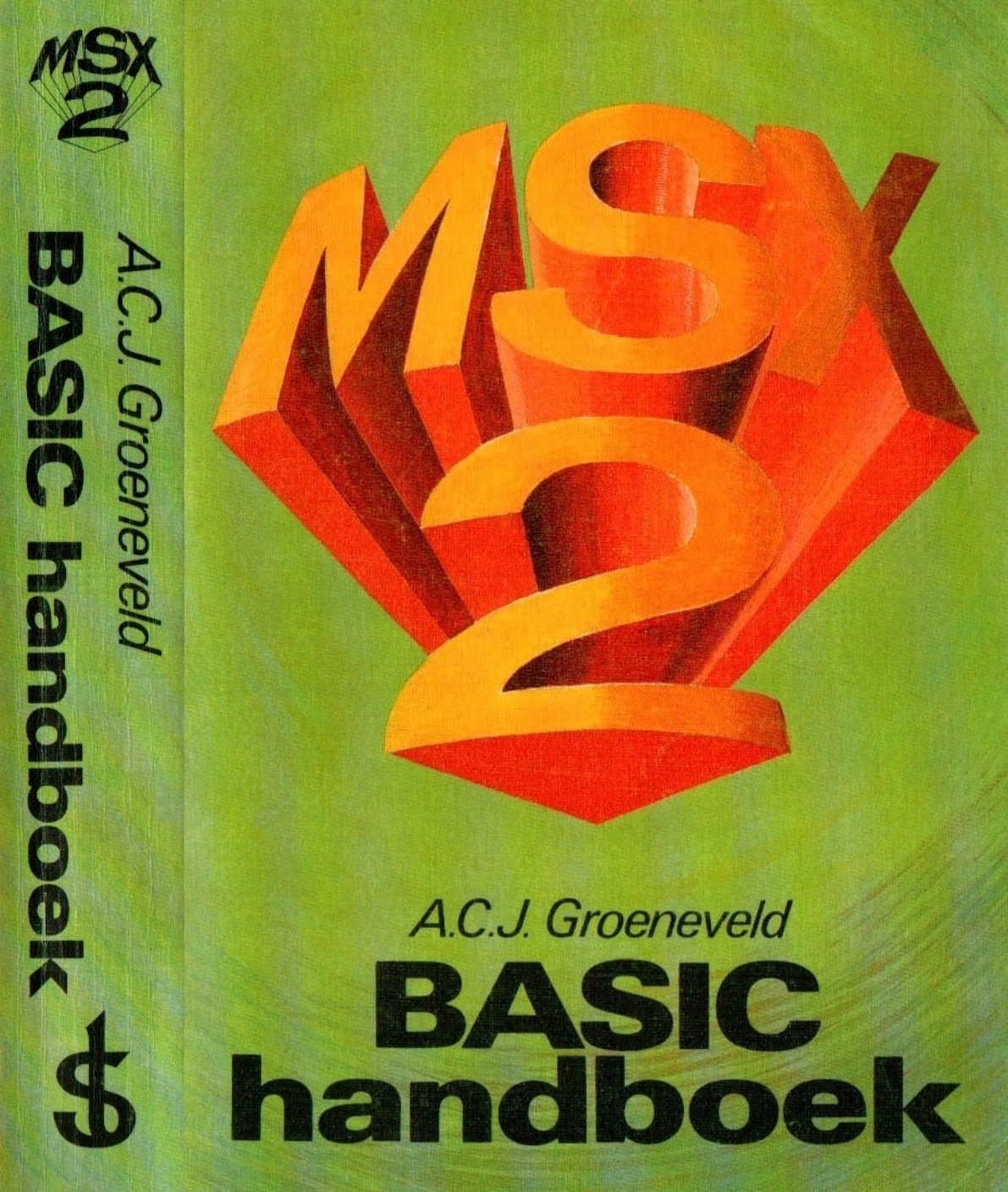

MSX
MSX2
A.C.J. Groeneveld
BASIC handboek
A.C.J. Groeneveld
BASIC
handboek

MSX 2
BASIC handboek

MSX 2

BASIC 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
Basic handboek / A.C.J. Groeneveld.
ISBN 90-6398-221-6
SISO 365.3 UDC 681.3.06:800.92 Basic
Trefw.: MSX 2-Basic (programmeertaal).

---

februari 1986
ISBN 90 6398 221 6

by uitgeverij Stark-Texel, Oosterend Nh.

INHOUD pag.

INLEIDING

MSX is een afkorting van MicroSoft eXtended basic. Met de naam MSX heeft MICROSOFT, het toonaangevende softwarehuis op het gebied van microcomputers, een zéér belangrijke en bitter noodzakelijke stap gedaan in de richting van de standaardisering van hobby- en microcomputers en hun taal.

Eindelijk is het mogelijk om apparatuur en programmatuur aan te schaffen die op vele merken micro-computers werkt; een behoefte die reeds jaren bestond.

MSX is meer dan alleen maar een software-voorschrift. MSX is een volledig standaard concept voor microcomputers. Niet alleen de taal maar ook de opbouw van het toetsenbord, de te gebruiken chips, de te hanteren geheugentechnieken en andere zaken zijn door het MSX-concept voorgeschreven. En hierdoor is het echt mogelijk om allerlei zaken aan elkaar te koppelen. Als er maar MSX op staat.

Nadat MICROSOFT in 1983 voor het eerst het begrip MSX lanceerde, heeft de MSX-computer een grote vlucht genomen. Reeds in 1984 kwamen er vele computers op de markt met het predikaat MSX. In 1985 waren er alleen op de nederlandse markt al tientallen verschillende merken MSX-computers te verkrijgen.

Eind 1985 lanceerde MICROSOFT het MSX-2, een concept dat veel uitgebreider is dan het 'oude' MSX-1 en dat gebruik op professioneel gebied zeer goed mogelijk maakt. Eén van de meest opvallende kenmerken van MSX-2 is, dat het beeldscherm nu ondermeer de professionele 80 bij 24 indeling kent. Maar MSX-2 biedt veel meer...

Belangrijk bij dit alles is, dat MSX-1 „upwards compatible" is met MSX-2. Alle MSX-1 software werkt dus ook op de MSX-2 computer. Zelfs de meeste randapparatuur (floppy disk!) die voor gebruik met een MSX-1 computer werd gekocht, werkt ook met een MSX-2 computer.

Dit handboek heeft tot doel, de steun en toeverlaat te zijn voor elke MSX-programmeur. De amateur maar ook de professionele programmeur vindt in dit handboek een duidelijk, overzichtelijk en nederlands naslagwerk. Het handboek behandelt de MSX-standaard, geeft wat

bruikbare tabelinformatie, behandelt de videochip en de soundchip maar gaat vooral zéér diep in op het belangrijkste gebied; het MSX-basic. Van dit basic wordt de exacte schrijfwijze en betekenis per kommando behandeld en wordt het één en ander met duidelijke, nederlandse voorbeelden toegelicht.

Om ook de nieuwkomer in de computerwereld van dienst te zijn, worden algemene zaken als konstanten, variabelen en uitdrukkingen breedvoerig behandeld.

Doordat een aparte tabel is opgenomen waarin de sleutelwoorden op geadviseerde leer-volgorde zijn opgenomen, kan dit handboek ook uitstekend dienen als een complete instruktie voor de beginnende programmateur.

Waar er verschillen tussen het MSX-1 en het MSX-2 zijn, wordt dit in dit handboek duidelijk aangegeven. Hierdoor is dit handboek zowel voor de MSX-1- als voor de MSX-2-computerbezitter volledig bruikbaar.

Dit handboek is het eerste deel van een serie bestaande uit drie handboeken. In het tweede deel van deze serie wordt het MSX-2 behandeld in verband met de schijveneenheid en andere mogelijke uitbreidingen. Zowel het MSX-2 disk-basic als het MSX-DOS operating system worden in dit tweede deel ondermeer uitgebreid behandeld.

In het derde deel tenslotte zijn een hoeveelheid praktische toepassingen, tabellen en tips opgenomen. Ondermeer komt in dit deel de opbouw van het MSX-ROM aan de orde. Ook zijn in dit deel een aantal programma's opgenomen waarmee u uw eigen programma's kunt onderzoeken, uw schijven blok voor blok kunt bekijken, enzovoorts.

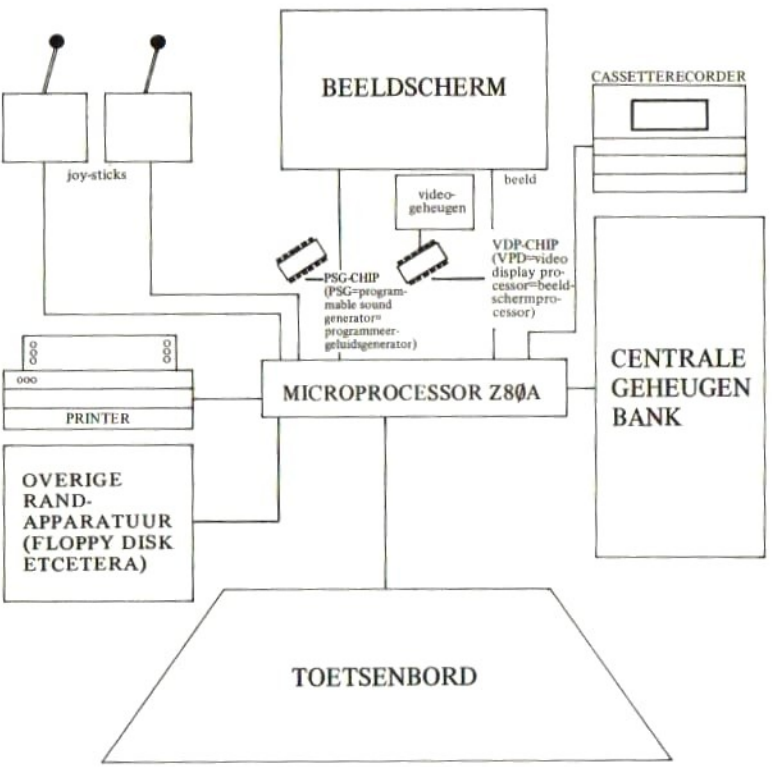
Ik hoop dat dit boek, samen met de andere twee uit de handboekenserie, een welkome ondersteuning voor vele MSX-programmateurs mag zijn.

januari 1986,
A.C.J. Groeneveld.

Zoals in de inleiding reeds werd opgemerkt, beperkt de MSX standaard zich niet alleen tot de BASIC taal. Ook het hardware-concept wordt duidelijk door deze standaard bepaald.

2.1 De hardware van een MSX-computer

Hieronder volgt een schema van de algemene opbouw van een MSX-computer.



Als standaard microprocessor is binnen de MSX-standaard gekozen voor de Z-80-A microprocessor (of een processor met gelijke specificaties). De Z-80 is de meest verbreide microprocessor en vormt al jaren het hart van vele microcomputers. Alhoewel de Z-80 een 8 bits microprocessor is en door velen als ouderwets wordt betiteld ten opzichte van de 16 bits microprocessoren, voldoet deze chip nog steeds uitstekend op hobby- en semi-professioneel gebied. De verwachting is, dat in de toekomst nog vele nieuwe computermodellen zullen worden ontworpen op basis van de Z-80 microprocessor.

Als geluidsgenerator is gekozen voor de AY-3-8912 geluidsprocessor (of een processor met gelijke specificaties). Deze processor is een computer op zich en regelt alle geluidseffecten van de MSX-computer. Ook voor deze geluidsgenerator geldt dat het een oude bekende is binnen de computerwereld. Echter, door zijn grote aantal mogelijkheden voldoet ook deze processor uitstekend. De AY-3-8912 bezit drie geluidskanalen, één ruisgenerator en een variëteit van effecten. Hierdoor is het bijvoorbeeld goed mogelijk om een driestemmig stuk muziek met slagwerk te programmeren.

Als beeldschermprocessor werd binnen de MSX-1 standaard gekozen voor een TMS 9918A processor (of een processor met gelijke specificaties). Deze processor heeft de beschikking over een eigen video-geheugen van 16 kilobytes waarin alle beeldschermgegevens worden opgeslagen. Hierdoor wordt het vaak toch al zo schaarse centrale geheugen gespaard.

Binnen de MSX-2 standaard werd gekozen voor een andere beeldschermprocessor: de V9938. Deze beeldschermprocessor heeft standaard een eigen video-geheugen van 64 kilobytes (!). Binnen de MSX-2 standaard kan dit video-geheugen eventueel zelfs tot 128 kilobytes worden uitgebreid! MSX-2 biedt ten opzichte van MSX-1 dan ook wel ongekende mogelijkheden op grafisch gebied!

Als magnetisch opslagmedium gaat de MSX-standaard uit van een heel gewone cassetterecorder. Op een normale geluidscassette worden programma's gezet en kunnen ook andere gegevens worden geplaatst. Bijzonder is dat de MSX-standaard uitgaat van een relais dat de recordermotor aan en uit kan schakelen. Dit relais is via het MOTOR-commando ook in MSX-BASIC te besturen; de handige programmeur kan op deze wijze dit relais ook voor andere besturingsdoeleinden gebruiken zoals bijvoorbeeld de besturing van een diaprojector.

Uiteraard voorziet de standaard in de aansluiting van een printer. Deze aansluiting geschiedt via het standaard Centronics parallelle protocol. De printer kan een afdrukeenheid zijn met de mogelijkheid om de standaard MSX grafische symbolen af te drukken maar mag ook een algemene printer zonder deze mogelijkheden zijn.

De joy-sticks vormen onmisbare attributen voor de spelletjes-fanaat! Als zodanig ontbreken deze niet in de MSX-standaard. Programmeurs die spelletjes in MSX-basic willen schrijven, vinden een keur van eenvoudige commando's, waarmee de joy-sticks volledig kunnen worden uitgebuit.

MSX ondersteunt de aansluiting van één of meer schijveneenheden. In sommige gevallen is een schijveneenheid reeds in de computer ingebouwd. In deel twee van deze handboekenreeks wordt deze schijveneenheid uitgebreid behandeld.

Als beeldscherm kan natuurlijk een mooie, dure kleurenmonitor worden gebruikt. Vooral met MSX-2, waarin 256 verschillende kleuren kunnen worden aangestuurd, is zo'n beeldscherm natuurlijk erg bruikbaar.

Echter, een eenvoudige zwart-wit-monitor of zelfs een draagbaar televisietoestel kunnen ook uitstekend voldoen.

De MSX-standaard voorziet in een gecombineerde audio/video-uitgang voor monitoren en een hoogfrequente tv-uitgang voor aansluiting van een televisietoestel.

De MSX-2 standaard voorziet daarbij ook nog in een zogenaamde RGB-aansluiting (RGB staat voor Rood/Groen/Blauw). Met deze aansluiting kan een monitor aan de MSX-2 computer worden gekoppeld waarbij de computer dan direkt de drie electronenkanonnen in de beeldbuis van de kleurenmonitor aanstuurt. Hiermee kan een zeer stabiel en uiterst scherp beeld worden verkregen; iets dat met de uitgebreide grafische mogelijkheden van de MSX-2-computer heel welkom is...

2.2 Het geheugen van een MSX-computer

Het geheugen van een computer heeft een bepaalde grootte, die men uitdrukt in bytes, kilobytes of megabytes.
Een byte is een geheugen-eenheid waarin één enkel karakter kan

worden opgeslagen. Om bijvoorbeeld de tekst "MSX-BASIC" in het computergeheugen op te slaan, zijn negen geheugeneenheden ofwel negen bytes benodigd.

Indien men over geheugengrootten spreekt, is het gebruikelijk om deze uit te drukken in kilobytes. Het merkwaardige is, dat in de computerwereld een kilobyte niet 1000, maar 1024 bytes telt. Indien men het over 32 kilobytes of 32 Kb heeft, dan bedoelt men dus een geheugengrootte van 32768 bytes. In dat geheugen kunnen dus 32768 lettertekens (of 32768 machinecode-instructies, zie hoofdstuk 4) worden opgeslagen.

Een megabyte telt weer 1024 kilobytes. Binnen de microcomputerwereld heeft men het meestal pas over megabytes, indien men het over de opslagcapaciteit op magneetschijf of diskette heeft.

Het centrale geheugen van een MSX-computer bestaat in principe uit vier geheugenbanken van elk 16 kilobytes. Deze geheugenbanken zijn genummerd vanaf 0 tot en met 3.

Per geheugenbank kan steeds één van de vier mogelijke slots tegelijk worden 'aangesloten'. De MSX-computer kan zélf kiezen, welk slot aangesloten moet worden en welke andere drie slots dus automatisch buiten beschouwing blijven.

Zodoende kan de MSX-computer 64 kilobytes geheugen selekteren uit een totaal aanbod van maximaal 256 kilobytes.

De MSX-2-computer kan per slot nog eens vier verschillende 'subslots' selekteren waardoor het beschikbare geheugen 1024 kilobytes (1 megabyte) kan bedragen!

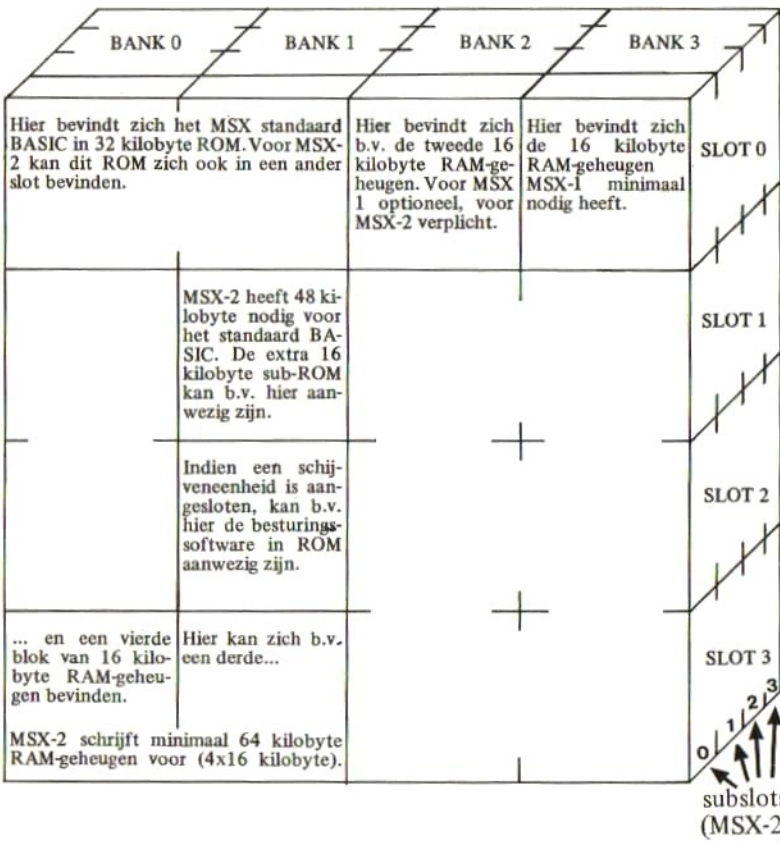
De MSX-1 standaard schrijft minimaal de aanwezigheid van 32 kilobyte ROM (Read Only Memory = leesbaar, onuitwisbaar geheugen) voor. In dit ROM is het MSX-basic gekodeerd; dankzij dit ROM-geheugen kan er op de MSX-computer in BASIC worden geprogrammeerd.

De MSX-1 standaard schrijft daarbij minimaal 16 kilobytes RAM (Random Access Memory = vrij toegankelijk geheugen) voor. In dit geheugen wordt ondermeer later uw BASIC-programma opgeslagen. De meeste MSX-1 computers hebben echter standaard 64 kilobytes RAM waarvan er 32 kilobytes onder MSX-basic beschikbaar zijn.

De MSX-2 standaard schrijft 48 kilobytes ROM voor waarin het MSX-2-basic is gekodeerd. Daarbij dienen er bij MSX-2 minimaal 64 kilobytes RAM aanwezig te zijn. Die 64 kilobytes zijn in basic volledig bereikbaar doordat 32k van dit geheugen als RAM-disk te gebruiken is.

Door middel van insteekcassettes (een floppy-disk-eenheid, een spelletje, enz.) kunnen blokken geheugen met daarin ROM of RAM worden toegevoegd aan het beschikbare MSX-geheugen.

Het volgende schema geeft aan, hoe de verschillende geheugenbanken zijn ingedeeld of kunnen worden ingedeeld.



Buiten dit schema valt het video geheugen. De MSX-1 standaard schrijft 16 kilobytes video-geheugen (RAM) voor. In dit geheugen worden alle beeldschermgegevens bewaard.
De MSX-2 standaard schrijft 64 kilobytes video-geheugen voor. Dit video-geheugen kan eventueel naar 128 kilobytes worden uitgebreid.

Het video-geheugen heeft een aparte koppeling naar de beeldschermprocessor en valt buiten de organisatie van het centrale geheugen van de MSX-computer.

2.3 MSX-BASIC en het RAM

Het MSX-BASIC gebruikt het beschikbare RAM op een wijze die is toegelicht in het schema op de volgende bladzijde.

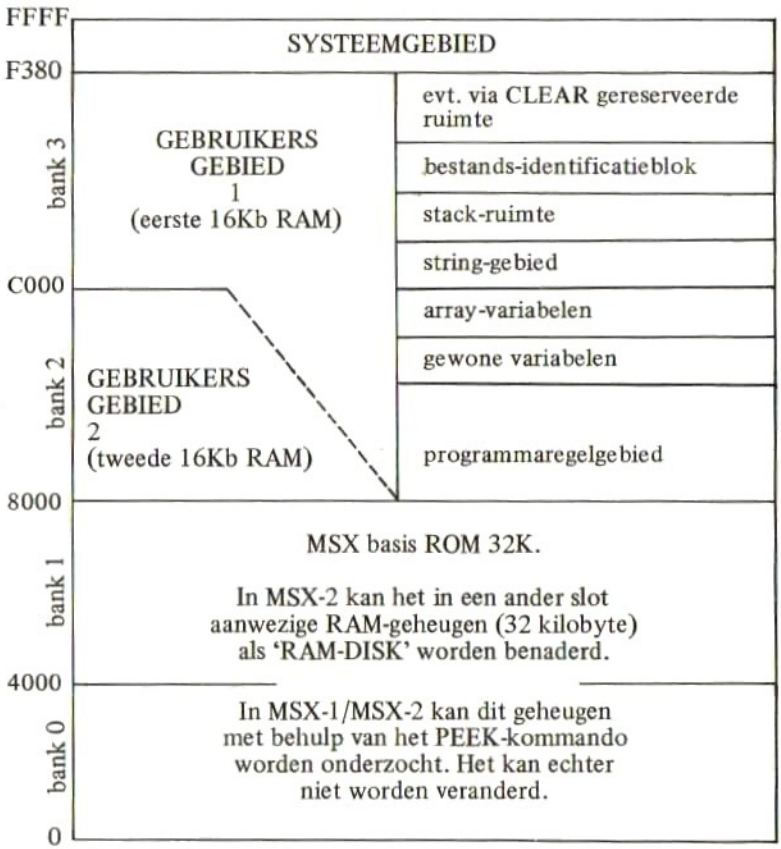
Het allerbovenste gedeelte is altijd gereserveerd voor het MSX systeem. Dit gedeelte dient als kladblaadje voor interne gegevens, dat het systeem zo nu en dan nodig heeft.

Onder dit systeemgedeelte volgen diverse blokken die elk moment, afhankelijk van het werkende programma, anders van grootte kunnen zijn.

Van boven naar beneden komen we als volgende blok de via CLEAR gereserveerde ruimte tegen. Deze ruimte kan binnen de BASIC-taal worden gecreëerd via het CLEAR sleutelwoord. Deze ruimte kan door de gevorderde programmeur vervolgens worden gebruikt voor het opslaan van routines, geschreven in de Z-80 machinetaal. Deze routines kunnen dan weer vanuit de BASIC-taal worden aangeroepen door middel van het DEFUSR en het USR sleutelwoord. In het normale geval bestaat dit blok niet; onder het systeem volgt dan onmiddellijk het bestandsidentificatieblok.

Het bestandsidentificatieblok bevat gegevens die het MSX-BASIC nodig heeft bij het uitwisselen van gegevens met bijvoorbeeld een cassetterecorder of een floppy disk eenheid.

Het stringgebied bevat de alfanumerieke gegevens (gegevens, bestaande uit letters, cijfers en leestekens) waarmee het betreffende programma manipuleert.



De stack-ruimte (stack=stapel) bevat een aantal gegevens in verband met de besturing van het programma. Later, bij de behandeling van bijvoorbeeld het GOSUB sleutelwoord zal blijken dat het benodigde terugkeeradres een gegeven is dat in deze stack-ruimte wordt opgeslagen.

Hieronder bevinden zich de array-variabelen. De namen, specificaties en numerieke inhoud liggen in dit gebied opgeslagen. De alfanumerieke inhoud van de array-variabelen ligt echter in het stringgebied opgeslagen. Voor de betekenis van variabelen: zie hoofdstuk 6.

Onder de array-variabelen bevinden zich de niet-array-variabelen. Deze variabelen zijn op een dergelijke manier binnen dit gebied opgeslagen.

Als laatste blok volgt het blok met programmaregels. Het feitelijke BASIC-programma ligt in dit blok opgeslagen.

Indien er nog geheugen over is, ligt dat tussen het programmaregelgebied en het gewone variabelen-gebied.

Onder het programaregelgebied ligt het 32 kilobytes grote MSX basis ROM. Dit geheugen kan alleen met het PEEK-kommando onder basic worden benaderd en kan (uiteraard) niet worden veranderd.

Onder MSX-2 kan het in een ander slot aanwezige 32 KB RAM-disk worden benaderd. Dit stuk geheugen wordt dan benaderd als ware het een schijven-geheugen. Programma's en gegevens kunnen in dit geheugen dan tijdelijk worden opgeslagen en later weer worden opgeroepen. Wanneer de MSX-2 computer echter wordt uitgezet, gaat de inhoud van dit geheugen verloren. De RAM-disk is dus alleen voor zeer tijdelijke opslag geschikt.

2.4 Het toetsenbord van een MSX-computer

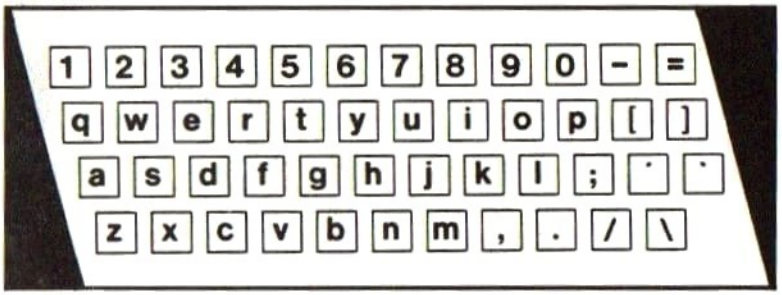
Zoals bij de meeste microcomputers het geval is, bezit ook de MSX-computer een toetsenbord met een standaard indeling. In Nederland is het zogenaamde QWERTY-toetsenbord het meest gebruikt.

Het toetsenbord valt onder te verdelen in drie soorten toetsen, te weten de karaktertoetsen, de controletoetsen en de functietoetsen.

De karaktertoetsen genereren bij intoetsing de vermelde letters en cijfers; op het eerste gezicht niets bijzonders dus. Echter, in combinatie met de functietoetsen SHIFT, CODE en GRAPH zijn er veel meer tekens te genereren dan op het eerste gezicht duidelijk is. Om deze tekens uit te proberen behoeft men slechts de computer aan te zetten en te wachten op de copyrightmelding, gevolgd door de Ok-melding. Hierna kunnen en mogen alle toetsaanslagen worden uitgeprobeerd. Let u in eerste instantie niet op de vele foutmeldingen die de MSX-computer kan geven als protest op uw door de computer niet begrepen intoetsingen.

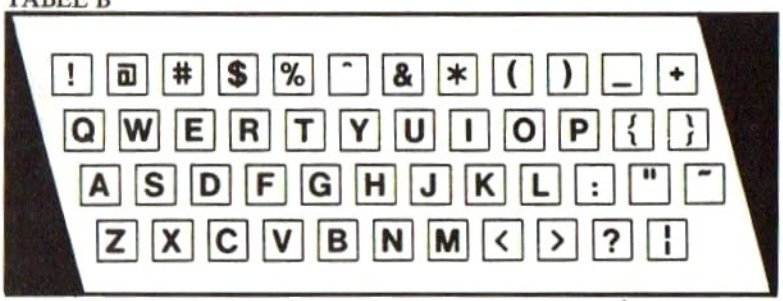
De zes schema's op de volgende twee pagina's geven de karakters weer die via het toetsenbord kunnen worden gegenereerd.

TABEL A



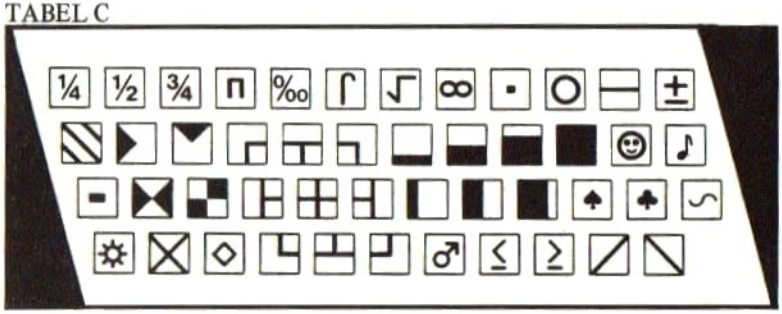
Karakters die normaal ontstaan bij het intoetsen.

TABEL B



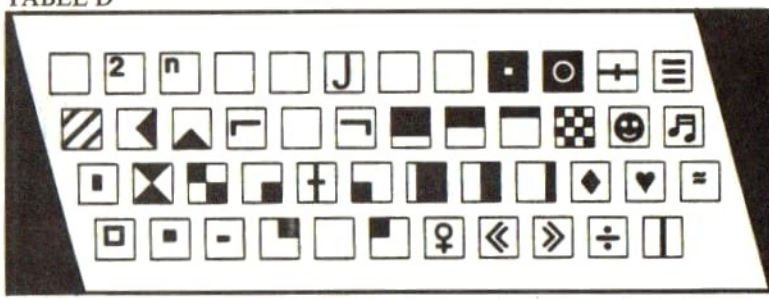
Karakters die ontstaan indien de toetsen tegelijk met de SHIFT-toets worden ingedrukt.

TABEL C



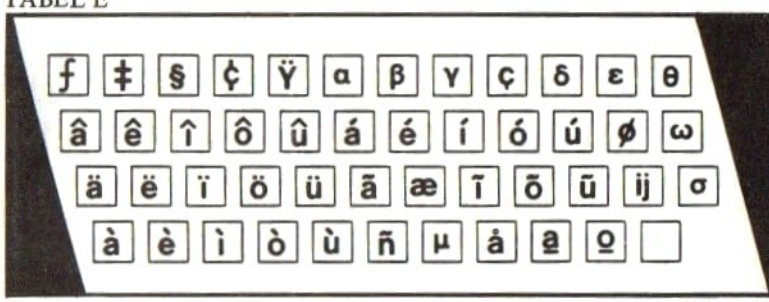
Karakters die ontstaan indien de toetsen tegelijk met de GRAFH-toets worden ingedrukt.

TABEL D



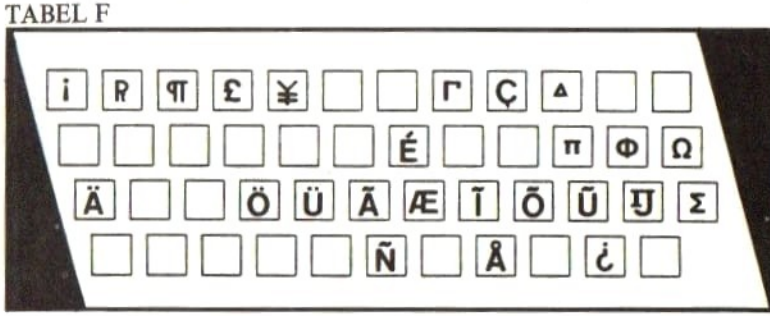
Karakters die ontstaan indien de toetsen tegelijk met de SHIFT- en GRAFH-toets worden ingedrukt.

TABEL E



Karakters die ontstaan indien de toetsen tegelijk met de CODE-toets worden ingedrukt.

TABEL F



Karakters die onstaan indien de toetsen tegelijk met de SHIFT- en CODE-toets worden ingedrukt.

Buiten deze uitgebreide set van karakters kunnen ook acties aan de MSX-computer worden ontlokt. Dit gebeurt via de zogenaamde controletoetsen. De MSX-standaard vereist de aanwezigheid van de volgende controletoetsen:

Toets	Functie
ESC	Kan in programma's worden gebruikt, maar heeft binnen het MSX-basic geen functie.
TAB	Met deze toets kunnen in één keer 8 spaties worden ingegeven.
CTRL	Deze toets heeft op zichzelf geen functie. Echter in combinatie met andere toetsen kunnen allerlei controlefuncties worden opgeroepen. Deze worden behandeld in hoofdstuk 3.
SHIFT	Deze toets heeft op zichzelf geen functie. Echter in combinatie met andere toetsen kunnen hoofdletters, leestekens en speciale tekens worden gegenereerd. Zie hiervoor de eerder geplaatste tabel B.
CAPS LOCK	Met deze toets kan de SHIFT-functie worden vastgezet zodat deze niet steeds behoeft te worden vastgehouden wanneer men bijvoorbeeld een groot stuk tekst in hoofdletters wenst in te tikken. De CAPS LOCK toets wordt ook wel eens met het teken ⇧ aangegeven. Wanneer de shift-toets is vastgezet, gaat er een speciaal CAPS LOCK lampje branden.
GRAPH	In combinatie met andere toetsen geeft deze toets toegang tot een extra set karakters zoals eerder in tabel C opgenomen.
CODE	In combinatie met andere toetsen geeft deze toets toegang tot de karakterset zoals opgenomen in tabel E.

N.B.: Door de controletoetsen SHIFT-GRAPH en SHIFT-CODE te combineren, kunnen de karakters zoals opgenomen in de tabellen D en F worden gegenereerd.

RETURN	Met deze toets worden ingaven op het toetsenbord bevestigd. Pas na ingave van RETURN 'weet' de MSX-computer dat de ingave is beëindigd en wordt de ingave op juistheid gecontroleerd. Indien u zo maar wat met het toetsenbord speelt, bijvoorbeeld om de karakterset uit te proberen, merkt u dat de computer pas na ingave van de RETURN-toets een eventuele foutmelding geeft.

SELECT Binnen het MSX-basic heeft deze toets geen speciale functie. Hij kan echter in programma's wel worden gebruikt.

BS Met deze toets kan het laatst ingetikte karakter weer worden verwijderd. Deze toets wordt ook wel met BACK-SPACE aangegeven.

CLS/HOME Op zichzelf gebruikt, zorgt deze toets ervoor, dat de cursor (het blokje op het scherm dat aangeeft waar de ingave is gebleven) links boven in de beeldschermhoek komt te staan. In combinatie met de SHIFT-toets wordt daarbij ook nog het gehele beeldscherm uitgewist.

INS Met behulp van deze toets kan in een bestaande regel een karakter of een stuk tekst worden tussengevoegd. Door een intoetsing wordt het tussenvoegen geactiveerd; door nog een intoetsing of door gebruik van één van de pijltoetsen wordt het tussenvoegen weer gedeactiveerd. De cursor verandert in een streepje wanneer het tussenvoegen actief is.

DEL Met behulp van deze toets kan één karakter worden verwijderd op elke plaats binnen een regel. De rest van de regel wordt dan aangeschoven.

STOP Met deze toets kan een werkend BASIC-programma worden stilgezet. Een volgende intoetsing zorgt ervoor dat het betreffende programma dan weer verder gaat. In combinatie met de CTRL-toets kan met deze toets een programma worden onderbroken.

PIJLTOETSEN Met deze toetsen kan de cursor in horizontale en vertikale richting worden verplaatst. Door intoetsing van twee toetsen tegelijk kan de cursor ook diagonaal (schuin) worden verplaatst. De pijltoetsten kunnen in een BASIC-programma te zamen als joy-stick dienst doen.

Alle functietoetsen zijn gekenmerkt door een afgekorte, Engelstalige naam. Ter verduidelijking volgt voor elke toets hier een verklaring van de naam:

ESC	ESCape=ontsnappen
TAB	TABulation=tabulatie (kolomindeling)
CTRL	ConTRoL=controle
SHIFT	SHIFT=opschuiven
CAPS	CAPitalS=hoofdletters

GRAPH	GRAPHics=grafische tekens
CODE	CODE=codering
RETURN	RETURN=terugkeren (naar de linkerkant van het beeldscherm, één regel lager)
SELECT	SELECT=selecteren, uitkiezen
BACKSPACE	BACKSPACE='terugspatie' een spatieteken wordt de 'verkeerde' kant uit over het foute karakter gezet.
CLS	CLear Screen=beeldscherm schoonmaken
HOME	HOME=huis. De cursor wordt naar 'huis' teruggestuurd.
INS	INSert=tussenvoegen
DEL	DELete=verwijderen
STOP	STOP=stoppen, stilstaan

De functietoetsen zijn genummerd van 1 tot en met 10. 'Onder' deze functietoetsen zijn vaste teksten opgenomen die veelvoudig terugkeren. Door middel van de intoetsing van de betreffende functietoets wordt dan de gehele tekst op beeldscherm geplaatst. De functietoetsen kunnen via het MSX-basic eventueel van functie worden veranderd. De functies worden (gedeeltelijk) onder in het beeld zichtbaar gemaakt. Deze regel is eventueel uitschakelbaar; ook dit kan in MSX-basic geschieden.
De functies 6 tot en met 10 worden opgeroepen door de functietoetsen te zamen met de SHIFT-toets te gebruiken.

De MSX-standaard voorziet in een full screen editor. Een editor is de Engelse naam voor een tekstverzorgend programma. Full screen wil zeggen dat tekst over het gehele beeld kan worden verzorgd.

Om wat met de MSX full screen editor te oefenen, is nog geen enkele kennis van het MSX-basic noodzakelijk. Slechts enkele gegevens zijn in dit stadium belangrijk:

1) Begin een regel tekst altijd eerst met een regelnummer dat groter is dan of gelijk is aan 0. Het regelnummer moet kleiner dan 65530 blijven.
2) Een ingetikte regel mag groter zijn dan de breedte van een beeldschermregel, maar moet kleiner blijven dan 256 tekens.
3) Een regel mag voorlopig elke volgorde van karakters bevatten.
4) Een regel dient altijd te worden beëindigd door de RETURN-toets.
5) Een op een of andere wijze veranderde regel wordt pas definitief opgenomen nadat een RETURN is ingegeven.

Een geldige tekst om in de MSX-editor mee te oefenen, is bijvoorbeeld:

```
1 IK OEFEN WAT MET DE MSX EDITOR
10 HDJS JSJS HSHSHSHS TETETE
123 IK TIK ZOMAAR WAT IN ...
```

maar elke zelf bedachte (nonsens-) tekst is uitstekend.

3.1 LIST

Nadat u een aantal regels op deze wijze heeft ingetikt, kunt u deze regels op volgorde van regelnummer weer opvragen door middel van het LIST-commando. Tikt u het woord LIST (hoofdletters of kleine letters maken hier geen verschil) in, gevolgd door de RETURN-toets. In een vaartje ziet u de regels keurig en in volgorde over het beeldscherm gaan. Maakt u zich niet ongerust over verdwijnende regels op het beeldscherm; deze zijn geregistreerd en kunnen door het list-commando weer worden opgeroepen.
Indien u slechts één regel wilt opvragen, tikt u dan LIST in, gevolgd door het betreffende regelnummer en een RETURN-toets. Een beperkte

hoeveelheid tekst kan worden opgeroepen door het LIST-commando in te toetsen, gevolgd door een eerste en een laatste regelnummer, door middel van een min-teken van elkaar gescheiden.
LIST biedt nog meer mogelijkheden, maar die zijn in het kader van dit hoofdstuk niet belangrijk.

3.2 De pijltoetsen, NEW

Op de boven beschreven wijze kunt u het beeldscherm naar believen met tekst vullen. De cursor, het blokje op het beeldscherm dat aangeeft waar u met intikken gebleven bent, loopt steeds keurig met uw ingave mee.
De cursor kunt u met behulp van de pijltoetsen in de aangegeven richtingen verplaatsen. Zo kunt u de cursor bijvoorbeeld midden in een stuk tekst zetten. Als u daarna wat letters intikt, dan komen die in de plaats van de lettertekens die eerder op die plaats stonden. Wanneer u na de verandering een RETURN-toets ingeeft, dan merkt u dat de cursor aan het begin van de volgende regel gaat staan. De door u aangebrachte wijziging is opgenomen in het programmageheugen. U kunt dit met het LIST-commando natuurlijk nog even controleren.

De pijltoetsen maken het mogelijk om op willekeurige plaatsen op het scherm wijzigingen aan te brengen en deze vervolgens permanent te maken.

Een andere mogelijkheid is het copiëren van regels. Geef bijvoorbeeld eens de tekst NEW in, gevolgd door een RETURN. Controleer daarna met een list of alle regels daadwerkelijk door de computer 'vergeten' zijn (NEW betekent nieuw).
Tik hierna in (en vergeet de RETURN niet):

```
10 DE EERSTE REGEL VAN VELE
```

Ga vervolgens met de pijltoetsen naar het regelnummer 10 en maak hier een regelnummer 20 van door alleen de 1 in een 2 te veranderen. Geef vervolgens (uiteraard) een RETURN in en doe daarna een LIST. U ziet dat de regel nu twee maal is opgenomen, eenmaal onder regelnummer 10 en eenmaal onder regelnummer 20.
Nu kan regel 20 bijvoorbeeld worden aangepast door met de pijltoetsen naar de juiste plaats te gaan en de tekst EERSTE te vervangen door de tekst TWEEDE. Na RETURN is ook deze verandering weer definitief.

Indien een regel niet meer is gewenst, kan deze worden verwijderd door alleen het regelnummer in te geven, gevolgd door een RETURN. Ook deze verwijdering kunt u natuurlijk weer controleren met de LIST-functie.

3.3 Verwijderen van tekens

Laten we nog eens een NEW-commando ingeven (gevolgd door de RETURN) en vervolgens met een SHIFT-CLS (of een CTRL-L) het beeldscherm schoonmaken. We beginnen met een schone lei. Als eerste regel tikken we in:

```
1 DIT IS DAN DE EERSTE REGEL
```

Nu gaan we om te oefenen het woordje DAN uit deze eerste regel verwijderen. Dit doen we door met de pijltoetsen naar de letter D van DAN te gaan en vervolgens vier keer de DEL-toets in te drukken. De DEL-toets dient vier keer te worden ingedrukt omdat ook het spatieteken na DAN moet worden verwijderd.

Op deze wijze kunnen stukken tekst worden verwijderd uit een regel. Na de RETURN is de verkorte regel definitief opgenomen; met een LIST kan dat worden gecontroleerd.

3.4 Invoegen van tekens

Het kan natuurlijk ook zijn dat we juist een stuk tekst willen tussenvoegen in een bestaande regel. Als oefening kunnen we bijvoorbeeld proberen om het woordje DAN weer in de regel uit het laatste voorbeeld terug te zetten.

Hiervoor zetten we met de pijltoetsen de cursor op de letter D van het woordje DE. Tik vervolgens de INS-toets in; de cursor verandert nu in een streepje.
Tik nu het woordje DAN in en vergeet de spatie na DAN niet. Geef vervolgens een RETURN om de regel definitief vast te leggen en controleer of de uitgebreide regel daadwerkelijk is opgenomen met een LIST.

Het zal blijken dat het woordje DAN netjes is tussengevoegd. Pas bij het invoegen op de volgende punten:

1) door de INS-toets nog een keer in te drukken, wordt de invoegfunctie weer opgeheven en wordt het volgende karakter weer gewoon over het oude karakter heen geplaatst.
2) een pijltoets heft de invoegfunctie eveneens op.
3) de BACKSPACE heft de invoegfunctie **niet** op.

3.5 De overige controletoetsen

Het zal duidelijk zijn dat met de MSX-editor het intikken van programmaregels alsmede het corrigeren van deze regels een eenvoudig karwei is. Met een beperkt aantal duidelijke controletoetsen zijn vele mogelijkheden te verwezenlijken, onder andere:

– het intoetsen van een stuk tekst
– het overschrijven van een stuk tekst
– het verwijderen van een stuk tekst
– het tussenvoegen van een stuk tekst.

Behalve de nu bekende controletoetsen (RETURN, INS, DEL en de pijltoetsen) zijn er nog meer functies op te roepen die weleens gemakkelijk zijn. Eén kwamen we er al tegen: de BACKSPACE (BS) voor het corrigeren van het laatste karakter.

De overige functies laten zich slechts oproepen door middel van het intoetsen van de CTRL-toets te zamen met een ander karakter. Eén zo'n functie kwamen we reeds tegen: de CTRL-L (druk tegelijk de CTRL en de L-toets in) voor het schoonmaken van het beeld.

Hieronder volgt een schema met de overige functies van de editor. Om te oefenen kunt u het beste eerst een flink stukje tekst intikken en vervolgens één voor één de functies uitproberen.

CONTROLEFUNCTIES VOOR DE MSX-EDITOR

toets die te zamen met de CTRL-toets moet worden ingetoetst	functie van deze toets
B	de cursor gaat terug naar het begin van het vorige woord.
C	onderbreken van de ingave zonder dat de regel

	definitief wordt opgenomen.
E	alle tekst op de betreffende regel, rechts van de de cursor, wordt verwijderd.
F	de cursor gaat verder naar het begin van een volgend woord.
G	een kort geluidssignaal is hoorbaar, verder geen functie.
H	heeft dezelfde functie als BACKSPACE (BS).
I	heeft dezelfde functie als de TAB; er worden 8 spaties in één keer afgedrukt.
J	de cursor gaat naar de volgende regel. Eventueel wordt de beeldscherminhoud één regel opgeschoven (gebeurt niet met de pijl-naar-beneden-toets).
K	heeft dezelfde functie als HOME.
L	heeft dezelfde functie als CLS.
M	heeft dezelfde functie als RETURN.
N	de cursor gaat helemaal naar het einde van de regel.
R	heeft dezelfde functie als INS.
U	de volledige regel wordt schoongemaakt.
X	heeft dezelfde functie als SELECT (geen).
\	heeft dezelfde functie als pijl naar rechts.
]	heeft dezelfde functie als pijl naar links.
^	heeft dezelfde functie als pijl naar boven.
_	heeft dezelfde functie als pijl omlaag.
DEL	heeft dezelfde functie als DEL zonder CTRL.

Pas op: hoegenaamd alle controlefuncties hebben tot gevolg, dat een eventueel geactiveerde tussenvoeging wordt gede-activeerd. Dit kan men zien aan de cursor, die dan weer de vorm van een blokje (in plaats van een streepje) krijgt.

Het is raadzaam om ervoor te zorgen, enige ervaring met de MSX-editor te verkrijgen alvorens de MSX-studie te vervolgen.

4.1 Computertalen

De Z-80-A microprocessor, het hart van het MSX-computersysteem, 'begrijpt' slechts één taal, de machinetaal. In hele reeksen van getallen, variërend vanaf 0 tot en met 255, dient deze microprocessor te worden duidelijk gemaakt wat van hem wordt verwacht. Om een stukje machinetaal te voorschijn te halen, dient men slechts het volgende korte programma in te toetsen nadat de computer is aangezet en de Ok-melding is verschenen:

```
NEW
Ok
10 FOR I=1 TO 100
20 PRINT PEEK(I);
30 NEXT I
RUN
```

(denk aan de RETURN aan het einde van elke regel)

Na het intikken van het RUN-commando (met daarachter een RETURN) verschijnen op het beeldscherm allemaal getallen tussen 0 en 255. Deze getallen vormen de eerste honderd instructies die in machinetaal in het ROM zijn opgenomen.

Elke machine-instructie heeft een bepaalde taak. Deze taken zijn echter zo elementair dat er al een heel programma dient te worden geschreven alleen om twee getallen met elkaar te vermenigvuldigen. Het programmeren in machinetaal is een langdurig en ingewikkeld werk dat alleen voor gevorderde hobbyisten en specialisten is weggelegd. Zelfs professionele programmeurs zijn meestal niet in staat om een programma in machinetaal te schrijven.

Om er nu voor te zorgen dat de computer vriendelijker is te benaderen zonder dat het noodzakelijk is dat men een specialist is in het moeilijke machinetaal-programmeren, heeft men al vrij lang geleden de zogenaamde HOGERE COMPUTERTALEN bedacht. Deze hogere computertalen stellen iemand die geen specialist is op het gebied van de computeropbouw toch in staat om vrij eenvoudig gebruik te maken van de computer. Zulke hogere computertalen zijn bijvoorbeeld het PASCAL, het ALGOL, het FORTRAN, het COBOL en zeker niet in de laatste plaats het BASIC.

Om er voor te zorgen dat een computer de door de mens bedachte

hogere programmeertaal 'begrijpt', is er een programma nodig dat deze hogere programmeertaal kan vertalen naar de elementaire machinecode die de computer 'begrijpt' en kan uitvoeren. Zo'n programma dat de vertaling van een hogere programmeertaal naar de elementaire machinetaal verricht, noemt men een INTERPRETER (interprete=betekenis geven aan) of in een ander geval een COMPILER (compile=samenstellen).

Het MSX-basic is een hogere programmeertaal die een INTERPRETER nodig heeft voor de vertaling naar machinetaal. Deze interpreter is éénmaal door zéér gespecialiseerde vakmensen van MICROSOFT samengesteld en toen opgeslagen in het niet uitwisbare gedeelte van elke MSX-computer (het ROM). Deze interpreter bestaat uit enkele tienduizende machinecode-instructies waarvan we in het eerder genoemde kleine programma er honderd lieten zien.

4.2 BASIC

De naam van de taal BASIC is een afkorting van Beginners All-purpose Symbolic Instruction Code. Zoals de naam doet vermoeden, werd de taal BASIC in eerste instantie ontworpen voor beginners op het gebied van de computerwereld. Sinds de introductie van BASIC is er door diverse software-ontwerpers een keur van zogenaamde dialecten ontworpen zodat het nu niet meer mogelijk is om van een standaard BASIC te spreken. De tegenwoordige BASIC-dialecten zijn vaak zeer uitgebreid en lijken niet meer op het eerste, eenvoudige BASIC waar alle, nu meer dan 300 verschillende dialecten van af stammen.

Het MSX-basic is één van de vele BASIC-dialecten. Echter door de bijzondere kwaliteit van dit BASIC en de ruime mate waarmee het door diverse computerfabrikanten wordt toegepast, kan het ondertussen wel de standaard BASIC voor micro-computers worden genoemd.

Het MSX-basic bevat meer dan 150 sleutelwoord-combinaties (het eerste BASIC bevatte slechts een tiental sleutelwoorden...). Met deze sleutelwoorden kunnen commando's worden gevormd waarmee de MSX-computer wordt opgedragen om te rekenen, te tekenen, geluid te maken, tekst af te drukken enzovoorts.

MSX betekent: MicroSoft eXtended basic ofwel 'het uitgebreide basic van Microsoft'. Uit de naamgeving blijkt wel dat de taal niet nieuw is; het normale Microsoft basic is al jarenlang de meest bekende

BASIC op de wat professionelere micro-computer.

Echter, met de ontwikkeling van de kleine microcomputer, de zogenaamde huiscomputer of home-computer, groeide de noodzaak om dit BASIC aan te passen aan de moderne mogelijkheden en toe te passen op de kleine microcomputers. Uit deze noodzaak ontstond het uitgebreide Microsoft basic ofwel het MSX-basic; een BASIC waarin ervaring van vele jaren is verwerkt en waarin de meest moderne mogelijkheden zijn ondervangen.

4.3 Commando's

Zoals reeds werd opgemerkt, heeft MSX-basic ongeveer 150 sleutelwoord-combinaties waarmee commando's kunnen worden geformuleerd. Indien de formulering van deze commando's juist geschiedt, zal de computer u perfekt gehoorzamen. Indien echter een klein foutje in deze formulering zit, zal de computer u niet begrijpen en reageren met een foutmelding.

Om alvast een beetje de smaak te pakken te krijgen van het programmeren in MSX-basic worden hieronder enkele eenvoudige sleutelwoorden behandeld. Aan het einde van deze paragraaf komen we tot de conclusie dat we het eerste echte computerprogramma in MSX-basic hebben geschreven.

Zet de MSX-computer eerst uit en weer aan en tik dan eens de volgende regel in:

```
PRINT "HALLO ALLEMAAL"
```

Nadat de RETURN-toets is ingedrukt, zal de computer netjes de tekst "HALLO ALLEMAAL" afdrukken om vervolgens met de Ok-melding aan te geven dat hij klaar is.

We droegen de computer (PRINT betekent AFDRUKKEN) op om een tekst af te drukken; de computer gehoorzaamde ons feilloos.

Probeer ook de volgende regel eens:

```
PRINS "HALLO ALLEMAAL"
```

Iedere BASIC-programmeur begrijpt wat er bedoeld wordt. Echter, de

computer, tenslotte maar een dom instrument, begrijpt niet dat het sleutelwoord eigenlijk PRINT had moeten zijn en geeft een syntax error (=fout in schrijfwijze).

Hieruit zal één ding duidelijk zijn:

DE COMPUTER IS EEN DOM INSTRUMENT EN ALS ZODANIG NIET IN STAAT OM FOUTEN TE MAKEN. DE COMPUTER GEHOORZAAMT U FEILLOOS WANNEER U IN STAAT BENT OM DE OPDRACHTEN PRECIES VOLGENS DE VOORSCHRIFTEN IN ZIJN TAAL TE FORMULEREN.

We zien dat we met de regel PRINT "HALLO ALLEMAAL" een eerder gestelde regel in verband met de editor hebben overtreden; we hebben voor deze regel namelijk geen regelnummer geplaatst. Echter, geen foutmelding volgde; de computer begreep onze bedoelingen en voerde het commando feilloos uit.

Wanneer we de regel PRINT "HALLO ALLEMAAL" wèl voorzien van een regelnummer, kunnen we deze regel later met het reeds bekende LIST-commando weer zichtbaar maken. Echter, na het intikken van deze regel voert de computer het commando niet meer uit.

Wanneer we het hele kleine programma dat we nu in feite hebben geschreven, willen laten uitvoeren door de computer, dan moeten we het commando RUN gebruiken. Na dit commando zal de tekst "HALLO ALLEMAAL" weer keurig worden afgedrukt. We kunnen dit enige malen herhalen door steeds weer het RUN-commando te gebruiken. Merk op dat het veel gebruikte RUN-commando standaard onder de vijfde functietoets aanwezig is en het LIST-commando onder de vierde toets kan worden gevonden.

We kunnen het nu ontstane programma uitbreiden met nog een regel:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ER MEE"
RUN
HALLO ALLEMAAL
HOE GAAT HET ERMEE
Ok
```

We zien dat na het RUN-commando de twee opgedragen teksten keurig

onder elkaar worden afgedrukt. Blijkbaar werkt de computer de ingetoetste programmaregels op volgorde van regelnummer af totdat alle regels 'op' zijn. We kunnen het programma op deze wijze willekeurig groter maken en steeds weer laten uitvoeren. Een leuk grapje kan men uithalen door het programma als volgt uit te breiden:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ER MEE"
30 GOTO 10
RUN
```

Na het commando RUN blijft de opgedragen tekst achter elkaar worden afgedrukt op het beeldscherm. Het commando GOTO 10 zorgt ervoor, dat wanneer de computer bij regel 30 is aangekomen, hij weer 'teruggaat' naar regel 10 waardoor het programma geen einde kan vinden. Nogmaals blijkt dat de computer een dom instrument is dat de moeilijkste maar ook de domste en meest onzinnige opdrachten zonder protest uitvoert. De enige voorwaarde is, dat de juiste formulering is gebruikt.

Het steeds maar ronddraaiende programma kan door de STOP-toets tijdelijk worden onderbroken; links onder in het beeld verschijnt dan de cursor. Door de STOP-toets een tweede maal aan te raken, gaat het programma weer verder. Definitief kan het programma worden onderbroken door de toetsen CTRL en STOP tegelijkertijd in te drukken. Met een LIST kunnen we dan het programma weer te voorschijn halen.

Vervang nu regel 10 eens door:

```
10        PRINT"HALLO ALLEMAAL"
```
(denk aan de spaties)

en probeer het programma nog eens.
Conclusie: het MSX-basic is niet erg gevoelig voor spaties; het programma werkt nog precies hetzelfde. We hadden dus net zo goed:

```
10PRINT"HALLO ALLEMAAL"
```
(helemaal geen spaties)

kunnen intikken.
Ook wanneer we:

```
10print "HALLO ALLEMAAL"
```
(kleine letters)

intikken, zien we dat het programma nog steeds uitstekend werkt. We kunnen concluderen dat het gebruik van kleine of grote letters bij sleutelwoorden niets uitmaakt. Bij een LIST zal zelfs blijken dat de computer het woordje print zelf in hoofdletters heeft veranderd.

Het zal duidelijk zijn dat het programma wel anders gaat werken wanneer we de tekst tussen de aanhalingstekens in kleine letters gaan zetten.

We kunnen vaststellen dat:

- sleutelwoorden met kleine en grote letters mogen worden geschreven.
- sleutelwoorden aan elkaar dienen te worden geschreven.
- er tussen aanhalingstekens precies de tekst dient te worden opgenomen die ook moet worden gebruikt.
- er verder naar believen spatietekens mogen worden gebruikt.

In deze paragraaf kwamen we slechts enkele van de vele sleutelwoorden tegen. In de volgende hoofdstukken worden eerst wat broodnodige begrippen behandeld waarna we in de hoofstukken 9 en 10 worden geconfronteerd met alle bestaande sleutelwoorden met hun preciese schrijfwijzen en betekenissen. Alhoewel de principes van het programmeren vrij eenvoudig zijn, zal het door het grote aantal sleutelwoorden toch een behoorlijke oefening vergen voordat u het MSX-basic volledig beheerst.

Om het MSX-basic goed te kunnen beheersen, is begrip nodig van enkele algemene termen. In dit hoofdstuk wordt de term CONSTANTE behandeld.

5.1 Wat is een constante

De definitie van een constante is vrij eenvoudig:

EEN CONSTANTE IS EEN WAARDE DIE NIET
AAN VERANDERING ONDERHEVIG IS.

Wanneer we het over waarden hebben, hebben we het niet alleen over numerieke waarden (waarden die een hoeveelheid aangeven) maar ook over alfanumerieke waarden (tekstwaarden, teksten of STRINGS).

In het volgende voorbeeld zijn enkele constanten genoemd:

123	22.5	JANSSEN
0.124	−100000	DIT IS EEN TEKST
BOEKJE	ABCDEFGHIJ	10203.4

Merk op dat we in de computerwereld in plaats van de decimale komma altijd een punt gebruiken.

In het vorige hoofdstuk zagen we reeds, dat we alfanumerieke constanten door de computer kunnen laten afdrukken. In het reeds eerder genoemde programma:

```
10 PRINT "HALLO ALLEMAAL"
20 PRINT "HOE GAAT HET ER MEE"
30 GOTO 10
RUN
```

worden op programmaregel 10 en 20 de alfanumerieke constanten HALLO ALLEMAAL en HOE GAAT HET ERMEE afgedrukt op het beeldscherm. Alfanumerieke constanten dienen altijd tussen aanhalingstekens (") te worden vermeld in een MSX-basicprogramma.

Numerieke constanten behoeven *niet* tussen aanhalingstekens te worden

vermeld. Indien numerieke constanten toch tussen aanhalingstekens worden geplaatst, noemen we deze constanten niet meer numeriek maar ALFANUMERIEK. Het volgende programma:

```
NEW (om eerste de oude regels te verwijderen)
Ok
10 PRINT 125
20 PRINT "125"
```

zal, wanneer het door middel van het commando RUN in werking wordt gezet, twee maal het getal 125 op het beeldscherm afdrukken. Toch noemen we de constante op regel 10 een numerieke constante en de constante op regel 20 een alfanumerieke constante.

5.2 Rekenen met constanten

We kunnen de computer gebruiken als rekenmachine door hem bijvoorbeeld constanten bij elkaar te laten optellen. Tik bijvoorbeeld eens in:

```
PRINT 123+200
```

De computer zal na het intoetsen van de RETURN-toets onmiddellijk met het antwoord, 323 reageren. Ook moeilijkere sommen maakt de computer zonder problemen. Bijvoorbeeld:

```
NEW (om eerst de oude regels te verwijderen)
Ok
10 PRINT 12.445+123.456
20 PRINT 12/6.5
30 PRINT 123*456
40 PRINT 1000-1
```

Dit voorbeeldprogramma zal na het RUN-commando in een oogwenk de uitkomsten van de opgegeven sommen op het beeldscherm laten zien. Merk op dat het sterretje(*) als vermenigvuldigingsteken en de schuine streep(/) als deelteken worden gebruikt.

Het rekenen gaat natuurlijk alleen met NUMERIEKE constanten. Wanneer we

```
PRINT "JAN"-"PIET"
```

intoetsen, dan zal de computer antwoorden met een foutmelding; de

berekening is niet uit te voeren.

Indien we echter de regel:

```
PRINT "HALLO ALLEMAAL "+"HOE GAAT HET ER MEE"
```

intoetsen, dan zal de computer de uitkomst HALLO ALLEMAAL HOE GAAT HET ERMEE afdrukken. Blijkbaar kan de computer twee alfanumerieke constanten wel *optellen.* Conclusie:

DE COMPUTER KAN REKENEN MET NUMERIEKE CONSTANTEN. ALFANUMERIEKE CONSTANTEN KUNNEN SLECHTS WORDEN OPGETELD; HET RESULTAAT IS DAN DE AANEENSCHAKELING VAN DEZE CONSTANTEN.

Aan de ingewikkeldheid van een opgedragen som zijn hoegenaamd geen grenzen. De opdracht:

```
PRINT 12*(2/3.5)-11*(12345+1.22)/13
```

zal door de computer in een fractie van een seconde worden uitgevoerd. Ook de opdracht:

```
PRINT "ABCDEFGH"+"IJKLMNOPQ"+"RSTUVWXYZ"
```

wordt onmiddellijk uitgevoerd, waarbij de uitkomst wordt gevormd door het alfabet.

5.3 Soorten van constanten

Het MSX-basic kent diverse soorten van constanten. Sommige soorten constanten bieden geen enkele moeilijkheid, andere zijn zeer lastig te begrijpen.

Een eerste onderverdeling van constanten zagen we reeds: de onderverdeling tussen NUMERIEKE en ALFANUMERIEKE constanten.

Van alfanumerieke constanten, die we vaak STRING-CONSTANTEN (string betekent keten) noemen, is er maar één soort. Van numerieke constanten zijn er echter meerdere soorten die we per stuk in de volgende paragrafen zullen behandelen.

5.4 Integere constanten

De integere constante is een eenvoudige vorm van numerieke constanten. Een integere constante dient te voldoen aan de volgende eisen:

- de integere constante mag niet kleiner zijn dan –32768
- de integere constante mag niet groter zijn dan 32767
- de integere constante mag geen decimalen en ook geen decimale punt bevatten

Om duidelijk te laten uitkomen dat een bepaalde constante een integere constante is, moet achter deze constante een procent-teken worden gezet. Enkele voorbeelden:

goede voorbeelden van integere constanten	foute voorbeelden integere constanten
32767%	32768
–32768%	–32769
12%	12.%
123%	–11.33

Een speciale integere constante wordt gevormd door het regelnummer dat voor een programmaregel dient te worden geplaatst. Bij de behandeling van de sleutelwoorden zal deze integere constante steeds weer als regelnummer worden aangeduid. Een regelnummer mag niet kleiner zijn dan 0 en niet groter dan 65529. Verder gelden de normale regels voor een integere constante ook voor een regelnummer.

5.5 Constanten met enkelvoudige precisie

De constante met enkelvoudige precisie heeft een veel grotere vrijheid in het gebruik. De volgende eisen worden aan een constante met enkelvoudige precisie gesteld:

- de minimum waarde voor een constante met enkelvoudige precisie is $-9.99... \times 10^{62}$
- de maximum waarde voor een constante met enkelvoudige precisie is $9.99... \times 10^{62}$
- de grootst mogelijke waarde kleiner dan 0 voor een constante met enkelvoudige precisie is -10^{-64}

- de kleinst mogelijke waarde groter dan 0 voor een constante met enkelvoudige precisie is 10^{-64}
- een constante met enkelvoudige precisie heeft een precisie (significantie) van 6 cijfers

Om het voorgaande te begrijpen dienen, voor sommige lezers ten overvloede, enkele zaken te worden uitgelegd.

De notatie 10^5 staat voor het getal 10 maal 10 maal 10 maal 10 maal 10 = 100000.

In deze notatie geven we in het grondtal (het eerste getal) aan welk getal dient te worden vermenigvuldigd. In de macht (het hoger geplaatste getal) geeft men aan, hoe vaak de vermenigvuldiging dient plaats te vinden. Dus

10^5

- heeft een grondtal, gelijk aan 10
- heeft een macht, gelijk aan 5
- wordt uitgesproken als *'tien tot de vijfde (macht)'*
- is het zelfde als 100000 (een 1 met *vijf* nullen)

Het zal duidelijk zijn dat het getal 10^{63} (een 1 met drieënzestig nullen erachter!) een bijzonder groot getal is.

Indien de macht gelijk is aan 1 of 0 of kleiner is dan 0, dan wordt het bovenstaande problematisch. Immers, een herhaalde vermenigvuldiging kan dan niet meer worden uitgevoerd. We spreken af:

- een macht gelijk aan 1 mag worden weggelaten. $10^1=10$
- een macht gelijk aan 0 heeft tot gevolg dat de waarde van de betreffende constante altijd gelijk is aan 1.

  Dus $10^0=1$ maar ook 2^0 en 12345^0 zijn gelijk aan 1.

- een macht kleiner dan 0 heeft tot gevolg dat de waarde van de betreffende constante gelijk is aan

$$\frac{1}{\text{deze constante, maar dan met positieve macht}}$$

10^{-20} is dus gelijk aan $1/10^{20} = 1/100000000000000000000 = 0.00000000000000000001$

Het zal duidelijk zijn dat het getal 10^{-64} een bijzonder klein getal is.

Bij wetenschappelijke notaties is het vaak gebruikelijk om de zogenaamde exponentiële notatie toe te passen. Indien een chemicus het over 1000 gram natriumchloride heeft, dan zal hij noteren:

1.0×10^3 g Natriumchloride.

De notatie 1.0×10^3 bevat twee elementen. Het eerste element (1.0) noemen we de mantisse. Het tweede element (10^3) noemen we de exponent.

Indien we de notatie uitwerken, blijkt het volgende:

$1.0 \times 10^3 = 1.0 \times 10 \times 10 \times 10 = 1000$

De chemicus die deze notatie op papier zette, bedoelde net als wij dus gewoon 1000 gram natriumchloride. Echter, hij noteerde meer dan alleen dat:

- met de notatie 1.0 geeft hij een betrouwbaarheid, precisie of significantie aan van de hoeveelheid. Met 1.0 wordt aangegeven dat slechts de eerste twee cijfers van deze notatie betrouwbaar zijn. Het zou net zo goed om 980 gram of om 1032 gram kunnen gaan, maar het is zeker geen 250 of 1080 gram.

- met de notatie 10^3 geeft hij de grootte-orde aan. Het gaat in dit geval om een aantal duizenden (in dit geval één maal duizend) grammen.

Omdat deze exponentiële notatie vooral bij grotere of kleinere getallen het voordeel biedt dat het een korte notatie is (probeer het getal 1.2×10^{60} maar eens gewoon uit te schrijven...) heeft Microsoft deze notatie ook in het MSX-basic mogelijk gemaakt. De notatievorm moest echter natuurlijk enigszins aan de mogelijkheden van de computer worden aangepast.

De constante 1.0×10^3 wordt in MSX-basic genoteerd als 1.0E3 of 1.0D3. Zo wordt 1.243×10^{16} bijvoorbeeld genoteerd als 1.243E16 of 1.1234D16 en wordt 3.14×10^{-33} genoteerd als 3.14E–33 of 3.14D–33.

Achter de mantisse wordt in de MSX-notatie (zoals in veel andere

BASIC-talen) eerst een D of een E geplaatst waarna de macht van de exponent wordt opgenomen. Het grondtal van de exponent is bij deze notatie altijd gelijk aan 10 en wordt weggelaten.

Voor de duidelijkheid is het raadzaam om met deze notatie eens wat te experimenteren op de computer. Probeer bijvoorbeeld eens een programma in de volgende vorm te schrijven en kijk eens wat er na een RUN gebeurt. Reken de uitkomsten na.

```
NEW
Ok
10 PRINT 1.2E-3
20 PRINT 5.22E-33*1E44/2E0
30 PRINT 123D22/23D21-10
... etcetera
```

Lees nu nog eens de bepalingen voor een constante met enkelvoudige precisie na. Met de hier behandelde stof zullen deze bepalingen nu duidelijk zijn.

Om nu aan te geven, dat een constante een constante met enkelvoudige precisie is,

- moet achter de constante een uitroepteken (!) worden geplaatst of:
- moet het exponentgedeelte, indien aanwezig, de exponentaanduiding E bevatten. De exponentaanduiding D is voor deze constanten verboden.

Het zal duidelijk zijn dat een constante met enkelvoudige precisie niet verplicht in de exponentiële notatie behoeft te zijn genoteerd. Enkele voorbeelden:

goede voorbeelden van constanten met enkelvoudige precisie	foute voorbeelden van constanten met enkelvoudige precisie
1234.5!	12%
1234!	1.234567D33
1.1E28	–12D–33
–32769!	1111111

5.6 Constanten met dubbele precisie

Het enige verschil tussen constanten met dubbele precisie en constanten met enkelvoudige precisie is, dat een constante met dubbele precisie geen precisie van 6 cijfers maar een precisie van 14 cijfers heeft.

Om aan te geven dat een constante een dubbele precisie heeft,
- moet achter de constante een hekje (#) worden geplaatst, of:
- moet het exponentgedeelte, indien aanwezig, de exponentaanduiding D bevatten. De exponentaanduiding E is voor deze constante *verboden*.

Enkele voorbeelden:

goede voorbeelden van constanten met dubbele precisie	foute voorbeelden van constanten met dubbele precisie
1234.5#	1234.5!
12D22	12E22
123#	123%
12D–60	1!

5.7 Bij twijfel...

Het is niet verplicht om bij een constante een achtervoegsel (#,% of !) te gebruiken. Bij sommige constanten is het dan echter niet precies uit te maken of het nu een integere constante, een constante met enkelvoudige precisie of een constante met dubbele precisie is. Bij de exponentiële notatie bestaat de twijfel nooit; uit de in de exponent genoteerde letter (D of E) is zonder meer het type constante af te leiden.

Indien het MSX-basic het verschil niet precies kan bepalen, dan maakt het om misverstanden te voorkomen, altijd een zo klein mogelijk, passende constante van de constante waarover wordt getwijfeld.

MSX-basic werkt intern met een nog grotere verscheidenheid aan constanten die echter voor de programmeur zelden van belang zijn.

5.8 Constanten in andere talstelsels

In MSX-basic is het ook mogelijk om constanten te formuleren in andere talstelsels. We onderscheiden binaire (tweetallige), octale (achttallige) en hexadecimale (zestientallige) constanten.

Om met deze constanten te werken, is kennis van de diverse talstelsel noodzakelijk. Het werken met deze vormen van constanten is echter voor de meeste programmeurs niet interessant. Slechts in een enkel geval is het plezierig om met binaire, hexadecimale of octale constanten te kunnen werken.

Het is zeker voor de beginnende programmeur maar ook voor de wat gevorderde programmeur die hier geen noodzaak toe ziet, verstandig om de rest van dit hoofdstuk over te slaan.

5.9 Het principe van talstelsels

Het talstelsel waarin wij normaal werken, is het decimale talstelsel. Het decimale talstelsel of tientallige talstelsel is zo genoemd omdat dit talstelsel tien cijfertekens kent (0,1,2...9). Wanneer we in het decimale talstelsel gaan tellen vanaf 0, dan komen we tot de conclusie dat we het tiende getal in twee cijfers moeten gaan noteren. Het honderdste getal dient in drie cijfers te worden genoteerd etcetera. Een decimaal getal kan als volgt worden ontleed:

Bijvoorbeeld het getal 4264:

4	2	6	4
aantal maal 10^3	aantal maal 10^2	aantal maal 10^1	aantal maal 10^0

Het getal 4264 is op deze wijze gelijk aan 4 x 10 x 10 x 10 = 4000
2 x 10 x 10 = 200
6 x 10 = 60
4 = 4+

4264

Het tweetallige stelsel kent geen tien maar slechts twee cijfertekens. Deze cijfertekens zijn 0 en 1. Een tweetallig of binair getal laat zich als volgt ontleden:

Bijvoorbeeld het binaire getal 11010:

1	1	0	1	0
aantal maal 2^4	aantal maal 2^3	aantal maal 2^2	aantal maal 2^1	aantal maal 2^0

Het binaire getal 11010 is dus gelijk aan

1 x 2 x 2 x 2 x 2	=	16
1 x 2 x 2 x 2	=	8
0 x 2 x 2	=	0
1 x 2	=	2
0	=	0+
		26

Merk op dat in het binaire stelsel *elke* waarde kan worden genoteerd. Een nadeel is, dat voor de notatie van een relatief klein getal al erg veel posities nodig zijn. Bijvoorbeeld het getal 511 moet binair al als 111111111 worden geschreven; met 9 cijfers maar liefst!

Het tweetallige of binaire stelsel is het stelsel waarin de computer zelf rekent. Wanneer een computergeheugen tot op het laatste element wordt ontleed, dan zal blijken dat de werkelijke opslag van gegevens in dit geheugen tweetallig geschiedt. De geheugenelementen waarin alleen het getal 0 of 1 kan worden opgeslagen, noemt men BITS (afkorting van binary digits=tweetallige cijfers). Om deze reden is het vaak voordelig om in ieder geval het binaire stelsel te beheersen.

Het achttallige of octale stelsel kent voor de notatie van waarden acht cijfertekens, namelijk de cijfers 0,1,2,3,4,5,6 en 7. Een octaal getal laat zich als volgt ontleden:

Bijvoorbeeld het octale getal 7761:

7	7	6	1
aantal maal 8^3	aantal maal 8^2	aantal maal 8^1	aantal maal 8^0

Het octale getal 7761 is dus gelijk aan 7 x 8 x 8 x 8 = 3584
7 x 8 x 8 = 448
6 x 8 = 48
1 = 1+
4081

Merk op dat ook in het octale stelsel *elke* waarde kan worden genoteerd.

Het zestientallige of hexadecimale stelsel kent voor de notatie van waarden maar liefst 16 cijfertekens. Omdat in ons normale cijfertekenstelsel slechts tien cijfers voorkomen, moeten er voor notatie van hexadecimale waarden dus zes cijfertekens bij gemaakt worden. Men koos voor de zes tekortschietende cijfers als symbolen de letters A tot en met F. Dus:

het hexadecimale cijfer	heeft decimaal gezien de waarde
0	0
1	1
.	.
.	.
.	.
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Een hexadecimaal getal laat zich als volgt ontleden:

Bijvoorbeeld het getal FE81:

F	E	8	1
aantal maal 16^3	aantal maal 16^2	aantal maal 16^1	aantal maal 16^0

Het hexadecimale getal FE81 is dus gelijk aan:

```
15(F) x 16 x 16  x 16 =  61440
14(E) x 16 x 16       =   3584
 8    x 16            =    128
       1               =     1+
                        -------
                        65153
```

Ook in het hexadecimale stelsel kan *elke* waarde worden genoteerd.

Het binaire, het hexadecimale en het octale stelsel zijn de drie belangrijkste stelsels in de computerwereld buiten natuurlijk het decimale stelsel. Alhoewel de theorie van de verschillende talstelsels niet zo erg moeilijk is, is het werken en rekenen met deze talstels vaak een groot probleem.

5.10 Binaire constanten

In het MSX-basic kan een binaire constante worden geformuleerd. De notatie dient dan te worden voorafgegaan door het teken & en de letter B. Probeer bijvoorbeeld eens het volgende programma:

```
NEW
Ok
10 PRINT &B10110
20 PRINT &B11110
```

Wanneer dit programma door een RUN wordt geactiveerd, zal het programma keurig worden uitgevoerd; de (decimale) waarden van de betreffende binaire constanten worden keurig afgedrukt.

Een binaire constante dient aan de volgende voorwaarden te voldoen:
- de constante dient door &B te worden voorafgegaan.
- de constante mag alleen de cijfers 0 en 1 bevatten.
- de constante mag maximaal gelijk aan &B1111111111111111

zijn (65535 decimaal).

Pas op, indien de decimale waarde van de binaire constante groter is dan 32767, dan trekt MSX-basic automatisch de waarde 65536 af van de waarde voordat deze wordt afgedrukt. Hierdoor blijft de uitkomst altijd tussen de waarden –32768 en 32767 waardoor de uitkomst altijd een integere waarde is. De meer doorgewinterde programmeur zal hierin het tweecomplementsysteem herkennen.

5.11 Octale constanten

In het MSX-basic kan een octale constante worden geformuleerd. De notatie dient dan te worden voorafgegaan door het teken & en de letter O. Probeer bijvoorbeeld eens het volgende programma:

```
NEW
Ok
10 PRINT &O177
20 PRINT &O1752
```

(Pas op voor het verschil tussen de letter O en het cijfer Ø)

Wanneer dit programma wordt uitgevoerd, worden keurig de decimale waarden van de vermelde octale constanten afgedrukt.

Een octale constante dient aan de volgende voorwaarden te voldoen:

- de constante dient door &O te worden voorafgegaan.
- de constante mag alleen de cijfertekens 0...7 bevatten.
- de constante mag maximaal gelijk zijn aan &O177777(65535 decimaal).

Pas op, indien de decimale waarde van de binaire constante groter is dan 32767, dan trekt MSX-basic automatisch de waarde 65536 af van de waarde voordat deze wordt afgedrukt. Hierdoor blijft de uitkomst altijd tussen de waarden –32768 en 32767 waardoor de uitkomst altijd een integere waarde is. De meer doorgewinterde programmeur zal hierin het tweecomplementsysteem herkennen.

5.12 Hexadecimale constanten

Ook hexadecimale constanten kunnen binnen het MSX-basic worden

gebruikt. De notatie dient dan te worden voorafgegaan door het teken & en de letter H. Probeer bijvoorbeeld eens het volgende programma:

```
NEW
Ok
10 PRINT &HFFFF
20 PRINT &H1023
```

Wanneer dit programma wordt uitgevoerd, worden keurig de decimale waarden voor de genoteerde hexadecimale constanten afgedrukt op beeldscherm.

Een hexadecimale constante dient aan de volgende eisen te voldoen:

- de constante dient door &H te worden voorafgegaan.
- de constante mag alleen de cijfertekens 0...9 en A...F bevatten.
- de constante mag maximaal gelijk zijn aan &HFFFF (65535 decimaal).

Pas op, indien de decimale waarde van de binaire constante groter is dan 32767, dan trekt MSX-basic automatisch de waarde 65536 af van de waarde voordat deze wordt afgedrukt. Hierdoor blijft de uitkomst altijd tussen de waarden –32768 en 32767 waardoor de uitkomst altijd een integere waarde is. De meer doorgewinterde programmeur zal hierin het tweecomplementsysteem herkennen.

6.1 Wat is een variabele

Ook de definitie van een variabele is vrij eenvoudig:

EEN VARIABELE IS EEN WAARDE DIE
(IN TEGENSTELLING TOT EEN CONSTANTE) WEL
AAN VERANDERINGEN ONDERHEVIG IS.

Een variabele kan dus niet met één vaste aanduiding worden genoteerd. In plaats daarvan dienen we variabelen een naam te geven. Wanneer we bijvoorbeeld intikken:

```
WAARDE=12.5
```

dan hebben we op dat moment aan de variabele WAARDE de constante 12.5 toegekend. Wanneer we later dan intoetsen:

```
PRINT WAARDE
```

dan zal de computer netjes de twee eerder aan de variabelen toegekende waarde weer projecteren.

Wanneer we vervolgens intikken:

```
HOEVEELHEID=26
```

dan is op dat moment de waarde 26 aan variabele HOEVEELHEID toegekend. Tikken we nu in:

```
PRINT WAARDE+HOEVEELHEID
```

Dan zal de computer netjes de twee eerder aan de variabelen toegekende waarden bij elkaar optellen en het resultaat op het beeldscherm laten zien.

We kunnen de waarde van een variabele ook herzien. Indien we intikken:

```
HOEVEELHEID=622.5
```

En daarna:

```
PRINT HOEVEELHEID+WAARDE
```

dan zien we, dat de variabele HOEVEELHEID een andere waarde heeft verkregen. De variabele HOEVEELHEID maar natuurlijk ook alle andere variabelen zijn dus onderhevig aan veranderingen.

Bovenstaande voorbeelden behandelen slechts één van de typen variabelen, namelijk de numerieke variabelen. We kennen echter ook een heel ander soort variabelen, namelijk de alfanumerieke variabelen.

Laten we bijvoorbeeld eens intikken:

```
LET NAAM$="GEERT-JAN"
```

Het woordje LET betekent: laat of laat zijn. LET NAAM$= "GEERT-JAN" betekent dus zoveel als laat variabele NAAM$ gelijk zijn aan GEERT-JAN. Het sleutelwoord LET zouden we ook in onze eerdere voorbeelden kunnen hebben gebruikt maar het mag in deze constructie zonder meer worden weggelaten. We hadden dus ook gewoon:

```
NAAM$="GEERT-JAN"
```

kunnen intikken.

De variabele NAAM$ is een alfanumerieke variabele. Deze heeft dan ook verplicht een dollarteken ($) als laatste teken. Alfanumerieke variabelen kunnen allerlei karakters bevatten. Net als bij alfanumerieke constanten kunnen we alfanumerieke variabelen alleen maar optellen. Probeer bijvoorbeeld eens in te tikken:

```
PRINT NAAM$
```

en

```
PRINT NAAM$+" IS MIJN NAAM."
```

In het laatste voorbeeld werden een alfanumerieke variabele en een alfanumerieke constante bij elkaar opgeteld. Het resultaat bestond uit een aaneenschakeling van deze variabele en constante.

Om het begrip variabele nog wat verder uit te diepen, hebben we de

medewerking van nog een MSX-sleutelwoord nodig, namelijk het sleutelwoord INPUT. Met dit sleutelwoord is het mogelijk om een variabele een waarde te geven door intoetsing terwijl het programma werkt. Tik bijvoorbeeld het volgende programma eens in:

```
NEW   (om oude programmaregels te verwijderen)
Ok
10 PRINT "REKENPROGRAMMA"
20 INPUT "HOE HEET U ";NAAM$
30 PRINT "GOEDENDAG "+NAAM$+" HOE IS HET ERMEE ?"
40 INPUT "GEEF EEN GETAL IN ";WAARDE
50 PRINT WAARDE+WAARDE;" IS HET DUBBELE VAN ";WAARDE
60 GOTO 40
```

Zo heuben we met relatief weinig sleutelwoorden al een heel programma geschreven. In regel 10 wordt op het beeldscherm vermeld dat het hier om een rekenprogramma gaat. Niets bijzonders, slechts een eenvoudige alfanumerieke constante wordt op het beeldscherm afgedrukt.
In regel 20 komt het nieuwe sleutelwoord, INPUT, aan de beurt.
Wanneer we het programma met een RUN in werking stellen, zien we dat de tekst HOE HEET U netjes op het beeldscherm wordt afgedrukt en dat de computer daarna wacht. De computer verwacht nu namelijk van u, dat u een alfanumerieke constante intoetst en dat u, zoals altijd, deze ingave met een RETURN-toets afsluit. Uw ingave wordt in variabele NAAM$ bewaard.
In regel 30 presenteert de computer de optelling van een alfanumerieke constante, een alfanumerieke variabele en weer een alfanumerieke constante. Hierdoor begroet de computer ons hoffelijk en noemt hij ons zelfs bij de naam; een bewijs dat in NAAM$ inderdaad de eerder gepleegde ingave is geplaatst.
In regel 40 vraagt de computer weer om een ingave. Dit maal dient de ingave een numerieke constante te zijn; dat is te zien aan de naam van de variabele in het INPUT-commando. Deze naam eindigt niet op een dollarteken en dit duidt op een numerieke variabele. Indien we tijdens deze INPUT proberen, een niet geldige numerieke constante in te geven, dan geeft de computer een foutmelding en krijgen we nogmaals de kans om een correcte numerieke constante in te geven.
In regel 50 wordt de optelling van de variabele WAARDE bij zichzelf afgedrukt. Op het beeldscherm verschijnt het dubbele van de ingegeven waarde. Na deze waarde wordt de alfanumerieke constante IS HET DUBBELE VAN afgedrukt, gevolgd door de waarde van de inhoud van numerieke variabele WAARDE. Uit deze programmaregel leren we meteen dat we in een PRINT-commando verschillende zaken achter

elkaar aan kunnen afdrukken door deze met een punt-komma van elkaar te scheiden.
Tenslotte gaven we de computer op regel 60 het commando om weer naar programmaregel 40 terug te gaan; de computer vraagt weer om een ingave. Wanneer we de computer niet met CTRL-STOP onderbreken, zal hij voortdurend met dit programma bezig blijven.

We zien dat het gebruik van variabelen een nieuwe dimensie geeft aan de computer. Door het gebruik van variabelen kunnen we eenzelfde programma steeds van andere waarden voorzien waardoor steeds nieuwe situaties worden berekend.

6.2 Soorten van variabelen

Net als bij constanten onderscheiden we ook bij variabelen enkele verschillende typen. De eerste onderverdeling, die tussen numerieke en alfanumerieke variabelen, zagen we reeds. Van de stringvariabelen of alfanumerieke variabelen is er slechts één type. Echter, van de numerieke variabelen onderscheiden we verschillende typen die in de volgende paragrafen worden behandeld.

6.3 De integere variabele

Een integere variabele herkennen we aan het procentteken (%) dat achter de variabelenaam is vermeld. Een integere variabele kan alleen een integere waarde bevatten (zie de beschrijving van integere constanten). Wanneer we proberen om een andere dan integere waarde aan een integere variabele toe te kennen, dan wordt de waarde aangepast of volgt een foutmelding. Bij:

```
LET WAARDE%=12.5
```

zal geen foutmeling ontstaan. Echter, wanneer we via een PRINT-opdracht de variabele weer op het beeldscherm laten verschijnen, dan zal blijken dat de variabele de waarde 12 heeft aangenomen; de decimalen werden verwaarloosd en de variabele bleef als zodanig integer. Wanneer we echter intoetsen:

```
LET WAARDE%=100000
```

dan volgt een foutmelding. De maximale integere waarde is 32767;

100000 kan als waarde niet in WAARDE% worden opgenomen.

6.4 De variabele met enkelvoudige precisie

Een variabele met enkelvoudige precisie kunnen we herkennen aan het uitroepteken (!) dat achter de variabelenaam is vermeld. Een variabele met enkelvoudige precisie is aan dezelfde beperkingen gebonden als een constante met enkelvoudige precisie tot zover van toepassing. Wanneer we proberen om een waarde, afwijkend van deze bepalingen, aan een variabele met enkelvoudige precisie toe te kennen dan wordt de waarde aangepast of krijgen we een foutmelding. Wanneer we bijvoorbeeld intoetsen:

```
WAARDE!=12345678
```

dan volgt geen foutmelding. Tikken we echter vervolgens in:

```
PRINT WAARDE!
```

dan presenteert de computer de waarde 12345700. Het zevende en achtste cijfer vallen buiten de precisie van een variabele met enkelvoudige precisie. Wel rondde de computer het zesde cijfer zo af dat de waarde in WAARDE! toch zo dicht mogelijk bij de bedoelde waarde ligt.

Wanneer we intoetsen:

```
WAARDE!=1E99
```

dan geeft de computer een foutmelding; de waarde die we aan WAARDE! wilden toekennen, ligt niet binnen de voor een variabele met enkelvoudige precisie bepaalde grenzen.

6.5 De variabele met dubbele precisie

Een variabele met dubbele precisie kunnen we herkennen aan het hekje (#) dat achter de variabelenaam is geplaatst. Een variabele met dubbele precisie is aan dezelfde beperkingen gebonden als de constante met dubbele precisie tot zover van toepassing. Wanneer we proberen om een waarde, afwijkend van deze bepalingen, aan een variabele met dubbele precisie toe te kennen dan wordt de waarde aangepast of

krijgen we een foutmelding. Wanneer we bijvoorbeeld intikken:

```
WAARDE#=123456789012345
```

dan geeft de computer geen foutmelding. Tikken we echter vervolgens in:

```
PRINT WAARDE#
```

dan presenteert de computer de waarde 1.2345678901235E+15. Behalve dat de computer besloot om dit grote getal in exponentiële vorm te presenteren, werd op het vijftiende cijfer afgerond en worden slechts veertien cijfers afgedrukt.

Indien we intikken:

```
WAARDE#=1D99
```

dan geeft de computer een foutmelding; de waarde die we aan WAARDE # wilden toekennen, viel buiten de gestelde bepalingen.

6.6 Bij twijfel...

Indien een variabelenaam geen dollarteken, hekje, uitroepteken of procentteken als laatste karakter heeft, dan is het niet meer duidelijk om wat voor soort variabele het hier gaat. De volgende regels gelden dan:

- in het normale geval neemt de computer voor zo'n waarde aan, dat het om een variabele met dubbele precisie gaat.
- indien echter door middel van een DEFINT, DEFSTR, DEFSNG of DEFDBL sleutelwoord een ander type is verbonden aan de eerste letter van de variabelenaam, dan wordt dat type automatisch aangenomen. Zie hiervoor de behandeling van de DEFSTR, DEFINT, DEFSNG en DEFDBL sleutelwoorden in hoofdstuk 9.

6.7 Waarschuwing

Aan de lengte van een variabelenaam is hoegenaamd geen beperking. Een variabele kan zowel W als AANTALGEHAALDEPUNTEN heten. Het is echter raadzaam om korte, duidelijke variabelenamen te kiezen.

Een andere factor komt nog om de hoek kijken. En omdat deze factor al zeer vaak aanleiding heeft gegeven tot dagen zoekwerk naar een vermeende programmafout door radeloze programmeurs (en niet alleen beginnelingen...), zetten we de volgende waarschuwing in grote letters neer:

HOE LANG DE VARIABELENAAM OOK IS, DE COMPUTER 'KIJKT' ALLEEN NAAR DE EERSTE TWEE LETTERS VAN DIE VARIABELENAAM!!!

Dat betekent bijvoorbeeld dat de variabele AA en de variabele AANTAL één en dezelfde variabele is. Om dit te bewijzen, kunt u het volgende kleine programma intoetsen:

```
NEW
Ok
10 INPUT "GEEF EEN GETAL IN ";AANTAL
20 PRINT "VARIABELE AANTAL=";AANTAL
30 PRINT "EN VARIABELE AA=";AA
40 STOP
```

Dit programma bewijst onomstotelijk dat slechts de eerste twee letters belangrijk zijn. Er is daarom zelfs wel wat voor te zeggen om er een gewoonte van te maken om variabelenamen nooit groter te maken dan twee letters, eventueel gevolgd door een dollarteken, een hekje, een uitroepteken of een procentteken.

6.8 Array-variabelen

Het kan voorkomen dat we meerdere waarden onder één en dezelfde variabelenaam wilen bewaren. In dat geval zullen we behalve de variabelenaam ook dienen aan te geven welke waarde van alle waarden die we hieronder hebben opgeslagen we bedoelen. Een variabele waaronder we meer dan één waarde willen bewaren, kunnen we vergelijken met een tabel. We noemen zo'n varabele een array-variabele, tabel-variabele of kortweg een array (rij).

Zo'n array dienen we een bepaalde grootte (een bepaald aantal mogelijkheden tot opslag van waarden) toe te kennen. Dit doen we met behulp van het sleutelwoord DIM. We gaan een programma schrijven:

```
NEW
Ok
10 DIM TABEL(3)
```

Wanneer we dit programma in werking stellen, dan wordt een variabele, genaamd TABEL, toegewezen. Deze variabele biedt mogelijkheid tot opslag van *vier* waarden, in dit geval numerieke waarden met dubbele precisie. Deze *vier* waarden worden opgeslagen onder:

TABEL(0)	...eerste waarde
TABEL(1)	...tweede waarde
TABEL(2)	...derde waarde
TABEL(3)	...vierde waarde

In het verdere programma kunnen we deze tabel gaan vullen:

```
20 LET TABEL(0)=12
30 LET TABEL(1)=96
40 LET TABEL(3)=-1E22
50 LET TABEL(2)=TABEL(3)*TABEL(1)/TABEL(0)
60 PRINT TABEL(0),TABEL(1),TABEL(2),TABEL(3)
```

Wanneer we dit programma laten werken via een RUN, dan merken we na enige narekening dat de variabele TABEL inderdaad vier afzonderlijke waarden bevat die we kunnen veranderen en waarmee we kunnen rekenen.

In programmaregel 60 zien we overigens, dat bij een PRINT-commando de afzonderlijke gegevens met een komma van elkaar mogen worden gescheiden. We zagen zoiets al in paragraaf 6.1, maar daar gebruikten we een punt-komma. Het gebruik van een komma heeft tot gevolg dat de gegevens ietwat uit elkaar, zo mogelijk op dezelfde regel worden afgedrukt.

We kunnen het programma uitbreiden met het volgende gedeelte:

```
60 INPUT "WELKE TABELWAARDE ";NUMMER
70 PRINT "WAARDE:";TABEL(NUMMER)
80 GOTO 60
```

(de oude regel 60 vervalt hiermee)

Wanneer we dit programma in werking stellen, vraagt de computer ons op regel 60, welk tabelelement we willen zien. We moeten dan een waarde 0,1,2 of 3 ingeven om geen foutmelding te krijgen. Wanneer we een waarde ingeven, wordt deze in NUMMER opgeslagen. Op programmaregel 70 wordt dan vervolgens het tabelelement met het zojuist ingegeven nummer op het beeldscherm getoond. We zien dat we het bedoelde tabelelement niet alleen met een getal maar ook met een andere variabele kunnen aanduiden (de professionele programmeur spreekt niet over het aanduiden maar over het *adresseren* van een tabelelement).

Een array-variabele kan in meer dan één richting bepaald zijn. Het volgende programmavoorbeeld geeft een voorbeeld van het werken met een tweedimensionale tabel of array. Merk op dat er tevens meerdere commando's op één programmaregel zijn geplaatst. Dit is toegestaan indien deze commando's met een dubbele punt van elkaar zijn gescheiden.

De eerste programmaregel bevat een REM-regel. REM is een afkorting van REMARK (=opmerking). Met deze regel wordt slechts een stukje commentaar in het programma opgenomen dat door de computer wordt genegeerd.

```
NEW
Ok
10 REM VOORBEELD TWEE-DIMENSIONELE TABEL
20 DIM TB%(2,2)
30 TB%(0,0)=12:TB%(0,1)=345:TB%(0,2)=-22
40 TB%(1,0)=99:TB%(1,1)=871:TB%(1,2)=5
50 TB%(2,0)=TB%(0,0)+TB%(1,0):TB%(2,1)=TB%(0,1)+TB%(1,1)
60 TB%(2,2)=TB%(0,2)+TB%(1,2)
70 PRINT TB%(0,0);TB%(1,0);TB%(2,0)
80 PRINT TB%(0,1);TB%(1,1);TB%(2,1)
90 PRINT TB%(0,2);TB%(1,2);TB%(2,2)
100 PRINT "EINDE":STOP
```

Aan het einde van het programma kan men tabel TB% als volgt zien opgebouwd:

tweede dimensie \ eerste dimensie	0	1	2	
0	12	99	111	◄(12 + 99)
1	345	871	1216	◄(345 + 871)
2	-22	5	-17	◄(-22 + 5)

Alhoewel tabellen met drie, vier, vijf of meer dimensies moeilijk voorstelbaar zijn, zijn deze binnen het MSX-basic toch mogelijk. Een constructie als:

```
60 DIM TABEL(3,4,7,8)
70 LET TABEL(0,1,3,8)=23
```

werkt perfekt en kan in latere, ingewikkelde programma's een uitkomst zijn.

Tot slot van deze paragraaf merken we op, dat tabellen of arrays niet alleen met numerieke maar ook met alfanumerieke variabelen zijn toegestaan. Zo kan in een tabel die met:

```
DIM ADRES$(100,3)
```

is toegewezen, de naam, straat, woonplaats en telefoonnummer van 101 kennissen worden opgeslagen.

Wanneer een tabel-variabele niet met behulp van een DIM-kommando is toegewezen, dan krijgt deze tijdens het eerste gebruik automatisch het aantal gebruikte dimensies. Per dimensie worden 11 tabel-elementen (0...10) aangenomen. Na het volgende programma heeft variabele Q automatisch de dimensie (10,10,10) als ware een DIM Q (10,10,10) uitgevoerd.

```
new
ok
10 LET Q(0,2,7)=26
```

6.9 Opslagruimte van variabelen

De computer dient van variabelen de geldende waarden te onthouden. Dit doet de computer door deze waarden in het geheugen op te slaan. Dit computergeheugen is beperkt. Daarom is het, vooral bij grote programma's, zaak om erop te letten of waarden, opgeslagen in variabelen met dubbele precisie, niet 'passen' in variabelen met enkelvoudige precisie of misschien wel in integere variabelen.
Integere variabelen nemen tussen de verschillende typen relatief de minste geheugenruimte in. Hun mogelijkheden zijn dan ook maar beperkt. Variabelen met enkelvoudige precisie nemen wat meer geheugenruimte in en bieden daarvoor in de plaats veel minder beperkingen. Variabelen met dubbele precisie nemen het meeste ruimte in; hun mogelijkheden zijn dan ook welhaast onbeperkt.

De geheugenruimte van een computer, we zagen dat al eerder, wordt uitgedrukt in een aantal bytes. Een byte is een geheugenelement waarin, ruw gezegd, één karakter of twee cijfers van een numerieke variabele kunnen worden opgeslagen. Op de manier waarop deze gegevens worden opgeslagen, gaan we hier niet verder in.

De wat gevorderde programmeur vraagt zich op een gegeven moment af, hoeveel geheugenruimte bepaalde variabelen nu kosten. Om deze vraag te beantwoorden, volgt hieronder een schema van typen variabelen met hun benodigde ruimte.

type variabele	ruimte die in ieder geval wordt bezet	per opgeslagen teken extra rekenen	per dimensie extra tellen	per array-element extra tellen
alfan. variabele	6 bytes	1 byte		
alfan. array-variabele	6 bytes	1 byte	2 bytes	3 bytes
num. integere variabele	5 bytes			
num. variabele met enkelvoudige precisie	7 bytes			
num variabele met dubbele precisie	11 bytes			
num. integere array-variabele	6 bytes		2 bytes	2 bytes
num. array-variabele met enkelv. precisie	6 bytes		2 bytes	4 bytes
num. array-variabele met dubbele precisie	6 bytes		2 bytes	8 bytes

Het aantal dimensies van een array is het aantal getallen die in het DIM-statement voor die variabele tussen haakjes worden genoemd. Het aantal array-elementen vindt men door alle getallen die tussen de haakjes van de betreffende variabele in het DIM-statement staan, elk met 1 te verhogen en deze vervolgens met elkaar te vermenigvuldigen. Een DIMA$ (2,3,10) geeft bijvoorbeeld 3x4x11=132 array-elementen. Het aantal dimensies van A$ is in dat geval 3.

7.1 Wat is een uitdrukking

We kunnen in MSX-basic de computer bijvoorbeeld de volgende som laten oplossen:

```
PRINT 2*(WAARDE+5.5)/(2+AANTAL)
```

De variabelen WAARDE en AANTAL dienen dan wel te zijn gevuld met bepaalde waarden.

De opgave (de som) die na het PRINT-commando staat, noemt men in de computerwereld een UITDRUKKING. Vormen van uitdrukkingen zijn bijvoorbeeld ook:

```
2+3              A/B                "JAN"+"KAREL"
25/(3-4*Q)       "HALLO"+NAAM$      12.55+3%
```

Deze uitdrukkingen, numeriek of alfanumeriek, geven aan:

- welke bewerkingen dienen te geschieden (optellen, aftrekken, vermenigvuldigen etcetera)
- op welke variabelen deze bewerkingen dienen te worden toegepast
- op welke constanten deze bewerkingen dienen te worden toegepast
- in welke volgorde de bewerkingen dienen te worden toegepast.

In feite kunnen we een uitdrukking beschouwen als een som die wij de computer opgeven. Zo'n uitdrukking noemen we numeriek indien het resultaat van deze uitdrukking numeriek is. We noemen een uitdrukking alfanumeriek indien het resultaat van deze uitdrukking alfanumeriek is. In een uitdrukking vinden we de volgende elementen:

- variabelen. Deze hebben een waarde die al eerder bepaald is.
- constanten. Deze hebben een vastgestelde waarde.
- bewerkingscodes. Deze geven aan welke bewerkingen dienen te geschieden.
- voorrangstekens. Deze worden altijd gevormd door haakjes. Met deze voorrangstekens kan een afwijkende voorrang in bewerking worden vastgelegd.

We kunnen een uitdrukking als volgt definiëren:

EEN UITDRUKKING IS EEN IN PRINCIPE UITVOERBAAR SAMENSTELSEL VAN CONSTANTEN, VARIABELEN, BEWERKINGEN EN VOORRANGSTEKENS.

Op de eerste twee elementen, de constanten en variabelen, zijn we in de voorgaande hoofdstukken reeds diep ingegaan. In de volgende paragrafen gaan we op bewerkingen, voorrang en voorrangstekens in.

7.2 Algebraïsche bewerkingen

Het MSX-basic kent de volgende algebraïsche bewerkingen:

- \+ optellen. Met het plus-teken geven we aan dat de uitdrukkingen links en recht van dit teken bij elkaar dienen te worden opgeteld. Dit is de enige bewerking die ook op alfanumerieke waarden mag worden toegepast.
- \- aftrekken. Met het min-teken geven we aan dat de uitdrukkingen links en rechts van dit teken van elkaar moeten worden afgetrokken. Deze bewerking mag alleen op numerieke bewerkingen worden toegepast.
- * vermenigvuldigen. Met het sterretje of vermenigvuldigingsteken geven we aan dat de uitdrukkingen rechts en links van dit teken met elkaar dienen te worden vermenigvuldigd. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- / delen. Met de deelstreep geven we aan dat de uitdrukkingen links en rechts van dit teken door elkaar moeten worden gedeeld. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- ^ machtsverheffen. Met het 'dakje' geeft men aan, dat de uitdrukking links van dit teken in een macht dient te worden verheven. De uitdrukking rechts van dit teken geeft aan om welke macht het gaat. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.
- \ integer delen. Met deze omgedraaide deelstreep geeft men aan dat de uitdrukkingen links en rechts van dit teken eerst dienen te worden gemaakt tot een integere waarde (een gehele waarde, minimaal gelijk aan −32768 en maximaal gelijk aan 32767). Vervolgens worden deze integere waarden op elkaar gedeeld. De uitkomst wordt een tweede maal gemaakt tot

een integere waarde. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.

MOD restbepaling. Met dit sleutelwoord geeft men aan dat de uitdrukkingen links en rechts van dit sleutelwoord eerst dienen te worden gemaakt tot een integere waarde. Daarna dienen deze twee integere waarden door elkaar te worden gedeeld. De restwaarde die bij deling overblijft (er wordt in dit geval niet na de decimale punt doorgedeeld) is de uitkomst van deze bewerking. Deze bewerking mag alleen op numerieke uitdrukkingen worden toegepast.

Met name de laatste twee bewerkingen kunnen wat problemen geven. Hieronder volgen enkele voorbeelden, verwerkt in één programma:

```
NEW
Ok
10 INPUT "EERSTE WAARDE ";A
20 INPUT "TWEEDE WAARDE ";B
30 PRINT "OPTELLING ";A+B
40 PRINT "AFTREKKING ";A-B
50 PRINT "VERMENIGVULDIGING ";A*B
60 PRINT "DELING ";A/B
70 PRINT "INTEGERE DELING ";A\B
80 PRINT "REST BIJ DELING ";A MOD B
90 PRINT "MACHT ";A^B
100 GOTO 10
```

Indien we voor A of B een waarde, groter dan 32767 of kleiner dan –32768 ingeven, dan zal programmaregel 70 een foutmelding geven; A of B zijn niet tot integere waarden te maken.
Wanneer we voor A bijvoorbeeld 25.68 ingeven en daarbij voor B bijvoorbeeld 6.99 ingeven, dan geeft programmaregel 70 de uitkomst 4. A wordt namelijk eerst tot een integere waarde (25) gemaakt. Hetzelfde gebeurt met B (6). Deze twee waarden worden op elkaar gedeeld (25/6=4.66...). De uitkomst wordt tenslotte weer integer gemaakt (4).
Programmaregel 80 geeft bij de genoemde waarden voor A en B de uitkomst 1. De deling 25/6 geeft de uitkomst 4 *rest 1*.
Wanneer B te groot wordt ingegeven, veroorzaakt regel 90 al snel een foutmelding omdat de uitkomst veel te groot wordt. Merk op dat ook oneigenlijke machten (gebroken getallen in de macht) zijn toegestaan.

MSX-basic vertoont bij de bewerking \ één foutje. Tik bijvoorbeeld eens in:

```
? -32768\-1          (? mag in plaats van PRINT worden gebruikt)
```

De computer zal als uitkomst de waarde 32768 geven. Deze waarde is niet integer want een integere waarde mag niet groter zijn dan 32767. Ten onrechte wordt hier geen foutmelding gegeven...

7.3 Functionele bewerkingen

Het MSX-basic kent een groot aantal functionele bewerkingen. In de hoofdstukken 9 en 10 zijn deze met N-FUNKTIE of A-FUNKTIE aangeduid.
Een functionele bewerking is een bewerking die kan worden toegepast op een uitdrukking of op een stelsel van uitdrukkingen. Een functionele bewerking heeft altijd de vorm van:

FUNCTIONELE BEWERKING (UITDRUKKING...)

Om het één en ander wat nader toe te lichten, volgen hieronder enkele voorbeelden van functionele bewerkingen.

Tik bijvoorbeeld eens in:

```
PRINT INT(12.9)
```

De computer zal antwoorden met het getal 12. De functionele bewerking INT heeft tot gevolg dat het grootste gehele getal kleiner dan de tussen de haakjes staande uitdrukking wordt berekend.

```
PRINT LEFT$("ABCDEFGHIJKLMNOP",5)
```

De computer antwoordt hier met de uitkomst ABCDE. De functie LEFT$ heeft tot gevolg dat het linkergedeelte van de opgegeven alfanumerieke uitdrukking wordt bepaald. Het aantal tekens is gegeven in de tweede tussen haakjes gegeven uitdrukking, die numeriek is.

```
PRINT SGN(-12.8)
```

De computer antwoordt hier met de uitkomst −1. De functie SGN kan drie verschillende waarden geven, namelijk 1 (indien de uitdruk-

king tussen haakjes positief is), 0 (indien de uitdrukking tussen haakjes gelijk is aan 0) of –1 (indien de uitdrukking tussen haakjes negatief is).

```
PRINT SIN(3.14)
```

Alleen die programmeurs (in spé) die wat van goniometrie afweten, zullen deze functie begrijpen. De functie geeft de sinus van de waarde van de uitdrukking tussen haakjes. SIN gaat ervan uit dat de uitdrukking tussen haakjes een hoek in radialen voorstelt.

In de hoofdstukken 9 en 10 wordt onderscheid gemaakt tussen A-FUNCTIES en N-FUNCTIES. A-FUNCTIES zijn functionele bewerkingen met een alfanumeriek resultaat (zoals LEFT$) en N-FUNCTIES zijn functionele bewerkingen met een numeriek resultaat (zoals INT).

Een voorbeeldprogramma:

```
NEW
Ok
10 INPUT "GEEF EEN WAARDE IN ";WAARDE
20 LET A=INT(WAARDE)
30 LET B=SGN(WAARDE)
40 PRINT "INTEGERE WAARDE ";A
50 IF B=-1 THEN PRINT "NEGATIEF"
60 PRINT LEFT$("ABCDEFGHIJKLMNOPQRSTUVWXYZ",WAARDE)
70 GOTO 10
```

Dit voorbeeldprogramma vraagt eerst een numerieke ingave in WAARDE. Vervolgens worden in programmaregel 20 en 30 twee functies op deze waarde toegepast; de resultaten worden onder de variabelenamen A en B bewaard. Op programmaregel 40 wordt dan de integere waarde op beeldscherm getoond.
Op programmaregel 50 komen we een nieuw sleutelwoord tegen: IF. Deze regel kan gelezen worden als: als variabele B gelijk is aan –1 (dus als een negatieve waarde werd ingegeven), druk dan de tekst NEGATIEF af. Met het IF-sleutelwoord kunnen we dus een voorwaarde stellen. Als aan deze voorwaarde wordt voldaan, wordt het daarna opgenomen commando uitgevoerd. Op dit sleutelwoord komen we verderop nog uitgebreid terug.
Op programmaregel 60 worden de eerste letters van het alfabet afgedrukt. Het aantal letters is bepaald door de waarde van variabele WAARDE die wij ingaven. Merk op dat programmaregel 60 een fout geeft bij een negatieve waarde.

7.4 Relationele bewerkingen

Het MSX-basic kent een aantal relationele bewerkingen. Een relationele bewerking is een bewerking die uitdrukking geeft aan een relatie.

Tik bijvoorbeeld eens in:

```
PRINT 2=2
```

De computer zal het antwoord –1 geven. Dit antwoord betekent dat de bewering 2=2 inderdaad waar is. De uitdrukking links van het gelijkteken en de uitdrukking rechts van het gelijkteken hebben de relatie, dat ze in waarde gelijk zijn aan elkaar. Tik bijvoorbeeld eens in:

```
PRINT (5+2)=(8-1)
```

Ook hier zal het oordeel van de computer WAAR (–1) luiden; 5+2 is inderdaad gelijk aan 8–1. Tik nu eens in:

```
PRINT 5+3=7
```

Het oordeel van de computer is nu 0 (NIET WAAR). De uitdrukkingen 5+3 en 7 hebben niet de relatie gelijkheid met elkaar.

Het gelijkteken in de hiervoor gegeven voorbeelden is een relationele bewerking. Een relationele bewerking dient altijd plaats te vinden tussen twee numerieke uitdrukkingen of tussen twee alfanumerieke uitdrukkingen. Het resultaat is altijd –1 (WAAR) of 0 (NIET WAAR). Nog een voorbeeld met alfanumerieke uitdrukkingen:

```
PRINT "JAN"="J"+"AN"
```

en

```
PRINT "JAN"="JANNEMAN"
```

De eerste opdracht zal resulteren in het antwoord –1 (WAAR), de tweede in het antwoord 0 (NIET WAAR).

De bewerking 'is gelijk aan' (=) is slechts één van de relationele bewerkingen die het MSX-basic kent. Hieronder volgt een tabel met alle relationele bewerkingen:

RELATIONELE BEWERKING	BETEKENIS
=	is gelijk aan
<	is kleiner dan
>	is groter dan
<> OF ><	is kleiner of groter dan (ofwel: is ongelijk aan)
<= OF =<	is kleiner dan of gelijk aan (ofwel: is ten hoogste gelijk aan)
>= OF =>	is groter dan of gelijk aan (ofwel: is ten minste gelijk aan)
=<> OF =>< OF <=> OF <>= OF >=< OF ><=	is kleiner dan, gelijk aan of groter dan

De laatste relationele bewerking geeft altijd een waar resultaat (−1) en heeft als zodanig geen enkele zin; we zullen deze bewerking verder vergeten.

In principe zijn er slechts drie relationele bewerkingen (<, = en >). De andere relationele bewerkingen zijn combinaties van deze drie basisbewerkingen.

Een voorbeeldprogramma:

```
NEW
Ok
10 INPUT "WAARDE ";A
20 INPUT "NOG EEN WAARDE ";B
30 IF A<B THEN PRINT "DE EERSTE WAARDE IS KLEINER"
40 IF A=B THEN PRINT "DE TWEE WAARDEN ZIJN GELIJK"
50 IF A>B THEN PRINT "DE EERSTE WAARDE IS GROTER"
60 IF A<=B THEN PRINT "DE EERSTE WAARDE IS KLEINER
 DAN OF GELIJK AAN DE TWEEDE WAARDE"
70 GOTO 10
```

Na enig proberen zal het volgende blijken: indien een uitdrukking de uitkomst –1 heeft (WAAR IS), wordt het commando (of worden de commando's) achter THEN in een IF...THEN constructie uitgevoerd. Is de uitdrukking echter niet waar (de uitkomst is dan 0) dan gebeurt dit niet.

Probeer nu eens het volgende programma:

```
NEW
Ok
10 INPUT "WAARDE ";A
20 IF A THEN PRINT "WAAR"
30 GOTO 10
```

en laat dit programma eens uitvoeren. Geef diverse waarden in, negatief, positief en ook eens een 0. De computer zal steeds antwoorden met de tekst "WAAR" tenzij een 0 werd ingegeven. Conclusie:

> *Een relationele bewerking geeft bij uitvoering de uitkomst waar of niet waar. Deze oordelen worden gesymboliseerd in de waarden –1 of 0. In de IF...THEN-constructie kan het al dan niet waar zijn van een uitdrukking worden getest. Indien een uitdrukking als waar wordt bevonden, wordt het programmaregelgedeelte achter het sleutelwoord THEN uitgevoerd; tussen IF en THEN mag elke uitdrukking worden geplaatst. Deze uitdrukking behoeft niet verplicht een relationele bewerking te bevatten. De IF...THEN-constructie beschouwt een uitdrukking als waar indien deze ongelijk is aan nul (zie het laatste voorbeeld, de uitkomst hoeft dus niet verplicht gelijk te zijn aan –1) en als niet waar indien deze wel gelijk is aan nul.*

Merk op dat het gelijkteken binnen MSX-basic een dubbele functie uitoefent.

Eerste functie: LET A=12.34	in een dergelijke constructie vormt het gelijkteken geen relationele bewerking maar geeft het slechts aan dat een variabele dient te worden gelijkgesteld aan een uitdrukking.
Tweede functie: PRINT A=B	in een dergelijke constructie vormt het gelijkteken een relationele bewerking.

7.5 Logische bewerkingen

Het MSX-basic biedt verscheidene logische bewerkingen. Logische bewerkingen dienen ondermeer om zeer ingewikkelde afvragingen te verrichten.
De beginnende programmeur kan, zeker in eerste instantie, misschien deze paragraaf beter overslaan.

Een logische bewerking vergelijkt twee uitdrukkingen en doet vervolgens een uitspraak in de zin van WAAR (ongelijk aan 0) of NIET WAAR (gelijk aan 0). Aan de hand van de meest gemakkelijke logische bewerking lichten we dit wat nader toe. Tik bijvoorbeeld eens in:

eerste regel:

```
IF 2+3=5 AND 4+6=10 THEN PRINT "HOERA"
```

tweede regel:

```
IF 2+4=6 AND 6+6=11 THEN PRINT "EEN VREEMDE ZAAK"
```

derde regel:

```
IF "JAN"<"KAREL" AND 4*4=16 THEN PRINT "HOERA"
```

De eerste regel resulteert evenals de derde regel in het afdrukken van de tekst HOERA op het beeldscherm. De tweede regel blijft zonder resultaat. Verklaring: De logische bewerking AND onderzoekt de uitdrukking links en rechts van deze bewerking. Alleen indien beide beweringen WAAR zijn (–1 als uitkomst hebben) dan krijgt de totale uitdrukking tussen IF en THEN de beoordeling WAAR (de waarde –1) en wordt het gedeelte achter THEN uitgevoerd. De derde regel geeft een extra moeilijkheid; er wordt een bewering gedaan met twee teksten waarbij wordt beweerd dat de eerste tekst kleiner is dan de tweede tekst. Indien de uitdrukkingen waarop een relationele bewerking wordt toegepast, alfanumeriek zijn, dan worden deze uitdrukkingen alfabetisch uitgewaardeerd. Een tekst die bij een alfabetische rangschikking voor een tweede tekst terecht zou komen, is ook kleiner dan deze tweede tekst.

Over het algemeen kan men stellen dat karakters volgens de MSX-karaktertabel (hoofdstuk 15) worden vergeleken waarbij een karakter met de laagst gecodeerde waarde ook het kleinst is. Dat betekent ondermeer dat kleine letters altijd groter zijn dan hoofdletters...

De eerste voorbeeldregel zou in normaal nederlands als volgt kunnen worden vertaald:

Als 2+3 gelijk is aan 5 *en* 4+6 is gelijk aan 10, druk dan de tekst HOERA af.

De logische bewerking AND kan men dus vertalen met het woordje EN.

Het MSX-basic kent diverse logische bewerkingen. Hieronder volgt een tabel met alle logische bewerkingen die in MSX-basic bestaan:

logische bewerking	betekenis ongeveer	uitleg
AND	en	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft de uitkomst WAAR (ongelijk aan nul) alleen indien de beide uitdrukkingen WAAR (ongelijk aan nul) zijn.
OR	of	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft de uitkomst WAAR indien minstens één van de twee uitdrukkingen WAAR is.
XOR	exclusief of	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst WAAR indien minstens en hoogstens één van de uitdrukkingen WAAR is.
EQV	gelijkwaardig met	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst WAAR indien beide uitdrukkingen WAAR zijn of beide uitdrukkingen NIET WAAR zijn (wanneer beide uitdrukkingen dus gelijkwaardig zijn).
IMP	impliceert dat	deze logische bewerking wordt tussen twee uitdrukkingen geplaatst. Hij geeft slechts de uitkomst NIET WAAR indien de eerste uitdrukking waar is maar de

		tweede niet (wanneer het WAAR zijn van de eerste uitdrukking niet impliceert dat de tweede uitdrukking WAAR is).
NOT	niet	deze logische bewerking wordt vóór een tussen haakjes gestelde uitdrukking geplaatst en draait de waarde van deze uitdrukking om. Indien deze uitdrukking WAAR is, veroorzaakt deze bewerking dus het antwoord NIET WAAR en omgedraaid.

Een voorbeeldprogramma ter afsluiting:

```
NEW
Ok
10 REM VOORBEELD LOGISCHE BEWERKINGEN
20 INPUT "HOEVEEL IS 2+3 ";A
30 INPUT "HOEVEEL IS 5+7 ";B
40 IF A=5 AND B=12 THEN PRINT "ALLEBEI DE SOMMEN GOED"
50 IF A=5 OR B=12 THEN PRINT "EEN OF TWEE SOMMEN GOED"
60 IF A=5 XOR B=12 THEN PRINT "SLECHTS EEN SOM GOED"
70 IF NOT(A=5 OR B=12) THEN PRINT "ALLES FOUT"
80 IF A=5 EQV B=12 THEN PRINT "ALLEBEI GOED OF ALLEBEI FOUT"
90 IF NOT(A=5 IMP B=12) THEN PRINT "DE EERSTE GOED MAAR DE T
WEEDE FOUT"
100 GOTO 20
```

Bij het proberen van dit programma moet u eens wat goede en wat foute antwoorden aan de computer geven. Let eens op zijn bevindingen. Indien u beide antwoorden goed gaf, geeft de computer het commentaar BEIDE SOMMEN GOED, EEN OF TWEE SOMMEN GOED en ALLEBEI GOED OF ALLEBEI FOUT. Dit zijn de drie juiste conclusies die via de AND, de OR en de EQV bewerkingen konden worden getrokken.

Het werken met logische bewerkingen vereist een logisch redeneervermogen maar ook een dosis oefening. Alhoewel het gebruik hiervan in eerste instantie niet gemakkelijk is, zal in latere instantie het nut van deze logische bewerkingen zich bewijzen.

7.6 Logische bewerkingen voor gevorderen

Deze paragraaf dient door beginnende programmeurs zeker te worden overgeslagen daar hier erg diep op logische bewerkingen wordt ingegaan. Het is niet aannemelijk dat een beginnende programmeur hier al veel aan heeft.

In de vorige twee paragrafen werd een beperking gehandhaafd die het één en ander wat gemakkelijker maakte. Wanneer we relationele en logische bewerkingen op de keper willen beschouwen, dienen we het volgende vast te stellen.

1) een relationele bewerking heeft de uitkomst –1 (WAAR) of 0 (NIET WAAR) tot gevolg. Deze uitkomst dient te worden beschouwd als een integere waarde die via de tweecomplementmethode wordt gepresenteerd. WAAR of –1 betekent in dat geval: alle bits binnen deze (twee bytes) waarde op 1 en NIET WAAR betekent: alle bits op 0.
2) de logische bewerkingen mogen slechts worden toegepast op uitdrukkingen die een integere waarde tot gevolg hebben of tot een integere waarde kunnen worden gemaakt.
3) logische bewerkingen oefenen hun bewerking op bit-niveau uit (vandaar de uitkomst –1 en niet 1 bij WAAR).

Het een en ander heeft tot gevolgd dat de geoefende programmeur in MSX-basic op bit-niveau zijn afvragingen kan doen waarbij het in sommige toepassingen mogelijk is om zestien afvragingen in één keer te doen.
Het toepassen van de logische bewerkingen op dit niveau vereist een paar kwaliteiten van de programmeur:

– een perfekte kennis van het binaire talstelsel
– een volledige beheersing van de tweecomplementmethode
– een gedegen kennis van de Boole algebra

Het voert te ver om in dit handboek op deze drie gebieden verder in te gaan; de geïnteresseerde kan zich het beste op de hoogte stellen met de vakliteratuur die op deze onderwerpen bestaat.

Eén tip: oefening kan het best geschieden met een volgende constructie:

```
PRINT$((&B1101) AND (&B1011))
```

In plaats van AND kan natuurlijk elke logische bewerking (behalve NOT) worden geplaatst. Het resultaat van de op bitniveau uitgevoerde logische bewerking wordt binair, dus ook weer bitsgewijs, afgedrukt waardoor een bitsgewijze studie van de werking van logische bewerkingen gemakkelijk mogelijk is. Zie ondermeer de behandeling van BIN$ (een A-FUNCTIE), in hoofdstuk 9.

Merk op dat de binaire constanten van het laatste voorbeeld tussen haakjes moeten. Een foutje in MSX-basic zorgt anders ten onrechte voor een foutmelding (die dan weer opgelost kan worden door de spatie voor AND te verwijderen).

Tot slot volgt voor de gevorderde lezer een waarheidstabel met betrekking tot de logische bewerkingen:

bewerking	eerste argument	tweede argument	resultaat
NOT	0		1
	1		0
AND	0	0	0
	0	1	0
	1	0	0
	1	1	1
OR	0	0	0
	0	1	1
	1	0	1
	1	1	1
XOR	0	0	0
	0	1	1
	1	0	1
	1	1	0
EQV	0	0	1
	0	1	0
	1	0	0
	1	1	1
IMP	0	0	1
	0	1	1
	1	0	0
	1	1	1

7.7 Voorrangsregels

Wanneer in een uitdrukking veel bewerkingen voorkomen, dan zal al snel de twijfel bestaan welke bewerking het eerste dient te geschieden. Bekijken we bijvoorbeeld de uitdrukking in de volgende programmaregel:

```
PRINT 2*3+4
```

De uitkomst zal luiden: 10. Dit wijst erop dat eerst de vermenigvuldiging en pas later de optelling is uitgevoerd. De vermenigvuldiging had blijkbaar voorrang op de optelling. We kunnen deze voorrang wijzigen door het een en ander tussen haakjes te plaatsen:

```
PRINT 2*(3+4)
```

In dit geval wordt eerst de uitdrukking tussen haakjes uitgewerkt en pas daarna de vermenigvuldiging gedaan. De uitkomst luidt dan ook: 14.

Indien de moeilijkheid zich zou beperken tot vermenigvuldigen en optellen, dan zou er niet zoveel aan de hand zijn. Echter, binnen MSX-basic hebben we een keur van algebraïsche, relationele, functionele en logische bewerkingen. In dat woud van bewerkingen is het nuttig om te weten welke bewerkingen voorrang hebben op welke andere bewerkingen in MSX-basic. De voorrangsregels zijn niet zo erg moeilijk; ze komen behoorlijk overeen met de voorrangsregels die algemeen in rekenwerk worden toegepast.

Allereerst noemen we de haakjes. Een gouden regel is dat *haakjes de allerhoogste voorrang* hebben. De uitdrukking die het diepst tussen haakjes staat, wordt altijd het eerst door de computer uitgewerkt. Dit geldt ook voor de haakjes die verplicht zijn bij een functionele bewerking.

Buiten deze haakjes liggen de voorrangsregels als volgt:

voorrangs-volgorde	bewerkingen	opmerkingen
1	functionele bewerkingen	Door het verplichte gebruik van haakjes kunnen onderling geen voorrangsconflicten ontstaan.
2	algebraïsche bewerkingen	eerst ^
		dan *, /, \ en MOD 1)
		dan + en - 1)
3	relationele bewerkingen	1)
4	logische bewerkingen	eerst NOT (haakjes)
		dan AND
		dan OR,XOR,EQV en IMP 1)

1) Indien er tussen gelijkwaardige bewerkingen conflictsituaties ontstaan, dan worden deze bewerkingen van links naar rechts in volgorde uitgevoerd.

7.8 Precisie tijdens berekeningen

Bij het uitwerken van uitdrukkingen dient de computer vaak vrij ingewikkelde berekeningen te maken. Alhoewel dit schijnbaar moeiteloos gebeurt, komt voor een berekening vaak veel kijken. Tik eens in:

```
PRINT 2+40\INT(2*5+4.1)
```

Hoegenaamd onmiddellijk produceert de computer het juiste antwoord. Laten we eens nagaan, welke bewerkingen de computer moest doen om tot dit antwoord te komen.

Uit de voorrangsregels volgt, dat de berekening als volgt dient te worden uitgevoerd:

uitdrukking	`2+40\INT(2*5+4.1)`	
stap 1	`2+40\INT(10+4.1)`	
stap 2	`2+40\INT(14.1)`	
stap 3	`2+40\14`	
stap 4	`2+2`	
stap 5	`4`	(uitkomst)

Tijdens deze uitvoerige berekeningen moeten nogal eens tussenuitkomsten door de computer worden bewaard. Deze tussenuitkomsten worden in zogenaamde naamloze systeemvariabelen bewaard en zijn in MSX-basic niet te benaderen. Om onnauwkeurigheden tijdens de berekeningen te voorkomen, worden deze naamloze systeemvariabelen altijd toegewezen als variabelen met *dubbele precisie.*

Een uitzondering wordt gevormd door de logische en relationele bewerkingen. Het resultaat hiervan wordt, indien dit als tussenuitkomst dient te worden opgeslagen, altijd als integere naamloze systeemvariabele opgeslagen.

8.1 De BNF-notatie

We zagen reeds eerder dat een computer een dom apparaat is. Om de computer te laten doen wat we van hem wensen, is het noodzakelijk om zijn taal te beheersen, in dit geval het MSX-basic.

Juist omdat de computer in feite een dom apparaat is, is hij niet in staat om gegevens te interpreteren, om een betekenis te geven aan begrippen. Elke fout, al is het maar het vergeten van een komma of een verkeerd aangeslagen letter, wordt door de computer dan ook onmiddellijk gemeld; de computer is niet in staat om te onderzoeken wat we eigenlijk bedoelen.

Om deze reden is het noodzakelijk om de computer te voeden met programmagegevens die tot op de laatste letter precies volgens de voorschriften van de computertaal zijn.

Om een computertaal precies volgens de voorschriften te kunnen gebruiken, dienen we deze voorschriften zélf eerst te kennen.

In de hoofdstukken 9 en 10 wordt van elk sleutelwoord de exacte schrijfwijze (syntaxis) gepresenteerd, alsmede een omschrijving van wat het sleutelwoord voor ons kan doen (semantiek).

Om in de preciese *schrijfwijze* van elk sleutelwoord geen twijfel te laten bestaan, is gekozen voor **een** speciale methode van notatie van de schrijfwijze. Deze speciale notatiemethode noemen we de BNF-notatie, naar de uitvinder van deze notatievorm (BNF=Backus Normal Form). Omdat het MSX-basic één van de meest uitgebreide talen is, schieten de mogelijkheden van de BNF-notatie soms te kort. Daarom zijn bij de normale symbolen uit de BNF-notatie wat extra symbolen toegevoegd.

Om bij de schrijfwijze snel te kunnen bepalen welke de juiste is, is een zekere kennis van de BNF-notatiewijze zeer welkom. Echter, de notatiewijze vereist een logisch denkvermogen dat zeker bij de beginnende programmeur niet altijd aanwezig behoeft te zijn.

Het is verstandig om dit hoofdstuk in ieder geval zo veel mogelijk door

te nemen. Het is daarbij geen probleem dat niet alles wordt begrepen. Wel dienen de voorgaande hoofdstukken grondig te zijn doorgenomen.

Indien later in hoofdstuk 9 nog problemen bestaan doordat de notatie van de voorgeschreven schrijfwijze niet helemaal duidelijk is, dan kan de laatste twijfel meestal worden weggenomen door de voorbeelden die bij de sleutelwoorden zijn opgenomen, te bestuderen.

8.2 Een eerste kennismaking met de BNF-notatie

In de volgende paragrafen volgt een uitleg van de BNF-notatievorm. Daarbij zijn wat algemene definities (b.v. die van een alfanumerieke constante) in BNF opgenomen. Naar deze algemene specificaties wordt in hoofdstuk 9 en 10 steeds weer verwezen.

Om de BNF in te leiden, volgt in deze paragraaf een eenvoudig en uitgewerkt voorbeeld.

Allereerst beginnen we met de standaard BNF-symbolen.
Deze zijn:

[ ]	het tussen vierkante haken opgenomen gedeelte mag naar wens al of niet worden geplaatst
{ }	het tussen accolades opgenomen gedeelte mag weggelaten worden of één- of meermalen worden geplaatst
< >	het tussen scherpe haken opgenomen gedeelte vormt een beschrijving. Deze beschrijving kan later weer verder worden behandeld
¦	dit teken betekent *of*. Naar keuze mag *of* het linkergedeelte *of* het rechtergedeelte worden opgenomen
: : =	dit teken betekent: is per difinitie gelijk aan

Laten we met behulp van deze symbolen eens proberen of we de schrijfwijze van een cijfer kunnen bepalen:

```
<CIJFER>::=0¦1¦2¦3¦4¦5¦6¦7¦8¦9
```

Deze definitie hebben we eenvoudig vastgelegd. We hebben bepaald

dat een cijfer of een 0, of een 1, of een 2 etcetera is. Nu we het begrip cijfer, hoe kinderachtig het ook lijkt, hebben vastgelegd, kunnen we dit begrip verder gebruiken.

Laten we eens proberen om het begrip integere constante vast te leggen. We hoeven ons in dit stadium niet druk te maken over geldende minima of maxima.

We proberen:

```
<INTEGERE KONSTANTE>::={<CIJFER>}
```

Het tussen de accolades geplaatste begrip mag herhaald voorkomen. We hebben met deze eenvoudige definitie dus bepaald dat bijvoorbeeld 0, 125, 3842 etcetera een integere constante is.

Echter, er zit een fout in deze definitie. Het tussen accolades geplaatste begrip mag volgens de omschrijving van de symbolen namelijk ook *weggelaten* worden (=0 keer voorkomen). Een integere constante zou volgens deze definitie dus ook uit *niets* kunnen bestaan!

Aangezien een integere constante minimaal één cijfer dient te bevatten, dienen we onze definitie wat aan te passen.
We proberen:

```
<INTEGERE KONSTANTE>::=<CIJFER>{<CIJFER>}
```

Uit deze definitie volgt dat een integere constante uit één of meer cijfers bestaat. Het eerste probleem is opgelost. Echter, een volgende onvolkomenheid dient zich aan: we weten dat −100 bijvoorbeeld een integere constante is. Volgens onze definitie is dit echter (nog) niet zo. We moeten deze definitie dus nog wat verder aanpassen:

```
<INTEGERE KONSTANTE>::=[+|-]<CIJFER>{<CIJFER>}
```

Met deze definitie hebben we ook die onvolkomenheid verholpen. Deze laatste definitie bepaalt dat een integere constante als eerste teken een + of een − mag bevatten; het is echter niet verplicht.

Omdat we bij het omschrijven van de syntaxis (schrijfwijze) ons niet hoeven druk te maken om geldende maxima of minima, is deze definitie verder in orde!

Het zal duidelijk zijn dat onze definitie slechts een héél eenvoudige is, in de volgende paragrafen en hoofdstukken komen we veel ingewikkeldere omschrijvingen tegen. Bedenk daarbij echter dat er naar is gestreefd om deze omschrijvingen zo correct mogelijk uit te voeren. De doorgewinterde BNF-lezer kan hier erg veel informatie uit halen; voor de beginnende programmeur is het voldoende dat hij of zij een *behoorlijke indruk* van de schrijfwijze verkrijgt!

8.3 Algemene opmerkingen inzake de notatie van MSX-basic in BNF

Voorafgaande aan de BNF-behandeling en de behandeling van enige algemene specificaties dienen eerst de volgende opmerkingen te worden gemaakt:

- Binnen de in dit hoofdstuk opgenomen notaties zijn geen restricties aangebracht inzake geldende minimale waarden, maximale waarden en maximale lengten. Zie hiervoor de behandeling van de constanten en variabelen in de voorgaande hoofdstukken. Daarbij dient te worden vermeld dat een ⟨kommando⟩, een ⟨rij van kommando's⟩ en een ⟨programmaregel⟩ nooit de lengte van 255 karakters mogen overschrijden.
- Behalve bij de definitie van ⟨letter⟩ mogen in alle in dit hoofdstuk genoemde notaties, kleine letters en hun corresponderende
- hoofdletters vrij met elkaar worden verwisseld.
  Sleutelwoorden dienen steeds 'aaneensluitend' te worden gebruikt. Buiten deze restrictie is het MSX-basic 'spatie-ongevoelig'; op willekeurige plaatsen buiten sleutelwoorden mogen naar believen spaties worden tussengevoegd.
- De behandelde sleutelwoorden zijn 'gereserveerd'. Variabelenamen mogen nooit gelijk zijn aan een sleutelwoord of een sleutelwoord bevatten. Zie voor een totale opsomming van alle gereserveerde sleutelwoorden hoofstuk 16.
- Om de schrijfwijze (syntaxis) van MSX-basic zo nauwkeurig mogelijk te omschrijven, moest een onderscheid gemaakt worden tussen numerieke en alfanumerieke variabelen en constanten. Echter, door het dynamische karakter van MSX-basic wat betreft de typering van variabelen (via DEFSTR, DEFINT, DEFSNG en DEFDBL) kunnen sommige definities niet volledig worden uitgedefinieerd, maar moet worden volstaan met een vrij breedvoerige omschrijving.

– De definitie voor ⟨N⟩ (de numerieke uitdrukking) is niet sluitend. In een enkel geval kan een volgens de voorgeschreven schrijfwijze opgebouwde ⟨N⟩ toch een 'type mismatch' foutmelding geven. Deze foutmelding is te wijten aan een niet ver genoeg gaande vaststelling van de volgorde van uitvoering van bewerkingen bij verschillende typen variabelen en/of constanten in MSX-basic.
– Alhoewel de ELSE constructie voor en achter elk commando is toegestaan en zelfs als enig commando is toegestaan, heeft deze constructie slechts zin indien gebruikt achter een IF...THEN-constructie. In elke andere constructie vindt executie van een navolgend gedeelte *nooit* plaats. Daar een syntactische controle pas op het moment van executie plaats vindt, mag achter een ELSE, niet voorkomend in een IF...THEN-constructie, ⟨...⟩ (niet ter zake doende opvolging van karakters) worden opgenomen (zie paragraaf 8.4).

8.4 Gebruikte symbolen

De BNF-notatie kent standaard de volgende symbolen:

symbool	betekenis
[ ]	de tussenliggende bepaling mag 0 of 1 maal worden opgenomen (optie)
{ }	de tussenliggende bepaling mag 0 of meer maal worden opgenomen (repetitie of herhaling)
< >	de tussenliggende tekst verwijst naar een andere definitie of bevat een eenduidige omschrijving
: : =	betekent: is per definitie gelijk aan
¦	betekent: of. Gekozen dient te worden tussen het links van dit teken of het rechts van dit teken vermelde gedeelte.

Om het zéér uitgebreide MSX-basic met BNF te kunnen beschrijven, dienen enige aanvullende symbolen te worden gebruikt. Deze zijn:

uitbreidings symbool	betekenis
<...>	(let op de puntjes). Betekent: een willekeurige, niet terzake doende opvolging van karakters. Dus: <...>::={<KARAKTER>} Vanwege de nadruk dat het een en ander niet ter zake doet, is gekozen voor een apart symbool
———	betekent *of*. Gekozen dient te worden tussen het boven dit teken of onder dit teken vermelde gegeven
\	betekent: met uitzondering van, behalve. De voor dit teken opgenomen definitie moet worden *beperkt* met de na dit teken opgenomen definitie
<?>	nog niet vastgelegde definitie of omschrijving
. . .	logische voortzetting

8.5 Elementaire definities

```
<NIETS>::=

<LETTER>::A|B|C|...|Z|a|b|c|...|z

<CIJFER>::=0|1|2|3|...|9

<KARAKTER>::=<ALLE KARAKTERS, VERMELD IN DE MSX-TABEL
             BEHALVE DE KARAKTERS MET EEN ASCII-WAARDE
             ONDER DE 32 DECIMAAL>

<OCTAAL CIJFER>::=0|1|2|3|4|5|6|7

<BINAIR CIJFER>::=0|1

<HEXADECIMAAL CIJFER>::=0|1|2|3|...|9|A|B|C|D|E|F
```

<ALFANUMERIEKE KONSTANTE>::=(<KARAKTER>)

<TEKENLOZE INTEGERE KONSTANTE>::=<CIJFER>(<CIJFER>)

<INTEGERE KONSTANTE>::=[+|-]<TEKENLOZE INTEGERE KONSTANTE>

<NUMERIEKE KONSTANTE>::=<MANTISSE>[#|!|% | D ---<EXPONENT> | E]|<OCTALE KON...

...STANTE>|<BINAIRE KONSTANTE>|<HEXADECIMALE KONSTANTE>

<MANTISSE>::=[<INTEGERE KONSTANTE>][.[<TEKENLOZE INTEGERE KONSTANTE>]]\<NIETS>

<EXPONENT>::=<INTEGERE KONSTANTE>

<OCTALE KONSTANTE>::=&O(<OCTAAL CIJFER>)

<BINAIRE KONSTANTE>::=&B(<BINAIR CIJFER>)

<HEXADECIMALE KONSTANTE>::=&H(<HEXADECIMAAL CIJFER>)

```
<VARIABELE-NAAM>::=<LETTER>{<LETTER>|<CIJFER>}[%|!|#|$]

<NUMERIEKE VARIABELE-NAAM>::=<VARIABELE-NAAM>\<...>$

                                                                  %
                                                                 ---
<ALFANUMERIEKE VARIABELE-NAAM>::=<VARIABELE-NAAM>\<...>          !
                                                                 ---
                                                                  #
```

MERK OP DAT <LETTER>{<LETTER>|<CIJFER>} ZOWEL EEN NUMERIEKE ALS EEN ALFANUMERIEKE VARIABELE-NAAM KAN ZIJN. IN HET NORMALE GEVAL ZAL EEN DERGELIJKE VARIABELE EEN NUMERIEKE VARIABELE MET DUBBELE PRECISIE VERTEGENWOORDIGEN. DE FUNKTIES DEFSTR, DEFINT EN DEFSNG KUNNEN HIER IN ECHTER VERANDERING BRENGEN (ZIE HOOFDSTUK 9).

```
<NUMERIEKE VARIABELE>::=<NUMERIEKE VARIABELE-NAAM>[<INDEXERING>]

<ALFANUMERIEKE VARIABELE>::=<ALFANUMERIEKE VARIABELE-NAAM>[<INDEXERING>]

<INDEXERING>::=(<DIMENSIE>{,<DIMENSIE>})

<DIMENSIE>::=<N> (ZIE PARAGRAAF 8.10)

<VARIABELE>::=<ALFANUMERIEKE VARIABELE>|<NUMERIEKE VARIABELE>
```

8.8 Definities van numerieke bewerkingen

```
                                     <ALGEBRAISCHE BEWERKING>
                                 ----------------------------------
                                     <RELATIONELE BEWERKING>
<NUMERIEKE BEWERKING>::=         ----------------------------------
                                      <LOGISCHE  BEWERKING>
                                 ----------------------------------
                                 <NUMERIEKE FUNKTIONELE BEWERKING>

<ALGEBRAISCHE BEWERKING>::=+¦-¦*¦/¦^¦\¦MOD

                                  ⎡=[>]⎤
                                < ⎢----⎥
                                  ⎣>[=]⎦
                               -----------
                                  ⎡<[>]⎤
<RELATIONELE BEWERKING>::=      = ⎢----⎥
                                  ⎣>[<]⎦
                               -----------
                                  ⎡<[=]⎤
                                > ⎢----⎥
                                  ⎣=[<]⎦

<LOGISCHE BEWERKING>::=NOT¦AND¦OR¦XOR¦EQV¦IMP

<NUMERIEKE FUNKTIONELE BEWERKING>::=<ALLE BIJ N-FUNKTIES BEHORENDE
                                     SLEUTELWOORDEN (ZIE HFDST. 9)>
```

```
<ALFANUMERIEKE BEWERKING>::=<VOEGBEWERKING>!<RELATIONELE BEWER-
                            KING>!<ALFANUMERIEKE FUNKTIONELE
                            BEWERKING>

<VOEG-BEWERKING>::= +

<ALFANUMERIEKE FUNKTIONELE BEWERKING>::=<ALLE A-FUNKTIE SLEUTEL-
                                        WOORDEN (HFDST. 9)>
```

8.10 Definitie van een numerieke uitdrukking

```
<NUMERIEKE UITDRUKKING>::=<N>

                                  <ALGEBRAISCHE BEWERKING>
                                  ------------------------
      <NUMERIEKE KONSTANTE>                               |
<N>::=------------------------ <N> <RELATIONELE BEWERKING> <N> <NU...
      <NUMERIEKE VARIABELE>       ------------------------
                                    <LOGISCHE BEWERKING>  |

  ...MERIEKE FUNKTIONELE BEWERKING>(<ZIE DE VOORSCHRIFTEN VOOR
  DE BETREFFENDE FUNKTIES IN HOOFDSTUK 9>)!<A><RELATIONELE BE-
  WERKING><A>!(<N>)
```

8.11 Definitie van een alfanumerieke uitdrukking

```
<ALFANUMERIEKE UITDRUKKING>::=<A>

        "<ALFANUMERIEKE KONSTANTE>"
<A>::=--------------------------|<A><VOEG-BEWERKING><A>|<ALFANU...
        <ALFANUMERIEKE VARIABELE>

       ...MERIEKE FUNKTIONELE BEWERKING>(<ZIE VOOR ELKE FUNKTIE DE
       IN HOOFDSTUK 9 OMSCHREVEN SCHRIJFWIJZE>)|(<A>)
```

8.12 Definitie van een uitdrukking

```
<UITDRUKKING>::=<U>

<U>::=<N>|<A>
```

<TYPE 1 INTEGERE KONSTANTE>::=<TEKENLOZE INTEGERE KONSTANTE>

<KOMMANDO>::=<ALLE SLEUTELWOORDEN, IN EEN UITWERKING ZOALS IN HOOFD-
STUK 9 ONDER SCHRIJFWIJZE IS VERMELD>

<RIJ VAN KOMMANDO'S>::=
{ : / ELSE } [<KOMMANDO>] { : / ELSE } { : / ELSE } [<KOMMANDO>]

<DIREKTE OPDRACHT>::=<RIJ VAN KOMMANDO'S>['<ALFANUMERIEKE KONSTANTE>]

<INDIREKTE OPDRACHT>::=<PROGRAMMAREGEL>

<PROGRAMMAREGEL>::=<REGELNUMMER><DIREKTE OPDRACHT>

<REGELNUMMER>::=<TYPE 1 INTEGERE KONSTANTE>

9 DE SLEUTELWOORDEN VAN MSX-BASIC

In dit hoofdstuk worden de sleutelwoorden van het MSX-basic behandeld tot zover deze in de MSX-2-versie door MICROSOFT zijn vrijgegeven.

De sleutelwoorden worden in alfabetische volgorde behandeld.

Indien u de sleutelwoorden één voor één *voor het eerst* wilt leren kennen, is het zeer raadzaam om dit te doen via de aanbevolen leervolgorde. Zie hiervoor hoofdstuk 10.

Per sleutelwoord worden de volgende gegevens verstrekt:

- de naam van het sleutelwoord
- de moeilijkheidsgraad bij het gebruik van dit sleutelwoord. Deze moeilijkheidsgraad variëert van zeer eenvoudig tot zeer moeilijk en is een indicatie voor de beginnende programmeur die kan helpen bij een eerste selectie van de te bestuderen sleutelwoorden
- de soort. We onderscheiden binnen de sleutelwoorden de A-FUNKTIES (functies met alfanumeriek resultaat), de N-FUNKTIES (functies met numeriek resultaat), de SYSTEEMVARIABELEN (door MSX vastgestelde variabelen) en de KOMMANDO'S (sleutelwoorden die een aktie aangeven)
- de afkomst. Alle in MSX-basic voorkomende sleutelwoorden zijn afkomstig uit de Engelse taal. Het is vaak veel gemakkelijker om een sleutelwoord met de bijbehorende betekenis te onthouden wanneer men de preciese afkomst kent. Daarom is van elk sleutelwoord de afkomst naar het Nederlands herleid
- de schrijfwijze. De toegestane syntaxis of schrijfwijze is in BNF opgenomen voor elk sleutelwoord. Zie voor uitleg van de BNF-notatie hoofdstuk 8.
- de betekenis. De functie van elk sleutelwoord wordt behandeld
- voorbeelden. Waar zinvol zijn voorbeelden opgenomen. De voorbeelden zijn waar nodig in verband met het gebruik van andere sleutelwoorden *gebaseerd op de aanbevolen leervolgorde.*

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **ABS is afkorting van absolute value – absolute waarde**

schrijfwijze

```
ABS(N)

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

betekenis

Deze funktie geeft van de tussen haakjes vermelde numerieke uitdrukking de ABSOLUTE WAARDE. De absolute waarde van een getal is de waarde van dit getal, ontdaan van enig teken.

Bijvoorbeeld:
de absolute waarde van 12.437 is 12.347 en de absolute waarde van -133.5 is 133.5

Voorbeeld:

```
PRINT ABS(7*(-5))
 35
Ok
PRINT ABS(111)
 111
Ok
```

moeilijkheidsgraad . . **vrij moeilijk, kennis van de MSX koderingswijze is noodzakelijk**

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

afkomst . . . . . **ASC is afkorting van ASCII – american standard code for information interchange – amerikaanse standaard kodering voor informatieoverdracht**

schrijfwijze

```
ASC(<A>)
```

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

betekenis

Deze funktie geeft de ASCII-waarde van het eerste teken van de tussen haakjes vermelde alfanumerieke uitdrukking als resultaat.

De ASCII-waarde is de waarde die de computer intern in het geheugen gebruikt om allerlei tekens op te slaan. Deze waarde is nooit kleiner dan 0 en nooit groter dan 255.

In hoofdstuk 15 vindt u de aan de ASCII-standaard ontleende MSX-tabel.

Voorbeeld:

```
NEW
Ok
10 LET G$="MSX"
20 PRINT ASC(G$)
RUN
 77
Ok
```

(77 is de ASCII-kodering voor de letter M)

moeilijkheidsgraad . . **vrij moeilijk, kennis van goniometrie is noodzakelijk**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **ATN is afkomstig van arctangent – arctangens**

schrijfwijze

```
ATN(<N>)

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

betekenis

Deze funktie geeft de arctangens van de tussen haakjes vermelde numerieke uitdrukking als resultaat. Het resultaat ligt altijd tussen $-\frac{1}{2}\pi$ en $\frac{1}{2}\pi$ en drukt als zodanig een hoek in radialen uit.

Voorbeeld:

```
NEW
Ok
10 INPUT "WAARDE ";A
20 PRINT "DE ARCTANGENS VAN";A;"IS";ATN(A)
RUN
WAARDE ? 22
DE ARCTANGENS VAN 22 IS 1.5253730473733
Ok
```

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **AUTO is afkomstig van automatic – automatisch**

schrijfwijze

```
AUTO[<REGELNUMMER>][,[<STAPGROOTTE>]]
<REGELNUMMER>::=<TYPE 1 INTEGERE KONSTANTE>
<STAPGROOTTE>::=<TYPE 1 INTEGERE KONSTANTE>
<TYPE 1 INTEGERE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Dit kommando kan bij invoer van lange programma's erg nuttig zijn. Het verstrekt namelijk automatisch regelnummers; alleen de kommando's behoeven nog maar te worden ingetoetst. Voorbeeld:

```
NEW
Ok
AUTO
10 PRINT "AUTO-TEST"
20
```

– Etcetera, de regelnummers worden automatisch gegeven, alleen de kommando's behoeven te worden ingetoetst.

AUTO kan worden onderbroken door CONTROL-STOP of CONTROL-C.

Indien een regel dient te worden overgeslagen, behoeft alleen een RETURN te worden gegeven; een programmaregel met het overgeslagen nummer wordt dan niet in het programma opgenomen.

Indien een programmaregel reeds bestaat, geeft AUTO dat aan door middel van een ster-teken (*). Bijvoorbeeld (uitgaande van het eerste

stukje programma):

```
AUTO
10*PRINT "TWEEDE TEST"
20 REM NOG EEN REGEL
30
```

(Regel 10 was reeds aanwezig en is nu overgetikt.)

Indien regelnummers met sterren worden overgeslagen door alleen RETURN in te toetsen, blijft de oude programmaregel gehandhaafd.

AUTO kan worden vergezeld van een regelnummer en een stapgrootte. Bijvoorbeeld:

`AUTO`	Vanaf regel 10 worden regelnummers met stappen van 10 gegeven.
`AUTO 100`	Vanaf regel 100 worden regelnummers met stappen van 10 gegeven.
`AUTO 5,2`	De regels 5,7,9,11... worden gegeven.
`AUTO ,2`	De regels o,2,4... worden gegeven.

Het AUTO-kommando mag, alhoewel dat niet erg zinvol is, in een programmaregel worden opgenomen. RENUM (zie de behandeling van dit kommando) beschouwt ten onrechte echter de achter AUTO opgenomen waarden als regelnummers en probeert deze bij het hernummeren ook mee te nemen, hetgeen vreemde resultaten kan opleveren.

Merk op dat AUTO standaard onder funktietoets 2 aanwezig is.

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het beeldscherm geheugen alsmede (globaal) de werking van de beeldschermcontrolechip is vereist**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **BASE is basis**

schrijfwijze

```
BASE(<ADRESNUMMER>)

<ADRESNUMMER>::=<N>

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

betekenis

Deze systeemvariabele geeft toegang tot de adressen van verschillende tabellen in het video RAM. Het nummer van de gewenste tabel dient te worden opgenomen tussen de haakjes na het sleutelwoord BASE. Dit nummer mag niet kleiner zijn dan 0 en niet groter zijn dan 44 terwijl een eventuele decimale fractie wordt verwaarloosd.

Alleen systeem-variabelen BASE(0) tot en met BASE(19) kunnen worden veranderd; de overige waarden kunnen alleen worden opgevraagd. In MSX-1 bestaat BASE(20) en verder niet.

De adrestabellen kunnen worden gebruikt voor verder onderzoek in het video RAM maar kunnen ook worden veranderd. Onoordeelkundige verandering van de tabeladressen leidt tot onvoorspelbare resultaten waarbij de computer zelfs vaak moet worden uit- en aangeschakeld om hem weer aan de gang te krijgen. Een voorbeeld:

```
NEW
Ok
10 BASE(5)=0:SCREEN 1
RUN
```

Op het beeldscherm ontstaan vreemde effecten doordat het adres van een tabel werd verminkt. Schakel de computer uit en weer aan om

weer normaal te kunnen werken.

Het sleutelwoord LET is voor BASE niet toegestaan.

Zie voor betekenis van de betreffende tabeladressen met hun bijbehorende tabellen en toegestane waarden hoofdstuk 14.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **BEEP is piepen**

schrijfwijze

```
BEEP
```

betekenis

Dit kommando veroorzaakt een kort piep-signaal, te gebruiken bij bijvoorbeeld een INPUT als attentiesignaal. Voorbeeld:

```
NEW
Ok
10 BEEP:INPUT "UW NAAM ";A$
20 PRINT "GOEDENDAG, ";A$
30 STOP
RUN
UW NAAM ? FREDERIK                    (een attentiesignaal klinkt)
GOEDENDAG, FREDERIK
Break in 30
Ok
```

Indien u een televisieaansluiting heeft, komt het geluid via uw televisieluidspreker. U dient het volume van uw toestel dus wel wat open te draaien.

moeilijkheidsgraad . . **vrij moeilijk, vereist kennis van talstelsels en algemene principes van het computergeheugen**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **BIN$ is afkorting van binary string – binaire (tweetallige) string**

schrijfwijze

```
BIN$(<N>)

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

betekenis

Deze funktie geeft de binaire waarde van de tussen haakjes vermelde uitdrukking weer in string-vorm. De waarde van de numerieke uitdrukking dient te liggen tussen – 32769 en 65536. Indien de waarde van de numerieke uitdrukking gebroken is, wordt een decimale fractie verwaarloosd.
Indien de waarde van de numerieke uitdrukking negatief is, geeft BIN$ een binaire representant, samengesteld volgens de tweecomplementmethode. Bijvoorbeeld:

```
PRINT BIN$(-32768)
1000000000000000
Ok
PRINT BIN$(65535)
1111111111111111
Ok
PRINT BIN$(-1)
1111111111111111
Ok
A$=BIN$(1023)
Ok
PRINT A$
1111111111
Ok
```

moeilijkheidsgraad . . **zeer moeilijk, kennis van machinetaal en computergeheugenopbouw is vaak een vereiste**
soort . . . . . **KOMMANDO**
afkomst . . . . . **BLOAD is afkorting van Binairy LOAD – binair (tweetallig) laden**

schrijfwijze

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

<VERSCHUIVING 1>::=<N>

<VERSCHUIVING 2>::=<N>\R<...>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Het is noodzakelijk dat eerst de behandeling van BSAVE wordt doorgenomen.

Met BLOAD kan een eerder via BSAVE vastgelegde geheugeninhoud weer in het geheugen worden teruggeladen. De toegestane bestandsnamen worden onder het OPEN-kommando behandeld.

Met alleen BLOAD, gevolgd door een bestandsnaam, wordt de geheugeninhoud op precies dezelfde wijze weer ingelezen als deze eerder werd vastgelegd.

Indien de letter R werd gespecificeerd, dan wordt direkt na het laden vervolgd met uitvoering van het machinekommando in het byte dat eerder met BSAVE werd aangegeven.

Indien een verschuiving werd gespecificeerd, dan worden de eerder met BSAVE opgegeven adressen (eerste byte, laatste byte en startadres) verhoogd met de opgegeven verschuiving voordat de geheugeninhoud wordt geladen. Deze verschuiving mag niet kleiner zijn dan

–32768 en niet groter dan 65535. Een eventuele decimale fraktie wordt verwaarloosd. Indien de verschuiving negatief is, wordt voorafgaand aan de uitvoering de waarde 65536 hierbij opgeteld.

Indien door de opgegeven verschuiving één van de adressen de waarde 65535 overschrijden, dan wordt dit adres met de waarde 65536 verlaagd.

Indien door een opgegeven opschuiving een gedeelte van de geheugeninhoud over de grens van 65535 zou worden ingelezen, dan wordt dit gedeelte verder vanaf geheugenlokatie 0 ingelezen.

Voorbeeld (zie voorbeeld BSAVE):

```
BLOAD "CAS:MSX"
Found:MSX
Ok
```

Band terugspoelen en recorder op afspelen zetten. Het laden duurt ongeveer vijf minuten.

De eerder in het voorbeeld van BSAVE op band gezette ROM-inhoud wordt weer ingelezen. Praktisch heeft dit voorbeeld geen zin daar het ROM-geheugen niet kan worden veranderd.

Het gebruik van BLOAD is alleen zinvol indien de opbouw van het geheugen van een MSX-computer tot in details bekend is. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek gaan we hierop niet verder in.

N.B.: BLOAD kan niet vanaf de RAM-disk ("MEM:...") geschieden (MSX-2).

moeilijkheidsgraad . . zeer moeilijk, kennis van machinetaal en computergeheugen opbouw is vereist
soort . . . . . KOMMANDO
afkomst . . . . . BSAVE is afkorting van binary save – binair (tweetallig) veiligstellen

schrijfwijze

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

<BESTANDSNAAM>::=<A>

<EERSTE BYTE>::=<N>

<LAATSTE BYTE>::=<N>

<STARTADRES>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met het BSAVE-kommando kunnen delen van het computergeheugen direkt op een extern medium worden opgeslagen. De toegestane bestandsnamen worden bij behandeling van het OPEN-kommando verder omschreven.

Welk gedeelte van het geheugen dient te worden vastgelegd, blijkt uit twee waarden die na de bestandsnaam dienen te worden opgenomen: het adres van het eerste vast te leggen byte en het adres van het laatste vast te leggen byte. Deze twee adressen mogen niet kleiner zijn dan –32768 en niet groter zijn dan 65535. Indien een adres negatief is, wordt de waarde 65536 voor uitvoering bij dit adres opgeteld. Een eventuele decimale fractie wordt verwaarloosd.

Een voorbeeld: in het onderstaande programma wordt de inhoud van het basis ROM, dus in feite het machinekode-programma dat er voor zorgt dat we in MSX-basic kunnen programmeren, op cassetteband vastgelegd.

```
NEW
Ok
10 BSAVE "CAS:MSX",0,32767
RUN                          (eerst cassetterecorder op opnemen zetten)
Ok                                        (duur: ongeveer 5 minuten)
```

Indien een startadres is opgegeven, dan wordt dat mee opgeslagen. Wanneer (zie BLOAD) het geheugendeel dan later met de R-optie wordt geladen, dan vervolgt de computer onmiddellijk na het laden met de uitvoering van het machinekodebevel dat in het byte op het start-adres is gekodeerd.

Dit startadres mag niet kleiner zijn dan –32768 en niet groter dan 65535. Indien het startadres kleiner is dan nul, wordt er voor uitvoering automatisch eerst de waarde 65536 bij opgeteld. Een eventuele decimale fractie wordt verwaarloosd.

Het gebruik van BSAVE kan alleen zinvol worden gedaan indien de opbouw van het geheugen van een MSX-computer tot in details bekend is. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek gaan we daar niet verder op in.

N.B.: BSAVE kan niet naar de RAM-disk ("MEM:...") geschieden (MSX-2).

moeilijkheidsgraad . . . . . **normaal, hangt sterk van het gebruik af**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CALL is (aan)roepen**

schrijfwijze

```
      CALL
---------------- {"}<NAAM KOMMANDO>{"}[(<U>{,<U>})]
<ONDERSTREPING>

<NAAM KOMMANDO>::=<A>

<ONDERSTREPING>::=_

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met het CALL-kommando is het mogelijk om aan MSX toegevoegde kommando's aan te spreken. Deze kommando's zijn vaak in een apart stuk ROM-geheugen opgeslagen dat apart, bijvoorbeeld via de daartoe dienende sleuf in de computer, dient te zijn aangesloten. De funkties van deze sleutelwoorden alsmede de te specificeren variabelen hangen natuurlijk af van het aangesloten randapparaat en dienen in de bijgaande gebruiksaanwijzing te zijn verklaard.

Enkele CALL's behoren tot de MSX-2 standaard. Zij worden direkt hierna behandeld.

CALL mag door _ (onderstreping) woden vervangen.

moeilijkheidsgraad . . . . . eenvoudig
soort . . . . . KOMMANDO (toepassing van CALL)
afkomst . . . . . zie CALL. MEMINI staat voor MEMory INItialisation – geheugen-initialisatie

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
      CALL
--------------- ("}MEMINI("}[(RAMDISK-GROOTTE>)]
<ONDERSTREPING>

<ONDERSTREPING>::=_

<RAMDISK-GROOTTE>::=<N>

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

betekenis

MSX-2 kent de zogenaamde RAM-DISK. De onder BASIC gewoonlijk ongebruikt blijvende 32 kilobytes aan RAM-geheugen die zich standaard minimaal in de onderste twee geheugenbanken bevindt, kan met behulp van dit verschijnsel onder BASIC als een soort schijvengeheugen worden gebruikt.

Natuurlijk is de RAM-DISK, net als het gewone geheugen, gevoelig voor stroomuitval. Nadat de spanning werd uitgeschakeld en weer ingeschakeld, zijn de gegevens van de RAM-DISK verdwenen. De RAM-DISK kan dus alleen dienen voor tijdelijke gegevensopslag.

De RAM-DISK kan ongeveer als een cassetterecorder worden gebruikt met dit verschil, dat het laden en opbergen van programma's en andere gegevens vele malen zo snel gaat. Bovendien is de foutkans met betrekking tot de RAM-DISK minimaal, dit in tegenstelling tot het gebruik van de cassetterecorder.

Voordat de RAM-DISK in gebruik wordt genomen, dient deze te wor-

den ingesteld met het kommando CALL MEMINI. Met dit kommando maakt men de RAM-DISK schoon en kan men opgeven, welk gedeelte van het totale blok van 32 kb RAM als RAM-DISK moet worden gebruikt.

Daar de RAM-DISK 768 bytes nodig heeft voor opslag van de inhoudsopgave, zal er op de RAM-DISK altijd minimaal 768 bytes minder beschikbaar zijn als werd opgegeven. En omdat er altijd gehele blokken van 256 bytes tegelijk worden bezet voor de RAM-DISK, zal de door u opgegeven waarde naar beneden worden afgerond tot een veelvoud van 256 nadat er eerst de waarde 1 bij werd opgeteld.

Wanneer achter CALL MEMINI geen RAM-DISK grootte wordt opgegeven, wordt automatisch een zo groot mogelijke RAM-DISK toegewezen. In dat geval kunt u op de RAM-DISK precies 32000 bytes aan gegevens kwijt.

De RAM-DISK grootte dient minimaal gelijk te zijn aan 1023 en mag maximaal gelijk zijn aan 32767. Wanneer u een waarde, kleiner dan 1023 in het kommando opneemt, dan resulteert dit automatisch in het geheel afkoppelen van de RAM-DISK onder de melding "No RAM disk".

In het volgende voorbeeld wordt een kleine RAM-DISK geïnstalleerd:

```
CALL MEMINI(2048)
1280 bytes allocated
Ok
```

N.B.: net zoals de cassetterecorder de randapparatuur-aanduiding CAS: heeft, heeft de RAM-DISK de randapparatuur-aanduiding MEM:. Zie in dit verband ook het OPEN-kommando.
Een programma kan bijvoorbeeld met een SAVE "MEM:..." worden opgeslagen op de RAM-DISK waarbij op ... de naam van het programma moet worden ingevuld.

Noot voor de schijveneenheid-bezitters: in tegenstelling tot wat u met bestanden op schijf gewend bent, zijn bestanden op RAM-DISK alleen sequentiëel benaderbaar. Random benadering is alleen op schijf mogelijk.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO (toepassing van CALL)**
afkomst . . . . . **zie CALL. MFILES staat voor Memory FILES – geheugenbestanden (op RAM-DISK)**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
      CALL
--------------- (")MFILES(")
<ONDERSTREPING>

<ONDERSTREPING>::=_
```

Betekenis

Zie eerst de behandeling van CALL MEMINI.

CALL MFILES biedt u de mogelijkheid om zichbaar te maken wat de inhoud van de RAM-DISK is. Na uitvoering van dit kommando worden op het beeldscherm de namen van de op de RAM-DISK aanwezige bestanden (maximaal 32 stuks) geprojekteerd waarna wordt aangegeven hoeveel bytes er op de RAM-DISK nog vrij zijn.

In het volgende voorbeeld wordt een klein programma op de RAM-DISK opgeslagen waarna de RAM-DISK met het kommando CALL MFILES wordt onderzocht.

```
CALL MEMINI
32000 bytes allocated
Ok
NEW
Ok
10 REM TESTPROGRAMMA
SAVE "MEM:TEST"
Ok
CALL MFILES
TEST
31744 bytes free
Ok
```

moeilijkheidsgraad . . . . . eenvoudig
soort . . . . . KOMMANDO (toepassing van CALL)
afkomst . . . . . zie CALL. MKILL staat voor MEMory KILL – verwijder een geheugenbestand (van de RAM-DISK)

Dit kommando is alleen onder MSX-2 beschikbaar.

schijfwijze

```
      CALL
--------------- {"}MKILL{"}(<BESTANDSNAAM>)
<ONDERSTREPING>

<BESTANDSNAAM>::=<A>

<ONDERSTREPING>::=_

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

betekenis

Zie eerst de behandeling van CALL MEMINI en CALL MFILES.

Met CALL MKILL kunt u een bestand van de RAM-DISK verwijderen. Hiertoe dient u tussen haakjes de naam van het te verwijderen bestand achter het kommando CALL MKILL op te nemen.

Wanneer het bestand niet op de RAM-DISK aanwezig is, volgt de foutmelding "FILE not found". In het andere geval wordt het bestand verwijderd en wordt de vrijgekomen ruimte weer aan de RAM-DISK toegevoegd.

Met CALL MKILL kan slechts één bestand tegelijk van RAM-DISK worden verwijderd.

In het volgende voorbeeld wordt eerst de RAM-DISK geïnstalleerd. Hierna wordt er een kort programma op gezet. Nadat met behulp van CALL MFILES is gecontroleerd of het programma daadwerkelijk op de RAM-DISK aanwezig is, wordt het programma er met behulp van CALL MKILL weer afgehaald. Uiteindelijk wordt met CALL MFILES

gecontroleerd of het programma ook daadwerkelijk is verdwenen.

```
CALL MEMINI
32000 bytes allocated
Ok
NEW
Ok
10 REM TESTPROGRAMMA
SAVE "MEM:TEST"
Ok
CALL MFILES
TEST
31744 bytes free
Ok
CALL MKILL("TEST")
Ok
CALL MFILES
File not found
Ok
```

moeilijkheidsgraad . . . . . eenvoudig
soort . . . . . KOMMANDO (toepassing van CALL)
afkomst . . . . . zie CALL. NMAME staat voor Memory reNAME – het voorzien van een in het geheugen (op RAM-DISK) opgenomen bestand van een andere naam

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
     CALL
--------------- {"}MNAME{"}(<OUDE NAAM>AS<NIEUWE NAAM>)
<ONDERSTREPING>

<OUDE NAAM>:::=<BESTANDSNAAM>

<NIEUWE NAAM>:::=<BESTANDSNAAM>

<BESTANDSNAAM>:::=<A>

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

<ONDERSTREPING>:::=_
```

betekenis

Zie eerst de behandeling van CALL MEMINI, CALL MFILES en CALL MKILL.

Met CALL MNAME kunt u een op de RAM-DISK aanwezig bestand een andere naam geven. Hiertoe dient u achter het kommando CALL MNAME tussen haakjes de naam te vermelden van het bestand gevolgd door het sleutelwoord AS en de nieuwe naam van dit bestand.

Wanneer er reeds een bestand onder de door u opgegeven nieuwe naam aanwezig is op RAM-DISK, geeft de computer u de melding "File already exists" en wordt de naamsverandering niet uitgevoerd.

In het volgende voorbeeld wordt eerst de RAM-DISK geïnstalleerd. Hierna wordt er een klein programma op de RAM-DISK geplaatst onder de naam TEST. Met CALL MFILES wordt gecontroleerd of het

programma ook daadwerkelijk onder deze naam op de RAM-DISK aanwezig is. Daarna wordt de naam van de file TEST veranderd in PIETJE en wordt wederom gecontroleerd of dit correct is verlopen.

```
CALL MEMINI
32000 bytes allocated
Ok
NEW
Ok
10 REM TESTPROGRAMMA
SAVE "MEM:TEST"
Ok
CALL MFILES
TEST
31744 bytes free
Ok
CALL MNAME("TEST" AS "PIETJE")
Ok
CALL MFILES
PIETJE
31744 bytes free
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **CDBL is afkorting van convert to double precision value – zet over naar dubbele precisie waarde**

schrijfwijze

```
CDBL(<N>)

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

betekenis

Deze funktie geeft als resultaat de waarde van de tussen haakjes vermelde numerieke uitdrukking nadat deze tot dubbele precisie is herleid.

Omdat MSX-basic standaard met dubbele precisie rekent, heeft deze funktie geen zin.

moeilijkheidsgraad . . vrij moeilijk, kennis van de MSX-koderingswijze is noodzakelijk
soort . . . . . A-FUNKTIE
afkomst . . . . . CHR$ is afkorting van character string – karakter string

schrijfwijze

```
CHR$(<ASCII-KODE>)

<ASCII-KODE>::=<N>

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

betekenis

Deze funktie geeft als resultaat de letter die korrespondeert met de gegeven ASCII-kode. De ASCII-kode is de interne kode die de computer gebruikt om allerlei tekens in het geheugen op te slaan en moet minimaal gelijk zijn aan nul en maximaal gelijk zijn aan 255. Een eventuele decimale fractie wordt verwaarloosd.

In hoofdstuk 15 vindt u de aan de ASCII-Standaard ontleende MSX-tabel.

Voorbeeld:

```
NEW
Ok
10 FOR I=65 TO 90
20 PRINT CHR$(I);
30 NEXT I
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
```

(de ASCII-kodes 65 ... 90 corresponderen met de letters A ... Z).

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **CINT is afkorting van convert to integer value – zet over naar integere waarde**

schrijfwijze

```
CINT(<N>)

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

betekenis

Deze funktie geeft als resultaat de waarde van de tussen haakjes vermelde numerieke uitdrukking nadat deze geconverteerd is naar een integere waarde. Hiertoe wordt een decimale fractie verwaarloosd. Indien de resulterende waarde groter is dan 32767 of kleiner is dan –32768, volgt een foutmelding; de waarde kan dan niet tot een integere waarde worden herleid. Voorbeeld:

```
NEW
Ok
10 PRINT CINT(12.4)
20 PRINT CINT(4/3)
30 PRINT CINT(120000)
RUN
 12
 1
Overflow in 30
Ok
```

SLEUTELWOORD CIRCLE

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CIRCLE is cirkel**

schrijfwijze

```
CIRCLE<LOCATIE>,<STRAAL>[,[<KLEUR>][,[<AANVANGSHOEK>][,[<EIND...
      ...HOEK>][,<AFPLATTING>]]]]\CIRCLE<...>,

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<STRAAL>::=<N>

<KLEUR>::=<N>

<AANVANGSHOEK>::=<N>

<EINDHOEK>::=<N>

<AFPLATTING>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met dit kommando kunnen cirkels, ellipsen, cirkeldelen en ellipsdelen worden getekend. Voor goed begrip van dit kommando is het vereist dat de behandeling van het PSET-kommando terdege is bestudeerd.

In de meest eenvoudige vorm kunnen we een cirkel laten tekenen door de locatie van het middelpunt van de cirkel en een straal te specificeren. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 CIRCLE (111,111),50
30 GOTO 30
RUN
```

Een cirkel wordt getekend met het middelpunt op (111,111) en een straal van 50 puntjes. In feite wordt geen zuivere cirkel maar een ellips getekend. Dit is het gevolg van het feit dat de puntjes vertikaal 'dichter op elkaar' zitten dan horizontaal.

De straal dient groter dan of gelijk aan nul te zijn. Daarbij mag de straal niet groter zijn dan 32767. Indien de straal bestaat uit een gebroken waarde, dan wordt deze waarde eerst ontdaan van de decimale fractie.
Een foutje in MSX-basic draagt er zorg voor dan een negatieve straal, mits groter dan of gelijk aan –32768, wel wordt geaccepteerd. De computer tekent in zo'n geval de cirkel echter niet; in plaats hiervan blijft de computer op het CIRCLE-kommando 'hangen' en kan *alleen met CONTROL-STOP* worden onderbroken.

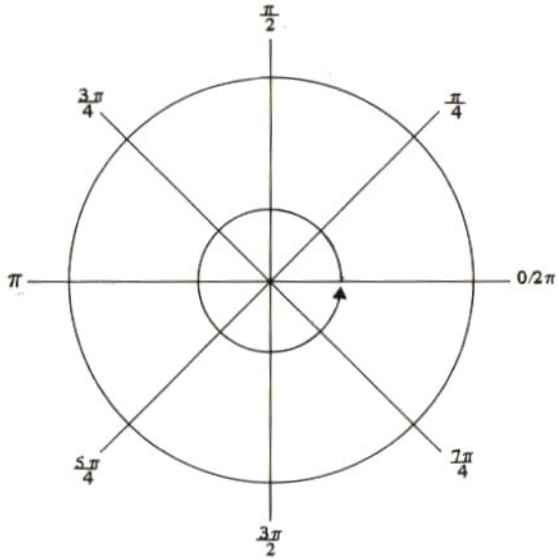
In plaats van de actieve voorgrondkleur kunnen we een andere voorgrondkleur specificeren. Bijvoorbeeld:

```
NEW
Ok
10 COLOR ,1,1
20 SCREEN 2
30 CIRCLE (111,111),50,12
40 GOTO 40
RUN
```

Een donkergroene cirkel wordt op een zwarte achtergrond getekend. Onder MSX-2 geldt dit alleen indien het standaard palet geactiveerd is. In MSX-2 biedt de kleurkodering vele mogelijkheden. Deze worden onder COLOR behandeld. Het CIRCLE-kommando mag alle voor het scherm toepasbare kleurinstellingen bevatten; zie hiervoor het SCREEN-kommando.

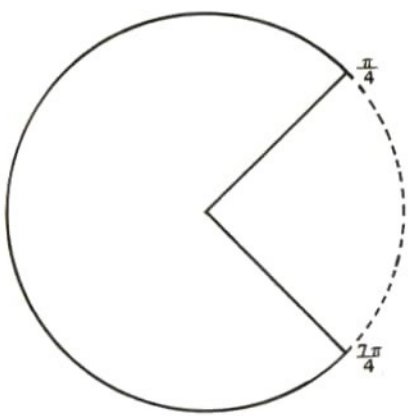
Een aanvangs- en een eindhoek kan worden gespecificeerd. De cirkel wordt dan gedeeltelijk getekend. Om precies te begrijpen welk deel van de cirkel gaat worden getekend, is een klein stukje goniometrie noodzakelijk.

De begin- en eindhoek dienen in radialen te worden opgegeven. De betekenis van de beginhoek en de eindhoek moge blijken uit het volgende schema:



$\pi = 3.141592653589793\ldots$

Om het volgende cirkeldeel te tekenen:



dient een beginhoek van $\frac{\pi}{4}$ en een eindhoek van $\frac{7\pi}{4}$ te worden opgenomen. $\frac{\pi}{4}$ is ongeveer gelijk aan 0.7854 en $\frac{7\pi}{4}$ is ongeveer gelijk aan 5.4978. Om dit cirkeldeel van een bepaalde cirkel te tekenen, kunnen we dus programmeren:

```
NEW
Ok
10 SCREEN 2
20 CIRCLE (111,111),50,,,.7854,5.4978
30 GOTO 30
```

Het betreffende cirkeldeel wordt getekend.

Indien we verbindingen naar het middelpunt willen tekenen, dienen we de begin- en eindhoek negatief op te geven. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 CIRCLE (111,111),50,,,-.7854,-5.4978
30 GOTO 30
RUN
```

Het cirkeldeel verschijnt; de verbindingen met het middelpunt zijn aangebracht.

Een kleine fout in MSX-basic komt nu aan het licht; als u de tekening goed bekijkt zal blijken dat de verbinding naar het middelpunt soms uit twee vlak naast elkaar lopende lijnen bestaat.

Probeer het bovenstaande voorbeeld ook eens met een SCREEN 3 kommando; precies dezelfde cirkel wordt, zij het in grovere punten, getekend.

In MSX-2 zijn er véél meer schermindelingen mogelijk. Zie hiervoor de behandeling van SCREEN. Probeer het voorbeeld bijvoorbeeld ook eens met de SCREEN 6 instelling.

Tenslotte kunnen we ook nog een afplattingsfaktor definiëren. Door deze faktor kunnen we in plaats van een cirkel een ellips tekenen. Indien de afplattingsfaktor groter is dan de waarde 1, dan wordt in vertikale richting de straal op de juiste maat gehouden en wordt de straal in horizontale richting gedeeld door deze afplattingsfaktor. Het

resultaat is dus een vertikaal gerichte ellips.
Indien de afplattingsfaktor kleiner is dan de waarde 1, dan wordt de straal in horizontale richting op de juiste maat gehouden en wordt de vertikale straal vermenigvuldigd met de afplattingsfaktor. Het resultaat is dus een horizontaal gerichte ellips. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 FOR P=.2 TO 5 STEP .2
30 CIRCLE(111,111),50,,,,P
40 NEXT P
50 GOTO 50
RUN
```

Op het beeldscherm verschijnt een samenspel van ellipsen.

N.B.: De afplattingsfaktor moet altijd groter zijn dan nul.

De cirkel behoeft niet volledig binnen het beeldschermbereik te liggen; hij mag zelfs volledig buiten het beeldschermbereik worden getekend. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 CIRCLE (-111,-111),200
30 GOTO 30
RUN
```

Een klein gedeelte van de cirkel, het gedeelte dat binnen het beeldschermbereik valt, wordt afgebeeld.

Uit het volgende voorbeeld blijkt dat indien een ellips of cirkel(deel) wordt getekend, de computer aanneemt dat het laatst getekende punt wordt gevormd door het middelpunt:

```
NEW
Ok
10 SCREEN 2
20 CIRCLE (111,111),50
30 LINE -STEP(100,100)
40 GOTO 40
RUN
```

Een cirkel met een vanuit het middelpunt naar rechtsonder lopende lijn is het resultaat. Deze lijn vangt aan in het laatst getekende punt; in dit geval het middelpunt (alhoewel dit niet daadwerkelijk is getekend).

Tot slot volgt hier een combinatievoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 X=RND(1)*512-256
30 Y=RND(1)*384-192
40 R=RND(1)*128
50 S=RND(1)*-6.28
60 E=RND(1)*-6.28
70 C=RND(1)*15+1
80 P=.01+10*RND(1)
90 CIRCLE (X,Y),R,C,S,E,P
100 GOTO 20
RUN
```

Het beeldscherm wordt volgeplaatst met cirkel- en ellipssegmenten in diverse kleuren. Vele figuren vallen geheel of gedeeltelijk buiten het beeldscherm. Merk op dat een kleine fout in MSX-basic veroorzaakt dat geheel boven in het beeld regelmatig een 'afkappingslijn' wordt getekend van een gedeeltelijk buiten het beeld vallende cirkel.

SLEUTELWOORD # CLEAR

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CLEAR is schoonmaken**

schrijfwijze

```
CLEAR[<STRINGRUIMTE>[,<BOVENGRENS BASICGEHEUGEN>]]

<STRINGRUIMTE>::=<N>

<BOVENGRENS BASICGEHEUGEN>::=<N>

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

betekenis

Eenvoudige betekenis

Met het enkele CLEAR-kommando worden de volgende acties ondernomen:

- alle eventueel nog openstaande kanalen worden gesloten (zelfde funktie als het CLOSE-kommando; zie aldaar)
- alle variabelen worden gewist
- alle array-variabelen worden van hun bijbehorende dimensies ontdaan (het DIM-kommando wordt als het ware voor elke variabele ongedaan gemaakt)

Bijvoorbeeld:

```
NEW
Ok
10 LET A$="DIT IS EEN TEST"
20 DIM B(20,20)
30 LET B(19,19)=22
40 CLEAR
50 PRINT A$;B(19,19)
RUN
Subscript out of range in 50
Ok
```

Uit bovenstaand programma blijkt dat de dimensionering voor variabele B is tenietgedaan. Hierdoor volgt een foutmelding. Tevens blijkt dat variabele A$ leeg is na uitvoering van CLEAR.

Gevorderd gebruik

Achter het CLEAR-kommando kunnen twee gegevens worden gespecificeerd. Om deze gegevens te vergelijken is het raadzaam om de geheugenindelingstabellen van hoofdstuk 1.3 nog eens te bestuderen.
Met het eerste gegeven kunnen we de maximaal door MSX te gebruiken string-ruimte bepalen. Indien we deze bepaling niet stellen, wordt een standaard grootte van 200 karakters aangenomen. Het onderstaande programma loopt bij een zojuist opgestarte MSX-computer fout:

```
NEW
Ok
10 FOR I=1 TO 255:A$=A$+"*":NEXT
RUN
Out of string space in 10
Ok
```

Deze foutmelding volgt uit het feit dat er standaard slechts een stringgebied van 200 karakters is toegewezen. Na intoetsen van:

```
CLEAR 1000:RUN
```

zal het programma nu wel tot een goed einde komen; er werd een string-gebied van 1000 karakters toegewezen.

Belangrijk is het om te weten dat MSX-basic bij het uitwerken van alfanumerieke uitdrukkingen een bepaalde werkruimte nodig heeft. Deze werkruimte is net zo groot als de grootste stringlengte die tijdens de uitwerking ontstaat. Hierdoor kan de bovengenoemde foutmelding eerder dan verwacht optreden. In het laatste voorbeeld trad deze reeds op bij de stringlengte van 100 tekens voor A$.

Met het tweede gegeven achter CLEAR kunnen we het laatste door MSX-basic te gebruiken geheugenadres bepalen. Hierdoor (zie geheugentabel) kan een vrije geheugenruimte worden gecreëerd waarin de specialist of zéér ver gevorderde amateur zijn of haar machinekodeprogrammatuur kan opslaan. Bijvoorbeeld:

```
CLEAR 200,50000
```

bepaalt dat het MSX-basic niet verder mag gaan in geheugengebruik dan geheugenadres 50000 en dat de stringruimte maximaal 200 karakters kan bevatten.

N.B.: Natuurlijk blijft MSX het systeemgedeelte (vanaf F380 hexadecimaal of 62336 decimaal) gebruiken. De werkelijke vrije geheugenruimte ligt dus tussen het met CLEAR opgegeven adres en adres 62336. Bij gebruik van schijveneenheden worden direkt onder het systeemgedeelte nog buffers toegewezen waardoor het bruikbare geheugengedeelte nog kleiner wordt.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CLOAD is afkorting van cassette load – laden van cassetteband**

schrijfwijze

CLOAD [PRINT / ?] [?][<BESTANDSNAAM>][,<...>]

<BESTANDSNAAM>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Zie eerst de behandeling van het CSAVE-kommando

Met het CLOAD-kommando kunnen programma's die met CSAVE op cassetteband zijn vastgelegd, weer worden ingelezen in het computergeheugen. Indien het vraagteken is opgenomen, wordt een programma niet van cassette geladen doch wordt slechts gecontroleerd of een programma op band overeen komt met het in het computergeheugen opgeslagen programma.
Indien geen programmanaam wordt gespecificeerd, wordt het eerst op de cassetteband voorkomende programma ingelezen. Indien wel een programmanaam is gespecificeerd, dan wordt de band afgezocht naar het juiste programma en wordt alleen dit programma in behandeling genomen. Bijvoorbeeld:

```
NEW
Ok
10 REM ********************
20 REM TESTPROGRAMMA
30 REM ********************
40 STOP
```

(nu eerst een cassetteband plaatsen en de cassetterecorder op opnemen zetten)

```
CSAVE "TEST",2
```

(met een schrijfsnelheid van 2400 baud wordt het programma op cassetteband geschreven)

(nu de cassetterecorder stoppen)

```
CLOAD? "TEST"
```

(nu de cassette terugspoelen en de recorder op afspelen zetten)

```
Found:TEST
Ok
```

(het programma is *vergeleken* en goed bevonden. Indien de vergelijking fout gaat, wordt de foutmelding Verify error gegeven)

(nu de computer uitschakelen en weer inschakelen; het programma is gegarandeerd uit het geheugen verdwenen.)

```
CLOAD
```

(nu de band terugspoelen en de recorder op afspelen zetten)

```
Found:TEST
Ok
LIST
10 REM **********************
20 REM TESTPROGRAMMA
30 REM **********************
40 STOP
Ok
```

(het programma is weer ingelezen)

Merk op dat bij de laatste keer geen programmanaam werd opgegeven; het eerste op de band voorkomende programma werd geladen.

Het is zéér raadzaam om *altijd* een met CSAVE vastgelegd programma te controleren op juistheid met een CLOAD?-kommando, direkt daarna uitgevoerd. De cassetteband is geen erg betrouwbaar opslagmedium; een eventuele 'drop-out' op een cassetteband kan vele uren programmeerwerk te niet doen!

Indien een programma dient te worden ingelezen terwijl we niet zeker weten of het het eerste programma op de band is, dienen we een programmanaam op te geven teneinde zekerheid te hebben dat het juiste programma wordt geladen. Stel voor dat op een cassetteband twee programma's TEST1 en TEST2 achter elkaar staan en we TEST2

willen laden. We kunnen dan de band terugspelen, de recorder op afspelen zetten en dan ingeven:

```
CLOAD "TEST2"
Skip:TEST1
Found:TEST2
Ok
```

Bij SKIP (= overslaan) vermeldt de computer dat hij een programma wel heeft gevonden maar niet inleest. Bij FOUND (gevonden) geeft de computer aan dat hij inleest.

MSX schrijft geen controle voor op de integriteit (betrouwbaarheid) van ingelezen gegevens. Een verkeerd ingesteld volume of een kwalitatief slechte apparatuur kan leiden tot verminkte programma's bij inlezen zonder dat daarvan tijdens het inlezen melding wordt gemaakt. Zet het volume bij afspelen altijd op ongeveer 80 procent en kies bij de CSAVE voor de laagste schrijfsnelheid indien u de apparatuur niet geheel vertrouwt.

Een CLOAD behoeft geen snelheidsindicatie te bevatten zoals een CSAVE. De CLOAD bepaalt de snelheid waarmee werd geschreven automatisch en leest ook weer op deze snelheid in.

Het CLOAD-kommando, opgenomen in een programmaregel, heeft tot gevolg dat het betreffende programma eerst wordt ingelezen en dat daarna de Ok-melding verschijnt (het programma loopt dus niet door). De SKIP- of FOUND-melding wordt dan echter niet gegeven.

N.B.: Het schrijven naar of laden van band van een groot programma kan tot verscheidene minuten duren.

Merk op dat het CLOAD-kommando standaard onder funktietoets 7 aanwezig is.

CLOAD? geeft, opgenomen in een programma, altijd een verify error bij uitvoering.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CLOSE is sluiten/dichtmaken**

schrijfwijze

```
CLOSE[[#]<KANAAL>{,<KANAAL>}]

<KANAAL>::=<N>

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

betekenis

Dit kommando wordt bij het OPEN-kommando reeds behandeld; zie aldaar.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CLS is afkorting van clear screen – scherm schoonmaken**

schrijfwijze

```
CLS
```

betekenis

Uitvoering van dit kommando resulteert in het geheel schoonvegen van het beeldscherm. Ook indien een grafisch beeldscherm geactiveerd is, zal dit volledig worden schoongemaakt. De grafische pointer (het 'laatst getekende punt') wordt dan niet veranderd.
De cursor wordt indien aanwezig, na het schoonmaken van het beeldscherm op de linker bovenpositie van het beeldscherm geplaatst. Voorbeeld:

```
NEW
Ok
10 INPUT "HOE HEET U ";NAAM$
20 CLS
30 PRINT "HALLO ";NAAM$
40 STOP
RUN
```

– Bij het uitvoeren van dit programma vraagt de computer eerst om uw naam. Daarna wordt het beeldscherm schoongemaakt en wordt de tekst HALLO, gevolgd door uw naam, afgedrukt.

moeilijkheidsgraad . . . . . **eenvoudig tot vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **COLOR is kleur**

schrijfwijze

```
COLOR[<VOORGROND>][,[<ACHTERGROND>][,<RAND>]]\<...>,

<VOORGROND>::=<N>

<ACHTERGROND>::=<N>

<RAND>::=<N>

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

betekenis MSX-1 en MSX-2 (SCREEN 0...3)

Met dit kommando kan de kleurinstelling van het beeldscherm worden bepaald. Een voorgrondkleur (de kleur van de letters of tekening), een achtergrondkleur (de kleur van het scherm) en een randkleur (de kleur van de boven- en onderrand van het beeldscherm) kan worden gespecificeerd. Een niet gespecificeerde kleur blijft onveranderd.

De volgende kleuren kunnen worden gekodeerd:

kleurnummer	kleur of effect
0	doorschijnend
1	zwart
2	groen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	cyaan (een soort blauw)
8	rood
9	lichtrood
10	donkergeel
11	lichtgeel
12	donkergroen
13	magenta (een soort paars)
14	grijs
15	wit

Kleurkode 0 (transparant) resulteert in zwart indien toegepast als randkleur. Toegepast als achtergrondkleur resulteert kleurkode 0 in het overnemen van de randkleur (SCREEN 1, 2 en 3) of in de kleur zwart (SCREEN 0).

Toegepast als voorgrondkleur, resulteert kleurkode 0 in het overnemen van de randkleur (SCREEN 1, 2 of 3) of in het overnemen van de achtergrondkleur (SCREEN 0). Het begrip 'transparant' gaat voor deze kleurkode dus maar ten dele op.

Onder SCREEN 0 is er geen randkleur van toepassing.

`SCREEN 1`	eerst scherm 1 instellen (zie SCREEN)
`COLOR 3,1,2`	de voorgrondkleur werd lichtgroen gekozen. De achtergrond is zwart en de rand is groen
`COLOR ,,14`	alleen de randkleur is naar grijs veranderd
`COLOR 0`	de voorgrond is nu doorschijnend en dus onzichtbaar
`COLOR ,0,0`	de achtergrond en rand zijn zwart (doorschijnend) gekleurd; de voorgrondkleur is ongewijzigd
`COLOR 15,4,4`	de standaard kleurinstelling (wit op donkerblauw, rand ook donkerblauw). Deze funktie is standaard onder funktietoets 6 aanwezig.

Merk op dat het COLOR sleutelwoord onder funktietoets nummer 1 aanwezig is.

Indien de numerieke uitdrukkingen die de kleuren aangeven, gebroken waarden bevatten, worden deze waarden ontdaan van de decimale fractie.
De kleurwaarden mogen niet kleiner zijn dan 0 en niet groter dan 15.

Indien het COLOR-statement vóór de SCREEN-instelling wordt opgenomen, zijn alle daarin vermelde kleuren onmiddellijk actief. Onder MSX-2 is dit echter niet altijd mogelijk en onder MSX-1 is dit niet altijd wenselijk.

Hieronder volgt een tabel waarin het kleurgedrag van een MSX-computer wordt uitgelegd wanneer het COLOR-statement ná de SCREEN-instelling is opgenomen:

	SCREEN 0 en SCREEN 1	andere SCREEN-instellingen
voorgrond-	alle geprojekteerde gegevens worden onmiddellijk naar de nieuwe kleur aangepast.	alleen de hierna geprojekteerde gegevens worden in de nieuwe kleur uitgevoerd.
achtergrond-kleur	de achtergrond wordt onmiddellijk in de nieuwe kleur uitgevoerd.	pas na een CLS-kommando is de nieuwe achtergrondkleur actief.
randkleur	(alleen SCREEN 1) is altijd onmiddellijk actief.	is altijd onmiddellijk actief.

MSX-2 betekenis

Onder MSX-2 gelden in eerste instantie dezelfde zaken die voor MSX-1 zijn beschreven. Echter, het COLOR-kommando is onder MSX-2 veel uitgebreider. Het is raadzaam om eerst het SCREEN-kommando een eerste keer door te nemen.

Onder scherminstellingen SCREEN 0 t/m SCREEN 5 en SCREEN 7 kan de kleurkodering variëren van 0 t/m 15. Onder scherminstelling SCREEN 6 kan de kleurkodering gelijk zijn aan 0, 1, 2 of 3.

De kleurkodering die onder SCREEN 0 tot en met SCREEN 7 kan worden opgegeven, verwijst naar zogenaamde PALETTES. Onder MSX-2 is een PALETTE een bepaalde menging van de kleuren rood, blauw en groen. Door deze kleuren op de juiste wijze te mengen, kunnen vele natuurlijke kleurschakeringen worden benaderd. Bedenk hierbij dat we het hebben over het mengen van licht en niet over het mengen van kleurstoffen. Lichtmenging is additioneel; de kleuren worden als het ware bij elkaar opgeteld. Hierdoor kan bij een bepaalde menging van rood, groen en blauw een indruk van wit licht worden verkregen. Volgens dit principe werkt bijvoorbeeld ook een kleurentelevisie.

Binnen een PALETTE kunnen de kleuren rood, groen en blauw in acht schakeringen met elkaar worden gemengd waarbij de waarde 0

betekent dat er van de betreffende kleur geen gebruik is gemaakt terwijl de waarde 7 betekent dat de betreffende kleur maximaal is toegevoegd. Alle tussenliggende waarden corresponderen met de mate van toevoeging.

Zo resulteert de menging van 6 delen rood, 6 delen groen en 4 delen blauw uiteindelijk in een lichtgele kleur. Door slechts één deel blauw toe te voegen, krijgen we een donkergele kleur.

De standaard PALETTES van MSX-2 hebben de volgende mengingen:

kleurkodering (PALETTE-nummer)		aantal delen rood	groen	blauw
0	transparant*	0	0	0
1	zwart	0	0	0
2	groen	1	6	1
3	lichtgroen	3	7	3
4	donkerblauw	1	1	7
5	lichtblauw	2	3	7
6	donkerrood	5	1	1
7	cyaan	2	6	7
8	rood	7	1	1
9	lichtrood	7	3	3
10	donkergeel	6	6	1
11	lichtgeel	6	6	4
12	donkergroen	1	4	1
13	magenta	6	2	5
14	grijs	5	5	5
15	wit	7	7	7

*deze kleur gedraagt zich niet als kleur (zie behandeling hiervoor). Echter, door een eenvoudige truc (zie verder) kan ook kleurkode 0 volwaardige kleur worden gebruikt.

Met het COLOR=(. . .) kommando (zie de behandeling aldaar) kan per PALETTE de kleursamenstelling worden veranderd door andere hoeveelheden rood, groen en blauw op te geven. Zo kunnen 512 verschillende kleuren worden gekodeerd (8 maal 8 maal 8) waarvan er maximaal 16 tegelijk kunnen worden gebruikt.

Merk op dat onder de SCREEN 6 instelling slechts een kleurkode 0, 1, 2 of 3 kan worden gebruikt. Om de kleur wit te activeren, moet dus één van de palettes 0, 1, 2 of 3 worden herzien met behulp van het COLOR=(. . .) kommando. Wanneer een kleurkode groter dan 3 wordt opgegeven onder een scherminstelling SCREEN 6, dan wordt deze waarde door 4 gedeeld en wordt de restwaarde als kleurkode aangenomen. Daarbij mag de kleurkode de waarde 31 nooit overschrijden.

Onder de SCREEN 6 instelling mag de kodering voor de randkleur eventueel de waarde 16 tot en met 31 bevatten. In dat geval worden voor de rand de kleursamenstellingen van twee palettes gecombineerd. De even beeldpuntjes krijgen dan een kleur volgens het tweede palette. Hierdoor onstaat een fijn gestreepte rand; op enige afstand versmelt deze beeldschermrand zich in onze waarneming tot één nieuwe combinatiekleur.

De twee palettes die onder SCREEN 6 als randkleur worden samengevoegd, laten zich als volgt bepalen:

kleurkode rand = 16 + 4 maal EVEN + ONEVEN

waarbij EVEN gelijk is aan het palette-nummer voor de even beeldschermpuntjes en ONEVEN gelijk is aan het palette-nummer voor de oneven beeldschermpuntjes.

Palettes 2 en 1 (2 voor de even- en 1 voor de oneven beeldpunten) laten zich volgens deze formule dus in kleurkode 16+4x2+1=25 combineren.

Onder de SCREEN 8 instelling hebben de verschillende kleurkoderingen weer een geheel andere betekenis. De maximale kleurkode is onder deze scherminstelling gelijk aan 255 en valt te berekenen met behulp van de volgende formule:

kleurkode = 4 maal het aantal delen rood plus
32 maal het aantal delen groen plus
1 maal het aantal delen blauw

Het aantal delen rood en groen kan maximaal gelijk zijn aan 7. Het aantal delen blauw is beperkt tot 3.

De kleur donkergeel (6 delen rood, 6 delen groen en 1 deel blauw) correspondeert in deze kodering dus met 4x6+32x6+1x1=217.

Zie voor een combinatievoorbeeld het voorbeeld, geplaatst bij de behandeling van COLOR=(. . .)

Het gedrag van de transparante kleur is in MSX-2 nogal verschillend onder de verschillende SCREEN-instellingen. Daarom volgen hieronder twee tabellen waarin per scherm-instelling het gedrag van kleurkode 0 wordt verklaard.

N.B.: onder SCREEN 8 worden de kleuren niet via palette-kodes aangestuurd. Daarom ontbreekt SCREEN 8 ook in deze tabellen.

TABEL 1, kleurnummer 0 in als transparante kleur gedefiniëerd.

Kleurnummer 0 kan wél of niet als transparante kleur worden ingesteld. Standaard wordt kleurnummer 0 door MSX-2 als transparante kleur beschouwd. Indien kleurnummer niet als transparante kleur is ingesteld (zie bij tabel 2 hoe dit mogelijk is), kan deze kleur weer wél transparant worden gemaakt door het bevel:

```
VDP(9)=VDP(9) AND 223
```

SCREEN	0	1	2	3	4	5	6	7
RAND	N.V.T.	kleurkode 0 wordt als normale kleur benaderd						
achtergrond	kleurkode 0 wordt als kleur behandeld	kleurkode 0 resulteert in het aannemen van de randkleur						
voorgrond	kleurkode 0 resulteert in het aannemen van de achtergrondkleur	kleurkode 0 resulteert in het aannemen van de randkleur						

TABEL 2, kleurnummer 0 is niet als transparante kleur gedefiniëerd. Kleurnummer 0 is normaal ingesteld als transparante kleur. Door uit-

voeren van het bevel

```
VDP(9)=VDP(9) OR 32
```

kan kleurnummer 0 als niet-transparante kleur worden gedefiniëerd. Toch blijft het gedrag van deze speciale kleur enigszins afwijkend:

SCREEN	0	1	2	3	4	5	6	7
achtergrond	als tabel 1	kleurkode 0 wordt normaal als kleur behandeld		als tabel 1	kleurkode 0 wordt normaal als kleur behandeld			
voorgrond								

Met het VDP-sleutelwoord (zie VDP) kunnen we de Video Display Processor zeer direkt besturen.

moeilijkheidsgraad ..... normaal
soort ..... KOMMANDO
afkomst ..... COLOR is kleur

De volgende behandeling is alleen voor MSX-2 van toepassing

schrijfwijze

```
                    =NEW!=RESTORE
COLOR[ ---------------------------------------------------- ]
      [=((<KLEURKODE>[,[<ROOD>][,[<GROEN>][,<BLAUW>]]])]\<...>,)]
```

```
<KLEURKODE>:::=<N>

<ROOD>:::=<N>

<GROEN>:::=<N>

<BLAUW>:::=<N>

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

betekenis

Het is noodzakelijk dat eerst het COLOR-kommando is bestudeerd.

Met de systeemvariabele COLOR kunnen de kleursamenstellingen per kleurkode worden aangepast. Hiertoe dient tussen haakjes eerst het palette-nummer te worden opgegeven, gevolgd door het aantal delen rood, eht aantal delen groen en het aantal delen blauw voor deze kleurencombinatie.

Zo kunnen we palette-nummer 1 (standaard zwart) de kleur wit laten geven door de volgende instruktie:

```
COLOR=(1,7,7,7)
```

terwijl we palette-nummer 15 (standaard wit) juist zwart kunnen maken door de instruktie:

```
COLOR=(15,0,0,0)
```

Met de instruktie:

```
COLOR
```

of

```
COLOR=NEW
```

herstelt u in één klap weer de standaard kleurencombinaties zoals deze in de tabel, behandeld onder het kommando COLOR, zijn gegeven.

De kleursamenstelling per kode wordt in het video-geheugen bewaard. Wanneer nu om één of andere reden de samenstelling van het video-RAM is veranderd, dan kunnen de nieuwe kleurensamenstellingen daadwerkelijk worden geactiveerd door het kommando:

```
COLOR=RESTORE
```

Dit kommando is vooral van belang wanneer komplete beeldschermen bijvoorbeeld van schijf worden geladen. Met dit kommando stelt men na het laden dan uiteindelijk de juiste kleurencombinaties in.

De instruktie:

```
COLOR=(12,6,5,4)
```

geeft een kleurencombinatie die niet in de standaard-tabel voorkomt. Deze kleur is dan als voorgrondkleur te activeren door bijvoorbeeld het kommando:

```
COLOR 12
```

Maximaal kunnen we op deze wijze 512 kleurschakeringen koderen waarvan we er maximaal 16 tegelijkertijd kunnen gebruiken.

Pas op: een SCREEN-kommando (zie aldaar) herstelt de standaard palette-waarden. Een kleurenpalette moet dus pas na een SCREEN-kommando worden ingesteld.

Het volgende, wat grotere programma spreekt qua werking voor zich. Het geeft u de mogelijkheid om op een eenvoudige wijze de 512 mogelijke combinaties te bestuderen. In een staafdiagram wordt de mengverhouding rood-groen-blauw aangegeven terwijl het beeldscherm de betreffende combinatiekleur aanneemt. Met behulp van dit programma kunt u op eenvoudige wijze zelf uw palette van kleuren samenstellen. Let u in dit voorbeeld nog niet op niet behandelde sleutelwoorden.

```
10 KEY OFF:COLOR 15,4,4:CLS:WIDTH 40:PRINT "palette-test: 51
2 kleuren":PRINT
20 PRINT "gebruik de toetsen":PRINT:PRINT"  4    5    6   (v
erhogen aantal delen)":PRINT:PRINT "  1    2    3   (verlage
n aantal delen)":PRINT:PRINT "rood groen blauw":PRINT:PRINT
"(return)"
30 A$=INPUT$(1)
40 SCREEN 3
50 COLOR 15,5,5
60 COLOR =(1,7,0,0)
70 COLOR =(2,0,7,0)
80 COLOR =(3,0,0,7)
90 I=0:J=0:K=0
100 CLS:COLOR=(5,I,J,K)
110 LINE (72,160)-(92,160-20*I),1,B
120 LINE (124,160)-(144,160-20*J),2,B
130 LINE (176,160)-(196,160-20*K),3,B
140 K$=INKEY$:IF K$="" THEN 140
150 IF K$<"1" OR K$>"6" THEN 140
160 L=VAL(K$):I=I+(L=1)-(L=4):J=J+(L=2)-(L=5):K=K+(L=3)-(L=6)
170 IF I<0 THEN I=0 ELSE IF I>7 THEN I=7
180 IF J<0 THEN J=0 ELSE IF J>7 THEN J=7
190 IF K<0 THEN K=0 ELSE IF K>7 THEN K=7
200 GOTO 100
```

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **COLOR SPRITE** betekent kleur van de 'geest'

De volgende behandeling is alleen voor MSX-2 van toepassing

schrijfwijze

```
COLOR SPRITE(<SPRITENUMMER>)=<KLEURKODE>

<SPRITENUMMER>::=<N>

<KLEURKODE>::=<N>

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

betekenis

COLOR SPRITE wordt onder PUT SPRITE behandeld; zie aldaar.

SLEUTELWOORD # COLOR SPRITE$

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **COLOR SPRITE** betekent kleur van de 'geest'

De volgende behandeling is alleen voor MSX-2 van toepassing

schrijfwijze

```
COLOR SPRITE$(<SPRITENUMMER>)=<KLEURKODERING>

<SPRITENUMMER>::=<N>

<KLEURKODERING>::=<A>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

COLOR SPRITE$ wordt onder PUT SPRITE behandeld; zie aldaar.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CONT is afkorting van continue – ga door**

schrijfwijze

```
CONT<...>

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

betekenis

Met CONT kunnen we een via het STOP-kommando of via de CONTROL-STOP afgebroken programma weer hervatten. Voorbeeld:

```
NEW
Ok
10 PRINT "REGEL 1"
20 STOP
30 PRINT "REGEL 2"
40 END
RUN
REGEL 1
Break in 20
CONT
REGEL 2
Ok
```

Indien een programma door middel van een STOP werd onderbroken, begint de hervatting van het programma na CONT bij de volgende programmaregel. Indien het programma echter door CONTROL-STOP werd onderbroken, dan begint de hervatting bij het kommando waarin het programma werd onderbroken. Voorbeeld:

```
NEW
Ok
10 GOTO 10
RUN              –Het programma staat in een zogenaamde lus.
Break in 10      –CONTROL-BREAK wordt nu ingetoetst.
CONT             –De oneindige lus wordt weer hervat...
```

Indien na onderbreking reeds wijzigingen in het programma werden aangebracht, zal de computer melden: Can't CONTINUE. Het is voor de computer dan niet mogelijk om de uitvoering te hervatten.

Merk op dat CONT standaard onder funktietoets 8 aanwezig is.

moeilijkheidsgraad . . . . . vrij moeilijk
soort . . . . . KOMMANDO
afkomst . . . . . COPY betekent kopiëren

De volgende behandeling is alleen voor MSX-2 van toepassing.

schrijfwijze

```
        <LOCATIE>-<LOCATIE>[,<SCHERMNUMMER>]
COPY ---------------------------------------- TO ...
     <NUMERIEKE VARIABELE-NAAM>[,<RICHTING>]

    <LOCATIE>[,[<SCHERMNUMMER>][,<LOGISCHE OPERATOR>]]
... ------------------------------------------------ \ <...>,
                <NUMERIEKE VARIABELE-NAAM>
```

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

<SCHERMNUMMER>:::=<N>

<RICHTING>:::=<N>

<HORIZONTAAL>:::=<N>

<VERTIKAAL>:::=<N>

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

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

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

<LOGISCHE OPERATOR>:::=XOR¦OR¦AND¦PSET¦PRESET¦TXOR¦TOR ...

                       ... ¦TAND¦TPSET¦TPRESET
```

betekenis

Met het COPY-kommando kunnen we delen van een grafisch scherm kopiëren. Dit kopiëren kan gebeuren naar een ander grafisch scherm, naar een andere plaats op het grafische scherm of naar een array-variabele. COPY werkt alleen maar met de scherminstellingen SCREEN 5 tot en met SCREEN 8.

Teneinde het COPY-kommando goed te kunnen begrijpen, is het noodzakelijk om eerst de behandeling van het SCREEN-kommando door te nemen.

Het eenvoudigste gebruik van COPY bestaat uit het kopiëren van een deel van het grafische scherm naar een andere plaats. Hiertoe geeft men na het COPY-kommando eerst twee beeldschermlocaties op, gescheiden van elkaar door een min-teken (ongeveer zoals bij het LINE-kommando). Deze twee punten bepalen de denkbeeldige rechthoek op het scherm waarbinnen de te kopiëren grafische gegevens staan.

Vervolgens geeft men na het sleutelwoord TO aan, waarnaar de grafische gegevens moeten worden gekopiëerd. Er behoeft in dit geval slechts één punt te worden opgegeven.

Het rechthoekige gebied **waarnaar** wordt gekopiëerd, wordt bepaald door het derde punt en een denkbeeldig vierde punt dat ten opzichte van dit derde punt ligt zoals het tweede punt ten opzichte van het eerste punt ligt.

Het volgende voorbeeld laat zien hoe een eenmaal gemaakte tekening meerdere keren op het scherm kan worden gedupliceerd.

```
10 SCREEN 5
20 CLS:FOR I=0 TO 30 STEP 2
30 CIRCLE(40,40),I:NEXT I
40 FOR X=10 TO 190 STEP 60:FOR Y=10 TO 1
30 STEP 60
50 COPY (10,10)-(70,70) TO (X,Y)
60 NEXT Y,X
100 GOTO100
```

In het vorige voorbeeld werd een tekening binnen hetzelfde scherm naar een andere plaats gekopiëerd.

Het is mogelijk om verschillende schermen te definiëren. Met het SETPAGE-kommando kunnen we een grafisch scherm benoemen dat zichtbaar moet zijn en een grafisch scherm benoemen waarop getekend kan worden. Afhankelijk van de instelling en het beschikbare video-geheugen kunnen meerdere verschillende grafische beeldschermen worden gedefiniëerd.

Door bij het COPY-kommando het schermnummer op te geven waarvandaan moet worden gekopiëerd (direkt na de denkbeeldige rechthoek) en het schermnummer op te geven waarnaartoe moet worden gekopiëerd (direkt na het punt waarna moet worden gekopiëerd), kunnen delen van tekeningen naar een ander grafisch scherm worden gekopiëerd. Hierdoor kan een tekening niet alleen op het actieve grafische

scherm worden gekopiëerd maar eventueel ook naar een (nog) niet actief scherm worden gekopiëerd.

In het volgende voorbeeld wordt een eenvoudige tekening gemaakt. Daarna wordt deze tekening naar een ander scherm en een andere plaats gekopiëerd. Uiteindelijk worden de twee schermen afzonderlijk steeds na elkaar geprojekteerd. De twee FOR-NEXT-regels zijn slechts ter vertraging aangebracht.

```
10 SCREEN 5:SET PAGE 0,0:CLS:SET PAGE 1,1:CLS
20 LINE (100,100)-(150,150),,B
30 LINE (90,110)-(125,75)
40 LINE (125,75)-(160,110)
50 COPY (70,70)-(180,180),1 TO (0,70),0
60 SET PAGE 0
70 FOR I=1 TO 100:NEXT I
80 SET PAGE 1
90 FOR I=1 TO 100:NEXT I
100 GOTO 60
```

Behalve dat een gedeelte van een grafisch scherm naar een andere plaats, eventueel op een ander scherm kan worden gekopiëerd, is het ook mogelijk om een kopie van een gedeelte van een grafisch scherm in een array-variabele te maken.

Deze array- of tabelvariabele wordt bij dit kopiëerproces slechts gebruikt als een gereserveerd stuk geheugen binnen het centrale computergeheugen. De afzonderlijke waarden in de tabelvariabele zijn na het kopiëren niet van belang en geven vaak rare getallen te zien. Allemaal bij elkaar bevatten ze echter een kopie van een gedeelte van een grafisch scherm en kunnen ze eventueel weer op een (ander) scherm worden geplaatst.

Voordat een stuk grafische informatie in een tabelvariabele kan worden gekopiëerd, moet deze tabelvariabele eerst met behulp van een DIM-kommando worden toegewezen. Hierbij doet zich natuurlijk onmiddellijk de vraag voor, hoe groot de tabelvariabele wel moet worden gedimensioneerd.

De vereiste grootte van een tabelvariabele laat zich als volgt berekenen:

```
10 KEY OFF:SCREEN 0:WIDTH 37:COLOR 15,4,4
20 PRINT "BEREKENEN ARRAY-DIMENSIE VOOR COPY"
30 PRINT:PRINT "LINE (X1,Y1)-(X2,Y2) TO (VARIABELE)"
40 PRINT:INPUT "WAARDE X1";X1
50 INPUT "WAARDE Y1";Y1
60 INPUT "WAARDE X2";X2
70 INPUT "WAARDE Y2";Y2
80 PRINT
90 INPUT "SCREEN ...";S
100 IF S<5 OR S>8 THEN 90
110 PRINT:PRINT "GEEF VARIABELE MET TYPE IN (%,!,#)"
120 PRINT
130 INPUT V$:IF LEN(V$)<2 THEN 120
140 T$=RIGHT$(V$,1):IF T$<>"#" AND T$<>"!" AND T$<>"%" THEN
120
150 D=4:IF S=6 THEN D=2
160 D=INT((D*(ABS(X2-X1)+1)*(ABS(Y2-Y1)+1)+7)/8)+4
170 T=2:IF T$="!" THEN T=4 ELSE IF T$="#" THEN T=8
180 D=INT(D/T)
190 PRINT:PRINT "DIM ";V$;"(";MID$(STR$(D),2);")"
200 PRINT:STOP
```

Nadat met behulp van dit programma de minimale dimensie van de tabelvariabele is bepaald, kan een stuk van het grafische scherm in de tabelvariabele worden gekopiëerd.

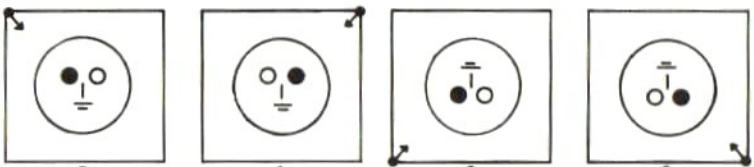
In het volgende voorbeeld wordt eerst een kleine tekening gemaakt. Deze tekening wordt in een variabele gekodeerd en daarna enige malen weer terug naar het beeldscherm gekopiëerd.

```
10 DIM Q(105)
20 SCREEN 5:CLS
30 CIRCLE (20,20),20:CIRCLE (10,10),5:CIRCLE (30,10),5:PAINT
 (10,10)
40 LINE (10,30)-(30,30):LINE (20,15)-(20,25):LINE (16,33)-(2
4,33)
50 COPY (0,0)-(40,40) TO Q
60 CLS
70 FOR X=0 TO 240 STEP 40:FOR Y=0 TO 160 STEP 40
80 COPY Q TO (X,Y)
90 NEXT Y:NEXT X
100 GOTO100
```

Wanneer een tekening of een deel van een tekening in een tabel-variabele is opgeslagen, kan deze weer worden teruggekopiëerd naar een ander gedeelte van het grafische scherm. Dit terugkopiëren kan zoals in het bovenstaande voorbeeld blijkt, gewoon rechtstandig gebeuren. Echter, bij dit terugkopiëren kan ook een richtingkode worden opge-

geven. Met deze richtingkode kan het opgeslagen schermgedeelte op een gespiegelde wijze weer worden geplaatst.

Het volgende schema geeft aan, welke betekenis de verschillende richtingkodes hebben:



Onder kode 0 wordt het schermgedeelte op normale wijze van linksboven naar rechtsonder geplaatst. Onder kode 1 wordt het schermgedeelte vanuit het rechterbovenpunt geplaatst en als zodanig gespiegeld. Onder kode 2 wordt het schermgedeelte vanuit het linker onderpunt geplaatst en onder kode 3 wordt het schermgedeelte vanuit het rechter onderpunt geplaatst.

In het vorige voorbeeld kunnen we regel 80 vervangen door één van de volgende programmaregels:

```
80 COPY Q,0 TO (X,Y)
80 COPY Q,1 TO (X,Y)
80 COPY Q,2 TO (X,Y)
80 COPY Q,3 TO (X,Y)
```

waarna de beschreven spiegeleffekten te bestuderen zijn.

Uiteindelijk kunnen we aan het COPY-kommando nog een logische operator toekennen wanneer we niet naar een tabelvariabele kopiëren.

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.

Het volgende voorbeeld laat op eenvoudige wijze zien hoe PRESET en TPRESET werken. Tik het voorbeeld eerst in, RUN en vervang daarna op regel 70 de logische operator TPRESET door PRESET. RUN het programma opnieuw en let op de wijze waarop de transparante kleur nu wordt behandeld.

```
10 SCREEN 5
20 COLOR 0,0,0
30 CIRCLE (10,10),10,1
40 DIM A(30)
50 COPY (0,0)-(20,20) TO A
55 COLOR 15
60 CLS:PAINT (100,100)
70 COPY A TO (RND(1)*250,RND(1)*200),,TPRESET
80 GOTO 70
```

Het volgende voorbeeld laat u de uitwerking van de logische operator XOR zien. Het voordeel van de XOR-bewerking is, dat een eventuele achtergrond bij een tweede, exact hetzelfde COPY-kommando weer verdwijnt. Zo kunnen zonder dat er SPRITES behoeven te worden gebruikt, bewegende figuurtjes worden geprogrammeerd langs een onveranderlijke achtergrond.

De XOR-bewerking kan, vooral in kleur, verrassende resultaten opleveren. Laat het volgende voorbeeld gedurende een wat langere tijd lopen en bestudeer het verbazende schouwspel.

```
10 DIM A(28)
20 SCREEN 5:COLOR=NEW:COLOR 0,0,0:CLS
30 CIRCLE (10,10),10,4:PAINT (10,10),8,4
40 COPY (0,0)-(20,20) TO A
50 X=0:Y=0:DX=2:DY=2
60 COPY A TO (X,Y),,XOR
70 X=X+DX:Y=Y+DY
80 IF X>234-DXTHENDX=-DX
90 IF Y>192-DYTHENDY=-DY
100 IFX<-DX THEN DX=-DX
110 IFY<-DYTHENDY=-DY
120 GOTO 60
```

Maxima en minima

De schermcoördinaten dienen tussen –32769 en 32768 te liggen terwijl een decimale fractie altijd wordt verwaarloosd. Het schermnummer is een geheel getal, minimaal gelijk aan 0 en maximaal gelijk aan het voor de betreffende SCREEN-instelling toegestane waarde. Een decimale fractie wordt ook hier verwaarloosd.

Het richtingnummer dient een waarde te hebben, minimaal gelijk aan 0 en maximaal gelijk aan 3 terwijl een decimale fractie wordt verwaarloosd.

Opmerkingen

Het COPY-kommando kan wanneer een gedeelte buiten de toegestane beeldgrenzen wordt gekopiëerd, vreemde gevolgen opleveren. Er kunnen plotseling rare figuurtjes midden op het beeldscherm verschijnen. Dit komt doordat (een klein foutje in MSX-2) het COPY-bevel een enkele keer toestaat dat er een gedeelte van het scherm over bepaalde controle-tabellen in het video geheugen wordt gekopiëerd.

Het COPY-kommando begint altijd links bovenaan met kopiëren en werkt vertikaal de beeldpuntjes af. Dit is vooral plezierig om te weten wanneer een gebied (gedeeltelijk) over zichzelf moet worden gekopieerd.
In het volgende, laatste voorbeeld wordt een onverwacht effect bij het kopiëren van een lijn, één beeldpuntje verder naar rechts, getoond. Met bovenstaande wetenschap is dit effect echter gemakkelijk te verklaren.

```
10 SCREEN 5:COLOR 15,4,4:CLS
20 LINE (100,10)-(100,100)
30 COPY (100,10)-(200,100) TO (104,10)
40 GOTO40
```

moeilijkheidsgraad . . vrij moeilijk, kennis van goniometrie is noodzakelijk
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **COS is afkorting van cosine – cosinus**

schrijfwijze

```
COS(<HOEK>)

<HOEK>:::=<N>

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

betekenis

Deze funktie geeft als resultaat de cosinus van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen. Het resultaat ligt uiteraard altijd tussen –1 en 1.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

Voorbeeld:

```
NEW
Ok
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926536
30 PRINT I;":";COS(HOEK)
40 NEXT I
RUN
```

(een cosinustabel verschijnt op het beeldscherm. Merk op dat op regel 20 de hoek van graden naar radialen wordt geconverteerd)

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **CSAVE is afkorting van cassette save – stel veilig op cassetteband**

schrijfwijze

```
CSAVE<BESTANDSNAAM>[,<SCHRIJFSNELHEID>]

<BESTANDSNAAM>::=<A>

<SCHRIJFSNELHEID>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met het CSAVE-kommando kunnen programma's op cassetteband worden vastgelegd. Dit vastleggen gebeurt gekodeerd; het programma wordt op cassetteband vastgelegd in de kodes zoals die zich in het geheugen bevinden; dit in tegenstelling tot het SAVE-kommando waarin het programma in tekstuele vorm wordt vastgelegd.

Programma's, vastgelegd met CSAVE, kunnen met een RUN-kommando niet automatisch worden opgestart; hiertoe dient gebruik te worden gemaakt van het SAVE-kommando.

Met het CSAVE-kommando dient de programmanaam op cassette te worden bepaald. Deze naam MOET minimaal één teken en MAG maximaal 6 tekens bevatten. Een teveel aan tekens wordt verwaarloosd.

Als tweede gegeven bij de CSAVE kan eventueel een schrijfsnelheid worden gespecificeerd. De numerieke uitdrukking die de schrijfsnelheid bepaalt, mag gelijk zijn aan 1 of 2. Indien deze numerieke uitdrukking een gebroken waarde bevat, dan wordt een decimale fractie verwaarloosd. De snelheidsaanduiding heeft de volgende betekenis:

1: er wordt met 1200 baud geschreven, hetgeen wil zeggen dat er 1200 bits (ongeveer 150 karakters) per seconde worden overge-

dragen op de band
2: er wordt met 2400 baud geschreven; de dubbele snelheid dus.

Indien geen schrijfsnelheid wordt gespecificeerd, wordt automatisch een snelheid van 1200 baud aangenomen of de laatste (via CSAVE of SCREEN) bepaalde schrijfsnelheid indien deze werd gespecificeerd.

Indien men over een cassetterecorder van goede kwaliteit beschikt, en daarbij kwalitatief goede cassettebanden gebruikt, kan voor de hoogste snelheid worden gekozen. Echter, wanneer de kwaliteiten wat lager liggen, is het zéér raadzaam om de laagste snelheid te kiezen.

MSX schrijft geen lees/schrijfcontrole voor; een slechte kwaliteit van apparatuur kan veroorzaken dat er fouten optreden tijdens het vastleggen.

voorbeeld

Zie het voorbeeld bij CLOAD

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **CSNG is afkorting van convert to single precision value – zet over naar enkelvoudige precisie waarde**

schrijfwijze

```
CSNG(<N>)

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

betekenis

Deze funktie geeft als resultaat de waarde van de tussen haakjes vermelde numerieke uitdrukking nadat deze is herleid naar enkelvoudige precisie. Bijvoorbeeld:

```
NEW
Ok
10 PRINT 1/3
20 PRINT CSNG(1/3)
RUN
 .33333333333333
 .333333
Ok
```

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **CSRLIN is afkorting van cursorline – cursor regel**

schrijfwijze

```
CSRLIN
```

betekenis

Deze systeemvariabele geeft als resultaat het regelnummer waarop de cursor zich op het moment bevindt. Voorbeeld:

```
NEW
OK
10 FOR I=0 TO 5:LOCATE ,I
20 PRINT CSRLIN:NEXT:STOP
RUN
```

(In de linkerbovenhoek van het beeld verschijnen onder elkaar de cijfers 0 tot en met 5.)

De systeemvariabele CSRLIN kan geen waarde worden toegekend; de waarde kan alleen worden gebruikt.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DATA is gegevens**

schrijfwijze

```
DATA(,)<DATA-ELEMENT>[,(,)<DATA-ELEMENT>]

<DATA-ELEMENT>::=["]<ALFANUMERIEKE KONSTANTE>["]|<NUMERIEKE KONSTANTE>

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

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

betekenis

Behalve het sleutelwoord DATA worden ook de sleutelwoorden READ en RESTORE hier behandeld.

Met het DATA-sleutelwoord kunnen we vaste gegevens in een programma opnemen. Deze gegevens mogen bestaan uit numerieke konstanten of alfanumerieke konstanten. Bijvoorbeeld:

```
10 DATA 1.2,3.4,123.6,-12.33,"MAANDAG","PIETJE"
```

De diverse gegevens dienen door middel van een komma van elkaar te worden gescheiden.

De op deze wijze in een programma opgenomen gegevens kunnen door middel van een READ-kommando in een variabele worden ingelezen. Bijvoorbeeld:

```
NEW
Ok
10 DATA "MAANDAG","DINSDAG","WOENSDAG"
20 DATA "DONDERDAG","VRIJDAG","ZATERDAG"
30 DATA "ZONDAG"
40 FOR I=1 TO 7:READ DAG$
50 PRINT DAG$:NEXT I:STOP
RUN
MAANDAG
DINSDAG
WOENSDAG
DONDERDAG
VRIJDAG
ZATERDAG
```

```
ZONDAG
Break in 50
Ok
```

In het laatste voorbeeld worden de alfanumerieke konstanten MAANDAG... ZONDAG achtereenvolgens in de variabele DAG$ ingelezen en vervolgens afgedrukt op het beeldscherm. Bij een READ 'onthoudt' de computer waar hij is gebleven met inlezen. Is MAANDAG reeds ingelezen, dan zal de volgende keer dat een READ wordt ingelezen, automatisch DINSDAG worden ingelezen. Het maakt hiervoor niet uit wáár precies de konstanten in een DATA-commando zijn genoemd; de computer begint altijd bij de eerste konstante en werkt deze konstante in volgorde van voorkomen af. Bijvoorbeeld:

```
NEW
Ok
10 DATA 33,44,55,66
20 READ A:PRINT A:GOTO 20
30 DATA 77,88,99
RUN
 33
 44
 55
 66
 77
 88
 99
Out of data in 20
Ok
```

We zien dat alle genoemde konstanten worden ingelezen en afgedrukt. Het maakt niet uit of de gegevens in een DATA-kommando voor of na de leesopdracht zijn genoemd. Pas als er geen volgende gegevens meer kunnen worden gevonden, geeft de computer een foutmelding.

Met een RESTORE-kommando kunnen we de computer opdracht geven, bij de volgende READ-opdracht weer bij een bepaald gegeven te beginnen met inlezen. Bijvoorbeeld:

```
NEW
Ok
10 DATA 1,2,3,4,5
20 RESTORE 20
30 DATA 111,222,333
40 READ A,B,C
50 PRINT A;B;C
```

```
RUN
 111   222   333
Ok
```

Met de RESTORE 20 opdracht gaven we in bovenstaand voorbeeld aan dat het inlezen op of na programmaregel 20 diende te geschieden. Alleen een RESTORE-opdracht (dus zonder regelnummer) heeft tot gevolg dat de computer weer bij het allereerste DATA-gegeven begint. Bijvoorbeeld:

```
NEW
Ok
10 READ A:IF A=0 THEN RESTORE:GOTO 10
20 PRINT A;:READ A$:PRINT A$:GOTO 10
1000 DATA 1,"ZONDAG",2,"MAANDAG",3,"DINSDAG"
1010 DATA 4,"WOENSDAG",5,"DONDERDAG",6
1020 DATA "VRIJDAG",7,"ZATERDAG",0
RUN
 1 ZONDAG
 2 MAANDAG
 3 DINSDAG
 4 WOENSDAG
 5 DONDERDAG
 6 VRIJDAG
 7 ZATERDAG
 1 ZONDAG
 2 MAANDAG ...
```

(etcetera, de dagen van de week blijven op het beeldscherm voorbijkomen. Onderbreken met CONTROL-STOP)

In bovenstaand voorbeeld wordt steeds een dagnummer en een dagnaam ingelezen in variabele A en A$. Indien op regel 10 echter voor variabele A een nul-waarde wordt ingelezen, wordt een RESTORE uitgevoerd waarna opnieuw de READ wordt uitgevoerd. Na deze RESTORE wordt op regel 10 weer de waarde 1 voor variabele A ingelezen en wordt op regel 20 de waarde ZONDAG voor variabele A$ ingelezen.

Merk op dat numerieke gegevens in een DATA-regel wel met READ in een alfanumerieke variabele kunnen worden gelezen, maar dat alfanumerieke gegevens niet in een numerieke variabele kunnen worden gelezen; dit laatste resulteert in een Sytax error op de betreffende DATA-regel.

moeilijkheidsgraad . . normaal, enige elementaire kennis van funktie-theorie is een voordeel

soort . . . . . **KOMMANDO**

afkomst . . . . . **DEF FN is afkorting van define function – definieer funktie/bewerking**

schrijfwijze

```
        <ALFANUMERIEKE VARIABELE-NAAM>[(<VARIABELE-NAAM>[,<VARIABE
DEF FN  ------------------------------------------------------- ...
        <NUMERIEKE VARIABELE-NAAM>[(<VARIABELE-NAAM>[,<VARIABELE-N

        LE-NAAM>])]=<A>
    ... ---------------
        AAM>])]=<N>
```

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

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

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

MSX-basic bevat vele ingebouwde funkties. SQR (worteltrekken), INT (integere waarde bepalen) en SIN (sinus berekenen) zijn enkele van deze funkties.
Het kan echter zijn dat men in een programma een specifieke funktie nodig heeft die helaas niet in MSX-basic bestaat. In dat geval kan deze funktie door middel van DEF FN zelf worden ontworpen.

Allereerst dient deze te ontwerpen funktie een naam te hebben. We geven de funktie de naam van een alfanumerieke variabele wanneer de uitkomst van deze funktie alfanumeriek gaat worden en we geven de funktie de naam van een numerieke variabele indien de uitkomst van deze funktie een numerieke wordt.

Eventueel komen tussen haakjes de variabelenamen te staan. Deze variabelenamen zijn slechts namen en vormen *geen echte variabelen.* Zij dienen slechts om aan te geven waar later bij het uitvoeren van de funktie de op dat moment ingevulde variabelen moeten komen te staan.

Indien echter rechts van het gelijk-teken een variabele wordt genoemd die niet tussen haakjes links van het gelijkteken is opgenomen, dan

wordt de *actuele* waarde van deze variabele gebruikt.

We ontwerpen als eerste een funktie met een numeriek resultaat. Deze funktie noemen we HYP. Hij rekent de lengte van de schuine zijde van een rechthoekige driehoek uit wanneer de lengten van de twee overige zijden gegeven zijn. De funktie luidt:

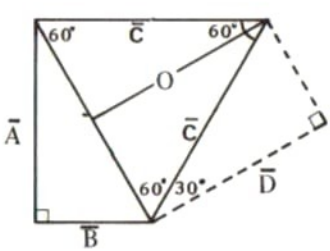
```
NEW
Ok
10 DEF FN HYP(A,B)=SQR(A*A+B*B)
```

Later kunnen we alleen met het woordje HYP deze funktie weer oproepen. Bijvoorbeeld:

```
20 PRINT FN HYP(12,13)
RUN
 17.691806012953
Ok
```

De ene funktie mag de andere funktie gebruiken. Zo kunnen we een volgende funktie ontwerpen die, uitgaande van de lengte van de schuine zijde van de rechthoekige driehoek, de oppervlakte berekent van de gelijkbenige driehoek met deze lengte. Zie de constructietekening. De funktie luidt:

```
20 DEF FN OPP(A,B)=SQR(3)*FN HYP(A,B)^2/4
30 PRINT FN OPP(12,13)
RUN
 135.53297569224
Ok
```



Verklaring:

$\bar{C}=\sqrt{(\bar{A}^2+\bar{B}^2)}$

$\bar{D}=\bar{C}\cdot\cos(30)=\frac{1}{2}\sqrt{3}\cdot\bar{C}$

De oppervlakte van een driehoek is gelijk aan half basis maal hoogte. Dus:

$O=\frac{1}{2}\cdot\bar{D}\cdot\bar{C}=\frac{1}{4}\sqrt{3}\cdot\bar{C}^2$

Uiteindelijk zouden we nog een derde funktie in het leven kunnen roe-

pen die dezelfde berekening uitvoert, uitgaande van een *gelijkbenige* driehoek. Er behoeft dan slechts één zijde te worden opgegeven:

```
30 DEF FN OVL(A)=FN OPP(A,A)
40 PRINT FN OVL(12)
RUN
 124.70765814495
Ok
```

Een éénvoudige funktie die erg nuttig kan zijn, is bijvoorbeeld een afrondfunktie op twee cijfers (geldbedragen):

```
NEW
Ok
10 DEF FN AFR(X)=FIX(X*100+SGN(X)/2)/100
20 PRINT FN AFR(100*RND(1)-50)
30 GOTO 20
```

Op het beeldscherm worden willekeurige getallen afgedrukt. Zij zijn allemaal op twee cijfers na de komma afgerond.

Een voorbeeld van een alfanumerieke funktie: we definieren een funktie die uit een gegeven string een tussen haakjes geplaatst gedeelte isoleert:

```
NEW
Ok
10 DEF FN HAA$(A$)=MID$(A$,INSTR(A$,"(")+1,INSTR(A$,")")-INS
TR(A$,"(")-1)
20 PRINT FN HAA$("HIJ ZEI (MET LUIDE STEM) HOERA!!!")
30 Q$="DEZE TAAL (BASIC) IS NIET MOEILIJK."
40 PRINT FN HAA$(Q$)
RUN
MET LUIDE STEM
BASIC
Ok
```

Moeilijke funkties kunnen als zodanig één keer worden uitgedacht en opgenomen en voortaan in het programma eenvoudig worden toegepast.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DEFDBL is afkorting van define double precision variables – definieer variabelen met dubbele precisie**

schrijfwijze

```
DEF DBL<LETTER>[-<LETTER>](,<LETTER>[-<LETTER>])

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

betekenis

Dit kommando wordt onder DEFSTR behandeld. Zie aldaar.

SLEUTELWOORD DEFINT

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DEFINT is afkorting van define integer variables – definieer integere variabelen**

schrijfwijze

```
DEF INT<LETTER>[-<LETTER>](,<LETTER>[-<LETTER>])

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

betekenis

Dit kommando wordt onder DEFSTR behandeld. Zie aldaar.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DEFSGN is afkorting van define single precision variables – definieer variabelen met enkelvoudige precisie**

schrijfwijze

```
DEF SNG<LETTER>[-<LETTER>](,<LETTER>[-<LETTER>])

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

betekenis

Dit kommando wordt onder DEFSTR behandeld. Zie aldaar.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DEFSTR is afkorting van define string variables – definieer string-variabelen**

schrijfwijze

```
DEF STR<LETTER>[-<LETTER>][,<LETTER>[-<LETTER>]]

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

betekenis

Voor een goed begrip van onderstaande funkties is het raadzaam, de hoofdstukken 5 en 6 nog eens door te nemen.

In MSX-basic wordt een variabele, indien de naam niet één van de achtervoegsels $, %, ! of # heeft, automatisch gemaakt tot een variabele met dubbele precisie. Indien we bijvoorbeeld programmeren:

```
NEW
Ok
10 LET A=1/3
20 PRINT A
RUN
 .33333333333333
Ok
```

Dan zien we dat voor de variabele A automatisch een numerieke variabele werd gekozen met een precisie van 14 cijfers, een dubbele precisie dus. Als het ware werd automatisch voor het achtervoegsel # gekozen.

Dit automatisme kunnen we sturen door gebruik van de DEF STR, de DEF INT, de DEF SNG en de DEF DBL. Achter het betreffende sleutelwoord dienen letterreeksen te worden aangegeven. Na het uitvoeren van één van de genoemde kommando's wordt voortaan voor een variabele met een begin letter, genoemd in een letterreeks, een ander variabeletype gehanteerd (indien het type niet via een achtervoegsel is bepaald).

De sleutelwoorden hebben het volgende effekt:

`DEF STR`	als standaard variabele-type wordt het type alfanumerieke variabele gehanteerd
`DEF INT`	als standaard variabele-type wordt het type integere variabele gehanteerd
`DEF SNG`	als standaard variabele-type wordt het type variabele met enkelvoudige precisie gehanteerd
`DEF DBL`	als standaard variabele-type wordt het type variabele met dubbele precisie gehanteerd

De letterreeksen die achter een DEFSTR, DEFINT, DEFSNG of DEFDBL worden opgenomen, kunnen uit twee soorten bestaan, namelijk de enkelvoudige opsomming en de reeksaanduiding. Bij de enkelvoudige opsomming worden alle geldende letters, gescheiden door een komma, opgegeven. Bij de reeksaanduiding wordt de eerste geldende letter, een min-teken en de laatste geldende letter opgegeven. Bijvoorbeeld:

`DEF STR A,C,F`	alle variabelen waarvan de naam begint met een A, een C of een F en geen achtervoegsel hebben ($, %, ! of #), worden **automatisch** beschouwd als alfanumerieke variabelen (krijgen als het ware automatisch een $ als achtervoegsel)
`DEF SGN C-Q,Z`	alle variabelen waarvan de naam begint met een C, D, E...Q of met een Z en daarbij geen achtervoegsel hebben, worden automatisch beschouwd als variabelen met enkelvoudige precisie (krijgen als het ware automatisch een ! als achtervoegsel)

Voorbeeld:

```
NEW
Ok
10 DEF STR A-C
20 A="ALFANU"
30 B="MERIEKE V"
40 C="ARIABELEN"
50 PRINT A;B;C
55 PRINT A$;B$;C$
60 A%=32000:PRINT A%:STOP
RUN
```

```
ALFANUMERIEKE VARIABELEN
ALFANUMERIEKE VARIABELEN
 32000
Break in 60
Ok
```

We zien dat de variabelen A, B, C, indien niet met achtervoegsel gebruikt, na de DEFSTR A–C automatisch als alfanumerieke variabelen worden beschouwd. Op programmaregel 60 zien we echter dat wanneer een achtervoegsel wordt gebruikt, dit achtervoegsel geldend is. Ook zien we dat het achtervoegsel (regel 55) eventueel wel mag worden gebruikt.

Merk op dat een reeksaanduiding waarbij de tweede letter kleiner is dan de eerste letter (bijvoorbeeld een DEFDBL C–A), ten onrechte een syntax error (fout in schrijfwijze) veroorzaakt. Deze foutmelding is niet de juiste; een kleine fout in MSX-basic.

moeilijkheidsgraad .. **zeer moeilijk, kennis van machinetaal en of ROM-inhoud is noodzakelijk**
soort ..... **KOMMANDO**
afkomst ..... **DEFUSR is afkorting van define user function – definieer een door de gebruiker ontworpen (niet basic) funktie**

schrijfwijze

```
DEF USR[<CIJFER>]=<GEHEUGENADRES>

<CIJFER>::=<ZIE ALGEMENE SPECIFICATIES>

<GEHEUGENADRES>::=<N>

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

betekenis

Met dit kommando kan een funktie worden geactiveerd die eerder in machinekode in het computergeheugen werd opgenomen. Maximaal 10 van deze funkties kunnen worden gedefinieerd. Deze funkties kunnen genummerd worden vanaf 0 tot en met 9; het cijfer direkt achter het DEF USR-sleutelwoord. Indien dit cijfer wordt weggelaten, wordt een 0 aangenomen. Na het gelijkteken dient een geheugenadres te worden opgenomen. Dit adres geeft aan op welk byte in het geheugen het eerste uit te voeren machinekommando staat voor de betreffende funktie. Het geheugenadres mag niet kleiner zijn dan –32768 en niet groter dan 65535. Indien de waarde van het geheugenadres kleiner is dan 0, wordt er ter berekening van het juiste adres de waarde 65536 bij opgeteld.

Nadat de funktie op deze wijze is gedefinieerd, kan hij met het sleutelwoord USR weer worden opgeroepen. Na USR dient het nummer van de funktie te staan (0 ... 9). Indien dit nummer wordt weggelaten, wordt een 0 aangenomen. Vervolgens dient tussen haakjes een uitdrukking te worden opgenomen. Deze uitdrukking bevat al dan niet een waarde die naar het betreffende stuk machinekode-programma moet worden doorgespeeld maar moet altijd worden opgenomen. Indien deze waarde verder niet van belang is, kan het beste een cijfer 0 worden opgenomen.

USR vormt een funktie net zoals bijvoorbeeld de INT en de ABS

funktie. Als zodanig kan ook deze funktie worden opgenomen in een uitdrukking of worden afgedrukt met een PRINT-kommando e.d. Steeds als de USR-funktie op welke wijze dan ook wordt aangesproken, wordt het stuk machinekodeprogrammatuur op het aangegeven geheugenadres uitgevoerd. MSX-basic gaat er dan van uit dat deze machinekodeprogrammatuur uiteindelijk resulteert in een terugkeer naar MSX-basic met een waarde die in een uitdrukking verder kan worden verwerkt.

Een voorbeeld: Het volgende programma start een machine-routine op geheugenadres 0. Op dit adres start de computer altijd op wanneer de stroom wordt ingeschakeld en hier bevindt zich dan ook de opstartroutine. Het effect van dit programma is, dat de machine opnieuw wordt opgestart als ware de stroom even uitgeschakeld geweest. Een eventueel programma is verdwenen:

```
NEW
Ok
10 DEF USR5=0
20 PRINT USR5(0)
RUN
```

Het samenstellen van een stuk machinetaal-programmatuur vereist een zeer uitgebreide kennis van zaken in verband met de Z80 microprocessor. Ook dient de samenstelling van de MSX-computer alsmede de samenstelling van het computergeheugen te worden beheerst. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop niet verder ingegaan.

Hieronder volgen nog wat gegevens voor de doorgewinterde computerman in verband met het sleutelwoord USR. De leek op machinekodegebied kan de rest maar beter overslaan.

USR-geheugenlocaties (hex.)

Type konstante of variabele	geheugenlocatie van de tussen haakjes ingevoerde waarde	geheugenlocatie waarop de geretourneerde waarde moet worden geplaatst
INTEGER	F663 bevat de waarde 2. F7F8-F7F9 bevat de integere waarde	F663 moet op de waarde 2 gesteld worden. F7F8-F7F9 moet met de betreffende waarde worden gevuld
STRING	F663 bevat de waarde 3. F7F8 bevat de lengte van de string. F7F9-F7FA bevat het geheugenadres	F663 moet op de waarde 3 gesteld worden. F7F8 moet op de lengte van de string worden gesteld. F7F9 -F7FA moet op het string-adres gesteld worden
Enkelvoudige precisie	F663 bevat de waarde 4. F7F6-F7F9 bevat de enkelvoudige precisiewaarde	F663 moet met de waarde 4 worden gevuld. F7F6-F7F9 moet worden gevuld met de betreffende waarde
Dubbele precisie	F663 bevat de waarde 8. F7F6-F7FD bevat de dubbele precisiewaarde	F663 moet met de waarde 8 worden gevuld. F7F6-F7FD moet worden gevuld met de betreffende waarde

Een integere waarde wordt binair in de 2-complement-methode opgeslagen. De mantisse van een waarde met enkelvoudige of dubbele precisie wordt in binary coded decimal (BCD) in respectievelijk 3 en 7 bytes opgeslagen. De exponent wordt samen met het teken van de mantisse in het eerste byte opgenomen. Het eerste bit van dit byte is 0 bij een positieve en 1 bij een negatieve mantisse. De overige 7 bits bevatten de waarde van de exponent + 65.

SLEUTELWOORD # DELETE

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DELETE is verwijderen**

schrijfwijze

```
        [<REGELNUMMER>][-<REGELNUMMER>]
DELETE  -------------------------------
                       •
```

betekenis

Met dit kommando kunt u:

- Een regel uit het programma verwijderen. Geef in dat geval DELETE in, gevolgd door het regelnummer of een punt voor het laatst behandelde regelnummer.
- Een eerste gedeelte van een programma verwijderen. Geef hiertoe DELETE in, gevolgd door een minteken en het laatste te verwijderen regelnummer.
- Een middengedeelte van een programma verwijderen. Geef hiertoe DELETE in, gevolgd door het eerste te verwijderen regelnummer, een minteken en het laatste te verwijderen regelnummer.

Voorbeelden:

```
DELETE 60
```

regel 60 wordt verwijderd

```
DELETE 60-120
```

regels 60 tot en met 120 worden verwijderd

```
DELETE -90
```

alle regels tot en met 90 worden verwijderd

```
DELETE .
```

de laatste behandelde regel wordt verwijderd

De aangegeven regelnummers dienen bestaande regelnummers te zijn.

Indien DELETE in een programmaregel is opgenomen, worden de daarbij vermelde regelnummers meehernummerd (zie de behandeling van RENUM).

moeilijkheidsgraad . . normaal, zeer elementaire kennis van matrix-berekeningen is een voordeel
soort . . . . . **KOMMANDO**
afkomst . . . . . **DIM is afkorting van dimension – dimensie**

schrijfwijze

```
DIM<RIJ VAN ARRAY-VARIABELEN>

<RIJ VAN ARRAY-VARIABELEN>::=<VARIABELE-NAAM>[(<DIMENSIE>...

... {,<DIMENSIE>})][,<RIJ VAN ARRAY-VARIABELEN>]

<DIMENSIE>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met het DIM-kommando kunnen we zogenaamde array-variabelen toewijzen. Een array-variabele is te vergelijken met een tabel waarin we meerdere waarden kunnen opslaan.
Zowel het aantal dimensies als het aantal elementen per dimensie is praktisch onbeperkt. Theoretisch is het maximum aantal dimensies gelijk aan 255 en het maximum aantal elementen per dimensie gelijk aan 65535. Indien tussen haakjes een numerieke uitdrukking voorkomt met een gebroken waarde, dan wordt een decimale fractie verwaarloosd. Een array kan één-dimensionaal (een rij van waarden), twee-dimensionaal (een matrix van waarden), drie-dimensionaal (een ruimtelijke matrix) maar ook méér-dimensionaal zijn. Alhoewel het werken met meerdere dimensies een abstract denkvermogen vereist, kan dit toch wel eens erg gemakkelijk zijn.

Voorbeeld:

```
NEW
Ok
10 DIM A(3,3):LET T=0
20 FOR I=0 TO 3:FOR J=0 TO 3
30 LET A(I,J)=T:LET T=T+1
40 NEXT J,I
50 INPUT "RIJ ";R:IF R=0 THEN STOP
60 INPUT "KOLOM ";K
```

```
70 PRINT "DE WAARDE OP";R;",";K
80 PRINT "IS";A(R,K):GOTO 50
RUN
RIJ ? 2
KOLOM ? 2
DE WAARDE OP 2 , 2
IS 10
RIJ ? 3
KOLOM ? 1
DE WAARDE OP 3 , 1
IS 13
RIJ ? 0
Break in 50
Ok
```

Indien een variabele-naam in een dim-kommando wordt genoemd zonder dimensiebepaling, dan wordt deze variabele slechts toegewezen en op nul gesteld.

Merk op dat meer arrays in een dim-kommando kunnen worden gedimensioneerd. Bijvoorbeeld:

```
10 DIM A(3,2),B$(2),C(1,2,5),B%(100,5)
```

Ook alfanumerieke tabellen kunnen worden aangelegd. Een element in een alfanumerieke tabel mag een andere lengte hebben. Bijvoorbeeld:

```
NEW
Ok
10 DIM NAAM$(100)
20 INPUT "NUMMER ";NUMMER%
30 IF NUMMER%<0 OR NUMMER%>100 GOTO 20
40 PRINT "OUDE NAAM: ";NAAM$(NUMMER%)
50 INPUT "NIEUWE NAAM: ";NAAM$(NUMMER%)
60 GOTO 20
RUN
NUMMER ? 1
OUDE NAAM:
NIEUWE NAAM: ? FERDINAND
NUMMER ? 55
OUDE NAAM:
NIEUWE NAAM: ? KAREL DE GROTE
NUMMER ? 1
OUDE NAAM: FERDINAND
NIEUWE NAAM: ? FREDERIK
NUMMER ? 55
OUDE NAAM: KAREL DE GROTE
NIEUWE NAAM: ?
```

Wanneer we een array-variabele in een programma gebruiken zonder deze eerst met DIM te dimensioneren, wordt deze variabele in de gebruikte dimensies met 10 gedimensioneerd.
Bijvoorbeeld:

```
LET A(2,4,6)=33.4
```

dimensioneert variabele A als ware:

```
DIM A(10,10,10)
```

uitgevoerd.

SLEUTELWOORD # DRAW

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **DRAW is tekenen**

schrijfwijze

```
DRAW<TEKENAANWIJZING>

<TEKENAANWIJZING>::=<A>

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

betekenis

Voor goed begrip van dit kommando is het noodzakelijk dat de PSET terdege is bestudeerd.

Met het DRAW-kommando kunnen tekeningen op het beeldscherm worden gemaakt. Achter het DRAW-kommando dient dan een alfanumerieke uitdrukking te worden opgenomen die de teken-aanwijzingen voor de computer bevatten. Deze alfanumerieke uitdrukking dient te zijn opgesteld volgens de GML (Graphics Macro Language) richtlijnen. GML kan als een eenvoudige, aparte taal worden gezien die door Microsoft is ontwikkeld en alleen een grafische toepassing heeft.

Binnen GML kennen we volgende, éénletterige kommando's:

M...,... de M staat voor MOVE. De getallen, achter MOVE opgenomen, geven aan naar welk punt moet worden gegaan op het beeldscherm. De geadresseerde beeldschermlocatie mag zich buiten het scherm bevinden. Vanuit het laatst getekende punt wordt een lijn getrokken naar het aangegeven punt. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 DRAW "M100,100M200,100M20,30M22,33"
30 GOTO 30
RUN
```

Een lijnenspel wordt getekend.

Indien voor de achter M opgenomen getallen een plus- of minteken staat, wordt de verplaatsing relatief beschouwd ten opzichte van het laatst getekende punt. Een M+100,100 heeft dan bijvoorbeeld tot gevolg dat een lijn vanuit het laatst getekende punt wordt getrokken naar een punt dat op de vertikale locatie 100 ligt en op een horizontale locatie, 100 beeldschermpunten verder naar rechts dan die van het laatst getekende punt.

Voorbeeld:

```
NEW
Ok
10 SCREEN 2
20 DRAW "M100,100M+11,+0M+0,-11M-11,+0M+
0,+11"
30 GOTO 30
RUN
```

Eerst wordt een lijn vanuit het laatst getekende punt naar (100,100) getrokken. Vervolgens wordt een vierkantje getekend door achtereenvolgens een lijn van 11 beeldpunten naar rechts, naar beneden, naar links en weer naar boven te tekenen.

U... de U staat voor UP. Vanuit het laatst getekende punt wordt een lijn naar boven getrokken ter lengte van een aantal aangegeven beeldschermpunten.

R... de R staat voor RIGHT. Vanuit het laatst getekende punt wordt een lijn naar rechts getrokken ter lengte van het aantal aangegeven beeldschermpunten.

D... de D staat voor DOWN. Vanuit het laatst getekende punt wordt een lijn naar beneden getrokken ter lengte van het aantal aangegeven beeldschermpunten.

L... de L staat voor LEFT. Vanuit het laatst getekende punt wordt een lijn naar links getrokken ter lengte van het aantal aangegeven beeldschermpunten.

Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
```

```
20 DRAW "M100,100U20R20D20L20"
30 GOTO 30
RUN
```

Vanuit het laatst getekende punt wordt eerst een lijn naar (100,100) getrokken. Daarna wordt vanuit dit punt een vierkantje getekend; twintig beeldpunten naar boven, naar rechts, naar beneden en weer naar links.

E... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten hoger en verder naar rechts gelegen. Er wordt dus een lijn rechts schuin naar boven getekend.

F... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten lager en verder naar rechts gelegen. Er wordt dus een lijn rechts schuin naar beneden getekend.

G... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten lager en verder naar links gelegen. Er wordt dus een lijn links schuin naar beneden getekend.

H... vanuit het laatst getekende punt wordt een lijn getrokken naar een punt, het aantal aangegeven beeldschermpunten hoger en verder naar links gelegen. Er wordt dus een lijn links schuin naar boven getekend.

Voorbeeld:

```
NEW
Ok
10 SCREEN 2
20 DRAW "M100,100E10F10G10H10"
30 GOTO 30
RUN
```

Vanuit het laatst getekende punt wordt eerst een lijn naar (100,100) getekend. Daarna wordt vanuit dit punt een ruitfiguur getekend; tien beeldpunten naar rechtsboven, tien beeldpunten naar rechtsonder, tien beeldpunten naar linksonder en tien beeldpunten naar linksboven.

B... de B staat voor BLANK. De eerst volgende tekenopdracht

wordt wel uitgevoerd maar de lijn wordt niet getrokken. In feite wordt met behulp van deze funktie een ander punt aangewezen als zijnde het laatst getekend. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 DRAW "BM100,100E10F10G10H10"
30 GOTO 30
RUN
```

Net als in het laatste voorbeeld wordt ook hier een ruitfiguur getekend. Echter, de eerste lijn (M100,100) die vanuit het laatst getekende punt naar (100,100) zou moeten lopen, wordt niet getekend.

N... de eerstvolgende tekenopdracht wordt niet van begin- naar eindpunt maar van eind- naar beginpunt uitgevoerd. Hierdoor blijft het laatst getekende punt vóór deze opdracht ook het laatst getekende punt ná deze opdracht. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 DRAW "BM100,100NU50ND50NL50NR50NE50NF50NG50NH50"
30 GOTO 30
RUN
```

Steeds wordt vanuit punt (100,100) een lijn getrokken naar diverse richtingen waardoor een sterfiguur ontstaat.

A... de letter A staat voor ANGLE. Een hoek van 0 graden (A0), 90 graden (A1), 180 graden (A2) of 270 graden (A3) kan worden gespecificeerd. Alle volgende tekenopdrachten (tot een volgend A-kommando) worden het aantal aangegeven graden naar links gedraaid. Bijvoorbeeld:

```
NEW
Ok
10 LINE INPUT "HOEK:";H$
20 SCREEN 2
30 DRAW "A0BM100,100A"+H$+"D20R20U20L20E10F10"
40 GOTO 40
RUN
```

Een hoek (0, 1, 2 of 3) kan worden ingegeven. Een eenvoudig tekeningetje van een huis wordt, gedraaid volgens de aangegeven hoek, getekend. RUN dit programma diverse malen met verschillende hoeken.

C... de C staat voor COLOR. Een voorgrondkleur kan worden opgegeven. Indien geen kleur wordt gespecificeerd, wordt de op dat moment actieve kleur gebruikt. Een eenmaal gespecificeerde kleur blijft geldig totdat binnen het DRAW-kommando een volgende kleur wordt gespecificeerd. Bijvoorbeeld:

```
15 LINE INPUT "KLEUR:";C$
25 DRAW "C"+C$
RUN
HOEK:
```

Bij het vorige voorbeeld werden twee programmaregels toegevoegd. Nu kan ook een kleurkode worden ingegeven waarna het tekeningetje in de gewenste kleur wordt uitgevoerd.

Onder MSX-1 kan slechts kleur 0 t/m kleur 15 (CO...C15) worden opgenomen. Onder MSX-2 kunnen achter de letters 'C' alle kleurnummers worden opgenomen die onder de ingestelde SCREEN-mode toegestaan zijn. Zie COLOR en SCREEN voor nadere gegevens met betrekking tot de betekenis van de kleurnummers.

S... de S staat voor SCALE. Een schaal kan worden opgegeven. Het achter de S opgenomen getal, gedeeld door vier, vormt het aantal werkelijk getekende puntjes ten opzichte van één geprogrammeerd puntje. Indien geen schaalfaktor wordt opgegeven, of een S0 of een S4 wordt opgegeven, komt een getekend puntje precies overeen met een geprogrammeerd puntje. Een S1 heeft tot gevolg dat een tekening vier maal zo klein wordt uitgevoerd terwijl een S16 juist tot gevolg heeft dat een tekening vier maal zo groot wordt uitgevoerd. Het M-kommando met absolute adressering (dus zonder gebruik van een plus- of minteken) wordt door het S-kommando als enige niet beïnvloed. Het S-kommando blijft geldig totdat een nieuw S-kommando wordt gegeven. Bijvoorbeeld:

```
16 LINE INPUT "SCHAAL:";S$
26 DRAW "S"+S$
RUN
HOEK:
```

Alleen regelnummers 16 en 26 werden aan het vorige voorbeeld toegevoegd. Nu kan ook een schaal worden opgegeven. Afhankelijk van de opgegeven schaal wordt het tekeningetje nu groter of kleiner afgebeeld.

X...; de tussen de X en de puntkomma vermelde alfanumerieke variabele wordt tijdens het tekenen tussengevoegd. Hierdoor kan een tekening die is gekodeerd in een aparte variabele, diverse malen worden herhaald. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 Q$="U10R10D10L10E10H5G5F10"
30 DRAW "S4C15A0BM99,99XQ$;"
40 GOTO 40
RUN
```

Een eenvoudig tekeningetje zoals in Q$ gekodeerd, wordt in regel 30 getekend nadat schaal, kleur, hoek en beginpunt zijn voorbereid.

=...; de tussen het gelijkteken en de puntkomma vermelde numerieke variabele wordt tijdens het tekenen als waarde ingevoegd. Hierdoor behoeft een string met een tekenopdracht niet moeizaam met behulp van onder meer het STR$-sleutelwoord worden opgebouwd maar kan onmiddellijk vanuit een numerieke waarde een afmeting of locatie worden bepaald. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "GROOTTE ";GR%:GT%=GR%/2
20 SCREEN 2
30 DRAW "S4C15A0BM99,99U=GR%;R=GR%;D=GR%;L=GR%;"
40 DRAW "E=GR%;H=GT%;G=GT%;F=GR%;"
50 GOTO 50
RUN
GROOTTE ?
```

Een grootte in aantal beeldschermpuntjes kan worden ingegeven. Een eenvoudig tekeningetje wordt op de aangegeven grootte gemaakt.

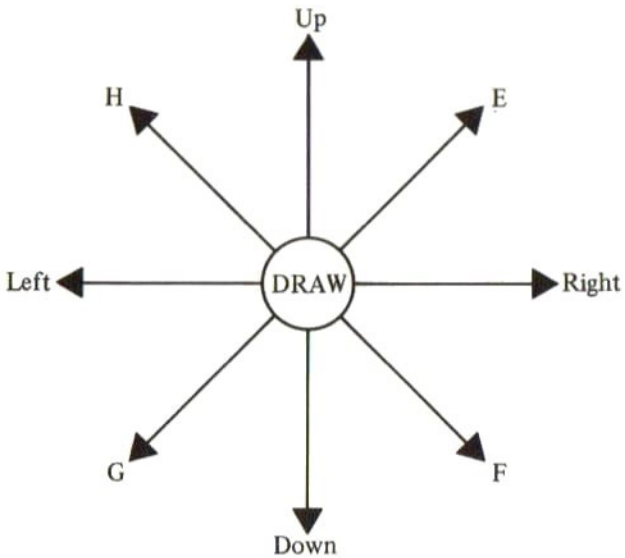
Algemene opmerkingen:

– DRAW gedraagt zich aan de randen van het beeldscherm niet geheel correct. Indien een lijn moet worden getrokken naar een buiten het beeld liggend punt, dan is het effect dat de lijn in werkelijkheid wordt getrokken naar de loodrechte projektie van dat niet bestaande punt op de betreffende beeldschermrand.

Maxima en minima:

– Voor numerieke variabelen die met behulp van het gebruik van een gelijkteken in een DRAW-string worden opgenomen geldt dat deze variabelen geheel in waarde dienen te zijn. Een eventueel aanwezige decimale fractie wordt genegeerd.
– Voor de in verband met het U, E, R, F, D, G, L, H en M gebruikte konstanten of variabelen gelden de volgende bepalingen:
variabele: minimaal –32768, maximaal 32767
konstante: minimaal –65535, maximaal 65535
deze waarden worden modulo 32768 behandeld, dat wil zeggen: indien een waarde groter is dan of gelijk is aan 32768 dan wordt 32768 van deze waarde afgetrokken en wanneer een waarde kleiner is dan of gelijk aan –32768 dan wordt 32768 bij deze waarde opgeteld.
– In verband met het S-kommando geldt een minimum van 0 en een maximum van 255 voor de betreffende konstante of variabele.
– In verband met het A-kommando geldt een minimum van 0 en een maximum van 3 voor de betreffende konstante of variabelen.
– In verband met het C-kommando geldt een minimum van 0 voor de betreffende variabele of konstante. Het maximum is 15 onder MSX-1 en hangt onder MSX-2 af van de gekozen scherminstelling.

Wanneer men het DRAW-kommando veelvuldig wil gaan gebruiken, is het gemakkelijk om een samenvatting van de uitgebreide mogelijkheden bij de hand te hebben. Om deze reden is op de volgende bladzijde een funktie-overzicht van het DRAW-kommando opgenomen. Het is misschien een goed advies om deze kaart over te tekenen of te kopiëren om apart bij de computer te kunnen gebruiken.



kode	betek.	funktie
M...,...	Move	ga naar het aangegeven punt en trek een lijn
B	Blank	voer het volgende kommando uit zonder een lijn te trekken
N	–	voer het volgende kommando in omgekeerde volgorde uit, ofwel vanaf het eindpunt naar het beginpunt
A.	Angle	teken de rest onder een hoek van 0 (A0), 90 (A1), 180 (A2) of 270 (A3) graden
C...	Color	teken de rest in de gekodeerde kleur
S...	Scale	teken de rest op de aangegeven schaal/4
X...;	–	voeg de vermelde stringvariabele in
=...;	–	voeg de waarde van de aangegeven numerieke variabele in
+/–	–	relatieve adressering

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **END is einde**

schrijfwijze

```
END
```

betekenis

Met het END-kommando geven we aan dat het programma is beeindigd. Meestal is het opnemen van een END-kommando slechts een formaliteit. Voorbeeld:

```
NEW
Ok
10 PRINT "DIT IS HET EINDE"
20 END
RUN
DIT IS HET EINDE
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **EOF is afkorting van end of file – einde van het bestand**

schrijfwijze

```
EOF(<KANAAL>)

<KANAAL>::=<N>

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

betekenis

Dit kommando wordt bij het OPEN-kommando reeds behandeld; zie aldaar.

moeilijkheidsgraad . . **normaal, zeer elementaire kennis van matrix-berekeningen is een voordeel**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ERASE is uitwissen**

schrijfwijze

```
ERASE<VARIABELE-NAAM>{,<VARIABELE-NAAM>}

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

betekenis

Het is noodzakelijk om eerst het DIM-kommando door te nemen.

Dit kommando heft een DIM-kommando voor de genoemde variabelen op. Voorwaarde is dat het betreffende DIM-kommando wel werd uitgevoerd.

Array's kunnen een enorm deel van het computergeheugen opeisen. Het is dus zaak om array's die niet meer worden gebruikt, op te heffen.

Een voorbeeld:

```
NEW
Ok
10 Y=0:Z=0
20 DIM A(20,20)
30 Y=FRE(0)
40 ERASE A
50 Z=FRE(0)
60 PRINT "A(20,20) KOST";Z-Y;"BYTES GEHEUGEN"
70 A(20,20)=12
RUN
A(20,20) KOST 3538 BYTES GEHEUGEN
Subscript out of range in 70
Ok
```

Uit dit voorbeeld blijkt, dat de A-array een ruimte van 3538 posities (bytes) inneemt. Na de ERASE is deze ruimte weer vrijgekomen. Programmaregel 70 veroorzaakt een fout; een bewijs dat de variabele A als array niet meer bekend is.

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **ERL is afkorting van error line-number –regel nummer van de fout**

schrijfwijze

ERL

betekenis

Deze systeemvariabele wordt behandeld onder ON ERROR GOTO; zie aldaar.

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **.ERR is afkorting van errornumber – foutnummer**

schrijfwijze

ERR

betekenis

Deze systeemvariabele wordt onder ON ERROR GOTO behandeld; zie aldaar.

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ERROR is fout**

schrijfwijze

```
ERROR<N>

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

betekenis

Dit kommando wordt onder ON ERROR GOTO behandeld; zie aldaar.

moeilijkheidsgraad . . **vrij moeilijk, kennis van logarithmische funkties is vereist**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **EXP is afkorting van exponent – macht**

schrijfwijze

```
EXP(<N>)

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

betekenis

Deze funktie geeft als resultaat de waarde *e* (2,7182818284...), verheven tot de macht die gevormd wordt door de waarde van de tussen haakjes vermelde uitdrukking.

Voorbeeld:

```
NEW
Ok
10 INPUT "WAARDE ";A
20 PRINT "E TOT DE MACHT A=";EXP(A)
30 GOTO 10
RUN
WAARDE ? 1
E TOT DE MACHT A= 2.7182818284588
WAARDE ? 200
E TOT DE MACHT A=
Overflow in 20
Ok
```

Merk op dat de maximale waarde van de numerieke uitdrukking ligt bij 145,06286085862 en de minimale waarde bij –75453,410912322. Deze laatste minimale waarde zou niet mogen bestaan maar is te wijten aan een klein foutje in MSX-basic.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **FIX is vastleggen, fixeren**

schrijfwijze

```
FIX(<N>)

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

betekenis

Het resultaat van deze funktie is een waarde, gelijk aan de waarde van de tussen haakjes vermelde numerieke uitdrukking maar dan ontdaan van decimale cijfers. FIX geeft dus altijd een geheel getal.

Voorbeeld:

```
NEW
Ok
10 LET A=12.5:LET B=-12.5
20 PRINT FIX(A);FIX(B)
RUN
 12 -12
Ok
```

moeilijkheidsgraad . . **normaal, enige elementaire kennis van de funktietheorie is een voordeel**
soort . . . . . **N-FUNKTIE of A-FUNKTIE**
afkomst . . . . . **FN is afkorting van function – funktie/bewerking**

schrijfwijze

```
FN<VARIABELE-NAAM>[(<VARIABELE>{,<VARIABELE>})]

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

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

betekenis

Deze funktie wordt bij DEF FN behandeld; zie aldaar.

SLEUTELWOORD # FOR-TO-STEP

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **FOR-TO-STEP is voor-tot-stapgrootte**

schrijfwijze

```
FOR <NUMERIEKE VARIABELENAAM>=<STARTWAARDE>TO<EINDWAARDE>[STEP...
<WAARDE>]

<STARTWAARDE>::=<N>

<EINDWAARDE>::=<N>

<STAPWAARDE>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

De FOR...TO...STEP combinatie is in samenwerking met het NEXT-kommando één van de krachtigste besturingsmogelijkheden van de computer.

Allereerst wijst FOR de startwaarde toe aan de genoemde variabele zoals een LET-kommando dat zou doen. Vervolgens vervolgt het programma op gebruikelijke wijze.
Wanneer nu vervolgens een bijbehorend NEXT-kommando wordt tegengekomen, dan wordt de betreffende variabele met de stapwaarde verhoogd. Hierna wordt gecontroleerd of de betreffende variabele ondertussen de eindwaarde niet is gepasseerd. Wanneer dit nog niet zo blijkt te zijn, wordt hervat met het kommando dat direkt op de FOR... TO...STEP volgt. Is de variabele deze eindwaarde echter wel gepasseerd, dan wordt het programma na de NEXT voortgezet.

Voorbeeld:

```
NEW
Ok
10 FOR I=1 TO 4 STEP 1.5
20 PRINT I
30 NEXT I
40 PRINT "KLAAR"
RUN
 1
```

```
 2.5
 4
KLAAR
Ok
```

De stapwaarde mag ook negatief zijn. Bijvoorbeeld:

```
NEW
Ok
10 FOR A=10 TO 0 STEP -1
20 PRINT A;
30 NEXT A
RUN
 10  9  8  7  6  5  4  3  2  1  0
Ok
```

Indien STEP wordt weggelaten, wordt automatisch een stapwaarde 1 aangenomen. Bijvoorbeeld:

```
NEW
Ok
10 FOR P=2 TO 5:PRINT P*P:NEXT:STOP
RUN
 4
 9
 16
 25
Break in 10
Ok
```

Uit het vorige voorbeeld blijkt dat bij de NEXT niet noodzakelijk een variabele behoeft te worden vermeld.

Het geheel vanaf FOR...TO...STEP tot en met NEXT noemt men wel de for-next loop (loop = lus). Twee of meer for-next loops mogen binnen elkaar plaatsvinden. Bijvoorbeeld:

```
NEW
Ok
10 FOR A=1 TO 3
20 FOR B=1 TO 3
30 PRINT A*B;
40 NEXT B:PRINT
50 NEXT A
RUN
 1  2  3
 2  4  6
 3  6  9
Ok
```

Kruisende loops zijn echter verboden.

GOED

```
NEW
Ok
10 REM GOEDE LOOP
20 FOR A=1 TO 5  <----.
30 FOR B=1 TO 5  <--. |
.                   | |
. (REST PROGRAMMA)  | |
.                   | |
90 NEXT B  <--------' |
100 NEXT A <----------'
```

FOUT

```
NEW
Ok
10 REM VERKEERDE LOOP
20 FOR A=1 TO 5  <----.
30 FOR B=1 TO 5  <--. |
.                   | |
. (REST PROGRAMMA)  | |
.                   | |
90 NEXT A  <--------+-'
100 NEXT B <-------'
```

Twee of meer NEXT-kommando's direkt ná elkaar mogen door één kommando worden vervangen. Voorwaarde is dan wel dat het betreffende NEXT-kommando de variabelen in de juiste volgorde noemt. Regel 90 en 100 van het bovenstaande goede voorbeeld mogen dus vervangen worden door de ene programmaregel:

```
90 NEXT B,A
```

In tegenstelling tot de meeste andere BASIC-dialecten voert MSX-basic een for-next loop altijd minimaal één keer uit, hoe de startwaarde en de eindwaarde zich ook tot elkaar verhouden.

Merk op dat in een FOR-NEXT loop geen tabelvariabele mag worden gebruikt als FOR-NEXT variabele.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **FRE is afkorting van free – vrij**

schrijfwijze

```
FRE(<U>)

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

betekenis

Deze funktie geeft als resultaat een getal dat de nog vrije geheugenruimte uitdrukt. Met de FRE-funktie kan

– de vrije ruimte in het geheugen, met uitzondering van het stringgeheugen, worden bepaald. Hiertoe dient de tussen haakjes op te nemen uitdrukking numeriek te zijn

– de vrije ruimte in het string-gebied worden bepaald. Hiertoe dient de tussen haakjes te vermelden uitdrukking alfanumeriek te zijn.

Bijvoorbeeld:

(zet eerst de computer uit en weer aan)

```
PRINT FRE(0)
 28815
Ok
PRINT FRE("")
 200
Ok
```

Uit het bovenstaande blijkt dat de computer een vrij geheugenruimtegebied heeft van 28815 tekens en dat in het vrije string-gebied nog plaats is voor 200 karakters.

De waarde van de tussen haakjes op te nemen uitdrukking is verder niet van belang.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **GET DATE – haal datum**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
GET DATE<ALFANUMERIEKE VARIABELE>[,A]

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

betekenis

Ook wanneer de spanning van uw MSX-2 computer is afgeschakeld, houdt deze datum en tijd bij. In de MSX-2 computer is namelijk een clock-chip ingebouwd die op een aparte accu-cel blijft doorlopen wanneer de computer is uitgeschakeld. Deze accu-cel wordt opgeladen zodra de MSX-2 machine wordt aangeschakeld.

Met de GET DATE kunt u de datum uit de clock-chip ophalen. Hiertoe dient u na het sleutelwoord GET DATE een alfanumerieke variabele op te geven. Na uitvoering van dit kommando staat de datum in deze alfanumerieke variabele en wel in de vorm "DD/MM/JJ" waarbij DD staat voor dagnummer van de maand, MM staat voor het maandnummer van het jaar en JJ staat voor het jaarnummer waarbij geen eeuwen worden opgenomen.

een voorbeeld:

```
NEW
Ok
10 GET DATE D$
20 PRINT D$
RUN
27/12/85
Ok
```

Met SET DATE kunt u de in de clock-chip opgeslagen datum veranderen; zie aldaar.

De clock-chip houdt rekening met het aantal dagen per maand en met schrikkeljaren. Een eenmaal goed ingestelde datum zal dan eigenlijk ook nooit meer behoeven te worden veranderd.

Wanneer na het GET DATE-kommando ook nog een komma, gevolgd door de letter A wordt opgenomen, dan wordt de apart in de clock-chip opgeslagen alarmdatum opgehaald. Ook deze datum kan met het SET DATE-kommando worden ingesteld. Wanneer de alarmdatum wordt opgevraagd, wordt alleen de alarm-dag gegeven. Bijvoorbeeld:

```
NEW
Ok
10 GET DATE Q$,A
20 PRINT Q$
RUN
31/00/00
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **GET TIME - haal tijd**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
GET TIME<ALFANUMERIEKE VARIABELE>[,A]

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

betekenis

Ook wanneer de spanning van uw MSX-2 computer is afgeschakeld, houdt deze datum en tijd bij. In de MSX-2 computer is namelijk een clock-chip ingebouwd die op een aparte accu-cel blijft doorlopen wanneer de computer is uitgeschakeld. Deze accu-cel wordt opgeladen zodra de MSX-2 machine weer wordt aangeschakeld.

Met de GET TIME kunt u de tijd uit de clock-chip ophalen. Hiertoe dient u na het sleutelwoord GET TIME een alfanumerieke variabele op te nemen. Na uitvoering van dit kommando staat in deze alfanumerieke variabele de tijd en wel in de vorm "UU:MM:SS" waarbij UU staat voor het uur (00/23), MM staat voor de minuten (00/59) en SS staat voor de seconden (00/59).

Een voorbeeld:

```
NEW
Ok
10 GET TIME T$
20 PRINT T$
RUN
16:16:43
Ok
```

De clock-chip zorgt ervoor, dat bij een dagwisseling (van 23:59:59 uur naar 00:00:00 uur) de datum automatisch wordt bijgesteld. De clock-chip is zeer nauwkeurig en behoeft na maanden hoogstens een enkele keer op de seconden te worden gelijkgezet. Dit gelijkzetten gebeurt met behulp van het kommando SET TIME; zie aldaar.

Wanneer na het GET TIME-kommando ook nog een komma, gevolgd door de letter A wordt opgenomen, dan wordt een apart in de clock-chip opgeslagen alarm-tijd opgehaald. Ook deze tijd kan met behulp van het SET TIME-kommando worden ingesteld. Wanneer de alarm-tijd wordt opgevraagd, worden alleen het alarm-uur en de alarm-minuten gegeven. Bijvoorbeeld:

```
NEW
Ok
10 GET TIME A$,A
20 PRINT A$
RUN
15:24:00
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **GOSUB is afkorting van go to subroutine – ga naar onderprogramma (subroutine)**

schrijfwijze

```
GOSUB<REGELNUMMER>

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

betekenis

Het is noodzakelijk dat eerst GOTO wordt bestudeerd.
In eerste instantie werkt de GOSUB hetzelfde als de GOTO; het programma gaat verder met het genoemde regelnummer. Echter, bij GOSUB doet de computer nog meer. De computer 'onthoudt' namelijk het punt *waarvandaan* wordt gesprongen, het zogenaamde terugkeeradres in een stukje geheugen, de stack (stapel) genaamd. Wanneer later een RETURN-kommando wordt gegeven (RETURN = keer terug), dan haalt de computer dit regelnummer weer van die stack en gaat door met het kommando dat direkt komt na de GOSUB (dus één stap verder dan wat de stack aangeeft).

Het voordeel van een GOSUB is, dat gedeelten van programmatuur meerdere malen kunnen worden gebruikt op diverse plaatsen in het programma. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "HOEVEEL IS 2+3 ";U
20 IF U<>5 THEN GOSUB 500:GOTO 10
30 INPUT "EN HOEVEEL IS 5+6 ";U
40 IF U<>11 THEN GOSUB 500:GOTO 30
50 PRINT "GOEDZO, KLAAR!":STOP
500 REM SUBROUTINE FOUTMELDING
510 PRINT "NEE HOOR, DE UITKOMST IS NIET";U
520 PRINT "NOG EEN KEER PROBEREN...":RETURN
RUN
HOEVEEL IS 2+3 ? 4
NEE HOOR, DE UITKOMST IS NIET 4
NOG EEN KEER PROBEREN...
HOEVEEL IS 2+3 ? 5
EN HOEVEEL IS 5+6 ? 12
```

```
NEE HOOR, DE UITKOMST IS NIET 12
NOG EEN KEER PROBEREN...
EN HOEVEEL IS 5+6 ? 11
GOEDZO, KLAAR!
Break in 50
Ok
```

We zien dat het programmadeel vanaf regel 500 een keer vanuit regel 20 en een keer vanuit regel 40 werd aangeroepen. Het programmadeel waarin de foutmelding wordt afgedrukt, behoeft door de GOSUB-mogelijkheid maar één maal in het programma te worden opgenomen.

Een programmagedeelte dat via een GOSUB wordt aangeroepen, noemt men meestal een onderprogramma of een *subroutine*.

Een subroutine mag weer een andere subroutine aanroepen. In dat geval wordt het te onthouden terugkeeradres bovenop de stack geplaatst. Dit heeft tot gevolg dat een RETURN altijd een terugkeer veroorzaakt naar het gedeelte na de laatste GOSUB.
Voorbeeld:

```
NEW
Ok
10 GOSUB 100:PRINT "TERUG IN HOOFDPROGRAMMA"
20 STOP
100 GOSUB 500:PRINT "TERUG IN EERSTE SUBROUTINE"
110 RETURN
500 PRINT "TWEEDE SUBROUTINE":RETURN
RUN
TWEEDE SUBROUTINE
TERUG IN EERSTE SUBROUTINE
TERUG IN HOOFDPROGRAMMA
Break in 20
Ok
```

Ga na dat de regelnummers waar de computer achtereenvolgens mee bezig is, de nummers 10, 100, 500, 100, 110, 10 en 20 zijn.

Een goede BASIC-programmeur stelt zijn programma uit vele subroutines samen. In elke subroutine verzorgt hij of zij dan een compleet afgerond geheel. Uiteindelijk stelt deze programmeur dan een HOOFDPROGRAMMA samen waarin al deze subroutines met GOSUB-kommando's worden aangesproken. Wanneer het hoofdprogramma, dat bijna alleen uit GOSUB's bestaat, daarbij ook nog met REM-kommando's goed van kommentaar is voorzien, ontstaat een programma dat

ook bij grote afmetingen voor iedereen nog zeer goed te volgen is. Een voorbeeld (niet intikken, dat heeft geen zin) van hoe zo'n hoofdprogramma er dan uit zou kunnen zien:

```
10 REM **********************
20 REM HOOFDPROGRAMMA ADRESSEN
30 REM **********************
40 GOSUB 1000'BEELDSCHERM SCHOON & KOP
50 GOSUB 2000'PROGRAMMAKEUZE
60 REM KEUZE 1...AANMAKEN ADRESSEN
70 IF KEUZE=1 THEN GOSUB 3000:GOTO 40
80 REM KEUZE 2...ADRESSEN OP BEELD
90 IF KEUZE=2 THEN GOSUB 4000:GOTO 40
100 REM KEUZE 3...ADRESSEN OP PAPIER
110 IF KEUZE=3 THEN GOSUB 5000:GOTO 40
120 REM KEUZE 4...EINDE PROGRAMMA
130 PRINT "EINDE PROGRAMMA":BEEP:STOP
140 REM *** EINDE HOOFDPROGRAMMA ***
1000 REM SUBROUTINE 1, BEELD SCHOON/KOP
.
.
(etcetera)
```

Indien men op deze wijze programmeert, is men struktureel bezig. Bij deze gestruktureerde vorm van programmeren is de GOSUB-RETURN combinatie onontbeerlijk.

Indien men ten onrechte niet met RETURN uit een subroutine terugkeert, kunnen na verloop van tijd vreemde situaties ontstaan. Elke keer dat een GOSUB wordt uitgevoerd, wordt namelijk een nieuw terugkeeradres op de stack bijgeplaatst. Hierdoor wordt er steeds een iets groter stukje geheugen door de computer bezet. Uiteindelijk kan het overblijvende geheugendeel zo klein blijken te zijn dat een foutmelding volgt. Een eenvoudig voorbeeld van dit verkeerde GOSUB-gebruik:

```
NEW
Ok
10 GOSUB 10
RUN
Out of memory in 10
Ok
```

Het bovenstaande programma vult continu de stack aan zonder dat er weer terugkeeradressen door RETURN worden afgehaald. Na ongeveer 4 seconden volgt een foutmelding; het hele geheugen is opgebruikt aan stack-ruimte.

Het is een gouden regel dat een subroutine ALTIJD met een GOSUB moet worden aangesproken en ALTIJD met een RETURN moet worden verlaten.

Indien tijdens het samenstellen van een programma de situatie ontstaat waarbij een normale RETURN niet meer kan worden gebruikt, kan een RETURN, gevolgd door een regelnummer, worden gebruikt. Dit *bijzonder weinig fraaie* gebruik van RETURN werkt als een GOTO met dit verschil dat een terugkeeradres van de stack wordt gehaald zonder dat er iets mee wordt gedaan; het geheugen wordt niet overbelast terwijl eventueel volgende RETURN-kommando's goed blijven werken (naar de juiste aanroeplocaties terugkeren). Voorbeeld:

```
NEW
Ok
10 GOSUB 100:STOP
100 INPUT "WAARDE TUSSEN 1 EN 10 ";A
110 IF A=INT(A) AND A>0 AND A<11 THEN RETURN
120 RETURN 10
RUN
WAARDE TUSSEN 1 EN 10 ? 2.5
WAARDE TUSSEN 1 EN 10 ? 3
Break in 10
Ok
```

(via een RETURN 10 wordt een verbetering geforceerd)

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **GOTO is samentrekking van go to – ga naar**

schrijfwijze

```
GOTO<REGELNUMMER>

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

betekenis

Met GOTO kan de programmaloop worden veranderd. Het regelnummer dat achter GOTO wordt vermeld, is het regelnummer waar de computer verder gaat. Een GOTO mag naar een eerder- of verdergelegen regelnummer verwijzen. Ook mag GOTO naar zichzelf verwijzen.

Voorbeeld:

```
NEW
Ok
10 GOTO 10
RUN
```

– Het programma staat in een eeuwigdurende lus en kan slechts met een CONTROL-STOP worden onderbroken.

```
NEW
Ok
10 REM DIT PROGRAMMA BLIJFT UW NAAM
20 REM CONTINU AFDRUKKEN.
30 INPUT "HOE HEET U ";NAAM$
40 PRINT NAAM$;"-";
50 GOTO 40
RUN
HOE HEET U ? KAREL
KAREL-KAREL-KAREL-KA...
L-KAREL-KAREL-KA...
REL-KAREL-KA...
```

– Het gehele beeldscherm wordt vol gezet, onderbreken met CONTROL-STOP.

Merk op dat GOTO standaard onder funktie-toets 3 aanwezig is.

moeilijkheidsgraad .. **vrij moeilijk, vereist kennis van talstelsels en algemene principes van het computergeheugen**
soort ..... **A-FUNKTIE**
afkomst ..... **HEX$ is afkorting van hexadecimal string – hexadecimale (zestientallige) string**

schrijfwijze

```
HEX$(<N>)

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

betekenis

Deze funktie geeft de hexadecimale waarde van de tussen haakjes vermelde uitdrukking weer in string-vorm. De waarde van de numerieke uitdrukking dient te liggen tussen –32769 en 65536. Een eventuele decimale fractie wordt verwaarloosd.
Indien de waarde van de numerieke uitdrukking negatief is, geeft HEX$ een hexadecimale representant, samengesteld volgens de tweecomplementmethode. Bijvoorbeeld:

```
PRINT HEX$(-32768)
8000
Ok
PRINT HEX$(65535)
FFFF
Ok
PRINT HEX$(-1)
FFFF
Ok
PRINT HEX$(1023)
3FF
Ok
```

SLEUTELWOORD IF-THEN/GOTO-ELSE

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **IF-THEN/GOTO-ELSE is als-dan/ga naar - anders (goto – samentrekking van go to)**

schrijfwijze

```
       THEN<DIREKTE OPDRACHT>
       ----------------------
IF<N>  THEN<REGELNUMMER>
       ----------------------
       GOTO<REGELNUMMER>
```

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

<REGELNUMMER>::=<ZIE ALGEMENE SPECIFICATIES>

<DIREKTE OPDRACHT>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Voor goed begrip van IF...THEN/GOTO...ELSE dienen de hoofdstukken 5, 6 en 7 terdege te zijn doorgenomen.

Deze kombinatie van sleutelwoorden biedt de mogelijkheid om numerieke of alfanumerieke relaties te onderzoeken en daarop beslissingen te nemen. Alleen indien de tussen IF en THEN/GOTO opgenomen numerieke uitdrukking WAAR is (of ongelijk aan nul is) worden de kommando's na THEN uitgevoerd of wordt met het regelnummer achter GOTO vervolgd.

Voorbeeld:

```
NEW
Ok
10 INPUT "HOEVEEL IS 2 MAAL 5 ";UITKOMST
20 IF UITKOMST=10 GOTO 100
30 IF UITKOMST<10 THEN PRINT "TE LAAG":GOTO 10
40 PRINT "TE HOOG":GOTO 10
100 PRINT "HOERA!!! GOED!!!"
110 STOP
RUN
HOEVEEL IS 2 MAAL 5 ? 9
TE LAAG
```

```
HOEVEEL IS 2 MAAL 5 ? 112.5
TE HOOG
HOEVEEL IS 2 MAAL 5 ? 10
HOERA!!! GOED!!!
Break in 110
Ok
```

Ook ingewikkelder zaken kunnen worden afgevraagd, bijvoorbeeld met behulp van de logische bewerkingen:

```
NEW
Ok
10 INPUT "DE LUCHT IS ";KLEUR$
20 INPUT "EN HET GRAS RUIKT ";GEUR$
30 IF KLEUR$="BLAUW" AND GEUR$="FRIS" THEN PRINT "GOEDZO!!":
STOP
40 PRINT "NOG EEN KEER PROBEREN":GOTO 10
RUN
DE LUCHT IS ? GEEL
EN HET GRAS RUIKT ? FRIS
NOG EEN KEER PROBEREN
DE LUCHT IS ? BLAUW
EN HET GRAS RUIKT ? FRIS
GOEDZO!!
Break in 30
Ok
```

Met de ELSE-optie kan worden gespecificeerd welke akties dienen te worden ondernomen indien de uitdrukking tussen IF en THEN/GOTO NIET WAAR (gelijk aan 0) is. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "HOE HEET U ";NAAM$
20 IF LEN(NAAM$)=5 THEN PRINT "UW NAAM IS PRECIES 5 LETTERS
LANG" ELSE PRINT "HALLO ";NAAM$;" HOE GAAT HET ERMEE ?":GOTO
 10
RUN
HOE HEET U ? JAN
HALLO JAN HOE GAAT HET ERMEE ?
HOE HEET U ? FRITS
UW NAAM IS PRECIES 5 LETTERS LANG
Ok
```

Merk op dat regel 30 en 40 uit het voorlaatste voorbeeld kunnen worden vervangen door één regel:

```
30 IF KLEUR$="BLAUW" AND GEUR$="FRIS" THEN PRINT "GOEDZO!!":
STOP ELSE PRINT "NOG EEN KEER PROBEREN":GOTO 10
```

moeilijkheidsgraad ..... **zeer eenvoudig**
soort ..... **SYSTEEMVARIABELE**
afkomst ..... **INKEY$ is afkorting van input key string – (vrij) ingetoetst karakter**

schrijfwijze

```
INKEY$
```

betekenis

Deze funktie geeft als resultaat het karakter van de laatst ingegeven toets. Indien niets werd ingegeven, geeft INKEY$ een lege waarde als resultaat. Bijvoorbeeld:

```
NEW
Ok
10 LET A$=INKEY$:IF A$="" THEN 10
20 PRINT "U GAF ";A$;" IN."
30 GOTO 10
RUN
```

(De computer wacht. Zodra u een toets indrukt, meldt hij dat waarna hij weer wacht. Programma onderbreken met CONTROL-STOP).

N.B.: Zodra INKEY$ één maal is gebruikt, is het betreffende karakter verdwenen. Daarom moet INKEY$ steeds eerst in een alfanumerieke variabele worden overgebracht (zie regel 10) en daarna pas worden afgevraagd.

Op de CONTROL-BREAK-intoetsing na, worden alle toetsaanslagen bij het gebruik van INKEY$ geregistreerd. Voor controletoetsen wordt ook een karakter gegeven. Om dit karakter te kunnen bestuderen, kunnen we de ASCII-waarde van dit karakter als volgt afvragen:

```
NEW
Ok
10 LET A$=INKEY$:IF A$="" THEN 10
20 PRINT "ASCII-WAARDE:";ASC(A$)
30 GOTO 10
RUN
ASCII-WAARDE: 13   (de RETURN-toets werd ingedrukt)
```

```
ASCII-WAARDE: 27
ASCII-WAARDE: 24
```
(de ESC-toets werd ingedrukt)
(de SELECT-toets werd ingedrukt)

(etcetera; probeer hoofdletters, kleine letters, controletoetsen en ook eens de funktietoetsen. Controleer het een en ander met hoofdstuk 15.

moeilijkheidsgraad . . **zeer moeilijk, kennis van de funkties van de poorten van het computersysteem is vereist**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **INP is afkorting van input – invoer, ingave**

schrijfwijze

```
INP(<POORT>)

<POORT>:::=<N>

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

betekenis

Een poortnummer dient te worden gespecificeerd. De waarde van het byte dat op het moment van aanspreken op de poort-buffer staat, vormt het resultaat van deze funktie.

Het poortnummer mag niet kleiner zijn dan –32768 en niet groter dan 65535. Een eventuele decimale factor wordt verwaarloosd. Indien het poortnummer negatief is, wordt 65536 bij dit nummer opgeteld voordat de betreffende poort wordt aangesproken.

Bijvoorbeeld:

```
NEW
Ok
10 PRINT INP(12)
RUN
 255
Ok
```

Voor een goed gebruik van INP is een gedetailleerde kennis van de hardware-opbouw van een MSX-computer noodzakelijk. Daarbij dienen de diverse componenten van een MSX-computer ook software-technisch te worden beheerst. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop niet verder ingegaan.

moeilijkheidsgraad . . . **normaal, in de eerste betekenis vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **INPUT is invoer/ingave**

schrijfwijze

```
                #<KANAAL>,
INPUT ---------------------- <VARIABELE>{,<VARIABELE>}
      ["<SATELLIETTEKST>";]

<SATELLIETTEKST>:::=<ALFANUMERIEKE KONSTANTE>

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

<KANAAL>:::=<N>

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

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

betekenis

Met het INPUT-kommando kunnen gegevens tijdens de programmaloop aan de computer worden toegevoerd.

eenvoudig gebruik

Met het INPUT-kommando kunnen gegevens via het toetsenbord worden ingevoerd. Deze worden dan in een variabele opgeslagen. Voorbeeld:

```
NEW
Ok
10 INPUT A
20 PRINT A
30 STOP
RUN
? 12.34                  – De computer geeft een vraagteken. Een
 12.34                     waarde dient te worden ingetoetst.
Break in 30
Ok
```

De in te voeren gegevens kunnen numeriek of alfanumeriek zijn. Bij invoer van alfanumerieke gegevens dient het aanhalingsteken als eerste karakter te worden vermeden omdat MSX-basic hierop niet correct

reageert. Voorbeeld:

```
NEW
Ok
10 INPUT A$
20 PRINT  "HALLO ";A$
RUN
? FRANS
HALLO FRANS
Ok
```

Indien meerdere gegevens dienen te worden ingevoerd, kan dat in één INPUT-kommando gebeuren. De variabelen achter INPUT dienen dan met een komma van elkaar te zijn gescheiden. Voorbeeld:

```
NEW
Ok
10 INPUT A,B,C
20 PRINT A;B;C
30 GOTO 10
RUN
? 1,2,3.5
 1   2   3.5
? 22
?? 33
?? 44
 22   33   44
? 1,2,3,4
?Extra ignored
 1   2   3
?
```

– De gegevens mogen in één keer achter elkaar worden ingegeven, gescheiden door een komma.

– De gegevens mogen ook elk met een RETURN worden afgesloten. Met een dubbel vraagteken vraagt de computer dan een volgende ingave totdat alle variabelen zijn gevuld.

– Indien te veel waarden worden ingevoerd geeft de computer eerst een melding dat het teveel aan ingevoerde gegevens wordt overgeslagen.

De komma is het scheidingsteken bij INPUT. Daarom kan een komma als *teken* via de input nooit in een stringvariabele worden overgeplaatst.

Een INPUT-kommando mag worden voorzien van een begeleidende tekst. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "HOE HEET U ";NAAM$
20 PRINT "HALLO ";NAAM$
30 STOP
RUN
HOE HEET U ? GERARD
HALLO GERARD
Break in 30
Ok
```

Indien voor MSX-basic onbegrijpelijke ingaven worden gedaan, vraagt de computer u om een herhaalde ingave. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "GEEF WAARDE IN ";A
20 PRINT "DANK U":STOP
RUN
GEEF WAARDE IN ? GERARD
?Redo from start
GEEF WAARDE IN ? 12.3
DANK U
Break in 20
Ok
```

Indien bij een INPUT slechts een RETURN-toets wordt ingegeven, dan behoudt of behouden de genoemde variabele(n) de vorige waarde(n). Voorbeeld:

```
NEW
Ok
10 INPUT A:PRINT A:GOTO 10
RUN
? 11.4
 11.4
? 12.5
 12.5
?                    ──────► Alleen een RETURN wordt ingegeven.
 12.5                ──────► En de oude waarde bleef behouden.
?
```

Gevorderd gebruik

Zie hiervoor ook de beschrijving van het PRINT-kommando, onderdeel gevorderd gebruik: schrijven naar randapparatuur alsmede het OPEN, CLOSE en MAXFILES-kommando.

Indien een kanaalnummer is gespecificeerd, kan de aanvoer van gegevens vanaf een ander apparaat dan alleen het toetsenbord plaatsvinden; bijvoorbeeld vanaf een cassetterecorder. Voorwaarde is dat de gegevens op precies dezelfde manier als via het toetsenbord worden aangevoerd. Dit betekent dat de diverse gegevens die op één regel staan, ook bijvoorbeeld door een komma dienen te worden gescheiden. In feite kan men stellen dat de gegevens naar het betreffende randapparaat (b.v. een cassetterecorder) dient te PRINTen zoals het INPUT-kommando deze later weer vereist. Een voorbeeld (zie ook het voorbeeld onder

PRINT):

eerste programma:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
20 PRINT #1,"DEZE GEGEVENS WORDEN"
30 PRINT #1,"STRAKS WEER OPGEHAALD"
40 PRINT #1,123,",";-12.33:CLOSE
```

– Tussen de twee numerieke gegevens op regelnummer 40 staat een komma, omdat zij straks door één INPUT weer worden opgehaald.

– Nu de cassetterecorder op opnemen zetten.

RUN	– De cassetterecorder begint te lopen.
Ok	– De cassetterecorder stopt weer; de gegevens zijn geschreven.

– Cassetterecorder nu eventueel uitzetten.

tweede programma:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
20 INPUT #1,A$:PRINT A$:INPUT #1,A$:PRINT A$
30 INPUT #1,A,B:PRINT A;B:CLOSE:STOP
RUN
```

– De cassetteband terugspoelen en de recorder op afspelen zetten.

```
DEZE GEGEVENS WORDEN
STRAKS WEER OPGEHAALD
 123 -12.33
Break in 30
Ok
```

Indien u dit voorbeeld uitprobeert, zorg er dan voor dat het volume van de cassetterecorder op ongeveer 80 procent staat. De MSX-standaard schrijft geen controles voor op de integriteit van gegevens. Afhankelijk van de kwaliteit en instelling van de cassetterecorder kunnen gegevensverminkingen optreden.

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open in dat geval "MEM:TEST" in plaats van "CAS:TEST". Zie ook CALL MEMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **INPUT$ is afkorting van input string – ingave string**

schrijfwijze

```
INPUT$(<AANTAL TEKENS>[,<KANAAL>])

<AANTAL TEKENS>::=<N>

<KANAAL>::=<N>

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

betekenis

Deze funktie geeft als resultaat de ingevoerde tekens nadat het vermelde aantal is bereikt. Deze ingevoerde tekens kunnen van een randapparaat komen (cassetterecorder of disk) maar ook van het toetsenbord.

Eenvoudig gebruik

Indien de in te voeren tekens van het toetsenbord dienen te komen, kan het kanaalnummer worden weggelaten. Bijvoorbeeld:

```
NEW
Ok
10 PRINT INPUT$(5)          (nu kunnen tekens worden ingegeven; deze worden in eerste instantie
RUN                          niet op het scherm weergegeven)
TEST.
Ok
```

Pas nadat er vijf tekens zijn ingegeven, verschijnen deze tekens op het beeldscherm en is het programma ten einde.

Gevorderd gebruik

Zie ook het OPEN, CLOSE en MAXFILES-kommando.
Indien een kanaalnummer werd gespecificeerd, kan de invoer van tekens ook via dit kanaal plaatsvinden. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS #1
20 PRINT #1,"DIT IS EEN TEST"
30 CLOSE
```

(plaats nu een cassetteband in de recorder en zet deze op opnemen)

```
RUN
Ok
```

(na enige tijd is de band geschreven)

(zet nu de recorder uit)

(tweede programma)

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
20 PRINT INPUT$(5,1)
30 PRINT INPUT$(3,1)
RUN
```

(spoel nu de band terug en zet de recorder op afspelen)

```
DIT I
S E
Ok
```

(eerst worden de eerste vijf en later de volgende drie karakters afgedrukt op het beeldscherm).

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open indat geval "MEM:TEST" in plaats van "CAS :TEST". Zie ook CALL MEMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **INSTR is afkorting van in string – binnen string**

schrijfwijze

```
INSTR([<STARTPOSITIE>,]<TE ONDERZOEKEN STRING>,<ZOEKSTRING>)

<STARTPOSITIE>::=<N>

<ZOEKSTRING>::=<A>

<TE ONDERZOEKEN STRING>::=<A>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Deze funktie geeft als resultaat een positiewaarde binnen de te onderzoeken string van de op te zoeken string. Bijvoorbeeld:

```
PRINT INSTR("LIESJE LEERDE LOTJE LOPEN","LEERDE")
 8
Ok
```

Bovenstaande voorbeeld laat zien hoe het woord LEERDE binnen de zin LIESJE LEERDE LOTJE LOPEN kan worden opgezocht. Het resultaat is de waarde 8; de positie van de eerste letter van de gevonden string.

Wanneer de te onderzoeken string niet de zoekstring bevat, levert de INSTR-funktie de waarde 0 op. Bijvoorbeeld:

```
PRINT INSTR("IN EEN GROEN GROEN GROEN","BLAUW")
 0
Ok
```

Wanneer slechts een gedeelte van de te onderzoeken string dient te worden onderzocht, dient een startpositie te worden opgegeven die niet kleiner mag zijn dan de waarde 1 en niet groter mag zijn dan 255.

Een eventuele decimale fractie wordt verwaarloosd. Bijvoorbeeld:

```
PRINT INSTR(3,"DE KLOK EN DE KLEPEL","DE")
 12
Ok
```

Indien in dit voorbeeld de startpositie (3) niet was opgenomen, zou het resultaat niet 12 maar 1 zijn (het eerste woord is al meteen gelijk aan DE).

Een laatste voorbeeld:

```
NEW
Ok
10 LINE INPUT "ZIN:";ZIN$
20 LET ST=1
30 PRINT "HET WOORDJE -DE- KOMT VOOR"
40 PRINT "OP POSITIE";
50 LET PS=INSTR(ST,ZIN$,"DE")
60 IF PS=0 THEN PRINT:GOTO 10
70 PRINT PS;:LET ST=PS+1:GOTO 50
RUN
ZIN:ALS DE KAT VAN HUIS IS, DANSEN DE MUIZEN OP TAFEL
HET WOORDJE -DE- KOMT VOOR
OP POSITIE 5  32
ZIN:DE DEERNEN DEELDEN DE DEENSE DEKEN
HET WOORDJE -DE- KOMT VOOR
OP POSITIE 1  4  12  16  20  23  30
ZIN:
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **INT is afkorting van integer – gehele waarde**

schrijfwijze

```
INT(<N>)

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

betekenis

Het resultaat van deze funktie is de grootste gehele waarde, kleiner dan of gelijk aan de waarde van de tussen haakjes vermelde numerieke uitdrukking. Om het verschil met FIX aan te geven, zijn beide in het voorbeeld verwerkt:

```
NEW
Ok
10 LET A=12.5:LET B=-12.5
20 PRINT INT(A);FIX(A)
30 PRINT INT(B);FIX(B)
RUN
 12  12
-13 -12
Ok
```

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **INTERVAL is pauze**

schrijfwijze

```
          ON
          ----
INTERVAL  OFF
          ----
          STOP
```

betekenis

Voorafgaand aan de behandeling van dit kommando dient het ON KEY GOSUB kommando grondig te zijn doorgenomen.

Behalve dat we in MSX-basic een programma tijdelijk kunnen onderbreken bij het intoetsen van een funktietoets, kunnen we het programma ook periodiek laten onderbreken op tijdsduur.

Wanneer we een mogelijke onderbreking van het programma op tijdsduur willen activeren, gebruiken we het kommando:

```
INTERVAL ON
```

Deactivering van deze onderbreking geschiedt met behulp van INTERVAL OFF.
Indien we een onderbreking willen laten 'onthouden' door de computer zonder dat er direkt actie op volgt, gebruiken we INTERVAL STOP.
Een eventueel geregistreerde wens tot onderbreking wordt dan bij een INTERVAL ON onmiddellijk gehonoreerd.

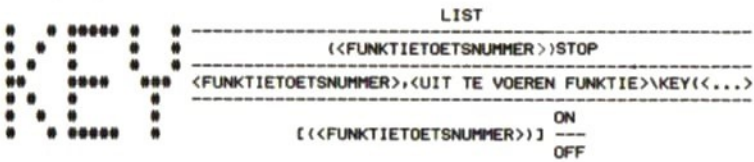
Een INTERVAL STOP heeft geen zin indien deze niet werd voorafgegaan door een INTERVAL ON.

Voor een zinvol voorbeeld: zie ON INTERVAL GOSUB.

INTERVAL vindt zijn toepassing bij uitstek in spelprogramma's en educatieve programma's (tijdsbewaking).

moeilijkheidsgraad . . normaal, wat lastig door de vele uiteenlopende mogelijkheden
soort . . . . . KOMMANDO
afkomst . . . . . KEY is toets

schrijfwijze



betekenis

Het KEY-sleutelwoord heeft vele verschillende betekenissen. Hieronder volgt een beschrijving per betekenis:

Eerste betekenis: het uitlijsten van de inhoud van de funktietoetsen

Door ingave via KEY LIST verschijnen onder elkaar op het beeldscherm de funkties die onder de funktietoetsen liggen opgeslagen. Eventuele controlekarakters (zoals de in CHR$(13) gekodeerde RETURN toets, zie de derde betekenis) worden als spaties afgedrukt.
Wanneer we KEY LIST op een zojuist aangeschakelde MSX-computer gebruiken, krijgen we de volgende lijst:

```
KEY LIST
color
auto
goto
list
run
color 15,4,4
cload"
cont
list.
 run
Ok
```

Wanneer we de funktietoetsen zelf indrukken, verschijnen de afgebeelde funkties daadwerkelijk. Zo heeft funktietoets F10 tot gevolg dat

het beeldscherm eerst wordt schoongemaakt (controlekarakter vóór run; zie de afgebeelde spatie) waarna het programma wordt opgestart.

Tweede betekenis; het ophouden van een onderbreking via een funktietoets

Deze betekenis, te activeren met de sleutelwoorden KEY en STOP, wordt bij de behandeling van ON KEY GOSUB behandeld; zie aldaar.

Derde betekenis; het veranderen van de funktie van een funktietoets

Via het KEY sleutelwoord, gevolgd door een numerieke uitdrukking die niet met een haakje begint, gevolgd door een komma en een alfanumerieke uitdrukking, kan de funktie van een funktietoets worden veranderd. Bijvoorbeeld:

```
KEY 2,"TRON"
Ok
```

Na het intikken van dit kommando heeft funktietoets 2 de funktie TRON verkregen. Wanneer we hierna deze funktietoets indrukken, verschijnt de tekst TRON. Er dient nog wel een RETURN te worden ingegeven om het kommando te bevestigen.

De RETURN-toets correspondeert met een ASCII-kode 13. Deze toets kan als karakter dus worden meegegeven in een funktie. Bijvoorbeeld:

```
KEY 2,"TRON"+CHR$(13)
Ok
```

Wanneer we nu funktietoets 2 indrukken, verschijnt wederom het TRON-kommando maar wordt daarbij ook meteen een return gegeven. Direkt na de funktietoets verschijnt de Ok-melding; het TRON-kommando is uitgevoerd.

Wanneer we een funktie willen creëren die een TRON en vervolgens een RUN activeert, kunnen we dat bijvoorbeeld als volgt bewerkstelligen:

```
KEY 2,"TRON"+CHR$(13)+"RUN"+CHR$(13)
Ok
```

Het ligt voor de hand dat we funktietoets 1 als volgt inrichten:

```
KEY 1,"TROFF"+CHR$(13)+"RUN"+CHR$(13)
Ok
```

Indien we funktietoets 3 de funktie 'maak het beeld schoon en doe een LIST' willen geven, kan dat als volgt:

```
KEY 3,"CLS"+CHR$(13)+"LIST"+CHR$(13)
Ok
```

of

```
KEY 3,CHR$(12)+"LIST"+CHR$(13)
Ok
```

ASCII-kode 12 heeft tot effect dat het beeld wordt schoongemaakt.

De numerieke uitdrukking voor de komma geeft het funktietoetsnummer aan en dient een gehele, positieve waarde te bevatten, kleiner dan 11. Indien de numerieke uitdrukking een gebroken waarde bevat, wordt de decimale fractie verwaarloosd.

Vierde betekenis; het aan- of uitzetten van de onderbrekingsmogelijkheid door middel van een funktietoets

Deze betekenis, die met behulp van de sleutelwoorden KEY,ON en OFF kan worden geactiveerd, wordt onder ON KEY GOSUB behandeld; zie aldaar.

Vijfde betekenis; het aan- of uitzetten van de funktietoetsregel

De MSX-computer heeft standaard de onderste beeldschermregel (in de alfanumerieke toestand) gereserveerd voor het presenteren van (een gedeelte van) de funkties van de funktietoetsen. Zolang deze funkties daar worden geprojekteerd, kan die onderste regel niet worden gebruikt door het programma.

Merk op dat de eerste vijf funkties zijn afgebeeld. Door de SHIFT-toets in te drukken, verschijnt onder in het beeld de tweede set van vijf funkties.

Door eenvoudigweg het kommando

```
KEY OFF
```

in te toetsen, kunnen we de onderste regel leeg maken en vrij maken voor gebruik in een programma. Door het

```
KEY ON
```

kommando halen we de funktietoetsregel weer te voorschijn en wordt deze regel weer verboden terrein voor het programma.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **LEFT$ is afkorting van left part of string – linker gedeelte van de string**

schrijfwijze

```
LEFT$(<BASISSTRING>,<AANTAL TEKENS>)

<BASISSTRING>::=<A>

<AANTAL TEKENS>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Deze funktie geeft van een alfanumerieke uitdrukking het linkergedeelte als resultaat. Hoe groot dit linkergedeelte is, wordt bepaald door het opgegeven aantal tekens. Bijvoorbeeld:

```
PRINT LEFT$("MSX-BASIC",4)
MSX-
Ok
```

Het aantal tekens moet groter dan of gelijk aan nul zijn en mag de waarde 255 niet overschrijden. Een eventuele decimale fractie wordt verwaarloosd.

Nog een voorbeeld:

```
NEW
Ok
10 LINE INPUT "HOE HEET U ?:";NAAM$
20 INPUT "HOEVEEL LETTERS HEEFT UW VOORNAAM";AANTAL
30 PRINT "HALLO ";LEFT$(NAAM$,AANTAL);"!"
RUN
HOE HEET U ?:JOHAN KARDINAAL
HOEVEEL LETTERS HEEFT UW VOORNAAM? 5
HALLO JOHAN!
Ok
```

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **LEN is afkorting van length –lengte**

schrijfwijze

```
LEN(<TE METEN STRING>)

<TE METEN STRING>::=<A>

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

betekenis

Deze funktie geeft als resultaat de lengte van de te meten string. Spaties en niet afdrukbare karakters die in de te meten string (een alfanumerieke uitdrukking) voorkomen, worden meegeteld.

Voorbeeld:

```
NEW
Ok
10 INPUT "HOE HEET U ";NAAM$
20 PRINT "UW NAAM IS";LEN(NAAM$);"LETTERS LANG"
RUN
HOE HEET U ? FERDINAND
UW NAAM IS 9 LETTERS LANG
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LET is laat**

schrijfwijze

```
[LET] <NUMERIEKE VARIABELE>=<N>
      ---------------------------
      <ALFANUMERIEKE VARIABELE>=<A>

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

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

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met het LET-kommando kan aan een variabele een bepaalde waarde worden toegekend. Indien een variabele nog niet eerder werd genoemd, wordt deze automatisch gecreëerd.

Bijvoorbeeld:

```
NEW
Ok
10 LET A=4
20 LET A$="TEST"
30 PRINT A;A$
RUN
 4 TEST
Ok
```

Het sleutelwoord LET mag volledig worden weggelaten. De volgende twee programmaregels zijn dan ook volledig identiek:

```
20 LET A$="TEST"
20 A$="TEST"
```

Indien een variabele zowel rechts als links van het eerste gelijk-teken

voorkomt, dan wordt eerst de uitdrukking rechts van het eerste gelijkteken uitgerekend en wordt daarna pas het eindresultaat aan de betreffende variabele toegekend. Voorbeeld:

```
NEW
Ok
10 INPUT "WAARDE ";WAARDE
20 LET WAARDE=WAARDE+WAARDE+WAARDE
30 PRINT "DRIE MAAL DEZE WAARDE IS";WAARDE
40 END
RUN
WAARDE ? 12.5
DRIE MAAL DEZE WAARDE IS 37.5
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LINE is lijn**

schrijfwijze

```
MSX-1: LINE[@][(<LOCATIE>)]-(<LOCATIE>)[,[(<KLEUR>)][,B[F]]]\(...>,

MSX-2: LINE[@][(<LOCATIE>)]-(<LOCATIE>)[,[(<KLEUR>)][,[B[F]][,<LOGISCHE ...
       ... OPERATOR>]]]\(...>,

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<KLEUR>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

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

betekenis

Voor goed begrip van dit kommando is het noodzakelijk dat eerst de PSET-behandeling volledig is doorgenomen.

Met dit kommando kunnen volledige lijnen of rechthoeken op het beeldscherm worden getekend. Een lijn wordt getrokken vanuit de eerste locatie naar de tweede in het LINE-kommando aangegeven locatie. Indien de eerste locatie niet wordt opgenomen, dan wordt het laatste getekende punt als beginlocatie genomen. Beide locaties mogen zich buiten het beeldscherm bevinden. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 LINE (10,10)-(100,100)
30 GOTO 30
RUN
```

(een lijn wordt getrokken)

```
20 LINE (10,10)-STEP(90,90)
RUN
```

(alleen regel 20 werd veranderd. Dezelfde lijn wordt getrokken)

```
15 PSET (-10,-10)
20 LINE -STEP(100,100)
RUN
```

Regel 15 werd toegevoegd en regel 20 werd veranderd. Een lijn wordt getrokken vanaf het denkbeeldige punt (−10,−10) naar (90,90).

```
15
20 LINE@(-100,-100)-(500,500)
RUN
```

Regel 15 werd weer verwijderd en regel 20 werd veranderd. Een lijn wordt tussen de twee denkbeeldige punten getrokken.

De toevoeging van het karakter @ is toegestaan. Deze toevoeging dient om LINE te onderscheiden van LINE INPUT en heeft verder geen funktie. We zullen deze toevoeging in alle voorbeelden verder weglaten.

Indien bij het LINE-kommando geen kleur wordt gespecificeerd, wordt de voorgrondkleur die op dat moment actief is, genomen als tekenkleur. Uiteraard kan een andere voorgrondkleur worden geactiveerd. Bijvoorbeeld:

```
NEW
Ok
10 COLOR 15,4,4
20 SCREEN 2
30 LINE (1,1)-(233,120),6
40 GOTO 40
RUN
```

Een donkerrode streep wordt op een donkerblauwe achtergrond getekend. Merk op dat het COLOR-kommando VOOR het SCREEN-kommando werd gegeven om de juiste achtergrondkleur te realiseren.

Indien de letter B (van BLOCK) wordt opgenomen, wordt er plotseling geen lijn meer getrokken maar wordt de rechthoek getekend waarvan de twee aangegeven punten de linker bovenhoek en de rechter onderhoek of de rechterbovenhoek en de linkeronderhoek vormen. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 LINE (20,20)-(200,200),12,B
30 GOTO 30
RUN
```

Een donkergroene rechthoek wordt getekend.

```
20 LINE (20,20)-(200,200),,B
RUN
```

Alleen programmaregel 20 werd gewijzigd. De rechthoek wordt nu in de geactiveerde voorgrondkleur getekend.

Indien een rechthoek gedeeltelijk of geheel buiten het beeldschermbereik valt, worden de buiten het beeld vallende lijnen van de rechthoek toch getekend. Dit gebeurt aan de corresponderende uiterste rand van het beeldscherm. Dit is onder meer in het bovenstaande voorbeeld het geval.

Indien de letters BF (block filled) worden opgenomen, wordt de rechthoek ook nog ingekleurd. Bijvoorbeeld:

```
20 LINE (20,20)-(200,200),12,BF
```

Alleen regel 20 werd vervangen. De rechthoek uit het voorlaatste voorbeeld wordt nu ook donkergroen ingekleurd.

Een combinatievoorbeeld:

```
NEW
Ok
10 COLOR 15,1,1
20 SCREEN 2
30 FOR I=10 TO 90 STEP 8
40 LINE (I+30,I)-(210-I,180-I),RND(1)*15,BF
50 NEXT I
60 GOTO 60
RUN
```

Een kleurig patroon van ingekleurde rechthoeken wordt getekend.

De volgende behandeling is alleen voor MSX-2 van toepassing

Uiteindelijk kan een logische operator worden toegevoegd aan het LINE-kommando. Deze logische operator bepaalt wat bij een LINE-kommando de resulterende kleur van uitvoering is, uitgaande van de bij het LINE-kommando gespecificeerde kleur en de kleur van de achtergrond waarop het LINE-kommando wordt uitgevoerd.

Een logische operator mag in kombinatie met het LINE-statement alleen onder SCREEN 5, 6, 7 of 8 worden uitgevoerd. Gebruik van een logische operator onder een andere screen-mode leidt tot een Syntax error.

De logische operatoren, toegevoegd aan het LINE-statement, hebben de volgende betekenis:

Log. oper.	UITWERKING
OR	de achtergrondkleur en de kleur waarin de lijn of het blok wordt uitgevoerd, ondergaan met elkaar een logische OR-bewerking. Het resultaat van deze bewerking is het nummer van de resulterende kleur.
AND	de achtergrondkleur en de kleur waarin de lijn of het blok wordt uitgevoerd, ondergaan met elkaar een logische AND-bewerking. Het resultaat van deze bewerking is het nummer van de resulterende kleur.
XOR	de achtergrondkleur en de kleur waarin de lijn of het blok wordt uitgevoerd, ondergaan met elkaar een logische XOR-bewerking. Het resultaat van deze bewerking is het nummer van de resulterende kleur.
PSET	de lijn of het blok wordt in de opgegeven kleur of in de voorgrondkleur uitgevoerd. PSET vertegenwoordigt als zodanig de normale uitvoering en behoeft eigenlijk niet te worden opgenomen.
PRESET	SCREEN 5 en 7: resulterende kleur = 15–opgegeven kleur SCREEN 6: resulterende kleur = 3–opgegeven kleur

	SCREEN 8: resulterende kleur = 255–opgegeven kleur Indien geen kleur werd opgegeven, dan wordt de geactiveerde voorgrondkleur als gegeven kleur beschouwd.
TOR TAND TXOR TPSET TPRESET	zelfde effect als corresponderende logische operatoren zonder de letter T daarvoor met dit verschil dat een lijn of blok in kleurkode 0 eenvoudigweg niet wordt uitgevoerd.

De diverse kleurbepalingen worden steeds tijdens uitvoering van het LINE-kommando beeldpunt voor beeldpunt uitgevoerd.

Een voorbeeld:

```
NEW
Ok
10 SCREEN 5
20 LINE (0,0)-(100,100),13,BF
30 LINE (20,20)-(40,40),15,BF
40 LET A=POINT(30,30)
50 SCREEN 0:PRINT A
RUN
```

In bovenstaand voorbeeld wordt eerst een blok in kleur 13 getekend. Daaroverheen wordt een blok in kleur 15 getekend met de logische operator XOR. Punt 30,30 is een punt dat binnen het tweede blok ligt.Variabele A wordt met behulp van het POINT-kommando gevuld met de resulterende kleur is gelijk aan 2. Berekening:

kleur 13	1101	
kleur 15	1111	XOR
resultaat	0010	(kleur 2)

In het volgende voorbeeld wordt met behulp van de XOR logische operator een kleurig vlakkenspel getekend:

```
NEW
Ok
10 SCREEN 5
20 LINE (RND(1)*255,RND(1)*255)-(RND(1)*255,RND(1)*255),RND(
1)*15,BF,XOR:GOTO 20
RUN
```

In dit laatste voorbeeld wordt getoond, hoe de XOR-operatie omkeer-

baar is. Door een tweede maal precies hetzelfde LINE-kommando met XOR uit te voeren, wordt een oude achtergrond hersteld.

```
NEW
Ok
10 DIM A%(500,4):FOR I=0 TO 500:FOR J=0 TO 4:A%(I,J)=RND(1)*
255:NEXT J,I
20 SCREEN 5:PSET (0,0):OPEN "GRP:" AS 1:FOR I=0 TO 150:PRINT
 #1,"*** MSX ":NEXT I:CLOSE
30 FOR I=0 TO 500:LINE (A%(I,0),A%(I,1))-(A%(I,2),A%(I,3)),A
%(I,4)/32,,XOR:NEXT I:GOTO 30
RUN
```

Wacht ongeveer een minuut op het resultaat.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LINE INPUT is regel ingave/invoer**

schrijfwijze

```
            ["<SATELLIETTEKST>";]
LINE INPUT ---------------------- <ALFANUMERIEKE VARIABELE>
                  #<KANAAL>,
```

```
<SATELLIETTEKST>::=<ALFANUMERIEKE KONSTANTE>

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

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

<KANAAL>::=<N>

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

betekenis

Eenvoudig gebruik

Zie ook de beschrijving van het INPUT-kommando.
De LINE INPUT is een kommando dat erg veel lijkt op de INPUT. Ook met het LINE INPUT-kommando kunnen tijdens de programmaloop gegevens aan de computer worden toegevoerd. De in te voeren gegevens moeten alfanumeriek zijn terwijl per LINE INPUT-kommando slechts één alfanumerieke variabele met gegevens kan worden gevuld. Bijvoorbeeld:

```
NEW
Ok
10 LINE INPUT "GEEF UW NAAM IN:";NAAM$
20 PRINT "HALLO ";NAAM$
30 STOP
RUN
GEEF UW NAAM IN:FREDERIK
HALLO FREDERIK
Break in 30
Ok
```

In tegenstelling tot de INPUT presenteert de LINE INPUT bij de ingave geen vraagteken. Ook in tegenstelling tot de INPUT is, dat bij

de LINE INPUT ook komma's en aanhalingstekens mogen worden ingegeven.

Indien in plaats van een ingave bij een LINE INPUT alleen een RETURN-toets wordt ingegeven, wordt een lege ingave aangenomen; de oude inhoud van de input-variabele wordt in dat geval (in tegenstelling tot hetgeen bij de INPUT het geval is) niet behouden.

Gevorderd gebruik

Zie hiervoor ook de beschrijving van het PRINT-kommando, onderdeel gevorderd gebruik: schrijven naar randapparatuur alsmede het OPEN, CLOSE en MAXFILES-kommando.

Indien een kanaalnummer is gespecificeerd, kan de invoer van gegevens vanaf een ander apparaat dan het toetsenbord plaatsvinden; bijvoorbeeld vanaf een cassetterecorder. Voorwaarde is, dat de gegevens regelgewijs op cassette zijn opgeslagen. Men kan stellen dat de gegevens van het betreffende randapparaat dienen te worden aangevoerd zoals deze ook vanaf het toetsenbord worden ingevoerd. Voorbeeld:

Eerste programma, schrijven van enkele gegevens naar cassetteband:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
20 LINE INPUT "GEGEVENS:";A$
30 IF A$="" THEN 100
40 PRINT #1,A$
50 GOTO 20
100 CLOSE:STOP
```

(nu eerst een cassetteband in de cassetterecorder plaatsen en de recorder op opnemen zetten)

```
RUN                                (band loopt even en stopt weer)
GEGEVENS:DIT IS EEN STUK TEST-TEKST
GEGEVENS:DAT STRAKS WEER WORDT INGELEZEN
GEGEVENS:TEST, TEST, TEST, TEST.
GEGEVENS:                          (hier wordt alleen een RETURN inge-
Break in 100                        geven; de band loopt even en stopt
Ok                                  weer)
```

(nu de cassetteband terugspoelen)

Tweede programma, lezen van gegevens van cassetteband:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
20 IF EOF(1) THEN CLOSE:STOP
30 LINE INPUT #1,A$:PRINT A$:GOTO 20
RUN
```

(cassetterecorder nu op afspelen zetten)

```
DIT IS EEN STUK TEST-TEKST
DAT STRAKS WEER WORDT INGELEZEN
TEST, TEST, TEST, TEST.
Break in 20
Ok
```

(de band stopt)

Indien u dit laatste voorbeeld uitprobeert, denk er dan wel aan, het volume van de cassetterecorder op ongeveer 80% te zetten. De MSX-standaard schrijft geen controles voor op het juiste inlezen van gegevens. Afhankelijk van de instelling en de kwaliteit van de cassetterecorder kunnen gegevens verminkt worden ingelezen.

Zie voor de verklaring van het gebruik van het sleutelwoord EOF de verklaring onder dit sleutelwoord.

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open in dat geval "MEM:TEST" in plaats van "CAS:TEST". Zie ook CALL MIMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LIST is lijst maken**

schrijfwijze

```
LIST[<REGELNUMMER>][-[<REGELNUMMER>]]

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

betekenis

Met dit kommando kunt u op het beeldscherm:

- Het volledige programma laten lijsten. Geef in dat geval slechts LIST in.
- Het eerste gedeelte van uw programma laten lijsten. Geef in dat geval LIST in, gevolgd door een minteken en het laatste te lijsten regelnummer.
- Het laatste gedeelte van uw programma laten lijsten. Geeft u in dat geval LIST in, gevolgd door het eerste te lijsten regelnummer en een minteken.
- Een middengedeelte van uw programma laten lijsten. Geeft u in dat geval LIST in, gevolgd door het eerste te lijsten regelnummer, een minteken en het tweede te lijsten regelnummer
- één enkel regelnummer laten lijsten. Geef in dat geval LIST in, gevolgd door het betreffende regelnummer of een punt voor het laatste behandelde regelnummer.

Voorbeelden:

```
LIST
```

lijst het gehele programma

```
LIST -100
```

lijst alle regels tot en met regel 100

```
LIST 100-
```

lijst alle regels vanaf regel 100

```
LIST 100-200
```

lijst alle regels vanaf 100 tot en met 200

```
LIST 100
```

lijst alleen regel 100

```
LIST
```

lijst het laatst behandelde regelnummer

De aangegeven regelnummers behoeven niet noodzakelijkerwijs aanwezig te zijn.

LIST zit in twee verschillende uitvoeringen standaard onder de funktietoetsen 4 en 9.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LLIST is afkorting van lineprinter list – lijst maken op afdrukeenheid (printer)**

schrijfwijze

```
LLIST[<REGELNUMMER>][-[<REGELNUMMER>]]

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

betekenis

Het LLIST-kommando werkt precies als het LIST-kommando met dit verschil, dat de programmaregels niet op het beeldscherm maar op de printer worden afgedrukt. Zie verder de behandeling van LIST.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LOAD is laden**

schrijfwijze

```
LOAD<BESTANDSNAAM>[,R]

<BESTANDSNAAM>::=<A>

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

betekenis

Het is noodzakelijk dat eerst de behandeling van SAVE wordt doorgenomen.

Programma's die met een SAVE zijn vastgelegd, kunnen met een LOAD weer in de computer worden teruggelezen.

Eenvoudig gebruik

een voorbeeld op cassetterecorder:

```
NEW
Ok
10 REM DIT IS EEN TESTPROGRAMMA
20 PRINT "TEST";
30 GOTO 20
```

(cassetterecorder op opnemen zetten)

```
SAVE"CAS:TEST"
Ok
NEW
Ok
LOAD"CAS:TEST"
```

(CAS: staat voor cassetterecorder; het programma heet TEST)

(terugspoelen en afspelen)

```
Found:TEST
Ok
LIST
10 REM DIT IS EEN TESTPROGRAMMA
```

```
20 PRINT "TEST";
30 GOTO 20
Ok
```

Met de achtervoeging ,*R* kan het programma direkt worden opgestart:

```
LOAD "CAS:TEST",R
```

(terugspoelen en afspelen)

Na verloop van tijd is het testprogramma geladen en begint de computer onmiddellijk TEST af te drukken.

Merk op dat wanneer het LOAD-kommando in een programmaregel is opgenomen, de mededelingen FOUND: (of SKIP:) achterwege blijven.

MSX schrijft geen controle voor op de integriteit van de ingelezen gegevens. Een verkeerd ingesteld volume of kwalitatief slechte apparatuur kan leiden tot verminkte programma's bij het inlezen zonder dat daar melding van wordt gemaakt. Zet het volume bij afspelen altijd op ongeveer 80 %.

N.B.: Het lezen van of schrijven naar cassetteband kan, wanneer het een groot programma betreft, wel tot enkele minuten duren.

Gevorderd gebruik

Wanneer de toevoeging ,*R* wordt gebruikt, dan worden kanalen opengelaten en het programma onmiddellijk opgestart. Voorbeeld:

Eerst gaan we een programma schrijven en SAVEn:

```
NEW
Ok
10 PRINT #1,"DIT IS EEN TEST"
20 GOTO 10
```

(cassetterecorder op opnemen)

```
SAVE "CAS:TEST"
Ok
```

Vervolgens schrijven we een tweede programma dat het alfanumerieke beeldscherm opent en vervolgens ons eerdere programma activeert:

```
NEW
Ok
10 OPEN "CRT:" AS 1
20 LOAD "CAS:TEST",R
RUN
```

(terugspoelen en afspelen)

```
DIT IS EEN TEST
DIT IS EEN TEST
DIT IS ...
etcetera
```

Uit dit voorbeeld blijkt:

- kanaal 1 bleef open staan
- programma TEST werd na laden onmiddellijk uitgevoerd.

Voor regel 20 mag in het bovenstaande voorbeeld eventueel:

```
20 RUN "CAS:TEST",R
```

worden ingetikt. Zie de behandeling van RUN.

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open indat geval "MEM:TEST" in plaats van "CAS :TEST". Zie ook CALL MEMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **SYSTEEM-VARIABELE**
afkomst . . . . . **LOC is een afkorting van LOCation –locatie, plaats**

schrijfwijze

```
LOC(<KANAAL>)

<KANAAL>::=<N>

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

betekenis

Het is noodzakelijk om voorafgaand aan deze behandeling eerst de (gevorderde) behandeling van PRINT, INPUT, LINE INPUT, MAXFILE, OPEN, CLOSE en EOF door te nemen.

Met de LOC-funktie kan worden opgevraagd, op welke positie in het betreffende bestand we ons op een bepaald moment 'bevinden'.

Hiertoe dient tussen haakjes een uitdrukking te worden opgenomen die in waarde het kanaalnummer voorstelt waarover het te onderzoeken bestand eerder werd geopend. Een decimale fractie wordt verwaarloosd.

De waarde die het resultaat van de LOC-funktie is, geeft aan hoeveel bytes (tekens) er reeds gelezen of geschreven zijn uit of in het betreffende bestand.

Een voorbeeld:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
20 FOR I=1 TO 5
30 PRINT LOC(1);"TEKENS GESCHREVEN"
40 PRINT #1,"TESTREGEL NUMMER";I
50 NEXT I:STOP
```

zet de cassetterecorder op opnemen

```
RUN
 0 TEKENS GESCHREVEN
 21 TEKENS GESCHREVEN
 42 TEKENS GESCHREVEN
 63 TEKENS GESCHREVEN
 84 TEKENS GESCHREVEN
Break in 50
Ok
```

spoel de cassette nu terug

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
20 IF EOF(1) THEN CLOSE:STOP ELSE INPUT #1,A$
30 PRINT A$;" (";LOC(1);"TEKENS GELEZEN)"
40 GOTO 20
```

zet de cassetterecorder op afspelen

```
RUN
TESTREGEL NUMMER 1   ( 21 TEKENS GELEZEN)
TESTREGEL NUMMER 2   ( 42 TEKENS GELEZEN)
TESTREGEL NUMMER 3   ( 63 TEKENS GELEZEN)
TESTREGEL NUMMER 4   ( 84 TEKENS GELEZEN)
TESTREGEL NUMMER 5   ( 105 TEKENS GELEZEN)
Break in 20
Ok
```

Ogenschijnlijk leest/schrijft de computer 3 tekens meer dan we in het voorbeeld per regel tellen. Echter, na elke regel wordt nog een spatie geschreven (het gevolg van een PRINT van een numerieke variabele). Daarbij wordt er na elke regel nog automatisch een CHR$(13) en een CHR$(10) geschreven ter afsluiting van de regel. Een heel bestand wordt (b.v. na CLOSE) met een CHR$(26) automatisch afgesloten.

Denk er aan, het volume van uw cassetterecorder op ongeveer 80% te zetten. MSX schrijft geen controles voor op het juiste inlezen van gegevens vanaf cassetterecorder. Slecht materiaal, vuile koppen of een verkeerde instelling kunnen leiden tot niet door uw computer opgemerkte verminking van gegevens.

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open in dat geval "MEM:TEST" in plaats van "CAS:TEST". Zie ook CALL MEMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **LOCATE is plaatsen**

schrijfwijze

```
LOCATE[<TEKENPOSITIE>][,[<REGELPOSITIE>][,<CURSOR>]]\LOCATE[<...>,]

<TEKENPOSITIE>::=<N>

<REGELPOSITIE>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met deze funktie kan de cursor gepositioneerd worden op een bepaalde plaats op het beeldscherm. Voorbeeld:

```
NEW
Ok
10 FOR X=0 TO 24:FOR Y=0 TO 20
20 LOCATE X,Y:PRINT "*"
30 NEXT Y,X:STOP
RUN
```

(In de linkerbovenhoek van het beeldscherm worden van boven naar beneden sterren afgedrukt.)
Merk op dat positie 0,0 wordt gevormd door de linkerbovenhoek van het scherm.

Eén van de twee beeldscherm-coördinaten mag worden weggelaten. Bijvoorbeeld:

```
NEW
Ok
10 LOCATE 20:PRINT "HALLO";
20 LOCATE ,0:PRINT "HIER BEN IK"
RUN
```

(De tekst HALLO wordt 20 posities uit de kantlijn afgedrukt.) Op de eerste regel van het beeldscherm wordt, vier posities verder uit de kantlijn, de tekst HIER BEN IK afgedrukt.

Met LOCATE kan ook de cursor zichtbaar of onzichtbaar worden gemaakt tijdens het projekteren. Het derde element achter LOCATE dient dan een 0 (onzichtbaar) of een positieve waarde, ongelijk aan 0 maar kleiner dan 256 (zichtbaar) te bevatten. Bij het opstarten is de cursor standaard onzichtbaar.

Indien alleen de cursor dient te worden veranderd, dient een LOCATE „0 (onzichtbaar) of bijvoorbeeld een LOCATE „1 (zichtbaar) te worden gebruikt.

De aangestuurde cursorposities mogen de beeldschermgrenzen natuurlijk niet overschrijden. De betreffende beeldschermmaten worden bij SCREEN behandeld.

Nog wat voorbeelden:

`LOCATE 10,20,0`	de cursor wordt op positie 10 (horizontaal), 20 (vertikaal) geplaatst en de cursor wordt onzichtbaar
`LOCATE ,5,1`	de corsor verplaatst zich naar de vijfde regel en wordt zichtbaar
`LOCATE 11,,0`	de cursor verplaatst zich naar de elfde positie horizontaal en wordt onzichtbaar.

Tijdens een INPUT of een LINE INPUT is de cursor altijd zichtbaar, hoe ook ingesteld. LOCATE heeft geen zin wanneer een grafisch scherm actief is.

„MSX-2: Een foutje in MSX-2 veroorzaakt dat bij aangeschakelde cursor CALL MEMINI verminkte bestandsnamen laat zien. Nadat de cursor werd uitgeschakeld, werkte CALL MEMINI weer naar behoren."

moeilijkheidsgraad normaal
soort ..... SYSTEEM-VARIABELE
afkomst ..... LOF is een afkorting van Lenth Of File – lengt van bestand

schrijfwijze

```
LOF(<KANAAL>)

<KANAAL>::=<N>

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

betekenis

Het is noodzakelijk om eerst de (gevorderde) behandeling van PRINT, INPUT, LINE INPUT, MAXFILES, OPEN, CLOSE en EOF door te nemen.

Met het sleutelwoord LOF kan de lengte in bytes (tekens) van een bestand worden opgevraagd. Tussen haakjes dient hiertoe een uitdrukking te worden opgenomen die in waarde het kanaalnummer voorstelt waarover het te onderzoeken bestand is geopend. Een eventuele decimale fractie wordt verwaarloosd.

Bijvoorbeeld

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
20 FOR I=0 TO 5:PRINT #1,"TEST-TEKST"
30 PRINT "HET BESTAND IS REEDS";LOF(1);"TEKENS LANG."
40 NEXT I:CLOSE:STOP
```

zet de cassetterecorder nu op opnemen

```
RUN
HET BESTAND IS REEDS 12 TEKENS LANG.
HET BESTAND IS REEDS 24 TEKENS LANG.
HET BESTAND IS REEDS 36 TEKENS LANG.
HET BESTAND IS REEDS 48 TEKENS LANG.
HET BESTAND IS REEDS 60 TEKENS LANG.
HET BESTAND IS REEDS 72 TEKENS LANG.
Break in 40
Ok
```

spoel de cassette nu terug

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
20 PRINT "HET BESTAND IS NU";LOF(1);"TEKENS LANG."
30 CLOSE:STOP
```

zet de cassetterecorder nu op afspelen

```
RUN
HET BESTAND IS NU 72 TEKENS LANG.
Break in 30
Ok
```

Ogenschijnlijk neemt het bestand steeds twee karakters per regel extra in beslag. Dat komt doordat MSX achter elke regel automatisch een CHR$(13) en een CHR$(10) plaatst ter afsluiting van de regel. Geheel onzichtbaar blijft de CHR$(26) die ter afsluiting van een heel bestand automatisch wordt geplaatst.

Denk er aan, het volume van uw cassetterecorder op ongeveer 80% te zetten. MSX schrijft geen controle voor op de juistheid van ingelezen gegevens van cassetterecorder. Slecht materiaal, vuile koppen of een verkeerde afstelling kunnen resulteren in niet door uw computer opgemerkte verminking van gegevens.

Voor MSX-2 gebruikers: probeer bovenstaande programma's ook eens met de RAM-disk. Open indat geval "MEM:TEST" in plaats van "CAS :TEST". Zie ook CALL MEMINI, CALL MKILL, CALL MFILES en CALL MNAME.

moeilijkheidsgraad . . **vrij moeilijk, kennis van logarithmische funkties is vereist**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **LOG is afkorting van natural logarithm – natuurlijke logarithmen**

schrijfwijze

```
LOG(<N>)

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

betekenis

Deze funktie geeft als resultaat de natuurlijke logarithme van de waarde van de tussen haakjes genoemde numerieke uitdrukking. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "WAARDE ";A
20 PRINT "NAT. LOG.=";LOG(A)
30 GOTO 10
RUN
WAARDE ? 12
NAT. LOG.= 2.4849066497879
WAARDE ? 2.7182818285
NAT. LOG.= 1.000000000015
WAARDE ? 0
NAT. LOG.=
Illegal function call in 20
Ok
```

Uiteraard dient de tussen haakjes vermelde waarde positief te zijn.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **LPOS is afkorting van line printer headposition – positie van de kop van de afdrukeenheid (printer)**

schrijfwijze

```
LPOS(<U>)

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

betekenis

Deze funktie geeft als resultaat de positie van de printercursor. De printercursor is niet zichtbaar, maar men moet zich deze voorstellen als iets dat de huidige positie op de printer aangeeft. Merk op dat de LPOS- en de POS-funktie veel met elkaar gemeen hebben.
De tussen haakjes op te nemen uitdrukking is een dummy-uitdrukking; in de praktijk kan het beste gewoon het cijfer 0 worden ingevuld. Bijvoorbeeld:

(voor dit voorbeeld is een aangesloten printer noodzakelijk)

```
NEW
Ok
10 LPRINT
20 PRINT LPOS(0)
30 LPRINT "DIT IS EEN TEST";
40 PRINT LPOS(0)
50 LPRINT "JE"
60 PRINT LPOS(0)
RUN
 0
 15
 0
Ok                    (op de printer verschijnt: DIT IS EEN TEST)
```

moeilijkheidsgraad . . **eenvoudig maar in complexe samenstellingen vrij moeilijk**

soort . . . . . **KOMMANDO**

afkomst . . . . . **LPRINT is samtrekking van lineprinter en print afdrukken op afdrukeenheid (printer)**

schrijfwijze

```
LPRINT[{<S>}<P>{<S>[<P>]}][USING<USING-STRING>;<U>{<S><U>}[<S>]]

<S>::=;|,

<P>::=<U>|TAB(POSITIE)|SPC(<AANTAL SPATIES>)

<POSITIE>::=<N>

<AANTAL SPATIES>::=<N>

<USING STRING>::=<A>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

De LPRINT werkt hoegenaamd volkomen identiek aan de PRINT. Het enige verschil is, dat de afdruk niet op het beeldscherm, maar op een eventueel aangesloten afdrukeenheid (printer) geschiedt.

Met LPRINT kunnen gegevens *alleen* (over de centronix parallelle poort) naar een afdrukeenheid worden gestuurd; bestandsbenadering is met LPRINT niet mogelijk; dit in tegenstelling tot de PRINT.

Zie voor een verdere behandeling de behandeling onder PRINT.

SLEUTELWOORD # MAXFILES

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **MAXFILES is afkorting van maximum number of files – maximaal aantal bestanden**

schrijfwijze

```
MAXFILES=<MAXIMUM AANTAL BESTANDEN>

<MAXIMUM AANTAL BESTANDEN>::=<N>

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

betekenis

Een bestand (file) is een verzameling van bij elkaar behorende gegevens. Deze gegevens kunnen in het computergeheugen staan maar kunnen ook op cassetteband of op een floppy disk staan.
Een bestand op cassette of floppy disk kan met MSX-basic op een aparte wijze worden benaderd.

Het aantal bestanden dat tegelijkertijd kan worden benaderd, wordt door MSX-basic bij het opstarten van de computer standaard op de waarde 1 gezet. Meestal is dit ook genoeg; het zal niet vaak voorkomen dat er meer dan één bestand tegelijkertijd wordt aangeroepen tenzij men in het bezit is van een schijveneenheid.

Wanneer men het aantal tegelijk te openen bestanden wil vergroten, dan kan dat door de systeemvariabele MAXFILES op een waarde te stellen, niet kleiner dan nul en niet groter dan 15. Een eventuele decimale fractie wordt verwaarloosd.

Elke extra mogelijkheid om een bestand tegelijkertijd te benaderen, kost wat geheugenruimte omdat de computer een gedeelte hiervan voor het lezen en schrijven van en naar het betreffende bestand dient te reserveren. Deze stukjes voor bestandsbenadering gereserveerd geheugen noemt men buffers. Per buffer is de grootte in MSX-basic 267 bytes (karakters).
Het is dus zaak om de systeemvariabele MAXFILES zo klein mogelijk te houden om zoveel mogelijk geheugen vrij te houden. Voorbeeld:

```
NEW
Ok
10 MAXFILES=0:I=0
20 A=A+FRE(0)
30 FOR I=0 TO 7:VPOKE I,PEEK(VARPTR(A)+I):NEXT I
40 MAXFILES=15
50 A=0:FOR I=0 TO 7:POKE VARPTR(A)+I,VPEEK(I):NEXT I
60 A=A-FRE(0)
70 PRINT "EEN BUFFER KOST";A/15;"POSITIES."
80 PRINT "15 BUFFERS KOSTEN SAMEN ONGEVEER";INT(100*A/25000)
;"% VAN HET TOTALE BASIC-GEHEUGEN."
RUN
EEN BUFFER KOST 267 POSITIES.
15 BUFFERS KOSTEN SAMEN ONGEVEER 16 % VAN HET TOTALE BASIC-G
EHEUGEN.
Ok
```

Op regel 30 wordt variabele A in het video RAM veilig gesteld. Op regel 50 wordt A weer teruggezet. Deze regels zijn verder niet van belang.

We zien dat alleen door MAXFILES aan een bepaalde waarde gelijk te stellen, aanzienlijke stukken geheugen al dan niet worden gereserveerd.

MAXFILES heeft nog een bijverschijnsel: MAXFILES voert automatisch ook een CLEAR uit. Bijvoorbeeld:

```
NEW
Ok
10 LET A=12.5
20 LET A$="TEST"
30 MAXFILES=0
40 PRINT A;A$
RUN
 0
Ok
```

Na MAXFILES zijn alle variabelen schoongemaakt.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **MERGE is samenvoegen**

schrijfwijze

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

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

betekenis

Het is noodzakelijk dat eerst de LOAD- en SAVE-behandeling is doorgenomen.

Programma's of programma-onderdelen die met een SAVE zijn vastgelegd, kunnen met een MERGE weer worden opgehaald. MERGE werkt op twee punten na identiek aan LOAD:

- LOAD maakt eerst het oude programmageheugen schoon, MERGE doet dat niet
- MERGE staat de ,*R* toevoeging niet toe.

Het eerste punt is zeer belangrijk. Doordat oude programmaregels niet worden verwijderd, kunnen verschillende programma's met elkaar worden *gecombineerd*. Bijvoorbeeld:

```
NEW
Ok
10 REM EERSTE PROGRAMMA
20 PRINT "TEST DEEL 1"
30 STOP
SAVE "CAS:TEST1"  (eerst cassetterecorder op opnemen zetten)
Ok
NEW
Ok
11 REM TWEEDE PROGRAMMA
30 PRINT "TEST DEEL 2"
40 STOP
SAVE "CAS:TEST2"
Ok
NEW
```

```
Ok
LOAD "CAS:TEST1"                    (cassetterecorder terugspoelen en
Found:TEST1                                    op afspelen zetten)
Ok
MERGE "CAS:TEST2"
Found:TEST2
Ok
LIST
10 REM EERSTE PROGRAMMA
11 REM TWEEDE PROGRAMMA
20 PRINT "TEST DEEL 1"
30 PRINT "TEST DEEL 2"
40 STOP
Ok
```

In bovenstaand voorbeeld werden twee onafhankelijke programma's, TEST1 en TEST2, apart op cassetteband gezet. Later werden deze twee in één programma samengevoegd. We zien:

- bij een MERGE wordt een programma bij een reeds in de computer aanwezig programma gevoegd
- indien een regelnummer al aanwezig is, wordt dit verwijderd ten behoeve van de in te lezen regel.

Men kan zich de MERGE-opdracht het beste voorstellen als ware het dat tijdens het uitvoeren van deze opdracht, het programma dat door MERGE wordt aangesproken, heel snel wordt ingetoetst (vanaf de cassetteband).

Door MERGE wordt het zinvol, grote programma's in gedeelten te ontwerpen en pas later, nadat de onderdelen zijn uitgetest, samen te voegen tot één geheel. Men zal dan al snel tot de conclusie komen dat sommige programma-onderdelen vaak voor meer dan één programma kunnen worden gebruikt.

Voor MSX-2 gebruikers: MERGE kan ook in combinatie met de RAM-disk worden gebruikt. Probeer bijvoorbeeld het bovenstaande voorbeeld maar eens met MEM: in plaats van CAS:. Zie ook CALL MEMINI, CALL MFILES, CALL MNAME en CALL MKILL.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **MID$ is afkorting van middle part of string – midden gedeelte van de string**

schrijfwijze

```
MID$(<TE WIJZIGEN STRING>,<EERSTE TEKEN>[,<AANTAL TEKENS>])=<A>

<EERSTE TEKEN>::=<N>

<AANTAL TEKENS>::=<N>

<TE WIJZIGEN STRING>::=<ALFANUMERIEKE VARIABELE>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met dit kommando (niet te verwarren met de MID$-*funktie*) kunnen delen van een alfanumerieke variabele worden vervangen. Welk gedeelte wordt vervangen, wordt bepaald door het opgegeven eerste teken en het aantal tekens. Waardoor wordt vervangen, wordt bepaald door de na het gelijkteken opgenomen alfanumerieke uitdrukking. Bijvoorbeeld:

```
NEW
Ok
10 LET A$="???-BASIC"
20 MID$(A$,1,3)="MSX"
30 PRINT A$
RUN
MSX-BASIC
Ok
```

Het opgegeven eerste teken mag niet kleiner zijn dan 1 en niet groter zijn dan de lengte van de te wijzigen string. Hieruit volgt dat alleen een reeds bestaande alfanumerieke variabele binnen het MID$-kommando mag worden genoemd.
Het aantal tekens mag niet kleiner zijn dan nul en niet groter dan 255.

Decimale fracties worden verwaarloosd.

Indien het aantal tekens wordt weggelaten, dan wordt een gedeelte vervangen tot aan de lengte van de te wijzigen string of totdat de alfanumerieke uitdrukking waarmee moet worden vervangen, 'op' is.
Bijvoorbeeld:

```
NEW
Ok
10 LET A$="DIT IS EEN TEST"
20 MID$(A$,5)="WAS EEN TEST"
30 PRINT A$
40 LET A$=A$+"T"
50 PRINT A$
60 MID$(A$,5)="IS"
70 PRINT A$
80 MID$(A$,7,1)=" "
90 PRINT A$
RUN
DIT WAS EEN TES
DIT WAS EEN TEST
DIT ISS EEN TEST
DIT IS  EEN TEST
Ok
```

De eerste regel laat zien dat er maximaal vervangen wordt tot aan de lengte van de te wijzigen alfanumerieke variabele.
De derde regel na RUN laat zien dat er vervangen wordt totdat de alfanumerieke uitdrukking waarmee moet worden vervangen, 'op' is.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **MID$ is afkorting van middle part of string – midden gedeelte van de string**

schrijfwijze

```
MID$(<BASISSTRING>,<EERSTE TEKEN>[,<AANTAL TEKENS>])

<BASISSTRING>::=<A>

<EERSTE TEKEN>::=<N>

<AANTAL TEKENS>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Deze funktie geeft als resultaat een middengedeelte van een alfanumerieke uitdrukking. Welk middengedeelte wordt gegeven als resultaat, hangt af van het gespecificeerde eerste teken en het gespecificeerde aantal tekens. Bijvoorbeeld:

```
PRINT MID$("MSX-BASIC-COMPUTER",5,5)
BASIC
Ok
```

Het aantal tekens mag niet kleiner zijn dan nul en niet groter zijn dan 255. Het opgegeven eerste teken mag niet kleiner zijn dan 1 en niet groter dan 255. Eventuele decimale fracties worden verwaarloosd.

Indien het aantal tekens niet wordt opgegeven, dan wordt vanaf het eerste teken de rest van de waarde van de alfanumerieke uitdrukking als resultaat gegeven. Bijvoorbeeld:

```
NEW
Ok
10 LET A$="DIT IS EEN TEST"
20 LET B$=LEFT$(A$,3)
30 LET C$=MID$(A$,5,2)
40 LET D$=MID$(A$,8,3)
50 LET E$=MID$(A$,12)
```

```
60 PRINT C$;" ";B$;" ";D$;" ";E$;"?"
RUN
IS DIT EEN TEST?
Ok
```

N.B.: De funktie MID$ en het kommando MID$ dienen niet met elkaar te worden verward.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **MOTOR is motor**

schrijfwijze

```
MOTOR [OFF]
      [---]
      [ON ]
```

betekenis

Alle cassetterecorderbesturende opdrachten in MSX kunnen de cassetterecordermotor ook aan- of uitschakelen. Hiertoe is een speciale REMOTE-aansluiting (afstandsbediening) voorgeschreven onder MSX. Bij een CSAVE-kommando en een CLOAD-kommando zien we, dat de cassetterecorder door de computer automatisch wordt ingeschakeld totdat de opdracht is voltooid. Na de opdracht wordt de cassetterecordermotor weer uitgeschakeld.

De cassetterecordermotor kan ook apart worden in- en uitgeschakeld zonder dat er direkt geladen of geschreven wordt. In dat geval dienen we het MOTOR-kommando te gebruiken. Wanneer we bijvoorbeeld de cassette in de recorder terug willen spoelen, kunnen we het afstandsbedieningsstekkertje uit de recorder halen en dan daadwerkelijk terugspoelen. Gemakkelijker is het om eerst

```
MOTOR ON
```

in te toetsen en dan terug te spoelen; de cassetterecorder is ingeschakeld. Om de cassetterecorder weer uit te schakelen, geven we in:

```
MOTOR OFF
```

In een programma kan het MOTOR-kommando worden gebruikt bij begeleide bediening van de cassetterecorder, bijvoorbeeld:

```
NEW
Ok
10 CLS:PRINT "----- SAVEN VAN HET PROGRAMMA -----"
20 LOCATE 0,3,1:LINE INPUT "RECORDER UIT OF AAN:";M$
```

```
30 IF M$="" THEN 50 ELSE IF LEFT$(M$,1)="U" THEN MOTOR OFF E
LSE IF LEFT$(M$,1)="A" THEN MOTOR ON
40 GOTO 10
50 MOTOR OFF:LOCATE 0,5:PRINT "ZET DE RECORDER OP OPNEMEN (R
ETURN)";
60 IF INPUT$(1)<>CHR$(13) THEN 60
70 LOCATE 0,7:PRINT "DE AANLOOPTAPE WORDT DOORGESPOELD";
80 MOTOR ON:FOR I=0 TO 5000:NEXT I:MOTOR OFF
90 LOCATE 0,9:PRINT "UW PROGRAMMA WORDT NU OP BAND GEZET";
100 CSAVE "TEST1",2
110 LOCATE 0,11:PRINT "UW PROGRAMMA STAAT NU OP BAND"
120 STOP
```

In het bovenstaande programma, dat als subroutine eventueel in een totaalprogramma zou kunnen worden opgenomen, wordt het op band zetten van een programma voor een groot deel begeleid.

Het MOTOR-kommando kent drie verschijningen:

`MOTOR ON`	–de cassetterecorder wordt ingeschakeld
`MOTOR OFF`	–de cassetterecorder wordt uitgeschakeld
`MOTOR`	–de cassetterecorder wordt ingeschakeld indien deze uitgeschakeld was en wordt uitgeschakeld indien deze ingeschakeld was.

Het MOTOR-kommando bedient een relais dat de stroomtoevoer naar de cassetterecordermotor al dan niet onderbreekt. Dit relais is licht belastbaar en kan voor allerlei doeleinden door de handige programmeur worden aangewend. Het automatisch besturen van een diaprojektor is bijvoorbeeld een vrij spectaculaire en in bepaalde toepassingen erg bruikbare mogelijkheid.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **NEW is nieuw**

schrijfwijze

```
NEW
```

betekenis

Het NEW-kommando heeft tot gevolg dat het programmageheugen geheel wordt schoongewist. Ook variabelen worden verwijderd en alle openstaande kanalen worden gesloten. Na een NEW komt de computer altijd met de Ok-melding, ook wanneer de NEW in een regel werd opgenomen.

Voorbeeld:

```
NEW
Ok
10 PRINT "HALLO ALLEMAAL"
30 NEW
RUN
HALLO ALLEMAAL
Ok
LIST
Ok
```

(programma is verdwenen)

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **NEXT is volgende**

schrijfwijze

```
NEXT[<NUMERIEKE VARIABELE>{,<NUMERIEKE VARIABELE>}]

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

betekenis

Dit kommando wordt volledig bij FOR...TO...STEP behandeld.
Zie aldaar.

moeilijkheidsgraad . . vrij moeilijk, vereist kennis van talstelsels en algemene principes van het computergeheugen
soort . . . . . A-FUNKTIE
afkomst . . . . . OCT$ is afkorting van octal string – octale (achttallige) string

schrijfwijze

```
OCT$(<N>)

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

betekenis

Deze funktie geeft de octale waarde van de tussen haakjes vermelde uitdrukking weer in string-vorm. De waarde van de numerieke uitdrukking dient te liggen tussen –32769 en 65536. Indien de waarde van de numerieke uitdrukking gebroken is, wordt de grootste gehele waarde, kleiner dan de gebroken waarde als geldende waarde genomen.
Indien de waarde van de numerieke uitdrukking negatief is, geeft OCT$ een octale representant, samengesteld volgens de tweecomplementmethode. Bijvoorbeeld:

```
PRINT OCT$(-32768)
100000
Ok
PRINT OCT$(65535)
177777
Ok
PRINT OCT$(-1)
177777
Ok
PRINT OCT$(1023)
1777
Ok
```

SLEUTELWOORD ON ERROR GOTO

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON ERROR GOTO is ga bij een fout naar (goto – samentrekking van go to)**

schrijfwijze

```
ON ERROR GOTO[<REGELNUMMER>]

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

betekenis

Met dit kommando kan een fout die in het programma optreedt, worden gedetecteerd en kan een bepaalde actie worden ondernomen.

Wanneer de computer de ON ERROR GOTO tegenkomt, wordt geen actie ondernomen. De computer 'weet' echter vanaf dat moment dat wanneer een fout optreedt, hij met het vermelde regelnummer dient verder te gaan.

Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 1000
20 INPUT "WAARDE ";A
30 PRINT 1/A:GOTO 20
1000 PRINT "FOUT ONTDEKT":STOP
RUN
WAARDE ? 1
 1
WAARDE ? 2
 .5
WAARDE ? 0
FOUT ONTDEKT
Break in 1000
Ok
```

Doordat de laatste ingave een nul werd ingegeven, moet op regel 30 door nul worden gedeeld. Dit resulteert in een fout waardoor de computer regel 1000 en verder ging uitvoeren.
De regel waarop de foutmelding plaatsvond en het nummer van de

fout zijn in de systeemvariabelen ERL en ERR terug te vinden. Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 500
20 LET A$=12
30 STOP
500 PRINT "FOUT NUMMER";ERR
510 PRINT "OP REGEL";ERL
520 STOP
RUN
FOUT NUMMER 13
OP REGEL 20
Break in 520
Ok
```

Op regel 20 werd een Type mismatch foutmelding (fout 13, zie hoofdstuk 11 en 12) geforceerd. Hierdoor werd verder gegaan met programmaregel 500. Het foutnummer en de regel waarop de fout optrad, worden vermeld.

Vanuit de ERROR-ROUTINE (het programmaonderdeel waarin de foutmeldingen worden behandeld) kan het programma eventueel weer worden doorgestart met een RESUME of een RESUME 0. Het effect hiervan is dat het kommando waarin de fout werd ontdekt, nogmaals wordt uitgevoerd. De error-routine moet dan natuurlijk een zodanige actie hebben ondernomen dat de fout niet meer optreedt. Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 200
20 A=0:B=1/A
30 STOP
200 A=1:RESUME
RUN
Break in 30
Ok
```

Dit weinig zinvolle programma wordt pas interessant wanneer we eerst een TRON ingeven en dan pas een RUN. Het zal dan blijken dat de programmaregels in de volgende volgorde worden uitgevoerd:

10 de error-routine wordt bepaald
20 A wordt op 0 gezet. B wordt op 1/A gesteld. Dit geeft een fout-

melding in verband met het delen door nul. Door deze foutmelding gaat het programma verder met regel:

200 de variabele A wordt op 1 gezet en het programma wordt vervolgd

20 wederom wordt B op 1/A gesteld. A heeft nu echter de waarde 1, waardoor geen foutmelding meer optreedt

30 het programma is ten einde.

Wanneer een kommando dat een fout veroorzaakt, dient te worden overgeslagen, kan een RESUME NEXT worden gebruikt, bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 1000
20 FOR I=10 TO -10 STEP -1
30 PRINT 1/I
40 NEXT I:STOP
1000 IF ERR=11 AND ERL=30 THEN RESUME NEXT
1010 STOP
RUN
```

Er worden 20 getallen op het scherm geprojekteerd die allemaal het resultaat zijn van de deling op regel 30. Eén van de delingen die op regel 30 wordt uitgevoerd, is de deling 1/0, namelijk wanneer I gelijk is aan nul. In dat geval wordt in de error-routine een RESUME NEXT gegeven; het fout veroorzakende kommando wordt overgeslagen.
Merk op dat in de error-routine eerst wordt gecontroleerd (in regel 1000) of inderdaad de verwachte fout op het verwachte regelnummer is opgetreden.

Indien vanuit de foutroutine dient te worden besloten dat hervatting van het programma op een geheel andere plaats dient plaats te vinden, kan een RESUME worden opgenomen, gevolgd door het regelnummer waarmee moet worden hervat. Regel 1000 uit het laatste voorbeeld had dus mogen luiden:

```
1000 IF ERR=11 AND ERL=30 THEN RESUME 40
```

Wanneer de foutafvanging moet worden afgezet, dus wanneer niet langer van de computer wordt verlangd dat bij een fout een speciale actie wordt ondernomen, kan de foutafvanging worden uitgeschakeld door de ON ERROR GOTO zonder regelnummer of met regelnummer 0 op te nemen. Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 200'FOUTAFVANG AAN
20 LET A$=12'EEN OPZETTELIJKE TYPE MISMATCH FOUT
30 ON ERROR GOTO'NU DE FOUTAFVANG WEER UIT
40 LET A$=12'EN NOG 'NS DE ZELFDE FOUT
50 STOP
200 RESUME NEXT'FOUT OVERSLAAN
RUN
Type mismatch in 40
Ok
```

(De fout op regel 20 werd afgevangen; die op regel 40 niet meer.)

Wanneer de ON ERROR GOTO zonder regelnummer of met regelnummer 0 IN de error-routine wordt geplaatst, heeft dat tot effect dat de laatste fout alsnog wordt afgedrukt op het beeldscherm en het programma alsnog wordt onderbroken. Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 1000
20 LET B=1/0
30 LET A$=12
40 STOP
1000 IF ERL=30 THEN ON ERROR GOTO 0
1010 RESUME NEXT
RUN
Type mismatch in 30
Ok
```

De fout op regel 20 wordt door de error-routine ontdekt en overgeslagen. De fout op regel 30 resulteert echter in een uitvoering van een ON ERROR GOTO 0 waardoor alsnog de foutmelding wordt gegeven en het programma wordt onderbroken.

In een ingewikkeld programma kan ook de error-routine ingewikkelde vormen aannemen. Om de error-routine uit te testen op goed funktioneren, moeten we in staat zijn om fouten te forceren. In de voorbeelden deden we dit al enkele malen, bijvoorbeeld met een LET A$ = 12 (geeft fout 13, Type mismatch). Veel eenvoudiger kan een fout worden geforceerd met het ERROR-kommando. Achter dit kommando behoeven we slechts het foutnummer (uit hoofdstuk 11 of 12) op te nemen. Bij uitvoering van dit kommando forceert de computer de opgegeven fout. Bijvoorbeeld:

```
NEW
Ok
10 INPUT "WELKE FOUT ";F
20 ERROR F
RUN
WELKE FOUT ? 5
Illegal function call in 20
Ok
RUN
WELKE FOUT ? 20
Verify error in 20
Ok
RUN
WELKE FOUT ? 99
Unprintable error in 20
Ok
```

Merk op dat de fout niet slechts wordt aangegeven maar ook daadwerkelijk wordt gesimuleerd.

Enkele opmerkingen:

- Indien een fout ontstaat IN een foutroutine, dan wordt deze fout niet meer afgevangen maar gewoon gemeld waarna het programma onderbreekt.

- Er zit een kleine fout in MSX-basic waardoor de fout-afvanging, eenmaal geactiveerd, actief blijft ook al is het programma reeds onderbroken. Hierdoor kan het voorkomen dat een programma plotseling vanuit de error-routine weer actief wordt wanneer bij het intypen van een stuk programma een fout wordt gemaakt. Het één en ander kan verwarrende resultaten opleveren. Geef ter voorkoming een ON ERROR GOTO 0 in nadat een programma met foutafvanging is onderbroken.
  Een voorbeeld van een mogelijk verwarrende situatie:

```
NEW
Ok
10 ON ERROR GOTO 100
20 STOP
100 RESUME NEXT
RUN
Break in 20
Ok
NU KAN IK VAN
Ok
ALLES INGEVEN, DE
```

```
Ok
COMPUTER GEEFT GEEN FOUT-
Ok
MELDINGEN MEER...
Ok
```

Nadat het bovenstaande programma eenmaal is uitgevoerd, is de computer niet meer in staat om een foutmelding te geven. Steeds bij een optredende fout wordt regel 100 geactiveerd. De RESUME NEXT geeft aan dat de regel met de fout gewoon moet worden overgeslagen; geen foutmelding verschijnt.

– RENUM (zie de behandeling van dit kommando) probeert ook de numerieke uitdrukking na ERL= te hernummeren. Wanneer deze numerieke uitdrukking bestaat uit één enkele konstante, geeft dat geen problemen. Echter, wanneer de numerieke uitdrukking complexer is, resulteert RENUM in weinig zeggende foutmeldingen of (en dit is kwalijker) helemaal geen foutmeldingen. Bijvoorbeeld:

```
NEW
Ok
10 ON ERROR GOTO 1000
20 ERROR 5:STOP
1000 IF ERL=20 THEN RESUME NEXT
1010 ON ERROR GOTO
RENUM 1
Ok
LIST
1 ON ERROR GOTO 21
11 ERROR 5:STOP
21 IF ERL=11 THEN RESUME NEXT
31 ON ERROR GOTO
Ok
```

In het bovenstaande voorbeeld ging ook het hernummeren van ERL goed. Indien we echter de volgende verandering in het programma aanbrengen:

```
21 LET A=10
22 IF ERL=21-A THEN RESUME NEXT
RENUM ,,1
Ok
```

dan volgt geen foutmelding. Het programma ziet er echter als volgt uit:

```
LIST
10 ON ERROR GOTO 12
11 ERROR 5:STOP
12 LET A=10
13 IF ERL=12-A THEN RESUME NEXT
14 ON ERROR GOTO
Ok
```

Regel 13 is door het onjuiste ingrijpen van RENUM nutteloos geworden (het voorkomen van een fout op regel 11 wordt niet meer afgevraagd); een foutmelding werd in dit geval niet gegeven omdat bij het hernummeren een regel met regelnummer 21 kon worden gevonden (het eerste gedeelte van de numerieke uitdrukking).

Over het algemeen kan men stellen dat men moet voorkomen, de systeem variabele ERL anders te gebruiken dan in een direkte afvraging.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON-GOSUB is samentrekking van on-go to subroutine – ga bij... naar onderprogramma (subroutine)**

schrijfwijze

```
ON<RANGWAARDE>GOSUB<REGELNUMMER>(,<REGELNUMMER>)

<REGELNUMMER>::=<ZIE ALGEMENE SPECIFICATIES>

<RANGWAARDE>::=<N>

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

betekenis

Het is noodzakelijk om eerste de behandeling van ON...GOTO goed door te nemen alsmede die van het GOSUB-kommando.

Dit kommando werkt ongeveer hetzelfde als het ON...GOTO-kommando met dit verschil dat er subroutines worden aangesproken. Deze subroutines of onderprogramma's eindigen allemaal met het RETURN-kommando. Na het RETURN-kommando wordt verdergegaan met het kommando na de ON...GOSUB tenzij in het RETURN-kommando iets anders is aangegeven.

Bijvoorbeeld:

```
NEW
Ok
10 FOR I=1 TO 3
20 ON I GOSUB 100,200,300
30 NEXT I:FOR I=1 TO 1E+20:NEXT
100 SCREEN 2:LINE (0,0)-(100,100):RETURN
200 CIRCLE (100,100),50:RETURN
300 LINE (10,10)-(60,40),,B:RETURN
RUN
```

Door elkaar worden een lijn (eerste subroutine), een cirkel (tweede subroutine) en een rechthoek (derde subroutine) getekend. Hierna staat de computer vast in regel 30.

SLEUTELWOORD # ON-GOTO

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON-GOTO is ga bij... naar**

schrijfwijze

```
ON<RANGWAARDE>GOTO<REGELNUMMER>{,<REGELNUMMER>}

<REGELNUMMER>::=<ZIE ALGEMENE SPECIFICATIES>

<RANGWAARDE>::=<N>

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

betekenis

Het is noodzakelijk dat eerste het GOTO-kommando is doorgenomen.

Met dit kommando is het mogelijk om afhankelijk van de waarde een numerieke uitdrukking naar diverse locaties in het programma te springen. Hiertoe worden de volgende acties bij een ON...GOTO ondernomen:

- allereerst wordt de numerieke uitdrukking uitgewerkt. Het resultaat wordt ontdaan van een eventuele decimale fractie.
- vervolgens wordt gecontroleerd of deze waarde groter is dan of gelijk is aan nul. Bij een negatieve waarde volgt een foutmelding.
- daarna wordt gecontroleerd of de berekende waarde niet groter is dan 255. Indien dit wel het geval is, volgt een foutmelding.
- de berekende waarde geeft aan, naar welk regelnummer moet worden gesprongen. Zo wordt bij de waarde 1 naar het eerste regelnummer gesprongen en bij de waarde 2 naar het tweede regelnummer enzovoorts. Indien geen regelnummer voor de betreffende waarde is gespecificeerd, wordt het programma vervolgd met het kommando na de ON...GOTO.

Bijvoorbeeld:

```
NEW
Ok
10 REM PROGRAMMAKEUZE
20 SCREEN 0:PRINT "PROGRAMMAKEUZE":PRINT:PRINT
30 PRINT "1...TEKEN EEN LIJN"
40 PRINT "2...TEKEN EEN CIRCEL"
50 PRINT "3...TEKEN EEN RECHTHOEK"
60 INPUT "KEUZE ";KEUZE
70 ON KEUZE GOTO 100,200,300
80 PRINT "ALLEEN EEN 1, 2 OF 3 INGEVEN":GOTO 60
90 IF INKEY$="" THEN 90 ELSE 20
100 REM LIJN
110 SCREEN 2:LINE (0,0)-(200,100):GOTO 90
200 REM CIRCEL
210 SCREEN 2:CIRCLE (100,100),50:GOTO 90
300 REM RECHTHOEK
310 SCREEN 2:LINE (20,20)-(100,100),,B:GOTO 90
RUN
```

Op het beeldscherm verschijnt een keuzemenu. Naar keuze kunnen een lijn, een cirkel of een rechthoek worden getekend. Nadat de tekening klaar is, behoeft slechts een toets te worden ingedrukt om weer het keuzemenu te verkrijgen.

SLEUTELWOORD ON INTERVAL GOSUB

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON INTERVAL GOSUB is samentrekking van on interval go to subroutine – bij pauze ga naar onderprogramma (subroutine)**

schrijfwijze

```
ON INTERVAL=<TIJDSDUUR>GOSUB[<REGELNUMMER>]

<TIJDSDUUR>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Voorafgaand aan de behandeling van dit kommando dient het ON KEY GOSUB kommando en het INTERVAL-kommando grondig te zijn doorgenomen.

Met dit kommando kunnen we de computer opdragen, de programmaloop met regelmatige tussenpozen te onderbreken. Men kan de ON INTERVAL GOSUB vergelijken met een soort ON KEY GOSUB waarbij de computer met regelmatige tussenpozen ZELF de funktietoets indrukt. Als zodanig gelden de algemene voorwaarden met betrekking tot ON KEY GOSUB ook voor ON INTERVAL GOSUB.

De tijdspanne die tussen twee onderbrekingen dient te verlopen, dient achter het gelijkteken te worden vermeld. Deze numerieke uitdrukking dient een waarde te vertegenwoordigen, groter dan –32769 en kleiner dan 65536 en stelt het aantal vijftigsten seconden voor dat de betreffende tijdspanne lang dient te zijn. Indien de waarde van de numerieke uitdrukking kleiner is dan 0, wordt er 65536 door de computer bij opgeteld waarna de tijdspanne is bepaald. De kenner herkent hierin de tweecomplementmethode. Een eventuele decimale fractie wordt verwaarloosd.

Voorbeeld:

```
NEW
Ok
10 INTERVAL ON:LET Q=0:CLS
20 ON INTERVAL=50 GOSUB 1000
30 GOTO 30
1000 LOCATE 0,0:LET Q=Q+1:PRINT Q:BEEP
1010 RETURN
RUN
```

Elke seconde wordt variabele Q verhoogd en linksboven in het scherm afgedrukt. Bovendien klinkt een attentie-signaal.

Opmerkingen:

- Tijdens de uitvoering van de onderbrekings-subroutine wordt automatisch door de computer een INTERVAL STOP uitgevoerd. Op het moment dat de RETURN uit het onderbrekingsgedeelte wordt uitgevoerd, wordt weer een INTERVAL ON uitgevoerd tenzij in de routine zelf reeds een INTERVAL STOP of INTERVAL OFF werd uitgevoerd.

- Tijdens het uitvoeren van de ERROR-routine wordt automatisch een INTERVAL STOP uitgevoerd. Direkt bij de RESUME wordt dan weer een INTERVAL ON uitgevoerd tenzij de ERROR-routine zelf een INTERVAL STOP of INTERVAL OFF uitvoerde.

- Wanneer een programma ten einde is, wordt automatisch een INTERVAL OFF uitgevoerd.

- Een kommando wordt altijd in het geheel uitgevoerd alvorens onderbreking plaatsvindt. Een INPUT-statement kan daarom bijvoorbeeld niet met ON INTERVAL-GOSUB worden onderbroken.

- De RETURN in de onderbrekings-subroutine mag elke voor RETURN toegestane vorm aannemen.

- Met een ON INTERVAL=... GOSUB (0) kan de mogelijkheid tot regelmatige onderbreking worden uitgeschakeld.

ON INTERVAL GOSUB vindt zijn toepassing bij uitstek binnen spelprogramma's en educatieve programma's (tijdsbewaking).

SLEUTELWOORD # ON KEY GOSUB

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON KEY GOSUB is afkorting van on key go to subroutine – bij toets ga dan naar onderprogramma (subroutine)**

schrijfwijze

```
ON KEY GOSUB[<REGELNUMMER>]{,{,}<REGELNUMMER>}

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

betekenis

De ON ERROR GOTO dient voorafgaand aan deze behandeling te zijn doorgenomen alsmede de GOSUB en de KEY.

Met dit kommando, dat in principe wel wat lijkt op de ON ERROR GOTO, kan worden bewerkstelligd dat een programma tijdelijk wordt onderbroken, in welk stadium dan ook, bij het intoetsen van een funktietoets. Deze onderbreking bestaat hier uit, dat een subroutine wordt aangesproken zodra de funktietoets in kwestie wordt ingetoetst. Hervatting van het programma kan aan het einde van deze subroutine met een RETURN plaatsvinden. Na deze return gaat het programma, tenzij anders aangegeven, door op de plaats waar het eerder werd onderbroken (vergelijk de RESUME bij ON ERROR GOTO).

Om ervoor te zorgen dat een programma door middel van een funktietoets kan worden onderbroken, dienen we de betreffende funktietoets eerst te activeren. Zonder deze activering kan een onderbreking via deze funktietoets nooit plaatsvinden. Het activeren van de funktietoets gaat als volgt:

```
KEY (numerieke uitdrukking) ON
```

De numerieke uitdrukking dient in waarde positief, geheel en kleiner dan 11 te zijn. Een eventuele decimale fractie wordt verwaarloosd.

Wanneer we bijvoorbeeld funktietoets 1 en 4 willen activeren, programmeren we:

```
NEW
Ok
10 KEY(1) ON:KEY(4) ON
```

De haakjes zijn verplicht; hierdoor wordt de mogelijkheid tot verwarring met een andere KEY-betekenis (zie behandeling van KEY, derde betekenis) voorkomen.

Vervolgens dienen we aan te geven, welke subroutines dienen te worden aangesproken wanneer deze funktietoetsen worden ingetoetst. Hiervoor gebruiken we het ON KEY GOSUB kommando:

```
20 ON KEY GOSUB 1000,,,1200
```

Merk op dat achter ON KEY GOSUB maximaal 10 regelnummers mogen worden vermeld. Deze regelnummers verwijzen naar de subroutines die worden aangesproken bij intoetsing van funktietoets 1, 2, 3... Vergelijk deze funktie voor het gemak met de ON...GOSUB waarbij de (verboden) variabele KEY de waarde van de funktietoets heeft.

Wanneer voor een funktietoets geen regelnummer is gespecificeerd, dan maakt het niet uit of deze funktietoets al dan niet is geactiveerd; een onderbreking zal in dat geval nooit plaatsvinden.

Belangrijk is het om te beseffen dat (ongeveer zoals bij de ON ERROR GOTO) de ON KEY GOSUB niet wordt uitgevoerd. Het enige dat de computer na uitvoering van ON KEY GOSUB 'weet' is, naar welk regelnummer moet worden gesprongen bij intoetsing van een funktietoets. *Daadwerkelijke* uitvoering vindt eventueel pas plaats op het moment van intoetsing van de funktietoets.

We maken ons voorbeeldprogramma af. Voor de duidelijkheid herhalen we de eerste twee regels ook nog even:

```
NEW
Ok
10 KEY(1) ON:KEY(4) ON
20 ON KEY GOSUB 1000,,,1200
30 LET A$="GEEN FUNKTIETOETS INGEDRUKT"
40 PRINT A$
50 GOTO40
1000 LET A$="FUNKTIETOETS 1"
1010 RETURN
1200 LET A$="FUNKTIETOETS 4"
1210 RETURN
RUN
```

Het beeld wordt volgeschreven met de tekst GEEN FUNKTIETOETS INGEDRUKT. Wanneer funktietoets 1 wordt ingetoetst, wordt plotseling de tekst FUNKTIETOETS 1 afgedrukt. Wanneer funktietoets 4 wordt ingetoetst, verschijnt continu de tekst FUNKTIETOETS 4. De overige funktietoetsen hebben geen effect.

Ondanks dat de computer in de eeuwigdurende lus (loop) 40-50-40 verkeert, heeft het aanraken van één van de geprogrammeerde funktietoetsen tot gevolg dat het programma wordt onderbroken en de desbetreffende subroutine wordt geactiveerd. In dit geval veranderen deze subroutines de inhoud van variabele A$. De RETURN aan het einde van deze subroutines zorgt ervoor, dat het programma de 'draad' weer oppakt waar het eerder met deze subroutine werd onderbroken.

Het zal duidelijk zijn dat ondermeer in spelprogramma's dit kommando enorme mogelijkheden biedt.

Wanneer de mogelijkheid tot onderbreken van het programma dient te worden opgeheven voor een bepaalde funktietoets, dan kan dat met

```
KEY (numerieke uitdrukking) OFF
```

Gebruik van dit kommando is hetzelfde als de KEY...ON.
Wanneer we bijvoorbeeld in het eerdere voorbeeld programmeren:

```
35 KEY(4) OFF
```

Dan zal funktietoets 4 na uitvoering van de programmaregel 35 niet meer het effect hebben dat er ten behoeve van subroutine 1200 het programma wordt onderbroken.

Alle funktietoetsen kunnen in één keer worden gedeactiveerd door het kommando:

```
ON KEY GOSUB ,,,,,,,,,
```
(9 komma's)

Wanneer het in een bepaald gedeelte niet wenselijk is dat het programma wordt onderbroken maar een intoetsing van de betreffende funktietoets wel dient te worden onthouden, dan kan dat worden geregeld met behulp van het kommando:

```
KEY (numerieke uitdrukking) STOP
```

Gebruik dit kommando hetzelfde als de KEY...ON.

Een eventuele intoetsing van de aangegeven funktietoets wordt door de computer 'onthouden' zonder dat het programma wordt onderbroken. Echter, op het moment dat de betreffende funktietoets weer wordt geactiveerd (KEY...ON) wordt de door de computer 'onthouden' intoetsing onmiddellijk gehonoreerd met een onderbreking ten behoeve van de aangesproken subroutine. Bijvoorbeeld:

```
NEW
Ok
10 ON KEY GOSUB ,,,,,,,,,1000
20 KEY(10) STOP
30 PRINT "DEZE REGEL WORDT ";
40 PRINT "ALTIJD AANEENGESLOTEN AFGEDRUKT"
50 KEY(10) ON:GOTO 20
1000 PRINT:RETURN
RUN
```

Continu wordt de tekst DEZE REGEL ... AFGEDRUKT steeds op één beeldschermregel afgedrukt. Door het intoetsen van funktietoets 10 kan wel een extra regelopschuiving worden geforceerd (routine 1000) maar wordt de regel nooit over twee beeldschermregels afgedrukt; een bewijs dat de KEY...STOP er voor zorgt dat nooit tussen regel 20 en 30 kan worden onderbroken. Door regel 20 te vervangen door bijvoorbeeld een REM-kommando kan (de KEY...STOP is nu weg) nu plotseling wèl een regelopschuiving tussen DEZE REGEL WORDT en ALTIJD AANEENGESLOTEN AFGEDRUKT worden geforceerd met funktietoets 10.

Opmerkingen:

- Tijdens het uitvoeren van een subroutine die door de ON KEY GOSUB werd aangeroepen, wordt voor de betreffende funktietoets automatisch een KEY...STOP uitgevoerd. Op het moment dat de RETURN wordt uitgevoerd, wordt automatisch een KEY...ON voor de betreffende funktietoets uitgevoerd, tenzij in de betreffende subroutine een KEY...STOP of KEY...OFF voor de betreffende subroutine werd uitgevoerd. Dit is om te voorkomen dat er merkwaardige fouten ontstaan doordat binnen de subroutine dezelfde routine een volgende maal wordt aangeroepen.

- Tijdens het uitvoeren van de ERROR-routine wordt voor elke actieve funktietoets automatisch een KEY...STOP uitgevoerd. Bij de RESUME worden de betreffende funktietoetsen wederom geactiveerd, tenzij in de ERROR-routine een KEY...STOP of KEY... OFF voor bepaalde funktietoetsen werd uitgevoerd.

- Wanneer het programma ten einde is, wordt automatisch voor elke funktietoets een KEY...OFF uitgevoerd.

- Een KEY...STOP heeft geen zin wanneer ervoor niet een KEY...ON voor dezelfde funktietoets werd uitgevoerd.

- Een kommando wordt altijd *volledig* afgewerkt voordat een onderbreking plaatsvindt.

- De RETURN in een onderbrekings-subroutine mag elke voor RETURN toegestane vorm hebben.

- Het gebruik van een regelnummer 0 in ON KEY GOSUB heeft tot gevolg dat een onderbreking voor de betreffende funktietoets niet meer mogelijk is.

- ON KEY GOSUB (0) schakelt de onderbrekingsmogelijkheid voor de eerste funktietoets uit.

- Een met KEY ON geactieveerde funktietoets die niet met KEY OFF is geactieveerd, geeft tijdens ingave niet de hieronder geprogrammeerde tekst.

SLEUTELWOORD # ON SPRITE GOSUB

moeilijkheidsgraad . . . . . **moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON SPRITE GOSUB is afkorting van on sprite go to subroutine – bij (botsing van) sprites ga naar onderprogramma (subroutine). Sprite is geest**

schrijfwijze

```
ON SPRITE GOSUB[<REGELNUMMER>]

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

betekenis

Voorafgaande aan deze behandeling dient het ON KEY GOSUB- en het PUT SPRITE-kommando te zijn doorgenomen.

ON SPRITE GOSUB geeft de mogelijkheid om het programma tijdelijk te onderbreken wanneer twee sprites met elkaar in botsing komen. Achter ON SPRITE GOSUB dient hiertoe een regelnummer te zijn vermeld. Dit regelnummer is het nummer van de eerste regel van de subroutine die in geval van een sprite-botsing dient te worden geactiveerd. Bij uitvoering van ON SPRITE GOSUB 'onthoudt' de computer slechts naar welk regelnummer bij een botsing dient te worden gesprongen. De daadwerkelijke sprong vindt pas bij botsing plaats. Een voorbeeld:

```
NEW
Ok
10 ON SPRITE GOSUB 100
20 SCREEN 2,1
30 SPRITE$(0)=STRING$(8,255)
40 PUT SPRITE 0,(100,100),1,0
50 SPRITE ON
60 FOR I=1 TO 200 STEP 2
70 PUT SPRITE 1,(110,I),1,0
80 NEXT I
90 GOTO 50
100 PUT SPRITE 0,,15
110 BEEP
120 PUT SPRITE 0,,1
130 SPRITE OFF
140 RETURN
RUN
```

Op het beeldscherm zijn twee zwarte sprites te zien, één stilstaande (40) en één vertikaal bewegende (60-80). Wanneer een botsing tussen deze twee sprites plaatsvindt, dan klinkt een attentiesignaal (110) en flitst de stilstaande sprite even wit op (100 en 120). Vervolgens wordt de mogelijkheid tot het onderbreken van het programma in verband met een botsing uitgezet (130) en wordt de for-next-lus afgemaakt. Aan het einde van de for-next-lus wordt de mogelijkheid tot het onderbreken van het programma voor een sprite-botsing weer aangezet (50).

Opmerkingen:

– Tijdens het uitvoeren van de onderbrekings-subroutine wordt automatisch door de computer een SPRITE STOP uitgevoerd. Op het moment dat de RETURN uit het onderbrekingsgedeelte wordt uitgevoerd, wordt weer een SPRITE ON uitgevoerd tenzij in de subroutine zelf een SPRITE STOP of een SPRITE OFF werd uitgevoerd.
De automatische SPRITE STOP heeft weinig nut. Omdat een botsing van sprites ook een tweede keer in de subroutine wordt opgemerkt, 'onthoudt' de computer meestal meteen een tweede botsing waardoor de RETURN uit de subroutine onmiddellijk resulteert in een nieuw aanspreken van de subroutine. Haal in het bovenstaande voorbeeld regel 130 er maar eens uit en probeer het programma nog eens. Het programma zal blijven 'hangen' in de onderbrekingssubroutine.
Om te voorkomen dat een dergelijke fout optreedt, kunnen twee acties worden ondernomen:

– of één van de botsing-veroorzakende sprites wordt in het allereerste kommando onmiddellijk op een plaats op beeldscherm gezet waar geen botsing meer wordt veroorzaakt

–of in de subroutine wordt een SPRITE OFF uitgevoerd die pas weer met een SPRITE ON teniet wordt gedaan nadat één van de botsing-veroorzakende sprites zo is verplaatst dat er geen sprake van een botsing meer is.

Opmerkingen:

– Tijdens het uitvoeren van de ERROR-routine wordt automatisch een SPRITE STOP uitgevoerd. Bij de RESUME wordt dan weer een SPRITE ON uitgevoerd tenzij de ERROR-routine zelf een SPRITE STOP of een SPRITE OFF uitvoerde.

- Wanneer een programma ten einde is, wordt automatisch een SPRITE OFF uitgevoerd.

- Een kommando wordt altijd in het geheel uitgevoerd voordat een onderbreking plaatsvindt.

- De RETURN in de onderbrekingsroutine mag elke voor RETURN toegestane vorm aannemen.

- Alleen twee zich geheel of gedeeltelijk in het zichtbare schermgebied bevindende sprites kunnen een botsingonderbreking veroorzaken. Deze sprites dienen zich dan beslist niet te bevinden achter een transparant waarop een sprite met y-coördinaat 208 (216 onder SCREEN 4...8 voor de MSX-2) is geplaatst (dit is een waarde voor de y-coördinaat waarbij alle op achterliggende transparanten geplaatste sprites onzichtbaar worden). Ook dienen zij niet geheel te zijn weggevallen in verband met het feit dat er zich te veel sprites op dezelfde horizontale lijn bevinden.

- Met ON SPRITE GOSUB (0) kan de mogelijkheid tot het detecteren van sprite-botsingen worden uitgeschakeld.

- Transparant gekleurde sprites (kleur 0) (of sprite-delen in MSX-2) kunnen geen met ON SPRITE GOSUB gedetecteerde botsingen veroorzaken.

- Onder MSX-2 kunnen sprites zo worden ingericht, dat een botsing met zo'n sprite nooit wordt gedetecteerd; zie PUT SPRITE.

SLEUTELWOORD # ON STOP GOSUB

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **ON STOP GOSUB is afkorting van on stop go to subroutine – bij stop ga naar onderprogramma (subroutine)**

schrijfwijze

```
ON STOP GOSUB[<REGELNUMMER>]

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

betekenis

Voorafgaand aan de behandeling van ON STOP GOSUB dient de werking van ON KEY GOSUB en STOP (tweede gedeelte) grondig te zijn doorgenomen.

De ON STOP GOSUB heeft hoegenaamd dezelfde funktie als de ON KEY GOSUB. De verschillen zijn:

– er is slechts één funktietoets die wordt afgevangen, namelijk de CONTROL-STOP (CTRL en STOP tegelijk ingedrukt)
– achter ON STOP GOSUB behoeft dus ook maar één regelnummer te worden vermeld
– er behoeft geen funktietoetsnummer te worden gegeven. Dus:

activeren van de STOP-toets	`STOP ON`	(vgl. KEY...ON)
deactiveren van de STOP-toets	`STOP OFF`	(vgl. KEY...OFF)
ophouden van de gevraagde onderbreking	`STOP STOP`	(vgl. KEY...STOP)

De ON STOP GOSUB biedt de mogelijkheid om een programma ononderbreekbaar te maken; de normale funktie van CONTROL-STOP, namelijk het onderbreken van een programma, kan worden gewijzigd. Ga na dat het volgende voorbeeld ononderbreekbaar is:

```
NEW
Ok
10 ON STOP GOSUB 1000:STOP ON
20 PRINT "ONONDERBREEKBAAR-";
30 GOTO 20
1000 RETURN
RUN
```

Het beeldscherm wordt met de tekst ONONDERBREEKBAAR– volgeplaatst; het intoetsen van CONTROL-BREAK helpt niet. Er is maar één manier om dit programma te onderbreken: de computer eerst uit en dan weer aan zetten.

Opmerkingen:

- Tijdens het uitvoeren van de door ON STOP GOSUB opgeroepen subroutine wordt door de computer automatisch een STOP STOP uitgevoerd. Op het moment dat de RETURN wordt uitgevoerd, wordt dan weer automatisch een STOP ON uitgevoerd tenzij in de betreffende subroutine een STOP STOP of een STOP OFF werd uitgevoerd.

- Tijdens het uitvoeren van de ERROR-routine wordt automatisch een STOP STOP uitgevoerd. Bij de RESUME wordt dan weer een STOP ON uitgevoerd, tenzij de ERROR-routine een STOP STOP of STOP OFF uitvoerde.

- Wanneer het programma ten einde is wordt automatisch een STOP OFF uitgevoerd.

- De RETURN in de onderbrekings routine mag elk voor RETURN toegestane vorm hebben.

- Met ON STOP GOSUB (0) schakelt men de mogelijkheid tot het aanroepen van een subroutine met CONTROL-STOP uit.

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **STRIG is samentrekking van shot en trigger (schot en trekker) vrij vertaald: ON STRIG GOTO is bij overhalen van trekker ga naar...**

schrijfwijze

```
ON STRIG GOSUB[<REGELNUMMER>][,[,]<REGELNUMMER>]

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

betekenis

Voorafgaand aan de behandeling van dit kommando dienen ON KEY GOSUB en STRIG grondig te zijn doorgenomen.

ON STRIG GOSUB heeft dezelfde funktie als ON KEY GOSUB, maar dan met betrekking tot resp. de spatiebalk, vuurknop 1 van joy-stick 1, vuurknop 1 van joy-stick 2, vuurknop 2 van joy-stick 1 en vuurknop 2 van joy-stick 2 als 'funktietoetsen'. Voorbeeld:

```
NEW
Ok
10 STRIG(0) ON
20 ON STRIG GOSUB 1000
30 GOTO 30
1000 PRINT "PANG!!!":RETURN
RUN
```

Steeds wanneer de spatiebalk wordt ingedrukt, wordt de tekst PANG!!! afgedrukt op het beeldscherm. Blijkbaar wordt de oneindige loop op regel 30 dan even onderbroken ten behoeve van de subroutine op regel 1000.

Nog een voorbeeld:

```
NEW
Ok
10 FOR I=1 TO 4:STRIG(I) ON:NEXT I
20 ON STRIG GOSUB ,1000,2000,3000,4000
30 GOTO 30
1000 PRINT "JOY-STICK 1 VUURKNOP 1":RETURN
2000 PRINT "JOY-STICK 2 VUURKNOP 1":RETURN
3000 PRINT "JOY-STICK 1 VUURKNOP 2":RETURN
4000 PRINT "JOY-STICK 2 VUURKNOP 2":RETURN
RUN
```

Dit voorbeeld is alleen zinvol indien er minimaal één joy-stick is aangesloten. Steeds wanneer een vuurknop wordt ingedrukt, wordt dit door de betreffende subroutine gemeld. De spatiebalk is ditmaal niet geactiveerd.

Opmerkingen:

- Tijdens de uitvoering van de door ON STRIG GOSUB opgeroepen subroutine wordt door de computer automatisch een STRIG... STOP uitgevoerd voor de betreffende vuurknop. Op het moment dat de RETURN wordt uitgevoerd, wordt weer automatisch een STRIG...ON uitgevoerd tenzij de betreffende subroutine de vuurknop deactiveerde met STRIG...STOP of STRIG...OFF.
- Tijdens het uitvoeren van de ERROR-routine wordt voor elke actieve vuurknop automatisch een STRIG...STOP uitgevoerd. Bij de RESUME worden de betreffende vuurknoppen dan weer geactiveerd, tenzij een vuurknop in de ERROR-routine met een STRIG...OFF of STRIG...STOP werd gedeactiveerd.
- Wanneer het programma ten einde is, wordt automatisch een STRIG...OFF voor elke vuurknop uitgevoerd.
- Een kommando wordt altijd in zijn geheel afgerond voordat onderbreking plaatsvindt.
- De RETURN in een onderbrekings-subroutine mag elke voor RETURN toegestane vorm hebben.
- Het gebruik van een regelnummer 0 in ON STRIG GOSUB heeft tot gevolg dat een onderbreking voor de betreffende vuurknop niet meer mogelijk is.
- ON STRIG GOSUB (0) schakelt de mogelijkheid tot onderbreken voor de spatiebalk uit.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **OPEN is openstellen/openmaken**

schrijfwijze

```
                 ┌FOR INPUT ┐
                 │----------│
OPEN<BESTANDSNAAM>│FOR OUTPUT│ AS[#]<KANAAL>
                 │----------│
                 └FOR APPEND┘  --> ALLEEN MSX-2 OF DISK BASIC
```

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

<KANAAL>:::=<N>

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

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

betekenis

Het is noodzakelijk dat eerst MAXFILES wordt bestudeerd.

Om een bestand, een verzameling van bij elkaar behorende gegevens, te kunnen gebruiken of inrichten, dient het eerst te worden geopend. Dit openen geschiedt met het OPEN-kommando en dient ondermeer om het bestand een KANAALNUMMER toe te kennen waardoor het gemakkelijk is te benaderen.

Als eerste gegeven na het OPEN-sleutelwoord dient de naam van het betreffende bestand te worden opgegeven. Deze naam bestaat uit één of uit twee gedeelten, van elkaar gescheiden door een dubbele punt. Het eerste gedeelte van de bestandsnaam geeft aan, op welk randapparaat het bestand staat of moet komen te staan. Het tweede gedeelte geeft aan hoe het betreffende bestand heet of moet worden genoemd.

Als tweede gegeven mag worden gespecificeerd op welke wijze het betreffende bestand moet worden gebruikt. FOR INPUT wil zeggen dat gegevens naar de computer toe dienen te worden ingevoerd vanuit het bestand en FOR OUTPUT wil zeggen dat gegevens vanuit de computer dienen te worden uitgevoerd naar het betreffende bestand.
FOR APPEND (MSX-2/DISK) wil zeggen dat een reeds bestaand bestand vanuit de computer met gegevens wordt aangevuld.
FOR INPUT en FOR OUTPUT behoeven niet te worden opgenomen wanneer het zondermeer duidelijk is op welke wijze het bestand dient te worden benaderd.

Als derde gegeven dient het kanaalnummer te worden aangegeven waaronder het eenmaal geopende bestand verder kan worden benaderd. Dit kanaalnummer is minimaal gelijk aan 1 en maximaal gelijk aan de waarde die de systeemvariabele MAXFILES (standaard 1) is gegeven. Een eventuele decimale fractie wordt verwaarloosd.

Allereerst volgt hieronder een tabel waarin diverse voorwaarden zijn opgenomen in verband met het OPEN-kommando:

rand-appa-raat	betekenis	benadering	opname FOR INPUT/ OUTPUT verplicht?	bestands-naam verplicht?	kommentaar
CAS:	cassette-recorder	FOR IN-PUT FOR OUT-PUT	ja	nee*	een cassetterecorder moet zijn aangesloten en ingesteld.
GRP:	grafisch scherm	FOR OUT-PUT	nee	nee, zinloos	een grafisch scherm moet m.b.v. het SCREEN-kommando zijn geactiveerd.
CRT:	alfanu-meriek scherm	FOR OUT-PUT	nee	nee, zinloos	een alfanumeriek scherm moet m.b.v. het SCREEN-kommando zijn geactiveerd.
LPT:	printer	FOR OUT-PUT	nee	nee, zinloos	een printer moet zijn aangesloten.
(niets)	cassette-recorder**	zie CAS:	zie CAS:	ja	zie CAS:
MEM:	RAM-disk	FOR IN-PUT FOR OUT-PUT FOR AP-PEND	ja	nee***	geldt alleen voor MSX-2. Zie ook CALL ME-MINI, CALL MFILES, CALL MNAME en CALL MKILL.

*Indien geen bestandsnaam wordt opgegeven:
- bij FOR OUTPUT wordt een bestand zonder naam aangemaakt.
- bij FOR INPUT wordt het eerste op tape aangetroffen bestand geopend.

** geldt alleen indien geen schijveneenheid staat aangesloten. Wanneer wel een schijveneenheid is aangesloten, wordt automatisch de schijf benaderd. Zie deel 2.

*** Indien geen bestandsnaam wordt opgegeven:
- bij FOR OUTPUT wordt een bestand zonder naam aangemaakt.
- bij FOR INPUT/APPEND wordt getracht, een naamloos bestand te openen. Indien een dergelijk bestand niet bestaat, wordt een foutmelding gegeven.

Per randapparaat worden de diverse mogelijke vormen van het OPEN-kommando nu nader behandeld:

Cassetterecorder

Om met MSX-basic optimaal van de cassetterecorder gebruik te maken is het noodzakelijk dat de cassetterecorder met drie kabeltjes is verbonden aan de computer, te weten:
- de microfoonkabel. Via deze kabel worden gegevens vanuit de computer naar de cassetterecorder gebracht
- de oortelefoonkabel. Via deze kabel worden gegevens vanuit de cassetterecorder naar de computer gebracht
- de afstandsbedieningskabel. Via deze kabel wordt de cassetterecorder door de computer aan- en uit gezet.

Enkele opmerkingen in verband met de cassetterecorder in koppeling aan de computer:

- wanneer gegevens naar de cassetterecorder worden gestuurd vanuit de computer, dan dient de cassetterecorder op opnemen te staan;
- wanneer gegevens vanuit de cassetterecorder naar de computer toe worden gestuurd, dan dient de cassetterecorder op afspelen te staan;
- indien gegevens vanuit de computer naar een randapparaat worden gestuurd dan spreekt men van het SCHRIJVEN van gegevens;
- indien gegevens vanuit een randapparaat naar de computer toe worden gestuurd dan spreekt men van het LEZEN van gegevens;
- bij het afspelen van de cassetterecorder dient het volume op ongeveer 80% te staan
- het MSX-basic schrijft geen controlemaatregelen voor bij het teruglezen van gegevens. Slecht materiaal kan ervoor zorg dragen dat gegevens verminkt worden ingelezen zonder dat daar door de computer melding van wordt gemaakt. Zorg dat u altijd cassettebanden van een goed merk gebruikt. Belangrijk is het om te weten of de cassettebanden geschikt zijn voor de betreffende recorder. Gebruik bij een eenvoudige recorder, tenzij anders aangegeven, altijd een ferro-kwaliteit cassettebandje (standaardkwaliteit). Het

gebruik van chroomdioxidebanden en dergelijke op een niet daarvoor geschikt apparaat geeft meestal slechts negatieve resultaten. Indien u dat kunt, maak dan zeer regelmatig de opnamekop en de weergavekop van uw cassetterecorder met een wattenstaafje en met zuivere alkohol (geen spiritus) schoon

– indien u een cassetteband wilt heen- of terugspoelen dan kan dat meestal niet doordat de computer de cassetterecorder heeft uitgeschakeld. Trek in dat geval de afstandsbediening aan de kant van de cassetterecorder even los of gebruik het MOTOR-kommando (zie aldaar).

Het schrijven van gegevens naar de cassetterecorder

Met het SAVE, CSAVE en BSAVE-kommando kunnen (programma-) gegevens naar de cassetterecorder worden geschreven. Zie de behandeling van deze kommando's. Indien men andere dan programma- of machinegegevens naar de cassetterecorder wilt schrijven, dan dient eerst een bestand te worden toegewezen en wel als volgt:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
```

Bij uitvoering van dit kommando wordt op de cassetterecorder (CAS:) een bestand toegewezen, genaamd TEST. CAS: mogen we in dit geval eventueel weglaten terwijl we voor TEST elke andere naam mogen invullen die niet groter is dan 6 letters. Een eventueel teveel aan letters wordt gewoon genegeerd.

In de volgende programmaregels kunnen vervolgens diverse gegevens naar dit bestand toe worden geschreven. Hiertoe dient men gebruik te maken van het PRINT-kommando. Een eenvoudig voorbeeld wordt hier gegeven; zie voor het gebruik van het PRINT-kommando verder de behandeling onder PRINT.

```
20 LINE INPUT "GEGEVENS:";A$
30 IF A$="" THEN GOTO 100
40 PRINT #1,A$
50 GOTO 20
```

Met dit programmagedeelte kunnen de op regel 20 ingegeven gegevens naar cassette worden geschreven. Het werkelijke schrijven gebeurt op regel 40. Merk op dat op regel 40 alleen maar het kanaalnummer

behoeft te worden genoemd; na het OPEN-kommando weet de computer alleen door dit kanaalnummer naar welk randapparaat de gegevens dienen te worden geschreven.
Op regel 30 wordt afgevraagd of er misschien niets werd ingegeven. In dat geval dient het programma te worden verlaten. Voordat het programma wordt beëindigd, dient eerst het betreffende bestand dat zojuist werd geopend, weer te worden afgesloten. Dit gebeurt via een CLOSE-kommando. Indien slechts een CLOSE wordt gegeven, dan worden alle kanalen achter elkaar afgesloten. Indien slechts één of enkele kanalen dienen te worden afgesloten, dan kan dat door achter het CLOSE-kommando de betreffende kanalen op te nemen, van elkaar gescheiden door een komma. In ons voorbeeld programmeren we:

```
100 CLOSE
110 STOP
RUN        (enige tijd verstrijkt)
GEGEVENS:DEZE GEGEVENS WORDEN STRAKS
GEGEVENS:WEER DOOR DE COMPUTER VAN
GEGEVENS:CASSETTEBAND TERUGGELEZEN
GEGEVENS:TEST 0123456789
GEGEVENS:                    (alleen de RETURN-toets wordt ingegeven)
Break in 110                                  (enige tijd verstrijkt)
Ok
```

Vergeet niet om, voordat het programma wordt gestart, de cassetteband te plaatsen, naar de juiste positie te spoelen en om de cassetterecorder op opnemen te zetten.

Nadat het programma is uitgevoerd (en er mogen natuurlijk andere gegevens worden ingevoerd), is een bestand op de cassetteband aangemaakt onder de naam TEST en met de ingegeven gegevens daarin geschreven.

Indien achter CAS: geen bestandsnaam werd opgegeven, dan werd een bestand zonder naam op de cassetteband aangemaakt.

Het lezen van gegevens van de cassetterecorder

Met het LOAD, CLOAD en BLOAD-kommando kunnen (programma-) gegevens van de cassetteband worden gelezen. Zie hiervoor de betreffende behandelingen. Indien men andere dan programma- of machinekodegegevens vanuit de cassetterecorder in wil lezen, dan dienen deze gegevens zoals in het vorige gedeelte behandeld als bestand op cassette-

band te zijn gezet. Eén uitzondering vormt een programma dat door SAVE op cassette is gezet. Dit bestand kan met OPEN worden geopend geopend en vervolgens worden ingelezen.

In het volgende voorbeeld gaan we de gegevens die zojuist in bestand TEST zijn opgeslagen, weer teruglezen. Hiertoe dienen we het bestand eerst te openen:

```
NEW
Ok
10 OPEN "CAS:TEST" FOR INPUT AS 1
```

Bij uitvoering van dit kommando mag CAS: eventueel worden weggelaten. Indien CAS: niet wordt weggelaten, mag de bestandsnaam (TEST) worden weggelaten; in dat geval wordt het eerste op de cassetteband voorkomende bestand in behandeling genomen.

Vervolgens programmeren we:

```
20 IF EOF(1) THEN CLOSE:STOP
30 LINE INPUT #1,A$
40 PRINT A$
50 GOTO 20
RUN              (enige tijd verstrijkt)
DEZE GEGEVENS WORDEN STRAKS
WEER DOOR DE COMPUTER VAN
CASSETTEBAND TERUGGELEZEN
TEST 0123456789
Break in 20
Ok
```

Vergeet niet om vooraf aan dit programma eerst de cassetteband naar het begin van het bestand terug te spoelen (gebruik de bandteller) en de cassetterecorder op afspelen te zetten. Zet het volume op ongeveer 80%.

We zien één nieuw sleutelwoord in regel 40, namelijk het sleutelwoord EOF. De funktie EOF dient te worden gevolgd door een numerieke uitdrukking die een kanaalnummer bevat. Een decimale fractie wordt eventueel verwaarloosd. De funktie geeft als resultaat een 0 indien het bestand nog niet ten einde is en een −1 indien er geen gegevens meer in het bestand beschikbaar zijn.

Om bestanden optimaal te kunnen gebruiken op cassette, is het na deze behandeling noodzakelijk om de gevorderde behandelingen van

PRINT, INPUT en LINE INPUT door te nemen tot zover zij betrekking hebben op het lezen en schrijven van gegevens van of naar randapparatuur.

Grafisch scherm

Een geactiveerd grafisch scherm wordt door MSX-basic ook als randapparaat beschouwd. Een bestandsnaam mag worden gespecificeerd bij het OPEN-kommando maar is zinloos.
De naar het grafische scherm geschreven gegevens worden vanaf het laatst getekende punt op het beeldscherm geplaatst. Het laatst getekende punt vormt de linker bovenhoek van de 8 bij 8 puntenmatrix waarin het eerste karakter kan worden geplaatst. Een voorbeeld:

```
NEW
Ok
10 SCREEN 2                          (probeer ook eens SCREEN 3)
20 OPEN "GRP:" AS 1
30 PSET (0,0)
40 PRINT #1,"DIT IS EEN TEST"
50 CIRCLE (100,100),50
60 GOTO 60
RUN
```

De tekst DIT IS EEN TEST wordt op het grafische beeldscherm geplaatst te zamen met de tekening van een cirkel.

Voor MSX-2, SCREEN 5 tot en met 8 geldt dat de laatste door welk kommando dan ook ingestelde logische operator ook voor de met PRINT # geplaatste karakters geldt.

Alfanumeriek scherm

Een geactiveerd alfanumeriek scherm wordt door MSX-basic eveneens als apart randapparaat beschouwd. Het printen op een alfanumeriek scherm via een kanaal heeft precies hetzelfde effect als het normale printen op een alfanumeriek beeldscherm. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "CRT:" AS 1
20 CLS
30 PRINT #1,"DIT IS EEN TEST"
RUN
```

de tekst DIT IS EEN TEST wordt linksboven in beeld geplaatst. Regel 10 kan vervallen wanneer regel 30 als volgt wordt veranderd;

```
30 PRINT "DIT IS EEN TEST"
```

Toch heeft het openen van het alfanumeriek beeldscherm zin; hierop komen we later terug.

Printer

Ook de printer is een randapparaat. Met het LPRINT-kommando kunnen we gegevens naar de printer sturen; zie de betreffende behandeling.
Door de printer te openen op een kanaal, kunnen we hetzelfde effect bereiken. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "LPT:" AS 1
20 PRINT #1,"DIT IS EEN TEST"
RUN
```

Indien een printer is aangesloten, verschijnt de tekst DIT IS EEN TEST op de printer. Indien regel 10 wordt verwijderd, kan regel 20 worden vervangen door:

```
20 LPRINT "DIT IS EEN TEST"
```

Toch heeft het apart openen van de printer in sommige gevallen nut. Bijvoorbeeld:

```
NEW
Ok
10 MAXFILES=2
20 OPEN "CAS:TEST" FOR INPUT AS 1
30 INPUT "PRINTER OF BEELDSCHERM (P/B)";A$
40 IF A$="P" THEN OPEN "LPT:" AS 2:GOTO 70
50 IF A$="B" THEN OPEN "CRT:" AS 2:GOTO 70
60 PRINT "FOUTE INGAVE":GOTO 30
70 IF EOF(1) THEN CLOSE:STOP
80 LINE INPUT #1,A$:PRINT #2,A$:GOTO 70
RUN                                    (enige tijd verstrijkt)
PRINTER OF BEELDSCHERM (P/B)? B
DEZE GEGEVENS WORDEN STRAKS
WEER DOOR DE COMPUTER VAN
CASSETTEBAND TERUGGELEZEN
TEST 0123456789
Break in 70
Ok
```

Als invoerbestand werd het bestand TEST zoals dat in het voorbeeld bij het schrijven naar cassetterecorder werd gegeven, gebruikt. Vergeet niet om voorafgaand aan dit programma de cassetterecorder op afspelen te zetten nadat de cassette tot op het juiste punt is teruggespoeld.

Op regel 10 wordt bepaald dat er twee verschillende bestanden naast elkaar worden geopend. Op regel 20 wordt dan in ieder geval het bestand TEST van cassetteband geopend. Op regel 30 wordt de keuze tussen het afdrukken van de gegevens op beeldscherm of printer mogelijk gemaakt. Afhankelijk van de ingave op deze regel wordt de printer of het alfanumerieke beeldscherm geopend. Vervolgens worden over kanaal 2 de vanuit de cassetterecorder ingelezen gegevens afgedrukt, hetzij op beeldscherm, hetzij op de printer.

Merk op dat een dergelijke keuze veel moeilijker is te programmeren indien we het PRINT-kommando zonder kanaalnummer en het LPRINT-kommando naast elkaar zouden gebruiken. In dit geval (en in vele andere denkbare gevallen) is het zinvol om een alfanumeriek scherm of een printer apart te openen.

RAM-disk (alléén MSX-2)

De RAM-disk bestaat uit de "onderste" 32 kilobytes RAM-geheugen van uw computer. Dit geheugenblok is normaal niet bruikbaar maar kan onder MSX-2 als een soort schijveneenheid worden gebruikt. Wel is het zo dat alle gegevens van de RAM-disk verloren gaan wanneer de spanning wordt uitgeschakeld. Daarom is de RAM-disk ook alleen maar voor tijdelijke opslag geschikt.

De RAM-disk werkt ongeveer als een cassetterecorder met dit verschil dat de uitwisseling van gegevens veel sneller gaat. Alle hiervoor gegeven voorbeelden met betrekking tot de cassetterecorder kunnen ook voor de RAM-disk toepasbaar gemaakt worden door "CAS:TEST" te vervangen door "MEM:TEST". Maak de RAM-disk wel eerst toegankelijk met behulp van CALL MEMINI.

Nog een verschil is, dat de RAM-disk ook FOR APPEND kan worden geopend waarna het betreffende bestand met gegevens kan worden AANGEVULD.

Zie in verband met de RAM-disk ook de behandeling van CALL MEMINI, CALL MNAME, CALL MFILES en CALL MKILL.

moeilijkheidsgraad . . **vrij moeilijk, kennis van de funkties van de poorten van het computersysteem is vereist**
soort . . . . . **KOMMANDO**
afkomst . . . . . **OUT is uit**

schrijfwijze

```
OUT<POORT>,<TE VERZENDEN WAARDE>

<POORT>::=<N>

<TE VERZENDEN WAARDE>::=<N>

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

betekenis

Een byte-waarde kan op een poortbuffer worden geplaatst.
Het poortnummer mag niet kleiner zijn dan –32768 en niet groter dan 65535. Een eventuele decimale fractie wordt verwaarloosd. Indien het poortnummer negatief is, wordt 65536 bij dit nummer opgeteld alvorens deze poort wordt aangesproken.
De via de poort te verzenden waarde dient in één byte te passen en mag als zodanig niet kleiner zijn dan 0 en niet groter dan 255, terwijl een decimale fractie wordt verwaarloosd.

Ondeskundig gebruik van dit kommando kan leiden tot vreemde effecten of het 'ophangen' van het systeem. In dat geval is er maar één remedie: computer uitzetten en daarna weer aanzetten.

Bijvoorbeeld:

```
OUT 168,255
```

De computer staat vast; uitzetten en weer aanschakelen helpt.

Voor een goed gebruik van OUT is een gedetailleerde kennis van de hardware-opbouw van een MSX-computer noodzakelijk. Daarbij dienen de diverse componenten van een MSX-computer ook softwaretechnisch te worden beheerst. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop niet verder ingegaan.

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **PAD is blocknote**

schrijfwijze

```
PAD(<N>)

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

betekenis

Aan de MSX-computers kan diverse randapparatuur worden aangesloten. Enkele randapparaten kunnen met behulp van PAD worden bestuurd en afgevraagd.
Deze randapparaten zijn:

Het aanraakpaneel (touch pad)

Het aanraakpaneel (PAD) is een paneel dat met een speciale stift moet worden aangeraakt. Op deze wijze kan een tekening worden gedigitaliseerd (in getallen worden omgezet) en worden door gezonden aan de computer. Met de funkties PAD (0....7) kan de status van het aanraakpaneel worden afgevraagd.

De lichtpen (light pen) (MSX-2)

Ook met de lichtpen kan een tekening worden gedigitaliseerd. De lichtpen wordt ter aanwijzing echter direkt op het beeldscherm gebruikt. Met PAD (8....11) kan de lichtpen worden afgevraagd.

De Muis (mouse) (MSX-2)

Met een muis worden bewegingen gedigitaliseerd. Een muis is een apparaat dat over een tafel kan worden bewogen. Onderaan de muis draait een kogel mee. De draaiing van die kogel wordt vertaald in een relatieve X- enY- coördinaat. Deze coördinaten worden vervolgens aan de computer doorgezonden. Met PAD (12....19) kan de muis worden afgevraagd.

Een volgbol (tracker-ball) (MSX-2)

Een volgbol is een kogel die met de vlakke hand in elke richting kan worden gedraaid. Deze draaiing wordt in een relatieve X- en Y- coördinaat vertaald, net zoals bij de muis. Een volgbol is in feite een tussenvorm van een joy-stick en een muis. Ook de volgbol wordt met PAD-(12...19) afgevraagd.

PAD (0....19) geeft altijd een integere waarde terug. In de volgende tabellen wordt aangegeven, welke de betekenissen van de diverse toepassingen zijn:

Tabel 1, PAD (0....7), het aanraak paneel (touch pad)

1e joy-stick aansluiting	2e joy-stick aansluiting	waarde	betekenis
PAD (0)	PAD (4)	0	er is geen aanraking gedetecteerd.
		-1	er is wel een aanraking gedetecteerd.
PAD (1)	PAD (5)	0....255	X-coördinaat van het aanraakpunt. Alleen geldig indien PAD (0) of PAD (4) eerst wordt afgevraagd en gelijk is aan -1
PAD (2)	PAD (6)	0....255	Y-coördinaat van het aanraakpunt. Alleen geldig indien eerst PAD (0) of PAD (4) werd afgevraagd en is gelijk aan -1
PAD (3)	PAD (7)	0	schakelaar op aanraakpaneel is niet ingedrukt.
		-1	schakelaar op aanraakpaneel is wel ingedrukt.

Tabel 2, PAD (8....11), de lichtpen (light pen) (alleen MSX-2)

PAD ( )	waarde	betekenis
PAD (8)	0	lichtpen is nog niet gereed.
	-1	lichtpen is gereed.
PAD (9)	0...1023	X-coördinaat van de lichtpen. PAD (8) moet eerst zijn afgevraagd en gelijk zijn aan -1.
PAD (10)	0...1023	Y-coördinaat van de lichtpen PAD (8) moet eerst zijn afgevraagd en gelijk zijn aan -1.
PAD (11)	0	lichtpen-schakelaar is niet ingedrukt.
	-1	lichtpen-schakelaar is wel ingedrukt.

Tabel 3, PAD (12....19), de muis of de volgbol (mous of tracker-ball) (alleen MSX-2)

1e joystick aansluiting	2e joystick aansluiting	waarde	betekenis
PAD (12)	PAD (16)	-1	geeft altijd -1 maar moet vóór PAD (13) /PAD (17) of PAD (14)/PAD (18) worden afgevraagd.
PAD (13)	PAD (17)	0....255	X-coördinaat van de muis. Alleen geldig nadat PAD (12) of PAD (16) is afgevraagd.
PAD (14)	PAD (18)	0...1023	Y-coördinaat van de muis. Alleen geldig nadat PAD (12) of PAD (16) is afgevraagd.
PAD (15)	PAD (19)	0	geen.

De schakelaars op de muis kunnen met STRIG (....) worden afgevraagd als bij joy-sticks, zie aldaar.

Tabel 4, geldigheidsvoorwaarden

PAD (...) geeft slechts een zinvolle waarde...	indien PAD (...) eerder werd afgevraagd en gelijk is aan -1
1	0
2	0
5	4
6	4
9	8
10	8
13	12
14	12
17	16
18	16
	MSX-2
...dienen deze waarden te worden afgevraagd. De juiste waarden kunnen anders worden verminkt	Zo snel mogelijk na deze afvraging ...

Het volgende voorbeeld is alleen zinvol met een aangesloten aanraakpaneel en geeft elke keer dat het aanraakpaneel wordt beroerd, een cirkel op de corresponderende beeldscherm-plaats.

```
NEW
Ok
10 SCREEN 2
20 IF PAD(0)=0 THEN 20
30 CIRCLE (PAD(1),PAD(2)),20
40 GOTO 20
RUN
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PAINT is schilderen**

schrijfwijze

```
PAINT<LOCATIE>[,[<SCHILDERKLEUR>][,<RANDKLEUR>]]\PAINT[<...>,]

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<SCHILDERKLEUR>::=<N>

<RANDKLEUR>::=<N>

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

betekenis

Met dit kommando kunnen binnen een tekening op scherm gehele vlakken worden ingekleurd. Door slechts een punt aan te geven, eventueel een schilderkleur te kiezen en eventueel een randkleur te kiezen, wordt het betreffende oppervlak ingekleurd.

Voor goed begrip van dit kommando is het noodzakelijk om de behandeling van PSET grondig door te nemen.

De wijze waarop PAINT werkt, kunnen we het beste vergelijken met de wijze waarop we zelf een tekening inkleuren.
Wanneer we een oppervlak gaan inkleuren is het noodzakelijk om eerst het juiste kleurpotlood te kiezen, om de juiste kleur te bepalen. Vervolgens dienen we te bepalen waar we het kleurpotlood op papier gaan neerzetten. Als laatste moeten we beslissen tot aan welke lijnen we gaan inkleuren.
Met het PAINT-kommando geven we het punt waar het kleurpotlood moet worden neergezet aan met de betreffende locatie. Met de schilderkleurkodering bepalen we de kleur. Met de randkleurkodering bepalen we de kleur van de lijnen waarbinnen de computer met kleuren dient te blijven.

Indien de schilderkleur wordt weggelaten, wordt aangenomen dat de schilderkleur gelijk is aan de op dat moment actieve voorgrondkleur. Indien de randkleur wordt weggelaten, wordt aangenomen dat ook deze kleur gelijk is aan de op dat moment actieve voorgrondkleur.

Indien SCREEN 2 of SCREEN 4 (MSX-2) is geactiveerd, dan is het overbodig om een randkleur te specificeren; de schilderkleur wordt altijd als randkleur genomen.

Enige voorbeelden:

```
NEW
Ok
5 COLOR 15,1,1
10 SCREEN 2
20 CIRCLE (111,111),75
30 CIRCLE (200,200),100
40 PAINT (160,160)
50 GOTO 50
RUN
```

Twee cirkels worden getekend. Het snij-oppervlak wordt ingekleurd.

```
10 SCREEN 3
40 PAINT (160,160),12,15
RUN
```

Alleen regels 10 en 40 werden veranderd. In lage resolutie worden de twee cirkels nu ingekleurd. Het snij-oppervlak wordt donkergroen gekleurd.

```
20 CIRCLE (111,111),75,12
30 CIRCLE (200,200),100,12
40 PAINT (160,160),4,12
RUN
```

Programmaregels 20, 30 en 40 werden vervangen. Twee groene cirkels worden getekend. Het snij-oppervlak wordt donkerblauw ingekleurd.

```
40 PAINT (160,160),,12
RUN
```

Alleen regel 40 werd vervangen. Het snij-oppervlak wordt met de op dat moment actieve voorgrondkleur ingekleurd.

Merk op dat het met PAINT voldoende is om ergens binnen het in te kleuren vlak een punt te prikken.

```
20 CIRCLE (111,111),75,0
30 CIRCLE (200,200),100,0
40 PAINT (160,160),12,0
RUN
```

De regels 20, 30 en 40 werden vervangen. De cirkels werden in de transparantkleur (onzichtbaar) getekend. Daarna werd het snijvlak ingekleurd. Alleen het ingekleurde snijvlak is zichtbaar. Dit laatste voorbeeld is uitstekend om te zien hoe de transparantkleur (kleurkode 0) funktioneel kan worden gebruikt.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **PDL is afkorting van paddle – schoep**

schrijfwijze

```
PDL(<PADDLE NUMMER>)

<PADDLE NUMMER>::=<N>

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

betekenis

Met deze funktie kan de waarde van een paddle worden afgevraagd. Een paddle bestaat uit een regelknop en meestal een vuurknop en kan aan één van de uitgangen van de computer worden aangesloten.

De numerieke uitdrukking tussen haakjes dient een waarde te hebben, gelijk aan 1, 2, 3, 4..., 12. Indien de tussen haakjes vermelde waarde gebroken is, wordt deze herleid naar de grootste gehele waarde, kleiner dan deze gebroken waarde.

De waarde tussen haakjes heeft de volgende betekenis:

PDL (oneven waarde) de eerste paddle wordt beschouwd

PDL (even waarde) de tweede paddle wordt beschouwd

De funktie heeft een waarde tot gevolg die minimaal gelijk is aan 0, maximaal gelijk is aan 255 en geheel is. De waarde 0 correspondeert met een geheel dichtgedraaide paddle, de waarde 255 correspondeert met een geheel opengedraaide paddle. Elke tussenliggende waarde correspondeert met een bepaalde stand van de regelknop.

Voorbeeld: (alleen zinvol met een aangesloten paddle)

```
NEW
Ok
10 LET A=PDL(1)*.09:IF AA=A THEN 10
20 CLS:LOCATE ,A,0:PRINT "====":LET AA=A:GOTO 10
RUN
```

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het computergeheugen is vereist**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **PEEK is gluren, kijken**

schrijfwijze

```
PEEK(<GEHEUGENADRES>)

<GEHEUGENADRES>::=<N>

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

betekenis

Met deze funktie kan de direkte inhoud van het computergeheugen worden opgevraagd. Achter het sleutelwoord PEEK dient hiertoe tussen haakjes het betreffende geheugenadres te worden opgenomen. Als resultaat wordt dan de inhoud van het betreffende byte numeriek weergegeven. Bijvoorbeeld:

```
NEW
Ok
10 FOR I=0 TO 99
20 PRINT PEEK(I);
30 NEXT I
RUN
```

Van de eerste honderd adressen wordt de inhoud numeriek op beeldscherm afgedrukt.

Het met PEEK toegankelijke geheugen wordt gevormd door:

- 0...32767 — 32 kilobyte basis ROM
- 32768...65535 — 32 kilobyte RAM, het voor MSX-basic beschikbare geheugen.

Het geheugenadres dat achter PEEK dient te worden gespecificeerd, mag niet kleiner zijn dan –32768 en niet groter dan 65535. Een eventuele decimale fractie wordt verwaarloosd. Indien het geheugenadres negatief is, wordt de waarde 65536 bij dit adres opgeteld voordat het geheugen wordt geadresseerd.

Indien het geheugenadres groter is dan 32767, dan wordt, indien er een decimale fractie is, deze naar BOVEN afgerond!

Het volgende programma tracht het ROM-geheugen karaktergewijs op het beeldscherm af te drukken. De geduldige toeschouwer zal tussen een wirwar van tekens regelmatig bekende delen herkennen en hele karaktertabellen zien voorbijkomen:

```
NEW
Ok
10 FOR I=0 TO 32767
20 IF PEEK(I)>32 THEN PRINT CHR$(PEEK(I));
30 NEXT I
RUN
```

Voor een goed gebruik van PEEK is gedetailleerde kennis van de opbouw van het geheugen van een MSX-computer noodzakelijk. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop verder niet ingegaan.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **PLAY is spelen, muziek maken**

schrijfwijze

```
PLAY(<STEMNUMMER>)

<STEMNUMMER>::=<N>

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

betekenis

Met de funktie PLAY kan worden afgevraagd of de toongenerator al klaar is met spelen. De toongenerator is een apart computertje binnen de MSX-computer en kan als zodanig nog bezig zijn met het afwerken van een stukje muziek terwijl de Ok-melding al is gegeven of terwijl het programma reeds verder gaat.
Tussen de haakjes dient het nummer van de stem in een numerieke uitdrukking te worden gegeven. Een eventuele decimale fractie wordt verwaarloosd. Het stemnummer mag gelijk zijn aan 0, 1, 2 of 3. Indien een 1, 2 of 3 wordt gebruikt, dan wordt de corresponderende stem afgevraagd. Indien stemnummer 0 wordt opgegeven, dan wordt *elke* stem afgevraagd.

Play geeft de waarde 0 indien de betreffende stem niet actief is en de waarde –1 indien de betreffende stem wel actief is. Bijvoorbeeld:

```
NEW
Ok
10 PLAY "CDEFGFEDCDEFGFEDC'
20 PRINT "KLAAR"
RUN
KLAAR                                (de muziek speelt echter nog)
Ok

15 IF PLAY(0) THEN 15                (regel wordt bijgevoegd)
RUN
KLAAR                                (de tekst 'klaar' wordt pas gegeven
Ok                                    nadat de muziek is uitgespeeld)
```

moeilijkheidsgraad . . **vrij moeilijk, een elementaire kennis van de muziektheorie is vereist**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PLAY is spelen, muziek maken**

schrijfwijze

```
PLAY<EERSTE STEM>[,<TWEEDE STEM>[,<DERDE STEM>]]

<EERSTE STEM>::=<A>

<TWEEDE STEM>::=<A>

<DERDE STEM>::=<A>

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

betekenis

Met het PLAY-kommando kunnen we op eenvoudige wijze een maximaal driestemmig stuk muziek vanaf een muziekmanuscript programmeren in de computer. Afhankelijk van het aantal stemmen dienen hiertoe achter het PLAY-kommando één, twee of drie alfanumerieke uitdrukkingen te zijn opgenomen.
Een geluids*effect*, bijvoorbeeld als begeleiding bij spelletjes, kan beter met het SOUND-kommando worden geprogrammeerd; zie aldaar.
De alfanumerieke uitdrukkingen dienen te zijn opgebouwd zoals de MML regels dat voorschrijven. MML (Music Macro Language) is te beschouwen als een aparte taal binnen MSX-basic, ontworpen door Digital Research, één van de grotere wereld-software producenten.

MML gaat uit van de volgende toonhoogte-symbolen:

C,D,E,F,G,A,B — deze letters stellen de namen van de noten voor van het te programmeren stuk muziek.

+, –, # — met een plus- of een min-teken achter de naam van de noot geeft men aan of deze halve toon moet worden verhoogd of verlaagd. Een plusteken komt dus overeen met een kruis en een minteken komt overeen met een mol. Dubbelmollen en dubbelkruizen zijn niet toegestaan. Het plus-teken mag door het echte kruis-teken worden vervangen.

Voorbeeld: de chromatische toonladder:

```
PLAY "CC+DD+EFF+GG+AA+B"
Ok
```

Achtereenvolgens worden de C, de Cis, de D, de Dis, de E, de F, de Fis, de G, de Gis, de A, de Ais en de B gespeeld door de computer.

Technisch heeft de volgende ingave exact hetzelfde effect:

```
PLAY "CD-DE-EFG-GA-AB-B"
Ok
```

Exact dezelfde klanken, voorstellende de C, de Des, de D, de Es, de E, de F, de Ges, de G, de As, de A, de Bes en de B, worden gespeeld.

Een zéér eenvoudig stukje driestemmige muziek:

```
NEW
Ok
10 PLAY "CDEFGABC","EFGABCDE","GABCDEFG"
RUN
Ok
```

Een zevental verschillende drieklanken wordt door de computer geproduceerd.

Door een getal (een integere konstante groter dan 0 en kleiner dan 65) achter de betreffende noot te plaatsen, bepaalt men de lengte van de noot. Een 1 staat voor een gehele noot, een 2 staat voor een halve noot, een 3 staat voor een derde noot etcetera.
In het voorbeeld worden enkele verschillende noten gespeeld; een hele, een halve, een kwart, een achtste, een zestiende en een tweeendertigste noot:

```
PLAY "C1D2E4F8G16A32"
Ok
```

Als standaard nootlengte wordt een kwartnoot aangenomen. Deze standaardlengte kan worden gewijzigd door het L-kommando. Met een L, gevolgd door een nootlengte (1 ... 64) wordt de standaard nootlengte voor de betreffende stem veranderd. Indien alleen de letter L wordt gebruikt, wordt een L4 aangenomen.

Het volgende voorbeeld resulteert in een snelle toonladder; de standaard nootlengte werd op een zestiende noot gezet:

```
PLAY "L16CDEFGAB"
Ok
```

Een rust wordt met het symbool R aangegeven, gevolgd door de duur van de rust (1 ... 64). Indien geen duur wordt opgegeven, wordt een kwartrust uitgevoerd. Voorbeeld: altijd is kortjakje ziek, eerste regel, tweestemmig:

```
PLAY "M6000S1L4CCGGAAGRL8FFFFL4EEDDCR","S1L4RREEFFERL8DD
DDL4 CCGGER"
Ok
```

Op de M6000S1 kodering die in beide stemmen werd gebruikt, gaan we later in. Voorlopig is het van belang om slechts te weten, dat deze toevoeging ervoor zorgt dat de tonen afzonderlijk hoorbaar zijn.

Met het O-kommando (pas op: de letter O en niet het cijfer Ø!) kunnen we het oktaaf sturen. Een oktaaf loopt altijd van C tot B. Indien geen oktaaf wordt gespecificeerd, wordt het vierde oktaaf aangenomen; het oktaaf waarin de centrale C zich als basis bevindt. Het volgende voorbeeld geeft de grote terts toonladder van C, klimmend over drie oktaven:

```
PLAY "O3CDEFGABO4CDEFGABO5CDEFGAB"
Ok
```

Achter het O-kommando dient dus een cijfer (1...8) te worden opgenomen dat aangeeft in welk oktaaf de betreffende noot dient te worden gespeeld. Het volgende voorbeeld laat de laagst en hoogst mogelijke toon binnen deze notatie horen:

```
PLAY "O1CO8B"
Ok
```

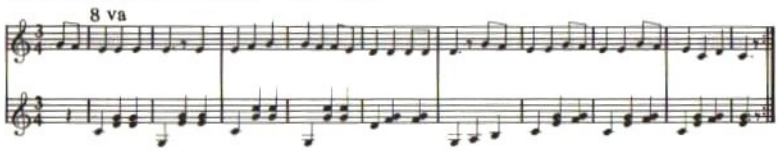
Indien achter de letter O geen cijfer wordt opgenomen, wordt een O4 uitgevoerd.
Indien achter een noot een punt wordt geplaatst, wordt deze analoog aan de normale muzieknotatie met de halve lengte verlengd. Een twee-

de punt resulteert op dezelfde wijze in een tweede verlenging die de helft in lengte duurt van de eerste verlenging, enzovoorts.

Voordat we op de wat meer technische kommando's van MML overgaan, volgt eerst een combinatievoorbeeld in de vorm van een eenvoudig driestemmig kinderliedje:

GOEDENAVOND SPEELMAN

MSX-ARRANGEMENT ♩=120




```
NEW
Ok
10 PLAY "V15T120","S1M5000T120","S1T120"
20 PLAY "L8O5GFL4EEEE.R8E","O4L4RCEEO3GO4EE","O4L4RRGGRGG"
30 PLAY "EFGGFL8FEL4DDL8DD","CGGO3GO4GGDFF","RO5CCRCCO4RGG"
40 PLAY "L4D.R8L8GFL4EEL8GFL4EEL8GF","O3GABO4CEFCEF","RRRRO4
GGRGG"
50 PLAY "L4ECDC.R8","CEFE.R8","RGGG.R8"
60 GOTO 20
RUN
```

Op de funktie van regel 10 gaan we later pas verder op in.

Met het V-kommando kan per stem het volume worden bepaald. Achter de letter V dient dan een integere konstante te worden opgenomen, niet kleiner dan nul en niet groter dan 15. 15 geeft het hoogste volume, 0 geeft geen volume. Indien een V-kommando zonder achtervoegsel wordt gebruikt, wordt een V10 uitgevoerd. In het voorgaande voorbeeld zien we in regel 10 ondermeer de volume-instelling van de eerste stem. Omdat deze het beste gehoord dient te worden, wordt deze op het hardst (V15) gezet. Probeer het voorbeeld ook eens met V0...V14. Wanneer niet eerder een volume werd gespecificeerd, staat het volume standaard op V10.

Met het T-kommando kan het tempo van de muziek per stem worden bepaald. Achter de letter T dient dan een integere konstante, niet kleiner dan 32 en niet groter dan 255 te worden opgenomen. Dit getal stelt het aantal kwartnoten voor dat per minuut dient te worden ge-

speeld. Standaard staat het tempo ingesteld op 120 kwartnoten per minuut (T120). Het laatste voorbeeld kan door verandering van de T120 in regel 10 naar T255 (drie maal, één maal voor elke stem) tot twee maal zo snel worden gespeeld door de computer. Indien alleen de letter T wordt gebruikt, dan wordt een T120 uitgevoerd.

Met het N-kommando kunnen toonhoogten op een andere wijze worden bepaald. Het MSX-basic staat de aansturing van 96 verschillende tonen toe, genummerd van 1 tot en met 96. Elke volgende toon ligt weer een halve toon hoger . De 36e toon komt overeen met de centrale C. Met het N-kommando kan een toon worden gespeeld niet door door de muzikale notatie te gebruiken maar door de juiste toon uit de rij van 96 te kiezen. Het nummer van deze toon dient dan achter de letter N te worden geplaatst.
Het volgende voorbeeld toont twee kommando's die precies dezelfde uitwerking hebben:

```
PLAY "O4L4CDEFGFEDC"
Ok
PLAY "L4N36N38N40N41N43N41N40N38N36"
Ok
```

Een nul achter de letter N resulteert in een rustpauze.

Met de letter X, gevolgd door een alfanumerieke variabele en een puntkomma, kan een alfanumerieke variabele worden ingelast.
Deze inlasting heeft echter beperkingen. Per PLAY-bevel per stem dient er liefst maar één alfanumerieke variabele op deze wijze te zijn opgenomen. Indien meerdere alfanumerieke variabele binnen één bevel en één stem worden opgenomen, of wanneer één variabele meerdere keren wordt opgenomen binnen één bevel en één stem, dan worden stuk voor stuk deze variabelen in de stem ingevoegd totdat zij gezamelijk de lengte van 23 posities hebben overschreden. De rest van de stem wordt in het geheel niet meer uitgevoerd terwijl de laatste variabele nog wel wordt 'uitgespeeld'. Een voorbeeld:

```
NEW
Ok
10 LET A$="CDEFGAB"
20 PLAY "O1XA$;O2XA$;O3XA$;O4XA$;O5XA$;O6XA$;O7XA$;O8XA$;"
RUN
Ok
```

Eigenlijk zou de grote test toonladder over alle oktaven moeten worden gespeeld. Echter, alleen de eerste vier oktaven worden uitgevoerd door

de bovengenoemde bepaling. Hoogstwaarschijnlijk betreft het hier een fout, die zowel in MSX-1 als MSX-2 voorkomt.

Met het gelijkteken (=), gevolgd door een numerieke variabele en een puntkomma, kan de waarde van een numerieke variabele worden tussengevoegd. De variabele wordt pas tussengevoegd nadat een eventuele decimale fractie is verwaarloosd. Indien de waarde niet ligt binnen de toegestane waarden, volgt een foutmelding.
Het voorbeeld speelt alle mogelijke toonhoogten:

```
NEW
Ok
10 FOR I=0 TO 96
20 PLAY "N=I;"
30 NEXT I
RUN
Ok
```

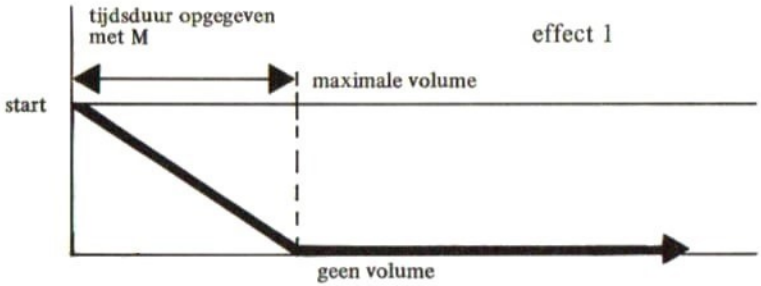
Merk op dat PLAY "N=96", de hoogste met het N-kommando aan te sturen toon een halve toon hoger is dan PLAY "08B", de hoogste op andere wijze aan te sturen toon.

Met de letter S kan men één van de acht mogelijke volume-effecten selekteren. Het gebruik van de letter V naast de letter S heeft geen zin; een volume-effect heeft altijd een vast verloop.

Indien gebruik wordt gemaakt van een volume-effect, dan is het M-kommando daarbij onmisbaar. Met het M-kommando kan men de tijdseenheid specificeren waarbinnen een geluidseffect dient te zijn afgewikkeld. Hiertoe dient achter de letter M een gehele konstante te worden opgenomen, niet kleiner dan –65535 en niet groter dan 65535. Indien de tijdseenheid niet via een konstante maar via een numerieke variabele wordt gespecificeerd met gebruikmaking van het gelijkteken en de puntkomma, dan mag de waarde van de gebruikte variabele niet kleiner zijn dan –32768 en niet groter dan 32767.
Bij een negatieve waarde wordt voor uitvoering eerst de waarde 65536 opgeteld. De kenner herkent hierin de tweecomplementmethode.

De duur in seconden van deze tijdseenheid rekent men uit door de achter het kommando M gespecificeerde waarde te delen door de waarde 6965.

Een voorbeeld: volume-effect nummer 1 heeft tot gevolg dat het volume op de navolgende wijze verloopt:



Een toon begint dus op vol volume en sterft vervolgens weg. De tijd die verstrijkt totdat de toon volledig is weggestorven, wordt via het M-kommando bepaald. We laten nu een toon in één seconde wegsterven waardoor we een soort nagalm-effekt krijgen:

```
PLAY "L2M6965S1T32A"
Ok
```

Vervolgens laten we de toon in een ..nde seconde wegsterven zodat een tokkel-effect ontstaat:

```
PLAY "L2M697S1A"
Ok
```

Volume-effect 8 is een continue herhaling van effect 1. Steeds wanneer de toon is uitgestorven, wordt deze weer op vol volume gebracht. We laten een toon tien maal per seconde uitsterven en weer op volume komen:

```
PLAY "L2M697S8A"
Ok
```

Een snel tokkeleffect ontstaat. We kunnen ook 6965 maal per seconde de toon laten uitsterven en weer op volume komen:

```
PLAY "L2M1S8A"
Ok
```

Door deze zeer snelle afwisseling ontstaat een schel geluidseffect.

Door veel met het M- en het S-kommando te combineren, kunnen allerlei geluidseffecten worden verkregen.

In de tabel op de volgende bladzijde staan alle geluidseffecten die in MSX-basic mogelijk zijn, bij elkaar genoemd.

Indien geen tijdsduur door middel van het M-kommando werd gespecificeerd, dan wordt een tijdsduur M255 aangenomen.
Indien een M-kommando zonder achtervoegsel wordt gebruikt, dan wordt de standaard waarde (255) aangenomen.
Indien een S-kommando zonder achtervoegsel wordt gebruikt, dan wordt een S0 uitgevoerd.
Een S-kommando kan worden opgeheven door gebruik van het V-kommando. Voor alle drie de stemmen kan maar één effect (S) en één tijdsduur (M) worden gespecificeerd. Steeds worden de laatst gespecificeerde waarden als geldend aangenomen voor alle stemmen waarvoor een effect werd aangezet.

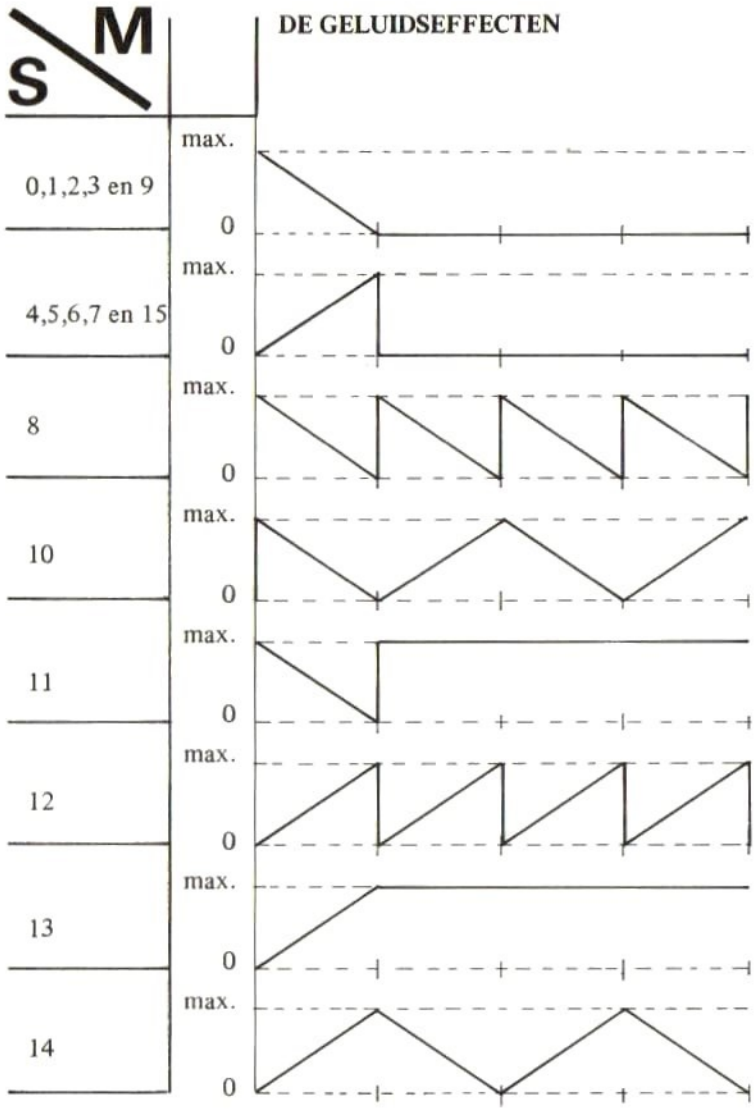
Enkele opmerkingen in verband met het PLAY-kommando:

- Door gebruik van het BEEP-kommando worden alle waarden weer op de standaardwaarden teruggebracht.

- Het maximale uitvoeringstempo is tien tonen per seconde. Indien een snellere passage wordt geprogrammeerd, blijft de uitvoering achter. Indien twee- of driestemmig wordt uitgevoerd, kan hierdoor een ongelijke uitvoering van de drie stemmen (een desynchronisatie) onstaat.

- Doordat besturingskommando's (zoals O, L, T en V) een kleine uitvoeringstijds inbeslag nemen, kan een desynchronisatie tussen de geprogrammeerde stemmen ontstaan bij meerstemmige programmering. Deze desynchronisatie kan worden opgelost door tussenvoeging van R64-kommando's (vierenzestigste rusten) op gehoor in de 'te snelle' stem. Ook verdient het aanbeveling om de stemmen per PLAY-kommando niet te lang te maken. Per PLAY-kommando vindt namelijk een algehele synchronisatie plaats; de stemmen worden bij elk PLAY-kommando altijd gelijk gestart.

- Indien geen effecten worden gebruikt, is de toonscheiding tussen twee gelijke tonen niet hoorbaar. Dit kan worden opgelost door


S
M
DE GELUIDSEFFECTEN
0,1,2,3 en 9
4,5,6,7 en 15
8
10
11
12
13
14
max.
0

een R64 tussen de twee noten in te lassen. Bijvoorbeeld:

```
PLAY "V10L4T120"
Ok
PLAY "AA"                          (een toon is hoorbaar)
Ok
PLAY "AR64A"                       (twee tonen zijn hoorbaar)
Ok
```

Pas met deze methode echter wel op voor desynchronisatie-effecten.

Nu volgt een kort overzicht van de kommando's van MML. Het is raadzaam om dit overzicht over te schrijven of te kopiëren zodat het als hulpmiddel kan fungeren wanneer men veel met de geluidsgenerator wil werken.

CDEF GAB – toonhoogten. Achter de naam van de noot kan de tijdsduur (1...64) worden opgenomen. Indien geen tijdsduur wordt opgenomen, wordt de standaard tijdsduur genomen.

+of # – kruis; halve toon verhoging

– – mol; halve toon verlaging

L.. – veranderen standaard tijdsduur voor een noot. L4 is standaard. De tijdsduur dient achter de letter L te worden opnomen (1...64). Alleen de letter L is gelijk aan een L4

R.. – rust. Achter de letter R dient de duur van de rust (1...64) te worden opgenomen. Alleen de letter R is gelijk aan kwart rust (R4)

O.. – oktaaf. Een oktaafnummer (1...8) kan worden uitgekozen. Een oktaaf loopt altijd van C naar B. O4 is het standaard oktaaf en heeft als eerste noot de centrale C. Indien achter de O geen cijfer wordt opgenomen, wordt een O4 uitgevoerd.

=...; – tussenvoeging van de waarde van een numerieke variabele

X...; – tussenvoeging van de waarde van een alfanumerieke variabele

S.. – selectie van een volume-effect. Achter S dient het effectnummer (0...15) te worden opgegeven. Indien geen effectnummer wordt opgegeven wordt effectnummer 0 aangenomen

M..... – tijdsduur van de effect-ontwikkeling. Vermenigvuldig de gewenste ontwikkelingstijd in seconden met 6965 en rond de verkregen waarde af. Deze waarde dient achter de M te worden opgenomen (0...65535). Bij M=...; dient de genoemde variabele een integere waarde te bezitten (-32768 ... 32767). Het M-kommando telt bij uitvoering bij negatieve waarde dan 65536 op. Indien alleen een M wordt opgenomen, wordt een M255 uitgevoerd

N.. – de toonhoogte dient na de letter N (0...96) te worden opgenomen. N0 is gelijk aan een rust en N36 is gelijk aan de centrale C. Elke volgende toon is weer een halve toon hoger

T... – tempo. Achter de letter T dient het aantal kwartnoten per minuut te worden opgenomen. Indien achter de letter T geen waarde wordt opgenomen, wordt een T120 uitgevoerd

V... – volume. Achter de letter V dient het volume te worden opgenomen. 0=geen volume, 15=hoogste volume. Alleen de letter V is gelijk aan een V10.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **POINT is punt**

schrijfwijze

```
POINT<LOCATIE>

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

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

betekenis

Voor goed begrip van deze funktie is het noodzakelijk dat eerste de PSET-behandeling volledig is doorgenomen.

Deze funktie geeft als resultaat de kode van de kleur die het gelocaliseerde puntje heeft. Indien het punt zich buiten het beeld bevindt, geeft deze funktie de waarde −1.

Bijvoorbeeld:

```
NEW
Ok
10 COLOR 15,0,0:SCREEN 3:CLS
20 FOR I=10 TO 100 STEP 8
30 FOR J=10 TO 100 STEP 8
40 PSET (I,J),-15*(POINT(I,J)=0)
50 NEXT J,I
60 GOTO 20
RUN
```

Op beeldscherm verschijnt een raster van punten die steeds weer verschijnen en verdwijnen. In regel 40 wordt de kleur van het betreffende puntje op 15 gesteld indien deze eerder 0 was en op 0 gesteld indien deze eerder 15 was.

De POINT-funktie geeft geen zinvolle resultaten wanneer een alfanumeriek beeldscherm is geactiveerd.

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het computergeheugen is vereist**
soort . . . . . **KOMMANDO**
afkomst . . . . . **POKE is wegstoppen**

schrijfwijze

```
POKE<GEHEUGENADRES>,<GEHEUGENWAARDE>

<GEHEUGENADRES>::=<N>

<GEHEUGENWAARDE>::=<N>

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

betekenis

Met het POKE-kommando kunnen geheugenadressen van andere waarden worden voorzien. Hiertoe dient een geheugenadres te worden opgegeven, gevolgd door een komma en de waarde die het byte op de betreffende geheugenplaats dient aan te nemen.
Het geheugenadres mag niet kleiner zijn dan –32768 en niet groter dan 65535. Een eventuele decimale fractie wordt verwaarloosd. Indien het geheugenadres groter is dan 32767, wordt een eventuele decimale fractie naar boven afgerond. Indien een geheugenadres negatief is, wordt voordat het geheugen wordt aangesproken eerst de waarde 65536 bij dit adres opgeteld.
De geheugenwaarde dient in één byte te passen en mag derhalve niet kleiner zijn dan 0 en niet groter zijn dan 255 terwijl een eventuele decimale fractie wordt verwaarloosd.

Het met POKE toegankelijke geheugen wordt gevormd door:

–	0...32767	32 kilobyte basis ROM
–	32786...65535	32 kilobyte RAM, het voor MSX-basic maximaal beschikbare geheugen.

Het veranderen van geheugenwaarden in de eerste 32 kilobyte geheugen heeft geen zin; dit gedeelte van het geheugen is ROM (Read Only Memory = alleen leesbaar geheugen) en heeft een vaste, onveranderde waarde. Bijvoorbeeld:

```
NEW
Ok
10 PRINT PEEK(32)
20 POKE 32,0
30 PRINT PEEK(32)
RUN
 195
 195
Ok
```

Op regel 20 werd een vergeefse poging gedaan om een stukje ROM te wijzigen.

In het tweede stuk geheugen van 32 kilobyte kunnen wel wijzigingen worden aangebracht; dit gedeelte van het geheugen bestaat uit RAM (Random Access Memory = vrij toegankelijk geheugen). Wijzigingen in dit geheugen dienen met grote terughoudendheid en met goede kennis van zaken te worden aangebracht; onoordeelkundig gebruik van POKE leidt bijna altijd tot het 'ophangen' van het computersysteem. Uit- en weer inschakelen van de computer is dan de enige remedie.

Bijvoorbeeld:

```
NEW
Ok
10 FOR I=32768 TO 65535
20 POKE I,0
30 NEXT I
RUN
Ok
LIST
Ok
```

Dit programma wist zichzelf uit; na uitvoering zijn de programmaregels verdwenen.

```
NEW
Ok
10 FOR I=65535 TO 32768 STEP -1
20 POKE I,0
30 NEXT I
RUN
```

Na korte tijd start de computer opnieuw op of staat hij vast (MSX-2); de geheugenwijziging had blijkbaar nogal destructieve gevolgen.

Voor een goed gebruik van POKE is gedetailleerde kennis van de opbouw van het geheugen van een MSX-computer noodzakelijk. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop verder niet ingegaan.

Nog twee handige POKE-voorbeelden:

```
POKE 64683,0
```

Zet uw computer op kleine letters alsof de CAPS LOCK toets werd uitgezet.

```
POKE 64683,255
```

Zet uw computer op hoofdletters alsof de CAPS LOCK toets werd aangezet.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **POS is afkorting van position –positie**

schrijfwijze

```
POS(<U>)

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

betekenis

Deze funktie geeft als resultaat de positie waarop de cursor zich op het moment in een regel bevindt. De uitdrukking tussen haakjes is een dummy-uitdrukking; in de praktijk kan het beste gewoon een nul worden gebruikt. Voorbeeld:

```
NEW
Ok
10 FOR I=0 TO 10 STEP 2
20 LOCATE I:PRINT POS(0)
30 NEXT I
RUN
 0
   2
     4
       6
         8
           10
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PRESET is afkorting van point resetting–punt terugzetten (verwijderen)**

schrijfwijze

```
MSX-1: PRESET<LOCATIE>[,<KLEUR>]

MSX-2: PRESET<LOCATIE>[,[<KLEUR>][,<LOGISCHE OPERATOR>]]\<...>,

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<KLEUR>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

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

betekenis

PRESET werkt bijna volledig identiek aan PSET. Het enige verschil is, dat bij het weglaten van de kleurkode niet de voorgrondkleur maar de achtergrondkleur wordt gekozen als kleur waarin het puntje wordt uitgevoerd. In de meeste gevallen wordt het puntje dan dus verwijderd (gereset).

moeilijkheidsgraad . . . **eenvoudig maar in complexe samenstellingen tot vrij moeilijk**

soort . . . . . **KOMMANDO**

afkomst . . . . . **PRINT is afdrukken**

schrijfwijze

```
PRINT
----- [#<KANAAL>,][{<S>}<P>{<S>[<P>]}][USING<USING-STRING> ...
  ?                                    ...;<U>{<S> <U>}[<S>]]

<S>::=;|,

<P>::=<U>|TAB(POSITIE)|SPC(<AANTAL SPATIES>)

<POSITIE>::=<N>

<AANTAL SPATIES>::=<N>

<KANAAL>::=<N>

<USING STRING>::=<A>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Het PRINT-kommando is één van de meest belangrijke maar ook ingewikkelde kommando's. Het kommando is op veel zeer verschillende manieren te gebruiken.

eenvoudig gebruik

Met het PRINT-kommando kan op beeldscherm een regel worden opgeschoven. Voorbeeld:

```
PRINT:PRINT:PRINT:PRINT
```

(4 regels worden overgeslagen)

```
Ok
```

Met het PRINT-kommando kunnen konstanten of de inhoud van variabelen worden afgedrukt op het beeldscherm. Voorbeeld:

```
LET A=2.3:LET A$="TEST"
Ok
PRINT "DIT IS EEN ";A$;A
DIT IS EEN TEST 2.3
Ok
```

Indien de verschillende af te drukken uitdrukkingen van elkaar gescheiden zijn met een punt-komma, worden deze uitdrukkingen tegen elkaar aan afgedrukt. Merk op dat voor een positieve waarde altijd een spatie wordt afgedrukt. Deze spatie kan worden beschouwd als het teken (plus-teken) van deze waarde. Ná een numerieke waarde wordt ook altijd een extra spatie afgedrukt.
Indien de verschillende uitdrukkingen van elkaar gescheiden worden door een komma, wordt elke uitdrukking een kolom verder afgedrukt. Het beeldscherm is hiertoe ingedeeld in kolommen van veertien tekens. Bijvoorbeeld:

```
PRINT "EEN","TEST"
EEN            TEST
Ok
```

Een TAB-funktie biedt de mogelijkheid om zelf tabulaties (kolomindelingen) door te voeren. Tussen de haakjes van de TAB-funktie dient de positie te worden vermeld waarop het afdrukken op het beeldscherm dient te beginnen. Bijvoorbeeld:

```
NEW
Ok
10 PRINT "EEN";TAB(20);"TWEE"
20 PRINT "APPELS";TAB(20);"PEREN"
RUN
EEN                  TWEE
APPELS               PEREN
Ok
```

Een SPC-funktie biedt de mogelijkheid om tussen twee af te drukken uitdrukkingen een **vast** aantal spaties te plaatsen. Bijvoorbeeld:

```
NEW
Ok
10 PRINT "EEN";SPC(20);"TWEE"
20 PRINT "APPELS";SPC(20);"PEREN"
```

```
RUN
EEN                         TWEE
APPELS                        PEREN
Ok
```

Indien de tussen de haakjes van TAB of SPC vermelde waarde gebroken is, zal de hoogste gehele waarde kleiner dan deze gebroken waarde geldig zijn.

Wanneer een PRINT-kommando wordt afgesloten met een puntkomma, dan zal een volgende printopdracht achter het laatst afgedrukte gedeelte doorgaan en dus niet op een volgende regel beginnen. Bijvoorbeeld:

```
NEW
Ok
10 PRINT "HALLO ";
20 PRINT "ALLEMAAL"
RUN
HALLO ALLEMAAL
Ok
```

Wanneer een PRINT-kommando wordt afgesloten met een komma, dan zal een volgende printopdracht niet op de volgende regel maar in een volgende kolom doorgaan. Bijvoorbeeld:

```
NEW
Ok
10 PRINT "EEN",
20 PRINT "TWEE"
RUN
EEN             TWEE
Ok
```

N.B.: Wanneer MSX-BASIC bij het afdrukken van met komma's gescheiden uitdrukkingen geen volledige kolom van 14 posities meer beschikbaar heeft, wordt het af te drukken gegeven in zijn geheel op de volgende regel afgedrukt.

gevorderd gebruik: USING

Indien een PRINT-kommando wordt geconstrueerd met daarin het USING-kodewoord, dan kan direkt na dat USING-kodewoord een formaat (of print-masker) worden gespecificeerd. De af te drukken gegevens dienen dan naar dit formaat te worden gevormd.

De diverse karakters die in een formaatspecificatie (using-string) mogen worden gebruikt, worden hieronder per stuk in een voorbeeld behandeld.

Het "!"-teken

Met het "!"-teken kan worden aangegeven dat slechts één karakter dient te worden afgedrukt. Voorbeeld:

```
PRINT USING "!";"MSX"
M
Ok
```

Het "\"-teken

Met het "\"-teken kan worden aangegeven dat slechts een bepaald aantal karakters dient te worden afgedrukt. Hiertoe dienen in de using-string twee "\"-tekens te worden opgenomen met daartussenin het aantal af te drukken tekens minus 2 **aan spaties**. Voorbeeld:

```
NEW
Ok
10 LET A$="ABCDEFGHIJKLMNOP"
20 PRINT USING "\   \";A$
RUN
ABCDE
Ok
```

Zoals ook voor de nog te behandelen tekens geldt, mogen de diverse speciale tekens ook bij elkaar in één using-string worden gebruikt. Daarbij mogen ook niet-speciale karakters in een using-string worden gebruikt; deze worden dan gewoon afgedrukt. Bijvoorbeeld:

```
NEW
Ok
10 LET U$="\   \/!"
20 PRINT USING U$;"ABCDEFGH";"919191"
RUN
ABCDE/9
Ok
```

Het "&"-teken

Met het "&"-teken kan worden aangegeven dat een alfanumerieke uitdrukking onverkort dient te worden afgedrukt. Voorbeeld:

```
NEW
Ok
10 INPUT "WOORD ";A$
20 PRINT USING "DE EERSTE LETTER VAN & IS !";A$;A$
RUN
WOORD ? KRAAI                                  (dit wordt ingegeven)
DE EERSTE LETTER VAN KRAAI IS K
Ok
```

Het "#"-teken

Met het "#"-teken geeft men aan dat op die plaats een cijfer van de uitkomst van een numerieke uitdrukking mag worden geplaatst. Met een veelheid van "#"-tekens kan men precies bepalen waar de uitkomst van een numerieke uitdrukking moet komen te staan en hoeveel posities deze mag hebben. Bijvoorbeeld:

```
NEW
Ok
10 FOR I=7 TO 12:PRINT USING "##";I
20 NEXT I
RUN
 7
 8
 9
10
11
12
Ok
```

Merk op dat in bovenstaand voorbeeld de getallen netjes **rechts** gericht onder elkaar staan. Zonder het gebruik van de USING-mogelijkheid zouden ze links gericht onder elkaar staan.
Indien de using-string te klein is, wordt het betreffende numerieke antwoord normaal afgedrukt maar met een voorlopend "%"-teken als waarschuwing. Bijvoorbeeld:

```
PRINT USING "### ##";100;100
100 %100                                   (de tweede "using" is te klein)
Ok
```

Indien nodig, wordt het af te drukken getal door dit USING-gebruik afgerond. Bijvoorbeeld:

```
NEW
Ok
10 PRINT USING "##";1.4;1.5
RUN
 1 2
Ok
```

Uit het voorgaande voorbeeld blijkt ook dat wanneer de using-string is "opgebruikt", hij weer van voor af aan opnieuw wordt gebruikt. In dit geval wordt hierdoor twee maal hetzelfde patroon gebruikt voor het afdrukken van de getallen 1.4 en 1.5. Voor een negatieve waarde dient een extra positie te worden gecreëerd in verband met het te plaatsen "–"-teken. Bijvoorbeeld:

```
PRINT USING "##";10;-10
10%-10
Ok
```

(voor –10 is de using-string te klein).

De punt

Met de decimale punt kan het numerieke print-masker met decimale posities worden gebruikt. Ook bij dit gebruik wordt eventueel weer afgerond. Voorbeeld:

```
PRINT USING "###.##";11.2;12.125
 11.20 12.13
Ok
```

Merk op dat posities na de decimale punt eventueel netjes met nullen wordt aangevuld.

Het "+"-teken

Het "+-teken mag vóór of na een numeriek print-masker worden geplaatst. Dit heeft tot gevolg dat het teken van het af te drukken getal voor of na dit getal wordt vermeld. Bijvoorbeeld:

```
NEW
Ok
10 LET A=22.3:LET B=-111
20 PRINT USING "+#####.##";A;B
30 PRINT USING "####.##+";A;B
RUN
  +22.30 -111.00
  22.30+ 111.00-
Ok
```

Het "–"-teken

Het "–"-teken heeft bijna dezelfde uitwerking als het "+"-teken. Het verschil is dat in plaats van een "+"-teken een spatie word afgedrukt in het uiteindelijke resultaat en dat het "–"-teken alleen rechts van een print-masker het gewenste effect heeft. Bijvoorbeeld:

```
PRINT USING "###.##-";1;-1
  1.00   1.00-
Ok
```

Het "**"-teken

Een dubbele ster, geplaatst voor een numeriek print-masker, heeft tot gevolg dat het print-masker met twee numerieke plaatsingsmogelijkheden voor cijfers wordt uitgebreid en dat daarbij alle spaties die door gebruik van dit print-masker voor het af te drukken getal zouden ontstaan, worden vervangen door sterren. Bijvoorbeeld:

```
PRINT USING "****.##";2
***2.00
Ok
PRINT USING "** ";2;22
*2 22
Ok
```

Uit het laatste voorbeeld blijkt dat de dubbele ster ook alleen mag worden gebruikt.

Het "$$"-teken

Een dubbel dollarteken, geplaatst voor een numeriek print-masker, heeft tot gevolg dat er één numerieke plaatsmogelijkheid extra wordt gecreëerd voor een cijfer en dat vóór het af te drukken getal een dollarteken wordt geplaatst. Bijvoorbeeld:

```
PRINT USING "$$#.##";2.5
 $2.50
Ok
```

Ook het "$$"-teken mag eventueel alleenstaand worden gebruikt.

Het "**$"-teken

Het gebruik van twee sterren en één dollarteken heeft tot gevolg dat de "$$" en de "**" worden gecombineerd. Het resultaat is twee extra plaatsingsmogelijkheden voor cijfers, een dollarteken voor het af te drukken getal en een opvulling van de linker spaties met sterren. Bijvoorbeeld:

```
PRINT USING "**$#.##";2.5
**$2.50
Ok
```

Ook het "**$"-teken mag alleenstaand worden gebruikt.

N.B.: Sommige computermodellen werken niet met het dollar-teken ($) maar met het Yen-teken (¥) of het pond-teken (£). Het één en ander hangt af van het land van herkomst van de machine en het land waarvoor de machine eigenlijk werd gefabriceerd. In Nederland worden normaal machines verkocht met Amerikaanse instelling. Deze werken binnen de using-string met het dollar-teken ($).

De komma

Een komma, opgenomen in een numeriek print-masker, niet als eerste teken en links van de eventueel aanwezige decimale punt heeft tot gevolg dat de duizendtallen met een komma worden afgescheiden in het af te drukken getal. Eventueel mogen meerdere komma's op de beschreven wijze worden opgenomen; zij resulteren alle in extra plaatsingsmogelijkheden voor cijfers. Voorbeeld:

```
NEW
Ok
10 PRINT USING "####,.##";1000
20 PRINT USING "#####.##,";1000
RUN
1,000.00
1000.00,
Ok
```

Merk op dat op programmaregel 20 de komma niet het beschreven effect heeft; de komma staat **na** de decimale punt.

Het " ^^^^" teken

Met het teken " ^^^^ " geeft men aan, dat een exponentiële vorm dient te worden gehanteerd bij het afdrukken. De "**", de "$$", de "**$", en het gebruik van de komma zijn in combinatie met " ^^^^" wel toegestaan maar geven zinloze resultaten. Het gebruik van een"+" of een "–" **ACHTER** het numerieke print-masker is ook toegestaan maar geeft eveneens een zinloos resultaat.
" ^ ^ ^ ^" dient direkt na het numerieke print-masker te worden opgenomen. Bijvoorbeeld:

```
PRINT USING "#.##^^^^";20000
0.20E+05
Ok
```

Merk op bij de exponentiele using-string het print-masker bepaald is voor de grootte-orde van de mantisse. De exponent wordt niet aangepast. voorbeeld:

```
PRINT USING ".#####^^^^";SQR(3)
.01732E+02
Ok
PRINT USING "#.######^^^^";SQR(3)
0.173205E+01
Ok
PRINT USING "#####.###^^^^";SQR(3)
 1732.051E-03
Ok
```

Steeds krijgt de mantisse een aantal voorkommaposities, gelijk aan het aantal gespecificeerde voorkommaposities minus 1.
Merk op dat het hierdoor

```
PRINT USING ".####^^^^";-1
%-.0100E+02
Ok
```

ondermeer fout gaat.

Gevorderd gebruik: schrijven naar randapparatuur.

Zie hiervoor ook de beschrijving van het **OPEN** en **CLOSE**-kommando.

Indien de kanaal-optie van het PRINT-kommando wordt gebruikt, kan men de gegevens in plaats van naar het beeldscherm ook naar een an-

dere plaats zenden. Dit kan bijvoorbeeld een cassetteband of het grafische scherm zijn. Ook kunnen de gegevens op deze wijze naar een printer (afdrukeenheid) worden verzonden alhoewel het LPRINT-kommando daar veel gemakkelijker voor is.

Indien de gegevens later met een INPUT weer dienen te worden gelezen (geldt natuurlijk niet voor de printer en het (grafische) beeldscherm), dan dient men er voor te zorgen dat de diverse gegevens die op één printregel staan door een komma van elkaar zijn gescheiden. Zie verder hiervoor het INPUT-kommando. Voorbeeld: gegevens schrijven naar cassetteband.

```
NEW
Ok
10 OPEN "CAS:TEST" FOR OUTPUT AS 1
20 PRINT#1,"DEZE GEGEVENS WORDEN"
30 PRINT#1,"STRAKS WEER OPGEHAALD"
40 PRINT #1,123;",";-12.33
50 CLOSE
```

(nu eerst de cassetterecorder op opnemen zetten)

```
RUN                                    (de cassetterecorder begint te lopen)
Ok          (de cassetterecorder stopt weer; de gegevens zijn op cassette-
                                                              band geschreven)
```

(N.B. : op dit voorbeeld komen we bij de behandeling van het INPUT-kommando nader terug)

Een tweede voorbeeld: schrijven naar het grafische scherm.

```
NEW
Ok
10 SCREEN 3
20 OPEN "GRP:" AS 1
30 PSET(0,0)
40 PRINT #1,"MSX";
50 GOTO 40
RUN
```

(het beeldscherm wordt met grote letters MSX volgezet)

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PSET is point setting – punt zetten**

schrijfwijze

```
MSX-1: PSET<LOCATIE>[,<KLEUR>]

MSX-2: PSET<LOCATIE>[,[<KLEUR>][,<LOGISCHE OPERATOR>]]\<...>,

<LOCATIE>:=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>:=<N>

<VERTIKAAL>:=<N>

<KLEUR>:=<N>

<N>:=<ZIE ALGEMENE SPECIFICATIES>

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

<LOGISCHE OPERATOR>:=XOR|OR|AND|PSET|PRESET|TXOR|TOR

                     ... |TAND|TPSET|TPRESET
```

betekenis

Voor een goed begrip van dit sleutelwoord is het raadzaam om eerst de betekenis van SCREEN en COLOR door te nemen.

Met het PSET-kommando kan een puntje op een grafisch beeldscherm worden geplaatst. Dit puntje kan in een bepaalde kleur worden uitgevoerd. Wanneer geen kleur wordt gespecificeerd, wordt de ingestelde voorgrondkleur als geldende kleur genimen.

De plaatsbepaling op het beeldscherm kan men zich als volgt voorstellen:

De linker bovenhoek van het grafische scherm krijgt altijd de plaatscoördinaten (0,0). De punten, rechts van dit linkerbovenpunt krijgen een opklimmende eerste coördinaat. Zo heeft het rechterbovenpunt onder de SCREEN 2-instelling de coördinaten (255,0).

De punten, onder het linkerbovenpunt krijgen een opklimmende twee-

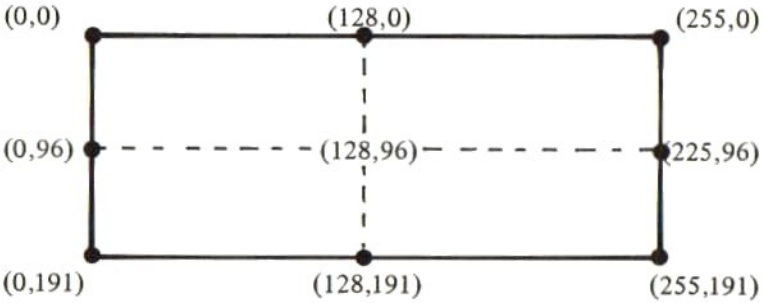
de coördinaat. Onder de SCREEN 2-instelling heeft het linkeronderpunt dan ook de plaatscoördinaten (0,191).

Het beeldscherm is als zodanig te vergelijken met een wiskundig coördinatenstelsel met dit verschil dat de Y-as om de oorsprong is gespiegeld.

Het rechteronderpunt van het beeldscherm heeft onder de SCREEN 2-instelling de coördinaten (255,191). Met twee coördinaatgetallen is elk punt op het grafische scherm aan te duiden.

De eerste coördinaat noemt men meestal de X-coördinaat en de tweede coördinaat noemt men dan de Y-coördinaat. Een punt is bepaald door een X-coördinaat en een Y-coördinaat.

Onder SCREEN 2 kan men zich het grafishe beeldscherm als volgt voorstellen:



Onder MSX-2 kent men een zevental grafische beeldscherm-instellingen met verschillende oplossende vermogens. Onder SCREEN 5 en SCREEN 8 wordt het rechteronderpunt va het beeldscherm gevormd door het punt (255,211). Onder SCREEN 6 en SCREEN 7 wordt dit punt gevormd door het punt (511,211). Onder deze scherminstellingen staan de puntjes horizontaal gezien dus tweemaal zo dicht op elkaar. Als zodanig kunnen onder SCREEN 6 en SCREEN 7 hele fijngetekende plaatjes op scherm worden ontworpen.

Hoe dichter de puntjes op elkaar staan, hoe fijner getekende plaatjes er op beeld kunnen worden gemaakt. Men zegt dan dat het grafische scherm een hoog oplossend vermogen of een hoge resolutie heeft.

SCREEN 3:

Een aparte beeldscherm-instelling wordt gevormd door SCREEN 3. Dit grafische beeldscherm heeft slechts een oplossend vermogen van 64 bij 48 puntjes. Men zou verwachten dat de rechteronderhoek van het grafische beeldscherm onder SCREEN 3 dan ook de coördinaten (63,47 zou hebben. Niets is echter minder waar! Ook onder SCREEN 3 heeft het rechteronderpunt de coördinaten (255,191). net zoals onder de SCREEN 2-instelling dus. Het verschil zit er hierin, dat één beeldschermpunt in feite overeenkomt met zestien beeldschermpunten onder de SCREEN 2-instelling. Door één van deze punten aan te zetten, zet men in feite alle zestien de punten aan die onderdeel van het ene grote beeldschermpunt uitmaken.

Onder SCREEN 3 staan de coördinaten (0,0), (0,1), (0,2), (0,3), (1,0) ...(3,3) allemaal voor het rechterbovenpunt.

Consequentie van dit gegeven is, dat grafische programmatuur die onder een SCREEN 2-instelling werkt, ook onder een SCREEN 3-instelling kan werken met dit verschil dat het grafische resultaat veel grover van vorm is.

Voorbeelden: (allemaal te onderbreken met CONTROL-STOP)

```
NEW
Ok
10 SCREEN 2
20 PSET (10,10)
30 PSET (30,30)
40 PSET (10,30)
50 PSET (30,10)
60 GOTO 60
RUN
```

Op het beeldscherm verschijnen vier puntjes als hoekpunten van een vierkant. Regel 60 voorkomt dat ongewenst weer een alfanumeriek scherm wordt geactiveerd en de tekening verdwijnt.

```
10 SCREEN 3
RUN
```

Alleen programmaregel 10 werd veranderd. Dezelfde punten worden afgebeeld; de punten zijn echter in beide richtingen vier maal zo groot.

```
NEW
Ok
10 SCREEN 2
20 FOR I=10 TO 30
30 FOR J=10 TO 30
40 PSET (I,J)
50 NEXT J,I
60 GOTO 60
RUN
```

Het vierkant waarvan in de eerdere voorbeelden alleen de hoekpunten werden afgebeeld, verschijnt nu in ingekleurde vorm doordat elk puntje binnen dit vierkant wordt geplaatst.

```
10 SCREEN 3
40 PSET (4*I,4*J)
RUN
```

Alleen de programmaregels 10 en 40 worden veranderd. Hetzelfde vierkant maar dan zestien maal vergroot (vier maal in elke richting) wordt afgebeeld. Merk op dat in programmaregel 40 steeds de locatie van de linker bovenhoek van het te plaatsen puntje wordt aangegeven.

```
10 SCREEN 2
RUN
```

Alleen programmaregel 10 werd veranderd. Het resultaat is dat nu een raster van puntjes wordt afgedrukt. Elk puntje geeft de linkerbovenhoek aan van een puntje dat zou zijn geplaatst bij een SCREEN 3-inrichting (voorlaatste voorbeeld).

```
10 SCREEN 3
40 PSET (4*I,4*J),15*RND(1)
RUN
```

Alleen programmaregels 10 en 40 werden veranderd. De punten zoals in het voorlaatste voorbeeld worden in willekeurige kleuren uitgevoerd.

```
10 SCREEN 2
RUN
```

Alleen programmaregel 10 werd veranderd. De punten zoals in het voorlaatste voorbeeld worden weer afgebeeld maar nu in diverse kleuren. De kleuren zijn niet helemaal willekeurig; in dit voorbeeld zijn de kleuren van de punten horizontaal twee aan twee gelijk.

De SCREEN 2 -instelling brengt met zich mee dat voor elk puntje niet elke kleur kan worden geselekteerd. De kleur wordt per groep van puntjes bepaald waarbij de laatst vastgestelde kleur geldig is. Kleuren worden per groep van acht naast elkaar liggende puntjes bepaald. De groepering is enigzins onduidelijk en is alleen goed te begrijpen wanneer men de exacte opbouw van het video-geheugen weet. Het is raadzaam om in de SCREEN 2 kleuren met mate te gebruiken en de diverse kleuringen op enige afstand van elkaar te gebruiken.

Voor de MSX-2-gebruiker gelden dezelfde restricties ook met betrekking tot de SCREEN 4-instelling.

De locatie op scherm kan ook met behulp van het STEP sleutelwoord worden bepaald. In dat geval spreekt men over een relatieve locatie; de beeldschermlocatie wordt gegeven ten opzichte van het laatste getekende punt. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2
20 PSET (0,0)
30 FOR I=1 TO 20
40 PSET STEP(10,10)
50 NEXT I
60 GOTO 60
RUN
```

Op één lijn, steeds 10 puntjes horizontaal en vertikaal verderop, verschijnen puntjes op het beeldscherm. Programmaregel 20 zet het eerste puntje in de linker bovenhoek; programmaregel 40 zet vervolgens steeds (10,10) verderop een nieuw puntje neer. Steeds wordt een volgende locatie bepaald door de aangegeven stapgrootten in horizontale en vertikale richting bij de vorige locatie op te tellen.

```
NEW
Ok
10 SCREEN 2
20 PSET (10,10)
30 PSET STEP (20,20)
40 PSET STEP (-20,0)
50 PSET STEP (20,-20)
60 GOTO 60
RUN
```

Dit programma geeft precies hetzelfde resultaat als het eerste voorbeeld; het geeft de vier hoekpunten van een vierkant aan.

De tussen haakjes vermelde locaties dienen integere waarden te bevatten. Indien één van deze uitdrukkingen een gebroken waarde als uitkomt heeft, wordt deze waarde ontdaan van een decimale fractie (de decimalen worden verwaarloosd). Indien de uiteindelijke waarde van een uitdrukking dan kleiner is dan –32768 of groter is dan 32767, volgt een foutmelding.

Het is opmerkelijk dat er puntjes *buiten* het beeldscherm kunnen worden gezet. Deze mogelijkheid heeft een reële betekenis. Hierop wordt in de diverse beschrijvingen van de grafische kommando's verder ingegaan.

Onder MSX-2 kunnen we onder SCREEN 5 tot en met SCREEN 8 aan het PSET-kommando nog een logische operator toevoegen.

De logische operator bepaalt de kleur waarin het beeldschermpuntje uiteindelijk uitgevoerd gaat worden.

Om de kleur van het uiteindelijke resulterende puntje te bepalen, worden de kleur van het te plaatsen puntje en de kleur van de betreffende beeldschermpositie voor de plaatsing in ogenschouw genomen. De kleurkoderingen ondergaan met elkaar een bewerking waarvan het resultaat uiteindelijk de kleurkode van het te plaatsen puntje zal worden.

De logische bewerkingen worden steeds op de binaire waarden van de kleurnummers uitgevoerd. Om met logische operatoren te kunnen werken, is dus een behoorlijk inzicht noodzakelijk in het binaire stelsel en dienen de logische operatoren 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	Het resulterende kleurnummer van het puntje is: SCREEN 5 en 7: 15 – opgegeven kleur SCREEN 6: 3 – opgegeven kleur SCREEN 8: 255 – opgegeven kleur
TAND TOR TXOR TPSET TPRESET	Al deze logische operatoren hebben dezelfde betekenis als bovengenoemde overeenkomstige operatoren. Het enige verschil is, dat kleurnummer 0 (transparant) eenvoudigweg nooit daadwerkelijk wordt geplaatst.

Een voorbeeld: tijdens een PSET-opdracht heeft het beeldschermpuntje waarop een nieuw punt geplaatst gaat worden, kleurkode 125 (dat kan onder de SCREEN 8-instelling). Het te plaatsen puntje krijgt kleurkode 210 mee. De logische operator is AND.

De resulterende kleur laat zich als volgt berekenen:

```
kleur 125:          binair   01111101
kleur 210:          binair   11010010    AND
                             --------
resulterende kleur           01010000 (80)
                             ========
```

Merk op dat MSX-basic op het kommando PRINT 125 AND 210 ook de uitkomst 80 geeft. Met MSX-basic kunnen we op deze wijze de resulterende kleur gemakkelijk berekenen.

In het volgende voorbeeld worden steeds weer puntjes geplaats en weer weggehaald. Dit voorbeeld laat zien dat met behulp van de XOR-operator een beeldpuntje weer in de oude staat kan worden hersteld door een tweemaal op precies dezelfde wijze het puntje te plaatsen.

```
NEW
Ok
10 COLOR 3,0,0:SCREEN 6:CLS
20 CIRCLE(255,106),150
30 PAINT (200,100),2,3
40 FOR X=0 TO 423 STEP 10:LINE (255,X/2)-(X,212-X/2),1:LINE
(512-X,X/2)-(X,0),1:NEXT
50 FOR X=200 TO 300 STEP 5
60 FOR Y=50 TO 150 STEP 5
70 PSET (X,Y),3,XOR
80 NEXT Y,X
90 GOTO 50
RUN
```

Ondanks de ingewikkeldheid van het patroon waarop de puntjes worden gezet wordt dit patroon toch steeds weer volledig hersteld.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PUT KANJI – plaats KANJI-teken.**

Dit bevel is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
PUT KANJI[<LOCATIE>],<KANJI-KODE>[,[<KLEURKODE>][,[<LOGISCHE ...
          ... OPERATOR>][,[<GROOTTE>]]]]\<...>,

<KANJI-KODE>::=<N>

<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

<KLEURKODE>::=<N>

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

<GROOTTE>::=<N>

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

betekenis

Voorafgaand aan deze behandeling dienen PSET en SCREEN te zijn bestudeerd. Alhoewel op alle europese MSX-2-machines het PUT KANJI bevel een bepaalde actie tot gevolg heeft, is dit bevel slechts zinvol indien een ROM-geheugen met daarin de JIS KANJI tekens in de machine is ingebouwd.

Met PUT KANJI kunnen japanse JIS KANJI tekens op een grafisch beeldscherm worden geprojekteerd. Hiertoe dient voorafgaand aan dit bevel SCREEN 5, SCREEN 6, SCREEN 7 of SCREEN 8 te zijn geactiveerd.

Indien het JIS KANJI ROM niet aanwezig is, worden er slechts complete blokken of niets geprojekteerd.

De opgegeven KANJI-KODE dient een waarde te hebben tussen –32769

en 32768 waarbij een eventuele decimale fraktie wordt verwaarloosd. Alleen de waarden vanaf 8481 tot en met 32382 hebben in dit geval echter zin. Indien een negatieve waarde als KANJI-KODE werd opgegeven, wordt er door het systeem automatisch 65536 bij opgeteld voordat het bevel wordt uitgevoerd.

De locatie heeft dezelfde vorm als bij PSET met dit verschil dat wanneer de locatie wordt weggelaten, er automatisch een STEP (16,0) wordt uitgevoerd. De locatie geeft het linkerbovenpunt aan van de te plaatsen KANJI-KODE.

De KLEURKODE mag alle voor het ingestelde beeldscherm geldige kleurkoderingen bevatten. Indien de KLEURKODE wordt weggelaten, wordt automatisch de ingestelde voorgrondkleur als kleur aangenomen.

De logische operator werkt hetzelfde als beschreven onder PSET met dit verschil dat de betreffende operator voor alle eventueel geprojekteerde beeldpunten van de KANJI-KODE geldt.

De GROOTTE mag de waarde 0,1 of 2 hebben. Een eventuele decimale fractie wordt verwaarloosd. Betekenis van GROOTTE:

0 de 16 bij 16 beeldpunten grote KANJI-KODE wordt volledig geprojekteerd.

1 Alleen de even beeldlijnen van de KANJI-KODE worden geprojekteerd. Het KANJI-teken wordt enigzins gedrongen geprojekteerd waarbij het nog slechts 8 beeldlijnen hoog is.

2 Als 1. Echter, nu worden juist alleen de oneven lijnen geprojekteerd.

Indien de grootte niet wordt opgegeven, wordt kode Ø toegepast.
In het onderstaande voorbeeld worden een tiental KANJI-tekens geprojekteerd.

```
NEW
Ok
10 SCREEN 5:COLOR 15,4,4:CLS
20 PRESET(40,80)
30 FOR I=0 TO 9:PUT KANJI ,8481+I:NEXT I
40 GOTO 40
RUN
```

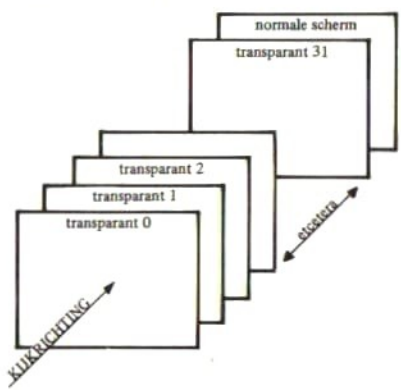
moeilijkheidsgraad . . . . . **moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **PUT SPRITE - plaats 'geest'**

schrijfwijze

```
PUT SPRITE<TRANSPARANT>[,[<LOCATIE>][,[<KLEUR>][,<SPRITE>]]]\<...>,
<LOCATIE>::=[STEP](<HORIZONTAAL>,<VERTIKAAL>)
<HORIZONTAAL>::=<N>
<VERTIKAAL>::=<N>
<TRANSPARANT>::=<N>
<KLEUR>::=<N>
<SPRITE>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
<...>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Alvorens dit kommando te bestuderen, is het noodzakelijk om eerst SPRITE$, COLOR, SCREEN en PSET grondig te hebben bestudeerd. Bovendien dienen de schrijfwijzen van COLOR SPRITE en COLOR SPRITE$ (alleen MSX-2) te zijn bestudeerd.



Voorafgaand aan het PUT SPRITE-kommando dient SCREEN 1 of een grafisch scherm te zijn ingesteld.

De indeling die het beeldscherm heeft voor SPRITE staat op de voorgaande bladzijde.

Het beeldscherm bestaat in feite uit een normaal (achtergrond) scherm met daarvoor 32 transparante schermen. Tot nu toe gebruikten we deze 32 transparanten niet maar tekenden of schreven we slechts op het achtergrondscherm.

Met PUT SPRITE kan een eerder met SPRITE$ (...) =... ontworpen sprite op een willekeurige plaats in een willekeurige kleur op een willekeurige transparant worden geplaatst. De volgende regels gelden:

- Onder MSX-1 kan een sprite slechts één kleur hebben. Onder MSX-2 kan een sprite onder SCREEN 4 tot en met SCREEN 8 meerdere kleuren aannemen; zie verder.
- Er kan altijd maar één sprite per transparant worden geplaatst. Indien men probeert, een tweede sprite op een transparant te plaatsen, verdwijnt onmiddellijk de eerder geplaatste sprite.
- Indien een sprite geheel of gedeeltelijk achter een andere sprite worden geplaatst, dan wordt inderdaad ook visueel de achterste sprite voor de voorste geplaatst. De achterste sprite is dus slechts gedeeltelijk of niet te zien. In MSX-2 bestaat hierop een uitzondering, zie verder.
- Er kunnen maximaal vier sprites op één horizontale lijn naast elkaar worden geplaatst. Indien meer dan vier sprites op verschillende transparanten op één horizontale lijn worden geplaatst, dan verdwijnen de vijfde en daarachterliggende sprites op de beeldlijnen waarover meer dan vier sprites tegelijk worden geprojekteerd. Onder SCREEN 4 tot en met SCREEN 8 (MSX-2) kunnen op deze wijze maximaal acht sprites op één lijn worden geprojekteerd.
  Meerdere sprites onder elkaar is nooit een probleem.
- Een sprite is op de niet getekende plaatsen (de witte ruitjes, zie SPRITE$) doorzichtig. Zo geeft het passeren van twee sprites altijd een visueel aanvaardbaar effekt.

Welke scherminstelling ook is geactiveerd, voor sprites geldt altijd de volgende schermindeling.

Een sprite kan zich dus geheel of gedeeltelijk buiten het beeld bevinden.

Achter PUT SPRITE dienen we als eerste gegeven het transparantnummer op te geven van het transparant waarop de sprite moet worden geplaatst. Een eventueel eerder op deze transparant geplaatste sprite verdwijnt hiermee automatisch. Het transparantnummer wordt gegeven in een numerieke uitdrukking die niet kleiner dan nul en niet groter dan 31 mag zijn. Een eventuele decimale fractie wordt verwaarloosd.

Als tweede gegeven achter PUT SPRITE kan de locatie van de betreffende sprite worden opgegeven. Deze locatie dient aan dezelfde eisen te voldoen als de locatie, beschreven onder het PSET-kommando en kan dus ook zowel relatief als absoluut zijn. Indien de locatie wordt weggelaten, wordt de laatste voor dat transparant geldende locatie behouden. Werd nog niet eerder locatie opgegeven, dan wordt automatisch de locatie (0,209) ingenomen onder SCREEN 1 tot en met SCREEN 3 of de locatie (0,217) onder andere SCREEN-instellingen (MSX-2). In beide gevallen blijven deze sprites zo onzichtbaar.

Pas op: indien een relatieve locatie werd opgegeven (STEP...), dan wordt deze relatieve locatie uitgevoerd ten opzichte van de laatst gegeven locatie. Op welk grafisch kommando deze laatste locatie van toepassing was, is dan niet van belang. De opgegeven locatie is in ieder geval niet zondermeer de locatie ten opzichte van de laatst opgegeven locatie VOOR HET BETREFFENDE TRANSPARANT.

Indien deY-coördinaat van een sprite (de vertikale positionering) gelijk is aan 208 (SCREEN 1,2, en 3) of 216 (SCREEN 4 en verder, MSX-2)

dan heeft dat tot effekt dat alle sprites die op achterliggende transparanten zijn geplaatst, onzichtbaar worden.

Als derde gegeven dient een kleur te worden opgegeven waarin de sprite dient te worden uitgevoerd. Deze kleurkode dient minimaal gelijk te zijn aan nul en maximaal gelijk te zijn aan 15 terwijl een decimale fraktie wordt verwaarloosd. Deze kleurkode verwijst naar het kleurenpalette zoals dat onder COLOR werd behandeld.

Indien geen kleurkodering werd opgenomen, wordt de laatst geldende kleur voor het betreffende transparant gehandhaafd. Indien nog niet eerder een kleur werd geselekteerd voor het betreffende transparant, wordt de ingestelde voorgrondkleur als geldende kleur genomen.

MSX-2: Pas op! Onder SCREEN 6 en SCREEN 8 zijn andere kleurkoderingsmethodieken van toepassing (zie COLOR). Toch blijft in dat geval de kleurkodering 0...15 gelden voor sprites en verwijst deze kleurkodering naar het kleurenpalette dat standaard is ingesteld en eventueel met COLOR= (...) werd veranderd.

Onder SCREEN 4 tot en met SCREEN 8 mogen bij de kleurkodering de volgende getallen worden opgeteld:

+16 maakt geen verschil

+32 een botsing van een sprite met de op deze wijze gekodeerde sprite wordt met ON SPRITE GOSUB niet meer gedetecteerd (zie ON SPRITE GOSUB).

+64 Als +32. Daarbij wordt de op deze wijze gekodeerde sprite onzichtbaar. Echter, het sprite-gedeelte dat zich ter hoogte van een voorliggende sprite bevindt die niet op deze wijze werd gekodeerd, wordt wel zichtbaar. Indien de sprite een overlapping heeft met deze voorliggende sprite, dan wordt op deze plaats niet alleen de voorste geprojekteerd maar worden zij beide door elkaar geprojekteerd waarbij de kleuren van de sprites voor deze plaats een logisch AND ondergaan.

Als vierde en laatste gegeven kan de betreffende sprite (het betreffende spelfiguur) worden genoemd. Dit figuur diende eerder met behulp van SPRITE$ te zijn ontworpen; zie aldaar. Indien dit laatste gegeven achterwege blijft, dan neemt MSX aan dat het nummer van de sprite gelijk is aan het nummer van het transparant.

Hieronder volgen enkele voorbeelden. Het is noodzakelijk om veel met sprites te oefenen alvorens aan wat meer serieus programmeerwerk te beginnen.

Voorbeeld 1, definitie van een 8 bij 8 sprite. Dit voorbeeld wordt bij de behandeling van SPRITE$ uitvoerig behandeld.

```
NEW
Ok
10 SPRITE$(0)=CHR$(24)+CHR$(60)+CHR$(60)+CHR$(86)+CHR$(126)+
CHR$(126)+CHR$(230)+CHR$(124)
```

Met dit kommando is spelfiguur nummer 0 bepaald. Het spelfiguur is gelijk aan het eerder bij SPRITE$ bepaalde figuur met dit verschil dat er decimale en geen binaire konstanten zijn gebruikt.
Om geen foutmelding te krijgen, dienen we voorafgaand aan dit kommando eerst de juiste beeldscherminstelling te realiseren:

```
5 SCREEN 2,1
```

Gekozen werd (zie SCREEN) voor een grafisch beeldscherm met een sprite-grootte van 8 bij 8 beeldpunten en een vier maal zo grote weergave van deze sprite.
Vervolgens gaan we de sprite op het beeldscherm plaatsen om te kijken hoe de sprite er nu werkelijk uit ziet:

```
20 PUT SPRITE 0,(100,100),15,0
30 GOTO 30
RUN
```

Op regel 20 werd de spelfiguur op transparant 0 geplaatst en wel op locatie (100,100). Als kleur werd voor wit gekozen en, in dit geval ten overvloede, werd voor sprite nummer 0 (SPRITE$(0)) gekozen. Op het beeldscherm zien we een soort 'spookje', bruikbaar in diverse reactiespelletjes. Regel 30 is een eeuwigdurende loop om het grafische beeld vast te houden. Onderbreek met CONTROL-STOP.
Vervolgens laten we een tweede spookfiguurtje in een andere kleur achter dit eerste spookje langs bewegen:

```
30 FOR I=0 TO 200
40 PUT SPRITE 1,(I,104),10,0
50 NEXT I
60 GOTO 30
RUN
```

Snel zien we een tweede spookje (donkergeel) achter het eerste spookje langs gaan, zich iets lager bewegend. Om deze beweging goed volgbaar te maken, kunnen we eventueel de volgende wijziging intoetsen:

```
30 FOR I=0 TO 200 STEP .1
```

Een derde spookje kunnen we weer achter het tweede langs laten gaan:

```
45 PUT SPRITE 2,(104,I),8,0
RUN
```

Om een grote snelheid in de beweging te realiseren, kunnen we bijvoorbeeld de volgende verandering aanbrengen:

```
30 FOR I=0 TO 200 STEP 2
```

Steeds zien we dat in de for-next-lus op de transparanten nummer 1 en 2 de figuren op een andere plaats worden afgebeeld. De eerder op deze transparant geplaatste afbeelding verdwijnt hierbij; er kan maar één sprite tegelijk op een transparant worden afgebeeld.

Vervolgens kunnen we deze figuurtjes zich voor een bepaalde achtergrond laten bewegen:

```
11 LET A=RND(-TIME)
12 FOR I=1 TO 20
13 CIRCLE (RND(1)*256,RND(1)*192),RND(1)*80
14 NEXT I
15 FOR I=1 TO 10
16 PAINT (RND(1)*256,RND(1)*192)
17 NEXT I
RUN
```

De figuurtjes verplaatsen zich nu voor een achtergrond van willekeurige cirkels waarvan gedeelten al of niet zijn ingekleurd. Zie de behandeling van RND en TIME voor een optimaal begrip. Elke keer dat het programma wordt opgestart, is de achtergrond weer wat anders.

Door op transparant nummer 1 de y-coördinaat 208 te gebruiken, kunnen we het spookje op transparant nummer 2 bijvoorbeeld laten knipperen:

```
40 PUT SPRITE 1,(0,208+Q)
41 LET Q=NOT(Q)
RUN
```

Regel 41 zorgt ervoor dat variabele Q afwisselend de waarde 0 en –1 krijgt. Hierdoor wordt op regel 40 afwisselend een figuurtje wel en niet op de y-coördinaat 208 van transparant 1 geplaatst. Hierdoor wordt het figuurtje op transparant 2 afwisselen wél en niet zichtbaar waardoor een knipperend effect ontstaat.

Een spelfiguur kan men laten bewegen door bijvoorbeeld afwisselend andere, iets gewijzigde sprites op dezelfde locatie op eenzelfde transparant af te beelden. Het volgende voorbeeld laat een zich schuin over het beeld verplaatsend en pulserend cirkeltje zien:

```
NEW
Ok
10 SCREEN 2,1
20 SPRITE$(0)=CHR$(126)+STRING$(6,129)+CHR$(126)
30 SPRITE$(1)=CHR$(0)+CHR$(60)+STRING$(4,66)+CHR$(60)+CHR$(0)
40 SPRITE$(2)=STRING$(2,0)+CHR$(24)+STRING$(2,36)+CHR$(24)+
STRING$(2,0)
50 SPRITE$(3)=STRING$(3,0)+STRING$(2,24)+STRING$(3,0)
60 FOR I=0 TO 3.9 STEP .05
70 PUT SPRITE 0,STEP(1,1),15,I
80 NEXT I
90 GOTO 60
RUN
```

Om het spelfiguur zich te laten verplaatsen, werd gebruik gemaakt van de STEP-mogelijkheid. Om het spelfiguur continu te laten veranderen, werd gebruik gemaakt van vier apart gedefinieerde sprites die achter elkaar op hoegenaamd dezelfde locatie en op dezelfde transparant worden geplaatst. Het gebruik van de STEP op regel 60 dient om de verandering van het spelfiguur niet te snel te laten geschieden; probeer de stepwaarde maar eens te veranderen.

Het voorbeeld laat een nog niet eerder genoemd verschijnsel zien; een sprite die aan de ene kant in verband met de locatie geheel buiten de toegestane grenzen treedt, wordt aan de tegenovergestelde kant weer in beeld teruggebracht. Dit gebeurt door bij/van een buiten de toegestane grenzen tredende coördinaat net zo vaak de waarde 256 op te tellen/af te trekken totdat de betreffende coördinaat binnen de toegestane grenzen valt.
De programmaregels:

```
10 PUT SPRITE 9,(100,100),15,1
10 PUT SPRITE 9,(356,356),15,1
10 PUT SPRITE 9,(-156,-156),15,1
```

hebben dus allemaal precies hetzelfde effekt.

Pas op: Indien een sprite zich op een horizontale positie vanaf -1 tot en met -32 bevindt, dan wordt deze niet op horizontale locatie 255 ... 224 geprojekteerd zoals men zou verwachten. Om een bepaalde reden die in hoofdstuk 14 wordt uitgelegd, wordt de sprite in dat geval in het geheel niet geprojekteerd. Ook wordt een links of rechts uit het beeld geschoven sprite-**gedeelte** nooit aan de andere kant van het scherm geprojekteerd.

Hieronder volgt een programma dat de eerder in de behandeling van SPRITE$ ontworpen 16 bij 16 sprite gebruikt:

```
NEW
Ok
10 COLOR 2,1,1:KEY OFF:LET A=RND(-TIME)
20 RESTORE 100:LET A$="":FOR I=1 TO 32
30 READ A:LET A$=A$+CHR$(A):NEXT I
40 SCREEN 1,3:SPRITE$(0)=A$
50 FOR I=1 TO 31
60 PUT SPRITE I,(RND(1)*256,RND(1)*256),RND(1)*16,0
70 PRINT "MSX-BASIC *** ";
80 NEXT I
90 GOTO 50
100 DATA 3,15,15,31,27,25,31,31,30,59,124,124,63,31,3,1
110 DATA 224,240,248,248,184,152,248,248,124,220,30,60,120
120 DATA 240,192,128
RUN
```

Op beeld wordt continu de tekst MSX-BASIC *** afgedrukt. Op de voorgrond worden op willekeurige posities in willekeurige kleuren, spookfiguurtjes voor- en achter elkaar afgebeeld. Ook zien we dat diverse malen een sprite slechts gedeeltelijk wordt afgedrukt; er staan op dat moment méér dan vier sprites op dezelfde regel. Daarbij zien we een enkele keer dat er plotseling een aantal sprites geheel verdwijnen. In dat geval heeft één van de sprites gedurende een for-next-loop een Y-coördinaat 208 aangenomen.

De verdere behandeling van PUT SPRITE is alleen voor MSX-2 van toepassing.

Onder SCREEN 4 tot en met SCREEN 8 is het mogelijk om meerdere kleuren per sprite toe te wijzen en om alleen de kleur van een sprite te wijzigen.

Alleen de kleur van een sprite kan men wijzigen met het COLOR SPRITE-kommando. Achter dit kommando dient tussen haakjes het transparantnummer te worden opgenomen waarop de betreffende sprite is geplaatst, gevolgd door een gelijk-teken en de kode van de gewenste kleur.

Een voorbeeld:

```
NEW
Ok
10 RESTORE 60:LET A$="":FOR I=1 TO 32
20 READ A:LET A$=A$+CHR$(A):NEXT I
30 SCREEN 5,3:SPRITE$(0)=A$
40 PUT SPRITE 5,(100,100),15,0
50 COLOR SPRITE(5)=RND(1)*16:FOR I=1 TO 100:NEXT:GOTO 50
60 DATA 3,15,15,31,27,25,31,31,30,59,124,124,63,31,3,1
70 DATA 224,240,248,248,184,152,248,248,124,220,30,60,120
80 DATA 240,192,128
RUN
```

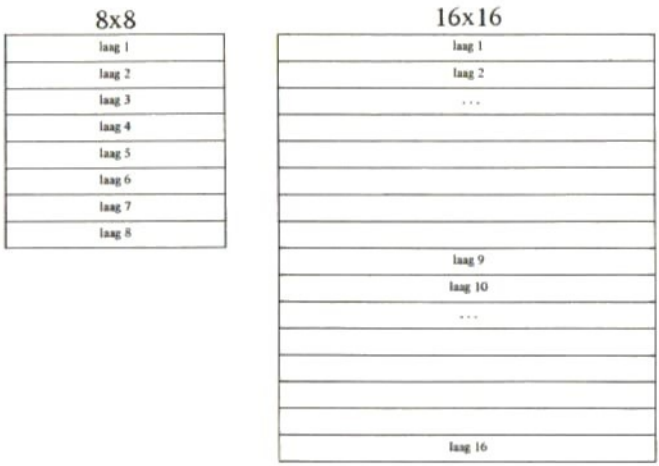
In het bovenstaand voorbeeld werd eerst een sprite geplaatst. Daarna wordt met het COLOR SPRITE-kommando de kleur voortdurend veranderd.

Voor de kleurkode, gebruikt met het SPRITE COLOR kommando gelden dezelfde voorwaarden en eigenschappen als vermeld onder de MSX-2 noot bij de kleurkode van PUT SPRITE hiervoor.

Een 8x8 en een 16x16 sprite kan men in de volgende schema's opgebouwd zien.

Per laag kan nu een verschillende kleur worden toegekend aan een sprite met behulp van het COLOR SPRITE$-kommando. Achter het COLOR SPRITE$-kommando dient tussen haakjes het transparantnummer te worden opgenomen waarop de in te kleuren sprite is afgebeeld, gevolgd door een gelijk teken. Na dit gelijkteken dienen de kleurkodes voor elke laag in een alfanumerieke uitdrukking te worden opgenomen. Dit kan het beste geschieden met behulp van de CHR$ (...)-funktie. Neem in dat geval na het gelijkteken de uitdrukking CHR$ (...)

+CHR$ (...) +CHR$ (...) +... op en zet tussen haakjes steeds de kleurkode van de volgende laag.



een voorbeeld:

```
NEW
Ok
10 RESTORE 70:LET A$="":FOR I=1 TO 32
20 READ A:LET A$=A$+CHR$(A):NEXT I
30 SCREEN 5,3:SPRITE$(0)=A$
40 PUT SPRITE 5,(100,100),15,0
50 A$="":FOR I=1 TO 16:A$=A$+CHR$(RND(1)*16):NEXT I
60 COLOR SPRITE$(5)=A$:GOTO 50
70 DATA 3,15,15,31,27,25,31,31,30,59,124,124,63,31,3,1
80 DATA 224,240,248,248,184,152,248,248,124,220,30,60,120
90 DATA 240,192,128
RUN
```

In dit voorbeeld wordt een 16x16 sprite vergroot geplaatst waarna per laag de kleuren steeds worden verwisseld.

Voor de kleurkodering, gebruikt in COLOR SPRIT$, gelden de volgende opmerkingen:

0 . . . 15 correspondeert met palette 0 . . . 15

Bij de kleurkoderingen mogen de volgende getallen worden opgeteld:

+16 geen betekenis

+32 de op deze wijze gekodeerde laag veroorzaakt bij „aanraking" met een andere sprite nooit meer een met behulp van ON SPRITE GOSUB te detecteren botsing.

+64 idem. Daarbij wordt de op deze wijze gekodeerde sprite-laag onzichtbaar. Echter, indien deze sprite-laag zich ter hoogte van een voorliggende sprite-laag bevindt die niet op deze wijze werd gekodeerd, wordt het gedeelte van de laag dat zich ter hoogte van deze sprite-laag bevindt, wel geprojekteerd. Indien met deze voorliggende sprite-laag een overlappingsgebied ontstaat, dan worden deze lagen door elkaar geprojekteerd waarbij de kleuren een logisch AND met elkaar ondergaan.

+128 een op deze wijze gekodeerde sprite-laag wordt 32 beeldpunten verder naar links geprojekteerd tenzij de x-coördinaat van deze sprite negatief werd opgegeven.

Een voorbeeld:

```
50 A$="":FOR I=1 TO 16:A$=A$+CHR$(RND(1)*256):NEXT I
```

Alleen regel 50 werd veranderd. Nu kunnen er ook kleurkoderingen groter dan 15 worden gegenereerd. De hierboven beschreven effekten verschijnen en verdwijnen willekeurig.

Opmerkingen:

– Indien de kleurkode in het PUT SPRITE-kommando wordt gebruikt, verdwijnt het eerdere effekt van COLOR SPRITE($). Andersom heft een later gebruik van COLOR SPRITE ($) de in het PUT SPRITE-kommando opgegeven kleurkode op.

– Een met een negatieve x-coördinaat geplaatste sprite schuift bij het gebruik van COLOR SPRITE ($) op deze sprite 32 beeldpunten naar rechts op tenzij er bij de kleurkodering de waarde 128 wordt opgeteld (alleen bij COLOR SPRITE$ mogelijk).

– Een sprite zélf heeft geen kleur. De kleur wordt bepaald door het transparant waarop deze is geplaatst. Kleuren kunnen ook niet per SPRITE maar moeten per TRANSPARANT worden ingesteld.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **READ is lezen**

schrijfwijze

```
READ<VARIABELE>{,<VARIABELE>}

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

betekenis

De sleutelwoorden READ, DATA en RESTORE worden onder het sleutelwoord DATA behandeld. Zie aldaar.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **REM is afkorting van remark – opmerking**

schrijfwijze

```
REM<KOMMENTAAR>

<KOMMENTAAR>::={<KARAKTER>}
               {----------}
               {    "     }

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

betekenis

Het REM-kommando dient slechts ter opname van kommentaar in het programma. Het programma zal bij uitvoering de REM-kommando's overslaan.

Pas op: Het REM-kommando dient altijd als LAATSTE kommando in een programmaregel te worden opgenomen. Voorbeeld:

```
NEW
Ok
10 REM DIT PROGRAMMA DOET NIETS
20 STOP
RUN
Break in 20
Ok
```

Indien kommentaar NAAST kommando's dient te worden opgenomen, kan ook het enkele aanhalingsteken (') worden gebruikt. Ook kommentaar, opgenomen achter een ('), dient als laatste in een programmaregel te zijn opgenomen. Voorbeeld:

```
NEW
Ok
10 REM VOORBEELDPROGRAMMA
20 INPUT A'DE WAARDE VAN A WORDT HIER INGEGEVEN
30 PRINT A'DE WAARDE VAN A WORDT HIER AFGEDRUKT
RUN
? 12.5
 12.5
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RENUM is afkorting van renumber – opnieuw voorzien van nummers**

schrijfwijze

```
RENUM[<NIEUW REGELNUMMER>][,[<OUD REGELNUMMER>][,<STAPGROOTTE>]]

<NIEUW REGELNUMMER>::=<REGELNUMMER>

<OUD REGELNUMMER>::=<REGELNUMMER>

<STAPGROOTTE>::=<TYPE 1 INTEGERE KONSTANTE>

<REGELNUMMER>::=<ZIE ALGEMENE SPECIFICATIES>

<TYPE 1 INTEGERE KONSTANTE>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met RENUM kunnen programmaregels van een nieuw regelnummer worden voorzien. Dit kan bijvoorbeeld praktisch zijn wanneer u programmaregels wilt tussenvoegen maar er tussen de twee programmaregels niet meer voldoende plaats is.

Indien u slechts RENUM intoetst, dan wordt het volledige programma hernummerd. Het eerste regelnummer is regelnummer 10 terwijl de volgende regels steeds een regelnummer krijgen dat 10 hoger ligt.
Indien u RENUM ingeeft, gevolgd door een regelnummer dan heeft dat tot gevolg dat de eerste programmaregel dit nummer krijgt en dat alle volgende programmaregels steeds een nummer krijgen dat 10 hoger ligt.
Indien u RENUM ingeeft, gevolgd door een regelnummer, een komma en weer een regelnummer, dan heeft dat tot gevolg dat het programma vanaf het tweede ingegeven regelnummer wordt hernummerd. Het eerste gedeelte blijft dus onveranderd. Het eerste te hernummeren regelnummer krijgt het eerste ingegeven regelnummer. Indien het eerste regelnummer wordt overgeslagen (dus alleen een komma en het tweede regelnummer) dan krijgt het eerste te hernummeren regelnummer het nummer 10 mee. In beide gevallen wordt er weer met stappen van 10 opgenummerd.
Indien u RENUM intoetst, gevolgd door een regelnummer, een komma, een regelnummer, een komma en een stapgrootte dan heeft dat tot effekt dat behalve dat er hernummerd wordt zoals zojuist beschreven

daarbij de ingegeven stapgrootte wordt aangenomen in plaats van de normaal gehanteerde stapgrootte van 10. Indien een RENUM wordt gegeven met daarbij alleen de stapgrootte gedefinieerd (RENUM „ stapgrootte), dan wordt het programma geheel hernummerd waarbij het eerste regelnummer gelijk is aan 10. De volgende regels liggen steeds een stapgrootte hoger.

Voorbeeld:

```
NEW
Ok
10 REM
20 REM
30 REM
RENUM 1,,1
Ok
LIST
1 REM
2 REM
3 REM
Ok
RENUM
Ok
LIST
10 REM
20 REM
30 REM
Ok
RENUM 1000,20,5
Ok
LIST
10 REM
1000 REM
1005 REM
Ok
```

RENUM werkt ALLE regelnummers naar behoren bij. Zo ook bijvoorbeeld de regelnummers die in een GOTO of een GOSUB of een ON... GOTO kommando vermeld staan.

Indien RENUM in een programmaregel is opgenomen, dan probeert MSX-basic ten onrechte ook om de regelnummers en de stapgrootte die eventueel achter het RENUM-kommando zijn vermeld, te hernummeren. Het één en ander kan leiden tot foutmeldingen en curieuze situaties. Het gebruik van een RENUM in een programmaregel is echter meestal vrij zinloos.

Kommando's die door RENUM kunnen worden herzien:

```
RENUM
GOTO
GOSUB
ON ... GOTO
ON ... GOSUB
AUTO
LIST
LLIST
DELETE
RUN
RESUME
RETURN
RESTORE
CONSTRUCTIES MET ERL
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RESTORE is terug zetten**

schrijfwijze

```
RESTORE[<REGELNUMMER>]

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

betekenis

De sleutelwoorden READ, DATA en RESTORE worden onder het sleutelwoord DATA behandeld. Zie aldaar.

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RESUME is hervatten**

schrijfwijze

```
RESUME[0¦NEXT¦<REGELNUMMER>]

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

betekenis

Dit kommando wordt behandeld onder ON ERROR GOTO; zie aldaar.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RETURN is keer terug**

schrijfwijze

```
RETURN[<REGELNUMMER>]

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

betekenis

Dit kommando wordt tesamen met GOSUB behandeld.
Zie aldaar.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **RIGHT$ is afkorting van RIGHT part of string – rechter gedeelte van de string**

schrijfwijze

```
RIGHT$(<BASISSTRING>,<AANTAL TEKENS>)

<BASISSTRING>:::=<A>

<AANTAL TEKENS>:::=<N>

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

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

betekenis

Deze funktie geeft van een alfanumerieke uitdrukking het rechtergedeelte als resultaat. Hoe groot dit rechtergedeelte is, wordt bepaald door het opgegeven aantal tekens. Bijvoorbeeld:

```
PRINT RIGHT$("MSX-BASIC",5)
BASIC
Ok
```

Het aantal tekens mag niet kleiner zijn dan nul en niet groter dan 255. Een eventuele decimale fractie wordt verwaarloosd.

Nog een voorbeeld:

```
NEW
Ok
10 LINE INPUT "HOE HEET U ?:";NAAM$
20 INPUT "EN HOEVEEL LETTERS TELT UW ACHTERNAAM ";AANTAL
30 PRINT "HALLO MENEER/MEVROUW ";RIGHT$(NAAM$,AANTAL)
RUN
HOE HEET U ?:JOHAN KARDINAAL
EN HOEVEEL LETTERS TELT UW ACHTERNAAM ? 9
HALLO MENEER/MEVROUW KARDINAAL
Ok
```

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **RND is afkorting van random – willekeurig**

schrijfwijze

```
RND(<N>)

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

betekenis

Deze funktie geeft als resultaat een toevalsgetal. Dit getal is minimaal gelijk aan nul maar altijd kleiner dan 1.
De tussen haakjes opgenomen numerieke uitdrukking heeft de volgende betekenis:

– kleiner dan nul:	het eerste toevalsgetal uit een door de numerieke uitdrukking bepaalde reeks wordt berekend en als resultaat gegeven. Voortzetting van de reeks toevalsgetallen dient te geschieden door bij elke volgende RND een uitdrukking met positieve waarde te gebruiken
– groter dan nul:	het volgende toevalsgetal uit een reeds aangevangen reeks wordt gepresenteerd. Indien de reeks niet eerder werd aangevangen, wordt het getal 0.59521943994623 als eerste uit de reeks gegenereerd
– gelijk aan nul:	het laatste toevalsgetal wordt herhaald. Indien nog niet eerder een toevalsgetal werd bepaald, wordt het getal 0.40649651372358 gegenereerd

De term toevalsgetal is niet correct. De toevalsgetallen worden door de computer BEREKEND waarbij een zo goed mogelijke verdeling van de getallen wordt betracht. Deze berekening is echter zo complex dat een volgend getal uit een reeks niet voorspelbaar is tenzij de reeks reeds eerder werd bestudeerd.

Een voorbeeld: een speltoepassing:

```
NEW
Ok
10 REM DOBBELSTEEN
20 INPUT "GEEF EEN WAARDE IN ";A
25 OGEN=RND(-A)
30 PRINT "GEEF EEN TOETS IN"
40 IF INKEY$="" THEN 40
50 OGEN=INT(RND(1)*6+1)
60 PRINT "IK GOOIDE";OGEN;"OGEN"
70 GOTO 30
RUN
GEEF EEN WAARDE IN ? 33
GEEF EEN TOETS IN
IK GOOIDE 1 OGEN
GEEF EEN TOETS IN
IK GOOIDE 5 OGEN
GEEF EEN TOETS IN
IK GOOIDE 1 OGEN
GEEF EEN TOETS IN
IK GOOIDE 2 OGEN
GEEF EEN TOETS IN
```

(etcetera; onderbreken met CONTROL-STOP)

Op regel 25 wordt naar aanleiding van een ingegeven getal een reeks toevalsgetallen aangevangen. Op regel 50 wordt dan steeds de volgende uit de reeks bepaald. Met dit toevalsgetal wordt een berekening uitgevoerd, zodanig dat het eindresultaat een willekeurig geheel getal is, gelijk aan 1, 2, 3, 4, 5, of 6.

Zie ook de behandeling van TIME.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **RUN is rennen, lopen, werken**

schrijfwijze

```
RUN [<REGELNUMMER>[{<KARAKTER>}\<CIJFER><...>]]
    [-----------------------------------------]
    [          <PROGRAMMANAAM>[,R]            ]
```

<REGELNUMMER>::=<ZIE ALGEMENE SPECIFICATIES>

<KARAKTER>::=<ZIE ALGEMENE SPECIFICATIES>

<CIJFER>::=<ZIE ALGEMENE SPECIFICATIES>

<PROGRAMMANAAM>::=<A>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

betekenis

Het RUN-kommando is het meest eenvoudige maar ook het meest belangrijke kommando dat MSX-basic kent. Met dit kommando kunnen we een in het werkgeheugen aanwezig MSX-basic programma activeren.

eenvoudig gebruik

In de meest eenvoudige vorm activeren we met RUN een programma:

```
NEW
Ok
10 PRINT "TEST 1"
20 PRINT "TEST 2"
RUN
TEST 1
TEST 2
Ok
```

Indien we achter het RUN-kommando een regelnummer opnemen, zal het programma vanaf dit regelnummer worden geactiveerd. Voorbeeld:

```
NEW
Ok
10 PRINT "TEST 1"
```

```
20 PRINT "TEST 2"
RUN 20
TEST 2
Ok
```

RUN sluit alle eventueel openstaande kanalen en maakt daarbij alle variabelen schoon. Merk op dat RUN standaard onder funktietoets 5 en 10 aanwezig is in twee verschillende uitvoeringen.

gevorderd gebruik

In verband met het gevorderde gebruik dienen eerst het SAVE-kommando en het OPEN-kommando te worden bestudeerd.
Een speciale vorm van het RUN-kommando is de RUN met daarachter een programmanaam. In deze vorm gebruikt, wordt na het RUN-kommando het eerder via SAVE vastgelegde programma eerst opgehaald van het aangesproken randapparaat (b.v. cassetterecorder) en pas dan geactiveerd. Met de toevoeging ,R bereiken we dat de eventueel geopende kanalen niet automatisch door het RUN-kommando worden gesloten. Bijvoorbeeld (in de cassetterecorder dient een cassette met daarop het programma TEST te worden geplaatst en de cassetterecorder dient op afspelen te worden gezet):

`RUN "TEST"`	Het programma TEST wordt van cassetteband geladen en uitgevoerd.
`RUN "CAS:TEST",R`	Idem, maar nu worden eventuele openstaande kanalen niet gesloten.

In het tweede voorbeeld mag RUN worden vervangen door LOAD; zie de behandeling van LOAD.

Zie voor de toegestane programma-namen de voorwaarde voor bestandsnamen bij het OPEN-kommando.

MSX-2: Natuurlijk kan ook MEM: als geldig randapparaat worden aangesproken. Zie CALL MEMINI, CALL MNAME, CALL MKILL en CALL MFILES.

SLEUTELWOORD # SAVE

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SAVE is veilig stellen**

schrijfwijze

```
SAVE<BESTANDSNAAM>[,A]

<BESTANDSNAAM>::=<A>

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

betekenis

Het is noodzakelijk dat eerst de behandelingen van CLOAD en CSAVE worden doorgenomen.

Met het SAVE-kommando kunnen programma's op een extern opslagmedium worden opgeslagen. Dit vastleggen gebeurt ongekodeerd; vastleggen geschiedt tekstueel, teken voor teken, zoals het programma ook werd ingegeven (volgens de ASCII-kodering). Hierdoor is een bestand dat ontstaat door middel van een SAVE ook later weer met een OPEN-kommando (zie de betreffende behandeling) te benaderen en regel voor regel als tekst in te lezen. *

Een programma, via SAVE vastgelegd, kan later via RUN direkt weer worden opgestart.

De achtervoeging ,A is toegestaan maar heeft geen zin.*

De toegestane bestandsnamen worden bij behandeling van het OPEN-kommando nader omschreven. We geven een voorbeeld op cassetterecorder.

Voorbeeld: het SAVEn van een programma op cassetteband

```
NEW
Ok
10 REM TESTPROGRAMMA
20 PRINT "DIT IS EEN TEST"
30 GOTO 20
SAVE"CAS:TEST"
Ok
```

(CAS: staat voor cassetterecorder; TEST is de programmanaam)

Voordat het SAVE-kommando wordt ingetoetst, dient een cassetteband in de recorder te worden geplaatst en dient deze recorder op opnemen te worden gezet. Na enige tijd verschijnt de Ok-melding; het programma is vastgelegd op band.

MSX schrijft geen lees/schrijfcontrole voor; een slechte kwaliteit apparatuur of bandmateriaal kan veroorzaken dat er tijdens het vastleggen fouten optreden.

Een in een programmaregel opgenomen SAVE-kommando heeft tot gevolg dat het programma eerst wordt vastgelegd en dat daarna de Ok-melding verschijnt; het programma gaat niet verder.

Zie voor verdere voorbeelden het LOAD-kommando.

*SAVE op schijf gaat **niet** in tekstuele vorm tenzij de ,A-optie werd toegepast. Hier heeft deze toevoeging dus wél zin. Zie deel 2.

SLEUTELWOORD SCREEN

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SCREEN is beeldscherm**

schrijfwijze

```
MSX-1:SCREEN[<INSTELLING>][,[<SPRITEGROOTTE>][,[<KLIK>][,[<CASSETTE ...
     ... SCHRIJFSNELHEID>][,<PRINTERTYPE>]]]]\<...>,

MSX-2:SCREEN[<INSTELLING>][,[<SPRITEGROOTTE>][,[<KLIK>][,[<CASSETTE ...
     ... SCHRIJFSNELHEID>][,[<PRINTERTYPE>][,<VERVLECHTING>]]]]]\<...>,

<INSTELLING>::=<N>
<SPRITEGROOTTE>::=<N>
<KLIK>::=<N>
<CASSETTESCHRIJFSNELHEID>::=<N>
<PRINTERTYPE>::=<N>
<VERVLECHTING>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
<...>::=<ZIE ALGEMENE SPECIFICATIES>
```

betekenis

Met het SCREEN-kommando kunnen diverse instellingen van de computer worden bepaald, voornamelijk in verband met het beeldscherm.

Alle numerieke uitdrukkingen die binnen het SCREEN-kommando worden gebruikt worden eventueel eerst ontdaan van een decimale fraktie.

Achter het SCREEN-kommando kunnen we als eerste gegeven de scherminstelling opnemen. Indien we de scherminstelling overslaan, blijft de laatste scherminstelling actief. In het andere geval wordt na de vernieuwde scherminstelling het beeld ook schoongemaakt.

De waarde van de scherminstelling kan gelijk zijn aan:

0: een alfanumeriek scherm van maximaal 24 regels van 40 tekens (MSX-1) of maximaal 24 regels van 80 tekens (MSX-2) wordt in-

gesteld. Onder deze scherminstelling kunnen geen sprites worden aangestuurd. Maximaal kunnen 2 uit 16 kleuren tegelijk worden geactiveerd; een achtergrondkleur en een voorgrondkleur. Deze scherminstelling kan op elke MSX-computer worden ingesteld. De MSX-1 computer staat bij aanzetten automatisch in SCREEN 0 onder MSX-2 bij regelbreedte 41 of hoger kunnen er op gebrekkige wijze 4 kleuren worden aangstuurd. Zie hoofdstuk 14.

1: een alfanumeriek scherm van maximaal 24 regels van 32 tekens wordt ingesteld. Onder deze scherminstelling kunnen enkelvoudig gekleurde sprites worden ingesteld. Van deze sprites kunnen er zich maximaal vier tegelijk zichtbaar op één horizontale lijn bevinden.
Maximaal kunnen er buiten de sprites 2 uit 16 kleuren worden aangestuurd; een voorgrondkleur en een achtergrondkleur. Met enige inspanning kunnen met behulp van hoofdstuk 14 op een gebrekkinge wijze alle 16 kleuren aansturen. Deze scherminstelling kan op elke MSX-computer worden ingesteld.

2: een grafisch scherm van 192 puntjes vertikaal en 256 puntjes horizontale lijn wordt ingesteld. Dezelfde sprites als onder SCREEN 1 kunnen worden gebruikt. Er kunnen maximaal 16 kleuren tegelijk worden gebruikt met deze beperking dat een groepje van acht naast elkaar liggende beeldschermpunten slechts één voorgrondkleur en één achtergrondkleur kan worden ingesteld. Alleen via een omweg (zie OPEN) kunnen karakters op het beeldscherm worden afgedrukt. Deze scherminstelling werkt slechts wanneer er minimaal 16 kilobytes aan video-geheugen is aangesloten.

3: een grafisch scherm van 48 puntjes vertikaal en 64 puntjes horizontaal wordt ingesteld. Elk puntje kan op dit grafische scherm in 16 verschillende kleuren worden uitgevoerd. Dezelfde sprites als onder SCREEN 1 kunnen worden gebruikt. Alleen via een omweg (zie OPEN) kan tekst op het beeldscherm worden afgedrukt. Deze scherminstelling werkt op elke MSX-computer.

De volgende scherminstellingen zijn alleen met een MSX-2-computer te activeren:

4: als SCREEN 2 met dit verschil dat er nu sprites met meerdere kleuren kunnen worden aangestuurd. Van deze sprites kunnen er acht tegelijk op één horizontale lijn staan.

5: een grafisch scherm van 212 puntjes hoog en 256 puntjes breed wordt ingesteld. Elk puntje apart kan in 16 verschillende kleuren worden uitgevoerd. Dezelfde sprites als onder SCREEN 4 kunnen worden gebruikt. Alleen via een omweg (zie OPEN) kan tekst op het beeldscherm worden afgedrukt. Deze instelling werkt op elke MSX-2 computer. In het video-geheugen kunnen twee (64 KB) of vier (128 KB) verschillende schermen onafhankelijk van elkaar worden opgeslagen.

6: een grafisch scherm van 212 puntjes hoog en 512 puntjes breed wordt ingesteld. Elk puntje apart kan in 4 verschillende kleuren worden uitgevoerd. Dezelfde sprite als onder SCREEN 4 kunnen worden gebruikt. Alleen via een omweg (zie OPEN) kan tekst op het beeldscherm worden afgedrukt. Deze instelling werkt op elke MSX-2 computer. In het video-geheugen kunnen twee (64 KB) of vier 128 KB) verschillende schermen onafhankelijk van elkaar worden opgeslagen.

7: een grafisch scherm van 212 puntjes hoog en 512 puntjes breed wordt ingesteld. Elk puntje apart kan in 16 verschillende kleuren worden uitgevoerd. Dezelfde sprite als onder SCREEN 4 kunnen worden gebruikt. Alleen via een omweg (zie OPEN) kan tekst op het beeldscherm worden afgedrukt. Deze instelling werkt allleen maar op MSX-2 computers met 128 kilobytes video-geheugen. In het video-geheugen kunnen twee verschillende schermen onafhankelijk van elkaar worden opgeslagen.

8: een grafisch scherm van 212 puntjes hoog en 156 puntjes breed wordt ingesteld. Elk puntje apart kan in 256 verschillende kleuren worden uitgevoerd. Dezelfde sprites als onder SCREEN 4 kunnen worden gebruikt. Allen via een omweg (zie OPEN) kan tekst op het beeldscherm worden afgedrukt. Deze instelling werkt alleen maar op MSX-2 computers met 128 kilobytes video-geheugen. In het video-geheugen kunnen twee verschillende schermen onafhankelijk van elkaar worden opgeslagen.

Zie voor de betekenis van de mogelijke kleurkoderingen onder de verschillende scherminstellingen de behandeling van COLOR. Zie voor het gebruik van meerdere beeldschermen per instelling SET PAGE (MSX-2).

Het tweede gegeven achter SCREEN geeft aan, welke sprite-grootte eventueel wordt gehanteerd. Indien dit gegeven wordt overgeslagen,

blijft de laatst ingestelde sprite-grootte actief.
De waarde van de sprite-grootte kan zijn:

0: de sprite-grootte is 8 bij 8 beeldschermpunten. Bij het aanzetten van de computer wordt automatisch deze grootte ingesteld.

1: de sprite-grootte is 8 bij 8 beeldschermpunten. De sprites worden echter vergroot weergegeven. Zowel in de breedte als in de hoogte worden de sprites twee maal vergroot.

2: de sprite-grootte is 16 bij 16 beeldschermpunten, onvergroot.

3: de sprite-grootte is 16 bij 16 beeldschermpunten. De sprites worden echter vergroot weergegeven. Zowel in de breedte als in de hoogte worden de sprites twee maal vergroot.

Zie voor de betekenis van sprites SPRITE$ en PUT SPRITE.

Het derde gegeven achter SCREEN geeft aan of elke toetsaanslag al dan niet door een klikgeluid wordt bevestigd. Indien dit gegeven de waarde 0 heeft, verdwijnt het klikgeluid. Indien deze waarde ongelijk is aan 0, wordt de klik weer geactiveerd. Deze aanduiding mag nooit groter dan 255 worden opgegeven. Indien dit gegeven wordt overgeslagen, blijft de laatste instelling van kracht. Wanneer de computer wordt aangezet, wordt het klikgeluid standaard geactiveerd.

Het vierde gegeven achter SCREEN geeft aan, op welke snelheid er gegevensoverdracht met de cassetterecorder plaats vindt. Indien dit gegeven wordt overgeslagen wordt de laagste instelling gehandhaafd.

De waarde van de cassette-schrijfsnelheid heeft de volgende betekenis:

1: het schrijven van een cassetteband gaat met een snelheid van 1200 baud hetgeen ongeveer overeenkomt met 150 tekens per seconde.

2: het schrijven van een cassetteband gaat nu met een snelheid van 2400 baud hetgeen overeenkomt met ongeveer 300 tekens per seconde.

Wanneer de computer wordt ingeschakeld, wordt standaard en de snelheid van 1200 baud ingesteld.

Het is slechts raadzaam om de snelheid op 2400 baud in te stellen wanneer men zeker weet dat de cassetterecorder in prima conditie en

de kwaliteit van het bandmateriaal onberispelijk is. In een ander geval is de kans op het optreden van fouten tijdens het schrijven of tijdens het later weer inlezen van de gegevens zeer groot!

Het vijfde gegeven achter SCREEN geeft aan, welke type printer aan de computer werd aangesloten. Indien dit gegeven wordt overgeslagen, blijft de laatste instelling van kracht.

Het printertype kan gelijk zijn aan:

0: MSX gaat ervan uit dat een standaard MSX-printer werd aangesloten. Deze printer kent alle door de computer te genereren karakters. Bij het inschakelen van de MSX-1 computer gaat deze automatisch uit van deze situatie.

1: MSX gaat ervan uit dat er een MSX-vreemde printer werd aangesloten. Alle specifieke MSX-karakters (zie hoofdstuk 15, alle grafische karakters boven CHR$ (127) of onder CHR$ (32) worden tijdens het afdrukken vervangen door spatietekens.

Onder MSX-2 kan uiteindelijk ook nog een zesde gegeven achter SCREEN worden opgenomen; de vervlechtingskode. Deze kode kan gelijk zijn aan:

0: de beeldscherminformatie wordt op normale wijze geprojekteerd.

1: de beeldscherminformatie wordt vervlochten weergegeven. Dit betekent dat de door uw televisietoestel of monitor geprojekteerde plaatjes die 50 maal per seconde steeds opnieuw worden geprojekteerd, steeds iets verschoven ten opzichte van elkaar worden geprojekteerd. Als zodanig worden de beeldlijnen niet over elkaar maar steeds iets naast elkaar geprojekteerd, als het ware in elkaar gevlochten.

2: indien met SET PAGE een oneven beeldscherm werd geactiveerd, worden dit beeldscherm en het beeldscherm hiervoor, het oneven en het even beeldscherm, afwisselend geprojekteerd zodat twee beelden door elkaar ontstaan.

3: als kode 2. Echter, in dit geval worden de beeldlijnen van het even en het oneven beeldscherm wederom vervlochten.

De vervlechtingskode laat zich slechts goed gebruiken indien een scherm wordt gebruikt met een lang nagloeiende beeldbuis. In het andere geval ontstaat een onrustig flikkerend beeld.

Toegepast op een daarvoor geschikte beeldbuis resulteert de vervlechting in een rustig en fijn gedetailleerd beeld. Bovendien is er de mogelijkheid om twee verschillende beeldschermen door elkaar heen te printen.

De MSX-2 kan worden uitgebreid met een video digitizer. Met dit apparaat kunnen video- of camerabeelden direkt worden gedigitaliseerd en in het video-geheugen worden vastgelegd. Tijdens deze digitalisatie kunnen twee aparte beelden in twee verschillende beeldschermen worden opgeslagen. Door elkaar geprojekteerd met kode 3 is het resultaat een beeldscherm van 424 bij 512 beeldschermpuntjes hetgeen een zeer fijn gedetailleerd beeld tot resultaat heeft.

Enige voorbeelden:

`SCREEN 1`	scherminstelling 1 wordt geactiveerd.
`SCREEN 1,2`	scherminstelling 1 wordt geactiveerd terwijl de sprite-grootte op 16 bij 16 beeldpunten onvergroot wordt bepaald.
`SCREEN ,,,2`	alleen de cassette-schrijfsnelheid wordt op 2400 baud gezet.
`SCREEN 0,,0,,1`	scherminstelling 0 wordt geactiveerd terwijl de toetsklik wordt uitgeschakeld en het printertype op MSX-vreemd wordt gesteld.

In het volgende programma wordt SCREEN 2 geactiveerd waarna enige cirkels worden getekend. Merk op dat bij het einde van het programma een terugkeer naar een alfanumeriek scherm door MSX wordt geforceerd.

```
NEW
Ok
10 SCREEN 2
20 FOR I=10 TO 95 STEP 5
30 CIRCLE (128,98),I
40 NEXT I:STOP
RUN
```

Een MSX-2 voorbeeld:

In het volgende voorbeeld worden twee verschillende beeldschermen ingetekend waarna deze vervlochten en door elkaar worden geprojekteerd.

```
NEW
Ok
10 SCREEN 5,,,,,0:CLS
20 OPEN "GRP:" AS 1
30 PRESET (20,10):PRINT #1,"SCHERM 1"
40 FOR I=0 TO 100 STEP 5:CIRCLE (128,106),I:NEXT I
50 SETPAGE 1,1:CLS
60 PRESET (20,20):PRINT #1,"SCHERM 2":CLOSE
70 FOR I=100 TO 0 STEP -3:LINE (128-I,106-I)-(128+I,106+I),,
B:NEXT I
80 SCREEN ,,,,,2
90 GOTO 90
RUN
```

Wanneer tijdens een grafische scherminstelling het programma op een einde komt of wanneer er een (LINE) INPUT wordt gepleegd vanaf het toetsenbord, dan wordt het laatst gebruikte alfanumerieke scherm (SCREEN 0 of SCREEN 1) geforceerd geactiveerd waarbij het grafische beeldscherm verloren gaat.

Een PRINT-kommando naar het beeldscherm, anders dan via het geopende grafische scherm (GRP:, zie OPEN), wordt genegeerd indien een grafisch scherminstelling actief is.

Met SET SCREEN kunnen bepaalde instellingen onder MSX-2 worden vastgelegd in een stukje geheugen dat met behulp van een accu onder spanning gehouden wordt. Deze accu wordt tijdens het gebruik van de computer automatisch weer opgeladen. Wanneer de machine later weer wordt aangezet, dan zijn deze instellingen onmiddellijk van kracht en behoeven zij niet met behulp van bijvoorbeeld SCREEN nogmaals te worden ingesteld. Zie SET SCREEN voor nadere gegevens.

moeilijkheidsgraad . . . . . eenvoudig
soort . . . . . KOMMANDO
afkomst . . . . . SET ADJUST – pas aan

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
SET ADJUST(<HORIZONTAAL>,<VERTIKAAL>)

<HORIZONTAAL>::=<N>

<VERTIKAAL>::=<N>

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

betekenis

Voor een goed begrip dienen eerst de kommando's WIDTH en SCREEN te worden doorgenomen.

De afstemming tussen de computer en het beeldscherm is vaak niet helemaal naar wens. Met name komt het veelvuldig voor dat bij een maximale scherminstelling (bijvoorbeeld WIDTH 80 onder SCREEN 0 instelling) de linker karakters van elke regel net achter de beeldrand verdwijnen en zo onzichtbaar worden.

Met SET ADJUST kan men de MSX-computer afstemmen op het gebruikte beeldscherm of televisietoestel. Hiertoe dient achter SET ADJUST een horizontale en vertikale verschuiving te worden opgenomen.

De horizontale verschuiving dient minimaal gelijk te zijn aan -7 en mag maximaal gelijk zijn aan 8. Een eventuele decimale fractie wordt verwaarloosd. Deze waarde geeft aan, hoeveel eenheden het totale beeldscherm ten opzichte van de standaardinstelling verder naar links (negatief) op naar rechts (positief) dient te worden geprojekteerd.

De vertikale verschuiving dient minimaal gelijk te zijn aan -7 en mag maximaal gelijk zijn aan 8. Een eventuele decimale fractie wordt verwaarloosd. Deze waarde geeft aan, hoeveel eenheden het totale beeldscherm ten opzichte van de standaardinstelling verder naar boven (nega-

tief) of verder naar beneden (positief) dient te worden geprojekteerd.

De aan SET ADJUST opgegeven waarde worden in een speciaal stukje geheugen geplaatst. Dit geheugen wordt door de in de MSX-computer ingebouwde accu continu onder spanning gehouden. De accu laadt zich op wanneer de computer gebruikt wordt.

Een eenmaal ingestelde SET ADJUST blijft van kracht, ook nadat de machine uitgeschakeld geweest is.

In het volgende voorbeeld kunt u met behulp van de pijltoetsen het beeldscherm op de juiste wijze instellen.

```
NEW
Ok
10 KEY OFF:COLOR 15,4,4:SCREEN 0:WIDTH 80
20 FOR I=BASE(0) TO BASE(0)+79:VPOKE I,219:VPOKE I+1840,219:
NEXT I
30 FOR I=BASE(0) TO BASE(0)+1919 STEP 80:VPOKE I,219:VPOKE I
+79,219:NEXT I
40 LOCATE 20,10,0:PRINT "ZET HET KADER HELEMAAL OP HET SCHER
M"
50 LOCATE 22,11:PRINT "GEBRUIK HIERVOOR DE PIJLTOETSEN"
60 LOCATE 31,12:PRINT "RETURN=EINDE"
70 X=0:Y=0
80 SET ADJUST (X,Y)
90 I$=INKEY$:IF I$="" THEN 90
100 IF I$=CHR$(13) THEN CLS:LOCATE 32,11:PRINT "INSTELLING G
EREED":STOP
110 I=ASC(I$):IF I<28 OR I>31 THEN 90
120 X=X-(I=29)*(X>-7)+(I=28)*(X<8)
130 Y=Y-(I=30)*(Y>-7)+(I=31)*(Y<8)
140 GOTO 80
RUN
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET BEEP – stel pieptoon in**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
SET BEEP[<TOON>][,<VOLUME>]\SET BEEP

<TOON>::=<N>

<VOLUME>::=<N>

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

betekenis

Met het kommando BEEP kunt u een attentiesignaal laten horen. Dit attentiesignaal laat de computer u automatisch horen indien een foutmelding wordt gegeven of een STOP-kommando werd aangetroffen.

Dit attentiesignaal is qua samenstelling en volume naar eigen wens in te stellen.

Als eerste gegeven achter SET BEEP dient een waarde te worden opgegeven die de samenstelling bepaalt van het attentiesignaal. Deze waarde dient minimaal gelijk te zijn aan 1 en mag maximaal gelijk zijn aan 4. Een eventuele decimale fractie wordt verwaarloosd.

Betekenis van deze waarde:

1 de normale, ééntonige pieptoon

2 een lagere, nagalmende toon

3 een tweetonige, nagalmende toon

4 een snel drietonig signaal

Als tweede gegeven achter SET BEEP dient het gewenste volume te worden opgegeven. Deze waarde dient minimaal gelijk te zijn aan 1 en mag maximaal gelijk zijn aan 4. Een eventuele decimale fractie wordt

verwaarloosd. Een waarde 1 correspondeert met het minimaal instelbare volume terwijl een waarde 4 een maximaal volume tot gevolg heeft.

De aan SET BEEP opgegeven waarde wordt in een speciaal stukje geheugen geplaatst. Dit geheugen wordt door de in de MSX-computer ingebouwde accu continu onder spanning gehouden. De accu laadt zich op wanneer de computer in gebruik is.

Een eenmaal ingestelde SET BEEP blijft van kracht, ook als de machine uitgeschakeld geweest is.

In het volgende voorbeeld kunt u zelf het beste attentiesignaal uitzoeken en vastleggen.

```
NEW
Ok
10 CLS:PRINT "0=EINDE"
20 INPUT "SOORT ATTENTIESIGNAAL (1,2,3 OF 4)";S:IF S=0 THEN
STOP
30 IF S<>INT(S) OR S<1 OR S>4 THEN 20
40 INPUT "VOLUME VAN DIT SIGNAAL (1,2,3 OF 4)";V
50 IF V<>INT(V) OR V<1 OR V>4 THEN 40
60 SET BEEP S,V:BEEP:GOTO 10
RUN
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET DATE – stel de datum in**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
SET DATE <DATUM>[,A]

<DATUM>::=<A>

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

De MSX-2 computer heeft een ingebouwde clock-chip. Deze chip heeft een eigen stukje geheugen en wordt door een ingebouwde accu continu onder spanning gehouden. Deze accu laadt zich tijdens het gebruik van de computer weer op.

De clock-chip loopt ook door wanneer de computer uitgeschakeld is. Als zodanig heeft de MSX-2 computer altijd de juiste datum en tijd tot zijn beschikking.

Met het kommando GET DATE kan men de huidige datum ophalen vanuit de clock-chip; zie aldaar. Met het SET DATE-kommando kan men de datum instellen.

Daar de clock-chip een vrij nauwkeurig instrumentje is, is het slechts bij uitzondering noodzakelijk om de datum bij te stellen.

Achter SET DATE dient men de in te stellen datum op te nemen. Deze datum dient in de vorm „DD/MM/JJ" te worden opgegeven waarbij DD staat voor het dagnummer van de maand, MM staat voor het maandnummer van het jaar en JJ staat voor het jaarnummer binnen de twintigste eeuw.

DD dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 31.
MM dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 12.
JJ dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 99.

DD, MM en JJ dienen altijd uit twee posities te bestaan waarbij de eerste positie eventueel een voorloop-nul mag zijn.

Als tweede gegeven achter SET TIME kan men een komma, gevolgd door de letter A opnemen. In dat geval wordt de opgegeven datum als alarm-datum opgenomen. Ook deze datum kan weer met GET DATE worden opgehaald.

Wanneer een alarm-datum wordt ingesteld, is het maandgegeven en het jaargegeven niet belangrijk. Het moet wel worden opgegeven maar wordt altijd op 00 gesteld.

Enkele voorbeelden:

```
SET DATE "26/01/86"
Ok
GET DATE A$:PRINT A$
26/01/86
Ok
SET DATE "27/01/86",A
Ok
GET DATE A$,A:PRINT A$
27/00/00
Ok
```

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET PAGE – stel beeldschermpagina in**

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
SET PAGE [<PROJEKTIESCHERM>][,<ACTIESCHERM>]\SET PAGE

<[PROJEKTIESCHERM>::=<N>

<ACTIESCHERM>::=<N>

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

betekenis

De MSX-2 computer heeft een erg groot video-geheugen; zelfs zo groot dat er gegegvens van meerdere beeldschermen tegelijk in kunnen worden bewaard.

Onder SCREEN 5 tot en met SCREEN 8 kunnen afhankelijk van het beschikbare video-geheugen, het volgende aantal schermen worden opgeslagen:

SCREEN	64 KB VRAM	128 KB VRAM
5	2	4
6	2	4
7	–	2
8	–	2

Het eerste achter SET PAGE op te nemen gegeven geeft aan, welk van de beschikbare beeldschermen dient te worden geprojekteerd.

Het tweede achter SET PAGE op te nemen gegeven geeft aan, op welk van de beschikbare beeldschermen eventuele grafische kommando's dienen te worden uitgevoerd.

Het is dus mogelijk dat het ene beeldscherm wordt geprojekteerd terwijl er op het andere beeldscherm wordt getekend!

De twee achter SET PAGE op te nemen gegevens dienen minimaal gelijk te zijn aan 0 en mogen maximaal gelijk zijn aan het aantal beschikbare beeldschermen -1. Een eventuele decimale fractie wordt verwaarloosd.

Het volgende voorbeeld laat u een serie tekeningen zien. Doordat er steeds op het niet zichtbare scherm wordt getekend, ziet u alleen maar de eindresultaten stuk voor stuk voorbijkomen.

```
NEW
Ok
10 Q=0*RND(-TIME)
20 SCREEN 5:COLOR 15,4,4
30 SET PAGE Q,-(Q=0):Q=-(Q=0):CLS
40 A=RND(1)*162+50
50 FOR I=0 TO 255 STEP A/10
60 LINE (I,A*SIN(I/A))-(A*COS(I/A),212-I)
70 NEXT I
80 GOTO 30
RUN
```

Wanneer een SCREEN 5 . . . t/m SCREEN 8 . . . wordt uitgevoerd, dan wordt daarbij automatisch een SET PAGE Ø,Ø en een CLS uitgevoerd.
Wanneer met SET PAGE een ander beeldscherm wordt geactiveerd, kan dat nog een oude inhoud hebben. Deze wordt door het SCREEN-kommando niet automatisch schoongemaakt.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET PASSWORD – stel toegangskode in**

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
SET PASSWORD<TOEGANGSKODE>

<TOEGANGSKODE>::=<A>

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

betekenis

Met het SET PASSWORD-kommando kunt u een toegangskode van maximaal 255 tekens instellen. Nadat u dit gedaan heeft, zal de computer u voortaan direkt na het inschakelen om deze toegangskode vragen. Alleen na het intikken van de juiste kode gaat de computer verder met de opstartcyclus en kunt u met het apparaat werken.

Om de toegangskode in te stellen, dient u na SET PASSWORD de betreffende toegangskode in te toetsen. Deze toegangskode wordt in een speciaal stukje geheugen bewaard dat altijd onder spanning blijft, ook wanneer de computer wordt uitgeschakeld. Hiertoe heeft elke MSX-2 computer een accu die zich oplaadt wanneer de machine aan staat.

Wanneer u de toegangskode vergeten bent, kunt u dit kodewoord op een speciale wijze omzeilen. Houdt u in dat geval tijdens het opstarten de STOP-toets en de GRAPH-toets beide ingedrukt. De computer zal niet om een „password" vragen maar direkt opstarten.

Een voorbeeld: u kunt een toegangskode "PIETER" instellen met behulp van het kommando:

```
SET PASSWORD "PIETER"
Ok
```

Het verwijderen van dit kodewoord is niet mogelijk met behulp van SET PASSWORD. In plaats daarvan dient u het SET PROMPT of het SET TITLE kommando uit te voeren. Omdat deze drie kommando's gebruik maken van hetzelfde stukje permanent geheugen, resulteert

het gebruik van één van deze bevelen altijd in het teniet doen van het effekt van de andere twee. Zo kunt u via een omweg een toegangskode verwijderen door bijvoorbeeld het volgende in te toetsen:

```
SET PROMPT "Ok"
Ok
```

moeilijkheidsgraad: . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET PROMPT – stel aanwijzing in**

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
SET PROMPT<GEREEDMELDING>

<GEREEDMELDING>::=<A>

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

betekenis

Wanneer de MSX-computer klaar is met een opdracht, besluit deze altijd met de Ok-melding. In bijna alle in dit boek opgenomen voorbeelden ziet u deze melding wel een keer verschijnen.

Deze melding kan met behulp van het SET PROMPT-kommando worden veranderd. Een op deze wijze veranderde melding wordt in een speciaal stukje geheugen van de computer opgeslagen. Dit stukje geheugen word continu door een accu onder spanning gehouden. Deze accu laadt zich op wanneer de computer ingeschakeld is. Een eenmaal ingestelde melding blijft als zodanig van kracht, ook wanneer de machine uitgeschakeld is geweest.

De veranderde melding mag maximaal 6 karakters lang zijn. Te veel opgegeven karakters worden verwaarloosd.

Omdat de kommando's SET PASSWORD, SET PROMPT en SET TITLE van hetzelfde stukje permanent geheugen gebruik maken, resulteert het gebruik van één van deze drie kommando's altijd in het teniet doen van het effekt van de andere twee.

Een voorbeeld:

```
STOP
Break
Ok
SET PROMPT "GOEDZO!"
GOEDZO
```

```
NEW
GOEDZO
SET PASSWORD "GEHEIM"
Ok
```

In dit voorbeeld ziet u ondermeer dat SET PASSWORD het effekt van SET PROMPT teniet doet.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET SCREEN – stel scherm (gegevens) in**

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
SET SCREEN
```

betekenis

Nadat het MSX-systeem is aangezet, kunnen we onder meer met SCREEN de juiste scherminstelling bepalen. Met COLOR kunnen we daarna voor- en achtergrondkleur alsmede de randkleur instellen. Daarna kunnen we wanneer we dit willen, de onderste regel met KEY OFF vrijmaken. Zo kunnen we uiteindelijk de MSX-computer precies naar onze wens aanpassen. Lastig is het echter, om dit elke keer na het inschakelen van de computer opnieuw te moeten doen.

Met SET SCREEN kunnen we diverse voor de hand liggende instellingen vastleggen in een speciaal stukje geheugen dat door een accu voortdurend onder spanning wordt gehouden, ook wanneer de computer uitgeschakeld is. Deze accu laadt zich op wanneer er met de computer gewerkt wordt.

Door een voudigweg SET SCREEN in te toetsen, leggen we vast:

–de scherminstelling. Wanneer SCREEN 0 of SCREEN 1 actief is, wordt dit vastgelegd. Een volgende keer wordt er onder deze scherminstelling opgestart.

–de breedte-instelling. De met WIDTH ingestelde regelbreedte wordt vastgelegd. Een volgende keer wordt deze regelbreedte automatisch weer geactiveerd.

–voorgrondkleur, achtergrondkleur en randkleur. De met COLOR ingestelde kleuren worden vastgelegd. Deze kleurinstelling wordt een volgende keer weer automatisch geactiveerd.

–de projektie van de funktietoetsen. De funktietoetsen kunnen

met behulp van KEY ON en KEY OFF al dan niet op de onderste beeldschermregel worden geprojekteerd. De laatste instelling wordt met SET SCREEN vastgelegd.

–de klikkode. Met het SCREEN-kommando kunnen we ondermeer bepalen of elke toetsaanslag al dan niet door een klikgeluid dient te worden bevestigd. Ook deze instelling wordt vastgehouden.

–de printerkode. De via SCREEN ingestelde printerkode (MSX-printer of vreemde printer) wordt vastgelegd.

–de cassette schrijfsnelheid. Dit eveneens met behulp van SCREEN bepaalde gegeven wordt vastgelegd.

–de vervlechtingskode. Ook deze kode wordt met behulp van SCREEN bepaald. Deze kode wordt eveneens vastgelegd.

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET TIME – stel tijd in**

Dit kommando is alleen onder MSX-2 beschikbaar

schrijfwijze

```
SET TIME <TIJD>[,A]

<TIJD>:::=<A>

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

betekenis

De MSX-computer heeft een ingebouwde clock-chip. Deze chip heeft een eigen stukje geheugen en wordt door een accu continu onder spanning gehouden. Deze accu laadt zich tijdens het werken met de computer weer op.

De clock-chip loopt ook door wanneer de computer is uitgeschakeld. Als zodanig heeft de MSX-2 computer altijd de juiste tijd tot zijn beschikking.

Met het kommando GET TIME kan men de huidige tijd ophalen vanuit de clock-chip; zie aldaar. Met het SET TIME-kommando kan met de tijd instellen.

Daar de clock-chip een vrij nauwkeurig instrumentje is, is het slechts bij uitzondering noodzakelijk om de tijd bij te stellen.

Achter SET TIME dient men de in te stellen tijd op te nemen. Deze tijd dient in de vorm „UU:MM:SS" te worden opgenomen waarbij UU staat voor de uren, MM voor de minuten en SS voor de seconden.

UU dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 23.
MM dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 59.
SS dient minimaal gelijk te zijn aan 00 en mag maximaal gelijk zijn aan 59.

UU, MM en SS dienen altijd uit twee posities te bestaan waarbij de eerste positie eventueel een voorloop-nul mag zijn.

Als tweede gegeven achter SET TIME kan men een komma, gevolgd door de letter A opnemen. In dat geval wordt de opgegeven tijd als alarmtijd opgenomen. Ook deze tijd kan met GET TIME weer worden opgehaald.

Wanneer een alarmtijd wordt ingesteld, is het seconden-gegeven niet belangrijk. Het moet wel worden opgegeven maar wordt altijd op 00 gesteld.

Enkele voorbeelden:

```
SET TIME "03:25:15"
Ok
GET TIME A$:PRINT A$
03:25:24
Ok
SET TIME "04:10:30",A
Ok
GET TIME A$,A:PRINT A$
04:10:00
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SET TITLE – stel titel in**

Dit kommando is alleen onder MSX-2 beschikbaar.

schrijfwijze

```
SET TITLE [<OPSTARTMELDING>][,<KLEUR>]\SET TITLE

<OPSTARTMELDING>::=<A>

<KLEUR>::=<N>

<A>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Op het opstartscherm kunnen we met behulp van SET TITLE een vermelding van maximaal 6 tekens laten verschijnen. Hiertoe dienen we achter SET TITLE de betreffende vermelding op te nemen. Een teveel aan karakters wordt hierbij verwaarloosd.

Wanneer u na het SET TITLE-kommando de computer opnieuw opstart, dan verschijnt de vermelding onder de MSX-aanduiding op het scherm. Wanneer de vermelding precies 6 tekens lang is, blijft de computer wachten totdat een toets wordt ingegeven.

De vermelding wordt in een speciaal stukje geheugen opgeslagen. Dit stukje geheugen wordt door een accu continu onder spanning gehouden waardoor de vermelding niet kan worden "vergeten". De accu wordt opgeladen wanneer de computer ingeschakeld is.

Omdat de kommando's SET TITLE, SET PASSWORD en SET PROMPT van hetzelfde stukje permanent geheugen gebruik maken, resulteert het gebruik van één van deze drie kommando's altijd in het teniet doen van het effekt van de andere twee.

Een voorbeeld:

```
SET TITLE "HALLO"
Ok                                  (Start nu de computer opnieuw op)
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **SGN is afkorting van sign – teken**

schrijfwijze

```
SGN(<N>)

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

betekenis

Deze funktie kan drie resultaten geven:

de waarde 1	wanneer de waarde van de tussen haakjes vermelde numerieke uitdrukking positief (groter dan nul) is
de waarde 0	wanneer de waarde van de tussen haakjes vermelde numerieke uitdrukking gelijk is aan nul
de waarde –1	wanneer de waarde van de tussen haakjes vermelde numerieke uitdrukking negatief (kleiner dan nul) is.

Voorbeeld:

```
NEW
Ok
10 LET A=-123.4:LET B=0:LET C=1E+33
20 PRINT SGN(A);SGN(B);SGN(C)
RUN
-1  0  1
Ok
```

moeilijkheidsgraad . . **vrij moeilijk, kennis van goniometrie is noodzakelijk**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **SIN is afkorting van sine – sinus**

schrijfwijze

```
SIN(<HOEK>)

<HOEK>::=<N>

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

betekenis

Deze funktie geeft als resultaat de sinus van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen. Het resultaat ligt uiteraard altijd tussen –1 en 1.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

Voorbeeld:

```
NEW
Ok
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926536#
30 PRINT I;":";SIN(HOEK)
40 NEXT I
RUN
```

(Een sinustabel verschijnt op het beeldscherm. Merk op dat op regel 20 de hoek van graden naar radialen wordt geconverteerd.)

moeilijkheidsgraad ..... **moeilijk, kennis van de betekenis van de register van de geluidscontrole chip is vereist**
soort ..... **KOMMANDO**
afkomst ..... **SOUND is geluid**

schrijfwijze

```
SOUND<REGISTERNUMMER>,<REGISTERINHOUD>

<REGISTERNUMMER>::=<N>

<REGISTERINHOUD>::=<N>

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

betekenis

Met het SOUND-kommando kunnen de registers van de PSG (Programmable Sound Generator) worden ingevuld. Hiertoe dient achter het SOUND sleutelwoord eerst het registernummer te worden gespecificeerd, gevolgd door een komma en de waarde die het betreffende register dient aan te nemen.

Het registernummer mag niet kleiner zijn dan 0 en niet groter dan 13. Een eventuele decimale fractie wordt verwaarloosd.
De registerinhoud mag niet kleiner zijn dan 0 en niet groter dan 255. Ook hier wordt een eventuele decimale fractie verwaarloosd.

Door de registers van de PSG te besturen, kunnen diverse effecten worden bereikt. Bijvoorbeeld de ruisgenerator dient op deze wijze te worden bestuurd. Een voorbeeld:

```
NEW
Ok
10 SOUND 6,&B1101
20 SOUND 7,&B110111
30 SOUND 8,&B1111
RUN
Ok
```

Een sterke ruis is hoorbaar. Deze kan met CONTROL-STOP worden afgezet.

In het bovenstaande voorbeeld werden registers 6, 7 en 8 van de PSG van bepaalden waarden voorzien, in dit geval opgegeven als binaire konstanten.

Zie voor de betekenis van de betreffende registers hoofdstuk 13.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **SPACE$ is afkorting van space string – spatie string**

schrijfwijze

```
SPACE$(<AANTAL SPATIES>)

<AANTAL SPATIES>::=<N>

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

betekenis

Deze funktie geeft als resultaat een aantal spaties. Het aantal wordt bepaald door de numerieke uitdrukking, opgenomen tussen de haakjes. De waarde van deze uitdrukking moet minimaal gelijk zijn aan 0 en mag maximaal gelijk zijn aan 255 terwijl een eventuele decimale fractie wordt verwaarloosd.

Voorbeeld:

```
NEW
Ok
10 FOR I=0 TO 9
20 PRINT SPACE$(I);"MSX"
30 NEXT I:STOP
RUN
MSX
 MSX
  MSX
   MSX
    MSX
     MSX
      MSX
       MSX
        MSX
         MSX
Break in 30
Ok
```

moeilijkheidsgraad . . . . . **moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SPRITE is 'geest'**

schrijfwijze

```
        ON
        ----
SPRITE  OFF
        ----
        STOP
```

betekenis

Voorafgaand aan de behandeling van dit kommando dienen de ON KEY GOSUB en de PUT SPRITE grondig te zijn doorgenomen.

MSX-basic biedt de mogelijkheid door een botsing van sprites het programma tijdelijk onderbreken. Met SPRITE ON geven we aan dat het programma bij een botsing van sprites tijdelijk dient te worden onderbroken. Met SPRITE OFF geven we aan dat botsingen van sprites niet meer behoeven te leiden tot een tijdelijke onderbreking van het programma. Met SPRITE STOP geven we aan dat een botsing van sprites wel door de computer dient te worden 'onthouden' maar dat het programma niet direkt mag worden onderbroken. Daadwerkelijke onderbreking vindt dan pas plaats bij een SPRITE ON kommando.

Voor een zinvol voorbeeld: zie het ON SPRITE GOSUB kommando.

moeilijkheidsgraad . . . . . **moeilijk**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **SPRITE is een afkorting van sprite string. Sprite is 'geest'**

schrijfwijze

```
SPRITE$(<SPRITENUMMER>)

<SPRITENUMMER>::=<N>

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

betekenis

Het is noodzakelijk dat het SCREEN-kommando reeds grondig is bestudeerd.

Het MSX-basic voorziet in vele kommando's die het schrijven van spelprogramma's in BASIC bijzonder goed ondersteunen. Gebruik van de sleutelwoorden SPRITE, SPRITE$, PUT SPRITE en ON SPRITE GOSUB maken het ondermeer mogelijk om:

- 32 verschillende vaste figuren (sprites) onafhankelijk van elkaar achter elkaar langs op het beeldscherm zich te laten verplaatsen;
- deze sprites of spelfiguren zelf te ontwerpen;
- een botsing van twee of meer sprites te detecteren en hierdoor (zoals bijvoorbeeld ook bij ON KEY GOSUB) het programma tijdelijk te laten onderbreken.

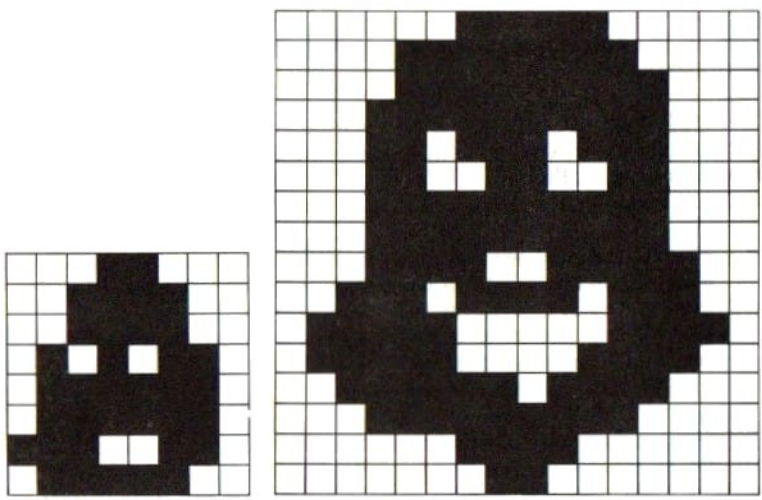
Spelprogramma's met bewegende beelden (packman-achtige spelletjes, aanvals- en schietspelletjes en dergelijke) kunnen in MSX-basic voor het eerst worden geprogrammeerd zonder dat kennis van machinetaal of assembler nodig is en met behoud van de benodigde snelheid. Daarbij zijn uiteraard ook wat serieuzere toepassingen bijzonder goed denkbaar.

Met SPRITE$ kunnen maximaal 64 (indien de sprite-grootte in het SCREEN-kommando op 2 of 3 is gezet) of 256 (indien de sprite-grootte in het SCREEN-kommando op 0 of 1 is gezet) spelfiguren onafhankelijk van elkaar worden ontworpen. Het ontwerpen van een sprite kan als volgt geschieden:

stap 1 beslis welke sprite-grootte gaat worden gebruikt. Is deze 8 bij 8 beeldpunten (sprite-grootte 0 of 1 in het SCREEN-kommando) of is deze 16 bij 16 beeldpunten (spritegrootte 2 of 3 in het SCREEN-kommando)

stap 2 maak een ontwerpblad. Dit kan het beste worden gedaan door een stuk ruitjespapier te nemen en daarop een vierkant van 8 bij 8 ruitjes of 16 bij 16 ruitjes af te bakenen, afhankelijk van de gewenste sprite-grootte.

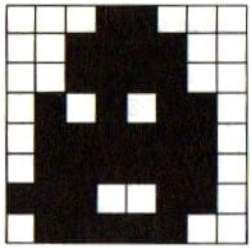
stap 3 ontwerp binnen dit vierkant het gewenste figuur door ruitjes in te kleuren of wit te laten. Een voorbeeld 8 bij 8 en 16 bij 16:



8x8 sprite

16x16 sprite

stap 4 digitaliseer de sprite. Voor een 8 bij 8 sprite is dat het gemakkelijkst. De ingekleurde ruitjes dienen voorgesteld te worden met het cijfer 1; de witte ruitjes met het cijfer 0. Zo krijgen we acht getallen van acht cijfers, allemaal nullen of enen. Voor deze acht cijfers zetten we "&B" waardoor we plotseling acht binaire konstanten hebben verkregen. Bijvoorbeeld:



8x8 sprite

```
&B 0 0 0 1 1 0 0 0
&B 0 0 1 1 1 1 0 0
&B 0 0 1 1 1 1 0 0
&B 0 1 0 1 0 1 1 0
&B 0 1 1 1 1 1 1 0
&B 0 1 1 1 1 1 1 0
&B 1 1 1 0 0 1 1 0
&B 0 1 1 1 1 1 0 0
```

digitalisatie

Voor een 16 bij 16 sprite is dat wat moeilijker. Verdeel hiertoe eerst het 16 bij 16 vierkant in vier 8 bij 8 vierkanten. Deze kunnen op de manier van de 8 bij 8 sprites weer worden gedigitaliseerd (in cijfers omgezet). Doe dit in de volgorde:

1: 8 bij 8 vierkant linksboven
2: 8 bij 8 vierkant linksonder
3: 8 bij 8 vierkant rechtsboven
4: 8 bij 8 vierkant rechtsonder

Op deze wijze worden 32 binaire kontanten verkregen.

stap 5 we kunnen 64 of 256 sprites definieren, afhankelijk van de gekozen sprite-grootte. Beslis welk sprite-nummer de door ons ontworpen sprite krijgt. De numerieke uitdrukking die als sprite-nummer wordt gebruikt, mag in waarde niet kleiner zijn dan nul en niet groter dan 63 of 255, afhankelijk van de gekozen sprite-grootte. Een eventuele decimale fractie wordt verwaarloosd. In ons voorbeeld zullen we sprite nummer 6 gaan definiëren.

stap 6 leg de sprite vast door SPRITE$(...) gelijk te stellen aan CHR$ (eerste binaire konstante)+CHR$(tweede binaire konstante)+ CHR$(derde...) etcetera totdat alle konstanten zijn opgebruikt. Bijvoorbeeld:

```
NEW
Ok
10 SCREEN 2,1:SPRITE$(6)=CHR$(&B00011000)+CHR$(&B00111100)+C
HR$(&B00111100)+CHR$(&B01010110)+CHR$(&B01111110)+CHR$(&B111
00110)+CHR$(&B01111100)
```

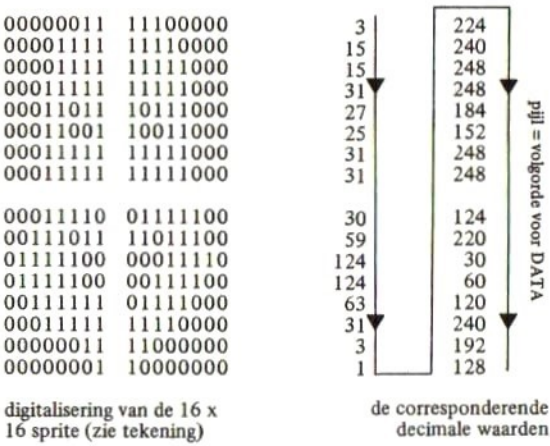
Voor een 16 bij 16 sprite kan dit niet daar de programmaregel dan onvermijdelijk groter wordt dan 255 tekens, hetgeen verboden is in MSX-basic. Om dit te voorkomen, dienen we een methode te gebruiken die kortere programmaregels toestaat. Omdat deze methode veel minder ruimte inneemt dan de eerste methode, is deze ook in tweede instantie voor de 8 bij 8 sprites aan te raden.
Allereerst dienen we alle binaire konstanten te converteren naar decimale konstanten. Dit laten we de computer doen en wel op de volgende manier:
Voor elke binaire konstante tikken we:

```
PRINT &B...          (de juiste binaire konstante invullen)
```

De computer antwoordt onmiddellijk met de decimale waarde van deze konstante die nooit meer dan drie cijfers groot is. Bijvoorbeeld:

```
PRINT &B11111111
 255
Ok
```

Voor elke binaire konstante laten we de computer de decimale waarde uitrekenen die we dan stuk voor stuk opschrijven. Uiteindelijk zetten we deze getallen in één of meer DATA-kommando's onder elkaar. Bijvoorbeeld (zie de eerder ontworpen 16 bij 16 sprite).



Uiteindelijke resultaat: de DATA-regel(s):

```
NEW
Ok
100 DATA 3,15,15,31,27,25,31,31,30,59,124,124,63,31,3,1
110 DATA 224,240,248,248,184,152,248,248,124,220,30,60,120
120 DATA 240,192,128
```

Vervolgens zetten we de sprite eerst in een gewone alfanumerieke variabele op een wijze zoals deze:

```
10 RESTORE 100:A$="":FOR I=1 TO 32:READ A:A$=A$+CHR$(A):NEXT
 I
```

Als laatste actie bepalen we de 16 bij 16 sprite definitief, bijvoorbeeld door het kommando:

```
20 SCREEN 2,2:SPRITE$(6)=A$
```

Indien met SPRITE$ een sprite wordt gedefinieerd, dient eerst de sprite-grootte te zijn bepaald (met een SCREEN). Indien dit niet gebeurt, volgt een foutmelding of worden (bij omschakelen van alleen de sprite-grootte) de sprites weer gewist.

N.B.: Het sleutelwoord LET mag niet voor SPRITE$ worden gebruikt.

N.B.: Bij 16 x 16 sprites kan SPRITE$(64) tot en met SPRITE$(255) wel worden toegewezen. Doordat echter bepaalde tabellen dan worden verminkt, is het resultaat onvoorspelbaar.

Zie het PUT SPRITE kommando voor gebruik van ontworpen sprites.

moeilijkheidsgraad . . normaal, kennis van vierkantswortels is vereist
soort . . . . . N-FUNKTIE
afkomst . . . . . SQR is afkorting van square root – vierkantswortel

schrijfwijze

```
SQR(<N>)

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

betekenis

Deze funktie geeft als resultaat de vierkantswortel van de waarde van de uitdrukking die tussen haakjes staat. Uiteraard mag deze waarde niet negatief zijn.

Voorbeeld:

```
NEW
Ok
10 INPUT "GETAL ";A
20 PRINT "VIERKANTSWORTEL:";SQR(A)
30 GOTO 10
RUN
GETAL ? 3
VIERKANTSWORTEL: 1.7320508075688
GETAL ? 9
VIERKANTSWORTEL: 3
GETAL ? 1024
VIERKANTSWORTEL: 32
GETAL ? 1E20
VIERKANTSWORTEL: 10000000000
GETAL ? -1
VIERKANTSWORTEL:
Illegal function call in 20
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **STICK is stok**

schrijfwijze

```
STICK(<JOY-STICK NUMMER>)

<JOY-STICK NUMMER>::=<N>

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

betekenis

Met deze funktie kan de ingestelde richting van een joy-stick worden opgevraagd.

De numerieke uitdrukking tussen haakjes dient gelijk te zijn aan 0, 1 of 2. Een eventuele decimale fractie wordt verwaarloosd.

De waarde tussen haakjes heeft de volgende betekenis:

`STICK(0)`	de pijltoetsen worden als joy-stick beschouwd
`STICK(1)`	de eerste joy-stick wordt beschouwd
`STICK(2)`	de tweede joy-stick wordt beschouwd

Deze funktie kan diverse waarden als resultaat hebben die allemaal een bepaalde richting aangeven. Deze waarden worden in het volgende schema gesymboliseerd:

Voorbeeld:

```
NEW
Ok
10 A$="1NOORD2NOORD-OOST3OOST4ZUID-OOST5ZUID6ZUID-WEST7WEST8
NOORD-WEST9"
20 A=STICK(0):IF A=0 THEN 20
30 PRINT MID$(LEFT$(A$,INSTR(A$,CHR$(49+A))-1),1+INSTR(A$,CH
R$(48+A)))
40 GOTO 20
RUN
```

(aan de hand van de pijltoetsen vermeldt de computer de aangegeven windrichting. Twee pijlen mogen tegelijk worden ingedrukt om een gecombineerde beweging te verkrijgen. Vul indien u een joy-stick aangesloten heeft, ook eens een andere waarde in de STICK-funktie in.)

moeilijkheidsgraad . . **normaal, in de eerste betekenis zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **STOP is stoppen, stilstaan**

schrijfwijze

```
      ┌ON  ┐
      │----│
STOP  │OFF │
      │----│
      └STOP┘
```

betekenis

eenvoudig gebruik

Met het STOP-kommando kunnen we het programma (tussentijds) beeindigen. Vooral bij het uittesten van een wat groter programma wordt vaak een stop ingelast om tussenresultaten te kunnen bekijken. Voorbeeld:

```
NEW
Ok
10 PRINT "REGEL 1"
20 STOP
30 PRINT "REGEL 2"
RUN
REGEL 1
Break in 20
Ok
```

gevorderd gebruik

Met het STOP-kommando kunnen we de afvanging van de CONTROL-STOP-toetsingave regelen.

`STOP ON` zet de afvanging aan.

`STOP OFF` zet de afvanging uit.

`STOP STOP` zet de afvanging stil; indien een CONTROL-STOP wordt ingedrukt, dan wordt dit door de computer slechts genoteerd. Pas wanneer een STOP ON wordt uitgevoerd, wordt de betreffende aktie

ondernomen.

Dit gebruik van het STOP-kommando is alleen zinvol in samenwerking met de ON STOP GOSUB. Zie voor een zinvol voorbeeld de behandeling van dit kommando.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **STR$ is afkomstig van (convert to) string – zet over naar een stringwaarde**

schrijfwijze

```
STR$(<N>)

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

betekenis

Deze funktie geeft de tegenovergestelde mogelijkheid van de VAL-funktie. Vanuit een numerieke uitdrukking kan een string worden samengesteld. Bijvoorbeeld:

```
NEW
Ok
10 LET A=12.3
20 LET A$=STR$(A)
30 PRINT A$;LEN(A$)
RUN
 12.3 5
Ok
```

Uit dit voorbeeld blijkt dat A$ een voorlopende spatie heeft. STR$ geeft aan positieve waarden altijd een voorloopspatie. Bij negatieve waarden wordt deze spatie door een min-teken vervangen. Bijvoorbeeld:

```
NEW
Ok
10 LET A=-12
20 PRINT LEN(STR$(A));"FL. "+STR$(A)
30 STOP
RUN
 3 FL. -12
Break in 30
Ok
```

moeilijkheidsgraad . . . . . **eenvoudig**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **STRIG is samentrekking van shot en trigger (schot en trekker)**

schrijfwijze

```
STRIG(<VUURKNOPNUMMER>)

<VUURKNOPNUMMER>::=<N>

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

betekenis

Met deze funktie kan worden getest of de vuurknop van een joy-stick al dan niet is ingedrukt.

De numerieke uitdrukking tussen haakjes dient gelijk te zijn aan 0, 1, 2, 3 of 4. Een eventuele decimale fractie wordt verwaarloosd.

De waarde tussen haakjes heeft de volgende betekenis:

`STRIG(0)`	de spatiebalk wordt als vuurknop beschouwd
`STRIG(1) of STRIG(3)`	de eerste of tweede vuurknop van de eerste joy-stick wordt beschouwd
`STRIG(2) of STRIG(4)`	de eerste of tweede vuurknop van de tweede joy-stick wordt beschouwd.

Indien de funktie de waarde 0 tot gevolg heeft, betekent dat dat de vuurknop van de betreffende joy-stick niet wordt ingedrukt. Indien deze funktie echter de waarde –1 geeft, betekent dat dat de vuurknop van de betreffende joy-stick WEL wordt ingedrukt.

Een voorbeeld met de spatiebalk:

```
NEW
Ok
10 IF STRIG(0) THEN PRINT "PANG!!":GOTO 10 ELSE 10
```

```
RUN
PANG!!
PANG!!
```

Telkens wanneer de spatiebalk wordt ingetoetst, verschijnt de tekst PANG. Onderbreek dit programma met CONTROL-STOP.
Indien u joy-sticks heeft aangesloten, verander dan de waarde tussen haakjes eens en probeer de echte vuurknoppen.

N.B.: Veel in de handel zijnde joy-sticks hebben slechts één vuurknop of hebben er twee waarvan er maar één tegelijk te gebruiken is. Meestal is een afvraging van STRIG(2) en STRIG(4) in verband hiermee zinloos.

moeilijkheidsgraad . . . . . **vrij moeilijk**
soort . . . . . **KOMMANDO**
afkomst . . . . . **STRIG is samentrekking van shot en trigger (schot en trekker)**

schrijfwijze

```
                    ON
                    ----
STRIG(<VUURKNOPNUMMER>) OFF
                    ----
                    STOP
```

```
<VUURKNOPNUMMER>::=<N>

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

betekenis

Het is noodzakelijk dat voorafgaand aan deze behandeling de behandeling van ON KEY GOSUB grondig is doorgenomen.

De vuurknoppen van de joy-sticks alsmede de spatiebalk kunnen als funktietoetsen worden geactiveerd. Met behulp van deze funktietoetsen kan net zoals met de gebruikelijke funktietoetsen een programma tijdelijk worden onderbroken.

STRIG...ON, STRIG...STOP en STRIG...OFF hebben met betrekking tot de bijbehorende funktietoetsen dezelfde werking als KEY...ON, KEY...STOP en KEY...OFF. De numerieke uitdrukking bepaalt welke funktietoets wordt geactiveerd:

0 spatiebalk
1 1e vuurknop van joy-stick 1
2 1e vuurknop van joy-stick 2
3 2e vuurknop van joy-stick 1
4 2e vuurknop van joy-stick 2

De waarde van de numerieke uitdrukking tussen haakjes mag dientengevolge niet kleiner zijn dan nul en niet groter dan 4 terwijl een decimale fractie wordt verwaarloosd.

Een STRIG...STOP heeft geen zin indien deze niet werd voorafgegaan door een STRIG...ON.
Zie voor een zinvol voorbeeld ON STRIG GOSUB.

moeilijkheidsgraad . . **normaal, kennis van de ASCII-tabel is een voordeel**
soort . . . . . **A-FUNKTIE**
afkomst . . . . . **STRING$ spreekt voor zich**

schrijfwijze

```
STRING$(<AANTAL TEKENS>,<TE HERHALEN TEKEN>)

<AANTAL TEKENS>::=<N>

<TE HERHALEN TEKEN>::=<A>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Met dit kommando is het mogelijk een lange string, opgebouwd uit allemaal dezelfde karakters, samen te stellen. Tussen haakjes dient dan als eerste het aantal te herhalen tekens te worden opgenomen. Dit aantal moet groter dan of gelijk aan nul zijn en kleiner dan 255. Indien de numerieke uitdrukking die het aantal bepaalt, een gebroken waarde bevat dan wordt de decimale fractie verwaarloosd.
Na de komma dient het te herhalen teken te worden gespecificeerd. Dit kan op twee manieren; via een numerieke uitdrukking en via een alfanumerieke uitdrukking. In het geval van een numerieke uitdrukking gelden dezelfde bepalingen als gegeven bij het aantal tekens. De waarde van de numerieke uitdrukking geeft de ASCII-kode aan van het af te drukken karakter. Zie hiervoor ondermeer hoofdstuk 15. In het geval van een alfanumerieke uitdrukking geldt dat het linker karakter van de waarde van deze alfanumerieke uitdrukking het teken vormt dat gaat worden herhaald. Bijvoorbeeld:

```
NEW
Ok
10 PRINT STRING$(20,"A")
20 PRINT STRING$(22,77)
30 LET A$=STRING$(10,"*")+"MSX"+STRING$(10,"*")
40 PRINT A$
RUN
AAAAAAAAAAAAAAAAAAAA
```

```
MMMMMMMMMMMMMMMMMMMMM
**********MSX**********
Ok
```

(77 is de ASCII-kode voor M)

SLEUTELWOORD # SWAP

moeilijkheidsgraad . . . . . **vrij eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **SWAP is verwisselen**

schrijfwijze

```
SWAP<ALFANUMERIEKE VARIABELE>,<ALFANUMERIEKE VARIABELE>
-------------------------------------------------------
SWAP<NUMERIEKE VARIABELE>,<NUMERIEKE VARIABELE>

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

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

betekenis

Met het kommando SWAP kunnen de waarden van twee numerieke of twee alfanumerieke variabelen worden verwisseld. Voorbeeld:

```
NEW
Ok
10 A=5:B=4:SWAP A,B:PRINT A;B
RUN
 4  5
Ok


NEW
Ok
10 DIM A(1):A(0)=5:A(1)=123
20 SWAP A(0),A(1)
30 PRINT A(0);A(1)
RUN
 123  5
Ok
```

De tweede in SWAP vermelde variabele moet in het programma reeds eerder zijn toegewezen (een waarde hebben gehad of zijn gedimensioneerd); een klein foutje in MSX-basic.

moeilijkheidsgraad . . **vrij moeilijk, kennis van goniometrie is noodzakelijk**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **TAN is afkorting van tangent – tangens**

schrijfwijze

```
TAN(<HOEK>)

<HOEK>:::=<N>

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

betekenis

Deze funktie geeft als resultaat de tangens van de waarde van de tussen haakjes vermelde uitdrukking. Deze waarde wordt beschouwd als de uitdrukking van een hoek in radialen.

Alhoewel elke waarde voor de hoek is toegestaan, is het in verband met de precisie van berekenen raadzaam om niet te grote hoekmaten te gebruiken.

```
NEW
Ok
10 FOR I=0 TO 90
20 LET HOEK=I/180*3.1415926536#
30 PRINT I;":";TAN(HOEK)
40 NEXT I
RUN
```

Een tangenstabel verschijnt op het beeldscherm. Merk op dat in regel 20 de hoek van graden naar radialen geconverteerd wordt. Merk ook op dat, in tegenstelling tot de theorie, TAN geen foutmelding geeft bij een hoek van 90 graden (of ½π radialen), doch slechts een grote waarde geeft.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **TIME is tijd**

schrijfwijze

```
TIME
```

betekenis

De MSX-computer bezit een interne klok. (MSX-2: niet te verwarren met de clock-chip). Deze klok wordt bij het aanzetten op 0 gezet en elke 1/50 seconde met de waarde 1 verhoogd. De waarden van de systeemklok kunnen we onder de systeemvariabele TIME opvragen en we kunnen de klok ook op 0 zetten.
Bijvoorbeeld:

```
NEW
Ok
10 TIME=0
20 FOR I=1 TO 50:LET A=SQR(I):NEXT I
30 LET T=TIME/50
40 PRINT "50 WORTELTREKKINGEN"
50 PRINT "KOSTEN";T;"SECONDEN."
RUN
50 WORTELTREKKINGEN
KOSTEN 6.56 SECONDEN.
Ok
```

In bovenstaand voorbeeld werden 50 worteltrekkingen gedaan. Gemeten werd hoeveel seconden hiervoor nodig waren.

Het is niet aan te bevelen om de systeemklok voor nauwkeurige en langdurige tijdmetingen te gebruiken; daarvoor is de systeemklok te onnauwkeurig.

Indien de systeemklok (na ongeveer 22 minuten) de waarde 65535 overschrijdt, begint hij weer bij 0. De systeemklok staat stil wanneer de computer bezig is met niet onderbreekbare acties zoals het lezen van of schrijven naar cassette.

In samenwerking met de RND-funktie kan de TIME-variabele nog een heel dankbare taak vervullen.

Het vervelende van de RND-funktie is, dat een willekeurige, volkomen van de bediening onafhankelijke bepaling van een toevalsgetal niet mogelijk is. Vooral met het programmeren van spelletjes is dit een groot probleem. Door nu op een slimme wijze van TIME gebruik te maken, kan dit probleem voor goed uit de wereld geholpen worden:

```
NEW
Ok
10 REM TOEVALSGETALLEN
20 LET A=RND(-TIME)
30 PRINT RND(1)
40 GOTO 30
RUN
```

Het bovenstaande programma levert steeds een andere reeks toevalsgetallen. Het geheim zit in regel 20 waar (zie de behandeling van RND) steeds weer een andere reeks toevalsgetallen wordt geselekteerd. Doordat TIME 50 maal per seconde verandert is het welhaast onmogelijk voor de mens om via de bediening twee maal dezelfde reeks bewust te selekteren.

N.B.: Het sleutelwoord LET mag niet voor TIME worden gebruikt.

N.B.: onder MSX-2 kunnen tijd metingen beter met behulp van GET TIME worden verricht, de speciale clock-chip die in de MSX-2 computer zit, is zeer nauwkeurig.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **TROFF is afkorting van trace off – ophouden met spoorzoeken**

schrijfwijze

TROFF

betekenis

Dit kommando wordt tesamen met TRON behandeld.
Zie aldaar.

moeilijkheidsgraad . . . . . **zeer eenvoudig**
soort . . . . . **KOMMANDO**
afkomst . . . . . **TRON is afkorting van trace on – beginnen met spoorzoeken**

schrijfwijze

```
TRON
```

betekenis

Dit kommando heeft tot gevolg dat het programma u bij uitvoering laat zien waar het binnen het programma bezig is. TRON is als zodanig een uitstekend hulpmiddel om funktionele fouten in een programma op te sporen. Voorbeeld:

```
10 TRON
20 LET A=1
30 LET B=2*A
40 LET A=2*B
50 IF A=16 THEN STOP
60 GOTO 30
RUN
[20][30][40][50][60][30][40][50]
Break in 50
Ok
```

De tussen de haken vermelde getallen zijn de regelnummers waar het programma mee bezig is. De loop van het programma kan op deze wijze perfekt worden bestudeerd.

TRON wordt door TROFF weer uitgeschakeld. Bijvoorbeeld:

```
10 TROFF
RUN
[10]
Break in 50
Ok
```

TRON en TROFF kunnen in een programmaregel worden gebruikt maar zijn veel bruikbaarder als direkte kommando's.

moeilijkheidsgraad . . zeer moeilijk, kennis van machinetaal en/of ROM-inhoud is vereist
soort . . . . . N-FUNKTIE of A-FUNKTIE
afkomst . . . . . USR is afkorting van user function – door gebruiker ontworpen (niet basic) funktie

schrijfwijze

```
USR[<CIJFER>](<U>)

<CIJFER>::=<ZIE ALGEMENE SPECIFICATIES>

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

betekenis

Deze funktie wordt onder DEF USR behandeld; zie aldaar.

moeilijkheidsgraad . . . . . **normaal**
soort . . . . . **FUNKTIE**
afkomst . . . . . **VAL is afkorting van value – waarde**

schrijfwijze

```
VAL(<A>)

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

betekenis

Met deze funktie kan een alfanumerieke uitdrukking numeriek worden onderzocht. De funktie VAL onderzoekt de tussen haakjes opgenomen alfanumerieke uitdrukking van links naar rechts. Zolang geen verboden karakters voor een numerieke konstante worden tegengekomen, blijft deze funktie de alfanumerieke variabele doorwerken. Zodra echter een karakter wordt tegengekomen dat verboden is, stopt de funktie VAL het verdere onderzoek en wordt de tot dat toe opgebouwde numerieke konstante als resultaat gegeven. Bijvoorbeeld:

```
NEW
Ok
10 LET A$="123GHJ4"
20 LET A=VAL(A$)
30 PRINT A
40 STOP
RUN
 123
Break in 40
Ok


10 LET A$="1E22HJKHJK"          (alleen regel 10 wordt vervangen)
RUN
 1E+22
Break in 40
Ok


NEW
Ok
10 LINE INPUT "WAARDE:";A$:LET A=VAL(A$)
20 PRINT "WORTEL=";SQR(A):GOTO 10
RUN
```

```
WAARDE:12
WORTEL= 3.4641016151377
WAARDE:14
WORTEL= 3.7416573867739
WAARDE:
```

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het computergeheugen is noodzakelijk**
soort . . . . . **SYSTEEMVARIABELE**
afkomst . . . . . **VARPTR is afkorting van variabele pointer – adres van variabele**

schrijfwijze

VARPTR (<VARIABELE> / #<KANAAL>)

<VARIABELE>::=<ZIE ALGEMENE SPECIFICATIES>

<KANAAL>::=<N>

<N>::=<ZIE ALGEMENE SPECIFICATIES>

betekenis

- bij een numerieke variabele: het geheugenadres van het eerste byte van deze variabele;
- bij een alfanumerieke variabele: het geheugenadres van de drie controlebytes van de alfanumerieke variabele (1e byte = lengte string, 2e en 3e byte = adres van het eerste karakter uit de string);
- bij een kanaal nummer: het geheugenadres van het eerste byte van het file control block van het over het betreffende kanaal geopende bestand.

Indien deze funktie een negatief resultaat geeft, dient de waarde 65536 bij dit resultaat te worden opgeteld teneinde het juiste geheugenadres te verkrijgen.

In het volgende voorbeeld wordt een alfanumerieke variabele byte voor byte geprojekteerd m.b.v. VARPTR.

```
NEW
Ok
10 LET A$="DIT IS EEN TEST"
20 LET A=VARPTR(A$)'ADRES CONTROLEBYTES
```

```
30 LET SA=PEEK(A+1)+256*PEEK(A+2)'ADRES EERSTE BYTE
40 LET L=PEEK(A)'LENGTE STRING
50 FOR I=0 TO L-1
60 PRINT CHR$(PEEK(SA+I));
70 NEXT I
RUN
DIT IS EEN TEST
Ok
```

moeilijkheidsgraad . . zeer moeilijk, kennis van de betekenis van de registers van de beeldscherm controlechip is vereist
soort . . . . . SYSTEEMVARIABELE
afkomst . . . . . VDP is afkorting van video display processor – beeldscherm processor

schrijfwijze

```
VDP(<REGISTERNUMMER>)

<REGISTERNUMMER>::=<N>

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

betekenis

De VDP-systeemvariabele representeert de inhoud van de registers van de VDP, de Video Display Processor. Achter het sleutelwoord VDP dient tussen haakjes een registernummer te worden opgegeven. Onder MSX-1 mag dit nummer niet kleiner zijn dan nul en niet groter dan 8. Een eventuele decimale fractie wordt verwaarloosd.
VDP (0) tot en met VDP (7) zijn data-registers. VDP (8) is een status-register en kan niet worden veranderd.

Onder MSX-2 laten de VDP-registers zich als volgt verdelen:

VDP (-9) tot en met VDP (-1)	status-registers. Deze kunnen alleen worden afgevraagd en niet worden veranderd.
VDP (Ø) tot en met VDP (7)	data-registers. Deze kunnen worden opgevraagd en veranderd.
VDP (8)	status-register. Dit register kan alleen worden afgevraagd en niet worden veranderd.
VDP (9) tot en met VDP (24)	data-registers. Deze registers kunnen worden opgevraagd en veranderd.

VDP (33) tot en met VDP (47) instructie-registers. Deze registers kunnen alleen worden veranderd en niet worden opgevraagd.

Door de registers van de VDP te besturen, kunnen diverse effecten worden bereikt. Onoordeelkundig gebruik van het VDP-sleutelwoord leidt meestal tot onvoorspelbare resultaten; vaak wordt de computer 'opgehangen' en kan alleen uit- en weer inschakelen de computer weer tot leven brengen.

Een voorbeeld.

```
NEW
Ok
10 VDP(2)=100
RUN
```

Het beeldscherm vertoont vreemde effecten. De meest eenvoudige oplossing is uit- en inschakelen.

Het sleuterwoord LET is voor VDP niet toegestaan.

Zie voor de betekenis van de betreffende VDP-registers hoofdstuk 14.

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het beeldschermgeheugen is noodzakelijk**
soort . . . . . **N-FUNKTIE**
afkomst . . . . . **VPEEK is afkorting van video memory peek – gluren in het beeldscherm geheugen**

schrijfwijze

```
VPEEK(<GEHEUGENADRES>)

<GEHEUGENADRES>::=<N>

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

betekenis

Het is noodzakelijk dat eerst de behandeling van het PEEK-kommando wordt doorgenomen.

VPEEK heeft hoegenaamd dezelfde funktie als PEEK met dit verschil dat de geheugenadressering niet kleiner mag zijn dan 0 en onder MSX-1 niet groter mag zijn dan 16383 en dat niet het computergeheugen maar het videogeheugen wordt aangesproken.

Voor een goed gebruik van VPEEK is een gedetailleerde kennis van de opbouw van het video-geheugen van een MSX-computer noodzakelijk. In hoofdstuk 14 wordt hierop wat nader ingegaan.

MSX-2: Merk op dat bij 128 kB video-RAM de 'bovenste' 64 kB niet zonder meer met VPEEK benaderbaar zijn.

moeilijkheidsgraad . . **zeer moeilijk, kennis van de opbouw van het beeldscherm geheugen is vereist**
soort . . . . . **KOMMANDO**
afkomst . . . . . **VPOKE is afkorting van video memory poke – stop weg in beeldscherm geheugen**

schrijfwijze

```
VPOKE<GEHEUGENADRES>,<GEHEUGENWAARDE>

<GEHEUGENADRES>::=<N>

<GEHEUGENWAARDE>::=<N>

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

betekenis

Het is noodzakelijk dat eerst de behandeling van POKE wordt doorgenomen.

VPOKE heeft hoegenaamd dezelfde funktie als POKE met dit verschil dat de geheugenadressering niet kleiner mag zijn dan 0 en onder MSX-1 niet groter mag zijn dan 16383 en dat niet het computergeheugen maar het video-geheugen wordt aangesproken.

Voor een goed gebruik van VPOKE is een gedetailleerde kennis van de opbouw van het video-geheugen van een MSX-computer noodzakelijk. In hoofdstuk 14 wordt hierop wat nader ingegaan.

MSX-2: Merk op dat bij 128 kB video-RAM de 'bovenste' 64 kB niet zondermeer met VPOKE benaderbaar zijn.

moeilijkheidsgraad . . zeer moeilijk, kennis van de funkties van de poorten van het computersysteem is vereist
soort . . . . . KOMMANDO
afkomst . . . . . WAIT is wachten

schrijfwijze

```
WAIT<POORT>,<FILTER 1>[,<FILTER 2>]

<POORT>::=<N>

<FILTER 1>::=<N>

<FILTER 2>::=<N>

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

betekenis

Met dit kommando kan worden gewacht totdat de waarde van het byte op een bepaalde poort een bepaalde waarde heeft. De condities waaronder het wachten niet meer behoeft te gebeuren, zijn gekodeerd in filter 1 en filter 2.
Het poortnummer mag niet kleiner zijn dan –32768 en niet groter dan 65535. Een eventuele decimale fractie wordt verwaarloosd. Indien het poortnummer kleiner is dan 0, wordt 65536 bij dit nummer opgeteld voordat de betreffende poort wordt benaderd.
De beide filterwaarden mogen niet kleiner zijn dan 0 en niet groter dan 255 terwijl een decimale fractie wordt verwaarloosd.
Indien de tweede filterwaarde wordt weggelaten, wordt een 0 voor deze waarde aangenomen.

WAIT voert continu op binair (bit-) niveau een XOR uit tussen de waarde die op dat moment in de buffer van de betreffende poort staat en filter 2. Tussen het resultaat van deze berekening en filter 1 wordt daarna een AND uitgevoerd. Indien het uiteindelijke resultaat ongelijk is aan 0, wordt er niet meer gewacht maar wordt het programma vervolgd. Indien het uiteindelijke resultaat gelijk is aan 0, wordt de controle wederom uitgevoerd.

Filter 2 kan men beschouwen als een waarde die aangeeft, welke bits dienen te worden geinverteerd. Filter 1 is dan een waarde die aangeeft

van welke bits er minimaal één op 1 moet staan voor verdere doorgang van het programma.

Voorbeeld:

```
WAIT 168,1
```

De computer wacht; alleen een CONTROL-BREAK kan dit wachten opheffen.

Voor een goed gebruik van WAIT is een gedetailleerde kennis van de hardware-opbouw van een MSX-computer noodzakelijk. Daarbij dienen de diverse componenten ook software-technisch te worden beheersd. Hiertoe dient specialistische literatuur te worden geraadpleegd; in dit handboek wordt hierop niet verder ingegaan.

SLEUTELWOORD # WIDTH

moeilijkheidsgraad ..... **eenvoudig**
soort ..... **KOMMANDO**
afkomst ..... **WIDTH is breedte**

schrijfwijze

```
WIDTH<TEKSTREGELBREEDTE>

<TEKSTREGELBREEDTE>::=<N>

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

betekenis

Met het WIDTH-kommando kan de gewenste regelbreedte worden ingesteld. Afhankelijk van de SCREEN-instelling (zie de behandeling van SCREEN) kan de breedte maximaal 32 of 40 karakters zijn. Onder MSX-2 kan de breedte onder de SCREEN 0-instelling maximaal gelijk zijn aan 80 karakters. Het minimum is natuurlijk gelijk aan 1. Indien de numerieke uitdrukking tussen de haakjes een gebroken waarde als uitkomst heeft, dan wordt een decimale fractie verwaarloosd. Bij het uitvoeren van een WIDTH wordt het beeldscherm schoongemaakt tenzij de betreffende breedte reeds geactiveerd was.

Voorbeeld:

```
NEW
Ok
10 WIDTH 4
20 PRINT "HALLO ALLEMAAL"
RUN                          (beeldscherm wordt schoongemaakt)
                    HALL
                    O AL
                    LEMA
                    AL
                    Ok
```

10 MSX SLEUTELWOORDEN/AANBEVOLEN LEERVOLGORDE

In hoofdstuk 9 zijn de sleutelwoorden op alfabetische volgorde opgenomen. Hieronder volgt een opsomming in de volgorde waaronder deze sleutelwoorden het beste kunnen worden bestudeerd door de nieuweling op MSX-basic gebied.

De behandeling van enkele kommando's is onderverdeeld in een eenvoudig gebruik en een gevorderd gebruik. Bestudeer eerst alleen het eenvoudige gebruik. Pas nadat u de aanbevolen leervolgorde heeft afgewerkt of nadat dat nodig blijkt, kunt u de gevorderde behandeling doornemen.
De opgenomen voorbeelden zijn afgestemd op de aanbevolen leervolgorde. De met een ster gemerkte sleutelwoorden vereisen een vrij specialistisch niveau en kunnen de eerste tijd misschien beter maar worden overgeslagen.

11 MSX-FOUTMELDING OP VOLGORDE VAN NUMMER

Hieronder volgen de mogelijke foutmeldingen van MSX-basic. Zij zijn op nummer gesorteerd opgenomen terwijl de betekenis van de foutmelding daar achter is geplaatst.

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 berekenen.
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

61 Bad file mode	MSX-2: het benaderde bestand is voor de uit te voeren actie op verkeerde wijze geopend.
64 File still open	MSX-2: met CALL MKILL wordt getracht, een nog niet met CLOSE gesloten bestand te verwijderen.
65 File already exists	MSX-2: met CALL MNAME wordt geprobeerd, een bestand de naam te geven van een reeds bestaand bestand.
66 RAM disk full	MSX-2: de RAM-disk is vol.
67 Too many files	MSX-2: er kunnen niet meer bestanden op de RAM-disk alhoewel er nog wel opslagruimte is (maximum aantal bestanden = 32).
70 RAM disk offline	MSX-2: de RAM-disk werd nog niet met CALL MEMINI benaderbaar gemaakt.

Indien een niet bestaand foutnummer met ERROR wordt gegenereerd, zal de foutmelding 'Unprintable error' verschijnen. Deze foutmeldingen kunnen door de programmeur een bepaalde betekenis worden toegedacht.

	is te lang voor MSX-basic (langer dan 255 karakters).
50 FIELD overflow	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met ERROR 50 gegenereerd.
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	deze foutmelding is wel aanwezig maar kan in het standaard MSX-1-basic niet voorkomen tenzij met ERROR 53 gegenereerd. Onder MSX-2 kan deze foutmelding in verband met de RAM-disk voorkomen, een gezocht bestand werd niet gevonden.
54 File already open	er werd getracht, een reeds geopend bestand nogmaals te openen.
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	deze foutmelding is wel aanwezig maar kan in het standaard MSX-1-basic niet voorkomen tenzij met ERROR 58 gegenereerd. Onder MSX-2 kan deze foutmelding in verband met de RAM-disk voorkomen indien tijdens het OPEN-statement geen FOR INPUT, FOR OUTSPUT of FOR APPEND werd gegeven.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.

12 MSX-FOUTMELDINGEN OP ALFABETISCHE VOLGORDE

Hieronder volgen de mogelijke foutmeldingen van MSX-basic. Zij zijn op volgorde van alfabet opgenomen. De betekenissen zijn achter de foutmeldingen geplaatst.

56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
61 Bad file mode	MSX-2: het benaderde bestand is voor de uit te voeren actie op verkeerde wijze geopend.
52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
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.
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te berekenen.
50 FIELD overflow	deze foutmelding is wel aanwezig maar kan in het standaard MSX-basic niet voorkomen tenzij met ERROR 50 gegenereerd.
54 File already open	er werd getracht, een reeds geopend bestand nogmaals te openen.
65 File already exists	MSX-2: met CALL MNAME wordt geprobeerd, een bestand de naam te geven van een reeds bestaand bestand.
53 File not found	deze foutmelding is wel aanwezig maar kan in het standaard MSX-1-basic niet voorkomen tenzij met

	ERROR 53 gegenereerd. Onder MSX-2 kan deze foutmelding in verband met de RAM-disk voorkomen, een gezocht bestand werd niet gevonden.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.
64 File still open	MSX-2: met CALL MKILL wordt getracht, een nog niet met CLOSE gesloten bestand te verwijderen.
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.
66 RAM disk full	MSX-2: de RAM-disk is vol.
70 RAM disk offline	MSX-2: de RAM-disk werd nog niet met CALL MEMINI benaderbaar gemaakt.
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.
58 Sequential I/O only	deze foutmelding is wel aanwezig maar kan in het standaard MSX-1-basic niet voorkomen tenzij met ERROR 58 gegenereerd. Onder MSX-2 kan deze foutmelding in verband met de RAM-disk voorkomen indien tijdens het OPEN-statement geen FOR INPUT, FOR OUTPUT of FOR APPEND werd gegeven.
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	MSX-2: er kunnen niet meer bestanden op de RAM-disk alhoewel er nog wel opslagruimte is (maximum aantal bestanden = 32).
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.

Indien een niet bestaand foutnummer met ERROR wordt genegeerd, zal de foutmelding 'Unprintable error' verschijnen. Deze foutmeldingen kunnen door de programmeur een bepaalde betekenis worden toegedacht.

De AY-3-8912 geluidsprocessor wordt bestuurd via veertien registers, genummerd vanaf 0 tot en met 13. Deze registers dienen via het SOUND-kommando (zie de behandeling in hoofdstuk 9) te worden voorzien van bepaalde waarden.

Bij het programmeren dienen per geluidsvariatie steeds de volgende stappen te worden overwogen:

- zet de betreffende geluidskanalen uit (register 7)
* zet de toonhoogte (register 0 ... 6)
* zet het volume of het betreffende effect op de juiste waarde (register 8, 9 of 10)
* zet de tijdsduur van de effectontwikkeling op de juiste waarde (register 11 en 12)
* zet het juiste soort effect aan (register 13)
- zet de betreffende kanalen aan (register 7).

Voor de met een * gemerkte stappen geldt dat deze alleen dienen te worden ondernomen wanneer zij van toepassing zijn en nog niet eerder zijn bepaald, of veranderd dienen te worden.

Op de volgende pagina is een schema geplaatst van de PSG-registers met hun betekenis. Daarna volgt per register een korte toelichting.

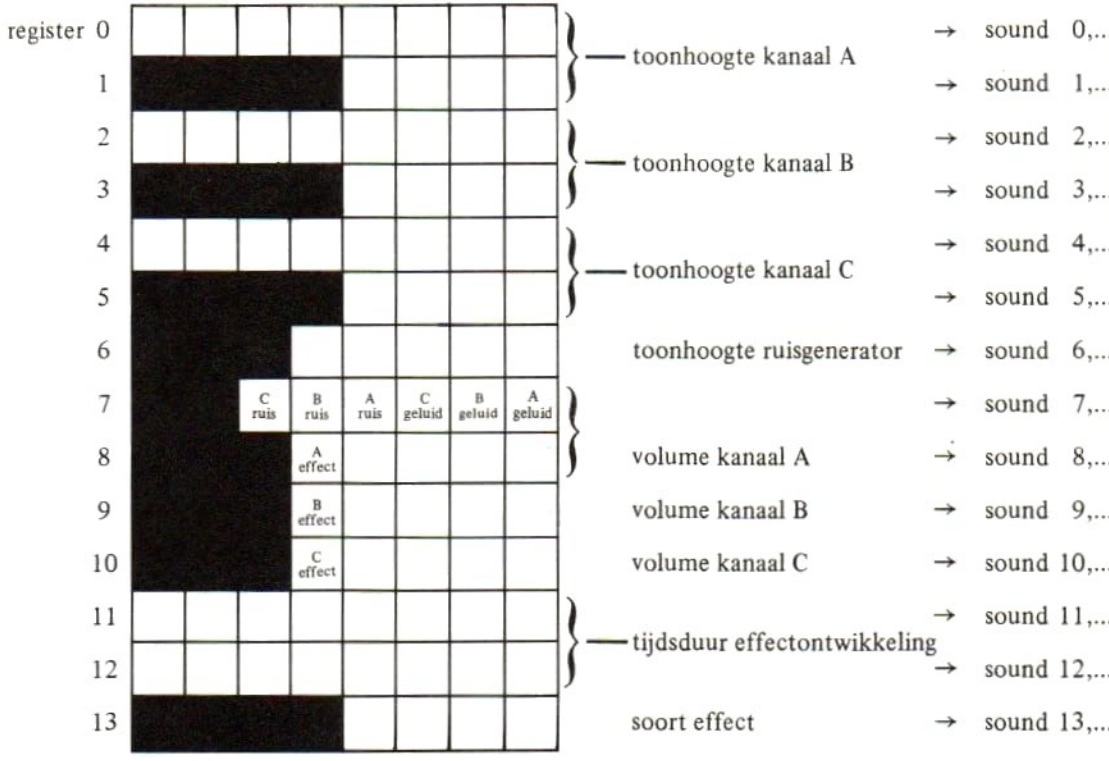
De geluidskanalen zijn de kanalen A, B en C. Per geluidskanaal kan in twee registers de toonhoogte worden gekodeerd. Voor de te programmeren toonhoogte geldt de volgende formule:

$$f = \frac{111760}{Hz} \qquad (f<4096)$$

Indien de toonhoogte in Herz bekend is, kan deze in de formule worden ingevuld; het resultaat noemen we f (frequentie).

Bijvoorbeeld: een toon van 440 Hz resulteert in een f van ongeveer 254.
Steeds dienen de eindwaarden te worden afgerond.

Nadat de waarde f is berekend, dient deze in het betreffende registerpaar te worden opgenomen. Dit gaat als volgt:


register 0
1
2
3
4
5
6
7
8
9
10
11
12
13
C ruis
B ruis
A ruis
C geluid
B geluid
A geluid
A effect
B effect
C effect
toonhoogte kanaal A
toonhoogte kanaal B
toonhoogte kanaal C
toonhoogte ruisgenerator
volume kanaal A
volume kanaal B
volume kanaal C
tijdsduur effectontwikkeling
soort effect
→ sound 0,...
→ sound 1,...
→ sound 2,...
→ sound 3,...
→ sound 4,...
→ sound 5,...
→ sound 6,...
→ sound 7,...
→ sound 8,...
→ sound 9,...
→ sound 10,...
→ sound 11,...
→ sound 12,...
→ sound 13,...

Tik in: PRINT BIN$(...) en vul op de puntjes de waarde voor f in. Bij f=254 wordt dat:

```
PRINT BIN$(254)
11111110
Ok
```

Vul vervolgens deze waarde vooraan op met nullen tot 12 posities. In ons voorbeeld met f=254 krijgen we dan:

```
000011111110
```

Kap hierna de verkregen binaire konstante van 12 posities in een deel van vier en een deel van 8 posities. Ons voorbeeld met f=254 geeft dan:

```
0000    11111110
```

Ken dan aan het betreffende registerpaar deze verkregen konstanten binair toe. Het gedeelte van 8 binaire cijfers moet in het eerste register van het betreffende kanaal worden geplaatst en het gedeelte van 4 binaire cijfers in het volgende register. Wanneer we geluidskanaal A een toon van 440 Hz willen laten produceren, programmeren we ondermeer:

```
NEW
Ok
20 SOUND 0,&B111110
30 SOUND 1,&B0000
```

Indien u dat kunt, mag u de binaire konstanten ook omrekenen naar decimale konstanten en deze in het SOUND-kommando opnemen.

Buiten de geluidskanalen is er ook een ruiskanaal aanwezig. Via dit ruiskanaal kunnen geluidseffecten (motoren, geweerschoten e.d.) worden gesimuleerd of kan het slagwerk bij een stuk muziek worden verzorgd.

Dit ruiskanaal heeft een vrij grove frequentie-aanduiding nodig. Deze aanduiding dient in register 6 te worden opgenomen. &B00000 geeft ruis met een hoge frequentie (licht) en &B11111 geeft een ruis met een lage frequentie (zwaar). Alle tussenliggende waarden mogen van licht naar zwaar worden gebruikt.

Een vrij zware ruis programmeren wij bijvoorbeeld als volgt:

```
40 SOUND 6,&B11001
```

Om een geluidskanaal daadwerkelijk te activeren, dient het te worden aangeschakeld. Een geluidskanaal kan worden uit- of aangeschakeld via register 7. Voor elk kanaal dat uitgeschakeld moet worden of blijven, dient het binaire cijfer 1 te worden ingevuld terwijl voor een kanaal dat aangeschakeld moet worden of blijven, een binaire 0 dient te worden ingevuld. In ons voorbeeld schakelen we eerst alle kanalen uit:

```
10 SOUND 7,&B111111
```

Later, als laatste actie schakelen we het geluid over kanaal A en de ruis over in kanaal A:

```
90 SOUND 7,&B110110
```

Indien geen (volume-)effect wordt gewenst voor een betreffend kanaal, dan dient in register 8, 9 of 10 op die plaats een binaire 0 te worden ingevuld. Wordt echter wèl een effect gewenst, dan dient een binaire 1 te worden ingevuld op die plaats. Indien een 1 is ingevuld, dan heeft het verder geen zin om een volume vast te stellen; elk effect heeft een eigen volume. Wanneer een 0 werd ingevuld, dan kan echter wel het volume worden bepaald. Het volume kan vanaf helemaal dicht (&B00000) tot helemaal open (&B01111) worden geregeld. Alle tussenliggende waarden zijn toegestaan.
In ons voorbeeld kiezen we voor een effect:

```
50 SOUND 8,&B10000
```

Indien voor één of meer kanalen voor een effect werd gekozen, dan dient de tijdsduur van dit effect te worden gespecificeerd in register 11 en 12. De te specificeren waarde is dezelfde als de waarde die na het M-kommando in MML (zie PLAY) dient te worden opgenomen.

Allereerst dient de tijdsduur in seconden te worden bepaald van de effectontwikkeling. Vervolgens wordt dit getal vermenigvuldigd met de waarde 6965. Deze waarde dient te worden afgerond en mag niet groter zijn dan 65535. Tik vervolgens in: PRINT BIN$(...) met op de puntjes de verkregen waarde ingevuld. Bij een gewenste ontwikkelingstijd van 2 seconden is dit bijvoorbeeld:

```
PRINT BIN$(2*6965)
11011001101010
Ok
```

Vul de verkregen binaire konstante vooraan aan met nullen totdat 16 cijfers zijn verkregen en splits deze konstante dan in twee delen van acht binaire cijfers. Ons voorbeeld (2 seconden):

```
    00110110     01101010
```

Vul vervolgens register 11 met het tweede gedeelte van deze konstante en register 12 met het eerste gedeelte. Ons voorbeeld:

```
60 SOUND 11,&B01101010
70 SOUND 12,&B00110110
```

Indien voor één of meer kanalen voor een effect wordt gekozen, dient uiteindelijk nog het soort effect te worden gespecificeerd in register 13. Voor de mogelijke effecten dient u het betreffende onderdeel met schema in hoofdstuk 9, de behandeling van PLAY te bestuderen. Voor ons voorbeeld kiezen we voor effect nummer 1, een éénmalig wegstervend geluid. Tikt u in: PRINT BIN$ (effectnummer). In ons voorbeeld:

```
PRINT BIN$(1)
1
Ok
```

Dit uiteindelijke binaire resultaat dient aan register 13 te worden toegekend:

```
80 SOUND 13,&B1
```

Het resulterende programma in verband met het voorbeeld ziet er in totaal nu als volgt uit:

```
LIST
10 SOUND 7,&B111111
20 SOUND 0,&B11111110
30 SOUND 1,&B0000
40 SOUND 6,&B11001
50 SOUND 8,&B10000
60 SOUND 11,&B01101010
70 SOUND 12,&B00110110
```

```
80 SOUND 13,&B1
90 SOUND 7,&B110110
RUN
Ok
```

Een 'geweerschot', begeleid door een toon van 440 Hz (A), is hoorbaar. Door regel 90 te veranderen naar

```
90 SOUND 7,&B111110
```

of

```
90 SOUND 7,&B110111
```

kunnen we of alleen het schot of alleen de toon hoorbaar maken.

Merk op dat het uiteindelijke voorbeeldprogramma volgens de regels is opgebouwd:

- op regel 10 worden de geluidskanalen allemaal uitgezet
- op regel 20 en 30 wordt de toonhoogte van kanaal A bepaald
- op regel 40 wordt de toonhoogte van de ruisgenerator bepaald
- op regel 50 wordt het effect voor kanaal A aangezet
- op regel 60 en 70 wordt de tijdsduur van de effectontwikkeling bepaald
- op regel 80 wordt het soort effect gekozen
- uiteindelijk wordt op regel 90 de betreffende kanalen (A geluid en A ruis) aangezet; het bedoelde effect is hoorbaar.

Er is maar één manier om de geluidsgenerator onder de knie te krijgen: proberen en nog eens proberen. Zelfs indien u geen muzikale aanleg heeft, zijn door proberen bijzonder leuke geluidseffecten te realiseren. Denk er bijvoorbeeld eens aan om tesamen met effect nummer 8 de ontwikkelingstijd bijzonder kort te houden en een ruis te genereren.

Tot slot één van de mogelijke effecten: men kan zich bij het volgende geluid een vertrekkende vliegende schotel voorstellen.

```
NEW
Ok
10 SOUND 7,&B111111
20 SOUND 8,&B10000
30 SOUND 11,&B01101010
40 SOUND 12,&B11110110
50 SOUND 13,&B1
60 FOR I=4095 TO 1100 STEP -1
```

```
70 SOUND 0,I MOD 256
80 SOUND 1,I\256
90 SOUND 7,&B111110
100 NEXT I
RUN
Ok
```

MSX-1 en MSX-2 hebben één van hun grootste verschillen in de video display processor. MSX-1 maakt gebruik van de TMS 9918 processor terwijl MSX-2 gebruik maakt van de veel uitgebreidere V 9938 processor. Een MSX-1 computer heeft maximaal 16 KB video-geheugen terwijl de MSX-2 computer tot 128 KB video-geheugen kan worden uitgebreid.

Het direkt werken met de video display processor met behulp van het sleutelwoord VDP en het manipuleren binnen het video-geheugen met behulp van de sleutelwoorden VPOKE, VPEEK en BASE is erg lastig. Er is een vrij breed inzicht in de opbouw van de MSX-computer voor nodig en het rekenen met binaire en hexadecimale getallen moet al helemaal geen problemen meer opleveren. . .

De beginnende computergebruiker doet er dan misschien in de eerste instantie ook wijs aan, dit hoofdstuk even links te laten liggen.

14.1 De video-mogelijkheden van MSX

Op de volgende bladzijden vindt u een uitvoerige tabel waarin alle video-mogelijkheden van de MSX-computer zijn samengevat. Een toelichting bij deze tabel:

BASE (0) tot en met BASE (44) geven als resultaat de adressen binnen het video-geheugen van allerlei daarin opgenomen tabellen. In de volgende paragraaf gaan we nader in op de funktie en opbouw van deze tabellen. BASE (0) tot en met BASE (19) zijn in basic te veranderen. BASE (10), BASE (11), BASE (12) en BASE (14) zijn per definitie gelijk aan BASE (20), BASE 21), BASE (22) en BASE (24). Door de eerstgenoemde systeemvariabelen te veranderen, worden ook automatisch deze systeemvariabelen veranderd. De overige BASE-systeemvariabelen zijn niet in basic te veranderen.

De verschillende BASE-systeemvariabelen dienen pas te worden afgevraagd nadat de bijbehorende scherminstelling is geactiveerd.

Bij BASE (0) tot en met BASE (19) zijn moduli vermeld. Deze BASE-systeemvariabelen kunnen alleen maar worden veranderd in een veelvoud van dit getal.

MSX-1 en MSX-2

	SCREEN 0	SCREEN 1	SCREEN 2	SCREEN 3
SCHERM TABEL MODULUS	BASE (0) 1024	BASE (5) 1024	BASE (10) 1024	BASE (15) 1204
KLEUR TABEL MODULUS	BASE (1) 64	BASE (6) 64	BASE (11) 8192	BASE (16) 64 geen funktie
MATRIX TABEL MODULUS	BASE (2) 2048	BASE (7) 2048	BASE (12) 8192	BASE (17) 2048
TRANSPARANT TABEL MODULUS	BASE (3) 128 geen funktie	BASE (8) 128	BASE (13) 128	BASE (18) 128
SPRITE$ TABEL MODULUS	BASE (4) 2048 geen funktie	BASE (9) 2048	BASE (14) 2048	BASE (19) 2048
PALETTE TABEL (alleen MSX-2)	40:1024 80:3840	8224	7040	8224
SPRITECOLOR-TABEL	–	–	–	–
ALFANUME-RIEK/GRAFISCH	A	A	G	G
MAXIMALE OP-LOSSENDE VERM.	MSX-1: 40x24 MSX-2: 80x24	32x24	256x192	64x48
AANTAL KLEU-REN TEGE-LIJK TE GE-BRUIKEN	4 waarvan 2 in MSX-basic	16 waarvan 2 in MSX-basic	16	16
BENODIGD VI-DEO-GEHEUGEN	8 KB	8 KB	16 KB	8 KB
MAX. AANTAL SPRITES NAAST ELKAAR	–	4	4	4
MAX AANTAL KLEUREN PER SPRITE	–	1	1	1
PROJEKTIE METHODE	karakter-georiënteerd	karakter-georiënteerd	karakter-georiënteerd	karakter-georiënteerd
MAX. AANTAL PAGES	–	–	–	–
MAX. AANTAL SPRITES	–	256 normaal 64 vergroot	256 normaal 64 vergroot	256 normaal 64 vergroot
MAX.AANTAL TRANSPARANTEN	–	32	32	32

Alleen MSX-2

SCREEN 4	SCREEN 5	SCREEN 6	SCREEN 7	SCREEN 8
BASE (20) = BASE (10)	BASE (25)	BASE (30)	BASE (35)	BASE (40)
BASE (21) = BASE (11)	BASE (26) geen funktie	BASE (31) geen funktie	BASE (36) geen funktie	BASE (41) geen funktie
BASE (22) = BASE (12)	BASE (27) geen funktie	BASE (32) geen funktie	BASE (37) geen funktie	BASE (42) geen funktie
BASE (23) —	BASE (28) —	BASE (33) —	BASE (38) —	BASE (43) —
BASE (24) = BASE (14)	BASE (29) —	BASE (34) —	BASE (39) —	BASE (44) —
7040	30336	30336	64128	64128
BASE (23) -512	BASE (28) -512	BASE (33) -512	BASE (38) -512	BASE (43) -512
G	G	G	G	G
256x192	256x212	512x212	512x212	256x212
16	16	4	16	256
64KB	64KB	64KB	128KB	128KB
8	8	8	8	8
8 normaal 16 vergroot	8 normaal 16 vergroot	8 normaal 16 vergroot	8 normaal 16 vergroot	8 normaal 16 vergroot
karakter- georiënteerd	bitmapped	bitmapped	bitmapped	bitmapped
—	64KB VRAM:2 128KB VRAM:4	64KB VRAM:2 128KB VRAM:4	2	2
256 normaal 64 vergroot	256 normaal 64 vergroot	256 normaal 64 vergroot	256 normaal 64 vergroot	256 normaal 64 vergroot
32	32	32	32	32

Niet alle BASE-systeemvariabelen hebben daadwerkelijk een funktie. In de tabel is vermeld, welke van deze variabelen funktieloos zijn.

SCREEN 4 tot en met SCREEN 8 zijn alleen onder MSX-2 beschikbaar. Voor MSX-1 geldt slechts het eerste gedeelte van de tabel.
BASE (Ø) tot en met BASE (19) laten zich niet hoger dan 16383 instellen. SCREEN Ø tot en met SCREEN 4 vormen dan ook instellingen die alleen van de eerste 16 KB video-geheugen gebruik kunnen maken. Onder MSX-1 is de maximale schermindeling SCREEN 3 en is slechts maximaal 16 KB video-geheugen aanwezig.

De overige gegevens in de tabel spreken voor zich of worden in de hierna komende behandeling duidelijk.

14.2 De videogeheugen-indeling

Het video-geheugen van de MSX-computer kan men onderverdeeld zien in diverse tabellen, te weten:

De SCHERM-TABEL	Deze tabel geeft in gekodeerde vorm aan, welke gegevens er op het scherm worden geprojekteerd.
De KLEUR-TABEL	Deze tabel geeft aan, welke kleuren de verschillende in de SCHERM-TABEL opgenomen gegevens hebben.
De MATRIX-TABEL	Deze tabel geeft aan hoe de in de SCHERM-TABEL gekodeerde gegevens er daadwerkelijk uit zien.
De TRANSPARANT-TABEL	Deze tabel bevat per transparant de gegevens van de hierop eventueel geprojekteerde sprite.
De SPRITE$-TABEL	Deze tabel geeft per sprite aan hoe deze er uit ziet.

Onder MSX-2 kennen we ook nog de volgende tabellen:

De SPRITECOLOR-TABEL	Deze tabel bevat per sprite (SCREEN 4 t/m SCREEN 8) de kleurinformatie van de verschillende lagen.

De PALETTE-TABEL — Deze tabel geeft per kleurkode aan, welke de samenstelling van de bijbehorende kleur is.

Met behulp van deze tabellen stelt de video display processor het uiteindelijke visuele beeld samen.

De in het beeldschermgeheugen opgenomen tabellen laten zich met behulp van hun bijbehorende BASE-systeemvariabelen localiseren; zie het hiervoor opgenomen schema.

De laatste twee, alleen onder MSX-2 beschikbare tabellen hebben geen eigen BASE-adres. De SPRITECOLOR-TABEL ligt altijd 512 bytes vóór de TRANSPARANT-TABEL terwijl de PALETTE-TABEL in het geheel niet relatief is te benaderen.

14.3 De videogeheugen-indeling onder SCREEN 0

De SCHERM-TABEL heeft een lengte van 960 bytes maar kan onder MSX-2 een lengte van 1920 bytes hebben wanneer een beeldschermbreedte van meer dan 40 karakters is toegewezen (bijvoorbeeld met WIDTH 80). Deze tabel bevat regel voor regel achter elkaar de ASCII-waarden van de geprojekteerde tekens. Welke beeldschermbreedte ook is ingesteld, deze tabel gaat altijd uit van de maximale regelbreedte (40 of 80 tekens).

```
NEW
Ok
10 REM IN DIT VOORBEELD WORDT EEN TEKST OP DE
20 REM BOVENSTE BEELDREGEL GEPROJEKTEERD DOOR
30 REM DEZE IN DE SCHERMTABEL TE PLAATSEN
40 REM MET HET VPOKE-BEVEL
50 Q$="*** TEST-TEKST ***"
60 SCREEN 0:WIDTH 40:FOR I=BASE(0) TO BASE(0)+LEN(Q$)-1
70 VPOKE I,ASC(Q$):Q$=MID$(Q$,2):NEXT I
80 LOCATE 0,10:END
RUN
```

De KLEUR-TABEL heeft alleen onder MSX-2 een funktie. Indien een beeldschermbreedte van meer dan 40 tekens per regel is toegewezen (bijvoorbeeld met een WIDTH 80), heeft deze tabel een lengte van 240 bytes.

De tabel is dus 240 maal 8 is gelijk aan 1920 bits groot. Elk bit binnen deze tabel correspondeert met een volgend geprojekteerd teken

op het beeldscherm. Indien het bit op 0 staat, wordt de normaal ingestelde kleur voor het bijbehorende teken gehanteerd. Staat dit bit echter op 1, dan worden de kleuren uit VDP-register 12 (=VDP (13)) eventueel toegepast. VDP-register 13 (=VDP (14)) bepaalt, hoe lang in dat geval de normaal in gestelde kleur en hoe lang de afwijkende kleur afwisselend wordt geprojekteerd.

In VDP(13) kunnen de alternatieve kleuren als volgt worden gekodeerd:

VDP (13) =16 maal de voorgrondkleur + de achtergrondkleur

In VDP (14) kan de afwisseling van normale kleuren en alternatieve kleuren als volgt worden gekodeerd:

VDP(14)=16 maal kleur 2 + kleur 1

waarbij kleur 2 staat voor het aantal vijfden van seconden dat de alternatieve kleurinstelling actief moet zijn en kleur 1 staat voor het aantal vijfden van seconden dat de normaal ingestelde kleur actief moet zijn. Kleur 1 en kleur 2 mogen maximaal gelijk zijn aan 15.

De kleurtabel wordt tijdens de scherminstelling niet schoongemaakt. MSX-basic maakt geen gebruik van deze mogelijkheid tot gebruik van vier kleuren onder de SCREEN 0-instelling.

```
NEW
Ok
10 REM IN DIT VOORBEELD WORDT EEN SCHERM
20 REM OPGEBOUWD MET KNIPPERTEKSTEN ER IN
30 REM
40 SCREEN 0:WIDTH 60:COLOR 15,4,4:CLS
50 LOCATE 0,10:PRINT "DIT IS EEN REGEL TEKST MET KNIPPERENDE
 WOORDEN ER IN"
60 FOR I=BASE(1) TO BASE(1)+239:VPOKE I,0:NEXT I
70 VDP(13)=63:VDP(14)=33'KLEUR EN TIJD
80 REM BITS IN KLEUR-TABEL
90 VPOKE BASE(1)+102,7:VPOKE BASE(1)+103,192
100 VPOKE BASE(1)+104,7:VPOKE BASE(1)+105,255
RUN
```

De MATRIX-TABEL heeft een lengte van 2048 bytes en bevat per acht bytes de opbouw van een karakter. Een op het beeldscherm geprojekteerd karakter bestaat uit puntjes die in een 8 bij 8 matrix passen. Elk groepje van acht bytes in deze tabel kan worden gezien als een 8 bij 8 bits matrix waarin met de waarde 1 en 0 is aangegeven, welk puntje in het betreffende karakter wel of niet voorkomt.

```
NEW
Ok
10 REM DIT PROGRAMMA LAAT DE MATRIX-OPBOUW
20 REM VAN ALLE MSX-KARAKTERS ZIEN ZOALS
30 REM IN DE MATRIXTABEL GEKODEERD
40 SCREEN 0:WIDTH 37:COLOR 15,4,4:CLS
50 FOR I=BASE(2) TO BASE(2)+2047
60 Q$=RIGHT$("00000000"+BIN$(VPEEK(I)),8)
70 FOR J=1 TO 8:IF MID$(Q$,J,1)="0" THEN MID$(Q$,J,1)=" " EL
SE MID$(Q$,J,1)="#"
80 NEXT J:IF I MOD 8=0 THEN PRINT:PRINT "ASCII-KODE";(I-BASE
(2))/8:PRINT
90 PRINT Q$:NEXT I
RUN
```

De TRANSPARANT-TABEL en de SPRITE$-TABEL hebben onder SCREEN 0 geen funktie

MSX-2

De SPRITECOLOR-TABEL heeft onder SCREEN 0 geen funktie.

De PALETTE-TABEL is 32 bytes lang. Per kleurkode (0 . . . 15) zijn twee bytes opgenomen. In het eerste byte hiervan zijn het aantal delen rood en het aantal delen blauw gekodeerd. De totale waarde van dit byte is 16 maal het aantal delen rood + het aantal delen blauw. In het daaropvolgende byte is alleen het aantal delen groen gekodeerd. Het aantal delen rood, groen en blauw mag de waarde 7 niet overschrijden.

```
NEW
Ok
10 REM DIT PROGRAMMA STELT U IN STAAT OM DE
20 REM PALETTE-SAMENSTELLING VANUIT VRAM OP
30 REM TE VRAGEN
40 REM
50 VRAM=3840'LOCATIE PALETTETABEL
60 DEF FND$(A)=MID$("DEELDELEN",SGN(ABS(A-1))*4+1,5+(A=1))
70 SCREEN 0:WIDTH 76:CLS
80 INPUT "PALETTE-NUMMER";PAL
90 IF PA<0 OR PA>15 OR PA<>INT(PA) THEN 80
100 ROOD=VPEEK(VRAM+2*PA)\16
110 BLAUW=VPEEK(VRAM+2*PA) MOD 16
120 GROEN=VPEEK(VRAM+2*PA+1)
130 PRINT "PALETTE";PA;"HEEFT";ROOD;FND$(ROOD);" ROOD,";BLAU
W;FND$(BLAUW);" BLAUW EN";GROEN;FND$(GROEN);" GROEN"
140 GOTO 80
RUN
```

Pas na het COLOR = RESTORE-bevel worden de nieuw in het beeld-

schermgeheugen gekodeerde kleuren geactiveerd.

14.4 De videogeheugen-indeling onder SCREEN 1

De SCHERM-TABEL heeft een lengte van 768 bytes en bevat regel voor regel achter elkaar de ASCII-waarden van de geprojekteerde tekens. Deze tabel gaat altijd uit van 32 tekens per regel.

De KLEUR-TABEL heeft een lengte van 32 bytes. Elke byte bevat 16 maal een voorgrondkleurkode + een achtergrondkleurkode.

Het eerste byte in deze tabel bepaalt de kleuring van de ASCII-kodes 0 t/m 7. Het tweede byte bepaalt de kleuring van de ASCII-kodes 8 t/m 15, enzovoorts. Het tweeëndertigste byte bevat tenslotte de kleurinformatie voor de ASCII-kodes 248 t/m 255.

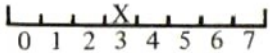
```
NEW
Ok
10 REM DIT VOORBEELD LAAT MEERDERE KLEUREN
20 REM IN SCREEN 1 ZIEN
30 SCREEN 1:WIDTH 32:CLS:COLOR 15,1,1
40 PRINT "Dit is een kleurtest onder"
50 PRINT "SCREEN 1. Deze tekst krijgt"
60 PRINT "dadelijk verschillende"
70 PRINT "kleuren!!!"
80 PRINT:PRINT
90 PRINT "ALLE MOGELIJKE MSX-TEKENS:"
100 PRINT "(let op de 8-groepering)"
110 PRINT STRING$(32,"-");
120 FOR I=0 TO 255
130 IF I<32 THEN PRINT CHR$(1)+CHR$(64+I); ELSE PRINT CHR$(I
);
140 NEXT I
150 PRINT:PRINT STRING$(32,"-")
160 VPOKE BASE(6)+64*RND(1),RND(1)*255:GOTO 160
RUN
```

MSX-basic ondersteunt onder SCREEN 1 slechts het gebruik van één voor- en één achtergrondkleur.

De MATRIX-TABEL heeft dezelfde funktie als onder SCREEN 0.

De TRANSPARANT-TABEL heeft een lengte van 128 bytes. Per transparant bevat deze tabel vier bytes.
In het eerste byte is de Y-coördinaat van de op de transparant geplaatste sprite opgenomen. In het tweede byte is de bijbehorende X-coördinaat opgenomen. Het derde byte bevat het spritenummer van de op deze transparant afgebeelde sprite. Het vierde en laatste byte bevat

tenslotte de kleurkode van de sprite en nog wat aanvullende gegevens. Dit byte is als volgt opgebouwd:



Bits 4-7 bevatten de kleurkode (0/15).

Bit 3 heeft geen funktie.

Bit 0: =0 normale situatie
=1 de sprite wordt 32 beeldpunten verder naar links geprojekteerd. Links uitgeschoven sprites of sprite-delen worden niet rechts geprojekteerd.
Wanneer onder MSX-basic een X-coördinaat, kleiner dan nul wordt opgegeven, dan wordt dit bit automatisch op 1 gesteld en wordt de X-coördinaat daaraan aangepast. Zo ontstaat links in het beeld een speling van 32 beeldschermpunten waarin een sprite kan verdwijnen zonder onmiddellijk rechts weer te verschijnen.

Bit 1: =0 normale situatie
(MSX-2) =1 De sprite wordt niet geprojekteerd. Indien een sprite met dit bit =1 een voorliggende sprite met dit bit =0 passeert, dan wordt hij zichtbaar over de samenvallende horizontale beeldlijnen. Indien deze sprite botsen, dan worden de kleuren van de beide sprites logisch met elkaar geORed op sommige beeldlijnen*. Op deze beeldlijnen wordt een sprite-botsing niet gedetecteerd en kan niet worden afgevraagd met ON SPRITE GOSUB.

Indien een sprite met dit bit =1 een achterliggende sprite ontmoet met dit bit =0 dan blijft deze onzichtbaar. Een sprite-botsing kan in dat geval niet worden gedetecteerd en worden afgevraagd met ON SPRITE GOSUB.

Indien twee sprites met dit bit =1 elkaar passeren, dan wordt de achterliggende sprite over sommige beeldlijnen* geprojekteerd. Sprite-botsingen worden niet gedetecteerd en kunnen dus ook niet met ON SPRITE GOSUB worden afgevraagd.

Bit 2: =0 normale situatie.
(MSX-2) = 1 Met betrekking tot deze sprite worden sprite-botsingen niet meer gedetecteerd. Zij kunnen dus niet met ON SPRITE GOSUB worden afgevraagd.

*: dit zijn de beeldlijnen:

8x8 sprites, normaal	: lijn 0, 4, 8 . . .
8x8 sprites, vergroot	: lijn 4 en 5, 12 en 13, 20 en 21 . . .
16x16 sprites, normaal	: lijn 0, 2, 4, 6 . . .
16x16 sprites, vergroot	: lijn 0 en 1, 4 en 5, 8 en 9 . . .

Enkele opmerkingen:

Een transparant gekodeerde sprite (kleurkode 0) heeft nooit een gedetecteerde botsing tot gevolg.

MSX-basic ondersteunt de funkties van de besproken bits 1 en 2 niet. Om deze bits te kunnen gebruiken, zal het kommando PUT SPRITE dienen te worden vermeden en moet de TRANSPARANT-TABEL met VPOKE direkt worden gevuld.

```
NEW
Ok
10 REM DIT VOORBEELD LAAT ZIEN:
20 REM
30 REM - HOE SPRITES DOOR DIREKTE VPOKE-KOMMANDO'S KUNNEN
40 REM   WORDEN GEPLAATST
50 REM - DAT BIT 0 EN 1 INDERDAAD DE HIERBOVEN BESPROKEN
60 REM   EFFEKTEN TOT GEVOLG HEBBEN IN DE KLEURKODERING
70 REM   VAN EEN SPRITE
80 REM
90 S0=&B10001100'SPRITE 0,KLEUR 12, BIT 0 AAN
100 S1=&B01000011'SPRITE 1,KLEUR 3, BIT 1 AAN
110 ON SPRITE GOSUB 180
120 SCREEN 1,3:WIDTH 28
130 SPRITE$(0)=STRING$(32,255):SPRITE$(1)=SPRITE$(0)
140 VPOKE BASE(8)+8,100:VPOKE BASE(8)+9,132:VPOKE BASE(8)+10
,0:VPOKE BASE(8)+11,S0'PUT SPRITE 2 SIMULATIE
150 SPRITE ON:FOR I=0 TO 191 STEP 4
160 VPOKE BASE(8)+16,I:VPOKE BASE(8)+17,116:VPOKE BASE(8)+18
,1:VPOKE BASE(8)+19,S1'PUT SPRITE 4 SIMULATIE
170 NEXT:GOTO 140
180 REM SPRITE-BOTSING
190 PLAY "S1M3000T255L8C":SPRITE OFF:RETURN
RUN
```

De SPRITE$-TABEL heeft een lengte van 2048 bytes en bevat de

samenstelling van de sprites zoals deze met SPRITE$ (. . .) = zijn bepaald. Bij 8x8 sprites bevat deze tabel maximaal 256 sprites die elk een groepje van 8 bytes beslaan. Bij 16x16 sprites bevat deze tabel maximaal 64 sprites die elk een groepje van 16 bytes beslaan.

MSX-2

De SPRITECOLOR-TABEL heeft onder SCREEN 1 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0.

14.5 De videogeheugen-indeling onder SCREEN 2

De SCHERM-TABEL heeft een lengte van 768 bytes en bevat regel voor regel achter elkaar de grafische informatie van het beeldscherm. De grafische informatie is KARAKTERGEORIENTEERD opgeslagen waarbij elk „karakter" correspondeert met een 8x8 gebiedje op het scherm.

De SCHERM-TABEL is onderverdeeld in drie blokken van 256 bytes. Het eerste blok verwijst naar de eerste 256 maal 8 bytes in de MATRIX-TABEL, het tweede blok verwijst naar de tweede 256 maal 8 bytes in de MATRIX-TABEL en het derde blok verwijst naar de derde 256 maal 8 bytes in de MATRIX-TABEL' De SCHERM-TABEL wordt door MSX-basic vast met drie maal de reeks 0 . . . 255 gevuld.

De MATRIX-TABEL bevat voor elk op het beeldscherm te projekteren karakter een aparte 8 bij 8 bitmatrix (8 bytes). In zo'n groepje van acht bytes is bit voor bit aangegeven, welk puntje de voorgrondkleur dient te krijgen (1) en welk puntje de achtergrondkleur moet hebben.

```
NEW
Ok
10 REM DIT PROGRAMMA LAAT ZIEN DAT DE SCHERMTABEL
20 REM DAADWERKELIJK VERWIJST NAAR DE MATRIXTABEL
21 REM (EN DE KLEURTABEL).
30 REM DOOR DE SCHERMTABEL TE VERANDEREN, WORDT
40 REM HET TOTALE BEELD DOOR ELKAAR GEGOOID. DOOR
50 REM DEZE TABEL DAL WEER TE HERSTELLEN, WORDT
60 REM HET BEELD OOK WEER HERSTELD.
70 SCREEN 2:COLOR 15,4,4:CLS
80 FOR X=0 TO 191 STEP 8:LINE (X,0)-(0,191-X),12:LINE (255-X
,191)-(255,X),7:NEXT X
90 FOR X=0 TO 20:LINE (RND(1)*128+64,RND(1)*46+98)-(RND(1)*1
28+64,RND(1)*46+98),RND(1)*16:NEXT X
100 OPEN "GRP:" AS 1:PRESET (80,48):PRINT #1,"MSX BEELDTEST"
:CLOSE
```

```
110 LINE (64,142)-(192,98),15,B
120 LINE (60,146)-(196,94),15,B
130 PAINT (62,144)
140 FOR I=0 TO 767:VPOKE BASE(10)+I,(I*7) MOD 256:NEXT I
150 FOR I=0 TO 767:VPOKE BASE(10)+I,I MOD 256:NEXT I
160 GOTO 140
RUN
```

De KLEUR-TABEL loopt parallel aan de MATRIX-TABEL. Elk byte in de MATRIX-TABEL correspondeert met een byte in de KLEUR-TABEL. Beide tabellen hanteren dezelfde volgorde.

Elke byte in de MATRIX-TABEL bevat de informatie voor acht naast elkaar gelegen beeldschermpunten. Het bijbehorende byte in de tabel bevat voor deze acht punten een voorgrondkleur en een achtergrondkleur. De waarde van elk byte in deze tabel is gelijk aan 16 maal de voorgrondkleur + de achtergrondkleur.

```
NEW
Ok
10 REM HET VOLGENDE VOORBEELD KLEURT, GEBRUIK
20 REM MAKEND VAN DE KLEURTABEL EN DE MATRIX-
30 REM TABEL, HET BEELDSCHERM WILLEKEURIG IN.
40 REM MOOI IS TE ZIEN, HOE HET BEELD IN
50 REM STROKEN VAN 8 PUNTEN IS VERDEELD
60 REM WAARBINNEN PER BEELDLIJN SLECHTS
70 REM TWEE VERSCHILLENDE KLEUREN KUNNEN
80 REM BESTAAN.
90 A=170:SCREEN 2:COLOR ,,1:FOR I=0 TO 6143
100 VPOKE BASE(12)+I,A:VPOKE BASE(11)+I,RND(1)*255:A=A
XOR 255
110 NEXT I
120 GOTO 120
RUN
```

De TRANSPARANT-TABEL en de SPRITE$-TABEL hebben dezelfde funktie als onder SCREEN 1.

MSX-2

De SPRITECOLOR-TABEL heeft onder SCREEN 2 geen funktie. De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0.

14.6 Formules SCREEN 2

Elk beeldschermpuntje wordt vertegenwoordigd door een bit binnen de MATRIX-TABEL dat op 1 of 0 staat al naar gelang het puntje ge-

'set' of ge'reset' is.

Bovendien wordt elk puntje in de KLEUR-TABEL vertegenwoordigd door een byte dat de voor- en achtergrondkleur van dit punt (samen met nog 7 andere) aangeeft.

Met de volgende formule kan een relatief video-geheugenadres worden bepaald: (X en Y zijn de coördinaten van het te onderzoeken punt)

```
BY=8*VPEEK(BASE(10)+X\8+(Y\8)*32)+2048*((X\8+(Y\8)*32)\256)+Y MOD 8
```

Opgeteld bij BASE (11) geeft deze waarde het adres in video-RAM aan van het byte waarin de kleur staat bepaald. Opgeteld bij BASE (12) geeft deze waarde het adres in video-RAM aan van het byte waarin het betreffende punt als bit is vertegenwoordigd.

In dat geval geeft

```
BI=X MOD 8
```

aan, welk bit (0, 1, 2 . . . 7) correspondeert met het geadresseerde puntje.

```
VPEEK(BASE(12)+BY) AND 2^(7-BI)
```

geeft een 0 of 1 aan naar gelang het beeldpunt ge'set' of ge'reset' is.

```
VPEEK(BASE(11)+BY)
```

geeft de inhoud van het byte waarin de kleur is gekodeerd.

```
VPOKE BASE(11)+BY,(VPEEK(BASE(11)+BY) AND 240) OR A
```

zet A als achtergrondkleur.

```
VPOKE BASE(11)+BY,(VPEEK(BASE(11)+BY) AND 15) OR 16*V
```

zet V als voorgrondkleur.

```
VPOKE BASE(11)+BY,16*V+A
```

zet V en A als voor- en achtergrondkleur.

```
VPOKE BASE(12)+BY,VPEEK(BASE(12)+BY) OR 2^(7-BI)
```

plaatst een puntje op beeld.

```
VPOKE BASE(12)+BY,VPEEK(BASE(12)+BY) AND (255-2^(7-BI))
```

verwijdert dit puntje weer.

```
VPOKE BASE(12)+BY,VPEEK(BASE(12)+BY) XOR 2^(7-BI)
```

inverteert het puntje.

```
NEW
Ok
10 REM IN DIT VOORBEELD WORDT EEN CIRCEL GETEKEND
20 REM DOOR DEZE PUNT VOOR PUNT UIT TE REKENEN
30 REM EN DAN VIA DE HIERBOVEN BEPAALDE FORMULE
40 REM IN HET VIDEO RAM TE PLAATSEN.
50 REM
60 COLOR 15,4,4
70 SCREEN 2:CLS
80 FOR I=0 TO 6.29 STEP .02:X=100+INT(50*COS(I)+.5):Y=100+IN
T(50*SIN(I)+.5)
90 BY=8*VPEEK(BASE(10)+X\8+(Y\8)*32)+2048*((X\8+(Y\8)*32)\25
6)+Y MOD 8
100 BI=X MOD 8
110 VPOKE BASE(12)+BY,VPEEK(BASE(12)+BY) OR 2^(7-BI):VPOKE B
ASE(11)+BY,(VPEEK(BASE(11)+BY) AND 15) OR 240
120 NEXT I
130 GOTO 130
RUN
```

14.7 De videogeheugen-indeling onder SCREEN 3

De SCHERM-TABEL heeft een lengte van 768 bytes en bevat regel voor regel achter elkaar de informatie van het beeldscherm. De grafische informatie is KARAKTERGEORIENTEERD opgeslagen waarbij elk "karakter" correspondeert met een 2 bij 2 gebiedje op het beeldscherm.

Elk byte in de SCHERM-TABEL verwijst naar een eenheid van twee bytes in de MATRIX-TABEL volgens de relatie:

SCHERM-TABEL-byte maal 8 + regelnummer MOD 4 = adres in MATRIX-TABEL.

De SCHERM-TABEL heeft onder MSX-basic een vaste vulling met vier

reeksen van 0 . . . 31, vier reeksen 32 . . . 63, vier reeksen 64 . . . 95, vier reeksen 96 . . . 127, vier reeksen 128 . . . 159 en vier reeksen 160 . . . 191.

De MATRIX-TABEL bevat per af te drukken karakter twee bytes. De byte-kodering is als volgt:

De KLEUR-TABEL heeft onder SCREEN 3 geen funktie. De TRANSPARANT-TABEL en de SPRITE$-TABEL hebben een funktie zoals onder SCREEN 1 beschreven.

MSX-2

De SPRITECOLOR-TABEL heeft onder SCREEN 3 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als besproken onder SCREEN 0.

14.8 Formules SCREEN 3

Het relatieve adres van een byte binnen MATRIX-TABEL waarin het betreffende punt is gekodeerd, laat zich als volgt berekenen: (X en Y zijn de coördinaten van het punt)

```
BY=VPEEK(BASE(15)+X\8+(Y\8)*32)*8+(Y\4) MOD 8
```

Het absolute adres binnen VRAM is natuurlijk BASE(17) + BY.

Binnen het geadresseerde byte zijn twee punten gekodeerd, één in het linker- en één in het rechtergedeelte van het byte.

```
(VPEEK(BASE(17)+BY) AND (15*(16-15*((X\4) MOD 2))))\(16- ...
                                          ... 15*((X\4) MOD 2))
```

geeft de kleurkode van het betreffende puntje aan.

```
VPOKE BASE(17)+BY,(VPEEK(BASE(17)+BY) AND 15*(1+15*((X\4) MOD 2))) ...
                                        ... OR C*(16-15*((X\4) MOD 2))
```

zet het punt op kleurkode C.

```
NEW
Ok
10 REM IN DIT VOORBEELD WORDT EEN CIRCEL GETEKEND
20 REM DOOR DEZE PUNT VOOR PUNT UIT TE REKENEN
30 REM EN DAN VIA DE HIERBOVEN BEPAALDE FORMULE
40 REM IN HET VIDEO RAM TE PLAATSEN.
50 REM ECHTER, VOOR DIT ALLES WORDT EERST DE
60 REM SCHERM-TABEL DOOR DE WAR GEMAAKT. NADAT
70 REM DE CIRCEL IS GETEKEND, WORDT DE SCHERM-
80 REM TABEL ACHTEREENVOLGENS HERSTELD EN WEER
90 REM DOOR DE WAR GEGOOID.
100 REM MERK OP DAT OF ALLEEN DE CIRCEL OF ALLEEN
110 REM DE ACHTERGROND GOED GEPROJEKTEERD WORDT.
120 REM DE PAINT OP REGEL 190 BEWIJST DAT HET
130 REM MSX-BASIC ZELF DE SCHERM-TABEL NIET
140 REM RAADPLEEGT ! (DIT GELDT TROUWENS OOK
150 REM VOOR DE SCREEN 2 INSTELLING.)
160 REM
170 COLOR 15,4,4
180 SCREEN 3:CLS
190 LINE (20,20)-(235,172),1,B
200 LINE (40,40)-(215,152),1,B
210 FOR I=0 TO 767:VPOKE BASE(15)+I,(I*7 MOD 32)+(I\128)*32:
NEXT I
220 PAINT (30,30),12,1
230 FOR I=0 TO 6.29 STEP .08:X=100+INT(50*COS(I)+.5):Y=100+I
NT(50*SIN(I)+.5)
240 BY=VPEEK(BASE(15)+X\8+(Y\8)*32)*8+(Y\4) MOD 8
250 VPOKE BASE(17)+BY,VPEEK(BASE(17)+BY) OR 15*(16-15*((X\4)
 MOD 2))
260 NEXT I
270 FOR I=0 TO 767:VPOKE BASE(15)+I,(I MOD 32)+(I\128)*32:NE
XT I
280 FOR I=0 TO 767:VPOKE BASE(15)+I,(I*7 MOD 32)+(I\128)*32:
NEXT I
290 GOTO 270
300 FOR I=0 TO 767:PRINT((I MOD 32)+(I\128)*32)*7 MOD (32+(I
\128)*32):NEXT I
RUN
```

14.9 MSX-1 en MSX-2

In de volgende paragrafen worden de scherminstellingen SCREEN 4 tot en met SCREEN 8 behandeld. Deze schermindelingen zijn alleen onder MSX-2 beschikbaar.

De volgende paragrafen, tot aan de behandeling van de VDP-registers, zijn dan ook alleen van toepassing voor een MSX-2 machine.

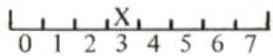
14.9 De videogeheugen-indeling onder SCREEN 4

De SCHERM-TABEL, de MATRIX-TABEL en de KLEUR-TABEL hebben dezelfde funktie als onder SCREEN 2. De SCREEN 4 - instelling is grafisch gezien dan ook gelijk aan de SCREEN 2 - instelling met dit verschil dat er op het gebied van sprites een afwijking is.

De TRANSPARANT-TABEL heeft een lengte van 128 bytes. Per transparant bevat deze tabel vier bytes.
In het eerste byte is de Y-coördinaat van de op de transparant geplaatste sprite opgenomen. In het tweede byte is de bijbehorende X-coördinaat opgenomen. Het derde byte bevat het spritenummer van de afgebeelde sprite. Het vierde byte heeft geen funktie.

De SPRITECOLOR-TABEL bevat de kleurinformatie per transparant. Voor elke transparant zijn er in de SPRITECOLOR-TABEL 16 bytes gereserveerd. Voor 8 bij 8 sprites zijn alleen de eerste 8 bytes van deze 16 bytes van belang.

Een 8 bij 8 sprite kan men opgedeeld zien in 8 horizontale lagen. Een 16 bij 16 sprite kan men op dezelfde manier opgedeeld zien in 16 lagen. In elk byte in de SPRITECOLOR-TABEL is de kleur van één zo'n spritelaag gekodeerd. Behalve de kleurkode (0 . . . 15) kunnen er per spritelaag ook nog wat andere funkties worden gekodeerd. Het "kleurbyte" ziet er als volgt uit:



bit 4 . . . 7 bevatten de kleurkode (0 t/m 15)

bit 0: =0 normale situatie
=1 spritelaag wordt 32 beeldpunten naar links afgebeeld. Links uitgeschoven sprites of sprite-delen worden niet rechts geprojekteerd.
Wanneer er onder MSX-basic een X-coördinaat, kleiner dan nul wordt opgegeven, dan wordt dit bit automatisch op 1 gesteld en wordt de X-coördinaat daaraan aangepast.

Zo ontstaat links in het beeld een spelling van 32 beeldschermpunten waarin een sprite kan verdwijnen zonder onmiddellijk rechts weer te verschijnen.

bit 1: =0 normale situatie
=1 de spritelaag wordt niet geprojekteerd behalve over de beeldlijnen waarin deze met een voorliggend sprite-gedeelte samenvalt waarvan dit byte niet op 1 staat. Eventuele overlappende delen worden in kleurkode logisch geORed.
Sprite-botsingen worden met betrekking tot deze spritelaag niet gedetecteerd en kunnen met ON SPRITE GOSUB dan ook niet worden afgevraagd.

bit 2: =0 normale situatie
=1 sprite-botsingen worden met betrekking tot deze spritelaag niet meer gedetecteerd en kan met ON SPRITE GOSUB dan ook niet worden afgevraagd.

bit 3: geen betekenis.

```
NEW
Ok
10 REM IN DIT VOORBEELD WORDT GETOOND
20 REM DAT BITS 0 EN 1 INDERDAAD DE
30 REM BESPROKEN EFFEKTEN HEBBEN IN DE
40 REM KLEURKODERING VAN DE SPRITE.
50 REM
60 S0=&B10011100'SPRITE 0,KLEUR 12, BIT 0 AAN
70 S1=&B01010011'SPRITE 1,KLEUR 3, BIT 1 AAN
80 ON SPRITE GOSUB 160
90 SCREEN 4,3
100 COLORSPRITE$(2)=STRING$(32,S0):COLORSPRITE$(4)=STRING$(3
2,S1)
110 SPRITE$(0)=STRING$(32,255):SPRITE$(1)=SPRITE$(0)
120 PUTSPRITE 2,(132,100),,0
130 SPRITE ON:FOR I=0 TO 191
140 PUTSPRITE 4,(116,I),,1
150 NEXT:GOTO 120
160 REM SPRITE-BOTSING
170 STOP
RUN
```

De SPRITES-TABEL heeft dezelfde funktie als onder SCREEN 1 en de PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0.

14.10 De videogeheugen-indeling onder SCREEN 5

SCREEN 5 is bit-mapped. Elk groepje van vier bits in de SCHERM-TABEL correspondeert met één enkel te projekteren beeldschermpuntje. De volgorde van de groepjes van 4 bits in de SCHERM-TABEL komt overeen met de projektievolgorde van de puntjes op het beeldscherm (beeldlijn voor beeldlijn van links naar rechts).

Binnen elk groepje van 4 bits is de kleurkodering van het puntje vastgelegd. De kleurkode van elk puntje kan dus variëren van 0 tot en met 15.

De SCHERM-TABEL is 256 x 212 / 2 = 27136 bytes groot.

De KLEUR-TABEL en de MATRIX-TABEL hebben onder SCREEN 5 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0. De SPRITE$-TABEL heeft dezelfde funktie als onder SCREEN 1. De SPRITECOLOR-TABEL en de TRANSPARANT-TABEL hebben dezelfde funktie als onder SCREEN 4.

14.11 Formules SCREEN 5

Van beeldpunt (X,Y) geeft de volgende formule de kleurkode:

```
(VPEEK(BASE(25)+INT((X+Y*256)/2)) AND (15*(16-15*(X MOD 2)))) ...
                                          ... \(16-15*(X MOD 2))
```

Beeldpunt (X,Y) kan met de volgende formule op kleurkode C gezet worden:

```
VPOKE BASE(25)+INT((X+Y*256)/2),(VPEEK(BASE(25)+INT((X+256*Y) ...
      ... /2)) AND (15*(1+15*(X MOD 2)))) OR C*(16-15*(X MOD 2))
```

```
NEW
Ok
10 REM DIT VOORBEELD TOONT DE JUISTHEID
20 REM VAN BOVENSTAANDE FORMULES.
30 REM
40 SCREEN 5
50 X=INT(RND(1)*256)
60 Y=INT(RND(1)*212)
70 C=INT(RND(1)*16)
80 VPOKE BASE(25)+INT((X+Y*256)/2),(VPEEK(BASE(25)+INT((X+Y*
256)/2)) AND (15*(1+15*(X MOD 2)))) OR C*(16-15*(X MOD 2))
90 IF POINT (X,Y)<>C THEN 130
100 Q=(VPEEK(BASE(25)+INT((X+Y*256)/2)) AND (15*(16-15*(X MO
D 2))))\(16-15*(X MOD 2))
```

```
110 IF Q<>C THEN 130
120 GOTO 50
130 SCREEN 0:PRINT "FOUT IN FORMULE":STOP
RUN
```

14.12 Paging SCREEN 5

Het video-geheugen kan men zich onder SCREEN 5 voorstellen als zijnde twee (64KB) of vier (128KB) banken van 32 KB. Page 0 bevindt zich in de eerste bank, page 1 in de tweede, page 2 in de derde en page 3 in de vierde.
De BASE-pointer naar de SCHERM-TABEL verwijst altijd naar het actieve scherm in de eerste bank. Indien met SET PAGE een ander actief scherm wordt geselecteerd, dan wordt de bijbehorende geheugenbank gewisseld met de eerste bank teneinde het actieve scherm altijd in de laagste bank te houden.

14.13 De videogeheugen-indeling onder SCREEN 6

SCREEN 6 is bit-mapped. Elk groepje van twee bits in de SCHERM-TABEL correspondeert met één enkel te projekteren beeldschermpuntje. De volgorde van de groepjes van 2 bits in de SCHERM-TABEL komt overeen met de projektievolgorde van de puntjes op het beeldscherm (beeldlijn voor beeldlijn van links naar rechts).

Binnen elk groepje van twee bits is de kleurkodering van het puntje vastgelegd. De kleurkode van elk puntje kan dus variëren van 0 t/m 3.

De SCHERM-TABEL is 512 x 212 /4 = 27136 bytes groot.

De KLEUR-TABEL en de MATRIX-TABEL hebben onder SCREEN 6 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0. De SPRITE$-TABEL heeft dezelfde funktie als onder SCREEN 1. De SPRITECOLOR-TABEL en de TRANSPARANT-TABEL hebben dezelfde funktie als onder SCREEN 4.

14.14 Formules SCREEN 6

Van beeldpunt (X,Y) geeft de volgende formule de kleurkode:

```
(VPEEK(BASE(30)+INT((X+512*Y)/4)) AND (3*4^(3-X MOD 4))) ...
                                        ... \(4^(3-X MOD 4))
```

Beeldpunt (X,Y) kan met de volgende formule op kleurkode C gezet worden:

```
VPOKE BASE(30)+INT((X+512*Y)/4),(VPEEK(BASE(30)+INT((X+512*Y) ...
          ... /4)) AND (255-3*4^(3-X MOD 4))) OR C*4^(3-X MOD 4)
```

```
NEW
Ok
10 REM DIT VOORBEELD TOONT DE JUISTHEID
20 REM VAN BOVENSTAANDE FORMULES.
30 REM
40 SCREEN 6
50 X=INT(RND(1)*512)
60 Y=INT(RND(1)*212)
70 C=INT(RND(1)*4)
80 VPOKE BASE(30)+INT((X+Y*512)/4),(VPEEK(BASE(30)+INT((X+Y*
512)/4)) AND (255-3*4^(3-X MOD 4))) OR C*4^(3-X MOD 4)
90 IF POINT (X,Y)<>C THEN 130
100 Q=(VPEEK(BASE(30)+INT((X+Y*512)/4)) AND (3*4^(3-X MOD 4)
))\(4^(3-X MOD 4))
110 IF Q<>C THEN 130
120 GOTO 50
130 SCREEN 0:PRINT "FOUT IN FORMULE":STOP
RUN
```

14.15 Paging SCREEN 6

De paging onder SCREEN 6 is identiek aan de paging onder SCREEN 5.

14.16 De videogeheugen-indeling onder SCREEN 7

SCREEN 7 is bit-mapped. Elk groepje van vier bits in de SCHERM-TABEL correspondeert met één enkel te projekteren beeldschermpuntje. De volgorde van de groepjes van 4 bits in de SCHERM-TABEL komt overeen met de projektievolgorde van de puntjes op het beeldscherm (beeldlijn voor beeldlijn van links naar rechts).

Binnen elk groepje van 4 bits is de kleurkodering van het puntje vastgelegd. De kleurkode van elk puntje kan dus variëren van 0 tot en met 15.

De SCHERM-TABEL is 512 x 212 / 2 = 54272 bytes groot.

De KLEUR-TABEL en de MATRIX-TABEL hebben onder SCREEN 7 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0. De SPRITE$-TABEL heeft dezelfde funktie als onder SCREEN 1. De SPRITECOLOR-TABEL en de TRANSPARANT-TABEL hebben

dezelfde funktie als onder SCREEN 4.

14.17 Formules SCREEN 7

Van beeldpunt (X,Y) geeft de volgende formule de kleurkode:

```
(VPEEK(BASE(35)+INT((X+Y*512)/2)) AND (15*(16-15*(X MOD 2)))) ...
                                          ... \(16-15*(X MOD 2))
```

Beeldpunt (X,Y) kan met de volgende formule op kleurkode C worden gezet:

```
VPOKE BASE(35)+INT((X+512*Y)/2),(VPEEK(BASE(35)+INT((X+Y*512)/2)) ...
               ... AND (15*(1+15*(X MOD 2)))) OR C*(16-15*(X MOD 2))
```

```
NEW
Ok
10 REM DIT VOORBEELD TOONT DE JUISTHEID
20 REM VAN BOVENSTAANDE FORMULES.
30 REM
40 SCREEN 7
50 X=INT(RND(1)*512)
60 Y=INT(RND(1)*212)
70 C=INT(RND(1)*16)
80 VPOKE BASE(35)+INT((X+Y*512)/2),(VPEEK(BASE(35)+INT((X+Y*
512)/2)) AND (15*(1+15*(X MOD 2)))) OR C*(16-15*(X MOD 2))
90 IF POINT (X,Y)<>C THEN 130
100 Q=(VPEEK(BASE(35)+INT((X+Y*512)/2)) AND (15*(16-15*(X MO
D 2))))\(16-15*(X MOD 2))
110 IF Q<>C THEN 130
120 GOTO 50
130 SCREEN 0:PRINT "FOUT IN FORMULE":STOP
RUN
```

14.18 Paging SCREEN 7

Het video-geheugen kan met zich onder SCREEN 7 voorstellen als zijnde twee banken van 64KB elk. Page 0 bevindt zich in de eerste bank en Page 1 bevindt zich in de tweede. De BASE-pointer naar de SCHERM-TABEL verwijst altijd naar het actieve scherm in de eerste bank. Indien met SET PAGE een ander actief scherm wordt geselekteerd, dan wordt de bijbehorende geheugenbank gewisseld met de eerste bank teneinde het actieve scherm altijd in de laagste bank te houden.

14.19 De videogheugen-indeling onder SCREEN 8

SCREEN 8 is bit-mapped. Elk byte in de SCHERM-TABEL corres-

pondeert met één enkel te projekteren beeldschermpuntje. De volgorde van de bytes in de SCHERM-TABEL komt overeen met de projektie-volgorde van de puntjes op het beeldscherm (beeldlijn voor beeldlijn van links naar rechts).

In elk byte is de kleurkodering van het puntje vastgelegd. De kleurkode van elk puntje kan dus variëren van 0 tot en met 255.

De SCHERM-TABEL is 256 x 212 = 54272 bytes groot.

De KLEUR-TABEL en de MATRIX-TABEL hebben onder SCREEN 8 geen funktie.

De PALETTE-TABEL heeft dezelfde funktie als onder SCREEN 0. Deze tabel bepaalt echter alleen de kleur-samenstellingen voor de eventueel gebruikte sprites.

De SPRITES-TABEL heeft dezelfde funktie als onder SCREEN 1. De SPRITECOLOR-TABEL en de TRANSPARANT-TABEL hebben dezelfde funktie als onder SCREEN 4.

14.20 Formules SCREEN 8

Van beeldpunt (X,Y) geeft de volgende formule de kleurkode:

```
VPEEK(BASE(40)+X+256*Y)
```

Beeldpunt (X,Y) kan met behulp van de volgende formule op kleurkode C gezet worden:

```
VPOKE BASE(40)+X+256*Y,C
```

```
NEW
Ok
10 REM DIT VOORBEELD TOONT DE JUISTHEID
20 REM VAN BOVENSTAANDE FORMULES.
30 REM
40 SCREEN 8
50 X=INT(RND(1)*256)
60 Y=INT(RND(1)*212)
70 C=INT(RND(1)*256)
80 VPOKE BASE(40)+X+256*Y,C
90 IF POINT (X,Y)<>C THEN 130
100 Q=VPEEK(BASE(40)+X+256*Y)
110 IF Q<>C THEN 130
```

```
120 GOTO 50
130 SCREEN 0:PRINT "FOUT IN FORMULE":STOP
RUN
```

14.21 Paging SCREEN 8

De paging onder SCREEN 8 is identiek aan de paging onder SCREEN 7.

14.22 De VDP-registers

De Video Display Processor wordt bestuurd met behulp van registers. Er zijn verschillende soorten registers die per stuk in de volgende paragrafen worden behandeld.

De registers die op de register-schema's een aparte MSX-1 vermelding hebben, zijn van toepassing op MSX-1 en MSX-2 computers. Alle andere registers zijn alleen onder MSX-2 van toepassing.

Vele registers verstrekken of vereisen specifiek hardware-technische gegevens of gegevens die alleen zin hebben wanneer men in machinetaal programmeert.

De VDP-registers worden genoemd en in enkele voorbeelden globaal behandeld. Voor nadere informatie dient specialistische literatuur te worden geraadpleegd.

Omdat specifieke literatuur met betrekking tot dit onderwerp naar aller waarschijnlijkheid alleen in het engels te verkrijgen is, zijn de schema's in de volgende paragrafen voorzien van de originele engelstalige termen.
Deze termen zijn ontleend aan de VDP system specification van ASCII-MICROSOFT en op sommige plaatsen aangevuld of gecorrigeerd.

14.23 De VDP mode-registers

In de mode-registers wordt de "toestand" van de VDP-chip gekodeerd. Ondermeer wordt in deze registers vastgelegd, welke SCREEN instelling actief is.
Er zijn 4 mode-register, te weten:

VDP(...)	register	BIT ø	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
ø	ø	ø	DG	IEø	IE1	M5	M4	M3	(D)
1	1	X	blank	IEø	M1	M2	ø	size	MAG
9	8	mouse	LCRS	TP	CBD	VRS1	VRSø	SPD	B/W
1ø	9	LN	ø	SYM1	SYMø	IL	E/O	NTSC	DCD

 = ook MSX-1  = alleen MSX-1

DG = digitze mode
IEØ = vertical retrace interrupt enable
IE1 = horizontal retrace interrupt enable
IE2 = light pen or mouse interupt enable
MAG = magnify of sprite patterns: Ø= x1, 1 = x2
SIZE = size of sprites: Ø = 8x8, 1 = 16x16
BLANK = enable (1) or disable (Ø) display
B/W = black and white mode, 32 gradations composite video output
SPD = sprite disable mode
M1...M5 = define mode:

M5	M4	M3	M2	M1	SCREEN
Ø	Ø	Ø	Ø	Ø	SCREEN 1
Ø	Ø	Ø	Ø	1	SCREEN Ø (4Ø)
Ø	Ø	Ø	1	Ø	SCREEN 3
Ø	Ø	1	Ø	Ø	SCREEN 2
Ø	1	Ø	Ø	Ø	SCREEN 4
Ø	1	Ø	Ø	1	SCREEN Ø (8Ø)
Ø	1	1	Ø	Ø	SCREEN 5
1	Ø	Ø	Ø	Ø	SCREEN 6
1	Ø	1	Ø	Ø	SCREEN 7
1	1	1	Ø	Ø	SCREEN 8

= ook MSX-1

D = external VDP-input (Ø/1 = ON/OFF)
MOUSE = mouse or light pen mode (Ø/1 = light pen/mouse interface)
LCRS = light pen registration/coincidence registration select (Ø/1= coincidence/light pen)
TP = transparent mode. Ø/1 = pallette code Ø becomes/not transparent
CBD = colorbus direction: Ø/1 = output/input

CBD	DG	Color-bus	Aplication
Ø	Ø	output	external encoder & external palette
1	Ø	input	mouse
Ø	1	input!	digitzing
1	1	input	digitzing

VRS Ø/VRS 1 = VIDEO RAM SELECT

VRS		RAM
1	Ø	
Ø	Ø	1x16KB
Ø	1	4x16KB
1	Ø	1x64KB
1	1	64KB high speed

SYM Ø/SYM 1 = synchronization mode

SYM		application mode
Ø	1	
Ø	Ø	internal
Ø	1	mixing
1	Ø	external video or digitize
1	1	none

IL = interlace mode. Ø = non-interlace, 1 = interlace and generate complete interlace timing

E/O = even/odd. Ø = normal display, 1 = even/odd screen alternative display

NTSC = NTSC or PAL/SECAM mode select (Ø/1 = NTSC/PAL).

DCD = dot clock direction (Ø/1 = dot clock input/output mode).

```
NEW
Ok
10 REM GEBRUIK VAN DE BLANK-FLAG
20 REM OM HET HELE SCHERM UIT EN AAN
30 REM TE ZETTEN.
40 REM
50 VDP(1)=VDP(1) XOR 64'INVERTEER BLANK-FLAG
60 FOR I=1 TO 100:NEXT:GOTO 50
RUN
```

```
NEW
Ok
10 REM GEBRUIK VAN DE MAG-FLAG
20 REM OM REEDS GEPLAATSTE SPRITES
30 REM VERGROOT OF NIET VERGROOT TE MAKEN.
40 REM
50 SCREEN 1,2
60 SPRITE$(0)=STRING$(32,255)
70 FOR I=0 TO 31
80 PUT SPRITE I,(RND(1)*256,RND(1)*192),RND(1)*16,0
90 NEXT
100 VDP(1)=VDP(1) XOR 1'INVERTEER MAG-FLAG
110 FOR I=1 TO 100:NEXT:GOTO 100
120 GOTO 100
RUN
```

```
NEW
Ok
10 REM GEBRUIK VAN DE CBD-FLAG OM
20 REM HET BEELDSCHERM AAN EN UIT
30 REM TE SCHAKELEN.
40 REM
50 VDP(9)=VDP(9) XOR 16'INVERTEER CBD-FLAG
60 FOR I=1 TO 100:NEXT:GOTO 50
RUN
```

```
NEW
Ok
10 REM GEBRUIK VAN DE SPD-FLAG
20 REM OM SPRITES AL DAN NIET TE
30 REM PLAATSEN.
40 REM
```

```
50 SCREEN 1,2
60 SPRITE$(0)=STRING$(32,255)
70 FOR I=0 TO 31
80 PUT SPRITE I,(RND(1)*256,RND(1)*192),RND(1)*16,0
90 NEXT
100 VDP(9)=VDP(9) XOR 2'INVERTEER SPD-FLAG
110 FOR I=1 TO 100:NEXT:GOTO 100
RUN
```

```
NEW
Ok
10 REM TV STORINGSIMULATIE DOOR
20 REM VERWISSELING NTSC-PAL MODE
30 REM
40 SCREEN 3
50 PRESET(25,70):OPEN "GRP:" AS 1
60 PRINT #1,"STORING"
70 VDP(10)=VDP(10) XOR 2'INVERTEER NTSC-FLAG
80 IF INKEY$="" THEN 80 ELSE VDP(10)=VDP(10) XOR 2'EN WEER T
ERUG
RUN
```

```
NEW
Ok
10 REM TV STORINGSIMULATIE DOOR
20 REM IN- EN UITSCHAKELEN VAN DE
30 REM SYNCHRONISATIE.
40 REM
50 SCREEN 3
60 PRESET(25,70):OPEN "GRP:" AS 1
70 PRINT #1,"STORING"
80 VDP(10)=VDP(10) XOR 48'INVERTEER SYM0-SYM1
90 GOTO 80
RUN
```

14.24 De VDP-adres-registers.

In de adresregisters worden de adressen van de actieve tabellen in het video-geheugen gekodeerd. Ondermeer de actieve BASE-adressen zijn in deze registers te vinden.

Er zijn 8 adres-registers, te weten:

VDP(...)	register	bit Ø	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
2	2	Ø	A16	A15	A14	A13	A12	A11	A1Ø
3	3	B13	B12	B11	B1Ø	B9	B8	B7	B6
11	1Ø	Ø	Ø	Ø	Ø	Ø	B16	B15	B14
4	4	Ø	Ø	C16	C15	C14	C13	C12	C11
5	5	D14	D13	D12	D11	D1Ø	D9	D8	D7
12	11	Ø	Ø	Ø	Ø	Ø	Ø	D16	D15
6	6	Ø	Ø	E16	E15	E14	E13	E12	E11
15	14	Ø	Ø	Ø	Ø	Ø	F16	F15	F14

= ook MSX-1

A16 . . . A10 = name table (SCHERM-TABEL) bar/1024*
B16 . . . B6 = color table (KLEUR-TABEL) bar/64
C16 . . . C11 = pattern-generator table (MATRIX-TABEL)/2048
D16 . . . D7 = sprite attribute table (TRANSPARANT-TABEL)/128
E16 . . . E11 = sprite pattern generator table (SPRITE$-TABEL)/2048
F16 . . . F14 = Video RAM acces bar

*SCREEN Ø (8Ø): A16 . . A12 = name table address/1024. A11 has no meaning and A10 =offset/1024 for second 1024 bytes of screen.

14.25 De VDP tekstcontrole-registers

In de tekstcontrole-registers worden enige zaken met betrekking tot SCREEN Ø . . . SCREEN 3 gekodeerd. Er zijn drie tekstcontrole-registers, te weten:

VDP(...)	register	bit Ø	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
7	7	TC3	TC2	TC1	TCØ	BCD3	BCD2	BCD1	BCDØ
13	12	C3	C2	C1	CØ	BC3	BC2	BC1	BCØ
14	13	ON3	ON2	ON1	ONØ	OF3	OF2	OF1	OFØ

= ook MSX-1

TC3 . . . TCØ = text color
BCD3 . . . BCDØ = back drop color
C3 . . . CØ = tekst color
BC3 . . . BCØ = back color
ON3 . . . ONØ = blink period/5 sec ON
OF3 . . . OFØ = blink period/5 sec OFF

} SCREEN Ø (8Ø)

Zie het voorbeeld, opgenomen onder de behandeling van de COLOR-TABEL onder SCREEN Ø

14.26 Pointers en controle-registers

In deze registers worden enige controle-waarden en pointer-waarden gekodeerd. Er zijn 8 pointers en controle-registers, te weten:

VDP(−)	register	bit Ø	bit 1	bit 2	bit 3	bit 4	bit 5	bit6	bit 7
16	15	Ø	Ø	Ø	Ø	RS3	RS2	RS1	RSØ
17	16	Ø	Ø	Ø	Ø	C3	C2	C1	CØ
18	17	Ø	Ø	RS5	RS4	RS3	RS2	RS1	RSØ
19	18	ΔV3	ΔV2	ΔV1	ΔVØ	ΔH3	ΔH2	ΔH1	ΔHØ
2Ø	19	IL7	IL6	IL5	IL4	IL3	IL2	IL1	ILØ
21	2Ø	Ø	Ø	CBX5	CBX4	CBX3	CBX2	CBX1	CBXØ
22	21	Ø	Ø	CBY5	CBY4	CBY3	CBY2	CBY1	CBYØ
23	22	Ø	Ø	CBZ5	CBZ4	CBZ3	CBZ2	CBZ1	CBZØ

RS3 . . . RSØ = register-number of status-register
C3 . . . CØ = relative palette address
RS5 . . . RSØ = register-number of control-register
ΔV3 . . . ΔVØ = vertical display adjust value
ΔH3 . . . ΔHØ = horizontal display adjust value
IL7 . . . ILØ = vertical line number of line interrupt
CBX5 . . . CBXØ = color burst value of phase Ø
CBY5 . . . CBYØ = color burst value of phase 1/3
CBZ5 . . . CBZØ = color burst value of phase 2/3

```
NEW
Ok
10 REM SET ADJUST ZONDER VASTLEGGING
20 REM IN PERMANENT GEHEUGEN M.B.V.
30 REM VDP REGISTER 19.
40 REM PIJLEN/RETURN=EINDE
50 REM
60 X=0:Y=0
70 K$=INKEY$:IF K$="" THEN 70
80 K=ASC(K$):IF K=13 THEN STOP
90 IF K=31 THEN Y=Y-1
100 IF K=30 THEN Y=Y+1
110 IF K=29 THEN X=X+1
120 IF K=28 THEN X=X-1
130 IF X<-8 THEN X=-8 ELSE IF X>7 THEN X=7
140 IF Y<-8 THEN Y=-8 ELSE IF Y>7 THEN Y=7
150 X1=X:Y1=Y
160 IFX1<0 THEN X1=X1+16
170 IFY1<0 THEN Y1=Y1+16
180 VDP(19)=16*Y1+X1
190 GOTO 70
RUN
```

14.27 De VDP programma-registers.

De Video Display Processor is een computer op zich. De VDP laat zich met behulp van de volgende 15 registers programmeren:

VDP(...)	register	bit Ø	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
33	32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SXØ
34	33	Ø	Ø	Ø	Ø	Ø	Ø	Ø	SX8
35	34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SYØ
36	35	Ø	Ø	Ø	Ø	Ø	Ø	SY9	SY8
37	36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DXØ
38	37	Ø	Ø	Ø	Ø	Ø	Ø	Ø	DX8
39	38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DYØ
4Ø	39	Ø	Ø	Ø	Ø	Ø	Ø	DY9	DY8
41	4Ø	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NXØ
42	41	Ø	Ø	Ø	Ø	Ø	Ø	NX9	NX8
43	42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NYØ
44	43	Ø	Ø	Ø	Ø	Ø	Ø	NY9	NY8
45	44	CH3	CH2	CH1	CHØ	CL3	CL2	CL1	CLØ
46	45	Ø	MXC	MXD	MXS	DIR Y	DIR X	EQ OR NEQ	MAJ MIN
47	46	CM3	CM2	CM1	CMØ	LO3	LO2	LO1	LOØ

SX8 . . . SXØ : source X coordinate
SY9 . . . SYØ : source Y coordinate
DX8 . . . DXØ : destination X coordinate
DY9 . . . DYØ : destination Y coordinate
NX9 . . .NXØ : number of X dots
NY9 . . . NYØ : number of Y dots
CH3 . . . CHØ
CL3 . . . CLØ } colorcode of color-register write
MAJ/MIN : Ø : X axis is major direction
1 : Y axis is major direction
EQ OR NEQ : equal or not equal detect (1=NEQ)

DIR X	: direction of X coordinate
DIR Y	: direction of Y coordinate
MXS	: source becomes external mermory
MXD	: destination becomes external memory
MXC	: CPU-access becomes external memory
CM3 . . . CMØ	: command
LO3 . . . LOØ	: logical operation

NB: deze registers zijn WRITE ONLY. De inhoud van deze registers kan worden veranderd maar is niet opvraagbaar.

VDP-commando's:

MNEMONIC	CM3...CMØ	direction	unit	operation
HMMC	1 1 1 1	CPU to VRAM	byte	perform
HMCM	1 1 1 Ø	VRAM to CPU	byte	a
HMMM	1 1 Ø 1	VRAM to VRAM	byte	high speed
HMMV	1 1 Ø Ø	VDP to VRAM	byte	move
LMMC	1 Ø 1 1	CPU to VRAM	dot	perform
LMCM	1 Ø 1 Ø	VRAM to CPU	dot	a
LMMM	1 Ø Ø 1	VRAM to VRAM	dot	logical
LMMV	1 Ø Ø Ø	VDP to VRAM	dot	move
LINE	Ø 1 1 1	VDP to VRAM	dot	line
SRCH	Ø 1 1 Ø	VRAM to VDP	dot	search
PSET	Ø 1 Ø 1	VDP to VRAM	dot	pset
PINT	Ø 1 Ø Ø	VRAM to VDP	dot	point
NOP	rest	–	–	no operation

Logical operations

MNEMONIC	LO3 . . .LOØ	operation
IMP	Ø Ø Ø Ø	SC → DC
AND	Ø Ø Ø 1	SC AND DC → DC
OR	Ø Ø 1 Ø	SC OR DC → DC
EOR	Ø Ø 1 1	SC XOR DC → DC
NOT	Ø 1 Ø Ø	SC → DC
TIMP	1 Ø Ø Ø	IF SC=Ø AND TP=Ø THEN DC→DC ELSE IMP
TAND	1 Ø Ø 1	IF SC=Ø AND TP=Ø THEN DC→DC ELSE AND
TOR	1 Ø 1 Ø	IF SC=Ø AND TP=Ø THEN DC→DC ELSE OR
TEOR	1 Ø 1 1	IF SC=Ø AND TP=Ø THEN DC→ELSE EOR
TNOT	1 1 Ø Ø	IF SC=Ø AND TP=Ø THEN DC→ELSE NOT
–	rest	illegal

SC = source color
DC = destination color

```
NEW
Ok
10 REM LOGICAL MOVE VRAM -> VRAM
20 REM VIA DIREKTE VDP-AANSTURING.
30 REM
40 COLOR 15,0:SCREEN 5
50 CIRCLE (50,50),49:FOR I=100 TO 0 STEP -5
60 VDP(33)=0:VDP(34)=0:VDP(35)=0:VDP(36)=0'COORDINATEN VAN
70 VDP(37)=I:VDP(38)=0:VDP(39)=I:VDP(40)=0'COORDINATEN NAAR
80 VDP(41)=100:VDP(42)=0:VDP(43)=100:VDP(44)=0'AANTAL PUNTEN
90 VDP(47)=152'VDP-KOMMANDO: LMMM TIMP
100 IF (VDP(-2) AND 1)=1 THEN 100'READY ?
110 NEXT I
RUN
```

```
NEW
Ok
10 REM OPHALEN KLEURKODE PUNT D.M.V.
20 REM DIREKTE VDP-PROGRAMMERING.
30 REM
40 SCREEN 5:COLOR ,1,1:CLS
50 X=INT(RND(1)*256):Y=INT(RND(1)*212):C=INT(RND(1)*16)
60 PSET (X,Y),C
70 REM
80 REM SIMULEER NU Q=POINT(X,Y)
90 REM
100 VDP(33)=X:VDP(34)=0:VDP(35)=Y:VDP(36)=0'COORDINATEN
110 VDP(47)=64' COMMANDO: PINT
120 IF (VDP(-2) AND 1)=1 THEN 120'READY?
130 Q=VDP(-7)'HAAL KLEURKODE
140 IF Q<>C THEN STOP ELSE 50'CONTROLE RESULTAAT
RUN
```

```
NEW
Ok
10 REM SIMULEER PSET MET BEHULP VAN
20 REM DIREKTE VDP-BESTURING.
30 REM
40 SCREEN 5:COLOR ,1,1:CLS
50 X=INT(RND(1)*256):Y=INT(RND(1)*212):C=INT(RND(1)*16)
60 REM
70 REM SIMULEER PSET(X,Y),C
80 REM
90 VDP(37)=X:VDP(38)=0:VDP(39)=Y:VDP(40)=0'COORDINATEN
100 VDP(45)=C'KLEUR
110 VDP(47)=80' KOMMANDO: PSET
120 IF (VDP(-2) AND 1)=1 THEN 120 ELSE 50'READY?
RUN
```

14.28 De VDP status-registers

De VDP status-registers geven aan, welke de status (toestand van de Video Display Processor) op een bepaald moment is. Er zijn 1Ø status-registers, te weten:

VDP(...)	status-register	bit Ø	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
8	Ø	F	SD	C	S4	S3	S2	S1	SØ
-1	1	FL	LPS	I 4	I 3	I 2	I 1	IØ	FH
-2	2	TR	VR	HR	BD	Ø	Ø	E/O	CE
-3	3	X7	X6	X5	X4	X3	X2	X1	XØ
-4	4	Ø	Ø	Ø	Ø	Ø	Ø	X9	X8
-5	5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	YØ
-6	6	Ø	Ø	Ø	Ø	Ø	Ø	Y9	Y8
-7	7	C7	C6	C5	C4	C3	C2	C1	CØ
-8	8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BXØ
-9	9	Ø	Ø	Ø	Ø	Ø	Ø	BX9	BX8

= ook MSX-1

F = interrupt flag of vertical retrace. This flag will be reset by the SRØ read operation
SD = 5th/9th Sprite delected flag
C = sprite coincidence flag
S4 . . . SØ = 5th/9th sprite number
FL = interrupt flag of light pen or mouse switch status
CDS = light pen switch status or status flag of the second mouse switch
FH = interrupt flag horizontal retrace FL, LPS and FH will be reset by the SR1 read operation
HR = horizontal retrace timing
VR = vertical retrace timing
CE = command executing status (Ø = ready)
BD = border detected
E/O = even or odd field status
I4 . . . IØ = LSI version number

TR = transfer ready (data transfer with CPU)
X9 . . . XØ = X-coördinate } sprite coincidence location or
Y9 . . . YØ = Y-coördinate } light pen coördinates or
BX9 . . . BXØ= border X-coördinate } mouse coördinates.
C7 . . . CØ = color code of color register read.

NB: deze registers zijn READ ONLY. Hun inhoud kan niet worden veranderd maar wél worden opgevraagd.

```
NEW
Ok
10 REM BEPALING VAN DE BOTSPLAATS VAN SPRITES
20 REM VIA DE STATUSREGISTERS. ONDER BASIC
30 REM VERLOOPT DEZE AFVRAAG NIET SOEPEL;
40 REM VAAK STAAN ER OUDE BOTSWAARDEN IN DE
50 REM REGISTERS.
60 REM
70 SCREEN 2,3:COLOR 15,4,4:CLS
80 DEF FNR=INT(RND(1)*256)
90 SPRITE$(0)=STRING$(32,255):SPRITE$(1)=SPRITE$(0)
100 X1=FNR:Y1=FNR:X2=FNR:Y2=FNR
110 PUT SPRITE 0,(X1,Y1),15
120 PUT SPRITE 1,(X2,Y2),15
130 FOR I=1 TO 50:NEXT I
140 IF (VDP(8) AND 32)=0 THEN 100'BOTSING?
150 FOR I=1 TO 100:PUT SPRITE 0,,(I MOD 2)*14+1:PUT SPRITE 1
,,(I MOD 2)*14+1:X=VDP(-3):Y=VDP(-5):NEXT I'KNIPPEREN & COOR
DINATEN VRAGEN
160 CIRCLE (X,Y),20'CIRCEL OP BOTSPLAATS
170 GOTO 100
RUN
```

BIT 4,5,6,7				BIT 0,1,2,3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
					0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
				HEX 1/2	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	à	ð	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	õ	Ñ	õ			σ	¬
0	1	1	0	6	♠		&	6	F	V	f	v	å	ũ	a	Ũ			μ	÷
0	1	1	1	7	•		'	7	G	W	g	w	ç	ð	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		⁄	Θ	•
1	0	1	0	A	◙		*	:	J	Z	j	z	e	Ũ	¬	¼		₩	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	¥	i	‰			ϕ	²
1	1	1	0	E	♫		.	>	N	^	n	~	Ã	Pt	≪	¶			∈	∎
1	1	1	1	F	☼		/	?	O	_	o	△	Å	ƒ	≫	§			∩	Transparant

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).

16 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

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 eèn 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 BASIC handboek

De MSX standaard is een feit. MSX computers veroveren Nederland. Geen wonder; met MSX is er eindelijk een standaard opgestaan tussen de méer dan drie honderd verschillende basic dialekten die er nu nog zijn.
Stelt u zich eens voor. U koopt een computer van merk X en 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 eerste handboek uit een serie van drie, omvat een volledige behandeling van het standaard MSX 2 basic. Voorafgaand hieraan wordt een zeer duidelijke inleiding gegeven tot het verschijnsel MSX 2. Het handboek wordt gekompleteerd met een hoeveelheid praktische tabellen, duidelijke afbeeldingen en zinvolle voorbeelden.
Door de volledige syntaxisbehandeling, de duidelijke behandeling per sleutelwoord en de vele voorbeelden beantwoordt dit handboek bijna elke vraag en vormt het een standaardwerk dat naast elke MSX 2 computer zou moeten liggen.

ISBN 90 6398 221 6

uitgeverij Stark-Texel