste waarde van de inhoud in de BCD code geven, zodat rechtstreeks de BCD waarde kan uitgevoerd worden. De DAA instructie is de enige instructie waarbij de AC gebruikt wordt.

4.2.3. Logische instructies

CMA	00101111	- - - - -	1	4

Complement accumulator
De inhoud van de accumulator wordt gecomplementeerd. Alle nullen worden vervangen door énen en vice versa.
(A) → ($\overline{A}$)

ANA r	10100sss	Z,S,P,AC,C	1	4

AND register with accumulator
De logische EN operatie wordt uitgevoerd op de inhoud van de accumulator en deze van het register. Het resultaat wordt in de accumulator geplaatst.
(A) AND (r) → A

ANA M 10100110 Z,S,P,AC,C 1 4

AND memory with accumulator
De EN operatie gebeurt nu met de inhoud van de accumulator en deze van een geheugencel geadresseerd door het registerpaar HL. Het resultaat wordt in de accumulator geplaatst.
(A) AND (M) → A met M = (HL)

ANI const 11100110 Z,S,P,C,AC 2 7

AND immediate with accumulator
De EN operatie wordt uitgevoerd op de inhoud van de accumulator en een constante (d_8). Het resultaat wordt in de accumulator geschreven.
(A) AND const → A

ORA r 10110sss Z,S,P,C,AC 1 4

OR register with accumulator
De OF operatie wordt uitgevoerd op de inhouden van de accumulator van een register. Het resultaat wordt in de accumulator geschreven.
(A) OR (r) → A

ORA M 10110110 Z,S,P,C,AC 1 7

OR memory with accumulator
De OF operatie wordt uitgevoerd op de inhouden van de accumulator en van een geheugencel geadresseerd door het HL registerpaar. Het resultaat wordt in de accumulator geschreven.
(A) OR (M) → A met M = (HL)

ORI const 11110110 Z,S,P,C,AC 2 7

OR immediate with accumulator
De OF operatie wordt uigevoerd op de inhoud van de accumulator en een constante d_8. Het resultaat wordt in de accumulator ingeschreven.
(A) OR const → A

XRA r 10101sss Z,S,P,C,AC 1 4

EXCLUSIVE OR register with accumulator
De EXCLUSIEVE OF operatie wordt uitgevoerd op de inhoud van de accumulator en van een register. Het resultaat wordt in de accumulator geschreven.
(A) ⊕ (r) → A

XRA M 10101110 Z,S,P,C,AC 1 7

EXCLUSIVE OR memory with accumulator
De EXCLUSIEVE OF operatie wordt uitgevoerd met de inhoud van de accumulator en met de inhoud van een geheugencel geadresseerd door het HL registerpaar. Het resultaat wordt in de accumulator geschreven.
(A) ⊕ (M) → A met M = (HL)

XRI const 11101110 Z,S,P,AC,C 2 7

EXCLUSIVE OR immediate with accumulator
De EXCLUSIEVE OF operatie wordt uitgevoerd op de inhoud van de accumulator en een constante (d_8). Het resultaat wordt in de accumulator geschreven.
(A) ⊕ d_8 → A

CMP r	10111sss	Z,S,P,AC,C	1	4

Compare register with accumulator
De inhoud van de accumulator wordt vergeleken met de inhoud van een register. De vergelijking wordt uitgevoerd door aftrekking van de registerinhoud van de accumulatorinhoud. Noch de registerinhoud, noch de accumulatorinhoud worden veranderd. Het resultaat beïnvloedt alleen de toestandenbits.
Bij C = 0 is (A) > (r) als Z = 0
is (A) = (r) als Z = 1
Bij C = 1 is (A) < (r)

CMP M	10111110	Z,S,P,C,AC	1	7

Compare with accumulator
De inhoud van de accumulator wordt vergeleken met de inhoud van een geheugencel geadresseerd door het HL registerpaar. Voor de modaliteiten van deze vergelijking zie CMP r.

CPI const	11111110	Z,S,P,C,AC	2	7

Compare immediate with accumulator
De inhoud van de accumulator wordt vergeleken met een constante (d_8). Voor de modaliteiten van deze vergelijking zie CMP r

4.2.4. Instructies voor de behandeling van een registerinhoud

a) Verschuivingen van accumulatorinhoud

RLC	00000111	- - - C -	1	4

Rotate left contenu
De inhoud van de accumulator wordt over één plaats naar links rondgeschoven.
Het rondschuiven gebeurt zó dat alle bits 1 plaats verschuiven naar links, met dien verstande dat D_7 de plaats inneemt van D_0. De waarde die D_7 had voor het rondschuiven komt in de carry.

C	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	voor uitvoering RLC

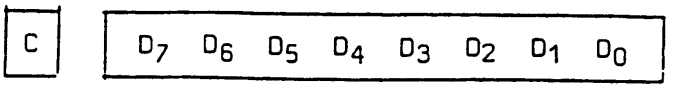
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	na uitvoering RLC

RRC	00001111	- - - C -	1	4

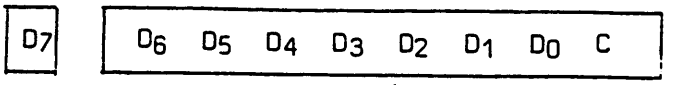
Rotate right contenu
De inhoud van de accumulator wordt over één plaats naar rechts rondgeschoven.
Alle bits schuiven 1 plaats naar rechts met dien verstande dat D_0 de plaats inneemt die D_7 had. De bit D_0 schuift eveneens door naar de carry.

RAL	00010111	- - - C -	1	4

Rotate accumulator left trough carry
Alle bits van de accumulatorinhoud, met inbegrip van de carrybit worden over één plaats naar links rondgeschoven.



voor uitvoering RAL



na uitvoering RAL

RAR 00011111 - - - C - 1 4

Rotate accumulator right trough carry

Alle bits van de accumulatorinhoud, met inbegrip van de carrybit worden over een plaats naar rechts rondgeschoven.

b) Instructies betreffende de carry bit

CMC 00111111 - - - C - 1 4

Complement carry
De carrybit wordt geïnverteerd

STC 00110111 - - - C - 1 4

Set carry
Carrybit wordt 1.

4.2.5. Springinstructies

a) Onvoorwaardelijke springinstructies

PCHL 11101001 - - - - - 1 5(6)

HL to program counter
Het adres dat zich in het HL registerpaar bevindt, wordt overgebracht naar de programmateller. Het programma wordt verder gezet vanaf dit adres.
(HL) → PC

JMP 11000011 - - - - - 3 10

Jump unconditional
Het programma wordt verder gezet vanaf adr. De programmateller wordt geladen met het adres aangegeven in de springinstructie

b) Voorwaardelijke springinstructies

Deze springinstructies worden slechts uitgevoerd als aan een bepaalde voorwaarde, aangegeven door de instructie, voldaan is, zoniet wordt het programma verder gezet met de instructie die volgt op de springinstructie. De voorwaarden die in de instructie voorkomen zijn aangegeven door één van de toestandenbits C,Z,P,S.

JC adr 11011010 - - - - - 3 10(7/10)

Jump on carry
Als de carrybit 1 is, wordt het programma verder gezet op het adres aangegeven door de instructie.

JNC adr	11010010	- - - - -	3	10(7/10)

Jump on no carry
Als de carry bit 0 is, wordt gesprongen naar het aangegeven adres.

JZ adr	11001010	- - - - -	3	10(7/10)

Jump on zero
Als de nul toestandsbit gelijk is aan 1, wordt gesprongen naar het aangegeven adres.

JNZ adr	11000010	- - - - -	3	10(7/10)

Jump on no zero
Als de nul toestandsbit nul is, wordt gesprongen naar het aangegeven adres.

JM adr	11111010	- - - - -	3	10(7/10)

Jump on minus
Als de tekenbit 1 is, wordt gesprongen naar het aangegeven adres.

JP adr	11110010	- - - - -	3	10(7/10)

Jump on positive
Als de tekenbit 0 is, wordt gesprongen naar het aangegeven adres.

JPE adr	11101010	- - - - -	3	10(7/10)

Jump on parity even
Als de pariteitsbit 1 is, wat overeenstemt met een even pariteit, wordt gesprongen naar het aangegeven adres.

JPO	11100010	- - - - -	3	10(7/10)

Jump on parity odd
Als de pariteitsbit 0 is, wat overeenstemt met oneven pariteit, wordt gesprongen naar het aangegeven adres.

4.2.6. Instructies betreffende de behandeling van subroutines

Deze instructies maken het mogelijk subroutines op te roepen hetzij onvoorwaardelijk, hetzij mits aan een bepaalde voorwaarde voldaan is.
Het beëindigen of afbreken van de subroutine kan eveneens onvoorwaardelijk gebeuren of mits beantwoording aan een bepaalde voorwaarde.

a) Onvoorwaardelijke oproep van subroutines

CALL adr	11001101	- - - - -	3	17

Call address
De subroutine die begint op het aangegeven adres wordt opgeroepen. Het betreft dus eveneens een soort springopdracht vermits het programma verder gezet wordt vanaf het aangegeven adres. Het adres van de instructie die onmiddellijk volgt op de CALL instructie, wordt in het stapelgeheugen (*stack*) geschreven om straks bij het verlaten van de subroutine het hoofdprogramma normaal te kunnen verderzetten.

b) Onvoorwaardelijke terugkeer uit een subroutine

RET	11001001	- - - - -	1	10

Return
Door deze instructie wordt de subroutine beëindigd en keert het programma terug naar het adres dat onmiddellijk volgt op de CALL opdracht. Dit adres staat ingeschreven in het stapelgeheugen.

c) Voorwaardelijke oproep van subroutine

De voorwaarden gebruikt in deze oproepen zijn weer aangegeven door de toestandenbits.

CC adr	11011100	- - - - -	3	11/17(9/18)

Call on carry
De subroutine wordt slechts opgeroepen als de carrybit gelijk is aan 1.
De duur van de instructie is 11 of 17 klokperiodes alnaargelang de subroutine opgeroepen wordt (17) of niet (11).

CNC adr	11010100	- - - - -	3	11/17(9/18)

Call on no carry
De subroutine wordt slechts opgeroepen als de carrybit gelijk is aan 0.

CZ adr	11001100	- - - - -	3	11/17(9/18)

Call on zero
De subroutine wordt slechts opgeroepen als de nulbit gelijk is aan 1.

CNZ adr	11000100	- - - - -	3	11/17(9/18)

Call on no zero
De subroutine wordt slechts opgeroepen als de nulbit 0 is.

CM adr	11111100	- - - - -	3	11/17(9/18)

Call on minus
De subroutine wordt slechts opgeroepen als de tekenbit 1 is.

CP adr	11110100	- - - - -	3	11/17(9/18)

Call on positive
De subroutine wordt slechts opgeroepen als de tekenbit 0 is.

CPE adr	11101100	- - - - -	3	11/17(9/18)

Call on parity even
De subroutine wordt slechts opgeroepen als de pariteitsbit 1 is, d.w.z. als de pariteit even is.

CPO adr	11100100	- - - - -	3	11/17(9/18)

Call on parity odd
De subroutine wordt slechts opgeroepen als de pariteitsbit nul is, d.w.z. als de pariteit oneven is.

d) Voorwaardelijke terugkeer opdrachten

Het afbreken van, of het terugkeren uit een subroutine kan weer afhankelijk zijn van de toestandenbits.

RC	11011000	- - - - -	1	5/11(6/12)

Return on carry
De terugkeer uit een subroutine gebeurt slechts indien de carrybit 1 is. De instructieduur is dan 11 klokperiodes, in het tegenovergestelde geval 5 klokperiodes.

RNC	11010000	- - - - -	1	5/11(6/12)

Return on no carry
De terugkeer uit de subroutine gebeurt slechts indien de carrybit nul is.

RZ	11001000	- - - - -	1	5/11(6/12)

Return on zero
De terugkeer uit de subroutine gebeurt slechts als de nulbit 1 is.

RNZ	11000000	- - - - -	1	5/11(6/12)

Return on no zero
De terugkeer uit de subroutine gebeurt slechts als de nulbit 0 is.

RM	11111000	- - - - -	1	5/11(6/12)

Return on minus
De terugkeer uit de subroutine gebeurt slechts als de tekenbit 1 is.

RP	11110000	- - - - -	1	5/11(6/12)

Return on positive
De terugkeer uit de subroutine gebeurt slechts als de tekenbit 0 is.

RPE	11101000	- - - - -	1	5/11(6/12)

Return on parity even
De terugkeer uit de subroutine gebeurt slechts als de pariteitsbit 1 is, d.w.z. als de pariteit even is.

RPO	11100000	- - - - -	1	5/11(6/12)

Return on parity odd
De terugkeer uit de subroutine gebeurt slechts als de pariteitsbit 0 is, d.w.z. als de pariteit oneven is.

4.2.7. Instructies betreffende de programma onderbreking (*interrupt*)

EI	11111011	- - - - -	1	4

Enable interrupt
Deze instructie laat onderbreking toe van het hoofdprogramma.

DI 11110011 - - - - - 1 4

Disable interrupt
In het programmagedeelte dat volgt na de DI instructie, is onderbreking onmogelijk, d.w.z. dat interruptaanvragen genegeerd worden.
Samen met deze twee instructies kan men gebruik maken van de zogenaamde *restart* (RST) instructies.

RST const 11nnn111 - - - - - 1 11(12)

Restart 0 to 7
Deze instructies ten getale van 8 (RST0 ... RST7) herstarten het programma na een interruptaanvraag op adressen verkregen door de constante 0 tot 7 te vermenigvuldigen met 8; dit geeft als adressen :

decimaal	hexadecimaal
0	Ø
8	8
16	10
24	18
32	2Ø
40	28
48	3Ø
56	38

Behalve voor RST 7 beschikken we dus over 8 geheugenplaatsen om een kleine subroutine te starten na elk van de RST opdrachten.
Bij de 8085 zijn nog vier verdere adressen voorzien voor de *restart* opdrachten :

RST 4,5 aangegeven onder de naam TRAP en met decimaal adres 36 (24 Hex).
RST 5,5 met decimaal adres 44 (2C Hex).
RST 6,5 met decimaal adres 52 (34 Hex).
RST 7,5 met decimaal adres 60 (3C Hex).

Drie van deze interruptinstructies nl. RST 5,5 6,5 en 7,5 zijn maskeerbaar.
De TRAP instructie daarentegen is niet maskeerbaar. Om de maskering te kunnen programmeren zijn twee bijkomende instructies voorzien, nl. RIM en SIM.

RIM 00100000 - - - - - 1 4

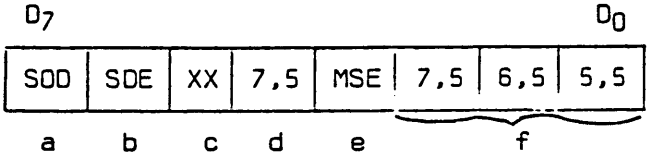
Read interrupt mask
Door deze instructie wordt de accumulator geladen met een 8 bits woord zodat de identificatie mogelijk wordt van de gemaskeerde interrupt adressen. Een bijkomende informatie geeft aan of voor één van de niveaus een interruptaanvraag heeft plaats gehad.
De samenstelling van het 8 bits woord is als volgt :

	I	I	I		M	M	M
SID	7,5	6,5	5,5	IE	7,5	6,5	5,5
a		b		c		d	

a. SID
Serial input data
De bit in dit vak indien aanwezig, geeft één bit van de informatie die in serie wordt ingebracht.
b. Een 1 in één van de vakken 7,5 6,5 of 5,5 geeft aan welke van de interruptadressen aangevraagd worden (I).
c. IE
Interrupt enable
IE heeft niet alleen betrekking op de specifieke interrupt adressen maar op het hele interrupt systeem (1 = enable).
d. Een 1 in één van deze vakken geeft aan welke van de interrupt adressen afgestopt is (M).

SIM 00110000 - - - - - 1 4

Set interrupt mask
De SIM instructie gebruikt de accumulatorinhoud van het ogenblik om :
a. de interrupt adressen 7,5 6,5 en (of) 5,5 af te stoppen.
b. de flank gevoelige input van RST 7,5 te resetten (RST 6,5 en RST 5,5 zijn niveau gevoelig).
c. D_7 van de accumulator te gebruiken als één bit van de informatie die in serie wordt buiten gebracht.
De samenstelling van het codewoord is als volgt :



a. SOD
Serial output data
De bit in vak D_7 wordt gebruikt als serie uitgang op voorwaarde dat de bit in vak D_6 een 1 is.
b. SDE
Serial data enable
SDE geeft aan of D_7 al (1) dan niet (0) als serie uitgang in aanmerking komt.
c. Geen betekenis.
d. Een 1 in dit vak veroorzaakt een terugzetten van RST 7,5.
e. MSE
Mask set enable
Een 1 in vak D_3 maakt het mogelijk één van de interrupt adressen af te stoppen.
Een nul geeft aanleiding tot negeren van de inhoud van D_0 tot D_2.
f. Geeft aan welk niveau afgestopt werd (M).
De D_1 heeft voorrang op de SIM instructie.

4.2.8. Speciale instructies

HLT 01110110 - - - - - 1 7(5)

Halt
Deze opdracht stopt het programma tot een interrupt aanvraag geregistreerd wordt.

EXTERNAL INTERRUPT

HOW TO USE AN EXTERNAL INTERRUPT

According to the DAI manual, an external interrupt signal can be connected to the DCE-bus. But how to use this interrupt signal is not described.

The external interrupt must be a low-to-high transition on pin 6 of the DCE-bus. It provokes an external interrupt on the 5501 interrupt controller (pin 22). As result, a RST2 instruction is generated by the 5501.

This small routine enables the use of an user interrupt routine in combination with the 'stack overflow' routine, which is generated by the DAI via the same external interrupt on the 5501.

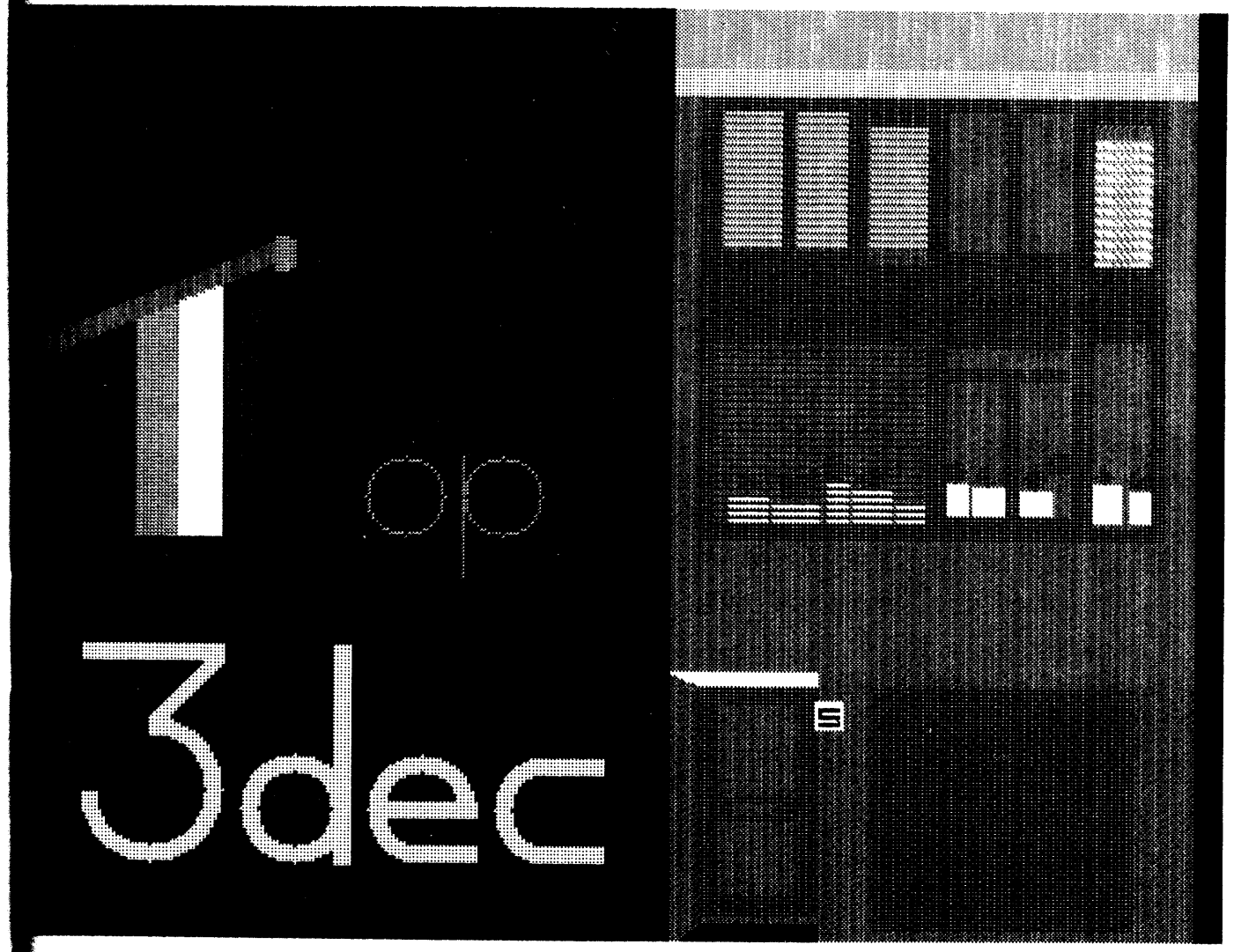
```
****        PUSH PSW
            PUSH B
            PUSH D
            LDA :ØØ5F     Get current interrupt mask
            PUSH PSW      and preserve it
            EI
            LXI D,:Ø8ØØ
            LXI H,:ØØØØ
            DAD SP        Get stackpointer in HL
            DAD D         Test if stackbottom reached
            JNC :D9E2     Then run 'stack overflow' error
              |
              |
              |           Your own interrupt routine
              |
              |
              |
            JMP :D9CD     General interrupt return
```

The interrupt vector address of RST2 - on address ØØ66/67 - must be changed to the startaddress of the routine above (****). This can be done under Utility via: V2.

JEROEN DEMO

```
1     REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2     REM Jeroen Overvoorde Helmbloem 5 3068 AC Rotterdam
3     REM Tel:(010)-210426 Nederland datum 2 dec 1983
4     REM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
5     REM ------------------------------------------------
6     REM 3 dec Helmbloem 5
7     REM ------------------------------------------------
10    MODE 5:X=184:FILL X,0 143+X,YMAX 12
11    MX=110:MY=121:R=11:K=10:GOSUB 3000:R=10:K=0:GOSUB 3000
12    MX=137:R=11:K=10:GOSUB 3000:K=0:R=10:GOSUB 3000:DRAW 126,99 126,132 1
0
20    FILL 36,110 47,190 9:FILL 48,110 59,190 15:FILL 60,110 71,190 3
30    FOR I=155 TO 190:DRAW 0,I 71,I+30 0:NEXT
40    FOR I=155 TO 160:DRAW 0,I 73,I+30 8:NEXT:IF T<>2 THEN T=T+1:GOTO 40
50    T=0:FILL 74,183 77,192 10:FILL 78,184 79,191 10
60    MX=40:MY=40:R=20:K=14:GOSUB 3000:R=15:K=0:GOSUB 3000:FILL MX-20,MY MX
,MY+20 0
65    FOR I=MY+15 TO MY+20:DRAW MX,I MX+20,I+20 14:NEXT:FILL MX-19,MY+36 MX
+20,MY+40 14
70    MX=80:MY=35:R=15:K=14:GOSUB 3000:R=10:K=0:GOSUB 3000:FILL 91,20 95,80
 14
80    MX=115:R=15:K=14:GOSUB 3000:R=10:K=0:GOSUB 3000
85    FILL MX,MY MX+20,MY-20 0:FILL MX,21 MX+15,25 14:FILL MX-14,MY-2 MX+15
,MY+2 14
90    MX=150:R=15:K=14:GOSUB 3000:R=10:K=0:GOSUB 3000:FILL MX,20 MX+15,50 0
95    FILL MX,21 MX+15,25 14:FILL MX,45 MX+15,49 14
100   FILL X,232 143+X,239 14:FILL X,110 7+X,231 8:FILL 136+X,110 143+X,231
 8
110   FOR Y=230 TO 116 STEP -6:FILL 0+X,Y 4+X,Y-1 7:FILL 6+X,Y 7+X,Y-1 7
120   FILL 136+X,Y 140+X,Y-1 7:FILL 142+X,Y 143+X,Y-1 7
130   FILL X,Y-3 1+X,Y-4 7:FILL 3+X,Y-3 7+X,Y-4 7
140   FILL 136+X,Y-3 137+X,Y-4 7:FILL 139+X,Y-3 143+X,Y-4 7:NEXT
150   FILL X,74 143+X,109 8
160   FOR V=109 TO 74 STEP -6:FOR H=0 TO 143 STEP 6:FILL H+X,V H+X+4,V-1 7:
NEXT H:NEXT V
170   FOR V=106 TO 74 STEP -6:FILL X,V X+1,V-1 7:FOR H=3 TO 140 STEP 6:FILL
 H+X,V H+X+4,V-1 7:NEXT H
180   FILL 142+X,V 143+X,V-1 7:NEXT
190   FILL 8+X,110 135+X,231 6
200   FOR I=1 TO 11:READ A,B,C,D:FILL A+X,B C+X,D 8:NEXT
210   FOR V=227 TO 224 STEP -2:DRAW 14+X,V 29+X,V 13:DRAW 34+X,V 47+X,V 13:
WAIT TIME 5:NEXT
220   FOR V=223 TO 190 STEP -2:DRAW 14+X,V 29+X,V 13:DRAW 34+X,V 47+X,V 13:
DRAW 54+X,V 69+X,V 13
230   WAIT TIME 5:NEXT:FILL 40+X,0 55+X,73 8
240   FOR V=73 TO 0 STEP -6:FOR H=45 TO 55 STEP 6:FILL 40+X,V 44+X,V-1 7:FI
LL H+X,V H+X+4,V-1 7:NEXT H:NEXT V
260   FOR V=70 TO 0 STEP -6:FILL 40+X,V 41+X,V-1 7:FILL 43+X,V 47+X,V-1 7:F
ILL 49+X,V 52+X,V-1 7
270   FILL 54+X,V 55+X,V-1 7:NEXT:FILL 128+X,0 143+X,73 8
275   FOR V=73 TO 0 STEP -6:FOR H=133 TO 143 STEP 6:FILL 128+X,V 132+X,V-1
7:FILL H+X,V H+X+4,V-1 7:NEXT H:NEXT V
280   FOR V=70 TO 0 STEP -6:FILL 128+X,V 129+X,V-1 7:FILL 131+X,V 135+X,V-1
 7:FILL 137+X,V 140+X,V-1 7
290   FILL 142+X,V 143+X,V-1 7:NEXT
295   FILL 56+X,0 127+X,73 8
300   FILL 56+X,3 127+X,69 6:FILL 89+X,50 94+X,51 8
310   K=5:FOR I=1 TO 7:READ A,B,C:FILL A+X,115 B+X,C 15:R=RND(3)+2:MX=(B-A)
/2+A+X:MY=C+R+1:GOSUB 3000:NEXT
315   FOR I=1 TO 4:READ A,B,C:FILL A+X,117 B+X,C 15:R=RND(3)+2:MX=(B-A)/2+A
+X:MY=C+R+1:GOSUB 3000:NEXT
320   FILL 8+X,5 40+X,70 5:FILL X,0 8+X,73 8:FOR I=0 TO 8:DRAW I+X,73 8+X,7
0 15:NEXT
325   FILL 8+X,70 40+X,73 15
```

```
330   FOR I=3 TO 10:DRAW I+X,0 8+X,4 3:NEXT:FILL 8+X,0 40+X,4 3
340   FILL 8+X,5 40+X,6 8:FILL 14+X,13 35+X,32 8
345   FILL 14+X,40 35+X,64 8:FILL 18+X,35 29+X,37 8
350   FILL 116+X,186 131+X,223 15
351   FOR I=131 TO 116 STEP -1:FOR U=186+0 TO 222:DRAW I+X,U I+X,U+1 9
355   U=U+2:IF U<223 THEN NEXT U
360   IF 0=0 THEN 0=1:NEXT I
370   IF 0=1 THEN 0=0:NEXT I
380   FOR I=164 TO 114 STEP -2:DRAW 12+X,I 69+X,I 6:WAIT TIME 3:NEXT
390   FILL 76+X,221 91+X,227 7:FILL 96+X,221 109+X,227 7:FILL 116+X,221 131
+X,223 7
400   FILL 40+X,58 47+X,65 15:FOR I=59 TO 63 STEP 2:DRAW 41+X,I 46+X,I 0:NE
XT
410   DOT 41+X,62 0:DOT 46+X,60 8
500   LOAD "JEROEN DEMO 3"
1000  DATA 14,190,29,227,34,190,47,227,54,186,69,223,76,190,91,227,96,190,1
09,227,116,186,131,223
1010  DATA 12,114,69,164,76,117,91,153,96,117,109,153,76,158,109,162,116,11
4,131,164
2000  DATA 16,26,122,28,40,120,43,48,125,50,60,123,62,69,120,116,123,125,12
6,131,123
2010  DATA 76,81,125,83,91,124,96,104,123,106,109,126
3000  D=(R*R):FOR Y=0 TO R:XX=SQR(D-Y*Y)
3010  DRAW MX-XX,MY-Y MX+XX,MY-Y K:DRAW MX-XX,MY+Y MX+XX,MY+Y K:NEXT Y
3020  RETURN
```



Beveren, 24 september

Geachte heer,

Bij het ontwerpen van het programma CRYPTOWRITER (zie bijlage) waarin enkele keren van integer variabelen naar string variabelen wordt overgegaan kwamen enkele foutjes aan het licht bij het gebruik van de funkties STR$(X) en VAL(X$).
Deze foutjes kwamen alleen voor als de getallen meer dan 6 cijfers bevatten.

+ De VAL(X$) funktie
  - werkt exact tot 5 cijfers
  - bij 6 cijfers werkt de funktie ook nog goed maar worden de getallen in wetenschappelijke notatie weergegeven
  - vanaf 7 cijfers treden er fouten op omdat dan in wetenschappelijke notatie wordt gewerkt en er afgerond wordt

Vb.	X$="12345"	VAL(X$)	geeft	12345
	X$="123456"	"	"	1.23456E5
	X$="1234567"	"	"	1.23457E6
	X$="12345678"	"	"	1.23457E7
	...			

Deze foutjes kunnen we omzeilen met onderstaande programmaregel die echter alleen werkt voor integer getallen bestaande uit 8 cijfers, maar met enige aanpassing is het ook bruikbaar voor floating point getallen. Het komt erop neer dat het getal in twee stukken verdeeld wordt die foutloos omgezet worden en dan worden die twee delen terug bij elkaar gevoegd.
Het kan vreemd lijken dat niet in één keer de waarde X berekend wordt maar dat eerst de tussenuitkomsten berekend worden en dan pas de optelling ervan. In de praktijk is namelijk gebleken dat je fouten krijgt als X in één keer berekend wordt.

```
10 X1=VAL(MID$(X$,0,4))*10000:X2=VAL(MID$(X$,4,4)):X=X1+X2
```

+ De STR$(X) funktie
  Hier treden dezelfde moeilijkheden op als bij de VAL(X$) funktie. Met onderstaand programaatje kunnen deze problemen omzeild worden. Het is eveneens voor integer waarden van 8 cijfers. (vooraf IMP INT)

```
1Ø ED=X/1ØØØØ:LD=X-ED*1ØØØØ
2Ø ED$=STR$(ED):LDs=STRs(LD)
3Ø ED$=MID$(ED$,1,4):LD$=MID$(LD$,1,LEN(LD$)-3)
4Ø IF LEN(LD$)<>4 THEN LD$="Ø"+LD$:GOTO 4Ø
5Ø X$=ED$+LD$
```

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

Tenslotte nog een verzoekje. Ik ben in het bezit van joysticks van een Atari spelcomputer. Is het mogelijk om een schakeling te publiceren waardoor ik de 9-polige connector ervan kan aansluiten op de DIN-bus van de DAI computer?

Met DAInamische groeten

Filip Verberckmoes
Meidoornlaan 16
2750 Beveren
Tel: 03/775.20.64

Filip Verberckmoes

Cedric Dufour has developed a splendid interface for ATARI-compatible joysticks (4 contacts - 8 directions) , more about this in one of our next issues ...

CRYPTOWRITER

Cryptowriter is een programma waarmee je een tekst kunt "vertalen" in geheimschrift m.b.v. een codeersleutel. Deze sleutel is een letter-,cijfer-, of tekencombinatie van maximaal 3 karakters.
Vb. DAI , 123 , ?/+ , JA ...
Het codebericht kan je op cassette bewaren of afschrijven van het scherm. Deze schrijfroutine kan je ook gebruiken, mits enige aanpassingen, om het codebericht te laten uitprinten.
Cryptowriter vervangt niet een letter door een bepaald getal zodat de code gebroken kan worden door de frequentie te tellen van bepaalde getallen. Cryptowriter maakt gebruik van de bit-operator IXOR. Het programma voert volgende bewerking uit om te coderen: CODE=BOODSCHAP IXOR SLEUTEL. Om te decoderen doe je eenvoudig BOODSCHAP=CODE IXOR SLEUTEL en het bericht verschijnt in alle duidelijkheid.
Eerst wordt de sleutel, vb. DAI omgezet in zijn ASCII waarde : (voor DAI geeft dat 686573). Daarna wordt uit INPT$( ), die de tekst onder de vorm van zijn ASCII waarden bevat, telkens 8 cijfers gehaald en wordt bovenstaande bewerking met de bit-operator uitgevoerd. Het resultaat komt in CBR$( ).
Vb. sleutel = DAI = 686573
boodschap = NEEN = 78696978
code = 79342591
Het valt meteen op dat de letters N en E telkens op een andere wijze gecodeerd worden.

Filip Verberckmoes
Meidoornlaan 16
2750 Beveren
Tel: 03/775.20.64

```
10      REM CRYPTOWRITER  DOOR FILIP VERBERCKMOES  9/1984
20      PRINT CHR$(12):CLEAR 10000:DIM INPT$(100),CBR$(255)
30      CURSOR 10,20:FOR A=1 TO 40:PRINT CHR$(25);:NEXT:CURSOR 20,18:PRINT
        "CRYPTOWRITER"
40      CURSOR 10,16:FOR A=1 TO 40:PRINT CHR$(25);:NEXT
50      CURSOR 15,12:PRINT "KEUZEMENU":CURSOR 15,10:PRINT "CODEREN.....
        1":CURSOR 15,9:PRINT "DECODEREN..... 2"
60      CURSOR 15,8:PRINT "UITLEG..... 3":CURSOR 15,6:PRINT "TIK UW
        KEUZE IN A.U.B.  >";
70      P=GETC:IF P=49 THEN PRINT "1":GOTO 110
80      IF P=50 THEN PRINT "2":GOTO 490
90      IF P=51 THEN PRINT "3":GOTO 1060
100     GOTO 70
110     GOSUB 1000:PRINT CHR$(12);"TIK NU JE BOODSCHAP IN"
120     P=GETC:IF P=0 GOTO 120
130     IF P=8 THEN CURSOR CURX-1,CURY:PRINT " ";:CURSOR CURX-1,CURY:LS=
        LEN(INPT$(N))-2:INPT$(N)=LEFT$(INPT$(N),LS):GOTO 120
140     IF P=9 THEN PRINT CHR$(12);"CODEBOODSCHAP":GOTO 170
150     IF P=13 THEN N=N+1:IF N>100 THEN PRINT :PRINT "MAXIMALE INPUTCAPACITEIT
        BEREIKT !":N=N-1:INPUT "RETURN";RET$:GOTO 170
160     PRINT CHR$(P);:P$=MID$(STR$(P),1,2):INPT$(N)=INPT$(N)+P$:GOTO 120
170     FOR I=0 TO N:REM Coderen
180      IF LEN(INPT$(I)) MOD 8<>0 THEN INPT$(I)=INPT$(I)+"32":GOTO 180
190      LS=LEN(INPT$(I))-8
200      FOR A=0 TO LS STEP 8
210       BS$=MID$(INPT$(I),A,8)
220       AB=VAL(MID$(BS$,0,4))*10000:BC=VAL(MID$(BS$,4,4)):BS=AB+BC
230       CODE=BS IXOR SLEUTEL:ED=CODE/10000
240       LD=CODE-ED*10000:ED$=STR$(ED):LD$=STR$(LD)
250       ED$=MID$(ED$,1,LEN(ED$)-3):LD$=MID$(LD$,1,LEN(LD$)-3)
260       IF LEN(ED$)<>4 THEN ED$="0"+ED$:GOTO 260
270       IF LEN(LD$)<>4 THEN LD$="0"+LD$:GOTO 270
280       CODE$=ED$+LD$
290       IF LEN(CBR$(M))=54 THEN PRINT CBR$(M):M=M+1
300     CBR$(M)=CBR$(M)+CODE$+" ":NEXT A:NEXT I:PRINT CBR$(M)
310     PRINT :PRINT "Wilt U het codebericht controleren (J/N) ?";
320     P=GETC:IF P=74 THEN PRINT CHR$(12);"CONTROLE CODEBERICHT":FLAG=1:GOTO
        600
330     IF P<>78 THEN 320
400     PRINT :PRINT "Wilt U het codebericht op cassette bewaren (J/N)";
410     P=GETC:IF P=74 THEN PRINT :INPUT "START TAPE,DRUK RETURN";RET$:SAVEA
        CBR$ "CODEBOODSCHAP":PRINT :GOTO 700
420     IF P<>78 GOTO 410
430     PRINT CHR$(12);"CODEBOODSCHAP":FOR A=0 TO M
440      PRINT CBR$(A):IF CURY=0 THEN INPUT "RETURN";RET$:PRINT CHR$(12);
450     NEXT A:GOTO 700

480     REM Decoderen  ** input codebericht
490     GOSUB 1000:PRINT CHR$(12);
500     PRINT "Codebericht manueel of via cassette invoeren (M/C)?";
510     P=GETC:IF P=67 THEN PRINT "C":PRINT "START TAPE":LOADA CBR$ :GOTO 600
520     IF P<>77 GOTO 510
530     PRINT "M":M=0
540     P=GETC:IF P=0 GOTO 540
550     IF P=8 THEN CURSOR CURX-1,CURY:PRINT " ";:CURSOR CURX-1,CURY:LS=
        LEN(CBR$(M))-1:CBR$(M)=LEFT$(CBR$(M),LS):GOTO 540
560     IF P=9 THEN CBR$(M)=CBR$(M)+" ":PRINT CHR$(12);"GEDEKODEERDE BOODSCHAP":
        GOTO 600
570     IF P=13 THEN CBR$(M)=CBR$(M)+" ":M=M+1:PRINT :GOTO 540
580     P$=CHR$(P):PRINT P$;:CBR$(M)=CBR$(M)+P$:GOTO 540
```

```
590     REM Decoderen codebericht
600     FOR B=0 TO 255:LS=LEN(CBR$(B))-9:FOR A=0 TO LS STEP 9
610       A$=MID$(CBR$(B),A,8)
620       AB=VAL(MID$(A$,0,4))*10000:BC=VAL(MID$(A$,4,4))
630       CBS=AB+BC:BS=CBS IXOR SLEUTEL
640       BD=BS/1000000:PRINT CHR$(BD);:LK=BD*1000000
650       BD=(BS-LK)/10000:PRINT CHR$(BD);:LK=LK+BD*10000
660       BD=FRAC(BS/10000.0)*100:PRINT CHR$(BD);:LK=LK+BD*100
670     PRINT CHR$(BS-LK);:NEXT A
680     IF CBR$(B+1)<>"" THEN NEXT B
690     IF FLAG=1 THEN FLAG=0:GOTO 400
700     PRINT :PRINT "Wilt U verder gebruik maken van CRYPTOWRITER (J/N)?";
710     P=GETC:IF P=74 GOTO 740
720     IF P=78 THEN END
730     GOTO 710
740     FOR A=0 TO N:INPT$(A)="":NEXT A
750     FOR A=0 TO 255:CBR$(A)="":IF CBR$(A+1)<>"" THEN NEXT A
760      GOTO 20

1000     REM Opstellem sleutel
1010     CURSOR 15,4:INPUT "SLEUTEL";S$:A=LEN(S$):PRINT :IF A>3 OR A=0 THEN
         1010
1020     SL$="":FOR D=0 TO A-1
1030      R$=MID$(S$,D,1):P=ASC(R$)
1040      ST$=MID$(STR$(P),1,2)
1050     SL$=SL$+ST$:NEXT D:SLEUTEL=VAL(SL$):RETURN
1060     PRINT CHR$(12);TAB(25);"UITLEG"
1070     PRINT "Met CRYPTOWRITER kunt U een tekst zodanig coderen dat alleen"
1080     PRINT "de bestemmeling, die de juiste sleutel kent, de oorspronke-"
1090     PRINT "lijke tekst kan lezen.  Bovendien wordt een letter op ver-"
1100     PRINT "schillende manieren gecodeerd zodat men de code niet meer"
1110     PRINT "kan breken door de frequenties te tellen van bepaalde":PRINT
         "tekens."
1120     PRINT "Om te starten tikken we eerst de codeersleutel in."
1130     PRINT "Dit is een letter-,cijfer- of leestekencombinatie bestaande"
1140     PRINT "uit maximaal 3 karakters.  Vb. DAI .  Daarna [RETURN]."
1150     PRINT "Dan kun je de tekst intikken die gecodeerd moet worden."
1160     PRINT "Als het intikken beeindigd is druk je [TAB] en het coderen"
1170     PRINT "wordt gestart.  Na het coderen heb je nog een controle-"
1180     PRINT "mogelijkheid die de volledige boodschap terug decodeert."
1190     PRINT "Je kan de codeboodschap op cassette bewaren of we van het"
1200     PRINT "scherm afschrijven.":PRINT "Bij het decoderen moet je eerst de
         sleutel in te tikken"
1210     PRINT "en daarna kan je het codebericht inladen van cassette of "
1220     PRINT "intikken (om te eindigen [TAB])."":PRINT
1230     PRINT "OPM. ALLE LETTERS MOETEN HOOFDLETTERS ZIJN ANDERS"
1240     PRINT "     LOOPT DE CODERING IN HET HONDERD !!!!!!!!!!"
1250     PRINT :PRINT "DRUK OP EEN TOETS >";
1260     IF GETC=0 GOTO 1260
1270     GOTO 20

1300     REM U kunt de codeboodschap ook laten uitprinten.
1310     REM Daarvoor moet U de regels 430 - 450 veranderen
1320     REM zodat de boodschap geprint wordt.  Het code-
1330     REM bericht zit in CBR$( ). Het aantal is M.
1340     REM *****  CRYPTOWRITER  *****  END  *****
```

NUMBERS UP

```
100     CLEAR 1000:DIM A(7,7),C$(3):COLORT 8 0 8 14:POKE #75,32
110     C$(0)=CHR$(136):C$(1)=CHR$(137):C$(2)=CHR$(94):C$(3)=CHR$(140)
120     MODE 0:PRINT CHR$(12):POKE #BC44,#D1:POKE #BC45,#4A:POKE #BC43,#4E
                                          :POKE #BC41,#55:POKE #BC3F,#4D
130     CURSOR 0,16:PRINT "BERS UP":POKE #BBBE,#D0
140     FOR A=9 TO 0 STEP -1:FOR S=10 TO 23:IF S=16 GOTO 160
150     IF S-A<>16 THEN CURSOR 40+A,S-A:PRINT A;
160     NEXT:FOR SZ=40+A TO 20 STEP -1
170     IF 23-A<>16 THEN CURSOR SZ+9-A,23-A:PRINTA;
180     NEXT:NEXT:COLORT 8 0 8 9:CURSOR 5,6:PRINT "MOEILIJKHEIDSGRAAD 1-5"
190     FOR S=#B6F8 TO S-46 STEP -2:POKE S,#FF:NEXT
200     H=GETC:IF H<49 GOTO 200:IF H>53 GOTO 200:B=H-46
210     FOR XX=1 TO B:FOR YY=1 TO B:A(XX,YY)=(XX-1)*B+YY:NEXT:NEXT
220     TX=21:TY=7:D=0:FOR H=3 TO B:TY=TY-1-D:D=1-D:TX=TX-3-D:NEXT
230     XB=B:YB=B:A(XB,YB)=0:PRINT CHR$(12);:GOSUB 510
240     GOSUB 650:POKE #B46A,#D1
250     CURSOR 5,1:PRINT "IN DEZE VOLGORDE MOETEN DE GETALLEN GEZET WORDEN"
260     WAIT TIME 150:CURSOR 5,1:PRINT SPC(7);"IK GOOI ZE NU NETJES DOOR
                                                    ELKAAR";SPC(9)
270     CURSOR 59,23:GOSUB 460:GOSUB 800
280     CURSOR 0,1:PRINT SPC(17);"WILT U WAT VOORBEELDEN ?";SPC(18)
290     H=GETC:IF H=0 GOTO 290:IF H<>74 GOTO 320
300     CURSOR 0,1:PRINT "IK GEEF U ENKELE VOORBEELDEN VOOR HET GEBRUIK VAN
                                                        DE PIJLEN"
310     FOR V=0 TO 10:GOSUB 690:GOSUB 780:NEXT
320     CURSOR 0,1:PRINT SPC(60):CURSOR 14,1:PRINT " UW BEURT NU:
                                                  VEEL SPEELGENOT"
330     Z=GETC:Z=GETC
340     Z=GETC:IF Z<16 GOTO 340:IF Z>19 GOTO 340:Q=Z-14-SGN(Z/18)*4
350     FOR Z=0 TO 4:SOUND 0 0 15 3 FREQ(400+RND(250))
360     WAIT TIME 2:SOUND OFF :WAIT TIME 2:NEXT
370     GOSUB 740:IF L=(-1) GOTO 330:GOSUB 780
380     FOR XX=1 TO B:FOR YY=1 TO B:IF XX=B THEN IF YY=B GOTO 400
390     IF A(XX,YY)<>(XX-1)*B+YY GOTO 330:NEXT:NEXT:GOTO 330
400     PRINT CHR$(12);:COLORT 8 1 0 0:Q=0
410     FOR Q!=0.0 TO 10.0 STEP 0.25:A=25.0+25.0*SIN(Q!):
                                        PRINT TAB(A);"FANTASTISCH":NEXT
420     POKE #BA2C,#D6:POKE #B9A6,#D1
430     CURSOR 12,12:PRINT "WIL JE NOGMAALS SPELEN (J/N)":GOSUB 800
440     H=GETC:IF H=0 GOTO 440:IF H=74 GOTO 100
450     PRINT CHR$(12):CURSOR 15,12:PRINT " JAMMER TOT DE VOLGENDE KEER":
                                                  POKE #75,29:END
460     FOR Z=0 TO B^3+70:NOISE 1 10:O=M:P=N
470     Q=RND(4.0):IF (Q+QO) MOD 4=1 GOTO 470:GOSUB 7*0
480     IF L=(-1) GOTO 470
490     IF XB=0 THEN IF YB=P THEN A(XB,YB)=A(M,N):A(M,N)=0:XB=M:YB=N:M=O:N=P:
                                                        GOTO 470
500     NOISE OFF :GOSUB 780:QO=Q:NEXT:RETURN
510     COLORT 8 8 8 9:FOR Y=0 TO B:S$="":FOR X=0 TO B*7:S$=S$+CHR$(11):NE''5=T
520     CURSOR TX+1,23-TY-3*Y:PRINT S$:NEXT
530     FOR X=0 TO B*7 STEP 7:FOR Y=0 TO (B*3)-1
540     CURSOR TX+1+X,22-Y-TY:PRINT CHR$(10):NEXT:NEXT
550     FOR X=1 TO B*7+1 STEP 7:CURSOR X+TX,23-TY:PRINT CHR$(25):NEXT
560     CURSOR 1+TX,23-TY:PRINT CHR$(21):CURSOR 1+B*7+TX,23-TY:PRINT CHR$(20)
570     FOR X=1 TO B*7+1 STEP 7:FOR Y=20 TO 26-B*3 STEP -3:
                                  CURSOR X+TX,Y-TY:PRINT CHR$(16):NEXT:NEXT
```

```
580       FOR Y=20 TO 26-B*3 STEP -3:CURSOR 1+TX,Y-TY:PRINT CHR$(23):NEXT
590       FOR Y=20 TO 26-B*3 STEP -3:CURSOR 1+B*7+TX,Y-TY:PRINT CHR$(22):NEXT
600       CURSOR 1+TX,23-TY-B*3:PRINT CHR$(19):CURSOR TX+1+B*7,23-TY-B*3:
                                                        PRINT CHR$(18)
610       FOR X=8 TO B*7-6 STEP 7:CURSOR X+TX,23-B*3-TY:PRINT CHR$(24):NEXT
620       FOR Y=0 TO B*3 STEP 3:FOR X=#BFE2-(TY+Y)*#86-TX*2 TO X-B*14 STEP -2:
                                                        POKE X,#FF:NEXT:NEXT
630       FOR X=#BFE2-TX*2 TO X-B*14 STEP -14:
                    FOR Y=X-TY*#86 TO Y-B*3*#86 STEP -#86:POKE Y,#FF:NEXT:NEXT
640       COLORT 8 14 8 9:RETURN
650       FOR YY=1 TO B:FOR XX=1 TO B
660       CURSOR TX+XX*7-4,24-TY-3*YY:PRINT A(XX,YY)
670       IF A(XX,YY)=0 THEN CURSOR TX-4+XX*7,24-TY-3*YY:PRINT "  "
680       NEXT:NEXT:RETURN
690       Q=RND(4.0):GOSUB 740:IF L=(-1) GOTO 690
700       X=M*7-3:Y=24-3*N
710       FOR Z=0 TO 5:CURSOR X+TX,Y-TY:SOUND 0 0 15 3 FREQ(400+RND(200))
720       PRINT C$(Q):WAIT TIME 3:SOUND OFF :WAIT TIME 2
730       CURSOR X+TX,Y-TY:PRINT " ":WAIT TIME 5:NEXT:RETURN
740       M=XB:N=YB:IF Q<2 THEN XB=XB+1-SGN(Q)*2:GOTO 760
750       YB=YB+1-SGN(Q-2)*2
760       IF XB<1 OR YB<1 OR XB>B OR YB>B THEN XB=M:YB=N:L=-1:NOISE OFF :RETURN
770       A(M,N)=A(XB,YB):A(XB,YB)=0:L=0:RETURN
780       CURSOR TX+XB*7-4,24-TY-3*YB:PRINT "   "
790       CURSOR TX+M*7-4,24-TY-3*N:PRINT A(M,N):RETURN
800       H=GETC:H=GETC:H=GETC:RETURN
```

**** fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp ****

T Heeft U ook wel eens de behoefte gehad om een stuk van de tekst in T
buffer 1 te verwijderen en te vervangen door de tekst in buffer 2 ?
I En plaatste U toen ook markers 06 en 07 om de te verwijderen tekst? I
En een marker 01 om de plaats niet te vergeten, waar de invoeging
P moest gaan plaatsvinden ? En kreeg U toen ook 'MARKERS NOT CORRECT' P
na een vergeefse poging tot 'Kill text'? Nu dit is zo gemaakt om de
vergissingen tussen M en K zoveel mogelijk te voorkomen. Vindt U de
F beveiliging overbodig en misschien zelfs lastig ; U kunt er wat aan F
doen. Verander de 06 op adres #966 in een 07 door bv POKE #966,7 en
W uw versie van FWP zal ook met marker 01 een 'Kill' uitvoeren. Maar, W
jammer genoeg moet U nu altijd marker 01 zetten. Vinden we hier een
P oplossing voor zal die direct gepubliceerd worden. P

**** fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp fwp ****

Very efficient 8080 program multiplies and divides

by Jerry L. Goodrich
Pennsylvania State University, University Park, Pa.

Making an appearance for the third time in this section of *Electronics*, an 8080 program is being presented that can compute 32-by-16-bit division and 16-by-16-bit multiplication. However, this subroutine betters its immediate predecessor's divide and multiply execution times [*Electronics*, March 27, 1980, p. 156] by 75% and over 60%, respectively. Also, this program's 8-by-16-bit multiplication subroutine surpasses all others with an execu-

8080 PROGRAM FOR 32-BY-16-BIT DIVISION

```
LABEL     SOURCE   CODE          COMMENT

DIV:      MOV      A, L          ; CHECK FOR OVERFLOW
          SUB      C
          MOV      A, H
          SBB      B
          RNC                    ; RETURN ON OVERFLOW
          MOV      A, B          ; 2'S COMPLEMENT BC
          CMA
          MOV      B, A
          MOV      A, C
          CMA
          MOV      C, A
          INX      B
          CALL     LOOP          ; DIVIDE INTO HIGHEST-ORDER 3 BYTES OF DIVIDEND
 LOOP DIVIDES 3-BYTE DIVIDEND BY 2-BYTE DIVISOR
LOOP:     MOV      A, D          ; MOVE THIRD BYTE TO BE DIVIDED INTO A
          MOV      D, E          ; SAVE LOWEST-ORDER BYTE DIVIDEND OR HIGHEST-ORDER BYTE QUOTIENT
          MVI      E, 8          ; LOAD LOOP1 COUNTER
LOOP1:    DAD      H             ; SHIFT DIVIDEND LEFT
          JC       OVER          ; JUMP IF DIVIDEND OVERFLOWED HL
          ADD      A
          JNC      SUB
          INX      H             ; CONVEY CARRY IF THERE
SUB:      PUSH     H             ; SAVE HIGHEST-ORDER 2 BYTES OF DIVIDEND
          DAD      B             ; SUBTRACT DIVISOR
          JC       OK            ; JUMP IF NO BORROW
          POP      H             ; UNSUBTRACT IF BORROW
          DCR      E             ; UPDATE LOOP1 COUNTER
          JNZ      LOOP1         ; LOOP UNTIL DONE
          MOV      E, A          ; PUT BYTE OF QUOTIENT IN E
          STC
          RET
OK:       INX      SP            ; CLEAN UP STACK
          INX      SP
          INR      A             ; PUT A 1 IN QUOTIENT
          DCR      E             ; UPDATE LOOP1 COUNTER
          JNZ      LOOP1         ; LOOP UNTIL DONE
          MOV      E, A          ; PUT BYTE OF QUOTIENT IN E
          STC
          RET
OVER:     ADC      A             ; FINISH DIVIDEND SHIFT, PUT 1 IN QUOTIENT
          JNC      OVERSUB
          INX      H             ; CONVEY CARRY IF THERE
OVERSUB   DAD      B             ; SUBTRACT DIVISOR
          DCR      E             ; UPDATE LOOP1 COUNTER
          JNZ      LOOP1         ; LOOP UNTIL DONE
          MOV      E, A          ; PUT BYTE OF QUOTIENT IN E
          STC
          RET
```

8080 PROGRAM FOR 16-BY-16-BIT MULTIPLICATION

```
LABEL     SOURCE    CODE      COMMENT
MULT:     MOV       A, E      ; LOAD LOWEST-ORDER BYTE OF MULTIPLIER
          PUSH      D         ; SAVE HIGHEST-ORDER BYTE MULTIPLIER
          CALL      BMULT     ; DO 1-BYTE MULTIPLY
          XTHL                ; SAVE LOWEST-ORDER BYTES PRODUCT, GET MULTIPLIER
          PUSH      PSW       ; STORE HIGHEST-ORDER BYTE OF FIRST PRODUCT
          MOV       A, H      ; LOAD HIGHEST-ORDER BYTE OF MULTIPLIER
          CALL      BMULT     ; DO SECOND 1-BYTE MULTIPLY
          MOV       D, A      ; POSITION HIGHEST-ORDER BYTE OF PRODUCT
          POP       PSW       ; GET HIGHEST-ORDER BYTE OF FIRST PRODUCT
          ADD       H         ; UPDATE THIRD BYTE OF PRODUCT
          MOV       E, A      ; AND PUT IT IN E
          JNC       NC1       ; DON'T INCREMENT D IF NO CARRY
          INR       D         ; INCREMENT D IF CARRY
NC1:      MOV       H, L      ; RELOCATE LOWEST-ORDER BYTES OF SECOND PRODUCT
          MVI       L, 0
          POP       B         ; GET LOWEST-ORDER 2 BYTES OF FIRST PRODUCT
          DAD       B         ; GET FINAL PRODUCT LOWEST-ORDER 2 BYTES
          RNC                 ; DONE IF NO CARRY
          INX       D         ; OTHERWISE UPDATE HIGHEST-ORDER 2 BYTES
          RET
; BMULT PERFORMS A 1-BYTE BY 2-BYTE MULTIPLY
BMULT:    LXI       H, 0      ; ZERO PARTIAL PRODUCT
          LXI       D, 7      ; D = 0, E = BIT COUNTER
          ADD       A         ; GET FIRST MULTIPLIER BIT
LOOP1:    JNC       ZERO      ; ZERO-SKIP
          DAD       B         ; ONE-ADD MULTIPLICAND
          ADC       D         ; ADD CARRY TO THIRD BYTE OF PRODUCT
ZERO:     DAD       H         ; SHIFT PRODUCT LEFT
          ADC       A
          DCR       E         ; DECREMENT BIT COUNTER
          JNZ       LOOP1     ; LOOP UNTIL DONE
          RNC                 ; DONE IF NO CARRY
          DAD       B         ; OTHERWISE DO LAST ADD
          ADC       D
          RET                 ; AND RETURN
```

tion time less than half that of its larger counterpart. The program works with 3 bytes of the dividend or product at a time to improve the execution time, which is the most appropriate measure of program efficiency.

The divider program works much like ordinary long division of a four-digit decimal number by a two-digit number. The divide routine of the program stores the dividend in registers HL-DE and keeps the most signficiant digits in HL. The divisor is for the operation placed in register pair BC. The quotient from the division builds up in DE with the remainder placed in HL. For a quotient that is longer than 16 bits, the carry flag is cleared in order to indicate an overflow. The worst-case execution time for the divide routine is 1,745 clock cycles.

In the multiplication routine, the multiplicand is stored in BC and the multiplier in DE. The result appears in registers DE-HL with the most significant digits being placed in DE. The execution time is 1,023 clock cycles.

For the 8-by-16-bit multiply routine, the 8-bit number is placed in register A and the 16-bit number in BC. The result then appears in A-HL with the high-order byte being placed in A. The worst-case execution time for this subroutine is 424 clock cycles. □

Engineer's notebook is a regular feature in *Electronics*. We invite readers to submit original design shortcuts, calculation aids, measurement and test techniques, and other ideas for saving engineering time or cost. We'll pay $75 for each item published.

```
10    REM **   CALCULUS  V.2 *****
20    REM ***   9 sous-programmes  de calcul *****
30    REM ****  AUTEUR:F.LEMOINE   NOVEMBRE 1982  *****
40    MODE 0:PRINT CHR$(12):COLORT 12 10 3 1:REM fond bleu clair
50    POKE #74,1:POKE #75,32:REM Efface le curseur
60    POKE #BCCA,#C3:REM 17' ligne -couleur rouge
70    POKE #BB39,#5A:REM moyenne definition
80    POKE #BB38,#DA:CURSOR 2,14:PRINT "CALCULUS":REM 14'ligne-couleur oran
ge
90    POKE #B921,#4A:POKE #B920,#D1:CURSOR 3,10:PRINT "V.2":REM basse defin
ition-10'ligne-couleur bleu fonce
100   POKE #B89A,#CC:REM 9'ligne-couleur bleu clair
110   CURSOR 45,8:PRINT "space"
120   CALLM #D6DA:REM attend le space
130   PRINT CHR$(12):MODE 0:COLORT 12 6 15 5:POKE #74,1:POKE #75,6:REM curs
eur:triangle
140   PRINT TAB(10);"C A L C U L U S  V.2":PRINT
150   PRINT TAB(4):PRINT "P.G.C.D. de deux nombres                    1"
160   PRINT TAB(4):PRINT "triangle de Pascal                          2"
170   PRINT TAB(4):PRINT "somme des n premiers nombres                3"
180   PRINT TAB(4):PRINT "nombres de Fibonacci                        4"
190   PRINT TAB(4):PRINT "factorielle                                 5"
200   PRINT TAB(4):PRINT "carre magique:nombre impair de cote         6"
210   PRINT TAB(4):PRINT "calcul des angles d'un triangle             7"
220   PRINT TAB(4):PRINT "moyenne.,variance et ecart-type             8"
230   PRINT TAB(4):PRINT "equation du 1 er et du 2e degre             9"
240   PRINT :INPUT "QUEL EST VOTRE CHOIX";H
250   IF H=0.0 OR H>9.0 THEN GOTO 240
260   COLORT 14 0 5 4:POKE #74,0:POKE #75,4
270   ON H GOSUB 340,440,590,620,700,800,1030,1160,1380
280   PRINT :PRINT " pour continuer/retour au menu/terminer[C/R/F]"
290   G=GETC:IF G=0.0 GOTO 290
300   IF G=ASC("C") THEN 270
310   IF G=ASC("R") THEN 130
320   IF G=ASC("F") THEN PRINT CHR$(12):END
330   IF G<>ASC("C") AND G<>ASC("F") AND G<>ASC("R") THEN 280
340   H=1:PRINT CHR$(12):PRINT TAB(15);"P.G.C.D de deux nombres"
350   PRINT TAB(15);"par l'algorithme d'Euclide":PRINT
360   INPUT "nombre 1";A!:PRINT :INPUT "nombre 2";B!
370   IF A!=0.0 OR B!=0.0 THEN PRINT "  ***pas de ZERO s.v.p***":GOTO 360
380   IF A!>B! THEN C!=A!:A!=B!:B!=C!
390   Q!=INT(A!/B!)
400   R!=A!-B!*Q!
410   IF R!<>0.0 THEN A!=B!:B!=R!:GOTO 390
420   V=B!:PRINT :PRINT "P.G.C.D = ";V
430   RETURN
440   PRINT CHR$(12):CLEAR 1500
450   PRINT "triangle de Pascal"
460   PRINT :INPUT "nombre de lignes ou de colonnes jusque 16";N:PRINT
470   IF N=1.0 THEN PRINT " **CE N'EST PAS SERIEUX**":GOTO 460
480   IF N>16.0 THEN PRINT :PRINT "pas au-dela de 16 s.v.p!":GOTO 460
490   DIM A(N,N)
500   PRINT CHR$(12):A(1.0,1.0)=1.0:CURSOR 0,22:PRINT A(1.0,1.0)
510   FOR I!=2.0 TO N
520   FOR J!=1.0 TO I!
530   A(I!,J!)=A(I!-1.0,J!)+A(I!-1.0,J!-1.0):NEXT:NEXT
540   Z=1:FOR Y!=21.0 TO 2.0 STEP -1.3:Z=Z+1:IF Z>N THEN 580
550   FOR X!=0.0 TO 59.0 STEP 5.0:CURSOR X!,Y!:K=K+1
560   IF K>Z THEN 570:PRINT A(Z,K):NEXT X!
570   K=0:NEXT Y!
580   H=2:RETURN
590   H=3:PRINT CHR$(12):PRINT TAB(7);"somme des n premiers nombres"
600   PRINT :INPUT "jusque combien";N:S=0:FOR I!=1.0 TO N:S=S+I!
```

```
610   NEXT:PRINT :PRINT "SOMME= ";S:PRINT :RETURN
620   H=4:PRINT CHR$(12):PRINT TAB(15);"nombres de Fibonnaci":PRINT
630   PRINT "chaque nombre est egal a la somme "
640   PRINT "des   deux    precedents  ":PRINT
650   INPUT "combien de nombres jusqu'a 91: ";N!:PRINT
660   IF N!>91.0 THEN PRINT "Pas au-dela de 91 s.v.p":GOTO 650
670   F0!=0.0:F2!=0.0:F1!=1.0:FOR I!=1.0 TO N!:F2!=F1!+F0!:PRINT F2!,:F0!=F
1!:F1!=F2!
680   NEXT I!:PRINT :PRINT
690   RETURN
700   H=5:PRINT CHR$(12):INPUT "factorielle d'un nombre ";A
710   IF A>=20 GOTO 740
720   F!=1.0
730   FOR I!=1.0 TO A:F!=F!*I!:NEXT I!:PRINT :PRINT A;" ! =";F!:RETURN
740   B!=(LOGT(A)-LOGT(EXP(1.0)))*A+LOGT(2.0*PI*A)/2.0
750   C!=INT(B!):D!=10.0^(B!-C!)
760   E!=(1.0/12.0+(1.0/288.0-1.0/373.0/A)/A)/A
770   F!=D!+D!*E!
780   PRINT :PRINT TAB(15):PRINT "factorielle par approximation :4 decimale
s ":PRINT TAB(5):PRINT A;"! =";F!;"  E  ";C!
790   RETURN
800   PRINT CHR$(12):PRINT "carre magique pour un nombre impair de cotes ju
sque 15"
810   CLEAR 1200
820   PRINT :INPUT "nombre de cotes";N!
830   IF N!=1.0 OR N!/2.0=INT(N!/2.0) THEN PRINT SPC(3);"nombre  non accept
e-recommencez s.v.p":GOTO 810
840   IF N!>15.0 THEN PRINT SPC(5);"pas au-dela de 15 je vous prie":GOTO 82
0
850   IF N!>=7.0 THEN PRINT "   ***patientez un instant je vous prie***"
860   DIM A(N!,N!)
870   M!=INT(N!/2.0)+1.0
880   FOR I!=1.0 TO N!:FOR J!=1.0 TO N!:A(I!,J!)=0.0:NEXT J!:NEXT I!
890   K!=1.0:I!=M!+1.0:J!=M!:A(I!,J!)=1.0
900   K!=K!+1.0
910   IF K!>N!*N! THEN 1000
920   IF I!+1.0>N! THEN I!=0.0
930   IF I!+1.0<=0.0 THEN I!=N!-1.0
940   IF J!+1.0>N! THEN J!=0.0
950   IF J!+1.0<=0.0 THEN J!=N!-1.0
960   IF A(I!+1.0,J!+1.0)=0.0 THEN 980
970   I!=I!+1.0:J!=J!-1.0:GOTO 920
980   I!=I!+1.0:J!=J!+1.0
990   A(I!,J!)=K!:GOTO 900
1000  PRINT :FOR I!=1.0 TO N!:PRINT :FOR J!=1.0 TO N!:PRINT A(I!,J!);:NEXT
J!:NEXT I!
1010  FOR J!=1.0 TO N!:S=S+A(1.0,J!):NEXT:PRINT :PRINT SPC(20);"SOMME =";S
1020  H=6:RETURN
1030  H=7:PRINT CHR$(12):PRINT "calcul des angles d'un triangle:soit a,b,c
les trois cotes"
1040  PRINT :INPUT "premier cote";A!
1050  PRINT :INPUT "second cote";B!
1060  PRINT :INPUT "troisieme cote";C!
1070  IF A!<=0.0 OR B!<=0.0 OR C!<=0.0 THEN PRINT :PRINT "impossible- veuil
lez recommencer":GOTO 1040
1080  CA!=(B!^2.0+C!^2.0-A!^2.0)/(2.0*B!*C!)
1090  IF CA!>1.0 OR CA!<(-1.0) THEN PRINT :PRINT "ce triangle n'existe pas"
:RETURN
1100  CB!=(A!^2.0+C!^2.0-B!^2.0)/(2.0*A!*C!)
1110  AA!=ACOS(CA!)*180.0/PI
1120  BB!=ACOS(CB!)*180.0/PI:CC!=180.0-AA!-BB!
1130  PRINT :PRINT "a= ";A!;SPC(3);"b= ";B!;SPC(3);"c= ";C!
1140  PRINT "angle A= ";AA!:PRINT :PRINT "angle B= ";BB!:PRINT :PRINT "angl
e C= ";CC!
1150  RETURN
1160  H=8:PRINT CHR$(12):PRINT "calcul de moyenne,variance et ecart-type"
```