MSX 2

DISK-DOS handboek

DISK DOS

A.C.J. Groeneveld

Uitbreidings handboek

MSX 2
DISK-DOS UITBREIDINGSHANDBOEK

MSX 2

DISK *DOS*

Uitbreidings handboek

A.C.J. Groeneveld

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

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

MSX

MSX 2. – Oosterend : Stark-Texel
Uitbreidingshandboek / A.C.J. Groeneveld.
ISBN 90-6398-222-4
SISO 365.3. UDC 681.3
Trefw.: MSX 2 (computer).

---

maart 1986,
ISBN 90 6398 222 4

INHOUD pag.

1 INLEIDING

In het grote MSX-2-handboek werd de MSX-computer en het MSX-basic uitgebreid behandeld, zowel voor MSX-1 als MSX-2-bezitters. De bezitter van een eenvoudige MSX-computer vindt in dit grote handboek dan ook alle geheimen van de MSX-computer uitgebreid beschreven.

Dit MSX DISK en DOS handboek is bedoeld als een uitbreiding op het eerste grote handboek en is geschreven voor de bezitter van een MSX-computer met een schijveneenheid te zamen met het grote MSX-handboek of eventueel de verkorte versie van dit handboek die door veel importeurs standaard bij de computer wordt geleverd. Ook dit deel is weer voor zowel MSX-1 als MSX-2 gebruikers geschikt.

In dit handboek wordt met name ingegaan op het MSX-basic in verband met de schijveneenheid. De beginnende programmeur dient dan ook eerst een redelijke ervaring te hebben opgebouwd met behulp van het grote handboek alvorens dit handboek ter hand te nemen. Daarbij is het verstandig om het grote handboek altijd klaar te hebben liggen om nog eens het één en ander te kunnen nazoeken.

Te zamen met het grote MSX-handboek vormt dit MSX-disk-handboek een zeer duidelijke en complete handleiding voor uw MSX-computer met schijveneenheid.

Indien u nog niet in het bezit bent van het 'grote' handboek, dan kunt u dit alles onthullende, meer dan 500 pagina's dikke nederlandstalige handboek bestellen bij uw computerboekhandel. Uw boekhandelaar heeft aan het ISBN-nummer 90 6398 221 6 genoeg om het voor u te kunnen bestellen. Eventueel kunt u dit handboek direkt bij de uitgever bestellen (telefoon 02223-661).

Ik hoop dat dit MSX DISK handboek het grote handboek in haar enorme sukses mag volgen.

maart 1986,
A.C.J. Groeneveld.

In het MSX-handboek werd reeds de geheugenopbouw van een MSX-computer behandeld. We zagen ondermeer dat we het geheugen van de MSX-computer kunnen indelen in vier geheugenbanken. In bank 0 en 1 zetelt het MSX basis ROM (read only memory, niet uitwisbaar geheugen). In bank 2 en 3 bevindt zich dan maximaal 32 kilobyte RAM (random access memory, vrij toegankelijk geheugen) waarin de MSX-computergebruiker bijvoorbeeld zijn programma kan koderen.

Het MSX-disk-basic brengt in de struktuur van de geheugenindeling enkele wijzigingen:

- allereerst resulteert het aansluiten van de disk-unit in een uitbreiding van het ROM-geheugen. 'Achter' geheugenbank 1 wordt een extra stuk ROM-geheugen geplaatst dat aktief wordt bij de benadering van de disk.

- Ten tweede wordt een gedeelte van het RAM-geheugen 'afgesnoept' voor de disk. MSX-basic heeft bij het gebruik van een disk wat extra geheugen nodig voor het opslaan van tussenresultaten (bufferruimte).

Wanneer de disk verder niet wordt gebruikt, is de aanwezigheid van het disk-basic in plaats van het gewone basic alleen te merken aan de volgende verschijnselen:

- het beschikbare geheugen is wat kleiner. Sommige veel ruimte vragende programma's kunnen hierdoor een foutmelding geven in het MSX-disk-basic.

- Indien bij de bevelen:
  BLOAD
  BSAVE
  LOAD
  MERGE
  OPEN
  RUN
  SAVE

  geen naam van het randapparaat wordt gegeven, werd bij het MSX-basic automatisch aangenomen dat de cassetterecorder als randapparaat werd bedoeld. In het MSX-disk-basic wordt plotseling automatisch aangenomen dat de disk unit als randapparaat wordt bedoeld.

  Het bevel SAVE "PROG" wordt in MSX-basic bijvoorbeeld auto-

matisch uitgevoerd op de cassetterecorder. In MSX-disk-basic wordt dit bevel echter automatisch op de disk uitgevoerd.

- Sommige fabrikanten leveren op floppy een kompleet disk operating system mee (MSX-DOS of CP/M). In dat geval dient het MSX-computersysteem te beschikken over 64 kilobyte RAM-geheugen. Het MSX-DOS operating system wordt in dit boek nader aan de orde gesteld.

- Onder MSX-1 is het MSX-disk-basic voorzien van een gewijzigde startmethode. Voorafgaand aan het normale gebruik dient eerst een datum en soms ook een tijd te worden ingegeven.

In het MSX-handboek werd reeds de opbouw van een MSX-computer behandeld. Eén van de onderdelen van de MSX-computer kan bestaan uit een (floppy) disk eenheid.

De disk eenheid wordt niet tot de standaard MSX-configuratie gerekend en werd als zodanig niet in het algemene handboek behandeld.

3.1 Permanente gegevensopslag

Het is bij een computer onontbeerlijk dat gegevens voor langere tijd kunnen worden bewaard. Een groot en ingewikkeld programma willen we niet elke avond opnieuw intikken, maar slechts éénmaal invoeren en vervolgens vastleggen. Standaard kan dit vastleggen van gegevens in MSX-basic gebeuren op een cassetteband met behulp van een cassetterecorder.

Elke MSX-programmeur stuit al snel tegen twee grote bezwaren van deze opslagmethode, namelijk:

- De opslag duurt vrij lang. 32 kilobytes (32768 tekens) op cassetteband vastleggen duurt ongeveer vijf minuten. Dit lijkt in eerste instantie snel, maar blijkt al vlug een groot bezwaar te zijn.

- De opslag is niet erg betrouwbaar. Vooral wanneer de cassetterecorder niet in topconditie is of verkeerd bandmateriaal wordt gebruikt, blijken gegevens tijdens opslag te kunnen worden verminkt. Het is dan ook noodzakelijk om uitstekend bandmateriaal aan te schaffen, de cassetterecorder regelmatig schoon te maken en af te stellen en de vastgelegde gegevens altijd te controleren.

Ondanks deze grote bezwaren is de cassetteband het meest voor de hand liggende opslagmedium voor hobby-computers; een cassetterecorder is relatief erg goedkoop, terwijl ook het bandmateriaal relatief zeer goedkoop is.

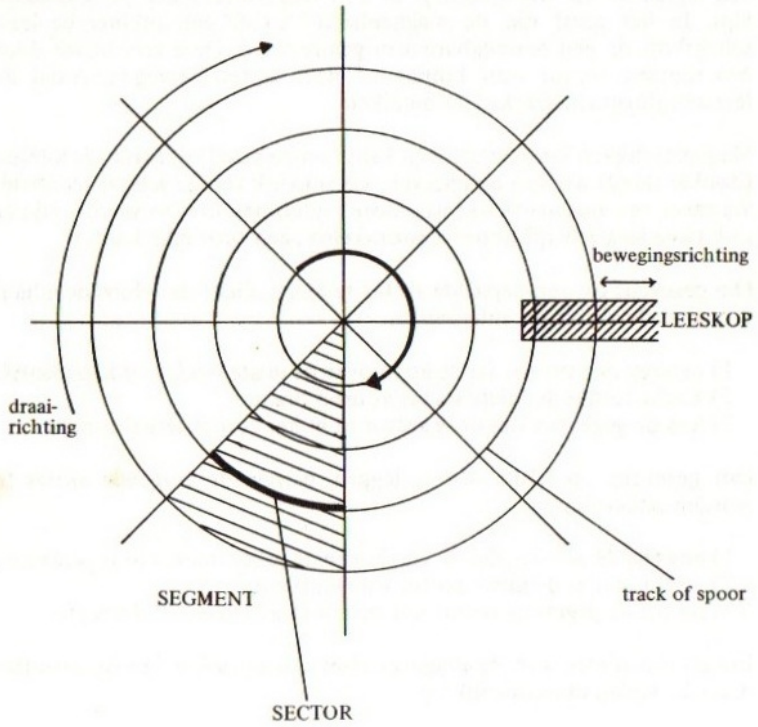
Wanneer snelheid bij het vastleggen en ophalen van gegevens een rol gaat spelen, wordt de cassetterecorder al snel een onbruikbaar randapparaat. Er dient dan naar een sneller opslagmedium te worden uitgekeken: de magneetschijf.

MSX-basic ondersteunt in een uitgebreide versie standaard het gebruik van deze magneetschijven.

In de volgende paragrafen wordt eerst het algemene principe van de magneetschijf uitgelegd. Daarna gaan we in op de logische indeling van een magneetschijf. Daarna, in hoofdstuk 4, wordt het MSX-basic met betrekking tot de magneetschijf behandeld.

3.2 De magneetschijf

De magneetschijfeenheid (disk unit) gaat uit van het principe dat gegevens worden gelezen van en geschreven naar een snel ronddraaiende schijf. Deze kunststof schijf is bedekt met een magnetisch geprepareerde laag, die vergelijkbaar is met de laag magnetisch materiaal die op een cassetteband is aangebracht.



We kunnen een magneetschijf ingedeeld zien in een aantal 'taartpunten' die we segmenten noemen. Daarbij kunnen we ons allemaal om elkaar heen liggende cirkels op het schijfoppervlak voorstellen, de sporen of tracks. De doorsnede van een segment met een track noemt men een sector. We kunnen een magneetschijf dus onderverdeeld zien in een groot aantal sectoren, verdeeld over de verschillende tracks op het schijfoppervlak.

Een sector is de eenheid van uitwisseling met een schijf. Op een sector kan een blok gegevens, meestal ter grootte van 512 of 1024 bytes, in één keer worden geschreven. Ook wordt een blok altijd in één keer ingelezen.

Om gegevens uit een bepaalde sector te kunnen lezen of om een bepaalde sector te kunnen (her) schrijven, is een lees-schrijfkop nodig, net zoals een opname- en weergavekop in een cassetterecorder noodzakelijk zijn. In het geval van de magneetschijf zit de gekombineerde lees-schrijfkop op een beweegbare arm gemonteerd. Deze arm wordt door een speciale motor naar binnen of naar buiten bewogen, zodat de leesschrijfkop alle tracks kan bereiken.

Magneetschijven kunnen aan één kant (single sided) of aan twee kanten (double sided) worden beschreven, afhankelijk van de schijveneenheid. Wanneer een magneetschijf aan twee zijden beschreven wordt, zijn er ook twee lees/schrijfkoppen noodzakelijk; één voor elke kant.

Om gegevens uit een bepaalde sector te lezen, dient de schijveneenheid de volgende akties te ondernemen:

1) beweeg de arm zo, dat de kop boven de juiste track wordt geplaatst.
2) wacht totdat de juiste sector voorbij draait.
3) lees de gegevens van deze sector in in het computergeheugen.

Om gegevens op schijf vast te leggen, dienen de volgende akties te worden ondernomen:

1) beweeg de arm zo, dat de kop boven de juiste track wordt geplaatst.
2) wacht totdat de juiste sector voorbijdraait.
3) schrijf de gegevens vanuit het computergeheugen in die sector.

Enkele voordelen van de magneetschijf ten opzichte van de cassette-recorder vallen onmiddellijk op:

– Er behoeft geen cassetterecorder meer op opnemen of afspelen te worden gezet; de schijveneenheid zorgt zelf voor in- en uitschakeling voor 'opnemen' en 'afspelen' van gegevens.

– Alle sectoren zijn in onderdelen van seconden te bereiken; lange door-of terugspoeltijden komen hierdoor te vervallen.

– De schijveneenheid is speciaal voor gegevensopslag ontworpen. In tegenstelling tot een gewone cassetterecorder behoeft de schijveneenheid niet geschikt te zijn om muziek te registreren. Hierdoor kan de kwaliteit van een disk unit worden toegespitst op de opslag van gegevens hetgeen een sneller en veel betrouwbaarder opslaan van gegevens mogelijk maakt.

3.3 Een snelheidsvergelijking

Voor diegene die hierin is geïnteresseerd, volgt hieronder een globale snelheidsvergelijking tussen een disk unit en een cassetterecorder.

DE CASSETTERECORDER

Om 16 kilobytes op cassetteband vast te leggen, is ongeveer de volgende tijd benodigd:

De hoogste schrijfsnelheid (onbetrouwbaar) op cassette is 2400 baud hetgeen overeenkomt met ongeveer 250 bytes per seconde. Om 16 kilobytes aan gegevens op band op te slaan, is dus een tijd nodig van ongeveer:

$$\frac{16 \times 1024}{250} = 66 \text{ seconden (ruim 1 minuut)}$$

Indien in een sector die 512 bytes kan bevatten, er 9 sectors per track zijn en de rotatiesnelheid van de disk ongeveer 5 maal per seconde is (allemaal redelijke aannamen voor een eenvoudige floppy disk eenheid) dan duurt het schrijven van 16 kilobytes aan gegevens ongeveer:

$$\frac{16 \times 1024}{512 \times 9 \times 5} = \text{nog geen seconde...}$$

Beide tijdsberekningen zijn slechts benaderingen; sommige invloeden zijn volledig verwaarloosd. Zonder veel gevaar kan men echter stellen dat de eenvoudigste schijveneenheid al snel HONDERD MAAL ZO SNEL is als een cassetterecorder...

3.4 Vormen van magneetschijven

De magneetschijfeenheid of disk unit komt in diverse verschillende vormen voor. Twee vormen van magneetschijfeenheden behandelen we hieronder:

– de 3½ inch floppy disk unit

  Deze schijveneenheid maakt gebruik van 3,5 inch floppy disks. Deze floppy disks hebben een doorsnede van 3,5 inch (ongeveer 9 cm) en zijn verpakt in een hard plastic omhulsel. Het leesvenster is afgedekt met een klepje en schijft bij het insteken van de floppy in de schijveneeheid automatishc open, waarna de lees/schrijfkop toegang heeft tot de daadwerkelijke schijf.
  De schijf zelf is gemaakt van een flexibele kunststof met daarop een magnetisch geprepareerde laag.
  Over het algemeen kent deze schijf-vorm een redelijk hoge opslag; per floppy disk kan meestal een hoeveelheid van 250.000 tot 1.000.000 tekens worden opgeslagen...

– de 5¼ inch floppy disk unit

  Deze schijveneenheid maakt gebruik van grotere floppy disks (5½ inch = ongeveer 13 cm doorsnede). Deze floppy disks hebben een open leesvenster en moeten daarom altijd in een speciale envelop worden bewaard. De verdere specificaties zijn ongeveer gelijk aan die van de 3½ inch floppy disk.

3.5 Write protect

Bij elke vorm van magneetschijven is de mogelijkheid aanwezig om de gehele schijf te beschermen tegen abusievelijk overschrijven van gegevens. Deze mogelijkheid noemt men meestal de WRITE PROTECT optie.

De 3½ diskette bezit hiertoe in één van de hoeken een klein schuifje.

Door dit schuifje zo te zetten dat een gaatje ontstaat, wordt het schijfje beschermd tegen het per ongeluk overschrijven van gegevens. Nadat het schuifje op beschreven wijze in de write protect stand is geschoven, is het voor de computer onmogelijk om nog gegevens op de schijf te schrijven. Het blijft echter mogelijk om gegevens van deze schijf te LEZEN.

De 5¼ inch floppy disk heeft geen schuifje. In plaats hiervan dient een plakkertje te worden geplakt over de kleine insparing die zich aan de kant van het leesvenster bevindt. Nadat dit plakkertje is aangebracht, is het voor de computer niet meer mogelijk om nog op de schijf te schrijven. LEZEN gaat natuurlijk nog wel.

Bij optredende fouten bij het schrijven geldt altijd dat u eerst dient te controleren of de schijf niet via de WRITE PROTECT optie is beschermd.

Wanneer dit niet het geval blijkt te zijn, onderzoek dan of de schijf wel geformatteerd is (zie de volgende paragraaf).

3.6 Formatteren

We zagen reeds dat een magneetschijf is onderverdeeld in zogenaamde sectoren. Elke sector staat op een vaste plaats op de magneetschijf. In elke sector kan een blok gegevens (meestal 512 of 1024 bytes) worden geschreven.

Voordat een magneetschijf in gebruik wordt genomen, dient de onderverdeling in sectoren eerst te worden aangebracht op de schijf. De magneetschijf moet als het ware voor het gebruik eerst in stukjes worden verdeeld.

Het verdelen van de magneetschijf in partjes noemt men het FORMATTEREN van de magneetschijf. Altijd geldt:

EEN SCHIJF MOET EERST WORDEN GEFORMATTEERD, EERDER KAN HIJ NIET WORDEN GEBRUIKT.

Daar de vorm van de magneetschijveneenheid door MSX niet wordt vastgelegd, is er ook geen vast omschreven manier van formatteren van een magneetschijf. Het formatteren van een schijf is als kommando niet in de MSX-standaard opgenomen.

Hoe een magneetschijf moet worden geformatteerd, dient in de gebruiksaanwijzing van de betreffende schijveneenheid te worden opgezocht. Normaal gebeurt dit door middel van een ingave als:

```
CALL FORMAT
```

Hierna vraagt de computer dan welke schijf dient te worden geformatteerd. Geef de letter A in indien u maar één schijveneenheid heeft. Heeft u twee schijveneenheden, geef dan een A in voor de eerste of een B in voor de tweede schijveneenheid. Bij meerdere schijveneenheden kan C,D... worden ingegeven.

Meestal duurt het formatteren één tot twee minuten. Na afloop van het formatteren kan het zijn dat de computer een foutmelding geeft. Probeer in dat geval een tweede formattering. Blijft het formatteren fout gaan, dan is de schijf kapot of is de schijveneenheid vervuild of kapot.

PAS OP: WANNEER U EEN VOORBEELDSCHIJFJE MET EEN OPERATING SYSTEM MEEGELEVERD HEBT GEKREGEN, FORMATTEERT U DEZE NATUURLIJK NIET. MET HET FORMATTEREN VERWIJDERT U NAMELIJK ALLE GEGEVENS VAN DE SCHIJF.

Ook indien u een gehele schijf in één klap leeg wilt hebben, kunt u dit door het formatteren bewerkstelligen.

Voordat u verder leest is het zaak dat u, wanneer u tenminste de voorbeelden uit wilt proberen, weet hoe u moet formatteren. Houd één geformatteerde schijf apart voor het oefenen met de voorbeelden.

N.B.: Sommige MSX-computers kunnen na het FORMAT-kommando nog wat aanvallende informatie verstrekken of vragen stellen, afhankelijk van het gebruikte type schijveneenheid.

3.7 De logische indeling van een magneetschijf

In de vorige paragrafen stelden we ondermeer vast dat we de magneetschijf uiteindelijk kunnen zien ingedeeld in een aantal sectoren. Elke sector bevat een blok van gegevens ter grootte van 512 of 1024 bytes. Zo'n blok is de kleinste eenheid die in zijn geheel kan worden gelezen of geschreven.

De indeling van de schijf in sporen, segmenten en sectoren (blokken) noemen we de fysische indeling van de schijf. Deze indeling kan als het ware op de schijf zelf worden aangewezen.

Met de logische indeling van de schijf bedoelen we de indeling die het MSX-basic hanteert, gebruik makende van de fysische indeling.

Doordat MSX-basic een intelligente en doordachte logische indeling van de magneetschijf ondersteunt, behoeft de gebruiker van de MSX-computer nooit stil te staan bij de fysische indeling van de magneetschijf. MSX bepaalt de fysische indeling, zorgt dat een blok niet ten onrechte door een ander kan worden overschreven en houdt bij in welke blokken welke gegevens zijn geschreven en welke blokken bij elkaar horen. Nooit behoeft de gebruiker dus een lijstje bij te houden van welke sectoren er werden gebruikt voor opslag van een programma. Door alleen maar de naam van het programma aan het MSX-systeem te verschaffen, kan worden bewerkstelligd dat de sectoren waarin het programma is opgenomen in de juiste volgorde worden ingelezen in het computergeheugen.

MSX-basic zorgt ervoor dat op elke disk een *inhoudsopgave* (index) wordt bijgehouden. In deze inhoudsopgave staat vermeld:

- wat de naam is van de verzameling van gegevens die op schijf werd geschreven.
- wat de soort is van deze verzameling van gegevens. Zijn het programmaregels of zijn het andere gegevens?
- welke sectoren worden bezet door deze gegevens. Totdat deze gegevens worden verwijderd, mogen de sectoren, waarin deze gegevens staan, niet voor andere doeleinden worden gebruikt. MSX-basic waakt hierover.

Een bij elkaar behorende verzameling van gegevens noemt men in de computerwereld over het algemeen een BESTAND of een FILE. De MSX-schijveneenheid is een bestandgeoriënteerd medium. Voordat gegevens op floppy worden geschreven, dient eerst een bestand te zijn gedefinieerd op schijf. Hierop gaan we bij de behandeling van de kommando's verder in.

MSX verzorgt de besturing van de bestanden (het file management) waardoor we als MSX-programmeur niet met de fysieke opbouw van de schijveneenheid worden gekonfronteerd.

3.8 FILES

Zoals in de vorige paragraaf werd opgemerkt, gaat MSX-basic uit van een bestandsstruktuur op schijf. De term FILE of BESTAND zal steeds weer terugkeren en is zeer belangrijk. Onthoudt:

EEN BESTAND OF EEN FILE IS EEN GROEP VAN BIJ ELKAAR BEHORENDE GEGEVENS.

We onderscheiden vooorlopig twee verschillende soorten FILES, namelijk:

1. de PROGRAM FILE (programmabestand). Zo'n programmabestand bestaat uit allemaal bij elkaar behorende programmaregels die te zamen een programma vormen. Een program file wordt met behulp van het SAVE-kommando op de schijf geschreven en met behulp van bijvoorbeeld het LOAD-kommando weer in zijn geheel van schijf opgehaald. Met een KILL-kommando kan een program file definitief van de schijf worden gewist.

2. de DATA FILE (gegevensbestand). Een data file bestaat uit allemaal bij elkaar behorende gegevens. In een data file kunnen we bijvoorbeeld de namen adressen en telefoonnummers van onze kennissen opslaan. Ook kunnen we een ander bestand ontwerpen bijvoorbeeld de namen, adressen en telefoonnummers van onze kennissen opslaan. Ook kunnen we een ander bestand ontwerpen met daarin de gegevens van onze postzegelverzameling, platenverzameling etcetera. Al deze bestanden mogen eventueel op één enkele disk staan; MSX-basic zorgt ervoor dat deze gegevens uit elkaar worden gehouden.

3.9 PROGRAM FILES

De PROGRAM FILE bestaat uit een verzameling van bij ekaar behorende programmaregels die tezamen een programma vormen. Laten we als voorbeeld eens het volgende programma nemen: (zorg dat er een geformatteerde schijf in de (eerste) schijveneenheid zit)

```
NEW
Ok
10 PRINT "MSX-BASIC *** ";
20 GOTO 10
SAVE "PROG
```

```
Ok
FILES
EDIT          KLARA          PROG
Ok
LOAD "PROG"
Ok
KILL "PROG"
Ok
FILES
EDIT          KLARA
Ok
```

In dit voorbeeld werd eerst een NEW-kommando gegeven waarna een programma van twee regels werd ingetikt. Hierna werd het programma met behulp van een SAVE-kommando op schijf gezet. Vervolgens werd het kommando FILES ingetikt; de computer geeft aan welke files nu op de schijf staan. Behalve de file PROG (het zojuist op schijf gezette programma) bleken er in dit voorbeeld nog twee files op de schijf te staan.
Vervolgens werd het zojuist op schijf gezette programma weer in het computergeheugen ingelezen met behulp van het LOAD-kommando. Met het KILL-kommando werd vervolgens het programma PROG weer van schijf verwijderd. Het laatste FILES-kommando laat zien dat de program file PROG daadwerkelijk van schijf is verdwenen.

Bovenstaand voorbeeld vertoont veel gelijkenis met de wijze waarop op cassetteband een programma kan worden opgeslagen. Over het algemeen wordt een program file met de sleutelwoorden SAVE, LOAD, KILL, MERGE en RUN benaderd.
Op deze sleutelwoorden gaan we in hoofstuk 4 verder in.

3.10 DATA FILES

Data files of gegevensbestanden kunnen we weer in twee verschillende groepen onderverdelen:

1 de SEQUENTIAL FILE (sequentieel=in volgorde)

2 de RANDOM FILE (random=willekeurig)

De sequential file kan alleen in opklimmende volgorde worden benaderd. Gegevens die in een bepaalde volgorde op schijf werden gezet, kunnen alleen ook weer in deze volgorde worden ingelezen.

De random file geeft de schijveneenheid zijn grote kracht. De random file staat toe dat gegevens in elke volgorde kan worden benaderd. Wanneer we bijvoorbeeld onze kennissen nummeren van 1 tot 100 en deze vervolgens op schijf zetten, is het bij de random file niet noodzakelijk dat eerst de gegevens van kennis 1 tot en met 99 worden ingelezen om uiteindelijk de gegevens van kennis 100 te weten te komen. De gegevens van kennis nummer 100 kunnen onmiddellijk in het computergeheugen worden ingelezen.

3.11 SEQUENTIAL FILES

De sequential file geeft de mogelijkheid om gegevens achtereenvolgend op schijf te zetten. Nadat deze gegevens op schijf zijn gezet, kunnen ze ook weer in het computergeheugen worden ingelezen, echter alleen in die volgorde waarin ze ook geschreven zijn.

Gegevensbestanden op cassetteband kunnen eveneens worden beschouwd als sequentiele bestanden. Ook van cassetteband kunnen gegevens alleen in die volgorde worden gelezen waarin ze ook geschreven zijn. Het voordeel van een sequential file op schijf is gelegen in snelheid en betrouwbaarheid. Gegevens worden vaak 100 maal zo snel van magneetschijf ingelezen, terwijl de integriteit (betrouwbaarheid) van deze gegevens bijzonder groot is.

Een voorbeeld:

```
NEW
Ok
10 OPEN "TEST" FOR OUTPUT AS #1
20 LINE INPUT R$
30 IF R$="" THEN 50
40 PRINT #1,R$:GOTO 20
50 CLOSE:OPEN "TEST" FOR INPUT AS #1
60 IF EOF(1)=-1 THEN CLOSE:STOP
70 INPUT #1,R$:PRINT R$:GOTO 60
RUN
DEZE TEKST WORDT IN EEN SEQUENTIAL
FILE GESCHREVEN EN ZO METEEN WEER
AFGEDRUKT.

DEZE TEKST WORDT IN EEN SEQUENTIAL
FILE GESCHREVEN EN ZO METEEN WEER
AFGEDRUKT.
Break in 60
Ok
```

Op regel 10 wordt een file aangemaakt, genaamd TEST. FOR OUTPUT wil zeggen dat gegevens naar deze file *geschreven* dienen te worden. Op regel 20 kan een tekst worden ingegeven. Wanneer daadwerkelijk een tekst werd ingegeven, volgt op regel 40 het kommando om deze tekst in de file te schrijven. Vervolgens kan dan een nieuwe regel tekst (GOTO 20) worden ingegeven.
Wordt op regel 30 gekonstateerd dat er geen tekst meer werd ingevoerd, dan wordt op regel 50 de file eerst gesloten en dan weer geopend, dit maal FOR INPUT. FOR INPUT wil zeggen dat gegevens vanuit de file dienen te worden *ingelezen.* Wanneer EOF(1) ongelijk is aan −1, dus wanneer nog geen einde bestand werd gekonstateerd, wordt op regel 70 een volgende regel aan gegevens ingelezen en geprojekteerd op het beeldscherm. Indien op regel 60 een einde bestand werd gekonstateerd, wordt het bestand gesloten en is het programma ten einde.

Merk op dat het programma in het geheel geen nieuwe kommando's bevat. Wanneer geen schijveneenheid zou zijn aangesloten, zou hetzelfde programma op band zijn werk doen. Het is dan echter wel noodzakelijk om de cassette-recorder eerst op opnemen, vervolgens op terugspoelen en uiteindelijk op afspelen te zetten.

Het is erg leerzaam om dit programma een keer op cassette en een keer op magneetschijf uit te proberen; het snelheidsverschil is verbazend...

Het bovenstaande voorbeeld bestuurt een sequentieel bestand. De gegevens die eerst werden geschreven, werden daarna noodzakelijkerwijs in dezelfde volgorde weer ingelezen.

3.12 RANDOM FILES

Om het hoe en waarom van random files goed te begrijpen, is een nadere uitleg noodzakelijk.

We zagen reeds dat we op een magneetschijf meerdere files kunnen opslaan. In feite kunnen we de magneetschijf opgedeeld zien in FILES. Zo'n file wordt gevormd door een groep van bij elkaar behorende gegevens.

Een file kunnen we weer opgedeeld zien in kleinere stukken; de RECORDS.

EEN RECORD IS EEN EENHEID VAN GEGEVENS BINNEN EEN FILE. MEESTAL BEVAT EEN RECORD DE GEGEVENS VAN ééN OBJEKT, PERSOON OF VERSCHIJNSEL BINNEN DE FILE WAARIN DE GEGEVENS VAN MEERDERE OBJEKTEN, PERSONEN OF VERSCHIJNSELEN ZIJN OPGENOMEN.

Een voorbeeld: we leggen een bestand aan met daarin verschillende gegevens van onze vrienden en kennissen. We spreken af dat elke kennis een vast nummer van 1 tot 100 krijgt.

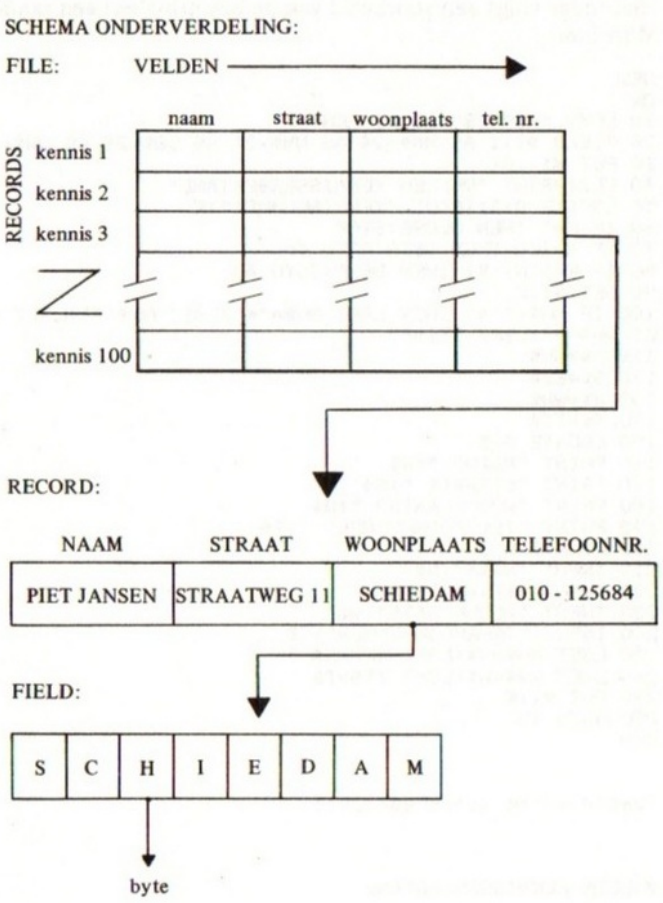
Per kennis willen we vastleggen:

1. Zijn/haar nummer
2. Zijn/haar naam
3. Zijn/haar straat
4. Zijn/haar woonplaats
5. Zijn/haar telefoonnummer

Het bestand, dat we KENNIS zullen noemen, bevat dus de gegevens van al onze vrienden en kennissen. Eén record uit het bestand bevat het nummer, de naam, de straat, de woonplaats en het telefoonnummer van één kennis. Het bestand bestaat dus uit diverse records terwijl elk record bij precies één kennis hoort.

Een record kunnen we weer verdeeld zien in verschillende FIELDS of velden. In één veld wordt een volledige gegevenseenheid opgeslagen. Zo vormt de naam of de woonplaats één van de velden binnen het record van een kennis.

Ter verduidelijking eerst een schema:



In het bovenstaande schema zien we ons kennissenbestand weer terug. We zien dat we per kennis één record hebben gereserveerd. Per record kennen we weer enkele velden. In het voorbeeld lieten we eerst het record van kennis nummer 3 zien (Piet Jansen). Vervolgens beschouwden we het veld van de woonplaats binnen dit record (Schiedam).

Uiteindelijk bestaat een veld natuurlijk uit bytes; de kleinste eenheid van gegevensopslag waarin één enkel letterteken kan worden gekodeerd.

Hieronder volgt een voorbeeld van de besturing van een random file in MSX-basic.

```
NEW
Ok
10 OPEN "KENNIS" AS #1 LEN=97
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$,24 AS WW$,24 AS TT$
30 PUT #1,101
40 CLS:PRINT "VULLEN KENNISSENBESTAND"
50 LOCATE 0,3:INPUT "KENNISNUMMER";K
60 IF K=0 THEN CLOSE:STOP
70 IF K>100 THEN BEEP:GOTO 40
80 IF K<>INT(K) THEN BEEP:GOTO 40
90 GET #1,K
100 IF MM$<>"*" THEN LSET MM$="*":LSET NN$="":LSET SS$="":LS
ET WW$="":LSET TT$=""
110 N$=NN$
120 S$=SS$
130 W$=WW$
140 T$=TT$
150 LOCATE 0,5
160 PRINT "NAAM? ";N$
170 PRINT "STRAAT? ";S$
180 PRINT "WOONPLAATS? ";W$
190 PRINT "TELEFOONNUMMER? ";T$
200 LOCATE 0,5
210 INPUT "NAAM";N$
220 INPUT "STRAAT";S$
230 INPUT "WOONPLAATS";W$
240 INPUT "TELEFOONNUMMER";T$
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
270 PUT #1,K
280 GOTO 40
RUN

(beeld wordt schoongemaakt)


VULLEN KENNISSENBESTAND


KENNISNUMMER? 3

NAAM? PIET JANSSENS
STRAAT? STRAATWEG 111
WOONPLAATS? SCHIEDAM
TELEFOONNUMMER? 010-123456
```

Met dit programma kunnen honderd verschillende kennissen op de schijf worden opgeslagen. Indien een kennis reeds is ingebracht en zijn of haar nummer een tweede maal wordt ingegeven, verschijnen de gegevens weer op het beeldscherm. Deze gegevens kunnen eventueel worden verbeterd of met alleen een return onveranderd worden opgenomen.

Merk op dat de kennissen door elkaar heen kunnen worden aangemaakt en ook weer door elkaar heen worden opgevraagd. Er is geen verplichting tot het handhaven van enige volgorde.

Het bovenstaande programma behandelen we hier globaal; voor de preciese betekenis van de kommando's dient hoofdstuk 4 te worden geraadpleegd.

Op regel 10 wordt het kennissenbestand geopend en eventueel aangemaakt als het nog niet eerder bestond. Ook werd bepaald dat een record in dit bestand een lengte heeft van 97 bytes (tekens). Vervolgens wordt op regel 20 met het FIELD-kommando de indeling van het record bepaald. Eén karakter, MM$, wordt gereserveerd als merkteken. Bij een gevuld record wordt deze MM$ op een sterretje gezet als teken dat er goede informatie in het record aanwezig is. De velden NN$, SS$, WW$ en TT$ zijn elk 24 posities lang en zijn bedoeld voor de opslag van naam, straat, woonplaats en telefoonnummer.

Op regel 30 wordt vervolgens record nummer 101 beschreven met behulp van het PUT-kommando. Omdat NN$, SS$, WW$ en TT$ nog niet werden gevuld, wordt een leeg record geschreven op positie nummer 101 in het kennissenbestand. Dit record wordt bij het begin geschreven om te voorkomen dat later een INPUT PAST END foutmelding verschijnt. Deze fout kan natuurlijk ook met de ON ERROR GOTO-konstruktie worden opgevangen. Echter, we houden het voorbeeld hier wat eenvoudig.

Op regel 40 wordt het beeldscherm schoongemaakt en een kopregel afgedrukt. Op regel 50 kan dan een kennisnummer worden ingegeven dat op regel 60, 70 en 80 wordt gekontroleerd op juistheid. Het programma stopt indien een kennisnummer nul werd ingegeven.

Op regel 90 wordt het record van de betreffende kennis ingelezen. Door deze GET-instruktie worden de op regel 20 genoemde variabelen (NN$, SS$, WW$, en TT$) automatisch gevuld met de gegevens van deze kennis. Ook wordt MM$ gevuld, het merkteken dat aangeeft of er zinnige informatie in het record aanwezig is. Indien MM$ ongelijk

is aan een sterretje, wordt dit veld alsnog gelijk gesteld aan een sterretje terwijl de andere velden uit het record worden schoongemaakt. Merk op dat de velden uit een record met een speciale instruktie (in dit geval LSET) dienen te worden gevuld. De velden uit een met behulp van het sleutelwoord FIELD gedefinieerd record mogen niet zomaar met een LET- of INPUT-kommando worden behandeld. Daarom worden deze velden vervolgens (110-140) ook overgenomen in de velden N$, S$, W$ en T$.

Op regel 150-190 worden de ingelezen gegevens op beeldscherm afgedrukt. Vervolgens kunnen op regel 200-240 de gegevens opnieuw worden ingegeven. Doordat op de plaatsen van de INPUT-kommando's al gegevens staan vermeld, worden deze automatisch als waarde voor de INPUT genomen wanneer alleen een RETURN-toets wordt ingegeven. Ook kunnen deze gegevens met behulp van de diverse besturingstoetsen worden gecorrigeerd.

Op regel 250 en 260 worden de recordvelden met het LSET-kommando gelijk gemaakt aan de ingegeven waarden waarna op regel 270 het record in het kennissenbestand op de plaats wordt gezet die door de variabele K (kennisnummer) wordt aangegeven.

Uiteindelijk (GOTO 40) kan weer een nieuw kennisnummer worden ingegeven.

In tegenstelling tot het vorige voorbeeld kwamen we in dit laatste voorbeeld wat nieuwe sleutelwoorden tegen. De belangrijkste sleutelwoorden die we in verband met random files gaan tegenkomen zijn:

FIELD	voor het definiëren van de opbouw van een record
LSET	voor het invullen van een veld
RSET	ook voor het invullen van een veld, maar nu rechts aangeschoven
GET	voor het lezen van een record
PUT	voor het schrijven van een record

Voor een uitgebreide behandeling van deze sleutelwoorden dient hoofdstuk 4 te worden geraadpleegd.

3.13 De grootte van files

Het zal de oplettende lezer zijn opgevallen dat in de voorbeelden nergens op enigerlei wijze de grootte van een bestand werd aangegeven. MSX kent een zogenaamde DYNAMISCHE behandeling van bestanden. Hierdoor behoeft een grootte van een bestand niet te worden opgegeven; waar nodig vult MSX een bestand steeds aan tot de juiste grootte. De enige grens die er aan de grootte van een bestand ligt, is de opslagcapaciteit van de magneetschijf. Een eenvoudige rekensom leert dat er op een floppy disk van bijvoorbeeld 250 kilobytes een aantal van ongeveer 2500 kennissen met hun namen, straten, woonplaatsen en telefoonnummers kunnen worden opgeslagen. De benadering van elke kennis afzonderlijk met behulp van het bovenstaande voorbeeldprogramma kost gemiddeld nog geen seconde...

4 DE MSX-DISK SLEUTELWOORDEN EN HUN BETEKENIS

In dit hoofdstuk worden de MSX-disk sleutelwoorden behandeld. Voorafgaand aan deze behandeling worden eerst wat sleutelwoorden behandeld die in het grote MSX-handboek reeds werden behandeld maar in MSX-disk basic een extra funktie krijgen. Daarna worden de nieuwe sleutelwoorden behandeld.

De schrijfwijzen worden in de BNF-notatie gepresenteerd. Deze notatievorm werd in het grote MSX-handboek uitgebreid behandeld.

Omdat het hier slechts om een beperkt aantal sleutelwoorden gaat, worden deze, in tegenstelling tot in het grote MSX-handboek, hier in aanbevolen leervolgorde gepresenteerd. Zij zijn in de inhoudsopgave op alfabetische volgorde te vinden.

Per nieuw sleutelwoord worden de volgende gegevens verstrekt:

- de naam van het sleutelwoord
- de moeilijkheidsgraad bij gebruik van dit sleutelwoord. Deze moeilijkheidsgraad varieert van zeer eenvoudig tot zeer moeilijk.
- de soort. We onderscheiden binnen de sleutelwoorden de A-FUNKTIES (funkties met een alfanumeriek resultaat), de N-FUNKTIES (funkties met een numeriek resultaat), SYSTEEMVARIABELEN (variabelen met een voorbestemde waarde) en KOMMANDO'S (sleutelwoorden die een aktie aangeven.
- de afkomst. Alle in MSX-basic voorkomende sleutelwoorden zijn afkomstig uit de Engelse taal. Het is vaak gemakkelijker om een sleutelwoord en de bijbehorende betekenis te onthouden wanneer men de preciese afkomst kent. Daarom is van elk sleutelwoord de afkomst naar het Nederlands herleid.
- de schrijfwijze in de BNF-notatievorm.
- de betekenis. De funktie van elk sleutelwoord wordt uitvoerig behandeld. Van sleutelwoorden die reeds in het grote MSX-handboek werden behandeld, is slechts een aanvullende beschrijving opgenomen.

– een voorbeeld. Waar zinvol is een voorbeeld opgenomen.

4.1 Enkele oude sleutelwoorden wat nader behandeld

OPEN

---

aanvullende betekenis

Het OPEN-kommando wordt gebruikt om een bestand, een groep van bij elkaar behorende gegevens, in te richten c.q. te kunnen gebruiken.

Het beeldscherm (CRT:), het grafische beeldscherm (GRP:), de printer (LPT:) of de cassetterecorder (CAS:) leerden we in het grote MSX-handboek reeds kennen. Als nieuwe randapparaten introduceert MSX-basic "A:...", "B:...", "C:...." enzovoorts als aanduidingen voor de eerste schijveneenheid, de tweede, de derde enzovoorts.

Als aanduiding op welke wijze het bestand dient te worden benaderd, zagen we reeds FOR OUTPUT en FOR INPUT. Als bijkomende aanduiding introduceert MSX-DISK-basic de aanduiding FOR APPEND waarmee we aangeven dat een bestand dient te worden AANGEVULD met gegevens. MSX-2 gebruikers kennen deze aanduiding al in verband met de RAM-DISK. Bovendien kan voor een random files de recordlengte worden aangegeven met behulp van het woord LEN.

De schrijfwijze dient er in BNF dus als volgt uit te zien:

```
                              INPUT
                              ------
                        FOR OUTPUT AS [#]<KANAAL>
                              ------
                              APPEND
OPEN<BESTANDSNAAM>  ------------------------------
                        AS [#]<KANAAL>[LEN=<N>]

<BESTANDSNAAM>::=<A>

<KANAAL>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Allereerst volgt hieronder een uitgebreide tabel met voorwaarden in verband met het OPEN-kommando:

randapparaat	betekenis	benadering	verplicht?	bestandsnaam	kommentaar
CAS:	cassette-recorder	for input for output	ja	nee*	een cassette-recorder moet zijn aangesloten
GRP:	grafisch scherm	for output	nee	nee, zinloos	een grafisch scherm moet zijn geaktiveerd
CRT:	alfanume-riek scherm	for output	nee	nee, zinloos	een alfanumeriek scherm moet zijn geaktiveerd
LPT:	printer	for output	nee	nee, zinloos	een printer moet zijn aangesloten
MEM:	ram-disk	for input for output for append	ja	nee***	alleen MSX-2
A:/B:/...	disk	for input for output for append	nee**	ja	een disk unit moet zijn aangesloten
(niets)	disk	..... zie A:/B:/...			

* indien geen bestandsnaam wordt opgegeven:
–bij FOR OUTPUT wordt een bestand zonder naam aangemaakt.
–bij FOR INPUT wordt het eerste bestand dat op de band wordt tegengekomen, geopend.

** indien geen benaderingswijze wordt gespecificeerd, wordt een RANDOM benaderingswijze aangenomen. Zie hiervoor de behandeling in hoofdstuk 3 en de sleutelwoorden vanaf FIELD.

*** indien geen bestandsnaam wordt opgegeven:

–FOR INPUT: een naamloos bestand wordt geopend. Indien dit FOR APPEND bestand niet aanwezig is, volgt een foutmelding.
–FOR OUTPUT: een naamloos bestand wordt aangemaakt.

Pas op dat een kommando als OPEN "FILE"...AS 1, dat in MSX-basic een bestand op cassetteband opende, nu plotseling een bestand op

schijf opent of creëert.

Met de LEN-optie kan de totale recordlengte van een random bestand worden bepaald. Indien deze optie achterwege gelaten wordt, wordt standaard een lengte van 256 bytes per record aangenomen.

Voor de disk unit volgen hieronder enkele voorbeelden, gekombineerd met de behandeling van de overige 'oude' sleutelwoorden.

MAXFILES

Met de schijveneenheid krijgt de systeemvariabele MAXFILES pas echt een betekenis. Doordat de schijf random benaderbaar is, is het mogelijk om meerdere bestanden tegelijk op één schijf te creëren en/of te openen. Zelfs lees- en schrijfopdrachten kunnen gekombineerd worden. Wanneer meerdere bestanden dienen te worden geopend, dan dient de systeemvariabele MAXFILES gelijk te worden gesteld aan het aantal tegelijk te openen bestanden. Een voorbeeld:

```
NEW
Ok
10 MAXFILES=2
20 OPEN "FILE1" FOR INPUT AS 1
30 OPEN "FILE2" FOR OUTPUT AS 2
40 REM VERDERE PROGRAMMA
```

Op regel 10 werd bepaald dat er maximaal twee bestanden tegelijk mogen worden geopend. Op regel 20 werd vervolgens het bestand FILE1 geopend. Indien het bestand niet aanwezig is op de disk, volgt een foutmelding. Op regel 30 werd het bestand "FILE2" geopend en meteen schoongemaakt (met FOR OUTPUT geeft men aan dat men NIEUWE gegevens in het bestand wil schrijven). Indien bestand FILE2 niet aanwezig was, werd het aangemaakt.
Op regel 40 vervolgt het programma tenslotte.

CLOSE

Voordat we dit sleutelwoord verder behandelen eerst een waarschu-

wing:

Een bestand dat nog niet is gesloten, is meestal niet helemaal bijgewerkt. Indien een disk wordt vervangen door een andere terwijl bestanden nog niet gesloten zijn, kan het gebeuren dat de eerste schijf niet volledig wordt bijgewerkt terwijl de tweede wordt 'opgeblazen' (de gegevens worden vaak volledig verminkt). Nadat een programma is onderbroken, kunnen bestanden toch nog openstaan; pas bij een poging om het programma te veranderen, kan het plotseling gebeuren dat de disk unit weer even gaat werken; de openstaande bestanden worden dan als nog gesloten.

Alvorens schijven te verwisselen, moet u zich aanwennen om eerst een close-kommando voor de zekerheid in te toetsen zonder regelnummer. Verwijder pas na de Ok-melding de schijf; er kan dan niets meer fout gaan.

Dus tik eerst in:

```
CLOSE
Ok
```

en verwijder dan pas de schijf.

Het CLOSE-kommando sluit een specifiek bestand; maakt het verder onbenaderbaar voor het programma. Alle kanalen kunnen in één keer worden gesloten door uitvoering van een enkele CLOSE. Door achter het sleutelwoord CLOSE een kanaalnummer te vermelden (eventueel voorafgegaan door een hekje) kan een specifiek bestand worden gesloten, namelijk het bestand dat eerder op het genoemde kanaalnummer werd geopend. Door meerdere kanaalnummers achter CLOSE op te nemen (gescheiden door een komma) kunnen selectief meerdere bestanden tegelijk worden gesloten.

SAVE

Met het SAVE-kommando kan een programma op een randapparaat worden veiliggesteld.

Het SAVE-kommando krijgt in het MSX-disk basic een iets andere

funktie dan in het MSX-basic:

MSX-basic	MSX-disk-basic
Indien in de programmanaam geen randapparaat wordt gespecificeerd, dan wordt de cassetterecorder als randapparaat aangenomen. De bevelen:	Indien in de programmanaam geen randapparaat wordt gekodeerd, dan wordt de laatste aktieve disk (meestal A:) als randapparaat genomen. De bevelen:
SAVE "PROG"	SAVE "PROG"
en	en
SAVE "CAS:PROG"	SAVE "A:PROG"
hebben dus dezelfde uitwerking.	hebben dus dezelfde uitwerking (wanneer de schijf A: tenminste het laatst aktief was).
Het programma wordt altijd teken voor teken volgens de ASCII-kodering geschreven.	Indien het programma naar disk wordt geschreven, dan gebeurt dat in de kodering waarmee het ook in het geheugen staat (verkort). Indien het programma NIET naar disk wordt geschreven, dan gebeurt dit via de ASCII-kodering.

Voor de bestandsnaam bij een SAVE-kommando gelden dezelfde eisen als genoemd bij het OPEN-kommando. Wanneer bij een SAVE naar schijf de ,A-optie wordt opgenomen, dan wordt het programma toch in ASCII-kodering geschreven. Dit is van belang wanneer het programma later met een MERGE bij een ander programma dient te worden gevoegd.

Dus:

```
SAVE"A:PROG",A
Ok
```

schrijft het programma PROG op schijf maar dan wel in ASCII-kodering; dezelfde kodering die met een SAVE op band wordt gebruikt.

Indien met behulp van SAVE een programma met de naam:

```
AUTOEXEC.BAS
```

op de schijf wordt geschreven, dan zal de computer mits de schijf met dit programma zich dan in de eerste schijveneenheid bevindt, de volgende keer dat hij wordt aangezet, onmiddellijk beginnen met het uitvoeren van dit programma, tenzij het MSX-DOS operating system op deze schijf staat (zie verder).

LOAD

Het LOAD-kommando werkt als eerder in het grote MSX-handboek beschreven met dit verschil dat wanneer geen randapparaat werd gespecificeerd, automatisch de laatst benaderde schijf wordt genomen. Zowel programma's die met de ,A-optie zijn geSAVEd als de programma's die zonder deze optie op schijf werden gezet, worden geladen.

LOC

De LOC-funktie heeft, toegpast op een op schijf geopend bestand, dezelfde funktie als beschreven in het grote handboek met dit verschil, dat LOC op sequentiële bestanden altijd een veelvoud van 256 geeft en op random bestanden altijd het recordnummer van het laatst benaderde record geeft.

Bijvoorbeeld:

```
NEW
Ok
10 REM LOC EN DE TAPE
20 OPEN "CAS:TEST" FOR OUTPUT AS 1
30 FOR I=1 TO 10:PRINT #1,"TEST"
40 PRINT LOC(1);:NEXT I:CLOSE:STOP
RUN
 6  12  18  24  30  36  42  48  54  60
Break in 40
Ok
```

```
NEW
Ok
10 REM LOC EN DE SEQUENTIAL FILE
20 OPEN "A:TEST" FOR OUTPUT AS 1
30 FOR I=1 TO 10:PRINT #1,STRING$(50,"*")
40 PRINT LOC(1);:NEXT I:CLOSE:STOP
RUN
 0  0  0  0  256  256  256  256  256  512
Break in 40
Ok

NEW
Ok
10 REM LOC EN DE RANDOM FILE
20 OPEN "A:TEST" AS 1 LEN=32
30 FOR I=1 TO 10:PUT #1,I
40 PRINT LOC(1);:NEXT I:CLOSE:STOP
RUN
 1  2  3  4  5  6  7  8  9  10
Break in 40
Ok
```

LOF

Net zoals van een bestand op tape geeft LOF ook van een bestand op schijf de actuele grootte aan.

```
NEW
Ok
10 REM LOF EN DE RANDOM FILE
20 OPEN "A:TEST" AS 1 LEN=32
40 PRINT LOF(1):CLOSE:STOP
RUN
 25600
Break in 40
Ok
```

RUN

Een met SAVE vastgelegd programma kan met een RUN "(programmanaam)" worden geladen en direkt worden opgestart. Onder MSX-diskbasic wordt indien geen randapparaat werd gespecificeerd, wederom de laatst benaderde schijf genomen. Zowel programma's mét als

zónder ,A-optie kunnen op deze wijze worden geaktiveerd.

MERGE

Het MERGE-kommando werkt als eerder in het grote MSX-handboek beschreven met dit verschil dat wanneer geen randapparaat wordt gespecificeerd, automatisch de laatst benaderde schijf wordt genomen. Wanneer een programma met MERGE van schijf wordt geladen, dan dient dat programma eerder met de ,A-optie op deze schijf te zijn geSAVEd.

PRINT#

Het PRINT#-kommando heeft PRECIES dezelfde werking in het MSX-disk basic als in het gewone MSX-basic. Het OPEN-kommando bepaalt op welk randapparaat het PRINT#-kommando wordt uitgevoerd; het printkommando zelf verandert niet.

(LINE) INPUT#

Ook het (LINE) INPUT#-kommando behoudt in het MSX-disk basic precies dezelfde werking. Door het open-kommando wordt bepaald vanuit welk randapparaat gegevens worden ingelezen.

EOF(...)

EOF behoudt eveneens dezelfde funktie; het OPEN-kommando is bepalend voor het type randapparaat.

VARPTR

Ook voor op schijf opgeslagen bestanden geeft deze funktie de waarde van het adres van het zogenaamde FILE CONTROL BLOCK in het geheugen.

CALL

Twee CALL's zijn met name belangrijk bij het gebruik van schijven-eenheden:

1. Om schijven te kunnen formatteren (zie hoofdstuk 3) dient een CALL te zijn voorgeschreven door de fabrikant van de schijven-eenheid. Meestal luidt deze call: CALL FORMAT.

2. Wanneer het MSX-basic vanuit het MSX-DOS operating system werd opgestart, dan kan door ingave van het kommando CALL SYSTEM naar dit operating system worden teruggekeerd.

BLOAD en BSAVE

Ook op schijf kunnen gegevens rechtstreeks uit het computergeheugen worden opgeslagen. BLOAD en BSAVE kiezen in de disk-versie van MSX automatisch voor de schijf indien geen ander randapparaat (b.v. CAS:) werd aangegeven.

De schrijfwijze van BLOAD en BSAVE is voor MSX-DISK-basic iets gewijzigd:

```
                          [  R                                  ]
                          [,---[,<VERSCHUIVING 1>]              ]
BLOAD<BESTANDSNAAM>       [  S                                  ]
                          [------------------------             ]
                          [   ,<VERSCHUIVING 2>                 ]
```

```
<VERSCHUIVING 1>::=<N>
```

```
                                R
<VERSCHUIVING 2>::=<N>\---<...>
                                S
```

```
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

```
<...>::=<ZIE ALGEMENE SPECIFICATIES>
```

```
<BESTANDSNAAM>::=<A>
```

```
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

```
BSAVE<BESTANDSNAAM>,<EERSTE BYTE>,<LAATSTE BYTE>[,<STARTADRES>]
                                                [------------]
                                                [    ,S      ]
```

```
<BESTANDSNAAM>::=<A>
```

```
<EERSTE BYTE>::=<N>
```

```
<LAATSTE BYTE>::=<N>
```

```
<STARTADRES>::=<N>\S
```

```
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

```
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met de ‚S-optie kan worden aangegeven dat niet het centrale werkgeheugen maar het videogeheugen moet worden geadresseerd. Bij 128 KB videogeheugen (MSX-2) is alleen de eerste 64 KB op deze wijze te benaderen.

In het volgende programma wordt eerst een tekening samengesteld waarna deze op schijf wordt opgeslagen. Het tweede programma laadt deze tekens weer van schijf. Merk op dat voorafgaand aan de BLOAD-opdracht de juiste scherminstelling moet zijn gekozen.

```
NEW
Ok
10 REM EERSTE PROGRAMMA
20 REM MAAK EERST EEN TEKENING
30 SCREEN 2:COLOR 15,4,4:CLS
40 FOR I=0 TO 90 STEP 10
50 LINE (I,I)-(255-I,191-I),,B
60 NEXT I
70 REM EN SLA DEZE OP SCHIJF OP (ONDERSTE 16 KB)
80 BSAVE "SCHERM",0,16383,S
RUN

(tekening wordt samengesteld)
(schijveneenheid werkt  even)

Ok


NEW
Ok
10 REM TWEEDE PROGRAMMA
20 REM HAAL DE TEKENING WEER OP
30 SCREEN 2:COLOR 15,4,4:CLS
40 BLOAD "SCHERM",S
50 GOTO 50
RUN

(tekening wordt vrij snel)
(weer   op  beeld   gezet)
```

Onder MSX-2 dient na de BLOAD-opdracht naar het videogeheugen een COLOR=RESTORE-opdracht te worden uitgevoerd teneinde de juiste palette-instelling te activeren. Deze palette-instelling kan door de BLOAD-instruktie wel zijn geladen maar wordt niet eerder actief dan na deze instruktie.

Indien onder MSX-2 precies 64 kilobytes videogeheugen wordt opgeslagen (bijvoorbeeld met BSAVE "TEST",0,65535,S), ontstaan problemen. Met de ,S-optie kunnen maximaal 65535 bytes worden opgeslagen.

De ,S-optie kan alleen in combinatie met de schijveneenheid worden gebruikt.

Tot slot volgen hier onder enkele kleine voorbeelden waarin op schijf wordt gewerkt en waarin de 'oude', hier boven behandelde sleutelwoorden worden gebruikt.

voorbeeld 1

In dit voorbeeld wordt bestand BES01 op de schijf toegewezen en schoongemaakt. Vervolgens kunnen regels tekst worden ingegeven die dan in dit bestand worden weggeschreven. Als zodanig kan bijvoorbeeld een brief op schijf worden bewaard.

```
NEW
Ok
10 REM VOORBEELD 1, INGAVE TEKST
20 CLS:OPEN "BES01" FOR OUTPUT AS 1
30 PRINT "GEEF TEKST IN (*=EINDE)"
40 LINE INPUT R$:IF R$="*" THEN 60
50 PRINT#1,R$:GOTO 40
60 CLOSE:STOP
RUN

(beeldscherm wordt schoongemaakt)


GEEF TEKST IN (*=EINDE)

Beste lezer,

Met deze brief testen we ons eerste
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.

*
Break in 60
Ok
```

voorbeeld 2

In dit voorbeeld wordt bestand BES01 op schijf geopend waarna regels tekst kunnen worden AANGEVULD in dit bestand.

```
NEW
Ok
10 REM VOORBEELD 2, AANVULLEN TEKST
20 CLS:OPEN "BES01" FOR APPEND AS 1
30 PRINT "GEEF TEKST IN (*=EINDE)"
40 LINE INPUT R$:IF R$="*" THEN 60
50 PRINT#1,R$:GOTO 40
60 CLOSE:STOP
RUN
```

```
(beeldscherm wordt schoongemaakt)

GEEF TEKST IN (*=EINDE)

P.S.: Als het goed is, wordt deze
      tekst nog aangevuld...

*
Break in 60
Ok
```

voorbeeld 3

In dit voorbeeld wordt bestand BES01 geopend en bestand BES02 toegewezen en schoongemaakt op schijf. Vervolgens worden de regels tekst die in BES01 liggen opgeslagen, stuk voor stuk 'overgeheveld' in bestand BES02. Er kan worden bepaald welke regels wel en niet worden overgeheveld. Ook kunnen regels worden tussengevoegd.

```
NEW
Ok
10 REM VOORBEELD 3, OVERHEVELEN TEKST
20 MAXFILES=2
30 OPEN "BES01" FOR INPUT AS 1
40 OPEN "BES02" FOR OUTPUT AS 2
50 CLS:PRINT "0=REGEL OVERSLAAN"
60 PRINT "1=REGEL OVERNEMEN"
70 PRINT "2=NA DEZE REGEL TUSSENVOEGEN"
80 PRINT
90 IF EOF(1) THEN 190
100 LINE INPUT #1,R$
110 PRINT R$
120 K$=INKEY$:IF K$="" THEN 120
130 IF K$<>"0" AND K$<>"1" AND K$<>"2" THEN BEEP:GOTO 120
140 IF K$="0" THEN PRINT "VERWIJDERD !!!":BEEP:GOTO 90
150 PRINT #2,R$:IF K$="1" THEN 90
160 REM REGELS TUSSENVOEGEN
170 PRINT "TUSSENVOEGEN (*=EINDE)"
180 LINE INPUT R$:IF R$="*" THEN PRINT "EINDE TUSSENVOEGEN":
GOTO 90 ELSE PRINT #2,R$:GOTO 180
190 CLOSE:STOP
RUN

(beeld wordt schoongemaakt)


0=REGEL OVERSLAAN
1=REGEL OVERNEMEN
2=NA DEZE REGEL TUSSENVOEGEN
```

```
Beste lezer,

TUSSENVOEGEN (*=EINDE)
Allereerst mijn hartelijke dank voor
de aanschaf van dit boekwerkje.

*
EINDE TUSSENVOEGEN
Met deze brief testen we ons eerste
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.


P.S.: Als het goed is, wordt deze
TUSSENVOEGEN (*=EINDE)
      laatste regel verwijderd...
*
EINDE TUSSENVOEGEN
      tekst nog aangevuld...
VERWIJDERD !!!

Break in 190
Ok
```

voorbeeld 4

In dit voorbeeld wordt BES02 in BES01 terug gekopieerd en vervolgens verwijderd. Zo kan het programma in voorbeeld 3 steeds worden herhaald.

```
NEW
Ok
10 REM VOORBEELD 4, BES02->BES01
20 MAXFILES=2
30 OPEN"BES02" FOR INPUT AS 1
40 OPEN"BES01" FOR OUTPUT AS 2
50 IF EOF(1) THEN 70
60 LINE INPUT #1,R$:PRINT #2,R$:GOTO 50
70 CLOSE:STOP
RUN
Break in 70
Ok
```

voorbeeld 5

In dit voorbeeld wordt bestand BES01 regel voor regel ingelezen en

op beeldscherm afgedrukt. Door op regel 40 de PRINT te vervangen door een LPRINT kan worden bewerkstelligd dat de regels op een eventuele printer worden afgedrukt.

```
NEW
Ok
10 REM VOORBEELD 5, PRINT BES01
20 CLS:OPEN "BES01" FOR INPUT AS 1
30 IF EOF(1) THEN CLOSE:STOP
40 LINE INPUT #1,R$:PRINT R$:GOTO 30
RUN

(beeld wordt schoongemaakt)


Beste lezer,

Allereerst mijn hartelijke dank voor
de aanschaf van dit boekwerkje.

Met deze brief testen we ons eerste
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.


P.S.: Als het goed is, wordt deze
      laatste regel verwijderd...

Break in 30
Ok
```

Deze vijf programma's bij elkaar vormen een héél eenvoudig tekstverwerkingspakketje. Misschien een uitdaging voor de amateur om het een en ander verder uit te bouwen en te verfraaien...

MSX-2

Bovenstaande programma's kunnen voor de RAM-DISK in orde worden gemaakt door BES01 en BES02 te vervangen door MEM:BES01 en MEM:BES02.

4.2 De nieuwe sleutelwoorden

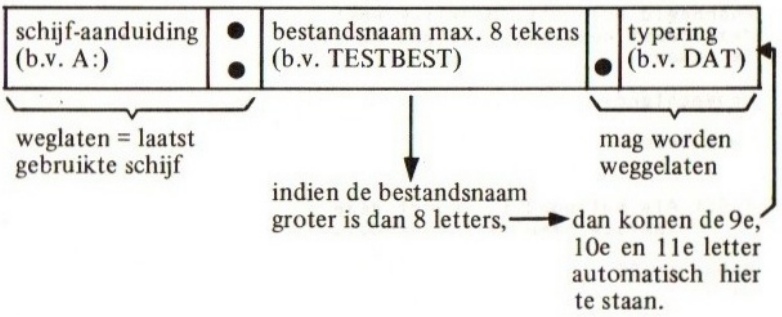
In deze paragraaf worden de sleutelwoorden behandeld die het MSX-disk basic extra biedt.
Echter, voordat we tot deze behandeling overgaan, dient eerst een veel voorkomende term nader te worden uitgelegd en wel de term:

⟨BESTANDSAANDUIDING⟩

Een bestandsaanduiding is een naamgeving waarmee we geen, één of een groep van bestanden kunnen aanduiden. Dit in tegenstelling tot een bestands*naam,* waarmee we precies één bestand kunnen aanduiden.

Om verder op deze term in te gaan, dienen we ons eerst te realiseren welke componenten een bestands*naam* op schijf in zich heeft of in zich kan hebben:

bestandsnaam: op schijf (b.v. A:TESTBEST.DAT)



Allereerst onderscheiden we in de bestandsnaam de schijfaanduiding (A:/B:/enz.).
Vervolgens onderscheiden we de naam van het bestand.
Achter de naam mag vervolgens een punt worden opgenomen, gevolgd door een drie-letterige typeaanduiding. Deze type-aanduiding is geheel vrij en mag naar believen worden toegepast. Het is echter raadzaam om bij het programmeren een vaste regel in verband met deze typering te hanteren. Bijvoorbeeld:

– een basic programma op schijf (geen ,A)	XXXXXXXX.BAS
– een basic programma op schijf (met ,A)	XXXXXXXX.ASC

– een bestand met daarin alleen tekst	XXXXXXXX.TXT
– een sequentiëel bestand	XXXXXXXX.SEQ
– een random bestand	XXXXXXXX.RAN

Door alle bestanden van een bepaalde soort ook een bepaalde type-aanduiding te geven, zijn deze later (wanneer er tientallen bestanden op schijf staan) weer snel bij elkaar te zoeken.

Een programma kan dus bijvoorbeeld als volgt worden geSAVEd:

```
SAVE "A:VB01.BAS"
```

Met A: geven we aan dat het programma op de eerste schijf moet worden geschreven. Het programma zelf heet VB01 (voorbeeld 1) en is door middel van .BAS tot basic programma getypeerd.

```
SAVE "B:OVL.ASC",A
```

Het programma OVL wordt op de tweede schijveneenheid geschreven. Omdat het een ASCII-bestand is (de ,A-optie), krijgt het programma de typering ASC mee.

Schijfaanduiding en typering mogen worden weggelaten. Indien de schijfaanduiding wordt weggelaten, wordt de laatst aangesproken schijveneenheid aangenomen.

In MSX-disk basic is het bij sommige sleutelwoorden noodzakelijk om een groep van bestanden tegelijk aan te kunnen aanduiden. Dit doen we met een zogenaamde bestandsaanduiding.

Een bestandsaanduiding kan allereerst een schijf-aanduiding bevatten (bijvoorbeeld A: of B:) maar dit is niet noodzakelijk.

Vervolgens kan de bestandsaanduiding een bestandsnaam bevatten. Deze naam mag geheel worden ingevuld, maar sommige tekens mogen worden vervangen door een vraagteken. De groep van bestanden die met die bestandsaanduiding wordt bedoeld, mogen in hun naam op deze positie *elke* letter bevatten.

Vervolgens kan een type-aanduiding worden opgenomen. Voor deze typering geldt eveneens dat sommige tekens door een vraagteken mogen worden vervangen. Ook mag een typering geheel worden weggelaten.

Enkele voorbeelden:

bestandsaanduiding	groep van bestanden die wordt aangeduid
"BES??"	alle bestanden op de laatst geaktiveerde schijveneenheid die een naam van maximaal 5 letters hebben waarvan de eerste drie letters gelijk zijn aan BES
"A:?FILE"	alle bestanden op schijveneenheid A: (de eerste schijveneenheid) met een naam van 5 letters waarvan de laatste vier letters gelijk zijn aan FILE
"A:????.BAS"	alle bestanden op schijveneenheid A: met een naam van maximaal vier letters en een typering BAS
A:PROGRAMM.???	alle bestanden op schijveneenheid A: met de naam PROGRAMM en eventueel een typering

Zowel de bestandsnaam als de typering mogen als laatste karakter een sterretje bevatten. Met dit sterretje geven we aan, dat de rest van de bestandsnaam of de typering er niet toe doet. Bijvoorbeeld:

bestandsaanduiding	groep van bestanden die wordt aangeduid
"BES*.DAT"	alle bestanden met een naam die begint met BES. De lengte van de naam doet er verder niet toe. Wel dient het bestand een typering DAT te hebben. De bestanden dienen aanwezig te zijn op de laatst benaderde schijveneenheid
"*.P*"	alle bestanden met een typering, beginnende met een P op de laatst aangesproken schijf
"A:*.*"	alle bestanden op schijveneenheid A:

Uiteraard mag een bestandsaanduiding ook gewoon een bestandsnaam bevatten. De 'groep' van bestanden die wordt bedoeld, bevat dan altijd maar één bestand.

NB: in bestandsnamen mogen de volgende tekens worden gebruikt:

A-Z	0-9	$	&	#	%	'
( )	-	@	^	{	}	!

Indien kleine letters worden gebruikt, zet MSX deze zelf om in hoofdletters.

SLEUTELWOORD FILES

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **FILES is bestanden**

schrijfwijze

```
FILES[<BESTANDSAANDUIDING>]

<BESTANDSAANDUIDING>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord FILES kan worden bekeken, welke bestanden zich op een schijf bevinden. Voorbeelden:

`FILES`	laat zien welke bestanden zich op de laatst aangesproken schijf bevinden.
`FILES "*.DAT"`	laat zien welke bestanden zich op de laatst aangesproken schijveneenheid bevinden met een typering DAT.
`FILES "A:P*.*"`	laat alle bestanden op schijveneenheid A: zien waarvan de namen met een P beginnen.
`FILES "A:TEST.BAS"`	laat zien of bestand TEST.BAS op de eerste schijveneenheid voorkomt.

Het volgende voorbeeld laat de inhoud zien van zo maar een schijf:

```
FILES "A:V*"
VDP01        VBPR         VDP02        VDP03        VDP04
VDP05        VDP06        VDP07        VDP08        VDP09
VDP10
Ok
```

SLEUTELWOORD LFILES

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LFILES is samentrekking van line printer en files – bestanden (bestandsnamen) op afdrukeenheid (printer) afdrukken**

schrijfwijze

```
LFILES[<BESTANDSAANDUIDING>]

<BESTANDSAANDUIDING>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord heeft op één aspekt na precies dezelfde werking als het sleutelwoord FILES. Het verschil is, dat LFILES het bedoelde overzicht op een eventueel aangesloten printer afdrukt in plaats van op het beeldscherm.

SLEUTELWOORD KILL

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **KILL is doden, elimineren**

schrijfwijze

```
KILL<BESTANDSAANDUIDING>

<BESTANDSAANDUIDING>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord KILL kunnen we een bestand of een groep van bestanden volledig en in één klap van schijf verwijderen. Bijvoorbeeld:

`KILL "BEST.DAT"`	verwijdert bestand BEST.DAT van de schijf
`KILL "*.DAT"`	verwijdert alle bestanden van schijf met typering DAT
`KILL "A:*.*"`	verwijdert alle bestanden van schijveneenheid A:

Het spreekt vanzelf dat met het sleutelwoord KILL met grote voorzichtigheid dient te worden omgesprongen.

SLEUTELWOORD # COPY

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **COPY is kopieren**

schrijfwijze

```
COPY<BESTANDSAANDUIDING>[TO<BESTANDSAANDUIDING>]

<BESTANDSAANDUIDING>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

De grafische betekenis die copy onder MSX-2 kan hebben, wordt op blz. 80 behandeld.

Met het sleutelwoord COPY kunnen complete bestanden of groepen van bestanden worden gekopiëerd. Bijvoorbeeld:

```
COPY "A:BES1" TO "A:BES2"
```

Bestand BES1 wordt gekopiëerd onder de naam BES2 zodat nu twee identieke bestanden onder twee verschillende namen op de eerste schijf zijn opgeslagen.

```
COPY "T*" TO "Q*"
```

Alle bestanden waarvan de naam begint met een T en die geen typering hebben, worden op dezelfde schijf gekopiëerd onder een andere naam. De nieuwe naam is gelijk aan de oude naam op één letter na. De eerste letter, oorspronkelijk een T, is nu namelijk door een Q vervangen.

```
COPY "A:*.BAS" TO "A:*.PRG"
```

Alle bestanden op schijveneenheid A: met typering BAS worden op dezelfde schijf onder dezelfde namen maar onder een andere typering (PRG) gekopiëerd.

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

Kopiëert alle bestanden van de tweede schijveneenheid naar de eerste schijveneenheid. Indien er slechts één schijveneenheid aanwezig is, zal MSX deze schijveneenheid steeds afwisselend de rol van eerste en

tweede schijveneenheid laten spelen. Tussendoor vraagt MSX u automatisch steeds om de schijf te verwisselen.

```
COPY "B:"
```

Indien TO... wordt weggelaten, neemt de MSX-computer automatisch aan dat er naar de laatst benaderde schijveneenheid dient te worden gekopieerd. Is dit schijveneenheid A:, dan zal de uitwerking van dit kommando dezelfde zijn als hierboven beschreven.

De COPY "B:" TO "A:" biedt ons de mogelijkheid om in één keer de hele schijf te kopiëren. Het volgende voorbeeld laat zien hoe dat in zijn werk gaat:

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

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

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

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

Steeds dient afwisselend de originele en de kopie-schijf te worden aangeboden aan de disk drive. Bij aanvang dient de originele schijf in de eenheid te zitten. Stel eventueel met WRITE PROTECT-optie het origineel veilig.

Met het COPY-kommando kunnen onder MSX-2 ook grafische schermen naar en van schijf worden gekopiëerd. Zie hiervoor de aparte behandeling verderop.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **NAME is naam geven**

schrijfwijze

```
NAME<BESTANDSAANDUIDING>AS<BESTANDSAANDUIDING>

<BESTANDSAANDUIDING>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord NAME kan een bestand of een groep van bestanden van een nieuwe naam worden voorzien. Bijvoorbeeld:

```
NAME "A:FILE.BES" AS "BEST.FIL"
```

verandert de naam van FILE.BES in BEST.FIL op schijf A:

```
NAME "A:*.BAS" AS "*.PRG"
```

verandert de typeringen van de betreffende bestanden van BAS naar PRG.

```
NAME "B???" AS "Q???"
```

op de laatst aangesproken schijveneenheid worden de bestanden met een naam van vier letters beginnende met een B van een andere naam voorzien. De nieuwe naam van die bestanden begint met een Q en is verder gelijk aan de oude naam.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **DKSF is samentrekking van disk en free – vrije ruimte op schijf**

schrijfwijze

```
DSKF(<SCHIJFNUMMER>)

<SCHIJFNUMMER>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord DSKF kan de vrije ruimte die nog op een schijf over is, worden opgevraagd. Als schijfnummer kan het werkelijke schijfnummer worden opgegeven (1,2,3 . . . komt overeen met A:,B:,C: . . .) of kan een nulwaarde worden opgegeven, waarmee dan de laatst benaderde schijf kan worden bedoeld. Indien de numerieke uitdrukking die de schijveneenheid bepaalt, een decimale fraktie bevat, dan wordt deze verwaarloosd.

De waarde die de DSKF-funktie als resultaat geeft, kan verschillende betekenissen hebben, afhankelijk van het merk en type van de schijveneenheid. Meestal wordt een numerieke waarde toegekend die gelijk is aan het aantal KILOBYTES dat nog vrij is op de schijf. Bijvoorbeeld:

```
PRINT DSKF(0)
 273
Ok
SAVE "TEST"
Ok
PRINT DSKF(0)
 272
Ok
```

In het bovenstaande voorbeeld bleken op een bepaalde schijf nog 273 kilobytes aan ruimte over te zijn. Het daarna op schijf gezette programma bleek één kilobyte aan vrije ruimte te bezetten.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **FIELD is veld, gebied**

schrijfwijze

```
FIELD[#]<KANAAL><RECORDINDELING>

<RECORDINDELING>::={,<VELDLENGTE>AS<VELDNAAM>}\<NIETS>

<KANAAL>::=<N>

<VELDLENGTE>::=<N>\<...><VARIABELE-NAAM>

<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<VARIABELE-NAAM>::=<ZIE ALGEMENE SPECIFICATIES>

<ALFANUMERIEKE VARIABELE>::=<ZIE ALGEMENE SPECIFICATIES>
```

Wanneer we in MSX-basic met RANDOM FILES willen werken, dienen RECORDS te definiëren (zie de in hoofdstuk 3 behandelde stof). Het definiëren van records dienen we in MSX-basic met het FIELD-sleutelwoord te doen. Voorafgaand aan het FIELD-kommando dient een OPEN-kommando te zijn opgenomen dat een RANDOM-bestand op het betreffende kanaal opent. Dit openen dient *niet* met één van de toevoegingen FOR INPUT, FOR OUTPUT of FOR APPEND te geschieden.

Na het openen dient de recordsamenstelling met het FIELD-kommando te worden gedefiniëerd.

Na het FIELD-sleutelwoord dient eerst, al dan niet voorzien van een hekje, het kanaalnummer te worden opgenomen. Dit kanaalnummer moet in waarde gelijk zijn aan het kanaalnummer waarover het betreffende bestand werd geopend. Een eventuele decimale fraktie wordt verwaarloosd.

Vervolgens dienen per veld de lengten van dat veld (in bytes) en de naam van dat veld (gewoon de naam van een alfanumerieke variabele) te worden opgenomen.

Bijvoorbeeld:

```
NEW
Ok
10 OPEN "KENNIS" AS #1 LEN=97
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$,24 AS WW$,24 AS TT$
```

In dit voorbeeld wordt op regel 10 eerst het bestand KENNIS geopend. Indien het betreffende bestand nog niet aanwezig was, wordt het gecreëerd. De recordlengte wordt op 97 posities bepaald.

Op regel 20 wordt van het RANDOM bestand KENNIS de recordsamenstelling bepaald. Het record bestaat uit een veld met MM$ (1 byte lang), een veld NN$ (24 bytes lang), een veld SS$ (24 bytes lang), een veld WW$ (24 bytes lang) en een veld TT$ (24 bytes lang). Het totale record is dus 1+24+24+24+24=97 bytes lang.

Het bovenstaande voorbeeld is een gedeelte van het voorbeeld dat in hoofdstuk 3 werd getoond. In de verdere behandeling zullen we ons waar mogelijk aan dit voorbeeld houden.

SLEUTELWOORD LSET

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LSET is afkorting van left set – links klaarzetten**

schrijfwijze

```
LSET<VELDNAAM>=<A>

<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>

<ALFANUMERIEKE VARIABELE>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Variabelen die met behulp van het FIELD-kommando als naam zijn toegekend aan een veld uit een record, mogen niet zo maar op de gebruikelijke wijze aan een waarde worden gelijkgesteld. De FIELD-variabelen zijn zeer speciale variabelen in MSX-basic die alleen met het LSET, RSET of GET-kommando van waarde mogen worden veranderd. Wanneer we bijvoorbeeld zouden programmeren:

```
NEW
Ok
10 OPEN "KENNIS" AS #1 LEN=97
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$,24 AS WW$,24 AS TT$
30 NN$="GERT-JAN JANSSEN"    ( ← dit mag niet!)
```

dan koppelen we het veld NN$ af van het FIELD-kommando en wordt het verder onbruikbaar voor de bestandsbenadering.

Met het LSET-kommando kunnen we een waarde aan een recordveld toekennen. Deze waarde wordt dan LINKS in het recordveld aangesloten terwijl het recordveld vervolgens eventueel met spaties wordt aangevuld. Bijvoorbeeld:

```
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
```

Uitgaande van het voorbeeld, gegeven onder FIELD, worden in dit voorbeeld de velden NN$, SS$, WW$ en TT$ gelijkgesteld aan N$, S$, W$ en T$. Alle vier de veldwaarden worden tot 24 posities aangevuld met spaties.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RSET is afkorting van right set – rechts klaarzetten**

schrijfwijze

```
RSET<VELDNAAM>=<A>

<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>

<ALFANUMERIEKE VARIABELE>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Het RSET-sleutelwoord heeft ongeveer dezelfde betekenis als het LSET-sleutelwoord. Het verschil is, dat RSET de betreffende waarde RECHTS in het recordveld aansluit. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "FILE" AS 1
20 FIELD #1,10 AS SA$
30 RSET SA$="123.45"
40 PRINT SA$
50 CLOSE:STOP
RUN
    123.45
Break in 50
Ok
```

Bovenstaand voorbeeld laat zien dat een met behulp van RSET aan een veld toegekende waarde rechts uitgelijnd wordt geplaatst.

SLEUTELWOORD PUT

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PUT is zetten, plaatsen**

schrijfwijze

```
PUT[#]<KANAAL>[,<RECORD>]

<KANAAL>:::=<N>

<RECORD>:::=<N>

<N>:::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het PUT-kommando kan een kompleet record (dus meerdere velden) in het bestand worden geschreven. Het record dat wordt geschreven, moet in een FIELD-kommando zijn gespecificeerd terwijl alle velden met een LSET of een RSET (of eventueel een GET) dienen te zijn gevuld.

Achter het PUT-kommando dient eerst eventueel een hekje en daarna het kanaalnummer te worden opgenomen. Een eventuele decimale fraktie wordt verwaarloosd. Vervolgens kan een recordnummer worden opgenomen. Dit recordnummer bepaald als hoeveelste record het betreffende record in het bestand wordt geschreven. Een eventuele decimale fraktie wordt verwaarloosd.

Uitgaande van het voorbeeld, gegeven onder FIELD en LSET, schrijft het kommando:

```
270 PUT #1,R
```

het record, bestaande uit MM$, NN$, SS$, WW$ en TT$, in bestand KENNIS. Als hoeveelste record het betreffende record wordt geschreven, wordt bepaald door de variabele R.

Wanneer een recordnummer wordt weggelaten, dan wordt het record automatisch als volgende record geschreven. Indien na het openen van het bestand nog niet eerder een PUT of GET werd uitgevoerd, dan wordt het eerste record geschreven.

SLEUTELWOORD GET

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **GET is krijgen, pakken**

schrijfwijze

```
GET[#]<KANAAL>[,<RECORD>]

<KANAAL>::=<N>

<RECORD>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Het GET-kommando werkt precies tegenovergesteld aan het PUT-kommando; de velden van een bepaald record worden juist ingelezen. Uitgaande van het voorbeeld, gegeven onder FIELD, LSET en PUT, leest het kommando:

```
280 GET #1,R
```

het record weer in. Door dit ene, eenvoudige kommando worden de velden MM$, NN$, SS$, WW$ en TT$ weer vanaf de schijf ingelezen.

Indien het recordnummer wordt weggelaten, wordt het volgende (of eerste) record ingelezen (als PUT).

Met de tot nu toe behandelde kommando's en funkties kan al volledig worden gewerkt met random bestanden.

In hoofdstuk 3 werd reeds een voorbeeld van een bestandsonderhoudsprogramma gegeven. Dat voorbeeld wordt hier nog eens geplaatst:

```
NEW
Ok
10 OPEN "KENNIS" AS #1 LEN=97
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$,24 AS WW$,24 AS TT$
30 PUT #1,101
40 CLS:PRINT "VULLEN KENNISSENBESTAND"
50 LOCATE 0,3:INPUT "KENNISNUMMER";K
60 IF K=0 THEN CLOSE:STOP
70 IF K>100 THEN BEEP:GOTO 40
80 IF K<>INT(K) THEN BEEP:GOTO 40
90 GET #1,K
```

```
100 IF MM$<>"*" THEN LSET MM$="*":LSET NN$="":LSET SS$="":LS
ET WW$="":LSET TT$=""
110 N$=NN$
120 S$=SS$
130 W$=WW$
140 T$=TT$
150 LOCATE 0,5
160 PRINT "NAAM? ";N$
170 PRINT "STRAAT? ";S$
180 PRINT "WOONPLAATS? ";W$
190 PRINT "TELEFOONNUMMER? ";T$
200 LOCATE 0,5
210 INPUT "NAAM";N$
220 INPUT "STRAAT";S$
230 INPUT "WOONPLAATS";W$
240 INPUT "TELEFOONNUMMER";T$
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
270 PUT #1,K
280 GOTO 40
```

```
(beeld wordt schoongemaakt)


VULLEN KENNISSENBESTAND


KENNISNUMMER? 3

NAAM? PIET JANSSENS
STRAAT? STRAATWEG 111
WOONPLAATS? SCHIEDAM
TELEFOONNUMMER? 010-123456
```

Op regel 10 wordt het bestand geopend en eventueel aangemaakt. Het bestand krijgt kanaal 1 toegewezen, terwijl de totale recordlengte op 97 bytes wordt gesteld.

Op regel 20 wordt de recordindeling gespecificeerd. Het record bestaat in dit geval uit:

- MM$ dit veld bevat een sterretje indien het record door het programma eerder werd gevuld.
- NN$ dit veld bevat de naam van de kennis waarvan we de gegevens willen opslaan.
- SS$ dit veld bevat straat en huisnummer van de betreffende kennis.

WW$ dit veld bevat postkode en woonplaats van de betreffende kennis.
TT$ dit veld bevat het telefoonnummer van de betreffende kennis.

Veld MM$ kan maximaal 1 positie lang zijn; de overige velden hebben allemaal een maximale lengte van 24 posities.

Op regel 30 wordt record nummer 101 geschreven. In het bestand KENNIS willen we maximaal 100 kennissen opslaan. Om er voor te zorgen dat we later bij een GET van een nog niet eerder geschreven record een foutmelding krijgen, passen we de truc toe dat we het hoogst mogelijke record plus 1 schrijven. Alle onderliggende records worden, alhoewel ze met onzinnige gegevens gevuld kunnen zijn, dan wel als gedefiniëerd beschouwd. Een GET op één van deze records geeft dan geen foutmelding meer.

Op regel 40 wordt het beeldscherm opgemaakt en op regel 50 kan een kennisnummer worden ingegeven. Op regel 60, 70 en 80 wordt gekontroleerd of het ingegeven nummer wel groter dan 0, kleiner dan 101 en geheel is. Indien een nul werd ingegeven, wordt het programma na een CLOSE (afsluiten van het bestand) beëindigd.

Op regel 90 wordt het record met het ingegeven (kennis)nummer gelezen. Op regel 100 wordt vervolgens gekontroleerd of het record reeds eerder werd geschreven (MM$ moet dan gelijk zijn aan een *). Indien dit niet het geval is, worden de recordvelden allemaal met behulp van het LSET-kommando op spaties gesteld terwijl MM$ nu wel aan een sterretje wordt gelijkgesteld.

Op regel 110, 120, 130 en 140 worden de variabelen N$, S$, W$ en T$ gelijk gesteld aan NN$, SS$, WW$ en TT$. De recordvelden worden ter verdere bewerking overgeheveld in niet aan het record gekoppelde variabelen.

De regels 150, 160, 170, 180 en 190 dragen er zorg voor, dat de inhoud van het zojuist ingelezen record netjes op het beeldscherm wordt geprojekteerd. De regels 200, 210, 220, 230 en 240 staan vervolgens nieuwe ingaven voor de betreffende variabelen (N$, S$, W$ en T$) toe. Doordat deze ingaven (zie de LOCATE op 150 en 200) over de projektie heen wordt uitgevoerd, wordt bewerkstelligd dat de ingave van alleen een RETURN de betreffende variabele gelijkstelt aan de oude waarde. Ook kunnen de reeds geprojekteerde gegevens met de

MSX full-screen editor worden bewerkt.

Dit over elkaar projekteren en ingeven is een leuke truc die in MSX mogelijk is en krachtig programmeren toestaat.

Op regel 250 en 260 worden op gebruikelijke wijze de recordvelden weer op de juiste waarden gebracht. De records worden netjes rechts aangevuld met spaties tot de geldende veldlengten.

Op regel 270 wordt het eerder opgehaalde record weer op dezelfde plaats in het bestand teruggeschreven waarna (GOTO 40) een volgend kennisnummer kan worden ingegeven.

Het spreekt voor zich dat dit programma naar eigen wens kan worden aangepast. De ervaren amateur maakt met behulp van dit programma in een kwartiertje zijn eigen bestandsonderhoudsprogramma waarin hij of zij de gegevens van de leden van een voetbalclub, een postzegelverzameling en dergelijke bijhoudt.

moeilijkheidsgraad . . .**vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . . **A-FUNKTIE**

afkomst . . . . . .**MKI$ is afkorting van make integer value string – maak string van integere waarde**

schrijfwijze

```
MKI$(<N>)
```

```
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Wanneer we numerieke gegevens in een bestand willen schrijven, dienen we deze gegevens eerst te converteren naar een alfanumeriek gegeven. Pas nadat we het kommando:

```
LET V$=STR$(V)
```

hebben uitgevoerd, kunnen we de waarde van de variabele V in een record opnemen en vervolgens schrijven. Andersom wordt V na het lezen van dit record weer op de juiste waarde gezet door een:

```
LET V=VAL(V$)
```

Wanneer in bovenstaand voorbeeld de variabele V een waarde heeft van 12345678, dan zijn er minimaal acht bytes nodig om deze waarde volgens bovenstaand recept in een bestand te schrijven. Echter, in het computergeheugen beslaat deze variabele V maar 4 bytes aan geheugenruimte. Dit komt doordat de interne opslag van numerieke variabelen in het MSX-computergeheugen op een bepaalde wijze zijn gekodeerd, afwijkend van de ASCII-kodering.

Om nu dit verlies van ongeveer 100 % aan schijfruimte te kunnen voorkomen, biedt MSX de mogelijkheid om een variabele in een string te plaatsen zoals deze ook in het geheugen is gekodeerd. Hierdoor kan een integere waarde altijd in twee bytes, een waarde van enkelvoudige precisie in vier bytes en een waarde van dubbele precisie in acht bytes worden gekodeerd. Een bijkomend verschijnsel is, dat de conversie van numerieke waarde naar string-waarde door de computer op deze wijze sneller kan worden uitgevoerd.

Het kommando:

```
LET V$=MKI$(V)
```

resulteert altijd in een alfanumerieke variabele van twee bytes lang. Wanneer we met een:

```
PRINT V$
```

de waarde van V$ proberen te achterhalen, krijgen we op het scherm slechts zinloze gegevens; de informatie is niet volgens de ASCII-kodering in de string opgenomen en dus niet zondermeer dekodeerbaar.

Echter, wanneer we intikken:

```
LET V=CVI(V$)
PRINT V
```

dan zal de computer met de juiste waarde reageren; de funktie CVI is in staat om van de schijnbaar zinloze informatie in V$ weer een juiste numerieke waarde te maken.

Voor deze optimale conversie van numerieke waarden naar strings en omgedraaid, kennen we zes verschillende funkties:

FUNKTIE	resultaat	lengte van de string	tegenovergestelde funktie
V$=MKI$(V)	een integere waarde wordt naar een alfanumerieke waarde geconverteerd	2 bytes	V=CVI(V$)
V$=MKS(V)	idem voor een waarde van enkelvoudige precisie	4 bytes	V=CVS(V$)
V$=MKD$(V)	idem voor een waarde van dubbele precisie	8 bytes	V=CVD(V$)

Steeds zijn als voorbeeld de variabelen V en V$ gebruikt; elke andere variabele is natuurlijk toegestaan.

Numerieke waarden kunnen in een random bestand als volgt worden verwerkt:

```
NEW
Ok
10 OPEN "TEST" AS 1
20 FIELD #1,2 AS S$,4 AS T$,8 AS U$
30 S%=32767:S!=123456!:S#=1.234E+36
50 LSET S$=MKI$(S%):LSET T$=MKS$(S!)
60 LSET U$=MKD$(S#)
70 PUT 1,20:S%=0:S!=0:S#=0
80 GET 1,20:S%=CVI(S$):S!=CVS(T$):S#=CVD(U$)
90 PRINT S%,S!,S#
100 CLOSE:KILL "TEST":STOP
RUN
 32767         123456          1.234E+36
Break in 100
Ok
```

In het bovenstaande voorbeeld worden drie verschillende typen numerieke variabelen gevuld en geconverteerd naar string-variabelen. Deze stringvariabelen worden in bestand TEST geschreven. Vervolgens worden ze weer ingelezen en worden deze variabelen weer naar numerieke variabelen geconverteerd en afgedrukt. Uiteindelijk wordt bestand TEST van schijf verwijderd.

PAS OP:
Wanneer via MKI$, MKS$ en MKD$ geconverteerde variabelen in een bestand dienen te worden geschreven, dan mag dat slechts een random bestand zijn. Het printen van dergelijke stringwaarden in een sequentieel bestand geeft onherroepelijk moeilijkheden bij het latere teruglezen.

moeilijkheidsgraad . . **vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . . **A-FUNKTIE**

afkomst . . . . . **MKS$ is afkorting van make single precision value string – maak string van enkelvoudige precisie variabele**

schrijfwijze

```
MKS$(<N>)

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

moeilijkheidsgraad . . .vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel.

soort . . . . . A-FUNKTIE

afkomst . . . . . MKD$ is afkorting van make double precision value string – maak string van dubbele precisie variabele

schrijfwijze

```
MKD$(<N>)

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

moeilijkheidsgraad . . **vrij moeilijk, kennis van de opslag methodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . . **N–FUNKTIE**

afkomst . . . . . **CVI is afkorting van convert to integer value – zet over naar integere waarde**

schrijfwijze

```
CVI(<A>)

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

moeilijkheidsgraad . . **vrij moeilijk, kennis van de opslag methodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . . **N-FUNKTIE**

afkomst . . . . . **CVS is afkorting van convert to single precision value – zet over naar enkelvoudige precisie waarde**

schrijfwijze

```
CVS(<A>)

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

moeilijkheidsgraad . . .**vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . . **N-FUNKTIE**

afkomst . . . . . **CVD is afkorting van convert to double precision value – zet over naar dubbele precisie waarde**

schrijfwijze

```
CVD(<A>)

<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

5 FOUTMELDINGEN OP VOLGORDE VAN NUMMER

Het MSX-disk basic kent de volgende foutmeldingen, hier op nummer opgenomen:

01 NEXT without FOR	gepoogd werd om een NEXT uit te voeren zonder dat een bijbehorende FOR werd uitgevoerd.
02 Syntax error	er werd een fout in de schrijfwijze ontdekt.
03 RETURN without GOSUB	gepoogd werd om een RETURN uit te voeren terwijl er niet eerder een GOSUB werd uitgevoerd.
04 Out of DATA	gepoogd werd om een READ uit te voeren terwijl er geen DATA meer kon worden gevonden.
05 Illegal function call	er werd gepoogd om een kommando of funktie uit te voeren met niet toegestane waarden.
06 Overflow	het resultaat van een berekening ligt boven het toegestane maximum of onder het toegestane minimum.
07 Out of memory	er werd gepoogd om een kommando uit te voeren dat meer geheugen nodig heeft dan er beschikbaar is.
08 Undefined line number	een niet bestaand programmaregelnummer werd benaderd.
09 Subscript out of range	een array-variabele werd buiten zijn dimensies aangesproken of er werd een verkeerd aantal dimensies genoemd.
10 Redimensioned array	er werd gepoogd om een array-variabele voor een tweede keer te DIMensioneren.
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te bepalen.
12 Illegal direct	er werd gepoogd om een kommando direkt in te tikken terwijl dit kommando slechts in een programmaregel mag voorkomen.

13 Type mismatch	een numerieke en een alfanumerieke uitdrukking werden met elkaar verwisseld.
14 Out of string space	gepoogd werd om een kommando uit te voeren waardoor het stringgeheugen te klein werd. Vergroot het string-geheugen met CLEAR.
15 String too long	er werd gepoogd om een string samen te stellen die langer werd dan 255 posities.
16 String formula too complex	de alfanumerieke uitdrukking die werd uitgewerkt is te ingewikkeld voor MSX-basic. Splits deze uitdrukking in enkele kleinere.
17 Can't continue	gepoogd werd on een programma met CONT te vervolgen terwijl dat niet (meer) gaat.
18 Undefined user function	gepoogd werd om met FN een niet gedefinieerde funktie aan te roepen.
19 Device I/O error	een fout werd ontdekt tijdens het lezen van/schrijven naar een randapparaat (cassetterecorder of printer e.d.).
20 Verify error	tijdens de controle met CLOAD? werd een verschil ontdekt tussen het programma op cassetteband en het programma in het geheugen.
21 No RESUME	de ERROR-routine werd aangeroepen en een RESUME kon niet worden gevonden.
22 RESUME without error	er werd gepoogd om een RESUME uit te voeren terwijl er geen fout via de ON ERROR GOTO werd gedetecteerd.
24 Missing operand	een noodzakelijke uitdrukking wordt in een kommando niet gevonden.
25 Line buffer overflow	een ingetikte programmaregel is te lang voor MSX-basic (langer dan 255 karakters).
50 FIELD overflow	de lengte van het met FIELD gedefinieerde record is groter dan 256 bytes.
51 Internal error	deze melding wijst op een fout die niet zou mogen kunnen voorkomen. Licht uw computerleverancier of MICROSOFT in.

52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
53 File not found	het te benaderen bestand werd niet op de schijf aangetroffen.
54 File already open	een reeds bezet kanaal wordt met OPEN een tweede maal gebruikt of een bestand wordt over twee kanalen geopend.
55 Input past end	er werd gepoogd om nog gegevens uit een bestand te lezen terwijl dit bestand reeds volledig was doorgewerkt.
56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
57 Direct statement in file	tijdens het LOADen van een programma werd een direkt kommando (zonder regelnummer) ontdekt; LOAD werd gestopt.
58 Sequential I/O only	een randapparaat dat alleen sequentiele benadering toelaat werd met GET/PUT benaderd.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.
60 Bad FAT	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 60 gegenereerd.
61 Bad file mode	het benaderde bestand is van een verkeerde soort (sequentieel terwijl random werd verwacht of andersom).
62 Bad drive name	er werd een niet bestaande schijfeneenheid aangeroepen.
63 Bad sector number	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 63 gegenereerd.
64 File still open	met NAME/COPY/KILL werd een bestand benaderd dat nog is geopend (zie OPEN).
65 File already exists	met NAME werd geprobeerd om de naam van een reeds op de schijf aanwezig bestand aan een ander bestand toe te kennen.
66 Disk full	er is niet genoeg ruimte op de schijf over.

67 Too many files	er werd geprobeerd om een honderddertiende file op een schijf toe te wijzen.
68 Disk write protected	er werd geprobeerd om te schrijven naar een schijf die WRITE PROTECTED is (zie H.3).
69 Disk I/O error	er werd een lees/schrijffout op schijf ontdekt. de schijveneenheid kan iets mankeren of de schijf is niet geinitialiseerd.
70 Disk offline	de schijveneenheid staat niet aangeschakeld of er is geen schijf geplaatst.
71 Rename across disk	er wordt geprobeerd om een bestand met NAME een andere schijveneenheid toe te kennen.

6 FOUTMELDINGEN OP ALFABETISCHE VOLGORDE

Het MSX-disk basic kent de volgende foutmeldingen, hier op alfabetische volgorde opgenomen:

60 Bad FAT	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 60 gegenereerd.
62 Bad drive name	er werd een niet bestaande schijfeenheid aangeroepen.
61 Bad file mode	het benaderde bestand is van een verkeerde soort (sequentieel terwijl random werd verwacht of andersom).
56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
63 Bad sector number	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 63 gegenereerd.
17 Can't continue	gepoogd werd on een programma met CONT te vervolgen terwijl dat niet (meer) gaat.
19 Device I/O error	een fout werd ontdekt tijdens het lezen van/schrijven naar een randapparaat (cassetterecorder of printer e.d.).
57 Direct statement in file	tijdens het LOADen van een programma werd een direkt kommando (zonder regelnummer) ontdekt: LOAD werd gestopt.
69 Disk I/O error	er werd een lees/schrijffout op schijf ontdekt. de schijfeenheid kan iets mankeren of de schijf is niet geinitialiseerd.
66 Disk full	er is niet genoeg ruimte op de schijf over.
70 Disk offline	de schijfeenheid staat niet aangeschakeld of er is geen schijf geplaatst.

68 Disk write protected	er werd geprobeerd om te schrijven naar een schijf die WRITE PROTECTED is (zie H.3).
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te berekenen.
50 FIELD overflow	de lengte van het met FIELD gedefinieerde record is groter dan 256 bytes.
65 File already exists	met NAME werd geprobeerd om de naam van een reeds op de schijf aanwezig bestand aan een ander bestand toe te kennen.
54 File already open	een reeds bezet kanaal wordt met OPEN een tweede maal gebruikt of een bestand wordt over twee kanalen geopend.
53 File not found	het te benaderen bestand werd niet op de schijf aangetroffen.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.
64 File still open	met NAME/COPY/KILL werd een bestand benaderd dat nog is geopend (zie OPEN).
12 Illegal direct	er werd gepoogd om een kommando direkt in te tikken terwijl dit kommando slechts in een programmaregel mag voorkomen.
05 Illegal function call	er werd gepoogd om een kommando of funktie uit te voeren met niet toegestane waarden.
55 Input past end	er werd gepoogd om nog gegevens uit een bestand te lezen terwijl dit bestand reeds volledig was doorgewerkt.
51 Internal error	deze melding wijst op een fout die niet zou mogen kunnen voorkomen. Licht uw computerleverancier of MICROSOFT in.
25 Line buffer overflow	een ingetikte programmaregel is te lang voor MSX-basic (langer dan 255 karakters).
24 Missing operand	een noodzakelijke uitdrukking wordt in een kommando niet gevonden.

01 NEXT without FOR	gepoogd werd om een NEXT uit te voeren zonder dat een bijbehorende FOR werd uitgevoerd.
21 No RESUME	de ERROR-routine werd aangeroepen en een RESUME kon niet worden gevonden.
04 Out of DATA	gepoogd werd om een READ uit te voeren terwijl er geen DATA meer kon worden gevonden.
07 Out of memory	er werd gepoogd om een kommando uit te voeren dat meer geheugen nodig heeft dan er beschikbaar is.
14 Out of string space	gepoogd werd om een kommando uit te voeren waardoor het stringgeheugen te klein werd. Vergroot het string-geheugen met CLEAR.
06 Overflow	het resultaat van een berekening ligt boven het toegestane maximum of onder het toegestane minimum.
22 RESUME without error	er werd gepoogd om een RESUME uit te voeren terwijl er geen fout via de ON ERROR GOTO werd gedetecteerd.
03 RETURN without GOSUB	gepoogd werd om een RETURN uit te voeren terwijl er niet eerder een GOSUB werd uitgevoerd.
10 Redimensioned array	er werd gepoogd om een array-variabele voor een tweede keer te DIMensioneren.
71 Rename across disk	er wordt geprobeerd om een bestand met NAME een andere schijfeneenheid toe te kennen.
58 Sequential I/O only	een randapparaat dat alleen sequentiele benadering toelaat werd met GET/PUT benaderd.
16 String formula too complex	de alfanumerieke uitdrukking die werd uitgewerkt is te ingewikkeld voor MSX-basic. Splits deze uitdrukking in enkele kleinere.
15 String too long	er werd gepoogd om een string samen te stellen die langer werd dan 255 posities.

09 Subscript out of range	een array-variabele werd buiten zijn dimensies aangesproken of er werd een verkeerd aantal dimensies genoemd.
02 Syntax error	er werd een fout in de schrijfwijze ontdekt.
67 Too many files	er werd geprobeerd om een honderddertiende file op een schijf toe te wijzen.
13 Type mismatch	een numerieke en een alfanumerieke uitdrukking werden met elkaar verwisseld.
08 Undefined line number	een niet bestaand programmaregelnummer werd benaderd.
18 Undefined user function	gepoogd werd om met FN een niet gedefinieerde funktie aan te roepen.
20 Verify error	tijdens de controle met CLOAD? werd een verschil ontdekt tussen het programma op cassetteband en het programma in het geheugen.

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **COPY is kopiëren**

schrijfwijze

```
COPY <LOCATIE>-<LOCATIE>[,<SCHERMNUMMER>] TO <BESTANDSAANDUIDING>

COPY <BESTANDSAANDUIDING>[,<RICHTING>] TO <LOCATIE>[,<SCHERM ...
     ... NUMMER>[,<LOGISCHE OPERATOR>]]\<...>,

COPY <BESTANDSAANDUIDING> TO <NUMERIEKE VARIABELE-NAAM>

COPY <NUMERIEKE VARIABELE-NAAM> TO <BESTANDSAANDUIDING>

<SCHERMNUMMER>::=<N>

<LOCATIE>::=(<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<RICHTING>::=<N>

<BESTANDSAANDUIDING>::=<A>

<LOGISCHE OPERATOR>::=XOR|OR|AND|PSET|PRESET|TXOR|TOR|TAND| ...
                      ... TPSET|TPRESET

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

<...>::=<ZIE ALGEMENE SPECIFICATIES>
```

De hieronder opgenomen behandeling geldt alleen voor MSX-2-computers en betreft de specifiek grafische aspecten van COPY.

betekenis

In het grote MSX-basic handboek werden reeds de diverse mogelijkheden van het COPY-kommando op grafisch gebied behandeld. In dit boek zagen we reeds welke de funkties van het COPY-kommando zijn in verband met het kopiëren van bestanden.

Onder MSX-2 heeft het COPY-kommando een nog wijdere betekenis. Het COPY-kommando stelt ons in staat om grafische gegevens direkt

vanaf het beeldscherm naar de schijf te kopiëren. Uiteraard kunnen op schijf opgeslagen gegevens ook weer onmiddellijk naar het grafische beeldscherm worden gekopieerd.

Ook hier geldt steeds dat het COPY-kommando alleen in verband met de scherminstellingen SCREEN 5 tot en met 8 kan worden toegepast.

Wanneer we direkt een gedeelte van een tekening vanaf scherm naar een bestand op schijf willen kopiëren, dan dienen we achter het COPY-kommando een rechthoekig gebied op te geven. Dit rechthoekige gebied wordt aangegeven met twee punten en geeft aan, welk gedeelte van het beeldscherm moet worden gekopieerd.

Vervolgens mogen we eventueel een beeldschermnummer opgeven. Wanneer we dit beeldschermnummer weglaten, wordt de informatie vanaf het actieve beeldscherm gekopieerd (zie SET PAGE). Door het opgeven van een beeldschermnummer kan de grafische informatie vanaf een niet actief scherm worden gekopieerd.

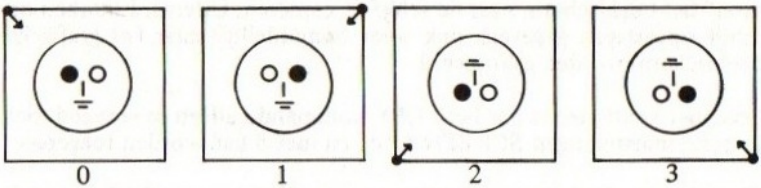
Nadat eventueel het beeldschermnummer is opgegeven, dient het sleutelwoord TO te worden opgenomen, gevolgd door de naam van het bestand waarnaar de gegevens dienen te worden gekopieerd.

In het volgende voorbeeld wordt er eerst een tekening op het beeldscherm gemaakt. Deze tekening wordt vervolgens naar een bestand gekopieerd. Een bestaand bestand wordt hierbij overschreven; een niet bestaand bestand wordt aangemaakt.

```
NEW
Ok
10 SCREEN 5:COLOR 15,4,4:CLS
20 FOR I=0 TO 64 STEP 4:LINE (0,I)-(64-I,0):NEXT I
30 COPY (0,0)-(80,80) TO "TEST"
RUN
```

Wanneer we een op schijf vastgelegde tekening later weer ergens op het scherm willen terugkopiëren, dienen we na het COPY-kommando eerst de bestandsnaam op te nemen van het bestand waarnaar de tekening eerder werd gekopieerd.

Vervolgens mogen we eventueel een richting opgeven. De betekenis van deze richtingkode wordt duideijk uit het volgende schemaatje:



Het weglaten van de richtingkode resulteert in het toepassen van richtingkode 0.

Nadat we eventueel de richtingkode hebben opgegeven, dient het sleutelwoord TO te worden opgenomen, gevolgd door een punt dat aangeeft, waarnaar moet worden gekopieerd. Het hierboven geplaatste schema geeft aan welk punt dit is.

Na dit punt mag eventueel nog een beeldschermnummer en een logische operator worden opgenomen.

In het volgende voorbeeld wordt de eerder gemaakte tekening weer in diverse standen naar het scherm gekopieerd.

```
NEW
Ok
10 SCREEN 5:COLOR 15,4,4:CLS
20 FOR I=0 TO 3
30 COPY "TEST",I TO (128,106)
40 NEXT I
50 GOTO 50
RUN
```

De logische operator bepaalt de kleuren waarin het uiteindelijk gekopieerde beeld wordt uitgevoerd. Voor de kleur van elk beeldpuntje worden steeds de kleur van het te kopiëren puntje en de kleur van het puntje waarop gekopiëerd gaat worden, in beschouwing genomen. Hun kleurnummers ondergaan een bepaalde, logische bewerking. De uitkomst van deze bewerking is het kleurnummer van het uiteindelijk resulterende puntje.

De logische bewerkingen worden steeds op de binaire waarden van de kleurnummers uitgevoerd. Om met deze logische operatoren te werken, is dus een behoorlijk inzicht noodzakelijk in het binaire stelsel en dienen de logische bewerkingen AND, OR en XOR te worden gekend.

De volgende logische operatoren kunnen worden gebruikt:

OPERATOR	UITWERKING
AND	De kleurnummers ondergaan de logische bewerking AND. De uitkomst is het kleurnummer van het resulterende beeldpunt.
OR	De kleurnummers ondergaan de logische OR-bewerking.
XOR	De kleurnummers ondergaan de logische bewerking XOR.
PSET	De kleurnummers ondergaan geen logische bewerking. De kleur van het te kopiëren punt wordt eenvoudigweg overgenomen.
PRESET	SCREEN 5 en 7: resulterende kleur = 15–originele kleur SCREEN 6: resulterende kleur = 3–originele kleur SCREEN 8: resulterende kleur = 255–originele kleur
TAND TOR TXOR TPSET TPRESET	zelfde betekenis als AND, OR, XOR, PSET of PRESET met dit verschil dat beeldpuntjes met kleurkode 0 niet worden meegekopiëerd. Op de plaats waar de kopie komt, blijft de originele kleur van deze plaats dus gehandhaafd.

Een voorbeeld: tijdens een COPY-opdracht heeft het beeldschermpuntje waarop gekopiëerd gaat worden, kleurkode 9. De kleurkode van het te kopiëren beeldpunt is 12. De logische operator is XOR.

De resulterende kleur laat zich als volgt berekenen:

```
kleur 9:    binair   1001
kleur 12:   binair   1100 XOR
resulterende kleur:  ----
                     0101 (5)
```

Merk op dat MSX-basic op het kommando PRINT 9 XOR 12 ook de uitkomst 5 geeft. Met MSX-basic kunnen we op een eenvoudige wijze de resulterende kleur aldus berekenen.

In het volgende voorbeeld wordt een tekening op schijf gezet en later weer meerdere malen naar het beeldscherm teruggekopieerd. Bij dit terugkopiëren wordt de logische operator XOR gebruikt.

```
NEW
Ok
10 SCREEN 5:COLOR 15,4,4:CLS
20 CIRCLE(20,20),20:PAINT (20,20),12,15
30 COPY (0,0)-(40,40) TO "TEST":CLS
40 DX=4:DY=2:X=10:Y=10
50 COPY "TEST" TO (X,Y),,XOR
60 X=X+DX:Y=Y+DY
70 IF X>200 OR X<10 THEN DX=-DX
80 IF Y>180 OR Y<10 THEN DY=-DY
90 GOTO 50
RUN
```

In het grote handboek zagen we, dat we met COPY ook in staat zijn om een gedeelte van een grafisch beeldscherm naar een NUMERIEKE TABELVARIABELE kunnen kopiëren. Later kan men deze tabel dan weer naar het scherm kopiëren. In feite verplaatsen we op deze wijze een gedeelte van het videogeheugen naar het centrale computergeheugen en vice versa.

Ook een numerieke tabelvariabele kunnen we met behulp van COPY naar een bestand op schijf kopiëren. In het onderstaande voorbeeld wordt tabelvariabele B naar bestand TEST gekopieerd.

```
NEW
Ok
10 DIM B(100)
20 FOR I=0 TO 100:B(I)=RND(1):NEXT
30 COPY B TO "TEST"
RUN
```

In het nu volgende voorbeeld wordt tabel B weer vanuit bestand TEST gevuld.

```
NEW
Ok
10 DIM B(100)
20 COPY "TEST" TO B
30 FOR I=0 TO 100:PRINT B(I);:NEXT
RUN
 .59521943994623  .10658628050158  .76597651772823  .57756392935958
 .73474759503023  .18426812909758  .37075377905223  .94954151651558
 .63799556899423  .47041117641358  .83212197605623  .49796192159158
 .18586284343823  .12951037284958  .49057636634023  .77256783898758
 .02818381196223  ... (etcetera)
```

Het is natuurlijk niet verplicht om de laatste twee vormen van het COPY-kommando alleen maar voor grafische doeleinden te gebruiken! Op prima wijze kunnen we dit kommando gebruiken om hele tabellen tegelijk op schijf te zetten en van schijf te halen.

Wanneer TABELVARIABELEN van en naar schijf worden gekopieerd, dienen deze variabelen wel te zijn gedimensioneerd met behulp van bijvoorbeeld het DIM-kommando. Indien de tabelvariabele waarnaartoe wordt gekopieerd, te klein is, dan wordt slechts een gedeelte van de waarden gekopieerd.

moeilijkheidsgraad . . zeer moeilijk, kennis van de fysische organisatie op schijf is een vereiste
soort . . . . . KOMMANDO
afkomst . . . . . DSKO is een afkorting van DISK OUTPUT

schrijfwijze

```
DSKO$<SCHIJFNUMMER>,<SECTORNUMMER>

<SCHIJFNUMMER>::=<N>

<SECTORNUMMER>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Het DSKO$-kommando is het tegenovergestelde van DSKI$. Met het DSKO$-kommando kunnen we een sector op schijf SCHRIJVEN, ongeacht de logische indeling van de schijveneenheid.

Onnodig te zeggen dat een verkeerd gebruik van DSKO$ al heel snel kan resulteren in het 'opblazen' van de gehele schijf. Oefen dit kommando slechts op een schijf zonder waardevolle gegevens en formatteer deze schijf na afloop voor de zekerheid.

ACHTER het kommando DSKO$ dienen twee waarden te worden opgenomen. De eerste waarde geeft aan, op welke schijveneenheid wordt geschreven. Een nul heeft tot resultaat dat de laatst aangesproken schijveneenheid wordt geselekteerd. Een 1 staat voor de eerste schijveneenheid, een 2 voor de tweede, enzovoorts.

Het tweede gegeven achter DSKO$ geeft het sectornummer aan van de sector die gaat worden geschreven. Sectornummers beginnen bij nul. Het hoogste sectornummer wordt bepaald door het type schijveneenheid.

Het onderstaande voorbeeld maakt gebruik van DSKI$ en DSKO$ en stelt u in staat om SECTORGEWIJS een kopie van een originele schijf te maken.

Sectorgewijs kopiëren heeft als nadeel, dat het een en ander een stuk langzamer gaat. Een voordeel is, dat werkelijk ALLE gegevens van de schijf worden gekopieerd! Wanneer men een logisch beschadigde schijf wil gaan repareren, kan het zijn dat dit programma de enige mogelijkheid biedt om van deze schijf eerst een veiligheidskopie te maken...

```
NEW
Ok
10 P1=PEEK(62289!)+256*PEEK(62290!)
20 A$="":P2=VARPTR(A$)
30 POKE P2,128:POKE P2+1,P1-256*(INT(P1/256)):POKE P2+2,INT(
P1/256)
40 S$=DSKI$(1,0)
50 N=ASC(MID$(A$,20))+256*ASC(MID$(A$,21))
60 PRINT "PLAATS ORIGINELE SCHIJF IN A:"
70 PRINT "EN KOPIE IN B: GEEF RETURN"
80 IF INKEY$<>CHR$(13) THEN 80
90 FOR I=0 TO N-1:S$=DSKI$(1,I)
100 DSKO$ 2,I:NEXT I
110 PRINT "KLAAR!"
RUN
PLAATS ORIGINELE SCHIJF IN A:
EN KOPIE IN B: GEEF RETURN
```

Met bovenstaand programma is het te hopen dat u twee schijveneenheden heeft. In het andere geval moet u namelijk vele malen steeds de schijven verwisselen.

moeilijkheidsgraad . . zeer moeilijk- kennis van de fysische organisatie op schijf is een vereiste
soort . . . . . SYSTEEMVARIABELE
afkomst . . . . . DSKI is een afkorting van DISK INPUT

schrijfwijze

```
DSKI$(<SCHIJFNUMMER>,<SECTORNUMMER>)

<SCHIJFNUMMER>::=<N>

<SECTORNUMMER::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het DSKI-kommando kunnen we sectorgewijs de schijf bestuderen. Onafhankelijk van de logische organisatie op de schijf geeft DSKI$ u de inhoud van elke gewenste sector op schrijf, gebruikt of ongebruikt.

Tussen haakjes dienen we aan het SSKI$-bevel twee waarden mee te geven. De eerste waarde geeft aan, op welke schijveneenheid de betreffende sector moet worden ingelezen. Een nul heeft tot resultaat dat de laatst geadresseerde schijveneenheid wordt genomen. Een 1 staat voor de eerste schijveneenheid, een 2 voor de tweede enzovoorts.

Het tweede getal tussen haakjes geeft aan, welke sector moet worden ingelezen. De nummering van de sectoren begint bij nul en eindigt bij een voor elke schijveneenheid wisselende waarde, afhankelijk van de opslagcapaciteit en sectorgrootte van deze eenheid.

Er zit een addertje onder het gras...

Wanneer men programmeert:

```
10 A$=DSKI$(1,4)
```

dan zou men verwachten dat A$ gelijk gesteld zou worden aan de inhoud van sector 4 van de eerste schijveneenheid. Niets is echter minder waar!

Wat er wél gebeurt is, dat in een buffer in het systeemgedeelte van de MSX-computer de sector wordt ingelezen. Het adres van het eerste byte uit deze buffer is gelijk aan PEEK(62289) + 256*PEEK(62290). Het is dus nog best vrij ingewikkeld om de inhoud van een sector te benaderen!

Enkele specifieke gegevens van de gebruikte schijveneenheid worden tijdens het formatteren automatisch in sector 0 van de schijf geschreven.

Het volgende voorbeeld geeft u de sectorgrootte in bytes, het aantal sectoren per schijf en de totale schijfcapaciteit van de door u gebruikte eenheid:

```
NEW
Ok
10 P1=PEEK(62289!)+256*PEEK(62290!)
20 A$="":P2=VARPTR(A$)
30 POKE P2,128:POKE P2+1,P1-256*(INT(P1/256)):POKE P2+2,INT(P1
/256)
40 S$=DSKI$(1,0):B=ASC(MID$(A$,12))+256*ASC(MID$(A$,13))
50 N=ASC(MID$(A$,20))+256*ASC(MID$(A$,21))
60 PRINT "UW EENHEID HEEFT";N;"BLOKKEN VAN";B;"BYTES"
70 PRINT "TOTALE CAPACITEIT:";B*N/1024;"KILOBYTES"
RUN
UW EENHEID HEEFT 720 BLOKKEN VAN 512 BYTES
TOTALE CAPACITEIT: 360 KILOBYTES
Ok
```

Het volgende voorbeeld geeft u een hexadecimale representatie van de eerste 128 bytes uit sector 0, de sector waarin diverse eigenschappen van de gebruikte schijveneenheid zijn gekodeerd:

```
NEW
Ok
10 P1=PEEK(62289!)+256*PEEK(62290!)
20 A$="":P2=VARPTR(A$)
30 POKE P2,128:POKE P2+1,P1-256*(INT(P1/256)):POKE P2+2,INT(
P1/256)
40 S$=DSKI$(1,0)
50 PRINT "  !00!01!02!03!04!05!06!07!"
60 PRINT "  !08!09!0A!0B!0C!0D!0E!0F!"
70 PRINT "--!--!--!--!--!--!--!--!";
80 FOR I=0 TO 127:IF I MOD 8=0 THEN PRINT:PRINT RIGHT$("0"+H
EX$(I),2);"!";
90 PRINT RIGHT$("0"+HEX$(ASC(MID$(A$,I+1))),2);"!";
100 NEXT I:PRINT:PRINT "--------------------------"
RUN
```

```
  !00!01!02!03!04!05!06!07!
  !08!09!0A!0B!0C!0D!0E!0F!
--!--!--!--!--!--!--!--!--!
00!EB!FE!90!41!53!43!20!20!
08!32!2E!32!00!02!02!01!00!
10!02!70!00!D0!02!F8!02!00!
18!09!00!01!00!00!00!D0!ED!
20!53!59!C0!32!D0!C0!36!56!
28!23!36!C0!31!1F!F5!11!AB!
30!C0!0E!0F!CD!7D!F3!3C!CA!
38!63!C0!11!00!01!0E!1A!CD!
40!7D!F3!21!01!00!22!B9!C0!
48!21!00!3F!11!AB!C0!0E!27!
50!CD!7D!F3!C3!00!01!58!C0!
58!CD!00!00!79!E6!FE!FE!02!
60!C2!6A!C0!3A!D0!C0!A7!CA!
68!22!40!11!85!C0!CD!77!C0!
70!0E!07!CD!7D!F3!18!B4!1A!
78!13!B7!C8!D5!5F!0E!06!CD!
---------------------------
```

7 OPERATING SYSTEMS

Het MSX DOS is een operating system voor de MSX computer. De gebruiker die dit leest, vraagt zich natuurlijk onmiddellijk af wat nu eigenlijk een operating system is. In dit hoofdstuk wordt uit de doeken gedaan wat de algemene principes van een operating system zijn en wat de achtergrond van het verschijnsel operating system is.

7.1 Operating systems overal

Iedereen die wel eens met een computer te maken heeft gehad, hoe klein ook, heeft al eens te maken gehad met een operating system. Met een beetje fantasie zou men zelfs kunnen beweren dat de eenvoudigste rekenmachine al een operating system heeft.

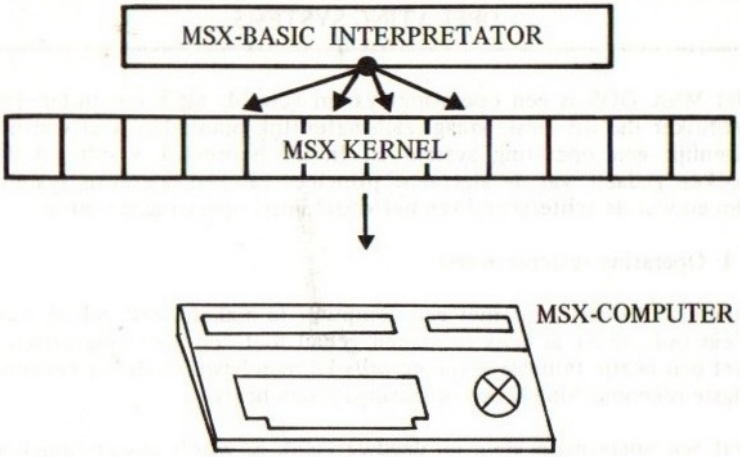
Wat een operating system nu daadwerkelijk is, wordt al wat duidelijk wanneer we de minder gebruikelijke nederlandse vertaling van het woord operating system nader beschouwen:

OPERATING SYSTEM = BESTURINGSSYSTEEM

De volgende definitie van een operating system is misschien niet helemaal correct maar in ieder geval wel duidelijk:

Een operating system (besturingssysteem) is een stuk programmatuur dat gedurende de werking van een computer altijd (gedeeltelijk) aanwezig is in het computergeheugen en dat de coördinatie van de verschillende computeronderdelen op het allerlaagste niveau verzorgt.

Wanneer we een eenvoudige MSX computer zonder schijveneenheid beschouwen, kunnen we wat betreft de aanwezige programmatuur in het ROM geheugen (Read Only Memory = permanent, alleen uitleesbaar geheugen) het volgende schema tekenen:



In het ROM geheugen van de computer onderscheiden we twee verschillende niveaux van (machinetaal-) software:

1. de MSX BASIC INTERPRETATOR. Deze software controleert de door u ingetoetste BASIC-kommando's op juistheid. Goed bevonden kommando's worden bij het uitvoeren van een door u samengesteld MSX basic programma door de interpretator daadwerkelijk uitgevoerd. De interpretator INTERPRETEERT in feite het door u ingetoetste BASIC programma en vertaalt 'uw woorden' naar aanwijzingen die door de computer kunnen worden begrepen.

2. het MSX KERNEL. Het MSX kernel wordt gevormd door een verzameling van tientallen meestal vrij kleine programma's. Deze programma's dragen elk zorg voor de daadwerkelijke besturing van een klein onderdeel van het totale computersysteem. Zo is er één programma dat de afhandeling met cassetterecorder verzorgt. Ook is er een programma dat de ingave via het toetsenbord mogelijk maakt. Weer een andere routine uit het MSX kernel draagt zorg voor het trekken van een lijn over het beeldscherm. Bij elkaar vormt het kernel de totale besturing van de MSX computer. De MSX basic interpretator die 'daar boven' staat, gebruikt deze routines uit het kernel en geeft deze de nodige gegevens mee. Het 'intelligente' werk wordt eerst door de MSX basic interpretator uitgevoerd waarna de elementaire akties door het kernel worden uitgevoerd.

Wanneer u auto rijdt, kunt u zichzelf vergelijken met de MSX basic-interpretator. Het intelligente werk van koppelen, gas geven, sturen, remmen, uitwijken en dergelijke wordt door u gedaan. Uw auto vormt in deze vergelijking het kernel. De akties die u onderneemt, worden door de auto uiteindelijk effektief gemaakt. U draait aan het stuur; de auto verandert van richting. U haalt een handle over; het elektrische systeem van uw auto zorgt ervoor dat er richting wordt aangegeven.

Samenvattend kunnen we stellen dat het kernel de mogelijkheden biedt om de computer te kunnen besturen. De software van het tweede niveau bestuurt de computer daadwerkelijk door van de kernel gebruik te maken.

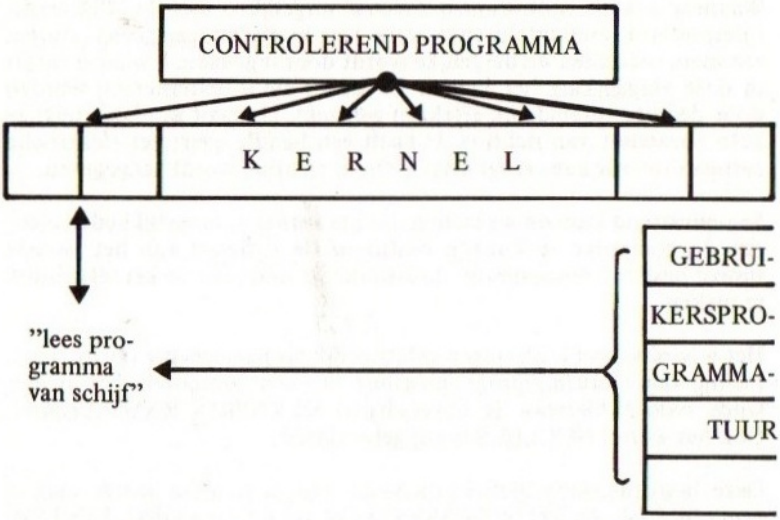
Het woord kernel is algemeen gebruikelijk als naamgeving van de verzameling van besturingsprogrammatuur van een computer. Een nederlands woord hiervoor is bijvoorbeeld KERNPROGRAMMATUUR. Ook het woord NUCLEUS is vrij gebruikelijk.

Deze besturingsmogelijkheid biedende programmatuur wordt vaak al aangeduid als BESTURINGSSYSTEEM of OPERATING SYSTEM. Als zodanig kan worden gesteld dat elke computer, hoe klein ook, een operating system heeft. Daar dit operating system meestal nogal verweven is met de basic-interpreter, wordt een dergelijk operating system ook wel een intrinsiek operating system genoemd. In de volgende paragraaf zullen we zien, dat er eigenlijk nog een element nodig is om daadwerkelijk van een operating system te kunnen spreken.

7.2 Een echt besturingssysteem

In de vorige paragraaf zagen we, dat in feite in elke computersysteem een operating system aanwezig is. Dit operating system bestaat uit een verzameling van besturende machinetaalprogramma's zonder dat er sprake is van een daadwerkelijke samenhang tussen deze programma's. De basic-interpreter die zich meestal op het tweede, coördinerende niveau bevindt, zorgt voor een samenhang van deze stukjes systeem en maken het operating system zinvol.

Een 'echt' operating system kan men zich schematisch als volgt voorstellen:



De taakstelling van een operating system kan men als volgt zien:

- een operating system biedt een NUCLEUS, een verzameling van programmatuur waarmee de onderdelen van de computer kunnen worden bestuurd.

- een operating system biedt de mogelijkheid om elementaire akties met betrekking tot het computersysteem te ondernemen. Deze elementaire akties worden meestal betiteld als het ONDERHOUD van het systeem.

- een operating system biedt de mogelijkheid om controle met behoud van de NUCLEUS over te dragen aan een gebruikersprogramma.

Het controlerende programma, dat zelf ook uitgebreid gebruik maakt van de nucleus, biedt de mogelijkheid om het computersysteem te onderhouden. Zo kunnen bestanden met dit controlerende programma worden ingericht, verwijderd, gekopiëerd en kunnen bijvoorbeeld ook schijven worden geformatteerd. Ook, en dit is één van de belangrijkste mogelijkheden van het controlerende programma, heeft het de mogelijkheid om zichzelf 'op te offeren' ten gunste van een ander programma. Via de betreffende besturing in het kernel kan een ander

gebruikersprogramma (een basic interpreter, een pascal compiler, een tekstverwerker enzovoorts) in de plaats van het controlerende programma worden gezet terwijl de besturende software, de kernel, aanwezig blijft.

Het operating system cijfert zichzelf dus gedeeltelijk weg, maar laat een gedeelte van zichzelf (de kernel) staan voor gebruik door het vervangende programma. Wanneer dit vervangende programma klaar is met de uit te voeren taken, kan het controlerende programma met behulp van een routine uit de kernel weer worden geplaatst.

7.3 DOS

Alhoewel in elke hobbycomputer wel een operating system is aan te wijzen, komt het verschijnsel operating system als een op zichzelf staande grootheid uit de professionele computerwereld. Immers, een groot en duur computersysteem 'hangt' men in deze wereld niet aan één computertaal op. Het enige dat een professioneel computersysteem moet bieden, is een uitstekend operating system. Van dit operating system kunnen de software fabrikanten dan gebruik maken voor het ontwerpen van de benodigde computertalen en andere gebruikerssoftware.

Een professioneel computersysteem zonder schijveneenheid is, behoudens enkele specifieke besturingstoepassingen, nauwelijks voor te stellen. Het spreekt dan ook voor zich dat een operating system meestal voorziet in de mogelijkheid tot besturing van één of meer schijveneenheden. Sterker nog, de meeste operating systems zijn volledig gebaseerd op het gebruik van een schijveneenheid.

Wanneer, en dit is in de meeste gevallen zo, een operating system is gebaseerd op het gebruik van minstens één schijveneenheid, noemt men een dergelijk operating system meestal een Disk Operating System, afgekort DOS.

Het MSX DOS is een speciaal voor de MSX computer door Microsoft ontwikkeld disk operating system.

In het vorige hoofdstuk zagen we, wat een Disk Operating System (DOS) is.

Het MSX DOS is een disk operating system, speciaal voor MSX-computers ontworpen door Microsoft, een software-fabrikant van wereldfaam op de wat professionelere microcomputer.

Een goede software-producent vindt niet steeds opnieuw het wiel uit. Steeds maakt hij gebruik van eerdere produkten en ervaringen om de volgende, nieuwere produkten beter, foutlozer, sneller, logischer en doordachter te maken.

Op deze wijze stelde Microsoft behalve het MSX basic en het MSX DISK basic ook het MSX DOS operating system samen, gebruik makend van professioneel software-materiaal en jarenlange ervaringen op professioneel gebied.

8.1 De afkomst van MSX DOS

Microsoft heeft al jarenlang een befaamdheid opgebouwd met het ontwikkelen van BASIC-dialecten voor diverse computersystemen. Het Microsoft Basic is voor zeer vele mensen een begrip. Voor zeer vele hobbycomputer-fabrikanten ontwikkelde Microsoft de bijbehorende BASIC-dialecten en baseerde zich hierbij steeds op de centrale taal, het Microsoft basic. Ook het MSX basic is hiervan afgeleid.

Toen enige tijd geleden de 16-bits computers in zwang kwamen, ontwikkelde Microsoft voor dit type computer een eigen operating system; het MS DOS (MicroSoft Disk Operating System). Met dit operating system stak Microsoft haar tegenpool, Digital Research, voor het eerst op operating system gebied de loef af. Tot dan toe gold Digital Research op 8-bits computers als de toonaangevende operating system fabrikant. Het operating system van Digital Research, CP/M (Control Program for Microprocessors) is echter, vooral op het gebied van de 8-bits computers, nog steeds een zeer wijdverbreid operating system. Onder CP/M is nog steeds een gigantische hoeveelheid software op verschillende gebieden te verkrijgen.

Het MSX-DOS is een afgeleide van het MS DOS, speciaal bedoeld voor MSX computers. MSX-DOS is dus op professionele leest geschoeid.

Echter, Microsoft ging bij het samenstellen van MSX-DOS nog een stap verder. Door de struktuur van het KERNEL voor een belangrijk gedeelte gelijk te houden aan de struktuur van het kernel van CP/M schiep Microsoft de mogelijkheid om de gigantische hoeveelheid onder CP/M verkrijgbare, professionele software ook onder MSX-DOS te gebruiken. Vaak moeten hiervoor door de software-fabrikanten kleine wijzigingen worden aangebracht; een enkele keer werkt een CP/M programma onmiddellijk onder MSX-DOS.

Met het MSX-DOS operating system krijgt de MSX computer dus plotseling een toegang tot het immense gebied van CP/M software...*

* Alhoewel het MSX DOS bij het ter perse gaan van dit boek nu al véél meer dan een jaar op de markt is, blijft het aanbod van deze software bijzonder beperkt. Dit is deels te wijten aan het feit dat de software-fabrikanten er nog geen brood in zien om hun programmatuur voor MSX-computers geschikt te maken. Met de opkomst van de MSX-2 computers is het te verwachten dat hierin verandering gaat komen...

9 HET NUT VAN EEN OPERATING SYSTEM

In de vorige hoofdstukken behandelden we het begrip operating system. Nu we precies weten wat een operating system is, is het misschien zinvol om in te gaan op het nut van de aanwezigheid van het MSX DOS op de MSX computer.

9.1 Compatibiliteit

Compatibiliteit is een woord dat zoveel wil zeggen als: onderlinge toepasbaarheid. Indien twee computers met betrekking tot een programma compatible zijn, betekent dat dat het betreffende programma op beide computers zonder wezenlijke wijzigingen kan worden uitgevoerd. De mate van compatibiliteit wordt bepaald door het aantal veranderingen dat moet worden aangebracht op een programma om het ook op een ander systeem te kunnen laten draaien. Een hoge compatibiliteit houdt in, dat slechts een enkele wijziging mogelijk moet worden aangebracht.

Door gebruik te maken van een kernel met een vaste, afgesproken struktuur, schept men de mogelijkheid om programma's samen te kunnen stellen die op verschillende computersystemen kunnen werken. Voor de daadwerkelijke besturing van de computeronderdelen (randapparaten) worden immers de kernel-routines gebruikt. Deze kernelroutines kunnen per merk computersysteem verschillen; het programma dat deze kernel gebruikt, kan voor elke systeem gelijk blijven.

Zo is bijvoorbeeld software die geschreven is voor het CP/M operating system, op honderden verschillende computersystemen te gebruiken. Al deze computersystemen maken namelijk gebruik van het CP/M operating system en hebben elk een eigen kernel dat machine-afhankelijk maar altijd overeenkomstig van struktuur is.

Het voordeel hiervan is, dat software, geschreven voor de ene machine, ook op een andere machine kan worden gebruikt. Software-fabrikanten kunnen op deze wijze verzekerd zijn van een grote schare van afnemers en als zodanig zonder veel risico software gaan ontwikkelen. Voor de gebruiker heeft dit weer het voordeel dat het aanbod van software bijzonder goed van kwaliteit en bijzonder ruim van afmeting is.

We kunnen over het algemeen dus stellen dat gebruik van een operating system, vooral van een standaard veel gebruikt operating system, direkt

leidt tot een groot en kwalitatief uitstekend software-aanbod. Doordat het aanbod zo hoog is en de fabrikanten kunnen rekenen op een behoorlijke afzet, blijven de prijzen meestal erg vriendelijk.

9.2 MSX-DOS en CP/M

We zagen reeds dat het CP/M operating system nog steeds een zeer wijdverbreid operating system is. In de afgelopen jaren is er een immens aanbod aan software ontstaan voor CP/M computers.

Omdat het MSX-DOS op kernel-niveau een hoge mate van compatibiliteit heeft met CP/M, is de meeste van deze software na een kleine aanpassing bruikbaar onder MSX-DOS.

Een bijzonder sterk en fraai voordeel van MSX-DOS is dus, dat dit operating system een deur opent naar een reeds bestaande, immens grote hoeveelheid professionele software. Zo kan de MSX-DOS gebruiker kiezen tussen diverse, zeer professionele tekstverwerkers of databases. Ook bedrijfssoftware (financiële administraties, faktureerprogramma's, voorraadcontroleprogramma's, etcetera) is te kust en te keur te verkrijgen. Ook de professionele gebruiker kan met zijn MSX systeem dus alle kanten uit.

9.3 Machine-onafhankelijkheid

Professionele software blijft duur. Hoe de concurrentieverhoudingen ook liggen; voor een goede database moet altijd nog een bedrag van enkele honderden guldens worden neergeteld. Nog veel duurder is natuurlijk de maatsoftware die men speciaal voor een bepaald doel door een software-fabrikant laat ontwikkelen.

Het is dan ook altijd een pijnlijke zaak om op een gegeven moment te moeten constateren dat men bij aanschaf van een nieuw, groter computersysteem verplicht is om ook totaal nieuwe software aan te schaffen. Niet alleen het aanschafbedrag vormt een struikelblok; ook de bediening van die andere software, het aanpassen van reeds aanwezige gegevens en dergelijke zaken vormen een probleem.

Wanneer men gebruik maakt van operating system dat algemeen in zwang is, vallen veel bezwaren weg. Doordat bij verandering van computer het operating system gelijk blijft, behoeft geen nieuwe software te worden aangeschaft maar behoeft de oude software slechts naar het nieuwe systeem te worden overgebracht.

Over het algemeen kan men dus stellen dat gebruik van een operating system er zorg voor draagt, dat bij uitbreiding of vervanging van apparatuur de software behouden kan blijven.

In paragraaf 3.8 zagen we, wat een bestand is en welke soorten we onder MSX-DISK-BASIC onderscheiden. Het MSX-DOS kent een andere indeling van bestanden of files, namelijk:

1. COMMAND FILES. Deze bestanden onderscheiden zich door het achtervoegsel .COM aan de naam van het bestand (zie hoofdstuk 12). Een commando file bevat een machinetaalprogramma en wordt door MSX DOS als kommando beschouwd. De gegevens uit een command file vormen bij elkaar dus een programma dat door MSX DOS direkt kan worden uitgevoerd zonder dat eerst het BASIC behoeft te worden geladen. Een command file is vaak het resultaat van een compilatie van bijvoorbeeld een PASCAL programma of van een assembleer-run. De professionele gebruikerssoftware die onder MSX DOS wordt aangeboden (databases, tekstverwerkers, compilers, enz.). wordt meestal in de vorm van één of meer command files aangeboden.

2. BATCH FILES. Deze bestanden onderscheiden zich door het achtervoegsel .BAT aan de naam van het bestand (zie hoofdstuk 12). Een batch file bevat een hoeveelheid kommando's zoals deze in hoofdstuk 13 worden behandeld en die direkt door het MSX DOS worden 'begrepen'. In een batch file kunnen opeenvolgingen van MSX DOS-kommando's die erg vaak voorkomen, worden gegroepeerd onder één naam. Zo kunnen onder MSX DOS door de gebruiker zelf nieuwe, complexe kommando's worden samengesteld. Een uitgebreide behandeling van de batch file volgt in hoofdstuk 14.

3. SYSTEEM FILES. Deze bestanden onderscheiden zich door het achtervoegsel .SYS aan de naam van het bestand (zie hoofstuk 12. Een systeem file bevat gegevens die van direkt belang zijn voor het MSX DOS zelf; de gebruiker krijgt met dit soort bestanden meestal niet te maken.

4. Overige bestandssoorten. MSX DOS maakt geen onderscheid tussen de overige nog mogelijke bestandssoorten. Het MSX DISK basic doet dit wel: met name maakt het MSX DISK basic een duidelijk onderscheid tussen RANDOM files en SEQUENTIAL files. Ook onderscheidt MSX DISK basic duidelijk de PROGRAM file van de DATA file. Het MSX DOS doet dit niet; alle overige bestandssoorten worden op een zelfde wijze behandeld.

10.1 Bestandgrootten

De gebruiker van het MSX DOS zal op geen enkel gebied de noodzaak tegenkomen, de GROOTTE van een bestand te vermelden. Dit komt omdat het MSX DOS een zogenaamde dynamische behandeling van bestanden kent. Een bestand wordt altijd zo groot toegewezen als nodig is en kan later altijd worden aangevuld. De enige grens aan de grootte van een bestand wordt uiteindelijk gevormd door de opslagcapaciteit van de magneetschijf zelf.

11 HET OPSTARTEN VAN MSX-DOS

In de voorgaande hoofdstukken behandelden we uitgebreid het onderwerp operating systems in het algemeen en MSX-DOS in het bijzonder.

In de volgende hoofdstukken gaan we in op de praktijk van het MSX DOS.

11.1 Volgorde van handelingen

Om het MSX-DOS operating system op te starten, hebben we de volgende dingen nodig:

- een MSX computer
- een aangekoppelde schijveneenheid (geen Quick Disk)
- een schijfje met daarop het MSX-DOS operating system

Het MSX-DOS is als volgt op te starten:

- plaats het schijfje met het MSX DOS operating system in de schijveneenheid. Wanneer de betreffende schijveneenheid een vergrendeling heeft, laat deze dan nog even open. Wanneer geen vergrendeling aanwezig is, plaats het schijfje dan niet helemaal maar laat het nog iets uitsteken.

- schakel vervolgens de schijveneenheid in.

- schakel vervolgens de MSX computer (en beeldscherm) in.

- vergrendel nu onmiddellijk de schijveneenheid of plaats de schijf volledig in de schijveneenheid.

Na enige seconden gaat de schijveneenheid werken en vrij snel volgt op beeld de melding:

```
MSX-DOS version X.XX
Copyright 1984 by Microsoft

Command version X.XX
```

X.XX staat hierbij voor het versienummer van de MSX-DOS-componenten. MSX-1 systemen vragen na deze melding om datum en tijd; door twee maal alleen de RETURN-toets in te geven, slaan we voor-

lopig de ingave van datum en tijd over. Zie voor de juiste ingaven de behandeling van het kommando DATE en TIME in hoofdstuk 8.

Uiteindelijk verschijnt de melding:

```
A>
```

Deze melding geeft aan welke schijveneenheid op dit moment aktief is. De letter A, meestal de PROMPT genoemd, staat voor de eerste aangekoppelde schijveneenheid, de letter B voor de tweede, enzovoorts.

Het)-teken geeft aan dat we een kommando kunnen ingeven.

Wat we zojuist deden, is het opstarten van het MSX DOS operating system. Dit opstarten wordt ook wel het 'booten' van het systeem genoemd. We onderscheiden de zogenaamde COLD BOOT (koude start) waarbij het systeem wordt opgestart door het aan te zetten en de WARM BOOT waarbij het systeem hernieuwd wordt opgestart zonder het systeem eerst uit te zetten.

Wanneer een gebruikersprogramma ten einde is en het controlerend programma weer binnenhaalt, voert dit programma een zogenaamde WARM BOOT uit. Door middel van het tegelijk indrukken van de toetsen CTRL en C (of CTRL en STOP) forceert men eveneens een WARM BOOT; de prompt wordt hernieuwd afgedrukt.

11.2 De eerste, eenvoudige handelingen

Nu het operating system met succes is opgestart, kunnen we de eerste eenvoudige besturing met behulp van dit operating system uitvoeren.

Eén van de meest eenvoudige besturingen is het veranderen van aktieve schijveneenheid. Na een COLD BOOT is altijd schijveneenheid A aktief. Wil men echter schijveneenheid B aktief maken, dan kan dat door ingave van:

```
A>B:
```

Het systeem reageert meteen met de prompt:

```
B>
```

als teken dat de tweede schijveneenheid nu aktief is. Zo kunnen in principe zesentwintig schijveneenheden met alle letters van het alfabet worden geaktiveerd. Deze besturingsmogelijkheid is meestal zinloos bij het gebruik van slechts één schijveneenheid.

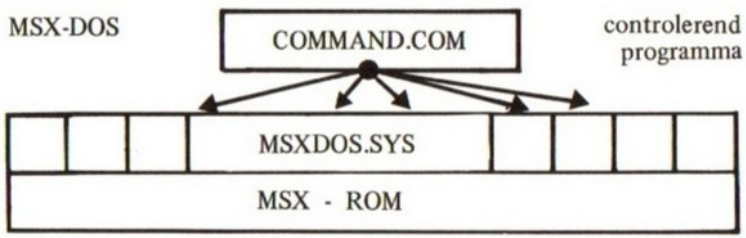
De inhoud van een schijf kan eenvoudig als volgt worden opgevraagd:

```
B>A:                                    (eerste eenheid weer actief maken)
A>DIR                                   (afkorting van DIRectory=index)
MSXDOS   SYS    2432  1-01-1984         (inhoud wordt geprojekteerd)
COMMAND  COM    6272  1-01-1984
TEST             312 10-15-1985 12:00p (MSX-2 geeft ook de tijd)
       3 files  302643 bytes free       (nog vrije schijfruimte)
A>
```

In het bovenstaande voorbeeld zijn de inhoud en de getalswaarden willekeurig; op uw systeem kunnen hele andere bestanden, andere data en andere grootten worden gegeven. Eén ding staat echter vast; twee bestanden staan in ieder geval in de zojuist opgevraagde directory:

MSXDOS.SYS en COMMAND.COM

In deze twee bestanden is het volledige operating system opgenomen. In hoofdstuk 4 behandelden we de opbouw van een operating system; welnu, het in dat hoofdstuk geschetste schema kunnen we als volgt invullen:



De werkelijke kernel van MSX-DOS staat voor het grootste gedeelte in het ROM (read only memory) van de MSX computer. Van ditzelfde kernel maakt ook het MSX basic bijvoorbeeld gebruik.
Het bestand MSX-DOS. SYS is een aanvulling op dit kernel. Met name maakt dit gedeelte van het kernel het ROM-geheugen van de MSX computer bereikbaar voor MSX-DOS.

Het bestand COMMAND.COM is het zogenaamde controlerende programma. Dit programma staat toe dat het vervangen wordt door een ander programma. Ook maakt het het eenvoudige systeemonderhoud (bestanden veranderen/aanmaken/verwijderen, systeemdatum veranderen etc.) mogelijk.

We zien dat deze twee bestanden meteen voorbeelden zijn van een SYSTEM FILE en een COMMAND FILE (hoofdstuk 3).

Dat COMMAND.COM een COMMAND FILE is, kunnen we aantonen door de volgende proefneming. Tik in:

```
A>COMMAND
```

De schijveneenheid gaat even werken waarna de prompt weer verschijnt. De opdracht COMMAND resulteerde hierin, dat het controlerend programma (COMMAND.COM) werd vervangen door een andere COMMAND FILE. Deze andere command file bleek hetzelfde controlerende programma te zijn; COMMAND.COM werd slechts herladen en opnieuw in werking gesteld. In feite voerden we zojuist een WARM BOOT uit.

12 BESTANDSAANDUIDINGEN

Het MSX-DOS is een DISK operating system. De onderhoudsakties die via het MSX-DOS kunnen worden uitgevoerd, bestaan dan ook hoofdzakelijk uit akties met betrekking tot de schijveneenheid. Zo kunnen bijvoorbeeld de volgende akties standaard onder MSX-DOS worden ontplooid:

- het kopiëren van bestanden
- het veranderen van bestandsnamen
- het verwijderen van bestanden
- het opvragen van bestandsspecificaties.

Om bij het manipuleren met bestanden aan te duiden om welk bestand of om welke groep van bestanden het gaat, leerden we reeds het begrip BESTANDSAANDUIDING kennen. Daarin onderscheidden we ondermeer de TYPERING.

System files, batch files en command files kennen een eigen, aparte typering. Het is raadzaam om deze typeringen alleen voor de bijbehorende bestandssoorten te gebruiken. De typeringen zijn als volgt:

–een command file	XXXXXXXX.COM	(bijv. COMMAND.COM)
–een system file	XXXXXXXX.SYS	(bijv. MSXDOS.SYS)
–een batch file	XXXXXXXX.BAT	

13.1 De intrinsieke kommando's van MSX-DOS

Het operating system heeft onder meer tot taak, het onderhoud van het betreffende computersysteem mogelijk te maken. Hiertoe dienen enige bevelen aan wezig te zijn waarmee dit onderhoud kan geschieden.

De direkt bij het operating system betrokken kommando's die niet zondermeer van dit operating system zijn los te maken, noemt men de intrinsieke kommando's van dat operating system.

In deze paragraaf worden de intrinsieke kommando's van MSX-DOS stuk voor stuk behandeld.

Per kommando is een schrijfwijze, een toelichting op de betekenis en een voorbeeld opgenomen.

De schrijfwijze is op de gebruikelijke wijze in BNF opgenomen.

schrijfwijze

```
BASIC [<PROGRAMMANAAM>]

<PROGRAMMANAAM>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Dit kommando is het eenvoudigste MSX-DOS kommando. Door eenvoudigweg na de schijfaanduiding het woord:

```
A>BASIC
```

in te toetsen, verschijnt de MSX-BASIC melding en is het MSX-BASIC aktief geworden.

Door onder MSX-BASIC de volgende CALL in te toetsen:

```
CALL SYSTEM
```

kan vanuit MSX-BASIC het MSX-DOS weer worden geactiveerd.

Het kommando BASIC zorgt ervoor, dat het 32 kilobytes grote ROM-geheugen (ROM=read only memory, onveranderbaar, alleen leesbaar geheugen) waarin het MSX-BASIC ligt opgeslagen, wordt geactiveerd. Hiertoe dient de eerste 32 kilobytes aan RAM-geheugen (RAM=random access memory, vrij toegankelijk geheugen), dat onder MSX-DOS actief is, te worden afgekoppeld. Bij CALL SYSTEM gebeurt natuurlijk precies het omgedraaide.

Wanneer achter het kommando BASIC een programmanaam wordt opgegeven, dan wordt het betreffende basic-programma onmiddellijk na het activeren van het MSX-BASIC uitgevoerd. Het programma SCREEN.BAS dat in het boek is opgenomen, (batch-files) kan vanuit MSX-DOS onmiddellijk worden opgestart met het kommando:

```
A>BASIC SCREEN.BAS
```

schrijfwijze

```
FORMAT
```

Met het FORMAT-kommando kunnen schijven worden geformatteerd. De betekenis en noodzaak van dit formatteren kwamen we reeds tegen in hoofdstuk 2. Daar zagen we ondermeer dat een nieuwe schijf altijd eerst een keer moet worden geformatteerd voordat deze kan worden gebruikt.

Pas op: een schijf verliest bij het formatteren alle daarop aangebrachte gegevens. Een gouden regel is dan ook om, hoe zeker u ook van uw zaak bent, voordat u daadwerkelijk gaat formatteren, twee maal te controleren of u de juiste schijf in de betreffende eenheid heeft geplaatst. Indien u meerdere schijfeenheden heeft, ontgrendel dan de overige eenheden zodat een eventueel daarin geplaatste schijf niet per ongeluk kan worden geformatteerd.

Nadat het kommando FORMAT is ingegeven, vraagt de computer om een specificatie van de schijveneenheid waarin de te formatteren schijf is geplaatst. Geef voor de eerste schijveneenheid de letter A in, voor de tweede schijveneenheid de letter B, enzovoorts. Indien u maar één schijveneenheid heeft, geeft u het beste altijd een A in.

Het formatteren van een schijf kan enkele minuten duren. Uiteindelijk geeft de computer de melding FORMAT COMPLETE wanneer het formatteren naar behoren is voltooid. Wanneer een fout bij het formatteren optreedt, dient het formatteren nog een keer te worden uitgevoerd. Een schijf die na drie maal formatteren nog niet met de juiste melding FORMAT COMPLETE komt, is niet betrouwbaar en dient te worden vervangen. Indien een foutmelding veelvuldig terugkomt, is het misschien tijd om de lees/schrijfkoppen van de betreffende schijveneenheid te (laten) schoonmaken. Hiertoe zijn speciale schoonmaaksets in de handel.

Uiteraard kan het geven van foutmeldingen bij het formatteren ook een technische oorzaak hebben. Het kan zijn dat de betreffende eenheid stuk is of de verkeerde schijven worden gebruikt.

Wanneer u de voorbeelden uit de rest van deze paragraaf wilt bestuderen, is het raadzaam om enige geformatteerde schijven te hebben klaarliggen. Geeft u hiertoe in:

```
A>FORMAT
```

De computer vraagt u onmiddellijk

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

Plaatst u op dat moment een nieuw, te formatteren schijf in de eenheid en toets een A (of een B) in.

De computer vraagt u vervolgens:

```
Strike a key when ready
```

Geef ter bevestiging een willekeurige toets in en wacht op de melding:

```
Format complete
```

en herhaal deze behandeling voor alle andere te formatteren schijven.

Wanneer u per ongeluk het FORMAT-kommando heeft ingetoetst en u wilt geen schijven initialiseren, druk dan de CTRL en de C-toets (of de CTRL en de STOP-toets) in. Het formatteren wordt afgebroken.

N.B.: Sommige MSX-computers kunnen na het FORMAT-kommando nog wat aanvullende informatie verstrekken of vragen stellen, afhankelijk van het gebruik type schijveneenheid.

schrijfwijze

```
DIR [<PROJEKTIEWIJZE>][<BESTANDSAANDUIDING>][<PROJEKTIEWIJZE>]

<BESTANDSAANDUIDING>::=<ALFANUMERIEKE KONSTANTE>

<PROJEKTIEWIJZE>::=/P¦/W

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het DIR-kommando kunnen we een inhoudsopgave van een schijf op beeldscherm verkrijgen. Door bijvoorbeeld alleen het volgende kommando in te geven:

```
A>DIR
```

verkrijgen we een inhoudsopgave van de schijf die het laatste werd geactiveerd; in dit geval schijf A.

Door ingave van:

```
A>DIR *.TXT
```

verschijnt op beeldscherm een opgave van alle bestanden op schijf A met typering TXT.

Eventueel kunnen we bij DIR een projektiemethode toepassen. Er zijn twee verschillende projektiemethoden mogelijk, namelijk:

`/P` de letter P staat voor page of bladzijde. Na elke 'bladzijde', steeds als het scherm opnieuw gevuld is zal de computer niet gewoon doorgaan met het projekteren van de inhoud maar zal eerst om een extra toetsaanslag worden gevraagd. Deze projektiemethode is alleen van nut bij schijven die betrekkelijk veel bestanden bevatten.

`/W` de letter W staat voor wide. Wanneer we deze projektiemethode opgeven, krijgen we de inhoud op beeldscherm voorgeschoteld in verkorte vorm over twee kolommen. De gegevens omtrent de grootte van de bestanden en de datum waarop deze bestanden werden aangelegd, worden in dat geval niet getoond.

De verschillende projektiemethoden mogen indien van toepassing zowel voor als achter de bestandsaanduiding worden opgenomen en mogen in elke volgorde met elkaar worden gekombineerd.

Enkele voorbeelden:

`DIR /P` geeft de inhoudsopgave van de laatst geactiveerde schijf. Wanneer het beeldscherm vol is, vraagt de computer u om een willekeurige toetsingave waarna verder wordt geprojekteerd.

`DIR /W B:*.BAS` De inhoud van schijf B wordt in gecomprimeerde vorm getoond. Alleen de bestanden met typering BAS worden getoond.

`DIR A:A??.*/P/W` Alle bestanden op de schijf in de eerste eenheid met een bestandsnaam van drie letters waarvan de eerste letter gelijk is aan de letter A, worden getoond. Eventueel wordt na projektie van een vol beeldscherm gewacht totdat een toets wordt ingedrukt. De inhoudsopgave wordt weer in verkorte vorm weergegeven.

Het volgende voorbeeld geeft de inhoudsopgave van zo maar een schijf:

```
A>DIR
TEST     BAS     6400  2-21-86 12.02p
SCREEN   BAS     3274  2-22-86 12.22p
AUTOEXEC BAS      132  3-03-86 10.04a
BYTES    BAT     1123 12-25-85 11.22p
       4 files   322645 bytes free
```

In de eerste kolom zien we de bestandsnamen genoemd. In de tweede kolom worden de typeringen opgesomd.
In de derde kolom zien we de grootte van elk bestand in aantal bytes uitgedrukt terwijl in de vierde kolom de data zijn opgenomen waarop de betreffende bestanden zijn toegewezen.
Onder MSX-2 wordt in de 5e kolom nog de tijd opgegeven waarop het bestand wordt vastgelegd.
Uiteindelijk wordt aan het einde van de inhoudsopgave het totaal aantal afgedrukte bestandsnamen opgegeven alsmede het aantal nog vrije bytes op de schijf.

schrijfwijze

```
MODE <SCHERMBREEDTE>

<SCHERMBREEDTE>:::=<NUMERIEKE KONSTANTE>

<NUMERIEKE KONSTANTE>:::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het MODE-kommando kan de breedte van het beeldscherm in aantal tekens worden opgegeven. Met het kommando:

```
MODE 40
```

kan de beeldschermbreedte bijvoorbeeld op 40 karakters worden gezet.

Wanneer een beeldschermbreedte van 32 karakters of minder wordt gespecificeerd, dan gaat MSX-DOS in beeldscherm mode 1 over (zie MSX-BASIC handboek). Boven de 32 karakters gaat MSX-DOS over op beeldscherm mode 0.

De opgegeven breedte dient altijd een geheel getal te zijn, groter dan 0 en kleiner dan 41. Dit geldt ook voor de eerste onder MSX-2 uitgebrachte versies van MSX-DOS! In de gecorrigeerde versies kan maximaal MODE 80 worden ingegeven.

schrijfwijze

```
PAUSE [<KOMMENTAAR>]

<KOMMENTAAR>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

Het PAUSE-kommando geeft de melding "Strike any key when ready..." en wacht totdat een toets wordt ingedrukt. Het achter het kommando PAUSE opgenomen kommentaar wordt verder niet in behandeling genomen en is alleen bedoeld als aanwijzing voor de lezer van het beeldscherm.

In het gewone gebuik heeft het PAUSE-kommando geen nut. In hoofdstuk 14 behandelen we het begrip BATCH-FILES. In dat hoofdstuk krijgt dit bevel plotseling groot nut.

schrijfwijze

```
REM [<KOMMENTAAR>]

<KOMMENTAAR>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Het kommando REM is een afkorting van REMARK (opmerking). Net als in BASIC staat het REM-kommando ons toe, een verduidelijking of kommentaar op te nemen. Een verschil is hierin gelegen dat MSX-DOS de REM-kommando's ook afdrukt.

Bij het gewone gebruik heeft het REM-kommando weinig waarde. Wat voor zin heeft het immers om een opmerking op het beeldscherm in te toetsen zonder dat er verder ooit nog iets mee gedaan wordt?

In hoofstuk 14 behandelen we het begrip BATCH-FILES. In dat hoofdstuk krijgt het REM-kommando pas zijn waarde wanneer we programma-achtige strukturen in MSX-DOS gaan ontwerpen.

schrijfwijze

```
TYPE <BESTANDSAANDUIDING>

<BESTANDSAANDUIDING>:::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>:::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Het kommando TYPE probeert voor u de inhoud van het door u aangegeven bestand op beeldscherm weer te geven. Wanneer het betreffende bestand alleen leesbare karakters bevat (een zogenaamde ASCII-file of textfile), dan lukt dit zonder problemen. Wanneer u echter een ander bestand (een BASIC-programma of iets dergelijks) met TYPE probeert te onderzoeken, wordt het op het beeldscherm meestal een enorme ravage. Doordat het betreffende bestand eigenlijk niet geschikt is om te worden afgedrukt, worden de gegevens vaak kris kras door elkaar afgedrukt. Veel van de karakters die TYPE probeert af te drukken, zijn geen zonder meer afdrukbare karakters en veroorzaken bij het projekteren vreemde effecten. Probeer dit maar eens met:

```
A>TYPE COMMAND.COM
```

Een tekstbestand, bijvoorbeeld een met de A-optie op schijf geplaatst programma, kan met TYPE zonder problemen op beeldscherm worden afgedrukt.

Indien geen vaste bestandsnaam maar een bestandsaanduiding aan kommando TYPE is opgegeven, dan wordt het eerste bestand uit de inhoudsopgave dat aan de opgegeven bestandsaanduiding voldoet, afgedrukt.

Voorbeeld: (bij voorkeur uit te voeren op een nieuwe, geformatteerde schijf)

Met het COPY-kommando, dat verderop wordt behandeld, kunnen we ondermeer eenvoudig een tekst in een bestand vastleggen:

```
A>COPY CON TEST.TXT
DEZE TEKST WORDT DADELIJK WEER
MET TYPE OP HET SCHERM GEPLAATST.
^Z
       1 file copied
```

De ingave beëindigen we door intoetsing van tegelijk de CTRL en de Z-toets en daarna de RETURN-toets.

Met het TYPE-kommando kunnen we de vastgelegde tekst dan weer afdrukken:

```
A>TYPE TEST.TXT
DEZE TEKST WORDT DADELIJK WEER
MET TYPE OP HET SCHERM GEPLAATST.
A>
```

schrijfwijze

```
DEL <BESTANDSAANDUIDING> of ERASE <BESTANDSAANDUIDING>
<BESTANDSAANDUIDING>::=<ALFANUMERIEKE KONSTANTE>
<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het kommando ERASE of DEL (beide doen precies hetzelfde) kunnen bestanden van schijf worden verwijderd. Eén bestand maar ook groepen van bestanden kunnen tegelijk worden verwijderd, afhankelijk van de opgegeven bestandsaanduiding.

Voorbeelden:

`A>DEL TEST.TXT`	verwijdert bestand TEST.TXT op de laatst geactiveerde schijf
`A>ERASE A:*.*`	verwijdert ALLE bestanden op de schijf in de eerste schijveneenheid
`A>DEL A:*.TXT`	verwijdert alle bestanden van schijf A met typering TXT
`A>DEL B:M??.*`	verwijdert van de schijf in schijveneenheid nummer 2 alle bestanden die een naam hebben van drie letters, beginnende met een M.

schrijfwijze

```
REN[AME] <BESTANDSAANDUIDING OUD> <BESTANDSAANDUIDING NIEUW>

<BESTANDSAANDUIDING>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het kommando REN of RENAME (hebben dezelfde funktie) kan een bestand of kunnen groepen van bestanden een andere naam worden gegeven. Hiertoe dient achter het REN(AME)-kommando de oude bestandsaanduiding te worden opgenomen, gevolgd door de nieuwe bestandsaanduiding.

Enkele voorbeelden:

`A>REN TEST.BAS TEST.BAK`	bestand TEST.BAS krijgt de nieuwe naam TEST.BAK
`A>RENAME *.BAS *.TXT`	alle bestanden met typering BAS krijgen nu de typering TXT
`A>REN *.* Q*.*`	alle bestandsnamen worden zo veranderd dat de eerste letter een Q wordt
`A>RENAME ???.TXT ???`	alle bestanden met een naam van drie letters en een typering TXT behouden hun eigen naam echter zonder typering.
`A>REN FILE1 ????2`	bestand FILE1 krijgt de nieuwe naam FILE2

```
A>RENAME A:FILE B:FILE
```

deze RENAME-opdracht resulteert onmiddellijk in een fout; een bestand kan natuurlijk niet naar een andere schijf hernoemd worden.

Wanneer REN of RENAME uitvoering het gevolg zou hebben dat er twee of meer bestanden met een zelfde naam op een zelfde schijf ontstaan, wordt de melding 'Rename error' gegeven en wordt de opdracht verder onderbroken.

Het volgende voorbeeld laat u een inhoudsopgave van een schijf, een RENAME-opdracht en weer een inhoudsopgave van dezelfde schijf na deze opdracht zien:

```
A>DIR T*.*
TEST01   TXT    3200  6-16-85
TEST     COM    7654  6-20-85
TEST03   TXT     234  6-21-85
       3 files   310123 bytes free
A>REN TEST??.TXT PROG??.TXT
A>DIR P*.*
PROG01   TXT    3200  6-16-85
PROG03   TXT     234  6-21-85
       2 files   310123 bytes free
A>
```

schrijfwijze

```
DATE [<DATUMOPGAVE>]

<DATUMOPGAVE>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het DATE-kommando kan de systeemdatum van de computer worden veranderd. Standaard staat deze datum op 1 januari 1984. Bij het opstarten van een MSXDOS-schijfje wordt, tenzij bestand AUTOEXEC.BAT op de schijf staat (zie hoofdstuk 10), onder MSX-1 het DATE-kommando automatisch uitgevoerd. Onder MSX-2 blijft de datum bewaard in de speciale clock-chip en behoeft deze niet steeds na het opstarten te worden opgegeven.

Achter het DATE-kommando kan onmiddellijk een datum worden opgegeven. Deze datum dient als volgt te zijn opgebouwd:

MM.DD.JJ of MM/DD/JJ of MM-DD-JJ

of

MM.DD.JJJJ of MM/DD/JJJJ of MM-DD-JJJJ

MM stelt hierbij het nummer van de maand voor en mag variëren van 1 tot 12
DD stelt het dagnummer in de maand voor en mag variëren van 1 tot 31
JJ stelt het jaarnummer voor en mag variëren van 0 tot 99
JJJJ stelt het volledige jaar voor en mag variëren van 1980 tot 2099

Wanneer alleen de laatste twee posities van het jaar worden gespecificeerd, dan neemt het MSX-DOS-systeem het volgende aan:

JJ=80-99	aangenomen wordt het jaar 1980-1999
JJ=0-79	aangenomen wordt het jaar 2000-2079

Het DATE-kommando kontroleert op de juiste datum en houdt daar-

bij ook rekening met schrikkeljaren. De datum 2/29/85 wordt bijvoorbeeld niet geaccepteerd terwijl de datum 2/29/84 wel wordt geaccepteerd.

Wanneer alleen het DATE-kommando, zonder achtergevoegde datum, wordt uitgevoerd, dan geeft het systeem de interne datum op beeldscherm weer. Met alleen de RETURN-toets kan de daaropvolgende ingave worden overgeslagen.

In het volgende voorbeeld wordt eerst de datum op 29 februari 1984 gesteld. Vervolgens wordt deze datum weer opgevraagd. Merk op dat MSX-DOS keurig de juiste dag bij de datum vermeldt.

```
A>DATE
Current date is Sun  1-01-1984
Enter new date: 02/29/84

A>DATE
Current date is Wed  2-29-1984
Enter new date:
```

schrijfwijze

```
TIME [<TIJDOPGAVE>]

<TIJDOPGAVE>:::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>:::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het TIME-kommando kan de tijd van de MSX-computer worden gelijkgezet. Standaard staat de tijd bij het opstarten altijd op twaalf uur middernacht precies. Bij het opstarten van een MSX-DOS-schijfje wordt het TIME-kommando onder MSX-1 op de meeste systemen standaard uitgevoerd. Er zijn echter systemen waarbij het TIME-kommando geen enkele zin heeft.

Elk MSX-2 systeem bezit een clock-chip waarin de tijd wordt bij gehouden, ook als het systeem is uitgeschakeld. Hierdoor behoeft de tijd op een MSX-2 systeem zelden te worden veranderd

Achter het TIME-kommando kan onmiddellijk een tijd worden ingevoerd. Deze tijd dient als volgt te zijn opgebouwd:

UU of UU:MM of UU:MM:SS

of

UUA of UU:MMA of UU:MM:SSA

of

UUP of UU:MMP of UU:MM:SSP

Zonder achtervoegsel of met het achtervoegsel P mag dit gegeven varieren van 0 tot 23. Met achtervoegsel A mag dit gegeven variëren van 0-12. Het achtervoegsel A geeft aan dat het om een tijd gaat tussen 12 uur 's nachts en 12 uur 's middags. Het achtervoegsel P geeft aan dat het om een tijd gaat tussen 12 uur 's middags en 12 uur 's nachts. De tijd kan dus volgens een 12 uurs klok en een 24 uurs klok worden ingegeven.

MM staat voor het minuut-gegeven van de tijd en mag variëren van 0 tot 59.

SS staat voor het seconden-gegeven van de tijd en mag eveneens varieren van 0 tot 59.

Enkele voorbeelden:

`TIME 12`	de tijd wordt op 12 uur 's middags gezet
`TIME 23:59:59`	de tijd wordt op één seconde voor middernacht gezet
`TIME 11:30P`	de tijd wordt op half twaalf 's avonds gezet

Wanneer alleen het kommando TIME wordt ingetoetst, vermeldt het MSX-DOS eerst de oude tijd en vraagt dan om een eventuele nieuwe tijd. Deze nieuwe ingave kan met alleen de RETURN-toets worden overgeslagen waarmee de oude tijd actief blijft.

schrijfwijze

```
        ON
VERIFY  ---
        OFF
```

betekenis

Met het VERIFY-kommando kan het MSX DOS worden bevolen, éénmaal op de schijf geschreven gegevens nog éénmaal ter controle onmiddellijk weer in te lezen en te vergelijken. Het resultaat hiervan is natuurlijk dat het schrijven van gegevens op schijf wat langzamer verloopt.

Met VERIFY ON zet men deze controlemogelijkheid aan en
met VERIFY OFF zet men deze controle weer uit.

Standaard staat deze controlemogelijkheid uit.

Sommige toepassingen van het MSX-DOS systeem kennen deze controlemogelijkheid niet. Weliswaar kunt u VERIFY ON en VERIFY OFF ingeven, maar deze ingaven hebben geen enkel resultaat.

schrijfwijze

```
COPY [<BESTANDSAANDUIDING>[<BESTANDSAANDUIDING>]]

<BESTANDSAANDUIDING>::=<ALFANUMERIEKE KONSTANTE>

<ALFANUMERIEKE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Het kopiëerkommando is het meest uitgebreide kommando dat onder MSX-DOS bestaat. Hieronder wordt het kopiëerkommando in verschillende fasen behandeld, in opkomende graad van moeilijkheid.

1 het kopiëren van een schijf

Een bestand bevat bepaalde gegevens. Op een moment kan het raadzaam zijn om deze gegevens te kopiëren. Wanneer er met de originele gegevens eventueel iets fout mocht gaan, kunt u in dat geval namelijk altijd nog van de eerder gekopiëerde gegevens uitgaan.

Eén van de belangrijkste dingen die u moet doen nog voordat u serieus bepaalde aktiviteiten op de MSX-DOS schijf gaat uitvoeren, is deze MSX-DOS schijf kopiëren.

Voordat u een gehele schijf gaat kopiëren, moet u eerst zorgen voor een lege, geformatteerde schijf waarop u de gegevens kan kopiëren. Formatteer deze schijf eerst met het FORMAT-kommando.

Daarna gaat, wanneer u de beschikking heeft over slechts één schijveneenheid, het kopiëren van een gehele schijf als volgt:

– plaats de te kopiëren schijf in de schijveneenheid

– geef in: `A>COPY A:*.* B:`

Met dit bevel geven we aan dat alle bestanden van de eerste schijveneenheid (A:*.*) naar de tweede schijveneenheid (B:) moeten worden gekopiëerd. We geven het COPY-kommando dus eerst aan, waarvandaan hij de gegevens moet kopiëren en dan pas waar naartoe deze gegevens dienen te worden gekopiëerd. Omdat één schijveneenheid

de rol van twee schijveneenheden kan vervullen onder MSX-DOS, kan op deze wijze een gehele schijf worden gekopiëerd.

De computer laat zien, welke bestanden worden geladen en dadelijk op de nieuwe schijf zullen worden gezet. Wanneer de computer u vraagt:

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

Plaats de nieuwe schijf dan in de schijveneenheid en toets een willekeurige toets in.

Wanneer de computer u vraagt:

```
Insert diskette for drive A:
and strike any key when ready
```

plaats dan weer de oude schijf in de schijveneenheid en geef een willekeurige toets in.

Het kan zijn, bij vrij volle schijven, dat de computer u meerdere malen om een schijvenwisseling vraagt. Doet u dit net zo lang totdat het kopiëren ten einde is.

Op deze wijze kunt u meerdere kopieën van uw schijven maken. Het is raadzaam om van belangrijke schijven minstens twee kopieën te bewaren. Door het goed bijhouden van kopieën is het gevaar, belangrijke gegevens (programma's) te verliezen, minimaal.

Bezitters van twee schijveneenheden kunnen op dezelfde manier kopiëren. Zij moeten er echter eerst voor zorgen dat in de eerste schijveneenheid de originele schijf is geplaatst en in de tweede schijveneenheid de geformatteerde schijf is geplaatst. Tijdens het kopiëren vraagt de computer in dit geval niet om schijfwisselingen.

We zien dat MSX-DOS, wanneer er maar één schijveneenheid beschikbaar is, de mogelijkheid heeft om deze schijveneenheid afwisselend de rol van eerste en van tweede schijveneenheid te laten spelen. Op deze wijze kunnen we bij vele kommando's net doen alsof er twee schijveneenheden aan de computer zijn aangesloten.

2 het kopiëren van een enkel bestand

We hoeven binnen MSX-DOS niet verplicht gehele schijven te kopiëren. Wanneer we dat willen, kunnen we ook enkele bestanden kopiëren. Dit kunnen we doen door alleen de betreffende bestandsnamen aan het COPY-bevel door te geven.

Wanneer we bijvoorbeeld alleen het bestand TEST.TXT naar een andere schijf willen kopiëren, kunnen we dat als volgt doen:

```
A>COPY TEST.TXT B:
```

De computer vraagt op de gebruikelijke wijze op een gegeven moment naar de schijf waarop dit bestand moet worden bijgeplaatst en geeft uiteindelijk de melding 1 file copied.

Wanneer we het bestand TEST.TXT naar een andere schijf willen kopieren en het meteen een andere naam willen geven, kan men dat als volgt doen:

```
A>COPY TEST.TXT B:TEKST.TXT
```

Het bestand TEST.TXT krijgt op de andere schijf nu de naam TEKST. TXT mee; de inhoud van het bestand blijft natuurlijk hetzelfde.

Het kan ook zijn dat we een kopie van een bestand onder een andere naam op *dezelfde* schijf willen hebben. Omdat een bestand op een schijf een unieke naam moet hebben, is het natuurlijk niet mogelijk om twee bestanden onder dezelfde naam op dezelfde schijf te hebben.

Wanneer we bijvoorbeeld het bestand COMMAND.COM onder de naam COMMAND1.COM op dezelfde schijf willen kopiëren, dan kan dat als volgt:

```
A>COPY COMMAND.COM COMMAND1.COM
```

Na een korte tijd geeft de computer een melding dat het bestand is gekopiëerd. Het bestand COMMAND.COM staat nu onder twee verschillende namen op dezelfde schijf. Met het DIR-kommando kan men dat natuurlijk voor de zekerheid even controleren.

3 het kopiëren van groepen van bestanden

We zagen in de bovenstaande behandeling reeds dat het COPY-kommando steeds twee aanduidingen verlangt. De eerste aanduiding geeft aan, welk bestand of welke bestanden dienen te worden gekopiëerd terwijl de tweede aanduiding aangeeft, wat de bestemming van de te kopiëren bestanden is.

Zo hebben de volgende kopiëeropdrachten de daaronder vermelde resultaten:

```
A>COPY A:*.* B:
```

kopiëer alle bestanden naar een andere schijf

```
A>COPY A:TEST.BAS B:
```

kopiëer het bestand TEST.BAS naar een andere schijf

```
A>COPY A:TEST.BAS B:TEST1.BAS
```

kopiëer bestand TEST.BAS naar een andere schijf en geef het daar de naam TEST1.BAS

```
A>COPY TEST.TXT TEST1.TXT
```

kopiëer de inhoud van TEST.TXT naar een ander bestand met de naam TEST1.TXT

Zowel de eerste bestandsaanduiding (die aangeeft waar vandaan moet worden gekopiëerd) als de tweede bestandsaanduiding (die aangeeft waar naar moet worden gekopiëerd) mogen volledig als een bestandsaanduiding worden opgebouwd. Deze aanduidingen mogen dus vraagtekens en sterretjes bevatten. Zo kan men bijvoorbeeld op een gemakkelijke wijze een groep van bestanden kopiëren. Enkele voorbeelden:

```
A>COPY *.TXT B:
```

kopiëert alle bestanden met typering TXT naar een andere schijf.

```
A>COPY TEST?.BAS B:
```

kopiëert alle bestanden met typering BAS die een naam hebben van vijf letters waarvan de eerste vier gelijk zijn aan TEST, naar een andere schijf.

```
A>COPY T*.* Q*.*
```

alle bestanden met een naam, beginnend met een T, worden op dezelfde schijf gekopiëerd. In de nieuwe namen is deze eerste T vervangen door een Q. Zo wordt een bestand TEST.BAS bijvoorbeeld gekopiëerd naar een bestand QEST.BAS.

```
A>COPY ?A?? ?B??
```

alle bestanden met een naam van vier letters waarvan de tweede letter een A is, worden gekopiëerd. De nieuwe bestanden krijgen de namen van oude bestanden met dit verschil dat A op de tweede plaats door een E is vervangen.

Bij deze kopiëerkommando's kan het gevaar schuilen dat een bestand wordt gekopiëerd onder een reeds in gebruik zijnde naam. Het bestand met deze naam wordt dan verwijderd ten behoeve van het nieuwe bestand. Dit is natuurlijk niet altijd de bedoeling...

Een kopiëerkommando als:

```
A>COPY *.* *.*
```

of

```
A>COPY TEST.* TEST.*
```

heeft tot gevolg dat bestanden op ZICHZELF worden gekopiëerd. De meeste MSX-DOS-systemen laten dit toe. Bij grote bestanden (groter dan ongeveer 60 kilobytes) kan dit echter problemen geven. Het is in dat geval mogelijk dat gedeelten van bestanden, soms zonder foutmelding, verloren gaan. Het kopiëren-opzichzelf van bestanden is dus sterk af te raden.

4 samenvoegend kopiëren

Het is mogelijk om meerdere bestanden tijdens het kopiëren samen te voegen tot één groot bestand. In dat geval worden de verschillende inhouden van de betreffende bestanden achter elkaar geplaatst en in één bestand overgenomen. Een voorbeeld:

```
COPY *.TXT ALLES.TXT
```

Deze kopiëeropdracht kopiëert de inhoud van alle bestanden met een typering TXT achter elkaar in één groot bestand, genaamd ALLES.TXT.

Het COPY-kommando staat het gebruik van een plus-teken toe om bestanden 'bij elkaar op te tellen'. Een voorbeeld:

```
A>COPY TEKST1.TXT+TEKST2.TXT TOTAAL.TXT
```

Deze opdracht kopiëert de inhoud van bestand TEST1.TXT, gevolgd door de inhoud van TEKST2.TXT in het bestand TOTAAL.TXT. Het kommando:

```
A>COPY A:TEKST1.TXT+B:TEKST2.TXT A:TOTAAL.TXT
```

kombineert op eenzelfde wijze twee bestanden die echter van verschillende schijven afkomstig zijn. Tijdens uitvoering van dit laatste kommando zal de computer twee maal om een schijfwisseling vragen (tenzij u natuurlijk twee schijveneenheden aan de computer heeft gekoppeld).

Het samenvoegend kopiëren kent vele mogelijkheden. Zo heeft het kommando:

```
A>COPY *.TXT+*.LST *.PRN
```

tot gevolg dat van elk bestand met typering TXT een in naam overeenkomstig bestand met typering LST wordt opgezocht. De samenvoeging van deze twee teksten komt dan terecht in een gelijknamig bestand met typering PRN.

Wanneer bij het samenvoegend kopiëren geen bestemming wordt opgegeven aan het kopiëerkommando, wordt de eerstgenoemde file als bestemming genomen. Zo heeft het kommando:

```
A>COPY TEKST1.TXT+TEKST2.TXT
```

tot gevolg dat de samenvoeging van TEKST1.TXT en TEKST2.TXT in het bestand TEKST1.TXT terechtkomt. De originele TEKST1.TXT gaat bij deze kopiëeropdracht dus verloren.

Ook wanneer men niet samenvoegend kopiëert, neemt het kopiëerkommando bij gebrek aan bestemming altijd het eerste bestand. Een bevel als:

```
A>COPY TEST.TXT
```

heeft bijvoorbeeld tot gevolg dat het bestand op zichzelf wordt gekopiëerd. Praktisch merkt men daar meestal niets van.

Een kommando als:

```
A>COPY *.LST+*.PRN
```

heeft tot gevolg dat voor elk bestand met typering LST een bijbehorend bestand met typering PRN wordt gezocht. Beide bestanden worden samengevoegd en onder de typering LST (eerste naam) weer op schijf gezet. Alle originele LST bestanden gaan hierbij verloren.

5 speciale bestandsnamen

Het MSX-DOS kent enkele voorbestemde bestandsnamen. Deze namen kunt u zelf niet gebruiken om gegevens in op te slaan.

Deze bestandsnamen zijn:

CON geeft de beeldscherm/toetsenbord-combinatie aan
AUX geeft de communicatiepoort aan (RS-232-C indien aanwezig)
LST geeft de printer aan
PRN geeft eveneens de printer aan
NUL geeft een loos kanaal aan

Deze bestandsnamen zijn niet door de gebruiker op schijf aan te leggen onder MSX-DOS, ook niet met één of andere typering. Deze bestandsnamen dienen slechts om de MSX-DOS-kommando's, in het bijzonder het COPY-kommando, extra mogelijkheden te geven.

CON

Deze aanduiding geeft de beeldscherm/toetsenbord-combinatie aan. Door ingave van het kommando:

```
A>COPY CON TEKST.TXT
```

kopiëert men de toetsenbord-ingave naar bestand TEKST.TXT. Na ingave van dit kommando kunnen regels tekst worden ingegeven. Wanneer dan uiteindelijk een CTRL-Z wordt ingegeven (tegelijk CTRL en Z indrukken), gevolgd door een RETURN, wordt de ingegeven tekst in het bestand gezet en verschijnt de normale prompt weer. Door vervolgens in te geven:

```
A>COPY TEKST.TXT CON
```

wordt deze tekst naar het beeldscherm gekopiëerd; de tekst die zojuist werd ingegeven en die in bestand TEKST.TXT staat, verschijnt op beeld.

LST of **PRN**

Deze aanduiding geeft de printer aan. Door ingave van het kommando:

```
A>COPY TEKST.TXT LST
```

kopiëert men de inhoud van bestand TEKST.TXT naar de printer; de tekst die in een vorig voorbeeld werd ingegeven, wordt nu op de printer afgedrukt.

Door ingave van het kommando:

```
A>COPY CON PRN
```

kopiëert men de via het toetsenbord ingegeven gegevens naar de printer. Na ingave van een aantal regels tekst kan men de CTRL-Z ingeven, waarna de tekst onmiddellijk op de printer wordt afgedrukt.

AUX

Wanneer uw MSX computer een communicatie-uitgang (RS-232-C) heeft, kunt u deze aansturen met behulp van de aanduiding AUX. Wanneer u een tweede printer aan deze uitgang heeft gekoppeld, kunt u deze aktiveren door bijvoorbeeld:

```
A>COPY TEKST.TXT AUX
```
(de inhoud van TEKST.TXT wordt afgedrukt)

```
A>COPY CON AUX
```
(de ingave wordt na CTRL-Z op de printer afgedrukt).

Met deze uitgang kan men bijvoorbeeld ook een tweede computer aankoppelen. Met het COPY-kommando kunnen tekstuele gegevens dan zeer gemakkelijk worden verzonden van de ene naar de andere computer. Wanneer beide computers MSX computers zijn, kan een bestand met daarin tekstuele gegevens bijvoorbeeld als volgt worden overgezonden:

```
COMPUTER 1:    A>COPY TEKST.TXT AUX
COMPUTER 2:    A>COPY AUX TEKST.TXT
```

De gegevens, verzonden vanuit computer 1, komen uiteindelijk in bestand TEKST.TXT van computer 2 terecht.

Het overhevelen van andere dan tekstuele gegevens op deze wijze, alsmede het op de juiste wijze koppelen van twee voor elkaar vreemde computers is nog een hele kunst. Hierop kunnen we in dit bestek niet nader ingaan.

NUL

Het kan zijn dat een bevel onder MSX-DOS een bestandsaanduiding nodig heeft terwijl wij deze niet willen geven. In dat geval passen we de aanduiding NUL toe. Twee praktijkvoorbeelden:

Wanneer we een bestand op leesbaarheid willen controleren, dus wanneer we willen onderzoeken of een bestand nog wel helemaal vanaf schijf door de computer is in te lezen, kunnen we dit als volgt doen:

```
A>COPY TEST.BAS NUL
```

We kopiëren het bestand naar 'niets'. Hiervoor moet de computer echter wel het bestand TEST.BAS volledig inlezen. Wanneer we geen leesfout-melding krijgen, is bewezen dat het bestand door de computer nog te benaderen is.

Wanneer we een leeg bestand op schijf willen toewijzen, dan kan dat als volgt:

```
A>COPY NUL+NUL TEST.TXT
```

Een leeg bestand, genaamd TEST.TXT, wordt aangemaakt. De aanduiding NUL+NUL is meestal nodig omdat (een foutje van COPY) anders geen bestand wordt aangemaakt.

6 binair en ASCII (moeilijk)

Bestanden onder MSX-DOS kunnen een variëteit aan gegevens bevatten. Het kan zijn dat een bestand bijvoorbeeld alleen maar tekst bevat. Een bestand als bijvoorbeeld COMMAND.COM bevat echter bijvoorbeeld een kompleet machinetaal-programma.

Het kopiëren van bestanden met verschillende types van gegevens geeft over het algemeen geen problemen. Een uitzondering hierop vormt het kopiëren via de communicatie-uitgang (AUX), dat zonder verdere hulpmiddelen alleen met tekstbestanden mag geschieden.

Wanneer men echter bestanden aan elkaar gaat koppelen (samenvoegend kopiëren), kunnen problemen onstaan.

Wanneer men twee tekstbestanden samenvoegend wil kopiëren, kan dat bijvoorbeeld als volgt:

```
A>COPY TEKST1.TXT+TEKST2.TXT TEKST3.TXT
```

De bestanden TEKST1.TXT en TEKST2.TXT worden achter elkaar gezet en onder de naam TEKST3.TXT weer op schijf gezet.

Om te bepalen waar de tekstbestanden hun laatste karakter hebben staan, is er een einde-bestand-karakter (CTRL-Z) achter de tekst opgenomen. Dit einde-bestand-teken genereert men bij het kommando COPY CON TEST.TXT bijvoorbeeld door de CTRL- en de Z-toets tegelijk in te drukken.

Bij het normale samenvoegend kopiëren worden de samen te voegen bestanden afgezocht tot aan het eerste CTRL-Z-karakter. Alleen het gedeelte vanaf het begin tot aan het CTRL-Z-karakter wordt voor het kopiëren dan gebruikt.

Echter, een enkele maal kan de wat verder gevorderde amateur de noodzaak hebben om twee niet-tekstbestanden aan elkaar te koppelen. Dat kan bijvoorbeeld het geval zijn wanneer een groot machinetaal-

programma dient te worden samengesteld.

Met de /B-optie kan men in dat geval aangeven dat bestanden BINAIR (dus niet tekstueel) dienen te worden geïnterpreteerd. Het volgende voorbeeld kopiëert twee bestanden op binaire wijze naar één nieuw bestand:

```
A>COPY BEST1.COM/B+BEST2.COM BEST3.COM
```

De /B-optie blijft binnen een kopiëer-kommando geldig totdat een /A-optie wordt gegeven. Deze blijft op zijn beurt weer geldig binnen een kopiëerkommando totdat de /B-optie weer gevonden wordt.

Met de /A-optie (de A staat voor ASCII, een kodering voor een tekstuele opslagwijze) geeft men aan, dat een bestand een tekstuele inhoud heeft.

Wanneer men met de /B-optie kopiëert, zoekt de computer niet naar het eerste voorkomen van het einde-bestands-karakter (CTRL-Z) maar kopiëert het systeem gewoon het gehele bestand.

Van deze eigenschap kan men gebruik maken om bijvoorbeeld twee einde-bestands-karakters achter elkaar aan het einde van een bestand te plaatsen. Wanneer dit nodig is, kan men dat bereiken door ingave van bijvoorbeeld:

```
A>COPY TEKST.TXT/B TEKST.TXT/A
```

Het bestand TEKST.TXT wordt in eerste instantie binair ingelezen. Het einde-bestands-karakter, de CTRL-Z achter de inhoud van het bestand, wordt niet als zodanig beschouwd en dus gewoon mee ingelezen.
Het bestand TEKST.TXT wordt in tweede instantie, dit keer als tekstbestand, over het oude bestand heen terug geschreven. Omdat het nu als tekstbestand wordt beschouwd, wordt aan het einde van het bestand een CTRL-Z-karakter (einde-bestands-karakter) toegevoegd. In het nieuwe TEST.TXT staan per saldo nu dus twee CTRL-Z-tekens aan het einde.

13.2 De extrinsieke kommando's van MSX-DOS

Behalve intrinsieke kommando's kent het MSX-DOS ook extrinsieke kommando's. Deze kommando's horen niet bij het operating system, maar zijn daaraan toegevoegd als machinetaalprogramma's.

Extrinsieke kommando's zijn als zodanig te herkennen aan de typering COM (afkorting van COMMAND). Met het kommando

```
A>DIR *.COM
```

kunnen dus precies alle extrinsieke kommando's worden opgevraagd.

De extrinsieke kommando's bevatten machinetaalprogramma's. Tekstverwerkende programma's, databases en dergelijke programmatuur worden meestal in de vorm van COM-files aangeboden. Deze programma's zijn dan op te starten door alleen de naam van de COM-file (zonder de typering COM) in te geven.

Een COM-file vervangt bij het opstarten het controlerend programma van het operating system (COMMAND.COM) en start dit controlerende programma bij beëindiging van zijn funktie weer op (een WARM BOOT). In een vorig hoofdstuk startten we in feite al een COM-file op door ingave van:

```
A>COMMAND
```

We vervingen het controlerende programma van het operating system in dat geval echter door zichzelf waardoor er niets speciaals gebeurde.

Omdat extrinsieke kommando's in feite bestaan uit machinetaalprogramma's die niet vast bij het operating system horen, kunnen deze in het bestek van dit boek verder ook niet worden behandeld. Wanneer u een hoeveelheid extra programmatuur in de vorm van COM-files aanschaft, is deze meestal voorzien van een uitgebreide handleiding.

MSX-DOS biedt ons de mogelijkheid om verzamelingen van veel achter elkaar voorkomende kommando's samen te vatten in een nieuw, eenvoudig kommando. We kunnen onder MSX-DOS dus vrij eenvoudig eigen kommando's maken, gebruik makend van de onder MSX-DOS aanwezig zijnde kommando's.

14.1 Kommando's vereenvoudigen

We kunnen een schijf kopiëren door ingave van

```
A>COPY A:*.* B:
```

Wanneer men deze ingave echter veel te ingewikkeld vindt, kan men voor deze ingave een eenvoudiger kommando ontwerpen en wel als volgt:

```
A>COPY CON C.BAT
COPY A:*.* B:
^Z      (CTRL-Z tegelijk intoetsen)
A>
```

We hebben met behulp van het COPY-kommando een klein tekstbestandje ontworpen met daarin het kopiëerkommando. Dit tekstbestandje bevat het moeilijke kopiëerkommando dat we voortaan op een eenvoudige wijze willen kunnen oproepen en heeft de typering BAT (afkorting van BATCH) gekregen.

Voortaan kunnen we het kopiëren opstarten door ingave van alleen:

```
A>C
```

Het MSX-DOS zoekt bij deze ingave naar het BAT-bestand met de naam C (en natuurlijk de typering BAT) en voert vervolgens de kommando's uit die in dat bestandje staan.

We hebben nu voor het kopiëerkommando een nieuw, heel eenvoudig kommando ontworpen, namelijk het kommando C.

Wanneer we veelvuldig de inhoud van een schijf willen raadplegen op een verkorte wijze, dan kunnen we steeds ingeven:

```
A>DIR/W/P
```

waarna de inhoud wordt getoond. Gemakkelijker is het om éénmaal de BAT-file D.BAT te ontwerpen en wel als volgt:

```
A>COPY CON D.BAT
DIR/W/P
^Z
A>
```

Nu hoeven we voor een inhoudsopgave alleen nog maar de letter D, gevolgd door een RETURN, in te geven.

Een BAT-file mag ook meerdere kommando's bevatten. Een voorbeeld:

```
A>COPY CON K.BAT
PAUSE PLAATS DE ORIGINELE SCHIJF
DIR/W/P
PAUSE GEEF RETURN VOOR HET KOPIEREN
COPY A:*.* B:
PAUSE PLAATS DE ORIGINELE SCHIJF
REM HET KOPIEREN IS TEN EINDE
BASIC
^Z
A>
```

Het bovenstaande voorbeeld geeft aan, hoe door middel van het vervaardigen van een BATCH-file een opvolging van kommando's kan worden vastgelegd. Het kommando K heeft voortaan tot gevolg dat het systeem netjes om de originele schijf vraagt, daar eerst een inhoud van laat zien, daarna pas op een teken van u gaat kopiëren, uiteindelijk weer om de originele schijf vraagt, meldt dat het kopiëren ten einde is en automatisch naar MSX-BASIC overschakelt.

Merk op dat de kommando's REM en PAUSE hier daadwerkelijk een funktie krijgen.

14.2 Variabelen in BAT-FILES

Men kan in een batch-bestand ook variabelen opnemen. Deze variabelen zijn:

%0 geeft vast de naam van het kommando weer
%1 eerste na het kommando ingegeven variabele
%2 tweede na het kommando ingegeven variabele
.
.
.
%9 negende na het kommando ingegeven variabele.

Deze variabelen worden bij uitvoeren van het betreffende kommando vervangen door de na het BAT-kommando ingegeven gegevens. Een voorbeeld:

```
A>COPY CON TEST.BAT
REM KOMMANDO %0
DIR %1
PAUSE GEEF RETURN VOOR VERWIJDEREN
DEL %1
REM EINDE KOMMANDO %0
^Z
A>
```

In bovenstaand voorbeeld werd een kommando ontworpen dat eerst de te verwijderen bestanden laat zien en pas na een RETURN-toets de bestanden daadwerkelijk verwijdert. De bestandsaanduiding dient als eerste variabele aan dit kommando te worden opgegeven. Wil men bijvoorbeeld met behulp van dit nieuwe kommando alle bestanden met typering BAS verwijderen, dan geeft men in:

```
A>TEST *.BAS
```

Gedurende de uitvoering van het door ons samengestelde batch-kommando zullen de variabelen %0 en %1 worden vervangen door TEST (de naam van het kommando) en *.BAS (de na TEST ingegeven typering).

14.3 Automatisch opstarten

Wanneer men de computer vanuit MSX-DOS automatisch bij het opstarten een bepaalde taak wil laten uitvoeren, dan dient men deze taak in een speciaal batch-bestand op te nemen. De naam van dit speciale batch-bestand is AUTOEXEC.BAT.

Wanneer men bijvoorbeeld direkt na het opstarten eerst de datum bij wil stellen en daarna het MSX-BASIC programma MENU.BAS wil opstarten, dan doet men dit als volgt:

```
A>COPY CON AUTOEXEC.BAT
DATE
BASIC MENU.BAS
^Z
A>
```

Het aldus ontworpen kommando AUTOEXEC wordt bij het opstarten van het systeem automatisch uitgevoerd. We kunnen dit testen door de computer na vervaardiging van dit kommando eerst uit en dan weer aan te zetten.

Het is niet zinvol om in AUTOEXEC.BAT variabelen te specificeren.

14.4 Voorbeelden

Er zijn talloze voorbeelden te bedenken waarbij het ontwerpen van BAT-files goed van pas komt. In deze paragraaf volgen enkele, uitgewerkte voorbeelden van batch-bestanden.

B. BAT: automatisch opstarten BASIC

De volgende BAT-file heeft als resultaat dat het MSX-BASIC automatisch wordt opgestart wanneer onder MSX-DOS de letter B wordt ingegeven. Behalve een BAT-FILE moet ook een klein BASIC-programma worden geschreven.

Tik eerst onder MSX-DOS in:

```
A>COPY CON B.BAT
BASIC B.BAS
^Z                          (druk de toetsen CTRL en Z tegelijk in)
A>
```

Start hierna het MSX-BASIC op:

```
A>BASIC
```

De MSX-BASIC-melding verschijnt.
Geef het volgende programma in:

```
NEW
Ok
10 KEY 1,"CALL SYSTEM"+CHR$(13)
20 NEW
SAVE "B.BAS"
Ok
CALL SYSTEM
A>
```

Vanaf nu kan het basic worden opgestart met de ingave:

```
A>B
```

Funktietoets 1 zorgt in MSX-BASIC voor een eenvoudige terugkeer naar het MSX-DOS.

WIDTH.BAT: instellen beeldscherm

Onder MSX-BASIC kan de beeldschermbreedte met het kommando WIDTH XX worden ingesteld waarbij op de plaats van XX dan de gewenste breedte dient te worden opgegeven. Onder MSX-DOS dient deze breedte met het MODE XX kommando te worden ingegeven. Om verwarring te voorkomen, kan een WIDTH-kommando onder MSX-DOS worden samengesteld en wel als volgt:

Tik onder MSX-DOS in:

```
A>COPY CON WIDTH.BAT
MODE %1
^Z                    (druk de CTRL- en de Z-toets tegelijk in)
A>
```

Vanaf nu kan het WIDTH-kommando ook onder MSX-DOS worden ingegeven, probeer maar eens met b.v.:

```
A>WIDTH 30
```

K.BAT: instellen MSX-DOS toetsen

Met de volgende BAT-file kan door alleen de ingave van de letter K de funktietoetsen op het MSX-DOS worden afgesteld. Behalve een BAT-file moet ook een BASIC-programma worden geschreven.

Tik eerst onder MSX-DOS in:

```
A>COPY CON K.BAT
BASIC K.BAS
^Z                          (toets de CTRL- en de Z-toets tegelijk in)
A>
```

Activeer nu eerst het MSX-BASIC en schrijf het volgende programma:

```
A>BASIC                     (de MSX-BASIC melding verschijnt)

10 DATA 1,"CALL SYSTEM@"
20 DATA 2,"DIR/P@"
30 DATA 3,"COPY "
40 DATA 4,"FORMAT@"
50 DATA 5,"BASIC@"
60 DATA 6,"ERASE "
70 DATA 7,"RENAME "
80 DATA 8,"TYPE "
90 DATA 9,"MODE "
100 DATA 10,"DATE@"
110 DATA -1
120 READ A:IF A=-1 THEN CALL SYSTEM
130 READ A$:IF RIGHT$(A$,1)="@" THEN MID$(A$,LEN(A$))=CHR$(1
3)
135 KEY A,A$
140 GOTO 120
SAVE "K.BAS"
Ok
```

Uiteraard mogen in programmaregels 10-100 de funktietoetsen naar keuze worden ingedeeld.

Help.BAT: een geheugensteuntje

De volgende BAT-file geeft u een kort overzicht van de aanwezige MSX-kommando's als een geheugensteuntje. Alleen het woord HELP hoeft maar te worden ingegeven.

Tik onder MSX-DOS in:

```
A>COPY CON HELP.BAT
TYPE HELP.TXT
^Z                                  (toets de CTRL- en de Z-toets tegelijk in)
A>
```

Tik vervolgens onder MSX-DOS in:

```
A>COPY CON HELP.TXT
-----------------------------------
DIR /W /P inhoudsopgave schijf
COPY      kopieren van bestanden
TYPE      uittikken tekstbestand
REN(AME)  bestanden anders noemen
DEL       verwijdenen van bestanden
ERASE     als DEL
DATE      instellen van de datum
TIME      instellen van de tijd
PAUSE     pauze tot toetsingave
REM       opmerking
VERIFY    ON/OFF controle aan/uit
BASIC     naar BASIC
FORMAT    formatteren schijven
MODE      beeldschermbreedte
-----------------------------------
^Z                                  (geef de CTRL- en de Z-toets tegelijk in)
A>
```

In het bovenstaande voorbeeld dienen ook de strepen te worden ingetoetst. Probeer de BAT–file nu uit ingave van:

```
A>HELP
```

14.5 Batch-files ingeven

Het is nogal omslachtig om met het COPY-kommando de gewenste BAT-files aan te maken. Daarom volgt hieronder een programma waarmee BAT-files tot 23 regels lang kunnen worden aangemaakt. Tijdens de invoer kan gebruik worden gemaakt van de pijltoetsen, INS en DEL. Met SELECT laat het programma zélf zien, welke funkties er nog meer zijn. Zo leggen we bijvoorbeeld met CONTROL-S een scherm met teksten vast op schijf.
De handige amateur breidt dit programma eenvoudig uit tot een leuk tekstverwerkingsprogrammaatje.

```
10 REM *************************
20 REM *      SCREEN.BAS       *
30 REM *    SCHERM EDITOR      *
40 REM *************************
50 REM
60 REM INITIALISATIE
70 REM
80 CLEAR 10000:SCREEN 0:KEY OFF
90 INPUT "SCHERMBREEDTE ";S
100 CLS:WIDTH S:B=BASE(0):COLOR 15,4,4:X=0:Y=0:DIM S$(22):FO
R I=0 TO 22:S$(I)=SPACE$(S):NEXT I:B$=S$(0)
110 REM
120 REM TOETSINGAVE
130 REM
140 LOCATE X,Y,1:A$=INPUT$(1)
150 A=ASC(A$):IF A>31 AND A<127 THEN MID$(S$(Y),X+1,1)=A$:PR
INT A$;:A=28:GOSUB 490:GOTO 140'KARAKTER INGEGEVEN
160 LOCATE ,,0:IF A=127 THEN S$(Y)=LEFT$(S$(Y),X)+MID$(S$(Y)
,X+2)+" ":LOCATE X,Y:PRINT MID$(S$(Y),X+1);:GOTO 140'DEL ING
EGEVEN
170 ON A GOSUB 220,230,240,250,260,270,280,290,300,310,320,3
30,340,350,360,370,380,390,400,410,420,430,440,450,460,470,4
80,490,500,510,520'CONTROL-INGAVEN
180 GOTO 140
190 REM
200 REM CONTROLETOETSEN
210 REM
220 B$=S$(Y):RETURN'CONTROL-A
230 RETURN'CONTROL-B
240 RETURN'CONTROL C
250 PRINT CHR$(27);"M";:IF Y<22 THEN FOR I=Y TO 21:S$(I)=S$(
I+1):NEXT I:S$(22)=SPACE$(S):RETURN ELSE S$(22)=SPACE$(S):RE
TURN'CONTROL-D
260 MID$(S$(Y),X+1,S-X)=SPACE$(S-X):PRINT CHR$(27);"K";:RETU
RN'CONTROL-E
270 CLS:PRINT "FILES:":PRINT STRING$(S,"-");:FILES:PRINT:PRI
NT STRING$(S,"-");:PRINT "(GEEF RETURN)";:A$=INPUT$(1):CLS:F
OR I=0 TO 22:PRINT S$(I);:NEXT I:RETURN'CONTROL-F
```

```
280 RETURN'CONTROL-G
290 X=X-1-S*(X=0):Y=Y+(X=S-1)-23*(Y=0 AND X=S-1):MID$(S$(Y),
X+1)=" ":PRINT CHR$(8);" ";CHR$(8);:RETURN'CONTROL-H/BACKSPA
CE
300 PRINT CHR$(27);"L";:LOCATE 0,23:PRINT CHR$(27);"1";:FOR
I=21 TO Y STEP-1:S$(I+1)=S$(I):NEXT I:S$(Y)=SPACE$(S):RETURN
'CONTROL-I
310 RETURN'CONTROL-J
320 X=0:Y=0:RETURN'CONTROL-K/HOME
330 GOSUB 660'CONTROL-L
340 X=0:Y=Y+1+23*(Y=22):RETURN'CONTROL-M/RETURN
350 RETURN'CONTROL-N
360 RETURN'CONTROL-O
370 FOR I=0 TO 22:LPRINT S$(I):NEXT I:RETURN'CONTROL-P
380 FOR I=X+1 TO S:J=ASC(MID$(S$(Y),I,1)):MID$(S$(Y),I,1)=CH
R$(J-32*(J>64 AND J<91)+32*(J>96 AND J<123)):LOCATE I-1,Y:PR
INT MID$(S$(Y),I,1);:NEXT I:RETURN'CONTROL-Q
390 S$(Y)=LEFT$(S$(Y),X)+" "+MID$(S$(Y),X+1,S-1-X):LOCATE X,
Y:PRINT MID$(S$(Y),X+1);:RETURN'CONTROL-R/INS
400 GOSUB 560:RETURN'CONTROL-S
410 RETURN'CONTROL-T
420 RETURN'CONTROL-U
430 RETURN'CONTROL-V
440 RETURN'CONTROL-W
450 GOSUB 750:RETURN'CONTROL-X/SELECT
460 RETURN'CONTROL-Y
470 S$(Y)=B$:LOCATE 0,Y:PRINT B$;:RETURN'CONTROL-Z
480 RETURN'CONTROL-?
490 X=X+1+S*(X=S-1):Y=Y-(X=0)+23*(Y=22 AND X=0):RETURN'RECHT
S
500 X=X-1-S*(X=0):Y=Y+(X=S-1)-23*(Y=0 AND X=S-1):RETURN'LINK
S
510 Y=Y-1-23*(Y=0):RETURN'BOVEN
520 Y=Y+1+23*(Y=22):RETURN'BENEDEN
530 REM
540 REM SCHRIJF SCHERM (CONTROL-S)
550 REM
560 CLS:LINE INPUT "FILE:";F$:IF F$="" THEN GOTO 620
570 OPEN F$ FOR OUTPUT AS 1
580 FOR J=22 TO 0 STEP -1:IF S$(J)=SPACE$(S) THEN NEXT J
590 FOR I=0 TO J:A$=S$(I)
600 IF RIGHT$(A$,1)=" "THEN A$=LEFT$(A$,LEN(A$)-1):GOTO 600
610 PRINT #1,A$:NEXT I:CLOSE
620 CLS:FOR I=0 TO 22:PRINT S$(I);:NEXT I:RETURN
630 REM
640 REM LEES SCHERM (CONTROL-L)
650 REM
660 CLS:LINE INPUT "FILE:";F$:IF F$="" THEN GOTO 710
670 OPEN F$ FOR INPUT AS 1
680 FOR I=0 TO 22:IF EOF(1)=-1 THEN S$(I)=SPACE$(S):GOTO 700
690 LINE INPUT #1,S$(I):S$(I)=LEFT$(S$(I)+SPACE$(S),S)
700 NEXT I:CLOSE
710 CLS:FOR I=0 TO 22:PRINT S$(I);:NEXT I:RETURN
```

```
720 REM
730 REM HULPFUNKTIE
740 REM
750 CLS:PRINT "CONTROL- FUNKTIE"
760 PRINT STRING$(S,"-");
770 PRINT "  A      ONTHOU DEZE REGEL"
780 PRINT "  D      VERWIJDER DEZE REGEL"
790 PRINT "  E      VERWIJDER DE REST"
800 PRINT "  F      WELKE BESTANDEN ?"
810 PRINT "  I      VOEG REGEL TUSSEN"
820 PRINT "  L      LAAD EEN SCHERM"
830 PRINT "  P      PRINT HET SCHERM"
840 PRINT "  Q      GROTE LETTERS <-> KLEINE"
850 PRINT "  S      SCHRIJF EEN SCHERM"
860 PRINT "  Z      PLAATS CONTROL-A REGEL"
870 PRINT STRING$(S,"-");"(GEEF RETURN)"
880 A$=INPUT$(1):CLS:FOR I=0 TO 22:PRINT S$(I);:NEXT I:RETUR
N
```

15 ENIGE MSX-DOS BIJZONDERHEDEN

Het MSX-DOS kent enkele bijzonderheden die in dit hoofdstuk worden samengevat.

15.1 CTRL-P/CTRL-N

Wanneer men dit wenst, kan men met een eventueel aangesloten printer alle gegevens die op beeldscherm verschijnen, meteen laten afdrukken.

Deze print-optie kan men door ingave van CTRL-P (de CTRL- en de P-toets tegelijk indrukken) inschakelen. Op eenzelfde wijze schakelt men deze optie ook weer uit met CTRL-N.

PAS OP: nooit de CTRL-P optie gebruiken wanneer geen printer is aangesloten. MSX-DOS 'hangt zich op' in dat geval.

15.2 Besturingstoetsen onder MSX-DOS

Het is in MSX-DOS mogelijk om de ingave enigszins te besturen. Een per ongeluk fout ingetoetst kommando behoeft niet altijd te worden overgetikt maar kan eventueel worden aangepast.

De volgende besturingstoetsen zijn aanwezig:

pijl naar rechts	herhaalt teken voor teken het laatst ingegeven kommando.
pijl naar links	verwijderdt één karakter.
pijl naar boven	maakt de regel helemaal leeg.
pijl naar beneden	herhaalt het volledige laatste kommando.
DEL	verwijdert het volgende karakter van het laatst ingegeven kommando.
INS	voegt de volgende karakters tussen.
BS	verwijdert het laatst ingegeven karakter (als pijl naar links).
SELECT	na SELECT moet een karakter worden ingegeven. Het laatst ingegeven kommando wordt tot aan dit karakter overgenomen.
CLS	na CLS moet een karakter worden ingegeven. Het vorige kommando wordt tot aan dit karakter overgeslagen.

HOME maakt het tot nu toe ingegeven bevel tot het laatst ingegeven bevel.

Een voorbeeld: stel, we toetsen in:

```
A>COPY A:*.BSA B:
```

Na uitvoering van dit kommando blijkt dat het fout was. We willen dit kommando nogmaals uitvoeren; BSA moet echter BAS worden.

Wanneer we de pijl naar beneden ingeven, zien we het gehele bevel weer tevoorschijn komen. Dit is echter niet de bedoeling; met de pijl omhoog verwijderen we dit laatste kommando weer. Met de pijl naar rechts kunnen we eventueel teken voor teken het oude kommando weer te voorschijn halen.

Met een SELECT-B ingave (eerst SELECT, dan de letter B) kunnen we het oude kommando echter in één klap tot aan de letter B overnemen. Met de pijl naar rechts herhalen we vervolgens de letter B. Door nu de letters AS in te geven, gevolgd door de pijl naar beneden, hebben we het uiteindelijke bevel nu goed op scherm staan. Door nu een RETURN in te geven, wordt het bevel daadwerkelijk uitgevoerd.

Wanneer we de bevelen TIME en DATE uitvoeren, kunnen we de huidige tijd en datum op een zelfde wijze aanpassen.

Alleen voor diegenen die onder MSX-DOS bijzonder veel ingave-werk moeten doen, heeft het zin om deze edit-mogelijkheden te bestuderen. Voor alle anderen staat het aanleren van de verschillende funkties waarschijnlijk niet in verhouding tot het nut dat men er ooit van heeft.

15.3 Scheidingskarakters

Tot nu toe hebben we in de MSX-DOS bevelen als scheidingskarakter steeds de spatie gebruikt. MSX-DOS staat echter meer scheidingskarakters toe.

De volgende scheidingskarakters mogen worden gebruikt:

TAB , (komma) ; (puntkomma) = (gelijkteken)

Dus een kopiëerkommando mag bijvoorbeeld ook als volgt worden ingetoetst:

```
A>COPY:A:*.*=B:
```

of

```
A>COPY A:*.*,B:
```

15.4 Foutmeldingen van MSX-DOS

Het MSX-DOS kent verschillende foutmeldingen. Deze meldingen worden niet gekodeerd in de vorm van een foutnummer maar in de vorm van een tekstje gegeven.

De volgende foutmeldingen kunnen door het MSX-DOS operating system worden gegeven:

Vanuit MSXDOS.SYS (vanuit het kernel)

Terminate batch file (Y/N)?

> Tijdens de uitvoering van een BAT-file heeft u een CTRL-STOP of een CTRL-C ingegeven. U kunt de uitvoering van het batchbestand verder laten gaan (Y) of de uitvoering stoppen (N).

Insert DOS disk in default drive
and strike any key when ready

> Na de uitvoering van een extrinsiek kommando probeerde MSX-DOS een WARM BOOT. COMMAND. COM werd echter niet aangetroffen; waarschijnlijk dient een andere schijf te worden geplaatst.

Bad FAT, drive...

> Op de aangegeven schijveneenheid is een foute File Allocation Table aangetroffen. In de praktijk betekent dat, dat een schijf is verminkt. De schijf zal opnieuw moeten worden geformatteerd alvorens deze weer kan worden gebruikt.

Write protect error writing drive...

> Bij een poging, gegevens op een schijf te schrijven, konstateerde

MSX-DOS dat de schijf in de aangegeven schijveneenheid write protected is. Doordat het write-protect stickertje is opgeplakt of doordat het write-protect schuifje is verschoven heeft men klaarblijkelijk te kennen gegeven dat de gegevens op deze schijf niet mogen worden overschreven.

Not ready error reading (writing) drive...

Tijdens een poging om gegevens te lezen/schrijven, konstateerde MSX-DOS dat de betreffende schijveneenheid hiervoor niet klaar was. Misschien is de spanning van de eenheid niet ingeschakeld of staat de vergrendeling nog los.

Disk error reading/writing drive...

Tijdens een lees/schrijfpoging werd een fout op schijf gekonstateerd. Misschien is de schijf nog niet geformatteerd, is hij versleten of is de eenheid vervuild of ontsteld.

Abort, Retry, Ignore?

Deze melding volgt altijd direkt op de drie voorgaande foutmeldingen. Met ingave van een R (retry) kan men de schijfaktie nog eens proberen, bijvoorbeeld nadat men de storing heeft opgeheven. Met I (ignore) kan men aangeven dat de fout verwaarloosd moet worden (niet aan te bevelen). Met A kan men tenslotte aangeven dat het gehele kommando dient te worden beëindigd.

Unsupported media type

(MSX-2) Het MSX-DOS systeem op een niet door MSX-DOS bestuurbare schijveneenheid gebruikt.

Vanuit COMMAND.COM (vanuit het controlerende programma)

Insert disk with batch file
and strike any key when ready

Tijdens het uitvoeren van de kommando's uit een batch-file kon MSX-DOS de batch-file plotseling niet meer vinden. Vermoedelijk is de schijf ondertussen verwisseld. De schijf met het batchbestand dient te worden geplaatst waarna een toets moet worden ingegeven.

Strike a key when ready...

Deze melding kan op verschillende plaatsen voorkomen. De computer verwacht dan een toetsingave van u nadat een bepaalde aktie is voltooid.

Invalid drive specification

Een verkeerde schijveneenheid-aanduiding werd aangegeven (bijvoorbeeld F:)

Program too big to fit in memory

Er werd getracht om een programma te laden dat te groot is om in het geheugen van de computer te passen.

Bad command or file name

Deze hele bekende melding verschijnt wanneer MSX-DOS niet weet wat het met een bepaalde ingave aan moet.

File not found

De gespecificeerde file werd niet gevonden

Are you sure (Y/N)?

Deze melding volgt als antwoord op een 'gevaarlijke' opdracht waarbij veel gegevens verloren kunnen gaan. Wanneer u zeker van uw zaak bent, geeft u een Y (YES) in, anders een N (NO).

Rename error

Tijdens het hernoemen van bestanden konstateerde REN(AME) een fout.

Invalid parameter

Een kommando werd voorzien van de verkeerde toevoegingen (bijvoorbeeld een VERIFY A:)

Insufficient disk space

Op schijf werd niet meer voldoende vrije ruimte gevonden om een kommando te voltooien.

File creation error

Tijdens het toewijzen van een bestand ontdekte MSX-DOS een fout.

File cannot be copied into itself

Er werd een bestand naar zichzelf gekopiëerd in een situatie waarin dat niet mogelijk is.

Content of destination lost before copy

Tijdens het kopiëren kwam een bestand zowel als te kopiëren bestand als naar te kopiëren bestand voor. Voordat het COPY-kommando het bestand kon inlezen, werd het al in de oorspronkelijke vorm aangetast. De gegevens uit dit bestand zijn verloren gegaan.

Write error

Een algemene schrijffout werd gekonstateerd.

Invalid date
Enter new date

Een verkeerde datum werd ingegeven. MSX-DOS vraagt alsnog om de goede datum.

15.5 Fouten in MSX-DOS

De huidige MSX-DOS en COMMAND-versie vertonen nog enige fouten. Deze fouten zijn bijna nooit hinderlijk, maar kunnen soms weleens verwarring wekken.

Met name het COPY-kommando vertoont in vele uitvoeringen nogal storende fouten. Zo geeft het bijvoorbeeld altijd de melding '1file copied' ook wanneer er geen enkele file werd gekopiëerd. Ook de ingave COPY NUL TEST.TXT werkt bijvoorbeeld niet. Indien een te kopiëren bestand niet bestaat, wordt daar vaak geen melding van gemaakt.

16 ESCAPE SEQUENTIES

Onder MSX-DOS verkrijgbare software dient vaak een eerste maal te worden geinstalleerd op de computer. Tijdens de installatieprocedure zullen u dan diverse vragen worden gesteld met betrekking tot de beeldschermbesturing.

Onder MSX-basic zijn er diverse kommando's om het beeldscherm te besturen. Zo kunnen we met de LOCATE-instruktie bijvoorbeeld de cursor op elk gewenst punt op het beeldscherm plaatsen.

Helaas kan de software die onder MSX-DOS verkrijgbaar is, deze specifieke MSX-basic bevelen niet uitvoeren. Deze programmatuur heeft een andere, uitvoerbare kodering nodig voor de diverse beeldschermbestuderingen.

MSX voorziet in een alternatieve beeldschermbestudering om de niet-basic programmatuur ter wille te zijn. Deze beeldschermbesturing bestaat steeds uit een escape-karakter, gevolgd door één of meer andere karakters.

Omdat het ESCAPE-karakter bij deze alternatieve beeldschermbestudering de hoofdrol speelt, noemen we al deze beeldschermaansturingen ESCAPE sequenties. En omdat het ESCAPE-karakter al deze beeldschermbesturingen in feite "inleidt", noemen we het escape-karakter het zogenaamde lead-in character.

In de tabel op de volgende bladzijde staan alle mogelijke alternatieve beeldschermbesturingen genoemd.

Merk op dat er in de tabel hele interessante, ook onder BASIC heel goed bruikbare funkties aanwezig zijn. Met name de mogelijkheid om op het scherm heel snel een regel tussen te voegen of te verwijderen, is heel interessant. Deze mogelijkheid werd verwerkt in het schermeditorprogramma dat in hoofdstuk 14 werd geplaatst.

FUNKTIE	AANROEP	HEXA	KODERING ONDER BASIC
CURSOR NAAR BOVEN	<ESC>A	1B41	PRINT CHR$(27);"A";
CURSOR NAAR BENEDEN	<ESC>B	1B42	PRINT CHR$(27);"B";
CURSOR NAAR RECHTS	<ESC>C	1B43	PRINT CHR$(27);"C";
CURSOR NAAR LINKS	<ESC>D	1B44	PRINT CHR$(27);"D";
CURSOR ADRESSEREN	<ESC>Y<Y+20H> <X+20H>	1B59YYXX	PRINT CHR$(27);"Y"; CHR$(Y+32);CHR$(X+32);
SCHERM SCHOONMAKEN	<ESC>j	1B6A	PRINT CHR$(27);"j";
SCHERM SCHOONMAKEN	<ESC>E	1B45	PRINT CHR$(27);"E";
REGEL VANAF CURSOR SCHOONMAKEN	<ESC>K	1B4B	PRINT CHR$(27);"K";
SCHERM VANAF CURSOR SCHOONMAKEN	<ESC>J	1B4A	PRINT CHR$(27);"J";
REGEL SCHOONMAKEN	<ESC>l	1B6C	PRINT CHR$(27);"l";
REGEL TUSSENVOEGEN	<ESC>L	1B4C	PRINT CHR$(27);"L";
REGEL VERWIJDEREN	<ESC>M	1B4D	PRINT CHR$(27);"M";
CURSOR=BLOK	<ESC>x4	1B7834	PRINT CHR$(27);"x4";
CURSOR=UIT	<ESC>x5	1B7835	PRINT CHR$(27);"x5";
CURSOR=ONDERLIJNING	<ESC>y4	1B7934	PRINT CHR$(27);"y4";
CURSOR=AAN	<ESC>y5	1B7935	PRINT CHR$(27);"y5";

Hieronder volgt nog een voorbeeld van cursoradressering met behulp van de alternatieve besturing:

```
NEW
Ok
10 REM DIREKTE CURSORADRESSERING MET ESCAPE-SEQUENTIES
20 SCREEN 0:WIDTH 40:COLOR 15,4,4:CLS
30 FOR X=0 TO 39:FOR Y=0 TO 23
40 PRINT CHR$(27);"Y";CHR$(32+Y);CHR$(32+X);"#";
50 NEXT Y,X
RUN
```

17 ALTERNATIEVE OPSTARTPROCEDURES

Wanneer een MSX-computer met schijveneenheid wordt aangezet, dan rekent deze op de aanwezigheid van een even aantal schijveneenheden.

Om met schijveneenheden gegevens te kunnen uitwisselen is er per eenheid een bepaald stuk geheugen nodig. Hoe meer aangesloten schijveneenheden, hoe minder geheugen er dus overblijft voor de opslag van een programma.

Soms is dit hinderlijk. Het kan zijn dat we voor een bepaald programma net een stukje geheugen te kort komen, terwijl we maar één of helemaal geen schijveneenheid nodig hebben.

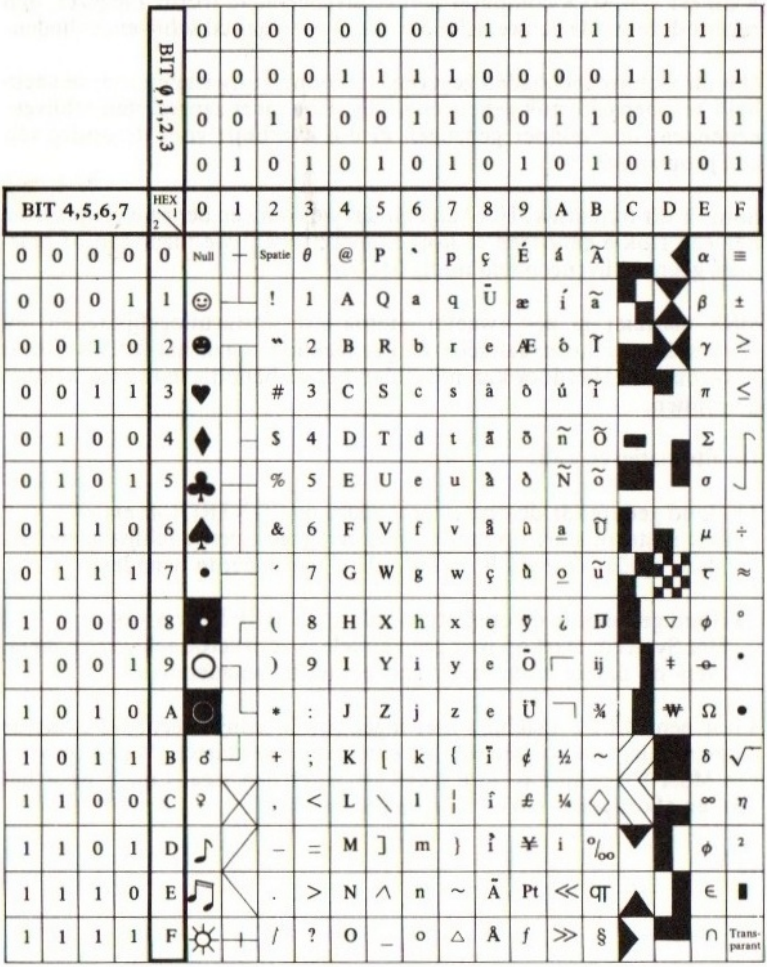
MSX voorziet in een tweetal alternatieve opstartmogelijkheden om slechts één of helemaal géén schijveneenheden aan het MSX-systeem te verbinden. Hierdoor kan misschien net de benodigde ruimte worden gewonnen.

De alternatieven zijn:

1) Houd gedurende de opstartprocedure de CONTROL-toets vast.
   Het systeem start op met slechts één geactiveerde schijveneenheid.
   Merk op dat eenheid B: in het geheel niet meer te benaderen is.

2) Houd gedurende de opstartprocedure de SHIFT-toets vast.
   Het Systeem start op met geen enkele geactiveerde schijveneenheid.
   Merk op dat de schijf in het geheel niet te benaderen is.

Voor een MSX-2 machine met ingebouwde schijveneenheid is dit de enige manier om zonder schijveneenheid op te starten. Maar ook voor een MSX-1 machine is deze methode beter dan steeds maar weer het aan- en afkoppelen van de schijveneenheid.



BIT 4,5,6,7				HEX 2/1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	Null		Spatie	θ	@	P	`	p	ç	É	á	Ã			α	≡
0	0	0	1	1	☺		!	1	A	Q	a	q	Ü	æ	í	ã			β	±
0	0	1	0	2	☻		"	2	B	R	b	r	e	Æ	ó	Ĩ			γ	≥
0	0	1	1	3	♥		#	3	C	S	c	s	å	ò	ú	ĩ			π	≤
0	1	0	0	4	♦		$	4	D	T	d	t	ä	õ	ñ	Õ			Σ	⌠
0	1	0	1	5	♣		%	5	E	U	e	u	à	ò	Ñ	õ			σ	⌡
0	1	1	0	6	♠		&	6	F	V	f	v	ª	û	a	Ũ			μ	÷
0	1	1	1	7	•		'	7	G	W	g	w	ç	ù	º	ũ			τ	≈
1	0	0	0	8	◘		(	8	H	X	h	x	e	ÿ	¿	Ð		▽	φ	°
1	0	0	1	9	○		)	9	I	Y	i	y	e	Õ	⌐	ij		‡	-φ-	•
1	0	1	0	A	◙		*	:	J	Z	j	z	e	Ü	¬	¾		₩	Ω	•
1	0	1	1	B	♂		+	;	K	[	k	{	ĩ	¢	½	~			δ	√
1	1	0	0	C	♀		,	<	L	\	l	¦	î	£	¼	◇			∞	η
1	1	0	1	D	♪		–	=	M	]	m	}	ì	¥	¡	‰			φ	²
1	1	1	0	E	♫		.	>	N	^	n	~	Ã	Pt	≪	¶			∈	■
1	1	1	1	F	☼		/	?	O	_	o	△	Å	ƒ	≫	§			∩	Trans-parant

In de bovenstaande tabel zijn alle MSX-karakters opgenomen met hun bijhorende binaire en hexadecimale waarden. Door de vermelde kon-

stanten in de CHR$-funktie op te nemen, kunnen de betreffende karakters worden gegenereerd. Een uitzondering vormen de karakters met een hexadecimale waarde kleiner dan &H20 (decimaal 32, binair &B00100000). Deze worden alleen gegenereerd met CHR$ indien voorafgegaan door een CHR$(1). Bij het kodenummer van het bepaalde karakter dient dan de waarde &H40 (decimaal 64, binair &B01000000) te worden opgeteld. Zo drukt PRINT CHR$(1)+CHR$(&H41) het grafische symbool onder kodenummer 1 af (eerste kolom, tweede karakter).

19 GERESERVEERDE SLEUTELWOORDEN

Deze sleutelwoorden zijn door MSX-basic gereserveerd en mogen niet als vrije variabele-namen worden gebruikt.

ABS	DEFDBL	INP	NOT	SIN
AND	DEFINT	INPUT	OCT$	SOUND
ASC	DEFSNG	INSTR	OFF	SPACE$
ATN	DEFSTR	INT	ON	SPC(
ATTR$	DELETE	IPL	OPEN	SPRITE
AUTO	DIM	KEY	OR	SQR
BASE	DRAW	KILL	OUT	STEP
BEEP	DSKF	LEFT$	PAD	STICK
BIN$	DSKI$	LEN	PAINT	STOP
BLOAD	DSKO$	LET	PDL	STR$
BSAVE	ELSE	LFILES	PEEK	STRIG
CALL	END	LINE	PLAY	STRING$
CDBL	EOF	LIST	POINT	SWAP
CHR$	EQV	LLIST	POKE	TAB(
CINT	ERASE	LOAD	POS	TAN
CIRCLE	ERL	LOC	PRESET	THEN
CLEAR	ERR	LOCATE	PRINT	TIME
CLOAD	ERROR	LOF	PSET	TO
CLOSE	EXP	LOG	PUT	TROFF
CLS	FIELD	LPOS	READ	TRON
CMD	FILES	LPRINT	REM	USING
COLOR	FIX	LSET	RENUM	USR
CONT	FN	MAX	RESTORE	VAL
COPY	FOR	MERGE	RESUME	VARPTR
COS	FPOS	MID$	RETURN	VDP
CSAVE	FRE	MKD$	RIGHT$	VPEEK
CSNG	GET	MKI$	RND	VPOKE
CSRLIN	GOSUB	MKS$	RSET	WAIT
CVD	GOTO	MOD	RUN	WIDTH
CVI	HEX$	MOTOR	SAVE	XOR
CVS	IF	NAME	SCREEN	
DATA	IMP	NEW	SET	
DEF	INKEY$	NEXT	SGN	

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

DE SLEUTELWOORDEN OP ALFABETISCHE VOLGORDE

Hieronder volgen alle MSX sleutelwoorden. De sleutelwoorden aangegeven met D/D komen in dit handboek voor. Alle andere sleutelwoorden kunt u vinden in het MSX-2 BASIC handboek.

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 LEERBOEKEN
door Wessel Akkermans en Piet den Heijer
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 bestudering 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. Een unieke serie leerboeken voor een ieder die meer over MSX wil weten en het betere werk met zijn computer wil maken.

MSX Basic leerboek deel 1 - ISBN 90 6398 649 1
Opdrachten bij deel 1 - ISBN 90 6398 596 7
MSX Basic leerboek deel 2 - ISBN 90 6398 769 2
Opdrachten bij deel 2 - ISBN 90 6398 556 8
MSX DOS leerboek deel 3 - ISBN 90 6398 519 3
Opdrachten bij deel 3 - ISBN 90 6398 516 9

MSX Verder uitgediept door H. Klopper
Eindelijk een Nederlandstalig boek over het altijd in de mist gehulde onderwerp – PEEKS EN POKES. In dit boek staan alle belangrijke RAM en VRAM adressen. De video chip en zijn registers worden volledig uitgelegd. Maar ook hoe men een machinetaal programma van cassette naar disk kan schrijven. Bovendien een diskloader utility en een uiterst geavanceerde programma beveiliging. Tenslotte zijn er een aantal interessante programma's opgenomen, waaronder een wereldkaart, waarmee verder kan worden geëxperimenteerd. Elke MSX gebruiker kan in dit boek iets van zijn gading vinden en nieuws leren.

ISBN 90 6398 447 2

MSX Machinetaal handboek door H. Klopper en M. Le Belle
Hoewel een MSX computer over een krachtig Basic beschikt, is het toch handig tijdens het programmeren de grondbeginselen van machinetaal te kennen. Daarvoor is dit boek een goede gids. De zaken worden niet puur theoretisch maar ook aan de hand van duidelijke voorbeelden, die direkt bruikbaar zijn, uitvoerig uitgelegd. Enkele onderwerpen zijn verder – scroll routine –machinetaal software (ook in disk Basic) op casette zetten –disassembler –Z80 essembler instrukties –lijst van ROM-routines –alle hook-adressen –bespreking van Basic tokens en een kompleet token-overzicht. Hèt handboek voor iedere MSX programmeur die zijn computer ten volle wil benutten.

ISBN 90 6398 735 8

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

MSX 2 Uitbreidings handboek

De MSX standaard is feit. MSX computers veroveren Nederland.
Geen wonder; met MSX is er eindelijk een standaard opgestaan tussen de meer dan drie honderd verschillende Basic dialekten die er nu nog zijn.
Stelt u zich eens voor. U koopt een computer van merk X en een floppy eenheid van merk Y. U schaft zich daarbij programmatuur aan van merk Z. Thuisgekomen zet u uw computer in elkaar, laadt de software en... alles werkt!
Tot voorkort ondenkbaar, maar met MSX een feit!
Na het sukses van MSX versie 1 is er nu een MSX 2. Deze standaard omvat het MSX 1 volledig, maar biedt daarbij nog meer. Vooral op grafisch gebied zijn er veel meer mogelijkheden gekreëerd. Daarbij maakt de mogelijkheid van 80 tekens per regel uw MSX 2 computer wel erg professioneel...
Net zoals eerder het geval was met MSX 1, zorgt uitgeverij Stark-Texel ook met MSX 2 weer voor een serie uitgebreide, volledige en duidelijke nederlandse handboeken.

Dit tweede handboek uit een serie van drie omvat een volledige behandeling van het MSX 2 Disk Basic en het MSX DOS operating system. Voorafgaand hieraan wordt een zeer duidelijke inleiding gegeven tot de fenomenen disk en operating system. Het handboek wordt gekompleteerd met een hoeveelheid praktische tabellen, duidelijke afbeeldingen en zinvolle voorbeelden.
Door de duidelijke behandeling per Disk en DOS sleutelwoord en de vele voorbeelden beantwoordt dit handboek aan bijna elke vraag en vormt het een standaardwerk dat naast elke MSX 2 Disk computer zou moeten liggen.

ISBN 90 6398 222 4 uitgeverij Stark-Texel