MSX LEERBOEK DOS DEEL 3

WESSEL AKKERMANS/PIET DEN HEIJER

MSX LEERBOEK DOS DEEL 3

AKKERMANS/DEN HEIJER

MSX DOS Leerboek
deel 3

MSX LEERBOEK DOS

DEEL 3

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. 3
ISBN 90 6398 519 3
SISO 365.3 UDC 681.3
Trefw.: MSX (computer)

1e druk 1985
ISBN 90 6398 519 3

VOORWOORD

Toen wij, nu bijna anderhalf jaar geleden, met deze serie leerboeken begonnen, konden wij nauwelijks vermoeden, hoeveel werk daarin zou gaan zitten. Gelukkig is ons nu reeds duidelijk geworden, dat al dat werk niet voor niets is geweest. Van een groot aantal lezers kregen wij zeer positieve kritiek op de eerste twee delen uit de serie.

Door deze positieve kritiek voelden wij ons voldoende geruggesteund om ook dit derde deel even grondig en even zorgvuldig af te maken als de vorige twee delen. Wij hopen, dat al diegenen, die voor het eerst met een schijveneenheid gaan werken, al hun vragen in dit boek krijgen beantwoord.

Wie een MSX-schijveneenheid koopt, krijgt daar in de meeste gevallen ook het MSXDOS-operating systeem bij. Daar de beschrijving van dat operating systeem nogal eens te wensen over laat, hebben wij ook de bediening en werking van dat systeem in dit boek opgenomen. Het gevolg hiervan is, dat er wederom een lijvig boekwerk is ontstaan, maar vooral, dat er een compleet werk is ontstaan.

Daar de drie leerboeken samen het gehele MSX-BASIC omvatten, hebben wij gemeend er goed aan te doen dit laatste deel te voorzien van een trefwoordenlijst, waarin alle trefwoorden van alle drie de leerboeken voorkomen. Op die manier hopen wij te bewerkstelligen, dat u ieder onderwerp, waarover u meer wilt weten, vanuit een en dezelfde lijst kunt benaderen.

Zoals gezegd, zijn wij bijna anderhalf jaar bezig geweest met het schrijven van deze serie leerboeken. Toen we hieraan

begonnen, konden we nog niet vermoeden, dat er een nieuwe versie van MSX-BASIC zou ontstaan, nog voordat we deze serie leerboeken af hadden. Deze nieuwe versie MSX-BASIC (MSX2) zullen wij, op aandringen van lezers en van onze uitgever, onmiddellijk na verschijnen van dit derde leerboek gaan schrijven. Op die manier hopen wij degenen onder u, die binnenkort gaan overschakelen of die al zijn overgeschakeld naar MSX2, het aanschaffen van een geheel nieuwe serie leerboeken, waarin dan voor u veel overbodige informatie zou staan, te besparen.

Rest ons nu nog u heel veel plezier toe te wensen bij het uitoefenen van uw hobby. Zoals steeds, na het verschijnen van een deel uit deze serie, hebben wij een tevreden doch gespannen gevoel. Tevreden omdat we een flink stuk werk hebben verzet, gespannen omdat we uw reacties nog niet kennen. Hopenlijk zult u ook door dit boek weer een flinke zet in de goede richting krijgen.

Mei 1986

De auteurs

INHOUDSOPGAVE

1 Manipuleren met gegevens

Organiseren, selecteren en sorteren van gegevens uit bestanden, zijn de meest voorkomende taken van een computersysteem. Onder een bestand (file) verstaan we een verzameling van gegevens, die in relatie tot elkaar staan en in principe dezelfde opbouw en organisatievorm hebben. Een bestand kan een adressenlijst zijn, de gegevens van een verzameling grammofoonplaten of recepten, een ledenlijst van een vereniging, enz.

Tijdens het organiseren, selecteren en sorteren, staat het zoeken naar gegevens en het vergelijken van gegevens met elkaar, centraal. Een aantal voorbeelden zijn:

> Het **selecteren** van alle leden uit een ledenbestand van een vereniging, die langer dan 25 jaar lid zijn.
>
> Het **sorteren** van een ledenbestand in alfabetische volgorde, wat namen betreft.
>
> Het opnieuw **organiseren** van een ledenbestand in een bestand bestaande uit actieve leden en een bestand bestaande uit ondersteunende leden.

In de hoofdstukken 2 tot en met 4 zal uitgebreid worden ingegaan op de opbouw van gegevensbestanden en het werken met deze bestanden. In dit hoofdstuk zal voornamelijk worden ingegaan op het **manipuleren** met gegevens als deze in het geheugen staan. Hierna volgt een lijst met functies, die kunnen worden gebruikt voor het manipuleren met gegevens.

LEFT$, RIGHT$, MID$:
Deze functies dienen voor het afscheiden van een gedeelte van een alfanumerieke constante (string). De alfanumerieke constante mag ook een alfanumerieke variabele (bijvoorbeeld A$) of een alfanumerieke uitdrukking (bijvoorbeeld "MSX"+D$) zijn. Voorbeeld: LEFT$("MSX-BASIC",3)="MSX".

LEN:
Deze functie dient voor het bepalen van het aantal tekens van een alfanumerieke constante. Voorbeeld: LEN("MSX-BASIC")=9.

ASC:
Deze functie dient voor het bepalen van de ASCII-waarde (numerieke constante) van een teken. Voorbeeld: ASC("C")=67.

CHR$:
Deze functie bepaalt van een numerieke constante het bijbehorende ASCII-teken. De functies ASC en CHR$ zijn elkaars omgekeerde. Voorbeeld: CHR$(67)="C".

VAL:
Deze functie bepaalt van een alfanumerieke constante (string) de numerieke waarde. Voorbeeld: VAL("20 gulden")=20.

STR$:
Deze functie bepaalt van een numerieke constante de alfanumerieke constante (string). De functies VAL en STR$ zijn elkaars omgekeerde. Voorbeeld: STR$(-147)="-147".

Behalve dat van voorgaande functies gebruik kan worden gemaakt voor het manipuleren met gegevens, kan ook nog gebruikt worden gemaakt van het **statement MID$.** Met het statement MID$ kan een gedeelte van een alfanumerieke constante (string) of variabele worden vervangen door een andere alfanumerieke variabele of constante.

1.1 Afscheiden van gedeelten van strings.

Bij het sorteren en selecteren van gegevens komt het nogal eens voor, dat er eerst gedeelten van strings moeten worden afgescheiden. Wanneer we bijvoorbeeld alle leden van een vereniging willen selecteren, die in het jaar 1961 lid zijn geworden, dan zullen we van elk lid de ingangsdatum van het lidmaatschap moeten onderzoeken. Wanneer we aannemen, dat de datum staat aangegeven in een alfanumerieke variabele als "DD-MM-JJ", dan zullen steeds de twee meest rechtse tekens moeten worden onderzocht. Voor het afscheiden van 1 of meer tekens uit een string, dienen de functies LEFT$, RIGHT$ en MID$.

Met de functie **LEFT$(X$,X)** worden de eerste X tekens van de uitdrukking X$ afgescheiden. Met de functie **RIGHT$(X$,X)** worden de laatste X tekens van de uitdrukking X$ afgescheiden.

X$ kan worden uitgedrukt als:
a. Alfanumerieke constante (string): **"18-04-61".**
b. Alfanumerieke variabele: **DA$.**
c. Alfanumerieke uitdrukking: **"18-04"+JA$.**

Parameter X kan worden uitgedrukt als:
a. Numerieke constante: **3.**
b. Numerieke variabele: **L.**
c. Numerieke uitdrukking: **L+1.**

Parameter X moet een gehele waarde hebben van 0 tot en met 255. Wanneer X groter is dan de lengte van X$, dan zal de totale inhoud van X$ worden gegeven. Wanneer X gelijk is aan 0, dan geeft de functie een lege alfanumerieke uitdrukking.

Voorbeeld:

```
100 REM * VOORBEELD SELECTIE *
110 CLS
120 INPUT "Datum (DD-MM-JJ)";DA$
```

```
130 PRINT
140 PRINT "Dag  :";LEFT$(DA$,2)
150 PRINT "Jaar :";RIGHT$(DA$,2)
160 IF RIGHT$(DA$,2)="61" THEN PRINT "25
 jaar lid."
170 IF INKEY$<>CHR$(13) GOTO 170
180 GOTO 110
```

Een voorbeeld van het invoeren van een datum met het INPUT-statement is:

19-11-57

Er wordt door het programma niet gecontroleerd of de ingevoerde datum een geldige is.

Met de functie **MID$(X$,X,Y)** worden Y opeenvolgende tekens uit X$ afgescheiden, te beginnen bij het X-de teken. X$ kan worden uitgedrukt, zoals voor LEFT$ en RIGHT$ is gesteld. Hetzelfde geldt voor X en voor Y.

De parameters X en Y mogen minimaal 1 en maximaal 255 zijn. Wanneer Y niet is gegeven, of wanneer er rechts van X minder tekens staan dan staan aangegeven in Y, dan worden alle tekens rechts van X afgescheiden. Wanneer X groter is dan de lengte van X$, dan wordt er geen enkel teken afgescheiden.

Voorbeeld:
Voeg aan het vorige programma de volgende programmaregel toe:

```
145 PRINT "Maand:";MID$(DA$,4,2)
```

1.2 Het vervangen van gedeelten van strings.

Met het statement **MID$(X$,X,Y)=Y$** kunnen gedeelten van strings door andere tekens worden vervangen. Wanneer we bijvoorbeeld in een zin het woord "MSX-Basic" willen vervan-

gen door het woord "MSX-BASIC", dan kan dit worden bewerkstelligd door het statement MID$. Een ander voorbeeld is het vervangen van de datum "01-01-85" door "01-01-86".

Met het statement MID$(X$,X,Y)=Y$ worden de tekens in X$, te beginnen vanaf positie X, vervangen door de tekens van Y$. Wanneer parameter Y is gegeven, dan geeft deze aan hoeveel tekens van X$ zullen worden vervangen door tekens van Y$. Wanneer parameter Y niet is gegeven, dan zullen alle tekens van Y$ worden gebruikt, voor zover de lengte van X$ dit toestaat.

Voorbeeld:

```
100 REM * WIJZIGEN VAN EEN WOORD *
110 A$="01-01-65"
120 MID$(A$,7)="86"
130 PRINT A$
140 B$="MSX-Basic"
150 V$="ASIC"
160 MID$(B$,6)=V$
170 PRINT B$
180 END
```

Opmerking:

Het aantal tekens, dat vervangen moet worden, moet gelijk zijn aan het aantal tekens, dat er voor in de plaats komt.

Wanneer in een zin het woord "heeft" moet worden vervangen door "had", kan dit eventueel als volgt worden gedaan.

```
100 A$="Dit heeft geen gevolgen."
110 MID$(A$,6,4)="ad  "
120 PRINT A$
130 END
```

Het statement MID$ is een nuttig en onmisbaar statement binnen tekstverwerkingsprogramma's. Met het statement kunnen woorden geheel of gedeeltelijk worden vervangen.

1.3 Het bepalen van de lengte van een woord of zin.

Wanneer voor een of andere verwerking de lengte van een woord of zin (string) nodig is, kan hiervoor de functie **LEN(X$)** worden gebruikt. De functie LEN(X$) geeft als resultaat de lengte van alfanumerieke variabele X$, uitgedrukt in het aantal tekens. Spaties en niet afdrukbare tekens (codes 1 tot en met 31) worden ook meegeteld.

X$ kan worden uitgedrukt als:
a. Alfanumerieke variabele: **LEN(A$).**
b. Alfanumerieke constante: **LEN("winkel").**
c. Alfanumerieke uitdrukking: **LEN(A$+"BASIC").**

```
100 REM * LENGTE WOORD *
110 CLS
120 A$="MSX":B$="BASIC"
130 PRINT LEN("MSX-BASIC")
140 PRINT LEN(A$+"-"+B$)
150 PRINT LEN(B$)
160 END
```

Met het volgende programma wordt geteld hoeveel maal een bepaald teken in een woord of zin voorkomt. In regelnummer 140 wordt met de functie LEN de lengte van het ingevoerde woord, of de ingevoerde zin, bepaald.

```
100 REM * FUNCTIE LEN(X$) *
110 CLS
120 LINE INPUT "Woord of zin:";W$
130 INPUT "Welk teken tellen";TE$
140 L=LEN(W$)
150 T=0
160 FOR I=1 TO L
170 IF MID$(W$,I,1)<>TE$ GOTO 190
180 T=T+1
190 NEXT I
200 PRINT
210 PRINT "Het woord of de zin"
220 PRINT "bevat";T;"maal het teken ";TE
```

```
$
230 IF INKEY$<>CHR$(13) GOTO 230
240 GOTO 110
```

Met de volgende subroutine wordt gecontroleerd of de ingevoerde tijd (UUMMSS) precies uit 6 tekens bestaat.

```
100 CLS
110 INPUT "Tijd (UUMMSS)";T$
120 IF LEN(T$)<>6 THEN PRINT "Lengte is
geen 6 tekens!":GOTO 110
130 RETURN
```

Het is ook wel eens interessant om te weten hoeveel maal elk teken in een bepaald stuk tekst voorkomt. Het volgende programma geeft hiervoor een mogelijke oplossing.

```
100 REM * TEKENS TELLEN *
105 DIM T(127)
110 CLS
120 LINE INPUT "Zin? ";Z$
130 L=LEN(Z$)
140 FOR I=1 TO L
145 H$=MID$(Z$,I,1)
150 T(ASC(H$))=T(ASC(H$))+1
160 NEXT I
165 PRINT
170 FOR J=32 TO 127
174 IF T(J)=0 GOTO 220
180 PRINT CHR$(J);":";T(J);SPC(2);
190 IF POS(0)>30 THEN PRINT:PRINT
200 IF CSRLIN<23 GOTO 220
210 IF INKEY$<>CHR$(13) GOTO 210
220 NEXT J
230 IF INKEY$<>CHR$(13) GOTO 230
240 GOTO 110
```

Opmerkingen:

Regelnummer 130 zou kunnen vervallen, wanneer regelnummer 140 als volgt werd veranderd:

140 FOR I=1 TO LEN(Z$).

Met behulp van regelnummer 150 worden de verschillende tellers bijgehouden. Zie voor uitleg van de functie ASC paragraaf 1.4.

Met regelnummer 170 worden alle afdrukbare tekens (codes 33 tot en met 127) en de spatie (code 32) aangegeven.

1.4 De functies ASC en CHR$.

Met de functie **ASC(X$)** wordt de ASCII-waarde (decimale waarde 0-255) van een teken bepaald. Voorbeeld:

ASC("M")=77

Wanneer de functie ASC op een reeks tekens wordt toegepast, levert de functie alleen de ASCII-waarde van het eerste teken. Voorbeeld:

ASC("MSX")=77

X$ kan worden uitgedrukt als:

a. Alfanumerieke variabele: **ASC(A$).**
b. Alfanumerieke constante: **ASC("BASIC")**

Daar de ASCII-waarden van de hoofdletters A tot en met Z (codes 65 tot en met 90) en de kleine letters a tot en met z (codes 97 tot en met 122) opeenvolgend zijn, kunnen deze waarden worden gebruikt voor het alfabetisch sorteren van items (bijvoorbeeld namen, adressen, woonplaatsen, etc.). Het verschil tussen de waarden van hoofdletters en kleine letters is voor alle letters constant, namelijk 32. Dat wil zeggen, dat het verschil tussen de hoofdletter A en de kleine letter a 32 is. Dit verschil geldt ook voor de letters B en b, C en c, enz.

Met het volgende programma kunnen we steeds twee woorden met elkaar vergelijken (sorteren). Een voorwaarde is echter, dat de woorden alleen uit hoofdletters of uit kleine letters bestaan.

```
100 REM * VERGELIJKEN VAN WOORDEN *
110 CLS
120 INPUT "Woord 1";W1$
130 INPUT "Woord 2";W2$
135 PRINT
140 IF LEN(W1$)<LEN(W2$) THEN L=LEN(W1$)
 ELSE L=LEN(W2$)
150 FOR I=1 TO L
160 IF ASC(MID$(W1$,I,1))<>ASC(MID$(W2$,
I,1)) GOTO 210
170 NEXT I
180 IF LEN(W1$)=LEN(W2$) THEN PRINT "Woo
rden zijn gelijk.":GOTO 230
190 IF LEN(W1$)>LEN(W2$) THEN PRINT W1$;
" > ";W2$:GOTO 230
200 PRINT W1$;" < ";W2$:GOTO 230
210 IF ASC(MID$(W1$,I,1))>ASC(MID$(W2$,I
,1)) THEN PRINT W1$;" > ";W2$:GOTO 230
220 GOTO 200
230 IF INKEY$<>CHR$(13) GOTO 230
240 GOTO 110
```

In regel 160 wordt teken voor teken met elkaar vergeleken. Met de functies MID$(W1$,I,1) en MID$(W2$,I,1) worden steeds uit beide woorden de tekens op positie I afgescheiden. Wanneer de tekens zijn afgescheiden, worden door middel van de functie ASC de decimale waarden van beide tekens bepaald. Deze waarden worden vervolgens met elkaar vergeleken. Indien de waarden ongelijk zijn, wordt naar regelnummer 210 gesprongen. Indien de waarden gelijk zijn, worden de tekens, die op de volgende positie (I) staan, met elkaar vergeleken.

De functie **CHR$(X)** geeft als resultaat een ASCII-teken of een functie, welke overeenkomt met de waarde van uitdrukking X. Voorbeelden:

CHR$(77)="M"	(teken)
CHR$(13)=RETURN	(functie)

Uitdrukking X kan een numerieke constante, een numerieke variabele of een numerieke uitdrukking zijn. Met PRINT CHR$(65) of PRINT CHR$(&H41) wordt de letter A afgedrukt. Met het volgende programma wordt een ASCII-codetabel afgedrukt.

```
100 REM * ASCII-CODETABEL *
110 WIDTH 40
120 CLS
130 FOR I=32 TO 127
140 PRINT CHR$(I);" =";I;SPC(2)
150 IF POS(0)>28 THEN PRINT:PRINT
160 IF CSRLIN<22 GOTO 190
170 IF INKEY$<>CHR$(13) GOTO 170
180 CLS
190 NEXT I
200 END
```

De functies ASC(X$) en CHR$(X) zijn elkaars inverse. Voorbeeld:

ASC("A")=65 CHR$(65)="A"

Hetzelfde geldt voor de functies VAL(X$) en STR$(X), die in de volgende paragraaf worden behandeld.

1.5 De functies VAL en STR$.

De functie **VAL(X$)** geeft als resultaat een numerieke representatie van de alfanumerieke uitdrukking X$. Wanneer het eerste teken van de uitdrukking X$ geen plusteken (+), minteken (-), &-teken of cijfer is, dan is het resultaat van de functie 0.
Voorbeelden:

```
100 REM * FUNCTIE VAL(X$) *
110 A$="142 gulden":B$="24A":C$="-12"
120 PRINT VAL(A$),VAL(B$)
```

```
130 PRINT VAL(C$),VAL("&H41")
140 PRINT VAL("A12")
150 END
```

De functie VAL wordt toegepast, wanneer er met alfanumerieke waarden berekeningen moeten worden uitgevoerd. In dit geval moeten de alfanumerieke waarden eerst worden omgezet in numerieke waarden.

Het kan zijn, dat een bepaalde waarde, bijvoorbeeld de prijs van een artikel, als alfanumerieke constante (string) is opgeborgen in een gegevensbestand. Voordat er met de prijs gerekend kan worden, zal deze dan eerst moeten worden omgezet in een numerieke waarde.

De functie **STR$(X)** geeft als resultaat een alfanumerieke weergave van de numerieke uitdrukking X.

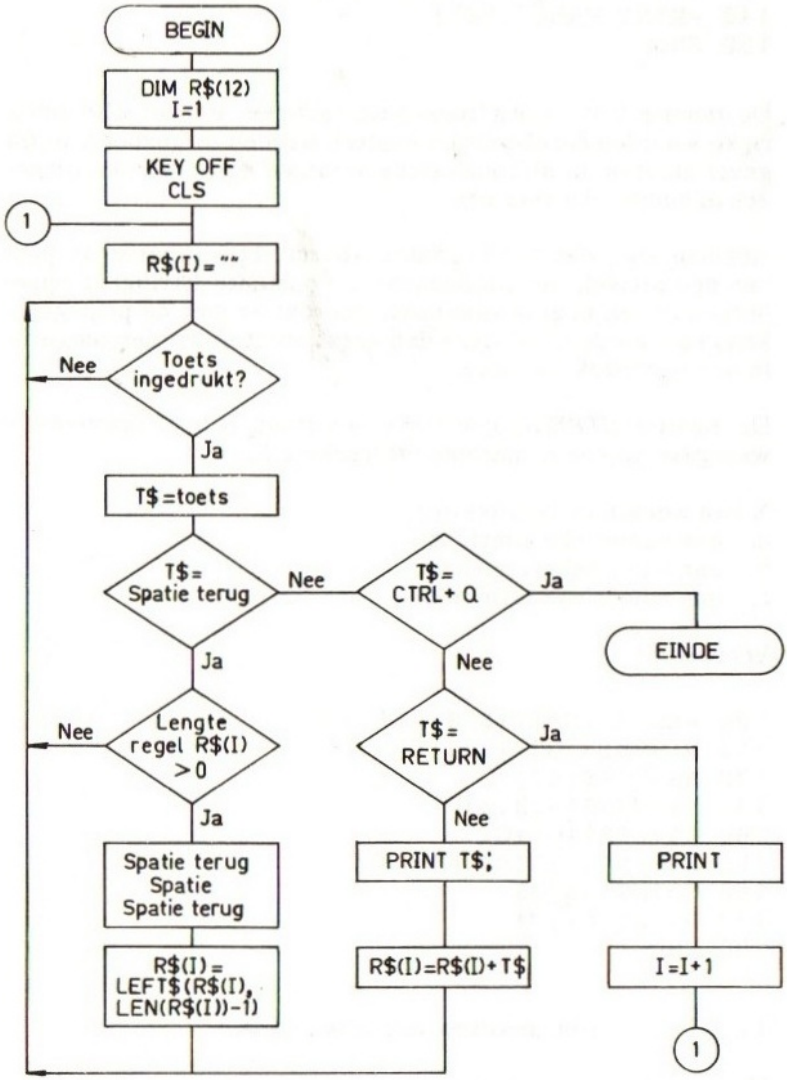
X kan worden uitgedrukt als:
a. een numerieke constante
b. een numerieke variabele
c. een numerieke uitdrukking

Voorbeeld:

```
100 REM * FUNCTIE STR$(X) *
110 G1=58.3:G2=123.15:N=16
120 A$=STR$(47)
130 B$=STR$(-23.5)
140 C$=STR$(G1+G2)
150 D$="NR."+STR$(N)
160 PRINT A$,B$
170 PRINT C$,D$
180 END
```

1.6 Het invoeren van tekstregels.

Met het volgende programma kunnen teksten worden ingevoerd en worden afgedrukt op het beeldscherm. De teksten



Afb. 1-1 Stroomdiagram tekstinvoering.

kunnen tijdens het invoeren worden gecorrigeerd.

Men moet er van uitgaan, dat het programma slechts een demonstratieprogramma is en dient voor het tonen van een aantal toegepaste technieken. Het stroomdiagram in afbeelding 1-1 geeft een duidelijk overzicht van het programma.

Opmerking:

CHR$(8)=	Spatie terug (BS-toets)
CHR$(17)=	CTRL + Q
CHR$(13)=	RETURN-toets

```
100 REM * INVOEREN TEKSTREGELS *
105 DIM R$(12):I=1
110 KEY OFF:CLS
115 R$(I)=""
120 T$=INKEY$:IF T$="" GOTO 120
125 IF T$=CHR$(8) AND LEN(R$(I))>0 THEN
PRINT CHR$(8);" ";CHR$(8);:R$(I)=LEFT$(R
$(I),LEN(R$(I))-1):GOTO 120
130 IF T$=CHR$(8) GOTO 120
135 IF T$=CHR$(17) GOTO 160
140 IF T$=CHR$(13) THEN PRINT:I=I+1:GOTO
 115
145 PRINT T$;
150 R$(I)=R$(I)+T$
155 GOTO 120
160 END
```

1.7 Sorteren.

Het hoofdstuk zal worden afgesloten met een sorteerprogramma. Het programma voert het volgende uit:

a. Plaatst 8 via het toetsenbord ingevoerde woorden in array A$.
b. Plaatst de 8 woorden in alfabetische volgorde.

c. Drukt vervolgens de alfabetisch gerangschikte woorden af op het beeldscherm.

Opbouw programma:

Regelnummers 115–140.
Invoeren door middel van een FOR-NEXT lus van de acht te sorteren woorden.

Regelnummers 145–180.
Zorgen voor het in alfabetische volgorde plaatsen van de acht woorden. Het alfabetisch rangschikken geschiedt in twee geneste FOR-NEXT lussen. Het eigenlijke sorteren geschiedt in regelnummer 165.
Er worden steeds twee opeenvolgende woorden uit array A$ met elkaar vergeleken, te beginnen met de eerste twee woorden uit de array. Als het eerste woord, wat waarde betreft, groter is dan het tweede woord, dan worden de twee woorden in de array van plaats verwisseld. Als het eerste woord een kleinere waarde heeft, dan behouden de twee woorden hun eigen plaats in de array. Deze procedure gaat door, tot het einde van de array is bereikt. Er hebben dan 7 (aantal elementen in de array min 1) vergelijkingen plaatsgevonden. Deze operatie zal 7 maal moeten plaatsvinden. Dit wordt bepaald door de buitenlus.
Wanneer twee woorden van locatie moeten wisselen, dan zal de hulpvariabele H$ moeten worden gebruikt.

Regelnummers 185–215:
Door middel van deze regelnummers vindt het in alfabetische volgorde afdrukken van de inhoud van de array A$ plaats.

Listing van het sorteerprogramma:

```
100 REM * SORTEREN VAN WOORDEN *
105 DIM A$(8)
110 CLS
115 FOR I=1 TO 8
120 INPUT "Woord";W$
125 A$(I)=W$
```

```
130 NEXT I
135 PRINT
140 PRINT "Woorden zijn ingevoerd."
145 PRINT "Sorteren begint.
150 PRINT
155 FOR I=1 TO 7
160 FOR J=1 TO 7
165 IF A$(J)>A$(J+1) THEN H$=A$(J):A$(J)
=A$(J+1):A$(J+1)=H$
170 NEXT J
175 NEXT I
180 PRINT "Einde sorteren."
185 PRINT "Druk RETURN-toets in."
190 IF INKEY$<>CHR$(13) GOTO 190
195 CLS
200 FOR K=1 TO 8
205 PRINT A$(K):PRINT
210 NEXT K
215 PRINT
220 END
```

2 Bestanden

2.1 Media.

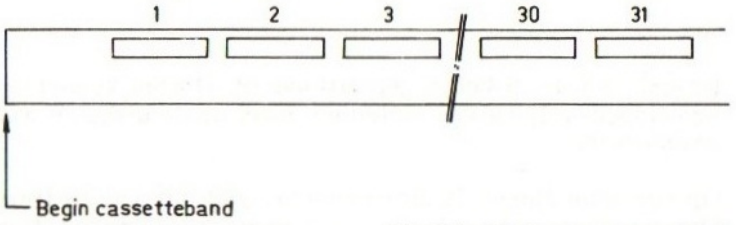
Daar het geheugen in een computer meestal beperkt is tot 64 kBytes en de gegevens in het geheugen verloren gaan bij het uitschakelen van de computer, zullen programma's en gegevensbestanden moeten worden opgeslagen op een permanent geheugenmedium. Men kan hiervoor floppy disks (diskettes) of cassetteband gebruiken. De hobbyist kan het best gebruik maken van de mini-floppy.

2.1.1 Cassetteband.

Cassetteband is tot nu toe het belangrijkste medium voor het vastleggen van gegevens. De cassetteband is voorzien van een magnetiseerbare oxydelaag, waarop informatie met een grote dichtheid kan worden vastgelegd. De gegevens worden in blokken op de cassetteband geschreven. Tussen de blokken bevinden zich ruimtes (inter block gaps), welke tijdens het schrijven (vastleggen van de gegevens) automatisch door het systeem worden gegenereerd. De snelheid tijdens het lezen en schrijven is 1200 baud (default-waarde). Er kan ook gekozen worden voor 2400 baud.

Gegevens worden op cassetteband vastgelegd in de volgorde, zoals deze tijdens de uitvoering van een programma worden aangeboden. Wanneer men na het vastleggen van de gegevens een bepaald gegeven wil terugvinden, dan is de tijdsduur (access-tijd) voor het opzoeken van het gegeven geheel afhankelijk van de plaats waar zich het gegeven op de cassetteband

bevindt. Moet men bijvoorbeeld een gegeven hebben, dat in het dertigste blok staat, dan moeten eerst de 29 blokken, die vooraf gaan aan het dertigste blok worden gelezen, voordat het dertigste blok kan worden gelezen (Afbeelding 2-1). Cassetteband wordt daarom ook wel een "sequentieel access" (in volgorde toegankelijk) medium genoemd.



Afb. 2-1 Blokken op cassetteband.

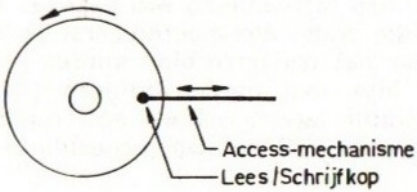
2.1.2 Floppy disk.

Floppy disks, ook wel diskettes genoemd, zijn media, die op een zodanige wijze gegevens kunnen bevatten, dat "random access" (willekeurige of a-selecte toegang) tot de gegevens mogelijk is. De tijd om een bepaald gegeven terug te vinden (access-tijd) op de diskette is in principe voor elk gegeven gelijk. In de praktijk is dit niet geheel waar. De access-tijd varieert, afhankelijk van de plaats van het gegeven op het medium. Dit wordt onder andere veroorzaakt door:

De afstand waarover de kop door het access-mechanisme moet worden verplaatst (spoor naar spoor).
De omwentelingssnelheid van het medium.

Het verschil in access-tijd voor het terugvinden van gegevens op diskette kan ten hoogste een aantal milliseconden bedragen. Dit is een grote verbetering ten opzichte van de vele seconden of minuten, die het verschil op een cassetteband kan uitmaken.

Voor het vastleggen of lezen van gegevens van diskette wordt een lees/schrijfkop gebruikt. Deze kop wordt boven een spoor



Afb. 2-2 Disk Drive.

(track) van de diskette gepositioneerd. Hierna kunnen gegevens (opeenvolging van nullen en enen) op de diskette worden geschreven.

Op een mini-floppy (5,25") bevinden zich 40 sporen (tracks). Om gegevens gemakkelijk te kunnen terugvinden (lezen), moet het systeem weten op welk spoor de gegevens staan en ook waar op het spoor. Hiervoor is elk spoor in 9 sectoren verdeeld. Elke sector kan 512 bytes bevatten. Gegevens kunnen nu worden gelezen door het doorgeven van het spoornummer en het sectornummer, waar de gegevens tijdens de schrijfactie zijn vastgelegd. De capaciteit van een minifloppy is 40 * 9 * 512 = 180 kBytes.

De gebruikers van een MSX-systeem zullen echter niet worden belast met het toekennen van sectoren aan een bestand. Het disk-gebruik wordt namelijk volledig beheerd door een "Disk Operating System", genaamd MSXDOS. Er wordt gewerkt volgens de symbolische bestandsbenaming. Bij het openen van een gegevensbestand zal MSXDOS voor het vastleggen van de gegevens aan het bestand een of meer sectoren toekennen.

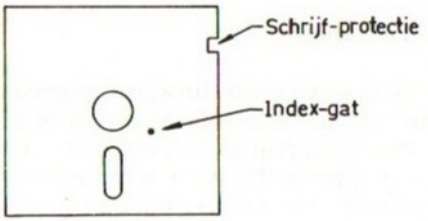
De floppy disk, vrij vertaald slappe schijf, is permanent opgeborgen in een kartonnen beschermhoes. De binnenkant van de hoes is voorzien van een materiaal, dat een kleine wrijvingscoefficient heeft. De disk draait namelijk rond in de hoes. Verder is de hoes voorzien van een uitsparing (opening), zodat de lees/schrijfkop kontakt kan maken met de floppy disk. Floppy disks zijn in tegenstelling tot hard disks gestandaardiseerd en kennen de volgende formaten:

8 inch floppy (geintroduceerd door IBM)
5.25 inch floppy (geintroduceerd door Shugart)
3.5 inch CFD (Cartridge Floppy Disk)
3 inch CFD (Cartridge Floppy Disk)

De floppy is aan de zijkant voorzien van een tweede uitsparing. Deze uitsparing dient voor het beschermen van bestanden, welke op de disk staan. Wanneer de uitsparing wordt afgedicht, kan er niet op de disk worden geschreven, zodoende is alle informatie op de disk beschermd tegen overschrijvingen, die niet gewenst zijn.

In de schijf zit ook nog een gat, welke dient voor het aangeven, waar de eerste sector op een spoor begint. Vanaf dit gat wordt door de software elk spoor in 9 sectoren gedeeld. Het indelen van de sporen op de schijf in sectoren wordt het formateren van de schijf genoemd. Voordat een nieuwe schijf kan worden gebruikt, moet deze eerst door het systeem worden geformateerd. Hiervoor bestaat in MSX-BASIC het commando CALL FORMAT.

Op de floppy disk staat meestal een etiket, waarop de namen van de bestanden, die op de floppy staan, kunnen worden geschreven. Schrijf nooit met een te hard potlood of ballpoint op het etiket, dit zou het disk-oppervlak beschadigen. Verder moet men ook geen magneten bij de floppies houden, dit zal de gegevens op de floppy vernietigen. Floppies mogen ook niet krom worden gebogen of op een verwarming of in de zon worden gelegd. Dit laatste veroorzaakt namelijk ook krombuigen. Verzorg de floppies altijd zeer zorgvuldig, dit voorkomt teleurstellingen.



Afb. 2-3 Diskette.

De mini-floppy boekt in de hobbysfeer groot succes, daar de floppy tegen een lage prijs een massageheugen biedt. De mini-floppy kan worden toegepast voor het opslaan van programma's en gegevensbestanden. Wanneer men bijvoorbeeld een programma op floppy wil opslaan, kan men dit doen door het commando SAVE "A:PROG" te geven. Een routine binnen het MSXDOS-systeem zorgt er dan voor, dat de symbolische programmanaam PROG wordt vertaald in het benodigde aantal sectoren, waar het programma wordt opgeslagen. De naam van het programma en de bijbehorende nummers van de sectoren worden in de directory (inhoudsopgave) geschreven. De inhoudsopgave staat ook op de disk. Het systeem behoeft bij het lezen alleen de inhoudsopgave te raadplegen, om te weten welke sectoren er gelezen moeten worden. Wanneer men bijvoorbeeld het programma PROG wil lezen, doet men dit met het commando LOAD "A:PROG". Het systeem zal de naam PROG in de inhoudsopgave opzoeken, en daar de nummers van de sectoren, waarin het programma PROG staat opgeslagen, vinden.

Voor professioneel gebruik is de capaciteit van een minifloppy wat aan de krappe kant. De capaciteit van een floppy kan worden vergroot door dubbele dichtheid (dual density) te gebruiken. Met dubbele dichtheid kunnen tweemaal zoveel gegevens worden opgeslagen. Hetzelfde resultaat kan worden bereikt door beide zijden van de floppy te gebruiken. Hiervoor is het wel nodig, dat de disk-drive (schijveneenheid) is uitgerust met twee lees/schrijfkoppen. Wordt van beide mogelijkheden gebruik gemaakt, zowel dubbele dichtheid als tweezijdig, dan kan de capaciteit worden vervierdubbeld.

2.2 Bestanden.

Een bestand (file) is een verzameling gegevens (een of meer records) die in relatie tot elkaar staan en in principe dezelfde opbouw en organisatievorm hebben. Een bestand kan een programma, een adressenlijst, een verzameling recepten en verzameling grammofoonplaten, enz. zijn.

Om snel over een bestand te kunnen beschikken, is het nodig, om elk bestand een naam te geven. Verder wordt van een bestand verwacht:

Dat het elke willekeurige lengte kan aannemen.

Dat het groter of kleiner kan worden.

Dat er op een efficiente en snelle wijze toegang kan worden verkregen tot alle informatie, ook als die in het midden van het bestand staat.

Dat er informatie beschikbaar is over het bestand, zoals de lengte, of de gegevens binair of in ASCII zijn uitgecodeerd, enz.

Dat de informatie geen fouten bevat en beschermd is.

Elk bestand kan in grootte varieren. Een adressenlijst kan groter worden, maar ook kleiner. Daar geheugenmedia, zoals diskette en cassetteband, eindig zijn, is de eenvoudigste oplossing elk bestand aan een complete diskette of cassetteband toe te wijzen. Wanneer men dit zo doet, is er voldoende ruimte voor een bestand om te groeien. Maar nu komt het ongemak. Daar er gewoonlijk maar 1 cassetterecorder en/of 1 schijveneenheid op een systeem is aangesloten, gaan we wel erg royaal om met de ruimte. Daar bij deze oplossing slechts een bestand per apparaat beschikbaar is, zijn we zeer beperkt in het aantal bestanden, waartoe de gebruiker gelijktijdig toegang zal kunnen krijgen. Een betere oplossing is, om een bestand een maximale lengte te geven van bijvoorbeeld een halve cassetteband of een halve diskette, zodat er al twee bestanden op een medium kunnen worden vastgelegd.

2.2.1 Indeling van een bestand.

Om het toewijzen van ruimte aan een bestand te vergemakkelijken, structureren de meeste bestandssystemen de beschikbare ruimte in blokken. Deze blokken kunnen gelijk in

lengte zijn, of varieren in lengte. In de meeste gevallen zijn de blokken, om het eenvoudig te houden, gelijk in lengte. Een bestand kan nu uit verschillende blokken bestaan. Een bestand kan groeien, door aan het bestand een of meer blokken toe te wijzen. Een probleem is, hoe kunnen alle blokken, die aan een bestand zijn toegewezen en niet in volgorde staan, in beheer worden genomen. De oplossing hiervoor is een "inhoudsopgave" (directory) te gebruiken. De inhoudsopgave bevat de adressen of verwijzingen (pointers) van alle blokken, die tot eenzelfde bestand behoren.

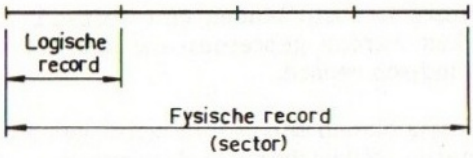
De inhoudsopgave voor het bestand "ADRESSESLIJST" op een diskette kan de volgende informatie bevatten:

Blok 1 staat op sector 2 van spoor 2
Blok 2 staat op sector 3 van spoor 2
Blok 3 staat op sector 4 van spoor 2
enz.

Het is natuurlijk efficient om de inhoudsopgave van een bestand in het geheugen van de computer te laden. Na elk blok moet namelijk de inhoudsopgave weer opnieuw worden geraadpleegd, om te bepalen welk blok het volgende is.

In het voorgaande is reeds meerdere malen het woord "blok" aan de orde geweest. Een blok bevat een of meer records. Een record is een basiseenheid informatie voor een gegevensbestandsprogramma. Deze basiseenheid wordt "logisch record" genoemd. Het is zinvol gebleken, voor het opbergen van gegevens op externe geheugenmedia (cassetteband en diskette) om meerdere logische records samen te voegen tot een "fysiek record". Een fysiek record is die informatie, die met een leesopdracht of schrijfopdracht respectievelijk gelezen of weggeschreven kan worden.

Dit samenvoegen van meerdere logische records tot een fysiek record heet het "blokken" van records. Het uitsplitsen in logische records wordt "ontblokken" genoemd. Een fysiek record komt overeen met een sector (512 bytes).



Afb. 2-4 Een fysisch record.

De gebruiker van bovenomschreven bestandsorganisatie zal echter in principe weinig geinteresseerd zijn in de organisatie. Het enige wat hem interesseert is het feit, dat hij op elk moment logische records moet kunnen lezen, modificeren en wegschrijven. De gebruiker van bestanden krijgt tevens te maken met het "openen" (aanvragen) en "sluiten" (vrijgeven) van een bestand. Hierbij moet informatie worden verstrekt, zoals:

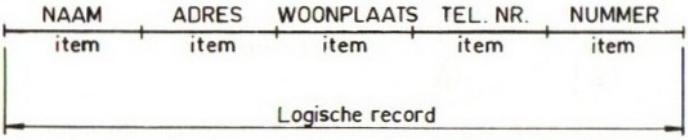
Toe te passen toegangsmethode (directe of sequentiele access).

Het apparaat waarop het bestand staat.

De naam van het bestand.

Grootte van de logische records.

De kleinste eenheid is een alfanumeriek teken en een aantal van deze tekens vormen een item (veld). Een of meerdere items vormen een logisch record (zie afbeelding 2-5). Een verzameling bij elkaar horende records vormt een gegevensbestand.

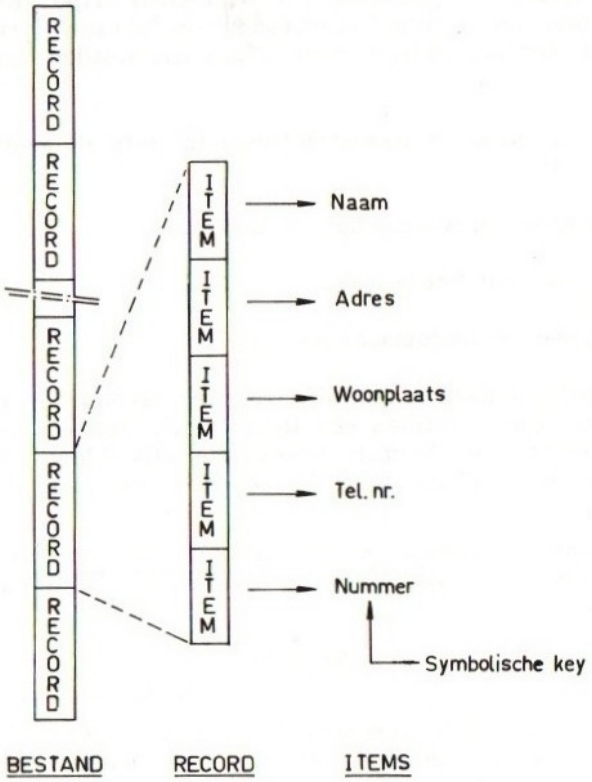


Afb. 2-5 Een logisch record.

Normaliter zijn alle gegevens in hetzelfde bestand logisch betrokken bij hetzelfde onderwerp, bijvoorbeeld personeel, telefoonnummers, artikelen in een magazijn, grammofoonpla-

ten. De kleinste eenheid binnen een bestand, dat door een programma kan worden gepresenteerd aan het computersysteem, is een logisch record.

Een van de items binnen een record wordt gewoonlijk gebruikt als identificatie (bijvoorbeeld salarisnummer, artikelcode). Zo'n item heet dan de "key" (sleutel) van het record. De key maakt het mogelijk een record te identificeren tussen alle andere records van eenzelfde bestand (zie afbeelding 2-6). De items binnen een logisch record kunnen verschillend van lengte zijn.



Afb. 2-6 Een bestand.

2.2.2 Access-methoden.

We zullen in deze paragraaf wat dieper ingaan op de verschillende mogelijkheden, die er bestaan om een bepaald bestand op te bouwen of te lezen. Deze mogelijkheden worden "access-methoden" (toegangsmethoden) genoemd. Om een duidelijk beeld te krijgen van deze access-methoden, moet eerst iets meer worden verteld over bestandsorganisatie. Er bestaan twee soorten bestanden:

1. Sequentieel toegankelijke bestanden.
2. direct (random) toegankelijke bestanden.

Een cassetterecorder kan uitsluitend sequentiele bestanden bevatten. Schijveneenheden daarentegen kunnen zowel sequentiele als direct toegankelijke bestanden bevatten.

Sequentiele bestandsorganisatie

Een sequentieel toegankelijk bestand is een bestand, waarin de logische volgorde van de records overeenkomt met de fysieke volgorde. De cassetterecorder is niet geschikt, om opgeborgen informatie te laden, te modificeren en daarna op dezelfde plaats in het bestand terug te schrijven (hardware probleem). Dit heeft tot gevolg, dat modificatie van records in een bestand het copieren van het gehele bestand vereist; dit wil zeggen, zowel de gemodificeerde als de niet gemodificeerde records.

Voor het verkrijgen van gegevens uit een sequentieel georganiseerd bestand moet de volgende procedure worden aangehouden:

De symbolische key van het record wordt gespecificeerd in het BASIC-programma.

Het volgende blok van het bestand wordt naar het geheugen getransporteerd en gepresenteerd aan het programma.

De gespecificeerde symbolische key wordt vergeleken met de symbolische keys van de records binnen het blok.

Indien het blok niet het gewenste record bevat, wordt het volgende blok gelezen en gepresenteerd aan het programma.

De laatste twee stappen worden net zo lang herhaald, totdat het record is gevonden.

Op diskettes is het ook mogelijk sequentieel georganiseerde bestanden vast te leggen. In dit geval worden de records een voor een gelezen respectievelijk geschreven in de fysieke volgorde. Bij een sequentieel georganiseerd bestand is het niet mogelijk direct toegang te verkrijgen tot het gewenste record. Hiervoor wordt dezelfde procedure toegepast als bij sequentiele bestanden op cassetteband, dat wil zeggen, door het vergelijken van een gespecificeerde symbolische key met de key in de records. Dus, het gehele bestand wordt afgezocht naar de gewenste informatie.

De diskette daarentegen kent de hardware problemen van de cassetterecorder niet. Een groot voordeel van de diskette is, dat elk record is voorzien van een adres. Als men in staat is het adres van een record te bepalen, dan zullen de gegevens vrijwel direct geladen of weggeschreven kunnen worden. Dit heeft ertoe geleid, dat men verschillende mogelijkheden heeft onderzocht om records binnen een bestand direct te adresseren. Hieruit zijn ondermeer de volgende bestandsorganisaties naar voren gekomen:

a. index-sequentiele organisatie.
b. directe (random) organisatie.

Index-sequentiele bestandsorganisatie.

Deze organisatie gaat uit van een sequentieel bestand, dat tevens een tabel bevat met unieke record-informatie en het adres van elk record. Dit laatste is gedaan, om directe ver-

werking van het bestand mogelijk te maken. Men kan een index-sequentiele organisatie vergelijken met de organisatie van een telefoonboek. De sequentiele verwerking is zonder meer duidelijk, de geindiceerde verwerking zal nu nader worden toegelicht. Wil men namelijk een telefoonnummer opzoeken, dan dient bekend te zijn:

1. Woonplaats.
2. Naam.
3. Straat.

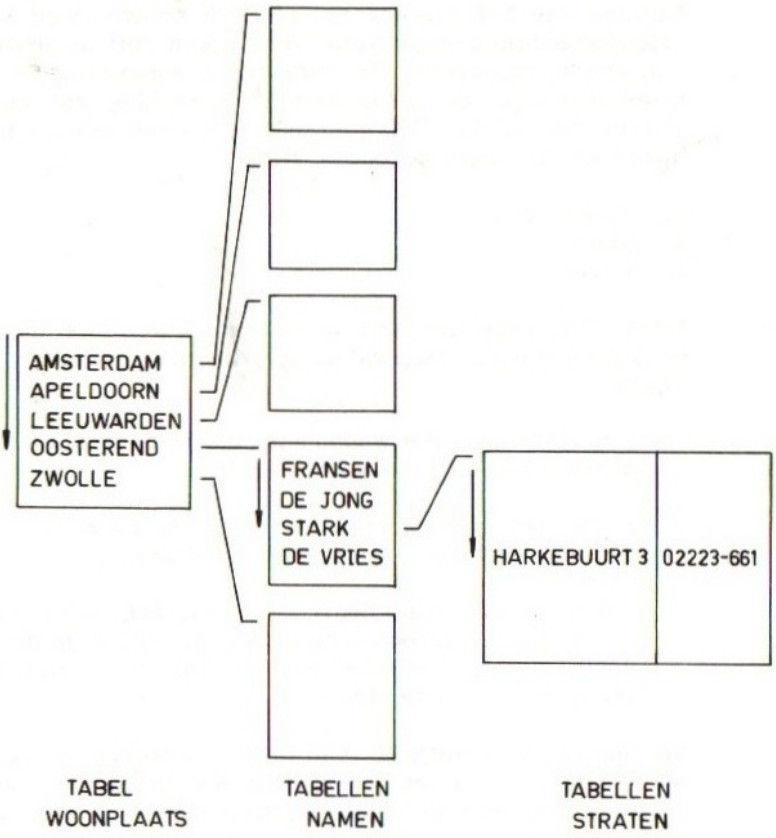
Deze drie gegevens kan men beschouwen als de unieke record-informatie. Men zal nu achtereenvolgens moeten doorlopen:

> Een tabel met alle woonplaatsen, om te bepalen waar de gezochte woonplaats zich bevindt.
>
> In de gezochte woonplaats moet de tabel met namen worden doorlopen, om de naam te vinden.
>
> Indien men de naam heeft gevonden, zal, indien de naam uniek is, het telefoonnummer bekend zijn. Is de naam niet uniek, dan zal de tabel met straten nodig zijn om het telefoonnummer te vinden.

Vooromschreven methode is in feite een steeds nauwkeuriger plaatsbepaling van het telefoonnummer in het telefoonboek, te vergelijken met het gezochte record op een diskette.

Directe bestandsorganisatie.

Met de directe (random) toegangsmethode is het mogelijk direct toegang te krijgen tot een record, met behulp van een adres. Er moet hierbij een relatie bestaan tussen het record en het adres. Dit betekent, dat het creeren van zo'n bestand zeer nauwgezet moet gebeuren. Het bepalen van het adres zal moeten geschieden met behulp van bepaalde parameters (items uit een record).



Afb. 2-7 Een index-sequentieel bestand.

Men kan gebruik maken van een relatief recordnummer. Hierbij wordt met behulp van dit recordnummer het adres bepaald. Het relatieve recordnummer dient dan uit een of meerdere velden (items) uit het record te worden samengesteld. Men kan bijvoorbeeld gebruik maken van een artikelnummer of een persoonsnummer. Een voorwaarde is, dat deze nummers met een vaste waarde, bijvoorbeeld 1, oplopen. Zo kunnen we bijvoorbeeld artikelnummers hebben van 1 tot en met n. Deze nummers dienen gelijktijdig als recordnummers en

worden in het BASIC-programma gebruikt om de betreffende records te selecteren. Vanuit het recordnummer wordt door het systeem het adres van het record bepaald.

3 Sequentiele bestanden

In hoofdstuk 2 hebben we geleerd, dat in MSX-BASIC twee typen gegevensbestanden kunnen worden onderscheiden, namelijk:

1. Sequentiele gegevensbestanden
2. Random (directe) gegevensbestanden

In gegevensbestanden kunnen we onder andere gegevens van postzegels, grammofoonplaten, recepten, enz. vastleggen. Verder is het ook mogelijk adressenbestanden en verzendlijsten (mailing lists) te creeren.

Buiten gegevensbestanden kennen we ook nog programmabestanden. Een programmabestand bestaat uit een aantal bij elkaar behorende programmaregels, die tezamen een programma vormen. Een programmabestand wordt met het commando SAVE op disk geschreven en met het commando LOAD van disk gelezen en in het geheugen geplaatst. Ook is het mogelijk door middel van het commando KILL een programmabestand van disk te wissen. Bovengenoemde commando's worden in hoofdstuk 5 uitgebreider behandeld. In dit hoofdstuk worden alleen sequentiele gegevensbestanden op disk behandeld.

Sequentiele gegevensbestanden zijn eenvoudiger te creeren dan random bestanden, echter het verkrijgen van toegang tot de gegevens van een sequentieel bestand is niet zo eenvoudig. Dit houdt vooral verband met de lange tijdsduur, die nodig is om over de gegevens uit het bestand te kunnen beschikken.

De gegevens, welke in een sequentieel bestand worden ge-

plaatst (geschreven), worden item na item (sequentieel) opgeslagen in de volgorde waarin de gegevens naar disk worden gestuurd. Het lezen van de gegevens van disk geschiedt in dezelfde volgorde, als waarin de gegevens tijdens het eerdere wegschrijven op disk werden opgeslagen.

De statements en functies, die voor sequentiele bestanden in programma's worden toegepast, zijn:

```
OPEN        PRINT#          CLOSE#      EOF
            INPUT#                      LOC
            LINE INPUT#
```

De statements en functies worden in de volgende paragrafen uitgebreid behandeld.

3.1 Inleiding tot sequentiele gegevensbestanden.

Voor het creeren van een sequentieel gegevensbestand op disk en het vervolgens toegankelijk maken van de gegevens in het bestand, dienen in het programma de volgende stappen te worden gemaakt.

1.
Openen van het bestand in de OUTPUT-mode:

```
OPEN "A:ADRES" FOR OUTPUT AS #1
```

2.
Schrijven van gegevens naar het geopende bestand:

```
PRINT #1,A$;",";B$;",";C$
PRINT #1,A;B;C
PRINT #1,USING "###.##";A,B,C
```

3.
Voor het toegankelijk maken van de gegevens in het bestand moet het bestand eerst worden gesloten, en vervolgens worden geopend voor de INPUT-mode:

```
CLOSE #1
OPEN "A:ADRES" FOR INPUT AS #1
```

4.
Lezen van gegevens uit het geopende bestand:

```
INPUT #1,A$,B$,C$
INPUT #1,A,B,C
LINE INPUT #1,A$
```

Als voorbeeld van sequentiele bestanden zullen we een programma behandelen, waarmee een "adressenbestand" kan worden gecreeerd, uitgebreid en gewijzigd. Verder kunnen natuurlijk door middel van het programma adressen worden opgevraagd. Het programma bestaat uit de volgende modulen:

Het hoofdprogramma voor het afdrukken van het menu en het kiezen van de gewenste subroutine.

Subroutine voor het creeren van het "adressenbestand".

Subroutine voor het toevoegen van nieuwe adressen aan het bestaande bestand.

Subroutine voor het opvragen (lezen) van adressen uit het bestand.

Subroutine voor het wijzigen of verwijderen van adressen uit het bestand.

Het hoofdprogramma en de subroutines zullen in de volgende paragrafen in de hiervoor beschreven volgorde worden behandeld.

3.2 Hoofdprogramma "Adressenbestand".

Het hoofdprogramma voor het "adressenbestand" bestaat uit een menu, het maken van een keuze uit het menu en het vervolgens aanroepen van de gekozen subroutine. De listing van het hoofdprogramma is als volgt:

```
100 REM * ADRESSENBESTAND *
105 WIDTH 40:KEY OFF
110 MAXFILES=2:CLS
115 PRINT TAB(3);"MENU ADRESSENBESTAND"
120 PRINT:PRINT
125 PRINT "1. Creeren van het bestand":P
RINT
130 PRINT "2. Toevoegen van nieuwe adres
sen":PRINT
135 PRINT "3. Opvragen van adressen":PRI
NT
140 PRINT "4. Wijzigen of verwijderen ad
ressen":PRINT
145 PRINT "5. Programma beeindiging"
150 PRINT:PRINT
155 INPUT "Welke keuze";K
160 IF K<1 OR K>5 THEN PRINT "Verkeerde
keuze!":GOTO 155
165 IF K=5 THEN END
170 ON K GOSUB 200,300,400,550
175 GOTO 110
```

3.3 Creeren van een bestand.

De volgende subroutine dient voor het creeren van het "adressenbestand". De gegevens worden via het toetsenbord ingevoerd. Elk adres, dat via het toetsenbord wordt ingevoerd, bestaat uit de volgende vijf elementen (velden):

1. Naam
2. Adres
3. Postcode
4. Woonplaats
5. Telefoonnummer

De listing van deze subroutine is als volgt:

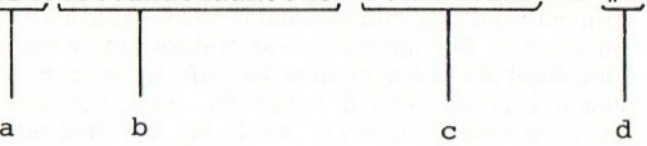
```
200 REM * CREEREN ADRESSENBESTAND *
205 OPEN "A:ADRBES" FOR OUTPUT AS #1
```

```
210 F=1
215 CLS
220 IF F=1 THEN PRINT "CREEREN ADRESSENB
ESTAND":PRINT
225 IF F=2 THEN PRINT "TOEVOEGEN NIEUWE
ADRESSEN":PRINT
230 INPUT "Naam        ";N$
235 INPUT "Adres       ";A$
240 INPUT "Postcode    ";P$
245 INPUT "Woonplaats  ";W$
250 INPUT "Telefoonnr.";T$
255 PRINT
260 INPUT "Gegevens juist (J/N)";J$
265 IF J$="N" OR J$="n" THEN PRINT "Voer
 gegevens opnieuw in!:PRINT:GOTO 230
270 PRINT #1,N$;",";A$;",";P$;",";W$;","
;T$
275 INPUT "Laatste adres (J/N)";J$
280 IF J$="N" OR J$="n" GOTO 215
285 CLOSE #1
290 RETURN
```

In de subroutine komen twee nieuwe statements voor, namelijk het OPEN-statement op regelnummer 205 en het PRINT-statement op regelnummer 270. Om toegang te kunnen verkrijgen tot de gegevens van een bestand, moet het bestand eerst worden geopend met het statement OPEN. Dit geldt ook, wanneer een nieuw bestand moet worden gecreeerd, hetgeen in onze subroutine het geval is.

We zullen in dit leerboek het statement OPEN alleen behandelen voor bestanden (sequentiele en random) op disk. Voor sequentiele gegevensbestanden op disk kunnen we voor het statement OPEN drie formaten onderscheiden.

```
OPEN "disk:bestandsnaam.ext" FOR OUTPUT AS #X
OPEN "disk:bestandsnaam.ext" FOR INPUT  AS #X
OPEN "disk:bestandsnaam.ext" FOR APPEND AS #X
```



In het statement OPEN betekent:

a.
Naam van de disk, waarop het bestand staat of komt te staan. In het geval van disk is dit de letter A of B. De letters geven de schijveneenheid A of B aan.

b.
Naam van het bestand. De naam mag uit maximaal 11 tekens bestaan. De eerste 8 tekens vormen dan de naam van het bestand én de daaropvolgende 3 tekens de bestandsnaamuitbreiding, die van de bestandsnaam wordt gescheiden door een punt. Ook mag men zelf de naam (van maximaal 8 tekens), gevolgd door een punt en de bestandsnaamuitbreiding (van maximaal 3 tekens) ingeven. Worden teveel tekens ingegeven, dan zal de computer reageren met het afdrukken van een foutboodschap.

c.
Geeft aan hoe het bestand gaat worden gebruikt:

FOR INPUT (bestand naar computer).
Gegevens worden van schijf gelezen en in het geheugen geladen (sequentieel lezen).

FOR OUTPUT (computer naar bestand).
Gegevens worden vanuit het geheugen op schijf geschreven (sequentieel schrijven).

FOR APPEND (computer naar bestand).
Nieuwe gegevens worden toegevoegd aan een bestaand bestand. De gegevens worden aan de achterkant (einde) van het

bestand toegevoegd.

d.
Een eenmaal geopend bestand is verder toegankelijk onder het nummer X. Dit betekent, dat tijdens het lezen of schrijven niet meer de bestandsnaam behoeft te worden gebruikt. Het nummer staat bekend onder de naam bestandsnummer of kanaalnummer. Normaal wordt bij het inschakelen van de computer het aantal bestanden, dat tegelijkertijd kan zijn geopend, op 1 gesteld. Wanneer men meerdere bestanden tegelijkertijd wil openen, dan kan dit door de waarde van de systeemvariabele MAXFILES te veranderen (default-waarde is 1). De waarde mag niet kleiner zijn dan 0 en niet groter dan 15. Het nummer geeft het aantal bestanden aan, dat tegelijkertijd geopend kan zijn. Een bestandsnummer kan minimaal 1 zijn en maximaal de waarde die aan MAXFILES is toegekend. De waarde 0 van MAXFILES betekent dus, dat er geen bestanden kunnen worden geopend.

Het is aan te raden, om het statement MAXFILES slechts eenmaal te gebruiken, waarbij dan direct het maximum aantal bestanden dat tegelijkertijd zal worden gebruikt, in enig deel van het programma, wordt gedefinieerd. Zijn er dus verschillende routines, waarbij in de ene routine slechts een bestand geopend is, terwijl in een andere routine meerdere bestanden tegelijk zijn geopend, dan dient het aantal bestanden dat maximaal tegelijkertijd wordt gebruikt, ergens aan het begin van het programma te worden vermeld in het MAXFILES-statement.

Voor het creeren van het "adressenbestand" moet het bestand worden geopend voor uitvoer (FOR OUTPUT), dat wil zeggen, dat het gegevenstransport plaats vindt van de computer naar het bestand (op de schijf).

Opmerking:
Men moet altijd voorzichtig zijn met het openen van een bestand voor OUTPUT. Het resultaat is namelijk, dat een nieuw bestand wordt toegewezen op schijf, en de ruimte hiervoor wordt schoongemaakt. Dit betekent, dat als men

een bestaand bestand opent voor OUTPUT, de informatie van het bestand als verloren moet worden beschouwd.

Het wegschrijven van de gegevens vanuit het geheugen naar het bestand geschiedt met een PRINT-statement. In het PRINT-statement behoeft alleen het bestandsnummer te worden genoemd. De computer weet dan precies voor welke schijf (A of B) en welk bestand die gegevens bestemd zijn. Dit is reeds bekend gemaakt in het OPEN-statement.

In het PRINT-statement staan, door puntkomma's van elkaar gescheiden, de numerieke en/of alfanumerieke variabelen, die naar het bestand zullen worden geschreven. Het algemene formaat van het statement is:

```
PRINT #bestandsnummer,lijst met variabelen
```

Alfanumerieke variabelen moeten in de lijst door puntkomma's van elkaar worden gescheiden. Om de alfanumerieke variabelen correct op de schijf te formateren, moeten afbakeningstekens worden gebruikt. Voorbeeld:

```
N$="NAAM"
A$="ADRES"
P$="POSTCODE"
```

Het statement PRINT #1,N$;A$;P$ schrijft dan naar disk: NAAMADRESPOSTCODE. Omdat er geen afbakeningstekens zijn gebruikt, kan hetgeen op schijf staat niet meer worden ingevoerd als drie aparte strings. De oplossing voor dit probleem is, het tussenvoegen van afbakeningstekens (komma's) in het PRINT-statement. Het statement voor voorgaand probleem gaat er dan als volgt uitzien:

```
PRINT #1,N$;",";A$;",";P$
```

Het patroon dat naar schijf wordt geschreven gaat er dan als volgt uitzien: NAAM,ADRES,POSTCODE. Dit kan dan met het statement INPUT #1,N$,A$,P$ worden teruggelezen van schijf en worden toegekend aan respectievelijk de variabelen N$, A$ en P$.

In het PRINT-statement mogen ook alfanumerieke constanten (strings) worden opgenomen. Bijvoorbeeld:

```
PRINT #1,"alfanumerieke constante"
```

Ook uitdrukkingen zoals STRING$(25,"-") zijn in een PRINT-statement mogelijk. Het statement gaat er dan als volgt uit zien:

```
PRINT #1,STRING$(25,"-")
```

Voordat de subroutine wordt beeindigd, wordt met het statement CLOSE eerst nog het bestand afgesloten. Achter CLOSE staat het bestandsnummer vermeld, van het bestand, dat moet worden afgesloten. Wanneer achter CLOSE niets staat, dan zullen alle bestanden, die zijn geopend op het moment van uitvoering van het CLOSE-statement, worden afgesloten. Dit betekent, dat vanaf dat moment bestanden niet meer toegankelijk zijn. Hiervoor zullen ze dan eerst weer moeten worden geopend met het statement OPEN.

Wanneer bijvoorbeeld de bestanden met de nummers 1 en 3 moeten worden afgesloten, dan gaat het statement er als volgt uitzien:

```
CLOSE #1,#3
```

3.4 Uitbreiden van een bestaand bestand.

Wanneer een bestaand sequentieel gegevensbestand moet worden aangevuld met nieuwe gegevens, dan moet het bestand op disk worden geopend voor de mode FOR APPEND.

Voor het toevoegen van nieuwe adressen aan ons sequentiele gegevensbestand ADRBES dient de volgende subroutine:

```
300 REM * TOEVOEGEN NIEUWE ADRESSEN *
305 OPEN "A:ADRBES" FOR APPEND AS #1
310 F=2
```

```
315 GOTO 215
```

In de subroutine wordt voor het invoeren van gegevens naar subroutine "Creeren adressenbestand" gesprongen. Het gedeelte voor het invoeren van gegevens en het wegschrijven naar disk is namelijk voor beide subroutines volkomen gelijk, vandaar dat het maar in een subroutine voorkomt.

3.5 Opvragen (lezen) van gegevens uit een bestand.

Het is natuurlijk de bedoeling, dat we op een zeker moment een adres uit ons bestand ADRBES gaan lezen. Het bestand zal dan eerst moeten worden geopend voor de mode FOR INPUT. Het formaat van het OPEN-statement gaat er dan als volgt uitzien:

```
OPEN "A:ADRBES" FOR INPUT AS #1
```

De gegevens (items) uit een sequentieel bestand worden met het statement INPUT van disk gelezen en toegekend aan programma-variabelen in het geheugen. In het statement behoeft alleen maar het bestandsnummer te worden genoemd. Het formaat van het statement gaat er dan als volgt uitzien:

```
INPUT #1,N$,A$,P$,W$,T$
```

In de volgende subroutine komt ook de functie EOF (End Of File) voor. De functie EOF controleert of het einde van een sequentieel gegevensbestand is bereikt. De functie geeft als resultaat -1 (waar), indien het einde van het bestand is bereikt. Indien het einde nog niet is bereikt, zal het resultaat van de functie 0 zijn. In EOF(X) is X het bestandsnummer, waaronder het bestand is geopend. Het bestand moet zijn geopend voor de INPUT-mode. De functie EOF wordt meestal toegepast, wanneer de lengte van een bestand niet bekend is.

Een adres kan op verschillende wijzen uit het sequentiele gegevensbestand ADRBES worden opgevraagd. Men kan bijvoorbeeld het programma een "naam" opgeven, waarna het

programma het bestand vanaf het begin gaat lezen en de ingevoerde naam gaat vergelijken met de gelezen namen uit het bestand. Als de naam is gevonden, worden alle gegevens, die bij de naam behoren, op het scherm afgedrukt.

De volgende listing is van de subroutine "opvragen van adressen":

```
400 REM * OPVRAGEN VAN ADRESSEN *
405 OPEN "A:ADRBES" FOR INPUT AS #1
410 CLS
415 T=0
420 INPUT "Welke naam";N1$
425 L=LEN(N1$)
430 IF EOF(1)=-1 GOTO 490
435 INPUT #1,N$,A$,P$,W$,T$
440 IF N1$=LEFT$(N$,L) GOTO 450
445 GOTO 430
450 T=T+1:CLS
455 PRINT N$:PRINT
460 PRINT A$:PRINT
465 PRINT P$;SPC(3);W$:PRINT
470 PRINT T$:PRINT:PRINT
475 INPUT "Juiste adres (J/N)";J$
480 IF J$="J" OR J$="j" GOTO 495
485 GOTO 430
490 IF T=0 THEN PRINT "Adres niet gevond
en!"
495 PRINT "Druk op RETURN-toets."
500 IF INKEY$<>CHR$(13) GOTO 500
505 CLOSE #1
510 RETURN
```

3.6 Wijzigen of verwijderen van gegevens uit een bestand.

Wanneer gegevens uit een sequentieel gegevensbestand moeten worden gewijzigd, zal een hulpbestand moeten worden gebruikt. Hetzelfde geldt voor het verwijderen (laten vervallen) van gegevens uit een sequentieel bestand. Het is namelijk niet mogelijk gegevens uit een sequentieel bestand te lezen,

vervolgens deze gegevens te wijzigen en daarna weer terug te schrijven naar het zelfde bestand.

Voor ons "adressenbestand" zal voor het wijzigen en verwijderen van gegevens uit het bestand de volgende procedure moeten worden gevolgd:

1.
Open het bestaande bestand ADRBES voor INPUT.

2.
Open een hulpbestand HULP voor OUTPUT.

3.
Controleer of het einde van het bestand is bereikt met EOF. Indien het einde van het bestand is bereikt, ga dan naar punt 9, anders naar punt 4.

4.
Lees de gegevens van het eerste of het volgende adres uit het bestand ADRBES met INPUT.

5.
Vergelijk de gelezen gegevens uit het bestand met de gegevens, die met een INPUT-statement via het toetsenbord zijn ingevoerd. Men kan bijvoorbeeld het gewenste adres, dat gewijzigd of verwijderd moet worden, door middel van de naam laten opzoeken. In dit geval wordt de naam van elk gelezen adres met de gewenste (ingevoerde) naam vergeleken.

6.
Indien de naam niet gelijk is, worden de gegevens van het gelezen adres weggeschreven naar het bestand HULP, met een PRINT-statement.

7.
Indien de naam wel gelijk is, worden alle gegevens van het adres eerst met PRINT-statements afgedrukt op het scherm, om te controleren of het wel het juiste adres is.

Is dit niet zo, dan worden de gegevens van het adres alsnog weggeschreven naar het bestand HULP. Is dit wel het juiste adres, dan zullen voor een wijziging de gegevens van het adres eerst worden gewijzigd en vervolgens worden weggeschreven naar het bestand HULP. Voor het verwijderen van een adres uit het bestand wordt dit eenvoudig gedaan door de gelezen gegevens niet naar het bestand HULP weg te schrijven.

8.
Ga naar punt 3.

9.
Sluit beide bestanden met het statement CLOSE.

10.
Open het bestand ADRBES voor OUTPUT.

11.
Open het bestand HULP voor INPUT.

12.
Copieer bestand HULP naar het bestand ADRBES.

13.
Sluit beide bestanden met het statement CLOSE.

14.
Indien nog meer wijzigingen moeten worden aangebracht, ga dan naar punt 1, anders is dit het einde van de procedure.

De listing van de wijzigingsroutine is als volgt:

```
550 REM * VERWIJDEREN VAN EEN ADRES *
555 REM *   WIJZIGEN VAN EEN ADRES   *
565 OPEN "A:ADRBES" FOR INPUT AS #1
570 OPEN "A:HULP" FOR OUTPUT AS #2
575 CLS
580 PRINT "WIJZIGEN OF VERWIJDEREN VAN E
```

```
EN ADRES."
585 PRINT
590 INPUT "Naam";N1$
595 L=LEN(N1$)
600 INPUT "Verwijderen (V) of Wijzigen (
W)";V$
605 IF V$<>"V" AND V$<>"W" THEN PRINT "V
erkeerde keuze gemaakt!":GOTO 600
610 IF EOF(1)=-1 GOTO 715
615 INPUT #1,N$,A$,P$,W$,T$
620 IF N1$=LEFT$(N$,L) GOTO 635
625 PRINT #2,N$;",";A$;",";P$;",";W$;","
;T$
630 GOTO 610
635 CLS
640 PRINT N$:PRINT A$:PRINT P$;SPC(3);W$
:PRINT T$
645 PRINT:PRINT
650 INPUT "Juiste adres (J/N)";J$
655 IF J$="N" OR J$="n" GOTO 625
660 IF V$="V" GOTO 610
665 PRINT "Voer nieuwe gegevens in":PRIN
T
670 INPUT "Naam        ";N$
675 INPUT "Adres       ";A$
680 INPUT "Postcode    ";P$
685 INPUT "Woonplaats ";W$
690 INPUT "Telefoonnr.";T$
695 PRINT
700 INPUT "Informatie juist (J/N)";J$
705 IF J$="N" OR J$="n" GOTO 665
710 GOTO 625
715 CLOSE #1,#2
720 OPEN "A:ADRBES" FOR OUTPUT AS #1
725 OPEN "A:HULP" FOR INPUT AS #2
730 IF EOF(2)=-1 GOTO 750
735 INPUT #2,N$,A$,P$,W$,T$
740 PRINT #1,N$;",";A$;",";P$;",";W$;","
;T$
745 GOTO 730
```

```
750 CLOSE #1,#2
755 CLS
760 PRINT "Wijziging is uitgevoerd."
765 INPUT "Volgende wijziging (J/N)";J$
770 IF J$="J" OR J$="j" GOTO 565
775 PRINT "Druk op RETURN-toets."
780 IF INKEY$<>CHR$(13) GOTO 780
785 RETURN
```

Opmerking:

Bij random gegevensbestanden is het verwijderen of wijzigen van gegevens uit een bestand niet zo'n bewerkelijke zaak als bij sequentiele gegevensbestanden.

De gegevens uit random bestanden behoeven niet vanaf het begin van het bestand sequentieel te worden opgezocht, maar kunnen direct worden geselecteerd uit het bestand, door het programma het record-nummer, waar de gewenste gegevens staan, op te geven. De gegevens kunnen dan direct worden gelezen en aan het programma beschikbaar worden gesteld. De gegevens kunnen, na te zijn bewerkt, door het programma direct naar het bestand worden teruggeschreven. Men moet bij random bestanden wel het record-nummer van de gewenste informatie kennen. Daarom moet er een relatie bestaan tussen record-nummers en de informatie in het bestand. Het record-nummer kan bijvoorbeeld overeenkomen met een artikelnummer, een persoonsnummer, enz.

3.7 De functies LOC en LOF.

Bij sequentiele bestanden op schijf geeft de functie LOC(X) als resultaat het aantal sectoren (blokken van 256 bytes), dat gelezen is uit of geschreven is naar een bestand, sinds dat bestand werd geopend. De parameter X is het bestandsnummer, waaronder het bestand is geopend. Steeds nadat een sector is gelezen, wordt de waarde, die door de functie LOC(X) wordt gegeven, met 256 verhoogd. Iedere verhoging met 256 wil dus zeggen, dat een volgende sector is gelezen of beschreven. Bij het bereiken van het einde van het bestand

geeft de functie LOC(X) de grootte van het bestand (in bytes) aan.

Bij random bestanden geeft de functie LOC(X) een ander resultaat dan bij sequentiele bestanden. Bij random bestanden geeft de functie namelijk het record-nummer van het zojuist met GET of PUT benaderde record als resultaat. Na het openen van het random bestand, doch voordat er een record werd benaderd, geeft de functie LOC(X) de waarde 0.

De functie LOF(X) geeft als resultaat de grootte van het gespecificeerde bestand X in bytes. Parameter X is het bestandsnummer, waaronder het bestand werd geopend. Met het volgende programma kunnen we de grootte van het sequentiele bestand ADRBES bepalen.

```
10 REM * GROOTTE BESTAND ADRBES *
20 OPEN "A:ADRBES" FOR INPUT AS #1
30 PRINT "BESTAND ADRBES IS";LOF(1);"BYT
ES GROOT"
40 CLOSE #1
50 END
```

3.8 Meer over het statement LINE INPUT.

Met het LINE INPUT-statement, gevolgd door het bestandsnummer en een alfanumerieke variabele, kan een complete regel (van maximaal 254 tekens) vanuit een sequentieel bestand worden gelezen en toegekend aan een alfanumerieke variabele.

Het statement leest net zolang tekens uit een sequentieel bestand, totdat een "carriage return" wordt bereikt. Het slaat vervolgens de Carriage return/Line feed combinatie over en het volgende LINE INPUT statement leest alle tekens tot de volgende carriage return. Wanneer eerst Line Feed en daarna Carriage Return wordt gelezen, ziet het programma dit als een string, welke eindigt op een Line Feed.

Met het volgende programma kan, via het toetsenbord, een tekst worden ingegeven met het statement LINE INPUT en naar een sequentieel bestand worden geschreven met het statement PRINT. Met hetzelfde programma kan de tekst weer worden gelezen vanuit het bestand, met een LINE INPUT statement en worden afgedrukt op het beeldscherm met het statement PRINT.

```
100 REM * STATEMENT LINE INPUT # *
110 CLS
120 OPEN "A:TEKST" FOR OUTPUT AS #1
130 PRINT "TEKST KAN WORDEN INGEVOERD."
140 PRINT "TYPE VOOR EINDE TEKST HET TEK
EN @ IN."
150 PRINT:PRINT
160 LINE INPUT T$
170 IF T$="@" GOTO 200
180 PRINT #1,T$
190 GOTO 160
200 CLOSE #1
210 PRINT:PRINT "Druk op RETURN"
220 IF INKEY$<>CHR$(13) GOTO 220
230 CLS
240 OPEN "A:TEKST" FOR INPUT AS #1
250 IF EOF(1)=-1 GOTO 290
260 LINE INPUT #1,T$
270 PRINT T$
280 GOTO 250
290 CLOSE #1
300 END
```

3.9 Het INPUT#-statement.

Met het statement:

```
INPUT #bestandsnummer,lijst met variabelen
```

kunnen gegevens (items) van een sequentieel bestand van schijf worden gelezen en worden toegekend aan programma-

variabelen. Het bestandsnummer is het nummer, waaronder het bestand voor INPUT (bestand naar computer) is geopend.

De lijst met variabelen bevat de namen van de variabelen (numerieke- en/of alfanumerieke), die zullen worden toegekend aan de items in het bestand. Voorbeeld:

```
100 INPUT #1,A$,B$,C
```

De items uit het bestand worden aan het systeem aangeboden net zoals deze met een INPUT-statement zouden worden ingevoerd via het toetsenbord.

Numerieke waarden:
Bij numerieke waarden worden spaties, carriage returns en line feeds, die voorafgaan aan een getal, genegeerd. Het eerste gelezen teken, dat geen spatie, carriage return of line feed is, wordt beschouwd als het begin van een getal. Het getal wordt als beeindigd beschouwd, als een spatie, carriage return, line feed of komma wordt ontvangen.

String items:
Wanneer een programma een sequentieel bestand afzoekt naar een string-item (alfanumerieke informatie), worden voorafgaande spaties, carriage returns en line feeds ook genegeerd. Het eerste teken, dat geen spatie, carriage return of line feed is, wordt beschouwd als het begin van een string-item. Wanneer het eerste teken een aanhalingsteken (") is, zal het string-item worden gevormd door alle tekens, die tussen het eerste aanhalingsteken en het eerste hierna volgende aanhalingsteken staan. Dus, een string-item, dat tussen aanhalingstekens staat, mag geen aanhalingstekens als normaal teken bevatten.

Wanneer het eerste teken van een string-item geen aanhalingsteken is, zal de string eindigen op een komma, carriage return of een line feed (of nadat 250 tekens zijn gelezen). Wanneer een string een komma, een puntkomma, gewenste voorafgaande spaties, een carriage return of een line feed bevat, is de beste oplossing de string tussen aanhalingstekens

(CHR$(34)) naar schijf te schrijven. Het volgende voorbeeld illustreert dit.

```
A$ = "schroef,  bolkop"
B$ = "    M6"
```

Het statement PRINT #1,A$;B$ zal dan het volgende op schijf schrijven:

```
schroef,  bolkop    M6
```

Gaan we dit teruglezen van schijf met het statement INPUT #1,A$,B$ dan zal "schroef" worden toegekend aan variabele A$ en "bolkop   M6" aan variabele B$.

Om de strings op de juiste wijze van elkaar op schijf te scheiden, moeten we dubbele aanhalingstekens naar de schijf schrijven. Het statement

```
PRINT #1,CHR$(34);A$;CHR$(34);CHR$(34);B$;CHR
$(34)
```

zal dan het volgende op schijf schrijven:

```
"schroef,  bolkop""    M6"
```

Gaan we dit teruglezen van schijf met het statement INPUT #1,A$,B$ dan zal "schroef, bolkop" worden toegekend aan variabele A$ en "   M6" aan variabele B$.

Einde bestand:
Wanneer het einde van het bestand is bereikt, op het moment dat een numeriek- of string-item moet worden gelezen (INPUT), wordt het item beeindigd.

Het volgende programma is een oefenprogramma, om het hiervoor geleerde in praktijk te kunnen brengen.

Listing OEFEN

```
100 REM * OEFENPROGRAMMA *
110 OPEN "A:OEFEN" FOR OUTPUT AS #1
120 CLS
130 PRINT "Voer via het toetsenbord alfa
nu-"
140 PRINT "merieke gegevens in zowel tus
sen als"
150 PRINT "niet tussen aanhalingstekens.
 Laat"
160 PRINT "in beide gevallen de tekstvoo
raf-"
170 PRINT "gaan door een aantal spaties.
"
180 PRINT "Voer via het toetsenbord ook
een"
190 PRINT "numerieke waarde in, al dan n
iet"
200 PRINT "voorafgegaan door spaties."
210 PRINT:PRINT
220 INPUT "Zonder aanhalingstekens";A$
230 INPUT "Tussen aanhalingstekens";B$
240 INPUT "Numerieke waarde       ";C
250 PRINT #1,CHR$(34);A$;CHR$(34);CHR$(3
4);B$;CHR$(34);C
260 CLOSE #1
270 OPEN "A:OEFEN" FOR INPUT AS #1
280 INPUT #1,A$,B$,C
290 PRINT:PRINT
300 PRINT A$:PRINT B$:PRINT C
310 CLOSE #1
320 IF INKEY$<>CHR$(13) GOTO 320
330 GOTO 100
```

4 Random bestanden

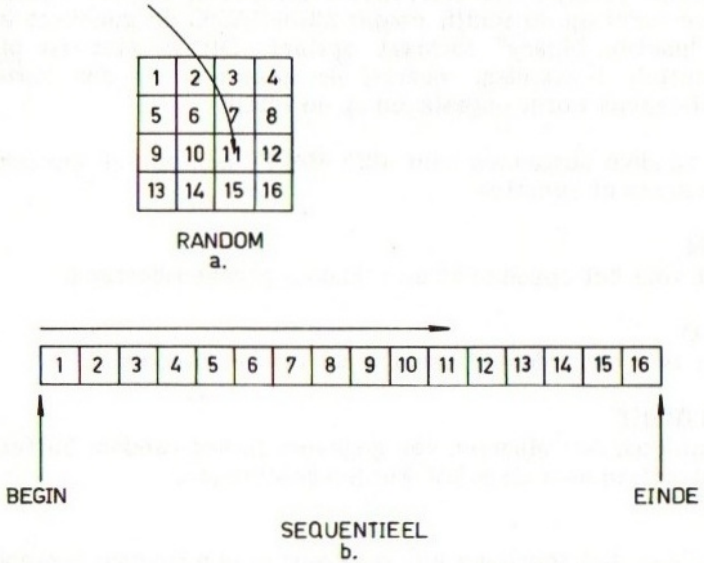
Bij een random gegevensbestand zijn gegevens direct (willekeurig) toegankelijk, waar deze zich ook op de schijf bevinden. Dit is in tegenstelling tot een sequentieel gegevensbestand, waarbij het bestand vanaf het begin van het bestand en in volgorde van hoe de gegevens op schijf zijn geschreven, moet worden afgezocht naar de gewenste gegevens. Het is bij random bestanden, om bij de gewenste gegevens te komen, dus niet nodig alle informatie te lezen, zoals bij sequentiele bestanden het geval is.

In afbeelding 4-1 wordt dit symbolisch voorgesteld. In afbeelding 4-1a worden de gegevens in record 11 direct geselecteerd en in afbeelding 4-1b moeten eerst de gegevens van de records 1 tot en met 10 worden gelezen, alvorens over de gegevens in record 11 kan worden beschikt.

De gegevens worden opgeslagen op vaste aparte plaatsen op de schijf, die records worden genoemd. Een record mag maximaal 256 bytes bevatten en minimaal 1 byte. Om een record te kunnen selecteren (adresseren) voor het schrijven of lezen van gegevens, is aan elke record een recordnummer toegekend. Met behulp van de recordnummers is binnen een random gegevensbestand elk record direct te bereiken, of met andere woorden, direct toegankelijk (direct access).

Wanneer een bepaald record moet worden gelezen, moet wel het recordnummer bekend zijn, anders is het niet mogelijk om het record direct te selecteren.

Het wijzigen van gegevens in een record in een random bestand is veel eenvoudiger dan het bij sequentiele bestanden



Afb. 4-1 Gegevensbestanden.

het geval is. De wijzigingen (mutaties) kunnen door het record te selecteren en de nieuwe gegevens over de oude gegevens heen te schrijven, direct worden aangebracht. Ook het uitbreiden van een random gegevensbestand met nieuwe records is een eenvoudige zaak. Een random bestand kan worden uitgebreid door het eerste vrije record (recordnummer) te bepalen en daar de nieuwe gegevens heen te schrijven.

Men zou bijvoorbeeld het adressenbestand uit hoofdstuk 3 in een random gegevensbestand kunnen opslaan, waarbij elk adres 1 record inneemt. Zouden we nu een bepaald adres willen opvragen, dan moeten we wel weten in welk record-(nummer) het gewenste adres staat. Het is natuurlijk ook mogelijk het random bestand sequentieel af te zoeken naar een adres, door alle records van het bestand in volgorde te lezen en de gewenste naam te vergelijken met de namen uit de gelezen records. Dit is dan ook niet het eigenlijke gebruik van een random gegevensbestand.

Een ander voordeel van een random bestand is, dat het minder ruimte vergt op de schijf, omdat MSX-BASIC de gegevens in een "packed binary" formaat opslaat. Dit is niet zo bij sequentiele bestanden, waarbij de gegevens als een serie ASCII-tekens wordt opgeslagen op de schijf.

Voor random bestanden kent MSX-BASIC een aantal speciale statements en functies.

OPEN
Dient voor het openen van een random gegevensbestand.

FIELD
Dient voor het indelen (formateren) van het random buffer.

LSET/RSET
Dienen voor het plaatsen van gegevens in het random buffer, voordat deze naar de schijf worden geschreven.

PUT
Dient voor het schrijven van gegevens in een random bestand op schijf vanuit het random buffer.

GET
Dient voor het lezen van gegevens uit een random bestand op schijf. De gegevens worden na het lezen in het random buffer geplaatst.

CLOSE
Dient voor het afsluiten van een random gegevensbestand.

MKI$/MKS$/MKD$
Dienen voor het converteren van numerieke gegevens in alfanumerieke waarden, voordat deze in het random buffer worden geplaatst en vervolgens op schijf worden geschreven.

CVI/CVS/CVD
Dienen voor het converteren van alfanumerieke waarden in numerieke waarden, voordat de computer de numerieke gegevens kan verwerken.

LOC
Dient voor het aangeven van welke record het laatst is gelezen of geschreven.

4.1 Het werken met random gegevensbestanden.

4.1.1 Openen van een random gegevensbestand.

Voor random gegevensbestanden bestaan er maar 1 OPEN-statement, dit in tegenstelling tot sequentiele gegevensbestanden, waarbij voor het vastleggen van gegevens in het bestand (OUTPUT), het toevoegen van nieuwe gegevens (APPEND) en het lezen van gegevens (INPUT) verschillende OPEN-statements bestaan. Voor de drie hiervoorgenoemde bewerkingen bestaat voor random gegevensbestanden maar 1 OPEN-statement.

Een random bestand wordt meestal in het begin van het programma eenmalig geopend en blijft dan geopend totdat het programma wordt beeindigd. Het aantal bestanden, dat tegelijkertijd kan worden geopend, wordt bepaald door de systeemvariabele MAXFILES (zie ook hoofdstuk 3).

Het formaat van het OPEN-statement voor random gegevensbestanden ziet er als volgt uit:

```
OPEN "apparaat:bestandsnaam" AS #X LEN=Y
```

Apparaat:
Voor random bestanden kan dit zijn A of B, hetgeen wil zeggen schijf A of schijf B.

Bestandsnaam:
Is de naam waaronder het bestand is vastgelegd (geschreven) op schijf of zal worden vastgelegd (gecreeerd). De naam mag maximaal uit 8 tekens bestaan, gevolgd door een bestandsnaamuitbreiding van drie tekens. De bestandsnaam en de bestandsnaamuitbreiding moeten van elkaar worden gescheiden door een punt.

Bestandsnummer X:
Parameter X is het bestandsnummer. Dit nummer geldt zolang het bestand is geopend en wordt in de plaats van de bestandsnaam gebruikt om het bestand aan te geven. Voor random bestanden wordt het bestandsnummer gebruikt in de statements FIELD, PUT, GET en CLOSE. Het bestand moet altijd eerst worden geopend, voordat deze statements kunnen worden gebruikt.

Recordlengte Y:
Parameter Y (een geheel getal) geeft de recordlengte aan in bytes, die voor het bestand wordt toegepast. De minimale lengte is 1 en de maximale lengte is 256. Wanneer de lengte niet wordt opgegeven in het OPEN-statement, dan is de default-waarde van kracht. Deze is 256.

Voorbeeld:

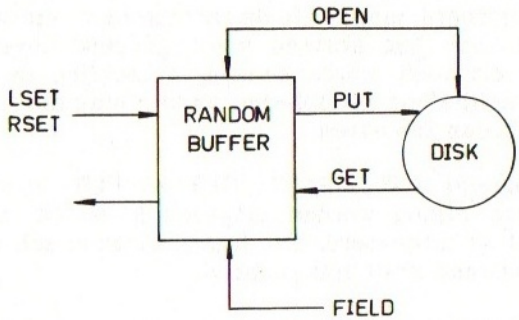
```
105 OPEN "A:VOORAD" AS #1 LEN=33
```

De naam van het bestand is VOORAD (voorraadadministratie), het wordt gecreeerd op schijf A en is verder bekend onder bestandsnummer 1. De lengte van de records in het bestand is 33 bytes.

4.1.2 Het indelen van het random buffer.

Voordat met een PUT-statement gegevens van een record naar schijf kunnen worden geschreven, moeten deze eerst in een random buffer worden geplaatst met een LSET- of RSET-statement. (afbeelding 4-2). In dit buffer komen ook de gegevens van een record te staan, die met een GET-statement uit een random bestand op schijf worden gelezen.

Het random buffer moet, voordat deze kan worden gebruikt, eerst worden ingedeeld (geformateerd). Voor het indelen van het buffer dient het statement FIELD. Het buffer moet overeenkomen met de indeling van het toegepaste record. Het formaat van het FIELD-statement is:



Afb. 4-2 Het Random Buffer.

```
FIELD #X,Y1 AS naam,Y2 AS naam, enz.
```

Bestandsnummer X:
Parameter X is het bestandsnummer waaronder het random bestand werd geopend met het OPEN-statement.

Aantal tekens per veld Y1, Y2, enz.:
Y1, Y2, enz. geven het aantal tekens voor elk veld in het buffer aan.

Naam:
Elk veld in het buffer krijgt een naam. De namen moeten altijd namen van alfanumerieke variabelen zijn.

Afbeelding 4-3 toont een voorbeeld van een record-indeling van een random bestand.

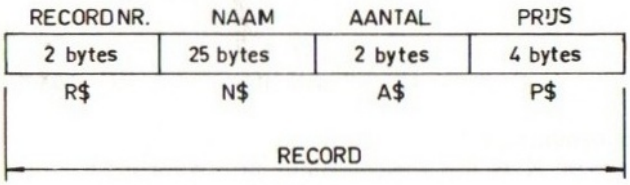
De velden van de records uit afbeelding 4-3 hebben respectievelijk de volgende namen, R$, N$, A$ en P$. Het FIELD-statement voor het indelen van het random buffer gaat er als volgt uit zien:

```
110 FIELD #1,2 AS R$,25 AS N$,2 AS A$,4 AS P$
```

Voordat een FIELD-statement in een programma kan worden uitgevoerd, moet eerst het bijbehorende random bestand zijn geopend. Het totaal aantal bytes, dat in een FIELD-statement

is gespecificeerd, mag nooit de recordlengte, die werd gespecificeerd toen het bestand werd geopend, overschreiden. Wanneer dit toch wordt gedaan, verschijnt de boodschap "Field overflow" op het scherm. Verder mag het buffer nooit groter zijn dan 256 bytes.

De statements LSET, RSET, GET en PUT kunnen in een programma alleen worden uitgevoerd, indien het FIELD-statement is uitgevoerd. Dit houdt automatisch in, dat het random bestand moet zijn geopend.



Afb. 4-3 De indeling van een record.

4.1.3 Plaatsen van gegevens in het random buffer.

Het FIELD-statement zorgt er niet voor, dat de gegevens in het random buffer worden geplaatst. Dit is een taak van de statements LSET en RSET. Hierbij moet nog worden opgemerkt, dat numerieke gegevens, in ons voorbeeld (afbeelding 4-3) het "record nr.", het "aantal" en de "prijs" eerst moeten worden omgezet in een alfanumeriek formaat (string), voordat deze met de statements LSET of RSET in het buffer kunnen worden geplaatst. Voor het converteren worden de volgende functies toegepast:

MKI$(X)
Converteert een geheel getal X in een alfanumerieke constante van 2 bytes groot.

MKS$(X)
Converteert een enkelvoudig nauwkeurig getal X in een alfanumerieke constante van 4 bytes groot.

MKD$(X)
Converteert een dubbelvoudig nauwkeurig getal X in een alfanumerieke constante van 8 bytes groot.

In het voorbeeld is de "naam" een alfanumeriek gegeven. Dit behoeft dus niet te worden geconverteerd, om in het random buffer te worden geplaatst.

Opmerking:
De namen van de velden in het random buffer mogen nooit voor andere doeleinden worden gebruikt, zoals voor INPUT- en LET-statements.

Wanneer we het buffer nu gaan vullen en stellen, dat "aantal" en "record nr." gehele getallen zijn, en "prijs" een enkelvoudig nauwkeurig getal, dan gaat het programmadeel dat zorgt voor het vullen van het buffer er als volgt uit zien:

```
125 INPUT "Record nr. (0-99)";R1
130 INPUT "Naam             ";N1$
135 INPUT "Aantal           ";A1
140 INPUT "Prijs            ";P1
145 LSET R$=MKI$(R1)
150 LSET N$=N1$
155 LSET A$=MKI$(A1)
160 LSET P$=MKS$(P1)
```

In het voorgaande programmadeel valt direct op te merken, dat de namen van de variabelen (alfanumeriek en numeriek), die worden gebruikt in de INPUT-statements, niet gelijk zijn aan de namen van de velden in het random buffer. Verder valt ook op, dat de numerieke waarden eerst worden geconverteerd naar alfanumerieke waarden (strings).

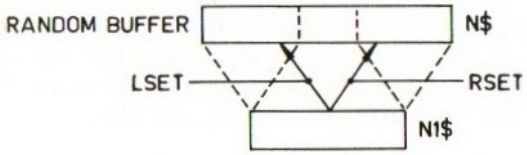
Opmerking:
De INPUT- en LSET-statements in het programma mogen elkaar ook afwisselen. Dat wil zeggen, na elk INPUT-statement eerst het bijbehorende LSET-statement.

Wanneer het veld N$ in het random buffer moet worden gevuld met N1$, dan kan dit op twee manieren gebeuren.

Hierbij moet worden aangenomen, dat de lengte van N1$ kleiner is dan N$.

N1$ kan links in het veld N$ worden geplaatst, waarbij de rest van het veld wordt aangevuld met spaties (afbeelding 4-4). In dit geval moet men het statement LSET gebruiken. Voorbeeld: LSET N$=N1$.

N1$ kan ook rechts in het veld N$ worden geplaatst, waarbij de rest van het veld wordt aangevuld met spaties. In dit geval moet het statement RSET worden gebruikt. Voorbeeld: RSET N$=N1$.



Afb. 4-4 Het gebruik van LSET en RSET.

4.1.4 Schrijven van gegevens naar een bestand.

Voor het schrijven van gegevens (1 record) uit een random buffer in een random bestand op schijf, wordt het statement PUT gebruikt. Het formaat van het statement PUT ziet er als volgt uit:

```
PUT #bestandsnummer,recordnummer
```

Het bestandsnummer is het nummer waaronder het random bestand werd geopend met het OPEN-statement. Het recordnummer is het nummer van het record, waarheen de gegevens uit het random buffer moeten worden geschreven. Het recordnummer moet een geheel getal tussen 0 en 32767 zijn. Wanneer het recordnummer niet is gegeven, zal het recordnummer worden gebruikt, dat volgt op het nummer van het laatst geschreven of gelezen record.

Wanneer we gegevens (1 record) vanuit het random buffer

weg willen schrijven naar het random bestand, gaat het PUT-statement er als volgt uitzien:

```
165 PUT #1,R1
```

In het statement geeft het nummer 1 het bestandsnummer aan van het random bestand op schijf. R1 geeft het nummer van het record in het bestand aan, waar de gegevens naartoe moeten worden geschreven.

We kunnen nu het programma voor het creeren van een bestand verder afmaken.

```
100 REM * VOORRAADADMINISTRATIE *
105 OPEN "A:VOORAD" AS #1 LEN=33
110 FIELD #1,2 AS R$,25 AS N$,2 AS A$,4
AS P$
115 CLS
120 PRINT "SCHRIJVEN VAN RECORDS":PRINT
125 INPUT "Record nr. (0-99)";R1
130 INPUT "Naam              ";N1$
135 INPUT "Aantal            ";A1
140 INPUT "Prijs             ";P1
145 LSET R$=MKI$(R1)
150 LSET N$=N1$
155 LSET A$=MKI$(A1)
160 LSET P$=MKS$(P1)
165 PUT #1,R1
170 PRINT
175 INPUT "Volgende record (J/N)";J$
180 IF J$="J" OR J$="j" GOTO 115
```

Het programma plaatst de informatie, die via het toetsenbord wordt ingevoerd in een random buffer en schrijft vervolgens de inhoud van het buffer (1 record) naar het random bestand VOORAD op schijf. Elke keer dat het PUT-statement wordt uitgevoerd, wordt een record naar het random bestand geschreven. Nummer R1, dat op regelnummer 125 met een INPUT-statement wordt ingevoerd, wordt dan het recordnummer.

4.1.5 Lezen van gegevens uit een random bestand.

Voor het lezen van gegevens uit een random bestand moet natuurlijk het random bestand eerst zijn geopend met een OPEN-statement. Verder moet het random buffer, waarin de gelezen gegevens (het record) als eerste worden geplaatst, met een FIELD-statement zijn geformateerd.

In een programma, waarin zowel records worden weggeschreven naar een bestand, als gelezen uit hetzelfde bestand, behoeft in principe maar een OPEN-statement en een FIELD-statement te worden toegepast.

Voor het lezen van een record uit een random bestand op schijf en het plaatsen van de gelezen gegevens in het random buffer, wordt het statement GET gebruikt. Het formaat van het GET-statement ziet er als volgt uit:

```
GET #bestandsnummer,recordnummer
```

Het bestandsnummer is het nummer, waaronder het random bestand werd geopend met het OPEN-statement. Het recordnummer is het nummer van het record, dat moet worden gelezen uit het bestand. Het nummer moet een geheel getal zijn, dat ligt tussen 0 en 32767. Wanneer het recordnummer niet is gegeven in het statement, zal het recordnummer, dat volgt op het recordnummer van het laatst geschreven of gelezen record, worden genomen.

Het recordnummer van het te lezen record kan worden opgevraagd met een INPUT-statement. De programmaregel voor het opvragen van het recordnummer zou er dan als volgt uit kunnen gaan zien:

```
210 INPUT "Record nr. (0-99)";R1
```

Voor het programma gaat het GET-statement er als volgt uit zien:

```
215 GET #1,R1
```

Met het GET-statement wordt het record met nummer R1 uit het random bestand (bestandsnummer 1) gelezen en in het random buffer geplaatst. Nadat de gegevens in het buffer zijn geplaatst, kan het programma vrij over de gegevens van het gelezen record beschikken. Getallen moeten, alvorens deze door het programma kunnen worden verwerkt, eerst worden geconverteerd van het alfanumerieke formaat in het numerieke formaat. Hiervoor dienen de volgende functies:

CVI(X$)
Converteert een alfanumerieke variabele X$ van 2 bytes in een geheel getal.

CVS(X$)
Converteert een alfanumerieke variabele X$ van 4 bytes in een enkelvoudig nauwkeurig getal.

CVD(X$)
Converteert een alfanumerieke variabele X$ van 8 bytes in een dubbelvoudig nauwkeurig getal.

Het programma voor het lezen van records kan als volgt worden afgemaakt:

```
200 CLS
205 PRINT "LEZEN VAN RECORDS":PRINT
210 INPUT "Record nr. (0-99)";R1
215 GET #1,R1
220 PRINT
225 PRINT "Record nr. :";CVI(R$)
230 PRINT "Naam       :";N$
235 PRINT "Aantal     :";CVI(A$)
240 PRINT "Prijs      :";CVS(P$)
245 PRINT:PRINT
250 INPUT "Volgende record (J/N)";J$
255 IF J$="J" OR J$="j" GOTO 200
260 CLOSE #1
265 END
```

4.1.6 Sluiten van een random bestand.

Wanneer een random bestand niet meer wordt gebruikt, kan het worden afgesloten met het statement CLOSE. Hiermee wordt ook gelijktijdig het bijbehorende random buffer in het geheugen vrijgegeven (zie ook hoofdstuk 3).

Met de statements END en CLEAR en het commando NEW worden de bestanden op de schijf automatisch afgesloten. Dit is echter niet het geval met het statement STOP. Dit houdt in, dat regelnummer 260 eventueel mag worden weggelaten, daar het statement END op regelnummer 265 het bestand ook afsluit.

4.1.7 De functie LOC.

Bij random bestanden geeft de functie LOC(X) als resultaat het nummer van het laatst gelezen of geschreven record uit het bestand met bestandsnummer X. Parameter X is het bestandsnummer, waaronder het bestand is geopend. Wanneer men bijvoorbeeld de uitvoering van een programma wil laten beeindigen na het lezen van het record met recordnummer 30, kan dit in het programma als volgt worden opgelost:

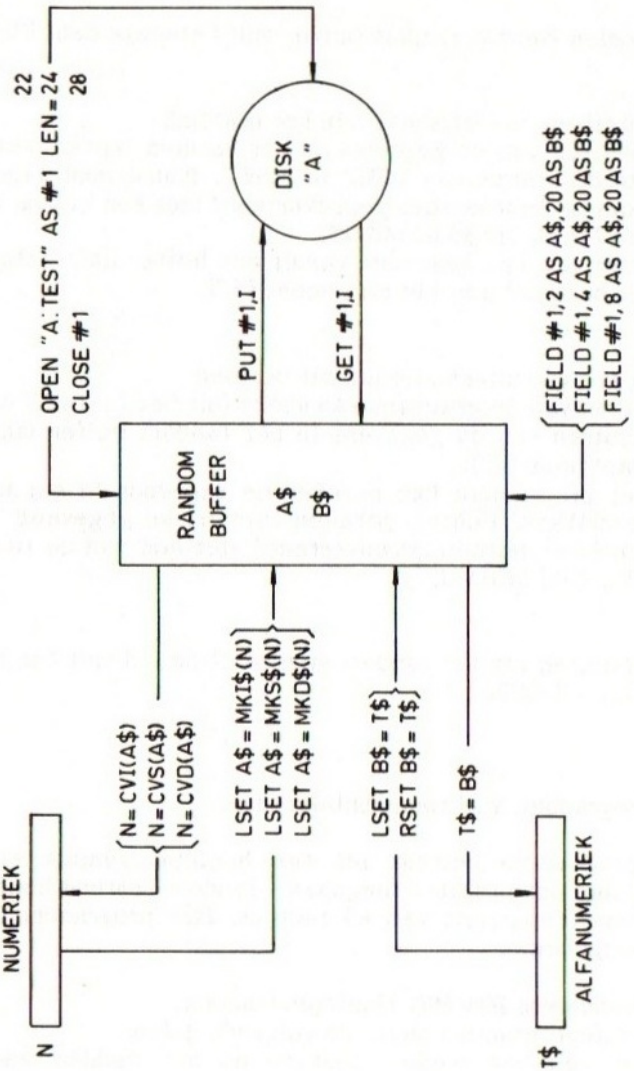
```
160 IF LOC(1)>30 THEN END
```

Het bestand is geopend onder bestandsnummer 1.

4.2 Overzicht statements en functies.

Afbeelding 4-5 geeft nog eens een overzicht van de statements en functies, die betrekking hebben op het werken met random gegevensbestanden. Hierna volgt een korte samenvatting van hoe toegang wordt verkregen tot random bestanden.

1.
Openen van het random bestand met het statement OPEN.



Afb. 4-5 Overzicht van disk-BASIC statements en functies.

2.
Indelen van het random buffer met het statement FIELD.

3.
Schrijven van informatie in het bestand:
Plaatsen van de gegevens in het random buffer met een van de statements LSET of RSET. Numerieke gegevens moeten eerst worden geconverteerd met een van de functies MKI$, MKS$ of MKD$.
Schrijven van gegevens vanuit het buffer naar schijf (in het bestand) met het statement PUT.

4.
Lezen van informatie uit het bestand:
Lezen van de gegevens van schijf (uit het bestand) en het plaatsen van de gegevens in het random buffer met het statement GET.
Het programma kan nu over de gegevens in het buffer beschikken. Echter, getallen (numerieke gegevens) moeten eerst worden geconverteerd met een van de functies CVI, CVS of CVD.

5.
Afsluiten van het random gegevensbestand met het statement CLOSE.

4.3 Programma Voorraadadministratie.

Het programma bestaat uit een hoofdprogramma en een aantal subroutines. Het toegepaste random bestand heeft een maximale capaciteit van 99 records. Het programma is als volgt opgebouwd:

Regelnummers 100-190: Hoofdprogramma.
Het hoofdprogramma bevat de volgende delen:
Openen van het random bestand en het indelen van het random buffer.
Menu.

Het maken van een keuze uit het menu en het aanroepen van de gewenste subroutine.
Het beeindigen van het programma.

Regelnummers 200-315: Toevoegen nieuw artikel.
Aanroepen van de subroutine op regelnummer 760 voor het opvragen van het eerste vrije recordnummer, waarin de nieuwe gegevens moeten worden geplaatst. Hiervoor is recordnummer 100 gereserveerd.
Controleren of het bestand niet vol is.
Invoeren van de gegevens van het nieuwe artikel.
Plaatsen van de gegevens in het bestand.
Aanroepen, na het opslaan van het laatste artikel in het bestand, van de subroutine op regel 745. Deze subroutine plaatst het eerste vrije recordnummer in recordnummer 100.
Terugspringen naar het hoofdmenu.

Regelnummers 350-405: Gegevens van een artikel.
Aanroepen van de subroutine op regelnummer 775 voor het invoeren van het artikelnummer.
Ophalen van het eerste vrije recordnummer.
Controleren of het ingevoerde artikelnummer juist is.
Lezen van de gegevens uit het bestand van het gewenste artikelnummer.
Afdrukken van de gegevens van het artikel op het scherm.
Terugspringen naar het hoofdmenu.

Regelnummers 450-480: Toevoegen aan de voorraad.
Aanroepen van de subroutine op regelnummer 775 voor het invoeren van het artikelnummer.
Ophalen van het eerste vrije recordnummer.
Controleren of het ingevoerde artikelnummer juist is.
Lezen uit het bestand van de gegevens van het gewenste artikelnummer.
Invoeren van het toe te voegen aantal via het toetsenbord.
Optellen van de bestaande voorraad bij het nieuwe aantal.
Plaatsen van de bijgewerkte gegevens in het bestand.
Terugspringen naar het hoofdmenu.

Regelnummers 500–545: Aftrekken van voorraad.
Aanroepen van de subroutine op regelnummer 775 voor het invoeren van het artikelnummer.
Ophalen van het eerste vrije recordnummer.
Controleren of het ingevoerde artikelnummer juist is.
Lezen van de gegevens uit het bestand van het gewenste artikelnummer.
Invoeren via het toetsenbord van het af te trekken aantal.
Bepalen van de bestaande voorraad.
Controleren of het af te trekken aantal niet groter is dan de bestaande voorraad.
Aftrekken van het via het toetsenbord ingevoerde aantal van de bestaande voorraad.
Plaatsen van de bijgewerkte gegevens in het bestand.
Terugspringen naar het hoofdmenu.

Regelnummers 600–635: Afdrukken minimum voorraden.
Aanroepen van de subroutine op regelnummer 760 voor het opvragen van het eerste vrije recordnummer.
Afzoeken van het bestand tot het eerste vrije recordnummer naar artikelen, waarvan de minimum voorraad is overschreden.
Afdrukken van de artikelen, waarvan de minimum voorraad is overschreden.
Terugspringen naar het hoofdmenu.

Regelnummers 650–680: Veranderen prijs.
Aanroepen van de subroutine op regelnummer 775 voor het invoeren van het artikelnummer.
Ophalen van het eerste vrije recordnummer.
Controleren of het ingevoerde artikelnummer juist is.
Lezen van de gegevens van het gewenste artikelnummer uit het bestand.
Afdrukken van de naam, het artikelnummer en de oude prijs.
Invoeren via het toetsenbord van de nieuwe prijs.
Plaatsen van de bijgewerkte gegevens in het bestand.
Terugspringen naar het hoofdmenu.

Regelnummers 700–755: Creeren van het bestand.
Afvragen of er werkelijk een nieuw (schoon) bestand moet

worden gecreeerd. Na uitvoering van deze subroutine zal namelijk alle oude informatie verloren zijn gegaan.
Het creeren van 100 records. In alle 100 records wordt het eigen recordnummer geplaatst.
Het plaatsen in recordnummer 100 van het eerste vrije recordnummer. Dit is in eerste instantie recordnummer 1. Recordnummer 100 kan niet worden gebruikt voor het opslaan van normale gegevens.
Terugspringen naar het hoofdmenu.

Programma-listing:

```
100 REM * VOORRAADADMINISTRATIE *
105 WIDTH 40:KEY OFF
110 OPEN "A:VORAAD" AS #1 LEN=35
115 FIELD #1,2 AS RE$,25 AS B$,2 AS A$,2
 AS M$,4 AS P$
120 CLS
125 PRINT TAB(3);"VOORRAADADMINISTRATIE"
:PRINT:PRINT
130 PRINT "1. Toevoegen van een nieuw ar
tikel":PRINT
135 PRINT "2. Opvragen gegevens van een
artikel":PRINT
140 PRINT "3. Toevoegen aan bestaande vo
orraad":PRINT
145 PRINT "4. Aftrekken van voorraad":PR
INT
150 PRINT "5. Afdrukken van artikelen, d
ie"
155 PRINT "   onder minimum voorraad zij
n":PRINT
157 PRINT "6. Wijzigen van prijs":PRINT
160 PRINT "7. Creeren van bestand":PRINT
165 PRINT "8. Programma beeindiging":PRI
NT:PRINT
170 INPUT "Uw keuze";K
```

```
175 IF K<1 OR K>8 THEN PRINT "Verkeerde
keuze!":GOTO 170
180 IF K=8 THEN END
185 ON K GOSUB 200,350,450,500,600,650,7
00
190 GOTO 120
200 REM * TOEVOEGEN VAN EEN *
205 REM *   NIEUW ARTIKEL   *
210 GOSUB 760
211 IF R>99 THEN PRINT "Bestand is vol!:
GOTO 305
215 CLS
220 PRINT "GEGEVENS NIEUW ARTIKEL":PRINT
225 INPUT "Beschrijving artikel";B1$
230 INPUT "Aantal in magazijn  ";A1
240 INPUT "Minimum voorraad    ";M1
250 INPUT "Prijs per artikel   ";P1
255 PRINT:PRINT
260 LSET B$=B1$
265 LSET A$=MKI$(A1)
270 LSET M$=MKI$(M1)
275 LSET P$=MKS$(P1)
280 PUT #1,R
285 R=R+1
290 INPUT "Laatste toevoeging (J/N)";J$
295 IF J$="N" OR J$="n" GOTO 215
300 GOSUB 745
305 PRINT "Druk op RETURN-toets."
310 IF INKEY$<>CHR$(13) GOTO 310
315 RETURN
350 REM * GEGEVENS VAN EEN ARTIKEL *
355 GOSUB 775
375 PRINT "Voorraad";CVI(A$);"stuks":PRI
NT
380 PRINT "Minimum voorraad";CVI(M$);"st
uks":PRINT
```

```
385 PRINT "Prijs";:PRINT USING "###.##";
CVS(P$):PRINT
390 PRINT:PRINT
395 INPUT "Volgend artikel (J/N)";J$
400 IF J$="J" OR J$="j" GOTO 355
405 RETURN
450 REM * TOEVOEGEN AAN VOORRAAD *
455 GOSUB 775
460 INPUT "Aantal toe te voegen";A2
465 A1=CVI(A$)+A2
470 LSET A$=MKI$(A1)
475 PUT #1,R
480 RETURN
500 REM * AFTREKKEN VAN VOORRAAD *
505 GOSUB 775
510 INPUT "Aantal af te trekken";A2
515 A1=CVI(A$)
520 IF A1-A2<0 THEN PRINT "Slechts";A1;"
in magazijn.":GOTO 510
525 A1=A1-A2
530 IF A1=<CVI(M$) THEN PRINT "Hoeveelhe
id nu";A1:PRINT "Minimum voorraad is";CV
I(M$)
535 LSET A$=MKI$(A1)
540 PUT #1,R
545 RETURN
600 REM * AFDRUKKEN MIN. VOORRADEN *
605 GOSUB 760
606 CLS
610 FOR I=1 TO R-1
615 GET #1,I
620 IF CVI(A$)>CVI(M$) THEN GOTO 625
622 PRINT B$
623 PRINT "Hoeveelheid       =";CVI(A$)
624 PRINT "Minimum voorraad =";CVI(M$):P
RINT
```

```
625 NEXT I
630 IF INKEY$<>CHR$(13) GOTO 630
635 RETURN
650 REM * VERANDEREN PRIJS *
655 GOSUB 775
660 PRINT "Oude prijs  :";CVS(P$)
665 INPUT "Nieuwe prijs";P1
670 LSET P$=MKS$(P1)
675 PUT #1,R
680 RETURN
700 REM "CREEREN VAN BESTAND *
705 CLS
710 INPUT "BENT U ER ZEKER VAN (J/N)";J$
715 IF J$="J" OR J$="j" GOTO 722
720 RETURN
722 FOR R=1 TO 100:'AANTAL ARTIKELEN
725 LSET RE$=MKI$(R)
730 PUT #1,R
735 NEXT R
740 R=1
745 LSET M$=MKI$(R)
750 PUT #1,100
755 RETURN
760 GET #1,100
765 R=CVI(M$)
770 RETURN
775 CLS
780 INPUT "Artikelnummer";RI
782 GOSUB 760
785 IF RI<1 OR RI>R-1 THEN PRINT "Verkee
rd nummer!":GOTO 780
790 GET #1,RI
792 R=RI
795 CLS
800 PRINT B$:PRINT
```

```
805 PRINT "Artikelnummer";R:PRINT
810 RETURN
```

5 Omgaan met programmabestanden

Het is belangrijk te weten hoe programma's vanuit het computergeheugen kunnen worden opgeslagen op schijf. Voor het opslaan van programma's op schijf dient het commando SAVE.

Wanneer een programma eenmaal op schijf staat, is het natuurlijk ook interessant het programma voor uitvoering weer te kunnen laden vanaf schijf in het computergeheugen. Het laden kan worden bewerkstelligd met het commando LOAD.

Als het programma eenmaal in het computergeheugen staat, kunnen we de uitvoering ervan starten met het commando RUN. Het is ook mogelijk, met hetzelfde commando RUN, het programma zowel te laden als te starten.

Net zoals met programma's, die op cassette staan, is het ook met programma's op schijf mogelijk, deze met het commando MERGE in het computergeheugen te laden en samen te voegen met het programma dat reeds in het geheugen staat.

Met het commando NAME kan een programma, dat op schijf staat, een andere programmanaam worden toegekend.

Is een programma niet meer nodig, dan kan het eventueel van schijf worden verwijderd met het commando KILL.

Bestaat op een gegeven moment de behoefte, een bepaald programma te copieren, dan kunnen we gebruik maken van het commando COPY.

Voor het opvragen van de inhoudsopgave (directory) van een schijf, kan gebruik worden gemaakt van het commando FILES. De inhoudsopgave wordt dan afgedrukt op het beeldscherm. Is het de bedoeling, de inhoudsopgave door middel van een printer op papier af te drukken, dan kan dit met het statement LFILES.

Wilt u weten wat er nog aan vrije ruimte op een schijf beschikbaar is, dan hebben we daarvoor de functie DSKF.

Voor de benaming van bestanden (programma- en gegevensbestanden) op schijf dient een bepaald formaat, wat bekend staat onder de naam "bestandsaanduiding". In de volgende paragraaf zal hier uitgebreid aandacht aan worden besteed. De bestandsaanduiding wordt toegepast in de commando's SAVE, LOAD, RUN, MERGE, NAME, KILL, COPY, FILES en LFILES.

5.1 Bestandsaanduiding.

Met de bestandsaanduiding kunnen we precies aangeven hoe een bestand heet en op welke schijveneenheid de schijf zich bevindt, waar het bestand (bijvoorbeeld een programma) moet worden opgeslagen of kan worden gevonden. De bestandsaanduiding bestaat uit de volgende delen:

a. Schijveneenheid (A of B).
b. Bestandsnaam (max. 8 tekens).
c. Bestandsnaamuitbreiding (max. 3 tekens).

Het algemene formaat van de bestandsaanduiding is:

"schijveneenheid : bestandsnaam . bestandsnaamuitbreiding"

Met **schijveneenheid** wordt schijveneenheid A of B aangegeven. Wanneer A of B wordt weggelaten, wordt door de computer automatisch de laatst gebruikte schijveneenheid genomen.

De maximale lengte van **bestandsnaam** is 8 tekens. Wanneer

geen gebruik wordt gemaakt van het veld "bestandsnaamuitbreiding", dan mag de bestandsnaam maximaal 11 tekens lang zijn. Het negende, tiende en elfde teken stromen dan automatisch over naar het veld "bestandsnaamuitbreiding". Wanneer de bestandsnaam "adressenbes" wordt ingegeven, zonder bestandsnaamuitbreiding, dan zal dat het volgende resultaat opleveren:

"A:adressen.bes"

De laatste drie tekens van de bestandsnaam "adressenbes" komen achter de punt (.) in het bestandsnaamuitbreidingsveld te staan. Wanneer een bestandsnaam langer is dan 11 tekens, wordt een foutboodschap op het beeldscherm afgedrukt. Hierbij wordt ervan uitgegaan, dat de bestandsnaamuitbreiding niet wordt gebruikt. Wordt deze wel toegepast, dan zal, indien de bestandsnaam uit meer dan acht tekens bestaat, een foutboodschap worden gegeven. Met de bestandsnaamuitbreiding kunnen we het type bestand aangeven. U bent vrij in de keuze van afkortingen, doch wij adviseren om een bepaald soort bestand altijd dezelfde bestandsnaamuitbreiding te geven. Er kunnen de volgende afkortingen worden gebezigd:

ASC - ASCII (programma)
BAS - BASIC (programma)
DAT - DATA (gegevensbestand)

Door bestanden een standaard bestandsnaamuitbreiding te geven, kunnen deze later, wanneer er meer bestanden op schijf staan, eenvoudiger worden opgezocht. De bestandsnaamuitbreiding is echter optioneel, hetgeen wil zeggen, dat deze ook mag worden weggelaten. Hier volgen enkele bestandsaanduidingen:

"A:TEST.BAS"	met uitbreiding BAS
"A:TEST.BAS"	met uitbreiding BAS
"A:TEST.ASC"	met uitbreiding ASC
"A:TEST"	zonder uitbreiding
"TEST"	zonder schijveneenheid en uitbreiding

De schijveneenheid (A of B), indien aangegeven, en de bestandsnaam zijn altijd gescheiden door een dubbele punt (:). De bestandsnaam en de bestandsnaamuitbreiding zijn gescheiden door een punt (.). Wanneer de uitbreiding niet wordt gebruikt, en de bestandsnaam uit meer dan 8 tekens bestaat, dan zal na het achtste teken automatisch een punt volgen. De naam mag nooit uit meer dan 11 tekens bestaan.

In de bestandsaanduiding kan, om bepaalde groepen bestanden te kunnen aangeven, gebruik worden gemaakt van het vraagteken (?) en van de asterisk (*).

Voor een vraagteken in de bestandsnaam of -uitbreiding kan elk teken worden gelezen. Met "A:TEST?" worden die bestanden op de schijf in schijveneenheid A bedoeld, waarvan de eerste vier posities uit de tekens TEST bestaan en de laatste positie elk teken kan zijn. De naam is echter in dit geval slechts vijf tekens lang. Voor het vraagteken kan dus elk teken worden ingevuld.

Met "A:TEST.???" worden alle bestanden TEST, van welk type (ASC, BAS, enz.) dan ook, bedoeld.

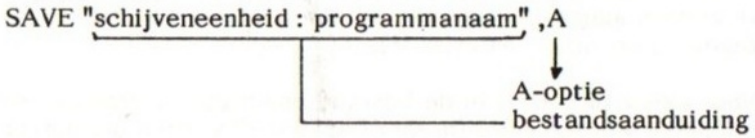
Met "B:?????.BAS" worden alle bestanden met de bestandsnaamuitbreiding BAS bedoeld, waarvan de bestandsnaam uit precies 5 tekens bestaat.

Zoals een vraagteken staat voor 1 teken, staat een asterisk (*) voor een of meer tekens.
Met "A:*.*" worden alle bestanden op schijveneenheid A bedoeld.
Met "B:*.ASC" worden alle bestanden met bestandsnaamuitbreiding ASC op schijveneenheid B bedoeld.
Met "B:P*" worden die bestanden op schijf B bedoeld, die beginnen met de letter P en die geen bestandsnaamuitbreiding hebben.
Met "*.B*" worden die bestanden op de laatst gebruikte schijveneenheid bedoeld, waarvan de bestandsnaamuitbreiding met de letter B begint.
Met "A:TEST*.ASC" worden die bestanden op schijveneenheid

A bedoeld, waarvan de bestandsnaam begint met de letters TEST en waarvan de bestandsnaam uitbreiding ASC is.

5.2 Opslaan en laden van bestanden.

Voor het schrijven van een programma, vanuit het computergeheugen naar schijf, dient het volgende commando:



Zie voor "schijveneenheid" en "programmanaam" paragraaf 5.1 (bestandsaanduiding).

Wanneer een programma in ASCII-formaat op schijf moet worden opgeslagen, moet de **A-optie** worden gebruikt. Dit wil zeggen, dat achter de bestandsaanduiding **,A** komt te staan. Wanneer de A-optie niet wordt gebruikt, zal het programma in het gecomprimeerde binaire formaat op schijf worden opgeslagen.

Voorbeelden:

```
SAVE "A:ADRES"
SAVE "A:ADRES.BAS"
SAVE "A:ADRES.ASC",A
```

Het ASCII-formaat neemt meer ruimte op schijf in dan het gecomprimeerde binaire formaat. Echter, sommige commando's, zoals MERGE, vereisen dat een programma in ASCII-formaat op schijf staat.

Overzicht:

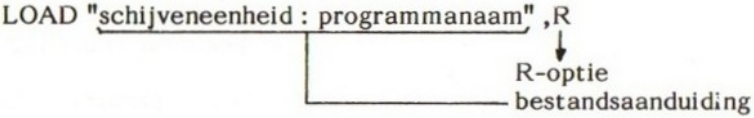
1.
Het commando CSAVE "programmanaam" wordt gebruikt voor het opslaan van een programma op cassette in het binaire formaat.

2.
Het commando SAVE "CAS:programmanaam" wordt gebruikt voor het opslaan van een programma op cassette in het ASCII-formaat.

3.
Het commando SAVE "bestandsaanduiding" wordt gebruikt voor het opslaan van een programma op schijf, in het binaire formaat.

4.
Het commando SAVE "bestandsaanduiding",A wordt gebruikt voor het opslaan van een programma op schijf, in het ASCII-formaat.

Voor het laden van een programma van schijf in het computergeheugen, kan onder andere gebruik worden gemaakt van het commando LOAD. Het formaat van het commando LOAD ziet er als volgt uit:



Zie voor schijveneenheid en programmanaam paragraaf 5.1 (bestandsaanduiding).

De programmanaam is de naam, die werd gespecificeerd toen het programma met het commando SAVE op schijf werd geschreven. Zowel programma's, die zijn opgeslagen op schijf met de A-optie (ASCII-formaat) als zonder de A-optie, kunnen met het commando LOAD van schijf in het computergeheugen worden geladen.

Wanneer de **R-optie** wordt gebruikt, zal het programma onmiddellijk nadat het in het geheugen is geladen, door de computer worden uitgevoerd.

Het commando LOAD wist altijd het geheugen en sluit, voor-

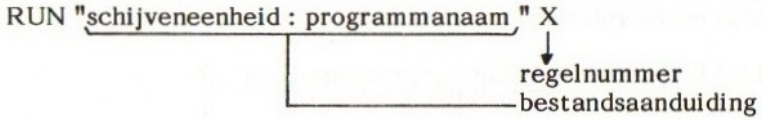
dat met laden wordt begonnen, alle bestanden. Wanneer echter de R-optie wordt toegepast, worden geopende gegevensbestanden open gehouden. Programma's, die na elkaar geladen en uitgevoerd worden, kunnen zodoende gebruik maken van dezelfde gegevensbestanden.

Voorbeelden:

```
LOAD "A:ADRES"
LOAD "A:ADRES.ASC",R
```

5.3 Starten van een programma.

Voor het starten van de uitvoering van een programma, dat in het computergeheugen staat, of nog op schijf, dient het commando RUN. Het algemene formaat van het commando RUN is:



Zie voor schijveneenheid en programmanaam paragraaf 5.1 (bestandsaanduiding).

Wanneer de bestandsaanduiding wordt weggelaten, zal de uitvoering van het programma, dat in het computergeheugen staat, worden gestart. Wanneer het commando RUN wordt gevolgd door een bestandsaanduiding, wordt het programma eerst van schijf geladen en vervolgens gestart.

De programmanaam is de naam, die werd gespecificeerd toen het programma met het commando SAVE op schijf werd geschreven.

Het commando RUN sluit alle bestanden en wist het geheugen, voordat het begint met het in het computergeheugen laden van het gespecificeerde programma.

Met **X** wordt een regelnummer aangegeven. Wanneer een regelnummer is gegeven, zal de uitvoering van het programma beginnen op het aangegeven regelnummer. Wanneer geen regelnummer is gegeven, zal de uitvoering van het programma beginnen op het eerste regelnummer, dat wil zeggen het laagste regelnummer van het programma.

5.4 Samenvoegen van programma's.

Met het commando MERGE kunnen programmaregels, die met het commando SAVE op schijf zijn gezet, van die schijf worden gelezen en aan het programma dat op dat moment in het geheugen staat, worden toegevoegd.

De regels, die moeten worden toegevoegd aan het programma in het geheugen, moeten met het commando SAVE in ASCII-formaat op schijf zijn geschreven. Indien dit niet het geval is, zal de boodschap "Bad file mode" op het scherm verschijnen.

Wanneer een of meerdere regels hetzelfde regelnummer hebben als die, welke reeds in het geheugen staan, zullen de in het geheugen staande regels worden overschreven door de regels die door middel van het commando MERGE van schijf worden gelezen.

MSX-BASIC keert na uitvoering van het commando MERGE altijd terug naar de directe (commando-) mode.

Het formaat van het commando MERGE is:

MERGE "schijveneenheid : programmanaam"

bestandsaanduiding

Het niet schoonmaken van de geheugeninhoud is het sterke punt van MERGE. Het is zodoende mogelijk programma's met elkaar te combineren. MERGE is ook zeer nuttig bij het in gedeelten (modulair) ontwikkelen van programma's. De verschillende gedeelten kunnen dan, nadat ze zijn geschreven en

getest, direct als losse modules naar schijf worden geschreven. Later kunnen die modules dan met behulp van MERGE worden samengevoegd in het geheugen. Ook komt het nogal eens voor, dat een bepaalde module in meerdere programma's wordt gebruikt.

5.5 Veranderen van een programmanaam.

Wanneer men een naam van een bestand op schijf wil veranderen, dan kan dit worden gedaan met het commando NAME. Het commando NAME heeft het volgende formaat:

NAME "schijveneenheid : bestaande naam " AS "nieuwe naam"

De bestaande naam moet bestaan en de nieuwe naam mag niet bestaan; anders zal dit resulteren in een foutboodschap. Na uitvoering van het commando NAME staat het bestand op dezelfde schijf en in hetzelfde gebied van de schijf, echter met de nieuwe naam. Wanneer geen schijveneenheid is gespecificeerd, wordt de laatst geselecteerde schijveneenheid genomen.

Voorbeelden:

```
NAME "A:ADRES" AS "ADRBES"
NAME "TEST" AS "TESTMOD"
```

5.6 Verwijderen van een bestand van schijf.

Wanneer men 1 of meerdere bestanden van schijf wil verwijderen, kan dit worden bewerkstelligd door het commando KILL. Het commando KILL heeft het volgende formaat:

```
KILL "bestandsaanduiding"
```

De schijveneenheid (A of B), welke in de bestandsaanduiding is gespecificeerd, is de eenheid, waarin de schijf is geplaatst met het bestand dat moet worden verwijderd. Een te verwij-

deren bestand kan een programmabestand, een sequentieel gegevensbestand of een random gegevensbestand zijn.

KILL kan ook worden gebruikt als statement in een programma. Wanneer in zo'n geval KILL in een programma een bestand moet verwijderen, dat reeds is geopend, dan wordt de foutboodschap "File already open" op het scherm afgedrukt.

Opmerking:
Met KILL moet altijd voorzichtig worden omgesprongen.

Voorbeelden:

KILL "A:ADRES.DAT"
Verwijdert het bestand ADRES.DAT van schijf.

KILL "A:*.DAT"
Verwijdert alle bestanden met de bestandsnaamuitbreiding DAT van de schijf.

KILL "A:*.*"
Verwijdert alle bestanden van de schijf.

5.7 Copieren van bestanden.

Voor het copieren van een bestand op schijf kan gebruik worden gemaakt van het commando COPY. Het formaat van het COPY-commando is als volgt:

COPY "schijveneenheid:bestandsnaam1" **TO** "schijveneenheid: bestandsnaam2"

Bestandsnaam 1 is de naam van het bestand dat moet worden gecopieerd.
Bestandsnaam 2 zal de naam worden van het gecopieerde bestand.

Er zijn systemen, die twee schijveneenheden (A en B) hebben, maar er zijn ook systemen met slechts 1 schijveneenheid. In

een systeem met twee schijveneenheden kunnen we bijvoorbeeld de schijf met het te copieren bestand in schijveneenheid A laden en de schijf, waarop het bestand moet worden gecopieerd in schijveneenheid B. Als dit is gebeurd, kan men het commando COPY geven. Wanneer we aannemen, dat het te copieren bestand de naam TEST heeft, en het gecopieerde bestand de naam TESTCOPY moet gaan voeren, dan gaat het commando er als volgt uit zien:

```
COPY "A:TEST" TO "B:TESTCOPY"
```

Bij een systeem met slechts 1 schijveneenheid moet u handelen alsof u twee schijveneenheden (A en B) heeft. Bijvoorbeeld:

```
COPY "A:TEST" TO "B:"
```

Nadat het bestand is geladen van de schijf in schijveneenheid A in het geheugen, zal door het systeem de volgende boodschap op het scherm worden geplaatst:

```
Insert diskette for drive B:
and strike a key when ready
```

U verwijdert nu de schijf uit schijveneenheid A en laadt de schijf voor schijveneenheid B. Vervolgens drukt u een willekeurige toets in (behalve CTRL + STOP). U moet de twee schijven blijven verwisselen, totdat het bestand volledig is gecopieerd. Wanneer het om een klein bestand gaat, zal het aantal verwisselingen klein blijven. Handelt het om een groot bestand, dan zal het aantal verwisselingen groter zijn. Er wordt namelijk steeds maar een klein gedeelte van het bestand in het geheugen geladen. Daarna zal, wanneer de schijven zijn gewisseld, de inhoud van het geheugen weer worden weggeschreven naar de copie-schijf. Dit gaat zoalng door, totdat het gehele bestand is gecopieerd.

Samengevat kan worden gesteld, dat de orginele en copieschijf afwisselend in de schijveneenheid moeten worden geladen. Voordat het commando COPY wordt geactiveerd, dient

de orginele schijf in de eenheid te zitten. Het is zinvol om de schijf met het orginele bestand veilig te stellen met behulp van een "Write Protect"-sticker.

Wanneer in het tweede gedeelte van het COPY-formaat alleen maar de schijveneenheid (bijvoorbeeld "B:") wordt aangegeven, wordt het te copieren bestand naar de aangegeven schijveneenheid gecopieerd met de oorspronkelijke bestandsnaam (in het voorbeeld de naam TEST).

Wanneer het tweede gedeelte alleen uit een bestandsnaam bestaat, wordt het oorspronkelijke bestand gecopieerd naar de schijf op de default-schijveneenheid met de gespecificeerde bestandsnaam.

5.8 Afdrukken inhoudsopgave van een schijf.

Wanneer men wil weten welke bestanden er op een bepaalde schijf staan, dan kan men dit nagaan met het commando FILES. Het formaat van het commando FILES ziet er als volgt uit:

FILES "bestandsaanduiding"

Wanneer de bestandsaanduiding (zie paragraaf 5.1) volledig wordt weggelaten, zullen alle namen van de bestanden op de schijf in de laatst gebruikte schijveneenheid op het beeldscherm worden afgedrukt.

Wanneer in de bestandsaanduiding de schijveneenheid en een bestandsnaam is gegeven, zal, wanneer het bestand op de schijf is gevonden, de bestandsnaam op het beeldscherm worden afgedrukt. Voorbeeld:

FILES "A:TEST"

Wanneer het bestand niet wordt gevonden, zal de boodschap "File not found" op het scherm verschijnen.

Wanneer er in de bestandsaanduiding met het vraagteken (?) en de asterisk (*) wordt gewerkt, kunnen er meerdere bestanden met hetzelfde commando worden geselecteerd (zie ook paragraaf 5.1).

Voorbeelden:

FILES "B:*.BAS"
Laat op het beeldscherm zien welke bestanden met de bestandsnaamuitbreiding BAS zich op de schijf in schijveneenheid B bevinden.

FILES "A:G*.*"
Laat op het beeldscherm zien welke bestanden zich op de schijf in schijveneenheid A bevinden, waarvan de namen met een G beginnen.

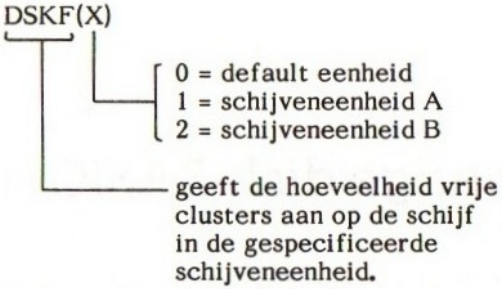
Wanneer men de bestandsnamen in plaats van op het beeldscherm door middel van een printer op papier wil laten afdrukken, dan kan hiervoor het commando LFILES worden gebruikt. Het formaat van het commando LFILES is volkomen gelijk aan dat van het commando FILES.

Voorbeeld:

LFILES "A:*.ASC"

5.9 Vrije ruimte op schijf.

Wanneer men wil weten wat de vrije ruimte is op een bepaalde schijf, kan dit worden opgevraagd met de functie DSKF. Bij de meeste systemen wordt de vrije ruimte op een schijf uitgedrukt in kilobytes. Dat wil zeggen, dat het resultaat van de functie DSKF een numerieke waarde is, die gelijk is aan het aantal kilobytes dat nog vrij is op de schijf. Het formaat van de functie DSKF is:



We kunnen de functie DSKF als volgt beproeven:

a. PRINT DSKF(1)

b. Schrijf een programma naar schijf met het commando SAVE.

SAVE "TEST"

c. PRINT DSKF(1)

Het resultaat uit punt c moet kleiner zijn dan het resultaat uit punt a.

6 Overige disk-BASIC statements

De hoofdstukken over disk-BASIC zullen worden afgesloten met de statements BSAVE, BLOAD, FORMAT en SYSTEM.

Het gebruik van BSAVE en BLOAD is alleen zinvol, wanneer u voldoende kennis hebt van de indeling van het geheugen en de instructieset van de microprocessor Z80.

Met het statement BSAVE zijn we in staat een machinetaalprogramma of de inhoud van het video-geheugen naar schijf te schrijven. Wil men de informatie welke met het statement BSAVE op schijf is opgeslagen weer van schijf in het geheugen laden, dan kan dit worden gedaan met het statement BLOAD.

Voordat een nieuwe schijf in gebruik kan worden genomen, dient de schijf eerst in sporen en sectoren te worden ingedeeld. Deze procedure wordt het "formateren" of het "initialiseren" van de schijf genoemd. Een schijf kan worden geformateerd met het statement CALL FORMAT.

Wanneer men MSX-BASIC wil verlaten en terug wil keren naar MSXDOS, kan dit worden bewerkstelligd met het statement CALL SYSTEM. Om dit statement beter te kunnen begrijpen, verdient het aanbeveling eerst de hoofdstukken 7 tot en met 12 door te nemen.

6.1 Opslaan en laden van machinetaalprogramma's.

Met het statement BSAVE kan een machinetaalprogramma direct vanuit het computergeheugen op schijf worden opgeslagen, onder een door u op te geven naam. Voor het opslaan van

machinetaalprogramma's op schijf, kent BSAVE de volgende twee formaten:

a.
BSAVE "schijveneenheid:bestandsnaam",
beginadres,eindadres

b.
BSAVE "schijveneenheid:bestandsnaam",
beginadres,eindadres,startadres

Voor **schijveneenheid** wordt A of B aangegeven. Wanneer de A of de B worden weggelaten, wordt door de computer automatisch de laatst gebruikte schijveneenheid genomen.

Bestandsnaam is de door uzelf gekozen naam voor het machinetaalprogramma. Zie voor de lengte van de naam hoofdstuk 5 (bestandsaanduiding).

Het machinetaalprogramma in het geheugen zal vanaf het in het formaat aangegeven **beginadres** tot en met het gegeven **eindadres** naar schijf worden geschreven. Het begin- en eindadres mogen niet kleiner zijn dan -32768 en niet groter dan 65536. Wanneer het adres kleiner is dan 0, wordt, voordat het adres wordt gebruikt, er eerst 65536 bij opgeteld. Het resultaat dient dan als begin- of eindadres.

De adressen kunnen ook worden opgegeven in het hexadecimale formaat. Bijvoorbeeld:

Beginadres: &H0000
Eindadres: &H4000

Het volgende statement plaatst een deel van het geheugen vanaf adres &H0000 tot en met adres &H4000 op de schijf in schijveneenheid A, onder de naam TEST.

```
100 BSAVE "A:TEST",&H0000,&H4000
```

Het **startadres** is optioneel. Het startadres is het geheugen-

adres, waar de uitvoering van het programma moet starten. Wanneer het startadres wordt weggelaten, is het beginadres automatisch het startadres van het programma. Voor de grootte van het startadres gelden dezelfde regels als voor het begin- en eindadres.

Voorbeeld van het statement BSAVE in een subroutine:

```
100 REM * VOORBEELD BSAVE *
110 CLS
120 INPUT "Programmanaam";P$
130 INPUT "Beginadres    ";B
140 INPUT "Eindadres     ";E
150 BSAVE "A:"+P$,B,E
160 RETURN
```

We kunnen bijvoorbeeld de totale inhoud van het ROM (machinetaalroutines), dat er voor zorgt, dat we in MSX-BASIC kunnen programmeren, op schijf vastleggen. Het begin adres is in dit geval 0 en het eindadres 32767.

Met het statement BLOAD kan een met het statement BSAVE naar schijf weggeschreven machinetaalprogramma weer worden geladen in het computergeheugen. Het programma zal in het geheugen worden geladen vanaf het adres, dat in het label (op schijf) van het programma staat. Dit wil zeggen, dat het programma op dezelfde plaats terecht komt, waar het ook stond, toen het met het statement BSAVE naar schijf werd geschreven.

Voor het laden van machinetaalprogramma's vanaf schijf kent BLOAD de volgende formaten:

a.
BLOAD "schijveneenheid:bestandsnaam"

b.
BLOAD "schijveneenheid:bestandsnaam",R

c.
BLOAD "schijveneenheid:bestandsnaam",offset

d.
BLOAD "schijveneenheid:bestandsnaam",R,offset

Voor **schijveneenheid** wordt A of B aangegeven. Wanneer de A of de B worden weggelaten, wordt door de computer automatisch de laatst gebruikte schijveneenheid genomen.

Bestandsnaam is de naam van het machinetaalprogramma, dat vanaf schijf in het computergeheugen moet worden geladen. Zie voor de lengte van de naam hoofdstuk 5 (bestandsaanduiding).

Wanneer de letter **R** (R-optie) is gespecificeerd, wordt het programma direct nadat het van schijf in het geheugen is geladen, automatisch gestart. Het startadres van het programma werd met het statement BSAVE ook op schijf vastgelegd.

Wanneer geen **offset** (numerieke waarde) is aangegeven, wordt het programma in het geheugen geladen vanaf het adres (beginadres), dat stond aangegeven in het statement BSAVE en dat tijdens uitvoering van BSAVE ook op schijf is geschreven. Wanneer een **offset** (verplaatsing) is aangegeven, wordt bij het beginadres, dat in het statement BSAVE stond aangegeven, een offset-waarde opgeteld. Dit geldt ook voor het eind- en startadres. Genoemde adressen worden tijdens de uitvoering van BSAVE ook op schijf geschreven.

Wanneer in het statement BLOAD een offset-waarde is aangegeven, wordt het programma in het geheugen geladen, vanaf het beginadres + offset. Hierdoor wordt een verplaatsing (verschuiving) van het programma in het geheugen verkregen. Wanneer geen offset is gegeven, zal de verschuiving 0 zijn. De regels voor de waarden, die voor offset mogen worden gebruikt, zijn dezelfde als die voor het beginadres en het eindadres in het statement BSAVE.

De programma's, die met een offset worden geladen, moeten "relocatable" zijn, dit wil zeggen, dat deze programma's voor uitvoering overal in het geheugen mogen worden geladen.

Voorbeeld van het statement BLOAD in een subroutine:

```
200 REM * VOORBEELD BLOAD *
210 CLS
220 INPUT "Programmanaam";P$
230 BLOAD "A:"+P$
240 RETURN
```

6.2 Opslaan en laden van inhoud video-geheugen.

MSX-BASIC beschikt over een 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.

Met het statement BSAVE zijn we in staat de inhoud van het video-geheugen in een bestand op schijf op te slaan. Dit kan worden uitgevoerd met de **S-optie** (screen-optie). Het formaat van BSAVE gaat er dan als volgt uit zien:

BSAVE "schijfveneenheid:bestandsnaam",
beginadres,eindadres,S

Voorbeeld:

```
100 REM * OPSLAAN VIDEO-GEHEUGEN *
110 SCREEN 3
120 OPEN "GRP:" AS #1
130 DRAW "BM40,0"
140 PRINT #1,"BSAVE"
150 DRAW "BM80,80"
160 PRINT #1,"EN"
170 DRAW "BM40,160"
180 PRINT #1,"BLOAD"
190 BSAVE "A:SCHERM",0,16383,S
200 END
```

Met voorgaand programma wordt het scherm samengesteld en vervolgens op schijf opgeslagen. Regelnummer 190 kan ook als volgt worden geschreven:

```
190 BSAVE "A:SCHERM",&H0000,&H4000,S
```

Wanneer we de beeldscherminformatie (screen image), die we met BSAVE op schijf hebben opgeslagen, weer willen laden in het video-geheugen, dan kan dit worden uitgevoerd met het statement BLOAD. Hiervoor dient weer de S-optie te worden toegepast. Het formaat van BLOAD ziet er voor het laden van beeldscherminformatie in het video-geheugen als volgt uit:

```
BLOAD "schijveneenheid:bestandsnaam",S
```

Voorbeeld:

```
200 REM * LADEN VIDEO-GEHEUGEN *
210 SCREEN 3
220 CLS
230 BLOAD "A:SCHERM",S
240 GOTO 240
```

Met voorgaand programma wordt de beeldscherminformatie, die met het vorige programma op schijf is geschreven, weer in het video-geheugen geladen. Zodra het video-geheugen door middel van het statement BLOAD is geladen, zal het beeld op het scherm verschijnen. In het programma dient er wel voor te worden gezorgd, dat voordat het statement BLOAD wordt uitgevoerd, de juiste beeldschermmode (SCREEN 0, 1, 2 of 3) is ingesteld.

Opmerking:

De S-optie kan alleen worden gebruikt in combinatie met schijveneenheid A of B.

6.3 Formateren van schijven.

Wanneer u een schijf koopt, dan is de schijf normaal helemaal leeg. Dit houdt in, dat er geen bestanden op staan en zelfs geen sporen (tracks) en sectoren. Op zo'n ongeformateerde schijf is het niet mogelijk gegevens te schrijven.

Het indelen van een schijf in sporen en sectoren en het voorbereiden van de inhoudsopgave (directory) van de schijf, wordt formateren of initialiseren genoemd. Zie voor een verdere uitleg van het formateren van een schijf hoofdstuk 9, paragraaf 2 (Formateren van een schijf).

Het formateren van een schijf kan in MSX-BASIC worden uitgevoerd door middel van het statement FORMAT. MSX-BASIC kan worden uitgebreid met een aantal statements, door een ROM-cartridge toe te passen. Tot deze statements behoort ook het statement FORMAT. De statements, die tot de uitbreidingsset behoren, kunnen worden opgeroepen met het statement CALL. Het statement FORMAT, uit de uitbreidingsset, wordt als volgt opgeroepen:

```
CALL FORMAT
```

Wanneer CALL FORMAT wordt uitgevoerd, volgt het volgende menu op het scherm:

```
Drive name? (A,B)
```

Na het invoeren van de naam van de schijveneenheid (A of B), volgt de volgende boodschap op het scherm:

```
Strike a key when ready
```

Nadat u de te formateren schijf in de schijveneenheid hebt geladen, dient u een willekeurige toets in te drukken. Wanneer het formateren is beeindigd, geeft MSX-BASIC de volgende boodschap op het scherm:

```
Format complete
```

Opmerkingen:

Wanneer een reeds gebruikte schijf wordt geformateerd, zullen na het formateren alle bestanden, die op de schijf stonden, zijn vernietigd.
Nieuwe schijven moeten altijd worden geformateerd.
De formateringsprocedure kan voor bepaalde systemen soms afwijken van de hiervoor genoemde beschrijving.

6.4 Terugkeren naar MSXDOS.

Met het statement SYSTEM kan MSX-BASIC worden verlaten en worden teruggekeerd naar MSXDOS. Zie voor verdere beschrijving hoofdstuk 9, paragraaf I.

Het statement SYSTEM, dat tot de uitbreidingsset behoort, kan worden aangeroepen met CALL. Het aanroepen gaat als volgt in zijn werk:

```
CALL SYSTEM
```

CALL SYSTEM kan alleen worden gebruikt, als BASIC werd ge-"boot" vanuit MSXDOS. Door middel van CALL SYSTEM zullen alle bestanden worden gesloten en het programma en de gegevens in het geheugen verloren gaan.

7 Het besturingssysteem MSXDOS

Tot nu toe hebben we in deze serie leerboeken alleen gesproken over het programmeren van de computer in BASIC. Daarbij hebben we er nauwelijks aandacht aan geschonken, dat er in de computer al een groot aantal subroutines aanwezig zijn. Alleen in het begin van het eerste leerboek is kort ingegaan op de "software", die door de fabrikant in het ROM-geheugen van de computer is gezet. Daarbij zagen we dat die software hoofdzakelijk bestaat uit een vertaalprogramma, dat de door ons ingevoerde BASIC-statements omzet naar voor de computer begrijpelijke instructies.

In deel twee van de serie leerboeken werd in het hoofdstuk over machinetaal en het aanroepen van machinetaalroutines nog eens teruggekomen op de software, die in het ROM zit opgeslagen. Daar zagen we, dat we een groot aantal routines uit die ROM kunnen aanroepen met behulp van de zogenaamde BIOS entry points.

Wij hebben er niet op gewezen, en u hebt het zich waarschijnlijk niet gerealiseerd, maar, al die routines, die in het ROM zijn opgeslagen en die geen deel uitmaken van het vertaalprogramma (de interpreter) vormen samen het operating systeem van de MSX-computer. Daar de basis uitvoering van de MSX-computer geen erg ingewikkelde apparaten te besturen heeft, kan dat operating systeem eenvoudig zijn. Je zou "operating systeem" kunnen vertalen met "besturingssysteem". Hoe ingewikkelder de te besturen apparatuur is, hoe ingenieuzer het besturingssysteem moet zijn.

Op het moment dat we een schijveneenheid aan onze computer aansluiten, voegen we een zeer ingewikkeld randapparaat

aan ons systeem toe. Voor die schijveneenheid moet de basis-uitvoering van de computer dan ook worden uitgebreid met een aantal tamelijk ingewikkelde routines. Zolang er echter niets anders dan een aantal onafhankelijk van elkaar werkende routines in het geheugen van de computer staat, spreken we nog niet van een operating systeem. Worden die losse routines echter vanuit een centraal programma aangeroepen, dan vormen al die losse routines tesamen een geheel. Dat geheel duiden we dan aan met een operating systeem.

Bij de aanschaf van een schijveneenheid leveren een aantal fabrikanten en importeurs zo'n operating systeem. Het betreft dan MSX-DOS. MSX-DOS is een Disk Operating Systeem voor MSX-computers. In de rest van dit leerboek zullen we gaan zien wat MSX-DOS precies is, wat het allemaal voor ons kan doen en hoe we MSX-DOS kunnen opdragen om dingen voor ons te doen.

7.1 Wat is MSX-DOS?

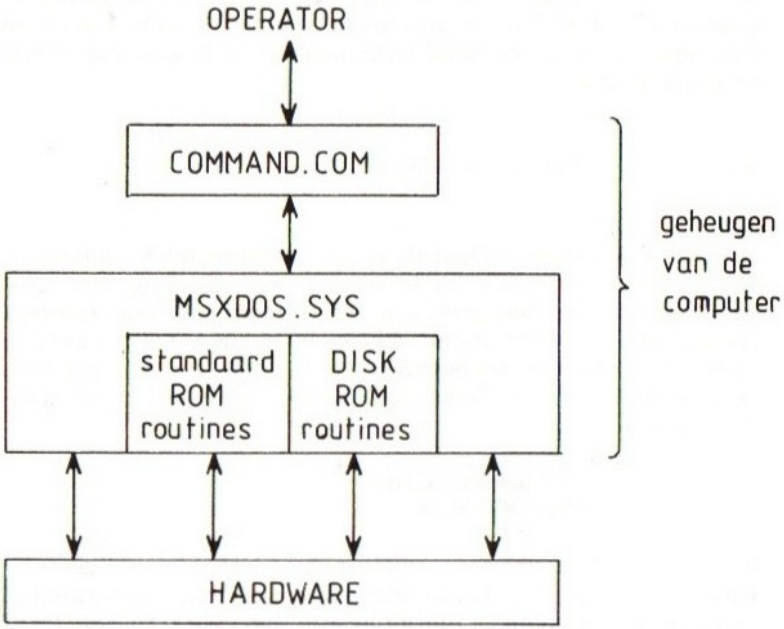
MSX-DOS is het Disk Operating System voor MSX-computers. Het is een programma (of liever een verzameling van routines), dat ons in staat stelt om een ingewikkeld computersysteem, waarin onder meer schijveneenheden voorkomen, op eenvoudige manier te besturen. MSX-DOS bestaat uit twee programma's, die op een schijf worden geleverd. Deze twee programma's zijn:

COMMAND.COM
MSXDOS.SYS

Bij het opstarten van de computer, met aangeschakelde schijveneenheid, worden beide hiervoor genoemde programma's automatisch geladen in het geheugen van de computer. Daartoe dient de schijf, met daarop de beide programma-files, natuurlijk wel in de schijveneenheid te zijn geladen.

Op deze plaats is een belangrijke aanwijzing op zijn plaats.

Zodra de schijf met de systeemprogramma's MSXDOS en COMMAND in de schijveneenheid is geladen, wordt MSXDOS automatisch in het geheugen gelezen, bij aanschakelen van de schijveneenheid en de computer. Het is echter mogelijk om dit te voorkomen, door tijdens het aanschakelen van de computer de SHIFT-toets in te drukken. In dat geval wordt MSXDOS niet geladen. In plaats daarvan wordt Disk-BASIC direct actief. Probeert u dit maar eens uit, voordat u verder gaat met het bestuderen van dit hoofdstuk. U kunt van dit verschijnsel later nog veel plezier hebben. Vandaar dat we er hier zo grote nadruk op leggen.



Afb. 7-1 Functie en plaats van MSXDOS.

Zijn de programma's eenmaal geladen in het geheugen van de MSX-computer, dan ontstaat de situatie zoals die is weergegeven in afbeelding 7-1. Via het toetsenbord geeft de "operator" opdrachten aan het systeem, ofwel aan het programmadeel COMMAND.COM. Dit overkoepelende programma vertaalt de opdrachten in MSXDOS opdrachten. MSXDOS.SYS bestaat uit een soort interface tussen COMMAND.COM en de in het ROM van de computer staande routines plus een aantal routines, die op hetzelfde niveau liggen als de ROM-routines. De ROM-routines en de MSXDOS-routines sturen vervolgens de hardware van de computer en de aangesloten randapparaten aan en vangen de resultaten van de uitgevoerde acties op, om die resultaten weer terug te geven aan de gebruiker.

7.2 De MSXDOS-commando's.

Zoals gezegd, kunnen we aan COMMAND.COM opdrachten geven. COMMAND.COM wordt dan ook wel aangeduid met een commando-interpreter. Het vertaalt de door ons gegeven commando's in door MSXDOS verstaanbare commando's. Natuurlijk kent COMMAND.COM slechts een beperkt aantal verschillende commando's. Hierna volgt een korte opsomming van alle commando's die kunnen worden gegeven. De commando's zullen in de volgende hoofdstukken uitgebreid worden behandeld.

BASIC
Met dit commando wordt overgeschakeld naar MSX-BASIC. Hierna werkt u niet langer met MSXDOS. Om terug in MSXDOS te komen, dient u het BASIC-commando CALL SYSTEM te geven.

COPY
Met dit commando kunnen zowel bestanden als gehele schijven worden gecopieerd. Bovendien is het niet beslist noodzakelijk dat de te copieren bestanden op een schijf staan. Het is bijvoorbeeld ook mogelijk om een bestand op een toetsenbord in te tikken en naar een schijf te copieren. Het BASIC-commando COPY heeft grotendeels dezelfde functie.

DATE
Na het ingeven van dit commando kan de datum worden ingegeven. De ingegeven datum wordt gebruikt in de administratie van de files op de schijf.

DEL
Met dit commando kunnen een of meerdere bestanden van de schijf worden gewist. Dit commando komt overeen met het BASIC-commando KILL.

DIR
Hiermee wordt een inhoudsopgave van de schijf op het beeldscherm, of indien gewenst op een printer, afgedrukt. Dit commando kunnen we zien als een uitgebreide versie van het BASIC-commando FILES.

ERASE
Dit is een synoniem voor het commando DEL. Het doet exact hetzelfde en is dus eveneens te vergelijken met het BASIC-commando KILL.

FORMAT
Met dit commando wordt een flexibele schijf geinitialiseerd, hetgeen wil zeggen, dat er sectoren op de schijf worden geschreven en dat een (nog lege) inhoudstabel op de schijf wordt geschreven. Vanuit BASIC kan de initialiseringsroutine worden aangeroepen met het statement CALL FORMAT.

MODE
Dit commando komt gedeeltelijk overeen met het BASIC-commando WIDTH. Wordt achter mode een getal van 32 of kleiner opgegeven, dan wordt SCREEN 1 geactiveerd. Daarboven wordt SCREEN 0 actief.

PAUSE
Met dit commando kan de uitvoering van programma's worden gestopt, totdat de gebruiker een toets heeft ingedrukt.

REM
Dit commando komt overeen met het BASIC-commando REM.

De tekst achter een MSXDOS-REM wordt echter bij uitvoering op het beeldscherm afgedrukt.

RENAME
Met dit commando kan aan een of meerdere files een nieuwe naam worden gegeven. De werking van dit commando komt overeen met het BASIC-commando NAME. RENAME mag ook worden geschreven als REN.

TIME
Na het ingeven van dit commando kan de juiste tijd worden ingegeven. De tijd wordt onder meer gebruikt in de administratie van de bestanden op de schijven.

TYPE
De inhoud van het opgegeven bestand wordt met dit commando op het beeldscherm afgedrukt. Indien de printer is geactiveerd, zal het bestand op de printer worden afgedrukt.

VERIFY
Door het commando VERIFY ON of VERIFY OFF te geven, wordt MSXDOS opgedragen gegevens die naar schijf worden geschreven na het schrijven terug te lezen en te vergelijken met de weggeschreven gegevens.

7.3 Een voorbeeld van het geven van MSXDOS-commando's.

Een MSXDOS-commando kan alleen worden gegeven, wanneer MSXDOS in het geheugen van onze MSX-computer is geladen. We zullen daarom eerst stap voor stap gaan zien hoe we dat doen. Daarbij gaan we ervan uit, dat de schijveneenheid is aangesloten aan de computer. Het aansluiten van de schijveneenheid is in de bij de schijveneenheid geleverde documentatie beschreven.

We plaatsen nu de schijf, waarop het MSXDOS-systeem staat, in de schijveneenheid. Bij sommige schijveneenheden is het noodzakelijk om de schijf te vergrendelen. Vervolgens schakelen we de schijveneenheid aan. Hierna schakelen we de com-

puter en het beeldscherm (TV of monitor) aan. U zult nu merken, dat de schijf na enige tijd gaat draaien en dat het wat langer duurt voordat u op het scherm gaat zien dat de computer werkelijk iets doet. Na enige tijd zult u de volgende tekst op het scherm zien verschijnen:

```
MSX-DOS version 1.00
Copyright 1984 by Microsoft

COMMAND version 1.01

Current date is Sun  1-01-1984
Enter new date:
```

Dit wil zeggen, dat MSXDOS in het geheugen is geladen. De software fabrikant, Microsoft, heeft er voor gezorgd, dat er altijd een datum aan de software bekend is. Er is gekozen voor de datum januari 01 1984. Die datum valt, zoals in voorgaand voorbeeld is te zien, op een zondag. Daar er mag worden verwacht, dat wij de computer dag na dag opnieuw aanzetten, dus steeds op een andere datum, geeft MSXDOS ons de gelegenheid om de werkelijke datum, van de dag waarop we de computer aanzetten, in te geven. MSXDOS vraagt ons naar die datum met de tekst:

```
Enter new date:
```

Zouden we nu alleen op de RETURN-toets drukken, zonder een datum in te geven, dan blijft de datum 1-01-1984 van kracht. Willen we wel een datum ingeven, dan moeten we aan de volgende regels voldoen.

```
Eerst het nummer van de maand.
Dan het nummer van de dag.
Tenslotte het jaartal.
```

Deze drie getallen dienen van elkaar te worden gescheiden door het koppelteken of door een breukstreep. Zouden we bijvoorbeeld de datum 14 maart 1986 willen ingeven, dan zullen we dat als volgt moeten doen:

3-14-1986 of 3/14/1986

Het is ook toegestaan om in plaats van het volledige jaartal, alleen de laatste twee cijfers van het jaartal in te geven. In dat geval dient u er echter rekening mee te houden dat alle waarden van 00 tot en met 79 zullen worden gerekend tot de jaren 2000 tot en met 2079 en dat de getallen 80 tot en met 99 tot de jaren 1980 tot en met 1999 zullen worden gerekend. Het verkort ingeven van het jaartal kunnen we weer laten zien aan de hand van de datum 14 maart 1986. We kunnen dat als volgt doen:

3-14-86 of 3/14/86

Na het ingeven van de datum, op de hiervoor beschreven manier, krijgen we de volgende tekst op het beeldscherm te zien:

```
Current date is Fri 3-14-1986
A>
```

Sommige systemen zullen in plaats van de tekst "A>" vertellen wat de momentele tijd is, en u in de gelegenheid stellen een nieuwe tijd in te geven. In dat geval zult u op het scherm zien:

```
Current time is 12:00:00a
Enter new time:
```

Nu kunt u weer kiezen of u wel of niet een nieuwe tijd wilt ingeven. Wilt u dat niet, dan drukt u gewoon op de RETURN-toets. Wilt u dat wel, dan moet u bij het ingeven van de tijd de volgende regels in acht houden:

Eerst het gewenste uur.
Eventueel gevolgd door de minuten.
Eventueel gevolgd door de seconden.

Indien u alleen een getal voor de uren intikt, dan wordt het aantal minuten en seconden door MSXDOS op 0 gezet. Indien

u dat wenst, kunt u het aantal uren laten volgen door het aantal minuten. Wilt u de tijd heel erg nauwkeurig invullen, dan kunt u eventueel de minuten nog laten volgen door het aantal seconden. Uren, minuten en seconden dienen van elkaar te worden gescheiden door een dubbele punt (:).

Op het beeldscherm ziet u dat achter de tijd een letter is afgedrukt, en wel de letter a. Deze letter stamt af van de Engelse aanduiding AM (Anti Meridium), hetgeen wil zeggen "Voor de Middag", ofwel voor 12 uur 's middags. Bij het ingeven van de tijd mogen wij deze letter ook intikken. Dit mag zowel wanneer we alleen de uren intikken, als wanneer we uren en minuten of uren, minuten en seconden intikken. Gebruiken we de letter a als toevoegsel, dan mogen we voor de uren alleen de getallen 0 tot 12 gebruiken. Gebruiken we het toevoegsel niet, dan mogen we voor de uren de getallen 0 tot 24 gebruiken.

Om aan te geven dat we een tijd willen ingeven, die na de middag ligt, mogen we het toevoegsel p gebruiken. Die "p" stamt af van de Engelse aanduiding PM (Post Meridium), hetgeen betekent "Na de Middag", ofwel na 12 uur 's middags. Ook bij gebruik van dit toevoegsel mogen we voor de uren alleen de getallen 0 tot 12 gebruiken. Zouden we bijvoorbeeld de volgende tijd ingeven:

```
2:30:00p
```

dan zal de "p" tot gevolg hebben, dat er bij het aantal uren 12 wordt opgeteld. De werkelijke tijd is dan dus 12 + 2 = 14 uur en 30 minuten. Anders gezegd: halfdrie 's middags.

Na het ingeven van de datum en, indien gevraagd, de tijd, zal uiteindelijk de MSXDOS-prompt op het scherm verschijnen. Dit is een aanduiding, waaraan u kunt zien, dat u een MSXDOS-commando mag ingeven. De MSXDOS-prompt ziet er als volgt uit:

```
A>
```

De A geeft hierin aan, dat op dit moment schijf A actief is. Er kunnen meerdere schijveneenheden tegelijk aan de computer worden aangesloten. De eerste schijveneenheid wordt dan aangeduid met de letter A, de tweede met de letter B. Laten we er voorlopig even van uitgaan, dat we maar een schijveneenheid aan onze computer hebben aangesloten. Die eenheid wordt dan aangeduid met de letter A. Zouden we nu willen weten, welke bestanden en programma's er op de schijf staan, die in de eenheid zit, dan kunnen we MSXDOS vragen om dat voor ons uit te zoeken. Hiertoe dienen we dan het MSXDOS-commando DIR in te geven.

```
A>dir
MSXDOS   SYS     2432  1-01-84
COMMAND  COM     6272  1-01-80
       2 files   170496 bytes free
```

Daar de schijveneenheid A actief is, zal het commando DIR tot gevolg hebben, dat er een inhoudsopgave van de schijf in eenheid A wordt gemaakt. In het voorbeeld ziet u, dat op die schijf slechts twee bestanden staan; MSXDOS.SYS en COMMAND.COM. De punt, tussen de bestandsnaam en de extensie, wordt niet afgedrukt.

Er wordt echter meer informatie gegeven dan alleen de bestandsnamen. Achter iedere naam ziet u een getal. Dit getal geeft de grootte van het bestand aan. U ziet, dat MSXDOS 2432 bytes groot is. In de laatste kolom ziet u de datum, waarop het bestand op de schijf is gezet, of beter uitgedrukt, de datum die werd ingetikt bij het opstarten van de computer, vlak voordat de bestanden naar de schijf werden geschreven.

Dit geeft meteen het belang aan van het intikken van de juiste datum tijdens het opstarten van MSXDOS. Zodra u een bestand creeert, wordt de huidige datum samen met de naam en de grootte van het bestand op de schijf genoteerd. Dit kan later handig zijn, omdat u dan altijd kunt zien wanneer u iets op de schijf hebt gezet. De laatste regel van de inhoudsopgave informeert ons over het totale aantal bestanden op de

schijf en het totale aantal bytes dat nog vrij is. Met andere woorden het aantal bytes dat we nog naar de schijf kunnen schrijven voordat deze vol raakt.

Bij bijna alle MSXDOS-commando's kunnen een of meerdere parameters worden opgegeven. Om aan te geven wat daarmee wordt bedoeld, het volgende voorbeeld, waarbij het DIR-commando wordt voorzien van een parameter.

```
A>dir /w
MSXDOS    SYS     COMMAND  COM
        2 files    170496 bytes free
```

Behalve het commando DIR is de parameter /w ingegeven. De letter w staat voor het Engelse woord "wide", hetgeen zoveel wil zeggen als "breed". U ziet dat het resultaat is, dat er twee bestandsnamen naast elkaar worden afgedrukt. Hiertoe is echter de informatie over de bestandsgrootte en de datum waarop het bestand werd gemaakt weggelaten. Indien er veel bestanden op een schijf staan, dan kan het echter overzichtelijker zijn op de inhoudsopgave als hiervoor beschreven te laten afdrukken.

Er kunnen bij DIR nog andere parameters worden opgegeven, doch bij de behandeling van het commando in een volgend hoofdstuk, zullen we alle mogelijkheden stuk voor stuk onder de loep nemen.

7.4 Commando-soorten.

MSXDOS onderscheidt twee soorten commando's. De eerste soort is die, die binnen MSXDOS zijn opgenomen. Al deze commando's zitten in COMMAND.COM verwerkt. Wanneer wij een commando ingeven, dan zal MSXDOS controleren of het dit commando kent. Zo ja, dan zal het commando worden uitgevoerd. Is het echter geen commando dat binnen MSXDOS bekend is, dan zal MSXDOS de schijf onderzoeken op het voorkomen van een bestand met de extensie .COM of .BAT, om te zien of er een bestand is waarvan de naam overeen-

komt met het gegeven commando.

De eerste groep commando's wordt aangeduid met intrinsieke commando's. Dit zijn dus commando's die binnen MSXDOS bekend zijn. De tweede groep commando's wordt aangeduid met extrinsieke commando's. Dit zijn commando's die van de schijf moeten worden ingelezen, voordat ze kunnen worden uitgevoerd.

De extrinsieke commando's, de op de schijf staande bestanden met de extensie .COM of .BAT, zijn in feite losse programma's. .COM- files worden, indien wij de naam daarvan intikken, tijdelijk in het geheugen geladen. Wordt zo'n programma in het geheugen geladen, dan wordt het daar al staande programma overschreven. Met andere woorden het programma COMMAND.COM zal worden overschreven. Zodra het andere programma echter is uitgevoerd, zal het oorspronkelijk programma worden teruggelezen in het geheugen.

Verderop in dit boek zullen we gaan zien hoe we zelf een bestaan in feite uit een verzameling van commando's, die stuk voor stuk zullen worden uitgevoerd door MSXDOS. Dit is te vergelijken met het uitvoeren van de programmaregels van een BASIC-programma door de interpreter.

De .COM-files worden gevormd door machinetaalprogramma's. Bij de standaard MSXDOS-software werden geen extra machinetaalprogramma's geleverd, en het zou te ver voeren om zelf dergelijke programma's te gaan maken. We zullen daarvan in dit boek dan ook geen voorbeelden zien. Er is echter een .COM-file, en wel het bestand COMMAND.COM. U kunt deze .COM-file gerust starten. Daartoe geeft u het volgende commando in:

```
A>COMMAND
```

U zult de schijf even horen werken, waarna op het scherm de melding:

A>

verschijnt. Ogenschijnlijk is er niets gebeurd, doch het programma COMMAND.COM, dat al in het geheugen stond, is nu overschreven met het programma COMMAND.COM, dat zojuist van schijf werd gelezen. Het eerste programma stond te wachten op een commando. Dat commando bestond eruit dat het programma moest worden overschreven met zichzelf. De tweede versie van het programma staat weer te wachten op een commando. Het lijkt dus of er helemaal niets is gebeurd.

8 MSXDOS helpt de operator

In het vorige hoofdstuk hebben we gezien wat MSXDOS is, welke commando's het voor ons kan uitvoeren en hoe we zo'n commando kunnen intikken. In dit hoofdstuk zullen we zien, dat MSXDOS ons ook nog behulpzaam kan zijn bij het intikken van de commando's. Verder zullen we in dit hoofdstuk zien, hoe we de resultaten van onze commando's kunnen stoppen of onderbreken en hoe we ze naar de printer kunnen sturen. Tenslotte zullen we zien hoe MSXDOS doet alsof er twee schijveneenheden zijn, terwijl er in werkelijkheid maar een schijveneenheid aan de computer is aangesloten.

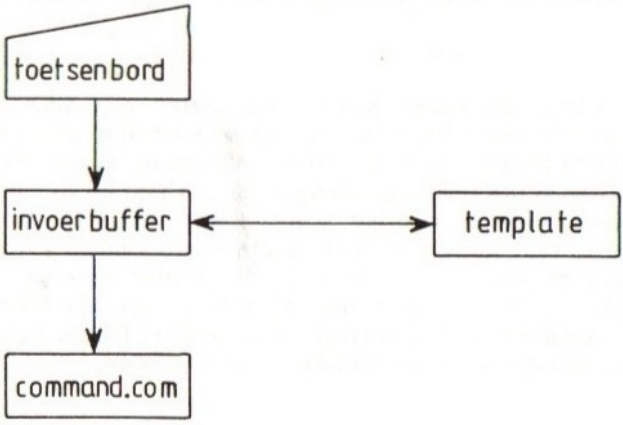
8.1 Hulp bij het intikken van commando's.

Wanneer MSXDOS in het geheugen van onze computer is geladen, gaat ook het intikken van commando's onder controle van MSXDOS. Om ons het leven wat gemakkelijker te maken, helpt MSXDOS ons bij het intikken van commando's, door te onthouden wat we de laatste keer hadden ingetikt. In afbeelding 8-1 is weergegeven, hoe dit in zijn werk gaat.

U ziet, dat wat wij intikken, in een buffer wordt gezet, het zogenaamde invoerbuffer. Op het moment dat we de RETURN-toets indrukken, wordt het buffer uitgelezen door MSXDOS, en wordt de in het buffer staande tekst door COMMAND.COM gecontroleerd. COMMAND.COM zal het commando, indien het geldig is, uitvoeren.

Zodra het invoerbuffer is uitgelezen, wordt het weer leeg gemaakt, opdat wij een volgend commando kunnen ingeven. Er gebeurt echter nog iets. Op het moment dat we de RETURN-toets indrukten, werd de inhoud van het invoerbuffer niet alleen aan COMMAND.COM doorgegeven, maar werd

het invoerbuffer ook gecopieerd naar een ander buffer. Dit andere buffer heeft van Microsoft (de fabrikant van MSXDOS) de naam "template" gekregen.



Afb. 8-1 Invoeren van commando's onder MSXDOS.

Als een commando, dat wij hebben ingegeven, is uitgevoerd, dan is het invoerbuffer leeg. Wij kunnen nu het invoerbuffer weer vullen, ofwel via het toetsenbord, door gewoon een nieuw commando in te tikken, ofwel via een aantal eenvoudige handelingen, met onder meer de cursor control toetsen. Met deze laatste manier kunnen we de inhoud van het "template" geheel of gedeeltelijk copieren naar het invoerbuffer. Bovendien kunnen we tijdens dat copieren letters wissen of toevoegen. In het vervolg van deze paragraaf zullen we zien hoe dit in zijn werk gaat.

Voor een goed begrip is het van belang te weten wat we op het beeldscherm zien. Achter de MSXDOS-prompt wordt onze ingave afgedrukt. Wat we daar zien is in feite het invoerbuffer. Steeds wanneer we een letter aan het invoerbuffer toevoegen, wordt het invoerbuffer opnieuw op het beeldscherm afgedrukt. Het lijkt dan net alsof we direct van het toetsen-

bord naar het beeldscherm typen, doch we dienen wel te bedenken dat wij naar het invoerbuffer schrijven en dat het invoerbuffer op het beeldscherm wordt afgedrukt.

Laten we eerst eens zien welke toetsen we bij het werken met de "template" allemaal kunnen gebruiken en wat de functie van iedere toets daarbij is. Het volgende overzicht laat dit zien:

Cursor rechts
Copieert het volgende teken uit de template naar het invoerbuffer. Iedere keer dat deze toets wordt ingedrukt, wordt slechts 1 teken gecopieerd.

Cursor naar beneden
Copieert alle nog resterende tekens uit de template naar het invoerbuffer. Indien nog geen enkel teken uit de template naar het invoerbuffer was gecopieerd, dan zal het gehele template naar het invoerbuffer worden gecopieerd.

Cursor links
Wist het laatste teken uit het invoerbuffer. In plaats van de cursor toets mag ook de Back Space toets (BS) worden gebruikt.

Cursor omhoog
Maakt het invoerbuffer leeg, terwijl de inhoud van het template wordt bewaard. Door na het indrukken van de cursor omhoog toets de cursor naar beneden toets in te drukken, ziet u dat het template inderdaad nog steeds met de orginele tekst is geladen.

SELECT
Door het indrukken van de SELECT-toets, gevolgd door een letter, zullen alle tekens uit het template tot aan de opgegeven letter naar het invoerbuffer worden gecopieerd.

DEL
Het indrukken van de DEL-toets heeft tot gevolg dat het volgende teken uit het template niet naar het invoerbuffer

wordt gecopieerd.

CLS
Het indrukken van de CLS-toets (dit is de SHIFT-toets + CLS-toets), gevolgd door een letter, worden alle tekens uit het template, tot de opgegeven letter, overgeslagen (niet gecopieerd).

INS
Door de INS-toets in te drukken wordt de insert-mode aangezet indien deze uit was en uitgezet indien deze aan was. Dit geeft de mogelijkheid om een of enkele tekens tussen te voegen in het invoerbuffer.

HOME
Het indrukken van de HOME-toets heeft tot gevolg dat het invoerbuffer naar het template wordt gecopieerd.

Dit zijn alle toetsen, die we bij het intikken van een commando, met gebruikmaking van het template, kunnen gebruiken. Omdat voorgaande omschrijving nog erg theoretisch is, zullen we nu een aantal voorbeelden laten zien. Daarbij zal de functie van alle toetsen volledig duidelijk worden. Daar we nog maar weinig commando's kennen zullen we zomaar een reeks tekens intikken. Dit heeft bovendien tot gevolg dat de resultaten van het gebruik van de toetsen duidelijker zichtbaar wordt.

Tik na het opstarten van MSXDOS (de MSXDOS-prompt moet op het scherm staan) het volgende commando in:

```
A>abcdefghijklmnop
```

Daar dit een niet bestaand commando is, zal MSXDOS hierop reageren met de boodschap "Bad command of file name". Desalniettemin is dit commando door het indrukken van de RETURN-toets wel in het template opgenomen. Vandaar, dat we nu met behulp van de speciale toetsen, het template terug kunnen copieren naar het invoerbuffer. De situatie op het beeldscherm is nu als volgt:

```
A>abcdefghijklmnop
Bad command or file name

A>
```

Drukken we nu op de **Cursor naar beneden toets** dan zien we dat het invoerbuffer plotseling weer is geladen met de tekens, die we bij het vorige commando hadden ingetikt. Met andere woorden, het template is gecopieerd naar het invoerbuffer. Op het scherm zien we nu staan:

```
A>abcdefghijklmnop
Bad command or file name

A>abcdefghijklmnop
```

Nu kunnen we met de **Cursor links toets** of de **BS-toets** het laatste teken uit het invoerbuffer wissen. Probeer het maar eens. Iedere keer dat u op de BS-toets drukt (of de Cursor links toets), wordt het op dat moment laatste teken uit het invoerbuffer gewist. Om alle tekens weer in het invoerbuffer terug te krijgen, behoeven we alleen de **cursor naar beneden toets** in te drukken. Als u dit hebt gedaan, dan staan de letters a tot en met p weer op het beeldscherm.

Door nu de **cursor omhoog toets** in te drukken, worden alle letters a tot en met p van het beeldscherm gewist. Deze letters staan echter nog wel in de template. Druk maar eens op de cursor naar beneden toets. Ziet u wel? Verwijder dan nu weer alle tekens uit het invoerbuffer, door op de cursor omhoog toets te drukken.

Op het beeldscherm staat nu alleen de MSXDOS-prompt. Druk nu eens op de **cursor rechts toets.** U ziet de letter a verschijnen. Druk nog twee maal op de zelfde toets, zodat er op het beeldscherm "abc" staat. Druk nu eenmaal op de **INS-toets** en tik een aantal keren de letter x in. Druk nogmaals op de INS-toets, om de insert-mode weer uit te schakelen, en druk nu op de cursor naar beneden toets. Het gevolg dient te zijn, dat er het volgende op het beeldscherm staat:

```
A>abcxxxdefghijklmnop
```

Nogmaals zij er op gewezen, dat het beeldscherm de inhoud van het invoerbuffer weergeeft. In het template staat dus nog steeds de tekst "abcdefghijklmnop". Zou u de inhoud van het template willen vervangen door de inhoud van het invoerbuffer, met andere woorden, zou u het invoerbuffer naar het template willen copieren, dan kunt u dat doen door op de **HOME-toets** te drukken.

Om te bewijzen, dat het template nu inderdaad de inhoud van het invoerbuffer heeft gekregen, kunt u het invoerbuffer wissen (door de cursor omhoog toets in te drukken) en vervolgens het template naar het invoerbuffer copieren (door de cursor naar beneden toets in te drukken). U ziet nu dat het beeldscherm er als volgt uit ziet:

```
A>abcxxxdefghijklmnop
```

Deze tekst is vanuit de template naar het invoerbuffer gecopieerd. De inhoud van het template is dus inderdaad gewijzigd.

Laten we nu eens zien, hoe we op eenvoudige manier de letters xxx weer verwijderen. Druk eerst op de RETURN-toets. Uitvoering van het commando zal resulteren in de boodschap "Bad command or file name". Het invoerbuffer is hierna leeg, terwijl het template nog is gevuld met de tekst "abcxxxdefghijklmnop". Vanuit deze situatie gaan we zien hoe we de het template naar het invoerbuffer kunnen copieren, zonder de letters xxx.

Druk op de **SELECT-toets** en tik vervolgens de letter x in. Nu worden alle tekens tot aan de letter x vanuit het template naar het invoerbuffer gecopieerd. In ons voorbeeld zijn dat de letters abc. Door nu de **DEL-toets** drie keer in te drukken, zullen de volgende drie tekens uit het template niet worden gecopieerd. Deze drie letters zijn xxx. Drukken we vervolgens de cursor naar beneden toets in, dan wordt de rest van het template (defghijklmnop) naar het invoerbuffer gecopieerd.

Het gevolg is, dat het invoerbuffer nu de tekst "abcdefghijklmnop" bevat. Denk er echter wel aan, dat het template nog steeds "abcxxxdefghijklmnop" bevat.

Laten we nu, om weer dezelfde uitgangspositie te verkrijgen als zoeven, de cursor omhoog toets indrukken. Dit maakt het invoerbuffer weer leeg. Het template bevat nog steeds de tekst "abcxxxdefghijklmnop". Stel dat we nu alleen de letters d tot en met p in het invoerbuffer willen copieren. Dat kunnen we bereiken door eerst de **CLS-toets** in te drukken en vervolgens de letter d. (Denk er om dat de CLS-toets wordt geactiveerd door eerst de SHIFT-toets en daarna, zonder de SHIFT-toets los te laten, de CLS-toets.) Het gevolg van de tot nu toe genomen actie is, dat alle tekens tot aan de letter d **niet** worden gecopieerd. Als we dan ook nú de cursor naar beneden toets indrukken, zal de rest van het template, vanaf de letter d dus, naar het invoerbuffer worden gecopieerd. Hierna bevat het invoerbuffer dus de tekst:

"defghijklmnop".

Hiermee hebben we voorbeelden gezien van alle functies van de toetsen, die bij het copieren van en naar het template van belang zijn. Om hiervan goed gebruik te kunnen maken, is verdere oefening nodig. Wij raden u dan ook aan, ook al zal dat in het begin niet snel gaan, zoveel mogelijk van deze toetsen gebruik te maken bij het invoeren van commando's. Na enige tijd zult u merken dat u er erg veel baat bij hebt en dat u veel minder tikwerk hoeft te verrichten.

8.2 Controle via het toetsenbord.

Door het indrukken van de CONTROL-toets samen met een letter, kunnen we de uitvoering van de commando's beinvloeden. Niet alle letters zijn toegestaan. De letters die wel zijn toegestaan worden hierna genoemd, samen met een korte beschrijving van het gevolg van het indrukken ervan.

CONTROL + P
Activeert de printer. Alles wat normaal naar het beeldscherm

wordt gestuurd, wordt hierna ook naar de printer gestuurd. Indien de printer niet aan staat, of helemaal niet is aangesloten, zal MSXDOS net zo lang wachten tot de printer wel is aangesloten en aangeschakeld. Heeft u geen printer, gebruik deze functie dan niet. Doet u dat toch, dan lijkt het alsof de computer in een "hang-up" situatie is terechtgekomen.

CONTROL + N
Hiermee wordt de na CONTROL + P verkregen situatie beeindigd. Er zal niets meer worden afgedrukt op de printer en MSXDOS zal niet meer wachten totdat de printer aangesloten en aangeschakeld is.

CONTROL + S
Hiermee wordt de output naar het beeldscherm onderbroken. Wanneer u na verloop van tijd een schijf hebt, waarop een groot aantal bestanden en programma's staan, en u maakt van die schijf een inhoudsopgave (DIR), dan kunt u dat overzicht onderbreken, door het intikken van CONTROL + S. U zult dan zien dat de inhoudsopgave niet verder wordt afgemaakt. Wilt u dat de inhoudsopgave verder wordt afgedrukt, dan is het voldoende om ongelijk welke toets in te drukken.

CONTROL + C
Hiermee wordt het in uitvoering zijnde commando afgebroken. Zou u deze toetsen indrukken terwijl het systeem bezig is om een inhoudsopgave van een schijf te maken, dan zal die inhoudsopgave niet af worden gemaakt. Nu is er echter geen mogelijkheid om die inhoudsopgave alsnog af te maken. Het commando is werkelijk afgebroken. U zult dan ook als laatste regel op het scherm de MSXDOS-prompt zien.

8.3 Schijven activeren.

Na het aanschakelen van de schijveneenheid en de computer, met de MSXDOS-schijf in de schijveneenheid, verschijnt uiteindelijk de MSXDOS-prompt A>. We zagen dit al in het vorige hoofdstuk. In deze prompt staat de letter A voor de op dat moment actief zijnde schijveneenheid. Het groter dan teken (>) geeft aan, dat we een commando mogen ingeven.

Zouden we nu het commando DIR ingeven, dan zal een inhoudsopgave van de schijf, die in de op dit moment actief zijnde schijveneenheid zit, worden gemaakt. In dit geval dus van de schijf in eenheid A.

De meesten onder ons zullen slechts 1 schijveneenheid hebben. In dat geval zou het erg moeilijk kunnen zijn een copie van de ene schijf naar een andere te maken. MSXDOS is ons hierbij echter behulpzaam. MSXDOS gaat er van uit dat er logisch gezien altijd twee schijveneenheden zijn. Op het moment dat wij aangeven dat we van schijveneenheid B gebruik willen maken, zal MSXDOS ons vragen om een schijf in eenheid B te laden en aan te geven dat we daarmee klaar zijn door een toets in te drukken. Vanaf dat moment is de actieve schijveneenheid logisch gezien eenheid B geworden. We zien dit ook op het beeldscherm, want de MSXDOSprompt ziet er nu als volgt uit:

```
B>
```

We zullen voorgaand verhaal met behulp van het DIR-commando, dat we al eerder hebben gezien, in de praktijk gaan bekijken. Als u de MSXDOS-prompt voor schijf A op het scherm hebt staan, dan tikt u **b:** in. Hierna ziet u dat schijveneenheid B nu actief is geworden. Tikken we nu het commando DIR in, dan zal MSXDOS een inhoudsopgave van de schijf in eenheid B willen maken. In die eenheid zat echter nog geen schijf, vandaar dat wij worden gevraagd een schijf in eenheid B te laden. U ziet dit in het volgende voorbeeld.

```
A>b:

B>dir

Insert diskette for drive B:
and strike a key when ready
MSXDOS   SYS     2432  1-01-84
COMMAND  COM     6272  1-01-80
       2 files    170496 bytes free
B>
```

Willen we nu eenheid A weer actief maken, dan tikken we **a:** in. Hierna zien we de MSXDOS-prompt voor schijf A weer verschijnen. Tikken we het commando DIR nogmaals in, dan zullen we weer worden gevraagd een schijf in de eenheid te laden, nu in eenheid A. We hadden immers de schijf uit eenheid A gehaald en in eenheid B gestopt, waardoor eenheid A leeg was.

Het voorgaande vergt nogal wat verbeeldingskracht, omdat er in werkelijkheid slechts 1 schijveneenheid is, terwijl MSXDOS net doet alsof er twee verschillende eenheden zijn. Door echter de aanwijzingen van de computer gewoon te volgen, zullen we niet gauw vergissingen maken.

Wie werkelijk twee physieke schijveneenheden heeft, zal merken dat MSXDOS iets anders reageert dan hierboven beschreven. Zodra MSXDOS weet dat er twee schijveneenheden zijn, en zodra MSXDOS weet dat er in iedere eenheid een schijf zit, zal ons niet meer worden gevraagd om schijven in de eenheden te laden. We behoeven dus geen schijven meer te wisselen. We kunnen dan echter nog steeds de andere schijveneenheid actief maken, door dezelfde handelingen te verrichten als in het vorige voorbeeld. We zullen alleen niet steeds worden gevraagd om een schijf in de eenheid te leggen.

Bij de uitleg van de verschillende commando's zullen we nog op dit onderwerp terugkomen. Bij een aantal commando's, zoals FORMAT, DIR en COPY is het erg belangrijk om te weten welke schijf actief is, omdat we dan vergissingen kunnen voorkomen.

9 Overeenkomsten tussen disk-BASIC en MSXDOS

In dit hoofdstuk zullen we een aantal MSXDOS-commando's behandelen, die een overeenkomstig MSX- of Disk-BASIC commando hebben. Het volgende tabelletje geeft een overzicht van die commando's.

MSXDOS	MSX-BASIC
COPY	COPY
DIR	FILES
ERASE	KILL
FORMAT	CALL FORMAT
MODE	WIDTH
REM	REM
RENAME	NAME

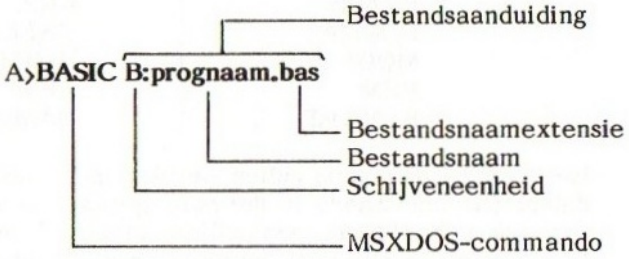
Het COPY-commando zullen we niet in dit hoofdstuk behandelen. Dit commando is dermate gecompliceerd, dat we er een apart hoofdstuk aan zullen wijden. Hoewel de beide COPY-commando's veel overeenkomsten hebben, is het aantal mogelijkheden van het MSXDOS COPY-commando is uitgebreider dan het MSXBASIC COPY-commando.

Om een goede vergelijking tussen de MSXDOS en de MSXBASIC commando's te kunnen maken, zullen we in staat moeten zijn, om over te schakelen van MSXDOS naar MSXBASIC. Hiertoe gebruiken we het MSXDOS-commando "BASIC". Dat overschakelen zullen we, naast de in voorgaande tabel gegeven commando's ook in dit hoofdstuk behandelen.

9.1 Van MSXDOS naar BASIC vice versa.

Er is een speciaal commando in MSXDOS, waarmee we kunnen overschakelen naar BASIC. Dit is het commando **BASIC.** Zoals gezegd, is dit een MSXDOS-commando, dus kunnen we dit commando alleen maar geven wanneer we een MSXDOS-prompt op het beeldscherm zien. Het ingeven van dit commando heeft tot gevolg, dat de MSXDOS-programmadelen die in het geheugen van onze computer staan, zullen worden overschreven met het DISK-BASIC. Hierna is MSXDOS dus niet meer in het geheugen aanwezig, en kunnen we geen MSXDOS-commando's meer geven.

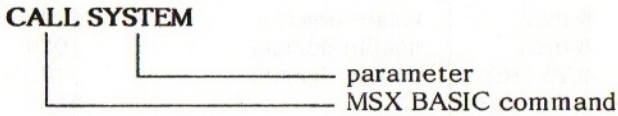
Het MSXDOS-commando BASIC kan ook nog worden gevolgd door een parameter. Deze parameter bestaat dan uit de bestandsnaam, waaronder het BASIC-programma dat u wilt starten op schijf staat, eventueel voorafgegaan door de aanduiding van de schijveneenheid, waar de schijf met het gewenste programma in zit. Het volgende voorbeeld maakt dit duidelijk:



Geeft u, zoals in voorgaand voorbeeld, aan dat u een programma van schijveneenheid B wilt uitvoeren, terwijl op dat moment eenheid A actief is, dan zult u, indien u maar een physieke schijveneenheid hebt, worden gevraagd een schijf in eenheid B te laden en daarna op een toets te drukken.

Na het overschakelen door middel van het BASIC-commando, is het MSX BASIC actief. We kunnen nu BASIC-commando's geven en BASIC-programma's draaien. Willen we, na verloop van tijd weer terugkeren naar MSXDOS, dan zullen we de

computer daartoe opdracht moeten geven door middel van een BASIC-statement. Het BASIC-statement dat ons hiertoe ter beschikking staat is het CALL statement. Als parameter bij dat CALL-statement dienen we dan het woord SYSTEM in te tikken.



Door dit BASIC-commando in te tikken, wordt de BASIC-interpreter vervangen door MSXDOS. MSXDOS wordt daartoe van de schijf met systeembestanden in het geheugen gelezen. Indien die schijf niet in de schijveneenheid zit, op het moment dat dit commando wordt gegeven, dan zal het systeem de boodschap "Boot error. Press any key to retry" geven. Hierop kunt u alsnog de schijf met de systeembestanden in de eenheid laden en op een toets drukken, waarop MSXDOS alsnog zal worden geladen.

9.2 Formateren van een schijf.

Wanneer u in de winkel een schijfje koopt, dan is dat schijfje nog helemaal leeg. Dit wil niet alleen zeggen, dat er geen bestanden op staan, er staan zelfs geen sporen en sectoren op. Het is niet mogelijk gegevens op een schijf te schrijven, zolang deze niet in sectoren is onderverdeeld. Dit proces van het indelen van een schijf in sporen (tracks) en sectoren, en het voorbereiden van de inhoudsopgave van de schijf, wordt met formateren aangeduid. Soms zult u de term initialiseren tegenkomen. Hiermee wordt precies hetzelfde bedoeld.
Het aantal sporen, dat op een schijf wordt geschreven, en het aantal en de grootte van de sectoren, die op iedere track worden geschreven, is afhankelijk van de gebruikte schijveneenheid. Soms worden beide zijden van de schijf gebruikt om er gegevens naar toe te schrijven, soms wordt slechts een zijde gebruikt. Volgens de MSX-standaard zouden schijven, die aan de volgende specificaties voldoen, mogen worden

gebruikt, waarbij moet worden aangetekend, dat het 8 inch formaat voor de MSX-computers in de praktijk niet wordt gebruikt:

diameter	schrijfdichtheid	bytes/sector
8 inch	single density	128
8 inch	double density	1024
5,25 inch	double density	512
3,5 inch	CFD	512
3 inch	CFD	512

Schijven, die u al eerder hebt gebruikt en waar al bestanden op staan, kunnen ook opnieuw worden geformateerd. MSXDOS doet dan net alsof het een nieuwe schijf is, en schrijft er nieuwe sporen en sectoren op. Hierna worden de sectoren van spoor (track) 0 van de schijf, net als bij een nieuwe schijf, voorzien van een lege inhoudsopgave. Het gevolg hiervan is natuurlijk wel, dat alle informatie die op de schijf stond voordat deze werd geformateerd, na het formateren is verdwenen.

Het is heel wel denkbaar, dat u per ongeluk een verkeerde schijf in de schijveneenheid zet bij het formateren. Zet u er per ongeluk een schijf in waarop bestanden staan, die u niet kwijt wilt, dan zullen deze programma's na het formateren toch zijn verdwenen. Weest u dus voorzichtig met het formateren van schijven. MSXDOS is u in zoverre behulpzaam, dat het u eerstvraagt op welke schijveneenheid u wilt formateren. Nadat u hierop hebt geantwoord vraagt het u nog om een toets in te drukken, voordat met het formateren wordt begonnen. Het volgende voorbeeld laat zien wat op het scherm wordt afgedrukt wanneer u een FORMAT commando geeft.

```
A>format
Drive name (A,B) a
Strike a key when ready
format complete
```

De vetgedrukte woorden zijn door u ingetikt, de overige tekst

wordt door het systeem afgedrukt. Tussen het moment, waarop u een willekeurige toets hebt ingedrukt (in antwoord op de boodschap "Strike a key when ready") en het moment waarop de boodschap "format complete" wordt afgedrukt, verstrijkt enige tijd. Die tijd wordt gebruikt voor het formateren van de schijf. Die tijd is afhankelijk van het type schijf dat moet worden geformateerd en kan varieren van een tot enkele minuten.

Hopelijk is het u al gelukt, om een copie te maken van de schijf met de systeemprogramma's. Mocht u nog niet zo'n copie hebben, dan zullen we die nu eerst gaan maken, voordat we verder gaan met het proberen van de andere MSXDOS-commando's. De copie moet worden gemaakt met behulp van het COPY-commando. Hoewel we dit commando nu niet gaan behandelen, zullen we hem toch even gaan gebruiken.

Neemt u een lege schijf en plaats deze in de schijveneenheid. Geef nu het commando FORMAT en wacht tot u de boodschap "format complete" hebt gekregen. Leg nu de systeemschijf (met de bestanden MSXDOS.SYS en COMMAND.COM) in de schijveneenheid en geef het volgende commando:

```
COPY a:*.* b:
```

Mocht het systeem nu vragen om de schijf voor eenheid A: te plaatsen, dan dient u er voor te zorgen, dat de systeemschijf in de schijveneenheid zit, waarna u op een toets drukt, om aan te geven dat het copieren mag beginnen. Nu worden alle bestanden van schijf A: in het geheugen van de MSX-computer geladen, net zo lang tot het geheugen van de computer vol is, of totdat alle bestanden in het geheugen staan.

Nu zult u worden gevraagd om de schijf in eenheid B: te plaatsen. Hierop haalt u de systeemschijf uit de schijveneenheid en zet u de zojuist geformateerde schijf in eenheid B:. Weer drukt u op een toets, om aan te geven dat het copieren mag worden voortgezet. Nu zullen alle bestanden vanuit het geheugen naar de nieuwe schijf worden geschreven.

Waren nog niet alle bestanden van schijf A: gelezen, dan zal het systeem u weer vragen om de schijf in eenheid A: te plaatsen. U zet dan de systeemschijf weer in de eenheid. Wordt u gevraagd om de schijf in eenheid B: weer te plaatsen, dan zet u de schijf waar de copie naartoe moet worden geschreven, weer in de eenheid. Wanneer u nog geen andere bestanden op de orginele systeemschijf had gezet, zal de copie in een keer kunnen worden gemaakt.

Nu u een goede copie hebt, doet u er goed aan de orginele schijf op te bergen op een veilige plaats. Wanneer de zojuist gemaakte copie onverhoopt defect zou raken, kunt u weer een nieuwe copie maken van de orginele schijf. De zojuist gevolgde procedure wordt in computer-jargon meestal aangeduid met het "maken van een **back-up**". Hebt u eenmaal een "back-up", dan kunt u met een gerust hart de oefeningen en opdrachten uit de rest van dit boek uitvoeren.

9.2 Manipuleren met bestanden.

In deze paragraaf zullen we zien hoe we bestanden van een schijf kunnen verwijderen, hoe we de naam van een bestand kunnen veranderen en hoe we kunnen controleren of onze commando's werkelijk zijn uitgevoerd. Daartoe gebruiken we respectievelijk het ERASE-, RENAME- en DIR-commando. Al deze commando's hebben een overeenkomstig commando in BASIC, namelijk KILL, NAME en FILES.

Voordat we deze commando's kunnen uitvoeren, zullen we eerst een aantal bestanden moeten maken. Dit zullen we doen door naar BASIC over te schakelen en een aantal korte BASIC-programma's te schrijven en op schijf op te slaan. Weet u nog hoe het overschakelen in zijn werk gaat? Juist, met het commando BASIC.

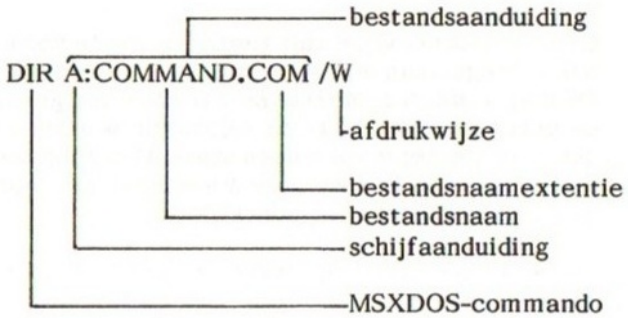
Eenmaal in BASIC, kunt u een programma maken (bijvoorbeeld 1 regel met daarin een REM-statement). Dit programma schrijft u naar schijf met behulp van het commando "SAVE programmanaam". Schrijft u op deze manier een aan-

tal programma's onder verschillende namen naar de schijf. Gebruik daarbij de namen TEST1, TEST2, TEST3, etc.

Bent u daarmee klaar, dan kunt u terugkeren naar MSXDOS door het BASIC-commando CALL SYSTEM te geven. Na korte tijd zult u de MSXDOS-prompt weer op het scherm zien verschijnen. Door nu het commando DIR te geven, krijgt u een overzicht van alle op de schijf staande files. Ook de zojuist gecreeerde files staan daarin, zoals hierna is te zien.

```
A>dir
MSXDOS    SYS      2432   1-01-84
COMMAND   COM      6272   1-01-80
TEST1                27   4-20-86
TEST2                27   4-20-86
TEST3                26   4-20-86
        5 files    168960 bytes free
```

Het DIR-commando kan worden gevolgd door een aantal parameters. De eerste parameter is de bestandsaanduiding. Deze bestaat uit een indicatie van de schijf en een indicatie van de bestandsnaam. De bestandsnaam hoeft niet beslist de volledige naam zijn. Het is toegestaan om slechts een gedeelte van de naam te geven, gevolgd door een asterix (*). De tweede parameter geeft aan op welke manier de inhoudsopgave moet worden afgedrukt. Straks zullen we de mogelijkheden zien. We zullen eerst het formaat van het DIR-commando bekijken.



De bestandsnaamaanduiding mag worden weggelaten. In dat geval wordt van de schijf die op dat moment in de eenheid zit een inhoudsopgave gemaakt. In die inhoudsopgave zullen dan alle bestandsnamen worden gegeven.

De afdrukwijze mag ook worden weggelaten. Tot nu toe hebben we het DIR-commando steeds zonder afdrukwijze gegeven. We zagen daarbij, dat van ieder bestand behalve de naam ook nog de grootte en de datum, waarop het bestand werd gecreeerd, werden gegeven. Door nu als afdrukwijze **/W** te kiezen, wordt het volgende resultaat verkregen.

```
A>dir /w
MSXDOS   SYS     COMMAND  COM
TEST1            TEST2
TEST3
       5 files    168960 bytes free
A>
```

U ziet het, alleen de bestandsnamen worden nog afgedrukt. Belangrijk daarbij is, dat er op iedere regel twee bestandsnamen worden afgedrukt. hierdoor kunnen er twee keer zoveel bestandsnamen op een scherm.

Een tweede afdrukwijze is **/P.** Hiermee wordt aangegeven, dat, indien er meer bestandsnamen zijn, dan er op een scherm passen, het maken van de inhoudsopgave moet worden gestopt, zodra er een scherm vol namen is en dat het volgende scherm met namen pas mag worden gemaakt, nadat er een toets is ingedrukt.

Over de afdruktwijze valt tenslotte nog te vermelden, dat het ook is toegestaan om een combinatie van de parameters **/P** en **/W** mag worden gemaakt. Zo zal **/W/P** tot gevolg hebben, dat de bestandsnamen in twee kolommen worden afgedrukt, totdat er een scherm vol namen staat. Het volgende scherm, met bestandsnamen in twee kolommen, zal pas worden gemaakt, nadat er een toets is ingedrukt.

Zoals al even kort opgemerkt, mogen de bestandsnaam en de

bestandsnaamextentie geheel of gedeeltelijk worden vervangen door een asterix (*). Deze asterix betekent dan dat het er niet toe doet welke letter of letters er nog volgen. Een voorbeeldje zal dit waarschijnlijk het snelst duidelijk maken.

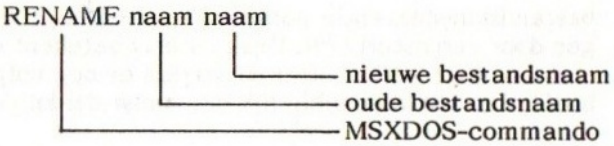
```
A>dir t*.*
TEST1            27   4-20-86
TEST2            27   4-20-86
TEST3            26   4-20-86
       3 files   168960 bytes free
A>
```

De asteriks neemt dus de plaats in van een of meerdere tekens. In plaats van een asterix mag ook een vraagteken worden gebruikt. Het vraagteken staat echter slechts voor 1 teken. Wel is het toegestaan om meerdere vraagtekens in de bestandsnaam te zetten. Tenslotte is het ook nog toegestaan om zowel vraagtekens als asteriksjes in de naam te gebruiken. Ook hiervan een voorbeeldje.

```
A>dir ??S*.*
TEST1            27   4-20-86
TEST2            27   4-20-86
TEST3            26   4-20-86
       3 files   168960 bytes free
A>dir ????2*.*
TEST2            27   4-20-86
       1 file    168960 bytes free
A>
```

Het gebruik van asteriks en vraagteken in bestandsnamen, ter vervanging van tekens, is niet alleen toegestaan in het DIR-commando, doch mag altijd worden gebruikt in alle commando's waarin bestandsnamen voorkomen.

Het RENAME-commando is redelijk eenvoudig. Het dient te worden gevolgd door de naam van het bestand, waaraan we een nieuwe naam willen toekennen en door de nieuwe bestandsnaam. Het formaat van dit commando ziet er als volgt uit:

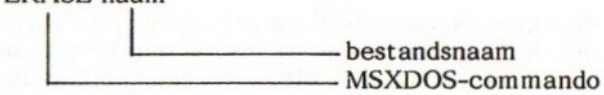


Ook bij dit commando weer een voorbeeldje. In dit voorbeeld wordt tevens aangetoond, dat er meerdere bestanden tegelijkertijd met hetzelfde commando kunnen worden geRENAMEd, door gebruik te maken van asteriksjes. In ons voorbeeld worden de bestandsnaamextenties van alle bestanden die beginnen met de letters test gewijzigd in .bas:

```
A>rename test*.* test*.bas

A>dir t*.*
TEST1    BAS        27   4-20-86
TEST2    BAS        27   4-20-86
TEST3    BAS        26   4-20-86
       3 files    168960 bytes free
```

Het commando RENAME mag ook worden afgekort tot REN. In dit boek zullen we dat niet doen, omdat dit de leesbaarheid niet verhoogt.

Het MSXDOS-commando ERASE moet worden gevolgd door slechts 1 parameter, de naam van het te wissen bestand. Het formaat van dit commando is dienovereenkomstig eenvoudig:



In plaats van het woord ERASE mag ook het woord DEL worden gebruikt. DEL staat voor het Engelse woord "delete", hetgeen verwijder betekent.

Ook in deze bestandsnaam mogen weer vraagtekens en asteriksjes worden opgenomen, zoals in het volgende voor-

beeld is te zien:

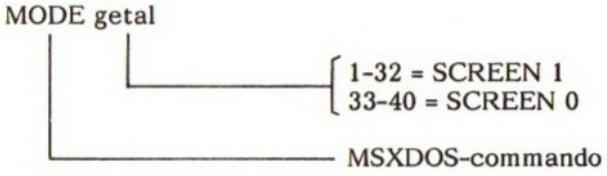
```
A>erase ????3*.BAS

A>
```

Het zal duidelijk zijn, dat het met dit commando mogelijk is om alle bestanden, of een groep van bestanden, in een keer van de schijf te wissen. Weest u daar voorzichtig mee. Geef het commando ERASE *.* alleen wanneer u werkelijk alle bestanden van de schijf wilt wissen.

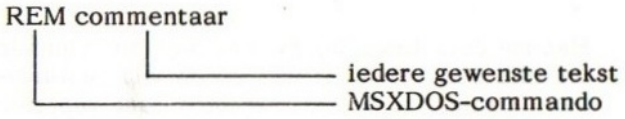
9.4 Het MODE- en REM-commando

De laatste twee commando's, die een overeenkomstig commando in disk-BASIC hebben, zijn MODE en REM. Beide commando's zijn in principe erg eenvoudig. De werkelijke functie van het REM-commando zal echter pas duidelijk worden in het hoofdstuk over het COPY-commando, waarin we zelf bestanden met MSXDOS-commando's zullen maken, die door middel van een zelfbedachte naam kunnen worden aangeroepen en door MSXDOS worden uitgevoerd. Dit soort bestanden wordt aangeduid met "Batch files".

Het commando MODE dient te worden gevolgd door een parameter. Deze parameter dient een getal te zijn, dat mag varieren van 1 tot en met 40. Indien het getal 32 of kleiner is, zal MSXDOS er voor zorgen, dat er wordt overgeschakeld naar SCREEN-mode 1. Is het getal groter dan 32, dan wordt er overgeschakeld naar SCREEN-mode 0. Het formaat is als volgt:



Het REM-commando heeft een al even eenvoudig formaat. Het commando mag worden gevolgd door commentaar. Het grote verschil met het BASIC-statement is, dat het commentaar van het REM-commando wordt afgedrukt. Bij de behandeling van de Batch files zullen we daarop terugkomen.

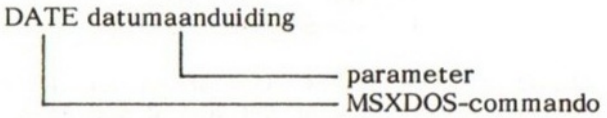


10 Unieke MSXDOS-commando's

Nadat we in het vorige hoofdstuk een aantal MSXDOS-commando's hebben gezien, die een overeenkomstig commando in disk-BASIC hadden, zullen we in dit hoofdstuk de MSXDOS-commando's gaan bekijken, waarvoor geen disk-BASIC variant bestaat. Twee van die commando's, DATE en TIME, zijn al niet helemaal nieuw meer voor ons. We zagen reeds hoe we de datum en de tijd moesten ingeven in hoofdstuk 7. In dit hoofdstuk zullen we echter zien hoe we de commando's kunnen ingeven, waarna we een nieuwe datum en/of tijd kunnen ingeven. De andere in dit hoofdstuk te behandelen commando's zijn VERIFY, PAUZE en TYPE.

10.1 Datum en tijd wijzigen.

Indien het MSXDOS-commando DATE wordt gegeven, zonder enige parameters, zult u door MSXDOS worden geinformeerd over de huidige datum, waarna u wordt gevraagd een nieuwe datum in te geven. Het formaat van dit commando is als volgt:



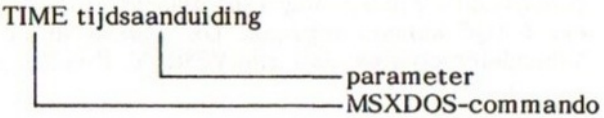
De parameter **datumaanduiding** moet bestaan uit de maand (MM), de dag (DD) en het jaar (JJ of JJJJ). De volgende formaten zijn toegestaan:

MM.DD.JJ	MM.DD.JJJJ
MM/DD/JJ	MM/DD/JJJJ
MM-DD-JJ	MM-DD-JJJJ

Het volgende voorbeeld laat zien, hoe het systeem reageert bij het intikken van de verschillende DATE-commando's.

```
A>date
Current date is Sun  4-20-1986
Enter new date: 4-20-86
A>date 4-20-86

A>date
Current date is Sun  4-20-1986
Enter new date: 4/20.1986
```

Voor het commando TIME geldt in grote lijnen hetzelfde als voor DATE. Het formaat van het TIME-commando is:



Ook hier geldt weer, dat het weglaten van de parameter achter het commando tot gevolg heeft dat u de momentele tijd krijgt te zien, waarna u zult worden gevraagd een nieuwe tijd in te tikken. Het formaat van de parameter **tijdsaanduiding** is als volgt:

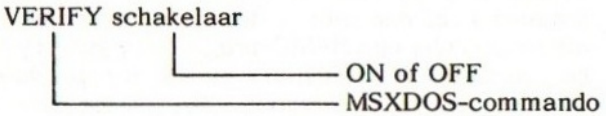
UU	UUP
UU:MM	UU:MMP
UU:MM:SS	UU:MM:SSP

Hierin staan de letters UU voor het uur, de letters MM voor de minuten en SS voor het aantal seconden. In de rechter kolom ziet u dat de tijd wordt gevolgd door de letter P. Hiermee wordt aangegeven dat de ingegeven tijd na het middaguur ligt. Indien de toevoeging P wordt gebruikt, mag het aantal uren niet hoger zijn dan 12. 12P betekent 12 uur 's nachts.

10.2 Is een bestand goed weggeschreven?

Schrijven naar schijf gaat meestal goed. Tot op de dag van vandaag is het mij bijvoorbeeld nog niet overkomen, dat er een fout werd gemaakt, bij het wegschrijven van een bestand naar schijf. Ik gebruik mijn schijveneenheid nu al ongeveer een jaar, bijna dagelijks. Toch is er altijd de kans, dat er toch iets fout gaat. Zo'n fout ontdek je altijd pas wanneer het te laat is, namelijk bij de eerstvolgende keer dat je het bestand wilt inlezen. Vandaar, dat het verstandig is, om bestanden, waarvan het belangrijk is dat ze absoluut foutvrij op schijf worden geschreven, weg te schrijven nadat het commando VERIFY ON is gegeven.

Met **VERIFY ON** wordt ieder bestand dat naar schijf wordt geschreven direct na het schrijven teruggelezen en vergeleken met het oorspronkelijke bestand. U begrijpt, dat dit tot gevolg zal hebben, dat het wat langer duurt voordat een bestand is weggeschreven. Wie echter met cassettes heeft gewerkt zal het met mij eens zijn dat het zelfs met VERIFY ON nog flitsend snel gaat. Het formaat van VERIFY is:



Is VERIFY eenmaal ON gezet, dan blijft dit gelden, totdat het commando **VERIFY OFF** wordt gegeven.

10.3 Afdrukken van bestanden.

Zoals u inmiddels weet, bestaan er diverse soorten bestanden. Gegevensbestanden bestaan uit records, die weer uit afdrukbare ASCII-tekens bestaan. Hetzelfde geldt voor BASIC-programma bestanden, wanneer dat programma als ASCII-file werd weggeschreven, met behulp van het commando SAVE "naam",A. Wordt een BASIC-programma weggeschreven zonder de toevoeging A, dan zal het ontstane bestand niet alleen afdrukbare tekens bevatten, maar ook niet afdrukbare

tekens. Immers voor ieder BASIC-statement wordt een speciale code in de plaats gezet. Behalve BASIC-programma's kunnen we ook nog machinetaalprogramma's en videogeheugeninhouden als bestand op schijf worden gezet. Hierin komen alle codes van 0 tot en met 255 voor, en dus een groot aantal niet afdrukbare tekens.

Met het MSXDOS-commando TYPE kan ieder gewenst bestand worden afgedrukt op het beeldscherm. Wordt echter een bestand aangegeven waarin niet afdrukbare codes voorkomen, dan kunnen de vreemdste gevolgen optreden. Een klein voorbeeldje hiervan kunt u zien, wanneer u een van de kleine BASIC-programma's, die we in het vorige hoofdstuk op schijf hebben gezet, gaat afdrukken. Geeft u hiertoe het volgende commando:

TYPE programmanaam

U ziet dat er, afhankelijk van het programma dat er in het bestand staat, een aantal vreemde tekens worden afgedrukt. Ook worden er naar alle waarschijnlijkheid een aantal regels zomaar overgeslagen en indien u normale tekst in het programma had, dan ziet u dat deze tekst ook normaal wordt afgedrukt. Om een BASIC-programma goed te kunnen afdrukken, dient het als ASCII-file te zijn weggeschreven. Laten we daarom voordat we verder gaan eerst een ASCII-bestand creeren.

Hiertoe schakelen we weer over naar BASIC, met behulp van het MSXDOS-commando BASIC. Eenmaal in BASIC typen we een eenvoudig BASIC-programma in, bijvoorbeeld een of enkele REM-regels. Dit programma zetten we met het commando SAVE "TEST2.ASC",A op schijf. Merkt u op, dat de bestandsnaamextentie nu ASC is. Hierdoor onderscheidt dit bestand zich van de andere bestanden. U zult zich nog herinneren dat we de andere BASIC-programma's de extentie BAS hadden gegeven.

Nu het programma als ASCII-file op schijf staat, kunnen we terugkeren naar MSXDOS, door het BASIC-commando CALL

SYSTEM te geven. Terug in MSXDOS geven we eerst het DIR-commando. We zien dan dat het zojuist gecreeerde bestand inderdaad op de schijf staat. Daarna geven we, zoals ook in het volgende voorbeeld is te zien, het commando:

```
TYPE TEST2.ASC
```

Dit commando heeft tot gevolg dat het BASIC-programma (TEST2.ASC) netjes op het scherm wordt afgedrukt, inclusief de regelnummers.

```
A>type test2.asc
10 REM testprogramma ASCII-file
20 FOR I=1 TO 100
30 NEXT I
40 CALL SYSTEM

A>
```

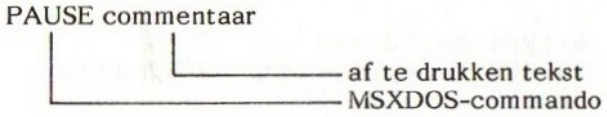
Net als bij de meeste andere commando's ook al het geval was, mogen we ook in het TYPE-commando asteriksen en vraagtekens gebruiken. Later zullen we zien dat we ook bestanden kunnen maken zonder MSXDOS te verlaten. Daarvoor is echter het COPY-commando nodig, dat in het volgende hoofdstuk zal worden behandeld.

10.4 Onderbreking van commando uitvoering.

Er zijn twee situaties te onderscheiden, waarin het wenselijk kan zijn de uitvoering van commando's te onderbreken.

De eerste situatie is, wanneer een commando in uitvoering is. Denk hierbij aan het commando TYPE. Indien een lange file (bestand) met behulp van TYPE op het beeldscherm wordt afgedrukt, zal de tekst zo snel langs het beeldscherm glijden, dat het onmogelijk is om alles te lezen. Om het afdrukken tijdelijk te onderbreken, kan de **control-toets samen met de letter S** worden ingedrukt. De eerstvolgende toetsaanslag zal de uitvoering van het commando weer voortzetten.

De tweede situatie doet zich voor, wanneer MSXDOS commando's uit een commandobestand aan het uitvoeren is. In de reeks commando's in dat bestand kan een **PAUSE-commando** zijn opgenomen. Dit PAUSE-commando zorgt er voor dat, nadat het vorige commando is uitgevoerd, gewacht wordt met het uitvoeren van het volgende commando, totdat de gebruiker op een toets heeft gedrukt. In het laatste hoofdstuk van dit boek zal daarvan een voorbeeld worden gegeven. Het formaat van dit commando is als volgt:



De tekst achter het commando zal op het beeldscherm worden afgedrukt, gevolgd door de boodschap "Strike a key when ready", zoals in het volgende voorbeeld is te zien.

Dit commando krijgt pas werkelijk nut, wanneer het in commandobestanden (Batch-files) wordt toegepast.

In beide voorgaande situaties is het mogelijk om de commando-uitvoering definitief te onderbreken, met andere woorden af te breken. Afbreken van de uitvoering van een commando wordt verkregen door de **control-toets samen met de letter C** in te drukken. Na het indrukken van deze toetsencombinatie verschijnt de MSXDOS-prompt weer op het scherm en kan een nieuw commando worden ingegeven.

11 Copiëren met MSXDOS

MSXDOS kent een vijftal soorten randapparatuur. Daaronder bevinden zich invoer-, uitvoer- en I/O-apparaten. Met behulp van het COPY-commando kunnen bestanden tussen de verschillende apparaten worden gecopieerd. De soorten apparaten zijn:

A: of B:	In/uit	Extern geheugen (diskette)
CON	In	Console (toetsenbord)
PRN	Uit	Printer (via parallelpoort)
AUX	In/uit	Auxiliary (RS232C-poort)
NUL	In/uit	Null (dummy ofwel niets)

De externe geheugens (de schijven A: en B:) zijn al eerder aan de orde geweest. Dat hierop bestanden staan, die van de ene schijf naar de andere kunnen worden gecopieerd, zal u duidelijk zijn. Daar er op een schijf meerdere bestanden tegelijk kunnen staan, zijn we genoodzaakt de bestandsnaam aan te geven, wanneer we een bestand van of naar een schijf willen copieren. Dit geldt echter alleen voor schijven.

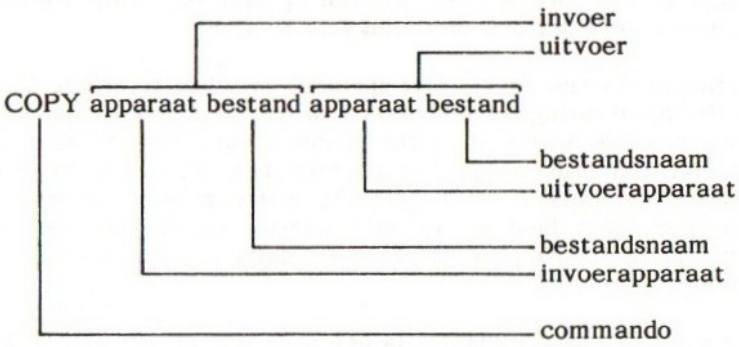
Bestanden hoeven echter niet speciaal op schijf te staan. Uit uw BASIC-ervaring tot nu toe is u al duidelijk geworden, dat bestanden ook op een cassette mogen staan. Voor MSXDOS, dat een afgeleide is van het professionele MS-DOS, dat op personal computers wordt gebruikt, bestaan geen cassetterecorders. Wel bestaan er een aantal andere apparaten, namelijk toetsenborden, printers en datacommunicatiekanalen.

Een heel bijzonder apparaat is het apparaat dat in MSXDOS wordt aangeduid met **NUL**. Dit is namelijk een niet bestaand apparaat. Indien we een copie naar dit apparaat maken, dan zal die copie als het ware "Ins Blaue hinein" gaan. Met andere woorden, de copie wordt in het niets, het luchtledige, ge-

maakt. Hetzelfde geldt, wanneer we proberen een copie te maken vanaf het NUL-apparaat naar een uitvoerapparaat, zoals een schijf of een printer. Er wordt niets, maar dan ook helemaal niets, gecopieerd. Het nut van dit apparaat kan zijn, dat een commando er mee wordt gecontroleerd op de juiste syntax.

Zoals gezegd, hoeven bestanden niet speciaal op een schijf te staan. Ze mogen ook op andere apparaten staan of gezet worden. Zelfs op een NUL-apparaat. Veel vaker zal het voorkomen, dat u een bestand wilt copieren van schijf naar printer of van het toetsenbord naar schijf of naar printer. Daar MSX-computers standaard nog niet zijn uitgerust met een RS232C-poort, zullen we die mogelijkheid nog niet gebruiken. Alle soorten apparaten, die MSXDOS kan onderscheiden, behalve de schijf, kunnen slechts 1 bestand tegelijkertijd bevatten. Dit geldt voor het toetsenbord, de printer en de RS232C-poort. In al die gevallen is het noemen van de bestandsnaam overbodig. Dit heeft ertoe geleid, dat wij ons niet realiseren, dat ook op een toetsenbord of een printer bestanden staan.

Het algemene formaat van het COPY-commando is als volgt:



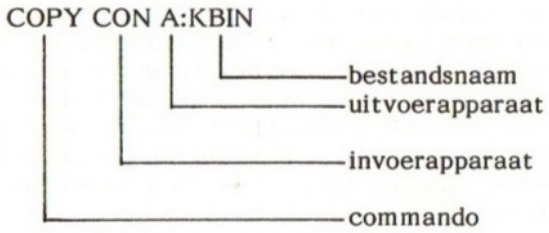
Indien de indicatie van het soort apparaat wordt weggelaten, dan zal MSXDOS aannemen, dat het om een schijf gaat, en

wel om de schijf die op dat moment actief is. De bestandsnaam mag in dit geval bestaan uit een naam en een bestandsnaam-uitbreiding, van elkaar gescheiden door een punt. Alle regels, die we tot nu toe hebben geleerd, zijn op de bestandsnaam van toepassing. Er mag dus gebruik worden gemaakt van asteriksjes en vraagtekens in de bestandsnaam. Indien het soort apparaat geen schijf is, mag de bestandsnaam worden weggelaten.

In de volgende paragrafen zullen we het copieren van en naar de verschillende apparaten aan de hand van vele voorbeelden behandelen. In deze voorbeelden zullen we weer gebruik maken van de in de vorige hoofdstukken geprepareerde schijf met bestanden.

11.1 Copieren vanaf het toetsenbord.

Wanneer we een bestand via het toetsenbord intikken, dan hoeven we geen bestandsnaam aan te geven, omdat er slechts 1 toetsenbord is, waarop slechts 1 bestand tegelijk kan worden ingetikt. Er kan dus geen enkel misverstand bestaan. Willen we dat ingetikte bestand naar een schijf copieren, dan zullen we aan het bestand, zoals dat op de schijf komt te staan, wel een naam moeten geven, omdat we het anders nooit op de schijf terug kunnen vinden. Een voorbeeld van het COPY-commando, waarmee we een bestand van het toetsenbord naar de schijf copieren, is als volgt:



Het COPY-commando zorgt er nu voor dat er een bestand op

het toetsenbord wordt geopend. Dat wil zeggen, dat er zal worden gewacht op invoer vanaf het toetsenbord. MSXDOS heeft er echter geen idee van hoe lang dat bestand is (uit hoeveel records dat bestand bestaat). Iedere keer dat we op het toetsenbord de RETURN-toets indrukken, geven we aan, dat er weer een record aan het bestand moet worden toegevoegd. Hoe vertellen we MSXDOS nu dat het hele bestand is ingetikt? Daartoe dienen we, op het moment dat we een nieuw record zouden beginnen in te tikken, de **CONTROL-toets samen met de letter Z** in te drukken. Het volgende voorbeeld laat dit zien.

```
A>copy con a:kbin
eerste record
tweede record
derde record
^Z
        1 file copied
A>type kbin
eerste record
tweede record
derde record

A>
```

Na ieder record dat we intikken, moeten we dus de RETURN-toets indrukken, om aan te geven dat er een compleet record is ingevoerd. Ook de CONTROL+Z moet worden gevolgd door het indrukken van de RETURN-toets. Daarmee wordt het laatste (in dit geval het vierde) record aan het bestand toegevoegd. Dat laatste record bevat dan, behalve de End Of File indicatie, geen enkele informatie. Dit is ook te zien na uitvoering van het commando TYPE. Er worden slechts drie records afgedrukt. De **^Z** wordt niet afgedrukt, doch heeft alleen als functie, MSXDOS te laten weten dat het einde van het bestand is bereikt.

Copieren we een bestand van het toetsenbord naar de printer, dan hoeven we helemaal geen bestandsnamen te gebruiken. Wel dienen we ons aan de regels voor het intikken van een bestand te houden. We moeten door middel van het indrukken van de RETURN-toets aangeven dat er een record is ingegeven, en als laatste record dienen we de CONTROL-toets samen met de letter Z in te drukken, gevolgd door de RETURN-toets. Het volgende voorbeeld laat zien hoe een dergelijke copie in zijn werk gaat.

```
copy con prn

A>copy con prn
van toetsenbord naar printer
^Z
van toetsenbord naar printer
       1 file copied
```

Als u dit commando uitvoert, zult u merken, dat het ingetikte record pas wordt afgedrukt nadat u de RETURN-toets hebt ingedrukt. Dit is ook logisch, want pas nadat u de RETURN-toets hebt ingedrukt is er een record ingetoetst. Dat record wordt dan naar de printer gecopieerd.

Voor het copieren van bestanden vanaf het toetsenbord naar de apparaten AUX en NUL gelden dezelfde regels als voor PRN. Daarbij dient wel te worden opgemerkt, dat het resultaat van het copieren naar NUL helemaal niets is. U kunt echter leuk oefenen met het intikken van bestanden, zonder de ingevoerde records ergens naar toe te schrijven. Probeert u het maar eens, en vergeet niet, dat het commando pas wordt beeindigd, nadat u CONTROL+Z en RETURN hebt ingedrukt.

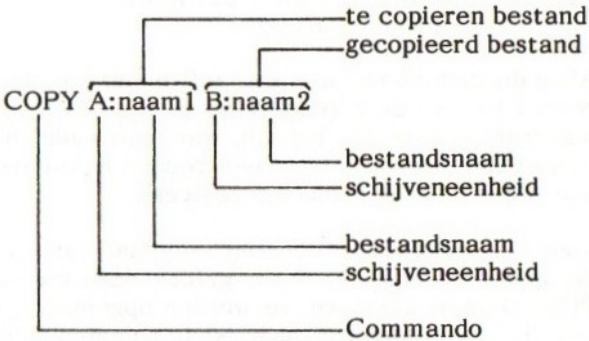
11.2 Copieren van schijf naar schijf.

Hiervan hebben we in een vorig hoofdstuk al een voorbeeldje gezien. We maakten toen een "back-up" van de systeemschijf. Het bestand, dat we willen copieren, dienen we altijd aan te

geven met behulp van een bestandsnaam. In plaats van een unieke naam, mogen we ook gebruik maken van de asteriks en van vraagtekens. Daarmee kunnen we dan alle bestanden of een groep van bestanden aangeven.

Het bestand waar we naartoe willen copieren hoeven we niet aan te geven. Laten we voor de uitvoer de bestandsnaam achterwege, dan zullen de gecopieerde bestanden dezelfde naam krijgen als de bestanden waar we de copie van maken. Wilt u echter een bestand copieren naar een bestand met een andere naam, dan zult u ook in het tweede deel van het copieer-commando een bestandsnaam moeten invullen.

Het formaat van het COPY-commando, voor het copieren van schijf naar schijf is als volgt:



Wordt de indicatie van de schijveneenheid voor het te copieren bestand weggelaten, dan zal de op dat moment actief zijnde schijveneenheid worden gekozen. Wordt de indicatie van de schijveneenheid voor het gecopieerde bestand weggelaten, dan zal dezelfde schijveneenheid als die van het te copieren bestand worden gekozen. Wordt de bestandsnaam van het gecopieerde bestand weggelaten, dan zal het gecopieerde bestand dezelfde naam krijgen als het te copieren bestand.

Hier volgen enkele voorbeelden:

1. Copie van 1 bestand onder dezelfde naam, naar een andere schijf.

```
A>copy a:test1.bas b:
Insert diskette for drive B:
and strike a key when ready
       1 file copied
A>
```

2. Copie van 1 bestand onder een andere naam naar dezelfde schijf.

```
A>copy test1.bas test1.cop
       1 file copied
A>
```

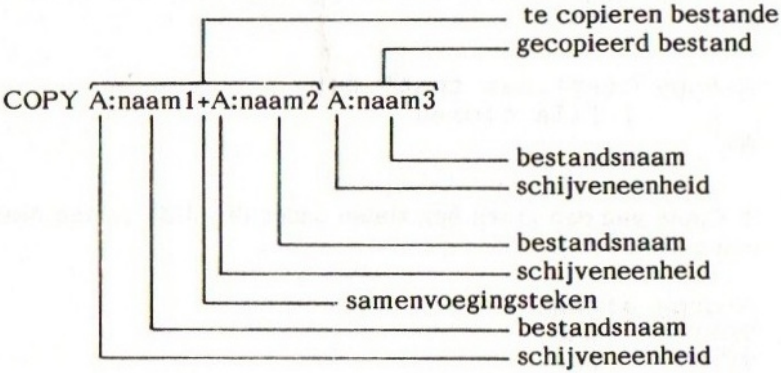
3. Copie van een groep bestanden onder dèzelfde namen naar een andere schijf.

```
A>copy a:test*.* b:
TEST1      BAS
TEST2      BAS
TEST2      ASC
TEST1      COP
       4 files copied
A>
```

4. Copie van alle bestanden van de ene schijf naar een andere schijf, met dezelfde bestandsnamen. (back-up)

```
A>copy a:*.* b:
MSXDOS     SYS
COMMAND    COM
TEST1      BAS
TEST2      BAS
TEST2      ASC
KBIN
TEST1      COP
       7 files copied
```

Met deze voorbeelden worden een aantal variaties getoond. Zelf kunt u de getoonde mogelijkheden gebruiken om nog meer variaties te maken. Daarbij kunt u ook het vraagteken gebruiken om een bepaald teken uit de bestandsnaam variabel te maken. Dit waren echter nog lang niet alle mogelijkheden. Een heel bijzondere mogelijkheid is, dat men meerdere bestanden tijdens het copieren kan samenvoegen tot 1 gecopieerd bestand. Het formaat van het COPY-commando waarmee deze mogelijkheid wordt gebruikt is als volgt:



Hierbij dienen we wel voorzichtig te zijn. Alleen tekstbestanden kunnen op deze manier worden samengevoegd in de copie. Zouden we een niet tekstbestand willen copieren, dan kunnen we daarvoor een extra parameter aan de bestandsnaam in het COPY-commando toevoegen. Er zijn twee parameters; /A en /B.

/A wil zeggen, dat het genoemde bestand een **tekstbestand** is.

/B wil zeggen, dat het genoemde bestand een **niet-tekstbestand**, zoals een machinetaalprogramma, is.

De parameters /A en /B hebben alleen betrekking op het COPY-commando waarin ze staan. Indien we deze parameters nog niet hebben toegepast, gaat MSXDOS er vanuit, dat alle genoemde bestanden tekstbestanden zijn. Zodra we de /B-

parameter hebben gebruikt, gaat MSXDOS er vanuit, dat alle daarna genoemde bestanden niet-tekstbestanden zijn, totdat de /A-parameter weer is gebruikt.

Het volgende voorbeeld laat zien, hoe een binair bestand (niet-tekstbestand) naar een tekstbestand kan worden gecopieerd:

```
COPY A:MCODE/B A:MCODE.ASC/A
```

Met de in dit hoofdstuk gegeven mogelijkheden van het COPY-commando kunt u naar hartelust experimenteren. Doe dat echter in het begin alleen met die bestanden, die geen belangrijke informatie bevatten. U zult waarschijnlijk in het begin nog wel eens een vergissing begaan, waarbij u een ander bestand per ongeluk overschrijft.

12 Zelf nieuwe commando's maken

In dit laatste hoofdstuk zullen we zien hoe MSXDOS ons in staat stelt, zelf nieuwe commando's samen te stellen, uit de bestaande commando's. Hiertoe kunnen we een aantal MSXDOS-commando's in een bestand zetten. Dit bestand geven we dan een bestandsnaam (naar eigen keuze) met als bestandsnaamuitbreiding de letters **BAT.** De letters BAT komen van het Engelse woord "batch", hetgeen zoveel wil zeggen als een "groep" of een "partij". Uiteraard heeft dit betrekking op de commando's die in het bestand staan. Er staan immers een aantal (een groep of partij) commando's in.

Bestanden met de bestandsnaamuitbreiding BAT worden meestal aangeduid met de naam **batch-files.** Vanaf nu zullen wij die naam gaan gebruiken.

Een heel bijzondere vorm van batch-files is de batch-file met de naam AUTOEXEC. Bij het aanschakelen van het systeem kijkt MSXDOS namelijk op de eerste aangesloten schijf, of daarop een bestand met de naam **AUTOEXEC.BAT** voorkomt. Is dit zo, dan worden de commando's, die in dat bestand staan, uitgevoerd. Wij zullen in dit hoofdstuk een aantal voorbeelden zien.

12.1 Bestanden met MSXDOS-commando's creeren.

Hoe kunnen we zelf een bestand, met daarin een of meer MSXDOS-commando's, op schijf creeren? MSXDOS zelf biedt ons daartoe de mogelijkheid, en wel in de vorm van het COPY-commando. In het vorige hoofdstuk hebben we al gezien, hoe we met behulp van het COPY-commando een

bestand via het toetsenbord kunnen intikken en opslaan op schijf. Door nu, als "records" van het bestand, MSXDOS-commando's in te tikken, ontstaat er een bestand met commando's, die door MSXDOS kunnen worden uitgevoerd.

MSXDOS stelt echter een eis aan zo'n bestand. De bestandsnaam van een bestand met MSXDOS-commando's moet namelijk zijn voorzien van de bestandsnaamuitbreiding "BAT", om te kunnen worden uitgevoerd. Bestanden met de bestandsnaamuitbreiding BAT noemen we Batch-files. Het volgende voorbeeld laat zien wat hiermee wordt bedoeld:

```
A>copy con commando.xxx
dir a:
^Z
       1 file copied
A>commando
Bad command or file name

A>rename commando.xxx commando.bat

A>commando

A>dir a:
MSXDOS   SYS     2432  1-01-84
COMMAND  COM     6272  1-01-80
COMMANDO BAT        9  4-20-86
       3 files   169984 bytes free
A>
```

In voorgaand voorbeeld ziet u, hoe we met behulp van een COPY-commando een bestand, met daarin het MSXDOS-commando DIR A:, maken. De naam van dat nieuw te creeren bestand is COMMANDO.XXX. Nadat het bestand is gecreeerd, proberen we het door MSXDOS te laten uitvoeren. Dit doen we door eenvoudigweg de naam van het bestand (in dit geval COMMANDO) in te tikken. U ziet echter, dat de reactie van het systeem is, "Bad command or file name". Dit wil zeggen, dat het systeem het gevraagde bestand niet heeft kunnen vinden, omdat de naam van het gegeven commando niet op de schijf voorkomt. Toch hebben we geen typefout

gemaakt. Bij het uitvoeren van een commando (of, zoals in dit geval een commando-bestand) gaat MSXDOS ervan uit, dat de bestandsnaamuitbreiding ".BAT" is. Dientengevolge zou de naam van het commandobestand dus COMMANDO.BAT moeten zijn. De naam is echter COMMANDO.XXX, en dus niet de juiste naam.

Na het veranderen van de naam van het commandobestand van COMMANDO.XXX in COMMANDO.BAT, zonder de inhoud van dat bestand te veranderen, geven we het commando COMMANDO nog eens. Zoals u ziet in het voorbeeld, wordt het commando nu wel gevonden en uitgevoerd. In het commandobestand staat het commando DIR A:. Dat commando zorgt er voor dat er een inhoudsopgave van schijf A: wordt gemaakt.

Nogmaals willen wij u er op wijzen, dat bij het intikken van een bestand, vanaf het toetsenbord naar een bestand op schijf, het laatste record leeg dient te zijn, op het indrukken van de CONTROL-toets samen met de letter Z na. Alleen dan weet MSXDOS dat het einde van het bestand, dat via het toetsenbord wordt ingetikt, is bereikt.

12.2 Zelf nieuwe commando's creeren.

Nu we weten, hoe we een "batch file" (een commando bestand) in principe kunnen maken, zullen we een paar voorbeelden gaan bekijken. Na deze voorbeelden te hebben bestudeerd, zult u zelf in staat zijn uw eigen commando's te maken.

Een commando voor het maken van een back up van een schijf zal door bijna iedereen wel eens worden gewenst. Daarom zullen we dat commando als eerste voorbeeld onder handen nemen. Het maken van een back up gaat met het COPY-commando. De daarbij te gebruiken parameters zijn bijvoorbeeld A:*.* B:. Er wordt dan een copie van schijf A: naar schijf B: gemaakt. We moeten de gebruiker echter gelegenheid geven om de juiste schijf in de juiste schijveneenheid te

leggen. Vandaar dat de totale procedure langer is dan alleen het COPY-commando.

```
A>copy con bu.bat
pause Zet de te copieren schijf in A:
copy A:*.* B:
rem   De copie bevat de bestanden:
dir a:
^Z
```

Voorgaand voorbeeld laat zien hoe het nieuwe commando wordt gemaakt. Met de eerste regel wordt aangegeven, dat op schijf A: het bestand BU.BAT moet worden gecreeerd. De invoer van de records komt van het toetsenbord (CON). De daaropvolgende regels zijn de records, die de inhoud van de "batch-file" vormen.

Met het PAUSE-commando vragen we de operator de te copieren schijf in de eenheid te laden. Zoals u zich wellicht herinnert, is het gevolg van het PAUSE-commando, dat MSXDOS wacht met het verder uitvoeren van commando's totdat de RETURN-toets is ingedrukt. Hier ziet u dus een nuttige toepassing van het PAUSE-commando.

Nadat de RETURN-toets is ingedrukt, wordt het volgende commando uitgevoerd. Dat is het commando dat de back up maakt, het COPY-commando. Het COPY-commando vraagt u vanzelf om, zodra dat nodig is, de schijf waarnaartoe moet worden gecopieerd, in de schijveneenheid te laden. Na uitvoering van de copie wordt u ook vanzelf gevraagd de orginele schijf weer in eenheid A: te laden.

Het hiernavolgende commando is een REM. Ook dit is weer een goed voorbeeld van het nut van een REM-commando. De tekst achter het REM-commando wordt op het beeldscherm afgedrukt, waarna MSXDOS onmiddellijk doorgaat met het uitvoeren van het volgende commando, in het voorbeeld het commando DIR A:.

Na het creeren van de batch-file BU.BAT kunnen we het

commando BU ingeven. BU is nu in feite een normaal MSXDOS-commando geworden. Na het intikken van dit commando zal MSXDOS eerst onderzoeken of BU een van de interne commando's is, doch zodra het ziet, dat het geen intern commando is, zal het op de schijf zoeken naar een bestand dat de naam BU.BAT heeft. Heeft MSXDOS zo'n bestand gevonden, dan worden de daarin voorkomende MSXDOS-commando's een voor een uitgevoerd. Wat er bij uitvoering van voorgaand voorbeeld allemaal gebeurt, is in de hiernavolgende beeldschermweergave te zien:

```
A>bu

A>pause Zet de te copieren schijf in A:
Strike a key when ready . . .

A>copy A:*.* B:
MSXDOS   SYS
COMMAND  COM
BU       BAT

Insert diskette for drive B:
and strike a key when ready
       3 files copied
A>
Insert diskette for drive A:
and strike a key when ready
rem    De copie bevat de bestanden:

A>dir a:
MSXDOS   SYS     2432  1-01-84
COMMAND  COM     6272  1-01-80
BU       BAT       99  4-20-86
       3 files   169984 bytes free
A>
A>
```

We zullen nu een voorbeeld gaan zien, waarbij een BASIC-programma wordt gestart vanuit MSXDOS. De te volgen procedure geldt voor elk BASIC-programma. Van het programma, waarmee we de procedure zullen toelichten, is hierna de listing afgedrukt.

```
10 KEY1,"dir A:"+CHR$(13)
20 KEY2,"dir B:"+CHR$(13)
30 KEY3,"A:"+CHR$(13)
40 KEY4,"B:"+CHR$(13)
50 KEY5,"BASIC"+CHR$(13)
60 CALL SYSTEM
```

Dit programma schrijven we weg naar schijf onder de naam KEYDEF.BAS.

Zoals u ziet doet dit programma niet veel anders dan het herdefinieren van de functietoetsen 1 tot en met 5. De nieuwe functies van de toetsen zijn echter MSXDOS-commando's. Hebben we dit programma eenmaal uitgevoerd, dan heeft het indrukken van een van de functietoetsen 1 tot en met 5 een MSXDOS-commando tot gevolg. Schakelen we nu over van DISK-BASIC naar MSXDOS, dan behouden de functietoetsen de laatst gedefinieerde functie. Een gevolg daarvan is, dat we daarna onder MSXDOS een aantal functietoetsen tot onze beschikking hebben, waarmee we MSXDOS-commando's kunnen geven. De laatste regel van het BASIC-programma zorgt ervoor, dat er automatisch naar MSXDOS wordt teruggekeerd.

Nu zouden we een MSXDOS-commandobestand kunnen maken, waarmee voorgaand BASIC-programma wordt uitgevoerd. We dienen dan de volgende records in te tikken in het bestand dat we KEYDEF.BAT zullen noemen.

```
A>copy con KEYDEF.BAT
BASIC keydef.bas
^Z
```

Tikken we hierna het MSXDOS-commando KEYDEF in, dan zal het BASIC-programma KEYDEF.BAS worden uitgevoerd. Dit programma herdefinieert de functietoetsen 1 tot en met 5 en keert daarna weer terug naar MSXDOS.

Een nadeel van het hebben van functietoetsen onder MSXDOS is, dat de functie van de toetsen niet op het beeldscherm wordt afgedrukt. Willen we weten wat de functie van iedere toets is, dan zullen we dat gewoon moeten onthouden, of we moeten een nieuw commando maken, waarmee we de waarde van de functietoetsen kunnen bekijken.

Om zo'n nieuw commando te kunnen maken, zullen we eerst weer een BASIC-programma schrijven, waarmee we een lijst van alle functietoetsen maken. De listing van dit programma volgt hierna:

```
10 CLS
20 KEYLIST
30 FOR I=0 TO 9
40 LOCATE 16,I:PRINT "= F";:PRINT USING
"##";I+1
50 NEXT I
55 PRINT
60 PRINT "RETURN = naar MSXDOS"
70 IF INKEY$<>CHR$(13) THEN GOTO 70
80 CALL SYSTEM
```

Na het intikken van dit programma, schrijven we het weg naar schijf onder de naam KEYLIST.BAS, voordat we weer terugkeren naar MSXDOS.

U ziet, dat we met dit programma de inhoud van iedere functietoets, gevolgd door het nummer van de betreffende functietoets, op het beeldscherm afdrukken. Hierna wacht het programma tot wij de RETURN-toets hebben ingedrukt. Na het indrukken van de RETURN-toets wordt het CALL SYSTEM statement uitgevoerd, waarmee wordt teruggekeerd naar MSXDOS.

U zult nu begrijpen, dat, indien we dit programma vanuit MSXDOS aanroepen, we de momentele inhoud van de functietoetsen krijgen te zien, waarna we direct weer terugkeren naar MSXDOS. Het lijkt nu alsof we binnen MSXDOS de functietoetswaarden hebben bekeken.

Om het voorgaande programma vanuit MSXDOS te kunnen aanroepen hebben we een .BAT-bestand nodig. In het volgende voorbeeld ziet u hoe we dit bestand maken. We hebben gekozen voor de naam HELP.BAT.

```
A>copy con help.bat
basic keylist.bas
^Z
```

Als we nu het commando HELP geven, krijgen we de waarden, die aan de functietoetsen zijn toegekend, te zien.

```
A>HELP

A>BASIC KEYLIST.BAS
DIR A:          = F 1
DIR B:          = F 2
A:              = F 3
B:              = F 4
BASIC           = F 5
COLOR 15,4,4    = F 6
CLOAD"          = F 7
CONT            = F 8
LIST.           = F 9
RUN             = F10

RETURN = naar MSXDOS
```

12.3 Automatisch starten van batch-files.

Tijdens het opstarten van het systeem kijkt MSX-BASIC of de schijveneenheid is aangeschakeld. Indien dit zo is, dan wordt gekeken of op de aangesloten schijf de bestanden

MSXDOS.SYS en COMMAND.COM voorkomen. Is dit het geval, dan wordt MSXDOS in het geheugen geladen en opgestart. MSXDOS kijkt nu of er op de schijf, waarvan het is opgestart, een bestand met de naam **AUTOEXEC.BAT** staat. Komt er een bestand met die naam voor op de schijf, dan wordt dat bestand uitgevoerd.

Als voorbeeld van wat we met deze faciliteit kunnen doen, geven we weer een BASIC-programma, dat vanuit een AUTOEXEC.BAT file wordt opgestart. De listing van het BASIC-programma volgt hierna. Tikt u dit BASIC-programma eerst in (nadat u bent overgeschakeld naar DISK-BASIC) en schrijft u dit BASIC-programma onder de naam PRORUN.BAS naar uw systeemschijf.

```
10 'Met dit programma wordt een over-
20 'zicht van alle op schijf staande
30 'files gegeven. Vervolgens kan een
40 'file met de cursor worden aange-
50 'wezen.
60 'Door op RETURN te drukken, wordt
70 'de naam van de aangewezen file in
80 'F$ gezet, en indien het een BASIC
90 'programma is, uitgevoerd.
100 '
110 SCREEN 0:WIDTH 40:KEY OFF:CLS:X=0:Y=
0:LOCATE X,Y:FILES
120 PRINT:PRINT:PRINT"Wijs de gewenste f
ile aan met de cursor.Druk vervolgens op
 RETURN."
130 LOCATE X,Y,1:I$=INKEY$
140 IF I$=CHR$(30) THEN Y=Y-1:IF Y<0 THE
N Y=0
150 IF I$=CHR$(28) THEN X=X+13:IF X>26 T
HEN X=0:Y=Y+1:IF Y>20 THEN Y=20
160 IF I$=CHR$(31) THEN Y=Y+1:IF Y>20 TH
EN Y=20
170 IF I$=CHR$(29) THEN X=X-13:IF X<0 TH
EN X=26:Y=Y-1:IF Y<0 THEN Y=0
180 IF I$<>CHR$(13) THEN 130
```

```
190 LOCATE 0,23,0:F$=""
200 FOR I=0 TO 11
210 F$=F$+CHR$(VPEEK(BASE(0)+X+I+Y*40))
220 NEXT I
230 PT$=RIGHT$(F$,3)
240 IF PT$="BAS" OR PT$="bas" THEN RUN F
$
250 PRINT "Dit is geen BASIC-programma";
260 FOR I=1 TO 500:NEXT I
270 GOTO 110
```

Het programma maakt een inhoudsopgave van de schijf, die op dat moment in de schijveneenheid ligt. Vervolgens kunt u een bestand aanwijzen, door de cursor met behulp van de cursor control toetsen op de naam van dat bestand te zetten. Drukt u daarna op de RETURN-toets, dan zal de naam van het aangewezen bestand in de variabele F$ worden gelezen.

Nu volgt er een controle op de bestandsnaamuitbreiding. Is de bestandsnaamuitbreiding niet "BAS" of "bas", dan gaat het programma ervan uit, dat het aangewezen bestand geen BASIC-programma is. Dit wordt ook door middel van een boodschap op het beeldscherm kenbaar gemaakt. Is het betreffende bestand wel een BASIC-programma, dan zal dat programma worden gestart.

Door nu dit programma automatisch te laten opstarten na aanschakelen van het systeem, wordt een eenvoudige manier van opstarten van een programma verkregen. We zien namelijk alle op de schijf staande programma's en wijzen eenvoudig het gewenste programma aan, zonder de naam daarvan te hoeven intikken. Het automatisch laten starten van het programma dienen we te bewerkstelligen door een batch-file te creeren, vanwaaruit het programma wordt gestart. Deze batch-file moet de naam AUTOEXEC.BAT hebben, om automatisch te worden opgestart na aanschakelen van de computer. De creatie van deze batch-file gaat als volgt (onder MSXDOS):

```
A>copy con autoexec.bat
```

```
basic prorun.bas
^Z
        1 file copied
A>
```

Verdere uitleg van de commando's in de batch-file lijkt me nu niet meer nodig. U hebt ze al in de andere commando bestanden gezien.

12.4 Variabelen in .BAT-bestanden.

Tot nu toe hebben we in onze zelfgedefinieerde commando's steeds vaste waarden gebruikt. Om aan te tonen, wat hiermee wordt bedoeld, het volgende voorbeeld:

```
A>copy con inhoud.bat
dir a:
^Z
```

Door dit MSXDOS-commando aan ons repertoire toe te voegen, kunnen we in het vervolg door het intikken van het woord INHOUD de inhoudsopgave van schijf A: op het beeldscherm afgedrukt krijgen. Wat we ook doen, we zullen altijd de inhoudsopgave van schijf A: krijgen. Onder MSXDOS bestaat echter de mogelijkheid om binnen de batch-file in plaats van de letter A een variabele te plaatsen. Hetzelfde commando gaat er dan als volgt uitzien:

```
A>copy con inhoud.bat
dir %1:
^Z
```

U ziet nu dat de letter A is vervangen door de tekens **%1.** Nu kunnen we het commando INHOUD laten volgen door een parameter. Stel dat we nu het volgende commando ingeven:

INHOUD B

Dan wordt de letter B ingevuld op de plaats waar in de batch-file de tekens %1 staan. Het gevolg is dus, dat het eerste

commando in de batch-file eigenlijk "DIR B:" is geworden. Uitvoering van dit commando heeft dan ook tot gevolg dat er een inhoudsopgave van schijf B: wordt gemaakt. Zouden we een volgende keer het commando INHOUD laten volgen door de letter A, dan zal er een inhoudsopgave van schijf A: worden gemaakt.

Het is ook toegestaan, om dezelfde variabele meerdere keren in de batch-file te laten voorkomen. In dat geval zal de variabele elke keer opnieuw door dezelfde parameter worden vervangen. Laten we, om dit te zien, nog een voorbeeldcommando maken.

```
A>copy con kopieer.bat
copy a:%1.asc a:%1.kop
^Z
```

Na het intikken van voorgaande batch-file hebben we een nieuw commando gecreeerd, namelijk het commando KOPIEER. Stel nu dat we van het bestand TEKST.ASC een kopie willen maken, dan zullen we het volgende commando geven:

KOPIEER TEKST
Het woord tekst wordt op twee plaatsen in dezelfde regel in de batch-file KOPIEER.BAT ingevuld, waardoor deze regel door MSXDOS wordt gelezen als:

COPY A:TEKST.ASC A:TEKST.KOP

We kunnen echter ook meerdere verschillende parameters vanuit het aanroepende commando doorgeven aan de batchfile. Het maximum aantal variabelen dat we binnen een batch-file kunnen specificeren is 10 (%0 tot en met %9). De variabelen %1 tot en met %9 hebben betrekking op de parameters, die achter het commando worden geplaatst, bij het aanroepen van de batch-file. De variabele %0 heeft betrekking op het commando zelf. Daar dit duidelijker kan worden gemaakt aan de hand van een voorbeeldje, dan met vele woorden, volgt hier nog een voorbeeld:

```
A>copy con merge.bat
copy a:%1.asc+a:%2.asc a:%3.asc
type a:%3.asc
^Z
```

Hiermee hebben we het commando MERGE gecreeerd, waarmee we twee bestanden kunnen kopieren naar een outputbestand. Het commando MERGE moet nu worden gevolgd door drie parameters. De eerste parameter na het commando MERGE wordt aan variabele %1 toegekend, de tweede parameter aan variabele %2 en de derde aan %3. Stel nu, dat we het volgende commando intikken:

MERGE ADRES TEKST BRIEF

Dan gaat de eerste regel van de batch-file er voor MSXDOS als volgt uitzien:

COPY A:ADRES.ASC+A:TEKST.ASC A:BRIEF.ASC

Het gevolg hiervan is dat er een nieuw bestand wordt gecreeerd met de naam BRIEF.ASC, waarin de inhoud van de bestanden ADRES.ASC en TEKST.ASC wordt samengevoegd. De laatste regel van de batch-file MERGE zal door MSXDOS worden gelezen als:

TYPE A:BRIEF.ASC

Uitvoering van dit commando heeft tot gevolg, dat het nieuw gecreeerde bestand BRIEF.ASC op het beeldscherm wordt afgedrukt. Immers, in het commando stond de variabele %3 en dit komt overeen met de derde parameter, die achter het commando MERGE werd ingetikt.

Na alle voorgaande voorbeelden zult u waarschijnlijk de eerste variabele (%0) hebben gemist. Deze variabele wordt altijd vervangen door de naam van de batch-file. Roepen we dus de batch-file aan met MERGE, en zouden we in die batchfile de variabele %0 gebruiken, dan zal MSXDOS de variabele %0 vervangen denken door het woord MERGE. Dit kunnen we

aantonen met het volgende voorbeeld:

```
A>copy con var0.bat
pause %1
pause %0
^Z
```

Roepen we nu deze batch-file aan, door het commando VAR0, gevolgd door de parameter "gezien" te geven, dan zien we het volgende op het beeldscherm verschijnen:

```
A>pause gezien
Strike a key when ready . . .

A>pause var0
Strike a key when ready . . .
```

Hieraan ziet u, dat %1 is vervangen door de parameter en dat %0 is vervangen door VAR0, de commando-naam. Ook ziet u aan dit voorbeeld, dat de variabelen niet in oplopende volgorde in de batch-file hoeven voor te komen. Iedere variabele mag op iedere willekeurige plaats binnen de batch-file worden gebruikt. Bovendien mag dezelfde variabele meerdere malen op verschillende willekeurige plaatsen worden gebruikt.

Appendix A
Een eenvoudige tekstverwerker

Na het opstarten van dit programma komt het volgende menu op het scherm, dat wordt gemaakt met programmaregels 120 tot en met 200:

1. Invoeren van tekst via toetsenbord
2. Schrijven van tekst op disk
3. Lezen van tekst van disk
4. Afdrukken op beeldscherm of printer
5. Wissen van regels tekst
6. Sorteren van regels tekst
7. Opzoeken van een woord
8. Wijzigen van een tekstregel
9. Programma beeindigen

1. Invoeren van tekst via toetsenbord.
De routine voor het invoeren van tekst staat op de regelnummers 250 tot en met 335. Heeft u nog niet eerder tekst gemaakt, dan zult u moeten beginnen met keuzemogelijkheid 1. Elke regel, die u intikt, krijgt een regelnummer. Het maximum aantal tekens per regel is 37. Om het invoeren van tekst te beeindigen, dient u de CONTROL-toets samen met de letter Q (Quit) in te drukken. Er wordt dan naar het hoofdmenu teruggesprongen.

2. Schrijven van tekst op disk.
De routine voor het wegschrijven van tekst naar disk staat op de regelnummers 350 tot en met 435. Nadat u tekst hebt ingetikt, zult u die tekst willen bewaren op disk. U zult worden gevraagd onder welke naam de tekst op de disk moet worden opgeslagen. Nadat de tekst is weggeschreven, wordt hiervan melding gedaan op het beeldscherm. Door op de

RETURN-toets te drukken keert het programma terug naar het hoofdmenu.

3. Lezen van tekst van disk.
Deze routine staat op de regels 450 tot en met 525. Heeft u deze keuze uit het hoofdmenu gemaakt, dan wordt u gevraagd welk bestand moet worden gelezen. Na het inlezen kunt u terugkeren naar het hoofdmenu door op de RETURN-toets te drukken.

4. Afdrukken op beeldscherm of printer.
Deze routine staat op de regels 550 tot en met 735. Na het kiezen van deze functie uit het hoofdmenu wordt een tweede menu op het beeldscherm afgedrukt, en wel:

1. Afdrukken op printer
2. Afdrukken op beeldscherm
3. Terug naar hoofdmenu

Kiest u voor afdrukken (op printer of beeldscherm), dan zult u worden gevraagd of de regels met of zonder regelnummer moeten worden afgedrukt. Deze vraag kan met de letter J of N worden beantwoord. Na de J of de N moet nog de RETURN-toets worden ingedrukt. Hierna wordt u gevraagd of alle regels moeten worden afgedrukt. Beantwoordt u de vraag met J, dan zal het afdrukken meteen beginnen. Beantwoordt u de vraag echter met N, dan zult u worden gevraagd vanaf welke regel de tekst dan wel moet worden afgedrukt. Raakt het scherm tijdens het afdrukken vol, dan kunt u door op de RETURN-toets te drukken steeds de volgende regel laten afdrukken.

5. Wissen van regels tekst.
Deze routine staat op de regels 750 tot en met 830. Na het kiezen van deze functie uit het hoofdmenu worden ons de volgende vragen gesteld. Vanaf welke regel? Tot en met welke regel? Na het beantwoorden van deze vragen zullen de opgegeven regels worden gewist. Bovendien worden de overblijvende regels hernummerd, zodat er weer een aansluitende tekst ontstaat.

6. Sorteren van regels tekst.
Deze routine staat op de regels 850 tot en met 910. Alle regels worden gesorteerd op ASCII-code. Dat wil zeggen, dat een regel die met een hoofdletter begint altijd voor een regel die met een kleine letter begint komt te staan. Afhankelijk van het aantal regels kan het enige tijd duren voordat het sorteren gereed is. U krijgt daarvan een melding op het beeldscherm. Door op de RETURN-toets te drukken keert u terug naar het hoofdmenu.

7. Opzoeken van een woord.
Deze routine staat op de regels 920 tot en met 980. U kunt het woord, dat u in de tekst wilt vinden, intikken, waarna deze routine dat woord gaat zoeken. Alle regels, waarin het te zoeken woord voorkomt, zullen op het scherm worden afgedrukt. Door uiteindelijk op de RETURN-toets te drukken kunt u terugkeren naar het hoofdmenu.

8. Wijzigen van een tekstregel.
Deze routine staat op de regels 1000 tot en met 1170. Na deze keuze te hebben gemaakt, krijgt u een korte hulppagina op het beeldscherm te zien:

-->	Cursor naar rechts
<--	Cursor naar links
INS	Tussenvoegen van een teken
DEL	Weghalen van een teken
HOME	Einde wijziging

Welke regel?

Op de vraag "Welke regel?", kunt u het nummer van de regel, die u wilt wijzigen, ingeven. U ziet dat u de cursor met de cursor control toetsen naar links en naar rechts kunt verplaatsen. Nadat u op de INS-toets hebt gedrukt, kunt u een letter tussenvoegen. De tussengevoegde letter wordt vlak voor de cursor ingevoegd. Door op de DEL-toets te drukken, verwijdert u de letter die onder de cursor staat. Wilt u het wijzigen van de regel beeindigen, dan drukt u op de HOME-toets. Hierdoor keert u terug naar het hoofdmenu.

Programma-listing:

```
100 REM * TEKSTVERWERKER *
105 CLEAR 4000
110 DIM I$(100)
115 WIDTH 40:LOCATE ,,1:KEY OFF
120 CLS
125 PRINT TAB(3);"T E K S T V E R W E R
K E R"
130 PRINT
135 PRINT "1. Invoeren van tekst via toe
tsenbord":PRINT
140 PRINT "2. Schrijven van tekst op dis
k":PRINT
145 PRINT "3. Lezen van tekst van disk":
PRINT
150 PRINT "4. Afdrukken op beeldscherm o
f printer":PRINT
155 PRINT "5. Wissen van regels tekst":P
RINT
160 PRINT "6. Sorteren van regels tekst"
:PRINT
165 PRINT "7. Opzoeken van een woord":PR
INT
170 PRINT "8. Wijzigen van een tekstrege
l":PRINT
175 PRINT "9. Programma beeindiging":PRI
NT:PRINT
180 INPUT "Uw keuze";K
185 IF K<1 OR K>9 THEN PRINT "Verkeerde
keuze!":GOTO 180
190 IF K=9 THEN END
195 ON K GOSUB 250,350,450,550,750,850,9
20,1000
200 GOTO 120
250 REM * INVOER VIA TOETSENBORD *
```

```
255 CLS
260 INPUT "Regelnummer (1-66)";R
265 IF R<1 OR R>66 THEN PRINT "Verkeerd
regelnummer!":GOTO 260
270 CLS
275 PRINT USING "##";R;:PRINT SPC(1);
280 F=1
285 I$=INKEY$:IF I$="" GOTO 285
290 IF I$=CHR$(8) AND LEN(I$(R))>0 THEN
PRINT CHR$(8);" ";CHR$(8);:I$(R)=LEFT$(I
$(R),LEN(I$(R))-1):GOTO 285
295 IF I$=CHR$(8) GOTO 285
300 IF I$=CHR$(17) GOTO 330
305 IF F=0 THEN I$(R)="":F=1
310 IF I$=CHR$(13) THEN PRINT:R=R+1:GOTO
 275
315 IF POS(0)=0 THEN PRINT TAB(3);
316 PRINT I$;
320 I$(R)=I$(R)+I$
325 GOTO 285
330 IF R>RM THEN RM=R-1
335 RETURN
350 REM * SCHRIJVEN VAN TEKST OP DISK *
355 CLS
360 PRINT "SCHRIJVEN VAN TEKST OP DISK":
PRINT
365 INPUT "Bestandsnaam";B$
370 OPEN "A:"+B$ AS #1 LEN=37
375 FIELD #1,37 AS F$
380 LSET F$=MKI$(RM)
385 PUT #1,1
390 FOR I=1 TO RM
395 LSET F$=I$(I)
400 PUT #1,I+1
405 NEXT I
410 CLOSE #1
```

```
415 PRINT
420 PRINT "Bestand ";B$;" is opgeslagen.
"
425 PRINT "Druk op RETURN-toets."
430 IF INKEY$<>CHR$(13) GOTO 430
435 RETURN
450 REM * LEZEN VAN TEKST VAN DISK *
455 CLS
460 PRINT "LEZEN VAN TEKST VAN DISK":PRI
NT
465 INPUT "Bestandsnaam";B$
470 OPEN "A:"+B$ AS #1 LEN=37
475 FIELD #1,37 AS F$
480 GET #1,1
485 RM=CVI(F$)
490 FOR I=1 TO RM
495 GET #1,I+1
500 I$(I)=F$
505 NEXT I
510 CLOSE #1
515 PRINT
520 PRINT "Bestand ";B$;" is geladen van
 disk."
525 GOTO 425
550 REM * AFDRUKKEN OP SCHERM/PRINTER *
555 CLS
560 PRINT TAB(9);"AFDRUKKEN VAN TEKST":P
RINT:PRINT
565 PRINT TAB(6);"1. Afdrukken op printe
r":PRINT
570 PRINT TAB(6);"2. Afdrukken op beelds
cherm":PRINT
575 PRINT TAB(6);"3. Terug naar hoofdmen
u":PRINT:PRINT
580 PRINT TAB(6);:INPUT "Uw keuze";K
585 IF K<1 OR K>3 THEN PRINT TAB(6);"Ver
```

```
keerde keuze!":GOTO 580
590 IF K=3 THEN RETURN
595 IF K=1 THEN OPEN "LPT:" AS #1:GOTO 6
05
600 IF K=2 THEN OPEN "CRT:" AS #1
605 PRINT
610 INPUT "Regels afdrukken met regelnrs
. (J/N)";N$
615 INPUT "Alle regels afdrukken
  (J/N)";J$
620 IF J$="J" OR J$="j" GOTO 685
625 INPUT "Vanaf regelnummer     ";E
630 INPUT "Tot en met regelnummer";L
635 IF E<1 OR L>RM THEN PRINT "Verkeerde
 regelnummers!":GOTO 625
640 CLS
645 FOR I=E TO L
650 IF N$="J" OR N$="j" THEN PRINT #1,US
ING "##";I;:PRINT #1,SPC(1);
655 PRINT #1,I$(I)
660 IF K=1 GOTO 675
665 IF CSRLIN<23 GOTO 675
670 IF INKEY$<>CHR$(13) GOTO 670
675 NEXT I
680 GOTO 725
685 CLS
690 FOR I=1 TO RM
695 IF N$="J" OR N$="j" THEN PRINT #1,US
ING "##";I;:PRINT #1,SPC(1);
700 PRINT #1,I$(I)
705 IF K=1 GOTO 720
710 IF CSRLIN<23 GOTO 720
715 IF INKEY$<>CHR$(13) GOTO 715
716 CLS
720 NEXT I
725 IF INKEY$<>CHR$(13) GOTO 725
```

```
730 CLOSE #1
735 RETURN
750 REM * WISSEN VAN REGELS TEKST *
755 CLS
760 PRINT "WISSEN VAN REGELS TEKST":PRIN
T
765 INPUT "Vanaf regelnummer     ";E
770 INPUT "Tot en met regelnummer";L
775 FOR I=E TO L
780 I$(I)=""
785 NEXT I
790 FOR I=L+1 TO RM
795 I$(E)=I$(I)
800 E=E+1
805 NEXT I
810 RM=E-1
815 FOR I=E TO RM
820 I$(I)=""
825 NEXT I
830 RETURN
850 REM * SORTEREN VAN REGELS TEKST *
855 CLS
860 PRINT "SORTEREN VAN REGELS TEKST":PR
INT
865 F=0:R1=-1
870 R1=R1+1
875 IF R1=RM AND F=0 GOTO 895
880 IF R1=RM GOTO 865
885 IF I$(R1)>I$(R1+1) THEN I1$=I$(R1):I
$(R1)=I$(R1+1):I$(R1+1)=I1$:F=1
890 GOTO 870
895 PRINT "Einde sorteerprocedure."
900 PRINT "Druk op RETURN-toets."
905 IF INKEY$<>CHR$(13) GOTO 905
910 RETURN
920 REM * OPZOEKEN VAN EEN WOORD *
```

```
925 CLS
930 PRINT "OPZOEKEN VAN EEN WOORD":PRINT
935 INPUT "Zoekwoord";Z$:PRINT:PRINT
940 Z1$=LEFT$(Z$,1)
945 FOR R=1 TO RM
950 FOR I=1 TO LEN(I$(R))
955 IF MID$(I$(R),I,1)<>Z1$ GOTO 965
960 IF MID$(I$(R),I,LEN(Z$))=Z$ THEN PRI
NT USING "##";R;:PRINT SPC(1);I$(R):GOTO
 970
965 NEXT I
970 NEXT R
975 IF INKEY$<>CHR$(13) GOTO 975
980 RETURN
1000 REM * WIJZIGEN VAN EEN TEKSTREGEL *
1005 CLS
1010 PRINT TAB(8);"WIJZIGEN VAN EEN TEKS
TREGEL":PRINT:PRINT
1015 PRINT TAB(5);"-->   Cursor naar rec
hts":PRINT
1020 PRINT TAB(5);"<--   Cursor naar lin
ks":PRINT
1025 PRINT TAB(5);"INS   Tussenvoegen va
n een teken":PRINT
1030 PRINT TAB(5);"DEL   Weghalen van ee
n teken":PRINT
1035 PRINT TAB(5);"HOME  Einde wijziging
":PRINT:PRINT
1040 PRINT TAB(5):INPUT "Welke regel";R
1045 PRINT:PRINT:PRINT USING "##";R;
1050 PRINT SPC(1);I$(R)
1055 P=1
1060 L=LEN(I$(R))
1065 LOCATE 3,17,1
1070 S$=INKEY$:IF S$="" GOTO 1070
1075 IF S$=CHR$(28) AND (P+2)<(L+3) THEN
```

```
 P=P+1:GOSUB 1155
1080 IF S$=CHR$(29) AND (P+2)>3 THEN P=P
-1:GOSUB 1155
1085 IF S$=CHR$(11) THEN RETURN
1090 IF S$<>CHR$(127) GOTO 1120
1095 IF P>L OR L=1 GOTO 1150
1100 IF P=L THEN I$(R)=LEFT$(I$(R),L-1):
GOTO 1115
1105 IF P=1 THEN I$(R)=RIGHT$(I$(R),L-1)
:GOTO 1115
1110 I$(R)=LEFT$(I$(R),P-1)+RIGHT$(I$(R)
,L-P)
1115 L=L-1:GOSUB 1155
1120 IF S$<>CHR$(18) GOTO 1150
1125 T$=INKEY$:IF T$="" GOTO 1125
1130 IF P=L+1 THEN I$(R)=I$(R)+T$:GOTO 1
145
1135 IF P=1 THEN I$(R)=T$+I$(R):GOTO 114
5
1140 I$(R)=LEFT$(I$(R),P-1)+T$+RIGHT$(I$
(R),L-P+1)
1145 L=L+1:GOSUB 1155
1150 GOTO 1070
1155 LOCATE 3,17
1160 PRINT I$(R);" "
1165 LOCATE (P+2),17
1170 RETURN
```

Appendix B Voetbalcompetitie

Het programma VOET dient ervoor, om alle voetbaluitslagen van het hele seizoen op te slaan. Hiertoe dienen alle voetbalverenigingen die aan de competitie deelnemen in DATAregels te worden opgeslagen. Tijdens de uitvoering van het programma wordt de operator-invoer met behulp van deze DATA-regels gecontroleerd op correctheid. Om de controle eenvoudig te houden, is ervoor gekozen, dat alle invoer in hoofdletters gebeurt. Voordat u het programma begint, dient u dus te zorgen dat de computer op hoofdletters staat ingesteld, door zonodig op de CAPS-toets te drukken.

Zodra u enige uitslagen hebt ingevoerd, kunt u gegevens opvragen, zoals "welke wedstrijden moet die en die club nog spelen?" en "wat is de momentele stand van de competitie" en "wat was de uitslag van die en die wedstrijd?".

Na het opstarten van het programma krijgt u het volgende menu:

1. Bijwerken uitslagen
2. Standen
3. Uitslag bepaalde wedstrijd
4. Nog te spelen wedstrijden
5. Invoeren uitslagen bestand
6. Genereren uitslagen bestand
7. Programma beeindigen

Wanneer u voor de eerste keer met dit programma gaat werken, dient u keuzenummer 6 als eerste te nemen.

1. Bijwerken uitslagen
Deze routine staat op de regelnummers 2000 tot en met 2235. Op het scherm verschijnt de tekst UITSLAGEN. Nu kunt u beginnen met het invoeren van de uitslagen. Tik eerst de

thuisspelende club in, dan het koppelteken (-), dan de bezoekende club, weer een koppelteken (-), en tenslotte de uitslag (Bijvoorbeeld: 1-2). Pas na de uitslag mag de RETURN-toets worden ingedrukt. Maakt u tijdens het intikken van een uitslagregel een fout, dan kunt u de betreffende uitslag het beste helemaal opnieuw intikken. Na iedere uitslag krijgt u de vraag of dit de laatste uitslag was. Zegt u nee (N), dan kunt u een volgende uitslag intikken. Zegt u ja (J), dan vraagt het systeem u een diskette in de drive te laden. Nadat u door middel van het indrukken van de RETURN-toets hebt aangegeven dat de diskette geladen is, zal het uitslagenbestand op schijf worden weggeschreven en keert u terug naar het hoofdmenu.

2. Standen.
Deze routines staan op de regelnummers 2500 tot en met 2980. Het oproepen van deze routine zorgt ervoor dat de huidige stand van de competitie wordt uitgerekend. Dit neemt even tijd, weest u dus niet ongerust als het wat langer duurt. Uiteindelijk krijgt u de stand te zien. De cursor blijft aan het einde van de tabel staan. Door nu op de RETURN-toets te drukken, keert u terug naar het hoofdmenu.

3. Uitslag bepaalde wedstrijd.
Deze routine staat op de regelnummers 3000 tot en met 3045. Na deze keuze uit het hoofdmenu te hebben gemaakt, wordt u gevraagd de door u gewenste wedstrijd in te tikken. Dit dient op de volgende manier te worden gedaan: Eerst de naam van de thuisspelende club, dan een koppelteken (-), dan de bezoekende club en tenslotte weer een koppelteken (-). Hierna zal het systeem voor u de uitslag van de wedstrijd opzoeken en op het scherm afdrukken. Was de gevraagde wedstrijd nog niet gespeeld, dan zal als uitslag "FF" worden afgedrukt.

4. Nog te spelen wedstrijden.
Deze routine staat op de regelnummers 3500 tot en met 3615. U zult worden gevraagd naar de club, waarvan de computer moet opzoeken welke wedstrijden deze nog moet spelen. Hierna verschijnen twee tabellen op het scherm. De linker tabel geeft de nog te spelen thuiswedstrijden, de rechter

tabel geeft de nog te spelen uitwedstrijden. Door op de RETURN-toets te drukken keert u terug naar het hoofdmenu.

5. Invoeren uitslagenbestand.
Deze routine staat op de regelnummers 4000 tot en met 4090. Met deze routine worden uitslagen, die met keuzenummer 1 van het hoofdmenu naar schijf zijn geschreven, in het geheugen van de computer gelezen. U zult worden gevraagd om de schijf (met daarop het bestand met uitslagen) in de drive te laden en met het indrukken van de RETURN-toets aan te geven dat de schijf is geladen. Hierna leest de computer het uitslagenbestand in het geheugen, meldt dit op het beeldscherm en wacht, tot u weer op de RETURN-toets hebt gedrukt. Hierna keert u terug naar het hoofdmenu.

6. Genereren uitslagenbestand.
Deze routine staat op de regels 4500 tot en met 4570. Deze routine dient u alleen de eerste keer, dat u de uitslagen wilt gaan bijhouden, te starten. De matrix met uitslagen wordt met deze routine geinitialiseerd. Dat wil zeggen, dat alle vakjes binnen de matrix met de letter FF worden geladen. Later, bij het bijwerken van de uitslagen, worden deze letters (FF) met de uitslagen overschreven. Om te voorkomen dat al bestaande uitslagen per ongeluk worden geinitialiseerd, vraagt deze routine of u er wel zeker van bent, dat deze routine moet worden gestart.

7. Programma beeindigen.
Deze routine staat op de regelnummers 1065 tot en met 1071. Hiermee wordt het programma beeindigd.

Programma-listing:

```
1000 REM * VOETBALUITSLAGEN *
1005 WIDTH 40:KEY OFF
1015 CLEAR 2000
1016 DIM A$(17,17),S(18,7)
1020 CLS
1025 PRINT TAB(4);"VOETBALUITSLAGEN EN S
```

```
TANDEN"
1030 PRINT:PRINT
1035 PRINT TAB(4);"1. Bijwerken uitslage
n":PRINT
1040 PRINT TAB(4);"2. Standen":PRINT
1045 PRINT TAB(4);"3. Uitslag bepaalde w
edstrijd":PRINT
1050 PRINT TAB(4);"4. Nog te spelen weds
trijden":PRINT
1055 PRINT TAB(4);"5. Invoeren uitslagen
 bestand":PRINT
1056 PRINT TAB(4);"6. Genereren uitslage
n bestand":PRINT
1060 PRINT TAB(4);"7. Programma beeindig
ing":PRINT:PRINT
1065 PRINT TAB(4):INPUT "Uw keuze";K
1070 IF K<1 OR K>7 THEN PRINT TAB(4);"Ve
rkeerde keuze!":GOTO 1065
1071 IF K=7 THEN CLOSE:END
1075 ON K GOSUB 2000,2500,3000,3500,4000
,4500
1080 GOTO 1020
1090 DATA AJAX,AZ'67,FC DEN BOSCH,EXELSI
OR,FEYENOORD,FORTUNA SITTARD,GO AHEAD EA
GLES,FC GRONINGEN,HAARLEM
1100 DATA HERACLES,MVV,NEC ,PSV,RODA JC,
SPARTA,FC TWENTE,FC UTRECHT,FC VVV
2000 REM * BIJWERKEN UITSLAGEN *
2005 FLAG=0
2010 CLS:LOCATE ,,1
2020 PRINT "UITSLAGEN"
2025 M=3:PRINT
2030 LOCATE 0,M:V1$="":RESTORE
2035 V$=INKEY$:IF V$="" GOTO 2035
2040 IF V$<>"-" THEN PRINT V$;:V1$=V1$+V
$:GOTO 2035
```

```
2045 FOR I=0 TO 17
2050 READ D$:L=LEN(V1$)
2060 IF V1$=LEFT$(D$,L) GOTO 2080
2065 NEXT I
2066 LOCATE 0,22:PRINT SPACE$(30):LOCATE
 0,22
2070 PRINT "Naam thuisclub verkeerd!":GO
TO 2030
2080 LOCATE 16,M:PRINT "-";
2085 LOCATE 18,M:V1$="":RESTORE
2090 V$=INKEY$:IF V$="" GOTO 2090
2095 IF V$<>"-" THEN PRINT V$;:V1$=V1$+V
$:GOTO 2090
2100 FOR J=0 TO 17
2105 READ D$:L=LEN(V1$)
2110 IF V1$=LEFT$(D$,L) GOTO 2126
2115 NEXT J
2120 LOCATE 0,22:PRINT SPACE$(30):LOCATE
 0,22
2125 PRINT "Naam bezoekende club verkeer
d!":GOTO 2085
2126 IF FLAG=1 GOTO 3030
2130 LOCATE 34,M:V1$=""
2135 V$=INKEY$:IF V$="" GOTO 2135
2140 IF V$<>CHR$(13) THEN PRINT V$;:V1$=
V1$+V$:GOTO 2135
2145 A$(I,J)=V1$:M=M+2
2150 LOCATE 0,22:PRINT SPACE$(30):LOCATE
 0,22
2155 INPUT "Laatste uitslag";J$
2160 IF J$="J" OR J$="j" GOTO 2170 ELSE
GOTO 2030
2170 CLS
2175 PRINT "Laad diskette in drive."
2185 PRINT "Druk op RETURN-toets voor sc
hrijfaktie."
```

```
2190 IF INKEY$<>CHR$(13) GOTO 2190
2200 OPEN "A:VOET" FOR OUTPUT AS #1
2205 FOR I=0 TO 17
2210 FOR J=0 TO 17
2215 PRINT #1,A$(I,J)
2220 NEXT J
2225 NEXT I
2230 CLOSE #1
2235 RETURN
2500 REM * STANDEN *
2505 CLS:GOSUB 2955
2510 PRINT "STAND EREDIVISIE":PRINT
2511 SW=0
2515 FOR I=0 TO 17
2520 FOR J=0 TO 17
2525 A1$=A$(I,J):IF A1$="FF" GOTO 2574
2530 A2$=MID$(A1$,2,1)
2535 IF A2$="-" THEN T=VAL(LEFT$(A1$,1))
ELSE T=VAL(LEFT$(A1$,2))
2540 A2$=RIGHT$(A1$,2)
2545 A3$=LEFT$(A2$,1)
2550 IF A3$="-" THEN U=VAL(RIGHT$(A2$,1)
) ELSE U=VAL(RIGHT$(A2$,2))
2551 IF SW=1 GOTO 2595
2555 IF T-U>0 THEN S(I,1)=S(I,1)+1:S(I,5
)=S(I,5)+2
2560 IF T-U<0 THEN S(I,3)=S(I,3)+1
2565 IF T=U THEN S(I,2)=S(I,2)+1:S(I,5)=
S(I,5)+1
2570 S(I,4)=S(I,4)+1:S(I,6)=S(I,6)+T:S(I
,7)=S(I,7)+U
2574 IF SW=1 GOTO 2611
2575 NEXT J
2576 NEXT I
2577 SW=1
2580 FOR J=0 TO 17
```

```
2585 FOR I=0 TO 17
2590 GOTO 2525
2595 IF T-U>0 THEN S(J,3)=S(J,3)+1
2600 IF T-U<0 THEN S(J,1)=S(J,1)+1:S(J,5
)=S(J,5)+2
2605 IF T=U THEN S(J,2)=S(J,2)+1:S(J,5)=
S(J,5)+1
2610 S(J,4)=S(J,4)+1:S(J,6)=S(J,6)+U:S(J
,7)=S(J,7)+T
2611 S(I,0)=I
2615 NEXT I
2620 NEXT J
2630 FOR I=0 TO 16
2635 FOR J=0 TO 16
2640 IF S(J,5)<S(J+1,5) GOTO 2665
2645 IF S(J,5)>S(J+1,5) GOTO 2680
2650 IF S(J,4)>S(J+1,4) GOTO 2665
2655 V1=S(J,6)-S(J,7):V2=S(J+1,6)-S(J+1,
7)
2660 IF V1-V2>=O GOTO 2680
2665 FOR K=0 TO 7
2670 H(0,0)=S(J,K):S(J,K)=S(J+1,K):S(J+1
,K)=H(0,0)
2675 NEXT K
2680 NEXT J
2685 NEXT I
2690 CLS
2695 PRINT "STAND ERE-DEVISIE":PRINT
2700 FOR I=0 TO 17
2705 RESTORE
2710 FOR J=0 TO S(I,0)
2715 READ V$
2720 NEXT J
2725 PRINT V$;TAB(15);
2730 PRINT USING "###";S(I,1);
2735 PRINT TAB(18);
```

```
2740 PRINT USING "###";S(I,2);
2745 PRINT TAB(21);
2750 PRINT USING "###";S(I,3);
2755 PRINT TAB(24);
2760 PRINT USING "###";S(I,4);
2765 PRINT TAB(27);
2770 PRINT USING "###";S(I,5);
2775 PRINT TAB(30);
2780 PRINT USING "###";S(I,6);
2785 PRINT TAB(33);"-";
2790 PRINT TAB(34);
2795 PRINT USING "###";S(I,7)
2800 NEXT I
2805 IF INKEY$<>CHR$(13) GOTO 2805
2810 RETURN
2955 FOR I=0 TO 17
2960 FOR J=0 TO 7
2965 S(I,J)=0
2970 NEXT J
2975 NEXT I
2980 RETURN
3000 REM * UITSLAG BEPAALDE WEDSTRIJD *
3010 CLS:M=4
3015 PRINT "UITSLAG BEPAALDE WEDSTRIJD":
PRINT
3020 PRINT "Welke wedstrijd?":PRINT
3025 FLAG=1:GOTO 2030
3030 LOCATE 34,M:PRINT A$(I,J):PRINT
3035 INPUT "Volgende wedstrijd";J$
3040 IF J$="J" OR J$="j" THEN M=M+5:IF C
SRLIN<17 GOTO 3020 ELSE GOTO 3010
3045 RETURN
3500 REM * NOG TE SPELEN WEDSTRIJDEN *
3515 CLS:RESTORE
3520 PRINT "NOG TE SPELEN WEDSTRIJDEN"
3525 INPUT "Welke vereniging";V$:PRINT
```

```
3530 PRINT TAB(4);"THUIS";TAB(24);"UIT"
3535 FOR I=0 TO 17
3540 READ D$:L=LEN(V$)
3545 IF V$=LEFT$(D$,L) GOTO 3560
3550 NEXT I
3555 LOCATE 0,22:PRINT "Vereniging verke
erd geschreven!":LOCATE 0,2:RESTORE:GOTO
 3525
3560 M=5:RESTORE
3565 FOR J=0 TO 17
3566 READ D$
3570 IF I=J GOTO 3580
3575 IF A$(I,J)="FF" THEN LOCATE 2,M:M=M
+1:PRINT D$
3580 NEXT J
3585 M=5:J=I:RESTORE
3590 FOR I=0 TO 17
3591 READ D$
3595 IF I=J GOTO 3605
3600 IF A$(I,J)="FF" THEN LOCATE 22,M:M=
M+1:PRINT D$
3605 NEXT I
3610 IF INKEY$<>CHR$(13) GOTO 3610
3615 RETURN
4000 REM * INVOEREN UITSLAGEN BESTAND *
4005 CLS
4010 PRINT "INVOEREN UITSLAGEN BESTAND":
PRINT
4015 PRINT "Plaats diskette in drive."
4030 PRINT "Druk op RETURN-toets voor le
esaktie."
4035 IF INKEY$<>CHR$(13) GOTO 4035
4040 OPEN "A:VOET" FOR INPUT AS #1
4045 FOR I=0 TO 17
4050 FOR J=0 TO 17
4055 INPUT #1,A$(I,J)
```

```
4060 NEXT J
4065 NEXT I
4070 PRINT:PRINT "Bestand is gelezen."
4075 PRINT "Druk op RETURN-toets."
4080 IF INKEY$<>CHR$(13) GOTO 4080
4085 CLOSE #1
4090 RETURN
4500 REM * GENEREREN UITSLAGEN BESTAND *
4505 CLS
4510 PRINT "GENEREREN UITSLAGEN BESTAND"
:PRINT
4515 INPUT "Bent U er zeker van (J/N)";J
$
4520 IF J$="J" OR J$="j" GOTO 4530
4525 RETURN
4530 FOR I=0 TO 17
4535 FOR J=0 TO 17
4540 A$(I,J)="FF"
4545 NEXT J
4550 NEXT I
4555 PRINT "Einde genereren bestand."
4560 PRINT "Druk op RETURN-toets."
4565 IF INKEY$<>CHR$(13) GOTO 4565
4570 GOTO 4525
```

Appendix C
Schaatstoernooi voor all-rounders

Met dit programma kunt u de stand van zaken in een internationaal schaatstoernooi voor all rounders prachtig bijhouden. Door alle uitslagen in te tikken, hebt u op ieder gewenst moment de beschikking over gegevens als "de momentele stand", "standen per afstand", "totaalstand na een bepaalde afstand, met de verschillen voor de volgende afstand", etc.

Na het opstarten van het programma krijgt u het volgende menu op het beeldscherm:

1. Invoeren en overzicht 500, 1500, 5000 en 10000 meter.
2. Totaal overzicht
3. Genereren van bestand
4. Programma beeindigen

Voordat u gegevens kunt invoeren, dient u eerst een bestand te genereren met keuzenummer 3 van het menu. Daarna dient u die functie niet weer te kiezen.

1. Invoeren en overzicht 500, 1500, 5000 en 10000 meter.
Deze routine staat op de regelnummers 2000 tot en met 2335. Het kiezen van deze functie uit het hoofdmenu heeft tot gevolg, dat u gevraagd wordt van welke afstand u gegevens wilt invoeren of een overzicht wilt hebben. Na het ingeven van de gewenste afstand verschijnt een volgend menu:

1. Invoeren tijden xxx meter

2. Totaal overzicht xxx meter
3. Terug naar het hoofdmenu

Kiest u voor functienummer 1 uit dit menu, dan wordt u gevraagd de naam en de tijd van een schaatser op de gekozen afstand in te geven. De naam moet voorkomen in het bestand, zoals dat met functie nummer 3 uit het hoofdmenu is gegenereerd. Na het invoeren van een of meer uitslagen kunt u terugkeren naar het laatste menu. Nu zou u een totaal overzicht kunnen maken van de afstand waarvoor u zojuis een uitslag hebt ingetikt, door functie nummer 2 te kiezen. In deze uitslag krijgt u van links naar rechts kolommen te zien met daarin de stand, de naam, het land en de tijd. De kolommen zijn opgesteld in volgorde van snelheid.

2. Totaal overzicht.
Deze routine staat op de regelnummers 3000 tot en met 3660. Door deze functie uit het hoofdmenu te kiezen, verschijnt er een tweede menu op het scherm, met de volgende inhoud:

1. Na 500 meter
2. Na 1500 meter
3. Na 5000 meter
4. Na 10000 meter
5. Terug naar hoofdmenu

De overzichten die na een keuze uit dit menu verschijnen, bevatten tabellen met de volgende gegevens, in volgorde van het puntentotaal van de schaatsers:

Plaats,
Naam van de schaatser,
Land,
Puntentotaal,
Tijdsverschil voor de volgende afstand.

Na het afdrukken van de tabel kunt u terugkeren naar het sub-menu door op de RETURN-toets te drukken. Vanuit dat menu kunt u terugkeren naar het hoofdmenu.

3. Genereren van bestand.
Deze routine staat op de regelnummers 4000 tot en met 4125. Deze functie dient alleen bij de eerste keer, dat gegevens voor een toernooi wordt bijgehouden, te worden gestart. De gegevensmatrix wordt geinitialiseerd (alle oude gegevens worden dus gewist) voordat u wordt gevraagd, de namen van de schaatsers en de landen die zij vertegenwoordigen, in te tikken. Om te voorkomen dat bestaande gegevens per ongeluk worden gewist, wordt u eerst gevraagd of u er zeker van bent dat u een nieuw bestand wilt genereren.

4. Programma beeindigen.
Deze routine staat op regelnummer 1080. Verder spreekt deze keuze voor zichzelf.

Programma-listing:

```
1000 REM * SCHAATSUITSLAGEN *
1005 WIDTH 40:KEY OFF
1010 CLEAR 4000
1015 DIM T$(35,3),T(35)
1020 OPEN "A:SCHAATS" AS #1 LEN=88
1025 FIELD #1,17 AS A1$,3 AS A2$,5 AS B1
$,1 AS B2$,7 AS B3$,7 AS C1$,1 AS C2$,7
AS C3$,7 AS D1$,1 AS D2$,7 AS D3$,8 AS E
1$,1 AS E2$,7 AS E3$,8 AS F1$,1 AS F2$
1030 CLS
1035 PRINT TAB(6);"MENU SCHAATSKAMPIOENS
CHAPPEN"
1040 PRINT:PRINT
1045 PRINT TAB(6);"1. Overzicht afstande
n 500,"
1050 PRINT TAB(6);"   1500,5000 en 10000
 meter":PRINT
1055 PRINT TAB(6);"2. Totaal overzicht":
PRINT
1060 PRINT TAB(6);"3. Genereren van best
and":PRINT
```

```
1065 PRINT TAB(6);"4. Programma beeindig
ing":PRINT:PRINT
1070 PRINT TAB(6);:INPUT "Wat is uw keuz
e";K
1075 IF K<1 OR K>4 THEN PRINT TAB(6);"Ve
rkeerde keuze!":GOTO 1070
1080 IF K=4 THEN CLOSE #1:END
1085 ON K GOSUB 2000,3000,4000
1090 GOTO 1030
2000 REM * OVERZICHT AFSTANDEN *
2005 GET #1,1:E=CVI(A1$)
2010 CLS
2015 INPUT "Afstand (500,1500,5000,10000
)";M
2020 IF M=500 OR M=1500 OR M=5000 OR M=1
0000 GOTO 2030
2025 PRINT "Verkeerde keuze!":GOTO 2015
2030 PRINT:PRINT
2035 PRINT "UITSLAGEN";M;"METER"
2040 PRINT:PRINT
2045 PRINT "1. Invoeren tijden";M;"meter
":PRINT
2050 PRINT "2. Totaal overzicht";M;"mete
r":PRINT
2055 PRINT "3. Terug naar hoofdmenu"
2060 PRINT:PRINT
2065 INPUT "Uw keuze is";K
2070 IF K<1 OR K>3 THEN PRINT "Verkeerde
 keuze!":GOTO 2065
2075 IF K=3 THEN RETURN
2080 IF K=2 GOTO 2170
2085 CLS:F=0
2090 INPUT "Naam                ";N$:L=LE
N(N$)
2095 INPUT "Uitslag (XX.)YY.ZZ ";U$
2100 FOR I=2 TO E-1
```

```
2105 GET #1,I
2110 IF N$=LEFT$(A1$,L) THEN F=1:GOTO 21
25
2115 NEXT I
2120 IF F=0 THEN PRINT "Naam niet gevond
en!":PRINT:GOTO 2155
2125 PRINT:PRINT "Naam is ";A1$;:PRINT
2130 IF M=500 THEN LSET B1$=U$:LSET B2$=
"F":X=VAL(U$):LSET B3$=MKS$(X):GOTO 2150
2135 IF M=1500 THEN LSET C1$=U$:LSET C2$
="F":X=(60*VAL(LEFT$(U$,1))+VAL(RIGHT$(U
$,5)))/3:LSET C3$=MKS$(X):GOTO 2150
2140 IF M=5000 THEN LSET D1$=U$:LSET D2$
="F":X=(60*VAL(LEFT$(U$,1))+VAL(RIGHT$(U
$,5)))/10:LSET D3$=MKS$(X):GOTO 2150
2145 IF M=10000 THEN LSET E1$=U$:LSET E2
$="F":X=(60*VAL(LEFT$(U$,2))+VAL(RIGHT$(
U$,5)))/20:LSET E3$=MKS$(X)
2150 PUT #1,I
2155 INPUT "Volgende ingave (J/N)";J$
2160 IF J$="J" OR J$="j" GOTO 2085
2165 CLS:GOTO 2030
2170 J=0
2175 FOR I=2 TO E-1
2180 GET #1,I
2185 IF M=500 THEN IF B2$="X" GOTO 2230
ELSE GOTO 2205
2190 IF M=1500 THEN IF C2$="X" GOTO 2230
 ELSE GOTO 2210
2195 IF M=5000 THEN IF D2$="X" GOTO 2230
 ELSE GOTO 2215
2200 IF M=10000 THEN IF E2$="X" GOTO 223
0 ELSE GOTO 2220
2205 T$(J,0)=A1$:T$(J,1)=A2$:T$(J,2)=B1$
:T(J)=CVS(B3$):GOTO 2225
2210 T$(J,0)=A1$:T$(J,1)=A2$:T$(J,2)=C1$
```

```
:T(J)=CVS(C3$):GOTO 2225
2215 T$(J,0)=A1$:T$(J,1)=A2$:T$(J,2)=D1$
:T(J)=CVS(D3$):GOTO 2225
2220 T$(J,0)=A1$:T$(J,1)=A2$:T$(J,2)=E1$
:T(J)=CVS(E3$)
2225 J=J+1
2230 NEXT I
2235 FOR N=0 TO J-2
2240 FOR I=0 TO J-2
2245 IF T(I)<T(I+1) GOTO 2270
2250 FOR K=0 TO 2
2255 T1$(0,0)=T$(I,K):T$(I,K)=T$(I+1,K):
T$(I+1,K)=T1$(0,0)
2260 T=T(I):T(I)=T(I+1):T(I+1)=T
2265 NEXT K
2270 NEXT I
2275 NEXT N
2280 CLS
2285 PRINT "OVERZICHT";M;"METER":PRINT
2290 PRINT "NR";TAB(4);"NAAM";TAB(25);"L
AND";TAB(31);"TIJD"
2295 PRINT
2300 FOR I=0 TO J-1
2305 PRINT USING "##";I+1;
2310 PRINT TAB(4);T$(I,0);
2315 PRINT TAB(25);T$(I,1);
2320 PRINT TAB(31);T$(I,2):PRINT
2325 NEXT I
2330 IF INKEY$<>CHR$(13) GOTO 2330
2335 CLS:GOTO 2030
3000 REM * TOTAAL OVERZICHT *
3005 GET #1,1:E=CVI(A1$)
3010 CLS
3015 PRINT "TOTAAL OVERZICHT":PRINT:PRIN
T
3020 PRINT "1. Na 500 meter":PRINT
```

```
3025 PRINT "2. Na 1500 meter":PRINT
3030 PRINT "3. Na 5000 meter":PRINT
3035 PRINT "4. Na 10000 meter":PRINT
3040 PRINT "5. Terug naar hoofdmenu":PRI
NT:PRINT
3045 INPUT "Uw keuze";K
3050 IF K<1 OR K>5 THEN PRINT "Verkeerde
 keuze!":GOTO 3045
3055 IF K=5 THEN RETURN
3060 ON K GOSUB 3100,3200,3300,3400
3065 GOTO 3010
3100 REM * NA 500 METER *
3105 J=0
3110 FOR I=2 TO E-1
3115 GET #1,I
3120 IF B2$="X" GOTO 3135
3125 T$(J,0)=A1$:T$(J,1)=A2$:T(J)=CVS(B3
$)
3130 J=J+1
3135 NEXT I
3140 FLAG=1
3145 GOSUB 3500
3150 RETURN
3155 GOSUB 3500
3160 RETURN
3200 REM * NA 1500 METER *
3205 J=0
3210 FOR I=2 TO E-1
3215 GET #1,I
3220 IF B2$="X" GOTO 3245
3225 IF D2$="X" GOTO 3245
3230 IF C2$="X" GOTO 3245
3235 T$(J,0)=A1$:T$(J,1)=A2$:T(J)=CVS(B3
$)+CVS(C3$)+CVS(D3$)
3240 J=J+1
3245 NEXT I
```

```
3250 FLAG=2
3255 GOSUB 3500
3260 RETURN
3300 REM * NA 5000 METER *
3305 J=0
3310 FOR I=2 TO E-1
3315 GET #1,I
3320 IF B2$="X" GOTO 3340
3325 IF D2$="X" GOTO 3340
3330 T$(J,0)=A1$:T$(J,1)=A2$:T(J)=CVS(B3
$)+CVS(D3$)
3335 J=J+1
3340 NEXT I
3345 FLAG=3
3350 GOSUB 3500
3355 RETURN
3400 REM * NA 10000 METER *
3405 J=0
3410 FOR I=2 TO E-1
3415 GET #1,I
3420 IF B2$="X" GOTO 3450
3425 IF C2$="X" GOTO 3450
3430 IF D2$="X" GOTO 3450
3435 IF E2$="X" GOTO 3450
3440 T$(J,0)=A1$:T$(J,1)=A2$:T(J)=CVS(B3
$)+CVS(C3$)+CVS(D3$)+CVS(E3$)
3445 J=J+1
3450 NEXT I
3455 FLAG=4
3460 GOSUB 3500
3465 RETURN
3500 REM * AFDRUKKEN VAN STANDEN *
3505 FOR N=0 TO J-2
3510 FOR I=0 TO J-2
3515 IF T(I)<T(I+1) GOTO 3540
3520 FOR K=0 TO 1
```

```
3525 T1$(0,0)=T$(I,K):T$(I,K)=T$(I+1,K):
T$(I+1,K)=T1$(0,0)
3530 NEXT K
3535 T=T(I):T(I)=T(I+1):T(I+1)=T
3540 NEXT I
3545 NEXT N
3550 IF FLAG=1 THEN S$="500":V$="5000"
3555 IF FLAG=2 THEN S$="1500":V$="10000"
3560 IF FLAG=3 THEN S$="5000":V$="1500"
3565 IF FLAG=4 THEN S$="10000"
3570 CLS
3575 PRINT "STAND NA ";S$;" METER":PRINT
3580 PRINT "NR";TAB(4);"NAAM";TAB(22);"L
AND";TAB(28);"PNT";TAB(35);
3585 IF FLAG=4 GOTO 3595
3590 PRINT V$;
3595 PRINT:PRINT
3600 FOR I=0 TO J-1
3605 PRINT USING "##";I+1;
3610 PRINT TAB(4);T$(I,0);
3615 PRINT TAB(22);T$(I,1);
3620 PRINT TAB(26);:PRINT USING "###.###
";T(I);
3625 IF FLAG=4 THEN PRINT:PRINT:GOTO 365
0
3630 IF FLAG=1 THEN W=10*(T(I)-T(0))
3635 IF FLAG=2 THEN W=20*(T(I)-T(0))
3640 IF FLAG=3 THEN W=3*(T(I)-T(0))
3645 PRINT TAB(34);:PRINT USING "###.##"
;W
3650 NEXT I
3655 IF INKEY$<>CHR$(13) GOTO 3655
3660 RETURN
4000 REM * GENEREREN VAN BESTAND *
4005 CLS
4010 PRINT "GENEREREN VAN BESTAND"
```

```
4015 PRINT
4020 INPUT "Bent U er zeker van (J/N)";J
$
4025 IF J$="N" OR J$="n" GOTO 4125
4030 X$="X":I=2
4035 CLS
4040 PRINT "U kunt de namen van de"
4045 PRINT "deelnemers nu geven.":PRINT
4050 INPUT "Naam";N$:LOCATE 0,4
4055 INPUT "Land";L$
4060 LSET A1$=N$:LSET A2$=L$
4065 LSET B2$=X$:LSET C2$=X$:LSET D2$=X$
:LSET E2$=X$:LSET F2$=X$
4070 PUT #1,I
4075 I=I+1
4080 PRINT
4085 INPUT "Volgende naam (J/N)";J$
4090 IF J$="N" OR J$="n" GOTO 4120
4095 FOR J=3 TO 7
4100 LOCATE 0,J:PRINT SPC(40)
4105 NEXT J
4110 LOCATE 0,3
4115 GOTO 4050
4120 LSET A1$=MKI$(I):PUT #1,1
4125 RETURN
```

Alfabetische trefwoordenlijst

De volgende trefwoordenlijst is een gecombineerde lijst van alle trefwoorden uit de drie delen van de leerboekenserie. Ieder trefwoord wordt gevolgd door een of meer paginanummers. Die paginanummers bestaan steeds uit twee getallen, van elkaar gescheiden door een koppelteken. Het cijfer voor het koppelteken (1, 2 of 3) geeft aan uit welk deel van de leerboekenserie dit trefwoord komt. Het getal achter het koppelteken is het nummer van de pagina, waarop het trefwoord voor komt.

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 DOS
deel 3

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 519 3