Englisch/Gerner/Scheuse/Thrun

Tips en trucs voor de CPC computers

CPC Bibliotheek 5

DATA BECKER
NEDERLANDS

Tips en trucs voor de CPC computers

Englisch/Germer
Scheuse/Thrun

Tips en trucs voor de CPC computers

CPC Bibliotheek 5

DATA BECKER
NEDERLANDS*

Oorspronkelijke titel: CPC 464 Tips & Tricks. Eine
Fundgrube für den CPC 464 Anwender

Voor de Nederlandse vertaling:

Omslagontwerp: Martin van Keulen
Vertaling: ProCread/Informatica, Groningen
ISBN 90 229 3344 x

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door;
A.W. Bruna & Zoons Uitgeversmij b.v.,
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.,
Antwerpsesteenweg 29a, 2630 Aartselaar.

INHOUDSOPGAVE

1 GRAFIEK

1.1 Inleiding

Grafiek is een van de interessantste onderdelen van de computer. De CPC 464 kent de volgende grafische mogelijkheden:

resolutie van 640*200 punten (=128.000 afzonderlijke punten)
20, 40 of 80 tekens (kolommen) en 25 regels
2, 4 of 16 kleuren tegelijk mogelijk
27 verschillende kleuren
produktieve grafiekinstructies
windowtechniek
de mogelijkheid om zelf tekens te definiëren
grafische tekens

De ontwikkeling op het gebied van grafiek heeft de computer tot een universeel instrument gemaakt. Mensen zijn in eerste instantie visueel georiënteerd. Ze nemen veel gemakkelijker informatie op als die wordt aangeboden als een mooi plaatje. Grafische weergave is vooral van belang als computerberekeningen resulteren in grote hoeveelheden getallen. De hulpmiddelen en mogelijkheden die de CPC op dit gebied heeft, komen in dit hoofdstuk uitvoerig aan de orde. We geven praktische voorbeelden, maar het is de bedoeling dat u met eigen experimenten aan de slag gaat.

1.2 Resolutie van grafiek

Resolutie is net zoiets als een potlood: hoe scherper het potlood is geslepen, des te preciezer kunt u tekenen. Met een hogere resolutie zijn fijnere grafische afbeeldingen mogelijk. Er staat een groter aantal punten tot uw beschikking, die afzonderlijk kunnen worden aangesproken. Bij de CPC heeft u een raster van 640 punten in horizontale en 200 punten in verticale richting. Met die 128.000 punten kunt u staaf-, schijf- of lijngrafieken maken die zeer nauwkeurig zijn en uitstekend ogen. Natuurlijk is de grafische modus er ook om te tekenen, spelletjes te ontwerpen enzovoort. De programma's in de volgende paragrafen belichten het grafische aspect van uw nieuwe computer. Hopelijk een stimulans

voor eigen ideeën.

1.2.1 Afbeeldingen die het aankijken waard zijn

In deze paragraaf wat kleine routines om grafische voorstellingen te maken. Het eerste programma tekent een cirkel. U voert de oorsprong, de straal en de stapgrootte in. Vervolgens beslist u of de cirkel alleen maar getekend wordt of dat die, of de achtergrond, ook nog wordt ingekleurd. Met de stapgrootte bepaalt u hoe dicht dat gebeurt. Kies de stapgrootte daarom altijd zo klein mogelijk.

```
10 MODE 2
20 INPUT "oorsprong (X) ";X
30 INPUT"oorsprong (Y) ";Y
40 INPUT"radius ";R
50 INPUT"stapgrootte ";S
60 INPUT "'a'chtergrond of 'c'irkel inkleuren";A$
70 CLS
80 Z=(1 AND A$="A")+(2 AND A$="C")
90 ORIGIN X,Y
100 FOR N=1 TO 360 STEP S
110 EX=EX+1
120 XP=R*COS(N)
130 YP=R*SIN(N)
140 PLOT XP,YP,1
150 IF YP=ABS(YP)THEN EY=400-Y ELSE EY=Y-400
160 IF XP=ABS(XP) THEN EX=640-X ELSE EX=X-640
170 IF Z=1 THEN DRAW XP,EY
180 IF Z=1 THEN PLOT XP,YP : DRAW XP,YP
190 IF Z=2 THEN MOVE 0,0 : DRAW XP,YP
200 NEXT N
210 IF Z<> 1 GOTO 280
220 FOR N=R TO 640-X
230 MOVE N,0 : DRAW N,400-Y
240 MOVE -N,0 : DRAW -N,400-Y
250 MOVE N,0 : DRAW N,Y-400
260 MOVE -N,0 : DRAW -N,Y-400
270 NEXT N
280 A$=INKEY$:IF A$="" THEN 280
290 RUN
```

Bijvoorbeeld:

```
oorsprong (x) = 320
oorsprong (y) = 200
straal        =  30
stapgrootte   =   0.5
```

Het tweede plaatje suggereert drie dimensies waardoor het afgebeelde object beter te herkennen is. Bij een pyramide of een gang is dat pertinent noodzakelijk. Een kleine verandering in het programma levert weer een heel andere voorstelling op.

```
10 MODE 2
20 ORIGIN 320,200
30 FOR T=0 TO 180 STEP 10
40 PLOT A,T:DRAW T,-A
50 PLOT-A,T:DRAW-T,-A
60 PLOT 180,180:DRAW 0,0
70 PLOT-180,180:DRAW 0,0
80 PLOT A,T:DRAW -A,T
90 PLOT 180,-180:DRAW 0,0
100 PLOT -180,-180:DRAW 0,0
110 PLOT -T,-A:DRAW T,-A
120 A=A+10
130 NEXT T
```

Verandert u bijvoorbeeld regel 30 in:

```
30 FOR T=180 TO 0 STEP -10,
```

dan krijgt u een heel ander plaatje. Door wat te experimenteren krijgt u verrassende resultaten. Het derde plaatje heet caleidoscoop. Als u het programma heeft gestart, ziet u wel waarom. Deze routine bewijst dat de factor toeval bijzonder aardige resultaten kan opleveren.

```
10 MODE 2
20 DEF FN F(X) = INT (RND(1)*X)+1
30 FOR X=1 TO 400 STEP FN F(25)+5
40 PLOT X,0:DRAW 400,X
50 PLOT 0,X:DRAW X,400
60 PLOT X,0:DRAW 0,400-X
70 PLOT 400,X:DRAW 400-X,400
80 PLOT X,0:DRAW 400-X,400
90 PLOT 400,X:DRAW 0,400-X
100 NEXT X
110 LOCATE 55,1
120 PRINT"nog een keer ?"
```

```
130 A$=INKEY$:IF A$="" THEN 130
140 IF A$="J" OR A$="j" THEN RUN
150 END
```

De vierde grafische afbeelding is een sinuswaaier. Na invoer van de stapgrootte tekent het programma een sinuswaaier met lijnen vanuit het punt (0,200) naar het actuele punt van de sinus. De figuur geeft weer een driedimensionale indruk doordat de lijnen een beetje schuin over elkaar liggen.

```
10 REM sinuswaaier
20 DEG
30 CLG
40 INPUT "stapgrootte ";S
50 DEF FN F(X)=SIN(X)
60 FOR N=1 TO 640 STEP S
70 X=X+(S*1.125)
80 MOVE 0,200
90 DRAW N,200+FN F(X)*200
100 NEXT N
110 A$=INKEY$: IF A$="" THEN 110
120 RUN
```

Probeert u dit programma ook eens met COS(X), -COS(X) of met -SIN(X). De hoge resolutie van de CPC is erg geschikt voor computer-art. Met een programma tekeningen maken, opslaan, laden enzovoort is geen probleem. Voor wiskunde-freaks is de kwaliteit van de CPC-grafiek natuurlijk het einde. Maar ook bij boekhoudkundige programa's bewijst de hoge resolutie goede diensten.

1.2.2 Lissajous-figuren

Jules Antoin Lissajous, een Frans fysicus die van 1822 tot 1880 leefde, maakte een studie van deeltjes in periodieke bewegingen waarbij hij ontdekte dat de deeltjes verschillende curves beschrijven. We hebben een programma gemaakt dat het verloop van deze curves laat zien. Na de start van het programma voert u de volgende variabelen in:

```
Y-frequentie: tussen 0 en 20
X-frequentie: tussen 0 en 20
stapgetal   : aantal punten
```

```
10 REM Lissajous-figuren
20 DEG
30 MODE 2
40 INPUT "Y-frequentie ";Y
50 INPUT "X-frequentie ";X
60 INPUT "aantal stappen ";S
70 CLS
80 FOR A=0 TO 2*PI STEP 2*PI/S
90 PLOT 150*SIN(A*X)+150,100*COS(A*Y)+100,1
100 NEXT A
110 END
```

Uitleg bij de listing:

10	titel
20	hoek in graden (DEG), niet in radialen (RAD)
30	beeldschermmodus
40-50	invoer van y- en x-frequentie
60	invoer van stapgetal
70	beelscherm wissen
80	begin van lus
90	punten zetten
100	einde van lus
110	programma afgelopen

1.3 Grafische editor

Om het programma te bedienen gebruikt u een joystick. Met de joystick kunt u een puntje, de grafische cursor, over het beeldscherm bewegen. Met de spatiebalk kiest u een modus: tekenen of bewegen. In de bewegingsmodus beweegt de minicursor over het beeldscherm zonder sporen achter te laten. Eigenlijk trekt u een lijn die de kleur heeft van de achtergrond. Op die manier kunnen punten of lijnen worden gewist. De C-toets wist het hele beeldscherm. Om een tekening op te slaan, moet u E indrukken. Voer de naam van de tekening in en druk op REC en PLAY. De grafische afbeelding wordt vervolgens blok voor blok opgeslagen. Daarna tekent u verder of beëindigt het programma. Met

```
10 WINDOW 1,79,24,25:LOAD"
RUN
```

laadt u de afbeelding. De grafische voorstelling wordt niet door lettertekens ontsierd en instructies komen alleen in de onderste twee regels te staan. Nog twee tips. (1) Als u het programma heeft beëindigd, toets dan MODE 2 in. Dat heeft tot gevolg dat een eerder gedefinieerde vensterindeling ongedaan wordt gemaakt en dat het grafische plaatje wordt gewist. (2) Al naar gelang het type kunnen joysticks andere waarden hebben. Dit programma is geschreven voor een Atari joystick. Als u een ander type heeft, moet u de waarden in de regels 90, 120 en 130-160 aanpassen.

```
10 MODE 2
20 CLS
30 GOSUB 240
40 X=1:Y=1:Z=1
50 PLOT X,Y,1
60 JO=JOY(0): IF JO<>0 THEN 80
70 A$=INKEY$:IF A$="" THEN GOTO 70
80 PLOT X,Y,Z
90 Y=Y+(1 AND JO=1)-(1 AND JO=2)
100 IF Y<0 THEN Y=0
110 IF Y>367 THEN Y=367
120 X=X+(1 AND JO=8)-(1 AND JO=4)
130 IF JO=9 THEN X=X+1:Y=Y+1
140 IF JO=10 THEN X=X+1:Y=Y-1
150 IF JO=6 THEN X=X-1:Y=Y-1
160 IF JO=5 THEN X=X-1:Y=Y+1
170 IF X<0 THEN X=0
180 IF X>367 THEN X=367
190 IF Z=0 AND A$=" " THEN Z=1:GOTO 210
200 IF Z=1 AND A$=" " THEN Z=0
210 IF A$="E" OR A$="e" THEN GOTO 280
220 IF A$="C" OR A$="c" THEN CLS:GOTO 10
230 GOTO 50
240 REM minicursor in de linkerbenedenhoek zetten
250 ORIGIN 0,33
260 WINDOW #0,1,79,24,25
270 RETURN
280 WINDOW SWAP 0
290 INPUT "hoe is de naam van de afbeelding ?",A$
300 SPEED WRITE 1
310 SAVE A$,B,49152,16383
320 PRINT "nieuwe 'g'rafiek of 'e'inde"
330 IN$=INKEY$: IF IN$="G" OR IN$="g" THEN RUN
340 IF IN$<>"E" THEN GOTO 330
```

1.3.1 Editor in multicolor

De grafische editor uit de vorige paragraaf hebben we wat uitgebreid. Ook voor dit tekenprogramma heeft u een joystick nodig. Na de start bepaalt u in welke modus u wilt werken. Naarmate een modus meer kleuren biedt, wordt het oplossend vermogen van het beeldscherm lager. Bevalt de kleur in modus 0 niet, kies dan met de INK-instructie een andere voordat u het programma start. Modus 0 telt 14 kleuren. De spatiebalk dient in dit programma als kleurschakelaar. Het programma kiest dan steeds de volgende stift. In de linker benedenhoek van het beeldscherm staat aangegeven met welke kleur u werkt. Denk eraan dat ook de achtergrond een kleur heeft waarmee u kunt tekenen of wissen. Om de cursor te bewegen zonder lijnen achter te laten kiest u de kleur van de achtergrond. U slaat het beeld op met de E-toets. Het programma vraagt dan naar een naam voor de afbeelding en zet het plaatje vervolgens op band. De C-toets wist het beeldscherm. Met L laadt u de afbeeldingen die op band staan in het computergeheugen. Weer vraagt het programma eerst naar de naam van de afbeelding. U kunt het plaatje vervolmaken omdat het programma na het laden in de normale tekenmodus terugkeert.

```
10 CLS: MODE 1
20 PRINT" MODE 0 = 14 kleuren"
30 PRINT" MODE 1 = 4  kleuren"
40 PRINT" MODE 2 = 2  kleuren"
50 INPUT "welke mode (0,1,2)";MX
60 IF MX = 0 THEN MO=13
70 IF MX= 1 THEN MO =3
80 IF MX= 2 THEN MO =1
90 IF MX > 2 THEN GOTO 50
100 MODE MX
110 CLS
120 GOSUB 350
130 X=1:Y=1:Z=0
140 PLOT X,Y,1
150 JO=JOY(0): IF JO<>0 THEN 170
160 A$=INKEY$:IF A$="" THEN GOTO 160
170 PLOT X,Y,Z
180 Y=Y+(1 AND JO=1)-(1 AND JO=2)
190 IF Y<0 THEN Y=0
200 IF Y>367 THEN Y=367
210 X=X+(1 AND JO=8)-(1 AND JO=4)
220 IF JO=9 THEN X=X+1:Y=Y+1
230 IF JO=10 THEN X=X+1:Y=Y-1
240 IF JO=6 THEN X=X-1:Y=Y-1
```

```
250 IF JO=5 THEN X=X-1:Y=Y+1
260 IF X<0 THEN X=0
270 IF X>639 THEN X=639
280 IF A$=" " THEN Z=Z+1: IF Z>MO THEN Z=0
290 IF A$=" " THEN FOR Q1=1 TO 6:FOR Q2=1 TO 3:PLOT Q1,Q2,Z: NEXT
Q2:NEXT Q1
300 IF A$="E" OR A$="e" THEN GOTO 380
310 IF A$="C" OR A$="c" THEN CLS:GOTO70
320 IF A$="L" OR A$="l" THEN GOTO 480
330 GOTO 140
340 REM minicursor in de linkerbenedenhoek zetten
350 ORIGIN 0,33
360 WINDOW #0,1,79,24,25
370 RETURN
380 WINDOW SWAP 0
390 INPUT "hoe is de naam van de afbeelding ?",BEELD$
400 BEELD$=BEELD$+".PIC"
410 SPEED WRITE 1
420 SAVE BEELD$,B,49152,16383
430 PRINT "nieuwe 'g'rafiek of 'e'inde of 'v'erder"
440 IN$=INKEY$: IF IN$=" " THEN GOTO 440
450 IF IN$="V" OR IN$="v" THEN PRINT "
": GOTO 120
460 IN$=INKEY$: IF IN$="G" OR IN$="g" THEN RUN
470 IF IN$<>"E" AND IN$<>"e" THEN GOTO 460
480 REM beeld lezen
490 INPUT "welke afbeelding ";BEELD$
500 BEELD$=BEELD$+".PIC"
510 WINDOW 1,79,24,25: LOAD BEELD$
520 PRINT "
530 GOTO 120
```

1.4 Functieplotter

Een functieplotter is een programma waarmee wiskundige functies grafisch kunnen worden weergegeven. In de functieplotter in deze paragraaf staat de functie in regel 60. Daarna voert u de minimumwaarde, de maximumwaarde en de stapgrootte in. Door 20 waarden te testen bepaalt de subroutine (370 e.v.) bij benadering het maximum en minimum van de functiewaarden om de Y-as goed te kunnen tekenen. Na de sprong terug tekent het programma het coördinatensysteem en benoemt het. Vervolgens wordt de functie getekend, eerst het negatieve en daarna het positieve gedeelte.

Na afloop wacht het programma tot u een toets indrukt. De voorbeeldfunctie levert een parabool op waarvan het negatieve gedeelte aan de andere kant van de X-as ligt. Neem als minimum (-2), als maximum (+2) en als stapgrootte (0.1). Na een korte rekentijd plot het programma een duidelijke curve. Gebruik als variabele steeds de letter X.

```
10 REM *******FUNCTIEPLOTTER*******
20 REM
30 MODE 2
40 CLEAR
50 DEG
60 DEF FN F(X)=-(X^2)
70 PRINT "                          nul is niet toegestaan"
80 INPUT "
MINIMUM (X)";A
90 PRINT "                          nul is niet toegestaan"
100 INPUT "
MAXIMUM (X) ";B
110 IF A=0 OR B=0 THEN PRINT"
nul is niet toegestaan!!":GOTO 70
120 INPUT "stapgrootte";C
130 IF A>B THEN 30
140 GOSUB 410
150 CLG
160 PLOT 0,200:DRAW 640,200:PLOT 320,0:DRAW 320,400
170 LOCATE 1,14
180 MP=320/ABS(A)
190 MQ=320/ABS(B)
200 IF MP=MQ THEN PR=A
210 IF MP<MQ THEN PR=A
220 IF MP>MQ THEN PR=-B
230 PRINT PR
240 PR$=STR$(ABS(PR))
250 LOCATE (80-LEN(DE$)),14
260 PRINT PR$
270 DE$=STR$(DE(1))
280 LOCATE 39 -LEN(DE$),1
290 PRINT DE$
300 DE$=STR$(-DE(1))
310 LOCATE 39 - LEN (DE$),25
320 PRINT DE$
330 MN=200/DE(1)
340 FOR S=0.1 TO A STEP -C
350 PLOT 320+S*MP,200+FN F(N)*MN,1
360 NEXT S
370 FOR N=0.1 TO B STEP C
380 PLOT 320+N*MQ,200+FN F(N)*MN,1
390 NEXT N
```

```
400 A$=INKEY$:IF A$="" THEN 400 ELSE 30
410 ML=ABS(A)+ABS(B)
420 DET=ML/20
430 ML=ML+2
440 DIM DE(22)
450 FOR I=A TO B STEP DET
460 VAR=VAR+1
470 de(var)=FN f(i)
480 NEXT I
490 DE(21)=FN F(A):DE(22)=FN F(B)
500 FOR J=21 TO 1 STEP -1
510 FOR I=1 TO J
520 IF ABS(DE(I+1))<ABS(DE(I)) THEN 560
530 DE=DE(I+1)
540 DE(I+1)=DE(I)
550 DE(I)=DE
560 NEXT
570 NEXT J
580 PRINT "verwachte maximum danwel minimum van de functie in dit
bereik is: ";DE(1)
590 DE(1)=ABS(DE(1))
600 PRINT"druk een toets!"
610 A$=INKEY$:IF A$="" THEN 610
620 RETURN
```

1.5 Kleuren en tekens

De CPC heeft 27 kleuren. Helaas kunt u het volledige palet niet in elke modus gebruiken. Let bij het programmeren op deze beperking, anders wordt u met de foutmelding IMPROPER ARGUMENT geconfronteerd. Sla het handboek er op na, daar staat het allemaal duidelijk in. Ook de verschillende modi (20, 40, 80 tekens) staan in dat boek goed uitgelegd. Hier volgen de instructies voor de modi en de kleurinstelling, omdat het handboek ze nergens bij elkaar vermeldt.

Mode 0 wist het scherm en schakelt naar 20 tekens

Mode 1 wist het scherm en schakelt naar 40 tekens

Mode 2 wist het scherm en schakelt naar 80 tekens

Border bepaalt de kleur van de rand (tot en met 27);

worden er 2 kleuren aangegeven, dan verspringt de kleur

Ink kent een kleur toe (tweede waarde) aan een bepaald kanaal (eerste waarde)

Pen kent een kleur (tweede waarde) aan een kanaal (eerste waarde) toe; het gaat om tekst

Paper kent een kleur (tweede waarde) aan een kanaal (eerste waarde) toe; het gaat om de achtergrond

1.6 Waardevolle grafiekinstructies

We behandelen de volgende instructies:

PLOT en PLOTR
DRAW en DRAWR
MOVE en MOVER
ORIGIN
TEST en TESTR
XPOS en YPOS

PLOT X,Y zet een punt op positie X,Y. X en Y zijn hier absolute coördinaten. Absoluut betekent dat de positie van het punt alleen afhangt van de actuele oorsprong. Gewoonlijk ligt die bij 0,0, de linker benedenhoek van het scherm.

PLOTR X,Y zet een punt op de relatieve positie X,Y. X en Y zijn afhankelijk van de plaats van de grafiekcursor.

DRAW X,Y trekt een lijn van de cursorpositie naar de absolute coördinaten X en Y; DRAWR naar de relatieve coördinaten X en Y.

MOVE X,Y beweegt de grafiekcursor zonder een punt te zetten naar de absolute coördinaten X en Y. MOVER X,Y doet hetzelfde, maar dan gaat de cursor naar de relatieve coördinaten, dus naar de positie die wordt berekend uitgaande van de oude plaats van de grafiekcursor.

ORIGIN verandert de oorsprong die maatgevend is voor de absolute coördinaten. Verder kunt u met deze instructie een grafiekvenster

vervaardigen. Het originele venster telt 640 punten in de breedte en 200 in de lengte.

TEST X,Y informeert over de kleur van het punt op positie X, Y. TESTR X,Y doet hetzelfde ten aanzien van de relatieve positie X,Y.

XPOS en YPOS betreffen de actuele positie van de grafiekcursor. PRINT XPOS, YPOS tovert die positie in duidelijke taal op het scherm.

Nog iets over het begrip relatief. Een absolute positie komt overeen met de relatieve waarden van X en Y plus de X en de Y van de cursorpositie:

```
X(rel) + X(cur) = X(abs)
Y(rel) + Y(cur) = Y(abs)
```

1.7 Het gebruik van vensters (windows)

Het woord WINDOW (venster) is een belangrijk begrip in de computerwereld. Een window is een vast bereik op het medium waarop de computer afbeeldingen projecteert (beeldscherm, printer, plotter enzovoort). U kunt het window als zelfstandig onderdeel behandelen. De CPC heeft meerdere windows en dus meer snelheid en bedieningsgemak. Vooral bij spelletjes en programma's voor het bedrijfsleven komt dit van pas. Het programma voor databeheer (paragraaf 6.2) is er een goed voorbeeld van. De informatie die de gebruiker voortdurend nodig heeft, staat daar in een apart window, bovenaan het beeldscherm. U kunt ook met de instructie ORIGIN een window definiëren. Met drie windows kunt u op 1 scherm bijvoorbeeld een getallentabel kwijt, een histogram en een schijfdiagram. Toepassingen op het gebied van spelletjes schudt iedereen natuurlijk zo uit zijn mouw. In het algemeen geldt dat u niet steeds opnieuw allerlei zaken hoeft aan te maken. Dit in tegenstelling tot programmeren in BASIC. Tijdrovende klussen bederven het hele spelplezier. Experimenteer eens met de WINDOW-instructie aan de hand van paragraaf 6.2 of schrijf zelf een programmaatje. Ook in het handboek staan leerzame voorbeelden. Om de experimenteerdrift wat aan te wakkeren volgen twee eenvoudige programma's.

Het eerste programma produceert 8 windows vol met onzichtbare tekens omdat de achtergrondkleur is veranderd. De regels 500-510

bevatten de waarden voor de windows, regel 500 betreft de X-richting (waarden tussen 1 en 80) en 510 de Y-richting (tussen 1 en 25). Rommel er naar hartelust mee. Nog meer windows krijgt u door de cijfers 8 in de lussen te verhogen.

```
10 MODE 2
20 FOR x=1 TO 8
30 READ kolom (x)
40 NEXT x
50 FOR y=1 TO 8
60 READ regel (y)
70 NEXT y
80 PAPER 5
90 FOR x=1 TO 8
100 FOR y=1 TO 8
110 WINDOW kolom (x), kolom (x)+5,regel (y),regel (y)+5
120 PRINT "*********"
130 NEXT y
140 NEXT x
500 DATA 1,80,20,60,30,45,27,50
510 DATA 9,19,12,25,1,3,18,1
```

Verklaring van het programma:

10	80-tekens modus instellen
20- 40	lus om waarde van de X-richting in te lezen
50- 70	lus om waarde van de Y-richting in te lezen
80	achtergrondkleur instellen
90-100	lus voor X/Y openen
110	window maken
120	tekenreeks uitvoeren
130-140	einde van de lussen
500-510	DATA-regels

Het tweede programma schrijft het beeldscherm met 1 teken vol en wist 1/3 als er op een toets wordt gedrukt. Als onderdeel van programma's voor het bedrijfsleven bewijst dit kneepje goede diensten. Bij dergelijke routines moeten vaak hele delen van het scherm worden gewist. Met een WINDOW gaat dat supersnel.

```
10 MODE 2
20 WINDOW #2,1,80,1,8
30 WINDOW #3,1,80,9,17
40 WINDOW #4,1,80,17,24
50 FOR n=2 TO 4
60 FOR mn=1 TO 640
70 PRINT"*";
80 NEXT mn
```

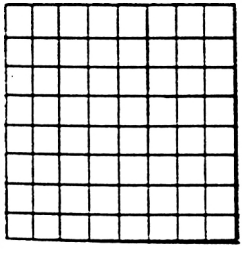
```
90 NEXT n
100 FOR x=2 TO 4
110 WINDOW SWAP x
120 a$=INKEY$:IF a$="" THEN 120
130 CLS
140 NEXT x
150 MODE 2
```

Verklaring van het programma:

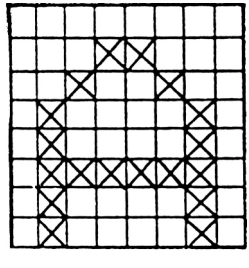
10	80-tekens modus instellen
20-40	3 windows definiëren
50-90	beeldscherm volschrijven met sterretjes
100	lus openen om van window te wisselen
110	window veranderen
120	wachten op toetsaanslag
130	window wissen
140	einde van de lus en naar het volgende window
150	de 3 windows terugzetten

1.8 Tekens die u zelf hebt bedacht

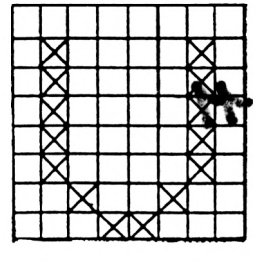
De CPC heeft nog een goede eigenschap: de mogelijkheid om zelf tekens te maken. Een teken bestaat uit gezette en niet gezette punten. Het raster waarin de tekens worden gedefinieerd, heet matrix. Afbeelding 1 toont een 8*8 matrix (8 hokjes verticaal, 8 horizontaal). Andere gebruikelijke matrixafmetingen zijn 7*8, 7*9 en bij professionele systemen zelfs 10*10 of 12*12. De grootte van de matrix is interessant omdat een groter aantal punten beter leesbare letters oplevert. Dit werkt net als hoge resolutie.



afb. 1

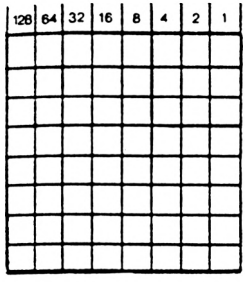


afb. 2

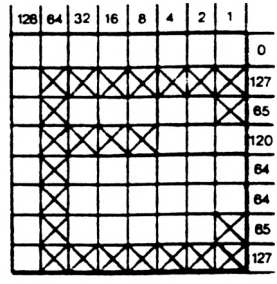


afb. 3

In figuur 2 en 3 hebben we punten in de matrix gezet die samen letters vormen. Zo is elke letter op het beeldscherm samengesteld. Bij de CPC kunt u kiezen uit 255 tekens; daarvan kunt u er 223 zelf definiëren. Maak altijd eerst een ontwerp op ruitjespapier; een 8*8 matrix is gauw uitgezet. Als de punten zijn gezet, moeten ze nog worden berekend om te kunnen worden opgeslagen. U maakt hierbij gebruik van de Boole rekenkunde. Afbeelding 4 is weer een 8*8 matrix. Boven de verticale kolommen staan de getallen 1-2-4-8-16-32-64-128. In de horizontale regels brengt u later nog getallen aan. Vul in deze matrix de punten voor uw teken in. Bekijk de eerste regel (horizontale lijn) en tel de getallen boven de gezette punten in die regel bij elkaar op. Schrijf de uitkomst naast de regel. Ga daarna verder met de volgende regel, enzovoort. De acht getallen zijn de waarden voor het teken. Afbeelding 5 geeft de letter E als voorbeeld.



afb. 4



afb. 5

U hoeft nu alleen nog ruimte voor uw teken te reserveren en het met een instructie aan een toets toe te kennen. De plaats voor uw teken(s) reserveert u met de instructie SYMBOL AFTER. Hierop volgt een getal uit het bereik 32-255. Als u bijvoorbeeld de waarde 240 kiest, kunt u alle tekens met een CHR$-code groter dan 239 (16 stuks) opnieuw definiëren. De CHR$-code staat in het handboek van de computer. Let erop dat u de instructie SYMBOL AFTER niet meer dan 1 keer gebruikt; anders worden alle oude tekens gewist. Futuristische, technische, wiskundige, chemische, Griekse, grafische - alle soorten tekens heeft u binnen handbereik als u er maar wat aandacht aan besteedt. Voor spelletjes biedt dit ongekende mogelijkheden. Een kanon dat een gevaarlijk fantasiewezen van het scherm knalt, is veel toffer dan een B die een i laat verdwijnen. Binaire getallen optellen doet niemand voor de lol en daarom hebben we een programma gemaakt dat u daarbij helpt.

1.8.1 Een handige tekengenerator

Met het volgende programma maakt u met de cursortoetsen uw eigen tekens binnen een 8*8 matrix. Na invoer en start van het programma verschijnt een rechthoek met in de linker bovenhoek een puntje. Dit is de cursor. U zet een punt door de COPY-toets te gebruiken. De computer zet dan een * op de plaats van de cursor. U wist een punt door de cursor links naast een sterretje te zetten en op de spatiebalk te drukken. Door vervolgens E in te drukken worden de codes berekend. Na korte tijd zet het programma de 8 codes van de tekening in de onderste beeldschermrand. Dan volgt de vraag welk teken u opnieuw wilt definiëren. Druk op de toets waarmee uw teken moet worden opgeroepen en op ENTER. Het is verstandig de codes van uw nieuwe teken te noteren, zodat u later nog weet welk teken bij welke toets hoort. Beantwoord de laatste vraag, of er nog andere tekens moeten worden gedefinieerd, met J of N. Als u het programma RUNt, worden alle al gedefinieerde tekens worden gewist. Wilt u uw tekens behouden, start het programma dan met GOTO 40.

```
10 SYMBOL AFTER 32
20 IF y=0 THEN y=1
30 DIM b$(8): DIM c$(8): DIM c(8)
40 MODE 1
50 GOTO 240
60 x=1 : y=1 : z=0
70 LOCATE x,y
80 PLOT x*16,(128-y*16)+8,1
90 a$=INKEY$:IF a$="" THEN 90
100 PLOT x*16,(128-y*16)+8,0
110 z=0
120 y=y+(1 AND a$=CHR$(241))-(a AND a$=CHR$(240))
130 IF y>8 THEN y=8
140 IF y=0 THEN y=1
150 x=x+(1 AND a$=CHR$(243))-(1 AND a$=CHR$(242))
160 IF x>8 THEN x=8
170 IF x=0 THEN x=1
180 IF a$=" " THEN z=2
190 IF a$ = CHR$(224) THEN z=1
200 IF z=1 THEN PRINT "*"
210 IF z=2 THEN PRINT " "
220 IF a$="e" OR a$="E" THEN 320
230 GOTO 70
240 WINDOW #0,10,17,10,18
250 ORIGIN 143,127
```

```
260 PLOT 0,0,1
270 DRAW 0,130
280 DRAW 132,130
290 DRAW 132,0
300 DRAW 0,0
310 GOTO 60
320 FOR I=1 TO 8:b$(I)="":NEXT I
330 PLOT 0,0
340 MOVER 10,120
350 FOR m=1 TO 8
360 FOR n=1 TO 8
370 IF TESTR (0,0)=1 THEN t$="1" ELSE t$="0"
380 b$(m)=b$(m)+t$
390 PLOTR 1,1,1
395 MOVER 15,-1
400 NEXT n
410 MOVER -8*16,-16
420 NEXT m
430 FOR x=1 TO 8:c$(x)="&x"+b$(x)
440 c(x)=VAL(c$(x)): NEXT x
450 WINDOW #1,1,39,20,25
460 WINDOW SWAP 0,1
470 FOR i=1 TO 8 :PRINT c(i);:NEXT i: PRINT
480 INPUT "welk teken moet opnieuw worden gedefinieerd";a$
490 a= ASC(a$)
500 SYMBOL a,c(1),c(2),c(3),c(4),c(5),c(6),c(7),c(8)
510 PRINT "moeten nog meer tekens opnieuw worden gedefineerd?"
520 in$=INKEY$:IF in$="" THEN 520
530 WINDOW SWAP 0,1
540 IF in$ = "j" OR in$="J" THEN 40
550 MODE 1
560 PRINT "einde van het programma"
```

Verklaring van de listing:

In regel 10-50 staat de zogenaamde initialisering: hier worden alle voorbereidingen voor de tekendefinitie getroffen. Dan volgt een sprong naar een subroutine, die de 8*8 matrix tekent. Het venster (window) is een regel groter gekozen om te voorkomen dat de tekening naar boven scrollt als er een punt wordt gezet in de rechter benedenhoek. Na de sprong terug zet het programma de tekstcursor in de linker bovenhoek. Hetzelfde geldt voor de tekencursor. Regel 90 leest het toetsenbord uit. Die gegevens worden verwerkt in 120-220. Als u op E drukt (regel 220) volgt een sprong naar de interpretatie van de figuur. In elke rij wordt eerst getest of een punt is gezet of niet (regel 370). Is dat het geval, dan voegt het programma aan variabele B$(M) een 1 toe, anders een 0. Zo ontstaat voor elke regel een binair getal van acht cijfers. De regels 430 en 440 veranderen de binaire getallen

in decimale waarden en slaan ze op in variabele C(W). Daarna vervaardigt het programma een tekstvenster, waarin het de codes zet en de vragen stelt. Regel 500 neemt het nieuwe teken op in de karakterset. Tenslotte maakt MODE 1 alle windows ongedaan.

1.9 Grafische karakters

Grafische tekens goed gebruiken is niet eenvoudig. Veel computerbezitters kunnen er na jaren nog niet mee omgaan. De tekens staan met hun codes in de bijlage van het handboek. Daar staan ook sprites zoals een poppetje, een bom, enzovoort. Dat zijn van tevoren gedefinieerde grafische voorstellingen, net als karakters, maar dan uitgebreider. U roept deze tekens op met de functie CHR$ (argument). Door PRINT CHR$ (252) in te voeren, verschijnt er een bom op uw beeldscherm. Met LOCATE en PRINT kunt u de sprites bewegen. Probeer het volgende programma maar eens:

```
10 MODE 2
20 FOR Y=1 TO 25
30 LOCATE 40,Y:PRINT CHR$(252)
40 LOCATE 40,Y:PRINT " "
50 NEXT Y
```

Dit programma laat een bom vallen van de bovenste naar de onderste beeldschermrand. Dat gebeurt door een lus en de LOCATE- en PRINT-instructies. Zo programmeert u goede spelletjes. Hieronder een programma dat de snelheid van deze methode aantoont. Het eindigt vreemd genoeg met een foutmelding, maar dat geeft niks. Roep na afloop de waarde voor Y maar eens op; dat is de sleutel tot de oplossing van de mysterieuze fout.

```
10 MODE 2
20 FOR x=1 TO 25
30 t=36
40 q=12
50 LOCATE t,x:PRINT STRING$(&8,CHR$(249))
60 NEXT x
70 y=1
80 FOR x=q TO q+4
90 LOCATE y,x:PRINT STRING$(&8,CHR$(250))
100 NEXT x
110 FOR x=q TO q+4
120 LOCATE y,x:PRINT STRING$(&8," ")
```

```
130 NEXT x
140 y=y+8
150 GOTO 80
```

Verklaring van het programma:

10	80-tekens modus instellen
20	lus openen voor neerwaartse beweging
30	beginwaarde voor neerwaartse beweging van de mannetjes
40	Y-waarde voor de onbeweeglijke mannetjes
50- 60	lus voor de uitvoer van de verticale mannetjes
70	Y wordt op 1 gezet
80-100	lus voor horizontale beweging in stappen van 4
110-130	lus om het laatste blok mannetjes te wissen
140	verhoging van de waarde, zodat de mannetjes zich in blokken voorwaarts kunnen bewegen
150	sprong terug naar de bewegingslus

1.10 Het beeldschermgeheugen

Het hoofdgeheugen van de CPC heeft een omvang van 64 Kbyte. Dat zijn 65536 bytes. Ongeveer 43 Kbyte is vrij voor eigen BASIC-programma's. Het complete geheugen is verdeeld in 4 blokken van 16 Kbyte. Deze blokken worden banks (banken) genoemd en zijn genummerd van 0 tot en met 3. Bank 0 komt overeen het bereik van de geheugenadressen 0 (decimaal) tot 16383 (dec) en bank 3 met het bereik van 49152 (dec) tot 65535 (dec). Als u de computer inschakelt, beslaat het beeldschermgeheugen bank 3. Onafhankelijk van de modus waarin u werkt, wordt dus 16 Kbyte voor het beeldschermgeheugen gereserveerd. Voer de volgende 4 regels in en start het programma met RUN.

```
10 REM beeldscherm 1
20 MODE 2 : PRINT "a"
30 FOR t = &C000 to &FFFF STEP &800
40 PRINT BIN$(PEEK(t) ,8) : NEXT t
```

Als u alles juist heeft gedaan, krijgt u de volgende informatie:

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0
0 0 0 0 1 1 0 0
0 1 1 1 1 1 0 0
1 1 0 0 1 1 0 0
0 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0
```

U herkent duidelijk het bitpatroon van de kleine a. Die hebben we in regel 20 in de linker bovenhoek van het beeldscherm gezet. Verander het programma als volgt:

```
10 REM beeldscherm 2
20 FOR t = &C000 to &FFFF STEP &800
30 FOR a = 0 TO 3
40 PRINT BIN$(PEEK(t+a),8);" ";
50 NEXT a : PRINT
60 NEXT t
```

Toets daarna in de directe modus MODE 2 in en daarna RUN. U ziet vier blokken met bitpatronen: de letters R e a d. Dat is niet vreemd, want in de eerste regel staat na elke MODE-instructie het woord READY. Als u MODE 1 en RUN invoert ziet u alleen nog een R en een e, maar het aantal gezette bits is in vergelijking met MODE 2 niet veranderd. Elke letter beslaat 2 bitpatroonblokken. Maar tegen de verwachting in zijn de 4 rechterbits van elk blok 0. Hoe kunnen de letters twee keer zo groot zijn terwijl er niet meer bits zijn gezet dan in de 80-tekens modus? De oplossing komt u op het spoor door het volgende in te toetsen:

```
PEN 2
MODE 1
RUN
```

Weer staan de eerste twee letters van READY in de eerste regel. Elke letter beslaat 2 bitpatroonblokken. Nu is steeds de linkerhelft van een blok gelijk aan 0 terwijl de gezette bits, die de letters weergeven, aan de rechterkant zijn te vinden. De verklaring ligt bij de opbouw van het beeldschermgeheugen. Die is afhankelijk van de gekozen MODE. Het 16 Kbyte beeldschermbereik is verantwoordelijk voor twee functies. Het bitpatroon van het teken wordt opgeslagen en dezelfde bits zorgen voor de kleur van het teken dat wordt weergegeven. Het is zo langzamerhand tijd om de modi aan een nader onderzoek te onderwerpen.

1.10.1 Modus 2

In modus 2 is de zaak vrij eenvoudig. Elke gezette bit komt overeen met een gezette pixel (beeldpunt) die de computer in de tekenkleur uitvoert. Een 1 in het bitpatroon staat voor de tekenkleur 24 (lichtgeel), een 0 voor kleur 1 (blauw). Voor de tekenkleur blauw dient u de volgende regels in de directe modus in te voeren:

```
MODE 2
PAPER 1: PEN 0
CLS
```

Het beeldscherm is nu geel geworden en de tekenkleur blauw. Start het programma BEELDSCHERM 2 uit de vorige paragraaf met RUN. In het bitpatroon van elk teken geven nullen aan dat een beeldpunt is gezet. De kleur blauw is de nieuwe tekenkleur geworden. Samengevat: in modus 2 bevat elke bit in het beeldschermgeheugen informatie over de kleur van de beeldpunten.

```
0 = blauw
1 = geel
```

1.10.2 Modus 1

Schakel de CPC op MODE 1. Deze modus met 40 tekens per regel heeft vier kleuren. De computer legt in elke byte van het beeldschermgeheugen vast welke kleur de beeldpunten heeft. Voordat we laten zien hoe de CPC de kleur en het te zetten punt opslaat, bekijken we eerst een byte:

```
7  6  5  4  ||  3  2  1  0
```

In modus 1 is elke byte verdeeld in 2 nibbles (halve bytes) van elk 4 bits, die hier door || zijn gekenmerkt. De twee bits met dezelfde positie in beide nibbles vormen een paar. Dat zijn dus de bits (3,7), (2,6), (1,5) en (0,4). Met deze vier paren moeten alle acht beeldpunten worden aangesproken, maar dat is onmogelijk. Daarom voegen we twee naast elkaar liggende

beeldpunten samen en kennen ze toe aan een van de vier genoemde bitparen. Zo kunt u elk pixelpaar 1 van de 4 mogelijke kleuren geven. De bits (3,7) bevatten de informatie voor het meest linkse pixelpaar. In de volgende tabel ziet u welk bitpatroon met welke kleur overeenkomt.

bitpatroon	kleur
0 0	blauw
0 1	lichtgeel
1 0	licht blauwgroen
1 1	lichtrood

Een voorbeeld uit de praktijk. Herinnert u zich nog het vreemde uiterlijk van de bitpatroonblokken die we in MODE 1 uitlazen? Probeer de voorbeelden bij het programma BEELDSCHERM 2 opnieuw. De eerste rij van het eerste bitpatroonblok ziet er als volgt uit:

1 1 1 1 0 0 0 0

De tabel laat zien dat de bitparen corresponderen met de volgende kleuren.

bitpaar	3,7	2,6	1,5	0,4
waarde	0 1	0 1	0 1	0 1
kleur	geel	geel	geel	geel

Teken nu op ruitjespapier naast elkaar 4 vierkanten van 8 bij 8 hokjes en verdeel ze in 4 kolommen (elk 2 hokjes breed) en 8 rijen. Elk hokjespaar komt overeen met twee naast elkaar liggende beeldschermpunten. Arceer elk hokjespaar als de twee corresponderende informatiebits 'op geel staan' en laat de hokjes leeg als ze blauw aangeven. Andere kleuren dan geel en blauw betekenen dat u een fout heeft gemaakt. Zijn alle regels van de bitpatronen op het scherm ontcijferd, dan staan er op papier een brede R en een brede e. De volgende stap die we ondernemen betreft de tekenkleur. Verander die met PEN 2 en toets vervolgens CLS in en RUN. De bitparen hebben nu de waarde 1 0 : licht blauwgroen. Zelf vierkleurentekens ontwerpen is op deze manier geen probleem. Let erop dat elk teken slechts de halve breedte van een normaal teken in MODE 1 heeft. Verder kunt u dit teken niet met SYMBOL of SYMBOL AFTER in de karakterset opnemen. Alleen met POKE kunnen tekens die uit vier of meer kleuren bestaan op het scherm worden gezet. Staan de waarden eenmaal in DATA-regels dan loopt het verder allemaal op rolletjes.

1.10.3 Modus 0

Deze modus laat zien hoe goed de CPC veel kleuren op een klein oppervlak kan weergeven. Helaas geldt wel: hoe groter het aantal kleuren, des te lager de resolutie. Om elke pixel een willekeurige kleur te kunnen geven uit een palet van 16 kleuren is een beeldschermgeheugen nodig van 64 Kbyte. Net als in modus 1 staat 1 kleurwaarde voor meerdere beeldpunten. In de vierkleurenmodus waren er per kleur twee beeldpunten. In de multicolormodus komen vier pixels overeen met een kleur. De volgende kleuren staan tot uw beschikking.

kleurentabel voor de multicolormodus:

bin	hex	dec	kleur
0000	0	00	blauw
0001	1	01	lichtgeel
0010	2	02	licht blauwgroen
0011	3	03	lichtrood
0100	4	04	helder wit
0101	5	05	zwart
0110	6	06	lichtblauw
0111	7	07	licht magenta
1000	8	08	blauwgroen
1001	9	09	geel
1010	A	10	pastelblauw
1011	B	11	rose
1100	C	12	lichtgroen
1101	D	13	pastelgroen
1110	E	14	knipperlicht: blauw-geel
1111	F	15	knipperlicht: roze-hemelsblauw

Met een klein programma tovert u deze 16 kleuren op het beeldscherm.

```
10 REM kleurenvoorbeeld
20 DATA &00,&11,&22,&33,&44,&55,&66,&77
25 DATA &88,&99,&AA,&BB,&CC,&DD,&EE,&FF
30 MODE 0
40 FOR d = 0 TO 31 STEP 2
50 READ x
60 FOR t = &C000 to &FFFF STEP &800
70 POKE t+d,x
80 NEXT t : NEXT d
90 GOTO 90
```

Met RUN start u het programma. Maar in plaats van de verlangde 16 kleuren krijgt u een bont patroon te zien. Dit komt doordat de kleurwaarden in een byte net als in modus 1 niet in vier samenhangende bits staan. De code is over de byte verstrooid. De bits 1, 3, 5 en 7 bevatten kleurinformatie voor de vier rechter beeldpunten en de bits 0, 2 ,4 en 6 voor de vier linker. Voordat de informatie de binaire vorm krijgt die in de tabel staat, heeft zij een vreemde weg afgelegd:

```
linker beeldpunten :  1, 5, 3, 7
rechter beeldpunten:  0, 4, 2, 6
```

Nu kunt u alle kleuren aanspreken. In de volgende tabel hebben de linker en de rechter beeldpunten dezelfde kleur. Lang niet alle mogelijke bitpatronen zijn er dus in opgenomen.

bitpatroon	links	rechts	byte
7 6 5 4 3 2 1 0			
0 0 0 0 0 0 0 0	00	00	0
1 1 0 0 0 0 0 0	01	01	192
0 0 0 0 1 1 0 0	02	02	12
1 1 0 0 1 1 0 0	03	03	204
0 0 1 1 0 0 0 0	04	04	48
1 1 1 1 0 0 0 0	05	05	240
0 0 1 1 1 1 0 0	06	06	60
1 1 1 1 1 1 0 0	07	07	252
0 0 0 0 0 0 1 1	08	08	3
1 1 0 0 0 0 1 1	09	09	195
0 0 0 0 1 1 1 1	10	10	15
1 1 0 0 1 1 1 1	11	11	207
0 0 1 1 0 0 1 1	12	12	51
1 1 1 1 0 0 1 1	13	13	243
0 0 1 1 1 1 1 1	14	14	63
1 1 1 1 1 1 1 1	15	15	255

Vervang de hexadecimale getallen in de DATA-regel van het programma KLEURENVOORBEELD door de decimale getallen (& is dus overbodig) onder het kopje 'byte':

```
20 DATA 0,192,12,204,48,240,60,252,
        3,195,15,207,51,243,63,255
```

Als u het programma nu start, krijgt u de 16 kleuren in de juiste volgorde. U ziet maar 15 balken omdat de eerste balk kleur 0 heeft (de achtergrondkleur).

1.10.4 Positie van een teken

```
10  REM positie van een teken
20  MODE 0
30  INPUT "welke kolom ";s
40  LOCATE 1,1
50  INPUT "welke regel ";z
60  LOCATE 1,1
70  INPUT "kleurpatroon";kleur
80  LOCATE 1,1
90  basis = &C000
100 FOR adres = basis TO basis + &3FFF STEP &800
110 POKE adres + (s-1) + ((z-1)*80),kleur
120 NEXT adres
130 GOTO 30
```

Dit programma POKEt kleine rechthoeken ter grootte van de cursor in MODE 2 op het scherm. Met regel 30 en 50 bepaalt u de plaats. De vraag naar het kleurpatroon heeft niets te maken met de kleurtabellen in het CPC handboek, het gaat om de waarde die u zelf heeft berekend (zie paragraaf 1.10.3). In dit voorbeeld is de waarde van de variabele 'basis' (regel 90) het standaardadres waar het beeldschermgeheugen begint. In hoofdstuk 5 presenteren we een machinetaalroutine waarmee u het begin van het beeldschermgeheugen kunt veranderen. Stapgrootte en eindadres stelt het programma automatisch in. Regel 110 zet het gekleurde vakje op de juiste plaats. Bedenk wel dat de HOME-positie de coördinaten 1,1 heeft. Dit programma test de ingevoerde waarden niet op juistheid, maar dat is wel in te bouwen.

STEP &800

U zult hebben gemerkt dat de stapgrootte STEP &800 steeds opduikt. Bij veel computers volgen de acht rijen (bytes) van een teken in het beeldschermgeheugen op elkaar: de karakters zijn keurig geordend in blokken van acht bytes. Bij de CPC daarentegen worden de eerste rijen (bytes) van alle 2000 beeldschermposities (25 regels maal 80 tekens) aaneengeregen tot een ketting. In de tweede ketting staan de tweede rijen, enzovoort. Het beginadres van de eerste ketting is tevens het begin van het beeldschermgeheugen. Alle andere waarden krijgt u door bij de vorige 2048 op te tellen. &800 is de hexadecimale schrijfwijze van 2048.

rij	hex	dec
1	C000	49152
2	C800	51200
3	D000	53248
4	D800	55296
5	E000	57344
6	E800	59392
7	F000	61440
8	F800	63488

We hadden het net over 2000 beeldschermposities, maar ook over stappen van 2048 bytes. 48 bytes aan het eind van elk blok worden niet gebruikt, daar zit 'm de kneep. Verandert u de offset van het beeldscherm, bijvoorbeeld om horizontaal te scrollen, dan worden ze zichtbaar. Wees voorzichtig met het gebruik van die ruimtes voor eigen machinetaalroutines. In paragraaf 5.6 staan offset en de mogelijkheid van horizontale scrolling uitvoerig beschreven.

1.11 Alternatieve tekens op een andere manier

Met de instructies SYMBOL en SYMBOL AFTER kunt u eigen tekens ontwerpen en gebruiken. Maar het komt ook voor dat alle gewone tekens op zijn en er toch nog andere tekens moeten worden gedefinieerd. Deze paragraaf voorkomt dat u in die gevallen de kluts kwijt raakt.

```
10 REM karakterset uitlezen
20 MODE 1
30 FOR t = &AB80 TO &AB87
40 PRINT BIN$(PEEK(t),8)
50 NEXT t
```

Op het beeldscherm verschijnt het bitpatroon van de CURSOR OMHOOG-toets (pijl naar boven). Dit teken heeft de waarde CHR$(240). De laatste 16 tekens (chr$240-chr$255) zijn bijzondere tekens. U kunt ze net zo eenvoudig veranderen als aanspreken. Probeer de volgende BASIC-instructie eens:

```
POKE &AB87,255
```

Start het programma dat nog in het geheugen zit met RUN. De naar

boven gerichte pijl heeft een voet gekregen. Dit nieuwe teken kunt u natuurlijk ook op het beeldscherm laten verschijnen.

```
PRINT CHR$(240): PRINT
```

Door de tweede PRINT-instructie staat het woord READY niet direct onder het symbool, zodat u de voet van de pijl beter ziet. Hier de lijst van tekens die u met deze methode kunt gebruiken.

hex		decimaal		chr$( )
van	tot	van	tot	
AB80	AB87	43904	43911	240
AB88	AB8F	43912	43919	241
AB90	AB97	43920	43927	242
AB98	AB9F	43928	43935	243
ABA0	ABA7	43936	43943	244
ABA8	ABAF	43944	43951	245
ABB0	ABB7	43952	43959	246
ABB8	ABBF	43960	43967	247
ABC0	ABC7	43968	43975	248
ABC8	ABCF	43976	43983	249
ABD0	ABD7	43984	43991	250
ABD8	ABDF	43992	43999	251
ABE0	ABE7	44000	44007	252
ABE8	ABEF	44008	44015	253
ABF0	ABF7	44016	44023	254
ABF8	ABFF	44024	44031	255

2 SOUND

2.1 De aansluiting van de CPC op een stereo-installatie

De ingebouwde luidspreker van de CPC levert geen hi-fi kwaliteit. Het uitgangssignaal is evenmin geschikt voor een koptelefoon. Voor een goede geluidsweergave moet u de computer op uw versterker aansluiten. Hiervoor heeft u een afgeschermde kabel nodig, met aan de CPC-kant een jackplug van 3.5 mm (3-polige uitvoering). De keuze van de stekker die in de versterker moet, is afhankelijk van het type microfooningang. De microfooningang is aan te bevelen, want die is het gevoeligst en lineair. Lineair wil zeggen dat het signaal niet wordt gecorrigeerd, zoals bij de phono-ingang (RIAA-correctie). Dat is de reden waarom de phono-ingang minder geschikt is om de computer op aan te sluiten. De tape-ingang is weliswaar lineair, maar veel minder gevoelig dan microfoon- of phono-ingang. Zet de apparaten uit voordat u ze op elkaar aansluit. Dat voorkomt complicaties. Als u ze weer heeft ingeschakeld, moet u de versterker instellen om niet door de geluidssterkte te worden verrast. Dat doet u zo:

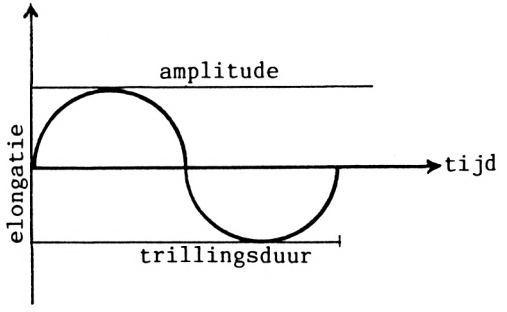
```
SOUND 7,284,32767,7
```

Deze instructie produceert de toon van de noot A die musici voor het stemmen van hun instrumenten gebruiken.

2.2 Elementaire kennis van geluid

Muziek of liever het produceren van geluid is een belangrijk onderdeel van de homecomputer. Om duidelijk te maken welke mogelijkheden de CPC heeft, gaan we eerst in op het menselijk gehoor. De doorsnee volwassene hoort geluiden van ongeveer 16 Hz tot 16 kHz. De CPC kan tonen produceren van 30 Hz - 125 kHz. U kunt dus niet alle tonen van de CPC horen. Een hond hoort de toon van de instructie SOUND 1,1,100 nog wel omdat zijn gehoorgrens hoger ligt. Voor een goed begrip moeten we ook wat weten van het uiterlijk van een toon. Tegenwoordig gaat men ervan uit dat een toon uit golven (of trillingen) bestaat. Een toon is hoger als hij een hogere frequentie heeft (er worden meer trillingen in dezelfde tijd voorgebracht). De meeteenheid voor de frequentie is

Hertz, afgekort Hz. Een voorbeeld: de centrale C heeft een frequentie (trillingsgetal) van 261,626Hz. De toon A heeft een frequentie van 440 Hz en is daarom hoger dan de C. Voor de duidelijkheid volgt hier een afbeelding van een trilling met commentaar.



amplitude

elongatie

tijd

trillingsduur

Verklaring van de begrippen:

1 elongatie (de actuele uitslag, gerelateerd aan de ruststand)

Dit begrip is alleen een dure naam voor de waarden op de Y-as.

2 amplitude (grootste elongatie of trillingswijdte)

De waarde van de geluidssterkte: hoe groter de amplitude, des te harder de toon.

3 tijd of toonduur

Deze factor speelt bij de vervaardiging van de omhullende curven een belangrijke rol. In ons voorbeeld staat dit begrip voor de X-as.

4 trillingsduur

De tijd die nodig is voor een volledige trilling. De trillingsduur is afhankelijk van de frequentie. Een hogere frequentie betekent een lagere trillingsduur.

2.3 Geluidsinstructies

De belangrijkste geluidsinstructies van de CPC zijn:

SOUND
ENV
ENT
RELEASE

SOUND is de basisinstructie voor het voortbrengen van tonen. De volgende parameters zijn daarbij in de genoemde volgorde van belang:

Kanaalstatus: deze waarde maakt de computer duidelijk welk kanaal wordt gebruikt en welke andere activiteiten worden ontplooid, zoals het samengaan met andere kanalen of een stop. De volgende byte informeert de computer over de gewenste actie:

bit	waarde	functie
0	1	stuurt de toon naar kanaal A
1	2	stuurt de toon naar kanaal B
2	4	stuurt de toon naar kanaal C
3	8	samen met kanaal A (rendez-vous)
4	16	samen met kanaal B (rendez-vous)
5	32	samen met kanaal C (rendez-vous)
6	64	stop
7	128	flush

Als u een toon door 3 kanalen tegelijk stuurt, moet de kanaalstatus dus 7 zijn. Via een combinatie van kanalen kunt u het toonverloop van de drie kanalen naadloos op elkaar laten aansluiten. Storende bijgeluiden bij pauzes behoren tot het verleden. Wilt u de tonen van een melodie op bepaalde momenten uit 2 of 3 kanalen laten klinken dan moet u rendez-vous arrangeren. Het volgende schema geeft een voorbeeld van een rendez-vous structuur. NO betekent dat een noot wordt gespeeld en R, gevolgd door A, B of C, kenmerkt een rendez-vous met een

bepaald kanaal. PA betekent pauze: er wordt op een rendez-vous gewacht. x <- y betekent dat y de oorzaak is van x.

	1	2	3	4	5	6	7	8
A	NO	NO	NO	PA<-RB	NO<-RB	NO<-RC	--	--
B	NO	NO<-RC	NO	NO	NO<-RA	NO	--	--
C	NO	NO<-RB	PA<-RA	PA<-RA	PA<-RA	NO<-RA	NO	NO

Om de verschillende kanalen te synchroniseren wordt een pauze ingelast in het kanaal dat voor een samenklank wordt aangesproken. In het volgende programma hebben we gebruik gemaakt van een aantal rendez-vous.

```
 10 SOUND 1,200,200
 20 SOUND 1,248,200 : REM kanalen A, B en C tegelijk met elk
                      een andere toon
 30 SOUND 4,284,200
 40 SOUND 1,248,200 : REM toon van kanaal A samen met de toon
                      van kanalen B en C
 50 SOUND 6,200,200 : REM 1 toon via kanalen B en C
 60 SOUND 1,284,200 : REM kanalen A en B tegelijk met elk een
                      andere toon
 70 SOUND 2,248,200 : REM terwijl er op kanaal C een pauze is
 80 SOUND 5,200,200 : REM kanaal C en A spelen dezelfde toon al
                      regel 60-70 klaar is
 90 SOUND 3,284,200 : REM kanaal A en B spelen dezelfde toon na
                      zodra regel 80 klaar is
100 SOUND 2,200,200 : REM kanaal B speelt pas als regel 90
                      klaar is
110 SOUND 4,248,200 : REM kanaal C speelt tegelijk met regel 90
120 SOUND 3,248,200 : REM kanaal A en B spelen dezelfde toon na
                      regel 100
130 SOUND 4,84,200  : REM kanaal C speelt direct na regel 110
```

U kunt de kanalen ook synchroniseren door een stop in te voeren. Het kanaal stopt met spelen en met de instructie RELEASE roept u het weer op. Het voordeel ten opzichte van het rendez-vous is dat het kanaal bij een stop kan worden bezet, terwijl het rendez-vous geen andere toon in het kanaal toelaat voordat de ontmoeting heeft plaatsgevonden.

Nootwaarde: raadpleeg de bijlagen van uw handboek. Normaal heeft u alleen de waarden van het nul-octaaf nodig, alle andere waarden berekent u als volgt:

nootwaarde = nootwaarde in octaaf 0 / 2 ^ octaaf

U berekent bijvoorbeeld de waarde van c in het octaaf -2 zo: 478/2^-2 of 478/0.25.

Toonduur: wordt in honderdste seconden aangegeven. Een negatieve waarde geeft aan hoe vaak de toonduur van ENV wordt herhaald. Bij 0 is de toonduur van ENV gehandhaafd.

Geluidssterkte: 0-7 als ENV niet is ingesteld, anders 0-15.

ENV/ENT daaraan zijn de twee volgende paragrafen gewijd.

Noise: aan het eind van de SOUND-instructie kunt u eigen geluiden toevoegen

De volgende instructie, die nauw samenhangt met de geluidsinstructie, is RELEASE. U kunt er een pauze mee ongedaan maken. Aleen het kanaalcijfer moet worden gespecificeerd (A=1, B=2, C=4). Zo langzamerhand is het tijd voor een aangename onderbreking. "Oh When The Saints Go Marching In" is een aardig voorbeeld van de geluidsmogelijkheden van de CPC 464 mogelijkheden. Sla het op voor later gebruik.

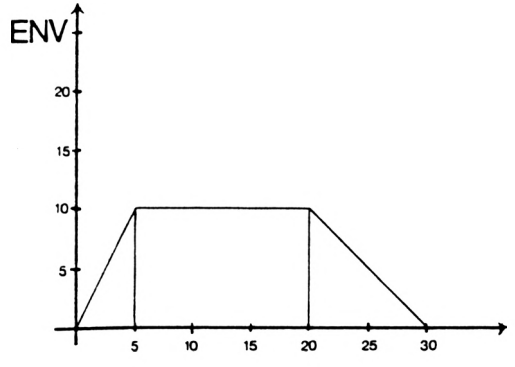
```
10 READ a,b
20 IF a=-1 THEN RESTORE: GOTO 10
30 SOUND 1,a,b
40 SOUND 2,0.5*a,b
50 SOUND 4,0.25*a,b
60 GOTO 10
70 DATA 478,50,379,50,358,50,319,200,0,5,319,50,478,50,379,
50,358,50,319,200,0,5,319,100
80 DATA 478,50,379,50,358,50,319,100,379,100,478,100,379,
100,426,200
90 DATA 0,5,426,50,379,50,0,5,379,50,426,50,478,150,0,5,
478,50
100 DATA 379,100,319,100,0,5,319,50,358,150,0,5,358,100,
379,50,358,50,319,100,379,100
110 DATA 478,100,426,100,478,200,0,5,478,50,-1,1
```

2.4 De ENV-instructie en de omhullende curve van het volume

De ENV-instructie verandert de geluidssterkte van een toon zodat het lijkt alsof een instrument die voortbrengt. U krijgt een beter resultaat door de geluidssterkte tijdens het spelen te variëren. Let erop dat het volume bij de geluidsinstructie 0 is zodat de omhullende curve volledig tot zijn recht komt. De omhullende curve krijgt een label of ENV-nummer, dat in de geluidsinstructie wordt gezet om de curve op te roepen. Dit label is de eerste waarde van de ENV-instructie. Daarna volgen drie parameters: het stapgetal, de stapgrootte en de pauzetijd. Ze kunnen vijf keer achter elkaar worden ingevoerd. Het stapgetal geeft het aantal stappen aan waarmee de geluidssterkte toe- of af moet nemen. Stapgrootte maal pauzetijd (=stapduur) is toonduur. Is deze waarde negatief, dan daalt het volume. De pauzetijd is de tijd waarin de geluidssterkte gelijk blijft. Deze wordt in honderdste seconden berekend. Probeer de volgende ENV eens in ons deuntje:

```
ENV 1,5,2,1,1,0,16,5,-3,2
```

Het resultaat is verbluffend. Maar hoe komen we aan die waarden? In het handboek zit een blad waarop u de omhulllende curve kunt tekenen. Let erop dat u voor elk deel van de curve drie waarden noteert: voor het attack-gedeelte (schuin omhoog), het sustain-gedeelte (horizontaal) en het decay-gedeelte (schuin omlaag). Bekijk het volgende plaatje maar eens.



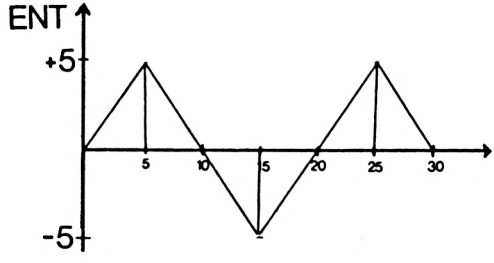
2.5 ENT en de omhullende curve van de toon

De omhullende curve van de toon is in principe hetzelfde als die van het volume. Een verandering daarvan heet vibrato, een

minimale verandering van de toonhoogte, waardoor de toon lijkt te trillen. De opbouw van de instructie met stapgetal, stapgrootte en pauzetijd is hetzelfde als bij ENV. In tegenstelling tot de drie onderdelen van ENV (attack, sustain en decay) zijn er bij ENT gewoonlijk vijf. Vergeet niet in het programma "Oh When The Saints" de soundregels zo te veranderen dat het programma de ENT ook aanspreekt. U moet de waarde 1 invoeren. Dit is het ENT-nummer, gelijk aan het label bij de ENV-instructie. Als u een negatief nummer invoert, wordt de omhullende curve herhaald. Toets de volgende instructie in:

```
ENT 1,5,1,1,10,-1,1,10,1,1,5,-1,1
```

Ook voor ENT is een curve getekend. Hier zijn dezelfde eenheden gekozen als bij ENV.



Eigenlijk zijn de omhullende curves geen curves. We zouden ze blokgolftrillingen moeten noemen omdat we alleen de hoekpunten van de curves kunnen berekenen. De kortste weg tussen twee punten is een rechte lijn en die heeft met een curve niets te maken. Met de beide instructies voor omhullende curves kunt u in theorie elke toon zo modificeren dat u hem nauwelijks meer herkent. Dat is handig bij een synthesizer. De soundeditor uit de volgende paragraaf dient daarbij als basis. Verander de tonen met de cursortoetsen en de ENV-instructie door de waarden voor de stapgrootte of het stapgetal te laten stijgen met de CURSOR OMHOOG-toets. Door met de parameters te experimenteren krijgt u een beter inzicht in de omhullende curves. Als u een instrument wilt simuleren, moet u precies weten hoe dat klinkt. Een snaarinstrument heeft meestal een korte nagalm door het trillen van de snaren. Om dit te bereiken moet bij ENV de derde fase (het afzwakken) langer duren dan het aanzwellen. Bij slaginstrumenten zoals een drumstel of een pauk is er geen nagalm. Hier kan het afzwakken korter duren dan het aanzwellen. Het wordt al

moeilijker de omhullende curve voor een blaasinstrument te berekenen, want er zijn verschillende manieren om trompet of schuiftrompet te spelen. Het belangrijkste is de geluidssterkte abrupt te laten aanzwellen, omdat de toon meer of minder zacht inzet, afhankelijk van de luchtkolommen die u moet laten trillen (dus afhankelijk van de lengte van het resonantielichaam). De manier van spelen bepaalt de decay-tijd.

2.6 De soundeditor

Muziek programmeren is moeilijk als u alleen met berekeningen werkt. Daarom staat in deze paragraaf een programma waarmee u een toon produceert door een toets in te drukken. Let wel: een toon en niet zomaar een geluid. De toetsen c, d, e, f, g, a en b produceren in dit programma tonen. Dat komt dus overeen met een toonladder in c. Halve tonen krijgt u door tegelijk SHIFT in te drukken. U kunt met dit programma melodieën

1 programmeren
2 opslaan
3 spelen
4 laden

door op 1, 2, 3 of 4 te drukken.

Ad 1 Met deze optie speelt u een wijsje (in het begin misschien nog wat hortend en stotend) dat de computer gelijk onthoudt. Gelukkig zorgt optie 4 voor een feilloze weergave. Een pauze programmeert u met de spatiebalk. De numerieke toetsen 1-9 corresponderen met de negen verschillende octaven die de CPC in zijn mars heeft. Met p laat u de laatst gebruikte toets op het beeldscherm verschijnen. De deuntjes mogen ten hoogste 1000 tonen groot zijn om het geheugen niet te veel te belasten. In de rechter bovenhoek van het scherm staat het aantal geprogrammeerde tonen. Een toets ingedrukt houden verlengt de toonduur. De pauze die hierdoor bij het programmeren ontstaat, is bij het afspelen van de melodie te verwaarlozen. Om vanuit punt 1 weer naar het menu terug te keren drukt u op ENTER.

Ad 2 Voer de naam van de melodie in en zet de cassetterecorder op opname. De noten worden als variabelenlijst opgeslagen. Met 4

kunt u ze weer laden. Cassettegebruikers, sla de melodieën niet te dicht bij elkaar op, want dan is de kans groot dat ze over elkaar heen komen te staan. Bovendien zijn de wijsjes met wat meer tussenruimte makkelijker terug te vinden.

Ad 3 De tonen van een wijsje dat u heeft geprogrammeerd, worden in de juiste volgorde afgespeeld.

Ad 4 U kunt de met 4 geladen melodie met 3 afspelen. Het programma in deze paragraaf heeft geen mogelijkheden de melodie te veranderen, maar dat euvel kan door een kleine ingreep worden verholpen.

Andere punten die voor wijziging in aanmerking komen, zijn:

a) Het programma werkt slechts met 1 kanaal. Door twee extra geluidsinstructies en een andere indeling van het toetsenbord heft u dit op.

b) Het volume kan niet met het programma worden veranderd. Dit regelt u via de volumeknop van de CPC of de stereo-installatie.

De volgende tabel behandelt de functie van de toetsen.

toets	betekenis	commentaar
c	c	noten in volgorde
C	cis	van toonladder in c
d	d	
D	dis	
e	e	
f	f	
F	fis	
g	g	
G	gis	
a	a	
A	ais	
b	b	
spatie	pauze	
p	toets op scherm	aan en uit
1	octaaf -3	octaafgetal volgens
2	octaaf -2	handboek

```
3       octaaf -1
4       octaaf  0
5       octaaf  1
6       octaaf  2
7       octaaf  3
8       octaaf  4
9       octaaf  5   staat niet in handboek

ENTER   eind        terug naar menu


10 DIM toon(1000)
20 MODE 2
30 LOCATE 5,10:PRINT "melodie programmeren      :1"
40 LOCATE 5,12:PRINT "melodie opslaan           :2"
50 LOCATE 5,14:PRINT "melodie spelen            :3"
60 LOCATE 5,16:PRINT "melodie laden             :4"
70 LOCATE 5,20:INPUT "uw keuze ";ke
80 ON ke GOTO 100,500,330,390
90 END
100 CLS
110 PRINT "M E L O D I E   P R O G R A M M E R E N"
120 N=0
130 FOR tonen=1 TO 1000
140 a$=INKEY$:IF a$= "" THEN 140
150 a=a+(478 AND a$="c")+(426 AND a$="d")
160 a=a+(379 AND a$="e")+(358 AND a$="f")
170 a=a+(319 AND a$="g")+(284 AND a$="a")
180 a=a+(253 AND a$="b")+(451 AND a$="C")
190 a=a+(402 AND a$="D")+(338 AND a$="F")
200 a=a+(301 AND a$="G")+(268 AND a$="A")
210 IF a$="p" AND pr=0 THEN pr=1:GOTO 140
220 IF a$="p" AND pr=1 THEN pr=0
230 IF a$=CHR$(13) THEN 20
240 nn= VAL(a$):IF nn=0 THEN n=n ELSE GOTO 320
250 IF pr=1 THEN PRINT a$;
260 toon(tonen)=(a*2  )
270 SOUND 1,toon(tonen)
280 LOCATE 3,3: PRINT tonen
290 a=0
300 NEXT
310 GOTO 20
320 n=4-nn:GOTO 140
330 CLS
340 LOCATE 5,15: PRINT "melodie wordt gespeeld"
350 FOR n=1 TO tonen
360 SOUND 1,toon(n)
370 NEXT
380 GOTO 20
```

```
390 CLS
400 LOCATE 5,5:PRINT "M E L O D I E   L A D E N"
410 LOCATE 1,20
420 INPUT "welke melodie moet ik laden ";na$
430 OPENIN na$
440 INPUT#9, tonen
450 FOR n=1 TO tonen
460 INPUT#9, toon(n)
470 NEXT n
480 CLOSEIN
490 GOTO 20
500 CLS
510 LOCATE 5,5
520 INPUT"hoe moet de melodie heten ";na$
530 OPENOUT na$
540 PRINT#9,tonen
550 FOR n=1 TO tonen
560 PRINT#9,toon(n)
570 NEXT
580 CLOSEOUT
590 GOTO 20
```

Verklaring:

```
 20- 90  menu en keuze van modus (1-4)
100-310  programmeermodus
130      lus openen voor aantal tonen
140      toetsenbord uitlezen
150-230  gebruikte toetsen beoordelen
240      is octaaf veranderd?
250      gebruikte toetsen weergeven
260      waarde in actuele variabele van array zetten
270      toon produceren met geluidsinstructie
310      sprong terug in menu
320      octaven veranderen, sprong terug in
         programmeermodus
330-380  speelmodus
350      lus openen voor aantal tonen
360      actuele toon spelen
380      sprong terug in menu
390-480  melodie laden
420      invoerfile op cassette openen
430      aantal tonen lezen
440      lus openen om toon in te lezen
450      afzonderlijke tonen inlezen
470      invoerfile op cassette sluiten
480      sprong terug in hoofdprogramma
490-580  melodie opslaan
```

```
510       naam van melodie invoeren
520       uitvoerfile op cassette openen
530       aantal tonen op band zetten
540       lus openen om tonen op band te zetten
550       afzonderlijke tonen op band zetten
570       sluiten van uitvoerfile op cassette
580       sprong terug naar hoofdmenu
```

2.7 Voorbeeldprogramma's

zi

Met de volgende kleine demo's (demonstratieprogramma's) voorziet u eigen programma's van de juiste toon. Deze voorbeelden komen ook van pas in de beide spelletjes. De demo's kunnen natuurlijk worden aangepast. Ze hebben een titel die met het geluid overeenkomt. De regelnummers zijn niet bindend; u kunt ze veranderen, mits dat met de sprongadressen ook gebeurt. De BASIC-instructie RENUM doet dat automatisch. Misschien kunt u de demo's met de omhullende curve verbeteren of een grafiek van de toon maken met eigengemaakte tekens. Laat uw fantasie de vrije loop.

```
10 REM politiesirene
20 FOR n=100 TO 200 STEP 10
30 SOUND 1,n,2
40 NEXT
50 FOR n=200 TO 100 STEP -10
60 SOUND 1,n,2
70 NEXT
80 GOTO 20
```

```
10 REM in gesprek-toon
20 SOUND 1,100,100,15
30 SOUND 1,0,100
40 GOTO 20
```

```
10 REM Star Wars
20 FOR n=90 TO 125 STEP INT(RND(1)*10)+1
30 SOUND 1,n,2,15
40 NEXT
50 SOUND 1,0,INT (RND(1)*20)
60 GOTO 20
```

```
10 REM toonladder
20 FOR n=127 TO 450 STEP 12
30 SOUND 1,n
40 NEXT n
50 FOR n=450 TO 127 STEP -12
60 SOUND 1,n
70 NEXT n
80 GOTO 20
```

```
10 REM explosie
20 FOR n=15 TO 1 STEP -1
30 SOUND 1,426,40,n,,,1
40 NEXT
```

```
10 REM alarm
20 FOR n=500 TO 100 STEP -15
30 SOUND 1,n,4
40 NEXT
50 SOUND 1,0,30
60 GOTO 20
```

```
10 REM treffer op afstand
20 FOR n=100 TO 800 STEP 15
30 SOUND 1,n,2,15
40 NEXT
50 FOR x=1 TO 7 STEP 0.5
60 SOUND 1,400,5,x,,,1
70 NEXT
80 SOUND 1,0,50
90 GOTO 20
```

```
10 REM CPC's favoriete melodie
20 a=INT (RND(1)*3500)+284
30 SOUND 1,a
40 b=INT(RND(1)*3400)+284
50 SOUND 2,b
60 c=INT (RND(1)*3300)+284
70 SOUND 4,c
80 GOTO 20
```

Het volgende programma heeft in principe dezelfde opbouw als "Oh When The Saints". Het speelt de melodie "Happy Birthday" en is alleen voorzien van tonen en toonlengtes. Verbeter de klank met ENV en ENT.

```
10 REM Happy Birthday
20 READ a,b
30 IF b=-1 THEN END
```

```
40 SOUND 1,a,b
50 GOTO 20
60 DATA  319, 50,319, 50,284,100,319,100,239,100
70 DATA  253,150,319, 50,319, 50,284,100,319,100
80 DATA  213,100,239,150,319, 50,319, 50,159,100
90 DATA  190,100,239,100,253,100,284,100,179, 50
100 DATA 179, 50,190,100,239,100,213,100,239,150
110 DATA 0,-1
```

verklaring:

```
10      titel
20      tonen of toonwaarden lezen
30      compositie afgelopen?
40      toon A spelen met toonlengte B
50      sprong terug naar 20
60-110  gegevens voor tonen en toonlengtes
```

2.8 Muziek is altijd goeiemorgen

Als u moeilijk wakker wordt, is dit programma een uitkomst. Het gaat om een digitale klok met een wekker. Bovendien klinkt er elke seconde een zachte toon en elke minuut een iets hardere. Op de hele uren hoort u een korte melodie. Met enige veranderingen maakt u van deze eenvoudige klok zelfs een tweede Big Ben. U kunt de wekker een zelfgekozen melodie laten spelen.

Gebruiksaanwijzing: na de start van het programma voert u de actuele tijd in de volgorde uur, minuut, seconde in. Elke invoer sluit u af met ENTER. Vervolgens vraagt het programma naar de tijd waarop u wilt worden gewekt. De invoer verloopt weer op dezelfde wijze. Als u alleen op ENTER drukt, is de wektijd 0.00 uur, dus twaalf uur 's nachts. Een druk op een willekeurige toets verlost u van de wektoon. De klok loopt dan normaal door. Loopt de klok achter, verlaag dan de stapgrootte van de EVERY-instructie in regel 150. Verhoog de waarde als de klok voorloopt.

```
10 REM soundwekker
20 MODE 2
30 INPUT "uren     ";z
40 INPUT "minuten  ";y
50 INPUT "seconden ";x
```

```
60 INPUT "wektijd  (u)";z2
70 INPUT "wektijd  (m)";y2
80 INPUT "wektijd  (s)";x2
90 MODE 0
100 LOCATE 3,9:PRINT CHR$(150);STRING$(&D ,CHR$(154));CHR$(156)
110 LOCATE 3,10:PRINT CHR$(149):LOCATE 17,10:PRINT CHR$(149)
120 LOCATE 3,11:PRINT CHR$(149):LOCATE 17,11:PRINT CHR$(149)
130 LOCATE 3,12:PRINT CHR$(149):LOCATE 17,12:PRINT CHR$(149)
140 LOCATE 3,13:PRINT CHR$(147);STRING$(&D ,CHR$(154));CHR$(153
150 LOCATE 5,5:PRINT "muziekwekker"
160 EVERY 50,2 GOSUB 180
170 GOTO 170
180 SOUND 1,284,5,5:x=x+1
190 IF x=60 THEN x=0:y=y+1:SOUND 2,71,5,6
200 IF y=60 THEN y=0:z=z+1:GOSUB 270
205 IF z=z2 AND y=y2 AND x=x2 THEN GOSUB 320
210 LOCATE 5,11:PRINT z
220 LOCATE 8,11:PRINT ":"
230 LOCATE 9,11:PRINT y
240 LOCATE 12,11:PRINT ":"
250 LOCATE 13,11:PRINT x
260 RETURN
270 READ no
280 IF no=-1 THEN RETURN
290 SOUND 4,no,30,7
300 GOTO 270
310 DATA 253,253,169,169,150,150,169,190,190,201,201,225,225,
253,169,169,190,190,201,201,225,169,169,190,201,201
315 DATA 225,253,253,169,169,150,150,169,190,190,201,201,225,
225,223,-1
320 SOUND 7,100,1000,15
330 IF INKEY$="" THEN 330
340 SOUND 135,0
350 RETURN
```

verklaring:

10- 80	op beeldschermmodus zetten, invoer van de verlangde waarden: uurtijd (h,m,s), wektijd (h,m,s)
90-140	kader van de klok tekenen
150	met de EVERY-instructie wordt elke seconde de subroutine opgeroepen die begint bij 170
170	secondetoon, bij secondentotaal wordt 1 opgeteld
180	minuut voorbij? Ja, dan minuuttoon
190	uur voorbij? Ja, dan naar subroutine (270 e.v.)
200	tijd om te wekken? Ja, dan naar subroutine (320 e.v.)
210-260	weergave van de tijd in cijfers

```
270       noten lezen
280       melodie afgelopen?
290       melodie spelen met alle drie de toonkanalen
310       dataregel met toonwaarden
320       wektoon
330       het programma wacht tot u een toets indrukt
340       sound-instructie onderbreekt wektoon; er wordt een
          stop naar de toonkanalen gestuurd
```

3 MACHINETAAL

3.1 Inleiding

Dit hoofdstuk behandelt zaken waar een wat gevorderde programmeur van op de hoogte moet zijn. Zoals u weet zijn er enkele problemen bij het programmeren in BASIC die alleen met kleine machinetaalroutines kunnen wordēn opgelost. Het uitvoeren van een sorteer- of zoekroutine duurt in BASIC soms minuten of zelfs uren. In machinetaal geschreven gebruikt zo'n programma slechts een honderdste van de tijd die het BASIC-programma nodig heeft. Het brein van de CPC 464 is de Z80, een van de meest gangbare 8-bits processors in de homecomputertechniek. Deze chip kent meer dan 600 instructies. De CPC begrijpt alleen binaire getallen, dat zijn cijferreeksen waarin alleen nullen en enen voorkomen. Door ze in een bepaalde volgorde te zetten, kunnen we elk getal weergeven. Elke 0 of 1 van een binair getal heet BIT (BInary digiT = binair cijfer). Bits komen meestal voor in een groep van 8, BYTE genoemd. Een voorbeeld: 01010101 staat voor 85 in het decimale stelsel. Dat zuigen we niet uit onze duim, reken maar eens mee. De meest rechtse bit staat voor 2 tot de macht 0, de bit links daarvan voor 2 tot de macht 1, de volgende voor 2 tot de macht 2 enzovoort. De meest linkse bit komt dus overeen met 2 tot de macht 7. Vervolgens vermenigvuldigen we elke waarde met het binaire cijfer van die positie en tellen de uitkomsten bij elkaar op. De som bedraagt 85. Netjes onder elkaar gezet ziet dat er zo uit:

```
1 * 2^0 =  1
0 * 2^1 =  0
1 * 2^2 =  4
0 * 2^3 =  0
1 * 2^4 = 16
0 * 2^5 =  0
1 * 2^6 = 64
0 * 2^7 =  0 +
          --
          85
```

De algemene vorm van een byte is als volgt:

```
b7 b6 b5 b4 b3 b2 b1 b0
```

b0 is de meest rechtse bit en b7 de meest linkse. De formule waar het decimale getal uitrolt, luidt dus:

```
b7 * 2^7 + b6 * 2^6 + b5 * 2^5 + b4 * 2^4 +
b3 * 2^3 + b2 * 2^2 + b1 * 2^1 + b0 * 2^0
```

Om het rekenen te vergemakkelijken hier de uitkomsten van 2^0 tot en met 2^7:

```
2^0=1  2^1=2  2^2=4  2^3=8  2^4=16  2^5=32  2^6=64  2^7=128
```

De cijfers 0-7 hebben dus te maken met het tot de zoveelste macht verheven cijfer 2. Met PRINT &X bin maakt u decimale getallen van binaire. PRINT BIN$(dec) doet het omgekeerde:

```
PRINT &X01010101
```

levert dus het decimale getal 85 op en

```
PRINT BIN$(85)
```

het binaire getal 01010101. Let wel: de nullen voor de eerste 1 van links worden meestal weggelaten.

3.2 Het hexadecimale of zestientallige stelsel

Het hexadecimale stelsel is de meest gangbare manier om computers te programmeren. Zestien cijfers en letters staan voor de decimale waarden 0-15.

hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

De minimale lengte van uit hexadecimale getallen opgebouwde machinetaalprogramma's en gegevens betekent een enorme tijdwinst bij de invoer. De hexadecimale code FF is aanmerkelijk korter dan de binaire code 11111111. Beide staan ze voor het decimale getal 255. Een byte (acht bits) kunt u dankzij het zestientallig stelsel met twee tekens weergeven. Een hexadecimaal getal van 1 byte verandert u als volgt in een decimaal getal. Vermenigvuldig het eerste deel van het getal met 16 en tel het tweede deel erbij op. A1 is hetzelfde als 161: A=10, dus 16xA=160; 160+1=161. Getallen van twee bytes (16 bits) splitst u in tweeën: highbyte en

lowbyte. Bij een getal als A1E1 is de linker byte (A1) de highbyte en de rechter de lowbyte (E1). Bereken de bytes op de manier die we net uit de doeken hebben gedaan, vermenigvuldig de highbyte met 256 en tel de lowbyte erbij op: highbyte=161, lowbyte=225; 161x256=41216; 41216+225=41441. De CPC 464 heeft twee functies om u het omrekenen te besparen. Voor hexadecimaal naar decimaal is dat

```
PRINT &hex
```

waarbij hex voor het hexadecimaal getal staat.

```
PRINT HEX$(dec)
```

doet het omgekeerde. U kunt bij de eerste functie alleen hexadecimale getallen tot en met 7FFF omrekenen. Bij grotere waarden wordt 65536 van het resultaat afgetrokken. Dat levert een negatief getal op. Bijvoorbeeld:

```
PRINT &A1E1
```

U verwacht het resultaat 41441, maar op het scherm verschijnt -24095 (41441-65536=). Zoiets is natuurlijk makkelijk te verhelpen:

```
10 INPUT A$:B$="&"+A$:IF VAL(B$)<0 THEN PRINT
   VAL(B$)+65536:END
20 PRINT VAL(B$)
```

We hebben een stringvariabele gebruikt, zodat het hexadecimale getal niet steeds door & vooraf hoeft te worden gegaan. In de volgende hoofdstukken komt u de begrippen highbyte en lowbyte nog vaak tegen, want geheugenplaatsen ofwel adressen zijn in machinetaal steeds in high- en lowbyte weergegeven. Het hoogste adres is 65535, want de highbyte kan als hoogste waarde alleen 255 of FF (hexadecimaal) zijn. Volgens de formule HIGHBYTE*256+LOWBYTE is de hoogste waarde 65535. Tot slot een tabel met de getallen van 0-16 in alle drie de stelsels.

dec	hex	bin
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

Met deze informatie moet u in staat zijn verdere uiteenzettingen te volgen.

3.3 Programmeertechnieken

U kunt verschillende methodes gebruiken om in BASIC machinetaal te programmeren. De meest tijdrovende methode is de codes voor de instructies in de geheugenplaatsen (adressen) te POKEn. Met een machinetaalmonitor gaat het veel handiger. Aan het eind van deze paragraaf staat een provisorische machinetaalmonitor om het programmeren van machineprogramma's te vergemakkelijken. Het enige wat deze monitor doet is waarden in adressen POKEn, maar dat scheelt toch aanzienlijk in het typwerk. POKE-instructies werken als volgt: POKE X,A zet de waarde A in adres X. In machinetaal hebben deze instructies een bepaald adres, vergelijkbaar met de regelnummers van een BASIC-programma. Wilt u een programma beginnen op adres &2000, POKE dan de waarde voor de eerste instructie in &2000 en de waarden die bij de instructie horen in &2001 en &2002. Als de instructie slechts 1 byte beslaat, staat op geheugenplaats &2001 al de tweede instructie. Hoe zit het nu met de codes die met de instructies overeenkomen? De computer begrijpt alleen binaire getallen. Daarom bestaat een instructie daar ook uit. Bij elke instructie hoort een aantal van dergelijke getallen. Zo krijgt u een lijst van getallen, die samen het programma vormen. Met de instructie POKE schrijft u ze in de juiste volgorde in de adressen. De instructie PEEK (X) is nodig om een geheugenplaats uit te lezen, dus de waarde te achterhalen. Een geschreven programma start u met de instructie CALL X. Deze roept het programma op dat op adres X begint.

3.3.1 Assembler

De assembler is een programmeertaal die afkortingen voor de eigenlijke machinetaalinstructies gebruikt, MNEMONICS genaamd. De afkorting voor de instructie ACCUMULATOR LADEN is bijvoorbeeld LDA. Omdat de machine deze afkorting niet begrijpt, heeft u een programma nodig om de instructies in binaire getallen te vertalen. Zo'n programma noemen we een assembler. U kunt bovendien de belangrijke adressen namen (zogenaamde labels) geven. Labels gebruiken we verderop om adressen van variabelen aan te duiden. Een assembler voor de CPC is inmiddels verkrijgbaar. Heeft u geen assemblerprogramma, dan moet u steeds de waarde van de instructie opzoeken in de tabel in de bijlage.

3.3.2 Registers en eerste instructies

Eerst de opbouw van de registers van de CPU (het brein van de computer):

Register A, meestal accumulator genoemd, dient voornamelijk voor opslag en verwerking van actuele waarden, dat zijn gegevens die in behandeling zijn.

Register BC: BC betekent bytecounter. Dit register wordt vaak voor telopdrachten gebruikt. Het gaat hier om een dubbelregister of registerpaar; register B en C kunnen afzonderlijk worden geadresseerd. In tegenstelling tot A, dat slechts een enkele byte kan opnemen, is er in register BC plaats voor twee bytes. Er kunnen dus getallen in worden verwerkt tot en met 65535.

Register HL: HL staat voor HIGH LOW. U kunt elk register afzonderlijk aanspreken, maar het is beter ze allebei gebruiken. De CPU (processor) heeft dit register namelijk nodig om adressen op te slaan. Die worden als 2 bytes (highbyte en lowbyte) weergegeven.

Register DE: dit registerpaar (apart inzetbaar) is een hulpregister om 16-bits getallen op te slaan die in de andere registerparen niet zinvol zijn. Het is ook te gebruiken als de andere registerparen vol zijn, of als een instructie drie registerparen nodig heeft.

Met de simpele instructie ld (laad) brengt u een waarde uit het geheugen in een register of verplaatst u het van het ene naar het andere. De instructie ld is de belangrijkste die de Z80 kent. De algemene formule om een waarde (w) in een register (x) te zetten is:

```
ld x,w
```

Een praktijkvoorbeeld:

```
ld a,&FF
```

&FF komt overeen met het decimale getal 255. In ons voorbeeld is w een directe invoer van 1 byte.

```
ld x,ww
```

is eveneens een directe invoer, maar hier gaat het om twee bytes, een highbyte en een lowbyte.

```
ld x,(ww)
```

wil zeggen dat het register wordt geladen met de inhoud van adres ww. Een voorbeeld moge dit illustreren:

```
ld hl,(&A000)
```

Het registerpaar HL wordt in dit geval geladen met de waarde uit de 2-bytes geheugenplaats &A000. De inhoud van register A overbrengen naar C gebeurt op de volgende wijze:

```
ld c,a
```

De inhoud van C gaat verloren, niet die van A. Een machinetaalprogramma moet op een veilige plaats in het geheugen worden opgeslagen. Dat wil zeggen niet in een gedeelte dat is gereserveerd voor een BASIC-programma of de variabelen daarvan. De volgende instructiereeks maakt ruimte vrij in het RAM. Voer direct in:

```
MEMORY &1FFF:PRINT HIMEM:NEW
```

Als u zelf bedachte tekens heeft geprogrammeerd, moet u ze eerst wissen door de computer uit te schakelen. Anders treden er complicaties op. Nu heeft u een vrije ruimte van meer dan 34 Kbyte voor uw machinetaalprogramma, maar er is nog maar 7.5 Kbyte over voor BASIC. De reeks POKE &2000,&3E: POKE&2001,&FF: POKE &2002,&C9 heeft tot gevolg dat de instructies in de

geheugenplaatsen worden geschreven. De drie waarden na POKE zijn de adressen waarin de instructies komen te staan. Ze hebben dezelfde functie als de regelnummers in BASIC. De waarde &3E is de instructie ld x,w: w is &FF. De waarde &C9 is de code voor RET (=RETURN), dus een sprong terug van de routine naar BASIC. Start het programma met CALL (&2000). Met CALL roept u machinetaalroutines op. Daarna meldt de computer zich met READY. U kunt niet zien wat er gebeurd is, maar toch bevat de accumulator inmiddels het hexadecimale getal &FF. We gaan nu &A1F1 in register BC zetten, dus ld bc,&A1F1. Voer achtereenvolgens in: POKE &2000,&1 (dat betekent ld bc,ww), POKE &2001,&F1 (lowbyte van het getal), POKE &2002,&A1 (highbyte van het getal), POKE &2003,&C9 (RETURN). Start het programma met CALL &2000. Weer merkt u niet wat er gebeurt, maar &A1F1 is wel de inhoud geworden van register BC. Zonder daar ruchtbaarheid aan te geven hebben we een gouden regel in praktijk gebracht: na een instructie komt bij een 16-bits operand altijd eerst de lowbyte en dan pas de highbyte.

We gaan ons nu wijden aan een optelprogramma voor getallen die een omvang hebben van 1 byte. We nemen twee getallen. Zet het eerste in geheugenplaats &3000, het tweede in &3002 en de uitkomst in &3004. Om de getallen op te tellen, zetten we het het eerste getal in de accumulator en het adres van het tweede in register HL. We laden register HL niet met het tweede getal (de haakjes ontbreken) om de programmeermogelijkheden te laten zien:

```
ld a,(&3000), ld hl,&3002
```

Optellen doet u met add a,(hl). Deze instructie zorgt ervoor dat de inhoud van de geheugenplaats, die door register HL is geadresseerd, wordt opgeteld bij de inhoud van de accumulator. Tenslotte slaan we met ld (&3004),a de uitkomst op in adres &3004. Hieronder het programma met achter de assemblerinstructies de hexadecimale waarden die u moet inPOKEn:

```
ld a,(&3000)    &3A &00 &30
ld hl,&3002     &21 &02 &30
add a,(hl)      &86
ld (&3004),a    &32 &04 &30
RET             &C9
```

In BASIC ziet dit programma er zo uit:

```
10 FOR N=&2000 TO &200A
20 INPUT WAARDE
30 POKE N,WAARDE
40 NEXT
```

Na de start van deze routine voert u de waarden in. U hoeft alleen nog twee getallen (tot maximaal 127) in te voeren met

```
POKE &3000, getal 1: POKE &3002, getal 2
```

Start het machinetaalprogramma met CALL &2000. Na READY kunt u de uitkomst zichtbaar maken met PEEK (&3004). De getallen mogen niet groter zijn dan 127 omdat anders het resultaat groter wordt dan 255. 1 byte is dan niet meer genoeg. 127+64=191 is in binaire getallen:

```
01111111
01000000 +
--------
10111111
```

Binaire getallen optellen is niet moeilijk. Is een bit gezet (1) en de overeenkomstige bit in het tweede getal niet (0), dan is het resultaat een gezette bit (1). Zijn beide bits gezet (1), dan wordt het cijfer onder de streep in dezelfde kolom een 0, maar er verschijnt 1 plaats naar links een gezette bit. We zeggen dat de bit wordt overgedragen (CARRY). Het voorbeeld demonstreert dit bij bit 6 en 7. Vindt de overdracht plaats in de meest linkse bit (bit 7) dan wordt deze bij een 8-bits operatie niet verwerkt. Bij getallen groter dan 127 zou zo'n overdracht moeten plaatsvinden. Een 16-bits optelling is in principe hetzelfde als een 8-bits optelling. Wel moeten er meer geheugenplaatsen worden benoemd en twee nieuwe instructies geïntroduceerd. Met het eerste getal zijn de adressen &3001 en &3000 gemoeid. Het tweede getal beslaat &3003 en &3002. De som komt in &3005 en &3004 te staan. Nu het programma: eerst telt het programma de beide lowbytes van de getallen op en slaat de uitkomst op in &3005. Bij een overdracht wordt de carryflag gezet. In het tweede deel telt het programma de highbytes op en verwerkt een eventuele overdracht.

```
ld a,(&3001)  laad lowbyte van getal 1 in accumulator
ld hl,&3003   laad adres van lowbyte van getal 2 in HL
add a,(hl)    tel lowbytes op
ld (&3005),a  sla lowbyte van resultaat op in &3005
ld a,(&3000)  laad highbyte van getal 1 in accumulator
dec hl        trek van HL 1 af
adc a,(hl)    tel highbytes en overdracht op
ld (&3004),a  sla highbyte van resultaat op in &3004
```

De eerste nieuwe instructie, dec hl, doet hetzelfde als ld hl,&3000, want omdat u van HL 1 aftrekt, staat daar &3000 in plaats van &3001. Het voordeel van dec hl is, dat de instructie slechts 1 byte lang en dus sneller is. De tweede nieuwe instructie, adc a,(hl), doet hetzelfde als add a,(hl), alleen

wordt nu de overdracht uit het eerste deel verwerkt. Het hele programma met de hexadecimale waarden die u moet invoeren luidt:

```
ld a,(&3001)  &3A &1 &30
ld hl,&3003   &21 &3 &30
add a,(hl)    &86
ld (&3005),a  &32 &5 &30
ld a,(&3000)  &3A 00 &30
dec hl        &2B
adc a,(hl)    &8E
ld (&3004),a  &32 &4 &30
RET           &C9
```

U heeft de BASIC-lus weer nodig om dit programma in te voeren. Verander regel 10 in 10 FOR N=&2000 TO &2012. Zet nu het eerste getal in de geheugenplaatsen &3000 (highbyte) en &3001 (lowbyte). Het tweede getal zet u in &3002 en &3003. Aangenomen dat getal 1 &5F4C is en getal 2 &8B5A, dan typt u in:

```
POKE &3000,&5F:POKE &3001,&4C:POKE &3002,&8B:POKE &3003,&5A
```

Start het machinetaalprogramma daarna met CALL(&2000). Na READY voert u in:

```
PRINT PEEK(&3004)*256+PEEK(&3005)
```

Op het scherm verschijnt 60070, de som van 24396 (&5F4C) en 35674 (&8B5A). Bij de optelling van twee 16-bits getallen waarvan de lowbytes samen groter zijn dan &FF ontstaat een overdracht. De computer slaat die op in de carryflag, een bit van register F. Dit register bevat alle belangrijke informatie die de CPU nodig heeft, bijvoorbeeld of er een overdracht heeft plaatsgevonden, of register B gelijk is aan 0, enzovoort. De computer verrekent een overdracht in bit 0 van de highbyte van het resultaat:

```
highbyte            lowbyte

00010000 (&10)      10000000 (&80)
00001000 (&08) +    10000001 (&81) +
--------------      --------------
00011000 (&18)      00000001 (&01)
```

Het resultaat van deze berekening zonder overdracht ligt 256 te laag. Om met de gezette carryflag rekening te houden verhoogt u de highbyte met 1. &1801 verandert dan in &1901. Dat is juist. De carryflag moet bij een 16-bits berekening dus altijd worden meegerekend. In het programma is dat gebeurd dankzij de instructie adc a,(hl).

Over naar iets heel anders. Met de instructie PLOT x,y kunt u punten op het beeldscherm zetten, maar u kunt ze ook in het beeldschermgeheugen POKEn. Dat loopt van &C000 tot &FFFF. Typ het volgende programma in:

```
10 MODE 2
20 FOR N= &C000 TO &FFFF
30 POKE N,&FF
40 NEXT
```

Na de start van het programma wordt het beeldscherm om de 8 regels gevuld. De punten hebben de hoogte van een regel. Dat houdt verband met de opbouw van de tekens. Om het volgende machinetaalprogramma goed te doorgronden, dient u zich ervan bewust te zijn dat de lus ook deze vorm kan hebben:

```
10 N=N+1
20 IF N=&FFFF THEN END
```

De lus heeft natuurlijk ook een equivalent in machinetaal. Dat wordt verderop wel duidelijk. Om u niet te veel te belasten met nieuwe instructies, worden alleen de eerste drie regels gevuld. In een adres moet &FF komen te staan. Dat doen we met ld. Daarna zijn de volgende aan de beurt. Herhaling van de instructie ld kunt u voorkomen door een lus te schrijven. Register B is daarvoor geknipt. We gebruiken de volgende variabelen:

```
lengte  lengte van het te vullen bereik
adr 1   beginadres van dit bereik
```

Eerst definiëren we de lus met de instructie ld b, lengte.. Vervolgens laden we &FF in de accumulator en daarna adr 1 in register HL. In machinetaal moeten we dan een wijzer op het startadres zetten. We laden de inhoud van de accu (&FF) in het adres van register HL, verhogen HL met 1 zodat het register op het volgende adres wijst en trekken van register B 1 af. Er moeten namelijk net zoveel punten worden gezet als register B aangeeft. De instructie inc hl verzorgt de optelling en dec b telt terug. De volgende instructie is een voorwaardelijke sprong, vergelijkbaar met IF...THEN GOTO... in BASIC. Wanneer aan de voorwaarde voldaan is, volgt een sprong. Die moet tot de instructie ld (hl),a leiden omdat anders steeds alleen de eerste geheugenplaats wordt geladen. De voorwaarde voor de sprong is NIET NUL, want er moet net zolang worden gesprongen tot register B niet gelijk is aan nul. De instructie is jp nz, adres; nz betekent NOT ZERO. Tot slot zorgt RET ervoor dat we in BASIC terugkeren. Het programma:

```
ld b,&C0        &6 &C0
ld a,0          &3E &FF
ld hl,&C000     &21 &0 &C0
ld (hl),a       &77
inc hl          &23
dec b           &5
jp nz,&2007     &C2 &7 &20
RET             &C9
```

Voer de waarden in een lus in die loopt van &2000-&200D. Voordat u het programma laadt, schakelt u de computer met MODE 2 nog een keer in de 80-tekens modus. Om complicaties te voorkomen doet u dit ook als de 80-tekens modus al actief is. Start met CALL &2000 en het programma tekent 3 regels. De spronginstructie (jp) springt naar het adres waarop ld (hl),a staat. Register PC (program counter) speelt hier een rol. Dit register bevat het adres van de uit te voeren instructie. Als u in PC een nieuw adres zet, voert de computer de instructie uit met het ingevoerde adres. Om een sprong te berekenen kunt u ook de waarde die nodig is om het betreffende adres te bereiken bij de program counter optellen of ervan aftrekken. Deze manier van springen heet relatief omdat de computer niet het werkelijke, absolute adres in PC laadt. De instructie jr e is de relatieve sprong. De factor e wordt bij de program counter opgeteld of afgetrokken. In werkelijkheid is de berekening van de sprong nog iets ingewikkelder, maar voor een inleiding is dit verhaal ruimschoots voldoende.

3.3.3 Tekens en instructieset

Deze paragraaf bestaat uit een tabel met tekens en mnemonics plus hun hexadecimale en decimale code. Dit alles natuurlijk voor de CPC 464.

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
0	Null	00	nop	rlc b	
1	SOH	01	ld bc,NN	rlc c	
2	STX	02	ld (bc),a	rlc d	
3	ETX	03	inc bc	rlc e	
4	EOT	04	inc b	rlc h	
5	ENQ	05	dec b	rlc l	
6	ACK	06	ld b,N	rlc (hl)	
7	BEL	07	rlca	rlc a	
8	BS	08	ex af,af"	rrc b	
9	HT	09	add hl,bc	rrc c	
10	LF	0A	ld a,(bc)	rrc d	
11	VT	0B	dec bc	rrc e	
12	FF	0C	inc c	rrc h	
13	CR	0D	dec c	rrc l	
14	SO	0E	ld c,N	rrc (hl)	
15	SI	0F	rrca	rrc a	
16	DLE	10	djnz DIS	rl b	
17	DC 1	11	ld de,NN	rl c	
18	DC 2	12	ld (de),a	rl d	
19	DC 3	13	inc de	rl e	
20	DC 4	14	inc d	rl h	
21	NAK	15	dec d	rl l	
22	SYN	16	ld d,N	rl (hl)	
23	ETB	17	rla	rl a	
24	CAN	18	jr DIS	rr b	
25	EM	19	add hl,de	rr c	
26	SUB	1A	ld a,(de)	rr d	
27	ESC	1B	dec de	rr e	
28	FS	1C	inc e	rr h	
29	GS	1D	dec e	rr l	
30	RS	1E	ld e,N	rr (hl)	
31	US	1F	rra	rr a	
32	SP	20	jr nz,DIS	sla b	

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld,h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44		2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
65	A	41	ld b,c	bit 0,c	out (c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld(NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	retn
70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	
72	H	48	ld c,b	bit 1,b	inc c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c.l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	
80	P	50	ld d,b	bit 2,b	in d,(d)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 2,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	[	5B	ld e,e	bit 3,e	ld de,(NN)
92	\	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95	-	5F	ld e,a	bit 3,a	
96		60	ld h,b	bit 4,b	in h,(c)

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld de,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld (hl),b	bit 6,b	
113	q	71	ld (hl),c	bit 6,c	sbc hl,sp
114	r	72	ld (hl),d	bit 6,d	ld (NN),sp
115	s	73	ld (hl),e	bit 6,e	
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123		7B	ld a,e	bit 7,e	ld sp,(NN)
124	G	7C	ld a,h	bit 7,h	
125	R	7D	ld a,l	bit 7,l	
126	A	7E	ld a,(hl)	bit 7,(hl)	
127	F	7F	ld a,a	bit 7,a	
128	I	80	add a,b	res 0,b	
129	S	81	add a,c	res 0,c	

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
130	C	82	add a,d	res 0,d	
131	H	83	add a,e	res 0,e	
132	E	84	add a,h	res 0,h	
133		85	add a,l	res 0,l	
134	T	86	add a,(hl)	res 0,(hl)	
135	E	87	add a,a	res 0,a	
136	K	88	adc a,b	res 1,b	
137	E	89	adc a,c	res 1,c	
138	N	8A	adc a,d	res 1,d	
139	S	8B	adc a,e	res 1,e	
140		8C	adc a,h	res 1,h	
141	E	8D	adc a,l	res 1,l	
142	N	8E	adc a,(hl)	res 1,(hl)	
143		8F	adc a,a	res 1,a	
144	T	90	sub b	res 2,b	
145	O	91	sub c	res 2,c	
146	K	92	sub d	res 2,d	
147	E	93	sub e	res 2,e	
148	N	94	sub h	res 2,h	
149	S	95	sub l	res 2,l	
150		96	sub (hl)	res 2,(hl)	
151		97	sub a	res 2,a	
152		98	sbc a,b	res 3;b	
153		99	sbc a,c	res 3,c	
154		9A	sbc a,d	res 3,d	
155		9B	sbc a,e	res 3,e	
156		9C	sbc a,h	res 3,h	
157		9D	sbc a,l	res 3,l	
158		9E	sbc a,(hl)	res 3,(hl)	
159		9F	sbc a,a	res 3,a	
160		A0	and b	res 4,b	
161		A1	and c	res 4,c	cpi
162		A2	and d	res 4,d	ini

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
163		A3	and e	res 4,e	outi
164		A4	and h	res 4,h	
165		A5	and l	res 4,l	
166		A6	and (hl)	res 4,(hl)	
167		A7	and a	res 4,a	
168		A8	xor b	res 5,b	ldd
169		A9	xor c	res 5,c	cpd
170		AA	xor d	res 5,d	ind
171		AB	xor e	res 5,e	outd
172		AC	xor h	res 5,h	
173		AD	xor l	res 5,l	
174		AE	xor (hl)	res 5,(hl)	
175		AF	xor a	res 5,a	
176		B0	or b	res 6,b	ldir
177		B1	or c	res 6,c	cpir
178		B2	or d	res 6,d	inir
179		B3	or e	res 6,e	otir
180		B4	or h	res 6,h	
181		B5	or l	res 6,l	
182		B6	or (hl)	res 6,(hl)	
183		B7	or a	res 6,a	
184		B8	cp b	res 7,b	lddr
185		B9	cp c	res 7,c	cpdr
186		BA	cp d	res 7,d	indr
187		BB	cp e	res 7,e	otdr
188		BC	cp h	res 7,h	
189		BD	cp l	res 7,l	
190		BE	cp (hl)	res 7,(hl)	
191		BF	cp a	res 7,a	
192		C0	ret nz	set 0,b	
193		C1	pop bc	set 0,c	
194		C2	jp nz,NN	set 0,d	
195		C3	jp NN	set 0,e	

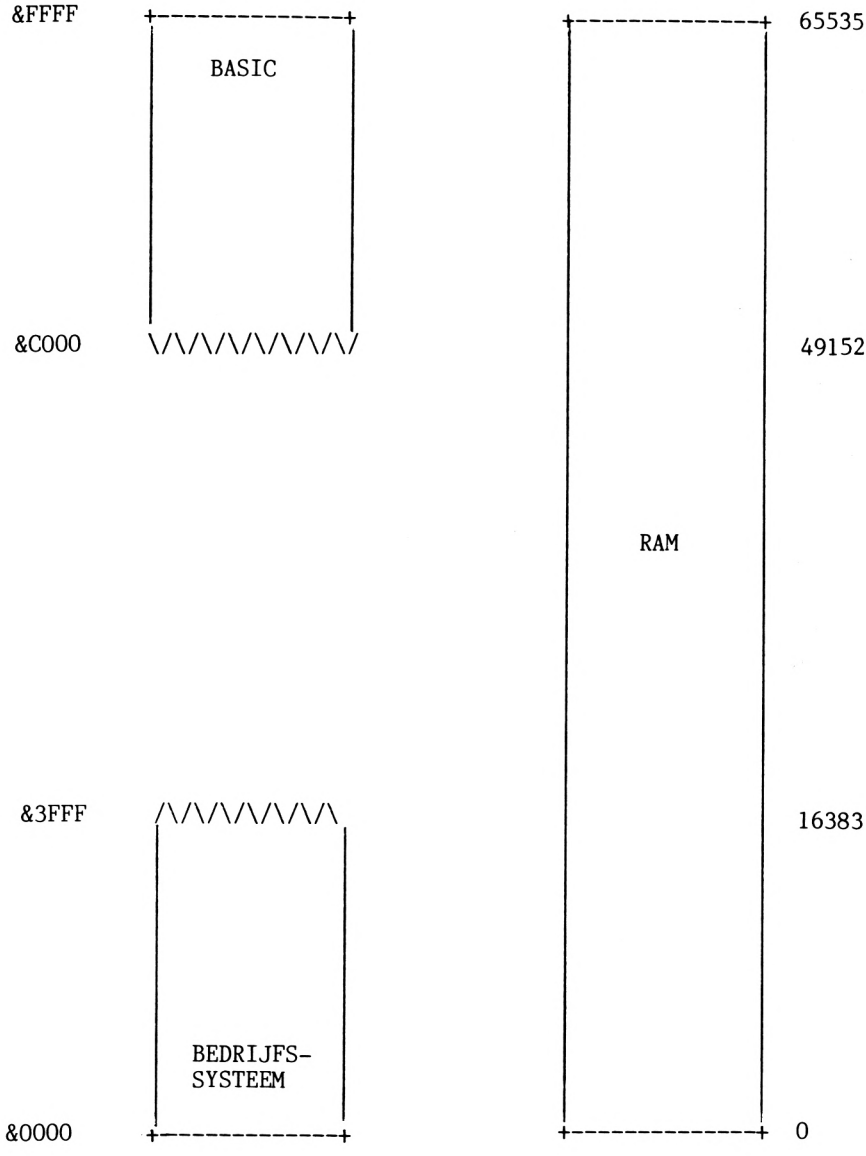
code	karakter	hex	Z80 assembler	naar CBh	naar EDh
196		C4	call nz,NN	set 0,h	
197		C5	push bc	set 0,l	
198		C6	add a,N	set 0,(hl)	
199		C7	rst 0	set 0,a	
200		C8	ret z	set 1,b	
201		C9	ret	set 1,c	
202		CA	jp z,NN	set 1,d	
203		CB		set 1,e	
204		CC	call z,NN	set 1,h	
205		CD	call NN	set 1,l	
206		CE	adc a,N	set 1,(hl)	
207		CF	rst 8	set 1,a	
208		D0	ret nc	set 2,b	
209		D1	pop de	set 2,c	
210		D2	jp nc,NN	set 2,d	
211		D3	out N,a	set 2,e	
212		D4	call nc,NN	set 2,h	
213		D5	push de	set 2,l	
214		D6	sub N	set 2,(hl)	
215		D7	rst 16	set 2,a	
216		D8	ret c	set 3,b	
217		D9	exx	set 3,c	
218		DA	jp c,NN	set 3,d	
219		DB	in a,N	set 3,e	
220		DC	call c,NN	set 3,h	
221		DD		set 3,l	
222		DE	sbc a,N	set 3,(hl)	
223		DF	rst 24	set 3,a	
224		E0	ret po	set 4,b	
225		E1	pop hl	set 4,c	
226		E2	jp po,NN	set 4,d	
227		E3	ex (sp),hl	set 4,e	
228		E4	call po,NN	set 4,h	

code	karakter	hex	Z80 assembler	naar CBh	naar EDh
229		E5	push hl	set 4,l	
230		E6	and N	set 4,(hl)	
231		E7	rst 32	set 4,a	
232		E8	ret pe	set 5,b	
233		E9	jp (hl)	set 5,c	
234		EA	jp pe,NN	set 5,d	
235		EB	ex de,hl	set 5,e	
236		EC	call pe,NN	set 5,h	
237		ED		set 5,l	
238		EE	xor N	set 5,(hl)	
239		EF	rst 40	set 5,a	
240		F0	ret p	set 6,b	
241		F1	pop af	set 6,c	
242		F2	jp p,NN	set 6,d	
243		F3	di	set 6,e	
244		F4	call p,NN	set 6,h	
245		F5	push af	set 6,l	
246		F6	or N	set 6,(hl)	
247		F7	rst 48	set 6,a	
248		F8	ret m	set 7,b	
249		F9	ld sp,hl	set 7,c	
250		FA	jp m,NN	set 7,d	
251		FB	ei	set 7,e	
252		FC	call m,NN	set 7,h	
253		FD		set 7,l	
254		FE	cp N	set 7,(hl)	
255		FF	rst 56	set 7,a	

3.4 Het geheugen van de CPC 464

Als u uw computer volledig wilt benutten, zult u iets over de interne opbouw moeten weten. De CPC heeft als processor een Z80. Dit is een 8-bits processor met 16 adreslijnen. Hiermee kunt u 65536 (2^16) geheugenplaatsen adresseren, die elk 8 bits breed zijn en 256 (2^8) verschillende waarden bevatten. De CPC bevat 64 Kbyte RAM en 32 Kbyte ROM. De hamvraag is natuurlijk hoe deze 96K met de processor aan te spreken als die maar 64K kan adresseren? Het antwoord luidt dat het adresbereik dubbel wordt bezet, een keer met RAM en daarboven nog eens met ROM. De processor bepaalt of hij het ROM of het RAM gebruikt. Om in het geheugen te schrijven kiest de processor natuurlijk het RAM, omdat het ROM (Read Only Memory) niet beschreven kan worden. De CPC heeft 1 ROM-bouwsteen die de complete 32 Kbyte bevat. De onderste 16 Kbyte vormen het bedrijfssysteem (0-16383(&3FFF)). De bovenste 16 Kbyte (&C000-&FFFF/49152-65535) bevatten de BASIC-interpreter. In schema:

Indeling van het geheugen



3.4.1 De plaats van BASIC en bedrijfssysteem

Het grootste gedeelte van het RAM-geheugen wordt gebruikt voor BASIC-programma's en -variabelen. Gewoonlijk ligt het bereik van &170 (368) tot &AB7F (43903) en is dus 43536 bytes groot. Als u in BASIC met PEEK het geheugen leest, krijgt u altijd de inhoud van het RAM. U kunt dus niet zonder meer de BASIC-interpreter of het bedrijfssysteem uitlezen. Met een monitorprogramma kan dat wel. We komen er in paragraaf 3.6 op terug. Het RAM is als volgt ingedeeld: de onderste 64 bytes van het geheugen zijn een kopie van het ROM. De kopie heeft hetzelfde adresbereik. In dit bereik staan de RESTART-routines die door een 1-bytes instructie kunnen worden opgeroepen. Deze acht RST-vectoren dienen bij de CPC 464 om routines in ROM en RAM op te roepen. Ze kiezen automatisch het juiste geheugen. Om deze routines zowel in het bedrijfssysteem (ROM) als in het RAM te kunnen gebruiken, bevat het RAM een kopie van de routines. Bedrijfssysteem en interpreter maken veelvuldig gebruik van RST-instructies. RST 3 passen we toe in het monitorprogramma in paragraaf 3.6.

3.5 De RST-instructies

RST 0 adres &0000

Deze vector zorgt voor een complete RESET. Er gebeurt hetzelfde als wanneer u de computer aanzet of tegelijkertijd CTRL SHIFT ESC indrukt.

```
0000  01 89 7F   LD  BC,&7F89
0003  ED 49      OUT (C),C
0005  C3 80 05   JP  &0580
```

Met de OUT-instructie wordt het ROM geselecteerd en vervolgens in het bedrijfssysteem de RESET-routine (beginadres: &0580) afgehandeld. U roept de RESET in BASIC op met CALL 0 en in machinetaal met CALL 0, JP 0 of RST 0.

RST 1 adres &0008

Deze instructie roept een routine op in het bedrijfssysteem of in het RAM met hetzelfde adresbereik. Het adres van de routine moet direct achter de RST-instructie staan. Voor het bereik van 0-&3FFF zijn 14 adresbits (0-13) voldoende, daarom dienen de twee hoogste bits voor de keuze van ROM of RAM:

```
bit 14 = 0  bedrijfssysteem gekozen
       = 1  RAM gekozen
bit 15 = 0  BASIC-ROM gekozen
       = 1  RAM gekozen
```

Bedrijfssysteemroutine &0826 kan als volgt worden opgeroepen:

```
RST 1
DW  &0826 + &8000
```

De gezette bit 15 kiest in het bereik &C000-&FFFF het RAM, terwijl de gewiste bit 14 het bedrijfssysteem adresseert. Op adres &0008 begint een sprong naar &B982.

RST 2 adres &0010

Deze RESTART-instructie roept een routine op in een uitbreidings-ROM. Het adres van de routine minus &C000 (relatief t.o.v. de start van het ROM) moet volgen op de instructie RTS 2. De hoogste twee bits dienen om vier verschillende ROMs uit te kiezen. Op adres &0010 begint een sprong naar &BA16.

RST 3 adres &0018

Met deze instructie roept u een routine in het ROM of RAM op. Achter de RST 3 instructie moet het adres van een parameterblok staan dat uit drie bytes bestaat. De eerste twee bytes bevatten het adres van de routine die wordt opgeroepen, de derde de ROM/RAM-status. Met de waarden 0-251 spreekt u het uitbreidings-ROM aan. De overige vier waarden hebben deze functies:

waarde	&0000-&3FFF	&C000-&FFFF
252	bedr.syst.	BASIC
253	bedr.syst.	RAM
254	RAM	BASIC
255	RAM	RAM

Op adres &0018 begint een sprong naar &B9BF.

RST 4 adres &0020

Deze instructie leest onafhankelijk van de ROM-status de RAM-inhoud van een machineprogramma. De instructie RST 4 vervangt de instructie ld a,(hl). HL moet dus het adres bevatten dat moet worden uitgelezen. Op adres &0020 begint een sprong naar &BACB.

RST 5 adres &0028

Hiermee springt u naar een routine in het bedrijfssysteem. Het adres volgt direct op de instructie. Op adres &0028 begint een sprong naar &BA2E.

RST 6 adres &0030

Het bedrijfssysteem gebruikt deze instructie niet. RST 6 kan bivoorbeeld worden gebruikt om een breakpoint in een debugger (=testprogramma) te zetten.

RST 7 adres &0038

De Z80 in de CPC 464 werkt in interruptmodus 1, daarom treedt bij een interrupt RST 7 op. Deze instructie verwerkt de interrupt. Op adres &0038 begint een sprong naar &B939.

3.6 Mini-monitor

Met dit monitorprogramma kunt u geheugenplaatsen adresseren, de inhoud veranderen en op band zetten. Met de RST-routines kunt u tevens het bedrijfssysteem en het BASIC-ROM aanspreken.

```
100 INK 0,1: INK 1,24: INK 2,23: INK 3,15,24: MODE 1
110 n=6: FOR i=0 TO n: READ cmd$(i): NEXT: DATA a,m,s,t,g,r,x
120 GOSUB 830
130 PEN 2: PRINT TAB (10) "M i n i - M o n i t o r" CHR$(13)
    TAB(10);: PEN 1: PRINT CHR$(22)CHR$(1)""""""""""""""""""""""""""""""
    PRINT:PRINT
140 PRINT:PRINT"I n s t r u c t i e:"CHR$(13)CHR$(22)CHR$(1)
    """""""""""""""""""""""""""""""CHR$(22)CHR$(0):PRINT
```

```
150 PRINT "m = geheugenbereik aangeven"
160 PRINT "a = geheugeninhoud veranderen"
170 PRINT "s = geheugenbereik opslaan"
180 PRINT "t = ASCII-tekst invoeren"
190 PRINT "g = machinetaalprogramma uitvoeren"
200 PRINT "r = ROM/RAM selecteren"
210 PRINT "x = terug naar BASIC"
220 PRINT
230 PRINT "Uitvoer op   ";
240 PEN 3:PRINT "B";: PEN 1:PRINT "eeldscherm":PRINT TAB(9) "of
    ";: PEN 3:PRINT "P";: PEN 1:PRINT"rinter ?";
250 WHILE a$<>"b" AND A$<>"B" AND A$<>"P" AND a$<>"p": a$=INKEY$:
    WEND
260 INK 3,24: rom=0
270 PRINT :IF a$="p" OR a$="P" THEN 1=16: c=8:GOTO 310
280 PRINT:PRINT:PRINT "beeldscherm-modus ? "
290 WHILE a$<>"1" AND a$<>"2": a$=INKEY$: WEND
300 a=VAL(a$): 1=a*8:c=0: IF a<>1 THEN MODE a
310 PRINT:PRINT "instructie ?";
320 a$=INKEY$: FOR i= 0 TO n: IF a$=cmd$(i) THEN PRINT: ON i+1
    GOSUB 490,540,590,700,750,780,940: GOTO 310
330 NEXT: GOTO 320
340 '
350 IF a>b THEN RETURN
360 PRINT#c, HEX$(a,4)"   ";
370 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
380 PRINT#c, HEX$(a,2)+ "  ";:a=a+1
390 NEXT: PRINT#c,  "  ";: a=a-1
400 IF c=0 THEN 450
410 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
420 x=x AND &7F
430 IF x<32 OR x=127 THEN x=46
440 PRINT#c, CHR$(x);: a=a+1: NEXT*PRINT#c: GOTO 350
450 IF 1=8 THEN PEN 2
460 FOR i=1 TO 1: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
470 PRINT CHR$(1)CHR$(x);:a=a+1: NEXT: PRINT: PEN 1:GOTO 350
480 '
490 INPUT "Adres ";x$: GOSUB 560: a=x
500 PRINT HEX$(a,4) ": ";
510 INPUT x$: a$=LEFT$(x$,1): IF a$<"0" OR a$>"9" AND a$<"a"
    OR a$>"f" THEN RETURN
520 GOSUB 560: POKE a,x: a=a+1: GOTO 500
530 '
540 GOSUB 650: GOSUB 350: RETURN
550 '
560 x=VAL("&"+x$): IF x<0 THEN x=x+2^16
570 RETURN
580 '
590 PRINT "opslaan:   ";
```

```
600 GOSUB 650
610 INPUT "naam ";a$
620 SAVE a$,b,a,b-a
630 RETURN
640 '
650 INPUT " van , tot ";a$,b$
660 x$=a$: GOSUB 560: a=x
670 x$=b$: GOSUB 560: b=x
680 RETURN
690 '
700 INPUT " adres ";x$: GOSUB 560
710 LINE INPUT "tekst invoeren:  ";a$
720 IF LEN(a$)>0 THEN FOR i=1 TO LEN(a$): POKE x-1+i,
    ASC(MID$(a$,i,1)): NEXT
730 RETURN
740 '
750 INPUT "adres ";x$
760 GOSUB 560: CALL x: RETURN
770 '
780 PRINT: INK 3,15,24
790 PRINT "select: ";: PEN 3: IF rom THEN PRINT "ROM";: ELSE
    PRINT"RAM";
800 PEN 1: a$= INKEY$: IF a$=" " THEN rom=1-rom: PRINT CHR$(13);:
    GOTO 790
810 IF a$=CHR$(13) THEN PRINT: INK 3,24: RETURN: ELSE 800
820 '
830 REM machinetaalprogramma
840 DATA &df,&04,&ab,&c9
850 DATA &07,&ab,&fc
860 DATA &3a,&00,&00,&32,&0e,&ab,&c9
870 FOR i=0 TO &D: READ a: POKE &AB00+1,a: NEXT: RETURN
880 '
890 REM ROM uitlezen
900 ah=INT(a/256): al=a-ah*256
910 POKE &AB08,al:POKE &AB09,ah
920 CALL &AB00
930 x=PEEK (&AB0E): RETURN
940 '
950 CLS:END
```

Programmabeschrijving van de mini-monitor:

100 Bovengrens van BASIC-geheugen verlagen, kleurindeling en 40-tekens modus kiezen.

110 Instructies inlezen in de variabele cmd$.

120 Machinetaalprogramma laden.

130-140 Uitvoer van de titel. De transparant-modus verzorgt onderstrepingen.

150-220 De instructieset en de letters om instructies op te roepen worden aangegeven.

230-260 Uitvoermedium voor de M-instructie kiezen (beeldscherm= b, printer= p). De letters knipperen (PEN3).

270-300 Afhankelijk van de invoer wordt bij de printeruitdraai het aantal bytes per regel (1) op 16 en het kanaalnummer (c) op 8 gezet. Bij uitvoer op scherm kunt u kiezen tussen 40 of 80 tekens per regel (1 of 2).

310-330 De hoofdlus wacht op de invoer van een instructie. het ingevoerde teken wordt met instructies vergeleken en als het overeenkomt wordt een subroutine opgeroepen.

350-390 hex-dump
Na uitvoer van het adres worden 1 (8 of 16) bytes als hex-getallen van twee cijfers uitgevoerd. Wordt het RAM uitgelezen dan gebeurt dit met PEEK, het ROM wordt uitgelezen met GOSUB 890.

400-470 ASCII-dump
Er is verschil tussen de uitvoer van ASCII-tekens op printer en op beeldscherm. Bij printeruitdraai wordt de hoogste bit van het teken gewist. De ASCII-code voor de punt vervangt een controleteken en de code voor DEL. De uitvoer op het beeldscherm in de 40-tekens modus vindt plaats in de tweede kleur. Dan verschijnen wel alle tekens (met CHR$(1)).

490-520 Deze routine verandert de inhoud van het geheugen. Als u een adres heeft gekozen, kan de nieuwe inhoud worden ingevoerd. Automatisch krijgt u daarna het volgende adres en wordt er gewacht op nieuwe invoer. U beëindigt deze modus met een ongeldig hex-teken, bijvoorbeeld een punt.

540 Deze twee GOSUBs betreffen de M-instructie. GOSUB 650 voert start- en eindadres van het bereik in waarom u heeft gevraagd terwijl de tweede oproep die adressen op het scherm tovert.

560-570 Deze routine verandert een ingevoerd hexadecimaal getal in een numerieke waarde. Dat maakt de & aan het begin overbodig.

590-630 Deze routine zet een RAM-bereik op band. U moet de grenzen van dit bereik plus de naam invoeren.

650-680 Deze routine voert een geheugenbereik in. De grenzen staan in de variabelen a en b.

700-730 Deze routine slaat een tekststring in het geheugen op. In regel 720 wordt de tekst teken voor teken in het geheugen gePOKEt.

750-760 Met deze programmaregels voert u een routine in machinetaal in het RAM uit, na het adres te hebben ingevoerd.

780-810 Deze routine kiest RAM of ROM. De actuele status verschijnt knipperend op het scherm. Met de spatiebalk schakelt u van RAM naar ROM en omgekeerd. Na de keuze maakt ENTER de invoer van instructies mogelijk.

830-870 Het machinetaalprogramma om het ROM uit te lezen in de vorm van DATA-regels. Regel 870 zet ze op de juiste plaats in het geheugen.

890-930 Een byte uit het ROM wordt gelezen. Het adres van de byte moet in variabele a staan. Na de scheiding in high- en lowbyte wordt de byte in de machinetaalroutine ingevoegd. Als u de routine met CALL oproept, slaat deze het resultaat op in &ABOE. Met PEEK leest u dit adres uit.

950 Na invoer van X wordt het scherm gewist en het programma beëindigd.

Een routine van het bedrijfssysteem leest het ROM van de CPC uit. Met RESTART-vector 3 (=oproep van een subroutine op &18) kunt u zowel in RAM als in ROM een routine oproepen. Hiertoe moet achter de instructie RST 3 het adres van een 3-bytes vector staan voor een FAR CALL. Deze vector bevat het adres van de routine en informatie over de keuze van ROM of RAM.

```
AB00                    ORG &AB00
AB00 DF                 RST 3
AB01 04 AB              DW  TABLE
AB03 C9                 RET
                ;
AB04 07 AB      TABLE   DW  READROM
AB06 FC                 DB  &FC ; ROMs selecteren
                ;
```

```
AB07 3A 00 00  READROM   LD  A,0 ; byte lezen
AB0A 32 0E AB            LD  BYTE,A ; en opslaan
AB0D C9                  RET
                ;
AB0E             BYTE    DS  1   ; ruimte voor gelezen byte
```

De tabel bevat de waarden voor de keuze van ROM en RAM.

waarde	geselecteerd geheugen
&00-&FB	uitbreidings-ROM
&FC	bedrijfssysteem + BASIC
&FD	BASIC
&FE	bedrijfssysteem
&FF	RAM

Met &FC kiest u het onderste (bedrijfssysteem) en het bovenste (BASIC) ROM. Voor de routine zelf (in het RAM) is dit niet van belang, want de adressen &4000-&BFFF zijn per definitie RAM. U kiest de bytes door voor de oproep van de routine low- en highbyte achter de instructie ld in adres READROM te POKEn. Het eerste voorbeeld voor een monitoruitvoer is een hex-dump van het begin van het programma. Het gaat dus om een gedeelte uit het RAM. Het tweede voorbeeld is het begin van het BASIC-ROM.

```
0160  00 00 00 00 00 00 00 00  ........
0168  00 00 00 00 00 00 00 00  ........
0170  31 00 64 00 AA 20 1C FF  1.d.* ..
0178  AA 01 20 A2 20 0E 2C 0F  *. " .,.
0180  01 20 A2 20 0F 2C 19 18  . " .,..
0188  01 20 A2 20 10 2C 19 17  . " .,..
0190  01 20 A2 20 11 2C 19 0F  . " .,..
0198  2C 19 18 01 20 AD 20 0F  ,... - .
01A0  00 41 00 6E 00 0D 05 00  .A.n....
01A8  EE EF 14 20 01 20 9E 20  no. . .
01B0  0D 0E 00 E9 EF 0E 20 EC  ...io. l
01B8  20 0D 05 00 EE 01 20 C3   ...n. C
01C0  20 03 09 00 63 6D E4 28  ...cmd(
01C8  0D 0E 00 E9 29 01 20 B0  ...i). 0
01D0  01 20 8C 20 61 2C 6D 2C  . . a,m,
01D8  73 2C 74 2C 67 2C 72 2C  s,t,g,r,
```

```
01E0  78 00 0A 00 78 00 9F 20  x...x..
01E8  1D 06 0B 00 6B 00 82 00  ....k...
01F0  BB 20 10 01 20 BF 20 EA  ; .. ? j
01F8  20 28 19 0A 29 20 22 4D   (..) "M
0200  20 69 20 6E 20 69 20 2D   i n i -
```

Hier begint het BASIC-ROM

```
C000  80 01 00 00 4C C0 31 00  ....L@1.
C008  C0 CD CB BC CD C4 F4 DA   MK<MDtZ
C010  00 00 21 00 AC 36 00 06  ..!.,6..
C018  1B 23 36 C9 10 FB 21 3F  .#6I.;!?
C020  C0 CD 37 C3 AF 32 00 AC  @M7C/2.,
C028  CD CB DD CD 84 CA CD 97  MKäM.JM.
C030  BD CD D3 C0 CD 3E C1 11  =MS@M>A.
C038  F0 00 CD 06 F7 18 25 20  p.M.w.%
C040  42 41 53 49 43 20 31 2E  BASIC 1.
C048  30 0A 0A 00 42 41 53 49  0...BASI
C050  C3 00 CD E1 CE C0 31 00  C.MaN@1.
C058  C0 CD 9A E7 CD 63 E1 CD  @M.gMcaM
C060  43 CA 38 54 CD 01 AC 31  CJ8TM.,1
C068  00 C0 CD 62 C1 CD D6 DD  .@MbAMVä
C070  DC B6 BC CD 48 BB CD 86  ö6<MH;M.
C078  C3 3A 45 AE B7 C4 3E C1  C:E.7D>A
```

4 DE OPSLAG VAN BASIC-REGELS, VARIABELEN EN TOKENS

4.1 Een BASIC-regel opslaan

Om deze paragraaf te begrijpen heeft u het monitorprogramma uit hoofdstuk 3 nodig. De opslag van een BASIC-regel is de schakel tussen machinetaal en BASIC. Verder bespreken we hoe variabelen in een BASIC-regel worden opgeslagen. Een computer begrijpt alleen binaire getallen. Daarom veranderen het bedrijfssysteem en de BASIC-interpreter alle BASIC-instructies in die code. Zonder BASIC-interpreter kunt u de computer nog zoveel vertellen, hij begrijpt er geen hout van. Maar zelfs met interpreter gaat het lang niet altijd goed, denk maar aan het fenomeen SYNTAX ERROR. Breid uw monitorprogramma voor machinetaal uit met de regel

```
1 PRINT "DATA BECKER"
```

Start vervolgens het monitorprogramma en kies het decimale stelsel en de leesmodus. Voer als beginadres 368 in en als eindadres 386. Om misverstanden te voorkomen geven we het volgende overzicht.

adres	waarde	commentaar
368	20	lowbyte van regellengte
369	0	highbyte van regellengte
370	1	lowbyte van regelnummer
371	0	highbyte van regelnummer
372	191	TOKEN voor PRINT
373	32	ASCII-waarde voor spatie
374	34	ASCII-waarde voor aanhalingstekens
375	68	ASCII-waarde voor D
376	65	ASCII-waarde voor A
377	84	ASCII-waarde voor T
378	65	ASCII-waarde voor A
379	32	ASCII-waarde voor spatie
380	66	ASCII-waarde voor B
381	69	ASCII-waarde voor E
382	67	ASCII-waarde voor C
383	75	ASCII-waarde voor K
384	69	ASCII-waarde voor E
385	82	ASCII-waarde voor R
386	34	ASCII-waarde voor aanhalingstekens

Zoals reeds eerder in dit boek ter sprake kwam luidt de formule voor de reconstructie van een getal dat is opgedeeld in highbyte en lowbyte:

```
highbyte * 256 + lowbyte
```

De regellengte is dus 20, maar er zijn slechts 19 adressen. De lengte van de regel is voor de CPC oninteressant, want hij gebruikt die waarde alleen om het beginadres van de volgende regel te berekenen. Het regelnummer bestaat ook weer uit high- en lowbyte, zodat het groter kan zijn dan 255. Het regelnummer is 1; dat kunt u zien aan de high- en lowbyte. De volgende waarde is een TOKEN, een belangrijk begrip voor BASIC-regels (zie paragraaf 4.2). De volgende waarden zijn ASCII-waarden voor "DATA BECKER". Alle programma's zijn op deze manier opgebouwd. De enige uitzondering is een BASIC-regel waarin een variabele wordt gedefinieerd. We komen daar in paragraaf 4.3 op terug.

4.2 Tokens

Een TOKEN (=teken) is een waarde die aan een BASIC-instructie wordt toegekend (net zoals binaire waarden aan machinetaalinstructies worden toegekend). Dit bespaart geheugenruimte. Het TOKEN voor PRINT is bijvoorbeeld maar 1 byte lang, terwijl voor het woord PRINT (5 letters) 5 bytes nodig zijn. Verder onderscheidt de computer de instructie beter van een variabele met een soortgelijke naam. Een variabele met de naam PRINT is uit den boze omdat de computer deze lettercombinatie als instructie interpreteert en een TOKEN zet. We hebben een lijst samengesteld van alle TOKENS van 1 byte (er zijn er ook van twee bytes). Een tip voor de beveiliging van programma's: schrijf in de eerste regel van het programma een REM-regel waarin u tussen aanhalingstekens een copyright plaatst. Een voorbeeld:

```
1 REM "COPYRIGHT BY DATA BECKER/NEDERLANDS*"
```

Zet in de tweede regel een belangrijke instructie, zonder welke het programma niet loopt, bijvoorbeeld een DIM of een functiedefiniëring. Zet midden in het programma een regel met de inhoud

```
POKE 372,191
```

Deze POKE-instructie verandert de eerste REM-regel in een PRINT-regel. Ontbreekt de REM-regel, dan wordt de tweede regel met de belangrijke instructie vernietigd. Onbevoegden die de eerste REM-regel uit het programma halen, werken zich dus aardig in de nesten.

instructie	waarde
AFTER	128
AUTO	129
BORDER	130
CALL	131
CAT	132
CHAIN	133
CHAIN MERGE	133&171
CLEAR	134
CLG	135
CLOSE IN	136
CLOSE OUT	137
CLS	138
CONT	139
DATA	140
DEF FN	141
DEF INT	142
DEF READ	143
DEF STR$	144
DEG	145
DELETE	146
DIM	147
DRAW	148
DRAWR	149
EDIT	150
ELSE	151
END	152
ENT	153
ENV	154
ERASE	155
ERROR	156
EVERY	157
FOR	158
GOSUB	159
GOTO	160
IF	161
INK	162
INPUT	163
KEY	164
KEY DEF	164&141
LET	165
LINE INPUT	166&163
LIST	167
LOAD	168
LOCATE	169
MEMORY	170

EI	220
ERL	227
FN	228
SPC	229
STEP	230
SWAP	231
TAB	234
THEN	235
TO	236
USING	237
>	238
=	239
>=	240
<	241
<>	242
<=	243
+	244
-	245
*	246
/	247
^	248
\	249
AND	250
MOD	251
OR	252
XOR	253
NOT	254
ABS	255

De TOKEN-waarden 221-226 en 232-233, die niet worden gebruikt, hebben een interessant resultaat. Schrijf in de eerste regel 1 REM en onthoud het tweede regelnummer. Schrijf vervolgens POKE 372,226. Wie daarna het programma probeert te listen, krijgt een SYNTAX ERROR. Het programma starten met RUN levert een SYNTAX ERROR in 1 op. Dat kan alleen worden verholpen met een RESET (gelijktijdig indrukken van CTRL, SHIFT en ESC), maar dan is men het programma wel kwijt. Starten gaat alleen met GOTO en het tweede regelnummer. U weet dat wel, een ander niet. Een onbekende zal het programma met RUN starten en het zo vernietigen. Met POKE 372,197 heft u deze beveiliging weer op. Let op: in een programma dat op deze manier is beveiligd, mag geen GOTO 1 of GOSUB 1 staan, want dan wordt het ook vernietigd.

4.3 Variabelen opslaan

Een BASIC-regel met een variabele is een uitzondering. U heeft voor deze paragraaf het monitorprogramma uit hoofdstuk 3 weer nodig. Toets de regel 1 C=100 in. Start daarna het monitorprogramma, kies het decimale stelsel en de leesmodus en voer als beginadres 368 in en als eindadres 379. U krijgt de volgende uitvoer op het scherm, die wij van commentaar hebben voorzien:

adres	waarde	commentaar
368	12	lowbyte van regellengte
369	0	highbyte van regellengte
370	1	lowbyte van regelnummer
371	0	highbyte van regelnummer
372	13	numerieke variabele
373	5	lengte van variabelenaam + 4
374	0	scheidingsnul
375	227	ASCII-waarde van variabelenaam + 128
376	239	TOKEN-waarde voor =
377	25	grootte van variabele
378	100	waarde van variabele
379	0	scheidingsnul

De verklaring voor de waarden van de eerste vier adressen vindt u in paragraaf 4.1. Gewoonlijk staat in adres 372 het TOKEN voor de instructie. Hier wordt daarentegen aangegeven dat variabele C een numerieke variabele is. Een stringvariabele zou een 3 hebben opgeleverd. De lengte van de variabelenaam in adres 373 is gecodeerd als waarde = lengte van de variabelenaam + 4. De volgende nul staat er alleen als scheiding. Dat geldt ook voor de nul in 379. De codering van de variabelenaam is als volgt: bij de laatste letter van de variabelenaam wordt 128 opgeteld. Alle andere letters van de variabelenaam behouden hun normale ASCII-waarde opgeslagen. De waarde 239 in adres 376 is de TOKEN-waarde voor het = teken. Hier is de ASCII-waarde niet genomen omdat duidelijk is dat = niet bij de variabelenaam hoort. De volgende waarde (25) is de gecodeerde waarde voor de grootte van de variabele. Deze vertelt de computer dat de waarde van de variabele in 1 byte past (dus niet groter is dan 255) en geen cijfers achter de komma heeft. Het gaat dus om een integer (=geheel) getal. In adres 378 is 100 de waarde van de variabele (immers C=100). Voor het tweede voorbeeld toetst u

```
1 C=1000
```

in. Voer als begin van de leesactie 372 in en als eindadres 380. De lijst met kort commentaar:

```
adres  waarde  commentaar
-------------------------
372      13     numerieke variabele
373       5     lengte van variabelenaam + 4
374       0     scheidingsnul
375     227     ASCII-waarde van naam + 128
376     239     TOKEN voor =
377      26     grootte van variabele
378     232     lowbyte van waarde van variabele
379       3     highbyte van waarde van variabele
380       0     scheidingsnul
```

Tot en met adres 376 is er niets veranderd In adres 377 staat 26. Dit betekent dat de waarde van de variabele groter is dan 255, kleiner dan 65535 en nog steeds een heel getal. De high- en lowbyte van de waarde van de variabele komen overeen met het getal 1000. De laatste scheidingsnul is bekend. Variabelen met een waarde van meer dan 65535 of getallen met cijfers achter de komma slaat de CPC op gelijke wijze op. Toets vervolgens

```
1 C=100000
```

in en als beginadres 372 en als eindadres 383. De lijst ziet er nu zo uit:

```
adres  waarde  commentaar

372      13     numerieke variabele
373       5     lengte van variabelenaam + 4
374       0     scheidingsnul
375     227     ASCII-waarde van variabelenaam + 128
376     239     TOKEN voor =
377      31     grootte van variabele
378       0     VAR 1
379       0     VAR 2
380      80     VAR 3
381      67     VAR 4
382     145     VAR 5
383       0     scheidingsnul
```

Tot en met adres 376 is alles bekend. De 31 in adres 377 geeft aan dat de waarde van de variabele groter is dan 65535 of geen heel getal is. De volgende 5 waarden zijn de waarde van de variabele. De formule voor de berekening van de waarde van de variabele is gecompliceerd:

```
waarde = (2 ^ (VAR 5 - 145)) * (65535 + (VAR 2 - 128) +
         (VAR 3 * 2) + (VAR 4 * 512) + (VAR 1 - 32800))
```

5.1 De joystick als muis

Met een muis zijn programma's veel makkelijker te hanteren. De muis is een kastje met een bal en een toets. Door de bal te bewegen kunt u op het scherm een symbool manipuleren, meestal een pijl. Hiermee kunt u op velden wijzen waarin functies staan. Door op de toets te drukken voert de computer de functie uit. Bent u geen ster in typen dan is de muis een uitkomst. Er is (nog) geen muis voor de CPC, maar de joystick kan deze functie ook vervullen. U heeft de joystick al bij het tekenprogramma gebruikt. De volgende routine (voor tekstverwerking) heeft een aantal menupunten. De joystick (AMSTRAD of ATARI-compatibel) zet een puntje op het beeldscherm. Breng dit puntje naar het teken (0) achter het menupunt van uw keuze en druk op de knop. U krijgt dan een subroutine. We geven eerst de routine, dan tekst en uitleg en tot slot verbeteringsmogelijkheden. Let bij het intikken op het juiste aantal spaties in de regels 30, 50, 70, 90, 110, 130, 150 en 170.

```
10 CLS : MODE 2
20 LOCATE 1,1
30 PRINT "tekst maken           0"
40 LOCATE 40,1
50 PRINT "tekst lezen           0"
60 LOCATE 1,8
70 PRINT "tekst veranderen      0"
80 LOCATE 40,8
90 PRINT "tekst wissen          0"
100 LOCATE 1,17
110 PRINT "tekst opslaan        0"
120 LOCATE 40,17
130 PRINT "tekst laden          0"
140 LOCATE 1,25
150 PRINT "tekst printen        0"
160 LOCATE 40,25
170 PRINT "programma afsluiten  0"
180 a=JOY(0)
190 PLOT x*8,400-y*16,0
200 IF a=1 THEN y=y-1
210 IF y<1 THEN y=25
220 IF a=2 THEN y=y+1
230 IF y>25 THEN y=1
240 IF a=4 THEN x=x-1
250 IF x<1 THEN x=80
```

```
260 IF a=8 THEN x=x+1
270 IF x>80 THEN x=1
280 PLOT x*8,400-y*16,1
290 IF a=16 THEN GOTO 300 ELSE GOTO 180
300 IF x=18 AND y=1 THEN GOTO 400
310 IF x=58 AND y=1 THEN GOTO 440
320 IF x=18 AND y=8 THEN GOTO 480
330 IF x=58 AND y=8 THEN GOTO 520
340 IF x=18 AND y=17 THEN GOTO 560
350 IF x=58 AND y=17 THEN GOTO 600
360 IF x=18 AND y=25 THEN GOTO 640
370 IF x=58 AND Y=25 THEN GOTO 680
380 GOTO 180
390 END
400 CLS
410 PRINT "tekst maken"
420 INPUT a$: IF a$<>"w" GOTO 420
430 GOTO 10
440 CLS
450 PRINT "tekst lezen"
460 INPUT a$: IF a$<>"w" GOTO 460
470 GOTO 10
480 CLS
490 PRINT "tekst veranderen"
500 INPUT a$: IF a$<>"w" GOTO 500
510 GOTO 10
```

Verklaring:

10-170	menu
180	waarde inlezen van joystickport in variabele
190	kleine cursor wissen
200-270	routine om kleine cursor met joystick te veranderen; variabelen voor x/y-richting worden verhoogd of verlaagd; daarna controle of waarde niet buiten scherm ligt, anders treedt later IMPROPER ARGUMENT op
280	zet grafiekcursor, de punt
290	knop ingedrukt, dan naar menupunt, anders joystickport uitlezen
300-370	als knop is ingedrukt, springt het programma hierheen en kijkt waar cursor stond
380	staat cursor bij druk niet op een 0 dan springt het programma terug naar uitlezing van port
400	vanaf hier kunt u routines inbouwen; regelnummers moeten worden aangepast!

Na het programma te hebben gestart, beweegt u met de joystick een

kleine cursor. In dit programma kunt u alleen de eerste drie punten met succes kiezen, omdat aan de andere geen routines zijn verbonden. U kunt het routine op twee punten verbeteren. U maakt de cursor groter door in plaats van de PLOT-instructie de instructie LOCATE te gebruiken. Ten tweede verdient het aanbeveling de plaats te vergroten waar u met de cursor heen moet gaan. Dat werkt veel makkelijker. Daartoe dient het bereik in de regels 300-370 te worden veranderd.

5.2 Bediening van de recordermotor

Twee adressen bedienen de cassettemotor, te weten:

&BC6E: start
&BC71: stop

Om de motor te laten lopen voert u dus CALL &BC6E in.

5.3 Een eenvoudige kopieerbeveiliging

In de CPC zit een kopieerbeveiliging en wel in de instructie SAVE. U beveiligt opgeslagen programma's door aan de programmanaam een komma en een p (voor protected: beschermd) toe te voegen. Het programma kan dan alleen met RUN worden geladen. Het start automatisch. Bij onderbreking is men het kwijt. Listen, opslaan, enzovoort is onmogelijk geworden.

5.4 Geheugenbereik opslaan

Er is een instructie om geheugenbereiken op te slaan. Die kunt u bijvoorbeeld gebruiken in een assembler om bepaalde bereiken op

band te zetten. Bij ons monitorprogramma hebben we dat gedaan. U kunt ook het hele beeldscherm of delen ervan opslaan zoals bij de grafiekeditor. Door deze bereiken weer te laden, verschijnen er snel weer beelden op het scherm. U omzeilt dus de langzame grafiekinstructies. Gaat u iets met het beeldscherm doen, bekijk dan eerst waar dat in het geheugen staat. In een vorige paragraaf zijn de adressen al eens ter sprake gekomen.

5.5 Zinvol gebruik van het toetsenbord

Met de eenvoudige instructies KEY en KEY DEF verandert u de functies van het toetsenbord. Het volgende programma helpt u daarbij.

```
10 MODE 2
20 LOCATE 22,5:PRINT "K E Y   M A N A G E R"
30 LOCATE 22,6:PRINT "====================="
40 PRINT:PRINT:PRINT:PRINT
50 PRINT "het volgende programma kent de getallen op uw
   toetsenbord enige hulpfuncties toe"
60 PRINT:PRINT"druk op een toets"
70 CALL &BB18
80 CLS
90 PRINT:PRINT:PRINT
100 KEY 137,d$
110 PRINT"toets 0     :LIST"
120 PRINT"toets 1     :RUN"
130 PRINT"toets 2     :LOAD"
140 PRINT"toets 3     :MODE"
150 PRINT"toets 4     :SAVE"
160 PRINT:PRINT:PRINT
170 PRINT "bovendien kunt u aan vijf andere toetsen per stuk
    maximaal 15 tekens toekennen"
180 INPUT "wanneer u dat wilt druk dan 'd' in";d$
190 IF d$="d" OR d$="D" THEN GOTO 260
200 CLS
210 PRINT:PRINT:PRINT:PRINT
220 PRINT"druk op een toets"
230 CALL &BB18
240 GOTO 320
250 CLS
260 PRINT:PRINT
```

```
270 INPUT "toets 6 :";a$
280 INPUT "toets 7 :";b$
290 INPUT "toets 8 :";c$
300 INPUT "toets 9 :";d$
310 GOTO 200
320 KEY 128,"list"+CHR$(13)
330 KEY 129,"run"+CHR$(13)
340 KEY 130,"load"
350 KEY 131,"mode "
360 KEY 132,"save"
370 KEY 133,"edit "
380 KEY 134,a$
390 KEY 135,b$
400 KEY 136,c$
410 KEY 137,d$
420 KEY 138,e$
430 CLS
```

5.6 Sprongadressen

In de volgende paragrafen staan een aantal handige adressen die in eigen programma's niet misstaan. We omschrijven de functie, vertellen wat over de noodzakelijke invoer/uitvoer-registers en geven bijzonderheden. Eerst nog de BASIC-lader om in BASIC alle functies op te roepen.

```
10 REM BASIC lader
20 MEMORY &2FFF : d = 0
30 INPUT "waarde ";w
40 IF w = 0 THEN GOTO 80
50 POKE &3000+d,w
60 d = d + 1
70 GOTO 30
80 CALL &3000
```

Deze BASIC-lader wijkt af van die in paragraaf 3.3. Het gaat hier om een oneindige lus met een afbreekcriterium. Regel 40 test of de ingevoerde waarde gelijk is aan 0. Is dit het geval, dan roept deze regel direct de subroutine op die de lader heeft geproduceerd. Als u een 0 als gegevenswaarde moet invoeren, verander dan de waarde in regel 40. Als u functies in een programma wilt invoegen, is het raadzaam de waarden van de

BASIC-lader in DATA-regels te zetten. Daarmee bouwt u dan de subroutine op. Met een POKE-instructie zet u variabele waarden in het juiste adres.

5.6.1 Bereik inkleuren

sprongadres: BC44 (hex)
8196 (dec)

Met deze routine geeft u gedeelten van het beeldscherm een kleurtje. De grootte is afhankelijk van de modus en de variabele grenspunten.

schrijven in het overdrachtsregister:

a	gecodeerde kleurwaarde (zie paragraaf 1.10)
h	linker kolom van bereik
d	rechter kolom van bereik
l	bovenste rij van bereik
e	onderste rij van bereik

Let erop dat het bereik niet buiten het beeldscherm komt, want dan hangt de CPC. De coördinaten van de linker bovenhoek van het beeldscherm zijn 0,0.

Hexadecimale waarden voor de BASIC-lader: 3E, (waarde voor tekenkleur/bitpatroon), 21, (boven), (links), 11, (beneden), (rechts), CD, 44, BC, C9, 0. De waarden voor boven, links enzovoort zijn variabel en komen overeen met regel- of kolomnummer.

assemblerformaat:

```
ld   a,N
ld   hl,hl
ld   de,NN
call 44,BC
ret
```

5.6.2 Van beeldscherm-bank wisselen

sprongadres: BC06 (hex)
48134 (dec)

Deze routine slaat het beeldschermgeheugen op in een van de vier banks (0-3). Om problemen te voorkomen met het bedrijfssysteem van de CPC kunt u het beste bank 1 of standaardbank 3 kiezen.

schrijven in het overdrachtsregister:

a de significante byte (highbyte) van het startadres van het nieuwe beeldschermbereik wordt aan de accu overgedragen

Gebruik alleen de waarden 00, 40, 80 en C0, want anders krijgt u niet gewenste resultaten.

Hexadecimale waarden voor de BASIC-lader: 3E, (nieuwe startwaarde), CD, 06, BC, C9, 0. Let bij de nieuwe startwaarde op de aanwijzingen in het overdrachtsregister.

Assemblerformaat:

```
ld    a,N
call  06,BC
ret
```

5.6.3 Wachten op een toets

sprongadres: BB06 (hex) 47878 (dec)

Deze routine zorgt voor een onderbreking van het programma. Deze onderbreking duurt tot een willekeurige toets wordt ingedrukt.

lezen uit het overdrachtsregister:

a deze routine leest de code van de ingedrukte

toets uit de accumulator

Deze functie roept u op een willekeurige plaats in het programma op met CALL &BB06.

assemblerformaat: call 06,BB

5.6.4 Scrolling

sprongadres: BC4D (hex) 48205 (dec)

Met deze routine schuift u het complete beeldscherm 1 regel naar boven of beneden. De kleurwaarde (PAPER) van de nieuw ingevoegde regel is variabel, maar natuurlijk afhankelijk van de MODE.

schrijven in het overdrachtsregister:

- b dit register bepaalt de verschuiving; bij 0 wordt er naar beneden gescrolled, een andere waarde zorgt voor een opwaartse verschuiving
- a geeft in de accumulator de gecodeerde kleurwaarde voor de nieuw te verschijnen regel aan

Hexadecimale waarden voor de BASIC-lader: (naar boven) 6, 1, 3E, (kleurcode van in te voegen regel), CD, 4D, BC, C9, 0.

Hexadecimale waarden voor de BASIC-lader: (naar beneden) 6, 0,3E, (kleurcode van de in te voegen regel), CD, 4D, BC, C9, 0.

```
assemblerformaat: ld   b,N
                  ld   a,N
                  call 4D,BC
                  ret
```

5.6.5 Gedeelten van het beeldscherm scrollen

sprongadres: BC50 (hex)
48208 (dec)

Met deze routine verschuift u vensters. De richting en kleurcode van de nieuwe regel komen overeen met de scroll-registers voor het hele beeldscherm.

schrijven in het overdrachtsregister:

a gecodeerde kleurwaarde
b richting waarin verschuiving plaatsvindt
h linker kolom van het te verschuiven venster
d rechter kolom van het venster
l bovenste rij van het venster
e onderste rij van het venster

Zorg dat u de beeldschermgrenzen niet overschrijdt, want dat leidt tot complicaties. Het veld links boven op het scherm heeft de waarde 0,0.

Hexadecimale waarden voor de BASIC-lader: 6, (richting), 3E, (kleurcode), 21, (boven), (links), 11, (beneden), (rechts), CD, 50, BC, C9, 0. De gekozen grenzen van het bereik zijn niet afhankelijk van een eerder gedefinieerd venster.

assemblerformaat:

```
ld   b,N
ld   a,N
ld   hl,NN
ld   de,NN
call 50,BC
ret
```

5.6.6 Modus kiezen

sprongadres: BC0E (hex)
48142 (dec)

Met deze routine wisselt u in een machinetaalprogramma van beeldschermmodus.

schrijven in het overdrachtsregister:

a de accumulator bevat de waarde van de nieuwe modus; de waarde van de accumulator ligt tussen 0 en 2

Hexadecimale waarden voor de BASIC lader: 3E, (nieuwe modus), CD, E, BC, C9, 0.

assemblerformaat:

```
ld   a,N
call OE,BC
ret
```

5.6.7 Een teken inverteren

sprongadres: BC4A (hex)
48202 (dec)

Met deze routine inverteert u een teken via twee bitpatroonmaskers. De invertering vindt plaats door twee maal de logische operator EXCLUSIVE OR.

schrijven in het overdrachtsregister:

b inverteringsmasker 1 (kleur 1)
c inverteringsmasker 2 (kleur 2)
h kolom
l rij

De positie is afhankelijk van de gekozen modus.

Hexadecimale waarden voor de BASIC-lader: 6, (masker 1), E, (masker 2), 21, (kolom), (regel), CD, 4A, BC, C9, 0.

Voorbeeld: modus 1, tekenkleur geel, achtergrondkleur blauw. Het teken op het beeldscherm dat moet worden geïnverteerd, is een 0. Laten we eens kijnen naar de invertering van de twee bovenste bitpatroonregels. Als inverteringsmasker 1 kiest u 1000 0000, masker 2 is 0000 1111.

```
beginsituatie       0111 0000  1100 0000
na invertering 1    1111 0000  0100 0000
na invertering 2    1111 1111  0100 1111
kleur van de pixel  RRRR RRRR  LLRR LLLL

R=rood
L=lichtblauw
```

assemblerformaat:

```
ld   b,N
ld   c,N
ld   hl,NN
call 4A,BC
ret
```

5.6.8 Horizontale scrolling

sprongadres: BC05 (hex)
48233 (dec)

Met deze routine verschuift u het beeldscherm horizontaal. De kolom die links wegvalt komt er aan de rechter kant van het beeldscherm weer bij (wrap around).

schrijven in het overdrachtsregister:

```
h  highbyte van de verschuifwaarde
l  lowbyte van de verschuifwaarde
e  verschuifwaarde moet deelbaar zijn door 2
```

Hexadecimale waarden voor de BASIC-lader: 21, (lowbyte van de verschuifwaarde), (highbyte van de verschuifwaarde), CD, 5, BC, C9, 0.

assemblerformaat:

```
ld   hl,NN
call 05 BC
ret
```

5.6.9 Cursorpositie kiezen

sprongadres: BB75 (hex)
47989 (dec)

Deze routine komt overeen met de BASIC-instructie LOCATE.

schrijven in het overdrachtsregister:

h nieuwe cursorkolom
l cursorregel

Er wordt niet gekeken of de gekozen waarden binnen het beeldscherm liggen.

Hexadecimale waarden voor de BASIC-lader: 21, (regel), (kolom), CD, 75, BB, C9, 0.

assemblerformaat:

```
ld   hl,NN
call 75 BB
ret
```

5.6.10 Kolom voor cursor kiezen

sprongadres: BB6F (hex)
47983 (dec)

Met deze routine kiest u een kolom in de regel waar de cursor staat.

schrijven in het overdrachtsregister:

a nieuwe kolomwaarde

Er wordt niet gecontroleerd of de gekozen waarde binnen het scherm ligt.

Hexadecimale waarden voor de BASIC-lader: 3E, (kolom), CD, 6F, BB, C9, 0.

assemblerformaat:

```
ld   a,N
call 6F BB
ret
```

5.6.11 Regel voor cursor kiezen

sprongadres: BB72 (hex)
47986 4(dec)

De plaats van de cursor kunt u definiëren door het kolom- en regelnummer aan te geven. Met deze routine wijzigt u die plaats door in de kolom van de cursor een andere regel te kiezen.

schrijven in het overdrachtsregister:

a nieuwe regelwaarde

Er wordt niet gecontroleerd of de nieuwe regel aanwezig is.

Hexadecimale waarden voor de BASIC-lader: 3E, (nieuwe regel), CD, 72, BB, C9, 0.

assemblerformaat:

```
ld   a,N
call 72,BB
ret
```

5.6.12 Joystick uitlezen

sprongadres: BB24 (hex)
47908 (dec)

Deze machinetaalroutine constateert wanneer de schakelaars van de joysticks zijn gesloten.

lezen uit het overdrachtsregister:

a na de uitlezing staat de code van de gesloten contacten in de accumulator
h dezelfde functie als de accumulator
l de status van de tweede joystick wordt tussentijds in register 1 opgeslagen

Een gezette bit komt overeen met een gesloten contact. Schakelopties en bits zijn op de volgende manier aan elkaar gerelateerd:

bit 0 boven
bit 1 beneden
bit 2 links
bit 3 rechts
bit 4 drukknop 2 (standaarddrukknop)
bit 5 drukknop 1
bit 6 ongebruikt
bit 7 ongebruikt

Hexadecimale waarden van de BASIC-lader: CD, 24, BB, 32, (lowbyte knop 2), (highbyte knop 2), 7D, 32, (lowbyte knop 1), (highbyte knop 1), C9, 0. Nadat deze routine is opgeroepen kunt u met twee PEEK-instructies de status van de beide joysticks uitlezen.

assemblerformaat:

```
call 24,BB
ld   (NN),a
ld   a,l
ld   (NN),a
ret
```

5.6.13 Invers in- en uitschakelen

sprongadres: BB9C (hex)
48028 (dec)

Met deze routine wisselt u de actuele kleuren van PAPER en PEN tegen elkaar uit. Met de eerste oproep schakelt u invers in en met de volgende uit. In BASIC sorteert PRINT CHR$(24) hetzelfde effect. In machinetaal roept u de functie op met CALL &BB9C.

assemblerformaat: call 9C,BB

5.7 Randomize - het toeval op het spoor

De CPC kent twee toevalsinstructies. Een daarvan is de RND-functie, waarmee u de volgende waarde uit een rij toevalsgetallen uitvoert. Voordat u zich over het voorbeeld buigt, moet u de computer resetten met CTRL + SHIFT + ESC.

```
10 REM toeval?
20 FOR t=1 TO 5
30 PRINT RND
40 NEXT t
```

Als u uw getallen met de volgende vergelijkt, zult u geen verschil zien.

```
0.271940658
0.528612386
0.021330127
0.175138616
0.657773343
```

Dat komt doordat deze random getallenreeks na een reset natuurlijk bij elke machine dezelfde is. Om te voorkomen dat u bij een spel van tevoren weet wat er gebeurt, is er een andere instructie: RANDOMIZE xx. Voor xx moet u een startwaarde invullen. Daar begint de reeks toevalsgetallen. Om dit te testen brengt u de CPC in de begintoestand. Vervolgens voert u het programma "toeval?" in. Om de RANDOMIZE-functie te gebruiken voegt u de volgende regel in:

```
15 RANDOMIZE 33
```

Start met RUN en noteer de toevalsgetallen. Hoewel de nieuwe

reeks niet op de oude lijkt, is het probleem van de terugkeer van de oude getallen na een RESET nog niet opgelost. Om een echt toevalsgetal te produceren, gebruikt u de real-time clock van de computer. Deze klok geeft in stapjes van 3/100 seconde aan hoeveel tijd er sinds het inschakelen van het apparaat is verstreken. We kunnen de startwaarde voor de RANDOMIZE-functie laten afhangen van de tijd die verstreken is. Mochten bij het volgende voorbeeldprogramma identieke rijen voorkomen dan is dat werkelijk toeval.

```
10 REM toeval
20 RANDOMIZE TIME
30 FOR t=1 TO 5
40 PRINT RND
50 NEXT t
```

Het programma draait om regel 20 waarin de variabele TIME de RANDOMIZE-functie aan de tijd relateert. RND levert altijd een getal dat groter is dan 0 en kleiner dan 1. Als u hele getallen wilt, moet de grondwaarde veranderd worden. Bij random getallen in het bereik 2-12 is 12 de bovengrens en 2 de ondergrens. Bovengrens minus ondergrens plus 1 levert het aantal mogelijke waarden: 12-2+1=11. Deze factor wordt met het door RND geleverde toevalsgetal vermenigvuldigd. Omdat RND nooit de waarde 1 levert, liggen de getallen tussen 0 en 10. Om op het bereik 2-12 uit te komen moet "+ ondergrens" in de formule worden verwerkt. INT selecteert het gehele gedeelte van het getal. De formule:

```
x = INT (RND * (bovengrens - ondergrens + 1) + ondergrens)
```

5.8 De CPC als rekenmachine

U kunt de CPC ook als rekenmachine gebruiken. Om bij berekeningen niet de hele tijd SHIFT te moeten gebruiken hebben we een programma gemaakt. Met de KEY DEF-functie zijn de toetsen zo gedefinieerd dat de bediening van verschillende toetsen tegelijk overbodig is. Eventueel kunt u de subroutine vanaf 200 weglaten. Die regelt de functies sin(x) en cos(x). Kort na de start van het programma meldt de CPC zich met de medeling VERANDERINGEN AANGEBRACHT. De wiskundige opties zijn een feit. Bovendien heeft de P de betekenis ? gekregen zodat de P indrukken voldoende is om de instructie PRINT op te roepen. Dat is bijzonder makkelijk omdat alle berekeningen vooraf dienen te gaan door PRINT. ? is de

afkorting van PRINT. Bij een vermenigvuldiging hoeft u niet meer SHIFT en dubbele punt tegelijk in te drukken: een van onze veranderingen is dat de dubbele punt voor * staat. Kijk maar eens in deze tabel!

wiskundige tekens	toets	toetswaarde
*	:	29
+	;	28
(		17
)		19
&		26
SIN(	0	128
COS(	1	129
TAN(	2	130
ATN(	3	131
ABS(	4	132
INT(	5	133
HEX$(	6	134
BIN$(	7	135
STR$(	8	136
?	p	27

Het verdient aanbeveling de toetsen waarvan de functie is gewijzigd, met stickertjes te voorzien van hun juiste functie.

```
10 REM rekenmachine
20 modus 2
30 KEY DEF 29,1,42
40 KEY DEF 28,1,43
50 KEY DEF 17,1,40
60 KEY DEF 19,1,41
70 KEY DEF 26,1,38
80 KEY DEF 27,1,63
90 GOSUB 200
100 PRINT"veranderingen aangebracht"
110 END
200 KEY 128,"sin("
210 KEY 129,"cos("
220 KEY 130,"tan("
230 KEY 131,"atn("
240 KEY 132,"abs("
250 KEY 133,"int("
260 KEY 134,"hex$("
270 KEY 135,"bin$("
280 KEY 136,"str$("
290 RETURN
```

De uitleg:

```
10        titel
20        80-tekens modus instellen
30- 80    functie van toetsen veranderen met instructie
          KEY DEF
90        sprong naar subroutine vanaf 200; als subroutine
          wordt weggelaten, vervalt deze regel
100       uitvoer van mededeling "veranderingen aangebracht"
110       veranderingen aangebracht, einde programma
200-280   cijfertoetsenblok definiëren als functietoetsen
          met instructie KEY
290       terug naar hoofdprogramma
```

Functies wijzigen gebeurt dus door de waarden van toetsen te veranderen. Mocht ons voorstel u niet aanstaan, dan kunt u dit naar eigen inzicht doen. De waarden van de toetsen staan in de bijlage van het CPC handboek.

5.8.1 Nauwkeurigheid van berekeningen

Nauwkeurigheid bij berekeningen is af te meten aan het aantal cijfers achter de komma. De CPC kan 9 plaatsen voor of achter de komma nog correct opslaan. Dat wil zeggen dat de onnauwkeurigheid bij grotere getallen groter is dan 1. Laten we die bewering eens natrekken. Toets in:

```
PRINT 5E9+1-5E9
```

De onnauwkeurigheid is 1, want als uitkomst krijgen we niet 1 maar 2. Het grootste integere (gehele) getal dat correct kan worden opgeslagen is 2^32-1 (4.294.967.295). Dit hangt samen met de opslag van variabelen (paragraaf 4.3). Om de nauwkeurigheid te verhogen berekent u de waarden met formules. In deze formules mogen geen vaste waarden zoals wortels of pi voorkomen, omdat die de precisie schaden. Als voorbeeld hebben we een dergelijke formule tot een programma omgebouwd, waarin de instructie DEF FN niet hoeft te worden gebruikt. De formule berekent exp(x) zo:

```
exp(x)=1+x/1!+x^2/2!+x^3/3/!...
```

Met dit programma kunt tot 31 faculteit rekenen, daarna is de grens van 1E34 bereikt. Omdat de CPC de functie faculteit niet bezit hebben we die gesimuleerd in de subroutine vanaf 100. Als u in dit programma de verlangde x invoert, krijgt u na korte tijd

het resultaat exp(x).

```
10 REM EXP(x)
20 INPUT"x-waarde";x
30 FOR n=1 TO 31
40 x1=x^n
50 GOSUB 100
60 erg1=x1/fa
70 erg2=erg2+erg1
80 NEXT
90 PRINT erg2+1
95 END
100 fa=1
110 FOR m=1 TO n
120 fa=fa*m
130 NEXT m
140 RETURN
```

Verklaring:

```
 10  titel
 20  invoer van x-waarde
 30  lus openen die aantal faculteit bepaalt
 40  vorming van x-machten
 50  sprong naar subroutine vanaf 100
 60  verwerking van faculteiten
 70  resultaat optellen
 80  lus sluiten
 90  uitvoer van resultaat
 95  einde programma
100  faculteit terugzetten
110  lus openen voor faculteit
120  formule om faculteit te berekenen
130  lus sluiten
140  sprong terug in hoofdprogramma
```

Omdat u niet alle 31 delen in de DEF FN instructie kunt onderbrengen, kunt u formules als exp(x) slechts onnauwkeurig met DEF FN berekenen. Een ander voorbeeld voor het gebruik van formules is de berekening van nulpunten. Nulpunten zijn punten waarin de grafiek van een functie de X-as snijdt. Deze punten zijn belangrijk voor de analyse van een functiegrafiek. Uit de tekening zijn ze niet nauwkeurig op te maken. We gaan uit van de tweede machts vergelijking

```
f(x)=x^2+p*x+q
```

De formule voor de berekening van de nulpunten luidt als volgt:

```
x1= -p/2+sqr(p^2/4*q)
x2= -p/2-sqr(p^2/4*q)
```

U voert eerst de waarde voor p in en dan de waarde voor q. Het programma levert vervolgens de nulpunten (x1,x2).

```
10 INPUT p
20 INPUT q
30 x1=-p/2+SQR(p*p/4-q)
40 x2=-p/2-SQR(p*p/4-q)
50 IF x1*x2<>q THEN 80
60 PRINT x1,x2
70 END
80 PRINT"geen nulpunten !"
90 END
```

Verklaring:

```
10  invoer p
20  invoer q
30  eerste nulpunt berekenen
40  tweede nulpunt berekenen
50  nulpunten testen met stelling van Vieta
60  uitvoer van nulpunten
70  einde programma
80  uitvoer, want geen nulpunten aanwezig
```

Als het programma door een foutmelding wordt onderbroken, zijn er geen nulpunten.

5.8.2 Rekensnelheid

Met een testprogramma dat op meerdere computers loopt, kunt u de rekensnelheid van computers met elkaar vergelijken. De standaardopgaven om de rekensnelheid te testen heten BENCHMARK-test. De BENCHMARK-test die we hier gebruiken heet ZEEF VAN ERATOSTHENES en wordt veel gebruikt om de snelheid van compilers met elkaar te vergelijken. Bij de rekensnelheid hoort ook de verwerking van uitlezingen, lussen en het beheer van datavelden. Die zaken test de ZEEF VAN ERATOSTHENES ook. Het resultaat van de berekening van deze opgave door twee computers luidt als volgt:

```
Apple II Europlus met MBASIC 5.2 ... 391,8 sec.
VIC 20 met 64K kaart ..... 394,8 sec.
```

Vis zelf eens uit of de CPC minder tijd nodig heeft.

```
10 REM Eratosthenes
20 DEFINT a-z
30 DIM f1(1000):1=1000:PRINT"start"
40 FOR j=1 TO 10
50 co=0
60 FOR i=1 TO 1:f1(i)=1:NEXT i
70 FOR i=1 TO 1
80 IF f1(i)<>1 THEN 160
90 priem=i+i+3
100 k=i+priem
110 IF k>1 THEN 150
120 f1(k)=0
130 k=k+priem
140 GOTO 110
150 co=co+1
160 NEXT i
170 NEXT j
180 PRINT co;"priemgetallen"
190 END
```

De CPC is dus sneller dan de andere twee.

5.9 Beeldschermbewegingen

Veel BASIC-dialecten kennen een SCROLL-instructie, waarmee het beeldscherm een regel naar boven of beneden kan schuiven. Een routine van het CPC-ROM simuleert deze instructie. Normaal beweegt u het beeldscherm naar boven door toetsen in te drukken. Is het beeldscherm vol, dan scrollt het verder. Maar naar beneden scrollen kan niet. Om dat voor elkaar te krijgen heeft u een lange machinetaalroutine nodig. Bij de CPC staat die routine in het ROM. De routine voor de beweging van het beeldscherm staat in het ROM (vanaf adres &BC4D) (zie paragraaf 5.6 Sprongadressen). Deze routine beweegt het beeldscherm 8 pixels (=1 teken). Is de waarde van het B-register nul, dan is de beweging naar beneden gericht, anders naar boven. U spreekt de ROM-routine aan met een kort machinetaalprogramma waarin u de waarde voor het B-register met een POKE verandert of met twee routines, een voor omhoog en

een voor omlaag. Omdat de laatste mogelijkheid het minst problematisch is, zullen we die gebruiken. Eerst moet het B-register met LD B,N worden geladen. De code daarvoor is 06. De dan volgende waarde is voor omlaag 0 en voor omhoog 255. Dit is in principe het enige verschil tussen de beide routines. U roept de ROM-routine op met CALL NN (code 205). Het adres &BC4D, dat hier op volgt, wordt in een high- en een lowbyte verdeeld. Na code 205 komt dus 77 (lowbyte 4D) en daarna 188 (highbyte BC). RET (code 201) zorgt voor een sprong terug. De machinetaalroutines zien er als volgt uit:

Naar boven:

```
LD B,FF         06;255
CALL &BC4D      205;77;188
RET             201
```

Naar beneden:

```
LD B,0          06;0
CALL &BC4D      205;77;188
RET             201
```

Het BASIC-programma dat de machinetaalroutines genereert, slaat ze op vanaf 43880 (omhoog) en vanaf 43886 (omlaag). Na afloop van het programma wist u dit BASIC-programma met DELETE -140. U kunt dan een eigen programma kwijt dat van de twee machinetaalroutines gebruik maakt.

```
10 REM omhoog
20 MEMORY 43879
30 DATA 6,255,205,77,188,201
40 FOR adr=43880 TO 43885
50 READ waarde
60 POKE adr,waarde
70 NEXT
80 REM omlaag
90 DATA 6,0,205,77,188,201
100 FOR adr=43886 TO 43891
110 READ waarde
120 POKE adr,waarde
130 NEXT adr
140 END
```

De uitleg:

```
10      titel
20      begrenzing van BASIC-geheugen
30      gegevens voor omhoog-routine
```

```
 40       lus openen om machinetaalroutine op te slaan
 50       waarden lezen
 60       waarden in geheugenplaatsen zetten
 70       lus sluiten
 80       titel
 90       gegevens voor omlaag-routine
100-130   gelijk aan 40-70
140       einde programma
```

U hoeft de routine in BASIC alleen met CALL op te roepen (CALL 43880 voor omhoog en CALL 43886 voor omlaag). Een voorbeeld om de scroll-mogelijkheden te illustreren. We zetten een tekst op het beeldscherm en gaan met de SCROLL-routines 5 regels naar boven en beneden. U bepaalt de snelheid met de toetsen.

```
10 REM SCROLL-DEMO
20 MODE 1
30 LOCATE 15,12:PRINT"scroll-demo"
35 REM OMHOOG
40 FOR n=1 TO 5
50 IF INKEY$="" THEN 50
60 CALL 43880
70 NEXT n
75 REM omhoog
80 FOR n=1 TO 5
90 IF INKEY$="" THEN 90
100 CALL 43886
110 NEXT n
120 END
```

De uitleg:

```
 10  titel
 20  beeldschermmodus instellen
 30  uitvoer van tekst
 35  titel
 40  lus openen voor aantal bewegingen
 50  uitlezing van toetsenbord
 60  oproep van SCROLL-routine (omhoog)
 70  lus sluiten
 75  titel
 80  lus openen voor aantal bewegingen
 90  uitlezing van toetsenbord
100  oproep van SCROLL-routine (omlaag)
110  lus sluiten
120  einde programma
```

De demo loopt niet als u niet eerst het programma hebt laten lopen dat machinetaalroutines genereert. Om de oproep te vereenvoudigen

kunt u twee variabelen de waarden voor de oproep toekennen:

```
omhoog=43880
omlaag=43886
```

Let er bij de routine op dat de kleurwaarde van de boven of onder ingevoegde regel in het A-register staat. Om richting en lengte van de beweging en de kleur van de ingevoegde regel met een invoer te kunnen veranderen, moet u de machinetaalroutine wat uitbreiden. We gebruiken het B-register waarin de richting van de beweging is opgeslagen als teller voor het aantal in te voegen regels. Het A-register bevat de kleurwaarde van de ingevoegde regels. Omdat de inhoud van het register bij een oproep van de ROM-routine verloren gaat, wordt die op de STACK (=stapelgeheugen) gelegd met de instructie PUSH. Na de oproep haalt u met POP de inhoud weer terug. De uitgebreide machinetaalroutine ziet er als volgt uit:

adres	instructie	code	verklaring
43870	LD B,N	06	aantal regels (N)
43871		N	wordt in B geladen
43872	LD A,M	62	kleurwaarde (M) wordt
43873		M	in A geladen
43874	PUSH BC	197	B wordt op STACK gelegd
43875	PUSH Af	245	A wordt op STACK gelegd
43876	LD B,R	06	B wordt met waarde
43877		R	van richting geladen
43878	CALL &BC4D	205	oproep van ROM-routine
43879		77	lowbyte van adres
43980		188	highbyte van adres
43881	POP AF	241	A wordt van STACK gehaald
43882	POP BC	193	B wordt van STACK gehaald
43883	DJNZ,e	16	volgt later
43884		245	
43885	RET	201	sprong terug naar BASIC

De letterlijke betekenis van DJNZ,e is: Decrement and Jump if Not Zero (decrement (verlaging) en spring e terug indien niet nul). Om te voorkomen dat het aantal regels niet tot 1 beperkt blijft, is een sprong naar adres 43874 nodig. Het B-register dient om te tellen. Bij elke ronde door het programma wordt het B-register met 1 verlaagd. Is het daarna niet gelijk aan 0 dan springt het programma 9 bytes terug, naar adres 43874. Omdat de programmateller al op de volgende instructie wijst, wordt 2 bij de 9 bytes opgeteld. Voor een sprong terug moet de spronggrootte van 256 worden afgetrokken. De uitkomst, 245, komt na de instructiecode te staan. Als het B-register gelijk is aan nul keert het programma naar BASIC terug.

```
10 REM SCROLL-DEMO
20 MEMORY 43869
30 ADR=43869: SCROLL=43870
40 DATA 6,10,62,0,197,245,6,255,205,77
50 DATA 188,241,193,16,245,201
60 FOR CO=1 TO 16
70 READ WAARDE
80 POKE ADR+CO,WAARDE
90 NEXT CO
100 MODE 1
110 INPUT"kleur";m
120 INPUT"aantal regels";n
130 INPUT"omhoog(1) of omlaag(2)";r
140 IF r=1 THEN r=255 ELSE r=0
150 POKE 43873,m
160 POKE 43871,n
170 POKE 43877,r
180 CALL scroll
190 GOTO 100
```

Verklaring:

10	titel
20	begrenzing van BASIC-geheugen
30	beginadres van programma bepalen
40-50	gegevens voor machinetaalroutine
60	lus openen om gegevens in te lezen
70	machinetaalcodes inlezen
80	waarden in geheugenplaatsen opslaan
90	lus sluiten
100	beeldschermmodus instellen
110	invoer van kleurwaarde
120	invoer van aantal regels
130	kiezen: omhoog of omlaag
140	verwerking van laatste invoer
150	kleurwaarde opslaan
160	aantal regels opslaan
170	bewegingsrichting opslaan
180	oproep van machinetaalroutine
190	sprong terug naar 100

De invoerwaarden moeten in de volgende bereiken liggen:

kleurwaarde	0-255
aantal regels	1- 25

De volgende adressen gebruikt u om de machinetaalroutine in uw eigen programma in te bouwen:

43873 kleurwaarde hier opslaan
43871 aantal regels hier opslaan
43877 richting van beweging hier opslaan
43870 startadres voor machinetaalroutine

5.10 Gegevens sorteren

U kunt numerieke gegevens sorteren maar ook gegevens van een adressenbestand. Het criterium voor het sorteren van getallen is hun grootte. Teksten sorteert u op grond van hun ASCII-code. De code is zo gekozen dat A kleiner is dan B en deze weer kleiner dan C. De cijfers en enkele bijzondere tekens gaan aan de hoofdletters vooraf. Na de hoofdletters komen de kleine letters. De BASIC-interpreter verzorgt het sorteerproces. Om te sorteren kunt u van allerlei algoritmen gebruik maken. Ze zijn complex maar efficiënt. Bij een beperkt aantal gegevens (10-50) is een eenvoudige procedure voldoende. Maar over grote bestanden (100-1000) doet die meerdere uren. Om u een indruk te geven introduceren we de eenvoudigste en de snelste procedures: BUBBLE-SORT en QUICK-SORT. BUBBLE-SORT vergelijkt steeds twee elementen met elkaar en verwisselt ze indien nodig. In BASIC gebeurt dat met twee geneste lussen.

```
100 FOR i=1 TO N: fl=0
110 FOR j=n TO i step -1
120 IF a$(j-1)>a$(j) THEN h$=a$(j):a$(j)=a$(j-1):a$(j-1)=h$:
    fl=1
130 NEXT j: IF fl=0 THEN RETURN
140 NEXT i: RETURN
```

Met GOSUB 100 roept u het programma als subroutine op. De te sorteren gegevens staan als string-array in A$. De grootte van het veld (de maximale index) zet u in variabele n, bijvoorbeeld:

```
10 n=50: DIM a$(n)
20 'gegevens inlezen
30 GOSUB 100 'gegevens sorteren
```

Dit eenvoudige sorteerprogramma is geschikt voor een klein aantal gegevens of een groter aantal al gesoteerde gegevens waar een nieuw stel moet worden ingevoegd. Het volgende programma, QUICK-SORT, is geschikt voor grotere hoeveelheden. Hier wordt ook een stringveld gesorteerd waarvan de indexgrens in n staat. Het

programma genereert een aantal woorden (dat u kiest), laat ze zien, sorteert ze en toont ze nog eens gesorteerd en al, inclusief de tijd die hiervoor nodig was.

```
10 DEFINT b-z
20 INPUT "aantal woorden ";n: DIM d$(n),o(20),u(20)
30 FOR i=1 TO n: b=(5+10*RND)
40 FOR j=1 TO b : d$(i)=d$(i)+CHR$(RND*25+65) : NEXT: NEXT
50 GOSUB 80: a=TIME: GOSUB 100: a=(TIME-a)/300
60 PRINT:GOSUB 80:PRINT:PRINT a "sec.": END
70 '
80 FOR i=1 TO n: PRINT d$(i): NEXT:RETURN
90 '
100 s=1: o(1)=1: u(1)=n
110 l=o(s): r=u(s): s=s-1
120 i=1: j=r: h$=d$((l+r)/2)
130 WHILE d$(i)<h$ AND i<r: i=i+1: WEND
140 WHILE d$(j)>h$ AND j>l: j=j-1: WEND
150 IF i<=j THEN d$=d$(i):d$(i)=d$(j):d$(j)=d$:i=i+1:j=j-1
160 IF i<=j THEN 130
170 IF r-i<=j-1 THEN 200
180 IF l<j THEN s=s+1: o(s)=l: u(s)=j
190 l=i: GOTO 220
200 IF i<r THEN s=s+1: o(s)=i: u(s)=r
210 r=j
220 IF r>l THEN 120
230 IF s>0 THEN 110
240 RETURN
```

Als u het programma met velden van verschillende grootte heeft laten lopen, krijgt u ongeveer de volgende resultaten:

n	sec.
50	3.8
100	7.9
200	18.6
500	110

De tijd voor 500 wooorden is verhoudingsgewijs hoog vanwege de garbage collection. Die komt in actie als er met veel strings wordt gewerkt. Nadat het voorbeeld met de 500 strings te hebben uitgevoerd, moet u eens

```
? FRE("")
```

intoetsen. Nu ruimt de BASIC-interpreter in ongeveer 29 seconden zijn stringgeheugen op. Hij doorzoekt het complete geheugen en wist de strings die niet meer nodig zijn of schrijft geldige

strings over de oude. Hoe meer geldige strings, hoe meer tijd deze procedure kost. Als er, zoals bij het sorteren, voortdurend strings gekopieerd moeten worden en er niet voldoende geheugenruimte meer is, is het hoog tijd voor een garbage collection. De sorteertijd loopt daardoor enorm op. Ter vergelijking nog even de tijden van BUBBLE-SORT:

n	sec.
50	12.1
100	49.0
200	238.2

5.11 Achtergrond printen

Als u met een printer werkt moet u met een nieuwe invoer steeds wachten tot de printer klaar is. Dit is vervelend en daarom introduceren we een routine waarmee u met het programma kunt doorgaan zonder op de uitvoer te hoeven wachten. Voeg aan uw programma de regels 65000-65240 toe. Voortaan kent u aan variabele pr$ de uit te voeren gegevens toe en GOSUB 65000 doet de rest. De computer vult de printerbuffer met pr$ en uw programma loopt verder (als de buffer tenminste niet vol is). U kunt de grootte naar behoefte aanpassen. Met de EVERY-instructie controleert de computer steeds opnieuw of er iets in de buffer staat. Is dit het geval, dan wordt er een regel uitgevoerd, enzovoort. Let erop dat het einde van het programma er uitziet als in de regels 220 en 240, zodat de buffer ook volledig wordt geprint. De regels 100-200 dienen als demonstratie. Als u serieus met het programma gaat werken, verander dan de 7 in regel 65200 in 8.

```
100 CLS:WINDOW #7,21,40,1,25
140 FOR i=0 TO 130
160 pr$=HEX$(i,6)
180 GOSUB 65000
200 NEXT
210 PRINT"klaar"
220 ON bf1 GOTO 220
240 END
```

```
65000 REM fill buffer
65020 ON f1 GOTO 65060
65040 DIM pr$(999):f1=1:pfc=1
65050 EVERY 30,0 GOSUB 65160
65060 IF pc=pfc THEN 65060
65080 DI:pr$(fc)=pr$:bf1=1
65100 fc=fc+1:IF fc>999 THEN pfc=0
65120 pfc=fc+1:IF pfc>999 THEN pfc=0
65140 RETURN
65160 REM print buffer
65180 IF pc=fc THEN bf1=0 : RETURN
65200 PRINT#7,pr$(pc)
65220 pc=pc+1:IF pc>999 THEN pc=0
65240 RETURN
```

6 PROGRAMMA'S VOOR DE GEBRUIKER

6.1 Inleiding

Voor de CPC verschijnt steeds meer software op de markt. Toch willen we in dit boek nog een aantal programma's naar voren brengen. Het accent ligt hierbij op een programma voor databeheer. Aan de uitleg hebben we verbeteringsvoorstellen toegevoegd, zodat u zelf aan de slag kunt. Veel succes.

6.2 Programma voor bestandsbeheer

Bestandsbeheer is universeel, in tegenstelling tot adressenbeheer of voorraadbeheer. De laatste twee zijn slechts voor een bepaald doel te gebruiken. Het beheer van een bestand kunt u aanpassen. In feite is bestandsbeheer niets anders dan werken met gegevensreeksen en velden. Het bestand is vergelijkbaar met een kaartenbak, het systeemkaartje met een gegevensreeks en de afzonderlijke informatie met een veld. Binnen een bestand is het aantal gegevensreeksen en velden variabel. Elk veld heeft een naam. Als u een adressenbestand opbouwt, definieert u het eerste veld als NAAM-VOORLETTERS, het tweede als STRAAT, het derde als PLAATS enzovoort. Met wat fantasie kunt u dit beheer ook voor tekstverwerking gebruiken. Het nu volgende programma is complex en lang en daarom modulair. Dat wil zeggen, gesplitst in afzonderlijke, zinvolle delen. Als u het programma overneemt, moet u wel alle delen achter elkaar zetten.

6.2.1 Menu

In het menu staan de functies van het programma, een opsomming van wat allemaal mogelijk is. Elk onderdeel heeft een nummer. Door dit in te toetsen en op ENTER te drukken kiest u een bepaalde optie. Als het programma-onderdeel is beëindigd, komt u terug in

het menu waar u opnieuw een keuze kunt maken.

```
10 REM
20 REM
30 REM
40 REM
50 REM
60 REM
70 REM
80 REM
90 MODE 2
100 LOCATE 20,3:PRINT"C P C  4 6 4  B E S T A N D S B E H E E R"
110 LOCATE 20,5:PRINT STRING$(&24,"=")
120 LOCATE 12,7:PRINT"COPYRIGHT:  D A T A  B E C K E R  1984"
130 LOCATE 15,10:PRINT"                              M E N U"
140 LOCATE 15,11:PRINT"                              -------"
150 LOCATE 15,13:PRINT"BESTAND openen                  - 1 -"
160 LOCATE 15,14:PRINT"BESTAND invoeren                - 2 -"
170 LOCATE 15,15:PRINT"BESTAND bijhouden               - 3 -"
180 LOCATE 15,16:PRINT"BESTAND opslaan                 - 4 -"
190 LOCATE 15,17:PRINT"BESTAND laden                   - 5 -"
200 LOCATE 15,18:PRINT"        zoeken                  - 6 -"
210 LOCATE 15,19:PRINT"BESTAND printen                 - 7 -"
220 LOCATE 15,20:PRINT"        beeindigen              - 8 -"
230 LOCATE 15,21:PRINT"        wissen                  - 9 -"
240 PRINT:PRINT:INPUT"uw K E U Z E   (1-9)   <return>";a
250 ON a GOSUB 280,630,1010,1440,1570,1700,1900,2020,2070
260 GOTO 90
```

De uitleg:

```
 10- 80  informatie in deze REM-regels is alleen zichtbaar als het
         programma wordt gelist; u kunt het beste zelf bedenken
         wat u daar als geheugensteuntje wilt hebben staan
 90      CPC BASIC-instructie schakelt over op 80-tekens modus
100-230  LOCATE-instructie voert menubeeld uit
240      keuze programma-onderdeel
250      routine om afzonderlijke delen te bereiken
260      na sprong terug met RETURN komt u weer aan begin van
         menu
```

Variabelen:

```
a  nummer van gewenste programma-onderdeel
```

Handleiding bij dit gedeelte:

U voert het verlangde nummer van het programma-onderdeel in en drukt op ENTER. Het programma doet de rest.

6.2.2 Bestand openen

In dit gedeelte definieert u het bestand. U geeft eerst aan hoe groot het moet zijn en daarna hoe het moet worden genoemd. Typefouten kunt u na de definitie veranderen.

```
270 REM
280 REM bestand openen
290 CLS
300 PRINT STRING$(&50,"-")
310 LOCATE 15,3:PRINT"B E S T A N D   O P E N E N"
320 PRINT STRING$(&50,"-")
330 LOCATE 1,: INPUT"hoeveel records moet het bestand bevatten";
    records
340 PRINT:INPUT"hoeveel velden moet een record hebben";velden
350 DIM benamingr$(records)
360 DIM benamingv$(velden)
370 DIM inhoud$(velden,records)
380 REM velden definieren
390 PRINT"velden van een record definieren"
400 hrecords=records
410 hvelden=velden
420 PRINT"--------------------------------"
430 PRINT:PRINT"(u moet";velden;" velden definieren)"
440 PRINT:PRINT
450 FOR velden = 1 TO hvelden
460 PRINT"benaming van het";velden;"e veld":INPUT"";
    benamingv$(velden)
470 NEXT velden
480 PRINT: INPUT"wilt u veranderingen aanbrengen";veranderingen$
490 IF LEFT$(veranderingen$,1)="n" OR LEFT$(veranderingen$,1)="N"
    THEN GOTO 90
500 CLS
510 PRINT"benaming van een veld veranderen"
520 PRINT"--------------------------------"
530 FOR velden = 1 TO velden -1
540 PRINT"benaming";velden;":",benamingv$(velden)
550 NEXT
560 PRINT:PRINT
570 INPUT"welk veld wilt u veranderen";velden
```

```
580 PRINT"oude benaming van veld"benamingv$(velden)
590 INPUT"nieuwe benaming van veld";benamingv$(velden)
600 INPUT"wilt u nog iets veranderen";veranderingen$
610 IF LEFT$(veranderingen$,1)="n" OR LEFT$(veranderingen$,1)="N"
    THEN GOTO 90 ELSE GOTO 560
620 RETURN
```

De uitleg:

290-320	titel van dit gedeelte
330+340	grootte van bestand invoeren
350+360	variabelen voor naam dimensioneren met juiste grootte
400+410	aantal velden en records in hulpvariabelen opslaan (hvelden, hrecords), omdat de oude variabelen door gebruik in lussen andere waarden kunnen krijgen
380-440	titel van subroutine
450-470	lus om namen van velden in te lezen
480	beslissing ten aanzien van veranderingen
490	nee, dan terug naar menu
500-520	titel van subroutine
530-550	lus om namen van velden uit te voeren
560-620	routine voor naamsverandering; geen verandering nodig, dan terug naar menu

Variabelen:

records	aantal records van bestand
velden	aantal velden van bestand
hrecords	aantal records permanent geregistreerd
hvelden	aantal velden permanent geregistreerd
naamg$ (records)	naam van records
naamv$ (velden)	naam van velden
veranderingen$	beslissing: ja of nee

Verbeteringen:

Zoals het begrip bestand nu is uitgelegd, ligt het aantal records en velden vast. Als uw bestand groter wordt dan gepland, heeft u een dynamisch concept nodig, waardoor het bestand kan meegroeien. Zo'n aanpassing heeft nogal wat voeten in de aarde en is alleen voor gevorderden weggelegd. Als beginner kunt u wel een aantal regels schrijven waarmee u foutieve invoer verhindert of waarvan u bij de invoer van vensters gebruik maakt. Sla daar eerst het handboek eens op na. Met het gebruik van windows verbetert u de outfit en het gemak aanzienlijk.

Handleiding bij dit gedeelte:

Het programma is interactief; dat betekent een vraag- en antwoordspel met de computer. U voert eerst het aantal records in (bij een adressenbestand bijvoorbeeld het aantal personen) en daarna het aantal velden (het aantal gegevens per persoon). Daarna voert u de namen van de velden in (naam, straat, plaats,...). Als dit proces is afgesloten, bepaalt u of u naar het menu terug wilt (= nee invoeren) of veranderingen wilt aanbrengen (= ja invoeren). Als u veranderingen aanbrengt, voert het programma eerst de namen van de velden uit. Daarna geeft u het nummer en de nieuwe naam aan. Bent u daarmee klaar dan bepaalt u opnieuw wat u wilt doen, een verandering aanbrengen of terug naar het menu.

6.2.3 Bestand invoeren

Dit gedeelte van het programma vult het gedefinieerde bestand met gegevens: namen, straten, plaatsen, enzovoort. Veranderingen kunt u hierna meteen aanbrengen. Aan het begin van dit onderdeel voert u de naam van het bestand in.

```
630 CLS
640 velden=hvelden
650 records=hrecords
660 PRINT STRING$(&50,"-")
670 LOCATE 15,3: PRINT"B E S T A N D   I N V O E R E N"
680 PRINT STRING$(&50,"-")
690 LOCATE 1,10: PRINT"(uw bestand heeft";records;" records en
   ";velden;" velden)"
700 PRINT:PRINT:INPUT"hoe moet het bestand heten";bestandsnaam$
710 CLS
720 REM zie regel 365
730 FOR records = 1 TO hrecords
740 velden=hvelden
750 FOR velden = 1 TO hvelden
760 WINDOW SWAP 0
770 WINDOW 1,80,1,8
780 PRINT STRING$(&50,"-")
790 PRINT"bestandsnaam :";bestandsnaam$
800 PRINT"maximaal aantal records :"hrecords," aantal velden";
    hvelden
810 PRINT"nummer van het actuele record :";records
```

```
820 PRINT"nummer van het actuele veld :";velden
830 PRINT STRING$(&50,"-")
840 WINDOW 1,80,9,25
850 CLS
860 PRINT benamingv$(velden);: INPUT""; inhoud$(velden,records)
870 NEXT velden
880 NEXT records
890 LOCATE 1,10
900 INPUT"wilt u veranderingen aanbrengen";veranderingen$
910 IF veranderingen$="n" OR veranderingen$="N" THEN GOTO 90
920 MODE 2
930 PRINT"veranderingen in de inhoud van een veld"
940 PRINT"--------------------------------------"
950 PRINT: INPUT"welk veld wilt u veranderen";velden
960 INPUT"in welke record";records
970 PRINT"oude inhoud van het veld :";inhoud$(velden,records)
980 INPUT"nieuwe inhoud van het veld :";inhoud$(velden,records)
990 INPUT"wilt u nog een verandering aanbrengen";veranderingen$
1000 IF veranderingen$="n" OR veranderingen$="N" GOTO 90 ELSE
     GOTO 920
```

De uitleg:

630-690	titel van dit gedeelte
650+660	de variabelen "record" en "velden" krijgen de oude waarden voor het geval ze veranderd waren
700	bestandsnaam invoeren
720	de variabele "inhoud$" wordt gedimensioneerd en krijgt de gegevens van het bestand
730+750	opent de lussen om gegevens in te lezen
770-830	begin van het invoerdeel, dat u informeert over velden, records, enzovoort
840-880	window; bestandsgegevens worden ingelezen
890-910	beslissing over veranderingen; nee, dan terug naar menu
920-1000	veranderingsproces

Variabelen:

record	bekend
velden	bekend
hrecords	bekend
hvelden	bekend
bestandsnaam	bestandsnaam
inhoud$ (velden,records)	alle bestandsgegevens, tweedimensionale array
veranderingen$	bekend

Verbeteringen:

U kunt voorkomen dat bij de invoer van een record steeds alle eerder ingevoerde velden worden gewist. Ook is het handig als er na elke record naar de veranderingsroutine wordt gesprongen. Deze veranderingen zijn makkelijk uit te voeren. Een laatste tip die voor het hele programma geldt: windows vergemakkelijken het gebruik aanzienlijk.

Handleiding bij dit gedeelte:

Bij het onderdeel INVOEREN vraagt de computer naar de naam van het bestand. Nu komt u in het INVOERMASKER. In de bovenste regels vindt u informatie over naam, maximum aantal velden en de nummers van de actuele velden en records. In het middelste gedeelte voert u (veld na veld) uw gegevens in. De computer slaat steeds veld en recordnummer op. Na de gegevens te hebben ingevoerd kunt u de velden nog veranderen. De computer vertelt u hoe u dit doen moet. Beantwoordt u de vraag of er nog meer veranderingen moeten worden aangebracht met nee dan wordt het menu hersteld.

6.2.4 Bestand bijhouden

Met dit gedeelte houdt u de stand van zaken bij. U kunt records maar ook hele velden veranderen of wissen. Om het bijhouden makkelijk te maken hebben we twee mogelijkheden ingebouwd: direct de plaats invoeren van het gegeven dat moet worden veranderd of in het bestand bladeren en dan beslissen of veranderingen nodig zijn.

```
1010 CLS
1020 PRINT STRING$(&50,"-")
1030 LOCATE 15,3: PRINT"       B E S T A N D   B I J H O U D E N"
1040 PRINT STRING$(&50,"-")
1050 PRINT:PRINT"inhoud van een veld direct veranderen     - 1 -"
1060 PRINT"bladeren en daarna veranderen              - 2 -"
1070 PRINT:INPUT"kies";a
1080 IF a=2 THEN GOTO 1170
1090 CLS
1100 PRINT:PRINT:PRINT:PRINT"veldinhoud direct veranderen"
1110 PRINT:PRINT:PRINT:INPUT"welk veld wilt u veranderen";velden
1120 INPUT"in welke record";records
```

```
1130 PRINT:PRINT:PRINT"oude inhoud van het veld";
     inhoud$(velden,records)
1140 INPUT"nieuwe inhoud van het veld";inhoud$(velden,records)
1150 PRINT:PRINT:INPUT"wilt u nog een verandering aanbrengen";
     veranderingen$
1160 IF veranderingen$="j" OR veranderingen$="J" GOTO 1090 ELSE
     RETURN
1170 CLS
1180 velden = hvelden
1190 records = hrecords
1200 FOR records = 1 TO hrecords
1210 WINDOW 1,80,1,8
1220 PRINT STRING$(&50,"-")
1230 PRINT"actuele bestand :"bestandsnaam$
1240 PRINT"actuele record:"records
1250 PRINT STRING$(&50,"-")
1260 GOSUB 1290
1270 INPUT"wilt u veranderingen aanbrengen";veranderingen$:IF
     LEFT$(veranderingen$,1)="n" OR LEFT$(veranderingen$,1)="N"
     THEN GOTO 1420
1280 GOTO 1350
1290 WINDOW 1,80,9,20
1300 CLS
1310 FOR velden =1 TO hvelden
1320 PRINT benamingv$(velden);":";inhoud$(velden,records)
1330 NEXT velden
1340 RETURN
1350 WINDOW 1,80,21,25
1360 CLS
1370 PRINT STRING$(&50,"-")
1380 INPUT"welk veld wilt u veranderen";velden
1390 INPUT"nieuwe inhoud van het veld";inhoud$(velden,records)
1400 INPUT"wilt u nog een verandering aanbrengen";veranderingen$
1410 IF LEFT$(veranderingen$,1)="j" OR LEFT$(veranderingen$,1)
     ="J" THEN GOTO 1360
1420 NEXT records
1430 RETURN
```

Verklaring:

1010-1060	titel en keuzemogelijkheden van dit gedeelte
1070	keuze van 1 of 2
1080	sprong naar 1 of 2
1090-1140	veranderingsproces door opgave van het nummer van de record en het veldnummer van het gegeven
1150+1160	beslissing of u nog een gegeven wilt veranderen; nee, dan terug naar menu, anders naar begin van subroutine

1170-1250	tekst
1200	opening van lus voor aanduiding van record
1260	sprong naar routine die de velden aanwijst
1290-1330	routine die via een lus benaming en inhoud van recordvelden uitvoert
1340	terug naar 1270
1270	beslissing of iets veranderd moet worden; nee, dan volgende record, ja, dan naar 1350
1350-1400	veranderingsroutine
1400+1410	beslissing of nog een veld in de record moet worden veranderd, anders volgende record

Variabelen:

a	waarde die bepaalt naar welke keuzemogelijkheid moet worden gesprongen
velden	bekend
record	bekend
hvelden	bekend
hrecord	bekend
inhoud$(velden,records)	bekend
veranderingen$	bekend
bestandsnaam$	bekend

Verbeteringen:

Er zijn alleen verbeteringen nodig in het tweede deel. Daar kunt u niet terug naar het menu door een toets in te drukken, maar moet u eerst alle records doorbladeren. Dit is vooral lastig als bekend is welke record moet worden veranderd. Bouw een routine in die controleert of een toets voor de sprong terug naar het menu is ingedrukt (bijvoorbeeld &). Is dit het geval, dan volgt met een simpele GOTO-instructie een sprong terug naar het menu.

Handleiding bij dit gedeelte:

Eerst beslist u of u direct wilt veranderen of eerst wilt bladeren. Direct veranderen kunt u alleen als veld- en gegevensnummer bekend zijn. Anders moet u onderdeel 2 kiezen. Heeft u onderdeel 1 gekozen, dan toetst u eerst het veldnummer in en dan het nummer van de record. Daarna voert het programma de oude veldinhoud uit. Breng de veranderingen aan en beslis daarna of u nog een veld wilt veranderen. Zo niet, dan komt u terug in het menu. Bij onderdeel 2 voert het programma de velden van de eerste record uit. Als u deze wilt veranderen, toetst u het

veldnummer en de nieuwe inhoud in. Daarna verandert u nog een veld of gaat naar de volgende record.

6.2.5 Bestand opslaan

Dit gedeelte van het programma zet de gegevens op band zodat ze niet verloren gaan als u de computer uitschakelt.

```
1440 CLS
1450 PRINT STRING$(&50,"-")
1460 PRINT"            B E S T A N D    O P S L A A N"
1470 PRINT STRING$(&50,"-")
1480 LOCATE 5,10:PRINT"uw bestand wordt opgeslagen, even geduld"
1490 OPENOUT bestandsnaam$
1500 FOR records =1 TO hrecords
1510 FOR velden = 1 TO hvelden
1520 WRITE#9,inhoud$(velden,records)
1530 NEXT velden
1540 NEXT records
1550 CLOSEOUT
1560 RETURN
```

Verklaring:

1440-1470	titel van dit gedeelte
1480	mededeling voor de gebruiker
1490	bestand openen onder de naam van het in de computer aanwezige bestand
1500-1540	lus om de gegevens op band te zetten
1550	bestand sluiten
1560	terug naar menu

Variabelen:

bestandsnaam$	bekend
records	bekend
velden	bekend
hrecords	bekend
hvelden	bekend
inhoud$(velden, records)	bekend

Verbeteringen:

U kunt misschien nog wat verfraaiingen aanbrengen, maar verder is dit gedeelte van het programma in orde.

Handleiding bij dit gedeelte:

De computer vraagt u de toetsen PLAY en RECORD in te drukken. Gebruik daarna een willekeurige toets en het programma doet de rest.

6.2.6 Bestand laden

Met dit onderdeel laadt u eerder opgeslagen bestanden. Het aantal velden en records van het bestand moet bekend zijn, zie paragraaf 6.2.2.

```
1570 CLS
1580 PRINT STRING$(&50,"-")
1590 PRINT"            B E S T A N D   L A D E N"
1600 PRINT STRING$(&50,"-")
1605 INPUT"welk bestand moet geladen worden";bestandsnaam$
1610 LOCATE 5,10:PRINT"uw bestand wordt geladen, even geduld"
1620 OPENIN bestandsnaam$
1630 FOR records =1 TO hrecords
1640 FOR velden = 1 TO hvelden
1650 INPUT#9,inhoud$(velden,records)
1660 NEXT velden
1670 NEXT records
1680 CLOSEIN
1690 RETURN
```

De uitleg:

```
1570-1600  titel van dit gedeelte
1610       mededeling voor de gebruiker
1620       bestand openen om gegevens op band in te lezen
1630-1670  lus om gegevens in te lezen die op band staan
1680       bestand sluiten
1690       terug naar menu
```

Variabelen:

records	bekend
velden	bekend
hrecords	bekend
hvelden	bekend
inhoud$(velden, records)	bekend

Verbeteringen:

U kunt misschien nog wat verfraaiingen aanbrengen, maar verder is dit gedeelte van het programma in orde. Wel verdient het misschien aanbeveling de inleesroutine zo te veranderen dat het aantal velden en records niet bekend hoeft te zijn.

Handleiding bij dit gedeelte:

Spoel de band tot het bestand dat u wilt laden. Druk daarna op PLAY. Elke toets op het toetsenbord regelt vervolgens de rest. Het programma wordt in de computer geladen, waarna het menu wordt hersteld.

6.2.7 Zoeken

Met dit onderdeel zoekt u in het bestand een bepaald gegeven, bijvoorbeeld een naam of een telefoonnummer. Hiertoe voert u het gegeven in plus het aantal plaatsen dat significant (van belang) is. Een klein aantal significante plaatsen heeft het voordeel dat de computer het begrip sneller vindt. Dit is een vrij snelle zoekroutine (voor ongesorteerde gegevens). U moet het getal niet te klein kiezen. Als u bijvoorbeeld slechts 3 plaatsen kiest, zijn voor de computer "Brakman" en "Brandsma" dezelfde personen, omdat de eerste drie letters gelijk zijn. Door vier plaatsen te kiezen, vermijdt u dit.

```
1700 CLS
1710 PRINT STRING$(&50,"-")
1720 PRINT"                              Z O E K E N"
1730 PRINT STRING$(&50,"-")
1740 LOCATE 5,10: INPUT"hoe heet het gezochte gegeven";woord$
1750 LOCATE 5,11: INPUT"hoeveel posities";plaatsen
```

```
1760 PRINT:PRINT:PRINT"even wachten, er wordt gezocht"
1770 PRINT"============================="
1780 zoek$=LEFT$(woord$,plaatsen)
1790 FOR records = 1 TO hrecords
1800 FOR velden = 1 TO hvelden
1810 help$=inhoud$(velden,records)
1820 vind$=LEFT$(help$,plaatsen)
1830 IF zoek$=vind$ THEN GOTO 1880
1840 NEXT velden
1850 NEXT records
1860 INPUT"gegeven niet gevonden, nog een gegeven zoeken ";keuze$
1870 IF LEFT$(keuze$,1)="n" OR LEFT$(keuze$,1)="N" THEN RETURN
     ELSE GOTO 1700
1880 PRINT"gegeven";woord$;"staat in veld";velden;"in record";
     records: INPUT"nog eens zoeken";keuze$
1890 IF LEFT$(keuze$,1)="n" OR LEFT$(keuze$,1)="N" THEN RETURN
     ELSE GOTO 1700
```

De uitleg:

1700-1730	titel van dit gedeelte
1740-1750	informatie inlezen over gezochte begrip
1760-1770	informatie voor de gebruiker
1780	kent significante plaatsen van gezochte begrip toe aan variabele zoek$
1790-1800	opent lus voor zoekproces
1810	met de lus wordt de inhoud van het bestand achter elkaar aan variabele hulp$ toegekend
1820	na elkaar worden de significante plaatsen van de inhoud aan variabele vind$ toegewezen
1830	als zoek$=vind$, is het gezochte begrip gevonden
1840+1850	einde van beide lussen
1860+1870	had de lus geen resultaat, dan is het begrip niet gevonden; de computer voert hier de melding uit; u kunt een nieuw begrip zoeken of teruggaan naar het menu
1880+1890	is het begrip gevonden, dan springt het programma hier naartoe; de computer voert het begrip zelf uit plus het nummer van record en veld; Bovendien kunt u nog een keer zoeken; anders terug naar menu

Variabelen:

begrip$	gezochte begrip
plaatsen	aantal significante plaatsen
zoek	significante plaatsen van gezochte begrip
hulp$	hulpvariabele, waaraan bestandsinhoud achter elkaar

```
          wordt toegekend
vind$     significante plaatsen van gegeven waarmee zoek$ moet
          worden vergeleken
records   bekend
velden    bekend
records   bekend
hvelden   bekend
kies$     bekend
```

Verbeteringen:

De zoekroutine zelf hoeft u niet te veranderen. Die kan ook met grote hoeveelheden gegevens goed overweg en is erg snel voor een BASIC-programma. U zou er wel voor kunnen zorgen dat in plaats van het nummer van de record en het veldnummer direct de totale record met alle velden wordt uitgevoerd. Dit levert geen problemen op, want het nummer van de record en het aantal velden (variabele hvelden) zijn bekend. 1 uitvoerlus is voldoende voor deze verbetering.

Handleiding bij dit gedeelte:

U voert het begrip in dat moet worden gezocht en vervolgens vraagt de computer u het aantal significante plaatsen in te voeren. Als het programma het begrip heeft gevonden, voert het het nummer van de record en het veldnummer uit. Daarna beslist u of u nog iets wilt zoeken (toets j), anders komt u terug in het menu. Als de computer het begrip niet vindt, deelt hij dat mee. U springt nu terug naar het menu of leidt een nieuw zoekproces in. Denk erom, typfouten worden ongenadigd afgestraft. Hiervoor zijn zogenaamde tolerante routines, die bijvoorbeeld bij 90% overeenstemming nog een begrip uitvoeren. Deze programma's vindt u bijna alleen in duurdere bestandsbeheren voor zakelijke toepassingen.

6.2.8 Bestand printen

Printen heeft het voordeel dat als het bestand verloren gaat, de gegevens in ieder geval nog op papier staan.

```
1900 CLS
1910 PRINT STRING$(&50,"-")
```

```
1920 PRINT"          B E S T A N D   P R I N T E N"
1940 PRINT STRING$(&50,"-")
1950 FOR records = 1 TO hrecords
1960 FOR velden = 1 TO hvelden
1970 PRINT#8,inhoud$(velden,records)
1980 NEXT velden
1990 NEXT records
2000 PRINT: INPUT"nog een keer printen";keuze$
2010 IF keuze$="j" OR keuze$="J" THEN GOTO 1950 ELSE RETURN
```

Verklaring:

1900-1930	titel van dit gedeelte
1940	informatie voor de gebruiker
1950-1990	routine voor de uitdraai met lus, PRINT-instructie en printeradres (8)
2000-2010	mogelijkheid voor nog uitdraai; als er op n wordt gedrukt, volgt een sprong naar het menu

Variabelen:

records	bekend
velden	bekend
hrecords	bekend
hvelden	bekend
kies$	bekend

Verbeteringen:

Deze routine levert een ongeformateerde uitdraai. De velden worden een voor een uitgevoerd. Een geformateerde uitdraai kan veel meer, bijvoorbeeld het nummer van de records en het veldnummer, de veldaanduiding enzovoort uitvoeren. Bovendien kunnen er dan alinea's tussen de records worden opgelaten. Met de instructie PRINT USING realiseert u al deze uitbreidingen. Sla het handboek erop na. Met de stringfuncties (right$, ,left$...) kunt u veel werk besparen.

Handleiding bij dit gedeelte:

Na de uitdraai bepaalt u of het bestand nog een keer moet worden geprint. Een ander schriftbeeld behoort ook tot de mogelijkheden. Daartoe dient u een stuurcode naar de printer te zenden voordat u het programma laat draaien. De tekens komen dichter op elkaar te staan met:

```
     PRINT #8;CHR$(15)
```

en verder uit elkaar met:

```
     PRINT #8;CHR$(14)
```

6.2.9 Einde programma

Met dit onderdeel breekt u het programma af. Het programma beëindigt zichzelf.

```
2020 CLS
2030 PRINT STRING$(&50,"-")
2040 PRINT"P R O G R A M M A   B E E I N D I G E N"
2050 PRINT STRING$(&50,"-")
2060 LOCATE 5,10:INPUT"zeker weten?";b$:IF b$="j" OR b$="J"
     THEN END ELSE RETURN
```

Verklaring:

```
2020-2050  titel van dit gedeelte
2060       keuze of programma werkelijk moet worden
           beëindigd; nee, dan terug naar menu
```

Variabelen:

```
b$  beslissing of het programma moet worden beëindigd
```

Verbeteringen:

Niet nodig, verfraaiingen zijn natuurlijk altijd welkom.

Handleiding bij dit gedeelte:

U beslist alleen nog of het programma werkelijk moet worden beëindigd. Deze vraag is ingebouwd om te voorkomen dat u per ongeluk dit gedeelte van het programma kiest en alle opgeslagen gegevens kwijtraakt.

6.2.10 Wissen

Met dit onderdeel wist u gegevens die overbodig zijn.

```
2070 CLS
2080 PRINT STRING$(&50,"-")
2090 PRINT"                         W I S S E N"
2100 PRINT STRING$(&50,"-")
2110 LOCATE 10,10: PRINT"alles wissen                    - 1 -"
2120 LOCATE 10,11: PRINT"record wissen                   - 2 -"
2130 LOCATE 10,14: INPUT"maakt uw keuze ";keuze$
2140 IF keuze$="2" THEN GOTO 2210
2150 CLS
2160 PRINT"alles wissen"
2170 PRINT"------------"
2180 LOCATE 5,10:INPUT"wilt u echt  a l l e s  wissen "; keuze$
2190 IF keuze$="j" OR keuze$="J" THEN CLEAR
2200 GOTO 90
2210 CLS
2220 PRINT"record wissen"
2230 PRINT"-------------"
2240 LOCATE 5,10: INPUT"welke record wilt u wissen ";records
2250 FOR velden = 1 TO hvelden
2260 inhoud$(velden,records)="   "
2270 NEXT velden
2280 LOCATE 5,15: INPUT"wilt u nog een records wissen ";keuze$
2290 IF keuze$="j" OR keuze$="J" GOTO 2210 ELSE RETURN
```

De uitleg:

```
2070-2120  titel van dit gedeelte
2130+2140  keuze tussen beide programma-onderdelen
2150-2200  routine die alle gegevens wist met de BASIC-instructie
           CLEAR; van tevoren wordt gevraagd of u dat echt wel
           wilt
2220-2230  titel van subroutine
2240       invoer van de te wissen record
2250       opening van lus voor de velden
2260       gegevens wissen door toekenning van lege string
2270       e         inde lus
        2280-2290  keuze of er nog een records moet worden ingevoerd;
                   nee, dan terug naar menu
```

Variabelen:

kies$	bekend
record	bekend
velden	bekend
hvelden	bekend
inhoud$(velden, records)	bekend

Verbeteringen:

De subroutine ALLES WISSEN behoeft geen verbeteringen. In het gedeelte RECORDS zou u de mogelijkheid kunnen aanbrengen om selectief een veld van een record te wissen. Dat valt te bereiken door de computer het veldnummer te laten opvragen en dan een lege string toe te kennen aan variabele inhoud$, waarin veld- en recordnummer staan.

Handleiding bij dit gedeelte:

U beslist of u alle gegevens wilt wissen of alleen een record. Kiest u voor ALLES, dan vraagt de computer of u daar zeker van bent. Hierop antwoordt u met N of J. Kiest u voor RECORD, dan moet u het recordnummer invoeren. Als de record is gewist, kunt u nog een andere wissen. Toetst u N in dan komt u weer in het menu.

6.3 Tekstverwerking

6.3.1 Inleiding

Tekstverwerking is een ruim begrip. Tekst in het geheugen opslaan, tekst printen, met stuurcodes verschillende lettertypes op papier zetten - het valt er allemaal onder en dit is nog lang niet alles. Al naar gelang de kwaliteit van het programma zijn de meest geavanceerde manipulaties mogelijk. Wat het ene programma mist, is in het andere doodnormaal. Er zijn op dit gebied nogal wat verschillen. Een ding is in ieder geval zeker: vergeleken met de meeste tekstverwerkingsprogramma's komt een gewone typmachine er maar bekaaid af. Omdat een tekstverwerkingsprogramma haast overal goed van pas komt, leek het ons geen overbodige luxe u er in dit boek nader kennis mee te laten maken. Onze tekstverwerker

is in BASIC geschreven. Een nadeel daarvan is dat dit programma niet zo snel en goed is als bijvoorbeeld TEXTOMAT, WORDSTAR of PERFECT WRITER. Maar om inzicht te verwerven in de fascinerende wereld van de tekstverwerking is het bijzonder geschikt.

6.3.2 Wat u nodig heeft

Erg belangrijk is de manier waarop teksten in het geheugen worden opgeslagen. Er zijn een aantal mogelijkheden, die echter lang niet allemaal even geschikt zijn. Voor de duidelijkheid geven we een puntsgewijs overzicht en behandelen de nadelen.

1 Voor elke tekstregel een stringvariabele gebruiken is niet aan te bevelen, want dan moet u later elke variabele met zijn precieze benaming (bijvoorbeeld variabele 1$) oproepen.

2 Ook veldvariabelen zijn uit den boze, want die moeten afhankelijk van de tekstregel worden geïndexeerd. Dit levert problemen op als een invoeging in een veld consequenties heeft voor de volgende regel. Een veld van een eendimensionale array kan maximaal 256 letters bevatten. Dit bezwaar kan worden weggenomen door een routine die de velden uit elkaar rafelt en vervolgens weer samenstelt. Dat is mogelijk, maar kost veel te veel tijd en geheugenruimte.

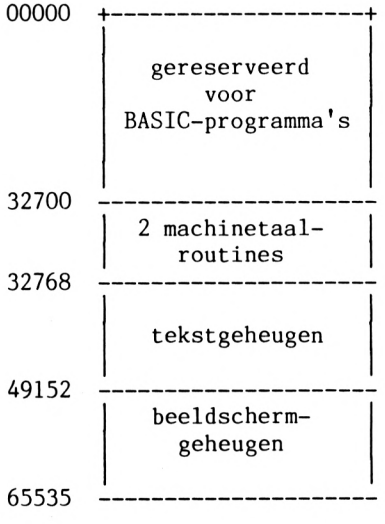
3 In een tweedimensionale matrix past in elk veld precies 1 letter. Het grote nadeel is dat de BASIC-interpreter per letter meerdere bytes nodig heeft voor het beheer. In ons geval is de verhouding 5:1. Dat betekent dat de BASIC-interpreter 10.000 bytes nodig heeft voor een tekst van 33 regels en dat is natuurlijk te dol.

4 In de professionele tekstverwerking gaat het anders. Elke letter die u intypt wordt in een directe geheugenplaats geschreven. Het betreft immers een machinetaalprogramma. U hoeft alleen maar te zorgen dat het bereik in de computer dat u voor de tekst nodig heeft, ook alleen voor de tekst blijft gereserveerd. Op de CPC is dat geen probleem, want er is een instructie voor.

```
1000 REM
1010 REM
1020 REM
1040 ON ERROR GOTO 4170
```

```
1050 MEMORY 32700: MODE 2:rrand=69
1060 storelen=&1000
1070 lrand=10:brand=5:orand=66
1080 paplen=72:papnum=0:blz=1
1090 FOR t=32701 TO 32715: READ a:POKE t,a :NEXT t
1100 FOR t=32721 TO 32735: READ a:POKE t,a :NEXT t
1110 CLS: regel=1:kolom=1: screen=1
1120 store=32768: lostore=store
1130 loline=1: histore=lostore
```

Deze eerste regels bevatten bijna alle belangrijke informatie die u later nodig heeft. In regel 1050 begrenst de MEMORY-instructie het BASIC-geheugen. Na de begrenzing ziet de indeling van het geheugen er zo uit:



Op deze manier beschikt u voor uw tekst na de begrenzing nog vrij over maximaal 16 Kbyte. Om er zeker van te zijn dat het bereik echt vrij is, wist u het van tevoren met de wisroutine in de regels 1230-1270. De wisroutine is een van de langzaamste delen van het programma. Het is daarom handig te weten dat het te wissen bereik verkleind of vergroot wordt met variabelé storelen uit regel 1060. In ons programma is een bereik van $1000 = 4096 tekens voor de tekst gereserveerd. De betekenis van de andere variabelen luidt als volgt:

rrand	rechter tekstrand voor uitdraai en beeldscherm
lrand	linker tekstrand voor uitdraai en beeldscherm
brand	bovenste tekstrand voor de uitdraai
orand	onderste tekstrand voor de uitdraai

paplen	aantal regels per blad
pagina	actueel paginanummer
regel	actuele cursorregel
kolom	actuele cursorkolom
screen	hulpwijzer, die aangeeft in welke regel de cursor staat
store	adres waar een teken wordt opgeslagen
lostore	laagste positie van het tekstgeheugen
histore	hoogste positie van het tekstgeheugen die werd aangesproken
loline	regelnummer van de bovenste op het beeldscherm uitgevoerde tekstregel

Met de twee machinetaalprogramma's uit regel 1090 en 1100 scrollt u de tekst snel over het beeldscherm. De gegevens voor deze routines staan aan het einde van het programma in de regels 4220 en 4230.

6.3.3 Informatie op het scherm

Op het beeldscherm was tot nu toe niets te zien. In deze paragraaf wordt dat rechtgezet, want hier gaat het om het titelmasker. Dit informeert u over alle belangrijke parameters (bijvoorbeeld in welke regel u bent) en is natuurlijk van belang voor de presentatie van het programma.

```
1140 REM opbouw van het beeldschermmasker
1150 REM locate positie in x en y
1160 x=1: y=1: GOSUB 3310
1170 GOSUB 3330
1180 FOR y=2 TO 4: GOSUB 3350: NEXT y
1190 GOSUB 3330
1200 x=5:y=3: GOSUB 3310
1210 PRINT "TEXT ME    1985 Data Becker      Regel:        Kolom:
Pagina:"
1220 GOSUB 3370
1230 LOCATE 30,5:PRINT 32768+storelen;" - "
1240 FOR t=32768 TO 32768+storelen: POKE t,32
1250 LOCATE 40,5:PRINT t:NEXT t: POKEt,ASC("E"):
     POKE t+1,ASC("I"):POKE t+2,ASC("N"):
     POKE t+3,ASC("D"):POKE t+4,ASC("E")
1260 LOCATE 30,5:PRINT STRING$(20,"*")
1270 FOR t =1 TO 3: PRINT CHR$(7): FOR i=1 TO 100: NEXT i: NEXT t
```

1280 GOSUB 3420

De variabelen x en y zijn twee hulpvelden die u gebruikt om de cursor op een bepaalde plaats te zetten. Dat gebeurt in een subroutine (regel 3310 e.v.). Tot en met regel 1210 wordt de titelpagina van het programma op het beeldscherm opgebouwd. De eigenlijke PRINT-instructies staan in subroutines, omdat ze veel worden gebruikt. De subroutine die in regel 3370 begint, geeft boven in het scherm informatie over de plaats waar u zich bevindt. Vanaf regel 1230 richt het programma een aantal geheugenplaatsen in ter grootte van de variabele storelen om de tekst te kunnen opnemen. Tijdens het wisproces kunt u op het scherm zien welke geheugenpositie op dat moment wordt gewist en ook het adres tot waar het tekstgeheugen wordt gereserveerd. Na enige minuten meldt de CPC met een lange pieptoon dat hij klaar is (regel 1270). Om onaangename verrassingen te voorkomen schrijft regel 1250 het woord EINDE aan het eind van het tekstgeheugen.

6.3.4 Nu gaat het gebeuren

In deze paragraaf gaat het om de kern van het programma. Regel 1320 test of er een toets is ingedrukt. Als dit niet is gebeurd, wacht het programma daarop. Is dat gebeurd, dan controleren de regels 1340-1410 of het om een normaal teken gaat of om een stuurfunctie. De volgende stuurfuncties worden direct uitgefilterd:

pijl naar rechts	cursor naar rechts
pijl naar links	cursor naar links
pijl naar beneden	cursor 1 regel lager
pijl naar boven	cursor 1 regel hoger
CLR	teken voor einde alinea uitvoeren, rest van de regel wissen en naar het begin van volgende regel
ENTER	naar begin van volgende regel

Regel 1420-1430 slaat de waarde van de ingedrukte toets (mits een geldig teken) op in het tekstgeheugen en zet het teken op het scherm. De subroutine vanaf regel 3470 zorgt ervoor dat de cursor invers wordt weergegeven.

```
1290 x=lrand: y=7: GOSUB 3310
1300 GOSUB 3470
1310 REM schrijfmodus
1320 toets$=INKEY$
1330 IF toets$= "" THEN GOTO 1320
1340 IF toets$=CHR$(243) THEN GOTO 1530
1350 IF toets$=CHR$(242) THEN GOTO 1560
1360 IF toets$=CHR$(241) THEN GOTO 1640
1370 IF toets$=CHR$(240) THEN GOTO 1710
1380 IF toets$=CHR$(16) THEN GOTO 1780
1390 IF toets$=CHR$(13) THEN GOTO 1840
1400 IF ASC(toets$) <32 THEN GOTO 2360 : REM ctrl codes
1410 IF ASC(toets$) >126 THEN GOTO 1910
1420 POKE store,ASC(toets$)
1430 PRINT toets$;
1440 IF kolom = rrand-lrand-3 THEN PRINT CHR$(7);
1450 x=x+1: kolom=kolom+1: store=store+1
1460 IF store>=histore THEN histore =store
1470 IF x>rrand THEN x=lrand: kolom=1: y=y+1:
     regel=regel+1:screen=screen+1
1480 IF screen>18 THEN screen=18: GOTO 3680
1490 GOSUB 3420: REM posities uitvoeren
1500 GOSUB 3310
1510 GOSUB 3470: REM tekens uitvoeren
1520 GOTO 1310
```

Na de uitvoer van een geldig teken schrijft het programma alle noodzakelijke wijzers en parameters weg. Nu moet het geheugenadres worden verhoogd en de nieuwe kolompositie van de cursor worden bepaald. Het kan zijn dat u het laatste teken in de linker benedenhoek van het beeldscherm heeft geschreven. Dan schuift de beeldscherminhoud een regel naar boven; een nieuwe regel wordt aangemaakt. Vervolgens zet het programma de cursor weer in de linker kolom en brengt de inhoud van variabele histore op een actuele waarde. Tenslotte worden alle nieuwe parameters in het kopje van het programma uitgevoerd. In regel 1520 wordt teruggesprongen naar de uitlezing van het toetsenbord. Een kleine maar nuttige uitbreiding vindt u in regel 1440. Als de computer bijna aan het eind van een regel is en nog maar drie tekens heeft te gaan klinkt er een signaal.

6.3.5 Cursorbesturing

De cursorbesturing is geregeld met de subroutines in de regels 1340-1370. Regel 1600 controleert of de cursor de meest linkse

kolom van het beeldscherm overschrijdt. Is dit het geval, dan verschijnt de cursor in de meest rechtse kolom van de vorige regel. Voordat dit gebeurt, test het programma of er wel een vorige regel is. Zo ja, dan zijn er twee mogelijkheden: de regel staat op het scherm of het scherm moet een regel naar beneden worden gescrolled.

```
1530 REM cursor naar rechts
1540 GOSUB 3550
1550 GOTO 1430
1560 REM cursor naar links
1570 GOSUB 3550
1580 PRINT toets$;
1590 x=x-1: kolom =kolom-1: store =store-1
1600 IF x<1rand THEN x=rrand: y=y-1:regel =regel-1:
     kolom=brand-1rand+1:screen=screen-1
1610 IF regel<1 THEN regel=1: kolom=1: store=32768: x=1rand: y=7:
     screen=1
1620 IF screen<1 THEN screen =1: GOTO 3750
1630 GOTO 1490
1640 REM cursor naar beneden
1650 GOSUB 3550
1660 PRINT toets$; CHR$(8);
1670 y=y+1: regel=regel+1: store=store+(rrand+-1rand+1)
1680 IF store>= histore THEN histore =store
1690 screen=screen+1: IF screen>18 THEN screen =18: GOTO 3680
1700 GOTO 1490
1710 REM cursor naar boven
1720 GOSUB 3550
1730 PRINT toets$;CHR$(8);
1740 regel=regel-1: IF regel<1 THEN regel=1:GOTO 1490
1750 y=y-1: store=store-(rrand-1rand+1)
1760 screen=screen-1:IF screen<1 THEN screen =1:GOTO 3750
1770 GOTO 1490
```

6.3.6 Terug naar het begin

Er zijn twee manieren om aan het begin van een volgende regel komen. Door op ENTER te drukken springt de cursor naar het begin van de volgende regel, CLR wist bovendien de rest van de regel waarin u bezig was. Verder zet CLR een pijl naar links op het

scherm ten teken dat de alinea is afgelopen. Alle latere invoegingen en wisacties hebben steeds alleen betrekking op de alinea waarin ze werden opgeroepen. Ook deze beide programma-onderdelen controleren of er bijzondere gevallen optreden (onderste beeldschermeinde bereikt).

```
1780 REM return uitvoeren
1790 POKE store,13: PRINT CHR$(242);
1800 IF x=rrand THEN GOTO 1450
1810 FOR t=x+1 TO rrand: store =store+1
1820 POKE store,32:PRINT " ";:NEXT t
1830 x=rrand : GOTO 1800
1840 REM return zonder wissen uitvoeren
1850 teken$=CHR$(PEEK(store))
1860 IF teken$=CHR$(13) THEN teken$=CHR$(242)
1870 PRINT teken$;
1880 IF x=rrand THEN GOTO 1450
1890 x=x+1: store=store+1
1900 GOTO 1880
```

6.3.7 Wissen en invoegen

Het programma doorloopt de volgende vier onderdelen, als een toets wordt ingedrukt met een code groter dan 126. Slechts 5 toetsencodes zijn hier van belang.

CTRL en pijl omlaag: regel invoegen

Deze functie zorgt ervoor dat een tekstregel wordt ingevoegd vanaf de actuele cursorpositie. Het beeldscherm wordt vervolgens opnieuw opgebouwd. Dat duurt een paar seconden.

```
1910 REM ctrl+cursor omlaag=regel invoegen
1920 IF ASC(toets$)<>249 THEN GOTO 2020
1930 breedte=rrand-lrand+1: help=(regel-1)*breedte
1940 FOR t=histore TO lostore+help STEP-1
1950 POKE t+breedte,PEEK(t)
1960 NEXT t
1970 FOR t=lostore+help TO lostore+help+breedte-1
1980 POKE t,32: NEXT t
1990 histore=histore+breedte
2000 IF merk=1 THEN merk=0:RETURN
2010 GOTO 2250
```

CTRL en pijl omhoog: regel wissen

Deze functie wist de regel waarin de tekstcursor zich bevindt. De andere regels schuiven op en het getal in variabele histore wordt met 1 regel verlaagd.

```
2020 REM ctrl+cursor omhoog=regel wissen
2030 IF ASC(toets$) <> 248 THEN GOTO 2120
2040 breedte=rrand-lrand+1: help=(regel-1)*breedte
2050 FOR t=lostore+help TO histore
2060 POKE t,PEEK(t+breedte)
2070 NEXT t
2080 FOR t=histore-breedte TO histore
2090 POKE t,32: NEXT t
2100 GOTO 2250
```

CTRL en pijl naar rechts: teken invoegen

Deze functie voegt op de plaats van de cursor een spatie in en verschuift alle andere tekens van de alinea een plaats naar rechts. Let erop dat het begin van de volgende regel wordt gewist als het teken voor einde alinea de rechter rand overschrijdt. Om dit te verhinderen voegt u eerst een lege regel in (CTRL + pijl omlaag).

```
2110 REM letter invoegen
2120 IF ASC(toets$) <> 251 THEN GOTO 2190
2130 FOR t=store TO histore: teken=PEEK(t): IF teken=13 THEN
     GOTO 2150
2140 NEXT t: t=t-1
2150 REM
2160 FOR help=t+1 TO store+1 STEP-1
2170 POKE help,PEEK(help-1):NEXT help
2180 POKE help,32: GOTO 2240
```

CTRL en pijl naar links)
DEL) teken wissen

Beide opties wissen het teken op de plaats van de cursor. Alle andere tekens van de alinea schuiven een plaats naar links.

```
2190 REM letter wissen
2200 IF ASC(toets$) <> 250 AND ASC(toets$) <>127 THEN GOTO 1310
2210 FOR t=store TO histore
2220 POKE t,PEEK(t+1): IF PEEK(t)=13 THEN POKE t+1,32: GOTO 2240
2230 NEXT t: POKE t,32
2240 breedte=rrand-lrand+1: help=(regel-1)*breedte
```

De volgende subroutine heeft een functie in elk van de 4 besproken routines, omdat na elke verschuiving het beeldschermgeheugen moet worden aangepast. Deze subroutine bouwt het beeldscherm opnieuw op vanaf de actuele cursorregel. Daarom is het raadzaam invoegingen of wisacties dichtbij de onderste beeldschermrand uit te voeren.

```
2250 REM gedeelten van scherm uitvoeren
2260 helpy=y
2270 FOR t=screen TO 18
2280 LOCATE lrand,helpy
2290 FOR i=lostore+help TO lostore+help+breedte-1
2300 teken$=CHR$(PEEK(i))
2310 IF teken$=CHR$(13) THEN teken$=CHR$(242)
2320 PRINT teken$;: NEXT i
2330 PRINT: help=help+breedte
2350 helpy=helpy+1: NEXT t:GOTO 1490
```

6.3.8 De functies

Deze programma-onderdelen roept u op met CTRL. Hiervoor hoeft u de schrijfmodus niet te verlaten. Deze functies lijken gecompliceerder dan ze zijn.

CTRL en X: programma beëindigen

Met deze functie verlaat u het programma. Eerst vraagt het programma of u TEXT ME echt wilt verlaten. Als u hierop n (=nee) antwoordt, wordt de normale schrijfmodus hersteld.

```
2360 REM controle op crtl codes
2370 IF ASC(toets$)<>24 THEN GOTO 2410 :
     REM ctrl+x: einde programma
2380 LOCATE 25,5:PRINT" TEXT ME beëindigen (j/n)  "
2390 a$=INKEY$: IF a$="n" OR a$="N" THEN LOCATE 25,5:
     PRINT STRING$(30,"*"):GOTO 1490
2400 IF a$="j" OR a$="J" THEN END ELSE GOTO 2390
```

CTRL en B: bovenste bladspiegelrand instellen

De bovenste marge is van belang al u gaat printen. Pas na die lege ruimte verschijnen er letters op papier.

```
2410 IF ASC(toets$) <> 2 THEN GOTO 2460:
     REM ctrl+b: bovenrand
2420 LOCATE 20,5: PRINT"bovenrand is ";brand;:
     INPUT"nieuwe waarde :";brand$
2430 IF brand$="" OR VAL(brand$)<1 OR VAL(brand$)=orand
     THEN GOTO 2450
2440 brand=VAL(brand$)
2450 LOCATE 20,5:PRINT STRING$(50,"*"):GOTO 1490
```

CTRL en O: onderste bladspiegelrand instellen

Zie boven. Waarden kleiner dan de bovenrand en groter dan de paginalengte zijn onmogelijk. Door op ENTER te drukken verlaat u beide functies zonder dat de waarde is veranderd.

```
2460 IF ASC(toets$) <> 15 THEN GOTO 2510:
     REM ctrl+o: onderrand
2470 LOCATE 20,5:PRINT"onderrand is: ";orand;:
     INPUT" nieuwe waarde :",orand$
2480 IF orand$="" OR VAL(orand$)>paplen OR VAL(orand$)<=brand
     THEN GOTO 2500
2490 orand=VAL(orand$)
2500 GOTO 2450
```

CTRL en Z: papierlengte (aantal regels) bepalen

Met deze functie verandert u het maximale aantal regels dat u op een vel papier kwijt kunt. Gewoonlijk kiezen we de letter die overeenkomt met de functie, maar de P is al gereserveerd. Vandaar hier een Z. 72 regels is de lengte van het gebruikelijke formaat computerpapier. Door op ENTER te drukken verandert u niets.

```
2510 IF ASC(toets$)<>26 THEN GOTO 2550:
     REM ctrl+z: papierlengte
2520 LOCATE 20,5:PRINT"papierlengte is:";paplen;:
     INPUT"nieuwe waarde :",paplen$
2530 IF paplen$="" OR VAL(paplen$)<orand THEN GOTO 2450
2540 paplen=VAL(paplen$):GOTO 2450
```

CTRL en S: tekst opslaan (saven)

De routine vanaf regel 2550 slaat de tekst op. Het programma vraagt u een naam voor de tekst in te voeren, waarna het de tekst onder deze naam op schijf of band zet. Het uitroepteken dat aan de filenaam voorafgaat voorkomt de systeemmeldingen die optreden bij de opslag van gegevens.

```
2550 IF ASC(toets$) <>19 THEN GOTO 2610:
     REM ctrl+s: opslaan
2560 LOCATE 20,5: INPUT"tekst opslaan! tekstnaam :",tekstnaam$
2570 IF LEN(tekstnaam$)<1 OR LEN(tekstnaam$)>8 THEN LOCATE 47,5:
     PRINT"**************" : GOTO 2560
2580 lengte=histore-lostore
2590 SAVE"!"+tekstnaam$+".txt",b,lostore,lengte
2600 GOTO 2450
```

CTRL en L: tekst laden

Met deze routine (tegenovergesteld aan de vorige) laadt u de tekst en kunt u er verder mee werken. Ook hier vraagt het programma naar de filenaam. Het laadproces vernietigt alle andere teksten in het computergeheugen, waarna het beeldscherm weer wordt opgebouwd.

```
2610 IF ASC(toets$) <> 12 THEN GOTO 2660:REM ctrl+l: laden
2620 LOCATE 20,5: INPUT "tekst inlezen ! tekstnaam :",tekstnaam$
2630 IF LEN(tekstnaam$)<1 OR LEN(tekstnaam$)>8 THEN LOCATE 47,5:
     PRINT"**************" : GOTO 2450
2640 LOAD"!"+tekstnaam$+".txt"
2650 GOTO 3000
```

CTRL en N: paginering aan/uit

Na CTRL/N te hebben ingedrukt, schakelt u heen en weer door op een lettertoets te drukken. U verlaat dit onderdeel met ENTER. Het paginanummer begint bij elke uitdraai met 1 en staat midden onder de laatste tekstregel. De opties kunt u veranderen in het programma-onderdeel PRINTEN.

```
2660 IF ASC(toets$) <> 14 THEN GOTO 2720 : REM ctrl+n: paginering
     aan/uit
2670 LOCATE 30,5:PRINT"paginanr. = ";:IF papnum=0
     THEN PRINT"uit"ELSE PRINT"aan"
2680 a$=INKEY$: IF a$="" THEN GOTO 2680
2690 IF a$=CHR$(13) THEN GOTO 2450
2700 papnum=papnum+1: IF papnum>1 THEN papnum=0
2710 GOTO 2670
```

CTRL en U: tekst uitdraaien

Met deze instructie maakt u in dit programma een uitdraai van een tekst op de printer. De breedte van de bladspiegel is daarbij gelijk aan het actuele beeldschermformaat. Lay-out op scherm en papier wijken dus niet van elkaar af. De subroutine waar in dit

programma-onderdeel naartoe wordt gesprongen, schuift het blad door tot aan de eerste printregel. Die is afhankelijk van de lengte van de bovenste rand. Regel 2880 bekijkt of de paginering actief is en of de regel is bereikt waarin het nummer moet komen te staan. Het paginanummer staat in variabele pagenum. De routine zet dit bij elke oproep terug naar 1 (regel 2860).

```
2720 IF ASC(toets$) <> 21 THEN GOTO 2920: REM ctrl+u: uitdraai
2730 GOSUB 3670
2740 breedte=rrand-lrand
2750 FOR index1=lostore TO histore STEP breedte+1
2760 pregel$=STRING$(lrand-1," ")
2770 FOR index2= 0 TO breedte
2780 pregel$=pregel$+CHR$(PEEK(index1+index2)):NEXT index2
2790 PRINT #8,pregel$:pregel=pregel+1
2800 IF pregel=orand THEN GOSUB 2850: GOSUB 3670
2810 NEXT index1
2820 FOR t=pregel+1 TO orand: PRINT #8,CHR$(13):NEXT t
2830 GOSUB 2850
2840 GOSUB 2450
2850 REM onderste regels opschuiven
2860 pagenum=1
2870 FOR t=orand+1 TO paplen: d$=CHR$(13)
2880 IF pagnum=1 AND t=orand+1 THEN d$=STRING$(lrand-2+INT
     ((rrand-lrand)/2)," ")+STR$(pagenum):pagenum=pagenum+1
2890 PRINT #8,d$
2900 NEXT t
2910 RETURN
```

CTRL en R: rechter kantlijn instellen

Met deze functie stelt u de rechter rand van uw bladspiegel in. De bladspiegel verschijnt overeenkomstig uw instelling direct op het beeldscherm. Ook deze routine kunt u met ENTER zonder veranderingen verlaten.

```
2920 IF ASC(toets$) <> 18 THEN GOTO 2960:
     REM ctrl+r: rechter kantlijn
2930 LOCATE 20,5:PRINT"rechter kantlijn oud :";rrand;:
     INPUT"nieuwe waarde :",rrand$
2940 IF rrand$="" OR VAL(rrand$)<lrand OR VAL(rrand$)>79
     THEN GOTO 2450
2950 rrand=VAL(rrand$): GOTO 3000
```

CTRL en Q: linker kantlijn instellen

Zie boven. We moesten onze toevlucht nemen tot de letter Q omdat

de L al is gereserveerd voor LADEN.

```
2960 IF ASC(toets$) <>11 THEN GOTO 3010:
     REM ctrl+q: linker kantlijn
2970 LOCATE 20,5:PRINT"linker kantlijn oud :";lrand;:
     INPUT"nieuwe waarde :",lrand$
2980 IF lrand$="" OR VAL(lrand$)<1 OR VAL(lrand$)>=rrand
     THEN GOTO 2450
2990 lrand=VAL(lrand$)
3000 GOSUB 3590:LOCATE 20,5:PRINT STRING$(40,"*"):x=lrand:y=7:
     GOSUB 3310:GOTO 1490
```

CTRL en H: HELP-tekst oproepen

Door deze toetsen in te drukken krijgt u een lijst van alle belangrijke CTRL-functies op het scherm. Het programma keert in de schrijfmodus terug als u op een willekeurige toets drukt. De HELP-teksten staan in de subroutine vanaf regel 3850.

```
3010 IF ASC(toets$) <> 8 THEN GOTO 3060: REM ctrl+h: HELP-tekst
3020 WINDOW #4,lrand,rrand,7,24
3030 CLS #4
3040 GOSUB 3820
3050 GOTO 3000
```

CTRL en W: tekstgeheugen wissen

Met deze functie wist u de tekst in het geheugen. Omdat het vervelend is als dit per ongeluk gebeurt, vraagt het programma eerst of u dat echt wel wilt.

```
3060 IF ASC(toets$) <>23 THEN GOTO 3120:
     REM ctrl+c: tekst wissen
3070 LOCATE 20,5:PRINT "tekst wissen! zeker weten? (j/n)"
3080 a$=INKEY$: IF a$="" THEN GOTO 3080
3090 IF a$="j" OR a$="J" THEN 3110
3100 IF a$="n" OR a$="N" THEN LOCATE 20,5:PRINT STRING$(40,"*"):
     GOTO 1490 ELSE 3080
3110 storelen=histore-lostore: GOTO 1110
```

CTRL en I: inhoudsopgave (catalogus)

Met deze functie krijgt u de directory van een schijf op het scherm. Elke toets herstelt de schrijfmodus.

```
3120 IF ASC(toets$) <> 9 THEN  GOTO 3200: REM ctrl+i: directory
```

```
3130 WINDOW #4,1,80,7,24
3140 CLS #4
3150 CAT
3160 PRINT #4,"druk op een toets!"
3170 a$=INKEY$: IF a$="" THEN GOTO 3170
3180 CLS #4: WINDOW 1,80,1,25
3190 GOTO 3000
```

CTRL en A: naar begin van venster springen

Met deze routine springt de cursor naar de linker bovenhoek van het actuele beeldschermvenster (de HOME-positie), dus niet naar het begin van de tekst. De B is al gereserveerd en daarom kozen we de A. In verband met het begrip "begin" kunt u dat waarschijnlijk toch makkelijk onthouden.

```
3200 IF ASC(toets$) <> 1 THEN GOTO 1310:
     REM ctrl+a: begin van actuele beeldschermvenster
3210 teken$=CHR$(PEEK(store))
3220 IF teken$=CHR$(13) THEN teken$=CHR$(242)
3230 PRINT teken$;
3240 IF screen=1 THEN GOTO 3280
3250 screen=screen-1: store=store-(rrand-lrand+1)
3260 regel=regel-1:y=y-1
3270 GOTO 3240
3280 IF kolom=1 THEN GOTO 1490
3290 kolom=kolom-1 :store=store-1
3300 x=x-1:GOTO 3280
```

6.3.9 Subroutines

In deze paragraaf vindt u nog een aantal subroutines. Ze zijn allemaal voorzien van commentaar. Om het programma op eenvoudige wijze uit te breiden kunt u het beste nog wat meer CTRL-functies definiëren. De kleine routines in machinetaal schrijven is overigens ook niet moeilijk en het levert flink wat tijdwinst op.

```
3310 REM cursorpositie kiezen
3320 LOCATE x,y: RETURN
3330 REM sterretjes
3340 PRINT STRING$(80,"*");: RETURN
3350 REM sterretjes
```

```
3360 LOCATE 1,y: PRINT"*";:LOCATE 80,y:PRINT"*";:RETURN
3370 REM papierbreedte aangeven
3380 LOCATE 1,6:PRINT STRING$(80," ")
3390 FOR a=lrand TO lrand+((rrand-lrand)/2):LOCATE a,6:
     PRINT"<";:NEXT a
3400 FOR a=lrand+((rrand-lrand)/2) TO rrand:LOCATE a,6:
     PRINT">";:NEXT a
3410 RETURN
3420 REM regel en kolom
3430 LOCATE 44,3: PRINT regel
3440 LOCATE 58,3: PRINT kolom
3450 LOCATE 71,3: PRINT INT(regel/(orand-brand))+1
3460 RETURN
3470 REM teken inlezen en printen
3480 teken =PEEK(store)
3490 IF teken =13 THEN teken =242
3500 PRINT CHR$(24);:REM reverse inschakelen
3510 PRINT CHR$(teken);: REM teken onder cursor simuleren
3520 PRINT CHR$(24);: REM reverse uitschakelen
3530 PRINT CHR$(8);: REM backspace functie
3540 RETURN
3550 REM TEXT PEEK-functie
3560 toets$=CHR$(PEEK(store))
3570 IF toets$=CHR$(13) THEN toets$=CHR$(242)
3580 RETURN
3590 REM opbouw van een beeldschermpagina na ctrl R, ctrl L
3600 WINDOW #2,1,80,6,24:CLS #2: WINDOW 1,80,1,24
3610 GOSUB 3370
3620 FOR index1=loline TO loline+17
3630 LOCATE lrand,index1+6: FOR index2=lostore+(index1-1)*
     (rrand-lrand+1) TO lostore+(index1*(rrand-lrand+1)-1)
3640 teken$=CHR$(PEEK(index2)):IF teken$=CHR$(13)
     THEN teken$=CHR$(242)
3650 PRINT teken$;:NEXT index2: NEXT index1
3660 regel=1:kolom=1:screen=1:store= lostore: RETURN
3670 pregel=0:FOR t=1 TO brand:PRINT#8,CHR$(13):pregel=pregel+1:
     NEXT t:RETURN
3680 REM scrollroutine (omlaag)
3690 CALL 32701
3700 y=y-1 : LOCATE lrand,24: FOR t=lostore+(regel-1)*
     (rrand-lrand+1) TO lostore+(regel*(rrand-lrand+1)-1)
3710 teken$=CHR$(PEEK(t))
3720 IF teken$=CHR$(13) THEN teken$=CHR$(242)
3730 PRINT teken$;:NEXT t
3740 GOTO 1490
3750 REM scrollroutine (omhoog)
3760 CALL 32721
3770 y=y+1:LOCATE lrand,7: FOR t=lostore+(regel-1)*
     (rrand-lrand+1) TO lostore+(regel*(rrand=lrand+1)-1)
```

```
3780 teken$=CHR$(PEEK(t))
3790 IF teken$=CHR$(13) THEN teken$=CHR$(242)
3800 PRINT teken$;:NEXT t
3810 GOTO 1490
3820 REM HELP-tekst printen
3830 PRINT #4,"* * * HELP-tekst: lijst van CTRL-functies * * * "
3840 PRINT #4
3850 PRINT #4,"CTRL-L  tekst laden"
3860 PRINT #4,"CTRL-S  tekst saven"
3870 PRINT #4,"CTRL-U  tekst uitdraaien"
3880 PRINT #4,"CTRL-R  rechter kantlijn instellen"
3890 PRINT #4,"CTRL-Q  linker kantlijn instellen"
3900 PRINT #4,"CTRL-B  bovenrand instellen"
3910 PRINT #4,"CTRL-O  onderrand instellen"
3920 PRINT #4,"CTRL-Z  bladlengte instellen"
3930 PRINT #4,"CTRL-N  bladzijdenummer aan/uit"
3940 PRINT #4,"CTRL-A  naar begin van actueel venster"
3950 PRINT #4,"CTRL-W  tekst wissen"
3955 PRINT #4,"CRTL-H  HELP-tekst op scherm"
3960 PRINT #4,"CTRL-X  einde programma"
3970 PRINT #4,"CTRL-I  inhoudsopgave van schijf"
3980 PRINT #4
3990 PRINT #4,"druk op een toets"
4000 a$=INKEY$: IF a$="" THEN GOTO 4000
4010 CLS #4: WINDOW 1,80,1,25
4020 RETURN

4100 REM op deze plaats stond een subroutine voor de Duitse
     karakterset; daarom valt hier een gat

4170 REM error routine
4180 LOCATE 15,6:PRINT" nummer van fout :";ERR;"in regel :";ERL
4190 a$=INKEY$: IF a$ <> CHR$(13) THEN GOTO 4190
4200 LOCATE 20,6:PRINT STRING$(40,"*")
4210 RESUME
4220 DATA &6,&1,&3e,&0,&21,&6,&0,&11,23,79,&cd,&50,&bc,&c9,&0
4230 DATA &6,&0,&3e,&0,&21,&6,&0,&11,23,79,&cd,&50,&bc,&c9,&0
```

6.4 Vang de bom

Spelletjes - wie heeft er nou een homecomputer en geen bandje of schijf met spelletjes? U kunt ze kant en klaar in de winkel kopen, maar het is eigenlijk nog veel leuker om ze zelf te maken.

In dit boek presenteren we er twee. Het eerste spel, VANG DE BOM, is kort maar niet zo eenvoudig als het lijkt. De opdracht is met een wagen in de onderste beeldschermrand bommen op te vangen. Elke gevangen bom betekent een punt, elke misser een punt eraf. De valsnelheid en het aantal bommen stelt u in aan het begin van het spel. Met de cursorbesturing beweegt u de wagen naar links en rechts. De twee spaties na de PRINT-instructie in regel 180 zijn uitermate belangrijk. De beste snelheid om de techniek een beetje onder de knie te krijgen is 0.5. Verlaat het programma niet met ESC. We hebben er namelijk met KEY SPEED voor gezorgd dat de toetsen vaker repeteren; daardoor zou u problemen krijgen bij het invoeren van een instructie. Als u kort op een toets drukt, wordt het teken meerdere malen op het beeldscherm uitgevoerd. Een tip voor waaghalzen: verander regel 140 in:

```
140 D=1:BEG=INT(RND(1)*30)
```

en regel 240 in:

```
240 IF KO >30 THEN KO=30
```

Bommen vangen wordt dan pas echt een lastig karwei. Het achtergrondplaatje verbeteren, aardige geluiden invoeren of de ontploffing van een bom met een paar zelf gedefinieerde tekens wat spectaculairder maken draagt natuurlijk altijd bij tot meer spelplezier. Dergelijke tekens kunt u met de LOCATE-instructie over het beeldscherm bewegen.

```
10 REM vang de bom
20 MODE 1
30 SYMBOL AFTER 57
40 SYMBOL 58,128,192,192,255,255,201,28,8
50 SYMBOL 59,1,3,3,255,255,147,58,16
60 SYMBOL 60,4,8,28,62,127,127,62,28
70 SYMBOL 61,255,227,255,255,255,255,255,255
80 INPUT"valsnelheid ";sn
90 INPUT"aantal bommen";bo
100 SPEED KEY 1,1
110 CLS
120 ko=10
130 FOR bommen=1 TO bo
140 CLS
150 LOCATE 1,22:PRINT STRING$(&28,"=")
160 d=1:beg=INT(RND(1)*20)+1
170 LOCATE beg,d:PRINT" "
180 LOCATE ko,21:PRINT":;"
190 LOCATE beg,d+sn:PRINT"<"
200 LOCATE ko,21:PRINT"  "
210 d=d+sn
```

```
220 IF INT(d)>=20 THEN GOTO 280
230 in$=INKEY$
240 ko=ko+(1 AND in$=CHR$(243))-(1 AND in$=CHR$(242))
250 IF ko<1 THEN ko=1
260 IF ko>=25 THEN ko=25
270 GOTO 170
280 IF ko=beg OR beg=ko+1 THEN tr=tr+1
290 IF ko<>beg AND ko<>beg-1 THEN tr=tr-1
300 NEXT
310 CLS
320 SPEED KEY 20,3
330 LOCATE 15,10:PRINT"punten: ",tr
340 LOCATE 10,20:PRINT"nog een keer ?"
350 INPUT a$
360 IF a$="j" OR a$="J" THEN RUN
370 SPEED KEY 20,3
```

6.5 Zeeslag anno 1985

Dit spel is niet de zoveelste variatie op het oude thema. In het eerste deel van het programma moet u een aantal parameters invoeren. Allereerst de moeilijkheidsgraad: die loopt van 0 tot 2. We gebruiken de verschillende modi voor het aantal tekens (20-40-80) van de CPC. Een 0 staat voor 20 tekens per regel. Het plaatje ziet er op deze manier het beste uit, omdat de x-omvang van een teken relatief groot is. De moeilijkheidsgraad is echter gering. Bij 80 tekens heeft u het het moeilijkst. De tweede parameter betreft de tijd die u heeft om het spel te spelen. Hoe kleiner u deze kiest, des te minder tijd en kans heeft u om de opdracht tot een goed einde te brengen. De derde parameter is de snelheid van de bommen en de laatste het aantal bommen dat u ter beschikking staat. Op het scherm ziet u vervolgens een groot schip, compleet met brug en kanon en verder een duikboot met daarnaast een bom. Het is de bedoeling de bom met de cursortoetsen naar de boeg van het schip te sturen en het naar de kelder te laten gaan. Hiervoor krijgt u een punt. Raakt u het schip niet binnen de beschikbare tijd, dan begint het programma opnieuw. Na afloop staan uw prestaties en de vaste parameters op het scherm: het aantal treffers, de tijd die u nodig had, de modus en de snelheid van de bommen. U kunt dit spel naar eigen inzicht met kleur en geluid uitbreiden.

```
10 MODE 2
```

```
15 treffer=0
20 SYMBOL AFTER 32
30 LOCATE 20,10:PRINT"Z E E S L A G  A N N O  1 9 8 5"
40 GOSUB 350
50 LOCATE 1,15:INPUT"welke...";a
60 INPUT"hoeveel tijd om de bom te besturen";b
70 INPUT"snelheid van de bommen;snel
80 INPUT"hoeveel bommen;bommen
85 DIM bo(bommen)
90 MODE a
95 FOR bb=1 TO bommen
96 CLS
100 IF a=0 THEN f=18
110 IF a=1 THEN f=38
120 IF a=2 THEN f=78
130 yp=INT(RND(1)*24)+2
140 xp=INT(RND(1)*f)+1
150 yp2=INT(RND(1)*25)+1
160 xp2=INT(RND(1)*f)+1
170 IF xp=xp2 OR yp=yp2 THEN GOTO 130
171 xp3=xp2+2:yp3=yp2
172 b1=b*100
175 SPEED KEY 1,snel
176 FOR mnn=1 TO b1
180 LOCATE xp+2,yp-1:PRINT"!"
190 LOCATE xp,yp:PRINT"&%#"
200 LOCATE xp2,yp2:PRINT"[]"
220 LOCATE xp3,yp3:PRINT"
230 in$=INKEY$
240 xp3=xp3+(1 AND in$=CHR$(243))-(1 AND in$=CHR$(242))
250 yp3=yp3+(1 AND in$=CHR$(241))-(1 AND in$=CHR$(240))
260 IF xp3<1 THEN xp3=1
270 IF xp3>f+2 THEN xp3=f+2
280 IF yp3<1 THEN yp3=1
290 IF yp3>25 THEN yp3=25
300 IF yp3=25 AND xp3=f+2 THEN xp3=f+1
310 LOCATE xp3,yp3:PRINT^"
320 IF xp3=xp AND yp3=yp THEN GOTO 440
330 NEXT mnn
335 NEXT bb
340 GOTO 3000
350 REM karakters definiëren
360 SYMBOL 33,62,8,28,28,124,124,124,252
370 SYMBOL 35,252,252,252,255,254,252,248,240
380 SYMBOL 37,1,129,129,255,255,255,255,255
390 SYMBOL 38,126,15,15,127,63,15,3,1
400 SYMBOL 91,128,192,252,254,255,255,254,0
410 SYMBOL 93,0,1,31,63,255,255,63,0
420 SYMBOL 94,0,0,156,190,127,190,156,0
```

```
430 RETURN
440 MODE 0
445 treffer = treffer+1
446 LOCATE 3,12
447 bo(bb)=((b1-mnn)/b1)*10
448 FOR tijd=1 TO 15
449 PRINT CHR$(224);
450 NEXT
451 LOCATE 3,13:PRINT CHR$(224);"J O E P I E  R A A K";CHR$(224)
452 LOCATE 3,14
453 FOR tijd =1 TO 15
454 PRINT CHR$(224);
455 NEXT
460 SOUND 129,450,150,15,,,15
465 FOR xx=1 TO 3000
466 NEXT xx
467 MODE a
470 GOTO 335
3000 MODE 2
3005 FOR ii=1 TO bommen
3006 punten=punten+bo(ii)
3007 NEXT
3008 punten=punten*treffer*(a+1)
3010 LOCATE 15,9:PRINT"E I N D E  V A N  H E T  S P E L"
3020 LOCATE 2,11:PRINT"u hebt met;bommen;" bommen;treffers;"
treffers gemaakt."
3021 LOCATE 2,13:PRINT"totaal aantal punten:";punten
3025 SPEED KEY 20,3
3030 LOCATE 2,15:INPUT"nog een keer?:";keuze$
3040 IF keuze$="J" OR keuze$="j" THEN RUN
3050 END
```

Een boek vol met informatie over de werking, opbouw en mogelijkheden van de CPC-computers. Diverse listings verduidelijken wat er in de tekst besproken is.

Een kleine greep uit de inhoud:
tekenen met de joystick * het gebruik van windows * geluids-editor * makkelijk te bedienen karaktergenerator * listings van diverse spelletjes.

Haal alles uit uw CPC!

De listings zijn geschikt voor de CPC-464, 664 en 6128

ISBN 90 229 3344 X

CPC Bibliotheek 5

DATA BECKER
NEDERLANDS

CPC85

DATA BECKER
NEDERLANDS*

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/