MSX LEERBOEK BASIC DEEL 2

WESSEL AKKERMANS/PIET DEN HEIJER

MSX LEERBOEK BASIC DEEL 2 AKKERMANS/DEN HEIJER

MSX BASIC leerboek
deel 2

MSX LEERBOEK BASIC DEEL 2

WESSEL AKKERMANS/PIET DEN HEIJER

uitgeverij STARK - TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Akkermans, Wessel

MSX leerboek / Wessel Akkermans, Piet den Heijer. – Oosterend: Stark-Texel
Dl. 2
ISBN 90 6398 769 2
SISO 365.3 UDC 681.3
Trefw.: MSX (computer)

. . . . . .

1e druk 1985
ISBN 90 6398 769 2

VOORWOORD

Bij de behandeling van de statements en functies in dit boek, zijn we er vanuit gegaan dat u de statements en functies uit het eerste deel van deze serie leerboeken al heeft bestudeerd en in de praktijk uitgeprobeerd. Er wordt slechts sporadisch terugverwezen naar de stof uit het eerste leerboek.

Een andere aanname die we voor dit boek hebben gedaan is, dat u alleen een basis-computer met cassetterecorder hebt, eventueel aangevuld met een printer. De floppy-disk komt in dit boek in het geheel niet ter sprake. De behandeling daarvan zal in het derde deel uit deze serie leerboeken aan de orde komen.

De nadruk ligt in dit boek op het toepassen van grafische- en geluidsmogelijkheden van de MSX-computer. Hiertoe zijn de beide macrotalen (de grafische macrotaal en de macrotaal voor het geluid) uitgebreid behandeld. Daarnaast is echter ook zeer uitgebreid aandacht besteed aan het maken van grafische voorstellingen met statements als CIRCLE, LINE, enz., en aan het rechtstreeks aansturen van de geluidsprocessor en de videoprocessor.

Een belangrijke inleiding in machinetaal is ook opgenomen. Het doel van deze, twee hoofdstukken innemende, uitleg is, u duidelijk te maken wat machinetaal is, en welke begrippen daarbij een rol spelen. Het is geenszins de bedoeling u te leren programmeren in machinetaal. Er zijn voldoende boeken

op de markt waaruit u kunt leren hoe u een Z80-microprocessor moet programmeren. Iedere computer heeft echter zijn eigen manier van aansturen van machinetaalprogramma's en wij hopen de specifieke MSX-manier duidelijk te hebben gemaakt.

In de appendici, tenslotte, vindt u een aantal grotere programma's. Deze programma's hebben een dubbele functie. Ten eerste dienen ze als voorbeeld van de mogelijkheden die u hebt na het lezen van dit boek. Ten tweede zijn het kant en klare toepassingsprogramma's. Door de door u opgedane kennis zult u echter in staat zijn om deze programma's aan uw eigen wensen aan te passen, of om ze naar eigen inzicht verder uit te breiden. Om u daarbij behulpzaam te zijn, hebben we de programma-listings van uitgebreid commentaar voorzien.

Net als in het eerste deel van deze serie leerboeken, is ook dit deel weer gecompleteerd met een trefwoordenlijst. Wij hopen daarmee van dit boek, nog tot lang nadat u het hebt bestudeerd, een waardevol naslagwerk te hebben gemaakt.

Om u verder behulpzaam te zijn bij het bestuderen, maar vooral bij het in praktijk brengen van de opgedane kennis, is bij dit boek weer een apart vragen en opdrachten boekje verkrijgbaar. Van alle vragen en opdrachten zijn in dat boekje ook de antwoorden en uitwerkingen gegeven. Mocht u zelf niet voldoende ideeen hebben voor praktijkoefeningen, dan is dit boekje zeker aan te raden.

De aard van de in dit boek behandelde statements en functies maakt het eigenlijk overbodig, maar wij wensen u toch veel plezier met het schrijven van programma's voor en het spelen met uw MSX-computer.

November 1985

De auteurs

INHOUDSOPGAVE

1 Standaard MSX-functies en eigen functies

We zullen dit deel beginnen met de behandeling van een aantal standaard functies. Standaard functies zijn functies, die niet in een programma behoeven te worden gedefinieerd, maar reeds in MSX-BASIC zijn ingebouwd. Wanneer bijvoorbeeld in een programma de functie SQR(25) voorkomt, dan weet de computer precies wat dit betekent. We behoeven dus niet zelf te programmeren hoe de vierkantswortel moet worden berekend. We gebruiken daarvoor gewoon de functie SQR. De standaard functies die in dit hoofdstuk zullen worden behandeld zijn:

ABS
Hiermee wordt de absolute waarde van een getal bepaald.

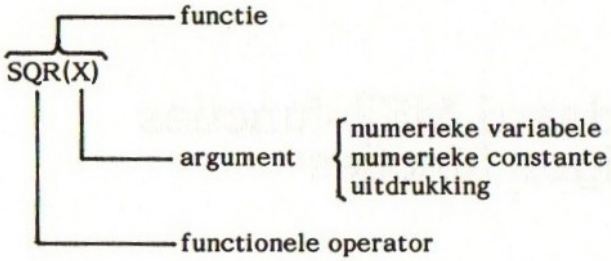
FIX
Hiermee worden alle cijfers, die achter de decimale punt (komma) van een getal staan, verwijderd.

SGN
Hiermee wordt het teken van een getal bepaald.

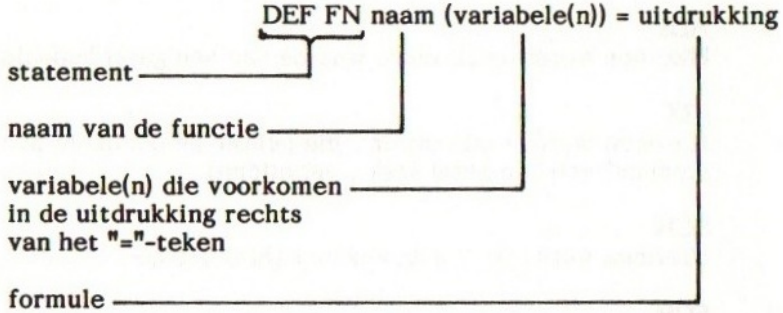
SQR
Hiermee wordt de vierkantswortel van een getal bepaald.

RND
Hiermee kunnen willekeurige getallen worden gegenereerd.

Het algemene formaat van een functie is:



Wanneer een gebruiker een functie nodig heeft, die in MSX-BASIC niet bestaat, dan is het mogelijk om in het programma zelf de functie te definieren. Laten we aannemen dat we in een programma de oppervlakte van een cirkel ($3.14*R^2$) willen berekenen. Met het **statement DEF FN** (definieer functie) kunnen we dan de uitdrukking $3.14*R^2$ definieren als een functie. We zouden zo'n zelf gedefinieerde functie dan een eigen functie kunnen noemen. Het algemene formaat van het DEF FN statement is:



De formule voor het berekenen van de oppervlakte van een cirkel wordt als volgt gedefinieerd:

```
DEF FNOPC(R)=3.14*R^2
```

In het statement is OPC (OPpervlakte Cirkel) de naam van de functie. Is de functie eenmaal gedefinieerd, dan kan deze onder de naam FNOPC(R) worden opgeroepen.

1.1 De functies ABS en SGN.

Met de functie ABS(X) kunnen we de absolute waarde van het argument X bepalen. Dit houdt in, dat het resultaat niet het teken min (-) of plus (+) voert. Bijvoorbeeld:

ABS(-27) = 27
ABS(+12) = 12

Met de functie SGN(X) kunnen we het teken van X bepalen. Hiervoor gelden de volgende regels:

Indien X kleiner is dan 0, dan is het resultaat -1.
Indien X gelijk is aan 0, dan is het resultaat 0.
Indien X groter is dan 0, dan is het resultaat 1.

Voorbeelden hiervan zijn:

SGN(-23) = -1
SGN(12.23) = 1

Het volgende programma geeft een aantal voorbeelden van de functies ABS en SGN.

```
100 REM * VOORBEELDEN ABS EN SGN *
110 CLS
120 INPUT "Variabelen a, b en c";A,B,C
130 PRINT
140 PRINT "Absolute waarde van ";A;"is";ABS(A)
150 PRINT "Absolute waarde van ";B;"is";ABS(B)
160 PRINT
170 PRINT "Teken van ";A;"is ";SGN(A)
180 PRINT "Teken van ";B;"is ";SGN(B)
190 PRINT "Teken van ";A;"+";B;"+";C;"is ";SGN
(A+B+C)
200 PRINT
210 A$=INKEY$:IF A$<>CHR$(13) GOTO 210
220 GOTO 110
```

Opmerking:
Druk voor het nogmaals uitvoeren van het programma eerst op de RETURN-toets.

Tussen de functies ABS en SGN bestaat de volgende relatie:

ABS(X) = X * SGN(X)

Het is ook interessant om na te gaan, hoe de absolute waarde van een getal zou moeten worden bepaald, indien de functie ABS in MSX-BASIC niet zou bestaan. Met het volgende programma wordt dit gedemonstreerd.

```
100 CLS
110 INPUT "Getal x";X
120 PRINT
130 Y=X
140 IF X>=0 GOTO 160
150 Y=-X
160 PRINT "Absolute waarde van ";X;"is";Y
170 A$=INKEY$:IF A$<>CHR$(13) GOTO 170
180 GOTO 100
```

Opmerking:
Druk voor het invoeren van het volgende getal eerst de RETURN-toets in.

Het volgende programma toont een oplossing voor het geval dat de functie SGN niet beschikbaar zou zijn:

```
100 CLS
110 INPUT "Getal x";X
120 PRINT
130 IF X>0 THEN T=1:GOTO 160
140 IF X=0 THEN T=0:GOTO 160
150 IF X<O THEN T=-1
160 PRINT "Teken van ";X;"is ";T
170 A$=INKEY$:IF A$<>CHR$(13) GOTO 170
180 GOTO 100
```

1.2 De functie FIX.

Wanneer we van een negatief getal het gedeelte achter de komma willen afkappen, kan dit niet zonder meer met de INT-functie worden gedaan. Het volgende voorbeeld laat dit zien:

INT(-47.12) = -48

In plaats van de gewenste waarde -47 krijgen we echter de waarde -48. Hoe kunnen we dit probleem nu oplossen? Eenvoudig, door gebruik te maken van de functie FIX. Met de functie FIX worden alle cijfers achter de decimale punt (komma) verwijderd. De werking komt overeen met de INTfunctie, met dit verschil, dat de functie FIX negatieve getallen niet naar beneden afrondt. Het eerder genoemde probleem kunnen we nu als volgt oplossen:

FIX(-47.12) = -47

Voor positieve getallen is het resultaat van de functies INT en FIX hetzelfde. Het volgende programma laat een aantal voorbeelden zien.

```
10 CLS
20 A=2.16:B=-12.47:C=6
30 PRINT INT(A);TAB(12);FIX(A)
40 PRINT INT(B);TAB(12);FIX(B)
50 PRINT INT(B+C);TAB(12);FIX(B+C)
60 PRINT
70 END
```

Voor het oplossen van voorgaand probleem kunnen we in plaats van de functie FIX ook gebruik maken van de functies SGN, INT en ABS. We bepalen met de functie SGN het teken van het getal, met de functie ABS de absolute waarde en met de functie INT het dichtstbijgelegen kleinere gehele getal van de absolute waarde. Door nu het resultaat hiervan te vermenigvuldigen met het teken, verkrijgen we zowel voor negatie-

ve- als voor positieve getallen de gewenste waarde, dat wil zeggen, alleen het gehele gedeelte van het getal. Het volgende programma laat dit zien.

```
10 CLS
20 INPUT "Getal";X
30 PRINT SGN(X)*INT(ABS(X))
40 PRINT
50 GOTO 20
```

1.3 De functie SQR.

Met de functie SQR(X) kunnen vierkantswortels worden berekend. Het argument X moet hierbij gelijk aan of groter dan 0 zijn. Wanneer het argument kleiner is dan 0, zal de foutboodschap "Illegal function call" op het scherm verschijnen. Het volgende programma toont een aantal voorbeelden met de functie SQR.

```
10 A=25:B=16:C=42
20 PRINT SQR(A)
30 PRINT SQR(A+C)
40 PRINT SQR(A+SQR(B)+7)
50 PRINT
60 END
```

Wanneer de functie SQR niet beschikbaar zou zijn, zou SQR(X) ook als X^0.5 kunnen worden geschreven.

We hebben voor de functie SQR(X) twee programmavoorbeelden. In het eerste voorbeeld wordt de functie toegepast voor het berekenen van de vierkantswortel voor verschillende waarden van X en in het tweede voorbeeld voor het berekenen van de wortels x1 en x2 van een vierkantsvergelijking.

De volgende listing toont het programma voor het genereren van een tabel. Hierbij kunnen, met behulp van de INPUT-statements op de regelnummers 115, 120 en 125, de begin-

waarde, de eindwaarde en de stapwaarde worden ingegeven. In de tweede kolom van de gegenereerde tabel staan de vierkantswortels van de verschillende waarden van X vermeld.

```
100 REM * VOORBEELD FUNCTIE SQR(X) *
105 CLS
110 PRINT "VIERKANTSWORTELS":PRINT
115 INPUT "Beginwaarde x";B
120 INPUT "Eindwaarde  x";E
125 INPUT "Stapwaarde  x";S
130 PRINT
135 PRINT "  x";TAB(10);"Vx"
140 PRINT
145 FOR X=B TO E STEP S
150 PRINT USING "####    ##.###";X,SQR(X)
155 NEXT X
160 A$=INKEY$:IF A$<>CHR$(13) GOTO 160
165 GOTO 105
```

Opmerking:

Wanneer men een volgende tabel wil genereren, zal eerst de RETURN-toets moeten worden ingedrukt.

Met het volgende programma kunnen de wortels x1 en x2 van een vierkantsvergelijking worden berekend. De algemene vorm van de vierkantsvergelijking ziet er als volgt uit:

$$ax^2 + bx + c = 0$$

In het programma worden de wortels als volgt berekend:

1. Indien a=0, dan bestaat er maar 1 wortel, en wel de wortel x=-c/b.

2. Indien c=0, dan zijn er twee gelijke reele wortels, namelijk x1=x2=-b/a

3.

Indien $b^2 - 4ac = 0$ (decrement D), dan zijn er twee gelijke reele wortels, $x1=x2=-b/2a$.

4.

Indien $b^2 - 4ac > 0$ (decrement D), dan zijn er twee reele wortels, namelijk:

$x1=(-b+\sqrt{D})/2a$ en $x2=(-b-\sqrt{D})/2a$.

5.

Indien $b^2 - 4ac < 0$ (decrement D), dan zijn er 2 complexe wortels. Voor het berekenen van de complexe wortels zullen we eerst een stuk kennis uit de rekenkunde gaan opfrissen.

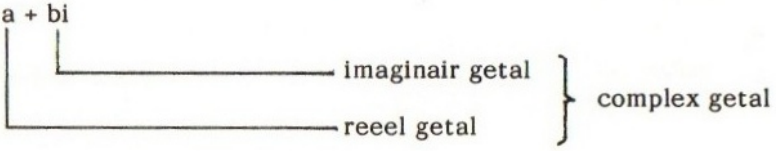
Indien $b^2 - 4ac < 0$, dat wil zeggen, D is negatief, dan is het niet mogelijk de vierkantswortel van D te berekenen. De vierkantswortel uit een negatief getal stelt noch een positief noch een negatief getal voor. Men noemt zo'n uitdrukking een imaginair getal (bijvoorbeeld: de wortel uit -25). In tegenstelling hiermee worden positieve- en negatieve getallen samengevat onder de benaming reele getallen. Het eenvoudigste imaginaire getal is de wortel uit -1. Het wordt de imaginaire eenheid genoemd en gewoonlijk voorgesteld door de letter i. Voorbeelden van imaginaire getallen zijn:

$\sqrt{-1} = i \qquad i^2 = -1$

$\sqrt{-25} = \sqrt{25} \times \sqrt{-1} = 5i$

$\sqrt{-3} = \sqrt{-1} \times \sqrt{3} = i\sqrt{3}$

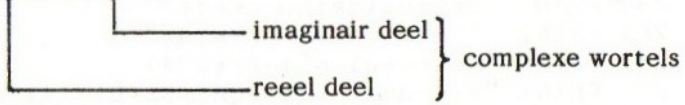
De som van een reeel getal en een imaginair getal noemt men een complex getal. Bijvoorbeeld:



Na deze uiteenzetting kunnen we met een gerust hart de complexe wortels geven.

$x1 = -b/2a - \sqrt{-D}/2a$

$x2 = -b/2a + \sqrt{-D}/2a$



De volgende listing toont het programma voor het berekenen van de wortels van een vierkantsvergelijking.

```
100 REM * VIERKANTSVERGELIJKING *
105 WIDTH 40
110 CLS
115 PRINT STRING$(27,"*")
120 PRINT "* BEREKENEN WORTELS VAN    *"
125 PRINT "* VIERKANTSVERGELIJKINGEN *"
130 PRINT "* a*x^2 + b*x + c = 0      *"
135 PRINT "* Invoeren  :a, b en c     *"
140 PRINT "* Berekenen :x1 en x2      *"
145 PRINT STRING$(27,"*")
150 PRINT
155 INPUT "Variabele a";A
160 INPUT "Variabele b";B
165 INPUT "Variabele c";C
170 PRINT
175 IF A<>0 GOTO 190
180 PRINT "Vergelijkng heeft 1 wortel:"
185 PRINT "x1 =";-C/B:GOTO 290
190 IF C<>0 GOTO 210
195 PRINT "Vergelijking heeft"
200 PRINT "2 gelijke reele wortels:"
205 PRINT "x1 = x2 =";-B/A:GOTO 290
210 D=B^2-4*A*C
215 IF D<>0 GOTO 235
220 PRINT "Vergelijking heeft"
225 PRINT "2 gelijke reele wortels:"
```

```
230 PRINT "x1 = x2 =";-B/(2*A):GOTO 290
235 IF D<0 GOTO 260
240 PRINT "Vergelijking heeft"
245 PRINT "2 reele wortels:"
250 PRINT "x1 =";(-B+SQR(D))/(2*A)
255 PRINT "x2 =";(-B-SQR(D))/(2*A):GOTO 290
260 PRINT "Vergelijking heeft"
265 PRINT "2 complexe wortels:"
270 PRINT "x1 reeel =";-B/(2*A)
275 PRINT "x1 imag. =";+SQR(ABS(D))/(2*A)
280 PRINT "x2 reeel =";-B/(2*A)
285 PRINT "x2 imag. =";-SQR(ABS(D))/(2*A)
290 PRINT
295 INPUT "Volgende opgave (J/N)";J$
300 IF J$="J" OR J$="j" GOTO 110
305 END
```

1.4 De functie RND

Met de functie RND(X) kunnen reeksen willekeurige getallen, met waarden die liggen tussen 0 en 1, worden gegenereerd. De getallen worden door de computer gegenereerd, waarbij wordt gezorgd voor een zo goed mogelijke verdeling van de getallen. Afhankelijk van de waarde van het het argument X, kunnen verschillende effecten worden verkregen. Voor X gelden de volgende regels:

X kleiner dan 0
Dit geeft het eerste getal uit een door de computer gegenereerde reeks getallen. De reeks getallen wordt bepaald door de waarde van X. Dit wil zeggen, dat bij elke waarde van X een andere reeks getallen behoort. Zou men het volgende getal uit de reeks willen hebben, dan moet X groter dan 0 zijn.

X groter dan 0
Geeft het volgende getal uit een reeds aangevangen reeks. Indien de reeks nog niet eerder is aangevangen, dan wordt als eerste getal het getal 0.595219439946623 gegeven.

X gelijk aan 0
Herhaalt het laatst gegeven getal. Wanneer nog niet eerder een reeks is aangevangen, dan wordt hetvolgende getal gegeven:
0.40649651372358

Met het volgende programma worden 10 willekeurige getallen tussen 0 en 1 gegenereerd. Voer eerst met behulp van het INPUT-statement op regelnummer 15 een aantal keren een positief getal in (X is groter dan 0). Start het programma steeds met het commando RUN. Het zal dan blijken, dat voor welke waarde van X dan ook, steeds dezelfde reeks getallen wordt afgedrukt. Voer hierna een aantal keren een negatief getal in (X is kleiner dan 0). Nu zal blijken dat voor elke waarde van X tien dezelfde getallen worden afgedrukt, echter steeds met een andere waarde. Voor X is kleiner dan 0 wordt steeds weer het eerste getal uit de reeks genomen, vandaar dat de getallen gelijk zijn.

Maak vervolgens de waarde van X gelijk aan 0, hetgeen tot gevolg heeft, dat 10 dezelfde getallen worden afgedrukt.

```
10 REM * REEKSEN GETALLEN *
15 INPUT "Argument X";X
20 FOR I=1 TO 10
25 PRINT RND(X)
30 NEXT I
35 PRINT
40 END
```

Wil men steeds bij het starten van een programma door middel van het commando RUN een andere reeks getallen genereren, dan kan dit op de volgende twee manieren.

Oplossing 1:
Genereer de reeks getallen met PRINT RND(-TIME) en bepaal steeds het volgende getal met PRINT RND(1). Daar de waarde van de systeemvariabele TIME bijna nooit hetzelfde is wanneer het programma wordt gestart, wordt steeds een andere negatieve waarde verkregen. Dit heeft tot gevolg dat steeds een andere reeks getallen door de computer wordt

gegenereerd.

```
10 PRINT "10 willekeurige getallen"
15 PRINT
20 PRINT RND(-TIME)
25 FOR I=1 TO 9
30 PRINT RND(1)
35 NEXT I
40 PRINT
45 END
```

Oplossing 2:
Genereer steeds een nieuwe reeks met PRINT RND(-TIME) en druk daarvan steeds het eerste getal af.

```
10 PRINT "10 willekeurige getallen"
15 PRINT
20 FOR I=1 TO 10
25 PRINT RND(-TIME)
30 NEXT I
35 PRINT
40 END
```

Wanneer we willekeurige gehele getallen tussen 0 en 9 willen genereren, dan moet het resultaat van RND(X) worden vermenigvuldigd met 10, waarna met de functie INT het gehele deel van het verkregen resultaat wordt bepaald. Het volgende programma laat een mogelijke oplossing zien.

```
10 CLS
15 PRINT "GETALLEN TUSSEN 0 EN 9"
20 PRINT INT(RND(-TIME)*10);
25 FOR I=1 TO 15
30 PRINT INT(RND(1)*10);
35 NEXT I
40 END
```

Wanneer we getallen tussen 1 en 10 willen genereren, dan

zullen we bij het resultaat van RND(-TIME)*10 nog 1 moeten optellen. Dit is te zien in het volgende programma.

```
10 CLS
15 PRINT INT(RND(-TIME)*10+1);
20 FOR I=1 TO 15
25 PRINT INT(RND(1)*10+1);
30 NEXT I
35 END
```

De meeste lezers zullen wel eens hebben gehoord van of zelfs meegedaan aan de lotto. Daarbij moeten zes getallen en een reservegetal, liggend tussen 1 en 41, worden ingevuld op een lotto-formulier. Met het volgende programma kunnen deze getallen worden gegenereerd.

```
100 CLS
110 DIM A(7)
120 PRINT STRING$(15,"*")
130 PRINT "*  L O T T O  *
140 PRINT STRING$(15,"*")
150 PRINT:PRINT
160 A=RND(-TIME)
170 FOR I=1 TO 7
180 A(I)=INT(RND(1)*41+1)
190 IF I=1 GOTO 240
200 FOR K=1 TO I-1
210 IF A(I)=A(K) GOTO 180
220 NEXT K
230 IF I=7 THEN PRINT SPC(5);
240 PRINT A(I);
250 NEXT I
260 PRINT:PRINT
270 END
```

Met de statements op de regelnummers 200, 210 en 220 wordt gecontroleerd of het getal al eerder werd gegenereerd. Is dat het geval, dan moet er een nieuw getal worden bepaald.

In het programma "Tollenaar", in appendix A, wordt op de regelnummers 1600 en 2350 gebruik gemaakt van de functie RND om respectievelijk getallen te genereren tussen 0 en 3 en tussen 0 en 7.

1.5 Het definieren van eigen functies.

Het statement DEF FN stelt de gebruiker in staat om functies te definieren, die niet in MSX-BASIC bestaan. Laten we aannemen, dat we een functie willen creeren voor het berekenen van de oppervlakte van een cirkel. Het statement gaat er dan als volgt uitzien:

```
DEF FNOPC(R)=3.14*R^2
```

FN is de afkorting van het woord FuNctie en wordt altijd gevolgd door de naam van de functie. In voorgaand voorbeeld is dit de naam OPC (OPpervlakte Cirkel). De functie krijgt de naam van een numerieke variabele, indien het resultaat van de functie numeriek is en de naam van een alfanumerieke variabele (moet eindigen met het $-teken), indien het resultaat van de functie alfanumeriek is.

In de gedefinieerde functie mogen meerdere variabelen worden gebruikt. De variabelen tussen de haakjes (in het voorbeeld R) moeten overeenstemmen met de variabelen rechts van het is-gelijk-teken. De variabelen (argumenten) worden gewoonlijk dummy-variabelen genoemd, omdat zij dienen om aan te geven waar later, bij het uitvoeren van de functies, de op dat moment ingevulde variabelen moeten komen te staan.

De uitdrukking 3.14*R^2 is in het voorbeeld gedefinieerd als functie FNOPC. Het statement DEF FN moet worden uitgevoerd voordat de gedefinieerde functie in het programma gaat worden gebruikt, anders verschijnt op het scherm de foutboodschap "Undefined user function".

Wanneer de functie volgens bovenstaande manier is gedefinieerd, kan deze in de rest van het programma net zoals een

standaard functie worden gebruikt. Zouden we bijvoorbeeld de functiewaarde van de functie FNOPC willen afdrukken voor R=6, dan kan dit als volgt:

```
PRINT FNOPC(6)
```

Met het volgende programma kunnen we de oppervlakte van een cirkel berekenen.

```
100 CLS
110 PRINT "OPPERVLAKTE CIRKEL"
120 PRINT
130 DEF FNOPC(R)=3.14*R^2
140 INPUT "Straal R";R
150 O=FNOPC(R)
160 PRINT "Oppervlakte cirkel is";O
170 PRINT:PRINT
180 A$=INKEY$:IF A$<>CHR$(13) GOTO 180
190 GOTO 140
```

In de uitdrukking (formule) rechts van het is-gelijkteken mogen in het DEF FN statement ook standaard functies worden gebruikt. Verder mogen er ook zelf gedefinieerde functies in voorkomen, mits deze maar eerder zijn gedefinieerd. In het volgende programma wordt een functie gedefinieerd voor de formule:

```
(A+2*B)/SQR(C)
```

```
100 CLS
110 DEF FNF1(A,B,C)=(A+2*B)/SQR(C)
120 PRINT "FUNCTIE (A+2*B)/SQR(C)"
130 PRINT
140 INPUT "Variabelen A, B en C";A,B,C
150 PRINT
160 R=FNF1(A,B,C)
170 PRINT "(A+2*B)/SQR(C) =";
180 PRINT USING "#####.###";R
190 PRINT:PRINT
```

```
200 INPUT "Volgende berekening (J/N)";J$
210 IF J$="J" OR J$="j" GOTO 130
220 END
```

Variabelen in het DEF-statement zijn dummy-variabelen.

Steeds wanneer de functie FN wordt uitgevoerd, worden de dummy-variabelen van het DEF-statement vervangen door de overeenkomstige numerieke variabelen. De dummy-variabelen en numerieke variabelen behoeven niet dezelfde namen te hebben. Het volgende programma toont dit aan.

```
100 CLS
110 DEF FNF(N)=4*N^2-4*N
120 INPUT "Variabele X";X
130 PRINT FNF(X)
140 PRINT
150 GOTO 120
```

Wanneer in voorgaand programma de functie FNF op regelnummer 130 wordt uitgevoerd, wordt de dummy-variabele N van het DEF-statement vervangen door de numerieke variabele X.

Het volgende programma laat een voorbeeld zien van een alfanumerieke functie. Het programma drukt de tekens uit de kolommen 0 en 1 uit de MSX-tekenset-tabel op het beeldscherm af.

```
100 CLS
110 DEF FNC$(I)=CHR$(1)+CHR$(I)
120 FOR J=65 TO 95
130 IF J=75 OR J=85 THEN PRINT:PRINT
140 PRINT FNC$(J);SPC(2);
150 NEXT J
160 PRINT:PRINT
170 END
```

2 Afdrukken op scherm en papier

Wanneer we gaan afdrukken op papier, krijgen we te maken met het commando LLIST, het statement LPRINT en de functie LPOS. De eerste twee kunnen respectievelijk worden vergeleken met het commando LIST en het statement PRINT, die dienen voor het afdrukken van informatie op het scherm. Met de functie LPOS kunnen we de positie van de printkop op de regel bepalen.

Met de functies CSRLIN en POS kunnen we respectievelijk de verticale en horizontale positie van de cursor op het beeldscherm bepalen.

2.1 Het commando LLIST

Het **commando LLIST** kan alleen worden gebruikt, wanneer een printer op de MSX-computer is aangesloten. Het aangesloten zijn alleen is geen garantie dat er iets op de printer wordt afgedrukt. Een veel voorkomende bedieningsfout is, dat de printer wel aangesloten en aangeschakeld is, doch dat deze niet "ready" staat. Op de meeste printers zit een schakelaar, waarmee de printer "ready" of "on line" kan worden gezet. Een lampje geeft dan aan of de printer "ready" of "on line" is. Pas wanneer dat het geval is, zal door de computer naar de printer gestuurde tekst kunnen worden afgedrukt.

Men kan van een programma, dat in het geheugen staat, te allen tijde een afdruk (listing) op papier maken. Hiertoe dient men de computer het commando LLIST te geven. Net als bij alle andere commando's, zal ook LLIST moeten worden gevolgd door het indrukken van de RETURN-toets. Het active-

ren van het commando LLIST heeft tot gevolg, dat de computer de programmaregels van het in het geheugen staande programma in volgorde van regelnummer op papier afdrukt.

Het enige verschil met het commando LIST is, dat de programmaregels niet op het beeldscherm, maar op papier (printer) worden afgedrukt. Het commando LLIST mag ook als statement in een programma worden gebruikt.

Aan het commando LLIST kunnen ook parameters worden toegevoegd. Het commando krijgt dan de volgende betekenissen:

LLIST	Het hele programma wordt afgedrukt.
LLIST n	Alleen regelnummer n wordt afgedrukt.
LLIST n-m	Regelnummers n t/m m worden afgedrukt.
LLIST -m	Alle regels t/m m worden afgedrukt.
LLIST n-	Alle regels vanaf n worden afgedrukt.
LLIST.	De laatst behandelde regel wordt afgedrukt.

Laad na het bestuderen van bovenstaande formaten een programma van cassette in het geheugen en oefen vervolgens met de verschillende formaten van het commando LLIST.

Wanneer er een printer is aangesloten op uw MSX-computer, is het erg zinvol, om de functie van een van de functietoetsen te veranderen in LLIST + chr$(13). We kunnen daarvoor bijvoorbeeld de functietoets F3 nemen. In de directe mode kan de functie van die functietoets als volgt worden veranderd:

```
KEY 3,"LLIST"+CHR$(13)
```

2.2 Het statement LPRINT

Het **statement LPRINT** komt, wat uitvoering betreft, volledig overeen met het statement PRINT, echter met dit verschil, dat er niet wordt afgedrukt op een beeldscherm, maar op het papier van een printer. Dit betekent wel dat er een printer moet zijn aangesloten op uw MSX-computer.

De verschillende formaten en mogelijkheden van het PRINT-statement werden reeds uitvoerig behandeld in deel 1 van deze serie leerboeken. We zullen dat hier dan ook niet herhalen. Het resultaat van de verschillende functies (TAB, SPC, SPACE$, etc.) in combinatie met het statement LPRINT is precies hetzelfde als in combinatie met het statement PRINT.

In het volgende programma worden een aantal versies van het LPRINT-statement toegepast.

```
100 CLS
105 PRINT "VIERKANTWORTELS":PRINT
110 INPUT "Beginwaarde x";B
115 INPUT "Eindwaarde  x";E
120 INPUT "Stapwaarde  x";S
125 LPRINT CHR$(12)
130 LPRINT "  x";TAB(10);"Vx"
135 LPRINT
140 FOR X=B TO E STEP S
145 LPRINT USING "####    ##.###";X,SQR(X)
150 NEXT X
155 PRINT
160 PRINT "Tabel is afgedrukt."
165 INPUT "Volgende tabel (J/N)";J$
170 IF J$="J" OR J$="j" GOTO 100
175 END
```

2.3 De systeemvariabelen CSRLIN en POS

De **systeemvariabele CSRLIN** geeft als resultaat het nummer van de regel (Y-coordinaat), waarop de cursor zich, op het moment van uitvoering van de functie, bevindt. De **systeemvariabele CSRLIN** wordt ook wel gezien als functie. Het resultaat van de "functie" CSRLIN ofwel de inhoud van de systeemvariabele CSRLIN kan 0 tot en met 23 zijn, waarbij 0 de bovenste regel van het scherm voorstelt.

De variabele kan alleen worden toegepast in de tekstmodes 1 en 2. Aan de systeemvariabele CSRLIN kan door de gebruiker geen waarde worden toegekend; de waarde kan alleen worden gebruikt (gelezen).

In het volgende programma wordt op grond van de waarde van CSRLIN het afdrukken van de tabel voortgezet of beeindigd. De lengte van de tabel is afhankelijk van het aantal regels dat op het beeldscherm kan worden afgedrukt. In het programma zal het afdrukken van de tabel worden gestopt, wanneer regel 20 op het scherm (zie regelnummer 70) is bereikt. Door nu op de RETURN-toets te drukken, zal worden verdergegaan met regelnummer 90. Daar ziet u hoe de systeemvariabele CSRLIN toch kan worden gewijzigd; door de cursor naar een andere regel te dirigeren met het LOCATE-statement.

```
10 CLS
20 PRINT "VIERKANTSWORTELS":PRINT
30 INPUT "Beginwaarde x";X:PRINT
40 PRINT "  x";TAB(10);"Vx":PRINT
50 PRINT USING "####    ###.##";X,SQR(X)
60 X=X+1
70 IF CSRLIN<20 GOTO 50
80 A$=INKEY$:IF A$<>CHR$(13) GOTO 80
90 LOCATE 0,4
100 GOTO 40
```

We hebben in het eerste deel van deze paragraaf geleerd hoe met de systeemvariabele CSRLIN kan worden bepaald op welke regel van het beeldscherm de cursor staat. Wanneer we nu willen bepalen op welke positie van de regel de cursor staat, kunnen we gebruik maken van de **systeemvariabele POS(0).**

De systeemvariabele POS(0) geeft als resultaat de momentele positie van de cursor op de regel (X-coordinaat). Het resultaat kan 0 tot en met 39 zijn, waarbij de meest linker positie 0 is. De uitdrukking tussen haakjes is een dummy-uitdrukking. Het beste kan hiervoor het cijfer 0 worden gebruikt. De

systeemvariabele kan alleen in de tekstmodes 1 en 2 worden toegepast. POS(0) wordt, evenals CSRLIN, ook wel eens als functie beschouwd.

In het statement LOCATE X,Y komt parameter X, de horizontale positie (0-39), overeen met de systeemvariabele POS(0) en komt parameter Y, de vertikale positie (0-23), overeen met de systeemvariabele CSRLIN. Met het volgende programma wordt dit bewezen.

```
10 CLS
20 LOCATE 19,11
30 A=CSRLIN:B=POS(0)
40 PRINT A;B
50 GOTO 50
```

Met het volgende programma worden de tekens uit de kolommen 0 en 1 van de MSX-tekenset-tabel afgedrukt. Wanneer tijdens het afdrukken van de tekens een positie, die groter is dan 20, wordt bereikt, wordt er eerst een blanco regel gegeven, voordat er wordt verdergegaan met het afdrukken. Het bepalen of de positie groter is dan 20, wordt gedaan op grond van de waarde van systeemvariabele POS(0).

```
10 CLS
20 DEF FNC$(I)=CHR$(1)+CHR$(I)
30 FOR J=65 TO 95
40 PRINT FNC$(J);SPC(2);
50 IF POS(0)>20 THEN PRINT:PRINT
60 NEXT J
70 PRINT:PRINT
80 END
```

2.4 De systeemvariabele LPOS(0)

De **systeemvariabele LPOS(0)** kan alleen worden gebruikt, wanneer er een printer is aangesloten op de MSX-computer.

De systeemvariabele LPOS(0) geeft een indicatie van de momentele print-positie in de printer-buffer. Het resultaat van LPOS(0) behoeft namelijk niet noodzakelijkerwijs de fysische positie van de printkop te zijn.

LPOS(0) heeft veel gemeen met POS(0). De uitdrukking tussen de haakjes is een dummy-uitdrukking. Het beste kan hiervoor het cijfer 0 worden gebruikt.

Wanneer de tekens uit het vorige programma op papier (printer) moeten worden afgedrukt, gaat het programma er als volgt uitzien:

```
10 LPRINT CHR$(12)
20 DEF FNC$(I)=CHR$(1)+CHR$(I)
30 FOR J=65 TO 95
40 LPRINT FNC$(J);SPC(2);
50 IF LPOS(0)>20 THEN LPRINT:LPRINT
60 NEXT J
70 PRINT "Tekens zijn afgedrukt."
80 END
```

3 Goniometrische functies

In de wis- en natuurkunde vormen de goniometrische functies een onmisbaar hulpmiddel. Het ligt niet in de bedoeling om uitgebreid in te gaan op de goniometrie zelf, de kennis hiervan dient aanwezig te zijn. Hebt u die kennis niet, dan kunt u wellicht dit hoofdstuk beter overslaan. MSX-BASIC kent voor de goniometrie de volgende standaard functies:

SIN(X)	de sinus van argument X
COS(X)	de cosinus van argument X
TAN(X)	de tangens van argument X
ATN(X)	de arctangens van argument X

Verder zal in dit hoofdstuk de functie EXP(X) worden behandeld, waarmee de X-de macht van het grondtal van de natuurkundige logaritme (e) kan worden berekend.

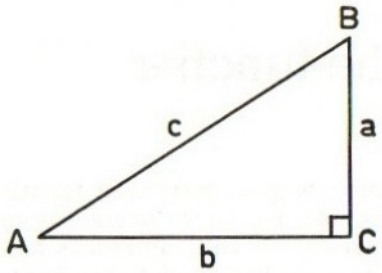
Als laatste zal in dit hoofdstuk de functie LOG(X) aan de orde komen. Met deze functie kan de natuurlijke logaritme van argument X worden berekend.

Het formaat van bovengenoemde functies staat vermeld in het begin van hoofdstuk 1.

3.1 De goniometrische functies SIN, COS, TAN en ATN.

Met de goniometrische functies SIN(X), COS(X), TAN(X) en ATN(X) zijn we in staat hoeken en zijden van meetkundige figuren te berekenen. Argument X van de **functies SIN, COS en TAN** moet hierbij worden uitgedrukt in radialen. Een radiaal is 180/pi=57.29578 graden. Is het aantal graden (Y) be-

kend, dan is het aantal radialen X=Y*pi/180.



Afb. 3-1 De rechthoekige driehoek ABC.

Beschouwen we de rechthoekige driehoek ABC uit afbeelding 3-1, dan is:

SIN(X) = a/c
COS(X) = b/c
TAN(X) = a/b

Het argument X mag een numerieke constante, een numerieke variabele of een uitdrukking zijn.

In het volgende programma wordt met het INPUT-statement op regelnummer 120 de hoek, uitgedrukt in graden, ingevoerd. Met de uitdrukking op regelnummer 130 worden de graden omgezet in radialen. Op de regelnummers 150 tot en met 170 worden achtereenvolgens de sinus, cosinus en tangens van de ingevoerde hoek berekend en afgedrukt.

```
100 REM * GONIOMETRISCHE FUNCTIES *
110 CLS
120 INPUT "Hoek in graden";H
130 R=H*3.14159/180
140 PRINT:PRINT
150 PRINT "sinus   van";H;"graden is";
155 PRINT USING "###.#####";SIN(R):PRINT
```

```
160 PRINT "cosinus van";H;"graden is";
165 PRINT USING "###.#####";COS(R):PRINT
170 PRINT "tangens van";H;"graden is";
175 PRINT USING "###.#####";TAN(R):PRINT
180 PRINT
190 INPUT "Volgende opgave (J/N)";J$
200 IF J$="J" OR J$="j" GOTO 110
210 END
```

Met de **functie ATN(X)** wordt de arctangens (in radialen) van de goniometrische verhouding X berekend. X stelt hierin de verhouding tussen de zijden a en b voor (zie afbeelding 3-1). Het tegenovergestelde van ATN(X) is TAN(X). Dit wil zeggen dat:

ATN(a/b) = X
TAN(X) = a/b

Opgave:
Schrijf een programma, waarmee hoek A (zie afbeelding 3-1) wordt berekend met de functie ATN. De zijden a en b moeten hierbij door middel van INPUT-statements worden ingevoerd. Hoek A moet worden afgedrukt in graden.

Met het volgende programma kunnen van een hoek, die is uitgedrukt in graden, minuten en seconden, de sinus en cosinus worden berekend.

```
10 CLS
20 INPUT "Hoek in gr,min,sec";G,M,S
30 R=(G+M/60+S/360)*3.14159/180
40 PRINT
50 PRINT USING "#.#####";SIN(R):PRINT
60 PRINT USING "#.#####";COS(R):PRINT
70 A$=INKEY$:IF A$<>CHR$(13) GOTO 70
80 GOTO 10
```

Het volgende programma definieert een functie, die aan een hoek in graden, de hoek in radialen toevoegt.

```
10 CLS
20 DEF FNR(X)=X*3.14159/180
30 INPUT "Hoek in graden";G
40 PRINT
50 PRINT G;"graden =";
60 PRINT USING "##.#####";FNR(G);
70 PRINT " radialen"
80 A$=INKEY$:IF A$<>CHR$(13) GOTO 80
90 PRINT:GOTO 30
```

Iedereen heeft wel eens gewerkt met een sinustabel. Met het volgende programma bent u nu in staat zelf een sinustabel te genereren voor een bepaald interval. De begin- en eindwaarde worden hierbij ingevoerd met INPUT-statements.

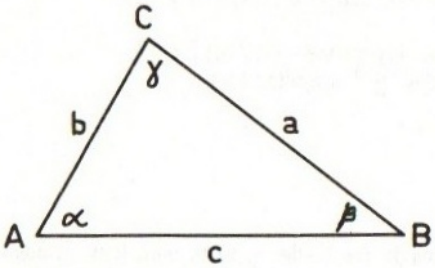
```
100 REM * SINUSTABEL *
110 CLS
120 INPUT "Beginwaarde in graden";B
130 INPUT "Eindwaarde in graden ";E
140 PRINT
150 FOR G=B TO E
160 R=G*3.14159/180
170 PRINT USING "###";G;
180 PRINT SPC(3);
190 PRINT USING "#.#####";SIN(R)
200 NEXT G
210 A$=INKEY$:IF A$<>CHR$(13) GOTO 210
220 GOTO 110
```

Voor het berekenen van de zijden van een driehoek (zie afbeelding 3-2) kunnen we gebruik maken van de sinusregel. De sinusregel luidt als volgt:

> De zijden van een driehoek zijn evenredig met de sinussen van de overstaande hoeken.

Dit is in formulevorm:

> a/sin alfa = b/sin beta = c/sin gamma



Afb. 3-2 Driehoek ABC

Met het volgende programma kan met de sinusregel zijde b worden berekend, indien zijde a en de hoeken A en B zijn gegeven. De gegevens worden ingevoerd met de INPUT-statements op de regelnummers 190 tot en met 210. Op de regelnummers 230 en 240 worden de graden van de hoeken A en B omgezet in radialen. Met het statement op regelnummer 250 wordt de grootte van zijde b berekend en afgedrukt.

Aan de lezer wordt overgelaten het programma zodanig uit te breiden, dat het ook geschikt is voor het berekenen van de zijden a en c. Maak hierbij gebruik van keuze via een menu.

```
100 REM * SINUSREGEL *
110 CLS
120 PRINT STRING$(30,"*")
130 PRINT "* TOEPASSING SINUSREGEL      *"
140 PRINT "* Gegeven: Zijde a           *"
150 PRINT "*          Hoek A en B       *"
160 PRINT "* Berek. : b=a*SIN(B)/SIN(A) *"
170 PRINT STRING$(30,"*")
180 PRINT
190 INPUT "Zijde a in cm   ";Z
200 INPUT "Hoek A in graden";A
210 INPUT "Hoek B in graden";B
220 PRINT
230 R1=A*3.14159/180
240 R2=B*3.14159/180
```

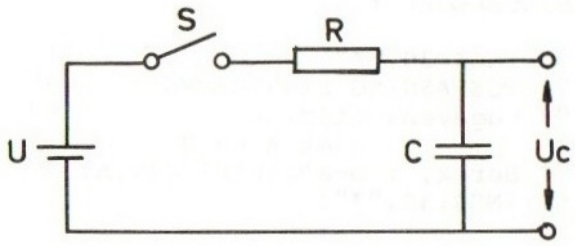
```
250 PRINT "b =";Z*SIN(R2)/SIN(R1);"cm"
260 PRINT:PRINT
270 INPUT "Volgende opgave (J/N)";J$
280 IF J$="J" OR J$="j" GOTO 110
290 END
```

3.2 De functie EXP

Met de **functie EXP(X)** wordt de X-de macht van het grondtal van de natuurlijke logaritme e berekend. De waarde van e = 2.718282. Het volgende programma laat een aantal voorbeelden zien.

```
10 A=8:B=6:C=3
20 PRINT "EXP(4) = 2.718282^4 =";EXP(4)
30 PRINT EXP(A)
40 PRINT EXP(B)
50 END
```

We zullen vervolgens de functie EXP(X) toepassen voor het berekenen van de momentele spanningen over condensator C, na het sluiten van schakelaar S in het circuit, zoals wordt getoond in afbeelding 3-3.



Afb. 3-3 Circuit met weerstand en condensator (RC-kring)

Voor het berekenen van de momentele spanningen wordt de volgende formule gebruikt:

$$Uc = U * (1-e^{(-t/C*R)})$$

In deze formule is:
Uc de momentele spanning in Volt op tijdstip t.
U de batterijspanning in Volt.
e het grondtal van de natuurlijke logaritme.
t het tijdstip in seconden na sluiten van schakelaar S.
C de capaciteit in Farad.
R de weerstand in Ohm.

In het programma wordt de formule als volgt geschreven:

```
170 UC=U*(1-EXP(-T/(C*R)))
```

Hierin stelt UC de momentele spanning voor over condensator C. In het programma wordt met een FOR...NEXT-lus begonnen op T=0 (het moment waarop de schakelaar S wordt gesloten) en doorgegaan tot T=1.9. Hierbij wordt na elke berekening de tijd met 0.1 seconde verhoogd. Met de INPUT-statements op de regelnummers 120 tot en met 140 worden respectievelijk de weerstandswaarde, de capaciteit en de batterijspanning ingevoerd.

```
100 REM * RC-KRING *
110 CLS
120 INPUT "R in Ohm  ";R
130 INPUT "C in Farad";C
140 INPUT "U in Volt ";U
150 PRINT
160 FOR T=0 TO 1.8 STEP .1
170 UC=U*(1-EXP(-T/(C*R)))
180 PRINT "t = ";
190 PRINT USING "#.#";T;
200 PRINT "sec";SPC(5);"Uc =";
210 PRINT USING "###.#####";UC;
220 PRINT "V"
230 NEXT T
240 A$=INKEY$:IF A$<>CHR$(13) GOTO 240
250 GOTO 110
```

Opdracht:
Voer de volgende waarden in : R=50000, C=0.00001 en U=8.

3.3 De functie LOG.

Met de functie LOG(X) kunnen we de natuurlijke logaritme (grondtal e) van argument X berekenen.

Met het volgende programma kunnen door middel van INPUT-statements verschillende waarden worden ingevoerd, waarvan vervolgens de natuurlijke logaritmen worden berekend.

```
100 CLS
110 INPUT "Waarden A en B";A,B
115 INPUT "Hoek in graden";G
120 PRINT:PRINT
130 PRINT "LOG(A) =";LOG(A)
140 PRINT
150 PRINT "LOG(A+B) =";LOG(A+B)
160 PRINT
170 R=G*3.14159/180
180 PRINT "LOG(SIN(G)) =";LOG(SIN(R))
190 PRINT:PRINT
200 INPUT "Volgende opgave (J/N)";J$
210 IF J$="J" OR J$="j" GOTO 100
220 END
```

Daar MSX-BASIC geen functie kent voor het berekenen van de normale of Briggse logaritme is de volgende conversieregel erg waardevol:

LOG10(X)=0.434294*LOG(X)

Wanneer in een programma de Briggse logaritme wordt toegepast, is het zinvol voor de Briggse logaritme een functie te definieren. Wanneer we de functie L10 noemen, gaat de definitie er als volgt uitzien:

DEF FNL10(X)=0.434294*LOG(X)

Het volgende programma is nagenoeg gelijk aan het voorgaande, met dien verstande, dat het nu de Briggse logaritme berekent.

```
100 DEF FNL10(X)=.434294*LOG(X)
110 CLS
120 INPUT "Waarden A en B";A,B
130 INPUT "Hoek in graden";G
140 PRINT:PRINT
150 PRINT "LOG10(A) =";
155 PRINT USING "#.#####";FNL10(A)
160 PRINT
170 PRINT "LOG10(A+B) =";
175 PRINT USING "#.#####";FNL10(A+B)
180 PRINT
190 R=G*3.14159/180
200 PRINT "LOG10(SIN(G)) =";
205 PRINT USING "#.#####";FNL10(SIN(R))
210 PRINT:PRINT
220 INPUT "Volgende opgave (J/N)";J$
230 IF J$="J" OR J$="j" GOTO 110
240 END
```

4 De grafische macrotaal

Tot nu toe hebben we nog weinig aandacht besteed aan het tekenen van figuren op het beeldscherm. MSX-BASIC heeft hiervoor een aantal krachtige statements en functies in zijn pakket. In de hoofdstukken 4, 6 en 8 zullen deze uitgebreid aan de orde komen. In dit hoofdstuk zullen we ons voornamelijk bezighouden met het statement DRAW, dat een bijzondere plaats inneemt in de rij van grafische statements.

Het statement DRAW heeft het volgende formaat:

DRAW "alfanumerieke uitdrukking"

Het bijzondere van het statement DRAW is, dat het een aantal grafische subcommando's kent. Deze **subcommando's** worden gespecificeerd in de **alfanumerieke uitdrukking** en hebben een vastgesteld formaat. Het geheel aan subcommando's moet worden gezien als een aparte grafische macrotaal, die ook wel "Graphics Macro Language" (GML) wordt genoemd. De subcommando's bestaan uit een letter plus een numerieke toevoeging (bijvoorbeeld: M10,90) en bevatten de tekeninstructies voor de computer.

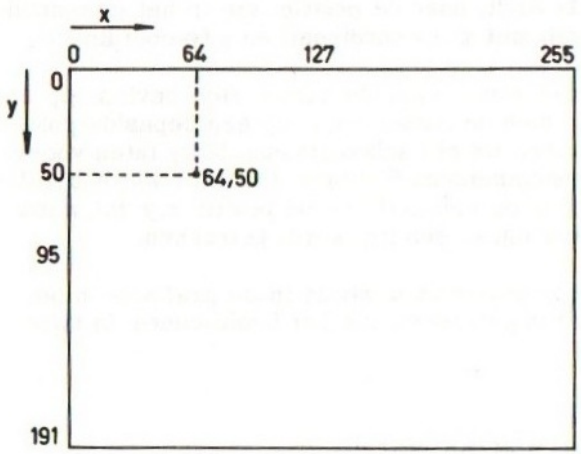
Het statement DRAW kan alleen worden gebruikt in de grafische modes 1 (SCREEN 2) en 2 (SCREEN 3).

Alvorens te beginnen met de uitleg van de verschillende subcommando's uit de grafische macrotaal, zal eerst nog kort worden stilgestaan bij de grafische modes 1 en 2 (zie hiervoor ook deel 1 uit deze serie leerboeken, hoofdstuk 6).

4.1 Grafische modes 1 en 2.

In de grafische mode kan het beeldscherm worden gebruikt voor het uitbeelden van allerlei grafische voorstellingen (figuren enz.) en het tegelijkertijd afdrukken van alfanumerieke gegevens (teksten en bijschriften).

In de grafische mode 1 is het beeldscherm verdeeld in 256 keer 192 beeldpuntjes, ook wel pixels genoemd (zie afbeelding 4-1). Elk beeldpuntje op het scherm wordt bepaald door een X-coordinaat (0-255) en een Y-coordinaat (0-191). In de grafische mode 1 (SCREEN 2) kan op 1 beeldpuntje nauwkeurig worden gepositioneerd en is de kleinste eenheid, waaruit de figuren op het scherm worden opgebouwd, ook 1 beeldpuntje.



Afb. 4-1 Coordinatensysteem voor grafische mode.

De indeling van het scherm in de grafische mode 2 is precies hetzelfde als in de grafische mode 1. Het verschil met de grafische mode 1 is echter, dat de figuren op het beeldscherm in plaats van uit beeldpuntjes worden opgebouwd uit blokjes, welke bestaan uit 4 keer 4 beeldpuntjes. De kleinste eenheid

in de grafische mode 2 (SCREEN 3) bestaat dus uit 4 keer 4 beeldpuntjes. Het positioneren in de grafische mode 2 kan, net zoals in de grafische mode 1, op 1 beeldpuntje nauwkeurig geschieden.

Voor het schoonmaken van het beeldscherm in de grafische modes (1 en 2) kan net zoals in de tekstmodes gebruik worden gemaakt van het statement CLS (CLear Screen).

4.2 Subcommando's voor het trekken van lijnen.

Voor het trekken van lijnen kunnen we gebruik maken van verschillende subcommando's. Het eerste subcommando dat zal worden behandeld is Mx,y. Met het **subcommando Mx,y** (Move) kan een lijn worden getrokken vanaf de positie waar zich de cursor bevindt, naar de positie, die in het commando wordt aangegeven met **x** (x-coordinaat) en **y** (y-coordinaat).

Wanneer men niet weet, waar de cursor zich bevindt op het beeldscherm, en men de cursor eerst op een bepaalde positie wil plaatsen, kunnen we het subcommando Mx,y laten voorafgaan door het **subcommando B** (Blank) Hiermee wordt bereikt dat de cursor naar de gespecificeerde positie x,y zal worden verplaatst, zonder dat er een lijn wordt getrokken.

Met het volgende programma wordt in de grafische mode 1 een horizontale lijn getrokken, die het beeldscherm in tweeen deelt.

```
10 SCREEN 2
20 DRAW "BM0,95M255,95"
30 GOTO 30
```

In het DRAW-statement staan in de alfanumerieke uitdrukking een aantal instructies voor de computer, namelijk:

BM0,95
Verplaats de cursor naar positie x=0 en y=95, zonder een

lijn te trekken.

M255,95
Trek een lijn vanaf de cursorpositie (in ons voorbeeld 0,95) naar positie x=255 en y=95.

De verschillende subcommando's kunnen in de alfanumerieke uitdrukking achter elkaar worden geplaatst zonder tussen de subcommando's scheidingstekens te plaatsen.

Laten we nu eens een horizontale en een verticale lijn trekken, zodat het beeldscherm in vieren wordt gedeeld. Hiervoor zal, na het trekken van de horizontale lijn, de cursor eerst naar positie x=127 en y=0 moeten worden verplaatst, zonder een lijn te trekken. Dit kunnen we bewerkstelligen met het subcommando BM127,0. Voor het trekken van de verticale lijn dient het subcommando M127,191. Het programma gaat er dan als volgt uitzien:

```
10 SCREEN 2
20 DRAW "BM0,95M255,95BM127,0M127,191"
30 GOTO 30
```

We kunnen het scherm ook in vieren verdelen door vanuit het middelpunt van het beeldscherm (x=127 en y=95) vier lijnen naar de randen van het scherm te trekken. We komen dan echter wel voor een probleem te staan, namelijk dat de cursor na het trekken van de lijn steeds weer naar het middelpunt terug moet worden geplaatst, voordat de volgende lijn kan worden getrokken. In de grafische macrotaal kennen we voor het terugplaatsen van de cursor naar de oorspronkelijke positie het **subcommando N.** Door nu subcommando N voor subcommando Mx,y te plaatsen, zal de cursor na het trekken van de lijn weer worden teruggeplaatst naar zijn oorspronkelijke beginpositie.

Voor het verdelen van het scherm volgens voorgaande procedure, zal de alfanumerieke uitdrukking van het DRAW-statement de volgende subcommando's moeten bevatten:

BM127,95NM0,95NM255,95NM127,0NM127,191

Met het subcommando BM127,95 wordt de cursor verplaatst naar positie x=127 en y=95, zonder dat er een lijn wordt getrokken. Met de M-subcommando's, die worden voorafgegaan door het subcommando N, zal na het trekken van de lijn de cursor weer worden teruggeplaatst naar de oorspronkelijke beginpositie. Het volgende programma toont hoe het beeldscherm vanuit het middelpunt in vieren wordt gedeeld.

```
10 SCREEN 2
20 DRAW "BM127,95NM0,95NM255,95NM127,0NM127,191"
30 GOTO 30
```

4.3 Tekst naar het grafisch scherm schrijven.

Het is ook mogelijk om alfanumerieke gegevens (teksten en bijschriften) naar het grafisch beeldscherm (SCREEN 2 en 3) te schrijven. Daar een grafisch beeldscherm in MSX-BASIC ook als randapparaat wordt beschouwd, dient het eerst te worden geopend, alvorens er gegevens naar toe kunnen worden geschreven. Dit openen geschiedt met het volgende statement:

```
OPEN "GRP:" AS #X
```

In dit statement stelt **"GRP:"** de naam van het grafisch scherm voor en is **X** een referentie, die in andere statements moet worden gebruikt om naar het grafisch scherm te kunnen schrijven. Het kanaalnummer X is een numerieke constante, die een minimale waarde kan hebben van 1 en een maximale waarde, die overeenkomt met de waarde van de systeemvariabele MAXFILES. De default-waarde van MAXFILES is 1.

De alfanumerieke gegevens kunnen met een PRINT-statement naar het grafisch scherm worden geschreven. De gegevens worden vanaf de actuele positie van de cursor op het scherm geschreven. Voor SCREEN 2 is de actuele positie van de

cursor, die op 1 beeldpuntje nauwkeurig is gepositioneerd, de linker bovenhoek van de 8*8 (beeldpuntjes) matrix, waarin het eerste teken wordt geplaatst. We zullen nu het vorige programma zodanig uitbreiden, dat in de vier velden respectievelijk de teksten VELD 1, VELD 2, VELD 3 en VELD 4 worden geschreven.

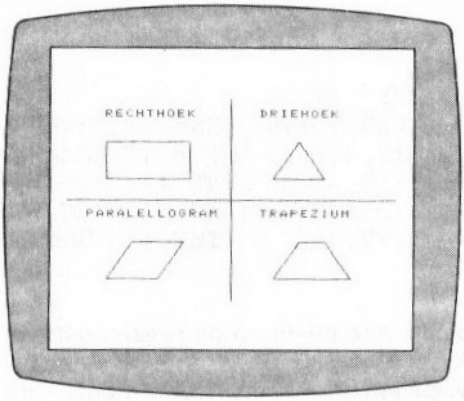
```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 DRAW "BM127,95NM0,95NM255,95NM127,0NM127,191"
40 DRAW "BM32,40":PRINT #1,"VELD 1"
50 DRAW "BM159,40":PRINT #1,"VELD 2"
60 DRAW "BM32,135":PRINT #1,"VELD 3"
70 DRAW "BM159,135":PRINT #1,"VELD 4"
80 GOTO 80
```

Met de DRAW-statements op de regelnummers 40 tot en met 70 plaatsen we de cursor op de positie waar we de tekst willen laten beginnen. In de grafische mode 1 en 2 kunnen we voor het positioneren van de cursor niet het statement LOCATE gebruiken, daar dit statement de cursor alleen maar kan positioneren op 1 tekenpositie (letter, cijfer of leesteken) nauwkeurig en niet op 1 beeldpuntje nauwkeurig.

Met het volgende programma worden in de vier velden van het beeldscherm respectievelijk een rechthoek, een driehoek, een paralellogram en een trapezium getekend. Bovendien wordt elke figuur nog van een bijschrift voorzien.

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 DRAW "BM0,95M255,95BM127,0M127,191"
40 DRAW "BM30,75M95,75M95,40M30,40M30,75"
50 DRAW "BM30,10":PRINT #1,"RECHTHOEK"
60 DRAW "BM160,75M200,75M180,40M160,75"
70 DRAW "BM150,10":PRINT #1,"DRIEHOEK"
80 DRAW "BM30,170M70,170M90,135M50,135M30,170"
90 DRAW "BM15,105":PRINT #1,"PARALELLOGRAM"
100 DRAW "BM160,170M220,170M200,135M180,135M16
```

```
0,170"
110 DRAW "BM150,105":PRINT #1,"TRAPEZIUM"
120 GOTO 120
```



Opmerking:

Raadpleeg voor het analyseren van het programma afbeelding 4-1.

4.4 Vervolg subcommando's voor het trekken van lijnen.

De x- en y-parameters in het subcommando Mx,y kunnen een absolute of een relatieve waarde hebben. Tot nu toe hebben we alleen gewerkt met **absolute waarden**, dat wil zeggen, dat de waarden de coordinaten zijn, waarheen de lijn moet worden getrokken. Het is echter ook mogelijk om **relatieve waarden** te gebruiken. In dat geval moeten x en y worden voorafgegaan door een **plusteken (+)** of een **minteken (-)**. De x- en y-coordinaten worden dan relatief bepaald ten opzichte van de cursorpositie. Het volgende voorbeeld illustreert dit:

Cursor-positie	Sub-commando	Uitwerking
x=40,y=80	M90,60	Er wordt een lijn getrokken vanaf positie 40,80 naar positie 90,60
	M+50,-20	

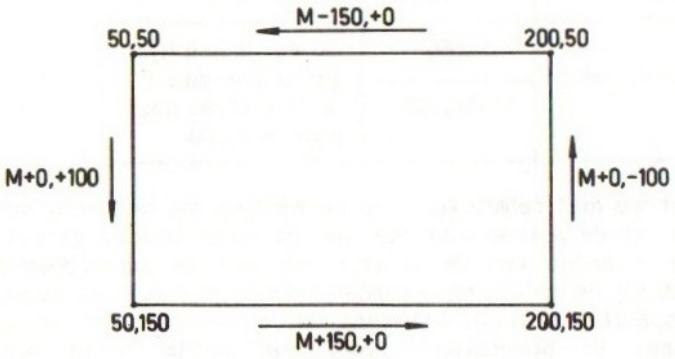
Wanneer we met relatieve waarden werken, worden voor het bepalen van de positie waarnaar de lijn moet worden getrokken, de waarden van de coordinaten van de cursorpositie opgeteld bij de relatieve waarden, die in het subcommando zijn gespecificeerd. Dit betekent dat de waarde van de x-coordinaat in bovenstaand voorbeeld gelijk wordt aan (40)+(+50)=90 en de waarde van de y-coordinaat gelijk wordt aan (80)+(-20)=60.

De volgende twee programma's, waarvan het ene werkt met absolute waarden en het andere met relatieve waarden, leveren hetzelfde resultaat, namelijk het tekenen van dezelfde rechthoek.

```
10 REM * ABSOLUTE WAARDEN"
20 OPEN "GRP:" AS #1
30 SCREEN 2
40 DRAW "BM50,150M200,150M200,50M50,50M50,150"
50 DRAW "BM60,95":PRINT #1,"ABSOLUTE WAARDEN"
60 GOTO 60
```

```
10 REM * RELATIEVE WAARDEN *
20 OPEN "GRP:" AS #1
30 SCREEN 2
40 DRAW "BM50,150M+150,+0M+0,-100M-150,+0M+0,+
100"
50 DRAW "BM60,95":PRINT #1,"RELATIEVE WAARDEN"
60 GOTO 60
```

Afbeelding 4-2 geeft een overzicht van wat er gebeurt, wanneer er met relatieve waarden wordt gewerkt.



Afb. 4-2 Relatieve positionering in grafische mode.

Opmerking:

Ook de waarde 0, als relatieve waarde, moet worden voorafgegaan door een plus- of minteken. Welk teken u kiest is niet belangrijk.

Met de volgende subcommando's kunnen, te beginnen vanaf de cursorpositie, rechte lijnen worden getrokken in acht verschillende richtingen.

Un

Het subcommando Un (Up = omhoog) zal vanaf de cursorpositie een lijn recht naar boven trekken, met een lengte van n beeldpuntjes.

Dn

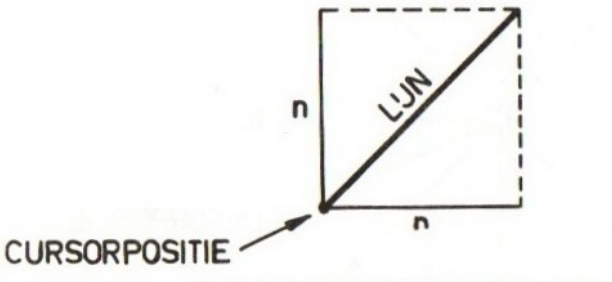
Het subcommando Dn (Down = naar beneden) zal vanaf de cursorpositie een lijn recht naar beneden trekken, met een lengte van n beeldpuntjes.

Ln

Het subcommando Ln (Left = links) zal vanaf de cursorpositie een lijn naar links trekken, met een lengte van n beeldpuntjes.

Rn
Het subcommando Rn (Right = rechts) zal vanaf de cursorpositie een lijn naar rechts trekken, met een lengte van n beeldpuntjes.

En
Het subcommando En zal vanaf de cursorpositie een lijn schuin naar boven trekken naar een punt, dat n beeldpuntjes hoger en n beeldpuntjes meer naar rechts ligt dan het beginpunt.



Fn
Het subcommando Fn zal vanaf de cursorpositie een lijn schuin naar beneden trekken naar een punt, dat ten opzichte van de cursorpositie n beeldpuntjes lager en n beeldpuntjes meer naar rechts ligt.

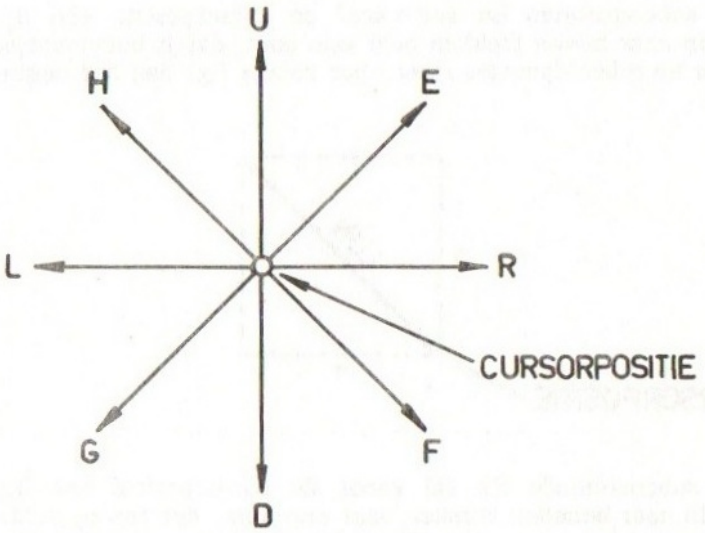
Gn
Het subcommando Gn zal vanaf de cursorpositie een lijn schuin naar beneden trekken naar een punt, dat ten opzichte van de cursorpositie n beeldpuntjes lager en n beeldpuntjes meer naar links ligt.

Hn
Het subcommando Hn zal vanaf de cursorpositie een lijn schuin omhoog trekken naar een punt dat, ten opzichte van de cursorpositie n beeldpuntjes hoger ligt en n beeldpuntjes meer naar links.

Ook de subcommando's Un, Dn, Ln, Rn, En, Fn, Gn en Hn

kunnen worden voorafgegaan door een van de subcommando's B of N (zie paragraaf 4.2).

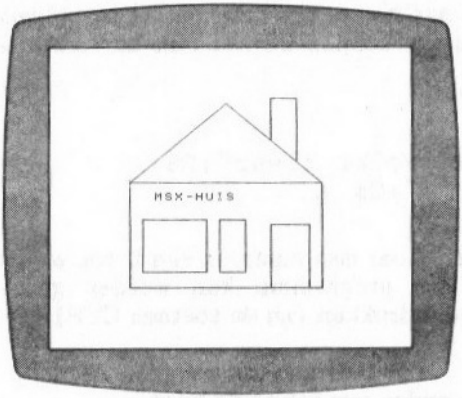
Het geheel aan voorgaande subcommando's wordt nog eens geïllustreerd in afbeelding 4-3.



Afb. 4-3 Subcommando's voor cursorbewegingen.

In het volgende programma zullen de verschillende subcommando's worden toegepast, voor het tekenen van een huis.

```
200 SCREEN 2
210 OPEN "GRP:" AS #1
220 DRAW "BM50,180R150U100L150D100BU100E75F75"
230 DRAW "BM60,165R50U50L50D50"
240 DRAW "BM120,165R20U50L20D50"
250 DRAW "BM160,40U60R20D80"
260 DRAW "BM160,180U60R30D60"
270 DRAW "BM70,90":PRINT #1,"MSX-HUIS"
280 GOTO 280
```



Met regel 220 worden de contouren van het huis getekend, met regelnummer 230 het grote raam, met regelnummer 240 het kleine raam, met regelnummer 250 de schoorsteen, met regelnummer 260 de deur en met regelnummer 270 wordt de naam op het huis geschreven.

4.5 Tekenen in een bepaalde kleur.

Met het **subcommando Cn** (Color) kunnen we opgeven in welke kleur (voorgrondkleur) we een figuur willen tekenen. In het subcommando stelt n het kleurnummer voor (0-15). Wanneer subcommando Cn niet wordt gebruikt, wordt de laatst gedefinieerde voorgrondkleur gebruikt. De default-waarde voor de kleur is 15 (wit). De volgende kleuren kunnen worden gespecificeerd:

0 = transparant
1 = zwart
2 = groen
3 = lichtgroen
4 = donkerblauw
5 = lichtblauw
6 = donkerrood
7 = cyaan
8 = rood
9 = lichtrood
10 = donkergeel
11 = lichtgeel
12 = donkergroen
13 = paars
14 = grijs
15 = wit

Wanneer we aan het vorige programma de volgende regelnummers toevoegen, kunnen we het huis in verschillende kleuren tekenen.

```
180 CLS
190 INPUT "Welke kleur";C$
205 DRAW "C"+C$
```

Voor de kleur moet een nummer van 0 tot en met 15 worden ingevoerd. Het programma kan worden gestopt door het tegelijkertijd indrukken van de toetsen CTRL en STOP.

4.6 Tekenen onder een bepaalde hoek.

De grafische macrotaal maakt het ook mogelijk een gespecificeerde figuur in een bepaalde stand (hoek) te plaatsen. De hoek kan worden aangegeven met het **subcommando An** (Angle = hoek), waarin n de waarde 0, 1, 2 of 3 kan hebben, met de volgende betekenis.

A0 = 0 graden
A1 = 90 graden
A2 = 180 graden
A3 = 270 graden

Met het volgende programma kan de invloed van het subcommando An worden geïllustreerd. Het programma kan een vlag in vier standen tekenen, waarbij ook nog voor de kleur kan worden gekozen. De hoek (0, 1, 2 of 3), waaronder de vlag moet worden getekend, kan worden ingegeven met het INPUT-statement op regelnummer 110.

```
100 CLS
110 INPUT "Hoek ";A$
120 INPUT "Kleur";C$
130 SCREEN 2
140 DRAW "C"+C$
150 DRAW "BM127,95A"+A$+"NR80D40R42U40BL14D40
BL14U40"
```

```
160 FOR I=1 TO 5000:NEXT I
170 GOTO 100
```

Wanneer het subcommando An niet wordt gebruikt, zal de laatst gedefinieerde hoek worden gebruikt. De default-waarde voor de hoek is 0 (0 graden).

4.7 Tekenen in een bepaalde schaal.

Met het **subcommando Sn** (Scale) kan worden aangegeven in welke schaal de computer een tekening moet uitvoeren. Met andere woorden, we kunnen de computer instrueren om de afmetingen van een tekening te vergroten of te verkleinen. In het subcommando kan n een waarde hebben van 1 tot en met 255, waarbij n gedeeld door 4 de schaalfactor is. De gegeven afstand in de subcommando's U, D, L, R, E, F, G, H en de relatieve x en y waarden in het subcommando M worden vermenigvuldigd met de schaalfactor. Het resultaat is dan de te verplaatsen afstand.

Met het statement DRAW "S6U50" wordt een lijn naar boven getrokken, echter niet met een lengte van 50 beeldpuntjes, zoals aangegeven door het subcommando U50, maar met een lengte van 75 beeldpuntjes (50 x 6 / 4 = 75), als resultaat van het subcommando S6.

Wanneer het subcommando Sn niet wordt gebruikt, zal de laatst gedefinieerde schaalfactor worden gebruikt. De default-waarde voor de schaal is 4 (schaalfactor 1).

Met het volgende programma kan een tekening van een huis in verschillende schalen worden uitgevoerd. De schaal kan door middel van het INPUT-statement op regelnummer 110 worden ingevoerd.

```
100 CLS
110 INPUT "Schaal";S$
120 SCREEN 2
130 DRAW "A0C15"
```

```
140 DRAW "BM50,180S"+S$+"R6U4L6D4BU4E3F3"
150 FOR I=1 TO 5000:NEXT I
160 GOTO 100
```

Opmerking:
Met het DRAW-statement op regelnummer 130 worden de default-waarden voor de hoek (0 = 0 graden) en de kleur (15 = wit) opnieuw ingesteld.

Het programma kan ook als volgt worden geschreven.

```
100 CLS
110 INPUT "Schaal";S$
120 SCREEN 2
130 DRAW "S"+S$
140 DRAW "BM50,180R6U4L6D4BU4E3F3"
150 FOR I=1 TO 5000:NEXT I
160 GOTO 100
```

4.8 Uitvoeren van subcommando's.

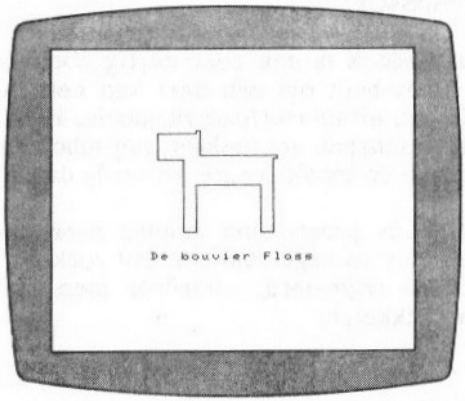
Het is mogelijk om subcommando's van de grafische macrotaal, die nodig zijn voor het tekenen van een bepaalde figuur, eerst toe te kennen aan een alfanumerieke variabele (bijvoorbeeld A$) en vervolgens uit te voeren met het subcommando X. Het formaat van het **subcommando X** ziet er als volgt uit:

X alfanumerieke variabele ;

In het volgende programma bevat de alfanumerieke variabele A$ de subcommando's voor het tekenen van een hond. De figuur wordt getekend door het subcommando X in het DRAW-statement.

```
100 SCREEN 2
110 OPEN "GRP:" AS #1
120 A$="R10U45R50D45R10U70R3U3L60U24L3D3L30D25R
20D68"
```

```
130 DRAW "A0S4C1BM80,100XA$;"
140 DRAW "BM55,120":PRINT #1,"De bouvier Floss"
150 GOTO 150
```



In het DRAW-statement zijn behalve het subcommando XA$; ook de subcommando's A0, S4, C1 en BM40,180 opgenomen. De subcommando's A0 en S4 dienen ervoor om de hoek in te stellen op 0 graden en de schaalfactor op 4:4=1. Met het subcommando C1 wordt de kleur op zwart ingesteld en met het subcommando BM40,180 wordt bepaald, waar wordt begonnen met het tekenen van de hond.

Wanneer men er zeker van wil zijn dat de default-waarden voor de hoek (0 = 0 graden), de schaal (4 = schaalfactor 1) en de voorgrondkleur (15 = wit) zijn ingesteld, zal men in de alfanumerieke uitdrukking van het DRAW-statement de volgende reeks subcommando's moeten opnemen:

DRAW "A0S4C15....."

- gegevens voor het tekenen van de figuur.
- voorgrondkleur wit.
- schaalfactor 1.
- hoek 0 graden.

Het is ook mogelijk om in het begin van het programma een DRAW-statement op te nemen, welke alleen de default-waarden instelt. Het statement gaat er dan als volgt uitzien:

```
DRAW "A0S4C15"
```

Het subcommando X is een zeer nuttig commando, omdat u hiermee in staat bent om een deel van een figuur apart te definieren in een alfanumerieke variabele. Hierdoor is dan de mogelijkheid geschapen om reeksen van subcommando's uit te voeren, waarvan de totale lengte groter is dan 255.

Met het volgende programma kunnen door middel van het INPUT-statement op regelnummer 110 reeksen met subcommando's worden ingevoerd, waardoor men zelf in staat is figuren te ontwikkelen.

```
100 CLS
110 INPUT "Subcommando's";A$
120 SCREEN 2
130 DRAW "A0C1BM50,100XA$;"
140 GOTO 140
```

Het programma tekent de figuren in zwart (C1). Nieuwe figuren kunnen worden ingevoerd na het indrukken van de CTRL-toets en STOP-toets, en het vervolgens weer starten van het programma met RUN.

Alle constanten (x, y en n) in de subcommando's kunnen worden vervangen door numerieke variabelen. Het formaat hiervoor ziet er als volgt uit:

Subcommando = numerieke variabele ;

De numerieke variabele moet worden voorafgegaan door een is-gelijk-teken (=) en eindigen met een puntkomma (;). Voorbeeld:

```
80 X=40
90 DRAW "U=X;"
```

Met het volgende eenvoudige programma kunnen we rechthoeken tekenen, waarvan de lengten van de rechthoekszijden kunnen worden ingevoerd met een INPUT-statement.

```
100 CLS
110 INPUT "Lengte zijden A en B";A,B
120 SCREEN 2
130 DRAW "BM40,150R=A;U=B;L=A;D=B;"
140 FOR I=1 TO 5000:NEXT I
150 GOTO 100
```

De volgende voorbeelden tonen hoe numerieke variabelen in het subcommando M kunnen worden toegepast:

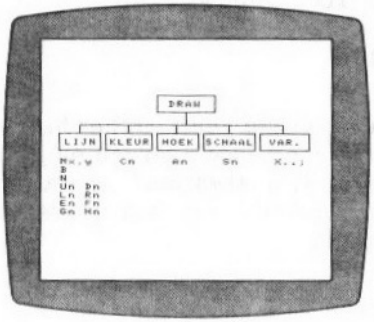
```
DRAW "M+=X1;,-=Y1;"
DRAW "M=X2;,=Y2;"
```

In het volgende programma wordt gebruik gemaakt van het subcommando X en de numerieke variabelen in het subcommando M. Het programma tekent vier driehoeken in verschillende schalen.

```
100 SCREEN 2
110 X=50:Y=90
120 DRAW "BM=X;,=Y;"
130 FOR I=0 TO 3
140 A$="S"+STR$(I*4+4)+"R40H20G20"
150 DRAW "XA$;"
160 FOR J=1 TO 2000:NEXT J
170 NEXT I
180 GOTO 180
```

Het volgende programma geeft op het beeldscherm een overzicht van alle subcommando's binnen de grafische macrotaal. U kunt dit programma eventueel op cassette bewaren. Het dient alleen als voorbeeld van hoe u overzichten kan maken op het beeldscherm.

```
100 SCREEN 2
110 OPEN "GRP:" AS #1
120 DRAW "A0S4C15"
130 DRAW "BM100,5R56D20L56U20BM112,12"
140 PRINT #1,"DRAW"
150 DRAW "BM5,50R40D20L40U20BM10,40"
170 DRAW "BM50,50R45D20L45U20"
180 DRAW "BM100,50R40D20L40U20"
190 DRAW "BM145,50R50D20L50U20"
200 DRAW "BM200,50R48D20L48U20"
210 DRAW "BM11,57":PRINT #1,"LIJN"
220 DRAW "BM55,57":PRINT #1,"KLEUR"
230 DRAW "BM106,57":PRINT #1,"HOEK"
240 DRAW "BM148,57":PRINT #1,"SCHAAL"
250 DRAW "BM210,57":PRINT #1,"VAR."
260 DRAW "BM25,50U10R47ND10R47ND10R51ND10R52ND
10BM128,25D15"
270 DRAW "BM7,80":PRINT #1,"Mx,y"
280 DRAW "BM65,80":PRINT #1,"Cn"
290 DRAW "BM116,80":PRINT #1,"An"
300 DRAW "BM165,80":PRINT #1,"Sn"
310 DRAW "BM215,80":PRINT #1,"X..;"
320 DRAW "BM7,90":PRINT #1,"B"
330 DRAW "BM7,100":PRINT #1,"N"
340 DRAW "BM7,110":PRINT #1,"Un Dn"
350 DRAW "BM7,120":PRINT #1,"Ln Rn"
360 DRAW "BM7,130":PRINT #1,"En Fn"
370 DRAW "BM7,140":PRINT #1,"Gn Hn"
380 GOTO 380
```



5 Besturing van buitenaf

Bij spelprogramma's is het meestal belangrijk, dat we van buitenaf invloed kunnen uitoefenen op bepaalde gebeurtenissen en voorvallen, die door middel van het programma op het beeldscherm plaatsvinden. Voorbeelden hiervan zijn, het verplaatsen van een figuur van de ene naar de andere plaats, het richten op een bepaald doel en het vervolgens afvuren van een schot op dit doel, het reageren op bepaalde vragen en voorvallen (reactiespelletjes) enz.

Op de MSX-computer kunnen hiervoor op de periferiepoorten 1 en 2 respectievelijk 1 of 2 joy-sticks worden aangesloten. Een joy-stick bestaat uit een stuurknuppel, die in acht posities kan worden geplaatst, en 1 of 2 actieknoppen, die ook wel vuurknoppen worden genoemd.

De posities van de stuurknuppel kunnen door middel van de functie STICK(n) worden opgevraagd. Het al of niet ingedrukt zijn van een actieknop kan worden afgevraagd met de functie STRIG(n). Hierna kan, afhankelijk van de status van de actieknop een bepaalde actie worden ondernomen. Het is ook mogelijk om, als gevolg van het indrukken van de actieknop, door middel van het statement ON STRIG GOSUB een bepaalde subroutine aan te roepen. Hiervoor zal eerst de betreffende actieknop moeten worden geactiveerd, waarvoor het statement STRIG(n) ON dient. Voor het deactiveren van de actieknop dienen de statements STRIG(n) OFF en STRIG(n) STOP. De laatste statements kunnen worden vergeleken met de statements ON KEY GOSUB en KEY(n) ON/OFF/STOP.

Buiten de joy-sticks kan op de periferiepoorten ook een "pad" worden aangesloten. Met een "pad" (tableau) kunnen op

digitale wijze tekeningen worden ingevoerd in de computer, vandaar dat dit apparaat ook wel digitizer wordt genoemd. De digitizer bestaat uit een paneel, waarop met een speciale tekenstift tekeningen of figuren kunnen worden overgetrokken. De tekenstift is voorzien van een drukknop (schakelaar). Door nu de tekenstift op een bepaalde positie op het paneel neer te zetten en voor elke positie de knop in te drukken,kan een tekening worden gemaakt en gecodeerd naar de computer worden gestuurd. Voor al deze acties dient de functie PAD(n).

Als laatste zal in dit hoofdstuk de functie PDL(n) worden behandeld. Met deze functie kan de positie van een regelknop (paddle) worden afgevraagd. De regelknop is meestal ook voorzien van een actieknop. De regelknop kan worden aangesloten op de periferiepoorten.

5.1 Het gebruik van de joy-stick.

De werking van de joy-stick zal worden verklaard met behulp van het volgende reactiespel. Op het beeldscherm zal steeds de naam van een land verschijnen. Het is de bedoeling, dat met de joy-stick hierop onmiddellijk wordt gereageerd door aan te geven of het land in Europa, Amerika, Afrika of Azie ligt. Wanneer het land in Europa ligt, zal de joy-stick in de positie "Noord" (naar boven) moeten worden geplaatst (zie afbeelding 5-1). Ligt het land in Amerika, dan moet de joy-stick in de positie "Zuid" (naar beneden) worden geplaatst. Afrika is "West" (links) en Azie "Oost" (rechts).

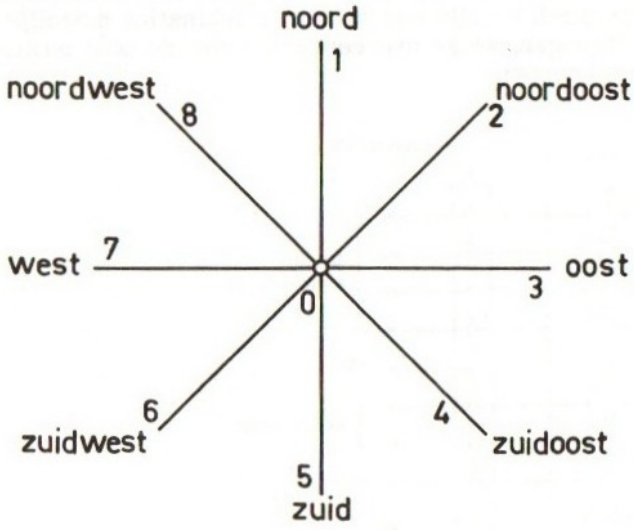
Met de **functie STICK(n)** kan in MSX-BASIC de ingestelde positie (richting) van een joy-stick aan de computer worden opgevraagd, zodat voor ons voorbeeld kan worden gecontroleerd of het antwoord al of niet juist is. Met parameter **n**, welke 0, 1 of 2 kan zijn, wordt aangegeven of de cursortoetsen (pijltjestoetsen), of een van de joy-sticks is geselecteerd.

n = 0
De cursortoetsen (pijltjestoetsen) zijn als joy-stick geselecteerd.

n = 1
De joy-stick, aangesloten op connector 1, is geselecteerd.

n = 2
De joy-stick, aangesloten op connector 2, is geselecteerd.

Opmerking:
Wanneer men geen joy-stick heeft, kunnen ook de cursor-toetsen (n=0) als joy-stick worden gebruikt.



Afb. 5-1 Joy-stick richtingen.

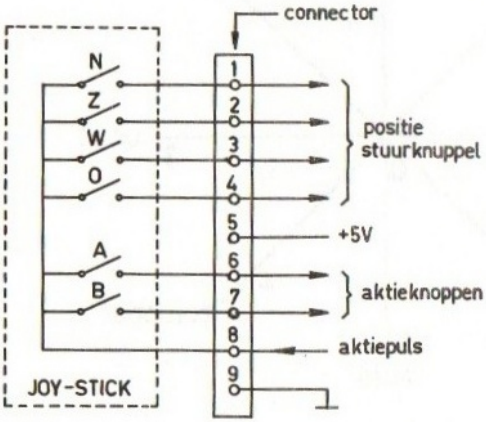
De functie STICK(n) kan 9 verschillende numerieke waarden als resultaat geven. De waarden vertegenwoordigen elk een bepaalde positie. De volgende tabel toont de waarden met de bijbehorende posities.

0 = neutraal	
1 = boven	(noord)
2 = rechtsboven	(noordoost)
3 = rechts	(oost)

4 = rechtsonder	(zuidoost)
5 = onder	(zuid)
6 = linksonder	(zuidwest)
7 = links	(west)
8 = linksboven	(noordwest)

Afbeelding 5-2 toont hoe een joy-stick is aangesloten op de MSX-computer.

De vier contacten van de joy-stick worden bediend door de stuurknuppel. Er zijn een achttal combinaties mogelijk op de vier uitgangen, welke overeenkomen met de acht posities van de stuurknuppel.



Afb. 5-2 Aansluitschema van de joy-stick.

De stuurknuppel van een joy-stick kan buiten de neutrale positie in acht verschillende posities worden geplaatst. Om met de 4 cursortoetsen ook 8 posities te kunnen bereiken, zullen voor de even posities (2, 4, 6 en 8) steeds 2 cursortoetsen tegelijkertijd moeten worden ingedrukt om de gewenste richting te verkrijgen. Voor de oneven posities hoeft maar 1 cursortoets te worden ingedrukt.

Alvorens door te gaan met het voorbeeld uit het begin van deze paragraaf, zal eerst een eenvoudig demonstratieprogramma worden gegeven. Hiervoor zal op connector 1 een joy-stick moeten worden aangesloten. Van de joy-stick zal alleen de stuurknuppel worden gebruikt en niet de actieknop(-pen). Met dit programma kunnen de verschillende posities worden gecontroleerd.

```
100 CLS
110 PRINT "POSITIES VAN DE JOY-STICK"
120 PRINT
130 P=STICK(1)
140 IF P=0 GOTO 130
150 RESTORE
160 FOR I=1 TO P
170 READ P$
180 NEXT I
190 PRINT "De positie is ";P$
200 A$=INKEY$:IF A$<>CHR$(13) GOTO 200
210 GOTO 120
220 DATA noord (boven),noordoost (rechtsboven)
230 DATA oost (rechts),zuidoost (rechtsonder)
240 DATA zuid (onder),zuidwest (linksonder)
250 DATA west (links),noordwest (linksboven)
```

Opmerking:

Om de computer de volgende positie te laten bekijken, moeten we eerst de stuurknuppel in de nieuwe stand zetten en dan op RETURN drukken.

Gebruik ook eens de cursor-toetsen als joy-stick. Hiervoor zal regelnummer 130 als volgt moeten worden veranderd: 130 P=STICK(0)

Met het volgende programma kan het landenspel, waarvan de spelregels in het begin van deze paragraaf staan, worden gespeeld. Sluit hiervoor een joy-stick op connector 1 aan.

```
100 CLS
110 WIDTH 40
```

```
120 PRINT "L A N D E N S P E L"
130 PRINT
140 RESTORE
150 FOR I=1 TO 12
160 READ A,L$,W$
170 PRINT L$;" ligt in?";TAB(21);
180 P=STICK(1):IF P=0 GOTO 180
190 IF A=P THEN PRINT W$:GOTO 210
200 PRINT "Antwoord is fout!"
210 A$=INKEY$:IF A$<>CHR$(13) GOTO 210
220 PRINT
230 NEXT I
240 DATA 7,Egypte,Afrika
250 DATA 5,Peru,Amerika
260 DATA 5,Argentinie,Amerika
270 DATA 3,Korea,Azie
280 DATA 7,Angola,Afrika
290 DATA 1,Roemenie,Europa
300 DATA 1,Polen,Europa
310 DATA 7,Goudkust,Afrika
320 DATA 3,Jordanie,Azie
330 DATA 5,Panama,Amerika
340 DATA 7,Kenia,Afrika
350 DATA 1,Portugal,Europa
360 END
```

Speel het spel ook eens door gebruik te maken van de cursortoetsen in plaats van de joy-stick. Vergeet hiervoor niet regelnummer 180 te veranderen.

5.2 Toepassing van de actieknoppen.

In het programma van het landenspel zouden we met het indrukken van de actieknop kunnen aangeven, dat de stuurknuppel nu in de gewenste positie staat en dat het programma verder kan gaan met onder andere het opvragen van de ingestelde positie. De nieuw ingestelde positie is zodoende pas relevant voor het programma, wanneer de actieknop wordt ingedrukt. We kunnen de stuurknuppel dan rustig instellen,

want zolang er niet op de actieknop wordt gedrukt, zal de ingestelde positie van de stuurknuppel nog niet worden afgevraagd.

Met de **functie STRIG(n)** kan worden getest of de actieknop van een joy-stick al dan niet is ingedrukt. Parameter **n** van de functie kan een waarde hebben van 0, 1, 2, 3 of 4. De betekenis hiervan is:

n = 0
De spatiebalk is als actieknop geselecteerd.

n = 1 of 3
De eerste respectievelijk tweede actieknop van de joy-stick, die is aangesloten op connector 1, is geselecteerd.

n = 2 of 4
De eerste respectievelijk tweede actieknop van de joy-stick, die is aangesloten op connector 2, is geselecteerd.

Wanneer de geselecteerde spatiebalk of actieknop niet is ingedrukt, levert de functie STRIG(n) de waarde 0. Is daarentegen de spatiebalk of actieknop wel ingedrukt, dan levert de functie STRIG(n) de waarde -1.

Met het volgende programma kan de toepassing van de spatiebalk als actieknop en de toepassing van de actieknoppen op de joy-sticks 1 en 2 worden gedemonstreerd.

```
100 CLS
110 PRINT "SELECTIE AKTIEKNOP"
120 PRINT
130 PRINT "0. Spatiebalk als aktieknop"
140 PRINT "1. Aktieknop joy-stick nr.1"
150 PRINT "2. Aktieknop joy-stick nr.2"
160 PRINT "3. Programma beeindiging"
170 PRINT
180 INPUT "Uw keuze";K
190 IF K<0 OR K>3 THEN PRINT "Verkeerde keuze
```

```
!":GO TO 180
200 IF K=3 THEN END
210 PRINT "Druk geselecteerde aktietoets in!"
220 S=STRIG(K)
230 IF S=0 GOTO 220
240 FOR I=1 TO 20
250 BEEP
260 NEXT I
270 GOTO 100
```

Voor het toepassen van de actieknop in het landenprogramma (paragraaf 5.1) zullen in het programma de volgende veranderingen moeten worden aangebracht.

```
175 S=STRIG(1): IF S=0 GOTO 175
180 P=STICK(1)
```

Wanneer het land op het scherm verschijnt, kan de stuurknuppel in de gewenste positie worden geplaatst. Zolang de actieknop nog niet is ingedrukt, kan de stuurknuppel nog van positie worden veranderd. Wordt er eenmaal op de actieknop gedrukt (regelnummer 175), dan zal de actuele positie van de stuurknuppel aan de numerieke variabele P worden toegekend (regelnummer 180).

5.3 Programma-onderbreking d.m.v. actieknoppen.

De spatiebalk en de actieknoppen van de joy-sticks kunnen ook worden gebruikt voor het onderbreken van programma's. Het is hierbij de bedoeling, dat bij het indrukken van de spatiebalk of een actieknop het programma dat in uitvoering is, tijdelijk wordt onderbroken en dat er een subroutine, die betrekking heeft op de ingedrukte spatiebalk of actieknop, wordt aangeroepen. Een onderbreking wordt pas in bewerking genomen, wanneer het statement dat in uitvoering is op het moment dat de onderbreking plaatsvindt, volledig is afgewerkt. Een onderbreking wordt ook wel gezien als een niet geprogrammeerde sprong.

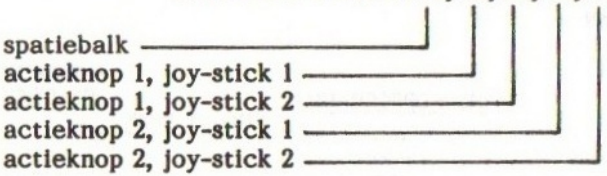
Opmerking:

In de rest van dit hoofdstuk zal de spatiebalk, toegepast als actieknop, niet meer expliciet worden genoemd. De spatiebalk zal verder als normale actieknop worden gezien.

Het statement RETURN aan het einde van de aangeroepen subroutine zorgt ervoor, dat het programma de draad weer oppakt waar het eerder, tijdens het indrukken van de actieknop, werd onderbroken. Een onderbreking vindt niet plaats, wanneer de computer niet bezig is met het uitvoeren van een programma.

Met behulp van het statement

ON STRIG GOSUB R1,R2,R3,R4,R5

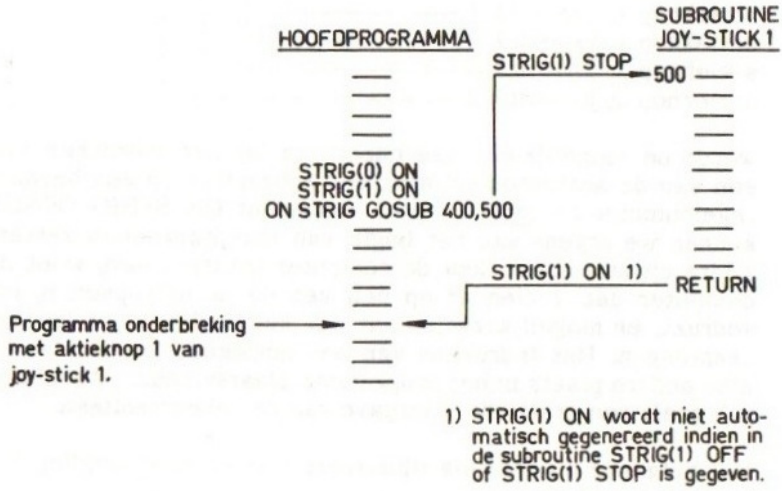


wordt de mogelijkheid geschapen om bij het indrukken van een van de actieknoppen naar een subroutine op een bepaald regelnummer te springen. Het statement ON STRIG GOSUB kunnen we ergens aan het begin van het programma zetten. Zodra dit statement door de computer is uitgevoerd, weet de computer dat, indien er op een van de actieknoppen wordt gedrukt, er mogelijkerwijs naar een subroutine moet worden gesprongen. Het indrukken van een actieknop kan echter op elke andere plaats in het programma plaatsvinden. Afbeelding 5-3 geeft een grafische weergave van de gebeurtenissen.

Het volgende programma illustreert wat er in afbeelding 5-3 wordt bedoeld.

```
100 CLS
110 STRIG(0) ON
120 STRIG(1) ON
```

```
130 ON STRIG GOSUB 400,500
140 PRINT "Demonstratie hoe de"
150 PRINT "subroutines behorende"
160 PRINT "bij de aktieknoppen"
170 PRINT "worden aangeroepen."
175 PRINT
180 GOTO 180
400 PRINT "Subroutine behorende"
410 PRINT "bij spatiebalk is aan-"
420 PRINT "geroepen."
430 PRINT
440 RETURN
500 PRINT "Subroutine behorende bij"
510 PRINT "aktieknop 1 van joy-stick"
520 PRINT "1 is aangeroepen."
530 PRINT
540 RETURN
```



Afb. 5-3 Grafische weergave van het ON STRIG GOSUB statement.

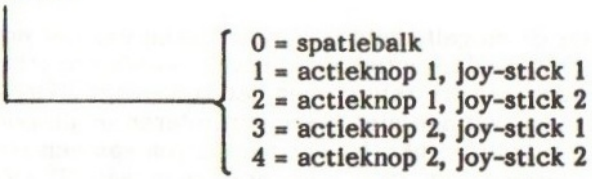
Het aantal regelnummers achter GOSUB mag maximaal 5

bedragen. De regelnummers komen als volgt overeen:

R1 spatiebalk
R2 actieknop 1, joy-stick 1
R3 actieknop 1, joy-stick 2
R4 actieknop 2, joy-stick 1
R5 actieknop 2, joy-stick 2

Zie voor verdere informatie over het gebruik van de regelnummers paragraaf 10.3 van deel 1 van deze serie leerboeken. In afbeelding 5-3 correspondeert regelnummer 400 met de spatiebalk en regelnummer 500 met de actieknop 1 van joy-stick 1.
Voordat de actieknoppen kunnen worden gebruikt, moeten deze eerst nog worden geactiveerd. Zonder deze activering heeft het indrukken van de actieknoppen geen onderbreking van het programma tot gevolg. Het statement waarmee actieknoppen kunnen worden geactiveerd, heeft het volgende formaat:

STRIG(n) ON



Wanneer als gevolg van het indrukken van een actieknop de betreffende subroutine wordt aangeroepen, wordt door de computer automatisch een STRIG(n) STOP voor genoemde actieknop uitgevoerd (zie afbeelding 5-3). Dit betekent, dat het opnieuw indrukken van de actieknop, tijdens het uitvoeren van de subroutine, geen effect zal hebben, zolang er geen STRIG(n) ON wordt gegeven.

Aan het einde van de subroutine keert het programma door middel van het statement RETURN weer terug naar het punt waar het eerder, door het indrukken van de actieknop, werd onderbroken. Tijdens de uitvoering van het RETURN-state-

ment wordt automatisch STRIG(n) ON gegeven. Dit gebeurt echter niet, wanneer in de subroutine het statement STRIG(n) OFF of STRIG(n) STOP werd uitgevoerd.

Tijdens het uitvoeren van een ERROR-routine worden alle actieknoppen, inclusief de spatiebalk, gedeactiveerd. Activering vindt pas weer plaats bij het uitvoeren van het statement RESUME, tenzij in de ERROR-routine een actieknop werd gedeactiveerd met STRIG(n) OFF of STRIG(n) STOP. Zie voor ERROR-routines hoofdstuk 9 van deel 1 uit deze serie leerboeken.

5.4 Activering en de-activering van actieknoppen.

Na uitvoering van het **statement STRIG(n) ON** zal MSXBASIC na uitvoering van elk statement controleren of de gegeven actieknop is ingedrukt. Indien dit zo is, zal het programma worden onderbroken en zal de bij de actieknop behorende subroutine worden aangeroepen. Het eerste regelnummer van de subroutine is gegeven in het ON STRIG GOSUB statement.

Wanneer de mogelijkheid tot onderbreken van het programma voor een bepaalde actieknop dient te worden opgeheven, kan dit worden gewerkstelligd met het **statement STRIG(n) OFF.** MSX-BASIC zal dan niet meer controleren of de betreffende actieknop is ingedrukt. Na het beeindigen van een programma wordt automatisch voor elke actieknop een STRIG(n) OFF gegeven.

Wanneer het in een bepaald gedeelte van het programma niet wenselijk is dat dit wordt onderbroken, maar dat het indrukken van een actieknop wel moet worden onthouden, dan kan dit worden geregeld met het **statement STRIG(n) STOP.** Wordt de betreffende actieknop ingedrukt, dan gebeurt er ogenschijnlijk niets. Zodra echter de betreffende actieknop weer wordt geactiveerd (met STRIG(n) ON), wordt het programma alsnog onderbroken en wordt de bij de actieknop behorende subroutine uitgevoerd. Een STRIG(n) STOP heeft geen zin, indien deze niet werd voorafgegaan door een

STRIG(n) ON.

De parameter **n** in de statements bepaalt voor welke actieknop het statement geldt:

n = 0 spatiebalk
n = 1 actieknop 1, joy-stick 1
n = 2 actieknop 1, joy-stick 2
n = 3 actieknop 2, joy-stick 1
n = 4 actieknop 2, joy-stick 2

We zullen deze paragraaf afsluiten met een educatief programma, waaraan twee personen kunnen deelnemen, die hoofdsteden van Europesche landen moeten raden. Hierbij werkt de ene deelnemer met de spatiebalk als actieknop en de andere met actieknop 1 van joy-stick 1.

```
100 CLS
105 WIDTH 40
110 ON STRIG GOSUB 500,600
115 PRINT TAB(10);"HOOFDSTEDEN RADEN"
120 PRINT:PRINT
125 PRINT TAB(4);"Deelnemer A gebruikt spatie
balk"
130 PRINT TAB(4);"als aktieknop."
135 PRINT
140 PRINT TAB(4);"Deelnemer B gebruikt aktiek
nop"
145 PRINT TAB(4);"van joy-stick 1."
150 PRINT:PRINT
155 PRINT TAB(4);:INPUT "Naam deelnemer A";NA$
160 PRINT TAB(4);:INPUT "Naam deelnemer B";NB$
165 PRINT:PRINT
170 PRINT TAB(4);"Veel geluk gewenst!"
175 PRINT TAB(4);"Voor het aanvangen van het"
180 PRINT TAB(4);"spel dient de RETURN-toets"
185 PRINT TAB(4);"te worden ingedrukt."
190 I$=INKEY$:IF I$<>CHR$(13) GOTO 190
195 TA=0:TB=0
200 CLS
```

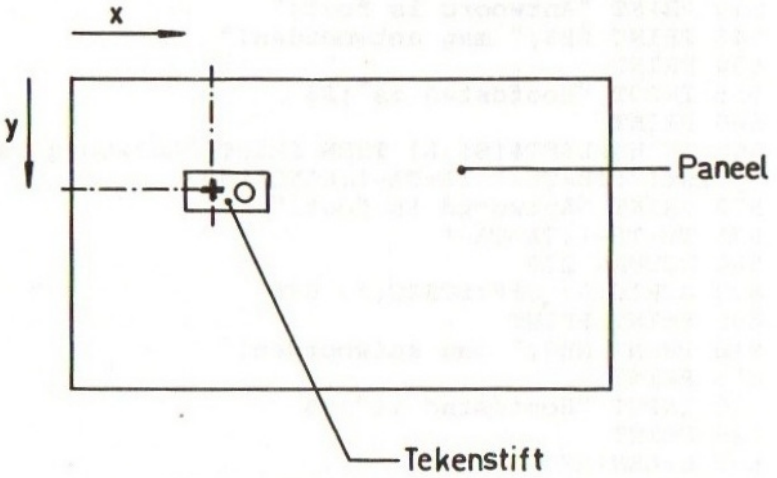
```
205 FOR I=1 TO 26
210 READ L$,H$
215 PRINT "Hoofdstad van ";L$;" is?"
220 STRIG(0) ON:STRIG(1) ON
225 GOTO 225
230 PRINT:PRINT
235 PRINT NA$;" heeft";TA;"punten."
240 PRINT NB$;" heeft";TB;"punten."
245 FOR J=1 TO 3000:NEXT J
250 CLS
255 NEXT I
260 PRINT "UITSLAG HOOFDSTEDEN RADEN"
265 PRINT
270 IF TA-TB<0 THEN PRINT "Hoera ";NB$;" heef
t gewonnen!":GOTO 290
275 IF TA-TB>0 THEN PRINT "Hoera ";NA$;" heef
t gewonnen!":GOTO 290
280 PRINT NA$;" en ";NB$;" speelden gelijk."
285 PRINT:PRINT:PRINT
290 INPUT "Volgend spel (J/N)";J$
295 IF J$="J" OR J$="j" GOTO 100
300 DATA Spanje,Madrid
305 DATA Zweden,Stockholm
310 DATA Sowjet Unie,Moskou
315 DATA Griekenland,Athene
320 DATA Groot Brittanie,Londen
325 DATA West-Duitsland,Bonn
330 DATA Roemenie,Boekarest
335 DATA Italie,Rome
340 DATA Finland,Helsinki
345 DATA Tsjecho-Slowakye,Praag
350 DATA Nederland,Amsterdam
355 DATA Zwitserland,Bern
360 DATA Joegoslavie,Belgrado
365 DATA IJsland,Reykjavik
370 DATA Oostenrijk,Wenen
375 DATA Belgie,Brussel
380 DATA Hongarije,Boedapest
385 DATA Noorwegen,Oslo
390 DATA Frankrijk,Parijs
```

```
395 DATA Bulgarije,Sofia
400 DATA Oost-Duitsland,Berlijn
405 DATA Ierland,Dublin
410 DATA Albanie,Tirana
415 DATA Polen,Warschau
420 DATA Denemarken,Kopenhagen
425 DATA Portugal,Lissabon
430 END
500 STRIG(0) OFF:STRIG(1) OFF
505 PRINT:PRINT
510 PRINT NA$;" mag antwoorden!"
515 PRINT
520 INPUT "Hoofdstad is";S$
525 PRINT
530 L=LEN(H$)
535 IF H$=LEFT$(S$,L) THEN PRINT "Antwoord is
 juist!":TA=TA+1:GOTO 580
540 PRINT "Antwoord is fout!"
545 PRINT NB$;" mag antwoorden!"
550 PRINT
555 INPUT "Hoofdstad is";S$
560 PRINT
565 IF H$=LEFT$(S$,L) THEN PRINT "Antwoord is
 juist!":TB=TB+1:TA=TA-1:GOTO 580
570 PRINT "Antwoord is fout!"
575 TB=TB-1:TA=TA-1
580 RETURN 230
600 STRIG(0) OFF:STRIG(1) OFF
605 PRINT:PRINT
610 PRINT NB$;" mag antwoorden!"
615 PRINT
620 INPUT "Hoofdstad is";S$
625 PRINT
630 L=LEN(H$)
635 IF H$=LEFT$(S$,L) THEN PRINT "Antwoord is
 juist!":TB=TB+1:GOTO 680
640 PRINT "Antwoord is fout!"
645 PRINT NA$;" mag antwoorden!"
650 PRINT
655 INPUT "Hoofdstad is";S$
```

```
660 PRINT
665 IF H$=LEFT$(S$,L) THEN PRINT "Antwoord is
 juist!":TA=TA+1:TB=TB-1:GOTO 680
670 PRINT "Antwoord is fout!"
675 TA=TA-1:TB=TB-1
680 RETURN 230
```

5.5 Digitizer (PAD).

Wanneer we in het Engelse woordenboek het woord PAD opzoeken, is een van de vertalingen "blocnote". In werkelijkheid is het "blocnote" een soort tekenpaneel, waarmee tekeningen kunnen worden gedigitaliseerd (in codes omgezet). Het paneel, dat op een van de periferiepoorten kan worden aangesloten, wordt meestal digitizer genoemd.



Afb. 5-4 Principe van een digitizer.

Het apparaat bestaat uit een paneel en een speciale tekenstift, welke is voorzien van een drukschakelaar. Door met de tekenstift ophet paneel bepaalde posities aan te raken, en voor elke positie de drukschakelaar in te drukken, kan een

tekening worden overgetrokken en gecodeerd naar de computer worden gestuurd. In afbeelding 5-4 wordt een digitizer getoond.

Met de **functie PAD(n)** kunnen we de verschillende toestanden van het paneel opvragen. In de functie specificeert parameter **n** welke toestand (status) is gewenst. Wanneer n een waarde heeft van 0, 1, 2 of 3 betreft het de digitizer aangesloten op connector 1 en wanneer n een waarde heeft van 4, 5, 6 of 7, betreft het de digitizer aangesloten op connector 2.

De volgende toestanden kunnen worden opgevraagd:

n = 0 of 4
Geeft resultaat 0, wanneer de tekenstift niet op het paneel wordt gedrukt en -1, wanneer de tekenstift wel op het paneel wordt gedrukt.

n = 1 of 5
Geeft als resultaat de x-coordinaat (de horizontale positie) van de plaats waar de tekenstift op het paneel is gedrukt.

n = 2 of 6
Geeft als resultaat de y-coordinaat (de verticale positie) van de plaats waar de tekenstift op het paneel is gedrukt.

n = 3 of 7
Geeft resultaat 0, wanneer de drukschakelaar op de tekenstift niet is ingedrukt en -1 wanneer de drukschakelaar wel is ingedrukt.

De functie PAD(n) kan alleen worden toegepast in een programma, wanneer een digitizer is aangesloten op de MSX-computer (op periferiepoort 1 of 2).

Het volgende programma is een voorbeeld van hoe de functie PAD(n) kan worden gebruikt. Met de digitizer wordt aangegeven waar op het beeldscherm een rechthoek van 30 bij 40 beeldpuntjes moet worden getekend. De opgevraagde coordinaten (X en Y) geven de linker bovenhoek van de rechthoek

aan.

```
10 CLS
20 SCREEN 2
30 IF PAD(0)=0 GOTO 30
40 X=PAD(1):Y=PAD(2)
50 DRAW "BM=X;,=Y;"
60 DRAW "R30D40L30U40"
70 GOTO 70
```

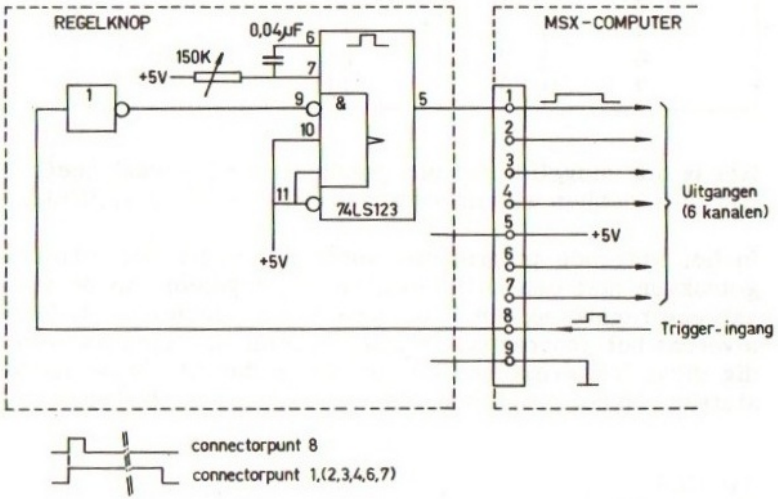
5.6 Werken met de regelknop (PADDLE).

Met de **functie PDL(n)** kan de waarde van een regelknop (paddle) worden opgevraagd. De functie kan alleen in een programma worden toegepast, wanneer op een of op beide periferiepoorten een regelknop is aangesloten. Parameter **n** kan een waarde hebben van 1 tot en met 12. De oneven waarden 1, 3, 5, 7, 9 en 11 hebben betrekking op de regelknop die is aangesloten op connector 1 en de even waarden 2, 4, 6, 8, 10 en 12 hebben betrekking op de regelknop die is aangesloten op connector 2.

De functie geeft als resultaat een waarde, die minimaal gelijk kan zijn aan 0 en maximaal aan 255. De waarde is altijd een geheel getal. De waarde 0 komt overeen met een geheel dichtgedraaide regelknop en de waarde 255 met een geheel opengedraaide regelknop. Elke tussenliggende waarde komt overeen met een tussenliggende stand van de regelknop.

Afbeelding 5-5 toont hoe een regelknop op de computer wordt aangesloten. Op 1 connector kunnen 6 kanalen (regelknoppen) worden aangesloten met een gemeenschappelijke triggeringang (connectorpunt 8). In de afbeelding is maar 1 kanaal getekend, dat bestaat uit een inverter en een monostabiele multivibrator (one-shot), waarop een regelbare weerstand R is aangesloten. Op het moment dat de functie PDL(n) wordt uitgevoerd, wordt een trigger-puls aangeboden op connectorpunt 8, die dan via de inverter de multivibrator triggert. Dit heeft tot gevolg, dat op de uitgang van de multivibrator een

puls met een bepaalde lengte komt te staan. De lengte van de puls is afhankelijk van de ingestelde waarde van de weerstand. De computer vertaalt op zijn beurt de lengte van de puls in een waarde, welke ligt tussen 0 en 255. De regelbare weerstand wordt bediend door de regelknop.



Afb. 5-5 Aansluitschema van de regelknop (PADDLE).

In plaats van 1 kanaal kunnen we ook meerdere kanalen toepassen. Het maximale aantal kanalen is 6. Elk kanaal heeft een eigen monostabiele multivibrator. De uitgangen van de multivibratoren zijn respectievelijk aangesloten op de connectorpunten 1, 2, 3, 4, 6 en 7. Er is echter maar 1 gemeenschappelijke trigger-ingang (connectorpunt 8), die via de inverter is aangesloten op de trigger-ingangen van de multivibratoren.
Met de functie PDL(n) kan per keer slechts de waarde van 1 kanaal worden opgevraagd. Welk kanaal zal worden geselecteerd, wordt bepaald door parameter n. Hiervoor geldt de volgende tabel:

connectorpunt	kanaal	waarde n connector 1	connector 2
1	1	1	2
2	2	3	4
3	3	5	6
4	4	7	8
6	5	9	10
7	6	11	12

Het is ook mogelijk dat een paddle maar 1 kanaal heeft. In dat geval hebben we alleen te maken met PDL(1) en PDL(2).

In het volgende programma wordt de lengte van een lijn, getrokken met een DRAW-statement, afgeleid van de stand van een regelknop, die is aangesloten op connector 1. Draai alvorens het programma te starten eerst de regelknop volledig dicht (resultaat van de functie is dan 0). Draai na het starten van het programma de regelknop geleidelijk open.

```
10 CLS
20 SCREEN 3
30 L=PDL(1)
40 DRAW "BM0,60"
50 DRAW "M=L;,60"
60 GOTO 30
```

6 Lijnen, cirkels, vlakken en kleuren

Het doel van dit hoofdstuk is het leren gebruiken van een aantal grafische statements, waarmee figuren, grafieken (wiskunde) en andere afbeeldingen op het beeldscherm kunnen worden getekend. Het ligt allerminst in de bedoeling om programma's te behandelen, waarmee ingewikkelde figuren kunnen worden getekend. De nadruk zal worden gelegd op de toepassing van de verschillende statements en op de mogelijkheden, die de statements te bieden hebben.

Alvorens te beginnen met de uitleg van de grafische statements, zal eerst worden stilgestaan bij de statements SCREEN en COLOR. Met SCREEN wordt de beeldschermmode bepaald. Met het statement COLOR worden de voorgrondkleur, de achtergrondkleur en de kleur van de randen op het beeldscherm bepaald.

Voor het tekenen van lijnen, driehoeken, rechthoeken, cirkels en andere figuren, dienen de grafische statements LINE en CIRCLE. Bovendien kan gebruik worden gemaakt van de grafische statements PSET en PRESET. Met deze statements kunnen figuren punt voor punt worden opgebouwd. Deze statements zijn daarom zeer geschikt voor het in beeld brengen van wiskundige functies (grafieken). Het is in MSX-BASIC ook mogelijk bepaalde vlakken (bijvoorbeeld een cirkel) in te kleuren. Hiervoor dient het grafisch statement PAINT. Als laatste zal in dit hoofdstuk de functie POINT worden behandeld, waarmee men de kleur van een gespecificeerd beeldpuntje kan opvragen.

De hiervoor genoemde statements en functie (LINE, CIRCLE, PSET, PRESET, PAINT en POINT) kunnen alleen in de grafische modes 1 en 2 (SCREEN 2 en 3) worden toegepast.

6.1 Beeldschermmodes en kleuren.

Daar de beeldschermmodes reeds in deel 1 zijn behandeld, zal alleen een overzicht van de vier modes worden gegeven. De vier beeldschermmodes kunnen worden ingedeeld in twee tekstmodes (SCREEN 0 en 1) en twee grafische modes (SCREEN 2 en 3). De modes staan in de volgende tabel vermeld.

statement	eigenschap beeldscherm	beeldscherm mode
SCREEN 0	24 regels van 40 tekens	tekstmode 1
SCREEN 1	24 regels van 32 tekens	tekstmode 2
SCREEN 2	256 pixels hor. 192 pixels ver.	grafische mode 1
SCREEN 3	64 blokjes hor. 48 blokjes ver.	grafische mode 2

Een pixel is een beeldpuntje. Dit is het kleinste lichtvlekje dat de computer op het beeldscherm kan maken. De kleinste eenheid in de grafische mode 2 (SCREEN 3) is een blokje, dat uit 4*4 beeldpuntjes bestaat. Het positioneren in de grafische mode 2 kan echter toch op 1 beeldpuntje nauwkeurig worden gedaan. De kleinste eenheid die in deze mode kan worden gebruikt voor het opbouwen van figuren en tekens (letters en cijfers) is echter 4*4 beeldpuntjes groot. Het oplossend vermogen is voor de grafische mode 2 dus laag te noemen ten opzichte van de grafische mode 1. De grafische mode 1 kan daarom worden gebruikt voor het maken van gedetailleerde tekeningen en de grafische mode 2 voor het maken van grove tekeningen. De volgende twee programma's zijn voorbeelden van beide grafische modes.

```
10 SCREEN 2
20 OPEN "GRP:" AS #1
30 DRAW "BM102,75R50D50L50U50"
40 DRAW "BM97,140"
50 PRINT #1,"SCREEN 2"
60 GOTO 60
```

```
10 SCREEN 3
20 OPEN "GRP:" AS #1
30 DRAW "BM96,30R50D50L50U50"
40 DRAW "BM0,100"
50 PRINT #1,"SCREEN 3"
60 GOTO 60
```

Wanneer in de grafische mode de uitvoering van een programma is beeindigd, wordt automatisch teruggekeerd naar de tekstmode.

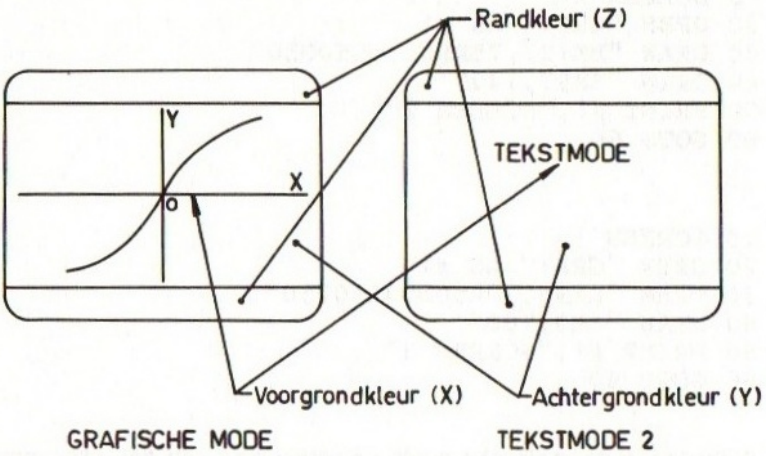
Wanneer de MSX-computer wordt ingeschakeld, wordt de beeldschermmode automatisch ingesteld op tekstmode 1 (SCREEN 0). Dit is dan ook de default-waarde.

Met het **statement COLOR** kan de kleurinstelling van het beeldscherm worden bepaald. Er kunnen met het statement drie kleuren worden gedefinieerd.

a) Voorgondkleur (kleur van tekst of tekening).
b) Achtergrondkleur (kleur van het scherm).
c) randkleur (de kleur van de boven- en onderrand bij een grafische instelling van het scherm).

Opmerking:
De randkleur geldt alleen voor de tekstmode 2 en de grafische modes 1 en 2.

Afbeelding 6-1 toont wat onder de verschillende instellingen wordt verstaan.



Afb. 6-1 Kleurinstellingen.

Het statement COLOR heeft het volgende formaat:

COLOR X,Y,Z

- Z — Randkleur
- Y — Achtergrondkleur
- X — Voorgrondkleur

Met de parameters X, Y en Z worden de kleurnummers gedefinieerd. In MSX-BASIC zijn 16 kleuren mogelijk. De volgende tabel bevat de kleurnummers met de bijbehorende kleuren.

0 = transparant	8 = rood
1 = zwart	9 = lichtrood
2 = groen	10 = donkergeel
3 = lichtgroen	11 = lichtgeel
4 = donkerblauw	12 = donkergroen
5 = lichtblauw	13 = paars
6 = donkerrood	14 = grijs
7 = cyaan	15 = wit

Het volgende programma illustreert de toepassing van het statement COLOR.

```
10 COLOR 1,2,10
20 SCREEN 2
30 OPEN "GRP:" AS #1
40 DRAW "BM10,70"
50 PRINT #1,"Voorgrondkleur   : zwart"
60 DRAW "BM10,90"
70 PRINT #1,"Achtergrondkleur: groen"
80 DRAW "BM10,110"
90 PRINT #1,"Randkleur        : donkergeel"
100 GOTO 100
```

Wanneer geen waarde is toegekend aan X, Y of Z, dan zal de laatste kleurinstelling voor de voorgrond, de achtergrond of de randen worden genomen. De default-waarden voor X, Y en Z zijn respectievelijk 15 (wit), 4 (donkerblauw) en 4 (donkerblauw). Het COLOR-statement moet in een programma voor het SCREEN-statement staan. Wanneer een COLOR-statement na het SCREEN-statement wordt uitgevoerd, blijft de oude achtergrondkleur actief en veranderen, indien van toepassing, alleen de voorgrondkleur en de randkleur.

Met het volgende programma worden alle mogelijke kleurcombinaties gedemonstreerd.

```
100 X=3:Y=2:Z=1
110 DIM A$(15)
120 RESTORE
130 FOR I=0 TO 15
140 READ A$(I):NEXT I
150 OPEN "GRP:" AS #1
160 COLOR X,Y,Z
170 SCREEN 2
180 DRAW "BM10,70"
190 PRINT #1,"Voorgrond    : ";A$(X)
200 DRAW "BM10,90"
```

```
210 PRINT #1,"Achtergrond : ";A$(Y)
220 DRAW "BM10,110"
230 PRINT #1,"Randen      : ";A$(Z)
240 Z=Y:Y=X:X=INT(RND(1)*15+1)
250 IF TIME>200 THEN TIME=0:GOTO 160
260 GOTO 250
270 DATA transparant,zwart,groen,lichtgroen,do
nkerblauw
280 DATA lichtblauw,donkerrood,cyaan,rood,lich
trood,donkergeel
290 DATA lichtgeel,donkergroen,paars,grijs,wit
300 END
```

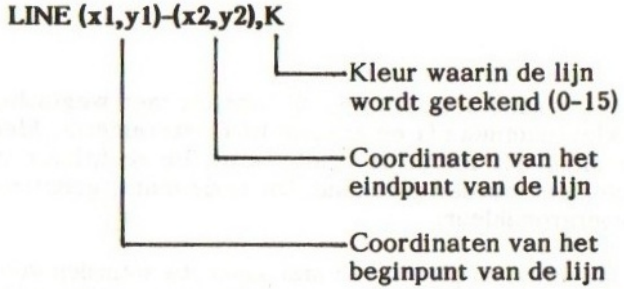
Wanneer als achtergrond- of randkleur transparant (kleurnummer 0) wordt gedefinieerd, zal dit resulteren in de kleur zwart. Wanneer als voorgrondkleur transparant wordt gedefinieerd, zal dit tot gevolg hebben, dat teksten en figuren niet zichtbaar zullen zijn op het beeldscherm.

6.2 Het tekenen van lijnen en figuren.

In hoofdstuk 4 hebben we reeds kennis gemaakt met de grafische macrotaal, waarmee door middel van het statement DRAW allerlei figuren op het beeldscherm kunnen worden getekend. Buiten deze macrotaal kent MSX-BASIC voor het tekenen nog een aantal andere statements. Een van deze statements is het statement LINE, waarmee vele soorten figuren kunnen worden opgebouwd, mits deze maar geen cirkels of afrondingen bevatten.

Met het statement LINE kan in de grafische mode 1 en 2 (SCREEN 2 en 3) op het beeldscherm een willekeurige rechte lijn of een rechthoek worden getekend. Door het achterelkaar plaatsen van verschillende LINE-statements in een programma kunnen allerlei andere figuren, zoals een huis, een gebouw, een poppetje, enz., op het beeldscherm worden getekend. De figuren moeten dan echter wel uit rechte lijnen zijn opgebouwd.

Het statement LINE kent verschillende formaten, die hierna in volgorde zullen worden behandeld. Het eenvoudigste formaat is:



Met bovenstaand formaat wordt op het beeldscherm een rechte lijn getrokken tussen twee punten. Het beginpunt van de lijn wordt aangegeven door de coordinaten x1 en y1 en het eindpunt door de coordinaten x2 en y2. De x-coordinaten kunnen varieren tussen 0 en 255, de y-coordinaten tussen 0 en 191. De kleur waarin de lijn zal worden getekend, wordt aangegeven door parameter K (kleur). K kan een waarde hebben, die ligt tussen 0 en 15. Wanneer K niet is gegeven, zal de laatst gespecificeerde voorgrondkleur worden gebruikt. De default-waarde van K is 15 (wit).

Met het volgende programma worden een driehoek en een rechthoek getekend op een lichtgroene achtergrond. De driehoek wordt getekend in de kleur zwart (K=1) en de rechthoek in de kleur donkerblauw (K=4). Met de LINE-statements op de regelnummers 120 t/m 140 wordt de driehoek getekend en met de statements op de regelnummers 150 t/m 180 de rechthoek.

```
100 COLOR 15,3,3
110 SCREEN 2
120 LINE (50,100)-(150,100),1
130 LINE (150,100)-(100,30),1
140 LINE (100,30)-(50,100),1
150 LINE (50,185)-(150,185),4
```

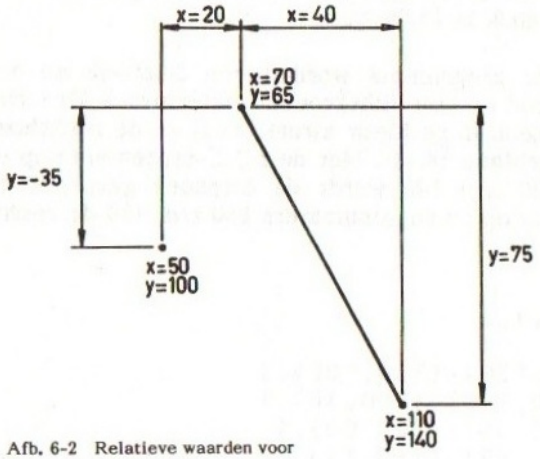
```
160 LINE (150,185)-(150,120),4
170 LINE (150,120)-(50,120),4
180 LINE (50,120)-(50,185),4
190 GOTO 190
```

Opdracht:

Voer bovenstaand programma ook eens uit met weglating van de kleurnummers (1 en 4) in de LINE-statements. Men zal dan zien, dat zowel de driehoek als de rechthoek in dezelfde kleur worden getekend. Dit is de laatst gedefinieerde voorgrondkleur.

Het is ook mogelijk, in plaats van met absolute waarden voor de coordinaten x en y, te werken met relatieve waarden. De parameters x en y geven dan de relatieve afstand aan van het beginpunt of het eindpunt van een lijn, ten opzichte van de huidige cursor-positie. Hierbij mogen x en y ook negatieve waarden hebben. Voorbeelden:

Huidige cursor-positie	x=50, y=100
Beginpunt van de lijn	x=70, y=65
Relatieve afstand	x=20, y=-35



Afb. 6-2 Relatieve waarden voor coordinaten.

Huidige cursor-positie	x=70, y=65	(beginpunt)
Eindpunt van de lijn	x=110, y=140	
Relatieve afstand	x=40, y=75	

De huidige cursor-positie kan, behalve een willekeurig punt, ook het beginpunt of het eindpunt van een lijn zijn.

Om aan te geven, dat x1 en y1 en/of x2 en y2 relatieve waarden vertegenwoordigen, moeten de x- en y-waarden worden voorafgegaan door het woord **STEP**. In de volgende formaten wordt dit getoond.

a. LINE STEP(x1,y1)-(x2,y2)
b. LINE (x1,y1)-STEP(x2,y2)
c. LINE STEP(x1,y1)-STEP(x2,y2)

In formaat a vertegenwoordigen x1 en y1 relatieve waarden en x2 en y2 absolute waarden. In formaat b is dit net andersom en in formaat c vertegenwoordigen zowel x1 en y1 als x2 en y2 relatieve waarden. We kunnen dus stellen, dat wanneer het woord STEP wordt gebruikt, de x- en y-waarden relatief moeten worden beschouwd ten opzichte van de huidige cursor-positie. De volgende twee programma's geven hetzelfde resultaat.

```
10 SCREEN 2
20 LINE (70,60)-(180,40),1
30 GOTO 30
```

```
10 SCREEN 2
20 LINE (70,60)-STEP(110,-20),1
30 GOTO 30
```

Wanneer de cursor-positie het beginpunt van een lijn is, kunnen de parameters x1 en y1 worden weggelaten. Het volgende programma toont dit aan.

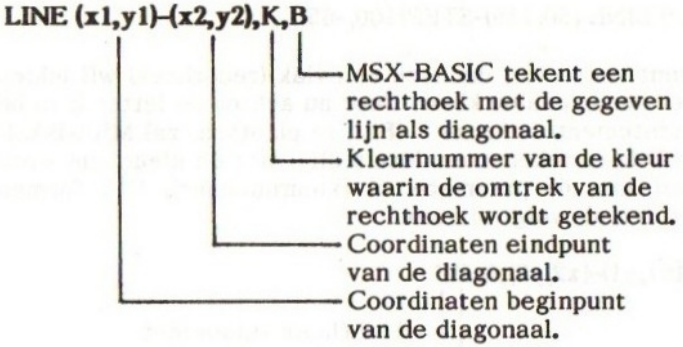
```
10 SCREEN 2
20 DRAW "BM20,40"
30 LINE -STEP(60,70),1
40 GOTO 40
```

Met het bovenstaande programma wordt op het beeldscherm een lijn getekend, met als beginpunt x=20 en y=40 en als eindpunt x=20+60=80 en y=40+70=110. Het programma met de driehoek en de rechthoek zouden we nu ook als volgt kunnen veranderen.

```
100 COLOR 15,3,3
110 SCREEN 2
120 LINE (50,100)-STEP(100,0),1
130 LINE -(100,30),1
140 LINE -STEP(-50,70),1
150 LINE (50,185)-STEP(100,0),4
160 LINE -(150,120),4
170 LINE -STEP(-100,0),4
180 LINE -STEP(0,65),4
190 GOTO 190
```

Analyseer voor uzelf het programma. Dit is een heel goede oefening.

Tot nu toe zijn met het LINE-statement alleen lijnen getekend. Het is echter ook mogelijk met een LINE-statement een complete rechthoek te tekenen. Dit wordt bereikt door achter het kleurnummer (K) de letter B (Blok) te plaatsen. Er zal dan een rechthoek worden getekend, met de gegeven lijn als diagonaal. De diagonaal zal niet op het beeldscherm worden getekend. Het formaat van het LINE-statement gaat er dan als volgt uitzien.



Met het volgende programma wordt een rechthoek getekend. De coordinaten x1=60, y1=10 en x2=200, y2=80 in regelnummer 20 geven respectievelijk het begin- en eindpunt van de diagonaal van de te tekenen rechthoek.

```
10 SCREEN 2
20 LINE (60,10)-(200,80),1,B
30 GOTO 30
```

Wanneer geen kleurnummer (voorgrondkleur) wordt gespecificeerd in het LINE-statement zal de laatst gedefinieerde voorgrondkleur worden gebruikt. Regelnummer 20 gaat er dan als volgt uitzien:

```
20 LINE (60,10)-(200,80),,B
```

Ook bij toepassing van de letter B voor het tekenen van rechthoeken, mag voor het specificeren van de diagonaal, gebruik worden gemaakt van relatieve waarden. In dit geval zou regelnummer 20 er als volgt kunnen gaan uitzien:

```
20 LINE (60,10)-STEP(140,70),,B
```

We zouden het programma van de driehoek en de rechthoek nogmaals kunnen aanpassen, door de regelnummers 150 tot en met 180 te laten vervallen en het volgende LINE-statement aan het programma toe te voegen:

```
150 LINE (50,185)-STEP(100,-65),4,B
```

Het komt vaak voor, dat men een vlak (rechthoek) wil inkleuren met een bepaalde kleur. Door nu achter de letter B in het LINE-statement de letter F (Fill) te plaatsen, zal MSX-BASIC een rechthoek tekenen en deze vullen met de kleur, die wordt gegeven door de parameter K (kleurnummer). Het formaat gaat er dan als volgt uitzien:

LINE (x1,y1)-(x2,y2),K,BF

- BF — Rechthoek vullen met kleur, die is gegeven door K.
- B — Rechthoek tekenen.

Wanneer er geen kleurnummer is gegeven in het LINE-statement, wordt de rechthoek gevuld met de laatst gespecificeerde voorgrondkleur. Met het volgende programma wordt een rechthoek gevuld met de kleur donkergeel.

```
10 SCREEN 2
20 LINE (50,185)-(150,120),10,BF
30 GOTO 30
```

Verander na uitvoering van het programma regelnummer 20 als volgt, en ga voor uzelf na wat de verschillen zijn.

```
20 LINE (50,185)-STEP(100,-65),,BF
```

Ter afsluiting van deze paragraaf volgt nog een programma, waarmee de kop van een marsmannetje wordt getekend.

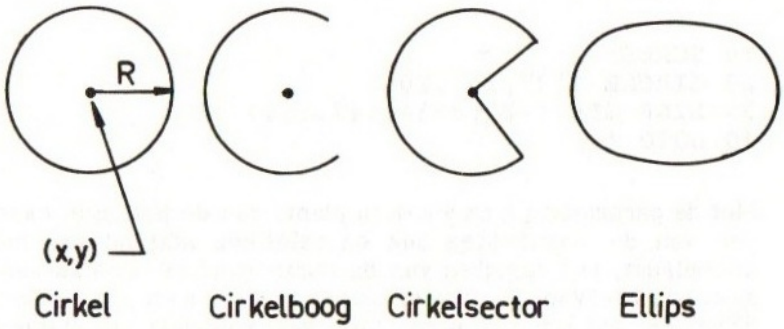
```
100 SCREEN 2
105 OPEN "GRP:" AS #1
110 LINE (70,5)-STEP(110,190),5,BF
115 LINE (80,45)-STEP(90,100),2,BF
120 LINE (100,65)-STEP(10,10),10,BF
125 LINE (140,65)-STEP(10,10),10,BF
```

```
130 LINE (120,95)-STEP(10,10),8,BF
135 LINE (95,125)-STEP(60,10),8,BF
140 LINE (105,45)-STEP(-25,-30),6
145 LINE (145,45)-STEP(25,-30),6
150 LINE (77,9)-(83,15),1,BF
155 LINE (167,9)-(173,15),1,BF
160 DRAW "BM80,170"
165 PRINT #1,"MARSMANNETJE"
170 GOTO 170
```

Om het statement LINE goed te leren kennen, is het zinvol om vanuit bovenstaand programma zelf de geprogrammeerde afbeelding te tekenen. Let hierbij ook op de verschillende kleuren. U zult opmerken, dat van de rechthoeken die over elkaar heen worden getekend, de kleur van de laatst getekende rechthoek bepalend is.

6.3 Het tekenen van cirkels en ellipsen.

In de vorige paragraaf hebben we gezien, dat het met het statement LINE niet mogelijk is om cirkels en afrondingen te tekenen. Voor het tekenen op het beeldscherm van cirkels, cirkelbogen, cirkelsectoren en ellipsen (afbeelding 6-3) beschikt MSX-BASIC over het statement CIRCLE.



Afb. 6-3 Mogelijkheden van het statement CIRCLE.

Het statement, dat alleen in de grafische modes 1 en 2 mag worden gebruikt, kent verschillende formaten. De formaten zullen hierna stap voor stap worden behandeld. Het basisformaat van het CIRCLE-statement is:

CIRCLE (x,y),R

Op het beeldscherm zal nooit een zuivere cirkel worden getekend, daar de beeldpuntjes op het scherm verticaal gezien dichter bij elkaar staan dan horizontaal. Straal R, die wordt uitgedrukt in beeldpuntjes, moet groter of gelijk zijn aan nul. Het laatst getekende punt van de cirkel is altijd het middelpunt (dit wil zeggen, dat na het tekenen van de cirkel de cursor blijft staan op het middelpunt van de cirkel).

Met het volgende programma wordt op het beeldscherm een cirkel getekend met een straal van 20 beeldpuntjes. Om te bewijzen, dat na uitvoering van het statement CIRCLE de cursor in het middelpunt van de cirkel staat, volgt op het statement CIRCLE een LINE-statement met relatieve waarden voor x en y.

```
10 SCREEN 2
20 CIRCLE (127,30),20
30 LINE STEP(-20,30)-(147,180),,B
40 GOTO 40
```

Met de parameters x en y kan in plaats van de absolute waarden van de coordinaten ook de relatieve afstand van het middelpunt, ten opzichte van de cursor-positie, worden aangegeven. Hiervoor dient voor de parameters x en y het woord **STEP** te worden geplaatst. Met het volgende programma verkrijgen we hetzelfde resultaat als met het vorige program-

ma. In dit geval kunnen x en y ook negatieve waarden hebben.

```
10 SCREEN 2
20 LINE (107,60)-(147,180),,B
30 CIRCLE STEP(-20,-150),20
40 GOTO 40
```

In het statement CIRCLE kan door middel van een kleurnummer ook worden aangegeven, in welke kleur de omtrek van de cirkel moet worden getekend. Wanneer geen kleurnummer in het statement is gespecificeerd, zal de laatst gegeven voorgrondkleur worden gebruikt. De default-waarde voor het kleurnummer is 15 (wit). Het formaat met kleurnummer ziet er als volgt uit:

CIRCLE (x,y),R,K

- K — Kleurnummer (0-15).
- R — Straal R, uitgedrukt in beeldpuntjes.
- (x,y) — Coordinaten middelpunt.

Als voorbeeld zal het vorige programma nu zodanig worden aangepast, dat de cirkel in zwart wordt getekend en de rechthoek in donkerrood.

```
10 SCREEN 2
20 LINE (107,60)-(147,180),6,B
30 CIRCLE STEP(-20,-150),20,1
40 GOTO 40
```

Opdracht:

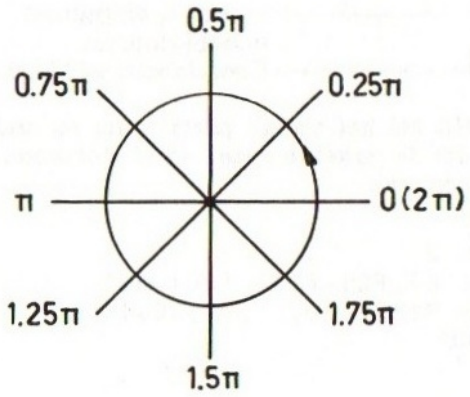
Verander in het voorgaande programma SCREEN 2 in SCREEN 3 en voer het programma nogmaals uit.

Het middelpunt van een cirkel mag ook buiten het beeldscherm worden geprojecteerd, dat wil zeggen x groter dan 255 en/of y groter dan 191. Dat betekent, dat slechts een

gedeelte van de cirkel op het scherm wordt getekend. Straal R mag hierbij niet groter zijn dan 32767. Het volgende programma geeft een voorbeeld van een buiten het beeldscherm geprojecteerd middelpunt.

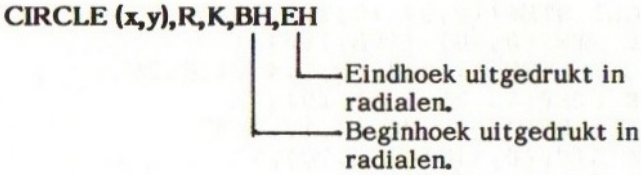
```
10 SCREEN 2
20 CIRCLE (290,30),80,1
30 GOTO 30
```

MSX-BASIC voorziet ook in het tekenen van cirkelbogen (gedeelten van cirkels). Hiervoor dient in het CIRCLE-statement de hoek vanwaar de cirkel moet starten en de hoek waar de cirkel moet eindigen te worden aangegeven. De hoeken worden hierbij uitgedrukt in radialen. De waarden van de hoeken liggen tussen 0 en 2pi radialen, waarbij pi = 3.1415926535898. Afbeelding 6-4 toont de ligging van de hoeken.



Afb. 6-4 Hoeken in het CIRCLE-statement.

Het formaat voor het tekenen van cirkelbogen ziet er als volgt uit:



Met het volgende programma wordt een cirkelboog getekend, die begint bij een hoek van 0.5 * 3.14 = 1.57 radialen en eindigt op een hoek van 1.5 * 3.14 = 4.71 radialen. De straal van de cirkel is 40 beeldpuntjes groot en de coordinaten van het middelpunt zijn x=120 en y=80. De kleur waarin de cirkelboog wordt getekend is zwart.

```
10 SCREEN 2
20 CIRCLE (120,80),40,1,1.57,4.71
30 GOTO 30
```

Voor de begin- en eindhoek mag elke willekeurige waarde tussen 0 en 2 pi worden genomen. De waarden in afbeelding 6-4 dienen alleen om aan te geven hoe de ligging van de hoeken is. Wanneer beginhoek BH niet is gegeven, zal 0pi als waarde worden genomen. Om dit te kunnen aantonen, zullen we in het voorgaande programma regelnummer 20 als volgt veranderen:

```
20 CIRCLE (120,80),40,1,,4.71
```

Opmerking:

Wanneer in het CIRCLE-statement geen kleurnummer of beginhoek wordt gegeven, moet men niet vergeten de komma's te plaatsen.

Met het volgende programma wordt op het beeldscherm een klein beeldscherm getekend.

```
100 SCREEN 2
105 OPEN "GRP:" AS #1
110 CIRCLE (50,20),10,1,1.57,3.14
115 LINE STEP(-10,0)-(40,152),1
```

```
120 CIRCLE STEP(10,0),10,1,3.14,4.71
125 LINE STEP(0,10)-(195,162),1
130 CIRCLE STEP(0,-10),10,1,4.71,6.28
135 LINE STEP(10,0)-(205,20),1
140 CIRCLE STEP(-10,0),10,1,,1.57
145 LINE STEP(0,-10)-(50,10),1
150 DRAW "BM80,85"
155 PRINT #1,"BEELDSCHERM"
160 GOTO 160
```

In het begin van deze paragraaf hebben we ook het tekenen van cirkelsectoren genoemd. Dit betekent, dat buiten het tekenen van een cirkelboog, ook nog het beginpunt en het eindpunt van de cirkelboog door rechte lijnen met het middelpunt moeten worden verbonden. Om dit te bereiken dient voor de waarden van de beginhoek (BH) en de eindhoek (EH een minteken (-) te worden geplaatst. Het volgende programma is hiervan een voorbeeld.

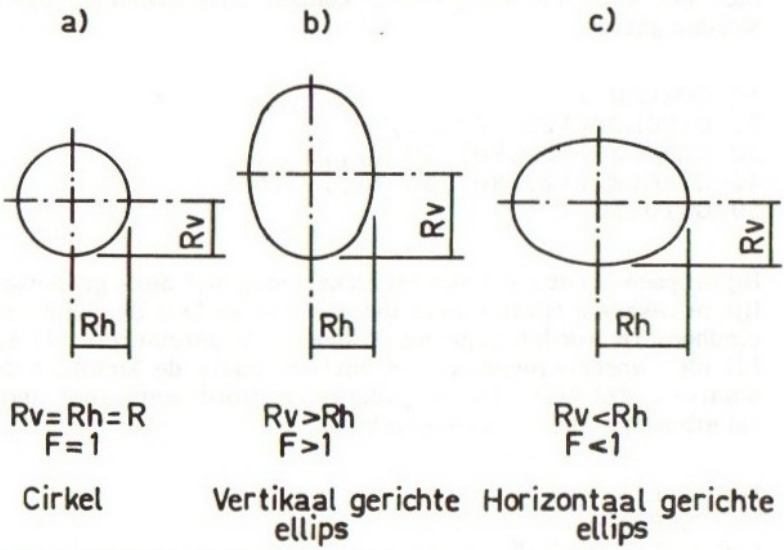
```
10 SCREEN 2
20 CIRCLE (120,90),40,1,-5.5,-3.9
30 GOTO 30
```

Het is ook mogelijk om maar een van beide punten met het middelpunt te verbinden. In dit geval wordt alleen de waarde, welke behoort bij het punt, dat met het middelpunt dient te worden verbonden, voorafgegaan door een minteken. In het volgende programma staat hiervan een voorbeeld.

```
100 SCREEN 2
110 CIRCLE (127,80),20,1,0,-2
120 LINE -STEP(-30,0),1
130 LINE -STEP(0,20),1
140 LINE -STEP(50,0),1
150 LINE -STEP(0,-20),1
160 LINE (97,80)-(87,70),1
170 CIRCLE (107,108),8,1
180 CIRCLE (137,108),8,1
190 GOTO 190
```

Als laatste zal in deze paragraaf het tekenen van ellipsen worden behandeld. Bij ellipsen kunnen we twee assen onderscheiden, namelijk de lange as en de korte as.

Voor het tekenen van een ellips in plaats van een cirkel zal in het statement CIRCLE een **afplattingsfactor** moeten worden gespecificeerd. De afplattingsfactor is altijd groter dan 0. Men kan bij het tekenen twee soorten ellipsen onderscheiden, namelijk horizontaal gerichte en verticaal gerichte ellipsen. De verhouding tussen Rv (verticale straal) en Rh (horizontale straal) wordt de afplattingsfactor F genoemd (F = Rv : Rh). Wanneer F groter is dan 1, dan wordt Rv gelijk aan R en Rh gelijk aan R / F. We verkrijgen dan een verticaal gerichte ellips (zie afbeelding 6-5b).



Afb. 6-5 Het effect van de afplattingsfactor

Wanneer F kleiner is dan 1, dan wordt Rh gelijk aan R en Rv gelijk aan R * F. We verkrijgen dan een horizontaal gerichte ellips (zie afbeelding 6-5c).

Daar voor een cirkel geldt dat Rv = Rh = R, is de afplattingsfactor voor een cirkel gelijk aan 1. Voor het tekenen van een cirkel behoeft in het CIRCLE-statement geen afplattingsfactor te worden gegeven. Het formaat voor het tekenen van een ellips is:

CIRCLE (x,y),R,K,BH,EH,F

└Afplattingsfactor.
Indien F groter dan 1
dan Rv=R en Rh=R/F.
Indien F kleiner dan 1
dan Rh=R en Rv=R*F.

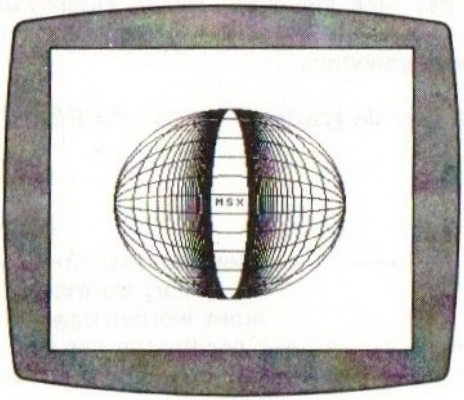
Met het volgende programma kunnen bovenstaande regels worden getest.

```
10 SCREEN 2
20 CIRCLE (120,90),50,1
30 CIRCLE (120,90),50,7,,,3
40 CIRCLE (120,90),50,13,,,.75
50 GOTO 50
```

Bij ellipsen is het net zoals bij cirkels mogelijk deze gedeeltelijk te tekenen. Hierbij dient net zoals bij cirkels de begin- en eindhoek te worden gegeven. Wanneer de parameters BH en EH niet worden toegepast, is het wel nodig de komma's te plaatsen. Met het volgende programma wordt een groot aantal ellipsen op het scherm getekend.

```
100 COLOR 1,2,1
110 SCREEN 2
120 OPEN "GRP:" AS #1
130 FOR F=1 TO 6 STEP .2
140 CIRCLE (120,95),90,15,,,F
150 NEXT F
160 FOR F=.1 TO 1 STEP .2
170 CIRCLE (120,95),90,13,,,F
180 NEXT F
```

```
190 DRAW "BM110,90"
200 PRINT #1,"MSX"
210 GOTO 210
```



6.4 Het kleuren van vlakken.

Het is natuurlijk leuk om bepaalde vlakken te kunnen opvullen met zelf gekozen kleuren. In MSX-BASIC is dit mogelijk door gebruik te maken van het statement PAINT. Het statement PAINT kan alleen worden gebruikt in grafische modes 1 en 2.

Inkleuren in grafische mode 1

Met het statement PAINT is het net zo gesteld als met de regels voor het kleuren in een kleurboek. Als men gaat kleuren, heeft men namelijk met de volgende regels te maken:

1. Het kiezen van het te kleuren vlak (x,y).
2. Het kiezen van de kleur (parameter Z).
3. Bepaal de grenzen (lijnen).

Met de parameters x en y worden de coordinaten aangegeven van een punt, dat binnen het in te kleuren vlak ligt.

In de grafische mode 1 moet de verfkleur Z, waarmee een vlak wordt ingekleurd, hetzelfde zijn als de kleur van de lijnen (randen) die het vlak insluiten. Wanneer de verfkleur en de kleur van de lijnen niet gelijk zijn, wordt met het inkleuren niet binnen het vlak gebleven. Indien kleurnummer Z wordt weggelaten, wordt als verfkleur de op dat moment actieve voorgrondkleur genomen.

Het formaat voor de grafische mode 1 (SCREEN 2) ziet er als volgt uit:

PAINT (x,y),Z

Kleurnummer (0-15) van de verfkleur, waarmee het vlak moet worden opgevuld.

Coordinaten van een punt binnen het in te kleuren vlak.

Met het volgende programma worden in de kleur zwart twee rechthoeken op het scherm getekend. De eerste rechthoek wordt door middel van het statement PAINT opgevuld met de kleur zwart en de tweede met de kleur groen. Men zal hierbij opmerken, dat meer dan alleen de tweede rechthoek zal worden ingekleurd met de kleur groen. Dit komt, omdat deze kleur niet gelijk is aan de kleur van de randen (zwart) van het in te kleuren vlak. De randen in dezelfde kleur als de verfkleur bepalen namelijk de grenzen, waarbinnen moet worden gebleven met het inkleuren.

```
10 SCREEN 2
20 LINE (10,10)-(50,50),1,B
30 LINE (80,10)-(120,50),1,B
40 PAINT (30,30),1
50 PAINT (100,30),10
60 GOTO 60
```

Opdracht:

Verander in regelnummer 50 de verfkleur 10 in 1. Draai het programma daarna nogmaals.

Wanneer het woord STEP wordt gebruikt in het statement PAINT, worden de waarden x en y relatief geinterpreteerd ten opzichte van de huidige cursor-positie. In dit geval kunnen x en y ook negatieve waarden hebben.

Met het volgende programma wordt een cirkel ingekleurd. Let ook hier weer op, dat het kleurnummer in het statement CIRCLE gelijk is aan het kleurnummer in het statement PAINT. Indien dit niet het geval zou zijn, wordt er meer gekleurd dan alleen de cirkel.

```
10 SCREEN 2
20 CIRCLE (120,90),50,10
30 PAINT (120,90),10
40 GOTO 40
```

Het statement PAINT zal de cirkel volledig vullen met donkergeel (kleurnummer 10), daar de cirkel ook is gekleurd in donkergeel. Verander na het uitvoeren van het programma het kleurnummer in regelnummer 30 eens in 6. U zult bemerken, dat bij uitvoering van het programma, het hele beeldscherm, behalve de randen, donkerrood zal worden ingekleurd. Dit komt omdat er geen grenzen zijn in de kleur donkerrood (kleurnummer 6).

Met het volgende programma worden twee paar cirkels getekend, die elkaar snijden. Het eerste paar cirkels is in zwart getekend. Van het tweede paar cirkels is de ene cirkel in zwart getekend en de andere in donkergeel. In het programma staat voor elk snijvlak een PAINT-statement. De twee PAINT-statements zijn volledig gelijk.

```
100 SCREEN 2
110 CIRCLE (50,60),40,1
120 CIRCLE (50,100),40,1
```

```
130 CIRCLE (190,60),40,10
140 CIRCLE (190,100),40,1
150 PAINT (50,80),1
160 PAINT (190,80),1
170 GOTO 170
```

Wanneer we het programma laten uitvoeren, zien we, dat bij het eerste paar cirkels het snijvlak zwart wordt ingekleurd en bij het tweede paar cirkels de zwart getekende cirkel.

Opdracht:

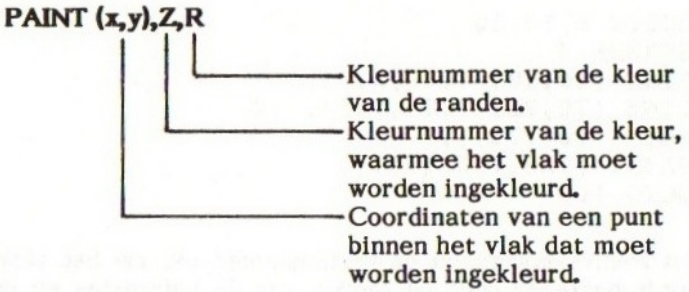
Verander het kleurnummer in regelnummer 130 in 1 en het kleurnummer in regelnummer 140 in 10. Start het programma dan nogmaals.

Toelichting:

Daar voor het tweede paar cirkels eerst de zwarte cirkel is getekend en daarna de donkergele, zitten er in de zwarte cirkel twee onderbrekingen (de laatste kleur is bepalend). Dit houdt in, dat het PAINT-statement op regelnummer 160 geen volledig ingesloten vlak, waarvan de randen zwart zijn, meer vindt. Dit heeft tot gevolg, dat de computer ook buiten de zwarte cirkel doorgaat met het inkleuren met zwart. Wees dus op uw hoede bij het inkleuren van vlakken.

Inkleuren in grafische mode 2

In de grafische mode 2 (SCREEN 3) mag de kleur waarmee een vlak wordt ingekleurd verschillend zijn van de kleur van de randen (lijnen), die het vlak insluiten. Het formaat van het statement voor de grafische mode 2 ziet er als volgt uit:



Wanneer de kleur van de randen van een vlak niet gelijk is aan de gespecificeerde randkleur (R) in het statement PAINT, zal ook buiten het vlak worden gekleurd, met de gespecificeerde verfkleur (Z).

Het volgende programma is een voorbeeld van het gebruik van het statement PAINT in de grafische mode 2.

```
100 SCREEN 3
110 LINE (50,50)-(100,100),1,B
120 LINE (150,50)-(200,100),10,B
130 PAINT (70,75),7,1
140 PAINT (170,75),6,10
150 GOTO 150
```

Opdracht:

Verander na uitvoering van het programma, in regelnummer 140 het kleurnummer 10 in kleurnummer 8. Voer het programma vervolgens nogmaals uit. Wat valt u daarbij op?

Wanneer de verfkleur (Z) in het statement PAINT wordt weggelaten, wordt als verfkleur de op dat moment actieve voorgrondkleur genomen. Wanneer de randkleur (R) wordt weggelaten, wordt als randkleur ook de op dat moment actieve voorgrondkleur genomen. Met het volgende programma wordt een deel van voorgaande regels aangetoond.

```
100 COLOR 6,10,10
110 SCREEN 3
120 LINE (50,50)-(200,200),1,B
130 LINE (75,75)-(175,175),1,B
140 PAINT (60,70),,1
150 PAINT (125,150),4,1
160 GOTO 160
```

Met het PAINT-statement op regelnummer 140 zal het vlak, dat wordt begrensd door de randen van de buitenste- en de binnenste rechthoek, worden ingekleurd met de voorgrondkleur 6 (donkerrood). Het binnenste vlak zal worden ingekleurd met verfkleur 4 (donkerblauw).

Zoals we ons nog kunnen herinneren, hebben we in hoofdstuk 4 met het statement DRAW een huis getekend. Deze paragraaf zullen we nu afsluiten met een programma, waarmee hetzelfde huis wordt getekend, echter nu met de statements LINE en PAINT.

```
200 COLOR 12,15,15
210 SCREEN 2
220 OPEN "GRP:" AS #1
230 LINE (50,180)-(200,80),1,B
240 LINE (50,80)-(125,5),6
250 LINE (125,5)-(200,80),6
255 LINE (49,80)-(201,80),6
260 LINE (60,165)-(110,115),14,B
270 LINE (120,165)-(140,115),14,B
280 LINE (160,180)-(190,115),4,B
290 LINE (160,39)-STEP(0,-60),6
300 LINE -STEP(20,0),6
310 LINE -STEP(0,80),6
320 PAINT (140,75),6
330 PAINT (70,150),14
340 PAINT (130,150),14
350 PAINT (170,150),4
360 DRAW "BM70,90"
370 PRINT #1,"MSX-HUIS"
```

```
375 LINE (0,180)-(255,180),1
380 LINE (115,30)-(135,50),14,BF
390 CIRCLE (20,20),10,10,,,1.25
400 PAINT (20,20),10
410 GOTO 410
```

6.5 Het punt voor punt tekenen van figuren.

Met de statements LINE en CIRCLE kunnen alleen rechte lijnen, cirkels, cirkelbogen, cirkelsectoren en ellipsen op het beeldscherm worden getekend. Wanneer we echter een willekeurige kromme of een grafische voorstelling van een wiskundige functie op het beeldscherm zouden willen tekenen, kan dit niet met de hiervoorgenoemde statements. We kunnen echter daarvoor wel gebruik maken van de statements PSET en PRESET. Deze statements werken alleen in de grafische modes 1 en 2.

Grafische mode 1 (SCREEN 2)

Het formaat van het statement PRESET ziet er als volgt uit:

PRESET (x,y),Z

- Z: Kleurnummer (0-15) van de kleur waarin het puntje wordt uitgevoerd.
- (x,y): Coordinaten van het puntje.

Met het statement PRESET kan een puntje op het beeldscherm worden geplaatst. Het puntje kan in een bepaalde kleur (Z) worden uitgevoerd. De positie, waar het puntje moet worden neergezet, wordt in het statement aangegeven met de coordinaten **x** en **y**. Wanneer het kleurnummer (Z) niet is gegeven, dan wordt de achtergrondkleur, die op dat moment actief is, als kleur genomen. De default-waarde voor Z is 4.

Met het volgende programma wordt door middel van het sta-

tement PRESET een donkergroen (kleurnummer 12) puntje in het middelpunt van een cirkel geplaatst.

```
10 COLOR 1,15,15
20 SCREEN 2
30 CIRCLE (127,91),20,1
40 PRESET (127,91),12
50 GOTO 50
```

Wanneer het kleurnummer 12 op regelnummer 40 wordt weggelaten, zal de op dit moment actieve achtergrondkleur (wit) worden gebruikt. Dit houdt in, dat het puntje niet zichtbaar is, daar de achtergrondkleur en de kleur van het puntje hetzelfde zijn.

In plaats van met absolute waarden te werken voor de coordinaten x en y, kan er ook met relatieve waarden worden gewerkt. In dat geval moeten de parameters x en y worden voorafgegaan door het woord STEP. Het volgende programma is hiervan een voorbeeld.

```
10 COLOR 1,15,15
20 SCREEN 2
30 LINE (105,75)-(145,115),4,B
40 PRESET (125,95),6
50 PRESET STEP(15,-15),6
60 PRESET STEP(-30,0),6
70 PRESET STEP(0,30),6
80 PRESET STEP(30,0),6
90 GOTO 90
```

In het volgende programma wordt het PRESET-statement zowel met als zonder kleurnummer toegepast. Het programma toont aan, dat het PRESET-statement zonder kleurnummer erg zinvol is. Hiermee kan men namelijk een beeldpuntje wissen, dat wil zeggen, de kleur van de achtergrond geven. Het programma laat een beeldpuntje aan en uit gaan (opflitsen).

```
10 COLOR 15,4,4
20 SCREEN 2
30 PRESET (127,91),1
40 FOR I=1 TO 50:NEXT I
50 PRESET (127,91)
60 FOR I=1 TO 50:NEXT I
70 GOTO 30
```

De werking van PSET is praktisch gelijk aan die van het statement PRESET, echter met dit verschil, dat bij het weglaten van het kleurnummer niet de achtergrondkleur, maar de voorgrondkleur wordt genomen. De default-waarde voor de voorgrondkleur is 15.

In het volgende programma wordt door middel van het statement PSET een vlak punt voor punt met de kleur donkergroen opgevuld.

```
10 COLOR 1,15,15
20 SCREEN 2
30 FOR I=100 TO 150
40 FOR J=70 TO 120
50 PSET (I,J),12
60 NEXT J
70 NEXT I
80 GOTO 80
```

Wanneer we nu kleurnummer 12 in regelnummer 50 weglaten, zal de op dit moment actieve voorgrondkleur worden gebruikt. Dit is de kleur zwart (kleurnummer 1).

Grafische mode 2 (SCREEN 3)

In de grafische mode 2 (SCREEN 3) wordt met de statements PRESET en PSET in plaats van een puntje een blokje van 2 * 2 puntjes op het beeldscherm gezet. De parameters x en y in het statement geven dan de positie aan van het puntje, dat zich links bovenaan het blokje bevindt. Met het volgende

programma wordt een kant van een dobbelsteen getekend.

```
10 COLOR 1,15,15
20 SCREEN 3
30 LINE (105,75)-(145,115),4,B
40 PSET (125,95),6
50 PSET STEP(13,-15),6
60 PSET STEP(-26,0),6
70 PSET STEP(0,26),6
80 PSET STEP(26,0),6
90 GOTO 90
```

De ogen van de dobbelsteen worden donkerrood gemaakt. Wanneer in de regelnummers 40 tot en met 80 de kleurnummers worden weggelaten, worden de ogen op de dobbelsteen zwart (actieve voorgrondkleur) gemaakt.

Deze paragraaf zal worden afgesloten met het tekenen van grafische voorstellingen in de grafische mode 1 (SCREEN 2). Onder een grafische voorstelling verstaan we een lijn, die het verband aangeeft tussen twee grootheden (bijvoorbeeld x en y), die onderling van elkaar afhankelijk zijn. Als voorbeeld zullen de grafische voorstellingen van de volgende functies worden getekend:

a) Y = SIN(X)
b) Y = COS(X)
c) Y = SIN(X) + COS(X)

Voordat de grafische voorstelling op het scherm kan worden getekend, zullen eerst de coordinatenassen moeten worden getekend.

```
120 COLOR 1,15,15
125 SCREEN 2
130 OPEN "GRP:" AS #1
135 LINE (0,90)-(255,90),1
140 LINE (40,0)-(40,192),1
145 DRAW "BM43,2":PRINT #1,"Y"
```

```
150 DRAW "BM43,92":PRINT #1,"0"
155 DRAW "BM240,80":PRINT #1,"X"
```

Wanneer we beginnen met de functie Y=SIN(X), zullen we eerst moeten bepalen voor welke waarden van X we Y willen berekenen. We kunnen bijvoorbeeld Y berekenen voor de waarden 0 tot en met 4pi radialen met een stapwaarde van 0.05. In het programma kunnen we dit met het volgende FOR-statement oplossen:

```
160 FOR X=0 TO 12.56 STEP .05
```

Voor het berekenen van de sinus van hoek X dient het volgende statement:

```
165 Y=SIN(X)
```

De grafische voorstelling van de functie zal met een PSET- of PRESET-statement in de FOR-NEXT-lus worden getekend. De oorsprong (0,0) van het assenstelsel bevindt zicht op het beeldscherm in de positie x=40 en y=90. Daar X een waarde kan hebben van 0 tot en met 12.56 en we op het beeldscherm horizontaal gezien over 256-40=216 beeldpuntjes beschikken, kunnen we de waarde van X met 15 vermenigvuldigen, zodat de totale breedte van het beeldscherm wordt gebruikt. De waarde X=0 komt dan op het beeldscherm overeen met x=40+(0*15)=40 en de waarde X=12.56 komt overeen met x=40+(12.56*15)=228.4=228.

Voor Y kunnen we dezelfde procedure aanhouden. De waarde van Y kan varieren van +1 tot -1. Om een duidelijk beeld te krijgen wordt Y vermenigvuldigd met een factor 30. Y=1 komt op het beeldscherm overeen met y=90-(1*30)=60, Y=-1 komt overeen met y=90-(-1*30)=120 en Y=0 komt overeen met y=90-(0*30)=90.

X en Y uit de functie Y=SIN(X) komen op het scherm overeen met x=40+X*15 en y=90-Y*30.

De coordinaten x en y kunnen nu in een PSET-statement

worden gebruikt voor het specificeren van het berekende punt. Het PSET-statement gaat er dan als volgt uit zien:

```
170 PSET (40+X*15,90-Y*30),1
```

We kunnen nu het programma als volgt afmaken:

```
175 NEXT X
180 GOTO 180
```

We zouden ook voor een bepaald interval kunnen kiezen, door de begin- en eindwaarde van X met INPUT-statements in te voeren. De waarde moet dan wel liggen tussen 0 en 12.56. Het programma moet dan als volgt worden uitgebreid:

```
100 COLOR 15,4,4
105 CLS
110 INPUT "Beginwaarde X";BX
115 INPUT "Eindwaarde  X";EX
```

Verder zal regelnummer 160 als volgt moeten worden aangepast:

```
160 FOR X=BX TO EX STEP .05
```

Voor het tekenen van de grafische voorstelling van de functie Y=COS(X) behoeft alleen regelnummer 165 te worden veranderd.

```
165 Y=COS(X)
```

Voor het tekenen van de grafische voorstelling van de functie Y=SIN(X)+COS(X) dient regelnummer 165 als volgt te worden veranderd:

```
165 Y=SIN(X)+COS(X)
```

Als laatste voorbeeld nemen we de functie Y=2X+4. De begin- en eindwaarde van X moet hierbij liggen tussen 0 en 40. Voor

deze functie zullen de regelnummers 165 en 170 als volgt moeten worden veranderd:

```
165 Y=2*X+4
170 PSET (40+X*5,90-Y),1
```

6.6 Het afvragen van de kleur van een beeldpuntje.

Met de functie POINT kan het kleurnummer van een gespecificeerd beeldpuntje worden afgevraagd. De functie kan alleen worden gebruikt in de grafische modes 1 en 2. Wanneer het gespecificeerde punt zich buiten het beeldscherm bevindt, geeft de functie als resultaat -1. Het formaat van de functie is:

POINT (x,y)

Coordinaten van het te onderzoeken beeldpuntje. (x=0-255, y=0-191)

Met het volgende programma worden drie beeldpuntjes op kleur onderzocht.

```
10 COLOR 1,3,3
20 SCREEN 2
30 FOR J=30 TO 50
40 FOR I=80 TO 160
50 PSET (I,J),9
60 NEXT I,J
70 FOR J=51 TO 70
80 FOR I=80 TO 160
90 PSET (I,J),5
100 NEXT I,J
110 X=POINT (100,40)
120 Y=POINT (100,60)
130 Z=POINT (100,200)
140 FOR I=1 TO 2000:NEXT I
```

```
150 COLOR 15,4,4
160 SCREEN 0
170 PRINT "Kleur 1 =";X
180 PRINT "Kleur 2 =";Y
190 PRINT "Kleur 3 =";Z
200 END
```

7 De macrotaal voor geluid

Zoals er voor het maken van grafische voorstellingen een macrotaal bestaat (het statement DRAW), zo bestaat er in MSX-BASIC ook een macrotaal voor het maken van muziek. Je zou kunnen zeggen, dat deze macrotaal de naam PLAY heeft. In de taal PLAY komen een groot aantal verschillende woorden voor. In werkelijkheid betreft het een statement (PLAY), waaraan een aantal subcommando's kunnen worden toegevoegd.

Naast het PLAY-statement zullen we in dit hoofdstuk ook nog de functie PLAY(X) behandelen. Deze functie maakt geen deel uit van de macrotaal PLAY, doch kan bij gebruik van het statement PLAY toch wel eens van pas komen. Met de functie PLAY(X) kan worden vastgesteld of de geluidsprocessor nog bezig is met het produceren van geluid. Afhankelijk van wat wij willen, kunnen we het programma met behulp van de functie PLAY(X) laten wachten tot de geluidsprocessor klaar is, of, ongeacht het resultaat van PLAY(X), doorgaan met het programma.

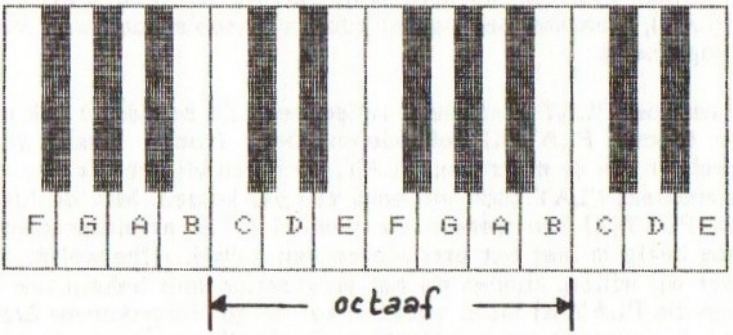
7.1 Vele tonen maken soms muziek

De macrotaal voor geluid kent maar een statement; het statement PLAY. Dit statement wordt gevolgd door 1, 2 of 3 strings. De geluidsprocessor in de MSX-computer heeft namelijk drie geluidskanalen. Voor ieder kanaal kan een string worden opgegeven. In die string staat welk geluid het betreffende kanaal geacht wordt te produceren. Zouden we bijvoorbeeld het volgende statement geven:

```
PLAY "CDE"
```

dan zal geluidskanaal A van de geluidsprocessor de noten C, D en E uit de toonladder van C spelen. Bent u niet thuis in muziek, dan zal de volgende verklaring u waarschijnlijk van nut zijn.

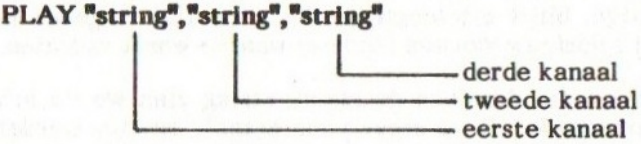
Op een piano zitten witte en zwarte toetsen. Die toetsen zijn opgedeeld in een aantal "octaven". Afbeelding 7-1 laat een stukje van het klavier van een piano zien. Daarbij is aangegeven wat een octaaf is. Een octaaf begint bij een witte toets die met de letter C wordt aangegeven, en eindigt bij de witte toets die met de letter B wordt aangegeven en die rechts van de C-toets ligt. Zoals uit de afbeelding blijkt, bestaat een octaaf dus uit de tonen C, D, E, F, G, A en B.



Afb. 7-1 Een octaaf op het pianoklavier.

Tussen de meeste witte toetsen zitten echter ook nog zwarte toetsen. Daar muziek niet mijn sterkste kant is, zal ik hier niet al te diep op in kunnen gaan. Voor de uitleg van de macrotaal is het echter voldoende om op te merken dat de zwarte toetsen, afhankelijk van de gebruikte toonladder, een verhoogde- of een verlaagde witte toets kunnen zijn. Zo zal, in de toonladder van C, de zwarte toets rechts van de witte C-toets, de Cis-toets zijn, een verhoogde C dus. Later in dit hoofdstuk zullen we zien hoe we dit in de macrotaal kunnen aangeven.

Met behulp van afbeelding 7-1 en het algem ormaat van het PLAY-statement moeten we in staat zijn om een aantal tonen ten gehore te laten brengen. Met een beetje goede wil is er zelfs een melodie te maken. Het algemene formaat van het PLAY-statement is als volgt:



Laten we ons voorlopig nog even beperken tot het genereren van geluid via 1 geluidskanaal. Om nu de eerste tonen van "Vader Jacob" te produceren, zullen we het volgende programma moeten schrijven:

```
10 PLAY "CDECCDEC"
20 END
```

U ziet, geen enkel probleem. De problemen gaan zich pas voordoen, wanneer we het liedje verder gaan uitwerken. Op een gegeven moment zullen we tonen tegenkomen die een andere lengte dienen te hebben. Bovendien zullen we op een gegeven ogenblik een toon ten gehore willen brengen, die niet binnen de octaaf valt. Het volgende programma laat het volledige lied "Vader Jacob" horen.

```
10 PLAY "cdeccdecef12gl4ef12gl8gagfl4ec
l8gagfl4ecco3go4l2cl4co3go4l2cl4"
20 END
```

In voorgaand voorbeeld is voor kleine letters gekozen, omdat dan het verschil tussen de letter o en het cijfer 0 goed opvalt.

In de bij het PLAY-statement behorende macro-string zien we, als we van links naar rechts lezen, op een gegeven moment de letter **l (subcommando)**, gevolgd door het cijfer 2. De letter l geeft aan dat de lengte van de noot gedefinieerd gaat worden. De waarde die achter de l wordt geplaatst, mag

varieren tussen 1 en 64. Standaard, na aanschakelen van de computer, is de waarde die bij l hoort 4. Dit wil zeggen dat de noot een vierde maat te horen zal zijn. De waarde 2 geeft aan dat de noot een halve maat te horen is, 1 is een hele maat. Met l geven we dus in feite de lengte van de noten ten opzichte van elkaar aan. Nadat de lengte door middel van l is gewijzigd, blijft die lengte geldig voor alle volgende noten, totdat l opnieuw van een (andere) waarde wordt voorzien.

In het laatste deel van de macro-string zien we de letter **o (subcommando).** De o staat voor octaaf, en kan worden gevolgd door een cijfer 1 tot en met 8. De geluidsprocessor kan namelijk noten uit 8 octaven ten gehore brengen. Een octaaf is een serie noten, die worden aangegeven met de letters C tot en met B, zoals is aangegeven in afbeelding 7-1. Na het aanschakelen van de computer is automatisch octaaf nummer 4 geselecteerd. Willen we op een gegeven moment een noot uit een lagere of hogere octaaf spelen, dan zullen we moeten aangeven uit welke octaaf die noten moeten worden gespeeld. Ook hier geldt weer, dat alle noten altijd uit de geselecteerde octaaf worden gespeeld, totdat er weer een andere octaaf wordt geselecteerd. Vandaar, dat in het voorbeeld na het spelen van de G uit octaaf 3, onmiddellijk wordt teruggegaan naar octaaf 4, om daaruit de noot C te spelen.

We hebben nu gezien, hoe we noten uit een octaaf kunnen selecteren, hoe we daarvan de lengte (ten opzichte van elkaar) kunnen bepalen, en hoe we verschillende octaven kunnen selecteren. Er is echter ook nog een mogelijkheid om het tempo, waarin wordt gespeeld, te beinvloeden. Hiertoe dienen we in de macro-string de letter **t (subcommando)**, gevolgd door een getal dat mag varieren van 32 tot 255, op te nemen. Na aanschakelen van de computer is het tempo vastgelegd op 120. Probeer nu eens zelf een ander tempo te kiezen, bijvoorbeeld t40, of t255. Speel het liedje eens in een aantal verschillende tempi en luister naar het resultaat.

Het liedje "Vader Jacob" is een canon. Onze geluidsprocessor heeft drie geluidskanalen. Laten we daarom eens proberen of we het lied driestemmig kunnen laten spelen. Het eerste

probleem dat we daarbij tegenkomen is, dat het PLAY-statement te lang wordt. Gelukkig is daar een oplossing voor.

In de strings die achter het PLAY-statement worden geplaatst, mogen ook variabelen worden opgenomen. De naam van die variabele zou echter wel eens kunnen overeenkomen met een macro-instructie. Stel bijvoorbeeld dat we variabele E willen opnemen in de macro-string. Hoe weet de macroprocessor dan dat we de variabele E bedoelen en niet de muzieknoot E? Dit zullen wij moeten aangeven. Hiertoe dienen we alle variabelen te laten voorafgaan door de letter **x (subcommando)**, die staat voor de macro-instructie "execute".

Zodra de macroprocessor de letter x tegenkomt, weet deze dat de daaropvolgende letter de eerste (en misschien wel de enige) letter van een variabelenaam is. Om aan de macroprocessor kenbaar te maken hoe lang de naam van de variabele is, zullen we dan ook de naam van de variabele moeten afsluiten met een **punt-komma (;)**. Pas wanneer we aan al deze eisen voldoen, zal de macroprocessor in staat zijn om variabelen, die in een string zijn opgenomen, uit te voeren. Als we dan bovendien zorgen dat de variabelen van te voren met de juiste waarden zijn gevuld, zal de geluidsprocessor de gewenste melodie ten gehore kunnen brengen.

Het volgende programma geeft een voorbeeld van voorgaande theorie. In dit programma zijn drie strings gedefinieerd, die elk het liedje "Vader Jacob" bevatten, doch met een verschuiving in de tonen. Regelnummer 40 bevat het PLAY-statement, waaraan drie strings zijn toegevoegd, een voor elk geluidskanaal. In die strings zijn nu echter de variabelenamen opgenomen, voorafgegaan door de letter x en afgesloten door de punt-komma. Tik het programma maar eens in en luister naar het resultaat.

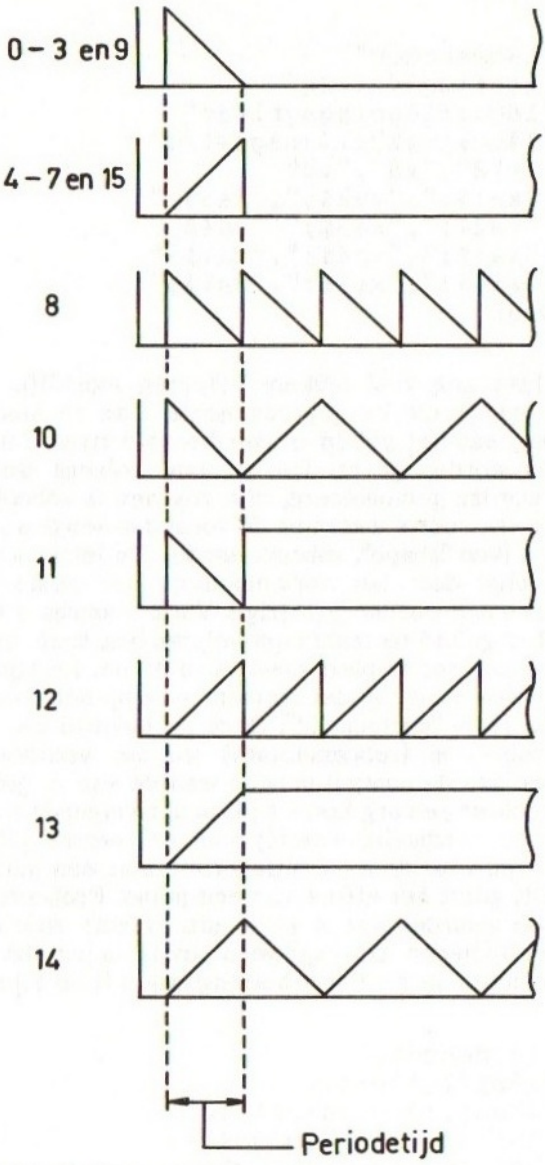
```
10 A1$="cdeccdecef12g14ef12g18gagf14ec18
gagf14ecco3go412c14co3go412c14"
20 A2$="ef12g14ef12g18gagf14ec18gagf14ec
co3go412c14co3go412c14cdeccdec"
30 A3$="18gagf14ec18gagf14ecco3go412c14c
```

```
o3go4l2cl4cdeccdecefl2gl4efl2gl4"
40 PLAY "xa1$;","xa2$;","xa3$;"
50 END
```

Hetzelfde lied kan ook nog op een andere manier worden geprogrammeerd. Dit wordt in het volgende programma getoond. De melodie is nu opgedeeld in 4 stukken van gelijke lengte (voor wat betreft de tijdsduur van de melodie). Hierdoor ontstaan kortere variabelen, hetgeen wat gemakkelijker in te tikken is. Vervolgens zorgen de vier PLAY-statements er voor dat het lied "Vader Jacob" in de juiste volgorde op de drie geluidskanalen wordt gespeeld.

```
10 A1$="l4cdeccdec"
20 A2$="l4efl2gl4efl2g"
30 A3$="l8gagfl4ecl8gagfl4ec"
40 A4$="l4co3go4l2cl4co3go4l2c"
50 PLAY "xa1$;","xa2$;","xa3$;"
60 PLAY "xa2$;","xa3$;","xa4$;"
70 PLAY "xa3$;","xa4$;","xa1$;"
80 PLAY "xa4$;","xa1$;","xa2$;"
90 GOTO 50
```

Om het lied nu wat minder eentonig te laten zijn, hebben we een aantal mogelijkheden. Laten we beginnen met de geluidssterkte. Er is een macro-instructie waarmee de geluidssterkte van ieder kanaal afzonderlijk kan worden geregeld. We kunnen de geluidssterkte regelen door in de macro-string de letter **v (subcommando)** op te nemen, en te laten volgen door een getal, dat mag varieren tussen 0 en 15. Het getal 0 wil zeggen dat we helemaal niets zullen horen, terwijl 15 het maximum volume geeft. Na aanschakelen van de computer staat de volumeregelaar automatisch in stand 8. In het volgende programma is een regel tussengevoegd waarmee de geluidssterkte van kanaal 1 op 12 wordt gezet, terwijl de geluidssterkte van de kanalen 2 en 3 op 8 wordt gezet. Hierdoor ontstaat het effect dat kanaal 1 de melodie gaat verzorgen, terwijl de kanalen 2 en 3 als begeleiding gaan functioneren. Probeer dit maar eens uit.



Afb. 7-2 De verschillende modulatievormen.

```
10 A1$="l4cdeccdec"
20 A2$="l4efl2gl4efl2g"
30 A3$="l8gagfl4ecl8gagfl4ec"
40 A4$="l4co3go4l2cl4co3go4l2c"
50 PLAY "v12","v8","v8"
60 PLAY "xa1$;","xa2$;","xa3$;"
70 PLAY "xa2$;","xa3$;","xa4$;"
80 PLAY "xa3$;","xa4$;","xa1$;"
90 PLAY "xa4$;","xa1$;","xa2$;"
100 GOTO 60
```

Er zijn echter nog veel leukere effecten mogelijk. Het is namelijk mogelijk om de geproduceerde toon te moduleren. Dit betekent, dat het geluid in een bepaald ritme harder en zachter kan worden gezet. De vormen, volgens welke het geluid kan worden gemoduleerd, zijn gegeven in afbeelding 7-2. De macro-instructie waarmee de vorm kan worden gekozen is de letter **s (van "shape", subcommando).** De letter s dient te worden gevolgd door het vormnummer. Het ritme van de modulatie kan ook worden gewijzigd. Vorm nummer 9 bijvoorbeeld laat het geluid op maximum volume beginnen, en geleidelijk afzwakken tot er niets meer is te horen. De tijd tussen het begin van de vorm, en het moment waarop het volume tot 0 is gedaald, is de "periodetijd". Deze periodetijd kan met de macro-instructie **m (subcommando)** worden veranderd. Na aanschakelen van de computer is de waarde van m gelijk aan 255. Dit is echter een erg korte tijd. In het volgende programma ziet u een voorbeeld, waarbij voor het eerste geluidskanaal is gekozen voor de modulatievorm 9 met een periodetijd van 4000. Dit geeft het effect van een piano. Probeert u eens verschillende waarden van s en m uit. U zult zien dat het aantal mogelijkheden buitengewoon groot is en dat de te bereiken effecten op z'n minst buitengewoon fraai zijn.

```
10 A1$="l4cdeccdec"
20 A2$="l4efl2gl4efl2g"
30 A3$="l8gagfl4ecl8gagfl4ec"
40 A4$="l4co3go4l2cl4co3go4l2c"
50 PLAY "s9m4000","v10","v10"
```

```
60 PLAY "xa1$;","xa2$;","xa3$;"
70 PLAY "xa2$;","xa3$;","xa4$;"
80 PLAY "xa3$;","xa4$;","xa1$;"
90 PLAY "xa4$;","xa1$;","xa2$;"
100 GOTO 60
```

We hebben nu de meeste macro-instructies gezien. Doch voordat we verder gaan met de nog overgebleven macro-instructies, volgen hier nog een paar programma's, waarmee u zelf naar hartelust kunt experimenteren. Het eerste programma laat zien hoe je een toon via het toetsenbord kunt starten. Daarmee wordt de computer dus tot orgel gepromoveerd. Het gebruikte principe is, dat er met behulp van de functie INKEY$ een letter van het toetsenbord wordt gelezen. Afhankelijk van de ingetoetste letter wordt dan de bijbehorende toon gegenereerd met behulp van het PLAY-statement. Dit programma bevat bovendien nog enige verfraaiing in de vorm van een grafische weergave van het piano-klavier. Bovendien wordt op het beeldscherm aangegeven welke toets van de piano u indrukt. Met behulp van dit programma is het erg gemakkelijk om een melodie in de vorm van de letters C tot en met B vast te leggen.

```
10 COLOR 1,15,3: SCREEN 2
20 OPEN "GRP:" AS #1
30 FOR I=8 TO 216 STEP 16
40 PSET (I,25):GOSUB 330
50 NEXT I
60 FOR I=20 TO 52 STEP 16
70 PSET (I,25):GOSUB 350
80 NEXT I
90 PSET (84,25):GOSUB 350
100 PSET (100,25):GOSUB 350
110 FOR I=132 TO 164 STEP 16
120 PSET (I,25):GOSUB 350
130 NEXT I
140 PSET (196,25):GOSUB 350
150 PSET (212,25):GOSUB 350
160 N$="FGABCDE"
170 FOR I=16 TO 112 STEP 16
```

```
180 PSET (I,105),15:PRINT #1,MID$(N$,I/1
6,1);
190 NEXT I
200 FOR I=128 TO 240 STEP 16
210 PSET (I,105),15:PRINT #1,MID$(N$,(I-
112)/16,1);
220 NEXT I
230 I$=INKEY$:IF I$=""GOTO 230
240 IF I$="C" OR I$="c" THEN PSET(72,135
):PLAY "c":GOSUB 370:GOSUB 390:PSET (72,
135):GOSUB 410:GOTO 230
250 IF I$="D" OR I$="d" THEN PSET(88,135
):PLAY "d":GOSUB 370:GOSUB 390:PSET (88,
135):GOSUB 410:GOTO 230
260 IF I$="E" OR I$="e" THEN PSET(104,13
5):PLAY "e":GOSUB 370:GOSUB 390:PSET (10
4,135):GOSUB 410:GOTO 230
270 IF I$="F" OR I$="f" THEN PSET(120,13
5):PLAY "f":GOSUB 370:GOSUB 390:PSET (12
0,135):GOSUB 410:GOTO 230
280 IF I$="G" OR I$="g" THEN PSET(136,13
5):PLAY "g":GOSUB 370:GOSUB 390:PSET (13
6,135):GOSUB 410:GOTO 230
290 IF I$="A" OR I$="a" THEN PSET(152,13
5):PLAY "a":GOSUB 370:GOSUB 390:PSET (15
2,135):GOSUB 410:GOTO 230
300 IF I$="B" OR I$="b" THEN PSET(168,13
5):PLAY "b":GOSUB 370:GOSUB 390:PSET (16
8,135):GOSUB 410:GOTO 230
310 GOTO 230
320 GOTO 320
330 LINE -STEP(16,100),1,B
340 RETURN
350 LINE -STEP(8,70),1,BF
360 RETURN
370 LINE -STEP(16,16),1,BF
380 RETURN
390 IF PLAY(0) GOTO 390
400 RETURN
410 LINE -STEP(16,16),15,BF
420 RETURN
```

Het volgende programma laat alle akkoorden van C tot en met B in majeur en in mineur horen. Al deze akkoorden zijn drieklanken. De keuze van de modulatievorm, periodetijd en volume voor de afzonderlijke kanalen kan tot bijzonder mooie klanken leiden. Experimenteer ook met dit programma ruimschoots, en noteer voor uzelf die klanken die u mooi vindt.

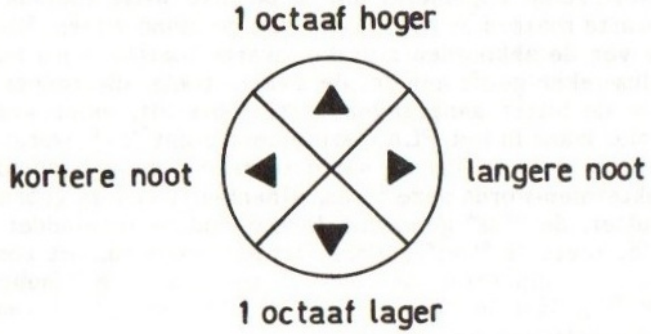
```
10 PLAY "v12","m6000s9","v12"
20 PLAY "l2","l2","l2"
100 PLAY "c","e","g":    'C majeur
105 PLAY "c","e-","g":   'C mineur
110 PLAY "d","f+","a":   'D majeur
115 PLAY "d","f","a":    'D mineur
120 PLAY "e","g+","b":   'E majeur
125 PLAY "e","g","b":    'E mineur
130 PLAY "f","a","c":    'F majeur
135 PLAY "f","a-","c":   'F mineur
140 PLAY "g","b","d":    'G majeur
145 PLAY "g","b-","d":   'G mineur
150 PLAY "a","c+","e":   'A majeur
155 PLAY "a","c","e":    'A mineur
160 PLAY "b","d+","f+": 'B majeur
165 PLAY "b","d","f+":   'B mineur
```

Bij het intikken van dit programma zal het u vreemd zijn voorgekomen dat er plotseling een aantal **plusjes (+) (subcommando)** moesten worden ingetypt. In het begin van dit hoofdstuk werd reeds opgemerkt dat er behalve witte toetsen ook nog zwarte toetsen op het klavier van de piano zitten. Bij het spelen van de akkoorden zijn die zwarte toetsen soms nodig. Het plus-teken geeft aan dat de zwarte toets, die rechts van de door de letter aangegeven witte toets zit, moet worden gespeeld. Waar in het PLAY-statement staat "f+", wordt dus de zwarte toets tussen de witte toetsen f en g bedoeld. In muziektermen wordt deze toets, afhankelijk van de gebruikte toonladder, de "Fis" genoemd. In een andere toonladder zou dezelfde toets de "Ges" kunnen worden genoemd. Dit zouden we dan in de macrotaal aan moeten geven met **"g-" (subcommando "-")**. Het betreft dan de zwarte toets links van de witte G-toets.

Het volgende programma stelt u in staat uzelf te begeleiden bij het zingen van een lied. De begeleiding bestaat uit akkoorden, die worden gegenereerd door het indrukken van een enkele toets. In dit programma zijn een aantal nog niet eerder genoemde mogelijkheden opgenomen. Op regelnummers 170 en 180 ziet u dat er in de string bij het PLAY-statement numerieke variabelen zijn opgenomen. De macro-instructie l moet worden gevolgd door een getal. In plaats van een getal mogen we ook een numerieke variabele plaatsen. Voorwaarde is dan wel, dat de letter l en de numerieke variabele van elkaar zijn gescheiden door een **"is gelijk teken" (=).** Net zoals we al eerder hebben gezien, dient de variabele te worden afgesloten door een **punt-komma.**

Op deze manier is het mogelijk om de tijdsduur van de tonen met behulp van de cursor control toetsen (CHR$(28) en CHR$(29) - links en rechts) te beinvloeden. Dezelfde methode is toegepast om een andere octaaf te kiezen. Door op de cursor control toetsen voor omhoog of omlaag te drukken wordt een hogere of lagere octaaf geselecteerd.

Een tweede bijzonderheid in dit programma is, dat er gebruik is gemaakt van een geindiceerde string-variabele (zie de regelnummers 190 tot en met 250). Of een variabele nu is geindiceerd of niet, doet niets af aan het feit dat het een variabele is, en dus kan deze gewoon in de bij het PLAY-statement behorende string worden geplaatst. Hiervoor gelden exact dezelfde regels als voor normale string-variabelen.



Afb. 7-3 Functies van de cursor control toetsen.

```
10 CLS
20 OC=4:LE=4
30 DIM N$(6,2)
40 FOR I=0 TO 6
50 FOR J=0 TO 2
60 READ N$(I,J)
70 NEXT J
80 NEXT I
90 DATA c,e,g:'    C majeur
100 DATA d,f+,a:'  D majeur
110 DATA e,g+,b:'  E majeur
120 DATA f,a,c:'   F majeur
130 DATA g,b,d:'   G majeur
140 DATA a,c+,e:'  A majeur
150 DATA b,d+,f+:' B majeur
160 PLAY "S9M4000","S9M4000","S9M4000"
170 I$=INKEY$:IF I$="" GOTO 170
180 PLAY "l=LE;","l=LE;","l=LE;"
190 PLAY "o=OC;","o=OC;","o=OC;"
200 IF I$="C" OR I$="c" THEN PLAY "xn$(0
,0);","xn$(0,1);","xn$(0,2);"
210 IF I$="D" OR I$="d" THEN PLAY "xn$(1
,0);","xn$(1,1);","xn$(1,2);"
220 IF I$="E" OR I$="e" THEN PLAY "xn$(2
,0);","xn$(2,1);","xn$(2,2);"
230 IF I$="F" OR I$="f" THEN PLAY "xn$(3
,0);","xn$(3,1);","xn$(3,2);"
240 IF I$="G" OR I$="g" THEN PLAY "xn$(4
,0);","xn$(4,1);","xn$(4,2);"
250 IF I$="A" OR I$="a" THEN PLAY "xn$(5
,0);","xn$(5,1);","xn$(5,2);"
260 IF I$="B" OR I$="b" THEN PLAY "xn$(6
,0);","xn$(6,1);","xn$(6,2);"
270 IF I$=CHR$(30) THEN OC=OC+1:IF OC>8
THEN OC=8
280 IF I$=CHR$(31) THEN OC=OC-1:IF OC<1
THEN OC=1
290 IF I$=CHR$(29) THEN LE=LE*2:IF LE>64
 THEN LE=64
300 IF I$=CHR$(28) THEN LE=LE/2:IF LE<1
THEN LE=1
```

```
310 LOCATE 5,5:PRINT "OCTAAF =";OC
320 LOCATE 5,7:PRINT "LENGTE =";LE;"/64
 * TEMPO "
330 LOCATE 5,9:PRINT "NOOT    = ";I$
340 GOTO 170
```

Er resten ons nu nog slechts enkele macro-instructies. Het zal u zijn opgevallen, dat alle tonen direct na elkaar worden gespeeld. Ze zitten als het ware aan elkaar vast. Om een scheiding tussen twee tonen mogelijk te maken is de macro-instructie **r (subcommando)** gecreeerd. Door de letter r te laten volgen door een getal tussen 1 en 64, wordt een rust tussen twee tonen gecreeerd. De lengte van die rust is een hele toonlengte wanneer r1 wordt geprogrammeerd, en is een vierenzestigste toonlengte wanneer r64 wordt geprogrammeerd. Hier gelden dus dezelfde regels als voor de toonlengte zelf (l).

Bij het schrijven van een muziekstuk voor meerdere kanalen, kan het gebeuren dat voor het ene kanaal veel meer macro-instructies worden gegeven dan voor het andere. Het uitvoeren van macro-instructies kost echter tijd. Hierdoor kunnen beide kanalen uit synchronisatie raken. Door op een verstandige manier gebruik te maken van rust-periodes tussen de tonen, kan die synchronisatie weer worden hersteld.

De lengte van tonen kan, behalve met behulp van de macro-instructie l, ook nog worden beinvloed door het plaatsen van een **punt (.)** achter de toon. In de muziekwereld is dit een gangbaar teken. Vandaar dat ook de macrotaal voor geluid daarin voorziet. Stel dat de lengte van een toon op 1 was gezet, en dat er daarna een toon C wordt geproduceerd. Door nu de letter C te laten volgen door een punt (.) zal de toon C met de helft worden verlengd.

De laatste macro-instructie, die nog moet worden behandeld is erg eenvoudig. Voor musici spreken de letters C tot en met B voor zichzelf. Voor niet-musici zeggen die letters echter niets. Bovendien zijn octaven maar onbegrijpelijke eenheden. Speciaal voor die mensen is er in de macrotaal voor geluid

een andere manier van aangeven van de toonhoogte opgenomen. In plaats van de letters en de octaven aan te geven, mag ook de letter **n (subcommando)** in de macro-string worden gezet, gevolgd door een getal. De toegestane getallen zijn 0 tot en met 95. Deze getallen komen overeen met de toetsen van het pianoklavier, van links naar rechts geteld. De centrale C (de C uit octaaf nummer 4) komt overeen met n36. Voor iedere volgende toets naar rechts (de zwarte toetsen ook meetellen), wordt de waarde van n met 1 verhoogd. Naar links wordt n voor iedere toets met 1 verlaagd. In plaats van een getal mag natuurlijk ook hier weer een numerieke variabele worden gebruikt.

7.2 Wordt er nog geluid gemaakt?

Met behulp van de functie PLAY(X) kunnen we te weten komen of de geluidsprocessor nog bezig is geluid te produceren. Nu is het niet zo belangrijk of wij dat zelf kunnen bepalen, want wij kunnen dat zo ook wel horen. Het kan echter wel belangrijk zijn dat het programma het kan bepalen. Het PLAY-statement is vaak al lang uitgevoerd, terwijl de geluidsprocessor nog een hele tijd bezig is om de opdrachten uit de macro-string uit te voeren. Ter illustratie het volgende programma.

```
10 CLS
20 PRINT "U hoort nu Vader Jacob"
30 A1$="l4cdeccdec"
40 A2$="l4efl2gl4efl2g"
50 A3$="l8gagfl4ecl8gagfl4ec"
60 A4$="l4co3go4l2cl4co3go4l2c"
70 PLAY "s9m4000","v10","v10"
80 PLAY "xa1$;","xa2$;","xa3$;"
90 PLAY "xa2$;","xa3$;","xa4$;"
100 PLAY "xa3$;","xa4$;","xa1$;"
110 PLAY "xa4$;","xa1$;","xa2$;"
120 PRINT "Nu is het afgelopen"
130 END
```

De tekst "Nu is het afgelopen" verschijnt al lang voordat de muziek werkelijk is afgelopen. Dit staat erg slordig. Er is echter een eenvoudige oplossing, waarbij de tekst pas verschijnt op het moment dat de muziek ook werkelijk is afgelopen. Dit wordt in het volgende programma getoond. Het enige verschil met het vorige programma is, dat regelnummer 120 is tussengevoegd.

```
10 CLS
20 PRINT "U hoort nu Vader Jacob"
30 A1$="l4cdeccdec"
40 A2$="l4efl2gl4efl2g"
50 A3$="l8gagfl4ecl8gagfl4ec"
60 A4$="l4co3go4l2cl4co3go4l2c"
70 PLAY "s9m4000","v10","v10"
80 PLAY "xa1$;","xa2$;","xa3$;"
90 PLAY "xa2$;","xa3$;","xa4$;"
100 PLAY "xa3$;","xa4$;","xa1$;"
110 PLAY "xa4$;","xa1$;","xa2$;"
120 IF PLAY(0) GOTO 120
130 PRINT "Nu is het afgelopen"
140 END
```

In voorgaand programma is op regel 120 de functie PLAY(0) gebruikt. Het algemene formaat van deze functie is:

Het opgegeven kanaalnummer kan 1, 2 of 3 zijn. In die gevallen zal de functie alleen naar het aangegeven kanaal kijken. Is als kanaalnummer het nummer 0 gegeven, dan zal de functie naar alle kanalen tegelijk kijken. Het resultaat van de functie is -1 indien het te onderzoeken kanaal nog geluid produceerd en 0 indien dit niet zo is.

In regel 120 van voorgaand programma zal het resultaat van de functie PLAY(0) alleen 0 zijn indien geen van de kanalen

nog geluid produceert. Een 0 komt overeen met "niet waar", zodat het IF-statement pas wordt verlaten wanneer alle kanalen uitgespeeld zijn. Zolang er nog een of meer kanalen geluid produceren, is het resultaat van de functie PLAY(0) -1. Dit komt overeen met "waar". Zolang de conditie "waar" geldt, zal worden gesprongen naar regelnummer 120 (GOTO 120).

Er zijn vele situaties denkbaar, waarin het wenselijk is pas verder te gaan met het uitvoeren van het programma nadat het geluid is afgelopen. Hopelijk kunt u in die gevallen nu zelf de functie PLAY(X) toepassen.

8 Sprites

Bij spelletjes is het aantrekkelijk, dat figuurtjes zich over het scherm kunnen bewegen. Denk hierbij maar aan pacman-achtige spelletjes. Bij MSX-BASIC is dit mogelijk gemaakt met behulp van "sprites". Sprites zijn figuren, die zich op de voorgrond van het scherm kunnen voortbewegen, zonder de figuren op de achtergrond, die zijn opgebouwd met de grafische statements DRAW, LINE, CIRCLE enz., aan te tasten. We kunnen bijvoorbeeld een huis op de achtergrond tekenen met het statement LINE en een auto, die als sprite is uitgevoerd, voor het huis langs laten rijden.

Voordat een figuurtje, uitgevoerd als sprite, op het scherm kan worden geplaatst en vervolgens kan worden voortbewogen, dient eerst te worden vastgelegd hoe het figuurtje er gaat uitzien. Het definieren of vastleggen van figuren (sprites) dient te gebeuren met de systeemvariabele SPRITE$(X). Als een sprite eenmaal is gedefinieerd, kan deze op elke willekeurige plaats op het scherm worden geplaatst. Hiervoor dient het statement PUT SPRITE. Met dit statement kunnen sprites ook worden voortbewogen en wel met elke gewenste snelheid.

Het is natuurlijk ook interessant, om wanneer twee figuurtjes (sprites) met elkaar in botsing of in aanraking komen, een bepaalde handeling uit te voeren. Denk hierbij bijvoorbeeld aan een pac-mannetje, dat figuurtjes opeet, of aan een vogel die wordt aangeschoten en hierna neervalt. In MSX-BASIC kan, wanneer twee sprites samenvallen of met andere woorden met elkaar in botsing komen, het programma tijdelijk worden onderbroken. Hierop kan een subroutine worden uitgevoerd, die betrekking heeft op de gebeurtenis. Hiervoor dient

het statement ON SPRITE GOSUB. In dit statement staat het eerste regelnummer van de subroutine vermeld.

Alvorens een subroutine kan worden aangeroepen, moet de mogelijkheid tot onderbreken van het programma, als gevolg van het samenvallen van twee sprites, eerst worden geactiveerd. Dit wordt gedaan met het statement SPRITE ON. De mogelijkheid tot onderbreken van het programma kan worden gede-activeerd door een van de statements SPRITE OFF of SPRITE STOP.

Wanneer in een programma sprites worden toegepast, zal in het SCREEN-statement de grootte van de sprites moeten worden aangegeven. Er zijn namelijk grote en kleine sprites. Daarom zal in dit hoofdstuk ook nog aandacht worden besteed aan het statement SCREEN.

Sprites kunnen alleen worden toegepast in de tekstmode 2 (SCREEN 1) en de grafische modes 1 en 2 (SCREEN 2 en 3).

8.1 Het statement SCREEN.

Buiten het specificeren van de grootte van de toe te passen sprites, zullen in deze paragraaf ook nog de andere mogelijke parameters in het SCREEN-statement worden behandeld.

In het SCREEN-statement kan, achter de mode (0, 1, 2 of 3), de grootte van de toe te passen sprites worden gegeven. De sprite-grootte wordt als volgt gedefinieerd:

0 - kleine sprites, 8*8 beeldpuntjes.
1 - kleine sprites, vergroot tot 16*16 beeldpuntjes.
2 - grote sprites, 16*16 beeldpuntjes.
3 - grote sprites, vergroot tot 32*32 beeldpuntjes.

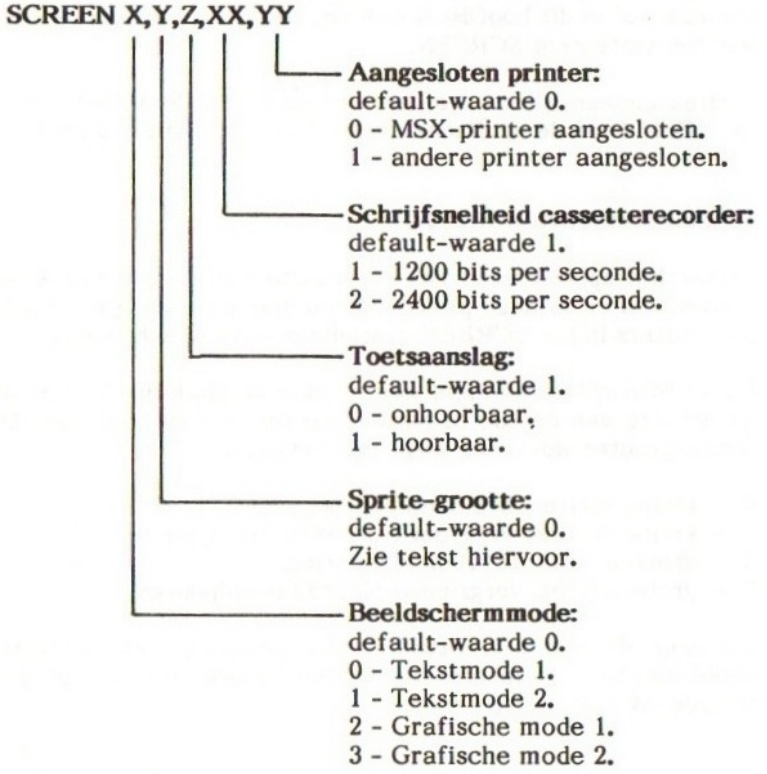
Wanneer de sprite-grootte niet is gegeven, zal de laatst gedefinieerde sprite-grootte worden gebruikt. De default-waarde voor de sprite-grootte is 0.

Wanneer in het SCREEN-statement de sprite-grootte wel is gegeven, zal tijdens het uitvoeren van het SCREEN-statement de inhouden van de variabelen SPRITE$(X) worden gecleared (schoongemaakt).

Voor het specificeren van grote sprites, bestaande uit 16*16 beeldpuntjes, in de grafische mode 1, gaat het SCREEN-statement er als volgt uitzien:

```
10 SCREEN 2,2
```

We zullen nu het volledige formaat van het SCREEN-statement tonen:



Opmerking:

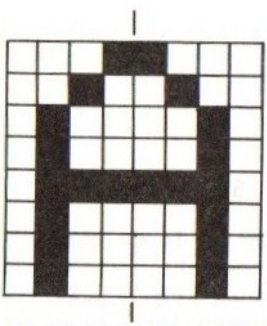
De gespecificeerde snelheid in het SCREEN-statement heeft alleen betrekking op schrijfoperaties. Wanneer van cassette wordt gelezen, wordt de snelheid automatisch door de computer bepaald, zodat het niet nodig is, dat de computergebruiker weet met welke schrijfsnelheid de informatie op cassette is geschreven.

De schrijfsnelheid kan ook worden gegeven in het statement CSAVE. Het formaat van het CSAVE-statement gaat er dan als volgt uitzien:

CSAVE "bestandsnaam",1 of
CSAVE "bestandsnaam",2

8.2 Het definieren van sprites.

Wanneer we een figuurtje willen laten uitvoeren als sprite, zullen we eerst moeten bepalen of het een kleine of grote sprite moet zijn. Een klein formaat sprite betekent, dat de sprite uit 8*8 beeldpuntjes bestaat. Dit wil zeggen, dat het figuurtje in principe moet worden vastgelegd in een raster van 8*8 beeldpuntjes. Wanneer kleine sprites worden toegepast, kunnen 256 sprites worden gedefinieerd. Een sprite wordt vastgelegd in de systeemvariabele **SPRITE$(X)**. Afbeelding 8-1 toont een sprite van 8*8 beeldpuntjes.



Hex. dec.	Dec.
18	24
24	36
42	66
42	66
7E	126
42	66
42	66
42	66

Afb. 8-1 Een sprite van 8 bij 8 beeldpuntjes.

Elke rij van acht beeldpuntjes vormt 1 byte (8 bits). De kleine sprite kan dus worden vastgelegd in 8 bytes. De inhoud van de bytes kan een binaire-, een hexadecimale- of decimale waarde zijn. Hoe de 8 bytes aan een alfanumerieke variabele kunnen worden toegekend, tonen de volgende voorbeelden. We hebben hiervoor de sprite uit afbeelding 8-1 genomen.

Hexadecimaal:

```
10 SCREEN 2,0
15 B$=""
20 FOR I=1 TO 8
25 READ A$
30 B$=B$+CHR$(VAL("&H"+A$))
35 NEXT I
60 DATA 18,24,42,42,7E,42,42,42
```

Binair:

```
10 SCREEN 2,0
15 B$=""
20 FOR I=1 TO 8
25 READ A$
30 B$=B$+CHR$(VAL("&B"+A$))
35 NEXT I
60 DATA 00011000,00100100,01000010,01000
010,01111110,01000010,01000010,01000010
```

Decimaal:

```
10 SCREEN 2,0
15 B$=""
20 FOR I=1 TO 8
25 READ A
30 B$=B$+CHR$(A)
35 NEXT I
60 DATA 24,36,66,66,126,66,66,66
```

Voor het definieren van de sprite (letter A) in de grafische mode 1 of 2, moet de inhoud van alfanumerieke variabele B$ worden toegekend aan systeemvariabele SPRITE$(X). De systeemvariabele SPRITE$ heeft het volgende formaat:

SPRITE$(sprite-nummer) = alfanumerieke variabele
of
SPRITE$(sprite-nummer) = string

Wanneer we met sprite-grootte 0 of 1 (zie paragraaf 8.1) werken, kunnen maximaal 256 onafhankelijke sprites worden vastgelegd. Elke sprite heeft hierbij zijn eigen nummer. Dit betekent dat het sprite-nummer een waarde kan hebben van 0 tot en met 255. Wanneer we met sprite-grootte 2 of 3 werken, kunnen maximaal 64 onafhankelijke sprites worden vastgelegd. Dit betekent, dat voor sprite-grootte 2 of 3 het sprite-nummer een waarde kan hebben van 0 tot en met 63.

De lengte van systeemvariabele SPRITE$(X) is vastgesteld op 32 bytes. Wanneer een string wordt toegekend, die korter is dan 32 bytes, wordt de inhoud van de variabele verder aangevuld met null-tekens (CHR$(0)).

Wanneer we de letter A uit afbeelding 8-1 willen vastleggen in de systeemvariabele SPRITE$(1), dan zullen we hiervoor in de voorgaande programma's de alfanumerieke variabele B$ moeten toekennen aan de systeemvariabele. De programma's kunnen dan als volgt worden uitgebreid:

```
40 SPRITE$(1)=B$
```

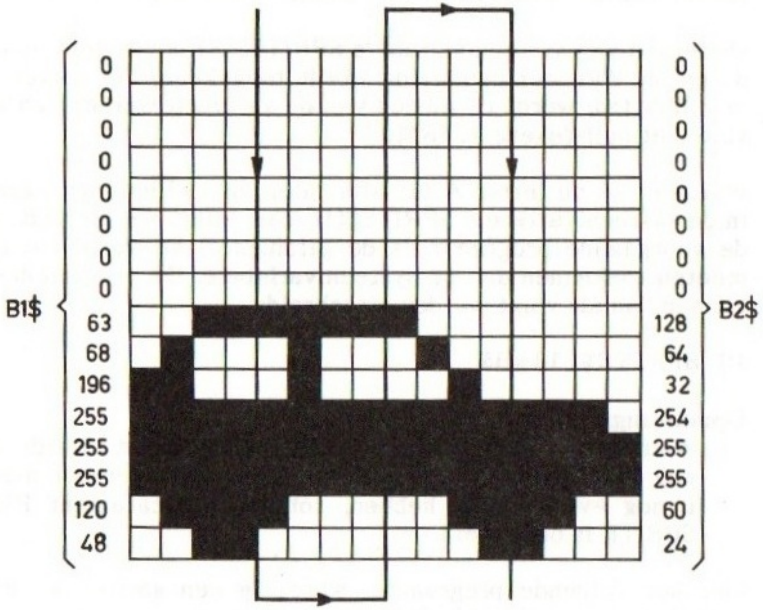
Opmerking:
Wanneer de programma's worden uitgevoerd, wordt er nog steeds niets op het scherm geplaatst. Hiervoor moet u nog even geduld hebben, totdat het statement PUT SPRITE is behandeld.

Met het volgende programma wordt in een sprite van 8*8 beeldpuntjes een spookje vastgelegd.

```
10 SCREEN 2,0
15 B$=""
20 FOR I=1 TO 8
25 READ A
30 B$=B$+CHR$(A)
```

```
35 NEXT I
40 SPRITE$(0)=B$
60 DATA 6,15,29,63,62,60,124,56
```

Wanneer een figuur niet in een sprite van 8*8 beeldpuntjes past, kan een sprite van 16*16 beeldpuntjes worden genomen. Hierbij vormen elke 16 horizontale beeldpuntjes 2 bytes. Dit houdt in, dat voor het vastleggen van de totale sprite 32 bytes nodig zijn.
Afbeelding 8-2 toont een sprite van 16*16 beeldpuntjes.



Afb. 8-2 Een sprite van 16 bij 16 beeldpuntjes.

Bij sprites van 16*16 beeldpuntjes moeten eerst de 16*8 beeldpuntjes links in de sprite worden vastgelegd en daarna de 16*8 beeldpuntjes rechts in de sprite. Hierbij wordt begonnen met de 8 horizontale beeldpuntjes links bovenaan in de sprite.

Wanneer de eerste 16 rijen van 8 beeldpuntjes zijn vastgelegd, worden de 16 rijen van 8 beeldpuntjes in het rechter gedeelte van de sprite vastgelegd. Ook voor het rechter gedeelte wordt weer bovenaan begonnen. De auto in afbeelding 8-2 wordt eerst in de alfanumerieke variabelen B1$ en B2$ vastgelegd en vervolgens worden de inhouden van deze variabelen toegekend aan de systeemvariabele SPRITE$(0). Het volgende programma laat dit zien:

```
10 SCREEN 2,3
15 B1$="":B2$=""
20 FOR I=1 TO 16
25 READ A
30 B1$=B1$+CHR$(A)
35 NEXT I
40 FOR I=1 TO 16
45 READ A
50 B2$=B2$+CHR$(A)
55 NEXT I
60 SPRITE$(0)=B1$+B2$
75 DATA 0,0,0,0,0,0,0,0,63,68,196,255,25
5,255,120,48
80 DATA 0,0,0,0,0,0,0,0,128,64,32,254,25
5,255,60,24
```

Bij het vastleggen van een sprite moet men er wel rekening houden, dat de maximaal toegestane lengte van een programmaregel 255 tekens is. Dat houdt in, dat soms niet alle bytes aan een alfanumerieke variabele kunnen worden toegekend. De regelnummers 15 tot en met 55 zouden als volgt kunnen worden vervangen:

```
15 B1$=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+C
HR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(63)+
CHR$(68)+CHR$(196)+CHR$(255)+CHR$(255)+C
HR$(255)+CHR$(120)+CHR$(48)
20 B2$=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+C
HR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(128)
+CHR$(64)+CHR$(32)+CHR$(254)+CHR$(255)+C
HR$(255)+CHR$(60)+CHR$(24)
```

Wanneer men de volledige sprite zou toekennen aan 1 alfanumerieke variabele, bijvoorbeeld aan B$, zou de lengte van de programmaregel meer dan 255 tekens tellen en dat is niet toegestaan. Vandaar dat de inhoud van de sprite wordt toegekend aan twee variabelen, namelijk B1$ en B2$. Deze variabelen worden op hun beurt weer toegekend aan systeemvariabele SPRITE$(0).

Voor de sprite met de letter A gaat het programma er dan als volgt uitzien:

```
10 SCREEN 2,0
20 B$=CHR$(24)+CHR$(36)+CHR$(66)+CHR$(66
)+CHR$(126)+CHR$(66)+CHR$(66)+CHR$(66)
40 SPRITE$(1)=B$
```

Voor sprites van 8*8 beeldpuntjes is het mogelijk de bytes direct toe te kennen aan de variabele SPRITE$(X). Het voorgaande programma gaat er dan als volgt uitzien:

```
10 SCREEN 2,0
20 SPRITE$(1)=CHR$(24)+CHR$(36)+CHR$(66)
+CHR$(66)+CHR$(126)+CHR$(66)+CHR$(66)+CH
R$(66)
```

Wanneer een figuur niet in 1 sprite past, kan de figuur worden vastgelegd in 2 sprites, met elk een eigen sprite-nummer. Het volgende voorbeeld is een figuur, dat bestaat uit twee sprites, van elk 16*16 beeldpuntjes. Het is een auto met een caravan.

```
100 SCREEN 2,2
105 B1$="":B2$="":B3$="":B4$=""
110 FOR I=1 TO 16
115 READ A
120 B1$=B1$+CHR$(A)
125 NEXT I
130 FOR I=1 TO 16
135 READ A
140 B2$=B2$+CHR$(A)
```

```
145 NEXT I
150 SPRITE$(0)=B1$+B2$
155 FOR I=1 TO 16
160 READ A
165 B3$=B3$+CHR$(A)
170 NEXT I
175 FOR I=1 TO 16
180 READ A
185 B4$=B4$+CHR$(A)
190 NEXT I
195 SPRITE$(1)=B3$+B4$
250 DATA 0,0,0,0,0,0,0,0,63,68,196,255,2
55,255,120,48
255 DATA 0,0,0,0,0,0,0,0,128,64,32,254,2
55,255,60,24
260 DATA 0,0,0,255,255,255,248,248,255,2
55,255,255,255,255,7,3
265 DATA 0,0,0,248,248,248,200,200,248,2
48,248,248,248,255,128,0
```

We hebben nu geleerd, hoe we een sprite kunnen vastleggen in de systeemvariabele SPRITE$(X). De volgende stap is, hoe we een sprite op het beeldscherm plaatsen en hoe we de sprite vervolgens laten bewegen. Hiervoor dient het statement PUT SPRITE, dat in de volgende paragraaf zal worden behandeld.

8.3 Het plaatsen van sprites op het beeldscherm.

In paragraaf 8.2 hebben we geleerd hoe figuren kunnen worden vastgelegd in sprites van 8*8 en 16*16 beeldpuntjes. Deze sprites kunnen eventueel vergroot worden weergegeven, dat wil zeggen, dat dan de dubbele grootte (16*16 of 32*32) van het oorspronkelijke formaat wordt verkregen.

Voordat het statement PUT SPRITE zal worden behandeld, zullen eerst een aantal belangrijke punten over het gebruik van sprites worden gegeven:

a)
Sprites kunnen niet worden toegepast in de tekstmodes 1 en 2 (SCREEN 0 en 1).

b)
Er mogen geen verschillende sprite-groottes door elkaar worden gebruikt. Men kan maar 1 sprite-grootte (0, 1, 2 of 3) tegelijkertijd toepassen. De sprite-grootte wordt aangegeven in het SCREEN-statement.

c)
Er mogen maximaal 32 sprites tegelijkertijd op het beeldscherm worden geplaatst, met een maximum van 4 sprites op dezelfde horizontale regel.

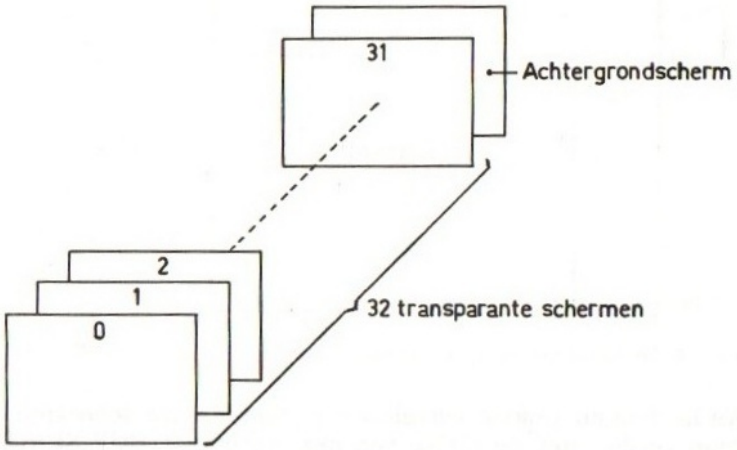
d)
Er mag, tegelijkertijd, op elk transparant scherm maar 1 sprite worden geplaatst.

Om de laatste twee punten te verduidelijken, zal eerst iets meer worden verteld over de opbouw van het beeldscherm bij gebruikmaking van sprites.

Tot nu toe is er alleen getekend op het achtergrondscherm en niet op de transparante schermen. Bij toepassing van sprites bestaat het beeldscherm uit een normaal achtergrondscherm en 32 transparante (doorzichtige) schermen (zie afbeelding 8-3).

Op de transparante schermen kunnen we nu met het statement PUT SPRITE van te voren gedefinieerde sprites plaatsen. Op elk transparant scherm kan maar 1 sprite worden geplaatst. De transparante schermen zijn van voor naar achter genummerd van 0 tot en met 31. Hierbij heeft scherm 0 de hoogste prioriteit en scherm 31 de laagste. Dat wil zeggen, dat wanneer een sprite op scherm 0 wordt geplaatst en een andere sprite op scherm 3, de sprite op scherm 0 prioriteit zal hebben boven de sprite op scherm 3. Worden bijvoorbeeld beide sprites langs elkaar heen bewogen, dan zal de sprite op scherm 3 tijdens het passeren achter de sprite op

scherm 0 verdwijnen. Denk hierbij maar aan twee auto's, die elkaar passeren.

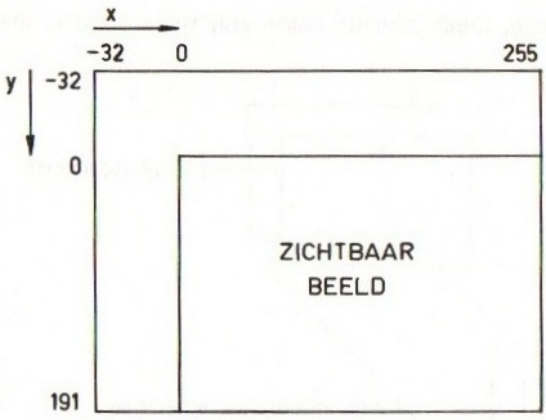


Afb. 8-3 Transparante schermen.

De nummers van de transparante schermen worden dan ook prioriteitsnummers genoemd. Soms worden de nummers ook wel transparantnummers genoemd.

Wanneer men een figuur wil opbouwen uit meerdere sprites, dan zullen de sprites elk op een apart scherm moeten worden geplaatst. Men zou bijvoorbeeld van een auto het voorwiel op scherm 3 kunnen plaatsen, het achterwiel op scherm 2, het voorstuk van de auto op scherm 1 en het achterstuk van de auto op scherm 0. Dit heeft als voordeel, dat de sprites verschillende kleuren kunnen worden gegeven. Normaal kan een sprite door middel van het statement PUT SPRITE maar in 1 kleur worden uitgevoerd. Verder kan dezelfde sprite (bijvoorbeeld een auto) op meerdere schermen worden geplaatst.

De gedefinieerde sprites kunnen door middel van het statement PUT SPRITE op elke willekeurige positie op het scherm worden geplaatst, zelfs buiten het scherm (zie afb. 8-4).



Afb. 8-4 Beeldschermafmeting voor sprites.

We hebben nu genoeg verteld over transparante schermen en gaan verder met de uitleg van het statement PUT SPRITE, waarmee sprites op de transparante schermen kunnen worden geplaatst. Het statement heeft het volgende formaat:

PUT SPRITE P,(x,y),K,X

- X — Sprite-nummer.
- K — Kleurnummer (0-15).
- (x,y) — Coordinaten van de positie waar de sprite moet worden geplaatst.
- P — Prioriteitsnummer (0-31).

Prioriteitsnummer

Het prioriteitsnummer kan varieren tussen 0 en 31 en geeft in wezen het nummer aan van het transparante scherm waarop de sprite met het sprite-nummer **X** zal worden geplaatst. Wanneer twee sprites op dezelfde positie worden geplaatst, zal de sprite met het laagste prioriteitsnummer zich voor de sprite met het hogere prioriteitsnummer bevinden. Met andere woorden, de sprite met het hogere prioriteitsnummer zal men niet zien.

Positie op het scherm
De positie waar de sprite op het scherm moet worden geplaatst, wordt aangegeven door de x- en y-coordinaten. De x-coordinaat kan een waarde hebben van -32 tot en met 255 en de y-coordinaat van -32 tot en met 191. Het is daarom mogelijk een sprite buiten het scherm te plaatsen.

Coordinaat y kan ook een waarde hebben van 208 en 209. Indien y=208, dan zullen alle sprites met een lagere prioriteit dan de gedefinieerde sprite in het PUT SPRITE statement van het scherm verdwijnen. Indien y=209, dan zal alleen de sprite, die in het PUT SPRITE statement is gedefinieerd, van het scherm verdwijnen.

De x- en y-coordinaten geven de positie aan van het beeldpuntje, dat zich in de linker bovenhoek van de sprite bevindt. De parameters x en y mogen ook relatieve waarden hebben. In dat geval moeten x en y worden voorafgegaan door het woord **STEP**.

Kleur van de sprite
De kleur, waarin de sprite zal worden uitgevoerd, wordt aangegeven door kleurnummer K (0-15). Wanneer K niet is gegeven, geldt de laatst gedefinieerde voorgrondkleur. De defaultwaarde voor de voorgrondkleur is 15 (wit).

Sprite-nummer
Het sprite-nummer **X** bepaalt welke sprite op het transparante scherm **P** (0-31) zal worden geplaatst. De sprite is vastgelegd in de variabele SPRITE$(X). Wanneer X niet is gegeven, is het sprite-nummer gelijk aan het prioriteitsnummer P.

Hierna volgen een aantal voorbeelden van het gebruik van sprites. Het eerste voorbeeld is het achtereenvolgens plaatsen van de letter A op het beeldscherm in steeds een andere kleur.

```
10 COLOR 15,15,15
15 SCREEN 2,0
20 B$=""
```

```
25 FOR I=1 TO 8
30 READ A
35 B$=B$+CHR$(A)
40 NEXT I
45 SPRITE$(1)=B$
50 FOR K=0 TO 15
55 PUT SPRITE 0,(128,90),K,1
60 FOR I=1 TO 1000:NEXT I
65 NEXT K
70 COLOR 15,4,4:END
75 DATA 24,36,66,66,126,66,66,66
```

Met de statements op de regelnummers 20 tot en met 45 wordt de letter A vastgelegd in de systeemvariabele SPRITE$(1). Met het statement op regelnummer 55 wordt de sprite op het transparante scherm 1 geplaatst in steeds een andere kleur. Door het statement op regelnummer 15 te veranderen in SCREEN 2,1, zal de sprite van 8*8 beeldpuntjes vergroot worden weergegeven op het scherm.

Met het volgende programma zal een auto over het scherm worden voortbewogen.

```
10 COLOR 1,15,15
15 SCREEN 2,2
20 LINE (0,90)-(255,90),1
25 B1$=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+C
HR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(63)+
CHR$(68)+CHR$(196)+CHR$(255)+CHR$(255)+C
HR$(255)+CHR$(120)+CHR$(48)
30 B2$=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+C
HR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(128)
+CHR$(64)+CHR$(32)+CHR$(254)+CHR$(255)+C
HR$(255)+CHR$(60)+CHR$(24)
35 SPRITE$(0)=B1$+B2$
40 FOR I=-32 TO 255 STEP .5
45 PUT SPRITE 1,(I,74),2,0
50 NEXT I
55 GOTO 40
```

Met het statement LINE op regelnummer 20 wordt op het achtergrondscherm (zie afbeelding 8-3) een zwarte lijn getekend. Met de statements op de regelnummers 25 tot en met 35 wordt de auto, uitgevoerd als sprite, vastgelegd in de systeemvariabele SPRITE$(0). Met het PUT SPRITE statement op regelnummer 45 wordt de sprite (auto) op het transparante scherm nummer 1 geplaatst. De beweging wordt verkregen door de x-coordinaat (variabele I) te varieren. De sprite wordt zodoende steeds iets verder op het scherm geplaatst, hetgeen wordt waargenomen als een vloeiende beweging. Door de stapgrootte in het FOR-statement kleiner of groter te maken, zal de snelheid van de auto respectievelijk kleiner of groter worden.

Door de volgende twee statements als volgt te veranderen:

```
15 SCREEN 2,2
45 PUT SPRITE 1,(I,58),2,0
```

zal de auto vergroot worden weergegeven op het scherm. Men zou ook nog de kleur kunnen varieren, door deze in het begin van het programma met een INPUT-statement in te voeren. De volgende regel zal dan aan het programma moeten worden toegevoegd:

```
11 INPUT "Kleur auto (1-14)";K
```

en de volgende regel zal als volgt moeten worden veranderd:

```
45 PUT SPRITE 1,(I,58),K,0
```

We kunnen het programma verder uitbreiden door op de achtergrond met LINE-statements een huis te tekenen. Het programma zal dan als volgt worden uitgebreid:

```
21 LINE (100,90)-(160,50),1,B
22 LINE (100,50)-(130,5),1:LINE (130,5)-
(160,50),1
23 LINE (110,80)-(130,60),1,B
24 LINE (140,90)-(155,60),1,B
```

Opmerking:
Het huis is geprogrammeerd voor SCREEN 2,2.

Wanneer we het voorgaande programma vanaf regelnummer 40 als volgt aanpassen, zullen er twee auto's over het scherm rijden.

```
40 I=-32:J=70
45 PUT SPRITE 2,(I,58),K,0
50 PUT SPRITE 1,(J,58),1,0
55 I=I+1:J=J+1
60 IF I>=255 THEN I=-32
65 IF J>=255 THEN J=-32
70 GOTO 45
```

De snelheid van de auto's kan worden veranderd door op regelnummer 55 de constanten, die bij I en J worden opgeteld, een andere waarde te geven. Door bijvoorbeeld bij I de waarde 2 op te tellen en bij J de waarde 1, zullen de auto's met verschillende snelheden over het scherm gaan.

Opmerking:
De waarde van de x-coordinaat mag niet kleiner zijn dan -32 en niet groter dan 255.

Het volgende programma laat een auto met een caravan erachter en nog een tweede auto over het scherm rijden. Voor de twee auto's wordt gebruik gemaakt van dezelfde sprite, namelijk SPRITE$(0). De caravan is vastgelegd in SPRITE$(1). De auto en de caravan worden op de transparante schermen 0 en 1 geplaatst en de tweede auto op scherm 2. Dat betekent, dat de auto op scherm 2 tijdens het passeren achter de auto met caravan langs gaat.

```
100 SCREEN 2,3
101 LINE (0,90)-(255,90),1
105 B1$="":B2$="":B3$="":B4$=""
110 FOR I=1 TO 16
115 READ A
```

```
120 B1$=B1$+CHR$(A)
125 NEXT I
130 FOR I=1 TO 16
135 READ A
140 B2$=B2$+CHR$(A)
145 NEXT I
150 SPRITE$(0)=B1$+B2$
155 FOR I=1 TO 16
160 READ A
165 B3$=B3$+CHR$(A)
170 NEXT I
175 FOR I=1 TO 16
180 READ A
185 B4$=B4$+CHR$(A)
190 NEXT I
195 SPRITE$(1)=B3$+B4$
200 I=0:J=-32:K=90
205 PUT SPRITE 0,(I,58),6,0
210 PUT SPRITE 1,(J,58),4,1
215 PUT SPRITE 2,(K,58),1,0
220 I=I+1:J=J+1:K=K+2
225 IF I>=255 THEN I=-32
230 IF J>=255 THEN J=-32
235 IF K>=255 THEN K=-32
240 GOTO 205
250 DATA 0,0,0,0,0,0,0,0,63,68,196,255,2
55,255,120,48
255 DATA 0,0,0,0,0,0,0,0,128,64,32,254,2
55,255,60,24
260 DATA 0,0,0,255,255,255,248,248,255,2
55,255,255,255,255,7,3
265 DATA 0,0,0,248,248,248,200,200,248,2
48,248,248,248,255,128,0
```

8.4 Samenvallen van sprites

Wanneer twee sprites op het scherm samenvallen of met elkaar in botsing komen, is het mogelijk het programma dat in

uitvoering is tijdelijk te onderbreken en vervolgens een subroutine, die betrekking heeft op de gebeurtenis, aan te roepen. Zo'n onderbreking wordt ook wel gezien als een niet geprogrammeerde sprong.

We zouden bijvoorbeeld een Pac-mannetje spookjes kunnen laten opeten. Elke keer dat het Pac-mannetje in aanraking komt met een spookje, kunnen we het Pac-mannetje het spookje laten opeten. Deze handeling kunnen we laten uitvoeren door de betreffende subroutine.

Het statement RETURN aan het einde van de aangeroepen subroutine zorgt ervoor, dat het programma de draad weer oppakt, waar het eerder door het samenvallen van de sprites werd onderbroken.

Met behulp van het statement

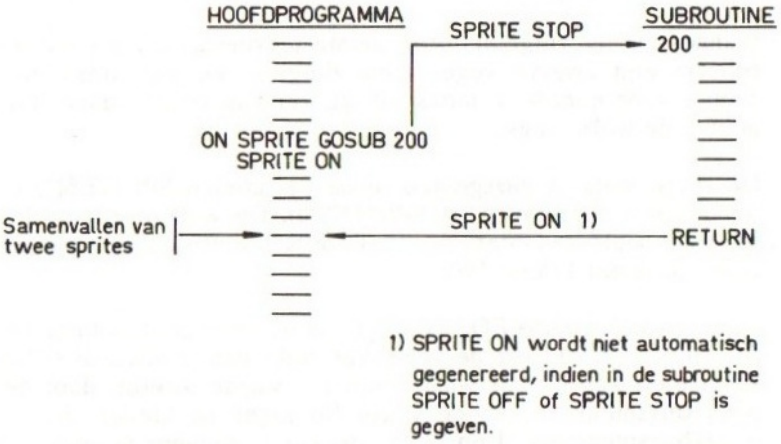
ON SPRITE GOSUB regelnummer

wordt de mogelijkheid geschapen om bij het samenvallen van twee sprites naar een subroutine op een gegeven **regelnummer** te springen. Dit statement kunnen we ergens aan het begin van het programma zetten. Zodra dit statement is uitgevoerd, weet de computer dat er, indien twee sprites samenvallen (in botsing komen), naar een subroutine moet worden gesprongen. Afbeelding 8-5 geeft een grafische weergave van de gebeurtenissen.

Het tijdelijk onderbreken van het programma en het vervolgens aanroepen van de bijbehorende subroutine als gevolg van het samenvallen van twee sprites, zal alleen plaats vinden wanneer dit is geactiveerd. Zonder activering heeft het samenvallen van twee sprites geen onderbreking van het programma tot gevolg. Voor het activeren dient het statement **SPRITE ON.** Na het uitvoeren van dit statement zal elke keer, voordat er met de uitvoering van een nieuw statement wordt begonnen, worden gecontroleerd of er twee sprites zijn samengevallen. Wanneer dit zo is, zal het programma worden onderbroken en de bij de sprites behorende subroutine worden

aangeroepen. Het eerste regelnummer van de subroutine is gegeven in het ON SPRITE GOSUB statement.

Wanneer de subroutine wordt aangeroepen, wordt door de computer automatisch een **SPRITE STOP** statement uitgevoerd (zie afbeelding 8-5). Dit betekent, dat het opnieuw samenvallen van sprites, tijdens het uitvoeren van de subroutine, geen effect zal hebben, zolang er geen SPRITE ON wordt gegeven.



Afb. 8-5 Het samenvallen van twee sprites.

Wanneer een SPRITE STOP is uitgevoerd, zal er geen onderbreking plaats vinden. Echter, het samenvallen zal wel worden onthouden, want zodra een SPRITE ON zal worden gegeven, wordt het programma alsnog onderbroken en de subroutine uitgevoerd. Een SPRITE STOP heeft geen zin, indien deze niet is voorafgegaan door een SPRITE ON.

Aan het einde van de subroutine keert het programma door middel van het statement RETURN weer terug naar het punt waar het eerder, door het samenvallen van twee sprites, werd onderbroken. Tijdens de uitvoering van het RETURN-statement wordt automatisch een SPRITE ON gegeven. Dit ge-

beurt echter niet, wanneer in de subroutine het statement SPRITE OFF of SPRITE STOP werd uitgevoerd.

Wanneer de mogelijkheid tot het onderbreken van het programma als gevolg van het samenvallen van twee sprites dient te worden opgeheven, kan dit worden bewerkstelligd met het statement **SPRITE OFF.** Er zal dan niet meer worden gecontroleerd of twee sprites zijn samengevallen. Na het beeindigen van een programma wordt automatisch een SPRITE OFF gegeven.

Wanneer u het volgende programma uitvoert, zult u over het scherm een zwarte vogel zien vliegen, die van links naar rechts voor een wolk langs vliegt, en van rechts naar links achter de wolk langs.

De witte wolk is vastgelegd in de variabelen SPRITE$(1) en SPRITE$(2) en de vogel in SPRITE$(3). De wolk wordt op het scherm geplaatst met de PUT SPRITE statements op de regelnummers 175 en 180.

Door middel van de FOR...NEXT lus op de regelnummers 185 tot en met 195 vliegt de vogel van links naar rechts over het scherm en wel voor de wolk langs. Dit wordt bereikt, door het prioriteitsnummer van de vogel (0) lager te kiezen dan de prioriteitsnummers (1 en 2) van de wolk. Wanneer de vogel de rechter kant van het scherm heeft bereikt, wordt het prioriteitsnummer 3 gemaakt. Dit betekent, dat de vogel van rechts naar links achter de wolk langs gaat. Zie regelnummers 200 tot en met 210. Wanneer de vogel de linkerkant van het scherm heeft bereikt, keert de vogel weer om en wordt het prioriteitsnummer weer 0. Dit betekent, dat de vogel weer voor de wolk langs vliegt.

```
100 SCREEN 2,1
105 FOR I=1 TO 2
110 B$=""
115 FOR J=1 TO 8
120 READ A
125 B$=B$+CHR$(A)
```

```
130 NEXT J
135 SPRITE$(I)=B$
140 NEXT I
145 B$=""
150 FOR J=1 TO 8
155 READ A
160 B$=B$+CHR$(A)
165 NEXT J
170 SPRITE$(3)=B$
175 PUT SPRITE 1,(150,40),15,1
180 PUT SPRITE 2,(166,40),15,2
185 FOR I=-32 TO 255 STEP .5
190 PUT SPRITE 0,(I,40),1,3
195 NEXT I
200 FOR I=255 TO -32 STEP -.5
205 PUT SPRITE 3,(I,40),1,3
210 NEXT I
215 GOTO 185
220 DATA 0,3,31,63,255,255,63,7
225 DATA 0,128,24,255,255,254,252,192
230 DATA 0,0,0,108,146,16,0,0
```

Men zou de vogel bij aankomst bij de wolk weer kunnen laten terugvliegen. Dit kan worden opgelost door gebruik te maken van het ON SPRITE GOSUB statement. Zodra de sprite van de vogel in aanraking komt met de sprite van de wolk, zal naar een subroutine worden gesprongen, die de terugkeer van de vogel bewerkstelligd. De volgende statements zullen dan aan het programma moeten worden toegevoegd:

```
101 ON SPRITE GOSUB 300
102 SPRITE ON


300 SPRITE OFF
305 FOR J=I TO -30 STEP -.5
310 PUT SPRITE 0,(J,40),1,3
315 NEXT J
320 SPRITE ON
325 RETURN 185
```

Men kan door middel van functietoetsen het terugkeren van de vogel, wanneer deze de wolk heeft bereikt, uitschakelen. Dit kan worden opgelost door een SPRITE OFF te geven. Laten we voor het geven van een SPRITE OFF functietoets F2 nemen en voor het geven van SPRITE ON functietoets F3. Het programma zal dan als volgt moeten worden uitgebreid:

```
103 ON KEY GOSUB ,400,450
104 KEY(2) ON:KEY(3) ON

400 SPRITE OFF
405 RETURN
450 SPRITE ON
455 RETURN
```

Opmerking:

De functietoetsen mogen alleen bediend worden als de vogel zich buiten de wolk bevindt.

9 Wat is machinetaal?

Over MSX-BASIC schrijven wij, zonder het gevoel te hebben er te diep op in te gaan, drie boeken vol. MSX-BASIC bestaat uit ongeveer 150 statements. Machinetaal is ook een programmeertaal, doch die bestaat uit nog meer statements. Bovendien is de aard van die statements meer gericht op de computer, dan op de mens die de computer moet programmeren. Het zal dan ook duidelijk zijn, dat we in dit hoofdstuk geen cursus machinetaalprogrammeren geven. Als dit hoofdstuk u een indruk geeft van wat machinetaal nu eigenlijk is, dan hebben wij ons doel bereikt. Wilt u meer over machinetaal weten, dan kunt u daarover alles lezen in afzonderlijke boeken, die in de boekhandel zijn te verkrijgen.

Machinetaal is de taal die de computer kan verstaan. Dit wil zeggen dat de computer in staat is om de instructies (statements) direct te interpreteren en om te zetten naar logische signalen. Met deze logische signalen worden de verschillende delen van de computer aangestuurd.

In het eerste deel van deze serie leerboeken hebben we al kort aandacht besteed aan de interne opbouw van een computer. We zagen daar onder meer dat in iedere microcomputer een uitvoerend orgaan zit, de microprocessor. De microprocessor is een chip waarin een rekenorgaan zit. Hierdoor is de microprocessor in staat om logische signalen bij elkaar op te tellen of van elkaar af te trekken. Om dit te kunnen doen heeft de microprocessor een aantal ingebouwde registers, waarin de waarde van de logische signalen kan worden opgeslagen. Het meestgebruikte register is het A-register, ook wel de Accu of accumulator genoemd. Alle berekeningen gaan via dit register. Het berekende resultaat (heel vaak een tussenre-

sultaat van een langere berekening) wordt, om tijd te sparen vaak opgeslagen in een ander register. Het opslaan in een register gaat namelijk veel sneller dan het opslaan in het geheugen. De microprocessor heeft dus verschillende registers.

In de MSX-computer is gebruikt gemaakt van een Z80-microprocessor. Deze microprocessor heeft de volgende registers:

Hoofdregisterset		Hulpregisterset	
accumulator A	vlaggen F	accumulator A'	vlaggen F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

Interrupt vector I	Memory refresh R
Index register IX	
Index register IY	
Program counter PC	

Afb. 9-1 De Z80-registerset.

Waar komen nu de gegevens, waarmee moet worden gerekend, vandaan? Dat kan zijn uit het geheugen, maar dat kan ook zijn uit een invoer-apparaat zoals het toetsenbord of de joystick. En waar gaan de berekende resultaten naar toe? Die kunnen ofwel naar het geheugen gaan, of naar een uitvoer-apparaat zoals het beeldscherm of de printer. Ook zijn er nog apparaten die zowel invoer als uitvoer kunnen verzorgen, denk maar aan de cassetterecorder en de schijveneenheid.

Om iets uit het geheugen te kunnen lezen, moet dat geheugen

eerst worden geaddresseerd. Je wilt immers niet het hele geheugen in een keer uitlezen, maar alleen de inhoud van een bepaald adres. De micròprocessor dient dus te beschikken over een aantal adreslijnen. Om een geheugen van 64 Kbytes te kunnen adresseren, zijn 16 adreslijnen nodig. Heb je eenmaal een adres geselecteerd, dan zul je aan het geheugen moeten kenbaar maken of je iets uit dat adres wilt lezen of dat je er juist iets in wilt schrijven. De signaallijnen waarmee dit soort informatie tussen de verschillende componenten van een computer wordt uitgewisseld, worden "control"-lijnen genoemd. Is een geheugenadres geselecteerd, en is aangegeven of er gegevens moeten worden gelezen of geschreven, dan moeten de gegevens nog getransporteerd worden van het geheugen naar een register van de microprocessor of omgekeerd. Dit transport vindt plaats over de "data-lijnen". In de meeste literatuur wordt niet over lijnen gesproken, maar over een "bus".

In een microcomputer vinden we dus een adres-bus, een databus en een control-bus. Met behulp van deze bussen wordt informatie tussen de verschillende delen van de computer en de microprocessor heen en weer gezonden. Dit geeft een druk verkeer over de verschillende lijnen. Om dat verkeer in goede banen te leiden is een politie-agent nodig. De microprocessor speelt die politie-agent. Alles wat er binnen de computer gebeurt, wordt bestuurd door de microprocessor. Die microprocessor werkt echter (gelukkig maar) in opdracht van zijn programmeur. Een programmeur kan hem vertellen wanneer hij gegevens uit het geheugen moet lezen, van welk adres dat moet worden gelezen, en waarheen hij de gelezen data moet sturen. Het vervelende is echter, dat de programmeur een mens is en de microprocessor een chip. Zij spreken niet dezelfde taal.

De microprocessor weet precies wat hem te doen staat als hem wordt opgedragen een 201 uit te voeren. Dat zal u waarschijnlijk niet veel zeggen. Met uw kennis van BASIC zal de opdracht RETURN u waarschijnlijk veel meer zeggen. Welnu, in machinetaal bestaat er ook een terugkeer van een subroutine. In machinetaal gesproken is dat het decimale getal 201.

Alle machinetaal-instructies (statements) zijn getallen, die varieren van 0 tot en met 255.

```
62 201 201
```

Bovenstaande reeks getallen zal op u waarschijnlijk overkomen als abracadabra. Voor de microprocessor is het echter zeer klare taal. In feite staat daar :

62 201 : Laad de constante 201 in register A (accu) en
201 : keer terug van de subroutine.

U ziet nu ook meteen, dat in een reeks van getallen, niet alle getallen instructies zijn. Soms staan daar getallen tussen die deel uit maken van een instructie. Het zijn dan parameters die bij een instructie behoren. Deze getallen kunnen dan, zoals in voorgaand voorbeeld, constanten zijn, of adressen (geheugen-adressen bijvoorbeeld). Dit maakt het voor ons, mensen, niet gemakkelijker om de machinetaal te lezen.

Gelukkig zijn er software-ontwikkelaars geweest die dit probleem voor ons hebben opgelost. Voor iedere machinetaal instructie hebben ze een woord (van maximaal 4 letters) bedacht. Machinetaal programmeurs kunnen nu hun machinetaalprogramma schrijven in woorden. Een vertaalprogramma zet deze woorden dan om in echte (voor de microprocessor verstaanbare) machinetaal. Zo'n vertaalprogramma heet een "assembler". Om voorgaand voorbeeld in assembler-taal te schrijven, zouden we het volgende moeten programmeren:

```
LD      A,201
RET
```

U ziet het, dit is al veel leesbaarder. LD is de afkorting van het Engelse woord LoaD en RET is de afkorting van RETURN. Zouden we dit nog niet duidelijk genoeg vinden, dan bestaat er ook nog een mogelijkheid om commentaar aan de programmaregels toe te voegen. Dit komt overeen met het REM-statement uit BASIC. Commentaar staat wel in het programma, maar wordt niet door de computer uitgevoerd. Voorgaan-

de programma-listing inclusief commentaar zou zijn:

```
LD      A,201       ;201 naar Accu
RET                 ;terugkeren
```

Als we ons nu even de situatie in een BASIC-programma voorstellen, dan zou de eerste regel in BASIC overeenkomen met LET A=201. In BASIC hebben we altijd te maken met regelnummers. In assembler is dit ook zo. Om voorgaand voorbeeld dus helemaal compleet te maken zou het er als volgt dienen uit te zien.

```
1000    LD      A,201       ;201 naar Accu
1010    RET                 ;terugkeren
```

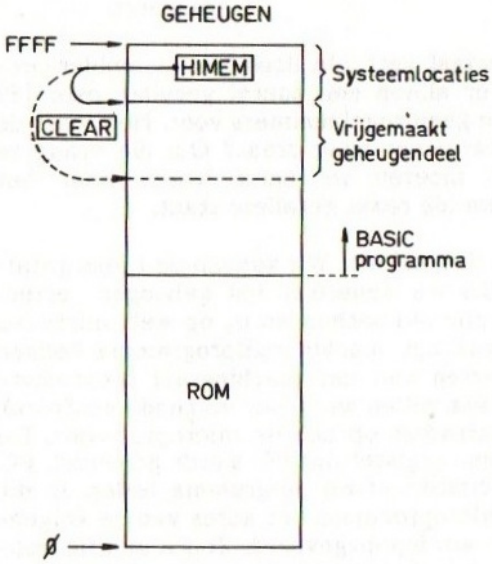
Is dit programma eenmaal vertaald door het assembler-programma, dan blijven er alleen een aantal getallen over (62 201 201). Hierin komen geen regelnummers voor. Hoe weet de computer dan welk statement waar staat? Om die vraag te kunnen beantwoorden moeten we eerst weten waar het machinetaal programma (de reeks getallen) staat.

In principe maken wij dat zelf uit. Wij kunnen de reeks getallen op iedere plaats die we wensen in het geheugen zetten. Het enige dat we dan moeten onthouden is, op welk adres we de eerste instructie van het machinetaalprogramma hebben weggezet. Bij het starten van het machinetaal programma (hoe dat in zijn werk gaat zullen we in het volgende hoofdstuk zien) geven we dat startadres op aan de microprocessor. De microprocessor heeft een register dat PC wordt genoemd. PC staat voor Program Counter, ofwel programma teller. In dit register onthoudt de microprocessor het adres van de volgende instructie die moet worden uitgevoerd. Zodra een instructie vanuit het geheugen in de microprocessor is geladen, om te worden uitgevoerd, wordt het PC-register opgehoogd, zodanig, dat het naar het adres van de volgende instructie wijst.

Stel nu dat de eerste instructie van ons voorbeeldprogramma op adres 50000 staat, dan zal de RET-instructie op adres

50002 staan. Na het laden van de LD-instructie staat register PC op 50002. Nu wordt de LD-instructie uitgevoerd. Zodra de microprocessor klaar is met de uitvoering, zal de volgende instructie vanaf het adres dat in PC staat worden geladen. Na het laden wordt PC weer opgehoogd en wordt de geladen instructie uitgevoerd, enz.

Hoewel we in principe vrij zijn om een machinetaal programma op iedere willekeurige plaats in het geheugen te zetten, zijn we toch gebonden aan een aantal beperkingen. Afbeelding 9-2 laat de lay-out van het geheugen van de MSX-computer zien.



Afb. 9-2 De geheugenindeling van de MSX-computer.

Meestal zullen we een machinetaal programma samen met een BASIC-programma in het geheugen willen hebben. Dat BASIC-programma heeft een vaste plaats in het geheugen. De lengte kan varieren. In onze computer zit ook een ROM-geheugen. Dit deel van het geheugen (de eerste 32 Kbytes) is

dus al bezet. In de ROM zitten een groot aantal machinetaal-routines die normaliter door het BASIC-programma worden gebruikt. In het volgende hoofdstuk zullen we zien dat we die routines ook vanuit een machinetaal-routine kunnen aanroepen. Hoe dan ook, we kunnen ons eigen machinetaal-programma niet in de door het ROM in beslag genomen geheugendeel plaatsen. De machinetaalroutines uit de ROM hebben ook een werkgeheugen nodig. ROM wil zeggen Read Only Memory. Vrij vertaald in het Nederlands is dat Alleen Uitleesbaar Geheugen. In een geheugen waaruit alleen maar informatie kan worden gelezen, kan dus niet geschreven worden. De ROM-routines willen hun resultaten toch ergens wegschrijven. Bovendien hebben deze routines wel eens variabele informatie nodig. Die wegschrijfplaatsen en die variabele informatie bevinden zich in het RAM, en wel vanaf het hoogste geheugenadres af naar beneden toe. De lengte van dit gebied is niet vast. Zodra een floppy-disk wordt aangesloten, zal de lengte van het "systeem"-RAM worden vergroot. Vandaar dat we hier geen vaste waarden noemen.

In het systeem-RAM is een adres gereserveerd, waar staat hoe groot het systeem-RAM-gebied is. Dit adres wordt **HIMEM** genoemd. In HIMEM vinden we het hoogste adres dat door het BASIC-programma mag worden gebruikt. Voor een standaard MSX-computer, zonder floppy-disk is het hoogste adres dat door BASIC mag worden gebruikt het adres &HF380, ofwel in decimalen 62336.

Als we nu weten, dat we ons machinetaalprogramma niet in het ROM-deel van het geheugen mogen zetten, en we mogen het niet in het systeem-RAM zetten, dan blijft alleen het BASIC-deel over. Dit deel is echter ook niet veilig, want als we ons BASIC-programma groter maken, kan het zijn dat we ons machinetaalprogramma overschrijven met een BASIC-statement. Er is echter een BASIC-statement dat ons te hulp schiet. Dit is het **CLEAR-statement.** In het eerste deel uit deze serie leerboeken hebben we al gezien, dat we met het CLEAR-statement de ruimte voor string-variabelen kunnen vergroten of verkleinen. CLEAR geeft echter nog meer mogelijkheden. Laten we eerst het algemene formaat van

CLEAR eens bekijken:

CLEAR string-ruimte , hoogste BASIC-adres

U ziet het al. Door in het CLEAR-statement achter de komma een adres te plaatsen, dat lager is dan het adres in HIMEM, creeren we een gat. Dat mag niet door het BASIC worden gebruikt, en wordt ook nergens anders voor gebruikt. Wat we op de adressen in dat niet gebruikte geheugendeel zetten, zal daar keurig blijven staan. Het is dus een prima plaats om er onze machinetaalroutine weg te zetten.

Nog een enkele tip. Om het hoogste geheugenadres dat door BASIC mag worden gebruikt te kunnen bepalen, dienen we te weten hoeveel geheugen het BASIC-programma gebruikt, en hoeveel geheugen er dus nog ongebruikt is. Hiertoe kunnen we gebruik maken van de functie FRE(0). Nadat we het BASIC-programma, dat straks samen met het machinetaalprogramma in het geheugen moet staan, hebben geladen, geven we het directe commando PRINT FRE(0). Het resultaat zal zijn dat we het aantal nog niet gebruikte geheugenbytes te zien krijgen. Stel dat dit aantal 5000 is. We weten dan dat we een stuk geheugen van maximaal 5000 bytes kunnen reserveren voor ons machinetaalprogramma. Op de standaard MSX-computer was het hoogste adres van het vrij te maken geheugengebied 62336. Hier kunnen we dus maximaal 5000 van af trekken. Machinetaalprogramma's zijn echter meestal vrij kort. Een stuk geheugen van bijvoorbeeld 1000 bytes zal vaak al ruim voldoende zijn. Om een stuk geheugen van 1000 bytes vrij te maken (in ons voorbeeld) zullen we het commando CLEAR 200,61336 moeten geven. We zouden maximaal (in ons voorbeeld) CLEAR 200,57336 mogen geven. Gaan we verder naar beneden, dan zal er onvoldoende ruimte voor het BASIC-programma overblijven.

Nu we weten, waar we een machinetaalroutine in het geheugen kunnen plaatsen, willen we natuurlijk ook nog weten hoe we dat kunnen doen. Hiervoor zijn verschillende mogelijkheden. In dit hoofdstuk zullen we ons beperken tot het gebruik van het POKE-statement.

Als we hebben gezegd, dat het hoogste adres dat door het BASIC-programma mag worden gebruikt, het adres 61336 is, dan mag de eerste instructie van ons machinetaalprogramma op adres 61337 starten. Het volgende BASIC-programma zal het voorbeeld machinetaalprogramma in het geheugen zetten.

```
10 CLEAR 200,61336
20 POKE 61337,62
30 POKE 61338,201
40 POKE 61339,201
50 END
```

U ziet, dat het **POKE-statement** twee parameters heeft, die van elkaar zijn gescheiden door een komma. De eerste parameter is het geheugenadres, waar de tweede parameter naartoe moet worden geschreven.

Om nu te controleren of het programma werkelijk heeft gedaan wat we er van verwachten, kunnen we het volgende programma intikken.

```
10 ADRES=61337
20 FOR I=ADRES TO ADRES+2
30 PRINT PEEK(I);
40 NEXT I
50 END
```

Na uitvoering van dit programma ziet u, als alles goed is gegaan, op uw beeldscherm de getallenreeks 62 201 201. Dit is de inhoud van de geheugenadressen 61337, 61338 en 61339. De inhoud van die adressen is gelezen met de **functie PEEK(adres)** op regelnummer 30. PEEK is dus een functie en kan nooit zelfstandig worden gebruikt. Dit in tegenstelling tot POKE. POKE is een statement en kan wel zelfstandig worden gebruikt. Achter PEEK volgt slechts 1 parameter, die tussen haakjes moet worden geplaatst. Die parameter geeft het uit te lezen geheugenadres aan.

Wie al eens artikelen in tijdschriften over machinetaal heeft gezien, of heeft gekeken in boeken over het programmeren in

machinetaal, die zal het zijn opgevallen, dat daarin meestal met hexadecimale waarden wordt gewerkt. Vroeger zag je zefs vaak machinetaalprogramma's in octale code. Waarom wordt er zo vaak in andere talstelsels gewerkt?

Laten we de LD-instructie nog eens onder de loep nemen. De code van die instructie was 62 (decimaal). De microprocessor bekijkt die code echter niet als een decimaal getal maar als een serie bitjes (8 bitjes). Om een serie van acht bitjes op te schrijven is tamelijk veel werk. Je kunt die acht bitjes echter ook in twee groepjes van vier verdelen en ieder groepje van vier als een hexadecimaal cijfer opschrijven. Op die manier ontstaat een korte en goed leesbare notatie, die een sterke relatie heeft met de binaire weergave. Het decimale getal 62 is hexadecimaal 3E. Binair komt dit overeen met de serie 00111110.

00111110 is de serie logische signalen die aan de microprocessor worden aangeboden (de instructie). De microprocessor decodeert deze signalen. In het begin van dit hoofdstuk zagen we dat de code 62 betekende: LD A,constante.
Register A zou met een constante moeten worden geladen. De microprocessor kent de letter A niet. Wel weet de microprocessor dat, als wij zeggen register A, dat hij dan naar een code 111 moet kijken in de instructie. Voor register B verwacht de microprocessor een code 000. Deze codes worden door de microprocessor op een bepaalde plaats binnen de instructie verwacht. We schrijven de instructie LD nogmaals op.

62 (dec) = 3E (hex) = 00 111 110 (bin)

De binaire cijfers zijn nu opgedeeld in een groepje van twee en twee groepjes van drie bits. Het middelste groepje bits vertelt de microprocessor welk register er in het geding is. In dit geval is dat groepje bits 111, hetgeen overeenkomt met register A. De microprocessor heeft echter voor ieder register een andere code. Hier volgen de verschillende codes.

A = 111

B = 000
C = 001
D = 010
E = 011
H = 100
L = 101

Willen we nu een LD H,constante uitvoeren, dan hoeven we niets anders te doen dan de achter H gegeven bits in de LD instructie in te voegen. We krijgen dan: 00 100 110.

00100110 (bin) = 26 (hex) = 38 (dec). De decimale waarde 38 betekent dus voor de microprocessor LD H,constante.

In het octale talstelsel worden getallen gemaakt door steeds de waarde van drie bits met 1 cijfer (0 t/m 7) weer te geven. Uit de indeling van de machinetaal instructie ziet u wel, dat het octaal weergeven van een instructie erg eenvoudig is. Daar, waar slechts 2 bits aanwezig zijn, wordt een bit met de waarde 0 aan de linkerkant toegevoegd. Voor ons voorbeeld geldt dan:

00 100 110 = 000 100 110 = 046 (octaal)

In dit geval is het register onmiddellijk uit de octale notatie te lezen. In de beginjaren van de microcomputer was de octale notatie erg populair. Tegenwoordig wordt praktisch alleen nog maar met hexadecimale notatie gewerkt.

Appendix B bevat een "utility"-programma (gebruiksprogramma) waarmee u hexadecimale codes kunt ingeven en opslaan in het geheugen van de computer. Met behulp van dit programma kunnen we de in het volgende hoofdstuk te behandelen korte machinetaalroutines op een gemakkelijke manier ingeven. Bovendien is in dit programma een editor ingebouwd, zodat gemaakte fouten gemakkelijk kunnen worden gecorrigeerd. Is een machinetaalroutine eenmaal goed ingegeven, dan bevat het programma functies waarmee het programma naar cassette kan worden geschreven. Er zijn twee versies van het programma opgenomen. De eerste versie is geheel in

BASIC geschreven. De tweede bevat een machinetaalroutine. Beide versies doen echter precies hetzelfde. De versie met de machinetaalroutine doet het alleen een stuk sneller.

Wij raden u aan een van beide programma's in te tikken. Indien u het programma met de machinetaalroutine intikt, weest u dan zo verstandig, om het programma na het intikken **eerst naar cassette te schrijven**, voordat u het start. Mocht er een tikfoutje in de machinetaalroutine zijn gemaakt, dan is de kans groot dat het programma "crashed". Dit wil zeggen dat u niets anders meer kunt doen dan de computer uitzetten en opnieuw beginnen. Dit kan u vele uren typewerk kosten. Lees vervolgens de aanwijzingen en de uitleg in appendix B aandachtig door. Hebt u al een programma, waarmee u hexadecimale code kunt intikken, dan kunt u dat ook gebruiken voor de oefeningen uit het volgende hoofdstuk.

10 Aanroepen van machinetaalroutines

Onmiddellijk na het aanschakelen van de computer, hebben alle interne registers van de microcomputer de waarde 0. Dit geldt ook voor het register PC (Program Counter). Zoals u zich uit het vorige hoofdstuk zult herinneren, wijst dit register naar de eerstvolgende machinetaalinstructie die door de microprocessor moet worden uitgevoerd. Daar de inhoud van PC nu 0 is, wijst dit register dus naar adres 0 van het geheugen. Op de adressen 0 tot en met 32768 (32 Kbytes) zit het ROM-geheugen. Adres 0 is dus het eerste adres van het ROM. En daar verwacht de microprocessor de eerstvolgende instructie te vinden.

Gelukkig wist de fabrikant dit ook, dus heeft die er voor gezorgd, dat er op dat adres inderdaad een machinetaalroutine start. De microprocessor kan dus rustig zijn eerste instructie uit het geheugen lezen, daarna de volgende, enz.

De routine die op adres 0 start, initialiseert de gehele computer. Dit wil onder meer zeggen, dat het geheugen wordt getest op goede werking en dat het systeemdeel van het RAM (voor standaard configuraties vanaf adres FFFF tot en met F380) met de gewenste waarden wordt gevuld. Na initialisatie wordt de BASIC-interpreter gestart (ook dit is een verzameling van machinetaal-routines) en wordt een bericht naar het beeldscherm gestuurd, om ons te laten weten, dat we kunnen beginnen met het invoeren van een commando of van een programmaregel.

Als wij nu zelf machinetaalroutines gaan schrijven, dan is het goed ons te realiseren, dat er al een hele ROM vol machinetaalroutines in onze computer aanwezig is. Veel van de routi-

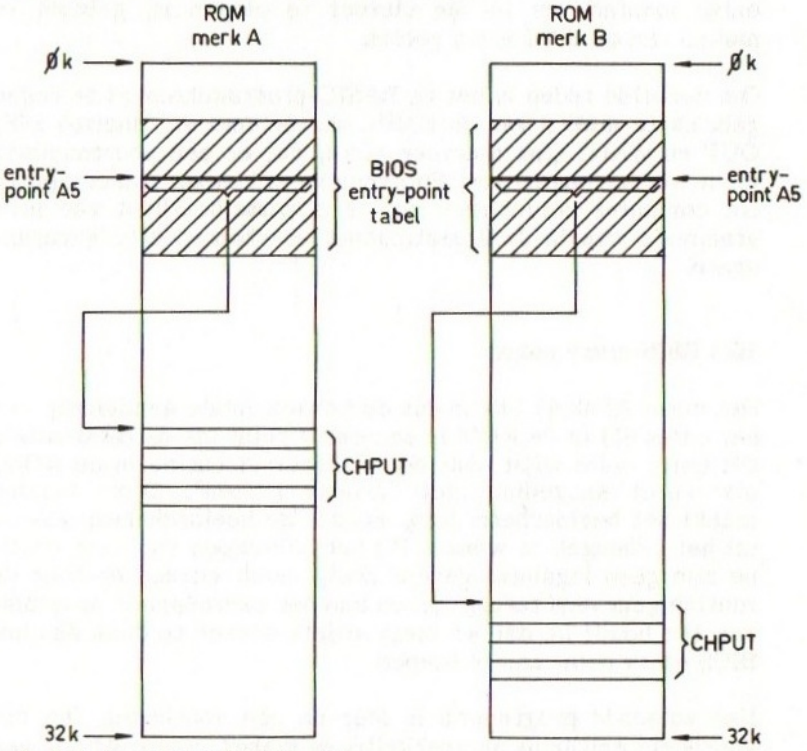
nes die we zelf zouden willen schrijven zitten ook al in die ROM. Een aantal van die routines zullen we in de loop van dit hoofdstuk tegenkomen. Daar we ons in dit boek niet toeleggen op het machinetaalprogrammeren, maar alleen willen laten zien wat machinetaalprogrammeren is, zullen we niet een complete lijst van alle in de ROM aanwezige machinetaalroutines geven. Boeken die speciaal over dit onderwerp zijn geschreven, zullen u de volledige informatie kunnen verschaffen.

We dienen ons wel goed te realiseren, dat iedere fabrikant zijn eigen ROM programmeert. Dit houdt ook in, dat de machinetaalroutines van computer tot computer op verschillende plaatsen binnen de ROM kunnen staan. Zouden wij de plaats van een bepaalde routine in de ROM van een bepaalde computer weten, dan zouden we die vanuit een programma kunnen aanroepen. Dat programma zal dan echter hoogstwaarschijnlijk op een andere computer niet werken, omdat dezelfde routine in die computer op een andere plaats staat. De ontwerpers van de MSX-standaard hebben dit echter voorzien. Ze hebben een tabel met startadressen van de belangrijkste routines tot standaard verheven.

In die tabel, die voor alle MSX-computers aanwezig dient te zijn, staan in alle computers dezelfde routines in dezelfde volgorde. Bovendien is de plaats van die tabel vastgelegd. Door nu machinetaalroutines uit het ROM altijd via deze tabel aan te roepen, verkrijgen we machinetaalprogramma's die op alle MSX-computers draaien. In afbeelding 10-1 is het geheugen van twee verschillende MSX-computers weergegeven. In beide computers start de tabel met startadressen van machinetaalroutines op hetzelfde adres. De machinetaalroutines die via de tabel worden aangeroepen bevinden zich echter, zoals de afbeelding laat zien, op twee heel verschillende plaatsen.

De startadressen uit de genoemde tabel worden aangeduid met BIOS-entry-points. De letters BIOS zijn een afkorting van Basic Input Output System. De tabel bestaat dus eigenlijk uit de basis in- en uitvoer routine adressen. Het is ook inderdaad mogelijk om met behulp van de BIOS-entry-points alle in- en

uitvoer naar scherm, toetsenbord, cassette, enz. te realiseren. Niet alleen is dat mogelijk, ook is dat ten zeerste aan te raden, en wel om de volgende redenen.



Afb. 10-1 Verschillen in ROM-inhouden.

Invoer en uitvoer van gegevens gaat via zogenoemde "poorten". Die poorten zijn hardware-componenten. Die hardware componenten zijn van een bepaald adres voorzien. Ze kunnen dus op dezelfde manier worden geadresseerd als een geheugen. De fabrikant is echter vrij om het werkelijke adres van een bepaalde poort te kiezen. Zou je nu in een machinetaal programma de instructie IN of OUT gebruiken, dan moet je ook aangeven van welke of naar welke poort gegevens moeten worden getransporteerd. Het zal duidelijk zijn dat, indien de poortnummers voor bijvoorbeeld een printer, van computer

tot computer verschillen, er op die manier nooit een programma kan worden geschreven dat op alle computers goed werkt. De beste manier (en uit een oogpunt van compatibility de enige manier) om in- en uitvoer te plegen is, gebruik te maken van de BIOS entry points.

Om dezelfde reden is het de BASIC-programmeur af te raden gebruik te maken van de BASIC-statements en functies INP, OUT en WAIT. Ook hiervoor geldt, dat er een poortnummer moet worden opgegeven. Poortnummers kunnen van computer tot computer verschillen, dus is uitwisselbaarheid van programma's waarin deze statements voorkomen niet gegarandeerd.

10.1 BIOS entry points

Het adres &H0041 (dit is dus de hexadecimale aanduiding van het adres 65) in de ROM is een entry point uit de BIOS-tabel. Dit entry point wijst naar de machinetaalroutine in de ROM, die wordt aangeduid met "disable screen". Deze routine maakt het beeldscherm leeg, zonder de beeldschermgegevens uit het geheugen te wissen. Bij het aanroepen van deze routine zijn geen ingangsgegevens nodig, noch worden er door de routine gegevens teruggegeven aan het aanroepende programma. Dit houdt in, dat we niets anders hoeven te doen dan het BIOS entry point aan te roepen.

Het volgende programma is hiervan een voorbeeld. Om het voorbeeld enigszins aantrekkelijk te maken, wordt er nog een BIOS entry point aangeroepen, en wel het entry point op adres &H0044. Hier staat het startadres van de machinetaalroutine "enable screen". Deze routine zorgt ervoor dat de informatie uit het videogeheugen weer op het scherm verschijnt. Door beide entry points afwisselend aan te roepen, ontstaat een knippereffect. Het gehele beeld gaat knipperen.

Vanuit BASIC kunnen de entry points op twee manieren worden aangeroepen; rechtstreeks of via een eigen machinetaalroutinetje. Van beide manieren volgt nu een voorbeeld, met daarbij de opmerking, dat van deze voorbeelden de recht-

streekse manier de beste is, omdat er geen gegevens tussen het BASIC-programma en de ROM-routine behoeven te worden uitgewisseld.

```
100 SCREEN 0: WIDTH 37: CLS: KEY OFF
110 FOR I=0 TO 23
120 PRINT "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
130 NEXT I
140 DEFUSR0=&H41: 'DISSCR
150 DEFUSR1=&H44: 'ENASCR
160 DUMMY=USR0(0): 'Disable screen
170 FOR I=0 TO 200: NEXT I
180 DUMMY=USR1(0): 'Enable screen
190 FOR I=0 TO 400: NEXT I
200 GOTO 160
```

Regels 100 tot en met 130 bouwen een beeldscherm op. Regels 140 en 150 definieren de startadressen van de machinetaalroutines 0 en 1. Regel 160 roept machinetaalroutine 0 aan (disable screen). Regel 170 laat het scherm gedurende korte tijd leeg (pauze). Regel 180 roept machinetaalroutine 1 aan (enable screen). Regel 190 laat het scherm gedurende korte tijd vol. Regel 200 ten slotte zorgt ervoor dat het scherm weer leeg wordt gemaakt.

Het volgende programma geeft exact hetzelfde resultaat, doch nu worden de entry points via een eigen machinetaalroutine aangeroepen. Vanuit BASIC roepen we onze machinetaalroutine aan, en de machinetaalroutine roept het entry point aan.

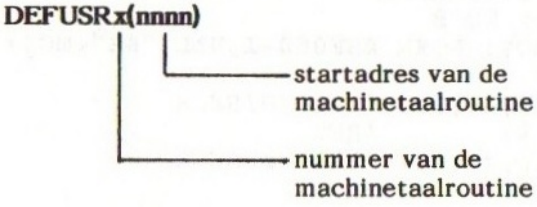
```
100 SCREEN 0: WIDTH 37: CLS: KEY OFF
110 CLEAR 200,&HF000
120 FOR I=1 TO 8
130 READ MC$: POKE &HF000+I,VAL("&H"+MC$)
140 NEXT I
150 DATA CD,41,00: 'CALL DISSCR
160 DATA C9:       'RET
170 DATA CD,44,00: 'CALL ENASCR
```

```
180 DATA C9:          'RET
190 FOR I=0 TO 23
200 PRINT "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
210 NEXT I
220 DEFUSR0=&HF001: 'DISSCR
230 DEFUSR1=&HF005: 'ENASCR
240 DUMMY=USR0(0): 'Disable screen
250 FOR I=0 TO 200: NEXT I.
260 DUMMY=USR1(0): 'Enable screen
270 FOR I=0 TO 400: NEXT I
280 GOTO 240
```

Met regel 110 wordt een stuk geheugen vrijgemaakt (vanaf adres &HF000) voor het opslaan van een eigen machinetaalroutine. Met de regels 120 tot en met 140 worden de machinetaalinstructies, die in de dataregels 150 tot en met 180 staan, in het vrijgemaakte geheugendeel geladen. De machinetaalinstructie op regel 150 roept het BIOS entry point &H0041 aan. Dit resulteert in het leegmaken van het beeldscherm. Op regel 160 staat de RET-instructie, waarmee wordt teruggekeerd naar het BASIC-programma. Regels 170 en 180 spreken nu voor zichzelf.

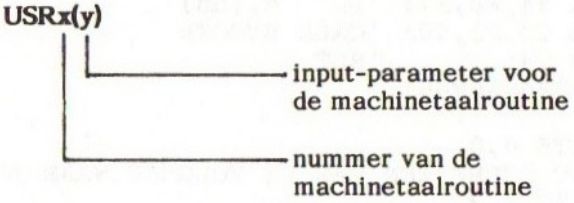
In regel 220 en 230 ziet u nu dat het begin van de routines niet meer het BIOS entry point is, maar het betreffende startadres van ons eigen machinetaalroutinetje. Voor het overige is er geen verschil tussen dit programma en het vorige.

Het startadres van iedere machinetaalroutine, die vanuit BASIC moet worden aangeroepen, moet met behulp van het **statement DEFUSR** worden vastgelegd. Het algemene formaat van DEFUSR is:



Er kunnen maximaal 10 verschillende machinetaalroutines tegelijkertijd worden vastgelegd. Iedere machinetaalroutine krijgt een uniek nummer, dat kan varieren van 0 tot en met 9. Dit nummer wordt in het statement aangegeven op de plaats van de letter x. Indien op die plaats geen nummer wordt geplaatst, dan zal MSX-BASIC aannemen dat nummer 0 wordt bedoeld.

Met de **functie USR0(0)** wordt, zoals uit het voorbeeldprogramma blijkt, een machinetaalroutine aangeroepen. Het algemene formaat van de functie USR is:



Ook hier geldt weer, dat op de plaats van de x het nummer van de aan te roepen machinetaalroutine wordt gezet. Tussen de haakjes kan een parameter, die moet worden doorgegeven aan het machinetaalprogramma, worden geplaatst. Daar we in de gegeven voorbeeldprogramma's geen gegevens wilden doorgeven, is er de waarde 0 ingevuld. Laten we nu echter eens gaan kijken wat er gebeuren moet, indien we wel een parameter aan de aan te roepen machinetaalroutine willen doorgeven.

10.2 Parameters van BASIC naar machinetaal.

We nemen als voorbeeld een programma, waarmee we het BIOS entry point STMOTR aanroepen. Entry point STMOTR zal voor ons, zonodig de motor van de cassetterecorder aan of uitschakelen. Deze machinetaalroutine (STMOTR) schakelt de motor aan, indien de inhoud van register A bij het aanroepen van de routine 1 is. Is de inhoud van register A bij het aanroepen van de routine 0, dan zal de cassettemotor worden

uitgeschakeld. Door in register A de waarde 255 (decimaal = FF hexadecimaal) te zetten, zal de cassetterecordermotor worden aangeschakeld, indien deze uit stond, en worden uitgeschakeld, indien deze aan stond. Het BIOS entry point van STMOTR staat op adres &H00F3.

```
100 SCREEN 0: WIDTH 40: KEY OFF
110 CLEAR 200,&HF000
120 FOR I=1 TO 7
130 READ MC$: POKE &HF000+I,VAL("&H"+MC$)
140 NEXT I
150 DATA 3A,F8,F7: 'LD   A,(nn)
160 DATA CD,F3,00: 'CALL STMOTR
170 DATA C9:       'RET
180 DEFUSR0=&HF001
190 CLS
200 LOCATE 0,0
210 PRINT "GEEF EEN VAN DE VOLGEND WAARDEN IN:";
220 LOCATE 3,3
230 PRINT "0     om de motor te STOPPEN";
240 LOCATE 3,4
250 PRINT "1     om de motor te STARTEN";
260 LOCATE 3,5
270 PRINT "255   om de motor om te schakelen";
280 LOCATE 0,8: INPUT "WAARDE";A%
290 DUMMY=USR0(A%)
300 GOTO 190
```

Regel 110:
Maakt het geheugengebied voor onze eigen machinetaalroutine vrij.

Regels 120 t/m 140:
Schrijven onze machinetaalroutine (vanuit de DATA-statements) naar het vrijgemaakte geheugengebied.

Regels 150 t/m 170:
De machinetaalroutine. De eerste instructie leest wat de inhoud van adres &HF7F8 is en zet die inhoud in de accumulator (register A). Adres &HF7F8 wijst naar het minst signifi-

cante byte van een twee bytes integer. Dit adres geldt altijd, wanneer er vanuit BASIC een integer variabele (in dit programma A%) als parameter wordt doorgegeven aan het machinetaalprogramma. Regel 160 roept de ROM-routine (STMOTR) aan. Deze routine kijkt in register A wat er moet worden gedaan. Na uitvoering van de ROM-routine wordt teruggekeerd in onze machinetaalroutine, waar de instructie RET wordt uitgevoerd. RET zorgt er voor dat we weer terugkeren vanuit machinetaal naar BASIC.

Regel 180:
Definieert het startadres van de machinetaalroutine (&HF001).

Regels 190 t/m 280:
Stellen ons in staat om de integer-variabele A% te vullen met een waarde.

Regel 290:
Start de machinetaalroutine en geeft de variabele A% mee als input-parameter voor de machinetaalroutine.

10.3 Parameters van machinetaal naar BASIC.

Indien een machinetaalroutine een output-parameter oplevert, die moet worden doorgegeven naar het aanroepende BASIC-programma, dan wordt daarvoor, net als voor de omgekeerde richting, een vaste procedure aangehouden. Ten eerste dient de programmeur te weten wat voor soort parameter er gaat worden teruggegeven; integer, enkele nauwkeurigheid, dubbele nauwkeurigheid of string. Afhankelijk van de soort parameter wordt dan in een LET-statement een overeenkomstig type variabele gedefinieerd, bijvoorbeeld:

```
LET RI%=USR0(RN%)
```

De variabele RN% bevat een integer waarde die aan de machinetaalroutine wordt meegegeven. Na uitvoering van de machinetaalroutine wil deze een integer waarde teruggeven

aan het aanroepende BASIC-programma. Die waarde wordt dan in het BASIC-programma opgevangen in de variabele RI%. Dit is in het volgende programma toegepast.

```
100 SCREEN 0: WIDTH 40: KEY OFF
110 CLEAR 200,&HF000
120 FOR I=1 TO 15
130 READ MC$: POKE &HF000+I,VAL("&H"+MC$)
140 NEXT I
150 DATA 3A,F8,F7: 'LD   A,(nn)
160 DATA CD,96,00: 'CALL RDPSG
170 DATA 32,F8,F7: 'LD   (nn),A
180 DATA 3E,02:    'LD   A,&H02
190 DATA 32,63,F6: 'LD   (nn),A
200 DATA C9:       'RET
210 DEFUSR0=&HF001
220 CLS
230 LOCATE 7,0: PRINT "LADEN PSG-REGISTER";
240 LOCATE 7,2: INPUT "REGISTERNUMMER";RN%
250 LOCATE 7,3: INPUT "REGISTERINHOUD";RI%
260 SOUND RN%,RI%
270 FOR I=0 TO 13
280 RN%=I
290 LOCATE 0,5+I
300 PRINT "De inhoud van register";I;"is";
310 RI%=USR0(RN%):PRINT RI%;
320 NEXT I
330 LOCATE 0,23
340 PRINT "Druk op RETURN om door te gaan";
350 I$=INKEY$: IF I$<>CHR$(13) GOTO 350
360 GOTO 220
```

Dit programma vraagt eerst, welk register van de programmeerbare geluidsgenerator (PSG) moet worden geladen met een door u in te geven waarde. Vervolgens worden alle registers van de PSG uitgelezen en op het beeldscherm afgedrukt. De programmeerbare geluidsgenerator wordt in het volgende hoofdstuk in detail behandeld. Na bestudering van dat hoofdstuk zult u ontdekken dat dit kleine programma toch een

handig utility is voor het componeren van de meest vreemde geluiden.

Regel 150:
Haalt de parameter die vanuit het BASIC-programma wordt doorgegeven op, en plaatst deze in register A van de microprocessor.

Regel 160:
Roept het BIOS entry point RDPSG aan. Hierdoor wordt een machinetaalroutine uit de ROM gestart, die het registernummer van de PSG, dat in register A staat aangegeven (doorgegeven vanuit BASIC), uitleest. De inhoud van dat PSG-register wordt in register A van de microprocessor gezet.

Regel 170:
Schrijft de inhoud van register A naar adres &HF7F8 (de integer-output parameter).

Regel 180 en 190:
Zorgen ervoor dat op adres &HF663 de waarde &H02 komt te staan. Dit is voor het BASIC-programma een aanduiding dat het machinetaalprogramma een integer-variabele teruggeeft. Zou er op adres &HF663 een waarde 3 worden gezet, dan betekent dit dat er een string wordt teruggegeven. Een 4 betekent een variabele met enkelvoudige nauwkeurigheid en een 8 wil zeggen, een variabele met dubbele nauwkeurigheid. De variabele, waarin de betreffende parameter moet worden opgevangen, moet van het juiste type zijn. Straks zullen we dieper ingaan op de verschillende soorten parameters, de mogelijkheden en de moeilijkheden.

Regel 200:
Keert terug naar het BASIC-programma.

Regel 210:
Definieert het startadres van onze machinetaalroutine.

Regel 260:
Hiermee wordt registernummer RN% geladen met de inhoud

van variabele RI%. Het SOUND-statement zal in het volgende hoofdstuk uitgebreid worden behandeld.

Regels 270 t/m 320:
De variabele RN% wordt 14 keer geladen met steeds een andere waarde (0 tot en met 13). Vervolgens wordt de machinetaalroutine gestart (regel 310), waarmee de inhoud van het in RN% aangegeven register van de programmeerbare geluidsgenerator in variabele RI% terecht komt. Zowel het registernummer van de geluidsgenerator als de inhoud daarvan, worden op het beeldscherm afgedrukt.

Regels 330 t/m 360:
Stellen ons in staat om nogmaals een register van de geluidsgenerator met een bepaalde waarde te laden, om vervolgens de nieuwe inhoud van alle registers te bekijken.

10.4 Parameter soorten.

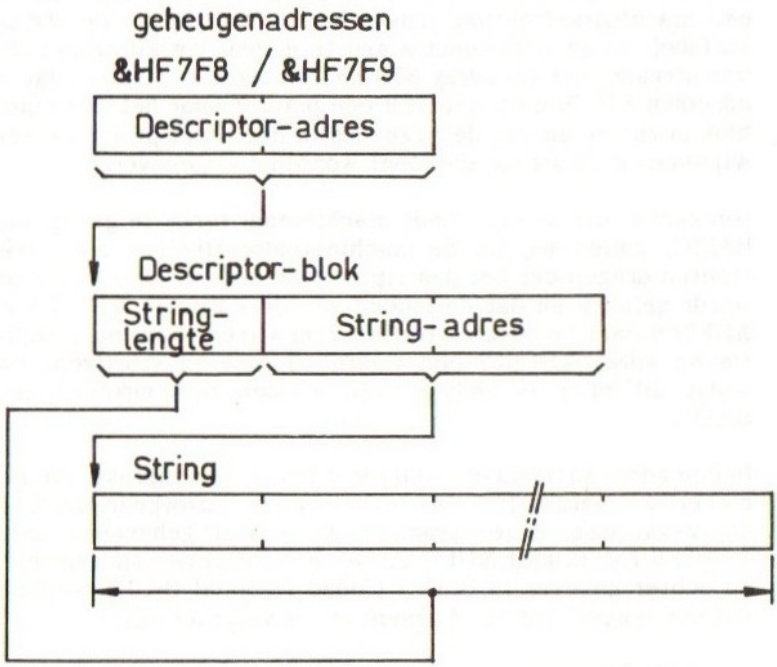
De parameters, die kunnen worden doorgegeven aan, of teruggontvangen van een machinetaalroutine, kunnen van een van de volgende soorten zijn:

- integer
- enkelvoudige nauwkeurigheid
- dubbele nauwkeurigheid
- string

Met welke soort parameter we te maken hebben, staat aangegeven op adres &HF663. Merk op, dat wij, bij het teruggeven van een parameter vanuit een machinetaalroutine naar BASIC, zelf verantwoordelijk zijn voor de inhoud van adres &HF663. Dit adres kan een van de volgende inhouden hebben:

2 = integer
3 = string
4 = enkelvoudige nauwkeurigheid
8 = dubbele nauwkeurigheid

De variabelen, waarin wij de parameters willen opvangen, dienen van het juiste type te zijn. Alle tot nu toe gegeven voorbeelden lieten het doorgeven van integer-parameters zien. Hierover hoeft dan ook geen uitleg meer te worden gegeven. We volstaan hier met vast te stellen, dat daarbij adres &HF663 de inhoud 2 heeft en dat de integer-parameter zelf op de adressen &HF7F8 en &HF7F9 staat. &HF7F8 bevat het minst significante byte van de integer en &HF7F9 het meest significante byte.



Afb. 10-2 Doorgeven van strings tussen BASIC en machinetaal.

Indien adres &HF663 de waarde 3 bevat, dan hebben we te maken met een string-parameter, een aantal tekens dus. In dat geval bevatten de adressen &HF7F8 en &HF7F9 een verwijzing naar een "descriptor block". In dat "descriptor block", dat uit drie bytes bestaat, vinden we als eerste een

byte waarin de lengte van de tekst (het aantal tekens) staat. De volgende twee bytes van het descriptor block bevatten het adres van het eerste teken van de string. Zowel de inhoud van adres &HF663, als van de adressen &HF7F8 en &HF7F9 en de inhoud en de plaats van het descriptor block en de string zelf, zijn bij het teruggeven van een string-parameter onze eigen verantwoordelijkheid. Afbeelding 10-2 geeft van het voorgaande verhaal een grafische weergave.

Wanneer we, vanuit BASIC een string willen doorgeven aan een machinetaalroutine, dan is het voldoende om de stringvariabele in de USR-functie aan te geven. We kunnen er dan van uitgaan, dat op adres &HF663 de waarde 3 staat, dat de adressen &HF7F8 en &HF7F9 een pointer naar het descriptor blok bevatten en dat de gegevens in het descriptor blok verwijzen naar de string, die moet worden doorgegeven.

Om echter een string vanuit machinetaal terug te geven naar BASIC, zullen wij (in de machinetaalroutine) er zorg voor moeten dragen dat het descriptor blok met de juiste waarden wordt geladen en dat de inhoud van de adressen &HF7F8 en &HF7F9 naar het descriptor blok verwijzen. Tenslotte zullen we op adres &HF663 een waarde 3 moeten schrijven. Pas nadat dit alles is gedaan, kan worden teruggekeerd naar BASIC.

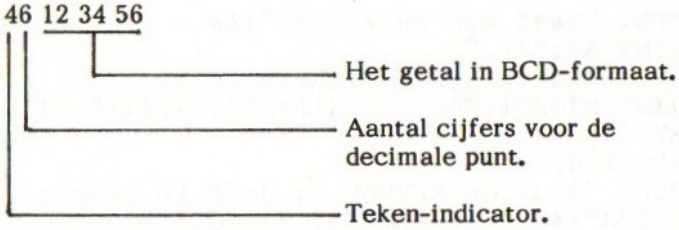
Indien adres &HF663 een waarde 4 bevat, is er sprake van een numerieke parameter met enkelvoudige nauwkeurigheid. In dat geval staat deze parameter in de vier geheugenlocaties &HF7F6 tot en met &HF7F9. De door te geven parameter is nu echter gegeven in Binary Coded Decimal (BCD) formaat. Dit wil zeggen, dat het formaat er als volgt uitziet:

&HF7F6 = exponent
&HF7F7 = twee digits (meest significant)
&HF7F8 = twee digits
&HF7F9 = twee digits (minst significant)

Een voorbeeldje zal waarschijnlijk duidelijker zijn dan een hoop woorden. Stel dat in een BASIC-programma de volgende statements voorkomen:

```
A!=123456
X=USR0(A!)
```

In dat geval zal de inhoud van adres &HF663 een 4 zijn. We weten dan, dat we te maken hebben met een parameter met enkelvoudige nauwkeurigheid en dat die parameter te vinden is op de adressen &HF7F6 tot en met &HF7F9. Als we die adressen bekijken, dan vinden we daar de volgende inhoud:



Een programma waarmee we het formaat van een parameter met enkelvoudige nauwkeurigheid kunnen onderzoeken is als volgt:

```
100 SCREEN 0: WIDTH 40: KEY OFF
110 CLS
120 INPUT "Geef een getal in:";A!
130 PRINT A!;"is ";
140 FOR I=VARPTR(A!) TO VARPTR(A!)+3
150 PRINT RIGHT$("00"+HEX$(PEEK(I)),2);" ";
160 NEXT I
170 LOCATE 0,20
180 PRINT "druk op RETURN om door te gaan";
190 I$=INKEY$: IF I$<>CHR$(13) GOTO 190
200 GOTO 110
```

Indien adres &HF663 de waarde 8 bevat, dan hebben we te maken met een parameter met dubbele nauwkeurigheid. Deze parameter bezet 8 adressen, en wel de adressen &HF7F6 tot en met &HF7FD. Ook deze parameter bestaat uit een exponent gevolgd door een aantal digits. De exponent staat weer op adres &HF7F6. De daarop volgende 7 adressen bevatten

elk twee digits, veertien digits in totaal dus. Voor deze soort parameters gelden verder dezelfde regels als voor de parameters met enkelvoudige nauwkeurigheid. Ook voor het onderzoeken van parameters met dubbele nauwkeurigheid volgt hier een programma.

```
100 SCREEN 0: WIDTH 40: KEY OFF
110 CLS
120 INPUT "Geef een getal in:";A#
130 PRINT A#;"is ";
140 FOR I=VARPTR(A#) TO VARPTR(A#)+7
150 PRINT RIGHT$("00"+HEX$(PEEK(I)),2);" ";
160 NEXT I
170 LOCATE 0,20
180 PRINT "druk op RETURN om door te gaan";
190 I$=INKEY$: IF I$<>CHR$(13) GOTO 190
200 GOTO 110
```

In beide laatste programma's werd een functie gebruikt, die nog niet was behandeld; de **functie VARPTR(variabele).** Met behulp van deze functie kunnen we het geheugenadres, van de variabele die achter de functie tussen haakjes is aangegeven, verkrijgen. In de voorbeeldprogramma's is de functie VARPTR toegepast in een FOR...NEXT-statement. Bekijken we wat er gebeurt in regelnummer 140 van het eerste voorbeeldprogramma, dan zien we het volgende.

Ergens in het geheugen staat de variabele A!. Door nu de naam van die variabele achter de functie VARPTR (variabelepointer) te plaatsen, levert de functie het geheugenadres op, waar de inhoud van de variabele is weggeschreven. Regelnummer 140 kan dus ook worden gelezen als:

140 FOR I = **adres van A!** TO **adres van A! + 3**

Probeer, met behulp van de voorbeeldprogramma's, een aantal verschillende waarden van zowel variabelen met enkelvoudige als met dubbele nauwkeurigheid uit. Pas nadat u in staat bent de inhoud van deze variabelen correct te bepalen, kunt u

proberen om een dergelijke parameter door te geven aan een machinetaalprogramma of vanuit een machinetaalprogramma terug te geven naar BASIC.

10.5 Het doorgeven van meerdere parameters.

Tot nu toe hebben we de officiele procedures gevolgd, waarbij steeds slechts 1 parameter tegelijkertijd kon worden doorgegeven. Er kunnen zich echter situaties voordoen, waarbij we meerdere parameters tegelijk willen doorgeven. Een dergelijke situatie doet zich onder meer voor, wanneer we gebruik willen maken van de machinetaalinstructie LDIR. Deze instructie verwacht in registerpaar HL het adres van een byte dat moet worden gecopieerd naar het byte waarvan het adres in registerpaar DE staat. Bovendien moet registerpaar BC het aantal bytes dat moet worden gecopieerd bevatten. Er moeten dus drie parameters bekend zijn. Als we deze parameters in het BASIC-programma met behulp van een INPUT-instructie opvragen, dan zullen we, voordat we de LDIR-instructie kunnen starten, drie parameters moeten doorgeven aan de machinetaalroutine.

Het volgende programma laat daarvan een voorbeeld zien.
Het programma maakt een copie van een door u op te geven geheugendeel naar het vrijgemaakte geheugendeel. Is de copie eenmaal gemaakt, en dit gaat razendsnel, dan wordt zowel de copie als het orgineel in binaire vorm afgedrukt. Straks zullen we zien wat hiervan de reden is.

```
100 SCREEN 0: WIDTH 40: KEY OFF: CLS
110 CLEAR 200,&HD000
120 FOR I=1 TO 14
130 READ MC$: POKE(&HD000+I),VAL("&H"+MC$)
140 NEXT I
150 DATA 2A,01,D1:'     LD   HL,(NN)
160 DATA ED,5B,03,D1:' LD   DE,(NN)
170 DATA ED,4B,05,D1:' LD   BC,(NN)
180 DATA ED,B0:'       LDIR
190 DATA C9:'          RET
```

```
200 DEFUSR0=&HD001
210 'OPVRAGEN PARAMETERS
220 LOCATE 0,0
230 PRINT "COPIEREN NAAR VRIJGEMAAKT RAM"
240 LOCATE 5,3
250 INPUT "VANAF ADRES";S!
260 LOCATE 5,5
270 INPUT "BLOK-LENGTE";L!
280 'WEGSCHRIJVEN PARAMETERS
290 POKE &HD101,S!MOD256
300 POKE &HD102,S!\256
310 POKE &HD103,1
320 POKE &HD104,&HD2
330 POKE &HD105,L!MOD256
340 POKE &HD106,L!\256
350 'START COPIEREN
360 DUMMY=USR0(0)
370 'CONTROLEER RESULTAAT
380 FOR I=0 TO L!
390 PRINT RIGHT$("0000"+HEX$(S!+I),4);" ";
400 PRINT RIGHT$("00"+HEX$(PEEK(S!+I)),2);
" ";
410 PRINT RIGHT$("00000000"+BIN$(PEEK(S!+I
)),8);"    ";
420 PRINT RIGHT$("0000"+HEX$(&HD201+I),4);
" ";
430 PRINT RIGHT$("00"+HEX$(PEEK(&HD201+I))
,2);" ";
440 PRINT RIGHT$("00000000"+BIN$(PEEK(&HD2
01+I)),8)
450 NEXT I
460 END
```

Regel 110
Maakt het geheugen vanaf adres &HD000 en hoger vrij voor gebruik door machinetaalroutines.

regels 120 - 140
Laden de machinetaalroutine uit de regels 150 tot en met 190

in het vrijgemaakte geheugendeel.

Regels 150 - 190
De machinetaalroutine. De eerste instructie zet de inhoud van geheugenlocaties &HD101 en &HD102 in registerpaar HL. De tweede instructie zet de inhoud van adressen &HD103 en &HD104 in registerpaar DE. De derde instructie zet de inhoud van geheugenlocaties &HD105 en &HD106 in registerpaar BC. Nu zijn alle registers met de juiste waarden geladen en kan de LDIR-instructie worden uitgevoerd. Deze instructie maakt de copie. Na uitvoering kan worden teruggekeerd naar BASIC, hetgeen met de RET-instructie wordt bewerkstelligd.

Regel 200
Definieert het startadres van de machinetaalroutine (&HD001).

Regels 210 - 270
Vragen de parameterwaarden op en zetten de ingegeven waarden in variabelen.

Regels 290 - 340
Schrijven de ingegeven waarden naar adressen in het vrijgemaakte geheugendeel, met behulp van POKE-instructies. De ingegeven waarden moeten worden omgerekend naar inhouden van enkele bytes, vandaar dat er in de POKE-statements numerieke uitdrukkingen staan, in plaats van constanten.

Regel 360
Start de machinetaalroutine (zie de beschrijving bij regelnummers 150 - 190).

Regels 380 - 450
Lezen de geheugenadressen waar vandaan werd gecopieerd en de adressen waar naartoe werd gecopieerd, en zetten beide inhouden naast elkaar op het beeldscherm. De inhoud van ieder adres wordt geconverteerd naar een serie van nulletjes en eentjes. Dit geeft u de mogelijkheid om te controleren of de copie correct is gemaakt. Bovendien geeft het u de mogelijkheid om te zien of er in de gecopieerde bytes een bepaald

patroon is te herkennen. Waar dit goed voor is zullen we hierna zien.

De tekenset is voor MSX-computers gestandaardiseerd. Dit houdt in, dat iedere MSX-computer dezelfde letters, cijfers en leestekens heeft. Dit is slechts ten dele waar, want er zijn een aantal MSX-versies gedefinieerd. Zo zullen MSX-computers voor de Japanse markt een "katakana"-tekenset hebben. We mogen er echter van uitgaan, dat alle officieel in Nederland geimporteerde MSX-computers dezelfde tekenset hebben.

Hoewel alle MSX-computers dezelfde tekenset hebben, is de plaats van die tekenset niet vastgelegd. De tekenset staat weliswaar in het ROM, doch waar precies, dat is fabrikantafhankelijk. Er is echter een eenvoudige manier om daar achter te komen. Hiervoor dienen we wel te weten hoe de tekenset in de ROM is opgeslagen.

Ieder teken, of dat nu een letter, een cijfer of een leesteken is, bestaat uit een matrix van 8 bij 8 puntjes. In een byte passen precies 8 puntjes, die ieder 1 of 0 kunnen zijn, hetgeen staat voor respectievelijk aan of uit, zichtbaar of niet zichtbaar, voorgrond of achtergrond. In de tekenset worden de tekens in acht horizontale rijen van ieder 8 bitjes bewaard. Afbeelding 10-3 laat een matrix zien, met daarin een letter. Op die manier zijn alle tekens opgeslagen.

Van alle afdrukbare tekens is het lachende poppetje het tweede dat in de tabel voorkomt. Vlak daarvoor, op de eerste plaats staat een teken, dat geheel uit nullen bestaat, het zogenaamde "null"-teken. Als we nu een programma schrijven, dat vanaf adres 0 in de ROM zoekt naar het voorkomen van de eerste twee tekens uit de tekenset, dan kunnen we het adres waarop de vergelijking opgaat onthouden, en weten we waar de tekenset in onze computer start. Het volgende programma doet dit voor ons.

```
100 'startadres tekenset-tabel zoeken
110 DATA 0,0,0,0,0,0,0,0:'null string
```

```
120 DATA 60,66,165,129,165,153,66,60:'poppetje
130 DEFINT A-Z
140 FOR I=0 TO 32767
150 D=PEEK(I)
160 READ K
170 IF D=K THEN Z=Z+1 ELSE Z=0:RESTORE:NEXT I
180 IF Z<16 THEN NEXT I
190 PRINT "tekenset-tabel start op:"
200 PRINT "hexadecimaal adres ";HEX$(I-15)
210 PRINT "decimaal adres     ";I-15
220 END
```

```
Voorbeeld: Na ruim 1 minuut geeft de
           Goldstar FC-200 dit resultaat:

tekenset-tabel start op:
hexadecimaal adres 1BBF
decimaal adres     7103
```

Met het verkregen resultaat kunnen we het voorlaatste programma opnieuw opstarten. Nu kunnen we echter als startadres voor het copieren het startadres van de tekenset opgeven. Op die manier zijn we in staat om de complete tekenset van ROM te copieren naar RAM. Staat de tekenset eenmaal in RAM dan kunnen we de orginele tekens naar eigen smaak wijzigen, of vervangen door andere tekens. Vervolgens is het mogelijk om deze gewijzigde tekenset naar het video-RAM te copieren. In het video-RAM moet de nieuwe tekenset dan naar de patronengeneratortabel worden gecopieerd.

Het volgende programma laat een voorbeeldje zien van deze mogelijkheden. Verdere uitleg gaat buiten het bestek van dit boek, doch de oplettende lezer zal nu zelf in staat zijn verder te experimenteren.

```
150 KEY OFF
160 SCREEN 1: WIDTH 32: CLS
170 LOCATE 5,15
```

```
180 PRINT "De a zal veranderen"
185 FOR I=0 TO 1000:NEXT I
190 '
200 '****************************
210 '*  laden van de teken-set. *
220 '****************************
225 '
240 RESTORE
250 FOR I=0 TO 7
260 READ BR
270 VPOKE BASE(7)+I+97*8,BR
280 NEXT I
282 '
283 '****************************
284 '* afdruk met nieuwe tekens *
285 '****************************
286 '
290 LOCATE 5,15
300 PRINT "De a zal veranderen"
305 GOTO 305
310 '
320 '****************************
330 '* definitie van de tekens  *
340 '****************************
350 '
360 DATA 0,124,6,126,198,198,126,0:'a
```

10.6 Systeemlocaties.

De routines in de ROM werken vaak met variabele gegevens. In het ROM kan echter niets worden geschreven. Er kunnen dus geen variabele gegevens in het ROM worden gezet. Daarom hebben de ROM-routines de beschikking gekregen over een geheugengebied in het RAM. Voor de basis MSX-computers, zonder schijveneenheden, loopt dit RAM-gebied vanaf adres &HF380 tot en met &HFFFF.

Nemen we als voorbeeld de ROM-routine die de cursor op het

beeldscherm moet schrijven, dan is het duidelijk dat die routine moet weten, waar de cursor op het scherm moet worden geschreven. Die plaats van de cursor is echter niet vast, dus moet er ergens in het RAM een adres zijn, waarin staat wat de horizontale en wat de verticale positie van de cursor moet zijn.

In onze machinetaalroutines kunnen we natuurlijk ook gebruik maken van deze adressen. We kunnen de systeemlocaties gewoon met behulp van POKE-statements vullen met waarden. Zolang u echter niet precies weet waarvoor een bepaalde locatie dient, kunt u er beter geen waarden in zetten. Zet u er namelijk een verkeerde waarde in, dan kan de computer in een "hang-up"-situatie terecht komen. Hier komen we alleen weer uit door de computer te re-setten (uit- en weer aanschakelen). Het gevolg hiervan is dat de gehele geheugeninhoud verloren gaat.

Het aantal systeemlocaties is zo groot, en vraagt zoveel uitleg, dat het niet doenlijk is om daaraan in dit boek verder aandacht te besteden. Wilt u er meer van weten, dan zult u speciale (machinetaal) boeken moeten raadplegen.

10.7 "Hook"-adressen.

Eveneens in het systeemgebied van het RAM, zijn een aantal zogenaamde "hook"-adressen gedefinieerd. De functie van deze adressen is als volgt.

De routines in de ROM zijn onveranderlijk. Het is dus niet mogelijk om de routines uit te breiden of te wijzigen. Dit betekent dat de computer niet erg flexibel is. Om toch voldoende flexibiliteit te verkrijgen, wordt aan het begin van een groot aantal ROM-routines een sprong gemaakt naar het systeem-RAM. Normaal, na het aanschakelen van de computer, staat op de plaats waar in het RAM naartoe wordt gesprongen een machinetaal-instructie RET. Hierdoor zal onmiddellijk worden teruggesprongen naar de ROM-routine. Wij hebben echter de mogelijkheid om in het RAM, in plaats

van de RET-instructie, een spronginstructie te plaatsen, waarmee naar een eigen machinetaalroutine wordt gesprongen. Na uitvoering van deze routine wordt dan alsnog teruggekeerd naar de oorspronkelijke ROM-routine.

Ook het gebruik van "hook"-adressen is, bij onvoldoende kennis van het systeem, niet aan te raden. Ook hierbij bestaat het gevaar van een "hang-up" situatie. Het volgende voorbeeld laat echter zien hoe een en ander in zijn werk gaat. Controleert u goed of u het programma foutloos hebt ingetikt, en schrijft u het programma eerst naar cassette, voordat u het gaat uitvoeren.

```
100 SCREEN 0:WIDTH 40:CLS
110 CLEAR 200,&HC000
120 DEFUSR0=&HC001
130 DEFUSR1=&HC012
140 FOR I=0 TO 29
150 READ MC: POKE &HC001+I,MC
160 NEXT I
170 DUMMY=USR1(0)
180 PRINT ".";
200 GOTO 180
210 DATA &H3E,&H06:      'LD   A,rij
220 DATA &HCD,&H41,&H01:'CALL SNSMAT
230 DATA &HFE,&HE9:      'CP   233
240 DATA &HC0:           'RET  NZ
250 DATA &H3E,&H2A:      'LD   A,*
260 DATA &HCD,&HA2,&H00:'CALL CHPUT
270 DATA &HC9:           'RET
280 DATA &HCD,&H01,&HC0:'data
290 DATA &H21,&H0F,&HC0:'LD   HL,data
300 DATA &H11,&H9F,&HFD:'LD   DE,FD9A
310 DATA &HED,&HA0:      'LDI
320 DATA &HED,&HA0:      'LDI
330 DATA &HED,&HA0:      'LDI
340 DATA &HC9:           'RET
```

Regel 280
Bevat een CALL instructie, waarmee de machinetaalroutine die start op adres &HC001 kan worden aangeroepen. Deze instructie wordt naar het "hook"-adres gecopieerd.

Regels 290 - 340
Dit is een machinetaalroutine, waarmee de instructie van regel 280 naar het "hook"-adres wordt gecopieerd. Het betreffende "hook"-adres is adres &HFD9A. Dit adres wordt aangeroepen door de ROM-routine, die 50 keer per seconde wordt uitgevoerd, om te zien of er een onderbreking van het programma is gewenst.

Regels 210 - 270
Dit is een machinetaalroutine, die wordt aangeroepen door de CALL-instructie die we met regels 290 - 340 op het "hook"-adres &HFD9A hebben gezet. Deze routine kijkt of er een bepaalde combinatie van toetsen tegelijkertijd is ingedrukt. Deze combinatie is CNTRL, GRAPH en CODE. Alleen wanneer deze drie toetsen alle drie tegelijkertijd zijn ingedrukt, worden de instructies van regels 250,260 en 270 uitgevoerd. In alle andere gevallen wordt onmiddellijk teruggekeerd naar de ROM-routine.

De uiteindelijke werking van het programma is, dat er continue punten op het beeldscherm worden afgedrukt (regels 180 en 200). Het indrukken van een toets resulteert in het starten van een ROM-routine. De ROM-routine springt naar het "hook"-adres. Daar staat een sprong-instructie naar een eigen machinetaalroutine. Die routine kijkt welke toets(en) is ingedrukt. Is de combinatie CNTRL, GRAPH en CODE ingedrukt, dan wordt een "*" afgedrukt. Daarna worden weer punten afgedrukt. De snelheid van het programma maakt het vrijwel onmogelijk om slechts 1 asterisk af te drukken, meestal zullen dat er veel meer zijn.

Wat we met dit programma hebben aangetoond is, dat het mogelijk is om meerdere programma's in het geheugen van de MSX-computer te zetten en die programma's met een druk op een knop te laten starten, ook al zijn we midden in een ander programma bezig.

11 De programmeerbare geluidsgenerator

In hoofdstuk 7 hebben we gezien hoe we met behulp van het PLAY-statement muziek kunnen maken. Het PLAY-statement is speciaal ontworpen om het ons gemakkelijk te maken. Er wordt echter niet van alle mogelijkheden van de geluidsprocessor gebruik gemaakt. Willen we alle mogelijkheden benutten, dan zullen we gebruik moeten maken van het SOUND-statement. Om het SOUND-statement goed te kunnen benutten, dienen we echter iets meer van de geluidsprocessor te weten.

We zullen daarom eerst het algemene formaat van het SOUND-statement bekijken, om daarna de toepassing van dat statement voor het laden van de verschillende registers aan een nader onderzoek te onderwerpen. Het formaat is:

SOUND register , inhoud

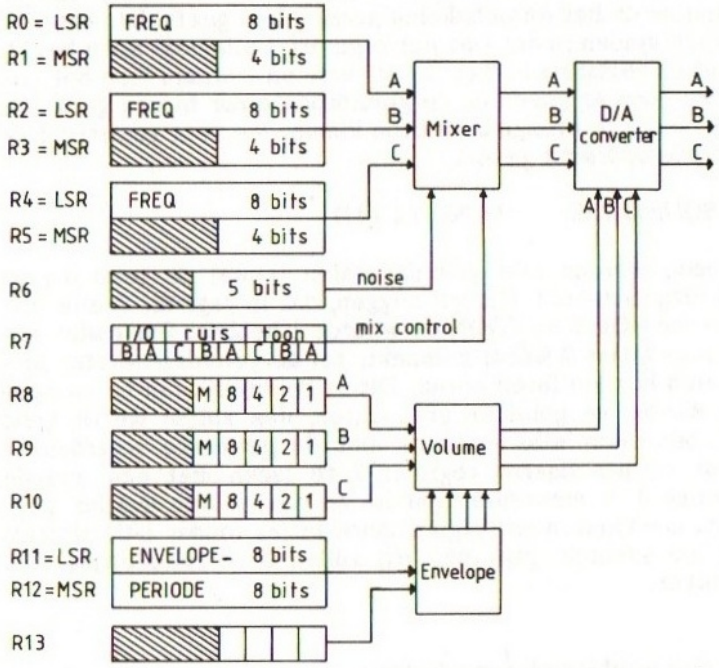
Het met dit statement aangegeven **register** zal worden geladen met de achter de komma opgegeven **inhoud.** Afhankelijk van het register is die inhoud aan beperkingen onderhevig. De toegestane waarden zullen in de loop van dit hoofdstuk verder worden toegelicht.

11.1 De logische opbouw van de geluidsprocessor.

Afbeelding 11-1 geeft een blokschema van de interne opbouw van de geluidsprocessor. Links in de afbeelding ziet u de registers. In totaal zijn er 14 registers, R0 tot en met R13.

Registers 0 en 1 bevatten een waarde, die de frequentie van

de toon voor kanaal A bepaalt. Registers 2 en 3 bepalen de frequentie voor kanaal B en registers 4 en 5 bepalen de frequentie voor kanaal C. Register 6 bepaalt de frequentie van de ruis (noise). De tonen van de drie kanalen A, B en C worden in de mixer gemengd met het ruissignaal. Bovendien kan de mixer onder besturing van de inhoud van register 7 de tonen al of niet doorlaten en al of niet mengen met het ruissignaal. De mixer bepaalt dus welke tonen er uiteindelijk zullen worden geproduceerd.



Afb. 11-1 Blokschema van de programmeerbare geluidsgenerator.

De D/A-converter zet de uit de mixer ontvangen digitale signalen om naar analoge signalen, die (na verdere versterking) door middel van een luidspreker hoorbaar kunnen worden gemaakt. De sterkte van de analoge signalen kan worden

beinvloed door de registers 8, 9 en 10, voor respectievelijk de kanalen A, B en C. Met deze registers kan het volume van het geluid dus worden geregeld. Is de inhoud van een van deze registers gelijk aan de waarde 16, dan zal het geproduceerde geluid worden gemoduleerd op een manier en met een fre- (envelope-vorm) en registers 11 en 12 (voor de periodetijd van de envelope).

In de MSX-standaard is de inhoud van de geluidsregisters, na het aanschakelen, niet gedefinieerd. Wel is zeker, dat de computer na het aanschakelen geen geluid geeft. Dit kan zijn oorzaak vinden in het feit dat register 7 allemaal enen bevat, of dat de registers 8, 9 en 10 alledrie op 0 staan. Hoe het ook zij, we doen er goed aan de geluidsprocessor in een gedefinieerde stand te brengen. Hiertoe kunnen we voor register 7 de volgende opdracht geven:

```
SOUND 7,63      (= 00 111 111)
```

Hiermee worden alle geluidskanalen (zowel de toon als de ruis) uitgeschakeld. Dit wil zeggen, dat in register 7 alle bitjes onder NOISE en TOON de waarde 1 hebben. Pas nadat een van deze bitjes 0 wordt gemaakt, zal de geluidsgenerator iets van zich kunnen laten horen. Dit is belangrijk om te onthouden. Willen we geluiden produceren, dan zullen we er goed aan doen eerst alle registers met de gewenste waarden te vullen, en pas daarna register 7 te laden met een waarde waarmee 1 of meerdere geluids- en ruiskanalen worden aangezet. De twee meest significante bitjes (onder I/O) worden door het systeem gebruikt. Wij zullen deze twee bitjes niet gebruiken.

11.2 Het produceren van een toon.

Om een toon te produceren, zullen we de volgende acties dienen te ondernemen:

- Bepaal de toonhoogte.
- Bepaal het volume.
- Zet het geluidskanaal aan.

Een toon heeft een bepaalde frequentie. Een lage bromtoon heeft bijvoorbeeld een frequentie van 50 herz, terwijl een hoge pieptoon een frequentie heeft van 10000 herz. De frequentie (voor kanaal A) kunnen we instellen, door registerpaar R0 en R1 met een bepaalde waarde te laden. Jammer genoeg is het niet zo, dat de frequentie rechtstreeks in een registerpaar kan worden gezet. Er dient eerst een formule op los te worden gelaten. Die formule luidt als volgt:

RP = 111860 / frequentie

Hierin is RP het registerpaar. In een registerpaar kunnen we alleen het gehele deel van het resultaat van de deling plaatsen. In een BASIC-programma zouden we dan ook de formule als volgt dienen te schrijven:

RP = INT (111860 / frequentie)

Laten we een voorbeeldje invullen. Stel, dat we een frequentie van 400 Herz willen produceren. In de formule vullen we die waarde in en krijgen dan als resultaat RP=111860/400=279. Met andere woorden, we moeten de waarde 279 in de registers 0 en 1 laden (voor geluidskanaal A). Register 0 bestaat uit 8 bitjes en in 8 bitjes kan maximaal een waarde van 255 worden opgeslagen. We hebben dus een negende bitje nodig. Dat negende bitje vertegenwoordigt dan een waarde 256. Daarna kunnen we de overblijvende waarde 279-256=23 in de overige 8 bitjes plaatsen.

Het negende bitje zit in het tweede register (1) de eerste acht bitjes zitten in het eerste register (0). Je zou ook kunnen zeggen, dat register 1 het meest significante register is (MSR). Immers de bitjes in dat register vertegenwoordigen een hogere waarde dan de bitjes in register 0. Zo kunnen we ook stellen dat register 0 dan het minst significante register is (LSR). LSR staat voor de Engelse woorden Least Significant Register.

Om nu te voorkomen, dat we zelf de inhoud van de registers voor een bepaalde frequentie moeten berekenen, zullen we nu

een klein programma schrijven dat de berekening voor ons uitvoert.

```
100 INPUT "Frequentie";F
110 RP=INT(111860!/F)
120 MSR=INT(RP/256)
130 LSR=RP-256*MSR
140 PRINT "LSR =";LSR
150 PRINT "MSR =";MSR
```

Bovendien kunnen we de berekende waarden voor LSR en MSR gebruiken om de geluidsprocessor te laden. Hiertoe kunnen we een SOUND-statement aan het programma toevoegen, en wel:

```
160 SOUND 0,LSR
170 SOUND 1,MSR
```

We horen echter nog steeds niets, want we hebben het volume nog niet ingesteld en we hebben het betreffende kanaal nog niet aangezet. Laten we daarom maar eens kijken hoe we het volume instellen. Het register dat het volume van kanaal A regelt is register 8. In dit register mogen we een waarde, die varieert tussen 0 en 16, invullen. De waarde 16 zullen we echter even overslaan. Daarop komen we later terug.

Laten we aannemen dat we de volumeregelaar voor kanaal A op 12 willen zetten. Dat kan dan eenvoudig door het volgende statement aan het programma toe te voegen:

```
180 SOUND 8,12
```

Toch horen we nog steeds niets. We zullen het betreffende kanaal (A) nog aan moeten zetten. Het aan en uitzetten van kanalen kan worden gedaan door het invullen van waarden in register 7. Indien in register 7 alle bitjes 1 zijn, staan alle kanalen uit. De drie minst significante bitjes zijn voor de kanalen A, B en C. Het bitje voor kanaal A heeft de waarde 1 indien het gezet is. Het bitje voor kanaal B heeft de waarde 2 indien het gezet is en het bitje voor kanaal C heeft de waarde 4 indien het gezet is. Om dus kanaal A aan te zetten, moet

het bijbehorende bitje 0 worden gemaakt. We zullen dus de binaire waarde 11111110 naar register 7 moeten sturen. Omgerekend naar decimaal is dat 254. Het volgende statement dient dus aan het programma te worden toegevoegd:

```
190 SOUND 7,254
```

En jawel, tenzij de volumeregelaar op het televisietoestel op 0 staat, knalt er een flinke pieptoon uit de luidspreker. Bovendien valt het op dat de toon niet ophoudt. Zolang de registers niet zodanig worden gewijzigd, dat het geluid moet veranderen, of dat een kanaal wordt uitgezet, zal dezelfde toon geproduceerd blijven. Het onderbreken van het programma (met de STOP-toets) beeindigt het geluid.

Om tenslotte voorgaand programma nog leuker te maken, kunnen we de volgende programmaregel toevoegen, waardoor we steeds een nieuwe waarde voor de frequentie kunnen ingeven.

```
200 GOTO 100
```

11.3 Ruis

Ruis heeft evenals een toon een bepaalde frequentie. Afhankelijk van de ruisfrequentie zal de ruis verschillend klinken. Het kiezen van een bepaalde ruisfrequentie is mogelijk, door register 6 een bepaalde inhoud te geven.

Zoals blijkt uit afbeelding 11-1, worden van register 6 slechts 5 bitjes gebruikt. Met vijf bitjes zijn slechts 32 combinaties mogelijk (0-31). De relatie tussen de registerinhoud en de ruisfrequentie is de volgende.

De maximale ruisfrequentie is 111860 Herz. De geproduceerde ruisfrequentie is de maximale frequentie gedeeld door de inhoud van register 6. Als bijvoorbeeld de inhoud van register 6 de waarde 2 is, dan zal de uiteindelijke ruisfrequentie 111860/2=55930 Herz zijn. Indien de inhoud van register 6 nul

is, dan zal er een ruis met onbepaalde frequentie worden geproduceerd. De volgende tabel somt alle mogelijke ruisfrequenties op.

Inhoud R6	Ruis-freq.	Inhoud R6	Ruis-freq.
0	-	16	6991
1	111860	17	6580
2	55930	18	6214
3	37286	19	5887
4	27965	20	5593
5	22372	21	5326
6	18643	22	5084
7	15980	23	4863
8	13982	24	4660
9	12428	25	4474
10	11186	26	4302
11	10169	27	4142
12	9321	28	3995
13	8604	29	3857
14	7990	30	3728
15	7457	31	3608

Behalve de ruisfrequentie dienen we ook het volume in te stellen. Dit gaat net als bij het volume van de toon. Indien we de ruis via kanaal B hoorbaar willen maken, zullen we register 9 met het gewenste volume moeten laden.

Nadat we de ruisfrequentie en het volume hebben ingesteld, dienen we nog aan te geven via welk kanaal of welke kanalen we die ruis hoorbaar willen maken. Dit kunnen we doen door een bitje in register 7, onder het kopje NOISE, op nul te zetten. We zullen weer een voorbeeld nemen. Laten we ruis via kanaal B gaan maken.

Om de ruis via kanaal B aan te zetten, dienen we het betreffende bitje 0 te maken. We moeten dus de volgende bitjes in register 7 zetten: 11 101 111, hetgeen decimaal de waarde

239 is.

Het volgende programma geeft een voorbeeld van ruis. Probeert u eens verschillende ruisfrequenties uit.

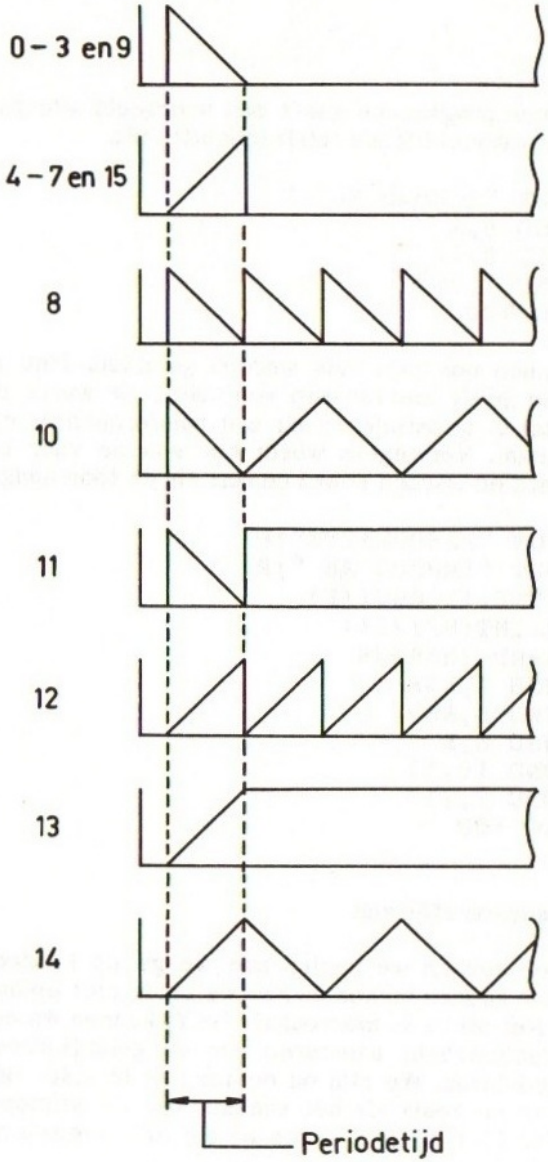
```
100 INPUT "Inhoud R6";R
110 SOUND 6,R
120 SOUND 9,12
130 SOUND 7,239
140 GOTO 100
```

Tonen kunnen ook met ruis worden gemixed. Het volgende programma geeft hiervan een voorbeeld. Er wordt een toon voor kanaal C gedefinieerd en een ruisfrequentie voor hetzelfde kanaal. Vervolgens wordt het volume voor kanaal C gedefinieerd en worden zowel de ruis als de toon aangezet.

```
100 INPUT "FREQUENTIE";F
110 INPUT "INHOUD R6 ";R
120 RP=INT(111860!/F)
130 MSR=INT(RP/256)
140 LSR=RP-256*MSR
150 SOUND 4,LSR
160 SOUND 5,MSR
170 SOUND 6,R
180 SOUND 10,12
190 SOUND 7,219
200 GOTO 100
```

11.4 Bijzondere effecten

Tot nu toe hebben we gezien hoe we geluid kunnen maken, hoe we ruis kunnen maken en hoe we beide met elkaar kunnen mengen. Net als in de macrotaal PLAY, kunnen we echter ook bij het rechtstreeks aansturen van de geluidsprocessor het geluid moduleren. We zijn nu echter ook in staat om ruis te produceren en zoals uit het schema van de geluidsprocessor (afbeelding 11-1) blijkt, kunnen we die ruis, evenals het geluid moduleren.



Afb. 11-2 De verschillende modulatievormen.

Voor het moduleren dienen we drie acties te ondernemen:

- Bepaal de periodetijd.
- Kies de gewenste modulatievorm.
- Zet de modulator aan.

De periodetijd (zie afbeelding 11-2) kunnen we bepalen door de registers 11 en 12 met een bepaalde waarde te laden. Deze waarde is echter niet het aantal seconden, maar is een waarde, die we moeten bepalen met behulp van een formule. Deze formule luidt als volgt:

RP = tijd (in seconden) * 6991

Stel dat we het geluid (of de ruis) in 1 seconde willen laten aanzwellen van niets tot maximum, dan zullen we het registerpaar (RP) moeten laden met de waarde 1 * 6991 = 6991. Net als bij de geluidsregisters, zullen we ook nu weer moeten uitrekenen wat de inhoud van ieder register moet zijn. Dit doen we weer in BASIC en wel als volgt:

```
MSR=INT(RP/256)
LSR=RP-256*MSR
```

Het resultaat zou dan zijn, dat register 12 (MSR) de inhoud 6991/256=27 zou krijgen, en register 11 (LSR) de waarde 6991-256*MSR=79. Om de geluidsgenerator nu met deze waarden te laden dienen we de volgende statements in te geven:

```
SOUND 11,LSR
SOUND 12,MSR
```

Wellicht zult u vaak in een bepaald programma slechts een bepaalde periodetijd willen gebruiken. Om nu niet steeds de periodetijd te hoeven berekenen, kunt u gebruik maken van de volgende tabel, waarin de meest gebruikte periodetijden zijn opgenomen.

Tijd in seconden	Inhoud van R11	R12
9	199	245
8	120	218
7	41	191
6	218	163
5	139	136
4	60	109
3	237	81
2	158	54
1	79	27
0.9	147	24
0.8	216	21
0.7	29	19
0.6	98	16
0.5	167	13
0.4	236	10
0.3	49	8
0.2	118	5
0.1	187	2

Nadat we de periodetijd hebben bepaald, dienen we nog de modulatievorm te kiezen. Dit is eenvoudig. Afbeelding 11-2 laat de verschillende modulatievormen zien. Door eenvoudigweg het cijfer, dat naast de door ons gewenste modulatievorm staat in register 13 te laden, geven we de geluidsprocessor opdracht het geluid volgens die modulatievorm te moduleren. Het BASIC-statement om dit te doen ziet er als volgt uit:

```
SOUND 13,8
```

De waarde 8 geeft een repeterend aanzwellend geluid. In plaats van de 8 mogen we natuurlijk ook een andere waarde ingeven.

De modulator moet nu nog worden aangezet. Dit gaat als volgt. Met de registers 8 tot en met 10 kunnen we het volume voor respectievelijk de kanalen A, B en C instellen. Indien we nu een van deze registers met de waarde 16 laden, zal voor

het bij dat register behorende kanaal de modulator aangeschakeld worden. Willen we bijvoorbeeld het geluid van kanaal C moduleren, dan zullen we register 10 met de waarde 16 moeten laden. In BASIC ziet dit er als volgt uit:

```
SOUND 10,16
```

Het volgende programma laat van het voorgaande een voorbeeld zien. Met regelnummer 190 wordt de ruisfrequentie ingesteld. Met regelnummer 200 wordt kanaal A op moduleren ingesteld. Regelnummer 220 bepaalt de modulatievorm en regelnummers 230 en 240 bepalen de modulatiefrequentie. Met regelnummer 250 wordt de ruis op kanaal A aangeschakeld. Het gevolg van deze acties en de daarbij gekozen waarden is, dat er een trommel wordt gesimuleerd.

```
100 INPUT "Laagste frequentie";LF
110 INPUT "Hoogste frequentie";HF
120 VOL=0
130 FB=HF-LF
140 RP=INT(111860!/FB)
150 MSR=INT(RP/256)
160 LSR=RP-256*MSR
170 SOUND 2,LSR:'kanaal B
180 SOUND 3,MSR:'kanaal B
190 SOUND 6,5:  'ruisfrequentie
200 SOUND 8,16: 'kanaal A moduleren
210 SOUND 10,12:'volume C
220 SOUND 11,0: 'modulatieperiode LSR
230 SOUND 12,4: 'modulatieperiode MSR
240 SOUND 13,8: 'modulatievorm
250 SOUND 7,241:'noise A, toon B/C
260 FOR I=LF TO HF STEP 5
270 RP=INT(111860!/I)
280 MSR=INT(RP/256)
290 LSR=RP-256*MSR
300 SOUND 4,LSR:'kanaal C
310 SOUND 5,MSR:'kanaal C
320 SOUND 9,VOL:'volume kanaal B
330 VOL=VOL+1
```

```
340 IF VOL>15 THEN VOL=0
350 NEXT I
360 FOR I=HF TO LF STEP -5
370 RP=INT(111860!/I)
380 MSR=INT(RP/256)
390 LSR=RP-256*MSR
400 SOUND 4,LSR:'kanaal C
410 SOUND 5,MSR:'kanaal C
420 SOUND 9,VOL:'volume kanaal B
430 VOL=VOL+1
440 IF VOL>15 THEN VOL=0
450 NEXT I
460 GOTO 260
```

Een ander speciaal effect, dat in voorgaand programma is toegepast, is een sirene. Dit effect wordt verkregen, door (in het programma) de frequentie van kanaal C te laten varieren tussen twee frequenties. Deze twee frequenties zijn in regelnummers 100 en 110 opgevraagd. Daarna worden deze frequenties in een FOR...NEXT-lus gebruikt. In feite gebruiken we twee FOR...NEXT-lussen, de ene om de toonhoogte te laten stijgen, de andere om de toonhoogte weer te laten dalen. De snelheid waarmee de maximum toonhoogte wordt bereikt hangt af van de gekozen stapgrootte.

Om het volume van het geluid te laten aanzwellen kunnen we behalve door gebruik te maken van de modulator, ook zelf het volume wijzigen. Ook hiervan is in het voorgaande programma een voorbeeld gegeven. In regelnummer 120 wordt de variabele VOL met de waarde 0 geladen. Regelnummers 130 tot en met 180 berekenen de waarden voor de geluidsregisters van kanaal B en laden deze waarden in de registers 2 en 3. In regel 320 wordt het volume voor kanaal B ingesteld, en wel met de waarde 0. Iedere keer dat de FOR...NEXT-lus wordt doorlopen, wordt de variabele VOL echter opgehoogd, totdat de waarde 15 is bereikt. Op dat moment wordt de waarde van VOL weer op 0 gezet. Het gevolg is, dat het geluid op kanaal B langzaam aanzwelt, totdat het maximum volume is bereikt. Daarna begint het geluid weer opnieuw aan te zwellen. Het leuke van dit effect is, dat het aanzwellen van dit geluid niet

synchroon verloopt met het aanzwellen van het geluid via de modulator. Op die manier zijn leuke effecten te bereiken.

Door verschillende waarden voor de verschillende registers in te vullen, kunt u met dit programma een groot aantal verschillende effecten genereren.

Ook met het programma uit hoofdstuk 10, waarmee de inhoud van de registers van de geluidsgenerator op het beeldscherm worden afgedrukt, en waarmee u de registers een nieuwe inhoud kunt geven, kunt u nu naar hartelust experimenteren. Met de nu verkregen kennis van de geluidsprocessor zult u in staat zijn een groot aantal leuke effecten te bereiken.

12 De Video Display Processor

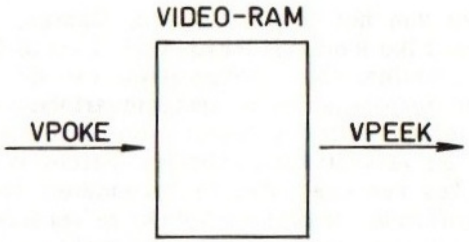
Elke MSX-computer is uitgerust met een TSM9129A Video Display Processor (VDP). De VDP zorgt voor de besturingssignalen en de informatie, die nodig zijn voor het produceren van de beelden (teksten, figuren, etc.) op het scherm. De VDP beschikt over een eigen RAM video-geheugen met een capaciteit van 16 kilobytes. De wijze waarop het video-geheugen wordt gebruikt is afhankelijk van de mode (SCREEN 0, 1, 2 of 3), waarin de computer werkt.

De VDP is uitgerust met 9 besturingsregisters, die elk 8 bits groot zijn. Deze register zorgen voor de besturing van de VDP en het 16 kilobytes video-geheugen.

Van de grafische statements PRINT, DRAW, LINE en CIRCLE, die worden gebruikt voor het plaatsen van informatie op het beeldscherm, worden de resultaten in tabellen in het video-geheugen geplaatst. Met de informatie uit deze tabellen wordt door de VDP het beeld op het beeldscherm samengesteld. De adressen van de tabellen zijn voor de programmeur toegankelijk gemaakt door middel van de systeemvariabele BASE(X). Deze systeemvariabele bergt alle tabeladressen.

Voor het lezen van gegevens uit het video-geheugen dient de functie VPEEK en voor het schrijven van gegevens naar het video-geheugen dient het statement VPOKE (zie afbeelding 12-1).

Voor het lezen van de inhoud uit de besturingsregisters van de VDP en voor het schrijven van gegevens naar de besturingsregisters heeft de programmeur de beschikking over de systeemvariabele VDP(X).



Afb. 12-1 Schrijven in en lezen uit video RAM.

Wanneer we de Video Display Processor zelf willen gaan besturen, dus niet via de gebruikelijke statements (PRINT, DRAW, LINE, CIRCLE, etc.), zal een betere kennis nodig zijn van de Video Display Processor, van de besturingsregisters en van het gebruik van de tabellen in het video-geheugen. De tabeladressen kunnen bijvoorbeeld niet zonder meer worden veranderd. Dit kan problemen geven, zoals het "ophangen" van de computer, waardoor deze niet meer normaal kan functioneren. In zo'n geval is het het beste de computer eerst uit te schakelen en dan weer in te schakelen. Tijdens het inschakelen zullen de tabellen in het video-geheugen weer door de computer worden geladen met de juiste informatie. Deze informatie staat in het ROM-geheugen van de computer.

In de volgende paragrafen zal worden uitgelegd, hoe in de verschillende modes (SCREEN 0, 1, 2 en 3) de tabellen kunnen worden gehanteerd en hoe zorgvuldig gebruik kan worden gemaakt van BASE, VDP, VPEEK en VPOKE. Maar nogmaals, de computer kan ook werken zonder dat u gebruik maakt van BASE, VDP, VPEEK en VPOKE. Het biedt u alleen wat meer mogelijkheden. Een voorbeeld hiervan is het creeren van eigen tekens, door in de betreffende tabel in het video-geheugen door middel van het statement VPOKE de matrix van bestaande tekens te veranderen.

12.1 Tabellen in het video-geheugen.

Zoals in de inleiding al naar voren is gebracht, dienen de inhouden van de tabellen in het video-geheugen voor het

samenstellen van het beeld (teksten, figuren, etc.) op het beeldscherm. Elke mode (SCREEN 0, 1, 2 en 3) beschikt over zijn eigen tabellen. De beginadressen van de verschillende tabellen zijn opgeslagen in de **systeemvariabele BASE(X).** De systeemvariabele wordt gebruikt voor het selecteren van adressen in de verschillende tabellen. Verder is het mogelijk het beginadres van een tabel te veranderen door de inhoud van de betreffende variabele BASE(X) te veranderen. Doe dit echter altijd met de grootste zorgvuldigheid.

De parameter X kan een waarde hebben van 0 tot en met 19. Zo vertegenwoordigen de inhouden van BASE(0) en BASE(2) de beginadressen van de tabellen, die worden gebruikt in de tekstmode 1 (SCREEN 0). De tabellen die in tekstmode 1 worden gebruikt zijn de **scherm-info-tabel** en de **matrix-tabel,** waarvan de beginadressen respectievelijk in BASE(0) en BASE(2) staan. De adressen worden tijdens het inschakelen van de computer automatisch toegekend aan de systeemvariabelen. De computer haalt de adressen uit het ROM-geheugen.

De beginadressen worden ook wel BASE-adressen genoemd. Zo is het BASE-adres voor de scherm-info-tabel in de tekstmode 1 gelijk aan de inhoud van BASE(0). De verschillende BASE-adressen beginnen elk op een veelvoud van een bepaalde waarde. Deze waarde is afhankelijk van de toegepaste tabel en kan 64, 128, 1024 of 2048 zijn. Het BASE-adres kan ook 0 zijn.

In tabel 12-1 staan voor elke mode de toegepaste tabellen, de bijbehorende BASE-adressen, de lengte van de tabellen en van welke waarde het BASE-adres een veelvoud moet zijn.

Met het volgende programma worden de beginadressen (BASE-adressen) van alle toegepaste tabellen op het beeldscherm afgedrukt.

```
100 CLS
110 PRINT "BEGINADRESSEN TABELLEN"
120 PRINT
130 FOR I=0 TO 19
```

```
140 PRINT "BASE(";I;")=";BASE(I)
150 NEXT I
160 END
```

Mode	Naam tabel	BASE-adres (beginadres)	Lengte in bytes	Veelvoud van
SCREEN 0 (tekstmode 1)	- Scherm-info tabel	BASE(0)	960	1024
	- Matrix-tabel	BASE(2)	2048	2048
SCREEN 1 (tekstmode 2)	- Scherm-info-tabel	BASE(5)	768	1024
	- Kleurtabel	BASE(6)	32	64
	- Matrix tabel	BASE(7)	2048	2048
	- Sprite-info tabel	BASE(8)	128	128
	- SPRITE$-tabel	BASE(9)	2048	2048
SCREEN 2 (grafische mode 1)	- Scherm-info tabel	BASE(10)	768	1024
	- Kleurtabel	BASE(11)	6144	64
	- Matrix-tabel	BASE(12)	6144	2048
	- Sprite-info tabel	BASE(13)	128	128
	- SPRITE$-tabel	BASE(14)	2048	2048
SCREEN 3 (grafische mode 2)	- Scherm-info tabel	BASE(15)	768	1024
	- Matrix-tabel	BASE(17)	2048	2048
	- Sprite-info tabel	BASE(18)	128	128
	- SPRITE$-tabel	BASE(19)	2048	2048

Tabel 12-1 Overzicht van tabellen in video-RAM.

Wanneer men een bepaald adres van een tabel wil uitlezen, zal bij het beginadres van de tabel altijd een bepaalde waarde moeten worden opgeteld. Deze waarde is afhankelijk van de positie van het adres in de tabel. Voor het eerste adres in de tabel zal bij het beginadres de waarde 0 moeten worden opgeteld. Bij het tweede adres moet de waarde 1 worden opgeteld, bij het derde de waarde 2, etc. (zie afb. 12-2). Wanneer bijvoorbeeld in de tekstmode 1 het derde adres van de matrix-tabel moet worden uitgelezen, dan zal de programmaregel, waarmee dat wordt gedaan, er als volgt gaan uitzien:

```
10 A=VPEEK(BASE(2)+2)
```

Opmerking:
Elk adres in het video-geheugen bestaat uit 8 bits.

Het formaat van de functie VPEEK is:

VPEEK(X)

└──────── geheugenadres (0-16383)

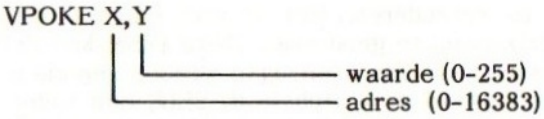
VPEEK leest de inhoud van het adres X uit het video-geheugen. Het resultaat kan een waarde hebben van 0 tot en met 255.

Met het volgende programma wordt de hexadecimale waarde van de inhoud van een adres afgedrukt op het scherm.

```
10 CLS
20 A=VPEEK(BASE(2)+8)
30 A$="00"+HEX$(A)
40 PRINT RIGHT$(A$,2)
50 END
```

Vervolgens zullen we eens kijken hoe we in het video-geheugen de inhoud van een bepaald adres in een tabel kunnen veranderen. Hiervoor dient het statement VPOKE. Laten we eens een adres veranderen in de matrix-tabel voor tekstmode 1. In de matrix-tabel zijn voor elk teken 8 bytes gereserveerd. Met deze 8 bytes wordt door middel van enen en nullen het hele teken gevormd (zie afbeelding 12-3). In de afbeelding staat de hoofdletter B uitgecodeerd. Elke positie op het beeldscherm bestaat uit 8x8 beeldpuntjes. Wanneer in de matrix een "1" staat, zal het overeenkomstige beeldpuntje de voorgrondkleur krijgen. Door nu op adres BASE(2)+528 de waarde &HF0 te veranderen in &H00, zal de bovenkant van de letter B niet meer op het scherm worden afgedrukt. De matrix-tabel wordt tijdens het inschakelen automatisch door de computer gegenereerd.

Voordat we het betreffende adres gaan veranderen, zal eerst het formaat van het statement VPOKE worden gegeven:



Met VPOKE kan een waarde Y naar adres X in het videogeheugen worden geschreven. Parameter Y kan een numerieke constante, een numerieke variabele of een numerieke uitdrukking zijn. Hetzelfde geldt voor parameter X.

Het programma voor het veranderen van adres BASE(2)+528 gaat er als volgt uitzien:

```
100 CLS
110 VPOKE BASE(2)+528,&H0
120 FOR I=0 TO 20
130 PRINT "B";
140 NEXT I
150 END
```

Het programma laat zien, hoe gemakkelijk het is de vorm van een teken te veranderen. Vandaar de opmerking, gemaakt in het begin van dit hoofdstuk, dat het veranderen van inhouden van tabellen zorgvuldig dient te geschieden.

Er zijn twee manieren om de matrix van de letter B in de tabel weer te herstellen, namelijk programmatisch of door het uitschakelen en daarna weer inschakelen van de computer. De laatste manier heeft tot gevolg dat de tabel weer automatisch wordt geladen met de juiste informatie. Door in het vorige programma regelnummer 110 als volgt te veranderen, wordt de matrix van de letter weer hersteld.

```
110 VPOKE BASE(2)+528,&HF0
```

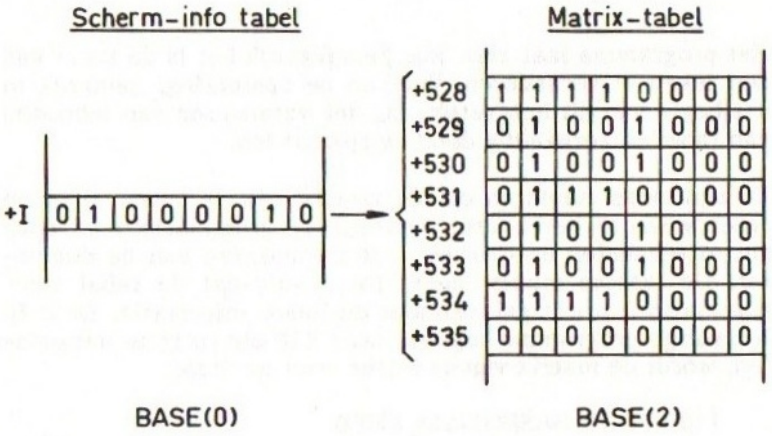
We hebben gezien hoe eenvoudig het is om door middel van BASE(X) en VPOKE X,Y veranderingen in het video-geheugen aan te brengen. Op deze manier kunnen we ook eigen tekens in de matrix-tabel plaatsen, door tekens, die toch niet vaak

worden gebruikt, te veranderen. Het is ook mogelijk een totaal nieuwe matrix-tabel te genereren. Deze tabel kan dan in een ander gebied in het video-geheugen worden geplaatst. Men moet hiervoor dan wel de variabele BASE(2) een andere waarde geven, zodat de door de computer geladen tabel blijft bestaan. We gaan hierbij uit van tekstmode 1 (SCREEN 0).

In de volgende paragrafen zullen voor elke mode de bijbehorende tabellen worden behandeld.

12.2 Tekstmode 1 (SCREEN 0).

In de tekstmode 1 wordt van twee tabellen gebruik gemaakt, namelijk van de **scherm-info-tabel** met beginadres BASE(0) en de **matrix-tabel** met beginadres BASE(2). Met behulp van deze twee tabellen worden door de Video Display Processor de beelden (teksten) op het scherm samengesteld (zie afbeelding 12-2).

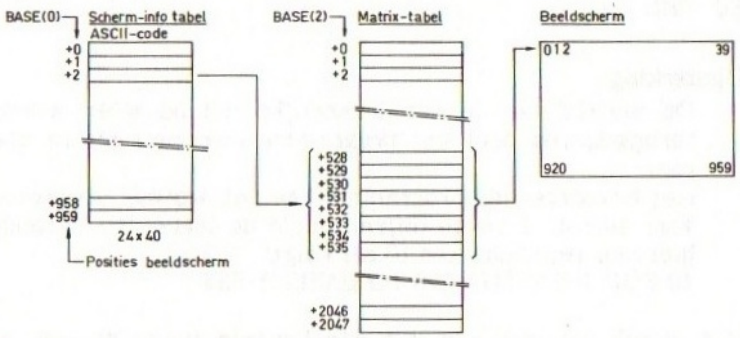


Afb. 12-2 Relatie tussen scherm-info- en matrix-tabel.

De scherm-info-tabel bevat alle ASCII-codes van de tekens die op het scherm verschijnen. De positie in de tabel correspondeert met de positie op het scherm. Daar in de tekstmode

1 op het scherm 24 x 40 = 960 tekens kunnen worden afgedrukt, is de lengte van de scherm-info-tabel 960 bytes.

In de matrix-tabel zijn voor elk teken 8 bytes gereserveerd. Met de inhoud (nullen en enen) van de 8 bytes wordt het teken uitgebeeld. In afbeelding 12-3 staat in de adressen BASE(2)+528 tot en met BASE(2)+535 de hoofdletter B uitgebeeld. Met behulp van de inhoud van deze adressen wordt door de Video Display Processor de letter B op het scherm afgebeeld. De positie, waar de letter B op het scherm moet worden afgebeeld, wordt bepaald door het adres in de scherm-info-tabel. Dit is het adres waar de ASCII-code van de af te beelden letter B staat.



Afb. 12-3 Tabellen in video-RAM na SCREEN 0.

De adressen in de matrix-tabel worden afgeleid van de inhoud (ASCII-code) van de scherm-info-tabel. Wanneer op een bepaald adres in de scherm-info-tabel de letter B staat (ASCII-code = 01000010 = 66), wordt het eerste adres van het blok van 8 bytes als volgt berekend:

BASE(2)+66x8 = BASE(2)+528.

Het teken bepaalt dus gelijk het adres in de matrix-tabel.
De matrix voor de hoofdletter P (ASCII-code = 01010000 = 80) start op adres BASE(2)+80x8 = BASE(2)+640.

Daar de MSX-tekenset uit 256 verschillende tekens bestaat, is

de lengte van de matrix-tabel 8x256 = 2048 bytes.

Met het volgende programma wordt de matrix-tabel dusdanig veranderd, dat de tekens bij het afdrukken op het scherm de achtergrondkleur krijgen, en de rest van de matrix (8x8 beeldpuntjes) de voorgrondkleur. Dit kan worden bereikt door in de matrix-tabel alle enen in nullen en alle nullen in enen te veranderen.

```
10 FOR I=BASE(2) TO BASE(2)+2047
20 A=255-VPEEK(I)
30 VPOKE I,A
40 NEXT I
50 END
```

Opmerking:

De matrix kan de oorspronkelijke inhoud weer worden teruggegeven door het programma nogmaals uit te voeren.
Het bovenstaande programma kan ook worden uitgevoerd voor slechts 1 teken (bijvoorbeeld de letter B). Verander hiervoor regelnummer 10 als volgt:
10 FOR I=BASE(2)+528 TO BASE(2)+535

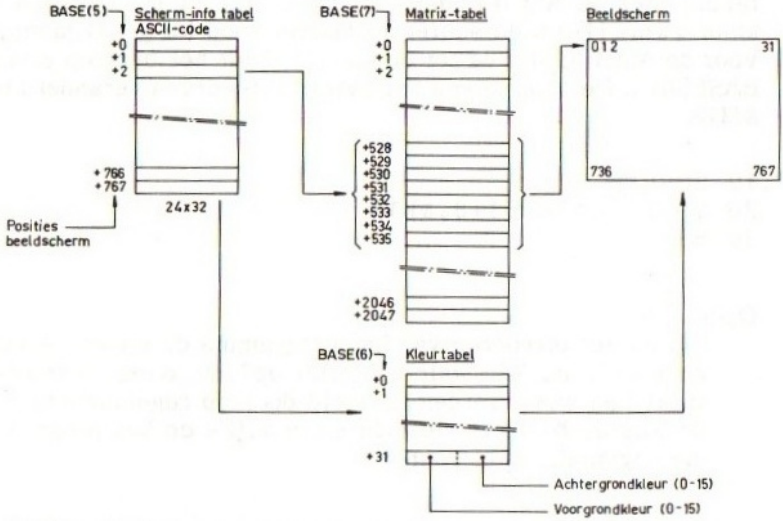
Het is ook mogelijk om van een bepaald teken de enen en nullen van de 8 bytes uit de matrix-tabel op het scherm af te drukken. Met het volgende programma kan dit worden verwezenlijkt:

```
100 CLS
110 INPUT "ASCII-code (0-255)";C
120 PRINT
130 A=BASE(2)+8*C
140 FOR I=A TO A+7
150 A$="00000000"+BIN$(VPEEK(I))
160 PRINT RIGHT$(A$,8)
170 NEXT I
180 I$=INKEY$:IF I$<>CHR$(13) GOTO 180
190 GOTO 100
```

In de tekstmode 1 kunnen geen kleuren worden gespecificeerd. Alleen de voorgrond- en achtergrondkleur zijn actief. In tekstmode 2 kunnen echter wel kleuren worden gecodeerd.

12.3 Tekstmode 2 (SCREEN 1).

In tekstmode 2 wordt door de Video Display Processor van drie tabellen gebruik gemaakt om de beelden op het scherm samen te stellen (zie afbeelding 12-4). De tabellen zijn de **scherm-info-tabel** met beginadres BASE(5), de **matrix-tabel** met beginadres BASE(7) en de **kleur-tabel** met beginadres BASE(6). Wanneer met sprites wordt gewerkt, wordt behalve van de drie hiervoor genoemde tabellen ook nog gebruik gemaakt van een **sprite-info-tabel** en een **SPRITE$-tabel.** Deze laatste twee tabellen worden behandeld in paragraaf 12.6.



Afb. 12-4 Tabellen in video-RAM na SCREEN 1.

De scherm-info-tabel en de matrix-tabel functioneren op dezelfde manier als in tekstmode 1. De lengte van de scherm-info-tabel in tekstmode 2 is echter 24x32 = 768 bytes. Behalve van de scherm-info-tabel en matrix-tabel, wordt in tekstmode 2 ook nog gebruik gemaakt van een kleur-tabel, welke uit 32 bytes bestaat. Door middel van deze tabel kunnen verschillende kleuren worden gecodeerd.

De kleur-tabel wordt als volgt gebruikt. Het eerste byte bevat de kleur van de tekens met de ASCII-codes 0 tot en met 7. Het tweede byte bevat de kleur van de tekens met de ASCII-codes 8 tot en met 15, enz. Van elk byte specificeert het minst significante deel de achtergrondkleur en het meest significante deel de voorgrondkleur. De default-waarde is voor de voorgrondkleur 15 en voor de achtergrondkleur 4 (dit maakt samen de waarde 244).

Met het volgende programma wordt in de kleur-tabel voor de tekens met de ASCII-codes 64 tot en met 71 de voorgrondkleur zwart (1) en de achtergrondkleur rood (8) gedefinieerd. Voor de ASCII-codes 64 tot en met 71 dient het byte op adres BASE(6)+8. De waarde van het byte moet worden veranderd in &H18.

```
10 SCREEN 1
20 VPOKE BASE(6)+8,&H18
30 END
```

Opdracht:

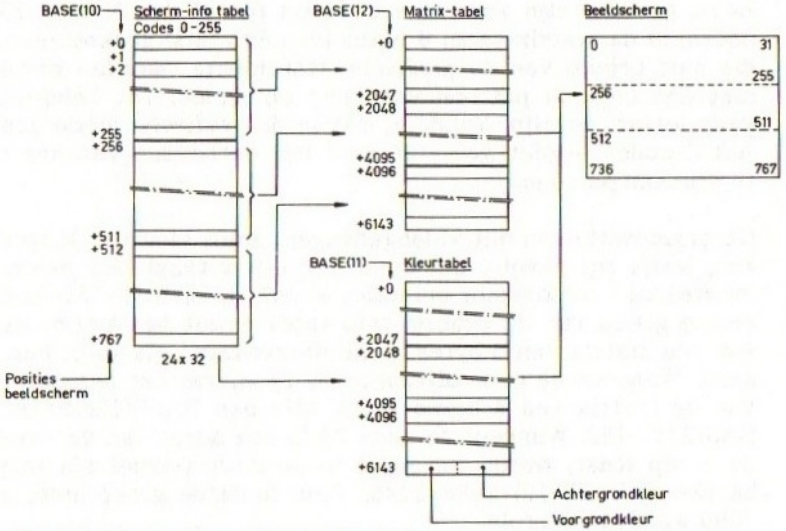
Tik na het uitvoeren van het programma de letters A tot en met K in. Wat valt u hierbij op? De oorspronkelijke staat kan weer worden hersteld door op regelnummer 20 de waarde &H18 te veranderen in &HF4 en het programma nogmaals uit te voeren.

12.4 Grafische mode 1 (SCREEN 2).

In de grafische mode 1 wordt door de Video Display Processor van drie tabellen gebruik gemaakt om de beelden (teksten en

figuren) op het scherm samen te stellen (zie afbeelding 12-5).

De tabellen, die in de grafische mode 1 worden gebruikt, zijn de **scherm-info-tabel** met beginadres BASE(10), de **matrix-tabel** met beginadres BASE(12) en de **kleur-tabel** met beginadres BASE(11).
Het beeldscherm is opgedeeld in drie stukken van elk 8 regels. Deze opdeling is ook terug te vinden in de scherm-info-tabel. Wanneer in de grafische mode 1 met sprites wordt gewerkt, wordt behalve van de drie hiervoor genoemde tabellen ook gebruikt gemaakt van een sprite-info-tabel en een SPRITE$-tabel. Deze tabellen worden behandeld in paragraaf 12.6.



Afb. 12-5 Tabellen in video-RAM na SCREEN 2.

Per tekenpositie (8x8 beeldpuntjes) op het beeldscherm is er in de scherm-info-tabel een code mogelijk van 0 tot en met 255. Deze codes zijn niet te vergelijken met ASCII-codes. Voor de grafische modes is het beeldscherm ingedeeld in 3x8x32=768 posities, elk bestaande uit 8x8 beeldpuntjes. De positie van een code in de scherm-info-tabel correspondeert met de positie op het beeldscherm.

Tijdens het uitvoeren van het statement SCREEN 2, wordt de scherm-info-tabel gevuld met drie reeksen codes, die een waarde hebben van 0 tot en met 255. De codes in de scherm-info-tabel verwijzen naar het begin van een blok van 8 bytes in de matrix-tabel. Dit blok van 8 bytes in de matrix-tabel is in feite een matrix van 8x8 bitjes. Deze matrix correspondeert met de matrix van beeldpuntjes op het beeldscherm. Daarbij geldt, dat de betreffende beeldschermpositie correspondeert met het adres van de code in de scherm-info-tabel (de eerste code in de scherm-info-tabel wijst dus de matrix aan, die op de eerste beeldschermpositie moet worden afgedrukt).

Zoals gezegd, zijn voor elke code (in totaal dus 3 keer 256 codes) in de matrix-tabel 8 bytes (van 8x8 bits) gereserveerd, die met behulp van de grafische statements worden gevuld met een bepaald patroon van enen en nullen. Dit behoeven geen letters te zijn. Vandaar, dat in de grafische mode geen ASCII-codes worden gebruikt voor het onthouden van het af te drukken patroon.

De organisatie van het videogeheugen, zoals hiervoor besproken, heeft tot gevolg, dat er in de matrix-tabel een gefragmenteerde beeldopbouw ontstaat. Vanuit de code (0-255) in de eerste groep van de scherm-info-tabel wordt het beginadres van een matrix van 8 bytes in de matrix-tabel als volgt berekend. Wanneer de code bijvoorbeeld 24 is, zal het beginadres van de matrix van 8 bytes gelijk zijn aan BASE(12)+24x8 = BASE(12)+192. Wanneer de code 24 in een adres van de tweede groep staat, wordt het adres in de matrix-tabel als volgt berekend; BASE(12)+2048+24x8. Voor de derde groep moet er 4096 worden bijgeteld.

De lengte van de scherm-info-tabel is 3x256=768 bytes. We hebben ook gezien dat er voor ieder adres in de scherm-info-tabel acht bytes in de matrix-tabel aanwezig zijn. De totale lengte van de matrix-tabel is zodoende 3x256x8 = 6144 bytes.

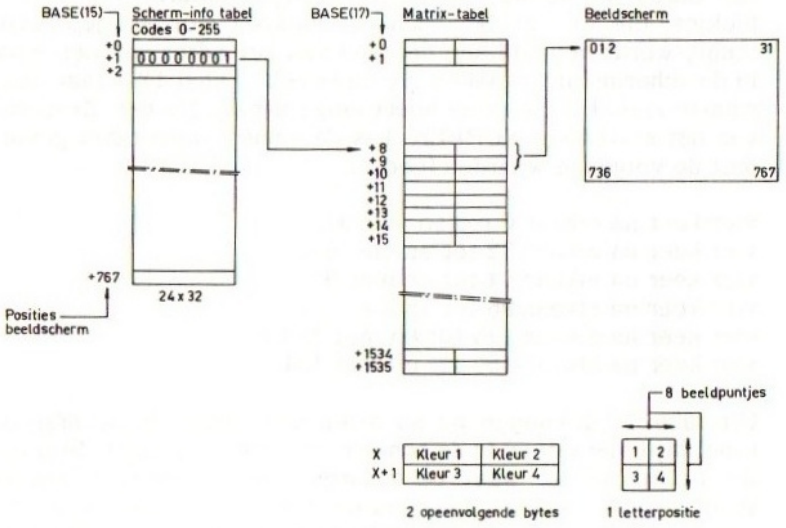
Per groep van 8 horizontale beeldpuntjes (1 byte in de matrix-tabel) kan een voorgrond- en een achtergrondkleur worden

gespecificeerd. De kleuren kunnen worden aangegeven in de kleur-tabel. De bytes in de kleurtabel lopen parallel aan de bytes in de matrix-tabel. In het meest significante deel van het byte wordt de voorgrondkleur aangegeven en in het minst significante deel de achtergrondkleur.

12.5 Grafische mode 2 (SCREEN 3).

In de grafische mode 2 (SCREEN 3) wordt door de VDP van twee tabellen gebruik gemaakt. Dit zijn de **scherm-info-tabel** met beginadres BASE(15) en de **matrix-tabel** met beginadres BASE(17). Een kleurtabel ontbreekt in grafische mode 2 (zie afbeelding 12-6.

Wanneer in de grafische mode 2 met sprites wordt gewerkt, wordt behalve van bovenstaande tabellen ook nog gebruik gemaakt van een sprite-info-tabel en een SPRITE$-tabel. Deze tabellen worden behandeld in paragraaf 12.6.



Afb. 12-6 Tabellen in video-RAM na SCREEN 3.

Het beeldscherm is in de grafische mode 2 verdeeld in 768 posities (24 rijen van 32). Iedere beeldschermpositie (van 8 bij 8 beeldpuntjes) is opgedeeld in vier blokjes. Elk blokje bestaat uit 16 beeldpuntjes (4 bij 4 beeldpuntjes). Aan elk blokje kan een kleur worden toegekend. De kleur kan vrij worden gekozen uit de 16 mogelijke kleuren van de MSX-computer. We hebben echter gezien, dat er geen gebruik wordt gemaakt van een kleuren-tabel. Hoe we dan wel te werk gaan, zowel bij het bepalen van de kleuren als bij het positioneren van blokjes, zullen we hierna zien.

De scherm-info-tabel bestaat uit 768 bytes. Voor iedere beeldschermpositie is er dus een byte in de scherm-info-tabel aanwezig. Als we echter de beeldschermposities doortellen, van links naar rechts en van boven naar onder, corresponderen de adressen van de scherm-info-tabel rechtstreeks met de beeldschermposities. De relatie tussen de scherm-info-tabel en de matrix-tabel is echter niet zo rechtstreeks.

Het adres van de matrix-tabel, waarin de kleuren van de vier blokjes, die op het beeldscherm moeten worden geplaatst, staan, wordt bepaald aan de hand van het adres van een code in de scherm-info-tabel en die code zelf. Tabel 12-1 laat deze relatie zien. Bij die tabel hoort enige uitleg. Na het uitvoeren van het statement SCREEN 3 is de scherm-info-tabel gevuld met de volgende waarden (codes):

vier keer na elkaar 0 tot en met 31,
vier keer na elkaar 32 tot en met 63,
vier keer na elkaar 64 tot en met 95,
vier keer na elkaar 96 tot en met 127,
vier keer na elkaar 128 tot en met 159 en
vier keer na elkaar 160 tot en met 191.

Uit tabel 12-2 kunnen we nu lezen welk adres in de matrix-tabel bij iedere code in de scherm-info-tabel behoort. Stel dat we willen weten, waar de kleuren, die bij adres 1 van de scherm-info-tabel, zich binnen de matrix-tabel bevinden. We moeten dan de code die op adres 1 van de scherm-info-tabel staat vermenigvuldigen met 8, en daar de waarde 0 bij optel-

Posities Codes	0 - 31 0 - 31	32 - 63 0 - 31	64 - 95 0 - 31	96 - 127 0 - 31
Posities Codes	128 - 159 32 - 63	160 - 191 32 - 63	192 - 223 32 - 63	224 - 255 32 - 63
Posities Codes	256 - 287 64 - 95	288 - 319 64 - 95	320 - 351 64 - 95	352 - 383 64 - 95
Posities Codes	384 - 415 96 - 127	416 - 447 96 - 127	448 - 479 96 - 127	480 - 511 96 - 127
Posities Codes	512 - 543 128 - 159	544 - 575 128 - 159	576 - 607 128 - 159	608 - 639 128 - 159
Posities Codes	640 - 671 160 - 191	672 - 703 160 - 191	704 - 735 160 - 191	736 - 767 160 - 191
8 x Code	+ 0	+ 2	+ 4	+ 6
	+ 1	+ 3	+ 5	+ 7

Beeldscherm-posities = Adressen Scherm-info tabel

Tabel 12-2 Relatie tussen codes en matrix-tabel posities.

len. Deze waarde 0 vinden we onderaan in de tabel, in de eerste kolom. De code op adres 1 is 1. Als we dit vermenigvuldigen met acht en daar nul bijtellen, krijgen we 1x8+0=8. Met andere woorden op adres 8 staat de kleur van beeldschermpositie 1.

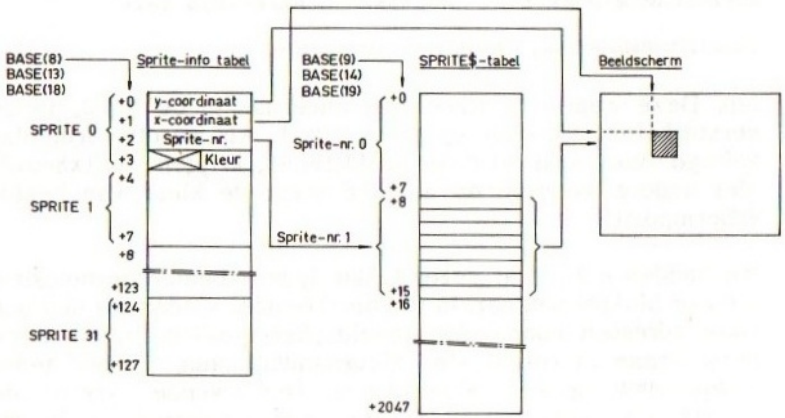
We hadden echter al gezegd, dat iedere beeldschermpositie uit vier blokjes bestaat. In de matrix-tabel vinden we dan ook twee adressen voor iedere beeldschermpositie. In die twee bytes staan in totaal vier kleuraanduidingen, 1 voor ieder blokje (4*4) op het beeldscherm. Het tweede byte in de matrix-tabel volgt direct op het eerstgevonden byte. In ons voorbeeld dus op adres 9 van de matrix-tabel (8+0=8 en 8+1=9).

De lengte van de matrix-tabel kunnen we eenvoudig bepalen. We weten dat de waarde van een code in de scherm-info-tabel mag varieren tussen 0 en 191. Dit zijn 192 verschillende codes. Iedere code komt vier keer voor in de scherm-info-tabel. In totaal zijn er dus 768 codes. Bij iedere code horen twee bytes in de matrix-tabel. Dit betekent dat de matrix-tabel 2x768=1536 bytes lang is.

12.6 Sprites.

Voor het afdrukken van sprites op het beeldscherm maakt de Video Display Processor gebruik van een **sprite-info-tabel** en een **SPRITE$-tabel** (zie afbeelding 12-7). De BASE-adressen van deze tabellen zijn afhankelijk van de mode waarin wordt gewerkt. Het volgende overzicht laat dit zien:

mode	sprite-info-tabel	SPRITE$-tabel
SCREEN 1	BASE(8)	BASE(9)
SCREEN 2	BASE(13)	BASE(14)
SCREEN 3	BASE(18)	BASE(19)



Afb. 12-7 Tabellen in video-RAM voor sprites.

In de sprite-info-tabel staan gegevens van de sprites vermeld. Daar er 32 transparante schermen bestaan en er per transparant scherm maximaal 1 sprite kan worden afgedrukt, is er in de tabel plaats voor de gegevens van 32 sprites. Voor elke sprite zijn vier bytes gereserveerd. Dit betekent dat de tabel een lengte heeft van 4x32=128 bytes. De gegevens voor de sprite die op het transparante scherm 0 wordt afgedrukt, staan op de adressen 0, 1, 2 en 3 van de sprite-info-tabel. De gegevens voor de sprite op het transparante scherm 1 staan op de adressen 4, 5, 6 en 7 van de sprite-info-tabel, etc., etc.

De inhoud van ieder van de vier bytes met sprite informatie is als volgt:

byte 1:
de y-coordinaat van de positie, waar de sprite op het scherm wordt afgedrukt. (de x- en y-coordinaten komen overeen met de positie van het beeldpuntje linksbovenin de sprite.)

byte 2:
de x-coordinaat van de positie, waar de sprite op het scherm wordt afgedrukt.

byte 3:
het sprite-nummer. Het sprite-nummer staat aangegeven in de variabele SPRITE$(X), waarmee de sprite is gedefinieerd.

byte 4:
het kleurnummer. In de vier minst significante bits wordt de kleur, waarin de sprite wordt afgedrukt, aangegeven.

De sprite-nummers in de sprite-info-tabel verwijzen naar de adressen in de SPRITE$-tabel. In deze tabel staan de sprites gedefinieerd. Uitgaande van sprites van 8 bij 8 beeldpuntjes, geldt dat de adressen BASE(X)+0 tot en met BASE(X)+7 dienen voor de systeemvariabele SPRITE$(0). De adressen BASE(X)+8 tot en met BASE(X)+15 dienen voor systeemvariabele SPRITE$(1), enz. In totaal kunnen er 256 sprites van 8 bij 8 beeldpuntjes in de SPRITE$-tabel worden geplaatst. Hiermee wordt de lengte van deze tabel op 256x8=2048 bytes

bepaald.

Gaan we uit van grote sprites (16 bij 16 beeldpuntjes), dan geldt dat de adressen BASE(X)+0 tot en met BASE(X)+31 voor de systeemvariabele SPRITE$(0) zijn gereserveerd. De volgende 32 adressen uit de SPRITE$-tabel zijn dan voor systeemvariabele SPRITE$(1) gereserveerd, enz. Er kunnen maximaal 64 grote sprites worden gedefinieerd. Dit geeft dan weer een lengte van 2048 bytes voor de SPRITE$-tabel (64x32=2048).

12.7 De besturingsregisters van de VDP.

Met behulp van de 9 besturingsregisters van de Video Display Processor (zie afbeelding 12-8) en de inhoud van de tabellen in het video-geheugen produceert de VDP de beelden op het beeldscherm.

VIDEO DISPLAY PROCESSOR

Bit	0	1	2	3	4	5	6	7
Register 0	0	0	0	0	0	0	A	D
1	1	0	E	B	C	R	S	M
2	0	0	0	0	Scherm-info tabel			
3	Kleurtabel							
4	0	0	0	0	0	Matrix-tabel		
5	0	Sprite-info tabel						
6	0	0	0	0	0	SPRITE$-tabel		
7	Voorgrondkleur tekstmode 1				Achtergrondkleur tekstmode 1			
8	T	N	V	Sprite-nummer 5e sprite				

Afb. 12-8 Registers in de Video Display Processor.

De 9 registers bestaan elk uit 8 bits en zijn genummerd van 0 tot en met 8. Voor de registers 0 tot en met 7 mogen zowel lees- als schrijfacties worden uitgevoerd. Register 8, ook wel status-register genoemd, mag alleen worden gelezen. Het lezen van de waarden uit de registers kan worden uitgevoerd met de **systeemvariabele VDP(X).** In deze variabele stelt parameter X het nummer (0-8) van het register voor.

Met het volgende programma worden alle 9 registers uitgelezen en worden de resultaten binair op het beeldscherm afgedrukt.

```
10 CLS
20 FOR I=0 TO 8
30 A$="00000000"+BIN$(VDP(I))
40 PRINT RIGHT$(A$,8)
50 NEXT I
60 PRINT
70 END
```

Door de regelnummers 30 en 40 als volgt te veranderen

```
30 A$="00"+HEX$(VDP(I))
40 PRINT RIGHT$(A$,2)
```

worden de gegevens hexadecimaal afgedrukt, in plaats van binair.

De inhouden van de registers 0 tot en met 7 kunnen ook worden veranderd met de systeemvariabele VDP(X). We kunnen bijvoorbeeld de voorgrondkleur en achtergrondkleur veranderen in respectievelijk zwart (1) en lichtgroen (3). Hiertoe moet de waarde in register 7 worden veranderd in &H13. Het programma waarmee dit wordt gedaan, ziet er als volgt uit:

```
10 SCREEN 0
20 VDP(7)=&H13
30 END
```

De kleuren zijn nu niet door middel van het COLOR-statement veranderd, maar direct in register 7 van de Video Display Processor.

Als laatste zal een overzicht worden gegeven van de inhoud van de verschillende registers in de VDP (zie ook afbeelding 12-8).

ABC	000	tekstmode 2
	100	grafische mode 1
	001	grafische mode 2
	010	tekstmode 1
D	0	geen externe VDP-input
	1	externe VDP-input
E	0	geen VDP-interrupt
	1	VDP-interrupt
R	-	gereserveerd
S	0	sprites van 8*8 pixels
	1	sprites van 16*16 pixels
M	0	sprites niet vergroot
	1	sprites wel vergroot

Scherm-info-tabel het beginadres van de actieve scherm-info-tabel, gedeeld door 1024.

Kleur-tabel het beginadres van de actieve kleur-tabel gedeeld door 64.

Matrix-tabel het beginadres van de actieve matrix-tabel gedeeld door 2048.

Sprite-info-tabel het beginadres van de actieve sprite-info-tabel gedeeld door 128.

SPRITE$-tabel		het beginadres van de actieve SPRITE$-tabel gedeeld door 2048.
Voorgrondkleur		actieve voorgrondkleur in tekstmode 1 (SCREEN 0).
Achtergrondkleur		actieve achtergrondkleur in tekstmode 1 (SCREEN 0).
T	0	statusregister (reg. 8) gelezen of de VDP is extern gereset.
	1	laatste rasterlijn van het scherm is geschreven.
N	0	statusregister (reg. 8) gelezen of de VDP is extern gereset.
	1	twee of meer sprites overlappen elkaar.
V	0	statusregister (reg. 8) gelezen of de VDP is extern gereset.
	1	er bevinden zich meer dan 4 sprites op 1 horizontale lijn (0 - 191).

Opmerking:
In dit laatste geval (V=1) wordt het nummer van het vijfde sprite in de vijf minst significante bits van register 8 geplaatst.

Appendix A

Het programma "Tollenaar" dient als voorbeeld van de mogelijkheden die MSX-BASIC heeft te bieden bij het schrijven van educatieve spelprogramma's. In dit programma zijn zoveel mogelijk verschillende statements opgenomen. Zo is er gebruik gemaakt van de statements CIRCLE, LINE, PAINT en DRAW voor het tekenen van een plaatje op het beeldscherm (regels 660 tot en met 780). Ook wordt er tekst naar het grafisch beeldscherm geschreven. De regels 1350 tot en met 1430 definieren de SPRITES, die in de DATA-regels 1040 tot en met 1330 staan.

Speciale aandacht verdient de routine, waarmee tekst op het grafisch scherm wordt afgedrukt. Indien een andere tekst steeds op dezelfde plaats op het grafisch scherm moet worden afgedrukt, dan zal de oude tekst eerst moeten worden gewist. De eenvoudigste manier om een tekst te wissen is, om met het LINE-statement een rechthoek over de te wissen tekst heen te tekenen en te vullen met de achtergrondkleur. Voorbeelden hiervan vindt u op de regelnummers 2390, 2460, 2700 en 2720.

Om het programma van uw eigen vragen en antwoorden te kunnen voorzien, is het goed het volgende te weten. U kunt de vragen en antwoorden in DATA-regels zetten. Deze DATA-regels dienen dan vanaf regelnummer 3040 te worden geplaatst. Iedere DATA-regel dient het volgende formaat te hebben:

DATA vraag , antwoorden

De vraag moet niet langer zijn dan 32 tekens. Anders past de

vraag niet meer op 1 regel op het beeldscherm. De antwoorden dient u zelf te bedenken. In de voorbeelden ziet u, dat als antwoorden op de vraag: 1+1?, de antwoorden 2, TWEE en twee zijn opgenomen. Met regels 2410 tot en met 2490 wordt het antwoord van de speler geaccepteerd. Het gegeven antwoord wordt dan met regelnummer 2510 gecontroleerd op correctheid. U ziet dat het antwoord als het ware over de toegestane antwoorden wordt geschoven (met de INSTR-functie) om te zien of het juist is. Op deze manier kunnen verschillende antwoorden toch als juist worden geinterpreteerd.

Om niet steeds dezelfde vragen te krijgen, en om de vragen niet steeds in dezelfde volgorde te krijgen, wordt met behulp van regelnummers 2350 en 2360 een willekeurige DATA-regel gekozen. De waarde achter de functie RND(-TIME) dient u aan te passen aan het aantal DATA-regels dat u in het programma heeft staan. In het voorbeeldprogramma staan slechts 8 DATA-regels en dus staat er een 8 achter de functie RND. Heeft u 30 DATA-regels met vragen, dan zult u de waarde 30 achter de RND-functie van regel 2350 moeten invullen. Hoe meer vragen u heeft hoe beter.

Een andere mogelijkheid om het spel te beinvloeden hebt u, door de snelheid van de tollenaar en de speler te beinvloeden. Ook kunt u de tijd die er verstrijkt tussen het opeten van de appel en het weer verschijnen van de appel veranderen.

De snelheid van de tollenaar wordt in variabele ST opgeslagen. De aanvangssnelheid van de speler staat in variabele SS. In regelnummer 790 wordt variabele ST gevuld met de waarde 4. Dit wil zeggen, dat de tollenaar steeds met 4 pixels tegelijk wordt verplaatst. Eveneens op regelnummer 790 wordt de variabele SS geladen. De inhoud van deze variabele wordt echter na verloop van tijd verlaagd. Dit ziet u in regelnummer 2670. Door SS hier, in plaats van met 1, met 2 te verlagen, zal de speler korter te leven hebben. De speler krijgt echter weer meer snelheid na het opeten van de appel. Dat ziet u gebeuren in regelnummer 2190. Door de variabele SS in die regel niet met 2 maar met een hogere waarde te verhogen, kan de speler langer in leven blijven.

Tenslotte kunt u de appel sneller terug laten komen door in regelnummer 2650 de waarde in de vergelijking (IF AW>30) te verlagen. Door die waarde te verhogen zal de appel langer op zich laten wachten.

Met enig experimenteren zult u in staat zijn om snelheden te kiezen, die overeenkomen met de moeilijkheidsgraad van de vragen. Vragen die veel intikwerk vergen, zullen het noodzakelijk maken dat de levensduur van de speler wordt verlengd.

```
100 '*******************************
110 '*      T O L L E N A A R      *
120 '*-----------------------------*
130 '*   copyright: STARK TEXEL.   *
140 '*   auteur:    W.Akkermans.   *
150 '*******************************
160 '
170 '
180 '*******************************
190 '*     ALGEMENE INFORMATIE     *
200 '*******************************
210 SCREEN 0:WIDTH 40:KEY OFF:CLS
220 PRINT "          T O L L E N A A R":
PRINT:PRINT
230 PRINT "Algemene informatie:"
240 PRINT:PRINT "Dit is een spel uit het
 boek:":PRINT:PRINT"          MSX-LEERBOEK
 Deel 2":PRINT:PRINT"geschreven door:"
250 PRINT:PRINT"     W.Akkermans en P. d
en Heijer,
260 PRINT:PRINT"Uitgegeven door":PRINT:P
RINT"          fa. STARK op Texel"
270 PRINT:PRINT:PRINT"Na het bestuderen
van het boek zult u instaat zijn zelf ee
n dergelijk programma te schrijven."
280 FOR I=0 TO 5000:NEXT I
290 CLS
400 '
410 '*******************************
420 '*        SPELREGELS           *
```

```
430 '*******************************
440 PRINT"SPELREGELS:"
450 PRINT:PRINT"Beweeg de speler met de
cursor-control- toetsen van het huisje n
aar de appel."
460 PRINT"Door het verstrijken van de ti
jd vermin-dert de levenskracht (evenals
de snel-  heid) van de speler. Het is da
arom van  belang om geregeld een appel t
e eten.":PRINT
470 PRINT"Tussen het eten van de appels
door, moetde speler echter naar huis ter
ugkeren.  Doet hij dat niet, dan zal het
 eten van de appel geen extra kracht gev
en."
480 PRINT:PRINT"Tussen het huis en de ap
pel waart een   tollenaar rond. Krijgt d
eze de speler tepakken, dan zal hij een
vraag stellen."
490 PRINT:PRINT"Druk op RETURN om door t
e gaan."
500 IF INKEY$<>CHR$(13) GOTO 500
510 CLS
520 PRINT:PRINT"Het goed beantwoorden va
n een vraag zal met een punt worden belo
ond. Bovendien  wordt de speler een voor
sprong gegeven. Bij het fout beantwoorde
n van de vraag  wordt de speler teruggez
et."
530 PRINT:PRINT"Rechtsboven in het beeld
 wordt het be-  haalde aantal punten wee
rgegeven. Ook   wordt daar de resterende
 levenskracht   bijgehouden."
540 PRINT:PRINT"Door zo lang mogelijk in
 leven te blij- ven kunnen zoveel mogeli
jk vragen wordenbeantwoord en dus zoveel
 mogelijk puntenworden behaald."
550 PRINT:PRINT"Druk op RETURN om te beg
innen"
560 IF INKEY$<>CHR$(13) GOTO 560
```

```
600 '
610 '*******************************
620 '*       INITIALISATIE         *
630 '*******************************
640 ON INTERVAL=100 GOSUB 2640
650 SCREEN 2,2:COLOR 11,12,1:CLS
660 CIRCLE (45,79),70,11,,,2
670 PAINT (65,79),11
680 LINE (0,159)-(255,191),1,BF
690 LINE (223,87)-(255,127),9,BF
700 LINE (231,95)-(247,127),3,BF
710 LINE (223,88)-(239,55),8
720 LINE -(255,88),8
730 LINE -(223,88),8
740 PAINT (239,63),8
750 OPEN "grp:" AS #1
760 DRAW "BM176,4":PRINT #1,"PUNTEN"
770 DRAW "BM176,16":PRINT #1,"KRACHT"
780 DRAW "BM222,20":PRINT #1,"....."
790 ST=4:T=1:XT=150:YT=65:SS=3:XS=239:YS
=110:AW=0:LT=0:PU=0:TH=0
800 GOSUB 1340:'DEFINIEER SPRITES
810 PUT SPRITE 3,(37,71),6,6
820 PUT SPRITE 2,(239,110),4,5
830 ON SPRITE GOSUB 2020
840 SPRITE ON
850 INTERVAL ON
900 '
910 '*******************************
920 '*      HOOFDPROGRAMMA         *
930 '*-------------------------*
940 '* Dit is een eideloze lus.  *
950 '*******************************
960 GOSUB 1520:'VERPLAATS TOLLENAAR
970 GOSUB 1820:'VERPLAATS SPELER
980 GOTO 960
1000 '
1010 '*******************************
1020 '*     SPRITE DEFINITIES       *
1030 '*******************************
```

```
1040 'tollenaar1
1050 DATA 01,03,03,83,F9,FF,07,03
1060 DATA 03,07,0E,1C,18,18,38,08
1070 DATA 80,C1,C1,C7,9E,F8,E0,C0
1080 DATA C0,C0,C0,60,60,60,60,70
1090 'TOLLENAAR2
1100 DATA 01,83,83,E3,79,1F,07,03
1110 DATA 03,03,03,06,06,06,06,0E
1120 DATA 80,C0,C0,C1,9F,FF,E0,C0
1130 DATA C0,E0,70,38,18,18,1C,10
1140 'SPELER L>R
1150 DATA 01,03,03,03,01,07,0F,1F
1160 DATA 37,67,07,0E,3C,38,20,10
1170 DATA 80,C0,C0,C0,80,FC,F0,80
1180 DATA 80,80,E0,70,30,30,30,3C
1190 'SPELER R>L
1200 DATA 01,03,03,03,01,3F,0F,01
1210 DATA 01,01,07,0E,0C,0C,0C,3C
1220 DATA 80,C0,C0,C0,80,E0,F0,F8
1230 DATA EC,E6,E0,70,3C,1C,04,08
1240 'SPELER OP EN NEER
1250 DATA 01,03,03,01,07,0B,0B,0B
1260 DATA 0B,0B,02,02,02,02,02,06
1270 DATA 80,C0,C0,80,E0,D0,D0,D0
1280 DATA D0,D0,40,40,40,40,40,60
1290 'APPEL
1300 DATA 18,04,02,01,01,1F,3F,7F
1310 DATA FF,FF,FF,FF,FF,7F,7E,3C
1320 DATA 3C,78,F0,80,7C,FE,FF,FF
1330 DATA FF,FF,FF,FE,FE,FC,FC,78
1340 'DEFINIEREN SPRITES
1350 RESTORE
1360 FOR I=1 TO 6
1370 SP$=""
1380 FOR J=1 TO 32
1390 READ S$
1400 SP$=SP$+CHR$(VAL("&H"+S$))
1410 NEXT J
1420 SPRITE$(I)=SP$
1430 NEXT I
```

```
1500 '
1510 '*******************************
1520 '*    VERPLAATSEN TOLLENAAR    *
1530 '*******************************
1540 IF XS<80 OR XS>205 GOTO 1590
1550 'ACHTERVOLGEN SPELER
1560 IF XS<XT THEN XT=XT-ST ELSE XT=XT+ST
1570 IF YS<YT THEN YT=YT-ST ELSE YT=YT+ST
1580 GOTO 1650
1590 'RANDOM VERPLAATSING
1600 R=INT(RND(-TIME)*4)
1610 IF R=0 THEN XT=XT-ST
1620 IF R=1 THEN XT=XT+ST
1630 IF R=2 THEN YT=YT-ST
1640 IF R=3 THEN YT=YT+ST
1650 'CHECK BEGRENZING
1660 IF XT<80 THEN XT=80
1670 IF XT>205 THEN XT=205
1680 IF YT<0 THEN YT=0
1690 IF YT>143 THEN YT=143
1700 'PLAATS TOLLENAAR
1710 PUT SPRITE 1,(XT,YT),10,1+T
1720 IF T=1 THEN T=0 ELSE T=1
1730 RETURN
1800 '
1810 '*******************************
1820 '*     VERPLAATSEN SPELER      *
1830 '*******************************
1840 S=STICK(0)
1850 IF S=0 THEN RETURN
1860 IF XS>224 THEN TH=0
1870 'JOYSTICK GEACTIVEERD
1880 IF S=6 OR S=7 OR S=8 THEN XS=XS-SS:
RI=1
1890 IF S=2 OR S=3 OR S=4 THEN XS=XS+SS:
RI=0
1900 IF S=1 OR S=2 OR S=8 THEN YS=YS-SS:
RI=2
1910 IF S=4 OR S=5 OR S=6 THEN YS=YS+SS:
RI=2
```

```
1920 'CHECK BEGRENZING
1930 IF XS<0 THEN XS=0
1940 IF XS>239 THEN XS=239
1950 IF YS<0 THEN YS=0
1960 IF YS>143 THEN YS=143
1970 'PLAATS SPELER
1980 PUT SPRITE 2,(XS,YS),4,3+RI:BEEP
1990 RETURN
2000 '
2010 '*******************************
2020 '*     COLLISION DETECTED      *
2030 '*******************************
2040 SPRITE OFF
2050 IF XS<60 GOTO 2190
2060 'MET TOLLENAAR
2070 INTERVAL STOP
2080 V=0:GOSUB 2320:'VRAAGSTELLING
2090 INTERVAL ON
2100 'VERPLAATS SPELER
2110 IF (V=0 AND RI=1) OR (V=1 AND RI=0)
 OR RI=2 THEN XS=XS+20
2120 IF (V=0 AND RI=0) OR (V=1 AND RI=1)
 THEN XS=XS-20
2130 IF XS>225 THEN TH=0
2140 PUT SPRITE 2,(XS,YS),4,3+RI:PLAY("T
255164CEG")
2150 IF PLAY(0)<>0 GOTO 2150
2160 SPRITE ON
2170 RETURN
2180 'MET APPEL
2190 IF TH=0 THEN SS=SS+2
2200 IF SS>6 THEN SS=6
2210 TH=1
2220 PUT SPRITE 3,(37,209),6,6
2230 SPRITE ON
2240 RETURN
2300 '
2310 '*******************************
2320 '*        VRAAGSTELLING        *
2330 '*******************************
```

```
2340 V$="":A$="":B$="":PO=0
2350 R=INT(RND(-TIME)*8):RESTORE 3040
2360 FOR I=0 TO R:READ V$,A$:NEXT I
2370 LINE (0,159)-(255,191),1,BF
2380 DRAW "bm0,163":PRINT #1,V$;
2390 LINE (8*PO,179)-(8*(PO+2),187),1,BF
2400 PSET (8*PO,179),1:PRINT #1,">";
2410 I$=INKEY$: IF I$="" GOTO 2410
2420 IF I$=CHR$(8) AND PO>0 THEN PO=PO-1
:B$=LEFT$(B$,LEN(B$)-1):GOTO 2390
2430 IF I$=CHR$(13) GOTO 2500
2440 IF I$<CHR$(32) GOTO 2410
2450 B$=B$+I$
2460 LINE (8*PO,179)-(8*(PO+1),187),1,BF
2470 PSET (8*PO,179),1:PRINT #1,I$;
2480 PO=PO+1
2490 GOTO 2390
2500 FOR I=1 TO 100:NEXT I
2510 IF INSTR(A$,B$)=0 THEN V=0 ELSE V=1
:PU=PU+1
2520 IF B$="" THEN V=0
2530 LINE (0,163)-(255,191),1,BF
2540 RETURN
2600 '
2610 '*******************************
2620 '*  TIJDSAFHANKELIJKE ACTIES   *
2630 '*******************************
2640 AW=AW+1
2650 IF AW>30 THEN PUT SPRITE 3,(37,71),
6,6:AW=0
2660 LT=LT+1
2670 IF LT>10 THEN SS=SS-1:LT=0
2680 IF SS=0 GOTO 2800
2690 INTERVAL STOP
2700 LINE (224,2)-(255,14),1,BF
2710 PSET (224,4),1:COLOR 15:PRINT #1,PU
2720 LINE (224,18)-(255,22),12,BF
2730 LINE (224,18)-(224+SS*5,22),15,BF
2740 INTERVAL ON
2750 RETURN
```

```
2800 '
2810 '*******************************
2820 '*      EINDRESULTAAT          *
2830 '*******************************
2840 SCREEN 3:DRAW "BM0,0"
2860 PRINT #1," aantal"
2870 PRINT #1," punten"
2880 PRINT #1,:PRINT #1,"  ";PU
2890 FOR I=1 TO 3000:NEXT I
2900 END
3000 '
3010 '*******************************
3020 '*    VRAGEN EN ANTWOORDEN     *
3030 '*******************************
3040 DATA 1+1=?,2TWEEtwee
3050 DATA 2+2=?,4VIERvier
3060 DATA 3+3=?,6ZESzes
3070 DATA 2+3=?,5VIJFVYFvijfvyf
3080 DATA 1+3=?,4vierVIER
3090 DATA 4+1=?,5VIJFVYFvijfvyf
3100 DATA "vul een letter in, in: b.ood"
,rR
3110 DATA "vul een letter in, in: gro.t"
,oOeE
```

Appendix B

In deze appendix zult u twee programma's aantreffen. Beide programma's hebben exact dezelfde functie, doch het eerste is geheel in BASIC geschreven, terwijl in het tweede programma een klein stukje BASIC is vervangen door machinetaal. De uitleg van de programma's zal als volgt zijn.

Bij het eerste programma zal eerst de gebruiksaanwijzing worden gegeven. Deze gebruiksaanwijzing geldt tevens voor het tweede programma. Bij de gebruiksaanwijzing zal op een aantal bijzondere punten in het programma worden gewezen. Bijna al deze bijzonderheden zullen ook in het tweede programma voorkomen. Bij het tweede programma zullen echter een aantal zaken, die speciaal met de machinetaalroutine te maken hebben, verder worden uitgewerkt. Voor de gebruiksaanwijzing bij het tweede programma zult u dus terecht moeten bij de beschrijving bij het eerste programma.

B1 Hexloader (BASIC)

Na het laden en starten van dit programma, zult u als eerste de vraag "Grootte machinecodegebied?" tegenkomen. Indien u van plan bent een machinetaalroutinetje van enkele bytes in te geven, dan kunt u als antwoord bijvoorbeeld 128 ingeven. Het programma kijkt dan met regelnummer 170 wat het hoogste adres is dat door BASIC mag worden gebruikt en verlaagt dat adres met de door u ingegeven waarde (met regelnummer 180).

Nu wordt de variabele BA (Begin Adres) gevuld met het eerste adres dat niet meer door BASIC mag worden gebruikt

(regelnummer 190). In regelnummer 200 ziet u nu een aantal variabelen, die allemaal worden geinitialiseerd. Voor een goed begrip van de werking van het programma volgt hier een korte beschrijving van deze variabelen.

BA	Begin Adres.
BW	Beginadres van het Window.
EW	Eindadres van het Window.
P1	Pointer 1 (cursor positie).
P2	Pointer 2 (hoogste gewijzigde adres).

Bij het opstarten van het programma heeft u zelf het beginadres van de machinecode bepaald. Het machinetaalgebied kan zo groot zijn, dat het niet in zijn geheel op het beeldscherm kan worden afgebeeld. Er is daarom gekozen voor een "window", ofwel een raam, waardoorheen wij in het geheugen kunnen kijken. Dit window laat 128 bytes zien. De bytes die we te zien zullen krijgen, liggen tussen de adressen BW en EW. Zodra we de cursor buiten het window bewegen, zullen de adressen BW en EW met een waarde 32 worden verhoogd of verlaagd.

De cursor is op het scherm te zien. De plaats van de cursor op het scherm komt overeen met een plaats in het geheugen. Die geheugenplaats wordt aangegeven met de variabele P1. Zodra de inhoud van een geheugenadres wordt gewijzigd, wordt er gekeken of de inhoud van P1 groter is dan de inhoud van P2. Is dit inderdaad zo, dan zal P2 gelijk worden gemaakt aan P1. Op die manier geeft P2 aan wat het hoogste geheugenadres is dat is gewijzigd. P2 geeft dus het laatste adres van ons ingegeven machinetaalprogramma aan.

Na het initialiseren van de variabelen, wordt het beeldscherm opgebouwd. We zien drie kolommen met informatie. De middelste kolom bevat per regel (in hexadecimaal formaat) de inhoud van 8 geheugenadressen. De inhoud van deze 8 adressen zijn ook nog eens weergegeven in de rechter kolom, doch nu in ASCII-formaat. Daarbij worden alle niet afdrukbare codes vervangen door een punt. De linker kolom tenslotte bevat de adressen van de eerste op iedere regel afgedrukte

geheugenlocaties.

Onder deze kolommen staat de opdracht: Kies een functie. Dit wil zeggen, dat u met de functietoetsen een functie kunt kiezen. U ziet dat er de volgende functies zijn:

BSAVE	Machinetaal wegschrijven.
BLOAD	Machinetaal inlezen.
EDIT	Hexadecimaal ingeven van machinetaal.
EXECUTE	Uitvoeren van machinetaalroutine.
ERASE	Wissen van een stuk geheugen.

We zullen nu deze functies stuk voor stuk nader gaan bekijken.

BSAVE

Met deze functie wordt het geheugengebied, dat wordt begrensd door de adressen in variabelen BA en P2, onder een door u op te geven naam, naar cassette geschreven.

BLOAD

Met deze functie wordt een machinetaalprogramma, dat op cassette staat, in het geheugen geladen. U kunt zelf de naam van het programma, dat moet worden ingelezen, bepalen. Het programma zal echter worden geladen vanaf het adres, dat in het cassette-label van dat programma staat. Dit wil zeggen, dat het programma op dezelfde plaats terecht komt, waar het ook stond, toen het naar cassette werd geschreven.

EDIT

Dit is de meest uitgebreide routine. U kunt nu uw machinetaalroutine in hexadecimaal formaat intikken. Daarbij hoeft u zich geen zorgen over hoofdletters of kleine letters te maken. Het programma maakt van eventueel ingetikte kleine letters

automatisch hoofdletters. Indien u een niet hexadecimaal cijfer intikt, zal de ingave worden genegeerd.

Behalve het intikken van waarden, kunt u ook de cursor verplaatsen. Hiertoe kunt u de cursor control toetsen gebruiken voor het links, rechts, omhoog en omlaag gaan. Bovendien kunt u de HOME-toets gebruiken. Daarmee zal de cursor naar de linkerbovenhoek van het op het scherm afgebeelde geheugengebied gaan. Ook kunt u de toetsen INS en DEL gebruiken.

Het indrukken van de INS-toets heeft tot gevolg dat u gevraagd wordt hoeveel bytes u wilt tussenvoegen. Indien u bijvoorbeeld 10 bytes wilt tussenvoegen, dan zullen alle bytes vanaf de plaats waar de cursor staat, tot aan het einde van het programma, tien plaatsen naar rechts worden verplaatst. De tien tussengevoegde bytes zullen de waarde 0 krijgen.

Ook na het indrukken van de DEL-toets zult u naar het aantal bytes worden gevraagd. Nu gaat het echter om het aantal bytes dat moet worden verwijderd uit het programma. Geeft u nu bijvoorbeeld in dat u 10 bytes wilt verwijderen, dan zullen de tien bytes vanaf de cursor naar rechts, worden verwijderd. Alle bytes vanaf het eerste byte dat niet meer moest worden verwijderd, tot en met het laatste byte van het programma, zullen 10 plaatsen naar links worden geschoven.

Een heel belangrijke toets is tenslotte de ESC-toets. Alleen door deze toets in te drukken kunt u de EDIT-mode verlaten. U dient de EDIT-mode te verlaten voordat u, met behulp van de functietoetsen, een andere functie kunt kiezen.

EXECUTE

Met deze functie kunt u een, in het geheugen staande, machinetaalroutine starten. U dient daartoe het startadres van die machinetaalroutine op te geven. Indien u alleen de RETURN-toets indrukt, zal het programma aannemen dat de machinetaalroutine start op het adres dat in variabele BA staat.

Deze functie dient u alleen te gebruiken, nadat u de machinecode naar cassette heeft geschreven. Bovendien zal de machinetaalroutine zelfstandig moeten kunnen werken. Na uitvoering moet de routine terugkeren naar BASIC. Is dat niet het geval, dan zal het programma in een "hang-up" situatie komen en dient u de computer uit en weer aan te zetten, voor u weer opnieuw kunt beginnen met het laden van "Hexloader" en machinecode.

ERASE

Met deze functie wordt een deel van het geheugen gewist. Het startadres en de lengte van het geheugengebied dat zal worden gewist is in regelnummer 5010 geprogrammeerd. In regelnummer 5010 staat voor het beginadres BW en voor het laatste te wissen adres de inhoud van variabele P2 (het laatste byte van het machinetaalprogramma). U zou hier zelf andere variabelen kunnen invoeren. Een mogelijkheid zou zijn om het hele machinetaalgebied te wissen (van BA tot en met P2), of om alleen het huidige window te wissen (van BW tot en met BE).

```
10 '*********************************
20 '*       H E X - L O A D E R     *
30 '*-------------------------------*
40 '* Een utility voor het invoeren *
50 '* van hexadecimale machine-code *
60 '*********************************
70 '
80 'Initialiseren
90 '
100 SCREEN 0: WIDTH 40: CLS: KEY OFF
110 BA=0: BW=0: P1=0: P2=0: EA=0: EW=0
120 FOR I=1 TO 10
130 READ K$: KEY I,K$
140 NEXT I
150 DATA BSAVE,BLOAD,EDIT,EXEC,ERASE, ,
, , ,
152 '*********************************
153 '* Opvragen en afbakenen van het  *
```

```
154 '* te bewerken geheugengebied.    *
155 '*********************************
156 '
160 LOCATE 0,2:INPUT "GROOTTE MACHINECOD
EGEBIED";GE
170 EA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
180 CLEAR 500,EA-GE
190 BA=PEEK(&HFC4A)+256*PEEK(&HFC4B)+1
200 BW=BA: P1=BA: P2=BA:EW=BA+128
205 '
210 '*********************************
215 '* Opbouwen van het beeldscherm.  *
220 '*********************************
225 '
260 CLS
270 GOSUB 500:'afdrukken adressen
280 GOSUB 600:'afdrukken geheugeninhoud
285 GOSUB 800:'afdrukken ASCII
290 GOSUB 700:'afdrukken cursor+adres
291 '
292 '*********************************
293 '* Activeren van de functietoetsen*
294 '*********************************
295 '
300 FOR I=1 TO 5:KEY (I) ON: NEXT I
310 ON KEY GOSUB 1000,2000,3000,4000,500
0
320 KEY ON
330 LOCATE 0,19,0:PRINT "kies een functi
e";SPC(20);:GOTO 330
490 '
491 '*********************************
492 '* Subroutine voor het afdrukken  *
493 '* van de adressen van de afge-   *
494 '* drukte geheugenlocaties.       *
495 '*********************************
500 '
510 FOR I=0 TO (EW-BW-1)\8
520 LOCATE 0,I:PRINT RIGHT$("0000"+HEX$(
BW+I*8),4);
```

```
530 NEXT I
540 RETURN
600 '
602 '***********************************
603 '* Afdrukken van de inhoud van de *
604 '* geselecteerde geheugenlocaties *
605 '***********************************
606 '
610 FOR I=0 TO EW-BW-1
620 HP=6+(I*3)MOD24: VP=I\8
630 LOCATE HP,VP
640 PRINT RIGHT$("00"+HEX$(PEEK(BW+I)),2
);
650 NEXT I
660 RETURN
699 '
700 '***********************************
702 '* Afdrukken van het huidige adres*
704 '***********************************
706 '
710 LOCATE 0,17: PRINT "huidig adres is:
";RIGHT$("0000"+HEX$(P1),4);
720 HP=6+((P1-BW)*3)MOD 24:VP=(P1-BW)\8
730 LOCATE HP,VP,1
740 RETURN
795 '
796 '***********************************
797 '*  Afdrukken ASCII-code van de    *
798 '* geselecteerde geheugenlocaties *
799 '***********************************
800 FOR I=0 TO EW-BW-1
810 HP=32+(I MOD 8): VP=I\8
820 LOCATE HP,VP
830 I$=CHR$(PEEK(BW+I))
840 IF I$<" " OR I$>"~" THEN I$="."
850 PRINT I$;
860 NEXT I
870 RETURN
999 '
1000 '**********************************
```

```
1001 '* Subroutine:    B S A V E      *
1002 '*********************************
1003 '
1010 CLS
1020 PRINT "ZET CASSETTERECORDER OP OPNA
ME"
1030 PRINT:PRINT
1040 INPUT "file-naam";F$
1050 BSAVE "cas:"+F$,BA,P2
1055 CLS: GOSUB 500: GOSUB 600: GOSUB 70
0: GOSUB 800
1060 RETURN
2000 '
2001 '*********************************
2002 '* Subroutine:    B L O A D      *
2003 '*********************************
2004 '
2010 CLS
2020 PRINT "ZET CASSETTERECORDER OP AFSP
ELEN"
2030 PRINT:PRINT:INPUT "File-naam";F$
2040 BLOAD "cas:"+F$
2050 CLS:GOSUB 500:GOSUB 600:GOSUB 700:
GOSUB 800
2060 RETURN
3000 '
3001 '*********************************
3002 '* Subroutine:      E D I T      *
3003 '*********************************
3004 '
3005 GOSUB 700
3006 KEY OFF
3007 LOCATE 0,19:PRINT SPC(30);
3008 LOCATE 0,19:PRINT "EDIT-MODE, input
:";
3009 LOCATE HP,VP,1
3010 I$=INKEY$: IF I$="" GOTO 3010
3015 LOCATE 18,19,1
3020 IF I$=CHR$(27) THEN KEY ON:RETURN:'
ESC
```

```
3030 IF I$=CHR$(28) THEN GOSUB 3300:'rec
hts
3040 IF I$=CHR$(29) THEN GOSUB 3400:'lin
ks
3050 IF I$=CHR$(30) THEN GOSUB 3500:'omh
oog
3060 IF I$=CHR$(31) THEN GOSUB 3600:'oml
aag
3070 IF I$=CHR$(11) THEN GOSUB 3700:'HOM
E
3080 IF I$=CHR$(18) THEN GOSUB 3800:'INS
3090 IF I$=CHR$(127) THEN GOSUB 3900:'DE
L
3095 LOCATE 17,19,1
3100 H1=INSTR("0123456789ABCDEFabcdef",I
$)-1
3110 IF H1<0 THEN 3005
3120 IF H1>15 THEN H1=H1-6
3125 PRINT I$;
3130 I$=INKEY$: IF I$="" GOTO 3130
3140 H2=INSTR("0123456789ABCDEFabcdef",I
$)-1
3150 IF H2<0 THEN 3130
3160 IF H2>15 THEN H2=H2-6
3165 PRINT I$;
3170 POKE P1,16*H1+H2
3172 GOSUB 700:PRINT RIGHT$("00"+HEX$(PE
EK(P1)),2);
3180 P1=P1+1
3190 IF P2<P1 THEN P2=P1
3192 IF P1>EW THEN BW=BW+32:EW=EW+32:GOS
UB 500:GOSUB 600:GOSUB 800
3200 GOSUB 700
3210 GOTO 3005
3300 'rechts
3310 P1=P1+1
3320 IF P1=>EW THEN BW=BW+32:EW=EW+32:GO
SUB 500:GOSUB 600:GOSUB 800
3330 GOSUB 700
3340 RETURN
```

```
3400 'links
3405 IF P1=<BA THEN RETURN
3410 P1=P1-1
3420 IF P1<BW THEN BW=BW-32:EW=EW-32:GOS
UB 500: GOSUB 600:GOSUB 800
3430 GOSUB 700
3440 RETURN
3500 'omhoog
3510 IF P1<BA+8 THEN RETURN
3520 P1=P1-8
3530 IF P1<BW AND P1>BA THEN BW=BW-32:EW
=EW-32:GOSUB 500:GOSUB 600:GOSUB 800
3540 GOSUB 700
3550 RETURN
3600 'omlaag
3610 P1=P1+8
3620 IF P1=>EW THEN EW=EW+32:BW=BW+32:GO
SUB 500:GOSUB 600: GOSUB 800
3630 GOSUB 700
3640 RETURN
3700 'HOME
3710 P1=BW:GOSUB 700:RETURN
3800 'INS
3805 IF P1=>P2 THEN RETURN
3807 LOCATE 0,19:PRINT SPC(30);
3808 LOCATE 0,19:INPUT "HOEVEEL BYTES";X
3810 FOR I=P2 TO P1 STEP -1
3820 POKE(I+X),PEEK(I)
3830 NEXT I
3840 FOR I=P1 TO P1+X-1:POKE I,0:NEXT I:
P2=P2+X
3850 GOSUB 600: GOSUB 700:GOSUB 800: RET
URN
3900 'DEL
3910 IF P1=>P2 THEN POKE P1,0:RETURN
3914 LOCATE 0,19:PRINT SPC(30);
3915 LOCATE 0,19:INPUT "HOEVEEL BYTES  "
;X
3920 FOR I=P1 TO P2
3930 POKE I,PEEK(I+X)
```

```
3940 NEXT I
3950 FOR I=P2 TO P2+X-1:POKEI,0:NEXT I:P
2=P2-X
3960 GOSUB 600:GOSUB 700:GOSUB 800:RETUR
N
3989 '
3990 '*********************************
3991 '* Subroutine:   E X E C U T E   *
3992 '*********************************
3993 '
4000 'EXECUTE
4010 CLS
4020 INPUT "startadres";SA
4025 IF SA=0 THEN SA=BA
4030 DEFUSR0=SA
4040 PRINT "m-code gestart op adres";SA;
"  ";USR0(0);
4050 LOCATE 0,19:PRINT "Druk op RETURN o
m door te gaan";
4060 I$=INKEY$:IF I$<>CHR$(13) GOTO 4060
4070 CLS:GOSUB 500:GOSUB 600: GOSUB 700:
GOSUB 800
4080 RETURN
4990 '
4991 '*********************************
4992 '* Subroutine:     E R A S E     *
4993 '*********************************
4994 '
5000 'erase
5010 FOR I=BW TO P2
5030 POKE I,0
5040 NEXT I
5050 GOSUB 600:GOSUB 800
5060 RETURN
```

B2 Hexloader (BASIC + machinetaalroutine).

Het opbouwen van het beeldscherm duurt in BASIC hinderlijk lang. Daarom heb ik een machinetaalroutine geschreven, waarmee hetzelfde wordt gedaan als met de BASIC-subroutines, die in het vorige programma op de regelnummers 500, 600 en 800 beginnen. Het resultaat van deze machinetaalroutine is, dat het beeldscherm nu in ongeveer 1 seconde is opgebouwd, terwijl dat daarvoor bijna 20 seconden duurde.

De machinetaalroutine is in de DATA-statements vanaf regelnummer 6000 opgenomen. Bij het intikken van de machinetaalroutine behoeft u niet alle commentaar in te tikken. U mag het ook als volgt doen:

```
6000 DATA 2A,7C,D0,06,10,C5,CD,3D,D0,3E
6010 DATA 20,CD,A2,00,CD,A2,00,E5,06,08
enz.
```

Dit scheelt u aanzienlijk typewerk en dus tijd. Voor het programma zijn alleen de hexadecimale codes van belang. Het commentaar dat er achter is gezet, is alleen voor u bedoeld. Dit commentaar is zoveel mogelijk in de vorm van assembler source listing gehouden. We zullen er dan ook niet verder op ingaan. Alleen het gebruik van een BIOS-entry point vraagt nog enige uitleg.

Adres "A2" is het BIOS-entry point voor de routine PUTCHAR. Met deze routine kan een teken naar het beeldscherm worden geschreven. Het af te drukken teken dient, bij het aanroepen van de routine, in de accumulator te staan. Als u dus ziet: LD A,0D, gevolgd door een CALL A2, dan wil dit zeggen dat er een Carriage Return (CR) teken naar het beeldscherm wordt gestuurd, ofwel dat de cursor naar het begin van de regel wordt gestuurd.

Het laatste in een DATA-regel gedefinieerde teken moet een asterisk (*) zijn. Dit heeft te maken met de manier waarop we de DATA-regels lezen. Op regelnummer 72 wordt een FOR...NEXT-lus gestart. Deze lus zou eventueel 500 keer

kunnen worden herhaald. Op regelnummer 74 ziet u echter, dat indien de ingelezen waarde in MC$ een "*" is, er naar regelnummer 90 wordt gesprongen. De lus wordt dan verlaten. Deze manier van werken heeft als voordeel, dat we niet hoeven uit te tellen hoeveel DATA-items er zijn.

Regelnummer 76 schrijft de met regelnummer 74 in MC$ gelezen waarde naar het geheugen, door middel van een POKE-statement. Het eerste adres waar een hexadecimale waarde naartoe wordt geschreven is adres &HD000. Ieder volgend DATA-item wordt naar een volgend adres geschreven. Het gebruik van **VAL("&H"+MC$)** in het POKE-statement, maakt het mogelijk om in de DATA-statements de hexadecimale waardes zonder voorvoegsels op te nemen.
Voor de bediening van dit programma kunt u de beschrijving bij het vorige programma lezen.

```
10 '*********************************
20 '*      H E X - L O A D E R      *
30 '*-------------------------------*
40 '* Een utility voor het invoeren *
50 '* van hexadecimale machine-code *
60 '*********************************
70 CLEAR 200,&HCFFF
71 RESTORE 6000
72 FOR I=0 TO 500
74 READ MC$: IF MC$="*" GOTO 90
76 POKE(&HD000+I),VAL("&H"+MC$)
80 NEXT I
90 DEFUSR=&HD000
100 SCREEN 0: WIDTH 40: CLS: KEY OFF
110 BA=0: BW=0: P1=0: P2=0: EA=0: EW=0
115 RESTORE 150
120 FOR I=1 TO 10
130 READ K$: KEY I,K$
140 NEXT I
150 DATA BSAVE,BLOAD,EDIT,EXEC,ERASE, ,
, , ,
152 '*********************************
153 '* Opvragen en afbakenen van het  *
```

```
154 '* te bewerken geheugengebied.    *
155 '*********************************
156 '
160 LOCATE 0,2:INPUT "GROOTTE MACHINECOD
EGEBIED";GE
170 EA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
180 CLEAR 500,EA-GE
190 BA=PEEK(&HFC4A)+256*PEEK(&HFC4B)+1
200 BW=BA: P1=BA: P2=BA:EW=BA+128
205 '
210 '*********************************
215 '* Opbouwen van het beeldscherm.  *
220 '*********************************
225 '
260 CLS
270 GOSUB 500
290 GOSUB 700:'afdrukken cursor+adres
291 '
292 '*********************************
293 '* Activeren van de functietoetsen*
294 '*********************************
295 '
300 FOR I=1 TO 5:KEY (I) ON: NEXT I
310 ON KEY GOSUB 1000,2000,3000,4000,500
0
320 KEY ON
330 LOCATE 0,19,0:PRINT "kies een functi
e";SPC(20);:GOTO 330
490 '
491 '*********************************
492 '* Subroutine voor het genereren  *
493 '* van het startadres van de af te*
494 '* drukken geheugenlocaties.      *
495 '*********************************
500 '
505 LOCATE 0,0
510 MSB=INT(BW/256):LSB=BW-MSB*256
520 POKE &HD07C,LSB:POKE &HD07D,MSB
530 MC=USR(0)
540 RETURN
```

```
699 '
700 '*********************************
702 '* Afdrukken van het huidige adres*
704 '*********************************
706 '
710 LOCATE 0,17: PRINT "huidig adres is:
";RIGHT$("0000"+HEX$(P1),4);
720 HP=6+((P1-BW)*3)MOD 24:VP=(P1-BW)\8
730 LOCATE HP,VP,1
740 RETURN
999 '
1000 '*********************************
1001 '* Subroutine:    B S A V E      *
1002 '*********************************
1003 '
1010 CLS
1020 PRINT "ZET CASSETTERECORDER OP OPNA
ME"
1030 PRINT:PRINT:INPUT "FILE-NAAM";F$
1050 BSAVE "CAS:"+F$,BA,P2
1055 CLS: GOSUB 500: GOSUB 700
1060 RETURN
2000 '
2001 '*********************************
2002 '* Subroutine:    B L O A D      *
2003 '*********************************
2004 '
2010 CLS
2012 FILES:PRINT:PRINT
2015 INPUT "file-naam";F$
2020 PRINT "ZET CASSETTERECORDER OP AFSP
ELEN"
2030 LOCATE 0,19: PRINT"Druk op RETURN o
m te laden";
2035 I$=INKEY$: IF I$<>CHR$(13) GOTO 203
5
2040 BLOAD "CAS:"+F$
2050 CLS:GOSUB 500:GOSUB 700
2060 RETURN
3000 '
```

```
3001 '*********************************
3002 '* Subroutine:      E D I T      *
3003 '*********************************
3004 '
3005 GOSUB 700
3006 KEY OFF
3007 LOCATE 0,19:PRINT SPC(30);
3008 LOCATE 0,19:PRINT "EDIT-MODE, input
:";
3009 LOCATE HP,VP,1
3010 I$=INKEY$: IF I$="" GOTO 3010
3015 LOCATE 18,19,1
3020 IF I$=CHR$(27) THEN KEY ON:RETURN:'
ESC
3030 IF I$=CHR$(28) THEN GOSUB 3300:'rec
hts
3040 IF I$=CHR$(29) THEN GOSUB 3400:'lin
ks
3050 IF I$=CHR$(30) THEN GOSUB 3500:'omh
oog
3060 IF I$=CHR$(31) THEN GOSUB 3600:'oml
aag
3070 IF I$=CHR$(11) THEN GOSUB 3700:'HOM
E
3080 IF I$=CHR$(18) THEN GOSUB 3800:'INS
3090 IF I$=CHR$(127) THEN GOSUB 3900:'DE
L
3095 LOCATE 17,19,1
3100 H1=INSTR("0123456789ABCDEFabcdef",I
$)-1
3110 IF H1<0 THEN 3005
3120 IF H1>15 THEN H1=H1-6
3125 PRINT I$;
3130 I$=INKEY$: IF I$="" GOTO 3130
3140 H2=INSTR("0123456789ABCDEFabcdef",I
$)-1
3150 IF H2<0 THEN 3130
3160 IF H2>15 THEN H2=H2-6
3165 PRINT I$;
3170 POKE P1,16*H1+H2
```

```
3172 GOSUB 700:PRINT RIGHT$("00"+HEX$(PE
EK(P1)),2);
3180 P1=P1+1
3190 IF P2<P1 THEN P2=P1
3192 IF P1>EW THEN BW=BW+32:EW=EW+32: GO
SUB 500
3200 GOSUB 700
3210 GOTO 3005
3300 'rechts
3310 P1=P1+1
3320 IF P1=>EW THEN BW=BW+32:EW=EW+32: G
OSUB 500
3330 GOSUB 700
3340 RETURN
3400 'links
3405 IF P1=<BA THEN RETURN
3410 P1=P1-1
3420 IF P1<BW THEN BW=BW-32:EW=EW-32:GOS
UB 500
3430 GOSUB 700
3440 RETURN
3500 'omhoog
3510 IF P1<BA+8 THEN RETURN
3520 P1=P1-8
3530 IF P1<BW AND P1>BA THEN BW=BW-32:EW
=EW-32: GOSUB 500
3540 GOSUB 700
3550 RETURN
3600 'omlaag
3610 P1=P1+8
3620 IF P1=>EW THEN EW=EW+32:BW=BW+32: G
OSUB 500
3630 GOSUB 700
3640 RETURN
3700 'HOME
3710 P1=BW:GOSUB 700:RETURN
3800 'INS
3805 IF P1=>P2 THEN RETURN
3807 LOCATE 0,19:PRINT SPC(30);
3808 LOCATE 0,19:INPUT "HOEVEEL BYTES";X
```

```
3810 FOR I=P2 TO P1 STEP -1
3820 POKE(I+X),PEEK(I)
3830 NEXT I
3840 FOR I=P1 TO P1+X-1:POKE I,0:NEXT I:
P2=P2+X
3850 GOSUB 500: GOSUB 700:RETURN
3900 'DEL
3910 IF P1=>P2 THEN POKE P1,0:RETURN
3914 LOCATE 0,19:PRINT SPC(30);
3915 LOCATE 0,19:INPUT "HOEVEEL BYTES  "
;X
3920 FOR I=P1 TO P2
3930 POKE I,PEEK(I+X)
3940 NEXT I
3950 FOR I=P2 TO P2+X-1:POKEI,0:NEXT I:P
2=P2-X
3960 GOSUB 500:GOSUB 700:RETURN
3989 '
3990 '*********************************
3991 '* Subroutine:   E X E C U T E   *
3992 '*********************************
3993 '
4000 'EXECUTE
4010 CLS
4020 INPUT "startadres";SA
4030 DEFUSR1=SA
4040 PRINT "m-code gestart op adres";SA;
"  ";USR1(0);
4050 LOCATE 0,19:PRINT "Druk op RETURN o
m door te gaan";
4060 I$=INKEY$:IF I$<>CHR$(13) GOTO 4060
4070 CLS:GOSUB 500: GOSUB 700
4080 RETURN
4990 '
4991 '*********************************
4992 '* Subroutine:     E R A S E     *
4993 '*********************************
4994 '
5000 'erase
5010 FOR I=BW TO P2
```

```
5030 POKE I,0
5040 NEXT I
5050 GOSUB 500:GOSUB 700
5060 RETURN
5995 '
5996 '*********************************
5997 '*     MACHINETAALPROGRAMMA      *
5998 '*********************************
5999 '
6000 DATA 2A,7C,D0:    'LD   HL,(D07C)
6010 DATA 06,10:       'LD   B,#10
6020 DATA C5:          'PUSH BC
6030 DATA CD,3D,D0:    'CALL D03D
6040 DATA 3E,20:       'LD   A,#20
6050 DATA CD,A2,00:    'CALL #A2
6060 DATA CD,A2,00:    'CALL #A2
6070 DATA E5:          'PUSH HL
6080 DATA 06,08:       'LD   B,#08
6090 DATA 7E:          'LD   A,(HL)
6100 DATA CD,46,D0:    'CALL D046
6110 DATA 3E,20:       'LD   A,#20
6120 DATA CD,A2,00:    'CALL #A2
6130 DATA 23:          'INC  HL
6140 DATA 10,F4:       'DJNZ D014
6150 DATA 3E,20:       'LD   A,#20
6160 DATA CD,A2,00:    'CALL #A2
6170 DATA E1:          'POP  HL
6180 DATA 06,08:       'LD   B,#08
6190 DATA 7E:          'LD   A,(HL)
6200 DATA CD,69,D0:    'CALL D069
6210 DATA 23:          'INC  HL
6220 DATA 10,F9:       'DJNZ D028
6230 DATA 3E,0D:       'LD   A,#0D
6240 DATA CD,A2,00:    'CALL #A2
6250 DATA 3E,0A:       'LD   A,#0A
6260 DATA CD,A2,00:    'CALL #A2
6270 DATA C1:          'POP  BC
6280 DATA 10,C9:       'DJNZ D005
6290 DATA C9:          'RET
6300 DATA 7C:          'LD   A,H
```

```
6310 DATA CD,46,D0:    'CALL D046
6320 DATA 7D:          'LD   A,L
6330 DATA CD,46,D0:    'CALL D046
6340 DATA C9:          'RET
6350 DATA E5:          'PUSH HL
6360 DATA F5:          'PUSH AF
6370 DATA E6,F0:       'AND  #F0
6380 DATA 0F:          'RRCA
6390 DATA 0F:          'RRCA
6400 DATA 0F:          'RRCA
6410 DATA 0F:          'RRCA
6420 DATA 21,7E,D0:    'LD   HL,D07E
6430 DATA 5F:          'LD   E,A
6440 DATA 16,00:       'LD   D,0
6450 DATA 19:          'ADD  HL,DE
6460 DATA 7E:          'LD   A,(HL)
6470 DATA CD,A2,00:    'CALL #A2
6480 DATA F1:          'POP  AF
6490 DATA E6,0F:       'AND  #0F
6500 DATA 21,7E,D0:    'LD   HL,D07E
6510 DATA 5F:          'LD   E,A
6520 DATA 16,00:       'LD   D,0
6530 DATA 19:          'ADD  HL,DE
6540 DATA 7E:          'LD   A,(HL)
6550 DATA CD,A2,00:    'CALL #A2
6560 DATA E1:          'POP  HL
6570 DATA C9:          'RET
6580 DATA 7E:          'LD   A,(HL)
6590 DATA FE,20:       'CP   #20
6600 DATA 38,08:       'JR   C,D076
6610 DATA FE,7F:       'CP   #7F
6620 DATA 30,04:       'JR   NC,D076
6630 DATA CD,A2,00:    'CALL #A2
6640 DATA C9:          'RET
6650 DATA 3E,2E:       'LD   A,#2E
6660 DATA CD,A2,00:    'CALL #A2
6670 DATA C9:          'RET
6680 DATA 00:          'NOP
6690 DATA 00:          'NOP
6700 DATA 30,31,32,33:'DEFM "0123"
```

```
6710 DATA 34,35,36,37:'DEFM "4567"
6720 DATA 38,39,41,42:'DEFM "89AB"
6730 DATA 43,44,45,46:'DEFM "CDEF"
6740 DATA *
```

Appendix C

Met het in deze appendix gegeven programma kunt u files, die op cassette staan, afdrukken op een printer. In plaats van files, die zijn aangemaakt met behulp van het PRINT-statement, kunt u natuurlijk ook programma's, die als ASCII-file zijn weggeschreven, inlezen en afdrukken.

Nu zult u zeggen, dat afdrukken van files niet zo moeilijk is. Hierin hebt u natuurlijk volkomen gelijk. Hoe vaak komt het echter niet voor, dat we de printer anders willen instellen dan normaal. In die gevallen moeten we een paar commando-strings naar de printer sturen. Het vervelende is dan echter, dat we steeds weer vergeten welke commando-strings ook al weer voor welke veranderingen gelden. Om dit probleem op te lossen zijn in dit programma een aantal commando-strings opgenomen, die we via een menu kunnen aanroepen.

Om het programma nog wat nuttiger te maken, is ook de mogelijkheid opgenomen, om aan te geven hoeveel regels er per bladzijde dienen te worden afgedrukt, en hoeveel tekens er per regel moeten worden afgedrukt. Bovendien wordt aan het einde van iedere afgedrukte pagina een bladzijdenummer afgedrukt, netjes midden onder de tekst.

Voor alle variabele gegevens, zoals aantal regels per pagina, aantal tekens per regel, aantal regels per inch en aantal tekens per inch, zijn default waarden opgenomen. Deze waarden vindt u op de regelnummers 110 tot en met 180. Met de routine van regels 270 tot en met 470 kunt u op eenvoudige manier een alternatieve waarde kiezen.

Hebt u eenmaal een keuze gemaakt, dan worden de gekozen

waarden omgezet naar commando-strings voor de printer. Dit gebeurt op de regels 540 tot en met 570. In die regels zijn commando-strings voor een MSX-printer opgenomen. Hebt u geen MSX-printer, dan dient u in het bij uw printer behorende manual de juiste waarden op te zoeken. Op regel 540 wordt de commando-string voor het instellen van de "character pitch" op 10 tekens per inch gegeven. Op regelnummer 550 die voor 12 tekens per inch. Op regel 560 wordt de commando-string voor 6 regels per inch gegeven en op regel 570 die voor 8 regels per inch.

Vanaf regel 630 vraagt het programma u om de file-naam van de file die moet worden afgedrukt. Geeft u geen naam op, dan zal de eerste de beste ASCII-file die op de cassette staat, worden ingelezen. Het openen van de file gebeurt in regelnummer 680.

Met de regels vanaf regelnummer 700 worden records uit de file gelezen. Indien het gelezen record langer is dan de opgegeven regellengte, dan zal een deel van het record (met een lengte die gelijk is aan de opgegeven regellengte) worden afgedrukt op een regel, terwijl de rest van het record op een volgende regel of op volgende regels zal worden afgedrukt.

Indien een pagina vol is (variabele R is gelijk aan of groter dan het opgegeven aantal regels per pagina), dan wordt naar de subroutine op regel 890 gesprongen. In deze routine wordt het paginanummer afgedrukt (op regelnummer 910). Bij het schrijven van dit programma is er vanuit gegaan dat er met losse A4-vellen wordt gewerkt. In dat geval moet het afdrukken worden onderbroken om u in de gelegenheid te stellen een nieuw papier in de printer te leggen. De tekst om u daarvan op de hoogte te stellen wordt met de regels 920 tot en met 940 op het scherm gezet. Regel 950 is weer een commando aan de printer. Met dit commando wordt de printer opgedragen een "Form Feed" uit te voeren (een hele pagina op te voeren). Zodra u op de RETURN-toets hebt gedrukt, wordt er weer begonnen met het afdrukken van de volgende pagina.

Als de hele file is afgedrukt, wordt dit gedetecteerd met

regelnummer 820. De file wordt dan afgesloten en er wordt teruggekeerd naar regelnummer 190. Daar wordt het scherm weer schoongemaakt en komt het instelkeuzemenu weer op het scherm.

U ziet, er zijn mogelijk een aantal aanpassingen van dit programma nodig, om het programma geschikt te maken voor uw printer. Indien u kettingbaanformulieren gebruikt kunt u de routine, die op regelnummer 890 begint ook nog sterk vereenvoudigen.

```
10 '************************
20 '*  FILE PRINT UTILITY  *
30 '*----------------------*
40 '*    initialisatie     *
50 '************************
60 '
70 KEY OFF
80 DIM D(3,1)
90 CLEAR 1000
100 T=0
110 D(0,T)=40:'   tekens per regel
120 D(0,T+1)=64:' tekens per regel
130 D(1,T)=24:'   regels per pagina
140 D(1,T+1)=66:' regels per pagina
150 D(2,T)=10:'   tekens per inch
160 D(2,T+1)=12:' tekens per inch
170 D(3,T)=6:'    regels per inch
180 D(3,T+1)=8:'  regels per inch
190 SCREEN 0
200 WIDTH 40
210 CLS
220 '
230 '*******************************
240 '*  Opvragen print parameters  *
250 '*******************************
260 '
270 LOCATE 10,1:PRINT"FILE PRINT UTILITY"
280 LOCATE 0,3:PRINT"Default waarden:"
290 LOCATE 0,5:PRINT"Tekens per regel
```

```
  =";D(0,T)
300 PRINT"Regels per pagina      =";D(1,T)
310 PRINT"Tekens per inch        =";D(2,T)
320 PRINT"Regels per inch        =";D(3,T)
330 LOCATE 0,12:PRINT"Nieuwe waarde?"
340 FOR I=0 TO 3
350 I$="":T=0
360 LOCATE 0,15
370 PRINT "RETURN = niet wijzigen"
380 PRINT "*      = opnieuw beginnen"
390 PRINT:PRINT"anders = volgende default"
400 LOCATE 26,5+I,1
410 I$=INKEY$:IF I$="" GOTO 410
420 IF I$="*" THEN CLS: GOTO 90
430 IF I$=CHR$(13) THEN S(I)=D(I,T):GOTO
 470
440 IF T=0 THEN T=1 ELSE T=0
450 LOCATE 23,5+I:PRINT D(I,T);"    ";
460 GOTO 400
470 NEXT I
480 '
490 '*********************************
500 '* Print Parameters naar Printer *
510 '*********************************
520 '
530 CLS:LOCATE 10,10:PRINT "SCHAKEL PRIN
TER AAN":LOCATE 0,0,0
540 IF S(2)=10 THEN LPRINT CHR$(27);"N";
550 IF S(2)=12 THEN LPRINT CHR$(27);"E";
560 IF S(3)=6 THEN LPRINT CHR$(27);"A";
570 IF S(3)=8 THEN LPRINT CHR$(27);"B";
580 '
590 '*********************************
600 '*  Opvragen filenaam en openen  *
610 '*********************************
620 '
630 LOCATE 5,10:PRINT SPC(80)
640 LOCATE 5,10:PRINT "FILE-NAAM";
650 F$=""
660 LOCATE 16,10,1
```

```
670 INPUT F$: F$="cas:"+F$
675 LOCATE 0,0,0
680 OPEN F$ FOR INPUT AS #1
690 '
700 '*********************************
710 '*  Records lezen en afdrukken   *
720 '*********************************
730 '
740 LINE INPUT #1,R$
750 L=LEN(R$)
760 IF L=0 GOTO 740
770 IF L=<S(0) THEN P$=R$:R$="":GOTO 800
780 P$=LEFT$(R$,S(0))
790 R$=RIGHT$(R$,L-S(0))
800 LPRINT P$:R=R+1
810 IF R=>S(1) THEN GOSUB 890
820 IF EOF(1)<>0 THEN GOSUB 890:CLOSE #1
:GOTO 190
830 GOTO 750
840 '
850 '*********************************
860 '* Printroutine met pagina-check *
870 '*********************************
880 '
890 CLS
900 P=P+1
910 LPRINT:LPRINT TAB(S(0)/2-1);P
920 PRINT "Pagina";P;"afgedrukt."
930 PRINT "Leg nieuw papier in printer."
940 PRINT "Druk op RETURN om door te gaan."
950 LPRINT CHR$(12);
960 IF INKEY$<>CHR$(13) GOTO 960
970 R=0:CLS:RETURN
```

Alfabetische trefwoordenlijst

Nederlandstalige MSX handboeken

MSX BASIC handboek voor iedereen, door A.C.J. Groeneveld
Een kompleet nederlandstalig handboek voor iedere MSX computergebruiker. Dit handboek omvat een volledige behandeling van het MSX-basic in het Nederlands. Het handboek geeft een antwoord op elke vraag die een programmeur, van welke scholing ook, over het MSX-basic zou kunnen stellen. De volledige syntaxisbehandeling rekent af met onzekerheden of een bepaalde schrijfwijze nu wel of niet is toegestaan. De duidelijke beschrijving geeft per sleutelwoord aan, welke de funktie hiervan is. De laatste mogelijk nog aanwezig onduidelijkheden worden vervolgens door de opgenomen, zinvolle voorbeelden weggenomen ISBN 90 6398 1007

MSX ZAKBOEKJE door Wessel Akkermans
Een vlot geschreven naslagwerk na of naast het handboek. U vindt er o.a. in: niet computergerichte tabellen; de MSX-BASIC instructieset; diverse tabellen die het BASIC-programmeren kunnen versnellen; de Z80 instructieset; hardware-gegevens (connectoren) en een aantal programmaatjes ISBN 90 6398 888 5

MSX DISK handboek voor iedereen, door A.C.J. Groeneveld
Handboek voor diskdrivebezitters om naast het grote handboek te gebruiken. Een zeer volledige behandeling van het disk-gebeuren zelf en de specifieke disk kommando's, uitgebreid met voorbeelden, tabellen en overzichten. Het handboek is aangevuld met interessante programma's, waaronder een tekentafelprogramma en een basisprogramma voor basisonderhoud ISBN 90 6398 407 3

MSX PRAKTIJKPROGRAMMA'S door Wessel Akkermans
Praktische programma's met waar nodig eerst een stukje theorie. Erg handig bij het maken van uw programma's. Een greep uit de onderwerpen: priemgetallen; zoeken en sorteren; trefwoordenlijsten; converteren van getallen; enz. ISBN 90 6398 437 5

MSX QUICK DISK handboek voor iedereen, door A.C.J. Groeneveld
Hèt handboek voor iedere QUICK DISK gebruiker. Uitvoerige behandeling van de sleutelwoorden aangevuld met duidelijke voorbeelden met listing ISBN 90 6398 254 2

MSX DOS handboek voor iedereen, door A.C.J. Groeneveld
Dit handboek geeft u op een heldere wijze een totaalbeeld van de mogelijkheden van het MSX-DOS. Ook is dit handboek voorzien van een inleiding op het begrip 'operating system' en dus echt een handboek voor iedereen ISBN 90 6398 674 2

MSX TRUUKS EN TIPS
door A.C.J. Groeneveld
Hoe laat ik de computer een cirkel arceren, hoe tover ik mijn computer om in een elektronisch orgeltje, hoe maak ik een mooie intro voor een spelletje. Allemaal vraagstukken die zich lastig laten programmeren maar die iedere MSX-er toch graag opgelost wil zien.
Dit boekje staat boordevol truuks en tips, allemaal in gewoon MSX basic geschreven. Bladerend door dit boek komt u tot de ontdekking dat er voornamelijk korte maar uiterst krachtige en bijzonder goed bruikbare routines zijn opgenomen. Dit boekje geeft kort maar krachtig een antwoord op al uw programmeervragen.
deel 1 ISBN 90 6398 900 8
deel 2 ISBN 90 6398 340 9

SOFTWARE PLUS IN MSX

INTROTAPE MSX door A.C.J. Groeneveld
Heeft u nog maar net een MSX computer gekocht en wilt u graag weten wat de computer kan en hoe u hem kunt leren programmeren? Deze cassette introduceert MSX op een uiterst vriendelijke en onderwijzende manier. U krijgt instructies hoe u de computer aan moet sluiten en de tape laden. Daarna volgt een demonstratie van de mogelijkheden in MSX, zoals het tekenen van sprites en het werken met de driestemmige toongenerator. Het geheel wordt afgesloten met twee 'les' gedeeltes. In anderhalf à drie uur weet u wat de MSX computer is, wat hij kan, en heeft u haast ongemerkt al wat regels geprogrammeerd.
ISBN 90 6398 148 1

MSX SCRIPT door Ton Weijters
Een menugestuurde nederlandstalige tekstverwerker. Het programma is geschikt om efficiënt grotere of kleinere teksten te bewerken. Paginaindeling (regellengte, paginalengte, marge, inspringen, centreren, enz.) wordt door het programma verzorgd. Dit geldt ook voor de paginatelling, toptitel en het eventueel uitvullen van de regels. Ook corrigeren, zoeken, string-substitutie, blokken tekst verplaatsen, copiëren of verwijderen, onderstrepen en vet zetten, is mogelijk met dit programma.
ISBN 90 6398 189 9

MSX DRAWS door A.C.J. Groeneveld
Een tekenprogramma in MSX basic, waarmee u al binnen 10 minuten uw eerste tekening kunt maken. Draws werkt erg vriendelijk en maakt gebruik van alle grafische mogelijkheden van de MSX computer. U kunt met Draws zowel technisch als creatieve tekeningen maken. Het programma heeft een effectief bereik van ruim 30.000 bij 30.000 puntjes met mogelijkheden als lijnen, cirkels, krommen, inkleuren, vergroten, verkleinen, verschuiven, verdraaien en andere tekeningen invoegen
ISBN 90 6398 754 4

LEERBOEK BASIC deel 2

De serie MSX Leerboeken geeft een komplete kursus MSX-BASIC programmeren, in drie delen.

Deze leerboeken zijn gericht op de beginnende programmeur. De moeilijkheidsgraad van de leerstof wordt dan ook slechts geleidelijk hoger. De gebruikte voorbeelden zijn zo praktisch mogelijk gekozen. Hierdoor kunnen al in een vroeg stadium bruikbare programma's worden gemaakt. Dit zal de lezer/leerling er toe aansporen om verder te gaan. Aan het eind van ieder deel is een groot voorbeeldprogramma opgenomen. Dit programma laat zien waartoe de lezer/leerling na bestuderen van het betreffende leerboek in staat zal zijn.

Bij ieder leerboek is een afzonderlijk „Opdrachten en uitwerkingen" boekje te verkrijgen. In deze boekjes staan, in volgorde van de hoofdstukken uit het leerboek, vragen en opdrachten met antwoorden en uitwerkingen.
Zowel voor gebruik op school als voor individueel gebruik zullen deze boekjes erg nuttig zijn.

ISBN 90 6398 769 2