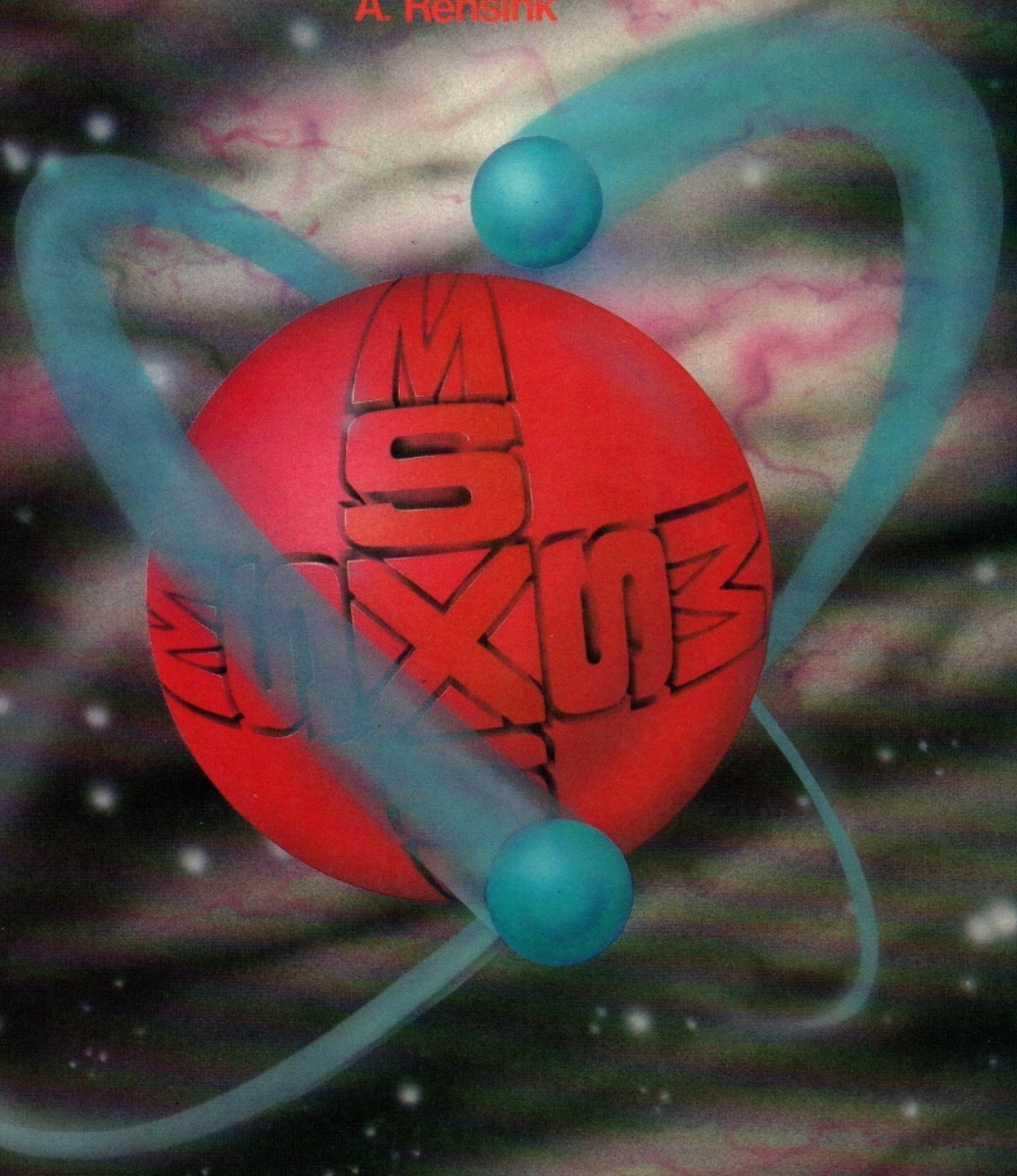
MSX-HANDBOEK voor gevorderden

A. Rensink



Kluwer Technische Boeken

MSX-HANDBOEK voor gevorderden

A.Rensink

MSX-handboek voor gevorderden

A. Rensink

MSX-handboek voor gevorderden

Kluwer Technische Boeken B.V.
Deventer – Antwerpen

Omslag: W. Niessink

ISBN 90 201 1925 7
D/1987/0108/165
NUGI 434

1e druk 1987

Woord vooraf

Dit boek is bedoeld voor degenen die hun MSX-computer niet alleen hebben gekocht om er spelletjes op te spelen of om er gekochte zakelijke programma's op te gebruiken, maar ook om er zelf goed mee te leren omgaan.
Met 'goed leren omgaan' bedoelen we een niveau van werken dat uitstijgt boven het zelf kunnen programmeren van Mastermind of Dieren Raden. We hebben het over een situatie waarin u niet staat te kijken van machinetaal en bitpatronen, waarin u niet de kluts kwijt raakt als we het hebben over hardware-registers en memory maps.

Het is niet onze bedoeling om u in dit boek kennis te laten nemen van de basisgegevens over machinetaal, bytes of bits. Deze kennis veronderstellen we reeds aanwezig. Veeleer maken we gebruik van deze basisgegevens om u de speciale eigenschappen en mogelijkheden van de MSX-computer duidelijk te maken.

Sommige van de gegevens in dit boek zijn u wellicht reeds bekend uit het oorspronkelijke handboek van de MSX-computer of uit andere bronnen. Andere zaken zijn echter volkomen nieuw, en zijn betrokken uit niet eerder gepubliceerde gegevens omtrent de MSX-standaard.

Inhoud

Inleiding: over computertermen en Engels

Een groot gedeelte van het vakgebied van de informatica – de computerkunde – is en wordt ontwikkeld op Engelstalig grondgebied. Alle vakliteratuur, zonder noemenswaardige uitzondering, is in het Engels. Als gevolg daarvan bestaat het computerjargon, de vaktaal van computergebruikers, geheel en al uit Engelse woorden.

Helaas is het niet mogelijk om in een boek als dit, dat diep ingaat op de werking van een computer, te ontkomen aan computerjargon. Het lijkt ons niet zinvol om iedere opduikende computerterm te vertalen. Niet alleen zou dat in sommige gevallen moeilijkheden opleveren waar geen voor de hand liggend equivalent te vinden is, maar ook vormt het jargon een tamelijk algemeen bekende en begrepen manier om sommige begrippen aan te duiden. Om die reden staat dit boek vol met Engelse termen, waarvan dikwijls geen uitleg of vertaling gegeven is.
Toch hebben we niet overal Engelse woorden gebruikt. In gevallen waarin een Nederlands alternatief bestaat hebben we de voorkeur gegeven aan het Nederlands. Zo hebben we het over geheugenadressen en niet over memory-addresses, over rijvariabelen en niet over arrayvariables, over stapelwijzers en niet over stack-pointers, over gleuven en niet over slots.

De discussies over het gebruik van computerjargon zijn nog lang niet uitgewoed. Bij gebrek aan een algemeen aanvaarde oplossing voor het probleem van het vervuilende Engels in een Nederlandse tekst, hebben we voor een aanpak gekozen die ons het meest redelijk voorkomt.

1 Het MSX-systeem

De meeste mensen zullen wel eens een printplaat of een foto ervan onder ogen hebben gehad. Eén blik op zo'n plaat is voldoende om je te overtuigen dat computers veel te ingewikkeld zijn om te begrijpen en dat mensen die er voor hun plezier aan gaan solderen óf geniaal, óf knettergek, maar waarschijnlijk beide zijn.
Welnu, zo'n printplaat is niets bijzonders. Weliswaar bepaalt het ontwerp van zo'n plaat enigszins de snelheid van de computer (omdat de verbindingen tussen chips langer of korter worden), maar het 'hart' van de computer is toch gelegen in de chips (micro-elektronische schakelingen) die in de computer gebruikt zijn. En het ontwerp van de printplaat wordt voor een groot deel bepaald door de chips zelf: een bepaalde keuze van chips leidt tot bepaalde, onveranderbare verbindingen.

Dit doet misschien denken dat het ontwerpen van een computer niets voorstelt: men neme een paar chips en soldere een paar verbindingen. Welnu, dat is ook zo. Op precies dat principe zijn al jarenlang computers gemaakt die allemaal hetzelfde doen en op precies dezelfde manier. Er zijn ook al jarenlang bouwpakketten te koop, doe-het-zelf computers voor de echte hobbyist.

Waarin, zal men zich na het lezen van het voorgaande afvragen, is nu die grote technologische vooruitgang gelegen waarover we zo vaak in de kranten lezen? Het antwoord is: in de chips zelf. Wat micro heet, is elk jaar micro-er; elke twee jaar (of daaromtrent) krijgt men weer twee keer zoveel schakelingen op dezelfde vierkante millimeter silicium; elke paar jaar zien we weer een radicale verbetering in processorcapaciteit.
Hoe dat ook zij, de MSX-standaard is opgebouwd uit oude, beproefde chips, en betekent dan ook geen grote stap voorwaarts op dat gebied. Maar daarbij moet worden opgemerkt dat de grootste successen op computergebied nooit zijn geboekt met technische hoogstandjes – integendeel.
Het voornaamste nieuwe aan MSX is het feit dat het een standaard is, en dat elke computer die aan die standaard voldoet alle apparatuur en programmatuur aankan die op die standaard is berekend. Dit in tegenstelling tot de meeste computers die op geen enkele manier met elkaars programma's of randapparaten kunnen werken.

In de volgende paragrafen van dit hoofdstuk zullen we kort ingaan op de MSX-standaard en op de zaken die deel uitmaken van die standaard. Veel van deze zaken zullen in meer detail worden besproken in komende hoofdstukken.

1.1 Onderdelen van het MSX-systeem

In deze paragraaf wordt een kort overzicht gegeven van de onderdelen die in de MSX standaard beschreven staan en die in elke MSX-computer voorkomen.

De processor

De microprocessor die in een MSX-systeem gebruikt wordt, is een Z80A (of gelijksoortige). Deze processor (of preciezer: zijn voorganger, de Z80), is al jaren met groot succes gebruikt in allerlei computers. De keuze van deze processor voor de MSX-standaard heeft wel wat kritiek met zich meegebracht vanwege de weinig vernieuwende werking die ervan uitgaat.
Van de Z80 kan worden gezegd dat hij veel kan, maar moeilijk volledig te benutten is. Dit laatste komt vooral door de beschikbare machinetaal: het grote aantal instructieformaten en het gebrek aan algemeenheid van de verschillende adresseringen maken het moeilijk om alle mogelijkheden van de Z80 uit te buiten.
De klokfrequentie waaronder de Z80 in het MSX-systeem werkt, is (bijna) 3,58 MHz, lang niet de maximumfrequentie waaronder hij kan werken (tot 8 MHz). Bovendien kent de Z80 in de configuratie van het MSX-systeem een wachtlus (WAIT-state) in de eerste machinecyclus. Dergelijke wachtlussen worden gebruikt om de processor gesynchroniseerd te laten werken met andere, langzamere delen van de computer (zoals het geheugen). Een wachtlus vertraagt de processor.

De interrupt-faciliteiten van de Z80 worden in het MSX-systeem nauwelijks gebruikt. INT wordt in het basissysteem alleen gebruikt voor de interrupt van de VDP (zie verderop) die plaatsvindt met een frequentie van 50 Hz in Europese, en 60 Hz in Amerikaanse systemen; NMI wordt helemaal niet gebruikt.

Voor details over de Z80, en Z80-machinetaal verwijzen we de lezer naar speciale literatuur over deze processor.

Het geheugen

De grootte van het programmageheugen van een MSX-computer kan variëren van 48 kbyte (32 kbyte ROM + 16 kbyte RAM) tot 1024 kbyte. Gedeeltelijk wordt dit bepaald door de fabrikant die aan de computer een bepaalde hoeveelheid geheugen geeft echter alle MSX-computers kunnen in theorie 1024 kbyte (=1 Mbyte) aan geheugen bevatten. Dit kan ROM- of RAM-geheugen zijn.

Aangezien de Z80-processor maar een adresbereik van 64 kbyte heeft, zijn er speciale mechanismen ontworpen om al dat geheugen aan te kunnen spreken. Deze mechanismen, en andere gegevens omtrent het geheugen, zullen aan de orde komen in hoofdstuk 2.

Naast programmageheugen bevat de MSX-computer ook 16 kbyte videogeheugen, dat enkel en alleen gebruikt wordt voor de opbouw van het beeld dat op de aangesloten TV of monitor zichtbaar is. Het gebruik van het videogeheugen wordt besproken in hoofdstuk 3.

Het videogedeelte

Voor het besturen van het videogedeelte van de MSX is een speciale chip ingebouwd, de Video Display Processor (VDP). De VDP-chip is een TI TMS-9918A (of gelijksoortige). Kenmerken van deze chip zijn:

- 256 tekens (ASCII-tekens en grafische tekens) met een 8×8-patroon
- 16 kleuren
- 32 sprites (max. 4 op één horizontale lijn)

Bij het videogedeelte hoort een apart videogeheugen dat geen deel uitmaakt van het gewone RAM-geheugen; zie hiervoor onder het kopje 'Het geheugen'. Dit videogeheugen kan alleen worden aangesproken via de VDP. Het videogeheugen is 16 kbyte groot.

Meer gegevens over de VDP en de werking van het videogedeelte van de MSX-computer staan in hoofdstuk 3.

Het audiogedeelte

Voor het regelen van het audiogedeelte is een speciale chip ingebouwd, soms aangeduid met Programmable Sound Generator (PSG). Dit is een GI AY-3-8910 (of gelijksoortige). Deze chip heeft als kenmerken:

- 3 stemmen toon/ruis
- 8 octaven bereik
- voorgeprogrammeerde volumeverlopen

Het MSX-systeem heeft standaard geen aparte audio-uitgang. In plaats daarvan wordt het geluid via de televisie weergegeven.

Merkwaardig genoeg verzorgt de PSG behalve het genereren van geluid ook het uitlezen van de speluitgangen die op de MSX aanwezig zijn.

Meer informatie over de PSG en de werking van het audiogedeelte is te vinden in hoofdstuk 4, en in par. 5.7 kunt u meer vinden over de speluitgangen.

Het toetsenbord

Het toetsenbord heeft standaard 72 toetsen. Elke fabrikant mag daaraan naar believen toetsen toevoegen.

De plaats van de toetsen op het toetsenbord verschilt per land. In Nederland geldt de 'gewone' internationale indeling van het toetsenbord, ook wel aangeduid met QWERTY.

Om te kunnen reageren op invoer vanaf het toetsenbord moet er een manier zijn waarop de processor kan opmerken dat er toetsen worden ingedrukt en testen welke toetsen dat zijn. De methoden die bestaan om dit te bereiken zijn te verdelen in twee categorieën: interrupt en scanning.

In het MSX-systeem wordt scanning toegepast. Dit houdt in dat het testen op ingedrukte toetsen gebeurt door middel van een regelmatige 'scan' van alle toetsen van het toetsenbord waarbij van elke toets wordt getest of hij ingedrukt is. De scan wordt aangestuurd door de VDP-interrupt (zie de uitleg over de processor).

Andere onderdelen
Behalve de reeds besproken onderdelen en chips van de MSX, grotendeels ter besturing van andere onderdelen, kent de MSX nog een chip: de Programmable Peripheral Interface (PPI).
Voorts is er in de computer een klok ingebouwd die alle andere onderdelen synchroniseert door middel van klokpulsen. De klok werkt, zoals eerder vermeld, met een frequentie van 3,58 MHz.

1.2 Aansluitingen van het MSX-systeem

Een computer zonder verbindingen met randapparatuur is niet veel nuttiger dan een stuk massief plastic. In dat opzicht zijn de aansluitingsmogelijkheden van een computer van groot belang.

In de loop der jaren zijn voor sommige aansluitingen – zoals van computer naar computer, van computer naar printer, en van computer naar televisie – enkele standaardverbindingen ontworpen. Een computer die kan werken met zo'n standaardverbinding kan meestal zonder al te veel problemen worden aangesloten op alle apparaten die deze verbinding kennen. Het is dus voordelig voor de gebruiker als een computer veel standaardverbindingen heeft.
Standaardverbindingen zijn bijvoorbeeld de Centronics-interface en de RS232-interface.

We zullen in deze paragraaf ingaan op de verbindingen die in de MSX-standaard zijn vastgelegd, en dus aanwezig zijn op alle MSX-computers.

Verbinding met het beeldscherm
Voor de verbinding met een gewone televisie kent de MSX een antenne-aansluiting. De verbinding kan gelegd worden met een normale antennekabel. Aan de kant van de computer moet voor de aansluiting een 2-pins RCA-aansluiting gebruikt worden. Behalve beeld reproduceert de TV ook geluid. Het geluid valt dan ook weg indien in plaats van een TV een (kleuren- of monochrome) monitor wordt gebruikt. Zo'n monitor geeft namelijk geen geluid weer.
Voor de verbinding met de monitor kent de MSX een aparte uitgang. Deze verbinding werkt volgens het composiet-videosysteem. De aansluiting aan de kant van de computer kan gebeuren met een RCA 2-pins aansluiting, net als bij de televisieverbinding, of met een DIN 5-pins aansluiting. De aansluiting aan de kant van de monitor verschilt per monitor: er bestaat in dit geval geen standaardverbinding.
Sommige TV's kunnen ook op de monitoruitgang in plaats van op de antenne-uitgang worden aangesloten en geven dan een beter beeld.

Verbinding met de cassetterecorder
De verbinding met de cassetterecorder gebeurt via een kabel die bij de computer wordt meegeleverd. Deze kabel heeft aan de computerkant een 8-pins DIN-plug (DIN 45326) en aan de recorderkant drie monostekkers.

Elke cassetterecorder met (tenminste) een MIC-(microfoon) en een EAR-(oortelefoon)aansluiting kan met behulp van deze kabel aangesloten worden op de computer. Details omtrent de aansluiting staan in het handboek van de computer.
De lay-out van de DIN-plug en meer gegevens over de verbinding met de cassetterecorder zijn te vinden in par. 5.4 en par. 6.2.6.

Verbinding met de cartridge
Elke MSX-computer kent ten minste één cartridge-aansluiting. Dit is een 50-pins aansluiting, genaamd 2.54 PACE. Alleen MSX-cartridges kunnen (en mogen) hierop worden aangesloten.
Meer informatie omtrent cartridges vindt u in par. 5.3.

Spelingangen
Elke MSX-computer kent ten minste één spelingang. Hierop kan een joystick, touch pad of een set paddle-controllers aangesloten worden.
De spelingang is een 9-pins AMP-aansluiting, die in vele andere huiscomputers wordt gebruikt. Als gevolg daarvan kunnen alle bestaande joysticks die van die ingang gebruik maken, zonder problemen met een MSX gebruikt worden.
De lay-out en andere gegevens over de spelingang zijn te vinden in par. 5.7.

Tot nu toe zijn er alleen verbindingen besproken die elk MSX-computersysteem moet hebben. In de MSX-standaard zijn echter ook andere verbindingen gespecificeerd die niet noodzakelijkerwijs in een minimumsysteem aanwezig hoeven te zijn. In het resterende deel van deze paragraaf zullen we ook deze verbindingen kort bespreken.

Verbinding met de printer
Voor het aansturen van een printer kent de MSX een parallelle Centronics-verbinding. Centronics is een standaard die garandeert dat elke computer die een Centronics-aansluiting heeft, kan werken met elke printer met zo'n aansluiting, en dat zijn er vele. (Helaas blijkt dit in de praktijk soms toch niet te werken vanwege onvolledige implementaties van de verbinding op sommige computers en/of printers.)

In feite kunnen via een Centronics-aansluiting ook andere apparaten dan printers worden aangesloten, maar verreweg de meest voorkomende toepassing van de aansluiting in huiscomputers is toch wel het aansturen van printers.
De verbinding wordt aan de kant van de computer gelegd via een 14-pins AMPHENOL-aansluiting. Meer informatie over de aansluiting is te vinden in par. 5.6.

RS232-verbinding
Zoals de Centronics-standaard een parallelle verbinding specificeert, zo doet de RS232-standaard dit voor seriële verbindingen.
Helaas is, nog meer dan bij de Centronics-aansluiting, een probleemloze verbinding van twee RS232-aansluitingen op verschillende apparaten lang niet zeker.
De RS232-aansluiting kent een breder toepassingsgebied dan de Centronics-aanslui-

ting. Zo wordt RS232 vaak toegepast voor het koppelen van twee computers, zodat deze gegevens uit kunnen wisselen. Ook wordt RS232 wel gebruikt voor het aansturen van printers.
Een RS232-interface kan alleen via de cartridge-gleuf worden aangesloten. Meer gegevens over RS232 vindt u in par. 5.7.

Audio-uitgang
Zoals gezegd, wordt het geluid dat de MSX-computer maakt, weergegeven via de televisie. Sommige MSX-computers zijn echter voorzien van een extra uitgang voor alleen geluid. Deze uitgang is in de vorm van een 2-pins RCA-aansluiting, net als de TV- en monitoraansluitingen.
Door gebruik van deze aansluiting kan men geluid altijd weergeven, zelfs bij gebruik van een monitor.
We zullen hier verder niet op terugkomen.

2 Het geheugen

Het MSX-systeem is standaard uitgerust met 32 kbyte ROM-geheugen en 16 kbyte tot 64 kbyte RAM-geheugen, afgezien van het videogeheugen.

Bij het lezen van deze zin zullen degenen die bekend zijn met de Z80 verwonderd opkijken. Deze microprocessor is namelijk maar in staat om 64 kbyte geheugen te adresseren. MSX-computers die 32 kbyte ROM én 64 kbyte RAM hebben, zullen, zou men zeggen, nooit al dat geheugen kunnen gebruiken.
In werkelijkheid kan dat natuurlijk wél. De manier waarop het geheugen van de MSX georganiseerd is, zodat de computer tot 1024 kbyte, oftewel 1 Mbyte, aan totale bruikbare geheugenruimte heeft (al is er dan meestal wat minder geïnstalleerd), zal in dit hoofdstuk uit de doeken worden gedaan. Daartoe zullen we echter eerst enige theoretische begrippen bespreken.

2.1 Het adresseren van geheugen

Om geheugen te kunnen gebruiken, moet men het kunnen bereiken: het moet gelezen kunnen worden, en (bij RAM-geheugen) er moeten nieuwe waarden aan toegekend kunnen worden.
Het bereiken van het geheugen wordt aangeduid met het adresseren van het geheugen. In deze paragraaf wordt eerst ingegaan op geheugenadressering in het algemeen, en vervolgens op het specifieke geval van de MSX.

2.1.1 Logische en fysische adressen

Wat is een geheugenadres? Het meest voor de hand liggende antwoord op deze vraag is: een getal, met een bereik van een aantal bits (vaak in huiscomputers 16 bits ofte wel twee bytes) dat een plaats in het geheugen aanduidt.
Een ander mogelijk antwoord is: het totaal aan (fysische) signalen waarmee één specifieke geheugenplaats wordt geselecteerd en onderscheiden van een andere geheugenplaats.
Bij bestudering van deze twee antwoorden en het verschil ertussen, valt ten eerste op dat het eerste antwoord een benadering is van de kant van de programmeur en het tweede een benadering vanuit de computer. In een programma heeft een adres, bijvoorbeeld een sprongadres voor een JMP-instructie, de vorm van een getal bestaande uit twee bytes. Evenzo kan een bepaalde geheugenplaats uitgelezen worden door het adres in de vorm van een getal op te geven.
Bij uitvoering van zo'n leesinstructie moet de waarde van de geheugenplaats worden

gevonden. Daartoe moet, uit de vele duizenden bytes van het geheugen, precies dat ene byte geselecteerd worden. Dit gebeurt aan de hand van het opgegeven adresgetal. De selectiesignalen, met behulp waarvan de geheugenplaats uiteindelijk gevonden wordt, hoeven geenszins precies de bits van dat getal te zijn. Een voorbeeld waarin dat niet het geval is, is wanneer er voor de selectiesignalen negatieve logica wordt toegepast.
We zullen in de rest van deze paragraaf het adresgetal dat de programmeur gebruikt aanduiden met het logische adres van de geheugenplaats, en het geheel aan selectiesignalen met het fysische adres.

Nu is in de meeste huiscomputers het totale fysische geheugen niet groter dan het logische adresbereik (bepaald door het aantal bits van het getal van een logisch adres) van de processor. In dat geval is het niet nodig om onderscheid te maken tussen het logische en het fysische adres; zo'n onderscheid zou alleen verwarring wekken. Er wordt dan gesproken van een één-op-één-relatie tussen het logische adres en het fysische adres van een geheugenplaats.
In grote computers is al jaren het logische adresbereik veel groter dan het fysische geheugen. Er wordt dan gesproken van een veel-op-één-relatie tussen het logische en het fysische adres. Op de precieze relatie tussen deze adressen gaan we verder niet in. Het is belangrijk om te onthouden dat het in dit geval niet a priori duidelijk is welk fysisch adres bij een bepaald logisch adres hoort.

In de MSX is het nu zo dat het logische adresbereik kleiner is (of kan zijn) dan de hoeveelheid fysisch geheugen: er bestaat een één-op-veel-relatie tussen logisch en fysisch adres.
Opnieuw is het zo dat uit een bepaald logisch adres niet zonder meer af te leiden valt welke fysische geheugenplaats daarbij hoort. Om dat te weten te komen, zijn er meer gegevens nodig. Waar deze gegevens vandaan komen en hoe ze gebruikt worden, zal in de volgende paragraaf aan de orde komen.

In het resterende deel van dit boek wordt met 'adres' logisch adres bedoeld, tenzij anders vermeld.

2.1.2 Adressering binnen het MSX-systeem

De Z80-processor, en dus de MSX-computer, kent twee soorten logische adressen: poortadressen en geheugenadressen.
Poortadressen bestaan uit één byte. Ze worden gebruikt bij de machinetaalinstructies IN en OUT. De 'poorten' die geadresseerd worden, zijn meestal communicatielijnen met andere onderdelen van de computer.
Geheugenadressen bestaan uit twee bytes. Ze worden in allerlei machinetaalinstructies gebruikt wanneer er iets van het geheugen gelezen of geschreven moet worden.
Het zij nogmaals benadrukt dat zowel de poortadressen als de geheugenadressen logische adressen zijn, en dat niet zonder meer vast staat wat er in feite geadresseerd wordt.

2.2 I/O-poorten

In de vorige paragraaf is het begrip 'poort', ook wel I/O-poort genoemd, aan de orde gekomen.
Als een microprocessor, zoals de Z80, alleen de beschikking zou hebben over 'gewoon' geheugen, zou er geen verbinding zijn met de buitenwereld. Programma's die in het geheugen staan kunnen worden uitgevoerd, maar de resultaten zouden niet bekend kunnen worden gemaakt aan de omgeving.
Voor communicatie met de buitenwereld zijn de I/O-poorten ontwikkeld. Deze poorten vormen een nieuw soort adresruimte die we hebben aangeduid met poortadressen.

Wanneer een poort wordt geadresseerd – en dit gebeurt in de Z80-machinetaalinstructies IN en OUT – wordt het logische adres, een getal met een bereik van acht bits, samen met een lees/schrijf-keuzesignaal gebruikt om andere chips dan de processor of het geheugen aan te spreken.
Deze andere chips hebben vaak zelf intern geheugen, aangeduid met 'registers'. De nummers of namen van deze registers kunnen we weer beschouwen als de fysische adressen van het registergeheugen. Ook kan het voorkomen dat een chip zelf een grote hoeveelheid privé-RAM heeft, zoals het geval is bij de VDP van het MSX-systeem. Dit geheugen moet weer fysisch geadresseerd kunnen worden.
De manier van communiceren is sterk chip-afhankelijk en vaak tamelijk gekunsteld. De reden voor het laatste is dat het logische poortadres niet veel informatie bevat; het bereik is maar acht bits, en met die acht bits moet meestal een aantal chips bestuurd worden. Het fysische adres van het geheugen binnen een chip in de poortadresruimte wordt in veel gevallen bepaald uit één of meer waarden die naar een poortadres geschreven worden, in plaats van uit het adres zelf, resulterend in een mengsel van adressen en andere gegevens die van en naar de poorten geschreven worden.

In de hoofdstukken 3 t/m 5 beschrijven we zulke 'andere chips' die via de poortadresruimte aangesproken moeten worden. We zullen het communicatie-protocol met dergelijke chips bespreken binnen het kader van logische en fysische adressering. Details die chip-afhankelijk zijn, worden ook per chip besproken.

2.3 Het gleufmechanisme

Waar we het in deze paragraaf hebben over 'geheugen', bedoelen we het RAM- en ROM-geheugen dat met behulp van de gewone 2-bytes geheugenadressering aangesproken wordt.
Het geheugen in het MSX-systeem is georganiseerd op een zeer speciale manier die het mogelijk maakt om het fysische geheugen uit te breiden tot vele malen het adresbereik. Deze manier wordt aangeduid met het gleufmechanisme.

De MSX kent vier zogenaamde primaire gleuven die elk of 64 kbyte aan geheugen kunnen bevatten, of uitgebreid kunnen worden tot vier secundaire gleuven, die

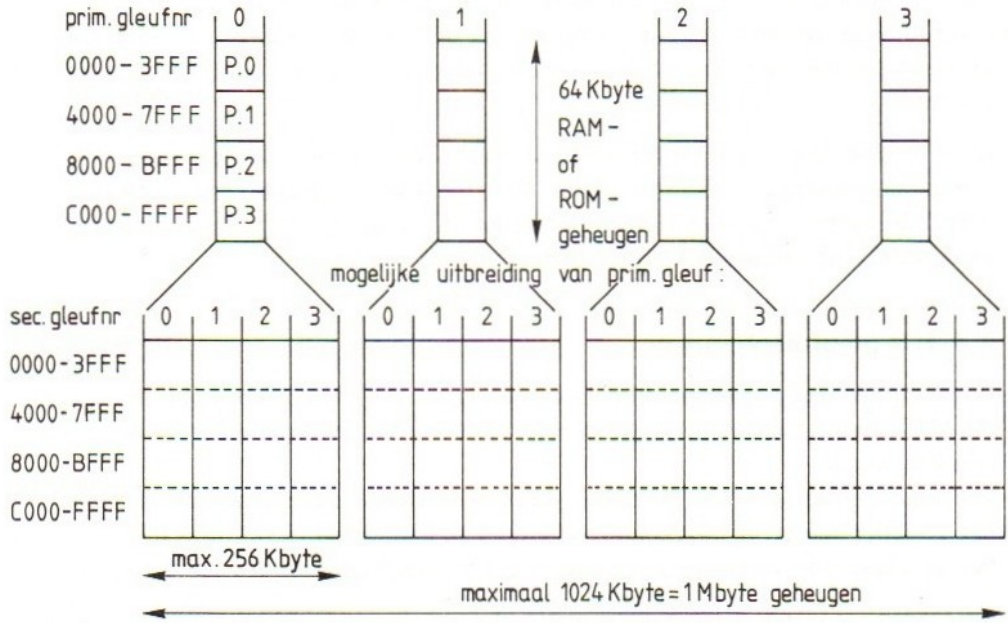
elk weer 64 kbyte kunnen bevatten. Samen kunnen de gleuven dus 4×4×64K= 1024 kbyte aan geheugen bevatten.
Het begrip 'gleuf' duidt op een interne structuur waarin mogelijk geheugen kan worden geïnstalleerd. De gleuven van het MSX-systeem zijn voor de programmeur alleen zichtbaar door het gleufselectiemechanisme zoals hiervoor beschreven. De gleufstructuur moet niet worden verward met een cartridge-gleuf.

Het fysische geheugen in een gleuf (hetzij niet-uitgebreid primair of secundair) bestaat uit vier installeerbare delen, pagina's genaamd. De pagina's van een gleuf zijn genummerd van 0 t/m 3. Elke pagina bevat maximaal 16 kbyte aan geheugen. Dit geheugen kan RAM of ROM zijn, of een mengsel van de twee.
Elke pagina van elke gleuf heeft een vast adresbereik, of er nu wel of geen geheugen in de pagina is geïnstalleerd. Het verband tussen het nummer van een pagina en het bijbehorende adresbereik is als volgt:

Pagina 0: 0000-3FFF
Pagina 1: 4000-7FFF
Pagina 2: 8000-BFFF
Pagina 3: C000-FFFF

Resumerend hebben we nu het volgende beeld van het fysische geheugen (afb. 2.1):

- Het systeem kent (maximaal) vier primaire gleuven.
- Elke primaire gleuf kan geheugen bevatten. Anderzijds kan een primaire gleuf worden uitgebreid tot vier secundaire gleuven, genummerd van 0 t/m 3.
- Een secundaire gleuf kan geheugen bevatten.



Afb. 2.1. Het gleufsysteem van de MSX-computer

- Elke gleuf (primair of secundair) bestaat uit vier pagina's, genummerd 0 t/m 3
- Elke pagina kan tot 16 kbyte geheugen bevatten. Dit kan RAM of ROM zijn, of een mengsel van de twee.

Een bepaalde pagina van een zekere gleuf kan nu op de volgende manier worden bereikt:

- Het nummer van de primaire gleuf waarin de pagina zich bevindt wordt ingesteld in poort A van de PPI (zie verderop).
- Als de primaire gleuf uitgebreid is, wordt het nummer van de secundaire gleuf waarin de bedoelde pagina zich bevindt ingesteld in het selectieregister van de primaire gleuf.
- De pagina kan nu aangesproken worden in het adresbereik waarin de pagina zich bevindt.

In deze beschrijving staan twee begrippen die nadere uitleg behoeven:

Poort A van de PPI is een register van de chip die PPI heet. Deze chip kan aangesproken worden via de poortadresruimte, zoals beschreven in par. 2.2. De PPI wordt in groter detail beschreven in par. 5.1. Het enige dat hier van belang is, is poort A, omdat die gebruikt wordt in het gleufmechanisme.
Poort A kan aangesproken worden via poortadres A8. Dit is de enige adressering die nodig is; het logische adres A8 komt overeen met het fysische register van poort A van de PPI.

Het register bestaat uit acht bits, genummerd van 7 t/m 0. Deze bits hebben de volgende betekenis:

Bit 7-6 prim. gleuf voor pag. 0 bereik: 0-3
Bit 5-4 prim. gleuf voor pag. 1 bereik: 0-3
Bit 3-2 prim. gleuf voor pag. 2 bereik: 0-3
Bit 1-0 prim. gleuf voor pag. 3 bereik: 0-3

Het selectieregister van een primaire gleuf is alleen aanwezig bij een primaire gleuf die uitgebreid is tot vier secundaire gleuven. In dat geval wordt het register gebruikt om een keuze te maken tussen de secundaire gleuven.

Het selectieregister kan niet aangesproken worden via een poortadres. In plaats daarvan moet het aangesproken worden via het geheugenadres FFFF van de desbetreffende primaire gleuf. In deze situatie wordt dus in plaats van een poortadres het 'gewone' logische adresbereik gebruikt om een register te adresseren. Deze manier van adressering wordt aangeduid met 'memory-mapped I/O'.

Bij het lezen van het selectieregister krijgt men de inverse van de werkelijke waarde. Bij het schrijven echter, komt de geschreven waarde rechtstreeks in het register. Doordat een geschreven waarde dus geïnverteerd is bij het uitlezen, kan men een selectieregister onderscheiden van RAM-geheugen dat eventueel via adres FFFF aangesproken wordt.

Alles te zamen is het selecteren van een pagina in een secundaire gleuf een tamelijk ingewikkelde zaak. Om het selectieregister van een primaire gleuf te kunnen bereiken, moet namelijk eerst pagina 3 van die primaire gleuf geselecteerd worden (omdat geheugenadres FFFF zich in pagina 3 bevindt); daarna kan de gewenste secundaire gleuf geselecteerd worden door het gleufnummer naar het selectieregister te schrijven. Ten slotte zal vaak pagina 3 weer op de oorspronkelijke gleuf moeten worden teruggezet.

De indeling van een selectieregister is als volgt:

Bit 7-2 ongebruikt
Bit 1-0 nummer van sec. gleuf bereik: 0-3

Het gebruik van adres FFFF voor het aanspreken van het selectieregister van de primaire gleuf, heeft overigens een neveneffekt. De fysische geheugenbytes op adres FFFF van de secundaire gleuven kunnen nu namelijk niet bereikt worden: immers, dat adres moet vrij zijn om het selectieregister te kunnen bereiken.
Het volledige algoritme voor het selecteren van een bepaalde gleuf voor een pagina luidt nu als volgt:

- Verander de bits in poort A van de PPI, behorend bij de bedoelde pagina, in het nummer van de gewenste primaire gleuf.
- Als de primaire gleuf niet uitgebreid is, is het selectieproces nu afgerond.
- Als de primaire gleuf wél uitgebreid is en de bedoelde pagina heeft nummer 3, schrijf dan het nummer van de gewenste secundaire gleuf naar adres FFFF. Het selectieproces is nu afgerond.
- Als de bedoelde pagina niet nummer 3 heeft, onthoud dan de huidige primaire gleufselectie van pagina 3 (af te lezen uit poort A) en selecteer voor pagina 3 dezelfde primaire gleuf als voor de bedoelde pagina door het nummer van die primaire gleuf naar bits 1-0 van poort A te schrijven.
- Schrijf naar adres FFFF het nummer van de te selecteren secundaire gleuf.
- Zet het nummer van de geselecteerde primaire gleuf van pagina 3 terug op de oude waarde die voor dat doel onthouden was.

Bij dit algoritme valt nog het volgende op te merken:

1. De machinetaalroutine waarin het voorgaande geïmplementeerd is, kan zich niet in het adresbereik bevinden van de pagina die opnieuw geselecteerd wordt. Bovendien kan de routine niet in pagina 3 staan indien de nieuw te selecteren primaire gleuf uitgebreid is; dit omdat het adresbereik van pagina 3 in dat geval tijdelijk een ander gedeelte van het fysische geheugen aanspreekt.

2. Het is niet mogelijk om tegelijkertijd twee secundaire gleuven, behorend bij dezelfde primaire gleuf, te selecteren voor twee verschillende pagina's.

Overigens is het gleufmechanisme binnen het OS (Operating System) van MSX-BASIC goed ondersteund. Het is dan ook over het algemeen onnodig, en zelfs niet wenselijk, om eigen machinetaal-routines te gebruiken of om direct naar de PPI of het selectieregister van een primaire gleuf te schrijven. In plaats daarvan kan men

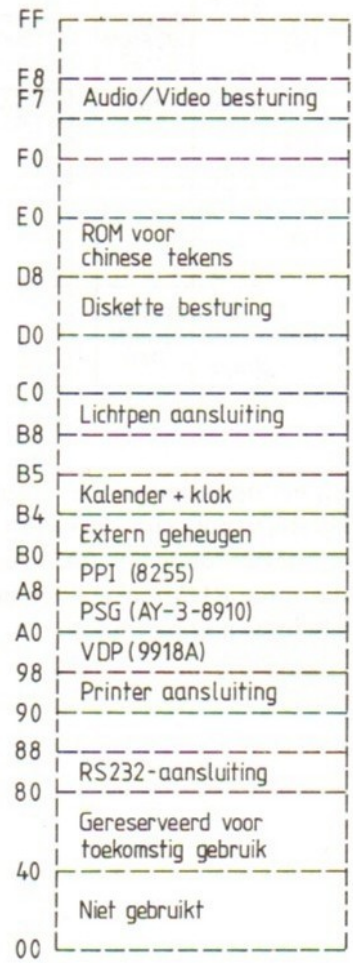
beter gebruik maken van de OS-ingangen die in par. 6.3 gedocumenteerd staan. Slechts als men permanent pagina 0 of 1 van een andere gleuf dan de systeemgleuf wil selecteren, is het nodig eigen routines te gebruiken.
Tussen het logische adres, waarmee in machinetaal het geheugen geadresseerd wordt, en het fysische adres, waarmee binnen de computer de gewenste geheugenplaats geselecteerd wordt, is nu het volgende verband aan te wijzen:

- De eerste twee bits van het logische adres vormen het paginanummer.
- De inhoud van poort A van de PPI vormt een tabel waarin aan de hand van het paginanummer het nummer van de primaire gleuf wordt opgezocht.
- Als de primaire gleuf niet uitgebreid is, vormt het gleufnummer samen met het logische adres het fysische adres.
- Als de primaire gleuf is uitgebreid, staat in het selectieregister het nummer van de secundaire gleuf.
- Het primaire en secundaire gleufnummer vormen samen met het logische adres het fysische adres.

2.4 De geheugenindeling van de MSX

In voorgaande paragrafen is het principe van logische en fysische adressen, poortadressering en het gleufmechanisme besproken. In deze paragraaf komen we toe aan de indeling van het geheugen zoals die in het MSX-systeem is.
Allereerst de indeling van de poortadresruimte. Afbeelding 2.2 geeft een geheugenkaart van de poortadressen.

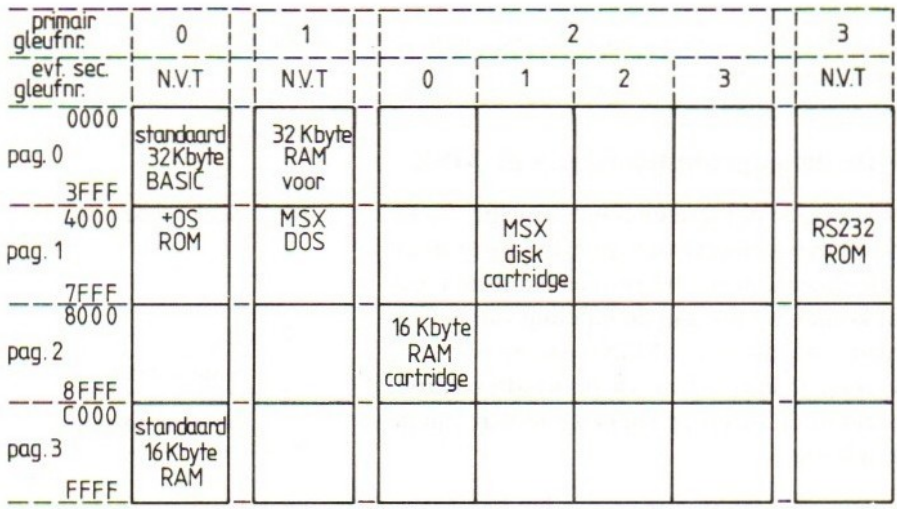
Poortadressen 00-3F zijn in het huidige MSX-systeem ongebruikt en zijn vrij voor eigen toepassingen. Lege gebieden in de adresruimte 40-FF zijn gereserveerd voor toekomstig gebruik door het systeem.
Welke functie elk van de systeempoortadressen heeft, wordt vermeld bij de bespreking van het onderdeel dat via dat adres aangesproken wordt.



Afb. 2.2. De poortadressen van de MSX-computer

Een geheugenkaart van het hoofdgeheugen levert problemen op, aangezien niet voorgeschreven is in welke gleuf een bepaalde pagina zich moet bevinden. Het enige voorschrift is dat de pagina's met het Operating System en BASIC-ROM zich in primaire gleuf 0 moeten bevinden, of in secundaire gleuf 0 van primaire gleuf 0 indien gleuf 0 uitgebreid is; en dat in een willekeurige gleuf pagina 3 gevuld moet zijn met 16 kbyte RAM. Verder moet er tenminste één gleuf vrij zijn voor het cartridgesysteem (zie par. 5.3).

Om een idee te geven van de geheugenkaart van de MSX is in afbeelding 2.3 een voorbeeldsysteem afgebeeld met vier primaire gleuven, geen van alle uitgebreid, waarin zich een pagina met een diskette station-aansluiting en 64 kbyte RAM-geheugen bevindt.



Afb. 2.3. Voorbeeld van een geheugenkaart

Bij het opstarten van de computer onderzoekt het OS de inhoud van de gleuven. Hierbij wordt een zo groot mogelijk blok RAM-geheugen opgezocht en het selectieregister wordt op dit blok ingesteld. Verder wordt een onderzoek gedaan naar geïnstalleerde ROM-pagina's en de functie daarvan. Het zoekalgoritme wordt in detail beschreven in par. 6.2.1.

3 Video

Het belangrijkste en meest gebruikte uitvoerapparaat van de computer is zonder twijfel het beeldscherm. In feite wordt het beeld dat iemand van een computer heeft voor het grootste deel bepaald door wat hij op het beeldscherm ziet.
Om die reden is het belangrijk dat de mogelijkheden voor schermaansturing groot zijn. Deze mogelijkheden worden voor het grootste deel gedicteerd door de gebruikte videochip.

In de MSX wordt het videogedeelte beheerd door de Video Display Processor, ofte wel VDP. De VDP is een chip die intern een groot aantal grafische mogelijkheden kent maar aan de andere kant op sommige punten nogal beperkt is.
In dit hoofdstuk wordt de VDP in detail beschreven. Eerst bespreken we de mogelijkheden en eigenschappen van de VDP, vervolgens gaan we in op de manier waarop van die mogelijkheden gebruik kan worden gemaakt.

3.1 Theorie: de principes van de VDP

Alle beelden die op het scherm van een computer verschijnen zijn opgebouwd uit puntjes. Deze puntjes – ook wel pixels genaamd – hebben een bepaalde kleur. De kleur van elk van de puntjes wordt door de VDP bepaald.
Wat nu belangrijk is, is de manier waarop de VDP van een gegeven pixel de kleur vaststelt. Kennis hiervan vormt de basis van het begrijpen van de werking van de VDP. Als namelijk bekend is hoe de kleur van een pixel vastgesteld wordt, is het ook mogelijk elk pixel de gewenste kleur te geven, en op die manier een zinvol beeld op te bouwen.
De gegevens die de VDP gebruikt zijn opgeslagen in de vorm van tabellen. De tabellen die de VDP kent zijn de naamtabel, de patroontabel, de kleurentabel, de sprite-patroontabel en de sprite-vlaktabel. Aan elk van deze tabellen is een paragraaf gewijd.

3.1.1 De naamtabel

Het woord 'naam' wordt hier gebruikt in de abstracte zin als: 'aanduiding van een object'. De namen die in de naamtabel staan, zijn geen woorden maar getallen. Deze getallen duiden inderdaad een object aan, namelijk een patroon in de patroontabel.

Elk element van de naamtabel beslaat één byte. De waarde van dat byte is dan de naam. Er zijn dientengevolge 256 mogelijke namen.

De elementen van de naamtabel corresponderen allemaal met een positie op het scherm. Anders gezegd, het schermbeeld is opgedeeld in hokjes en bij elk van die hokjes, hoort een element van de naamtabel. Zoals gezegd, is de waarde van dat element een aanduiding van een patroon uit de patroontabel. Op elke positie van het beeldscherm komt nu het patroon te staan dat wordt aangeduid door het element van de naamtabel dat correspondeert met die positie.

3.1.2 De patroontabel

De patroontabel bevat beeldpatronen, ofte wel: elk element van de patroontabel bevat informatie omtrent de vorm van een beeldpatroon. Een beeldpatroon is een matrix van patroonpunten. Elke patroonpunt kan 'aan' of 'uit' zijn. De matrix kan uitgebeeld worden door een ruitjesvierkant waarin elk ruitje overeen komt met een patroonpunt. Het ruitje is zwart als de patroonpunt aan is, en wit als de punt 'uit' is.

In de elementen van de patroontabel zijn de patroonpunten gerepresenteerd als bits. Elk bit van een element correspondeert met een patroonpunt van het bijbehorende beeldpatroon. Uit alle bits van een element samen is de vorm van het patroon af te leiden.
Met de naamtabel als tussenstap, is bij elke positie op het scherm een patroon uit de patroontabel aan te wijzen. Bij de schermpositie hoort namelijk een element van de naamtabel die een naam van een patroon bevat; dit patroon hoort dan bij de schermpositie.

Het resultaat is dat op elke positie van het scherm het bijbehorende beeldpatroon komt te staan. Aangezien de schermposities samen het hele beeld vormen, is op deze manier voor het hele scherm vastgelegd welk patroon erop verschijnt. We weten van elke beeldpunt op het scherm of hij 'aan' of 'uit' is.

Het is echter nog niet bekend wat dat 'aan' of 'uit' zijn van een beeldpunt inhoudt. Met andere woorden, de kleuren van het beeld zijn nog niet vastgelegd. Dat gebeurt met behulp van de kleurentabel.

3.1.3 De kleurentabel

In de keurentabel worden de kleuren van de patronen vastgelegd. Bij elk patroon zijn via deze tabel de kleuren bekend die een patroonpunt heeft als hij aan of uit is. In een bepaalde situatie is het bijvoorbeeld mogelijk dat een punt wit is als hij aan is en zwart als hij uit is; met die gegevens is dan het patroon volledig vastgelegd. Op een ander moment kan een punt dat aan is groen zijn, en een punt dat uit is geel; het patroon is dan weliswaar niet veranderd, maar omdat de kleuren zijn veranderd is het aanzicht van het patroon – zoals het op het beeldscherm verschijnt – wel degelijk anders geworden.

De kleur die een patroonpunt heeft als hij aan is, noemen we de voorgrondkleur van het patroon; de kleur die een pixel heeft dat uit is, heet de achtergrondkleur.

Als van alle patronen de kleuren bekend zijn, is de informatie omtrent het scherm-

beeld compleet; het scherm kan nu helemaal in de juiste kleuren gezet worden door de drie besproken tabellen.
Het verband tussen de patroontabel en de kleurentabel, dat aangeeft waar de kleuren van een bepaald patroon kunnen worden gevonden, kan variëren. Het is niet altijd zo dat er voor elk patroon precies één element van de kleurentabel aanwezig is.

3.1.4 De sprite-patroontabel

Met behulp van de naamtabel, de patroontabel en de kleurentabel (par. 3.1.1 t/m 3.1.3) is het mogelijk een schermbeeld volledig vast te leggen en te besturen. Deze tabellen samen zullen we aanduiden met het basissysteem van de VDP.
Naast het basissysteem van de VDP bestaat er nóg een besturingsmogelijkheid voor het beeldscherm. Deze mogelijkheid duiden we aan met het sprite-systeem ('sprite' is het Engelse woord voor een soort geestverschijning).
Een sprite in het MSX-systeem is een patroon. Het patroon van een sprite wordt opgeslagen in de sprite-patroontabel. Elk element van die tabel bevat het patroon van één sprite. De toepassing van een sprite-patroon wordt vastgelegd in de sprite-vlaktabel.
Sprites kunnen gebruikt worden om animatie toe te passen door het patroon van een sprite snel over het scherm te laten bewegen of door de patronen van twee of meer sprites af te wisselen.

3.1.5 De sprite-vlaktabel

In de sprite-patroontabel is alleen vastgelegd hoe een sprite eruit ziet. Het feitelijke gebruik van sprites gebeurt door middel van de sprite-vlaktabel.
Elk element van de sprite-vlaktabel bevat een aantal gegevens. Dit zijn:
- het sprite-nummer. Dit is het nummer van het sprite-patroon waarop het element betrekking heeft.
- de plaats op het scherm. Door deze plaats in te stellen komt de sprite pas op het scherm te staan.
- de kleur. Spreekt voor zichzelf.

Het dient benadrukt te worden dat de elementen van de sprite-vlaktabel onafhankelijk zijn van de elementen van de sprite-patroontabel. Het is niet zo dat er bij elke sprite in de sprite-patroontabel een element van de sprite-vlaktabel hoort, of andersom.
Het sprite-systeem van de VDP gaat uit van de sprite-vlaktabel. Op elk sprite-vlak kan een sprite-patroon verschijnen dat dan zichtbaar is op het scherm (afgezien van enkele beperkende bepalingen die nog aan de orde zullen komen). De gegevens in het sprite-vlak zijn voldoende om vast te stellen om welk sprite-patroon het gaat, waar het op het scherm moet staan, en in welke kleur.
Elk sprite-vlak wordt evenwijdig aan het beeldscherm gedacht, en de sprite-patronen op een sprite-vlak bedekken de patronen van het basissysteem en van de sprite-vlakken 'onder' dit vlak. Het bovenste sprite-vlak is het eerste element in de sprite-vlaktabel; het vlak direct daaronder is het volgende element; enzovoort.

Samengevat wordt het schermbeeld op de volgende wijze vastgesteld:

1. Van elk pixel van het schermbeeld moet de kleur worden vastgesteld. Daartoe wordt voor elk pixel afzonderlijk het volgende algoritme gevolgd:
2. De sprite-vlaktabel wordt doorlopen.
3. Als een sprite-patroon in de sprite-vlaktabel 'aan' is op de plaats van het betreffende pixel, is de kleur van dat patroon de kleur van het pixel. De sprite-vlaktabel wordt echter nog wel verder onderzocht, op zoek naar sprite-botsingen.
4. Als er na het onderzoeken van de sprite-vlaktabel geen kleur is gevonden, wordt de naamtabel onderzocht.
5. De naam uit de naamtabel die hoort bij de schermpositie waarin het pixel zich bevindt, wordt bepaald.
6. Het element van de patroontabel waar de naam naar verwijst, wordt onderzocht. Er wordt vastgesteld of ons pixel in dat patroon aan of uit is;
7. Naargelang het pixel aan of uit is, wordt voor het pixel de voor- of achtergrondkleur die bij dat patroon hoort, gekozen.

Het is niet gezegd dat de VDP inderdaad voor elk pixel dit algoritme volgt, maar het effect voor het schermbeeld is wel alsof dat het geval is.

3.2 Praktijk: de werking van de VDP

De VDP werkt volgens de principes die in de vorige paragraaf zijn besproken. De precieze uitvoering en vorm van de besproken tabellen worden in deze paragraaf besproken.
De videotabellen die aan de orde geweest zijn, worden opgeslagen in een geheugengebied dat speciaal voor dit doel is gereserveerd. Dit geheugengebied duiden we aan met het videogeheugen. Het videogeheugen, 16 kbyte groot, is in principe alleen voor de VDP bereikbaar. Via de aansluiting van de VDP met de processor kan het videogeheugen echter ook door de Z80 bereikt worden.

Naast het videogeheugen heeft de VDP de beschikking over een klein lokaal geheugen van negen bytes, aangeduid met de VDP-registers. In de VDP-registers wordt alle informatie over het videosysteem bewaard, anders dan de videotabellen.
In de volgende paragrafen worden de organisatie van de tabellen en de betekenis van de VDP-registers besproken.

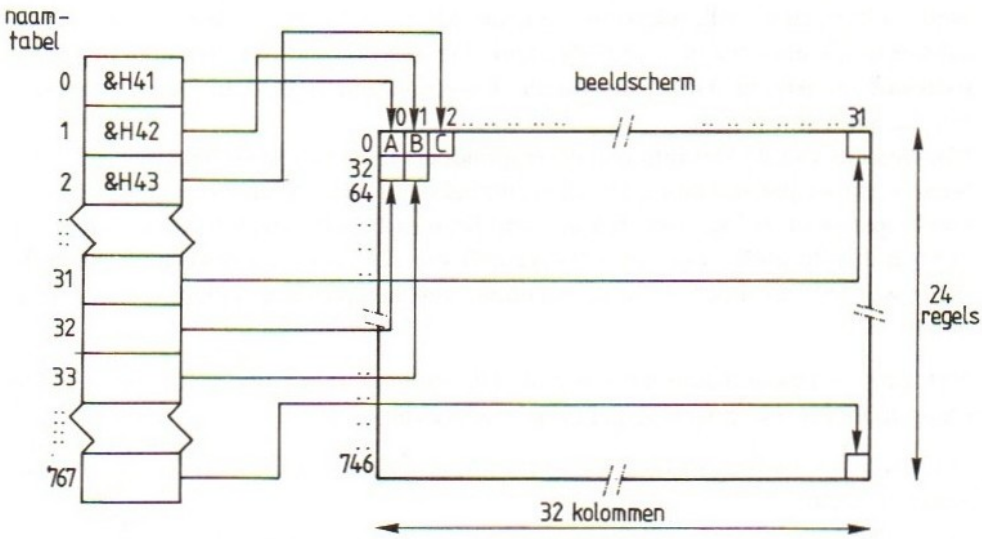
3.2.1 De instellingen van de VDP

De VDP kent verschillende scherminstellingen. Deze instellingen hebben betrekking op de manier waarop de tabellen georganiseerd en gebruikt worden.

De scherminstellingen zijn te beschouwen als een basisinstelling met variaties. Er zijn drie zelfstandig instelbare variaties. Gecombineerd zijn er dus acht mogelijke scherminstellingen; twee hiervan zijn echter niet bruikbaar.

De basisinstelling van de VDP

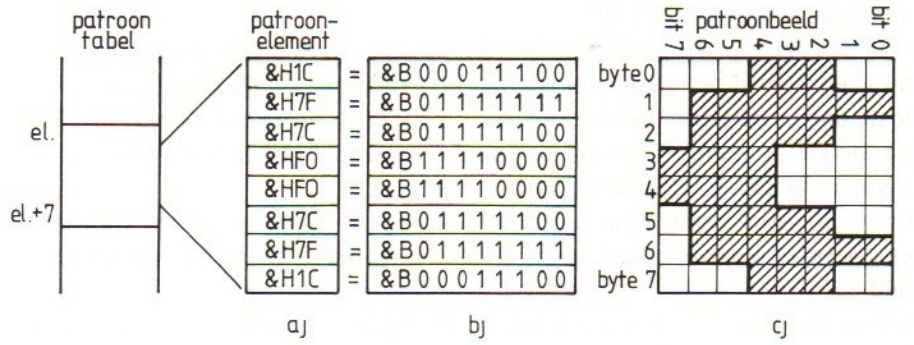
In de basisinstelling van de VDP wordt van alle tabellen gebruik gemaakt.
Het schermbeeld bestaat uit een rand en een middenstuk. De rand is een vlak in één kleur, boven en onder het middenstuk. De kleur van de rand wordt niet bepaald door de kleurentabel. Voor alle praktische doeleinden bestaat de rand niet, aangezien er niets aan te veranderen valt.
Het middenstuk wordt bestuurd door de tabellen. Wanneer we het voortaan over het scherm of schermbeeld hebben, bedoelen we het middenstuk, tenzij anders vermeld.
Het middenstuk wordt opgedeeld in 24 regels met elk 32 posities, in totaal 24×32=768 posities. Elke schermpositie correspondeert met een element van de naamtabel. De naamtabel telt dus 768 elementen.
Als we de elementen van de naamtabel nummeren van 0 tot en met 767, is het verband tussen schermpositie en elementnummer zoals vastgelegd in afbeelding 3.1.



Afb. 3.1. Verband tussen naamtabel en schermposities

De elementen van de naamtabel zijn één byte groot. Elke 'naam' is dus een getal tussen 0 en 255. Zoals gezegd zijn de namen in feite indices in de patroontabel. De patroontabel telt dus 256 elementen, genummerd van 0 tot en met 255.

De patroonbeelden die zijn gecodeerd via de elementen van de patroontabel zijn vierkanten van 8×8 patroonpunten. Elke patroonpunt kan aan of uit zijn; een patroonpunt kan dus gecodeerd worden door één bit. Voor elk element van de patroontabel zijn dan 8×8bits=8 bytes nodig. De inhoud van deze 8 bytes correspondeert met het resulterende patroon op de manier zoals aangegeven in afbeelding 3.2.



Afb. 3.2. Verband tussen element uit patroontabel en patroonbeeld

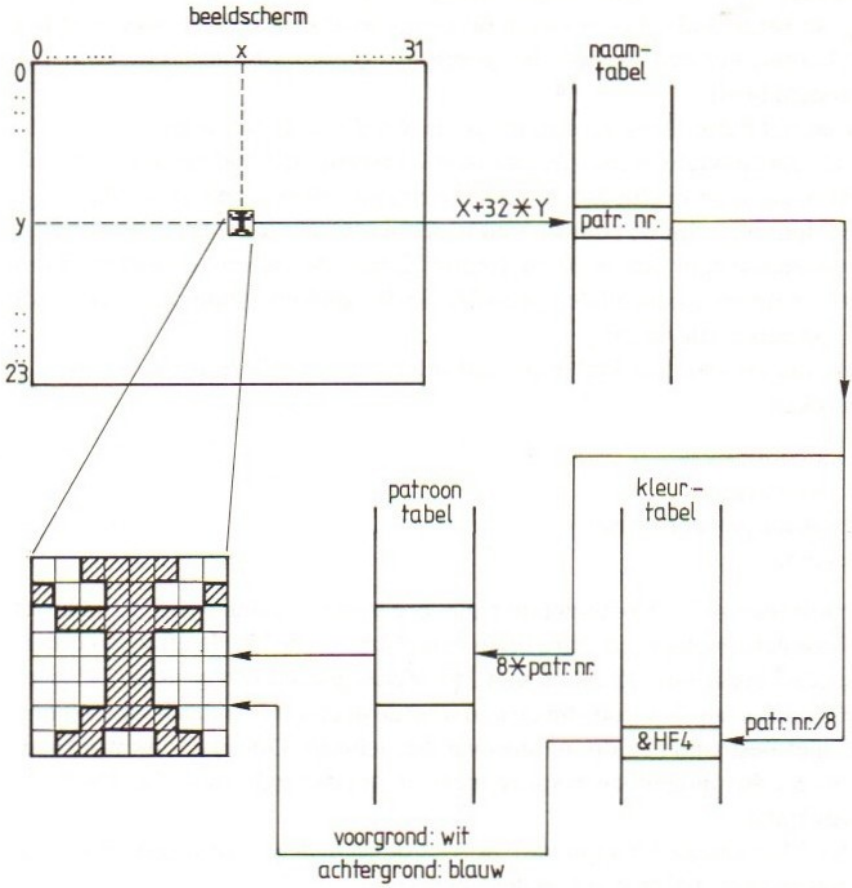
De kleurtabel bestaat uit elementen van één byte. Een kleur is een getal van 0 tot en met 15, dat gecodeerd kan worden in vier bits. Elk byte kan de code voor twee kleuren bevatten. Elk element van de kleurtabel bevat twee kleuren: een achtergrondkleur en een voorgrondkleur. De achtergrondkleur is de kleur die de patroonpunten hebben als ze uit zijn; de voorgrondkleur is de kleur van patroonpunten die aan zijn.
Elk element van de kleurentabel correspondeert in de basisinstelling met acht elementen van de patroontabel. De kleurentabel heeft dus 32 elementen. De patronen van de patroontabel worden dan per acht bestuurd door hetzelfde kleurenbyte, en hebben dus dezelfde voor- en achtergrondkleur. Bij een gegeven patroon kan de kleur worden gevonden door het nummer van het patroonelement door acht te delen.

Het totale verband tussen het schermbeeld, de naamtabel, de patroontabel en de kleurentabel is nog eens weergegeven in afbeelding 3.3.

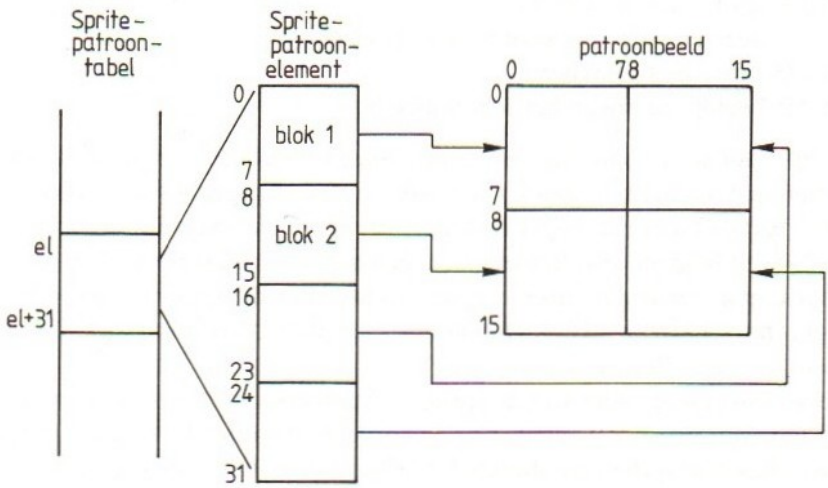
Tot zover het basis-grafische systeem in de basisinstelling. We komen nu op het sprite-systeem.

Het sprite-systeem wordt bestuurd door de sprite-patroontabel en de sprite-vlaktabel. Het sprite-systeem kent echter als beperking dat het niet mogelijk is meer dan vier sprites op één horizontale lijn tegelijk op het scherm te zetten.
De sprites van het sprite-systeem kunnen twee afmetingen hebben: 8×8 of 16×16 patroonpunten. Elke patroonpunt kan gecodeerd worden door één bit; de elementen van de sprite-patroontabel zijn 8 of 32 bytes groot, afhankelijk van het formaat van de sprites. Aangezien de sprite-patroontabel in zijn geheel altijd 2048 bytes groot is, heeft de tabel, afhankelijk van het formaat van de sprites, 256 elementen (genummerd van 0 tot en met 255) of 64 elementen (genummerd van 0 tot en met 63).

Als het formaat van de sprites 8×8 is, wordt het patroonbeeld gecodeerd op de manier zoals aangegeven in afbeelding 3.2. Als het formaat 16×16 is, is de codering zoals is aangegeven in afbeelding 3.4.



Afb. 3.3. Basisinstelling van het totale systeem



Afb. 3.4. Verband tussen 16×16 spritepatroon en codering

De sprite-vlaktabel bevat 32 elementen, genummerd van 0 tot en met 31. De elementen van de sprite-vlaktabel bevatten de plaats waar een sprite-patroon op het schermbeeld komt, het nummer van het gebruikte sprite-patroon, en de kleur die het patroonbeeld heeft.
De plaats waar het patroon op het scherm komt, wordt aangegeven met een coördinatenpaar. De coördinaten verwijzen niet naar de posities die in de naamtabel worden gebruikt, maar naar beeldpunten. Aangezien elke positie een patroon herbergt van 8×8 beeldpunten, zijn de coördinaten te relateren aan de schermposities door de coördinaatwaarde door acht te delen. Er zijn 32×8=256 punten, genummerd van 0 tot en met 255, op een horizontale lijn en 24×8=192 punten, genummerd van 0 tot en met 191, op een verticale lijn.
Een element van de sprite-vlaktabel bestaat uit vier bytes. Deze bytes hebben de volgende betekenis:

Byte 0: x-coördinaat
Byte 1: y-coördinaat
Byte 2: sprite-patroonnummer
Byte 3: kleur

De x- en y-coördinaten zijn 1-byte-getallen met een waarde tussen 0 en 255 (inclusief grenzen). Door gebruik van een extra bit in byte 3 (zie verderop) heeft de x-coördinaat in feite een bereik van –32 tot en met 255. Zoals gezegd zijn er maar 255 horizontale beeldpunten; als de x-coördinaat een waarde heeft kleiner dan 0, duidt dit op een virtueel niet-bestaand beeldpunt links van het scherm. Door gebruik van virtuele beeldpunten is het mogelijk een sprite-patroon aan de rand van het beeldscherm te laten verdwijnen.
Er zijn slechts 192 verticale beeldpunten; de waarden van de y-coördinaat die niet op een beeldpunt duiden, hebben een andere betekenis.

0 t/m 191	beeldpunt met dat nummer
192 t/m 207	sprite niet op scherm
208	deze en volgende sprites niet op scherm
209 t/m 233	sprite niet op scherm
234 t/m 255	beeldpunt boven het schermbeeld.

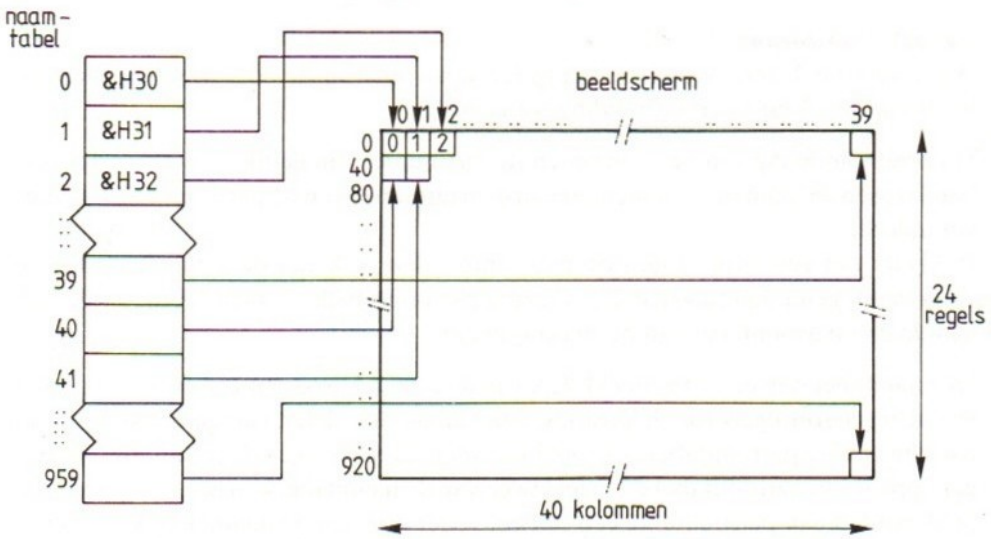
De waarde 208 voor de y-coördinaat heeft een bijzonder effect: niet alleen komt deze sprite niet op het scherm te staan, maar ook worden alle sprites van elementen van de sprite-vlaktabel met een hoger nummer van het scherm verwijderd.
De waarden 234-255 hebben hetzelfde effect als een negatieve waarde van de x-coördinaat: het sprite-patroon begint op een virtueel punt, dat in dit geval boven (en niet links van) het echte schermbeeld ligt. Het nummer van dit virtuele punt is gelijk aan de waarde van de y-coördinaat –256.
Het derde byte van een element van de sprite-vlaktabel bevat het sprite-nummer, ofte wel het nummer van het element van de sprite-patroontabel. Als het sprite-formaat 8×8 is, bevat de sprite-patroontabel 256 elementen; in dat geval worden alle acht bits van byte twee gebruikt voor het nummer. Als het sprite-formaat 16×16 is,

worden alleen bits 7-2 gebruikt voor het sprite-patroonnummer; de sprite-patroontabel bevat dan maar 64 elementen.
Byte drie (het vierde byte) van een element van de sprite-patroontabel is op de volgende manier ingedeeld:

Bit 7 hulpbit x-coördinaat T1 = –32
Bit 6-4 ongebruikt
Bit 3-0 kleur van het patroonbereik: 0–15

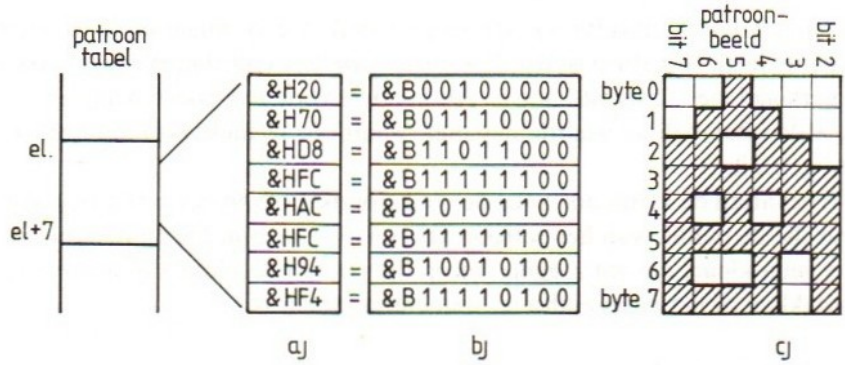
Het eerste bit wordt gebruikt om het bereik van de x-coördinaat uit te breiden met een negatief deel. Het effect is, dat de waarde voor de x-coördinaat in het eerste byte wordt verlaagd met 32 als bit 7 aan is. Als het byte voor de x-coördinaat zelf nu 0–31 bevat, is de uiteindelijke waarde negatief, waardoor virtuele beeldpunten kunnen worden gebruikt.
Bits 3-0 bevatten de kleurcode voor de voorgrondkleur van het sprite-patroon, dus de kleur die de punten van het patroon krijgen die aan zijn. De punten die uit zijn, hebben geen kleur, dat wil zeggen dat op die punten de kleur van achterliggende sprite-vlakken of het basis-grafische systeem zichtbaar is.

Variatie 1: vereenvoudiging
De basisinstelling van de VDP, zoals besproken in de vorige paragraaf, kent een aantal variaties. Hoe de variaties ingesteld worden, komt in par. 3.3 aan de orde.
De eerste van de variaties heeft als effect dat het totale grafische systeem aanzienlijk eenvoudiger van bediening, maar ook beperkter van mogelijkheden wordt.
De indeling van het scherm in posities verandert. De rand bestaat niet meer. Er zijn nu 40 horizontale posities, dus in totaal 40×24=960 posities, genummerd van 0 tot en met 959. De naamtabel heeft dan ook 960 elementen. Het verband tussen schermposities en naamtabel is vastgelegd in afbeelding 3.5.



Afb. 3.5. Verband tussen naamtabel en schermposities in variatie 1

De uitbreiding van het aantal posities op een regel is bereikt door de patronen te verkleinen. De elementen van de naamtabel bestaan nog steeds uit één byte. De patroontabel telt dan ook nog steeds 256 elementen, genummerd van 0 tot en met 255. De patronen uit de patroontabel zijn nu echter 6×8 patroonpunten groot, in plaats van 8×8. De codering van deze 6×8 punten gebeurt alsof het patroon nog steeds 8×8 groot is, echter de twee rechter kolommen worden niet gebruikt. Dit principe is weergegeven in afbeelding 3.6.



Afb. 3.6. Verband tussen patroontabel en patroonbeeld in variatie 1

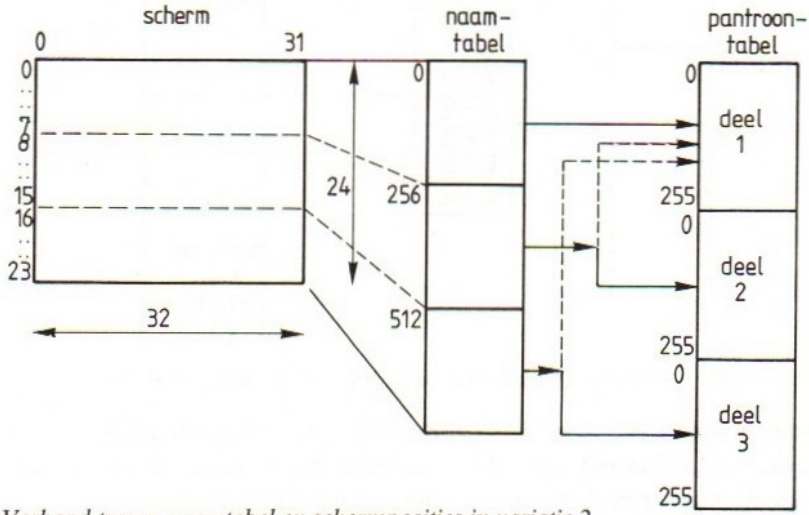
De kleurentabel wordt niet gebruikt. In plaats daarvan wordt voor elk patroon dezelfde voor- en achtergrondkleur gebruikt. Deze kleuren worden niet in een tabel opgeslagen.
Het sprite-systeem kan in variatie 1 niet worden gebruikt. De sprite-patroontabel en sprite-vlaktabel zijn niet aanwezig.

Variatie 2: driedeling
Waar variatie 1 een vereenvoudiging ten opzichte van de basisinstelling betreft, heeft variatie 2 juist een uitbreiding tot gevolg.

De positie-indeling van het scherm en de naamtabel zijn gelijk gebleven. Het verband tussen de 'namen' – elementen van de naamtabel – en de patroontabel is echter veranderd.
In plaats van één, zijn er nu drie patroontabellen. Elk van deze tabellen telt 256 elementen, genummerd van 0-255. De drie patroontabellen zijn in elk opzicht gelijk aan de ene patroontabel van het basissysteem.

De naamtabel is in drieën gedeeld. Elk van de drie delen omvat 256 elementen, ofte wel acht schermregels van 32 posities. Elk van de drie delen van de naamtabel kan aan een andere patroontabel gekoppeld worden. Deel één van de naamtabel is altijd gekoppeld aan patroontabel één; deel twee van de naamtabel kan naar keuze gekoppeld worden aan patroontabel één of twee; deel drie van de naamtabel kan gekoppeld worden aan patroontabel één of drie.

Een koppeling van een deel van de naamtabel aan een andere patroontabel betekent dat in het schermdeel van dat deel van de naamtabel bij elke naam een ander patroon hoort.
Deze situatie is uitgebeeld in afbeelding 3.7.

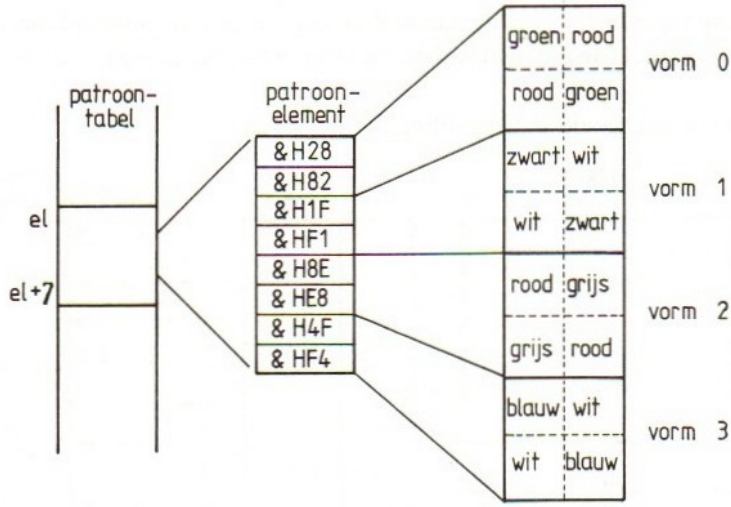


Afb. 3.7. Verband tussen naamtabel en schermposities in variatie 2

De kleurentabel is in variatie 2 aanzienlijk uitgebreid. De elementen van de kleurentabel zijn acht bytes groot in plaats van één byte; en de tabel heeft meer elementen dan in het basis systeem.
Elk van de bytes van de kleurentabel hoort bij één byte van de patroontabel. De kleuren worden dus bepaald per acht naast elkaar liggende patroonpunten.
Het verband tussen patroonnummer en element van de kleurentabel is veranderd. Ten eerste zijn er drie kleurentabellen, één per patroontabel; ten tweede wordt het patroonnummer gemaskerd voordat het bijbehorende element van de kleurentabel opgezocht wordt. De maskering gebeurt met een 5-bits masker over bits 4-0 van het patroonnummer.
Het sprite-systeem is in deze variatie precies hetzelfde als in de basisinstelling van de VDP.

Variatie 3: veelkleurig
De naamtabel is in deze variatie onveranderd gebleven. De patronen en de patroontabel zijn sterk veranderd. De kleurentabel wordt in deze variatie niet gebruikt. Het sprite-systeem is hetzelfde gebleven.
Een patroon is niet langer 8×8 pixels groot, maar is in deze variatie 8×32 pixels groot. De pixels worden bestuurd in blokken van 4×4. Van elk van deze blokken wordt de kleur via het patroon vastgesteld.

De manier waarop deze zaken in het patroon zijn gecodeerd, is weergegeven in afb. 3.8.



Afb. 3.8. Verband tussen patroontabel en patroonbeeld in variatie 2

De namen van de naamtabel verwijzen niet langer naar een heel patroon, maar naar een gedeelte van het patroon. Welk gedeelte dat is, hangt af van de schermregel waarmee de naam in verband staat en dus van het elementnummer van de naam zelf. Elk patroon bestaat uit vier delen, namelijk vier boven elkaar liggende blokken van 8×8 pixels. Deze blokken kunnen we nummeren van 0-3, te beginnen bij het bovenste blok. Het nummer van het patroonblok dat bij een bepaalde naam hoort, kan dan uit het elementnummer van de naam berekend worden met de formule elnr=MOD (32*4).

Combinaties

De variaties die in de vorige paragrafen besproken zijn, kunnen worden gecombineerd. Het resultaat is dan een nieuwe variatie op het basissysteem, waarin de afwijkingen van beide samenstellende variaties terug te vinden zijn.
Hier volgt een lijst van combinatiemogelijkheden en de resulterende eigenschappen van het systeem.

Variatie 1+2: drie patroontabellen, en een driedeling van de naamtabel. Patronen worden maar voor 6/8 deel gebruikt. De kleuren worden globaal vastgesteld: de kleurentabel wordt niet gebruikt. Het sprite-systeem is uitgeschakeld.

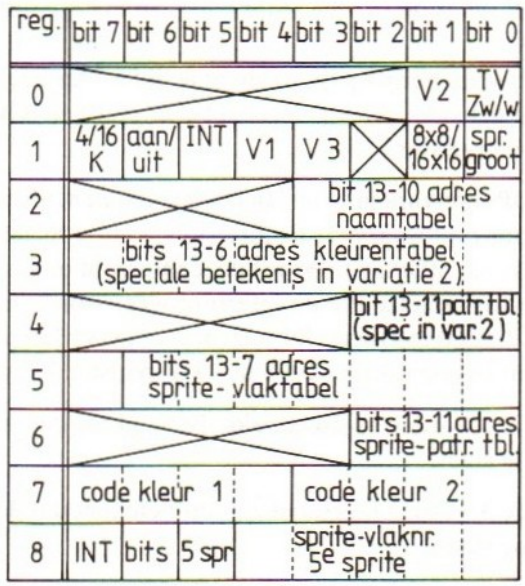
Variatie 1+3: intern conflict van de VDP. Het scherm vertoont permanent kolommen van afwisselend voor- en achtergrondkleur. Het sprite-systeem is uitgeschakeld.

Variatie 2+3: drie patroontabellen, en een driedeling van de naamtabel. Patronen beslaan 8×32 pixels, onderverdeeld in blokken van 4×4. De kleurentabel wordt niet gebruikt. Het sprite-systeem is normaal werkzaam.

Variatie 1+2+3: intern conflict van de VDP. Het scherm vertoont permanent kolommen van afwisselend voor- en achtergrondkleur. Het sprite-systeem is uitgeschakeld.

3.2.2 De VDP-registers

De VDP heeft negen registers, genummerd van 0 tot en met 8. Registers 0–7 kunnen zowel worden gelezen als beschreven; register 8 kan alleen worden gelezen. De VDP-registers bevatten informatie over instellingen van de VDP en de beginadressen van de verschillende tabellen in het videogeheugen. Afbeelding 3.9 geeft een algemeen overzicht van de betekenis van de registers.



Afb. 3.9. Overzicht van de VDP-registers

Register 0

De betekenis van de bits van register 0 is als volgt:

Bit 7-2 ongebruikt
Bit 1 variatie 2 ja/nee 1 = ja
Bit 0 externe invoer ja/nee 1 = ja

Met bit 1 kan variatie 2 worden ingeschakeld; dit betekent dat de VDP in variatie 2 werkt als dit bit hoog staat.

Het hoog zetten van bit 0 heeft tot gevolg dat het TV-beeld alleen nog in zwart-wit zichtbaar is, zelfs op een kleurentelevisie. De toepassing van dit bit is niet duidelijk.

Register 1

De betekenis van de bits van register 1 is als volgt:

Bit 7	videogeheugen 4 kbyte/16 kbyte	1 = 16 kbyte
Bit 6	beeldscherm aan/uit	1 = aan
Bit 5	interrupt aan/uit	1 = aan
Bit 4	variatie 1 ja/nee	1 = ja
Bit 3	variatie 3 ja/nee	1 = ja
Bit 2	ongebruikt	
Bit 1	sprite-grootte 8×8 of 16×16	1 = 16×16
Bit 0	sprites vergroot ja/nee	1 = ja

Bit 7 geeft aan of er 4 kbyte dan wel 16 kbyte videogeheugen beschikbaar is. In de MSX is dit altijd 16K, en is dit bit dus altijd hoog. Het laag zetten van het bit heeft geen effect.
Bit geeft aan of het beeldscherm op dit moment actief is. Als het beeld niet actief is, wordt de inhoud van het videogeheugen geheel genegeerd en is het scherm blank in één kleur: kleur 2 van register 7.

Bit 5 schakelt de VDP-interrupt in of uit. In het standaard MSX-systeem is de VDP-interrupt de enige interrupt waarvan alle interrupt-gestuurde acties, zoals het scannen van het toetsenbord, afhangen. Wordt deze interrupt uitgeschakeld dan 'loopt het systeem vast' tenzij in hetzelfde de interrupt weer wordt ingeschakeld. Het is overigens beter om de Z80-instructie DI te gebruiken voor het uitschakelen van de interrupts dan om dit te doen door bit 5 van VDP-register 1 laag te zetten.

Met bit 4 kan variatie 1 worden ingeschakeld; dit betekent dat de VDP in variatie 1 werkt als dit bit hoog staat.

Met bit 3 kan variatie 3 worden ingeschakeld; dit betekent dat de VDP in variatie 3 werkt als dit bit hoog staat.

Als bit 1 hoog staat, neemt de VDP aan dat sprite-patronen 16×16 pixels groot zijn. Dit heeft ook gevolgen voor de beginadressen van de elementen van de sprite-patroontabel. Uiteraard heeft dit alleen effect als het sprite-systeem werkzaam is, dus niet in variatie 1.
Als bit 0 hoog staat, worden sprites vergroot weergegeven. Dit wil zeggen dat elk pixel van een sprite op het scherm de grootte krijgt van een blokje van 2x2 pixels. Uiteraard heeft dit alleen effect als het sprite-systeem werkzaam is, dus niet in variatie 1.

Register 2

De betekenis van de bits van register 2 is als volgt:

Bit 7-4 ongebruikt
Bit 3-0 bits 13-10 adres naamtabel bereik: 0–15

Bits 3-0 van register 2 vormen bits 13-10 van het beginadres van de naamtabel in het

videogeheugen. Bits 9-0 van dit beginadres zijn 0. Het beginadres kan dus berekend worden door de waarde van bits 3-0 van dit register te vermenigvuldigen met &H400 (=1024).

Register 3

In het basissysteem van de VDP vormt de waarde van VDP-register 3 een deel van het beginadres van de kleurentabel. Bits 7-0 van het register komen overeen met bits 13-6 van dit beginadres. Het adres kan dus berekend worden door de waarde van het register te vermenigvuldigen met &H40 (=64).

In variatie 2 van het videosysteem hebben de bits van register 3 een andere betekenis:

Bit 7 bit 13 adres kleurtabel 1
Bit 6 deel 2 naamtabel=kleurtabel 2 1=ja
Bit 5 deel 3 naamtabel=kleurtabel 3 1=ja
Bit 4-0 masker voor patroonnummer

In deze variatie zijn er 3 kleurentabellen. De tabellen sluiten op elkaar aan in het videogeheugen. Elk van de tabellen is 2 kbyte groot.

Bit 7 is gelijk aan bit 13 van het beginadres van de kleurentabellen. Dit beginadres is dus 0000 of 2000.

Bit 6 geeft aan of deel 2 van de naamtabel is gekoppeld aan kleurentabel 1 dan wel 2 – met andere woorden, of tabel 2 wordt gebruikt. Zie par. 3.1.3.

Bit 5 doet hetzelfde voor kleurentabel 3.

Bits 4-0 vormen een masker over bits 4-0 van een patroonnummer bij het bepalen van het elementnummer van de kleur die bij dat patroon hoort. Zie par. 3.1.3.

Register 4

In het basissysteem van de VDP hebben de bits van register 4 de volgende betekenis:

Bit 7-3 ongebruikt
Bit 2-0 bits 13-11 adres patroontabel bereik: 0–7

Bits 2-0 van register 4 vormen bits 13-11 van het beginadres van de patroontabel in het videogeheugen. Het beginadres van de patroontabel kan dus berekend worden door de waarde van bits 2-0 van het register te vermenigvuldigen met &H2008=512).

In variatie 2 van het videosysteem hebben de bits van VDP-register 4 een andere betekenis:

Bit 7-3 ongebruikt
Bit 2 bit 13 adres patroontabel
Bit 1 deel 2 naamtabel=patroontabel 2 1 = ja
Bit 0 deel 3 naamtabel=patroontabel 3 1 = ja

In deze variatie zijn er drie patroontabellen. De tabellen sluiten op elkaar aan in het videogeheugen. Elk van de tabellen is 2 kbyte groot.

Bit 2 is gelijk aan bit 13 van het beginadres van de patroontabellen. Dit beginadres is dus &H0 of &H2.

Bit 1 geeft aan of deel 2 van de naamtabel gekoppeld is aan patroontabel 1 dan wel 2 – met andere woorden, of tabel 2 wordt gebruikt. Zie par. 3.1.3.

Bit 0 doet hetzelfde voor patroontabel 3.

Register 5
De bit van VDP-register 5 hebben de volgende betekenis:

Bit 7 ongebruikt
Bit 6-0 bits 13-7 van adres sprite-vlaktabel

Bits 6-0 van register 5 vormen bits 13-7 van het beginadres van de sprite-vlaktabel in het videogeheugen. Bits 6-0 van dit beginadres zijn 0. Het beginadres kan dus worden berekend door de waarde van bits 6-0 van dit register te vermenigvuldigen met &H80 (=128).

Register 6
De bits van VDP-register 6 hebben de volgende betekenis:

Bit 7-3 ongebruikt
Bit 2-0 bits 13-11 adres sprite-patroontabel

Bits 2-0 van register 6 vormen bits 13-11 van het beginadres van de sprite-patroontabel in het videogeheugen. Bits 10-0 van dit beginadres zijn 0. Het beginadres kan dus berekend worden door de waarde van bits 2-0 van dit register te vermenigvuldigen met &H800 (=2048).

Register 7
De bits van VDP-register 7 hebben de volgende betekenis:

Bit 7-4 code voor kleur 1 bereik: 0-15
Bit 3-0 code voor kleur 2 bereik: 0-15

Register 7 bevat twee kleurcodes. De kleuren die hierin gecodeerd zijn worden gebruikt als standaardkleur en/of randkleur, afhankelijk van de instelling van het videosysteem. Kleur 1 wordt gebruikt als voorgrondkleur in variatie 0. Kleur 2 is de randkleur in het basissysteem, en de achtergrondkleur in variatie 0.

Register 8
VDP-register 8 kan niet beschreven worden, alleen gelezen. De waarde ervan is een aanduiding van de toestand van de VDP.

De bits van register 8 hebben de volgende betekenis:

Bit 7 interrupt opgetreden	1=ja
Bit 6 sprite-botsing opgetreden	1=ja
Bit 5 vijf sprites horizontaal	1=ja
Bit 4-0 sprite-vlaknr vijfde sprite	

Bit 7 is hoog als er een VDP-interrupt is opgetreden en register 8 daarna nog niet is uitgelezen. Het bit wordt laag gezet bij het uitlezen van het register.

Bit 6 is hoog als de VDP een sprite-botsing heeft ontdekt, en het register daarna nog niet is uitgelezen. De detectie op sprite-botsingen vindt plaats bij elke interrupt. Het bit wordt teruggezet bij het uitlezen van register 5.

Bit 5 is hoog als er ergens op het scherm vijf of meer sprites op een horizontale lijn voorkomen. Als dit het geval is, worden de vijfde en alle volgende sprites niet of gedeeltelijk weergegeven.

Bits 4-0 geven het nummer van het eerste sprite-vlak waarop een sprite voorkomt die niet of gedeeltelijk wordt weergegeven vanwege de beperking van 4 sprites op een horizontale lijn. Het nummer in bits 4-0 is alléén geldig als bit 5 hoog is!

3.3 De besturing van de VDP

De vorige paragrafen zijn gewijd aan de werking van de VDP als los onderdeel, alsof de chip geen deel uitmaakt van een computersysteem. In deze paragraaf gaan we in op de manier waarop de VDP in het MSX-systeem is ingepast en kan worden gebruikt.

3.3.1 Schermmodi

De basisinstelling van het videosysteem van de VDP, en de variaties op die basisinstelling, zijn binnen de MSX vertaald naar zogenaamde schermmodi. Elke schermmodus wordt op een bepaalde manier behandeld, en de verschillende schermmodi dienen verschillende doelen.

Het is van belang om onderscheid te blijven maken tussen de mogelijkheden van de VDP en het gebruik dat daarvan binnen de MSX wordt gemaakt. Niet alle capaciteiten van de VDP worden benut. Door het gebruik van de schermmodi wordt op een specifieke manier gebruik gemaakt van de VDP-tabellen, en worden andere manieren min of meer uitgesloten.
Het is echter moeilijk om los te komen van de beperkingen van de schermmodi. De ondersteuning voor de schermmodi komt namelijk vanuit het Operating System (OS) en het is erg ingewikkeld om programma's te schrijven zonder gebruik te maken van het OS. Hierop komen we terug in par. 6.2.

De schermmodi zijn genummerd van 0 tot en met 3. Schermmodi 0 en 1 worden wel tekstmodi genoemd; schermmodi 2 en 3 zijn grafische modi. We zullen de schermmodi hier onder de loep nemen.

Schermmodus 0

Schermmodus 0 maakt gebruik van variatie 1 van de VDP. Het is een tekstmodus. In schermmodus 0 wordt de patroontabel gevuld met de patronen voor de tekens van de uitgebreide ASCII-tabel van de MSX. Er zijn precies 256 patronen beschikbaar voor 256 ASCII-tekens (inclusief de grafische extensie-symbolen).
De inhoud van de patroontabel is permanent, tenzij een gebruiker direct in het videogeheugen gaat schrijven in het gebied van de patroontabel. Dit laatste is een manier om tekens van de ASCII-tabel voor eigen doeleinden te herdefiniëren. Overigens is er daarvoor nog een andere methode; zie par. 6.2.6.

De naamtabel, die overeenkomt met posities op het scherm, wordt gebruikt om tekst op het scherm zichtbaar te maken. Alles wat in schermmodus 0 op het scherm plaatsvindt, gebeurt door verandering van de naamtabel.
Het voordeel van gebruik van variatie 1 van het videosysteem is dat er 40 kolommen beschikbaar zijn. Een nadeel is dat niet de hele patroonbeelden van de patroontabel worden gebruikt, maar slechts 6/8 deel daarvan. De patronen van de gewone ASCII-tekens zijn zo gedefinieerd dat alle pixels ervan staan in het 6×8 gebied dat in variatie 1 werkzaam is.
Het gebruik van kleuren is in schermmodus 0 maar beperkt mogelijk, omdat de kleurentabel in variatie 1 niet gebruikt wordt. Er kunnen nooit meer dan twee kleuren tegelijk op het scherm staan.

De standaard-instelling van de VDP-registers in schermmodus 0 is als volgt (de registerwaarden zijn hexadecimaal):

regnr	*waarde*
0	00
1	F0
2	0
3	00
4	01
5	00
6	00
7	F4

Schermmodus 1

Schermmodus 1 werkt met het basissysteem van de VDP. Het is een tekstmodus.

Net als in schermmodus 0 is de patroontabel gevuld met de patronen van de ASCII-tekens. De patronen worden nu echter in hun geheel (8×8 pixels) op het scherm gezet.
Het gebruik van de naamtabel in schermmodus 0 is kenmerkend voor een tekstmodus. De naamtabel wordt in schermmodus 1 op dezelfde manier gebruikt.
Anders dan in schermmodus 0 zijn er in deze modus wel verscheidene kleuren beschikbaar; het basissysteem van de VDP kent immers een kleurentabel. De kleuren worden echter niet werkelijk door het OS ondersteund: de elementen van de tabel

kunnen niet afzonderlijk worden veranderd. Bij verandering van kleuren door OS-routines wordt altijd de hele tabel behandeld. De enige manier om verschillende ASCII-tekens verschillende kleuren te geven is direct naar het videogeheugen te schrijven. De kleur is dan gebonden aan het ASCII-teken, niet aan een schermpositie.

Kleuren kunnen alleen per groep van 8 tekens veranderd worden, vanwege de koppeling tussen de patroontabel en de kleurentabel; zie par. 3.2.1.
Het sprite-systeem is volledig werkzaam in schermmodus 1 en wordt ondersteund door het OS.
De standaardinstelling van de VDP-registers in schermmodus 1 is als volgt (de registerwaarden zijn hexadecimaal):

regnr	*waarde*
0	00
1	E0
2	6
3	80
4	00
5	36
6	07
7	04

Schermmodus 2
Schermmodus maakt gebruik van variatie 2 van het videosysteem van de VDP. Het is een grafische modus.
In grafische modi – schermmodi 2 en 3 – is niet de patroon- maar de naamtabel constant. De invulling van de naamtabel is zo gekozen dat de patronen van alle schermposities onafhankelijk van elkaar gekozen en veranderd kunnen worden. Door verandering van patroon- en kleurentabel kunnen er tekeningen op het scherm gezet worden.

In schermmodus 2 is de naamtabel gevuld met een reeks van alle patroonnummers. Omdat variatie 2 van het videosysteem wordt gebruikt, waarin alle delen van de naamtabel naar een andere patroon- en kleurentabel kunnen verwijzen, kan aan elke schermpositie een eigen patroon en kleur gekoppeld worden.
Elk deel van de naamtabel telt 256 elementen; deze krijgen de waarden 0–255. Tevens worden de VDP-registers 3 en 4 zo ingesteld dat de verschillende delen van de naamtabel inderdaad naar verschillende kleur- en patroontabellen verwijzen. Bij elk pixel op het scherm hoort nu een ander bit in de patroontabel. Bij elke rij van 8 pixels hoort een andere voor- en achtergrondkleur.
De uitvoering van grafische commando's gebeurt nu door patronen en kleuren te veranderen. Het schrijven van tekst op het scherm moet gebeuren door de letters te tekenen, aangezien de ASCII-tekens niet meer voorhanden zijn.

De standaardinstelling van de VDP-registers in schermmodus 2 is als volgt (de registerwaarden zijn hexadecimaal):

regnr	*waarde*
0	02
1	E0
2	6
3	FF
4	03
5	36
6	07
7	04

Schermmodus 3

Schermmodus 3 maakt gebruik van variatie 3 van het videosysteem van de VDP. Het is een grafische modus.

Net als in schermmodus 2 is de invulling van de naamtabel constant. Tekeningen worden gemaakt door de patroontabel te veranderen.

De invulling van de naamtabel is zodanig dat steeds een heel patroon (van 8×32 pixels, zie par. 3.1.4) in zijn geheel op het scherm staat. Dit vereenvoudigt het tekenwerk. De naamtabel is zodoende gevuld met series patroonnumers van 32 nummers achter elkaar, één voor elke positie op een regel, die telkens vier keer voorkomen om alle vier delen van het patroon direct onder elkaar op het scherm te krijgen.

Overigens is er geen enkele mogelijkheid wat betreft tekeningen of kleuren die schermmodus 3 wel heeft en schermmodus 2 niet. Schermmodus 3 is echter wat sneller te besturen.

De standaardinstelling van de VDP-registers in schermmodus 3 is als volgt (de registerwaarden zijn hexadecimaal):

regnr	*waarde*
0	00
1	E8
2	2
3	00
4	00
5	36
6	07
7	04

3.3.2 Poortadressering

De VDP heeft de volgende poortadressen tot zijn beschikking:

Adres	*Modus*	*Toepassing:*
98	L/S	videogeheugeninhoud
99	L/S	status/commandoregister

Intern houdt de VDP twee adressen bij: een adres van de geheugenplaats in het videogeheugen waar gelezen wordt (het leesadres), en een adres van de geheugenplaats waar geschreven wordt (het schrijfadres).

Door het lezen van de I/O-poort op adres 98 wordt de waarde gelezen van de videogeheugenplaats waarnaar verwezen wordt door het leesadres, en wordt dit adres opgehoogd. Door het schrijven van een nieuwe waarde naar poortadres 98 wordt deze waarde neergezet op de plaats waarnaar verwezen wordt door het schrijfadres, en wordt dit adres opgehoogd.
Door de I/O-poort op adres 99 te lezen, krijgt men de waarde van VDP-register 8. Door een commando naar poortadres 99 te schrijven kan men de waarden van de VDP-registers of het lees- of schrijfadres veranderen.

Een commando dat naar poortadres 99 wordt geschreven, bestaat altijd uit twee bytes. Het tweede byte geeft aan om wat voor commando het gaat. De commando's zijn:
- veranderen van een VDP-register: het eerste byte van het commando is de nieuwe waarde voor het register; het tweede byte is het nummer van het register, aangevuld door een 1 in bit 7.
- veranderen van het leesadres: het eerste byte is het lage-orde-byte van het adres, het tweede byte is het hoge-orde-byte.
- veranderen van het schrijfadres: het eerste byte is het lage-orde-byte van het adres, het tweede byte is het hoge-orde-byte, aangevuld met een 1 in bit 6.

Het tweede byte van een commando ziet er als volgt uit:

Bit 7	adres/registerverandering	1 = register
Bit 6	lees/schrijfadres	1 = schrijf
Bit 5-0	hi-byte adres/regnr	

Bit 6 is alleen geldig als bit 7 hoog is.

3.3.3 OS-routines

Voor het lezen en schrijven van het videogeheugen en de VDP-registers zijn een aantal Operating System-routines beschikbaar. Het is over het algemeen zinvol om deze routines te gebruiken in plaats van zelf direct naar de I/O-poorten te schrijven. Het gebruik van OS-routines bevordert de begrijpelijkheid van een programma en vergroot de waarschijnlijkheid dat het programma op een volgende versie van de MSX zal blijven werken.

Is het vanwege de snelheid toch wenselijk om gegevens voor het videogeheugen direct via de VDP-poorten te schrijven of te lezen, dan is het aan te raden om de poortadressen niet als constante in het programma op te nemen, maar ze in plaats daarvan aan het OS te ontlenen. Zo is nog steeds een zekere mate van compatibiliteit aanwezig. Speciaal voor dit doel zijn op adressen 0006 en 0007 van de OS-ROM de adressen van de videogeheugen-schrijfpoort en -leespoort opgeslagen. In het huidige MSX-systeem zijn deze adressen beide gelijk aan 98.

Een andere reden waarom het zinvol is om OS-routines te gebruiken, is het feit dat de waarden van VDP-registers niet gelezen kunnen worden. Het OS houdt daarom een duplicaat van alle VDP-registers bij in het RAM-geheugen (zie par. 6.4.1, RG0SAV (F3DF) t/m STATFL (F3E7)). Alle waarden die naar VDP-registers worden geschreven, worden ook in deze adressen gezet. Deze RAM-adressen geven dus de huidige waarde van de VDP-registers, maar alleen als OS-routines worden gebruikt voor het schrijven van nieuwe waarden naar de registers.

De OS-routines die direct de VDP aanspreken, zijn de volgende:

adres	*naam*	*doel*
0047	WRTVDP	schrijft naar VDP-register
004A	RDVRM	leest geadresseerd videogeheugen
004D	WRTVRM	schrijft geadresseerd videogeheugen
0050	SETRD	schrijft leesadres
0053	SETWRT	schrijft schrijfadres
0056	FILVRM	vult blok videogeheugen met waarde
0059	LDIRMV	kopieert videogeheugen naar gewoon
005C	LDIRVM	kopieert gewoon geheugen naar video

Details over deze routines staan in par. 6.3. Andere OS-routines die op het videogedeelte van de MSX-computer betrekking hebben, maar die gespecialiseerd zijn op het gebied van de scherm-editor of het grafische systeem, komen aan de orde in par. 6.2.6 en par. 6.2.7.

3.3.4 BASIC-aansturing

Ook vanuit BASIC kunnen het videogeheugen en de VDP-registers worden aangesproken, afgezien van het gewone gebruik van de scherm-editor of het grafische systeem. Voor dat doel zijn een aantal sleutelwoorden toegevoegd.

Het verdient aanbeveling om dubbel voorzichtig te zijn bij het gebruik van deze sleutelwoorden, omdat door onzorgvuldig gebruik het MSX-systeem onhandelbaar wordt – bijvoorbeeld als variaties van het videosysteem worden geprobeerd door rechtstreeks naar de VDP-registers te schrijven.

De BASIC-sleutelwoorden die direct op de VDP betrekking hebben, zijn:

VDP: wordt gebruikt om de VDP-registers aan te duiden. De registers kunnen met behulp van dit sleutelwoord gelezen en beschreven worden. Bij het lezen worden niet de werkelijke waarden van de registers gelezen, aangezien dit niet mogelijk is; in plaats daarvan worden de duplicaten van de registers gelezen die in RAM worden bewaard.

VPEEK: functie met behulp waarvan het videogeheugen kan worden gelezen. De manier van gebruik van VPEEK is zoals het gebruik van PEEK voor gewoon geheugen.

VPOKE: wordt gebruikt om direct naar het videogeheugen te schrijven. De manier van gebruik is zoals het gebruik van POKE voor gewoon geheugen.

Naast deze sleutelwoorden kunnen natuurlijk ook INP en OUT worden gebruikt om de poorten direct te adresseren. Hiervoor gelden echter dezelfde bezwaren als voor het direct adresseren van de poorten in machinetaal; zie par. 3.3.3.

4 Audio

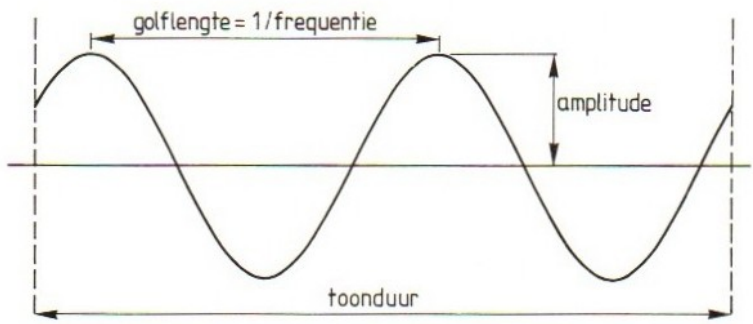
Vergeleken met sommige andere huiscomputers kent de MSX-computer nogal uitgebreide geluidsmogelijkheden. Vergeleken met computers met méér mogelijkheden is het geluid op de MSX tamelijk gemakkelijk te besturen. Zodoende is het audio-subsysteem (met een mooi woord) van de MSX een compromis tussen veelzijdigheid en gebruiksgemak.
Het besturen en genereren van geluid in de MSX gebeurt allemaal via één chip, de zgn. Programmable Sound Generator (Programmeerbare Geluids Generator), kortweg PSG.

In dit hoofdstuk zullen we achtereenvolgens ingaan op de mogelijkheden van de PSG en de besturing van de PSG. Eerst zullen we echter de achterliggende theorie van het produceren van geluid bestuderen.

4.1 Algemene muziektheorie

Om de geluidsmogelijkheden van de PSG op een logische en volledige manier te behandelen, moeten we eerst een klein stukje algemene muziektheorie behandelen. Mocht deze theorie al bekend zijn, dan is het wellicht toch nuttig om het volgende door te nemen, om een beter inzicht in de mogelijkheden van de PSG te verkrijgen.

Een zuivere toon (wat we met 'zuiver' bedoelen, komt straks aan de orde) heeft drie eigenschappen: toonhoogte, volume en duur (lengte). Fysisch gezien is een toon een golfpatroon dat zich door de lucht voortplant; de frequentie (of omgekeerd, de golflengte) van de golf bepaalt de toonhoogte, de amplitude bepaalt het volume, en de lengte van het golfpatroon de toonduur. Zie afbeelding 4.1.



Afb. 4.1. Een geluidsgolf

Het is relatief eenvoudig om een computer zuivere tonen te laten genereren. Bovenstaande drie eigenschappen zijn gemakkelijk te besturen.

Geluiden die we om ons heen horen, zijn niet zuiver in de zin waarin we dat woord hierboven gebruikt hebben: bestaande uit één frequentie. We krijgen namelijk te maken met ruis en boventonen (overtonen).
Met ruis worden alle geluiden aangeduid waaraan geen toonhoogte te onderscheiden is: van het ruisen van de zee en het fluisteren van de wind in de bomen tot het geraas van een vliegtuig of het geluid van storing op een radio. Natuurkundig gezien, is ruis samengesteld uit een heleboel zuivere tonen van heel hoge frequentie.

Boventonen zijn tonen die de zgn. klankkleur van een geluid of muziekinstrument bepalen: tonen met een andere frequentie (meestal hogere, vandaar boventonen), en zachter dan de hoofdtoon.
De meeste geluiden die we horen, zijn samengesteld uit een hoofdtoon, boventonen en ruis. Daarnaast kennen geluiden nog een klankverloop, een toonhoogteverloop en een volumeverloop.
Het zal duidelijk zijn dat dit alles tot gevolg heeft dat het erg moeilijk is om natuurgeluiden synthetisch (kunstmatig) te reproduceren. Om dit enigszins mogelijk te maken, is een synthesizer van vele duizenden guldens een vereiste.

Het geluid van de MSX kan maar op enkele van bovenstaande punten bestuurd worden. Met de bestaande mogelijkheden is echter ook al heel wat te bereiken.

De PSG heeft drie 'stemmen'. Hij kan drie zuivere tonen produceren, onafhankelijk van elkaar in toonhoogte en volume. Elk van de stemmen kan in plaats van een toon, of tegelijkertijd, ruis produceren. De 'kwaliteit' van de ruis kan worden bijgesteld. Er is echter maar één kwaliteit ruis tegelijkertijd beschikbaar.

Wat het volume van het geluid betreft: dit kan per stem rechtstreeks worden ingesteld, of er kan een voorgeprogrammeerd volumeverloop worden aangeduid. Er kan maar één volumeverloop tegelijk werkzaam zijn; dit verloop geldt dan voor alle stemmen waarvan het volume niet rechtstreeks is ingesteld.
De lengte van een geluid (toon of ruis) kan niet van tevoren worden ingesteld, dat moet worden gedaan door na de gewenste tijdsduur het volume van het geluid op 0 te zetten.

We zullen deze mogelijkheden in de volgende paragrafen bespreken.

4.2 De mogelijkheden van de PSG

De PSG-chip kent 16 interne registers. 14 daarvan worden er gebruikt voor het besturen van geluid; de andere twee voor het uitlezen van de joysticks. De registers zijn opgesomd in par. 4.2. In het hierna volgende wordt ingegaan op de functie van deze registers. Dat gebeurt aan de hand van de theorie van de vorige paragraaf.

4.2.1 Instelling van de toonhoogte

Voor het instellen van de toonhoogte zijn voor elke stem twee registers gereserveerd. Dit zijn:

Stem A: Registers 0-1
Stem B: Registers 2-3
Stem C: Registers 4-5

Wat er in feite wordt ingesteld, is een getal, recht evenredig met de golflengte van de te produceren toon. Dit getal kan worden ingesteld met een nauwkeurigheid van 12 bits. De 8 minst significante bits staan in het eerste register, de 4 meest significante bits vormen de laatste 4 bits van het tweede register; zie afb. 4.2.

reg. nr.	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	bits 7-0 toonhoogte stem A							
1					bits 11-8 toonhoogte stem A			
2	bits 7-0 toonhoogte stem B							
3					bits 11-8 toonhoogte stem B			
4	bits 7-0 toonhoogte stem C							
5					bits 11-8 toonhoogte stem C			
6					kwaliteit van de ruis			
7	poort B in/uit	poort A in/uit	stem C ruis	stem B ruis	stem A ruis	stem C toon	stem B toon	stem A toon
8				ver-loop	volume-instelling stem A			
9				ver-loop	volume-instelling stem B			
10				ver-loop	volume-instelling stem C			
11	snelheid van bits 7-0 volumeverloop							
12	snelheid van bits 15-8 volumeverloop							
13					CONT	ATT	ALT	HOLD
14	PSG poort A							
15	PSG poort B							

Afb. 4.2. De PSG-registers

Als de gewenste frequentie voor een stem bekend is, kan het in te voeren getal op de volgende manier worden bepaald:

getal=KF/(16*frequentie)

Hierin is KF de klokfrequentie van het systeem, dat wil zeggen KF=3.579.545.

Als de exacte frequentie niet belangrijk is, kan men werken met de vuistregel dat een hoger getal resulteert in een lagere toon.
Let wel, het instellen van een frequentie voor een van de stemmen resulteert nog niet in een geluid! Daarvoor moet eerst een volume 0 ingevoerd worden.

4.2.2 De kwaliteit van de ruis

Hoewel ruis, zoals gezegd, bestaat uit een heleboel willekeurige tonen van een hoge frequentie, kan het resulterende geluid enigszins bijgesteld worden door de gemiddelde frequentie van deze willekeurige tonen in te stellen. Dit noemen we de kwaliteit van de ruis.
Het getal waarmee de kwaliteit kan worden ingesteld, heeft een nauwkeurigheid van vijf bits. Dit zijn de vijf laatste bits van register 6. Het in te stellen getal kan worden afgeleid van de gewenste (gemiddelde) frequentie met behulp van dezelfde formule als in par. 4.2.1, met het verschil dat de instelling nu een veel kleiner bereik heeft.

4.2.3 De toewijzing van de stemmen

Elk van de drie stemmen kan naar keuze alleen toon, alleen ruis, beide, of geen van beide weergeven. Dit wordt ingesteld in register 7. Tevens kan via dit register het gebruik van de beide I/O-poorten die de PSG rijk is, worden ingesteld. In de MSX is deze instelling echter vast en onveranderbaar, zodat we ons daar niet om hoeven te bekommeren.

De indeling van register 7 is als volgt:

Bit 7	poort B invoer/uitvoer	1 = uitvoer
Bit 6	poort A invoer/uitvoer	1 = uitvoer
Bit 5	stem C ruis ja/nee	1 = nee
Bit 4	stem B ruis ja/nee	1 = nee
Bit 3	stem A ruis ja/nee	1 = nee
Bit 2	stem C toon ja/nee	1 = nee
Bit 1	stem B toon ja/nee	1 = nee
Bit 0	stem A toon ja/nee	1 = nee

4.2.4 Instelling van het volume

Per stem kan een constant volume worden opgegeven, of kan worden verwezen naar het ingestelde volumeverloop.
Voor elke stem is een register gereserveerd waarin de volumegegevens kunnen worden gezet. Deze registers zijn:

Stem A: Register 8
Stem B: Register 9
Stem C: Register 10

De indeling van elk van deze registers is:

Bit 0-2	ongebruikt	
Bit 3	gebruik verloop ja/nee	1 = ja
Bit 4-7	volume-instelling	bereik: 0–15

Zoals uit het bovenstaande valt op te maken, kan een constant volume ingesteld worden met een nauwkeurigheid van vier bits. Het getal dat ingevoerd wordt is recht evenredig met de amplitude van het resulterende geluid, dus hoe groter het getal, hoe harder het geluid. Bij instelling 0 is de amplitude 0, en is er geen geluid hoorbaar.

Als het gebruik van het volumeverloop is ingesteld (bit 3=1), heeft de instelling van het constante volume (bits 4-7) geen effect meer.
Zodra voor een stem een constant volume 0 is ingesteld, begint het geluid van die stem, en houdt pas weer op als het volume terug is gezet op 0. Als echter voor een stem het gebruik van het volumeverloop is gespecificeerd, dan is er alleen geluid te horen op momenten die het verloop aangeeft (zie de volgende paragraaf).

4.2.5 Instelling van het volumeverloop

De PSG kent acht voorgeprogrammeerde volumeverlopen. Het gewenste verloop kan worden ingesteld via register 13.
De snelheid, of frequentie, van het volumeverloop kan in registers 11 en 12 worden vastgelegd.

Register 13 is als volgt ingedeeld:

Bit 0-3	ongebruikt	
Bit 4	CONT ja/nee	1 = ja
Bit 5	ATT ja/nee	1 = nee
Bit 6	ALT ja/nee	1 = ja
Bit 7	HOLD ja/nee	1 = ja

De betekenis van de hierboven gebruikte afkortingen is:

CONT (van CONTinue=doorgaan): het volumeverloop blijft doorgaan, dat wil zeggen dat het patroon van het volumeverloop steeds wordt herhaald. Het alternatief is, dat het verloop ophoudt nadat het patroon één keer wordt doorlopen.

ATT (van ATTenuate=verzwakken): het volume begint op volle sterkte, en vermindert meteen in sterkte. Het alternatief is, dat het volume begint met 0, en steeds sterker wordt.

ALT (van ALTernate=afwisselen): het volume wordt afwisselend sterker

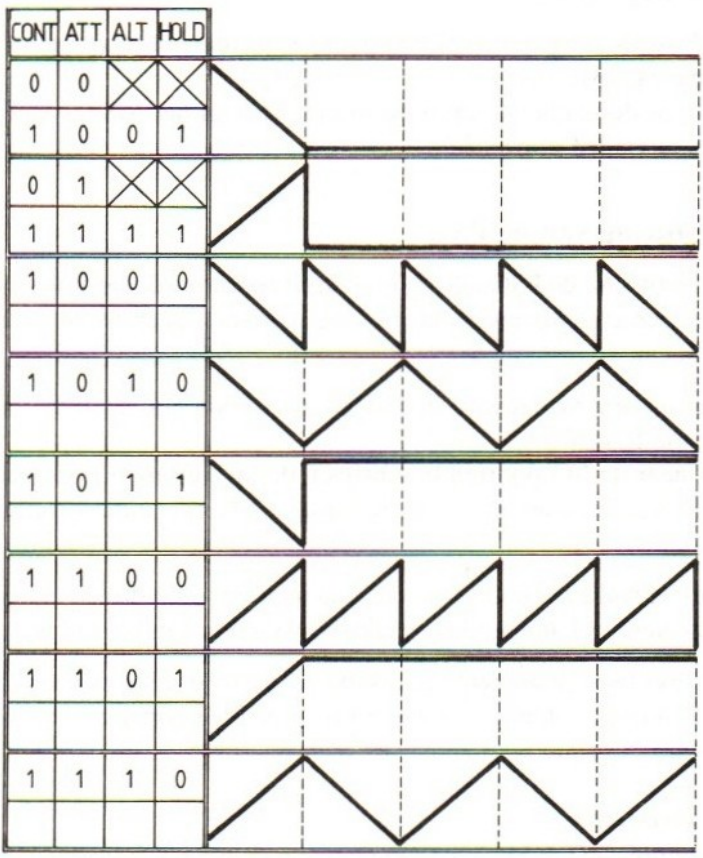
en zwakker. Het alternatief is, dat nadat het volume op z'n sterkst (zwakst) is gekomen, direct weer teruggesprongen wordt naar het zwakste (sterkste) volume.

HOLD (van HOLD=vasthouden): na één patroon te hebben doorlopen, blijft het volume constant op een niveau dat door de ATT- en ALT-bits bepaald wordt.

Elk volumeverloop is samengesteld uit onderdelen van een vaste lengte. In één zo'n onderdeel kan het volume van 0 naar maximum gaan, van maximum naar 0, of constant blijven. De manier waarop een verloop is samengesteld uit die onderdelen, wordt bepaald door de besturende bits waarvan de betekenis hiervoor is uitgelegd.

De volumeverlopen die kunnen worden bereikt, zijn vastgelegd in afbeelding 4.3.

De lengte van één onderdeel van een volumeverloop kan worden ingesteld door een getal in registers 11 en 12 te schrijven. Dit getal heeft een nauwkeurigheid van 16 bits. De meest significante acht bits komen in register 12, de 8 minst significante bits in register 11.



Afb. 4.3. Voorgeprogrammeerde volumeverlopen

Het in te stellen getal is recht evenredig met de duur van een onderdeel van een volumeverloop. Als de duur gegeven is (in seconden) dan is het getal daar met behulp van de volgende formule uit af te leiden:

getal=duur*KF/256

Hierbij is KF weer de klokfrequentie: KF=3.579.545 Hz.
Zodra naar register 13 een getal wordt geschreven, begint het bijbehorende volumeverloop bij het begin, en met de snelheid die bepaald wordt door registers 11 en 12.

4.2.6 Poortregisters

De PSG kent twee I/O-poorten die door de processor gelezen en beschreven kunnen worden. In het MSX-systeem worden deze poorten gebruikt om de joysticks uit te lezen.
Voor elk van deze beide poorten is een register gereserveerd dat op elk moment de waarde weergeeft die de poort heeft. Dit zijn de registers 14 en 15:

Poort A: Register 14
Poort B: Register 15

In register 7 wordt aangegeven of een poort als invoer- of uitvoerpoort wordt gebruikt (zie par. 4.2.3).
Details omtrent de manier waarop de I/O-poorten in het MSX-systeem gebruikt worden, zijn te vinden in par. 5.7.

4.3 De besturing van de PSG

Tot nu toe hebben we de PSG alleen op zichzelf beschouwd, alsof de chip geen deel uitmaakt van een computersysteem. In deze paragraaf gaan we in op het verband tussen de PSG en de rest van het MSX-systeem.

De PSG kan op drie niveaus worden aangestuurd: via poort-adressering, via OS-routines, en via BASIC.
Door direct naar de I/O-poorten te schrijven die de verbinding vormen tussen de PSG en de processor, moet men zelf zorg dragen voor de administratie en verwerking van geluiden.

Door gebruik te maken van OS-routines kan men op een indirecte manier de PSG-registers bereiken, of kan men het geluids-subsysteem van het OS gebruiken.

In BASIC zijn er twee 'statements' gereserveerd voor het gebruik van geluid: PLAY en SOUND. Het ene statement adresseert weer de PSG-registers, het andere maakt gebruik van het OS-geluidssysteem.

4.3.1 Poortadressering

In de adresruimte van het MSX-systeem bevindt de PSG zich tussen de poortadressen. Met andere woorden, de I/O-poorten vormen de verbinding tussen de PSG en de rest van de computer (de processor).

De toewijzing van poortadressen aan de PSG is als volgt:

Adres	*Modus*	*Toepassing*
A0	S	Invoer registernummer
A1	S	Invoer nieuwe waarde register
A2	L	Registerinhoud

'Modus' kan twee waarden hebben: L (lezen) of S (schrijven).

Zoals uit deze tabel blijkt, zijn er voor de PSG maar drie poortadressen gereserveerd. Met deze drie adressen kunnen toch alle zestien registers van de PSG gelezen en beschreven worden, en wel op de volgende manier:

1. Schrijf naar adres A0 het nummer van het register dat gebruikt gaat worden.
2. Gedurende een korte periode na het uitvoeren van stap 1 kan het register gelezen en beschreven worden.
3. De huidige inhoud van het register is te vinden op adres A2 gedurende de periode genoemd in 2.
4. Aan het register kan een nieuwe waarde worden toegekend door deze nieuwe waarde naar adres A1 te schrijven binnen de periode genoemd in 2.

Zoals hierboven aangegeven, is het register slechts gedurende een korte periode na het schrijven van het registernummer beschikbaar. De oorzaak hiervan is dat het OS als onderdeel van het interruptsysteem ook de waarde van PSG-register 15 opvraagt; dit om de vuurknoppen van de joysticks uit te lezen. Als bij het lezen van de registers interrupts uitgeschakeld worden (DI), is dit verholpen.
Na afloop van deze periode en overigens altijd zolang er geen registernummer naar A0 is geschreven, is de inhoud van A2 ongedefinieerd en heeft het schrijven van een nieuwe waarde naar A1 geen effect.

N.B.: In het huidige MSX-systeem worden de poortadressen A3-A7 niet gebruikt. Als gevolg daarvan hebben A0, A2, A4 en A6 precies hetzelfde effect bij het lezen en schrijven van deze adressen. Hetzelfde geldt voor A1, A3, A5 en A7. Echter, in de officiële MSX-standaard staan de adressen vermeld zoals hiervoor. Dit betekent dat deze adressen in ieder geval hun functie zullen blijven behouden, zelfs als A3-A7 voor een ander doel gebruikt gaan worden. Uit het oogpunt van overdraagbaarheid is het dus raadzaam om bij het schrijven van programma's alleen gebruik te maken van 'officiële' adressen.

Overigens is het aan te bevelen om zelfs helemaal geen gebruik te maken van de poortadressen, maar in plaats daarvan OS-ingangen te gebruiken voor het lezen en schrijven van de PSG-registers.

4.3.2 OS-routines

De OS-routines die de PSG ondersteunen, vallen in twee categorieën: routines die direct de PSG aansturen en routines die te maken hebben met het geluids-subsysteem van het OS.

In de eerste categorie vallen drie routines: het initialiseren van de PSG, het lezen van een PSG-register, en het geven van een nieuwe waarde aan een PSG-register.
De tweede categorie omvat alle routines van het 'queue'-systeem, de initialisatie van het geluidssysteem en het opstarten van het geluidssysteem. Voor gegevens over het queue-systeem en het geluidssysteem zie par. 6.2.3-6.2.4.
Het verdient de voorkeur om waar mogelijk gebruik te maken van het geluidssysteem, en te vermijden om direct de registers te adresseren. Ten eerste kan het direct adresseren van de PSG-registers het geluidssysteem in de war schoppen, aangezien dit systeem bepaalde informatie krijgt over de toestand van de registers die niet meer opgaat als de registers elders veranderd worden. Ten tweede is met het geluidssysteem een mogelijkheid geboden om geluid te produceren als achtergrondtaak, dus zonder verder omkijken naar de precieze werking van de PSG.
Aan de andere kant maakt het geluidssysteem niet van alle mogelijkheden van de PSG gebruik. Met name het gebruik van ruis wordt niet door het geluidssysteem ondersteund.

Hier volgt een lijst met alle OS-ingangen die op de PSG betrekking hebben.

003B	INITIO	initialisatie van de PSG (e.a)
0093	WRTPSG	geeft PSG-register nieuwe waarde
0096	RDPSG	leest waarde van PSG-register
0090	GICINI	initialisatie van het geluidssysteem
0099	STRTMS	start geluidssysteem op
00C0	BEEP	kort piepje en initialisatie
00F6	LFTQ	aantal bytes in een (geluids)queue
00F9	PUTQ	zet waarde in (geluids)queue

Voor details omtrent de routines zie par. 6.3.

4.3.3 BASIC-aansturing

Ook vanuit BASIC is het mogelijk om de PSG te gebruiken. Er zijn in BASIC drie sleutelwoorden die met geluid te maken hebben. We zullen deze sleutelwoorden de revue laten passeren.

SOUND: het SOUND-statement kan worden gebruikt om direct de PSG-registers te adresseren. Voor het gebruik hiervan gelden dezelfde bezwaren als voor het gebruik van de OS-routines die hetzelfde doen: SOUND gooit het geluidssysteem in de war en is over het algemeen niet nodig. Sommige eigenschappen van de PSG, zoals het gebruik van ruis, zijn echter zonder SOUND niet vanuit BASIC toe te passen.

PLAY: dit statement is de BASIC-versie van het geluidssysteem van het OS. PLAY biedt de mogelijkheid om via geluidscommando's in stringparameters alle stemmen tegelijk en gesynchroniseerd te laten spelen, en wel als achtergrondtaak, dat wil zeggen tegelijk met het uitvoeren van andere BASIC-statements.

BEEP: het BEEP-statement zorgt voor een kort piepje, dat hetzelfde is als het piepje dat klinkt bij een foutmelding (als dat piepje tenminste niet uitgeschakeld is). Tegelijk wordt het geluidssysteem opnieuw geïnitialiseerd. Als door een SOUND-statement het geluidssysteem onbruikbaar is geworden, kan dat vanuit BASIC met BEEP opgelost worden. BEEP is een direct equivalent met de OS-routine BEEP (00C0) zie par. 4.3.2.

Het is ook nog mogelijk om direct de poortadressen van de PSG te adresseren met behulp van de BASIC-sleutelwoorden INP en OUT. Dit is echter ten sterkste af te raden, omdat het geen enkel voordeel biedt ten opzichte van het gebruik van SOUND en het bovendien veel ondoorzichtiger is.

5 Overige apparatuur

In dit hoofdstuk wordt de rest van de apparatuur van de MSX-computer besproken. Hieronder vallen de PPI-chip, het toetsenbord en randapparatuur.

De MSX-standaard beschrijft een groot aantal aansluitingen voor randapparatuur. Alle apparaten die volgens deze specificaties werken, zullen in principe met elke MSX-computer kunnen werken. In feite maken deze aansluitingen net zo goed deel uit van de MSX-computer als het video- en audio-gedeelte, aangezien de aansturing ervan is vastgelegd.

5.1 De PPI

PPI staat voor Programmable Peripheral Interface (Programmeerbare Randapparatuur-Verbinding). Deze chip verzorgt een flexibele verbinding van de processor met andere onderdelen van het computersysteem.
In de MSX-computer wordt deze chip gebruikt voor het besturen van het gleufselectieregister, voor het uitlezen van het toetsenbord, voor het communiceren met de cassetterecorder, voor het aan- en uitzetten van het CAPS-lampje en voor de 1-bits geluidsuitgang van de PPI die de toetsenbordklik verzorgt.

De PPI heeft vier interne registers. Deze registers worden aangeduid met het A-, B-, C- en commandoregister. In het MSX-systeem zijn de PPI-registers op de volgende manier ingedeeld:

PPI-register A: het gleufselectieregister. Bij elke pagina van het adresbereik horen twee bits van dit register. Het getal in die twee bits bevat het nummer van de primaire gleuf die op dit moment voor die pagina geselecteerd is.

Bit 7-6	prim. gleuf voor pagina 0 bereik: 0-3
Bit 5-4	prim. gleuf voor pagina 1 bereik: 0-3
Bit 3-2	prim. gleuf voor pagina 2 bereik: 0-3
Bit 1-0	prim. gleuf voor pagina 3 bereik: 0-3

Register A kan zowel worden gelezen als beschreven.

PPI-register B: de bits van dit register corresponderen met een kolom van de toetsenbordmatrix. Elk van de bits geeft de waarde van het kolomelement van de rij die op dit moment via register C ingesteld is. Deze waarde is laag als de toets ingedrukt is, en anders hoog.

Meer informatie over de toetsenbordmatrix en de werking van het toetsenbord staat in par. 5.2.

Register B kan alleen worden uitgelezen. Het schrijven van een nieuwe waarde naar dit register heeft geen effect.

PPI-register C: via dit register wordt de cassetterecorder, het CAPS-lampje en de 1-bits geluidsuitgang bestuurd en wordt een nieuw rijnummer van de toetsenbordmatrix opgegeven.

Bit 7	geluidsuitgang	1 = aan
Bit 6	CAPS-lampje	1 = uit
Bit 5	cassette-schrijfsignaal	1 = hoog
Bit 4	cassette-motorbesturing	1 = uit
Bit 3-0	rijnummer toetsenbordmatrix	bereik: 0-10

De 1-bits geluidsuitgang is onafhankelijk van de PSG-geluids-chip. Deze uitgang wordt gebruikt voor de klik die klinkt bij een toetsaanslag (als die klik ingeschakeld is).
Meer informatie over het cassettesysteem is te vinden in par. 5.4. Meer over het toetsenbord en de -matrix staat in par. 5.2.
PPI-register C kan zowel worden gelezen als beschreven.

Commandoregister: naast de dataregisters A, B en C kent de PPI een commandoregister. Dit register kan alleen worden beschreven. Door een waarde naar het commandoregister te schrijven, kunnen bits van de andere registers afzonderlijk hoog of laag worden gezet.

In het MSX-systeem is er maar één PPI-register dat op die manier veranderd kan worden, namelijk register C. De waarde die naar het commandoregister geschreven moet worden om register C te veranderen ziet er als volgt uit:

Bit 7	moet hoog zijn	= 1
Bit 6-4	ongebruikt	
Bit 3-1	bitnummer binnen register C	bereik: 7-0
Bit 0	zet hoog of zet laag?	1 = zet hoog

De volgende poortadressen zijn gereserveerd voor de PPI:

Adres	*Modus*	*Toepassing*
A8	L/S	PPI-register A
A9	L	PPI-register B
AA	L/S	PPI-register C
AB	S	commandoregister

In het algemeen gesproken, is het niet verstandig om zelf de PPI-registers te lezen of te beschrijven. Ten eerste kan het beter worden overgelaten aan de routines die horen bij de apparaten die via de PPI aangesloten zijn om hun eigen registers aan te

spreken, aangezien het gebruik van deze apparaten heel complex kan zijn. Ten tweede kan het zelf veranderen van registerwaarden tot gevolg hebben dat de MSX-computer niet meer correct werkt, aangezien er op onbekende manier iets veranderd is in de toestand van de aangesloten apparatuur.

De volgende OS-routines schrijven, lezen of veranderen de waarde van een PPI-register:

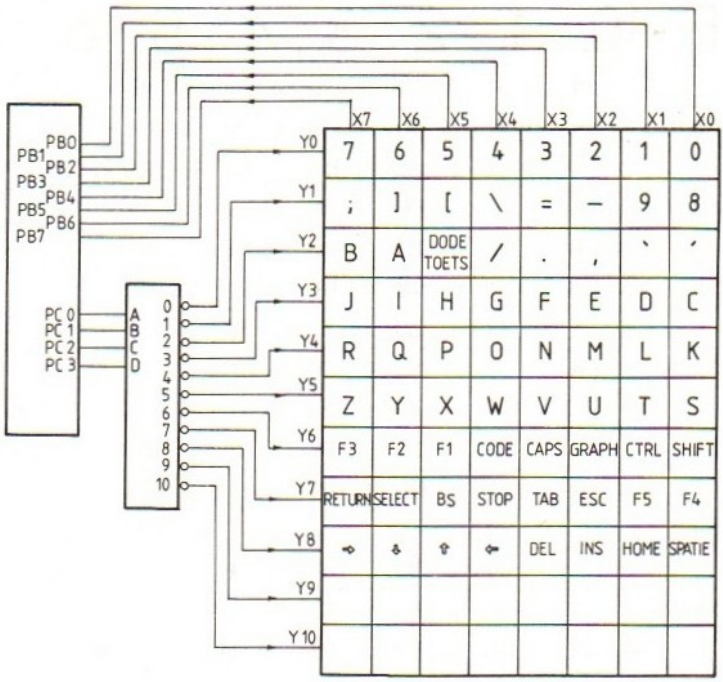
0000	INIT	zet registers op beginwaarde
000C	RDSLT	leest waarde uit andere gleuf
0014	WRSLT	schrijft waarde naar andere gleuf
001C	CALSLT	roept routine in andere gleuf aan
0024	ENASLT	schakelt gleuf permanent in
0030	CALLF	roept routine in andere gleuf aan
0038	KEYINT	leest toetsenbordmatrix
00E1	TAPION	start lezen van cassette
00E7	TAPIOF	beëindigt lezen van cassette
00EA	TAPOON	start schrijven naar cassette
00ED	TAPOUT	schrijft waarde naar cassette
00F0	TAPOOF	beëindigt schrijven naar cassette
00F3	STMOTR	zet cassettemotor aan/uit
0132	CHGCAP	zet CAPS-lampje aan/uit
0135	CHGSND	zet 1-bits geluidsuitgang aan/uit
0141	SNSMAT	leest rij van toetsenbordmatrix

5.2 Toetsenbord

Het toetsenbord kan beschouwd worden als een matrix van knooppunten. Elk van de knooppunten correspondeert met een toets van het toetsenbord. De waarde van een knooppunt geeft aan of de bijbehorende toets ingedrukt is of niet.

De matrix heeft de afmetingen 8 (horizontaal)×11 (verticaal). De onderste twee rijen worden in de MSX-standaard niet vastgelegd, en kunnen dus gebruikt worden voor extra toetsen.
In afbeelding 5.1 is de matrix afgebeeld en het verband tussen de elementen van de matrix en de toetsen van het toetsenbord vastgelegd. Tevens is een verbinding gelegd tussen de matrix en registers B en C van de PPI (zie par. 5.1).

Elk element van de matrix wordt gerepresenteerd door een bitwaarde. Het bit is hoog als de toets niet ingedrukt is, en laag als de toets wel ingedrukt is.
De waarde van een bit van een bepaalde toets wordt op een tamelijk ingewikkelde manier vastgesteld. In bits 3-0 van PPI-register C wordt het nummer van een rij van de matrix ingesteld; vervolgens corresponderen de bits van PPI-register B met de kolomelementen van die rij. Om alle ingedrukte toetsen vast te stellen, moeten achtereenvolgens de rijnummers 0-10 in register C worden ingesteld, en voor elke rij de waarde van register B ingelezen.



Afb. 5.1. Toetsenbordmatrix

Een probleem van deze organisatie is dat de matrix-structuur niet altijd correcte resultaten oplevert. Als bijvoorbeeld in die matrix toetsen N, V en W zijn ingedrukt, reageert ook toets O. De elementen van de matrix zijn niet zo onafhankelijk als het plaatje wil doen geloven.

Het toetsenbord kan worden uitgelezen door zelf naar register C te schrijven en de waarde van register B in te lezen. De poortadressen van deze registers zijn:

Adres	*Modus*	*Toepassing*
A9	L	PPI-register B
AA	S	PPI-register C

(Zie ook par. 5.1.)

Zelf de PPI-registers te gebruiken is niet aan te raden. In toekomstige versies van de MSX-computer kunnen de poortadressen of de betekenis van de bits van de registers veranderen. Voor het uitlezen van de toetsenbordmatrix is de OS-routine SNSMAT aanwezig, en aan te roepen via de OS ingang op adres 0141 (zie par. 6.3).

In de meeste gevallen is het niet nodig om zelf de matrix te doorzoeken. Het OS-interrupt-systeem (zie par. 6.2.2) leest de hele matrix uit en slaat de waarde op in geheugengebied NEWKEY (vanaf FBE5), met een kopie van de vorige waarde in OLDKEY (vanaf FBDA). Elk byte van een van deze gebieden correspondeert met een rij van de matrix.

INT			0	1	2	3	4	5	6	7
0	Normal		0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
		Shift	) 29	! 21	@ 40	# 23	$ 24	% 25	^ 5E	& 26
	Graph		○ 09	¼ AC	½ AB	¾ BA	η EF	‰ BD	◘ F4	√ FB
		Shift	◘ 0A		² FD	″ FC			F5	
	Code		δ EB	ƒ 9F	‡ D9	§ BF	¢ 9B	ÿ 98	α E0	β E1
		Shift	Δ D8	¡ AD	Pt 9E	π BE	£ 9C	Y 9D		
1	Normal		8 38	9 39	– 2D	= 3D	\ 5C	[5B	] 5D	; 3B
		Shift	* 2A	(28	— 5F	+ 2B		7C	{ 7B	} 7D	: 3A
	Graph		∞ EC	• 07	– 17	± F1	\ 1E	☺ 01	0D	♠ 06
		Shift		08	+ 1F	≡ F0	| 16	02	0E	♦ 04
	Code		E7	ç 87	ε EE	θ E9		φ ED	ω DA	ũ B7
		Shift	E2	Ç 80				ϕ E8	Ω EA	Ũ B6
2	Normal		' 27	` 60	, 2C	. 2E	/ 2F	`	a 61	b 62
		Shift	" 22	~ 7E	< 3C	> 3E	? 3F	´	A 41	B 42
	Graph		♣ 05	BB	≦ F3	≧ F2	/ 1D	`	▬ C4	11
		Shift	♥ 03	≈ F7	« AE	» AF	÷ F6	´	▌ FE	
	Code		ij B9	σ E5	å 86	a A6	o A7	^	ä 84	ù 97
		Shift	IJ B8	Σ E4	Å 8F		¿ A8	¨	Ä 8E	
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		◇ BC	C7	▼ CD	14	+ 15	13	▬ DC	▌ C6
		Shift	- FA	C1	▲ CE	■ D4	┼ 10	■ D6	▬ DF	▇ CA
	Code		ì 8D	ï 8B	î 8C	ö 94	ü 81	ä B1	í A1	æ 91
		Shift				Ö 99	Ü 9A	Ã B0		Æ 92
4	Normal		k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72
		Shift	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52
	Graph		▌ DD	▇ C8	♂ 0B	1B	▬ C2	▇ DB	CC	18
		Shift	▌ DE	| C9	♀ 0C	■ D3	▬ C3	D7	CB	A9
	Code		ĩ B3	o B5	μ E6	ñ A4	ó A2	ú A3	â 83	ô 93
		Shift	Ĩ B2	Õ B4		Ñ A5		E3		
5	Normal		s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A
		Shift	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A
	Graph		⋈ D2	12	▬ C0	1A	▶ CF	× 1C	19	☼ 0F
		Shift	⧗ D1		▇ C5	■ D5	◀ D0	• F9	AA	○ F8
	Code		ë 89	û 96	é 82	ò 95	ê 88	è 8A	á A0	à 85
		Shift			É 90					

Afb. 5.2. Verband tussen toetsaanslag en ASCII-teken

Het interrupt-systeem verzorgt tevens het interpreteren van de matrixwaarde. Deze interpretatie bestaat uit het omzetten van ingedrukte toetsen naar ingetypte ASCII-tekens. De ingetypte tekens worden in de toetsenbordbuffer gezet (zie par. 6.2.2.)
Het interrupt-systeem wordt 50 maal per seconde aangeroepen. Bij elke aanroep wordt er een teller verlaagd (namelijk SCNCNT, adres F3F6). Als de teller 0 heeft bereikt, wordt de waarde van NEWKEY gekopieerd naar OLDKEY, en wordt de huidige waarde van de toetsenbordmatrix bepaald en in NEWKEY gezet. Tevens wordt de teller teruggezet op 3.
Als de waarde van NEWKEY gelijk is aan OLDKEY is er niets veranderd aan de toetsenbordmatrix: er zijn geen nieuwe toetsen ingedrukt of oude losgelaten. In dat geval wordt de waarde van REPCNT (F3F7) verlaagd. Als deze teller de 0 bereikt heeft, wordt de waarde van NEWKEY geïnterpreteerd als nieuw ingedrukte toetsen; anders wordt de waarde van NEWKEY genegeerd. Dit heeft tot gevolg dat bij het vasthouden van toetsen een tijdje niets gebeurt, waarna de toetsen zich gaan herhalen.

Als NEWKEY wel verschilt ten opzichte van OLDKEY wordt REPCNT teruggezet op 13, en wordt NEWKEY beschouwd als de nieuwe toetsaanslag.
Het bepalen van de ASCII-tekens uit een nieuwe toetsaanslag gebeurt volgens de tabel van afbeelding 5.2. Hierin is voor elke combinatie van ingedrukte toetsen het resulterende ASCII-teken vermeld.

Het uitlezen van de toetsenbordbuffer gebeurt met de OS-routine CHGET (ingang 009F). De routine CHSNS (ingang 009C) bepaalt of er iets in de buffer staat. Met behulp van KILBUF (ingang 0156) wordt de buffer leeggemaakt. Zie ook par. 6.2.6.

OS-routines die betrekking hebben op het toetsenbord:

009C	CHSNS	test of er een teken in de buffer staat
009F	CHGET	leest een teken uit de toetsenbordbuffer
0141	SNSMAT	leest een rij van de toetsenbordmatrix
0156	KILBUF	maakt de toetsenbordbuffer leeg

Geheugenplaatsen die gebruikt worden voor het toetsenbord:

F3F6	SCNCNT	teller van interrupts
F3F7	REPCNT	teller van gelijke toetsaanslagen
FBDA	OLDKEY	vorige waarde van toetsenbordmatrix
FBE5	NEWKEY	nieuwe waarde van toetsenbordmatrix

5.3 Cartridges

Via de cartridge-gleuf kan programmatuur en/of apparatuur aan de MSX-computer worden gekoppeld. Elke cartridge-gleuf komt overeen met een primaire of secundaire gleuf in het gleufsysteem van de MSX-computer (zie hoofdstuk 2). Een cartridge kan dan ook tot 64 kbyte adresseerbaar geheugen bevatten – hoewel niet al dat geheugen even bruikbaar zal zijn.

PIN.NR.	NAAM	I/O	PIN.NR.	NAAM	I/O
1	$\overline{CS1}$	0	2	$\overline{CS2}$	0
3	CS12	0	4	$\overline{SLTSL}$	0
5	ONGEBRUIKT	-	6	$\overline{RFSH}$	0
7	$\overline{WAIT}$	I	8	$\overline{INT}$	I
9	$\overline{M1}$	0	10	$\overline{BUSDIR}$	I
11	$\overline{IORQ}$	0	12	$\overline{MERQ}$	0
13	$\overline{WR}$	0	14	$\overline{RD}$	0
15	$\overline{RESET}$	0	16	ONGEBRUIKT	-
17	A9	0	18	A15	0
19	A11	0	20	A10	0
21	A7	0	22	A6	0
23	A12	0	24	A8	0
25	A14	0	26	A13	0
27	A1	0	28	A0	0
29	A3	0	30	A2	0
31	A5	0	32	A4	0
33	D1	I/O	34	D0	I/O
35	D3	I/O	36	D2	I/O
37	D5	I/O	38	D4	I/O
39	D7	I/O	40	D6	I/O
41	GND	-	42	CLOCK	0
43	GND	-	44	SW1	-
45	+5V	-	46	SW2	-
47	+5V	-	48	+12V	-
49	SOUNDIN	I	50	-12V	-

Afb. 5.3. Cartridge-aansluiting

De cartridge-aansluiting is een 2.54 PACE 50-pin stekker. De indeling en betekenis van de verschillende pins zijn weergegeven in afbeelding 5.3.

De inhoud van een cartridge kan zeer uiteenlopen. De eenvoudigste mogelijkheid is een 16 kbyte ROM-pagina waarin een spelletjes- of ander programma staat, of een RAM-pagina die dient als geheugenuitbreiding. Via een cartridge kan echter ook een diskettestation of een RS232-verbinding worden verzorgd. Dit zijn cartridges met een zgn. apparaatuitbreiding (zie par. 6.5.2).

De cartridge-aansluiting voorziet, behalve in een adres- en databus, in een aantal andere verbindingen die een zeer intensieve communicatie tussen de cartridge en de processor mogelijk maken. Zo is het mogelijk interrupts vanuit de cartridge te geven (INT), om de cartridge te synchroniseren met de processor door een kloksignaal, etc.

Van alle pin-aansluitingen volgt hier een korte beschrijving. (De aanduiding 'negatieve logica' geeft aan dat het signaal 'aan' wordt verondersteld te zijn als het laag is, en 'uit' als het hoog is. Gewoonlijk is dit omgekeerd).

CS1, CS2, CS12 (pins 1-3, negatieve logica): worden gebruikt om van het geheugen dat eventueel in de cartridge geïnstalleerd is, een bepaald adresbereik te selecteren. De verschillende signalen selecteren de volgende adresbereiken:

signaal	*adresbereik*
CS1	4000-7FFF
CS2	8000-BFFF
CS12	4000-BFFF

SLTSL (pin 4, negatieve logica): geeft aan dat de cartridge geselecteerd is voor een of andere actie. Dit signaal moet aan (laag) zijn, wil de adres- of databus van de computer vrij zijn voor gebruik door de cartridge.

RFSH (pin 6, negatieve logica): wordt gebruikt om eventuele dynamische RAM-geheugens in de cartridge te verversen.

WAIT (pin 7, negatieve logica): staat in verbinding met de WAIT-pin van de processor. Door dit signaal kan de processor in een wachttoestand worden gebracht, waarin hij niets doet tot het signaal weer uit (hoog) is.

INT (pin 8, negatieve logica): staat in verbinding met de INT van de processor. Dit signaal kan worden gebruikt om vanuit de cartridge een interrupt te genereren. De RS232 maakt hier bijvoorbeeld gebruik van.

M1 (pin 9, negatieve logica): gaat aan (wordt laag) als de processor bezig is aan zijn M1 (eerste machine-) cyclus.

BUSDIR (pin 10, negatieve logica): geeft aan of de databus (D7-D0, zie hierna) op dit moment gebruikt wordt voor in- of uitvoer. Als het signaal aan is, wordt de databus gebruikt voor uitvoer (vanuit de computer), anders voor invoer.

IORQ (pin 11, negatieve logica): dit signaal staat in verbinding met de processor, en geeft aan op welk moment de adresbus wordt gebruikt voor poortadressering. Wanneer dat het geval is, vormen A7-A0 (zie hierna) het adres waarvan moet worden gelezen of waar naar toe moet worden geschreven.

MERQ (pin 12, negatieve logica): dit signaal staat in verbinding met de processor, en geeft aan op welk moment de adresbus gebruikt wordt voor gewone geheugenadressering. Wanneer dat het geval is, vormen A15-A0 (zie hierna) het geheugenadres waarvan moet worden gelezen of waar naar toe moet worden geschreven.

WR (pin 13, negatieve logica): dit signaal staat in verbinding met de processor, en geeft aan of er data geschreven wordt.

RD (pin 14, negatieve logica): dit signaal staat in verbinding met de processor, en geeft aan of er data gelezen wordt.

RESET (pin 15, negatieve logica): staat in verbinding met de processor. Dit signaal wordt laag als de computer teruggezet wordt in begintoestand (zoals bij het aanzetten). De cartridge kan op dit signaal reageren door hetzelfde te doen.

A15-A0 (pins 32-17): de adresbus. A15-A0 vormen samen een geheugenadres als MERQ aan is (zie hiervoor, pin 12); A7-A0 vormen samen een poortadres als IORQ aan is (pin 11, zie hiervoor).

D7-D0 (pin 40-33): de databus. Wordt gebruikt om 1 byte gegevens van of naar de cartridge te transporteren. Als BUSDIR aan (laag) is, is de transportrichting naar de computer toe, en anders van de computer vandaan.

CLOC kbyte (pin 42): staat in verbinding met het kloksignaal van de computer (frequentie 3.58 MHz). Kan worden gebruikt om de cartridge te synchroniseren met de computer.

SW1, SW2 (pins 44, 46): detectie van insteken of verwijderen van de cartridge.

SOUNDIN (pin 49): wordt gebruikt om geluidssignalen vanuit de cartridge uit te voeren.

De verbinding en communicatie met een cartridge vindt geheel plaats op hardwareniveau; dat wil zeggen dat het OS geen verschil ziet tussen een gleuf waarop een cartridge aangesloten is en een gleuf met gewoon geheugen. Het in cartridges geïnstalleerde geheugen wordt door het OS dan ook op dezelfde manier behandeld als het geheugen in een gewone pagina (zie hoofdstuk 2). De andere verbindingssignalen gedragen zich ook op dezelfde manier als overeenkomstige signalen uit de computer zelf.

De manier waarop de computer de gleuven onderzoekt op geheugenpagina's en de manier waarop die pagina's behandeld worden, zijn te vinden in par. 6.2.1. De manier waarop interrupts worden verwerkt, staat in par. 6.2.2.

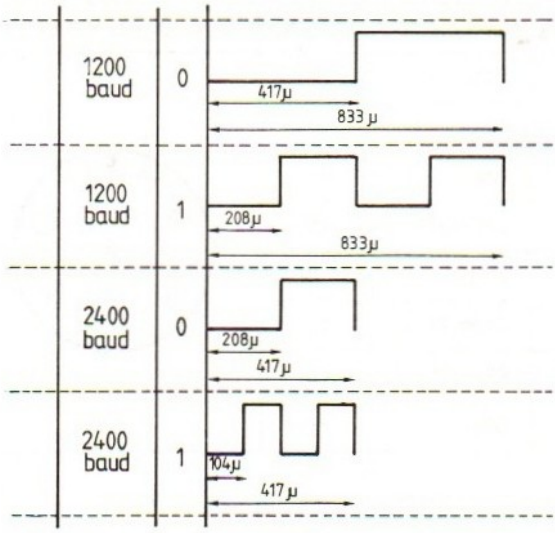
5.4 Cassette

De cassetterecorder kan worden gebruikt om op magnetische wijze informatie op een cassette op te slaan. De cassette kan dan als opslagmedium gebruikt worden. De informatie staat op cassette in de vorm van blokpulsen. De codering van een bit gebeurt met een methode die wordt aangeduid met Frequency Shift Keying (FSK). Dit houdt in dat de frequentie van de blokpuls (het aantal keren per seconde dat de puls hoog en weer laag wordt) aangeeft of er een 1 dan wel een 0 op de cassette staat.

De volgende tabel geeft een overzicht van de verschillende frequenties en de duur van de signalen voor één bit, gerelateerd aan de baudsnelheid waarmee het bit op cassette is gezet.

snelheid	*bit*	*frequentie*	*tijdsduur*
1200	0	1200 Hz	833 s
1200	1	2400 Hz	833 s
2400	0	2400 Hz	417 s
2400	1	4800 Hz	417 s

In afbeelding 5.4 zijn de mogelijke blokpulsen weergegeven.



Afb. 5.4. Blokpulsen op cassette

Een byte wordt op cassette gezet als een serie bits. Eerst komt een startbit, gelijk aan 0, dan volgen de acht bits die de waarde van het byte vertegenwoordigen, in volgorde van bit 0 (minst significante bit) tot bit 7 (meest significante bit); ten slotte twee stopbits, gelijk aan 1.

Behalve een byte als informatie-eenheid, kan er een kopblok (Engels: header) op cassette komen te staan. Zo'n kopblok is te horen (wanneer men de cassette direct beluistert) als continue toon van één toonhoogte; dit in tegenstelling tot de codering van een byte, dat een geraas oplevert.

Er zijn twee soorten kopblok: een kort en een lang. Een lang kopblok duurt vier maal zo lang als een kort blok.

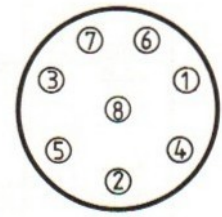
De volgende tabel legt een verband tussen baudsnelheid en kopbloklengte:

snelheid	*soort*	*frequentie*	*aant. pulsen*
1200	*kort*	*2400 Hz*	*4000*
1200	*lang*	*2400 Hz*	*16000*
2400	*kort*	*4800 Hz*	*8000*
2400	*lang*	*4800 Hz*	*32000*

Voor de verbinding van de cassetterecorder met de computer is een speciale aansluiting op de computer aanwezig. Dit is een vrouwelijke 5-pins DIN-verbinding. In afbeelding 5.5 is de indeling en betekenis van de pins aangegeven.

Vanuit de computer zijn de signalen van de verschillende pins op uiteenlopende wijzen bereikbaar. Het uitvoer- en motorbesturingssignaal zijn aangesloten via register C van de PPI, en het invoersignaal via register 14 van de PSG.

PIN.NR.	NAAM v. SIGNAAL	RICHTING	AANSLUITING VAN DE PINS
1	GND	—	
2	GND	—	
3	GND	—	
4	CMTOUT	UIT	
5	CMTIN	IN	
6	REMOTE +	UIT	
7	REMOTE -	UIT	
8	GND	—	



Afb. 5.5. Cassette-aansluiting

De precieze aansluiting van de signalen is als volgt:

CASW	PPI-register C	bit 5
CASON	PPI-register C	bit 4
CASR	PSG-register 14	bit 7

PPI-register C kan worden geadresseerd via poortadres AA of via het PPI-commandoregister, poortadres AB (zie par. 5.1). PSG-register 14 kan worden geadresseerd door eerst naar poortadres A0 het getal 14 (het nummer van het register) te schrijven, en vervolgens van poortadres A2 de waarde in te lezen; (zie par. 4.3.1).

Het is niet aan te raden direct het PPI- of PSG-register te gebruiken. De cassette is een nogal onbetrouwbaar medium, en om een enigszins goed functionerend cassettesysteem te implementeren is een grote hoeveelheid rekenwerk en timing vereist die gemakkelijk verstoord zou kunnen worden wanneer iemand buiten de OS-routines om de cassetterecorder probeert te besturen.

De manier waarop de cassetterecorder door het OS is ondersteund, wordt uitgebreid behandeld in par. 6.2.5.

5.5 Diskette

Een diskette (ook wel floppy of schijf genoemd) is een opslagmedium dat vele voordelen biedt boven de cassette: grotere betrouwbaarheid, grotere opzoeksnelheid, grotere overdrachtssnelheid, mogelijkheid van random access. Nadelen zijn: hoge kosten van het diskette-station (diskdrive), en hogere prijs per byte opslag van het medium.

Algemeen

Een diskette is een platte ronde schijf. Deze gegevens worden op de schijf bewaard met behulp van magnetische polarisatie. De ordening van de gegevens op de schijf is als volgt:

- de schijf wordt opgedeeld in een serie concentrische cirkels met steeds kleinere straal. Deze cirkels heten de sporen (Engels: tracks) van de diskette. Een diskette heeft over het algemeen 40 of 80 sporen. Dit is afhankelijk van het diskettestation.
- elk van de sporen is opgedeeld in sectoren, in feite stukken cirkelboog.
- op elke sector staan gegevens van een vaste hoeveelheid bytes (vaak 128, 256 of 512). Het aantal bytes op een sector wordt bepaald door de dichtheid (Engels: density) waarmee de sector is beschreven.

Voordat een diskette kan worden gebruikt, moet hij worden geformatteerd. Bij dit formatteren worden er magnetisch merktekens op de diskette gezet die aangeven waar sectoren beginnen en eindigen. Hierdoor kunnen sectoren op een diskette teruggevonden worden, en krijgt elke sector een vast nummer.
Het nummer van een sector wordt berekend uit het nummer van het spoor en de plaats van een sector op het spoor. De sporen zijn genummerd van 0–39 of van 0–79. Het sectornummer is gelijk aan het spoornummer vermenigvuldigd met het aantal sectoren per spoor, plus het volgnummer van die sector (afgeleid uit de plaats op het spoor).

MSX specifiek

In de MSX-standaard zijn details over de indeling van de diskette niet vastgelegd. Op het meest gebruikte diskette, 3.5 inch, staan 80 sporen van 9 sectoren die 512 bytes bevatten. Er zijn dus totaal 720 sectoren, genummerd van 0–719, en op een diskette past 360 kbyte aan gegevens.

Alle acties op een diskette werken met sectoren tegelijk. Lezen en schrijven gebeurt per hele sector. Wanneer er één byte gelezen moet worden dat midden op een sector staat, wordt eerst de hele sector ingelezen, en wordt daaruit vervolgens het gewenste byte gekozen.

Aansluiting

Een diskettestation moet op het MSX-systeem worden aangesloten met behulp van een cartridge. De eigenschappen van deze aansluiting zijn besproken in par. 5.3.

Voor handig gebruik van het diskettestation zijn vele speciale voorzieningen getroffen. Zo zijn er een groot aantal BASIC-sleutelwoorden gereserveerd voor gebruik van de diskette (zie par. 7.3), en zijn er vele RAM-haken aanwezig die enkel en alleen voor het diskettestation disk drive toegevoegd zijn (zie par. 6.4.2). Ook zijn er een aantal poortadressen (D0-D7) gereserveerd voor het diskettestation. Ten slotte zijn er twee OS ingangen voor routines die alleen op diskette van toepassing zijn: PHYDIO (ingang 0144) en FORMAT (ingang 0147).
De cartridge van het diskettestation bevat onder meer een ROM-pagina in het adresbereik 4000-7FFF die een apparaatuitbreiding bevat (zie par. 6.2.1). Verder

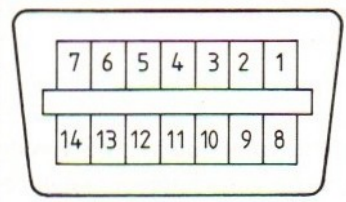
staan in deze ROM ook de routines die de DISK BASIC sleutelwoorden moeten verwerken. Daarnaast bevat de cartridge een aansluiting op het fysieke, dus het apparaat dat de diskette kan lezen en beschrijven, wel te vergelijken met een platen- of compact disk-speler.

Zoals gezegd zijn er in de MSX-standaard geen details over het diskettestation vastgelegd. Om die reden valt er in dit boek verder niet veel over te vertellen. Wie over zijn eigen diskettestation meer wil weten, zal op zoek moeten gaan naar documentatie die specifiek op dat betrekking heeft.

5.6 Printer

De printer zet tekens op papier. Via de printer-aansluiting wordt de printer verteld welke tekens er gezet moeten worden. De printer-aansluiting is een parallelle aansluiting van het type Centronics. De stekker is een 14-pins AMP of soortgelijke. In afbeelding 5.6 is de indeling en betekenis van de pinnen gegeven.

PIN.NR.	NAAM v. SIGNAAL	I/O	AANSLUITING VAN DE PINS
1	$\overline{\text{PSTB}}$	0	
2	PDB0	0	
3	PDB1	0	
4	PDB2	0	
5	PDB3	0	
6	PDB4	0	
7	PDB5	0	
8	PDB6	0	
9	PDB7	0	
10	N.V.T.	–	
11	BUSY	I	
12	N.V.T.	–	
13	N.V.T.	–	
14	GND	–	



Afb. 5.6. Aansluiting van de printer

De printer heeft twee poortadressen tot zijn beschikking. Deze adressen zijn de volgende:

Adres	*Modus*	*Toepassing*
90	L	Printer klaar (Busy-signaal)
90	S	Iets te printen (Strobe-signaal)
91	S	Code van te printen teken

Als bit 1 van de poort met adres 90 wordt gelezen, komt de waarde ervan overeen met pin 11 van de stekker. Het bit is hoog als de printer bezig is en laag als de printer klaar staat.

Als bit 1 van de poort met adres 90 wordt beschreven, komt de geschreven waarde terecht bij pin 1 van de stekker. Als er iets te printen is, wordt het bit laag gezet. De code van het te printen teken moet dan eerst naar poortadres 91 geschreven zijn. Vervolgens wordt het bit weer hoog gezet, ten teken dat er geen geldige code meer staat in de poort met adres 91.

In het OS zijn voor de printer op twee niveaus voorzieningen getroffen. Er zijn twee routines op laag niveau: LPTSTT (00A8) en LPTOUT (00A5), die respectievelijk testen op het klaar staan van de printer en het sturen van een ASCII-code naar de printer zonder te letten op de waarde van die code. Er is een routine op hoger niveau: OUTDLP (014D), die een foutmelding geeft als de printer niet klaar staat, en die besturingscodes kan herkennen en interpreteren.

De routine OUTDLP kan op verschillende manieren werken. Als de waarde van RAWPRT (F418) dat aangeeft, worden alle codes direct naar de printer gestuurd, net als in LPTOUT. Als dat niet het geval is, bepaalt de waarde van NTMSXP (F417) in hoeverre besturingscodes worden geïnterpreteerd.
Tabulatorsprongen worden door het OS uitgeteld en vervangen door een reeks spaties. In LPTPOS (F415) wordt het aantal tekens bijgehouden dat naar de printer is gestuurd na de laatste RETURN (code 0D). Als de printer een MSX-printer is (te zien aan NTMSXP), worden alle grafische codes en extensiecodes gewoon doorgestuurd. Als dat niet het geval is, worden deze codes vervangen door spaties. Als de waarde van PRTFLG (F416) dat aangeeft, schrijft de routine OUTDO (0020) alle meegegeven tekens naar de printer door een aanroep van OUTDLP.
Een MSX-printer, dat wil zeggen een printer die aan de standaard voldoet die door MSX wordt opgelegd, kent in ieder geval de volgende besturingscodes (hexadecimaal):

01 grafische extensie-kopbyte
0A een regel naar beneden
0C een pagina naar beneden
0D printerkop naar linkerkant van de regel
1B ESC

Het grafische extensie-kopbyte kan gevolgd worden door een code tussen 40 en 5F

(hex); samen duiden deze codes een van de MSX grafische-extensietekens aan. ESC kan gevolgd worden door een van de volgende codes (hex):

41 A: 6 regels per inch
42 B: 8 regels per inch
53 S: printer in grafische modus

Als de printer geen zes regels per inch kan printen, wordt in plaats daarvan elke geprinte regel gevolgd door een lege regel. De combinatie ESC+S, waarmee een grafische modus wordt binnengegaan, moet worden gevolgd door vier cijfers die samen een decimale representatie vormen van het aantal grafische tekens dat erop volgt. Een grafisch teken is een byte waarvan elk bit overeen komt met een puntje op het papier.

OS-routines die betrekking hebben op de printer:

0020	OUTDO	schrijft naar file, scherm of printer
00A5	LPTOUT	schrijft een code naar printer
00A9	LPTSTT	test op klaar staan printer
014D	OUTDLP	onderzoekt code en schrijft naar printer

Geheugenplaatsen die voor de printer worden gebruikt:

F415	LPTPOS	aantal geprinte tekens sinds RETURN
F416	PRTFLG	geeft aan of uitvoer naar printer gaat
F417	NTMSXP	geeft aan of printer MSX-printer is
F418	RAWPRT	geeft aan of codes worden geïnterpreteerd

5.7 Spelingangen

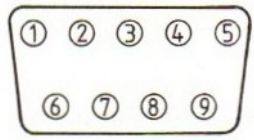
Op de MSX-computer is standaard één spelingang aanwezig. Bij uitgebreide versies kunnen twee spelingangen aangebracht zijn. De spelingangen – zo genoemd omdat ze voornamelijk worden gebruikt bij spelletjes – dienen om randapparatuur aan te sluiten waarmee op een alternatieve manier (alternatief ten opzichte van het toetsenbord) invoer naar de computer kan worden verzorgd.

In verreweg de meeste gevallen worden de spelingangen gebruikt voor het aansluiten van joysticks (spelpookjes). Daarom worden ze ook wel joystick-ingangen genoemd. Er is in het OS echter ook voorzien in andere invoerapparatuur, zoals een paddle-controller of een touch pad.

5.7.1 Het lezen van de spelingangen

De aansluiting voor de spelingangen is een vrouwelijke 9-pins AMP-aansluiting. In afbeelding 5.7 is aangegeven hoe de aansluiting is ingedeeld.

De spelingangen worden uitgelezen via registers 14 en 15 van de PSG. De bits van deze registers staan in verbinding met de pins van de ingangen, of worden gebruikt voor besturing. Deze situatie is niet afhankelijk van het soort apparaat dat op de

PIN.NR.	NAAM v. SIGNAAL	RICHTING	AANSLUITING VAN DE PINS
1	FWD	IN	
2	BACK	IN	
3	LEFT	IN	
4	RIGHT	IN	
5	+5V	——	
6	TRG 1	IN/UIT	
7	TRG 2	IN/UIT	
8	OUTPUT	UIT	
9	GND	——	

Afb. 5.7. Spelingang-aansluiting

ingangen is aangesloten. De betekenis van het hoog of laag zijn van de bits is dat wel.

PSG-register 14 kan alleen worden gelezen. De bits van het register hebben de volgende betekenis:

Bit 7	cassette-ingangssignaal
Bit 6	alleen gebruikt in Japanse MSX-computers
Bit 5	invoersignaal pin 7
Bit 4	invoersignaal pin 6
Bit 3	invoersignaal pin 4
Bit 2	invoersignaal pin 3
Bit 1	invoersignaal pin 2
Bit 0	invoersignaal pin 1

PSG-register 15 kan zowel gelezen als beschreven worden. De bits van het register hebben de volgende betekenis:

Bit 7	alleen gebruikt in Japanse MSX-computers	
Bit 6	keuze spelingang 1 of 2	1=ingang 2
Bit 5	uitvoersignaal pin 8	ingang 2
Bit 4	uitvoersignaal pin 8	ingang 1
Bit 3	uitvoersignaal pin 7	ingang 2
Bit 2	uitvoersignaal pin 6	ingang 2
Bit 1	uitvoersignaal pin 7	ingang 1
Bit 0	uitvoersignaal pin 6	ingang 1

De PSG-registers kunnen worden geadresseerd door eerst het nummer van het register te schrijven naar poortadres A0, en vervolgens een nieuwe waarde naar poortadres A1 te schrijven of de bestaande waarde van poortadres A2 te lezen.

Via register 14 moeten zowel de pins van spelingang 1 als van ingang 2 gelezen worden. De keuze tussen de twee ingangen wordt bepaald aan de hand van de waarde van bit 6 van register 15. Als dit bit laag is, wordt spelingang 1 gelezen; als het bit hoog is, wordt ingang 2 gelezen.

Zoals uit de uitleg bij register 15 blijkt, kunnen pins 6 en 7 van de spelingangen gebruikt worden voor uitvoer zowel als voor invoer. Om de ingangssignalen van pin 6 of 7 te lezen moeten de uitgangssignalen van die pin hoog zijn.

In plaats van direct de PSG-registers te adresseren kan men beter gebruik maken van de OS-routines WRTPSG (ingang 0093) en RDPSG (ingang 0096); zie par. 6.3. De correcte werking van deze routines is gegarandeerd onafhankelijk van de precieze manier van adressering. Als die manier mocht veranderen in een nieuwe versie van het MSX-systeem, zullen de OS-routines nog altijd werken.

5.7.2 Joysticks

Over het algemeen zullen joysticks (spelpoken) wel bekend zijn; een pook die onafhankelijk naar boven/beneden en naar links/rechts bewogen kan worden. Deze vier richtingen worden alle gedetecteerd met een schakelaar die 'aan' gezet wordt als de pook in de bijbehorende richting wordt bewogen. Verder kent de joystick één of twee vuurknoppen, die ook een schakelaar aansturen.

Elke schakelaar stuurt een signaal aan; er zijn maximaal zes signalen. Elk van deze signalen wordt verbonden aan een van de pins van de spelingang. De betekenis van de verschillende pins is:

1 IN: joystick naar boven
2 IN: joystick naar beneden
3 IN: joystick naar links
4 IN: joystick naar rechts
5 +5 V spanning
6 IN: vuurknop 1 ingedrukt; UIT: hoog signaal
7 IN: vuurknop 2 ingedrukt; UIT: hoog signaal
8 UIT: moet laag zijn
9 aarde

De uitvoerfunctie van pins 6 en 7 wordt niet gebruikt. Om de invoer via deze pins mogelijk te maken moet het bijbehorende uitvoersignaal hoog gemaakt worden. Het uitvoersignaal op pin 8 moet continu laag zijn, wil de joystick enig signaal geven.

Tot zover de signalen van de joysticks. Het verband tussen signalen en de waarde van registers 14 en 15 van de PSG is zoals eerder aangegeven. Om de joysticks te gebruiken, kan men deze registers zelf uitlezen, of gebruik maken van routines die door het OS beschikbaar worden gesteld voor dit doel. Het gebruik van deze routines verdient de voorkeur.
Er zijn twee OS-routines beschikbaar voor het uitlezen van joysticks. Een ervan, GTSTCK (ingang 00D5), bepaalt de stand van de spelpook (pinsignalen 1-4), de

andere, GTTRIG (ingang 00D8), reageert op het indrukken van een vuurknop (pins 6 en 7). Zie verder par. 6.3.

Vanuit BASIC zijn er verschillende manieren om gebruik te maken van de joysticks. Sleutelwoorden die op de joysticks betrekking hebben, zijn: STICK, STRIG en ON STICK.

5.7.3 Paddles

Een paddle of paddle-controller is een apparaat met een draaiknop. Deze draaiknop kan over een bepaald bereik worden gedraaid. De stand kan worden opgevraagd.

Het opvragen van de waarde van de paddle gebeurt door een korte puls uit te zenden via pin 8 van de spelingang waarop de paddle aangesloten is. Het mechanisme van de paddle reageert hierop door een signaal terug te zenden over een van de ingangspins van die spelingang. De lengte van het teruggezonden signaal is de waarde van de paddle.

Uit het voorgaande blijkt dat de paddle maar één uitgangs- en één ingangssignaal nodig heeft; bovendien kan hetzelfde uitgangssignaal meer dan een paddle aansturen. Op een spelingang kunnen dan ook zes paddles worden aangesloten. De aansluiting is dan als volgt:

1 IN: ingangssignaal paddle 1
2 IN: ingangssignaal paddle 2
3 IN: ingangssignaal paddle 3
4 IN: ingangssignaal paddle 4
5 +5 V spanning
6 IN: ingangssignaal paddle 5; UIT: hoog
7 IN: ingangssignaal paddle 6; UIT: hoog
8 UIT: aansturingspuls paddles 1-6
9 aarde

Bij het sturen van een puls via pin 8 reageren alle aangesloten paddles. Men kan vervolgens de invoer van de gewenste paddle inlezen.

Tot zover de signalen van de paddles. Het verband tussen signalen en de waarde van registers 14 en 15 van de PSG is zoals eerder aangegeven. Om de paddles te gebruiken kan men deze registers zelf uitlezen, of gebruik maken van routine GTPDL (ingang 00DE) die door het OS beschikbaar wordt gesteld voor dit doel. Het gebruik van deze routine verdient de voorkeur.
Vanuit BASIC kunnen de paddles worden uitgelezen met de functie PDL.

5.7.4 Touch pads

Een 'touch pad' is een apparaat met een plat vlak waarop aanrakingen worden geregistreerd. De plaats waar het vlak is aangeraakt, kan door de computer worden opgevraagd in de vorm van een coördinatenpaar. Ook wordt het indrukken van een 'vuur'knop op de touch pad gesignaleerd.

De touch pad die op de MSX-computer kan worden aangesloten, is een NEC PC-605. De aansluiting op de pins van de spelingangen en het protocol waarmee de coördinaten en de vuurknopstatus bepaald worden, is niet gedocumenteerd. Dit protocol is echter geïmplementeerd in de OS-routine GTPAD (ingang 00DB); zie par. 6.3. Het is dus noodzakelijk om van deze routine gebruik te maken bij toepassingen met een touch pad.
Vanuit BASIC kan een touch pad uitgelezen worden door het aanroepen van de functie PAD; zie par. 7.3.

5.8 RS232

RS232 is een verbinding die kan worden gebruikt voor communicatie met allerlei soorten randapparatuur. Er zijn een groot aantal printers die via een RS232-verbinding kunnen worden aangestuurd, maar niet via een parallelle verbinding. Ook kan RS232 worden gebruikt voor communicatie met andere computers.
De aansluiting van RS232 gebeurt via de cartridge-gleuf, net als de aansluiting van het diskettestation. Dit brengt met zich mee dat beide apparaten niet tegelijkertijd kunnen worden aangesloten, tenzij men de beschikking heeft over meer dan één cartridge-gleuf. Details over de aansluiting van de cartridge-gleuf zijn te vinden in par. 5.3.

De cartridge van de RS232-aansluiting bevat een ROM-geheugenpagina in het adresbereik 4000-7FFF(Hex). Deze pagina bevat een apparaat- en een statement-uitbreiding (zie par. 6.2.1). Verder bevat de cartridge een i-8251 communicatie-chip en een i-8253 programmeerbare klokchip.

Eigenschappen van de RS232
Een RS232-verbinding is een verbinding voor het serieel verzenden van gegevens. Seriële verzending betekent dat de bits van één byte één voor een worden verstuurd via hetzelfde signaal, in plaats van alle tegelijk zoals bij een parallelle verbinding.

De RS232 is een standaard voor seriële verbindingen die enige eigenschappen vastlegt. Helaas houdt deze standaard niet in dat alle aansluitingen die eraan voldoen ook met elkaar verbonden kunnen worden.
We zullen hier globaal de eigenschappen bespreken van de RS232 die op de MSX-computer kan worden aangesloten. Deze aansluiting heeft meer mogelijkheden dan de meeste andere RS232-verbindingens. Een gedetailleerde bespreking valt buiten de doelstelling van dit boek, aangezien het een uitbreiding van de MSX-computer betreft.

De belangrijkste signalen in de RS232-verbinding zijn de datasignalen. Dit zijn er twee: een voor inkomende, en een voor uitgaande signalen. Via dit signaal worden de bits verstuurd die samen de databytes vormen.

Het versturen van de bytes kan op een aantal manieren gebeuren. Deze manieren betreffen het aantal besturingsbits dat met het byte mee gestuurd worden, zoals

startbits vóór elk byte, stopbits erna, parity-checkbits (deze dienen als test of de overige bits wel correct zijn overgekomen).
Voorts kan het versturen van gegevens via de RS232 met een hele reeks snelheden gebeuren. De snelheid van versturen en de snelheid van ontvangen zijn beide in te stellen.
Door een slimme selectie van de besturingsbits en de snelheid van versturen/ontvangen kan meestal wel een verbinding worden gelegd tussen twee RS232-aansluitingen.

Naast de signalen voor versturen van gegevens kent de RS232 een aantal besturingssignalen. Deze signalen worden wel aangeduid met 'handshake'-signalen. Het doel is te voorkomen dat de RS232-aansluitingen aan beide kanten van een verbinding uit fase raken ten opzichte van elkaar, en bijvoorbeeld proberen te lezen op een moment dat er geen geldige gegevens verstuurd worden.

Poortadressen
Voor de RS232 zijn in de MSX-computers een aantal poortadressen gereserveerd, namelijk 80-87. In de nu volgende tabel staat het gebruik van deze adressen vermeld. Deze tabel is alleen van nut voor degenen die bekend zijn met de chips in de RS232-cartridge.

Adres	*Modus*	*Toepassing*
80	L/S	i-8251 gegevenstransmissie
81	L/S	i-8251 commando/statusregister
82	L	toestand van de RS232-verbinding
83	S	interruptmasker
84	L/S	i-8253 teller 0
85	L/S	i-8253 teller 1
86	L/S	i-8253 teller 2
87	S	i-8253 instelling

De ROM van de RS232 bevat een apparaatuitbreiding en een statement-uitbreiding. Voor algemene uitleg van deze begrippen zie par. 6.2.1. Een gedetailleerde bespreking valt buiten het bereik van dit boek.

6 Het Operating System

Het belangrijkste stuk programmatuur dat bij elke computer wordt geleverd, is wel het Operating System – af te korten tot OS. Het is dit programma dat in principe alle in- en uitvoer van en naar het toetsenbord, beeldscherm, cassette, diskette enzovoort verzorgt. Als het Operating System van een computer goed ontworpen is, kan dit voor een programmeur – in BASIC, machinetaal of welke andere taal dan ook – heel wat werk besparen.

Op het gebied van Operating Systems is al heel wat onderzoek verricht. Voor 'grote' microcomputers is al jaren een heel scala van OS beschikbaar. Veel OS zijn op meer dan één computer geïmplementeerd. Dit heeft voor de gebruikers van die computers als voordeel dat ze elkanders bestanden en soms zelfs programma's kunnen gebruiken, zodat een heleboel werk maar één keer hoeft te worden gedaan.

6.1 Wat doet het OS?

Een Nederlands woord voor Operating System is 'besturingssysteem' (we zullen hier echter de term Operating System, kortweg OS, aanhouden). Het woord besturingssysteem geeft echter wel duidelijk aan waar het om gaat: het besturen van de computer.

Als 'kale' machine – dat wil zeggen, zonder enige programmatuur – is een computer nauwelijks bruikbaar. Voordat er iets mee gedaan kan worden, moet de computer (bijvoorbeeld) eerst op de een of andere manier kunnen ontdekken dat er iets op het toetsenbord wordt ingetypt. Dat betekent dat er op het indrukken van een toets getest moet worden en er een programma moet zijn dat het testwerk doet. Daarmee is de cirkel rond: dat programma is er juist niet.
Waar het op neer komt is dat zonder programmatuur een computer niet meer is dan een brok onhandelbare apparatuur. Om de apparatuur te beheersen, is een besturingssysteem nodig. Het OS is in feite een verzameling routines die gebruikt kunnen worden zodra er in een programma iets anders gedaan moet worden dan puur en alleen rekenwerk.
Bij het zelf schrijven van machinetaalprogramma's is het sterk aan te raden om waar mogelijk gebruik te maken van OS-routines. Dit kan men doen d.m.v. de OS-ingangen die in dit hoofdstuk besproken zullen worden.
Het voordeel van het gebruik van OS-routines boven het zelf schrijven van soortgelijke routines, is de onafhankelijkheid van veranderingen in apparatuur – zoals die eventueel zouden kunnen optreden in nieuwe versies van de MSX-computer omdat

deze veranderingen worden opgevangen in het OS. Dit wordt ook wel aangeduid met overdraagbaarheid van programma's.

Waar mogelijk proberen we in dit hoofdstuk het OS gescheiden te houden van het BASIC-systeem. De programmatuur van deze twee systemen heeft namelijk in principe niets met elkaar uit te staan. In het ontwerp van de MSX-computer zijn deze systemen echter niet strikt gescheiden gehouden, en zodoende is het niet mogelijk om het OS en BASIC helemaal los van elkaar te bespreken. Dit komt vooral naar voren bij het bespreken van de RAM-lokaties van het OS (par. 6.4).
Het OS van de MSX-computer is op drie punten te benaderen: de OS-ingangen, die toegang geven tot OS-routines; het werkgeheugen, waarin de huidige toestand van het OS opgeslagen is; en de RAM-haken, die een uitbreidingsmogelijkheid van de MSX-computer vormen. Deze punten zullen in de volgende paragrafen aan de orde komen.
In de laatste paragraaf van dit hoofdstuk gaan we kort in op de mogelijkheden van MSX-DOS, een alternatief of uitbreiding van het huidige OS van de MSX-computer.

6.2 De werking van het OS

Het OS van de MSX-computer is opgebouwd uit modules. De modulaire opbouw is tamelijk ver doorgevoerd. Elke module verzorgt een subsysteem van het OS dat (binnen zekere grenzen) onafhankelijk is van de andere subsystemen.
In deze paragraaf worden de modules van het OS stuk voor stuk besproken. Van elk subsysteem bespreken we de eigenschappen en de globale werking. De manier waarop de subsystemen kunnen worden gebruikt, met behulp van de OS-ingangen, komt in de volgende paragraaf aan de orde.

6.2.1 Het gleufsysteem

Het gleufsysteem van het OS omvat het beheer en gebruik van de primaire en secundaire gleuven van het geheugen van de MSX-computer.
Zoals uit hoofdstuk 2 bekend is, is het geheugen van de MSX-computer georganiseerd als een stelsel van gleuven. Er is ruimte voor vier primaire gleuven die elk zijn uit te breiden tot vier secundaire gleuven. Elke gleuf – primair of secundair – kan 64 kbyte aan geheugen bevatten. Het geheugenbereik van een gleuf kan opgedeeld worden in vier pagina's van elk 16 kbyte. In totaal heeft de MSX-computer een capaciteit van $4\times4\times4=64$ pagina's. Elke pagina kan RAM- of ROM-geheugen bevatten.

Bij de initialisatie – dit gebeurt normaal gesproken bij het aanzetten van de computer, en daarna niet meer – is het eerste wat het OS doet, de gleuven aflopen en een overzicht maken van de geïnstalleerde pagina's met de nummers 1 en 2. Dit overzicht bestaat uit een aantal tabellen waarin gegevens over de situatie van de gleuven en de inhoud van de verschillende pagina's staan
Er zijn vier tabellen: EXPTBL, SLTTBL, SLTATR en SLTWRK.

EXPTBL (FCC1) bevat per primaire gleuf een byte waarvan de waarde aangeeft of de gleuf is uitgebreid.

SLTTBL (FCC5) bevat per primaire gleuf het nummer van de secundaire gleuf die op dit moment is ingeschakeld. De waarde van het nummer in SLTTBL is alleen geldig als het byte in EXPTBL aangeeft dat de gleuf inderdaad is uitgebreid.

SLTATR (FCC9) bevat per pagina één byte waarin aangegeven is wat voor soort inhoud de pagina heeft, aangenomen dat de pagina ROM-geheugen het volgende bevat: statement-uitbreiding, apparaatnaam-uitbreiding een BASIC-programma, of een combinatie van deze drie.

Hoewel bij elke pagina een element in SLTATR hoort, worden alleen pagina's met nummer 1 en 2 werkelijk onderzocht op inhoud.

SLTWRK (FD09) bevat per pagina twee bytes werkgeheugen, waarvan het gebruik vrij staat aan de pagina.

Een meer gedetailleerde beschrijving van deze tabellen is te vinden in par. 6.4.1.

We kunnen nu een indeling maken van de geïnstalleerde pagina's op grond van de manier waarop ze door het gleuf-systeem van het OS behandeld worden. Alle geïnstalleerde pagina's vallen in een van de volgende categorieën:

1. Systeempagina's. Dit zijn pagina's die ROM-geheugen bevatten, en die noodzakelijk zijn voor het functioneren van het MSX-systeem.
2. RAM-pagina's. Alle pagina's die alleen RAM-geheugen bevatten.
3. Uitbreidingspagina's. Dit zijn pagina's met ROM-geheugen die hetzij uitbreidingen van het systeem, hetzij toepassingsprogramma's (zoals spelletjes) bevatten.

Systeempagina's

In het standaard MSX-systeem zijn twee systeempagina's geïnstalleerd, die zich moeten bevinden in primaire gleuf 0 (en daarvan secundaire gleuf 0 indien primaire gleuf 0 uitgebreid is). Deze systeempagina's hebben de nummers 0 en 1. Ze bevatten het OS en de BASIC-interpreter.

RAM-pagina's

Het standaard MSX-systeem heeft tenminste één pagina RAM-geheugen nodig, met nummer 3. De maximale hoeveelheid RAM-geheugen die door het OS en BASIC gebruikt kan worden, is twee pagina's, genummerd 2 en 3, met samen 32 kbyte geheugen.

De RAM-pagina's mogen zich in elke willekeurige primaire of secundaire gleuf bevinden. Bij initialisatie van de computer zoekt het OS in alle gleuven naar geïnstalleerde RAM-pagina's met in het geheugenbereik paginanummers 2 en 3. Als er meer dan één RAM-pagina met nummer 2 of 3 geïnstalleerd is, wordt die pagina

gebruikt waarvoor de uitdrukking 4*prim. gleufnr + sec. gleufnr het kleinst is (dit is namelijk de pagina die het eerste gevonden wordt).

Natuurlijk kunnen er in de MSX-computer meer dan twee pagina's RAM-geheugen geïnstalleerd zijn; dit geheugen wordt echter niet door het OS of BASIC gebruikt. Het geheugen kan wel met behulp van enige OS-routines worden gelezen en beschreven. Met behulp van deze routines kan het geheugen worden benut door er gegevensstructuren in op te bergen.
Alle RAM-pagina's kunnen in principe ook worden benut om machinetaalprogramma's in op te slaan en uit te voeren. Echter, wil men gebruik maken van een of andere OS-routine (en het is bijna ondenkbaar dat dit niet zo zou zijn, aangezien men anders in feite een nieuw OS moet schrijven) dan moeten de pagina's 0 en pagina 3 bewaard blijven. Een alternatief als men toch pagina 0 of 3 wil omschakelen, is om de relevante delen van het OS en het werkgeheugen eerst naar de te gebruiken pagina's te kopiëren.

Uitbreidingspagina's
Standaard bevat het MSX-systeem geen uitbreidingspagina's. Sommige MSX-fabrikanten hebben hun computer echter wél voorzien van uitbreidingspagina's.
Het OS onderscheidt vier functies of onderdelen van uitbreidingspagina's. Dit zijn:

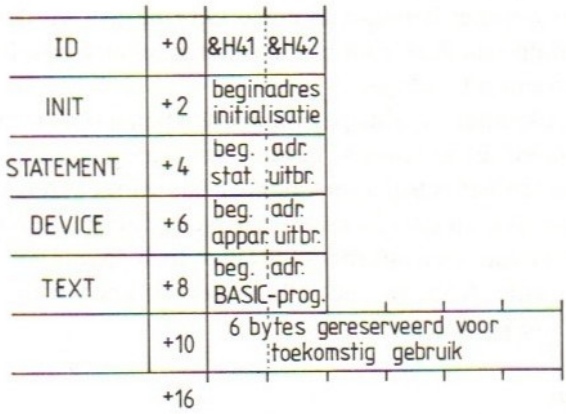
1. Een machinetaalprogramma.
2. Een apparaat-uitbreiding. Dit is programmatuur die het mogelijk maakt een nieuwe apparaatnaam te gebruiken in een file-specificatie; zie par. 6.5. Alles dat een file kan verwerken kan een apparaatnaam krijgen.
   In de meeste gevallen hoort bij een apparaatnaam ook een feitelijk apparaat. Dit apparaat kan via de cartridge aangesloten worden als de pagina zich in een cartridge bevindt, of kan binnen de computer aanwezig zijn als de pagina intern geïnstalleerd is.
3. Een BASIC-uitbreiding. Met behulp van het CALL-statement kan de verzameling BASIC-statements uitgebreid worden via een ROM-pagina.
4. Een BASIC-programma.

Bij initialisatie zoekt het OS naar uitbreidingspagina's in het geheugenbereik van 4000-BFFF (dus met nummers 1 of 2). Elke ROM-pagina in dat bereik waarvan de eerste twee geheugenplaatsen &H41 respectievelijk &H42 bevatten (de ASCII-codes voor 'AB') wordt beschouwd als uitbreidingspagina.
Het eerste gedeelte van elke uitbreidingspagina moet ingedeeld zijn volgens afbeelding 6.1.

ID is het herkenningsveld van een ROM-pagina. Het bestaat uit twee bytes, die respectievelijk &H41 en &H42 bevatten: de ASCII-codering voor de letters 'AB'.

INIT bevat het beginadres van de initialisatieroutine van deze pagina. Als INIT=0000 dan heeft de pagina geen initialisatieroutine. Het OS roept via het veld INIT de routines van alle geïnstalleerde pagina's aan alvorens de besturing aan de BASIC-interpreter over te dragen.

De initialisatieroutine kan worden gebruikt als de pagina een apparaatnaam- of statement-uitbreiding bevat, om allerlei beginwaarden te zetten en eventueel eigen geheugen te reserveren. Normaal gesproken eindigt de routine met een RET-instructie.



Afb. 6.1. De eerste bytes van een uitbreidingspagina

Het veld INIT kan echter ook worden gebruikt als entree voor een machinetaalprogramma dat zich in de pagina bevindt. Dit programma hoeft dan niet noodzakelijkerwijs te eindigen: als het programma bijvoorbeeld een spelletje is, hoeft de besturing nooit meer aan een ander programma of het OS te worden overgedragen.

STATEMENT bevat het beginadres van de BASIC-statement-uitbreiding in de pagina, indien aanwezig. STATEMENT=0000 als er geen statement-uitbreiding is.
Alleen pagina's met nummer 1 kunnen een statement-uitbreiding bevatten. Als in pagina's met een ander nummer het veld STATEMENT is ingevuld, kan dit tot problemen leiden.
Als voor een bepaalde ROM-pagina met nummer 1 of 2 het veld STATEMENT is ingevuld, wordt in het element van de SLTATR-tabel dat bij die pagina hoort, aangegeven dat de pagina een statement-uitbreiding bevat.

Als er bij het uitvoeren van een BASIC-programma een CALL-statement wordt gevonden dan wordt de 'identifier', die volgt op het woord CALL in het RAM-geheugen van de OS gezet, in het gebied van PROCNM (vanaf FD89) en afgesloten door ASCII-code 0. Vervolgens worden alle aanwezige uitbreidingspagina's afgelopen die een statement-uitbreiding bevatten.
Bij binnenkomst van een statement-routine wijst het HL-registerpaar naar het eerste niet-spatieteken in de tekst van het CALL-statement. Als de naam van het statement door de routine wordt herkend, moet het statement worden uitgevoerd en moet de routine worden verlaten met de carry-vlag van het F-register laag. Tevens moet HL naar het eerste teken achter het statement wijzen (een dubbele punt of het einde van een regel). Als de statement-naam niet wordt herkend, moet de routine

verlaten worden met de carry hoog en HL onveranderd. Alle andere registers mogen verloren gaan.

DEVICE bevat het beginadres van de apparaat-uitbreiding in de pagina, indien aanwezig. DEVICE=0000 als er geen apparaat-uitbreiding is.
Alleen pagina's met nummer 1 kunnen een apparaat-uitbreiding bevatten. Als in pagina's met een ander nummer het veld DEVICE is ingevuld, kan dit tot problemen leiden.
Als voor een bepaalde ROM-pagina met de nummers 1 of 2 het veld DEVICE ingevuld is, wordt in het element van de SLTATR-tabel dat bij die pagina hoort, aangegeven dat de pagina een apparaat-uitbreiding bevat.
Elke pagina kan routines voor maximaal vier apparaatnamen bevatten. Deze namen worden genummerd van 0 t/m 3.

Een pagina kan worden aangeroepen via het adres in DEVICE met een waarde in register A die aangeeft wat voor soort actie er ondernomen moet worden. Dit is een waarde uit de volgende lijst:

waarde	*actie*
00	open een file
02	sluit een file
04	doe random 1/0
06	die sequentiële uitvoer
08	doe sequentiële invoer
0A	voer de functie LOC uit
0C	voer de functie LOF uit
0E	voer de functie EOF uit
10	voer de functie FPOS uit
12	ga een teken terug
FF	geef het nummer dat bij de naam hoort

Bij een waarde van A kleiner dan &HFF staat het nummer van de apparaatnaam in het geheugenadres DEVICE (FD99). Als A=&HFF, dan staat de tekst van de apparaatnaam in PROCNM (vanaf FD89). In dat geval moet bij het verlaten van de routine het nummer dat bij de naam hoort in A staan, en de carry laag zijn. Als de naam niet bekend is, moet de carry bij het verlaten hoog zijn.
Eventuele andere in- en uitvoer-parameters, zoals te lezen of te schrijven tekens, zijn te vinden in het geheugengebied van de BASIC-interpreter: het type staat in VALTYP (F663), de waarde in DAC (vanaf F7F6). Zie par. 6.4.1. voor meer details.

TEXT is het beginadres van een eventueel BASIC-programma dat zich in de ROM-pagina bevindt. TEXT=0000 als er niet zo'n BASIC-programma is. Als TEXT is ingevuld, moet het veld wijzen naar het byte vóór het eerste regelnummer; dit byte moet 0 bevatten.
Alleen pagina's met nummer 2 kunnen een BASIC-programma bevatten. Als in

pagina's met een ander nummer het veld TEXT is ingevuld, kan dit tot problemen leiden.
Als voor een bepaalde ROM-pagina met de nummers 1 of 2 het veld TEXT ingevuld is, wordt in het element van de SLTATR-tabel dat bij die pagina hoort, aangegeven dat de pagina een BASIC-programma bevat.
Een BASIC-programma in een ROM-pagina wordt uitgevoerd nadat de INITialisatie-routines van alle uitbreidingspagina's zijn uitgevoerd. Tijdens het uitvoeren van een BASIC-programma geeft de switch in BASROM (FBB1) aan dat het programma dat wordt uitgevoerd in ROM staat. Dit heeft onder meer gevolgen voor de plaats van het variabelengebied binnen het geheugen.
De manier om een BASIC-programma te coderen en in ROM te zetten, wordt uitgebreid behandeld in hoofdstuk 7. Het is aan te raden om alle regelnummers in het programma om te zetten naar regelwijzers alvorens het programma in ROM te zetten: dit zal de werking versnellen.

6.2.2 Het interrupt-systeem

De Z80- microprocessor kent drie soorten interrupts. Dit zijn de BUSRQ, NMI en INT, in volgorde van afnemende prioriteit. In het MSX-systeem wordt alleen de INT gebruikt, hoewel er voor NMI in de vorm van een RAM-haak een voorziening is getroffen.

NMI

De NMI, of Non Maskable Interrupt, wordt standaard niet toegepast in de MSX-computer. Het is alleen mogelijk deze interrupt te gebruiken door intern onderdelen aan de computer toe te voegen.
Een NMI resulteert in een sprong naar adres 0066. Dit adres is dan ook vrijgelaten voor de NMI. Het adres bevindt zich tussen de OS-ingangen.
Het resultaat van het aanroepen van de NMI-ingang is een sprong naar de RAM-haak H.NMI (FDD6). Wie de NMI aan het MSX-systeem toevoegt moet dus zorgen dat de RAM-haak verbonden wordt met een interrupt-afhandelingsroutine.
Onder MSX-DOS is het niet mogelijk de NMI te ondersteunen. Adres 0066 wordt daar gebruikt voor FCB-informatie.

INT

De interrupt-modus van INT is permanent ingesteld op 1; dat wil zeggen dat bij elke INT een RST &H38 wordt uitgevoerd. Het hele OS is ingericht op interrupt-modus 1, en het is niet mogelijk deze modus binnen het MSX-systeem te veranderen.
Standaard is er in de MSX-computer maar één onderdeel dat interrupts genereert, en dat is de VDP. De algemene eigenschappen van de VDP zijn in detail besproken in hoofdstuk 3; we zullen hier alleen ingaan op het geven van de interrupt.
Het is mogelijk via het cartridge-systeem, of eventueel intern, onderdelen toe te voegen die een INT genereren. Hierin is voorzien in het OS door middel van de RAM-haak H.KEYI (FD9A). Deze RAM-haak wordt bij elke INT aangeroepen. Het is de taak van degene die een INT heeft toegevoegd om via H.KEYI 'zijn' INT te scheiden van de VDP-interrupt.

De rest van deze paragraaf zal worden besteed aan de afhandeling van de INT die door de VDP wordt gegenereerd. (Het zou misschien juister zijn om te zeggen dat de interrupt wordt gegenereerd door de interne klok, aangezien het die klok is die de VDP aanstuurt. Aangezien het fysische signaal van INT echter uit de VDP-chip komt, en deze chip ook de frequentie van de interrupt bepaalt, zullen we het verder hebben over de VDP-interrupt).
De belangrijkste actie die aangestuurd wordt door de VDP-interrupt is de toetsenbord-scan, die eens in de drie INTs plaatsvindt. Een teller, SCNCNT (F3F6), houdt bij of het tijd is voor de scan. Dit komt verder aan de orde in par. 6.2.6.
Het zij opgemerkt dat deze situatie niet leidt tot een erg goede respons van het toetsenbord. Als er net een toets wordt ingedrukt terwijl er niet wordt gescand, wordt de toetsaanslag over het hoofd gezien. Dit euvel zou verholpen zijn als het toetsenbord zelf bij elke toetsaanslag een interrupt zou genereren. Helaas is dit in het MSX-systeem uitgesloten.

Een volgende functie van de VDP-interrupt is het bijhouden van een software-klok. Deze klok wordt bij elke INT opgehoogd. In de USA-versie van de MSX-computer, waar de VDP-interrupt met een frequentie van 60 Hz gegenereerd wordt, loopt de software-klok dus sneller dan in de Europese versie, waarin volstaan wordt met een frequentie van 50 Hz.
In situaties waar de INT dikwijls wordt uitgeschakeld, bijvoorbeeld tijdens interactie met randapparatuur, gaat de software-klok onregelmatig lopen.

Verder wordt de VDP-interrupt gebruikt voor het bijhouden van het geluids-subsysteem. Dit komt aan de orde in par. 6.2.4.

Tenslotte wordt, tegelijk met het scannen van het toetsenbord, getest op allerlei voorwaarden die aanleiding kunnen geven tot zogenaamde software-interrupts.

Software-interrupts
Een software-interrupt is een signaal dat wordt gegeven wanneer er aan een of andere voorwaarde is voldaan, door aan een of andere geheugenplaats een van tevoren afgesproken waarde te geven. Het signaal kan worden opgevangen binnen elke willekeurige routine door te testen op de waarde van de betreffende geheugenplaats.

De kenmerkende eigenschap van een software-interrupt is dat de test op de interrupt-voorwaarde regelmatig gebeurt, op de maat van een hardware-interrupt. Dat betekent dat deze test los staat van elke routine die van de software-interrupt gebruik wil maken. Anderzijds is het binnen de routine waarin getest wordt op de voorwaarde ook niet bekend waar het signaal wordt gebruikt.

Het MSX-systeem kent de volgende software-interrupts:
- het indrukken van een functietoets
- het indrukken van CTRL+STOP
- het indrukken van een vuurknop
- het verstrijken van een tijdsinterval
- een sprite-botsing

Bij elk van deze interrupts hoort een element van de tabel TRPTBL (vanaf adres FC4C). Dit element wordt gebruikt voor het signaleren van de interrupt-voorwaarde, en om gegevens op te slaan omtrent de reactie op de interrupt. Elk element van TRPTBL bestaat uit drie bytes:

byte 0	-toestandgegevens
byte 1+2	-reactiegegevens

De toestandgegevens bepalen de instelling van de interrupt. Het gebruik van de bytes voor reactiegegevens is vrij aan de gebruiker van de interrupt.

Byte 0 (voor de toestandgegevens) is op de volgende manier ingedeeld:

Bit 7-3	- ongebruikt	
Bit 2	- interrupt opgetreden ja/nee	1 = ja
Bit 1	- in wachttoestand ja/nee	1 = ja
Bit 0	- ingeschakeld ja/nee	1 = ja

Als er aan een van de interrupt-voorwaarden voldaan is, wordt het bijbehorende element van TRPTBL opgezocht en wordt daarin bit 2 van byte 0 gezet om de interrupt te signaleren, mits de interrupt ingeschakeld is – dit is te zien aan de waarde van bit 0. Als de interrupt niet in wachttoestand is – dit is te zien aan bit 1 – wordt bovendien de waarde van ONGSBF (FBD8) opgehoogd.
In ONGSBF wordt bijgehouden hoeveel interrupts er zijn opgetreden die ingeschakeld en niet in wachttoestand waren.
Door de waarde van ONGSBF te testen, kan snel worden vastgesteld of er interessante interrupts zijn opgetreden zonder heel TRBTL te hoeven doorlopen.
Een interrupt die in wachttoestand staat, wordt wel gesignaleerd maar niet in ONGSBF aangegeven.

De nu volgende tabel geeft aan welk element van TRPTBL bij welke interrupt horen.

elementnr.	*beginadr.*	*interrupt*
0	FCAC	functietoets 1
1	FC4F	functietoets 2
2	FC52	functietoets 3
3	FC55	functietoets 4
4	FC58	functietoets 5
5	FC5B	functietoets 6
6	FC5E	functietoets 7
7	FC61	functietoets 8
8	FC64	functietoets 9
9	FC67	functietoets 10
10	FC6A	STOP-toets
11	FC6D	sprite-botsing
12	FC70	spatiebalk (vuurknop 0)

13	FC73	joystick 1 knop 1 (vuurknop 1)
14	FC76	joystick 2 knop 1 (vuurknop 2)
15	FC79	joystick 1 knop 2 (vuurknop 3)
16	FC7C	joystick 2 knop 2 (vuurknop 4)
17	FC7F	interval

In BASIC worden de software-interrupts volledig ondersteund. Elk van de interrupts kan in- of uitgeschakeld worden of in wachttoestand worden gezet. In bytes 1 en 2 van het element in TRPTBL komt het adres te staan van de regel waarheen gesprongen moet worden wanneer de interrupt optreedt.

Aan het eind van het uitvoeren van een BASIC-statement wordt getest op een verandering in TRPTBL, en wordt indien nodig een interrupt-routine aangeroepen.

6.2.3 Het queue-systeem

Een 'queue' is een gegevensstructuur die zich gedraagt als een buis. Aan de ene kant worden er dingen in gestopt die er in dezelfde volgorde aan de andere kant weer uitkomen. Deze structuur wordt ook wel aangeduid met 'FIFO-structuur' (First In First Out) of 'buffer'.

Het queue-systeem van de MSX-computer maakt het moeilijk om een queue te gebruiken zonder kennis van de interne structuur van de queue. Het enige wat er van het systeem 'zichtbaar' is, zijn twee OS-ingangen: LFTQ (00F6) en PUTQ (00F9), en een paar RAM-geheugenplaatsen: QUEUES (F3F3), QUETAB (vanaf F959), QUEBAK (vanaf F971), en de ruimte voor de queues zelf.

Het queue-systeem kan de volgende acties op een queue ondernemen:

INITQ: het initialiseren van een queue. Dit houdt in dat er voor de queue geheugen wordt gereserveerd.

PUTQ: een element aan een queue toevoegen (achter in de buis stoppen). Voor PUTQ is op adres 00F9 een OS-ingang verzorgd.

GETQ: een element van de queue afhalen (uit de voorkant van de buis halen).

BCKQ: een element voorin de queue zetten (in de voorkant van de buis terugduwen), zodat het element bij de volgende GETQ gelezen wordt. Dit kan maar met één element tegelijk.

LFQT: het aantal elementen bepalen dat in een queue aanwezig is. Voor LFTQ is op adres 00F6 een OS-ingang verzorgd.

Per queue houdt het queue-systeem een 'descriptor' bij. Deze descriptor bevat de volgende elementen:
- het beginadres van de ruimte die voor de queue gereserveerd is.
- het aantal elementen dat de queue maximaal kan bevatten.
- een wijzer naar de eerste lege plaats in de queue (de achterkant van de buis).

- een wijzer naar het 'oudste' element van de queue (de voorkant van de buis).
- een vlag die aangeeft of er een element voorin de queue is teruggezet (met BCKQ).

De descriptoren van alle queues zijn te vinden vanaf het adres dat in QUEUES (F3F3) staat. In het huidige MSX-systeem is dit adres gelijk aan QUETAB (F959).

Behalve ruimte voor de queue zelf is er ruimte gereserveerd voor een element dat eventueel voorin de queue is teruggezet. Zo'n element komt dus niet werkelijk voorin de queue te staan; het effect is echter hetzelfde. De ruimte voor de teruggezette tekens is QUEBAK (vanaf F970).
Het queue-systeem van het MSX-systeem wordt gebruikt door het geluids-subsysteem en voor de RS232-interface.

6.2.4 Het geluidssysteem

Het is mogelijk om waarden direct naar de PSG-registers te schrijven. Sommige geluidseffecten kunnen alleen op die manier bereikt worden. Het OS van de MSX-computer voorziet echter in een geluidssysteem dat het mogelijk maakt om het aansturen van de PSG als achtergrondtaak te laten plaatsvinden.

Het geluidssysteem is geënt op het queue-systeem en het interrupt-systeem, en is in dat opzicht eigenlijk meer een subsysteem. drie van de queues van het queue-systeem, met nummers 0-2, worden gebruikt voor het geluidssysteem. Elk van deze queues komt overeen met een van de stemmen van de PSG.
De werking van het geluids-subsysteem berust erop dat op gezette tijden, vastgesteld door het interrupt-systeem, geluidscommando's van de geluids-queues worden gehaald en uitgevoerd. De geluidscommando's bepalen de waarden die naar de registers van de PSG worden geschreven. De geluidscommando's moeten door de gebruiker op de queues gezet worden. Dit kan bijvoorbeeld met behulp van het BASIC-statement PLAY, of met behulp van de OS-ingangen van het queue-systeem (zie par. 6.2.3.).

Niet aan alle registers van de PSG kan met gebruik van een geluidscommando een nieuwe waarde worden gegeven. De belangrijkste uitzondering wordt gevormd door de registers die de ruis besturen. Deze registers zullen op een meer directe manier beschreven moeten worden. Hiervoor zijn aan het OS de ingangen WRTPSG (0093) en RDOSG (0096) toegevoegd.

Een geluidscommando van het geluids-subsysteem ziet er als volgt uit:
- Elk commando bestaat uit elementen. De elementen van een commando vallen in twee groepen: synchronisatie-elementen en besturingselementen.
- Het eerste element van een commando is een synchronisatie-element: alle volgende elementen, indien aanwezig, zijn besturingselementen.
- Een synchronisatie-element bestaat uit twee bytes, die in de queue staan met het hoge-ordebyte voorop. De bits van een synchronisatie-element hebben de volgende betekenis:

Bit 15-13	aantal bytes dat volgt	bereik: 0-7
Bit 12-0	wachttijd na uitvoering	bereik: 0'8096

Het aantal bytes dat volgt is het aantal bytes van besturingselementen die bij dit geluidscommando horen. De wachttijd na uitvoering slaat op de tijd die gewacht moet worden nadat dit geluidscommando is uitgevoerd totdat het volgende commando uit de queue mag worden gehaald. Deze tijd wordt opgegeven in tijdseenheden van de VDP-interrupt, aangezien de tijd door deze interrupt wordt bijgehouden. De frequentie van de VDP-interrupt is 50 Hz.

- Een besturingselement bestaat uit één tot drie bytes. Er bestaan vier vormen van het besturingselement. De vorm die een besturingselement heeft, wordt bepaald door de bits 7-6 van het eerste byte van het element:

bits 7/6	*#bytes*	*besturing*
00	2	toonhoogte
01	3	snelheid van volumeverloop
10	1	volume
11	3	volume + snelheid van volumeverloop

- Het toonhoogte-besturingselement bestaat uit twee bytes. De 12 minst significante bits leveren de waarde op die naar het register voor de toonhoogte van deze stem geschreven moet worden.

- Het besturingselement voor de snelheid van volumeverloop bestaat uit drie bytes. Het tweede en derde byte worden naar de PSG-registers 12 respectievelijk 11 geschreven, die de snelheid van het volumeverloop bepalen.

- Het volume-besturingselement bestaat uit één byte. De bits van dit byte hebben de volgende betekenis:

Bit 7-6	soort besturingselement (zie boven)
Bit 5	ongebruikt
Bit 4	instelling verloopnr. ja/nee ' 1 = ja
Bit 3-0	instelwaarde bereik: 0–15

  De waarde van bits 4-0 wordt het volumeregister dat bij deze stem hoort geschreven.
  De waarde van bit 4 geeft aan of dit element tevens het nummer van het gebruikelijke volumeverloop bepaalt. Als het bit hoog is, staat het nummer van het volumeverloop in bits 3-0. Dit nummer wordt naar register 13 geschreven.
- Het besturingselement voor volume + snelheid van verloop is een combinatie van de losse besturingselementen voor die instellingen. Het element bestaat uit drie bytes, waarvan het eerste byte gelijk is aan het besturingsbyte voor volume, en de volgende twee bytes gelijk zijn aan de besturingsbytes voor de snelheid van het volumeverloop.

De precieze betekenis van de waarden die met behulp van de geluidscommando's ingesteld worden, is besproken in hoofdstuk 4.

Behalve een geluidscommando kan er nog een ander commando op een geluidsqueue staan: het EOS-commando (End Of Sound). Dit commando bestaat uit één enkel byte, met waarde &HFF. Op de betekenis van dit commando komen we nog terug.

De werking van het geluidssysteem berust op de vlaggen in MUSICF (FB3F) en de tellers van de verschillende stemmen in het gebied van VCBA (vanaf FB41), VCBC (vanaf FB8B). Het interrupt-systeem van het OS gaat uit van de waarde van MUSICF. Als dit byte aangeeft dat een bepaalde stem op dit moment actief is, wordt de queue van die stem in behandeling opgenomen.
Allereerst wordt de teller van de stem verlaagd. Als daardoor de teller op 0 blijkt te zijn gekomen, wordt het volgende commando van de queue gehaald en uitgevoerd. Als op een gegeven moment een EOS-commando in een queue gevonden wordt, wordt de vlag in MUSICF die hoort bij de stem van die queue teruggezet, en wordt het volume van de stem op 0 gezet.
Voor het beëindigen van de werking van het geluidssysteem, en het opnieuw initialiseren van alle geheugenplaatsen die door het BASIC-statement PLAY worden gebruikt, kan de OS-ingang GICINI (0090) worden gebruikt.

Voor het opstarten van het geluidssysteem wanneer alle stemmen beëindigd zijn, kan de OS-ingang STRTMS (0099) worden gebruikt. Deze routine gaat alleen tot opstarten over als alle stemmen op dit moment zijn beëindigd, en als de geheugenplaats PLYCNT (FB40) een positieve waarde heeft.
Alle stemmen worden tegelijkertijd opgestart; het is dus zaak te zorgen dat er in de queues van stemmen die niet worden gebruikt tenminste een EOS-commando staat.
Het is mogelijk een stem van het geluidssysteem op te starten zonder gebruik te maken van STRTMS, door de vlag in MMUSICF die bij de stem hoort hoog te zetten en de teller van de stem in VCBA, VCBB of VCBC op 1 te zetten! Dit laatste is nodig omdat de teller eerst wordt verlaagd voordat er op 0 wordt getest.

6.2.5 Het cassettesysteem

Met het cassettesysteem worden de OS-routines met werkruimte aangeduid die ervoor zorgen dat er gegevens van en naar de cassetterecorder kunnen worden gestuurd.

Het cassettesysteem van het OS is op een laag niveau geïmplementeerd. Dat houdt in dat de OS-routines die de communicatie met de cassetterecorder verzorgen gegevens per byte wegschrijven, en bijvoorbeeld geen gebruik maken van filebuffers.
Wanneer een machinetaalprogramma gebruik maakt van het cassettesysteem, moet het programma er zelf zorg voor dragen dat opvolgende bytes op de cassette tijdig worden uitgelezen omdat de recorder dóórloopt en de gegevens anders de leeskop al gepasseerd hebben. Ook moet bij het schrijven van gegevens niet te lang worden gewacht tussen twee opvolgende bytes.

Het cassettesysteem kent lees- en schrijfsessies.
Een sessie bestaat uit een initialisatiedeel, een invoer- of uitvoerdeel en een afslui-

tend deel. Deze delen dienen direct na elkaar te worden uitgevoerd, aangezien de routines die de delen vertegenwoordigen zo geschreven zijn dat dat nodig is.
Voor elk deel van beide soorten sessies is een OS-ingang aanwezig. Deze ingangen zijn:

Voor een leessessie:

00E1	TAPION	leest kopblok van cassette
00E4	TAPIN	leest byte van cassette
00E7	TAPIOF	beëindigt lezen van cassette

Voor een schrijfsessie

00EA	TAPOON	schrijft kopblok naar cassette
00ED	TAPOUT	schrijft byte naar cassette
00F0	TAPOOF	beëindigt schrijven naar cassette

Tenslotte is er een OS-ingang voor het aansturen van alleen de motor van de cassetterecorder (via de remote control):

00F3	STMOTR	start of stopt motor cassetterecorder

Voor meer details omtrent deze OS-routines zie par. 6.3.

In het voorgaande wordt gesproken over een kopblok. Dit is in feite een lang aangehouden reeks pulsen met een en dezelfde frequentie, die dienen om bij het inlezen de computer te synchroniseren met de cassette. Aan de hand van de breedte van de pulsen golflengte wordt namelijk de snelheid vastgesteld waarin de bytes naar cassette zijn opgeschreven zodat ze met dezelfde snelheid kunnen worden ingelezen.
Bij het inlezen van het kopblok wordt aan de RAM-geheugenplaatsen LOWLIM (FCA4) en WINWID (FCA5) een waarde toegekend. Deze waarden worden bij het inlezen van de bytes van de cassette gebruikt.
De snelheden waarmee bytes op cassette kunnen worden geschreven en die door het OS worden ondersteund, zijn 1200 baud (bits/sec) en 2400 baud. De gegevens die nodig zijn voor het schrijven met een van die snelheden staan in CS120 (F3FC). Afhankelijk van de huidig ingestelde schrijfsnelheid wordt een gedeelte van dit gebied gekopieerd naar de geheugenplaatsen LOW (F406), HIGH (F408) en HEADER (F40A).

Meer gedetailleerde informatie over de manier waarop bits op cassette zijn gecodeerd staat in par. 5.4. Meer informatie over de gebruikte geheugenplaatsen is te vinden in par. 6.4.1.

6.2.6 Het toetsenbord

Het gedeelte van het OS dat zorgt voor het toetsenbord is geheel interrupt-gestuurd. Als zodanig maakt het eigenlijk deel uit van het interrupt-systeem (par. 6.2.2). Omdat het toetsenbord echter een belangrijk onderdeel is van de computer en omdat de manier waarop het toetsenbord door het OS wordt behandeld enige aandacht behoeft, is er een paragraaf aan gewijd.

In de VDP-interrupt wordt een teller bijgehouden, SCNCNT (F3F6), die terugloopt

van 3 naar 0. Zodra 0 is bereikt, wordt de teller weer op 3 gezet en wordt het toetsenbord gescand.
Het scannen van het toetsenbord verloopt volgens het principe dat in par. 5.2 is beschreven. Het resultaat van het scanproces komt te staan in NEWKEY (vanaf FBE5). Eerst wordt echter de oude waarde van NEWKEY overgeheveld naar OLDKEY (vanaf FBDA). Zolang de waarde van NEWKEY gelijk blijft aan de waarde van OLDKEY terwijl er minstens één toets is ingedrukt, wordt REPCNT (F3F7) verminderd. Zodra NEWKEY en OLDKEY niet meer gelijk zijn wordt REPCNT op 13 gezet.
Als er in NEWKEY een verandering ten opzicht van OLDKEY plaatsvindt omdat er een nieuwe toets is ingedrukt, leidt die nieuwe toets (of nieuwe toetsen) tot het invoeren van een teken in de toetsenbordbuffer. Als REPCNT tot 0 is afgenomen, gaat de toetsaanslag zich herhalen, en leidt elke ingedrukte toets tot een teken in de buffer.

Het verband tussen de ingedrukte toets of toetscombinatie is vastgelegd in afbeelding 5.2. Deze tabel geldt alleen voor toetsenborden met een internationale indeling; zie de ID-bytes in 002B-002C, par. 6.3.

De dode toets
Speciale vermelding behoeft de zogenaamde 'dode toets' (Engels: dead key). Dit is de toets op het toetsenbord die geen opschrift heeft. Het indrukken van deze toets resulteert niet in een ASCII-code maar wordt geregistreerd in KANAST (FCAC). Deze registratie wordt geraadpleegd bij elke geldige toetsaanslag om te kijken of het teken dat door die aanslag gegenereerd wordt misschien gemodificeerd moet worden.
In de internationale versie van het toetsenbord wordt de 'dode toets'-registratie gebruikt om geaccentueerde versies van de klinkers gemakkelijk te kunnen invoeren. Van alle klinkers zijn namelijk in de ASCII-tabel versies opgenomen met vier soorten accenten: accent grave, accent aigu, accent circumflex en trema (umlaut). Deze versies zijn normaal gesproken alleen te gebruiken door een of andere CODE- of SHIFT-toetscombinatie. Met gebruikmaking van de dode toets is het echter eenvoudig om de accenten toe te passen.
De gebruiker dient eerst de dode toets in te drukken, en vervolgens de klinker die een accent moet krijgen. Het indrukken van de dode toets kan op vier verschillende manieren gebeuren, om vier verschillende accenten te krijgen. De volgende tabel geeft aan wat het verband is tussen de manier van indrukken van de dode toets, de registratie in KANAST en het resulterende accent.

indruk	*KANAST*	*accent*
normaal	1	accent grave
SHIFT	2	accent aigu
CODE	3	accent circumflex
SHIFT + CODE	4	trema

Een teken dat via het toetsenbord is ingevoerd, komt in de toetsenbordbuffer te staan. Deze bevindt zich in KEYBUF (vanaf FBF0). In PUTPNT (F3F8) en GETPNT (F3FA) zijn twee wijzers opgeslagen die in de buffer de plaats aanwijzen waar een nieuwe code komt te staan respectievelijk waar de eerste (oudste) code moet worden uitgelezen.
Er zijn drie OS-ingangen die toegang geven tot de toetsenbordbuffer. Dit zijn:

009C	CHSNS	test of er iets in de buffer staat
009F	CHGET	haalt een code uit de buffer
0156	KILBUF	maakt de buffer leeg

6.2.7 De scherm-editor

Onder het systeem van de scherm-editor vallen de routines die in een tekstmodus de uitvoer naar het scherm verzorgen en via het scherm invoer van het toetsenbord lezen. Het is de taak van de scherm-editor om zaken als schermgrenzen in de gaten te houden, om besturingstekens op de juiste manier te verwerken en om invoer uit de toetsenbordbuffer op een zinvolle manier te verwerken.

Schermuitvoer

Alle schermuitvoer wordt verzorgd door de OS-routine CHPUT (ingang 00A2, zie par. 6.3). We bespreken hier globaal de speciale eigenschappen van deze routine.

Alle uitvoer komt terecht in het gedeelte van het videogeheugen dat begint op het adres in NAMBAS (F922). Dit adres wordt bij wisseling van schermmodus altijd gelijk gemaakt aan het beginadres van de naamtabel.

De schermregels worden begrensd door twee waarden: het aantal tekens dat op een regel past, bewaard in LINLEN (F3B0), en het aantal regels van het scherm, bewaard in CRTCNT (F3B1).
Het zij opgemerkt dat deze begrenzingen niet de fysische grenzen van het scherm zijn; deze zijn onveranderlijk. De fysische grenzen zijn vastgelegd in par. 3.2.1. De schermgrenzen worden alleen in stand gehouden door de scherm-editor.

Elk uitgevoerd teken verschijnt op een bepaalde plaats op het scherm. Deze plaats wordt de cursor genoemd. Nadat op de plaats van de cursor een teken is gezet, wordt de cursor normaal gesproken een positie naar rechts verplaatst, zodat het volgende teken rechts van het vorige verschijnt.
Soms is de cursor tijdens uitvoer op het scherm zichtbaar als een speciaal teken; zie verderop.

Als bij uitvoer naar het scherm de cursor het laatste teken op een schermregel passeert, wordt de cursor verplaatst naar het begin van de volgende regel. Als de onderste schermregel wordt gepasseerd, schuift alle tekst op het scherm een regel naar boven, zodat de onderste schermregel vrijkomt. Dit laatste wordt het 'scrollen' van het scherm genoemd.

Besturingstekens (tekens met een ASCII-code kleiner dan &H20) die naar het

scherm 'uitgevoerd' worden, hebben een speciale betekenis. Veel besturingstekens verplaatsen de cursor zonder verder iets op het scherm te zetten.

De routine CHPUT gebruikt de volgende geheugenplaatsen:

F3B0	LINLEN	aantal tekens op een regel
F3B1	CRTCNT	aantal regels van het scherm
F3B2	CLMLST	aantal tekens tussen tabulatorstops
F3DC	CSRY	regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	kolomnummer cursor (linker kolom nr. 1)
F922	NAMBAS	beginadres naamtabel

De cursor

Het meest opmerkelijke element van de scherm-editor is de cursor. Op de plaats waar de cursor staat, worden tekens toegevoegd of verwijderd. De cursor kan verplaatst worden door het uitvoeren van speciale besturingstekens, zodat overal op het scherm uitvoer kan verschijnen.
De cursor kan al dan niet zichtbaar zijn. Als de cursor wel zichtbaar is, wordt op de plaats van de cursor het patroon dat daar normaal gesproken staat, geïnverteerd; dat wil zeggen dat alle patroonpunten die eerst aan waren, uitgezet worden en vice versa.

Als er een teken op het scherm wordt gezet, komt dat teken te staan op de plaats waar eerst de cursor stond, en wordt de cursor naar rechts verplaatst.
De cursor wordt vertegenwoordigd door ASCII-code &HFF. Deze code staat in het element van de naamtabel dat hoort bij de positie van de cursor. De code van het teken dat 'onder' de cursor staat – dus dat op dat moment vervangen wordt door de cursor – wordt bewaard in CODSAV (FBCC).
Het patroon van de cursor – dus het element in de patroontabel met elementnummer &HFF – wordt bij het verplaatsen van de cursor bepaald aan de hand van het huidige patroon op de nieuwe schermpositie. Dit patroon wordt gecomplementeerd, en in het patroonelement &HFF gezet.
De cursor kan op twee manieren verzet worden: door het gebruik van besturingscodes of met behulp van de routine POSIT (00C6). Het regel- en kolomnummer van de huidige positie van de cursor zijn te vinden in CSRY (F3DC) en CSRX (F3DD).

Invoer via het tekstscherm

Een belangrijke functie van het tekstscherm is om een eenvoudige manier van invoer mogelijk te maken. Bij invoer via het tekstscherm worden alle tekens die op het toetsenbord zijn ingetypt direct op het scherm gezet, zodat er meteen correcties in kunnen worden aangebracht.

Dit invoermechanisme bepaalt in feite het beeld dat men van de computer heeft, aangezien het altijd tijdens het intypen of wijzigen van een programma of tijdens het invoeren van gegevens wordt gebruikt.
De tekens die op het toetsenbord worden ingetypt, worden door het interrupt-

systeem (zie par. 6.2.2) in de toetsenbordbuffer gezet, en worden door de routine CHGET (ingang 009F, zie par. 6.3) daar weer uit gelezen. Vervolgens worden deze tekens door middel van de routine CHPUT (ingang 00A2, zie boven) op het scherm gezet, met uitzondering van enkele besturingscodes, die binnen de invoerroutine een andere betekenis krijgen.

Het invoermechanisme levert als resultaat de tekst op van een logische regel, namelijk de logische regel waar de cursor stond bij het beëindigen van de invoerroutine.

Logische regels

De werking van de scherm-editor is gebaseerd op het begrip logische regel, als tegengestelde van schermregel. Een logische regel beslaat een of meer schermregels. De tekst van de logische regel wordt beschouwd als één geheel.

Wanneer tekens in een schermregel worden tussengevoegd of verwijderd, schuift de hele logische regel waarvan die schermregel deel uitmaakt op. Er zijn besturingscodes om de logische regel in zijn geheel te verwijderen en om naar het einde van een logische regel te springen.
In de beginsituatie van het scherm, en elke keer na het schoonvegen van het scherm, beslaan alle logische regels één schermregel. De tekst van de logische regel loopt van de eerste kolom tot het eerste teken van de schermregel dat geen spatie is.

Wanneer er over de grens van een schermregel heen wordt geschreven – dus wanneer er een teken in de laatste kolom van de schermregel wordt gezet – behoort daarna de volgende schermregel tot dezelfde logische regel. Als op dat moment het tussenvoegen binnen de scherm-editor is ingeschakeld, wordt er voor die volgende regel eerst ruimte gemaakt door een hele regel tussen te voegen (door gebruik van de escape-sequence 1B+4C, zie CHPUT).
Wanneer het scherm omhoog scrollt (door het geven van code 0A terwijl de cursor op de onderste schermregel staat) komt er onderaan het scherm een nieuwe logische regel, die op dat moment maar één schermregel beslaat.
De administratie van de logische regels wordt bijgehouden in een gebied van het RAM-geheugen, namelijk LINTTB (vanaf FBB2). Dit is een tabel waarin per schermregel een element is opgenomen waarin staat of de logische regel van die schermregel overloopt in de volgende schermregel.

Besturingscodes

Zoals gezegd worden de tekens en codes die door CHGET uit de toetsenbordbuffer worden gehaald door de scherm-editor verwerkt. Sommige tekens worden niet op het scherm gezet, maar leiden tot speciale acties. De ASCII-codes van deze tekens duiden we aan met besturingscodes.

De tekens worden verwerkt door de routine CHPUT (00A2). Deze routine zet tekens op het scherm en reageert, op bepaalde besturingscodes. De besturingscodes waarop CHPUT reageert zijn in de volgende tabel opgesomd. De codes zijn aangeduid door middel van hun hexadecimale representatie.

01 grafische extensie-kopbyte
07 piepje door aanroep van BEEP (00C0)
08 BS: cursor naar links, niet naar vorige regel
09 TAB: tabulatorsprong
0A cursor een regel naar beneden, evt. met scroll
0B HOME: cursor naar linker bovenhoek
0C CLS: scherm schoon en cursor naar linksboven
0D RETURN: cursor naar begin volgende regel
1B ESC: begin escape-sequence
1C cursor 1 positie naar rechts
1D cursor 1 positie naar links
1E cursor 1 positie naar boven
1F cursor 1 positie naar beneden

Meer informatie over de werking van CHPUT is te vinden in par. 6.3. Waar in voorgaande tabel de naam van een toets is genoemd, leidt het indrukken van die toets tot de betreffende code.
De nu volgende tabel geeft aan op welke manier het invoermechanisme reageert op besturingscodes. Deze reactie is aangegeven voor die codes die in het mechanisme een andere betekenis hebben dan voor de routine CHPUT. De tekens zijn gegeven door middel van de hexadecimale representatie van de code.

02 verzet cursor naar begin van dit woord
03 beëindigt scherm-editor
04 wordt onderdrukt
05 verwijdert tekst vanaf cursor tot regeleinde
06 verzet cursor naar begin volgend woord
08 BS: verwijdert teken direct voor cursor
0D RETURN: beëindigt scherm-editor
0E verzet cursor naar het einde van logische regel
12 INS: schakelt om tussen wel/niet tussenvoegen
15 verwijdert hele logische regel
1B ESC: wordt onderdrukt
7F DEL: verwijdert teken op cursorpositie

Het 'woord' waarvan bij code 02 en 06 wordt gesproken, is elke serie letters en cijfers of elk speciale teken.

Code 03 kan worden ingevoerd door CTRL+STOP of CTRL+C. De scherm-editor wordt erdoor uitgeschakeld, en er wordt geen invoer bepaald (zie verderop code 0D).

Code 04 is de STOP-toets. Deze wordt nooit naar CHPUT gestuurd. De uitvoering van de scherm-editor wordt dus nooit tijdelijk gestopt.

Code 08 voert na het verplaatsen van de cursor naar links, dat door CHPUT wordt verzorgd, nog de actie van code 7F uit (DEL).

Code 0D schakelt de scherm-editor uit nadat de cursor door CHPUT verplaatst is. Dit uitschakelen houdt in dat de tekst van de logische regel waarop de cursor staat in een buffer wordt gezet, namelijk BUF (F55E). Vervolgens wordt nog code 0A gegeven. Op het begrip logische regel komen we nog terug.

Code 12 is de INS-toets. Bij deze code wordt het tussenvoegen van de ingetypte tekst in- en weer uitgeschakeld. Na het inschakelen van het tussenvoegen wordt voor elk ingetypte teken eerst ruimte gemaakt binnen de logische regel, en wordt de logische regel uitgebreid indien nodig. Tevens wordt de vorm van de cursor veranderd in een streep in plaats van een blok. Bij het nogmaals geven van code 12 worden deze veranderingen weer ongedaan gemaakt.

Code 1B (de ESC-toets) wordt onderdrukt. Het indrukken van ESC heeft dus geen effect. Er kunnen geen escape-sequences via het toetsenbord worden ingevoerd.

Code 7F (de DEL-toets) vernietigt een teken van de tekst, namelijk het teken waarop de cursor zich bevindt. Het vernietigen heeft tot gevolg dat de overige tekst van de logische regel opschuift, zodat er geen ruimte meer overblijft waar het teken stond.

De scherm-editor wordt uitgeschakeld door code 03 of code 0D. Het effect van beide methoden van uitschakelen is verschillend. Code 03 (CTRL+STOP of CTRL+C) is een vorm van geforceerd uitschakelen, waarbij er geen resultaat door de editor wordt afgeleverd. Bij uitschakeling door code 0D (RETURN) wordt er echter een tekst opgeleverd, namelijk de tekst van de logische regel waarop de cursor stond ten tijde van het geven van de code. Deze tekst staat bij het verlaten van de routine in BUF (vanaf F55E).

De volgende routines hebben op de scherm-editor betrekking:

009F	CHGET	leest een code uit de toetsenbordbuffer
00A2	CHPUT	verwerkt ASCII-tekens en besturingscodes
00AE	PINLIN	versie van scherm-editor in directe modus
00B1	INLIN	hoofdroutine van editor
00B4	QINLIN	zet '?' op het scherm en roept INLIN aan
00C0	BEEP	geeft piepje
00C3	CLS	maakt scherm schoon
00C6	POSIT	verplaatst cursor

6.2.8 Het grafische systeem

Het grafische systeem van het MSX-systeem kleurt punten op het scherm, bepaalt de huidige kleur van een punt of verplaatst de grafische cursor in de gewenste richting of naar de gewenste plaats.

Het grafische systeem is alleen werkzaam in een grafische modus. De werking verschilt enigszins in modus 2 en 3. Het verschil is gelegen in het verschil in afmetingen van de kleinst adresseerbare eenheid in een grafische modus. In modus 2 is deze

kleinst adresseerbare eenheid een pixel; in modus 3 is het een blok van 4×4 pixels. Zo'n blok zullen we verder aanduiden met een modus 3-blok.

De grafische cursor

Binnen het grafische systeem is er sprake van een coördinatenpaar, ook wel de grafische cursor genaamd, dat altijd een plaats op het scherm aanwijst. De plaats waar de grafische cursor naar verwijst is de plaats waar grafische acties uitgevoerd worden. De grafische cursor vervult zo de rol die de 'gewone' cursor vervulde in de scherm-editor.

De grafische cursor is altijd in twee vormen aanwezig. Extern, zichtbaar op het scherm, heeft de cursor een x- en een y-coördinaat die samen het punt aanwijzen waar de cursor staat. Intern, op het niveau van de patroon- en kleurentabel wordt elk punt gerepresenteerd binnen een element van de patroontabel, en in modus 2 ook binnen een element van de kleurentabel.
De interne codering van de grafische cursor is een adres en een bitmasker. Het adres geeft het byte in de patroontabel aan waar de informatie over het bedoelde pixel staat; het bitmasker selecteert uit dat byte precies de bits die betrekking hebben op het pixel.

In modus 3 is het aantal punten in horizontale en verticale richting anders dan in modus 2. De coördinaatwaarden die worden gebruikt, hebben desondanks hetzelfde bereik. In de routine SCALXY (ingang 010E) worden de coördinaten van een punt in modus 3 omgezet naar modus 3-bloknummers. Alle overige OS-routines gaan uit van modus 3-blokken in plaats van coördinaten. Routines die de grafische cursor een punt naar rechts bewegen, bewegen in modus 3 de cursor een modus 3-blok naar rechts, etcetera.

Het verband tussen de waarde van de coördinaten en het adres en bitmasker van de grafische cursor is als volgt: als we de bits van de x-coördinaat aanduiden met X7-X0 en de bits van de y-coördinaat met Y7-Y0, dan is:

in modus 2:

CLOC = Y7–Y6–Y5–Y4–Y3–X7–X6–X5–X4–X3–Y2–Y1–Y0

X2	*X1*	*X0*	*CMASK*
0	0	0	00000001
0	0	1	00000010
0	1	0	00000100
0	1	1	00001000
1	0	0	00010000
1	0	1	00100000
1	1	0	01000000
1	1	1	10000000

in modus 3:

CLOC = Y7–Y6–Y5–X7–X6–X5–X4–X3–Y4–Y3–Y2

X2	*CMASK*
0	11110000
1	00001111

De coördinaten van de grafische cursor worden bewaard in de geheugenplaatsen GXPOS (FCB3) en GYPOS (FCB5). Het adres en bitmasker van de interne representatie staan in CLOC (F92A) resp. CMASK (F92C).

Kleuren

In het grafische systeem worden altijd drie standaardkleuren bewaard: de voorgrond-, achtergrond- en randkleur. Alleen de randkleur, bewaard in BDRCLR (F3EB) wordt vanzelf gebruikt, namelijk waar de kleur 'transparant' (code 0) is gebruikt, en in het gebied buiten het grafische schermdeel. De voor- en achtergrondkleur worden wel opgeslagen maar door het OS niet vanzelf bij grafische acties gebruikt.

De kleur waarmee een grafische actie uitgevoerd wordt, is altijd gelijk aan de werkkleur, bewaard in ATRBYT (F3F2). Dit hoeft niet dezelfde kleur te zijn als de voorgrondkleur, bewaard in FORCLR (F3E9). Alleen bij het wisselen van een grafische modus door de routine CHGMOD (ingang 005F) en bij het uitvoeren van de routine CHGCLR (ingang 0062) worden deze kleuren aan elkaar gelijk gemaakt.

Bij het 'tekenen' van een ASCII-teken op het grafische scherm door de routine GRPPRT (ingang 008D) wordt het teken in de voorgrondkleur (FORCLR) op het scherm gezet.
Bij inkleuracties wordt nog gebruik gemaakt van een andere geheugenplaats: BDRATR (FCB2), waarin de kleur wordt bewaard die gebruikt wordt voor het inkleuren.
De achtergrondkleur, bewaard in BAKCLR (F3EA), wordt alleen gebruikt in de routine CHGMOD. Deze routine maakt de kleur van het hele grafische schermdeel gelijk aan BAKCLR.

Routines

In de volgende tabel staan de routines die betrekking hebben op het grafische systeem. Meer informatie over deze routines is te vinden in par. 6.3.

0062	CHGCLR	verandert standaardkleuren
008D	GRPPRT	zet ASCII-teken op grafisch scherm
00FC	RIGHTC	grafische cursor een punt naar rechts
00FF	LEFTC	grafische cursor een punt naar links
0102	UPC	grafische cursor een lijn naar boven
0105	TUPC	cursor naar boven en test grenzen
0108	DOWNC	grafische cursor een lijn naar beneden

010B	TDOWNC	cursor naar beneden en test grenzen
010E	SCALXY	omzetten coördinaten en test grenzen
0111	MAPXY	omzetten coördinaten naar adres en masker
0114	FETCHC	ophalen adres en masker grafische cursor
0117	STOREC	wegzetten adres en masker grafische cursor
011A	SETATR	werkkleur een nieuwe waarde geven
011D	READC	kleur van huidige punt lezen
0120	SETC	huidige punt een nieuwe kleur geven
0123	NSETCX	horizontale rij punten inkleuren
012C	SCANR	onderzoekt een rij punten naar rechts
012F	SCANL	onderzoekt een rij punten naar links

6.3 OS-ingangen

De OS-ROM begint met een aantal adressen die aangeroepen kunnen worden om OS-routines uit te voeren. Op deze adressen staat een spronginstructie naar de betreffende routine.
Deze adressen zijn officieel gedocumenteerd als OS-ingangen. Dat wil zeggen dat de garantie bestaat dat bij wijzigingen in het OS of het MSX-systeem deze adressen behouden blijven. Door het OS via deze routines aan te roepen is het zeker dat een programma ook werkt op latere versies of andere merken van de MSX-computer.

In deze paragraaf worden alle OS-ingangen genoemd. Bij elke ingang is vermeld wat de ingangs- en uitgangsvoorwaarden zijn, en wat de routine doet waartoe dit een ingang is.
De ingangen zijn onder te verdelen in verschillende groepen. Deze onderverdeling is niet altijd even strikt aangehouden. Toch zullen we in deze paragraaf de categorieën van de onderverdeling noemen.

RST- en andere routines
Sommige van de volgende routines kunnen aangeroepen worden door een RST- in plaats van door een CALL-instructie. Omdat een RST korter en sneller is dan een CALL zijn een paar van de ingangen in deze categorie gereserveerd voor de BASIC-interpreter in plaats van het OS.

CHKRAM
Adres: 0000
Invoer: geen

Deze routine loopt de gleuven af, op zoek naar RAM-pagina's in het adresbereik 8000-FFFF. Het grootste aaneensluitende blok RAM-geheugen wordt geselecteerd als 'permanent' geheugen waarin programma's, en het werkgebied van BASIC en OS worden opgeslagen.
Tevens worden de geïnstalleerde pagina's in het adresbereik 4000-BFFF onderzocht op de vraag of ze ROM-pagina's zijn die een apparaatnaam-uitbreiding, een state-

ment-uitbreiding, een machinetaalprogramma of een BASIC-programma bevatten; zie par. 6.2.1.
Nadat deze onderzoekingen zijn gedaan, worden de initialisatieroutines van de gevonden ROM-pagina's uitgevoerd. Ten slotte wordt de BASIC-interpreter aangeroepen.
De routine CHKRAM kan worden aangeroepen met RST 0.

CGTABL
Adres: 0004

CGTABL is geen OS-ingang, maar een adres dat ten behoeve van programmeurs is toegevoegd. CGTABL bevat het adres waar de tabel met de codering van de patronen van de ASCII-tekens begint. Dit adres wordt bij initialisatie gebruikt om de RAM-plaats CGPNT (F91F) van de juiste waarde te voorzien.

VDP.DR
Adres: 0006

Dit is geen OS-ingang, maar een poortadres dat ten behoeve van de programmeur is toegevoegd. VDP.DR bevat het poortadres van de I/O-poort die dient om waarden uit het videogeheugen te lezen. Dit adres is in het huidige systeem 98. Mocht dit adres veranderen dan staat de juiste waarde in VDP.DR.

VDP.DW
Adres: 0007

Dit is geen OS-ingang, maar een poortadres dat ten behoeve van de programmeur is toegevoegd. VDP.DW bevat het poortadres van de I/O-poort die dient om waarden naar het videogeheugen te schrijven. Dit adres is in het huidige systeem 98. Mocht dit adres veranderen dan staat de juiste waarde in VDP.DW.

SYNCHR
Adres: 0008
Invoer: HL
Uitvoer: A, C-vlag, Z-vlag, HL
Verandert: Af, HL

Deze ingang hoort bij de BASIC-interpreter. De routine wordt gebruikt om te testen of het volgende teken of code in het BASIC-programma gelijk is aan het verwachte teken of code.

HL wijst bij aanroep van de routine naar het eerstvolgende teken in het BASIC-programma. Het verwachte teken is het teken dat achter de RST-instructie staat waarmee de routine is aangeroepen. Dit teken wordt gevonden door het terugkeeradres op de stapel te bestuderen. Dit adres wijst naar het verwachte teken.

Als de test niet slaagt, wordt een 'Syntax error' gegeven; anders wordt het terugkeeradres met één opgehoogd, terug op de stapel gezet en wordt de routine CHRGTR (0010) uitgevoerd. Bij het verlaten van de routine wijst HL naar het volgende teken van het BASIC-programma, A bevat het zojuist gelezen teken of code, de C-vlag is hoog als de zojuist gelezen code een getalconstante was, de Z-vlag is hoog als het einde van een statement is bereikt.

De routine SYNCHR kan worden aangeroepen door RST 8.

RDSLT

Adres:	000C
Invoer:	A, HL
Uitvoer:	A
Verandert:	AF, BC, DE

Deze routine leest de inhoud van een geheugenplaats van een willekeurige gleuf zonder permanente verandering van gleuf-selectie.

Bij aanroep van de routine staat in HL het adres van de geheugenplaats die gelezen moet worden. De gleuf waarvan dat adres gelezen moet worden, wordt aangeduid door de waarde van A. De bits van A hebben de volgende betekenis:

Bit 7	gleuf is wel/niet uitgebreid	1 = wel
Bit 6-4	ongebruikt	
Bit 3-2	nummer van secundaire gleuf	bereik: 0-3
Bit 1-0	nummer van primaire gleuf	bereik: 0-3

Het nummer van de secundaire gleuf (bits 3-2) is alleen geldig als bit 7 hoog is, ten teken dat er een secundair gleufnummer nodig is.
Bij terugkeer uit de routine is het gleufsysteem weer net zo ingesteld als bij aanroep. Interrupts worden binnen deze routine uitgeschakeld, maar niet weer ingeschakeld.

CHRGTR

Adres:	0010
Invoer:	HL
Uitvoer:	A, HL, C-vlag, Z-vlag
Verandert:	AF, HL

Deze routine leest het volgende teken of code van het BASIC-programma dat op dit moment uitgevoerd wordt.
Bij aanroep van de routine staat het adres van het te lezen teken in HL. Bij verlaten van de routine wijst HL naar het eerstvolgende teken en bevat A het zojuist gelezen teken. De C-vlag is hoog als het een getalconstante betreft; de Z-vlag is hoog als het zojuist gelezen teken aangeeft dat het einde van een statement bereikt is.
De routine CHRGTR kan worden aangeroepen door RST &H10.

WRSLT

Adres: 0014
Invoer: A, E, HL
Uitvoer: geen
Verandert: AF, BC, D

Met behulp van deze routine kan aan een geheugenplaats in een willekeurige primaire of secundaire gleuf een nieuwe waarde worden toegekend.
Het adres van de geheugenplaats die van een nieuwe waarde moet worden voorzien staat bij aanroep van de routine in HL. De waarde in A geeft aan welke gleuf bedoeld wordt. De gleufaanduiding is op dezelfde wijze gecodeerd als bij RDSLT (000C). De waarde die aan de geheugenplaats moet worden toegekend staat in E.
Bij terugkeer uit de routine is het gleufsysteem weer net zo ingesteld als bij aanroep.
Interrupts worden binnen deze routine uitgeschakeld, maar niet weer ingeschakeld.

OUTDO

Adres: 0018
Invoer: A, PRTFLG (F416), RAWPRT (F418), PTRFIL (F864)
Uitvoer: geen
Verandert: geen

Deze routine schrijft het teken waarvan de ASCII-code in A staat naar een geselecteerde uitvoer.

Als het om file-uitvoer gaat (te zien aan de waarde van PTRFIL die ongelijk 0 is) wordt de BASIC-interpreter aangeroepen om de uitvoer te verzorgen.
Als het om uitvoer naar de printer gaat (te zien aan de waarde van PRTFLG) wordt het teken naar de printer gestuurd. Als RAWPRT=0 dan wordt hierbij de routine OUTDLP (014D) gebruikt; anders wordt de code in A direct verstuurd, met gebruikmaking van routine LPTOUT (00A5).
In alle andere gevallen wordt CHPUT (00A2) aangeroepen voor het teken in A.
Binnen OUTDO wordt als eerste actie de RAM-haak H.OUTD (FEE4) aangeroepen. De routine OUTDO kan worden aangeroepen door RST &H18.

CALSLT

Adres: 001C
Invoer: IX, IY, evt. andere registers
Uitvoer: geen
Verandert: geen

Deze routine voert een aanroep uit van een subroutine in een willekeurige gleuf, en verzorgt een ordelijke terugkeer.
Bij aanroep van CALSLT staat het adres dat aangeroepen moet worden in IX en een aanduiding van de gleuf waarin de routine staat in IY. De codering van de gleufaanduiding in IY is gelijk aan de codering in A bij RDSLT (000C).

Alle andere registers (behalve de alternatieve registers) zijn vrij voor parameteroverdracht naar de aan te roepen routine.
Bij terugkeer uit de routine is het gleufsysteem weer net zo ingesteld als bij aanroep.
Interrupts worden binnen deze routine uitgeschakeld, maar niet weer ingeschakeld.

DCOMPR

Adres:	0020
Invoer:	DE, HL
Uitvoer:	vlaggen
Verandert:	AF

Deze routine vergelijkt de waarden van HL en DE en zet de vlaggen aan de hand van die vergelijking. De routine wordt gebruikt om ruimte te besparen, aangezien het aanroepen van DCOMPR kan gebeuren met een RST-instructie die maar één byte kost.

De routine DCOMPR kan worden aangeroepen door RST &H20.

ENASLT

Adres:	0024
Invoer:	A, HL
Uitvoer:	geen
Verandert:	alle registers

Deze routine selecteert permanent een pagina van een willekeurige gleuf.
Het adres in HL wordt alleen gebruikt om vast te stellen om welke pagina het gaat: namelijk de pagina waar de waarde van HL in het adresbereik valt. De waarde van A duidt de gleuf aan die geselecteerd moet worden. De codering van deze aanduiding is gelijk aan de codering bij RDSLT (000C).
Binnen deze routine worden interrupts uitgeschakeld en niet weer ingeschakeld.

GETYPR

Adres:	0028
Invoer:	VALTYP (F663)
Uitvoer:	vlaggen
Verandert:	AF

Deze routine wordt gebruikt door de BASIC-interpreter. In GETYPR wordt het type van de laatst geëvalueerde expressie bepaald. Dit type is te vinden in VALTYP.

Bij het verlaten van de routine geven de vlaggen het type aan:

C-vlag:	*Z-vlag:*	*S-vlag:*	*type:*
laag			dubbele precisie
hoog	hoog	laag	string
hoog	laag	hoog	integer
hoog	laag	laag	enkele precisie

De routine GETYPR kan worden aangeroepen door RST &H28.

ID-bytes
Adres: 002B, 002C

Adressen 002B en 002C worden gebruikt om een aantal gegevens in op te slaan omtrent de versie of uitvoering van de MSX-computer.
De bits van deze identificatie-bytes (ID-bytes) hebben de volgende betekenis:

Adres 002C:

Bit 7	interruptfrequentie 50/60 Hz	1 = 50 Hz
Bit 6-4	formaat van de datum	bereik: 0-2
Bit 3-0	soort lettertype	bereik: 0-2

Het formaat van de datum slaat op de volgorde waarin dag, maand en jaar ingevoerd moeten worden. Mogelijke waarden hiervoor zijn:

0 Japans (jaar-maand-dag)
1 Amerikaans (maand-dag-jaar)
2 Europees (dag-maand-jaar)

Het soort lettertype heeft betrekking op de inhoud van de ASCII-tabel. Dit veld kan de volgende waarden hebben:

0 Japans
1 Internationaal (ASCII)
2 Koreaans

Adres 002C:

Bit 7-4	Versie van de BASIC	bereik: 0-1
Bit 3-0	Versie van het toetsenbord	bereik: 0-4

De versie van de BASIC kan de volgende waarden hebben:

0 Japans
1 Internationaal

De versie van het toetsenbord kan de volgende waarden hebben:

0 Japans
1 Internationaal (QWERTY)
2 Frans (AZERTY)
3 Engels
4 Duits (DIN)

De instelling van de ID-bytes is in Nederland als volgt:

adres	*instelling*	*waarde*
002B	1-001-0001	91
002C	0001-0001	11

CALLF

Adres:	0030
Invoer:	geen
Uitvoer:	vlaggen
Verandert:	AF

Deze routine biedt een manier om met gebruik van weinig ruimte een aanroep te doen van een machinetaalroutine in een willekeurige primaire of secundaire gleuf.

CALLF kan op de volgende manier aangeroepen worden:

RST	&H30
DB	gleuf van bestemming
DW	adres van bestemming

Dit kost alles bij elkaar vier bytes. Deze manier van aanroepen kan dan ook uitstekend worden gebruikt in RAM-haken: daar zijn vijf bytes beschikbaar, waarvan de eerste vier dan een aanroep met gebruikmaking van RST &H30 kunnen bevatten, en het vijfde byte een RET-instructie kan zijn.
De gleuf van bestemming kan gecodeerd worden op de manier zoals uit de doeken is gedaan bij RDSLT (000C). Aan de hand van het adres wordt bepaald voor welke pagina die bepaalde gleuf geselecteerd moet worden.

CALLF werkt via CALSLT (001C). Eerst worden de waarden van de bestemmingsgleuf en het -adres gelezen door het terugkeeradres van de stapel te halen en op te hogen; deze waarden worden in IY respectievelijk IX gezet. Het terugkeeradres wordt opgehoogd terug op de stapel gezet, zodat het wijst naar het eerste byte na het bestemmingsadres. Als dit alles gebeurd is, wordt CALSLT aangeroepen.
Bij terugkeer uit de routine is het gleufsysteem weer net zo ingesteld als bij aanroep.
Interrupts worden binnen deze routine uitgeschakeld, maar niet weer ingeschakeld.

KEYINT

Adres:	0038
Invoer:	geen
Uitvoer:	geen
Verandert:	geen

Adres 0038 is het adres dat automatisch wordt aangeroepen bij elke interrupt, aangezien interrupt-modus 1 is ingeschakeld. Voor gegevens over de werking van het interrupt-systeem zie par. 6.2.2.
Tijdens het uitvoeren van de routine worden alle registers op de stapel bewaard, en bij terugkeer worden de oude waarden teruggezet.
Binnen KEYINT wordt als eerste actie de RAM-haak H.KEYI (FD9A) aangeroepen. Verderop wordt de RAM-haak H.TIMI (FD9F) aangeroepen. De routine KEYINT kan worden aangeroepen door RST &H38.

Initialisatie-routine

De volgende twee routines dienen om een paar zaken van hun beginwaarde te voorzien. Deze routines worden in de regel maar één keer aangeroepen, namelijk bij de initialisatie na het aanzetten van de computer.

INITIO

Adres:	003B
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers

Deze routine verzorgt de initialisatie van I/O-apparatuur. Onder meer worden de PSG en de printer geïnitialiseerd. Met initialisatie van apparatuur worden alle acties verstaan die nodig zijn voordat er goed gebruik van de apparatuur kan worden gemaakt. Hieronder valt bijvoorbeeld de instelling van een begintoestand.
De initialisatie van apparatuur hoeft normaal gesproken maar één keer te gebeuren.

INIFNK

Adres:	003E
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers, FNKSTR (F87F)

Door het aanroepen van deze routine worden de definities van de functietoetsen op hun beginwaarde (terug)gezet.

Routines voor de VDP

De volgende routines vormen de basis van het videosysteem van het OS. Op deze basis zijn de scherm-editor (par. 6.2.7) en het grafische systeem (par. 6.2.8) gebouwd.
Deze basis omvat de routines die direct de VDP aanspreken, en de routines die de schermmodi initialiseren.

DISSCR

Adres:	0041
Invoer:	geen
Uitvoer:	geen
Verandert:	AF, BC

Deze routine maakt het beeld blank. Het beeldscherm krijgt in zijn geheel de randkleur (de kleur waarvan de code in bits 3-0 van VDP-register 7 staat).
Deze routine verandert niets in het videogeheugen. Het blank maken van het beeld gebeurt door bit 6 van VDP-register 1 laag te zetten. Zie par. 3.3.

ENASCR

Adres: 0044
Invoer: geen
Uitvoer: geen
Verandert: AF, BC

Deze routine doet het tegengestelde van DISSCR (0041): na het aanroepen van ENASCR wordt het beeld op het scherm weer aangestuurd door de inhoud van het videogeheugen. Het herstellen van het beeld gebeurt door het hoog zetten van bit 6 van VDP-register 1.

WRTVDP

Adres: 0047
Invoer: B, C
Uitvoer: geen
Verandert: AF, BC, RG0SAV (F3DF)-RG7SAV (F3E6)

Deze routine schrijft een nieuwe waarde naar een VDP-register. Bij aanroep van de routine staat de te schrijven waarde in B en het nummer van het register waarin de waarde terecht moet komen in C.
Behalve naar de VDP wordt de nieuwe waarde ook naar een van de adressen RG0SAV-RG7SAV geschreven. Dit gebeurt om de inhoud van die geheugenplaatsen gelijk te houden aan de waarden van de VDP-registers.

RDVRM

Adres: 004A
Invoer: HL
Uitvoer: A
Verandert: AF

Deze routine dient om de waarde van een geheugenplaats van het videogeheugen te lezen.
Bij aanroep van de routine staat in HL het adres van de geheugenplaats. Bij het verlaten van de routine bevat A de waarde van die geheugenplaats.
Tevens is bij het verlaten van de routine het leesadres van de VDP (zie par. 3.3.2) ingesteld op het eerste adres na HL.

WRTVRM

Adres: 004D
Invoer: A, HL
Uitvoer: geen
Verandert: AF

Deze routine dient om een geheugenplaats van het videogeheugen van een nieuwe waarde te voorzien.

Bij aanroep van de routine staat in HL het adres van de geheugenplaats, en in A de waarde die de geheugenplaats moet krijgen.
Bij het verlaten van de routine is het schrijfadres van de VDP (zie par. 3.3.2) ingesteld op het eerste adres na HL.

SETRD

Adres:	0050
Invoer:	HL
Uitvoer:	geen
Verandert:	AF

Deze routine zet het leesadres van de VDP op een bepaalde waarde. De waarde die het leesadres moet krijgen staat bij aanroep van de routine in HL.
Na het uitvoeren van deze routine kan door het herhaaldelijk lezen van een waarde uit VDP.DR (de poort met als poortadres de waarde van adres 0006) de waarde van de geheugenplaatsen vanaf adres HL worden bepaald.

SETWRT

Adres:	0053
Invoer:	HL
Uitvoer:	geen
Verandert:	AF

Deze routine zet het schrijfadres van de VDP op een bepaalde waarde. De waarde die het schrijfadres moet krijgen staat bij aanroep van de routine in HL.

Na het uitvoeren van deze routine kunnen door herhaaldelijk schrijven van een waarde naar VDP.DW (de poort met als poortadres de waarde van adres 0007) de geheugenplaatsen vanaf adres HL van een nieuwe waarde worden voorzien. Zie par. 3.3.2.

FILVRM

Adres:	0056
Invoer:	A, BC, HL
Uitvoer:	geen
Verandert:	AF, BC

Deze routine beschrijft een aaneengesloten stuk van het videogeheugen met één dezelfde waarde.
Bij aanroep van de routine staat in A de waarde waarmee het videogeheugen gevuld moet worden, in BC het aantal bytes van het videogeheugen die met die waarde gevuld moeten worden, en in HL het adres van de eerste geheugenplaats van het videogeheugen die op die manier gevuld moet worden.

LDIRMV

Adres: 0059
Invoer: BC, DE, HL
Uitvoer: geen
Verandert: alle registers

Deze routine kopieert een stuk van het videogeheugen naar het gewone geheugen. Bij aanroep van de routine staat in BC de lengte van het stuk dat gekopieerd moet worden, in DE het adres in het gewone geheugen van de geheugenplaats waar naar toe het eerste byte gekopieerd gaat worden en in HL het adres in het videogeheugen van de eerste geheugenplaats waarvan de waarde gekopieerd gaat worden.
Na terugkeer uit de routine is de waarde van het blok gewoon geheugen dat begint op DE, ter lengte van BC, gelijk aan de waarde van het blok videogeheugen met dezelfde lengte dat begint op HL.

LDIRVM

Adres: 005C
Invoer: BC, DE, HL
Uitvoer: geen
Verandert: alle registers

Deze routine kopieert een stuk van het gewone geheugen naar het videogeheugen.

Bij aanroep van de routine staat in BC de lengte van het stuk dat gekopieerd moet worden, in DE het adres in het videogeheugen van de geheugenplaats waar naar toe het eerste byte gekopieerd gaat worden en in HL het adres in het gewone geheugen van de eerste geheugenplaats waarvan de waarde gekopieerd gaat worden.
Na terugkeer uit de routine is de waarde van het blok videogeheugen dat begint op DE, ter lengte van BC, gelijk aan de waarde van het blok gewoon geheugen met dezelfde lengte dat begint op HL.

CHGMOD

Adres: 005F
Invoer: A
Uitvoer: geen
Verandert: alle registers, SCRMOD (FCAF), OLDSCR (FCB0)

Deze routine initialiseert het scherm op de modus in A. Dit houdt in dat het scherm opnieuw in die modus gezet wordt, zelfs als het op dit moment al in die modus staat. Het initialiseren van het scherm gebeurt met één van de routines INITXT (006C), INIT32 (006F), INIGRP (0072) of INIMLT (0075), afhankelijk van de waarde van A.

Bij het verlaten van de routine bevat SCRMOD de waarde van A. Als A op een tekstmodus duidt, bevat ook OLDSCR de waarde van A. Daarnaast zijn er, afhankelijk van de nieuwe modus, beginadressen van VDP-tabellen veranderd.

CHGCLR

Adres: 0062
Invoer: FORCLR (F3E9), BAKCLR (F3EA), BDRCLR (F3EB)
Uitvoer: geen
Verandert: alle registers

Deze routine verandert de standaardkleuren (voorgrond-, achtergrond- en randkleur) aan de hand van de waarden van respectievelijk FORCLR, BAKCLR en BDRCLR.
De veranderingen die aangebracht worden, zijn afhankelijk van de schermmodus. In modus 0 worden de waarden van FORCLR en BAKCLR in bits 7-4 respectievelijk bits 3-0 van register 7 van de VDP gezet. In alle andere modi wordt register 7 gelijk gemaakt aan de waarde van BDRCLR. In modus 1 wordt bovendien de hele kleurentabel gevuld met de waarden van FORCLR en BAKCLR.

NMI

Adres: 0066
Invoer: geen
Uitvoer: geen
Verandert: geen

Deze routine wordt aangeroepen bij het optreden van een NMI-interrupt. In het standaard MSX-systeem wordt deze interrupt niet gebruikt; er is alleen in voorzien door middel van de RAM-haak H.NMI (FDD6).
De routine NMI wordt afgesloten met een RETN-instructie in plaats van de gewone RET-instructie.

CLRSPR

Adres: 0069
Invoer: SCRMOD
Uitvoer: geen
Verandert: alle registers

Deze routine zet het sprite-systeem van de VDP terug op de begintoestand. Dit gebeurt door de sprite-patroontabel en de sprite-plaatstabel van hun beginwaarden te voorzien.
In de begintoestand van de sprite-patroontabel zijn alle patronen blank; dat wil zeggen dat de patroonpunten uitgezet zijn.

In de begintoestand van de sprite-plaatstabel voldoen alle elementen van de tabel (oftewel de sprite-vlakken) aan de volgende voorwaarden:
- de y-coördinaat van de sprite in het sprite-vlak is ingesteld op 209; dit is een waarde waarbij de sprite niet op het scherm getekend wordt.
- de kleur van de sprite is ingesteld op de voorgrondkleur (de waarde van FORCLR (F3E9)).

– het nummer van de sprite in het sprite-vlak is gelijk aan het vlaknummer (ofte wel: het elementnummer binnen de sprite-plaatstabel).

Voor meer informatie over het sprite-systeem en de tabellen zie par. 3.1.1.

INITXT

Adres:	006C
Invoer:	TXTNAM (F3B3)-TXTPAT (F3BB), FORCLR (F3E9)-BDRCLR (F3EB)
Uitvoer:	geen
Verandert:	alle registers, NAMBAS (F922), CGPBAS (F924), LINLEN (F3B0), SCRMOD (FCAF), OLDSCR (FCB0)

In deze routine wordt het scherm geïnitialiseerd op modus 0. Dit houdt in dat:
– de waarde van LINLEN gelijk wordt gemaakt aan LINL40 (F3AE).
– NAMBAS en CGPBAS worden gevuld met de waarden van TXTNAM resp. TXTCGP.
– de naam- en patroontabel in het videogeheugen in hun begintoestand gezet worden.
– de werkkleuren worden aangepast door aanroep van CHGCLR (0062).
– de VDP-registers worden voorzien van hun beginwaarden door aanroep van SETTXT (0078).

De begintoestand van de videotabellen en de VDP-registers voor modus 0 zijn te vinden in par 3.3.1.

Tijdens het initialiseren van het scherm wordt het beeld tijdelijk blank gemaakt met behulp van de routine DISSCR (0041).
SCRMOD en OLDSCR krijgen de waarde 0, ten teken dat het scherm ingesteld is op modus 0.

INIT32

Adres:	006F
Invoer:	T32NAM (F3BD)-T32PAT (F3C5), FORCLR (F3E9)-BDRCLR (F3EB)
Uitvoer:	geen
Verandert:	alle registers, NAMBAS (F922)-ATRBAS (F928), LINLEN (F3B0), SCRMOD (FCAF), OLDSCR (FCB0)

In deze routine wordt het scherm geïnitialiseerd op modus 1. Dit houdt in dat:
– de waarde van LINLEN gelijk wordt gemaakt aan LINL32 (F3AF).
– NAMBAS en CGPBAS worden gevuld met de waarden van T32NAM resp. T32CGP.
– de naam- en patroontabel in het videogeheugen in hun begintoestand gezet worden.
– het sprite-systeem in de begintoestand gezet wordt door aanroep van CLRSPR (0069).

- de werkkleuren worden aangepast door aanroep van CHGCLR (0062).
- de VDP-registers worden voorzien van hun beginwaarden door aanroep van SETT32 (007B).

De begintoestand van de videotabellen en de VDP-registers voor modus 1 zijn te vinden in par. 3.3.1.

Tijdens het initialiseren van het scherm wordt het beeld tijdelijk blank gemaakt met behulp van de routine DISSCR (0041).
SCRMOD en OLDSCR krijgen de waarde 1, ten teken dat het scherm ingesteld is op modus 1.

INIGRP

Adres:	0072
Invoer:	GRPNAM (F3C7)-GRPPAT (F3CF), FORCLR (F3E9)-BDRCLR (F3EB)
Uitvoer:	geen
Verandert:	alle registers, NAMBAS (F922)-ATRBAS (F928), SCRMOD (FCAF)

In deze routine wordt het scherm geïnitialiseerd op modus 0. Dit houdt in dat:
- NAMBAS en CGPBAS worden gevuld met de waarden van GRPNAM resp. GRPCGP.
- de naam- en patroontabel in het videogeheugen in hun begintoestand worden gezet.
- het sprite-systeem in de begintoestand gezet wordt door aanroep van CLRSPR (0069).
- de werkkleuren worden aangepast door aanroep van CHGCLR (0062).
- VDP-registers worden voorzien van hun beginwaarden door aanroep van SETGRP (007E).

De begintoestand van de videotabellen en de VDP-registers voor modus 2 zijn te vinden in par. 3.4.1.
Tijdens het initialiseren van het scherm wordt het beeld tijdelijk blank gemaakt met behulp van de routine DISSCR (0041).
SCRMOD krijgt de waarde 2, ten teken dat het scherm ingesteld is op modus 2.

INIMLT

Adres:	0075
Invoer:	MLTNAM (F3D1)-MLTPAT (F3D9), FORCLR (F3E9)-BDRCLR (F3EB)
Uitvoer:	geen
Verandert:	alle registers, NAMBAS (F922)-ATRBAS (F928), SCRMOD (FCAF)

In deze routine wordt het scherm geïnitialiseerd op modus 0. Dit houdt in dat:
- NAMBAS en CGPBAS worden gevuld met de waarden van MLTNAM resp. MLTCGP.
- de naam- en patroontabel in het videogeheugen in hun begintoestand worden gezet.
- het sprite-systeem in de begintoestand wordt gezet door aanroep van CLRSPR (0069).
- de werkkleuren worden aangepast door aanroep van CHGCLR (0062).
- de VDP-registers worden voorzien van hun beginwaarden door aanroep van SETMLT (0081).

De begintoestand van de videotabellen en de VDP-registers voor modus 3 zijn te vinden in par. 3.3.1.
Tijdens het initialiseren van het scherm wordt het beeld tijdelijk blank gemaakt met behulp van de routine DISSCR (0041).
SCRMOD krijgt de waarde 3, ten teken dat het scherm ingesteld is op modus 3.

SETTXT

Adres:	0078
Invoer:	TXTNAM (F3B3)-TXTPAT (F3BB)
Uitvoer:	geen
Verandert:	alle registers

Deze routine geeft de registers van de VDP de waarde die ze in schermmodus 0 moeten hebben. Registers 0 en 1 worden ingesteld volgens variatie 1 van het grafische systeem. De waarden die registers 2-6 krijgen (waarin de beginadressen van de VDP-tabellen staan), worden bepaald aan de hand van de waarden van TXTNAM-TXTPAT. Register 7 blijft onveranderd.

SETT32

Adres:	007B
Invoer:	T32NAM (F3BD)-T32PAT (F3C5)
Uitvoer:	geen
Verandert:	alle registers

Deze routine geeft de registers van de VDP de waarde die ze in schermmodus 1 moeten hebben. Registers 0 en 1 worden ingesteld volgens het basis-grafische systeem. De waarden die registers 2-6 krijgen (waarin de beginadressen van de VDP-tabellen staan), worden bepaald aan de hand van de waarden van T32NAM-T32PAT. Register 7 blijft onveranderd.

SETGRP

Adres:	007E
Invoer:	GRPNAM (F3C7)-GRPPAT (F3CF)
Uitvoer:	geen
Verandert:	alle registers

Deze routine geeft de registers van de VDP de waarde die ze in schermmodus 2 moeten hebben. Registers 0 en 1 worden ingesteld volgens variatie 2 van het grafische systeem. De waarden die registers 2-6 krijgen (waarin de beginadressen van de VDP-tabellen staan), worden bepaald aan de hand van de waarden van GRPNAM-GRPPAT. Register 7 blijft onveranderd.

SETMLT

Adres:	0081
Invoer:	MLTNAM (F3D1)-MLTPAT (F3D9)
Uitvoer:	geen
Verandert:	alle registers

Deze routine geeft de registers van de VDP de waarde die ze in schermmodus 3 moeten hebben. Registers 0 en 1 worden ingesteld volgens variatie 3 van het grafische systeem. De waarden die registers 2-6 krijgen (waarin de beginadressen van de VDP-tabellen staan), worden bepaald aan de hand van de waarden van MLTNAM-MLTPAT. Register 7 blijft onveranderd.

CALPAT

Adres:	0084
Invoer:	A
Uitvoer:	HL
Verandert:	AF, DE, HL

Deze routine berekent het beginadres in het videogeheugen van een sprite-patroon. Bij aanroep van de routine staat in A het nummer van een sprite. Dit nummer geldt als elementnummer binnen de sprite-patroontabel. Bij het verlaten van de routine staat in HL het beginadres (in het videogeheugen) van de codering van het patroon dat bij die sprite hoort, rekening houdend met het formaat (8×8 of 16×16) van de sprites.

CALATR

Adres:	0087
Invoer:	A
Uitvoer:	HL
Verandert:	AF, DE, HL

Deze routine berekent het beginadres in het videogeheugen van een sprite-vlak-descriptor.
Bij aanroep van de routine staat in A het nummer van een sprite-vlak. Dit nummer geldt als elementnummer binnen de sprite-plaatstabel. Bij het verlaten van de routine staat in HL het beginadres (in het videogeheugen) van de descriptor van dat sprite-vlak.

GRPSIZ

Adres: 008A
Invoer: geen
Uitvoer: A, C-vlag
Verandert: AF

Deze routine bepaalt het sprite-formaat waarop het sprite-systeem op dit moment is ingesteld. Bij het verlaten van de routine staat in A het aantal bytes van een sprite-patroon (8 voor een 8×8 sprite, 32 voor een 16×16 sprite); de C-vlag is hoog als het formaat 16×16 is, laag als het 8×8 is.

GRPPRT

Adres: 008D
Invoer: A, SCRMOD (FCAF)
Uitvoer: geen
Verandert: geen

Deze routine verzorgt het schrijven van een ASCII-teken in een grafische schermmodus. Het teken moet op het scherm 'getekend' worden, d.w.z. het patroon wordt puntje voor puntje op het scherm gezet.
Bij het aanroepen van de routine staat de ASCII-code van het teken in A. Codes &H00-&H1F stellen hierbij de grafische extensie-codes voor.

Routines voor de PSG

De volgende ingangen geven toegang tot routines die de PSG besturen; ofwel in directe vorm, door direct naar de PSG-registers te schrijven, ofwel via het geluidssysteem.

GICINI

Adres: 0090
Invoer: geen
Uitvoer: geen
Verandert: alle registers, MUSICF (FB3F)-VCBC (FB8B)

Deze routine initialiseert het geluidssysteem. Dit houdt in:
- de VCB's worden op hun beginwaarden gezet. De standaard waarden voor toonduur, octaafnummer etc. worden ingevuld.
- de geluids-queues worden geïnitialiseerd, d.w.z. de descriptoren van deze queues worden van hun beginwaarden voorzien.
- de PSG wordt stilgelegd.

WRTPSG

Adres: 0093
Invoer: A, E
Uitvoer: geen
Verandert: geen

Deze routine schrijft een waarde naar een register van de PSG. Bij aanroep van de routine staat in A het nummer van het register dat een nieuwe waarde moet krijgen, en in E de nieuwe waarde.

RDPSG

Adres:	0096
Invoer:	A
Uitvoer:	A
Verandert:	A

Deze routine leest de waarde van een PSG-register. Bij aanroep van de routine staat in A het nummer van het register dat gelezen moet worden; bij het verlaten van de routine staat in A de waarde die het register heeft.

STRTMS

Adres:	0099
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers, MUSICF (FB3F)

Deze routine test of het geluidssysteem iets te doen heeft, en start het geluidssysteem op als het niet werkzaam is.
Bij het opstarten moet in elke geluids-queue tenminste een EOS-commando staan; zie par. 6.2.4.

Routines voor de scherm-editor

De volgende routines worden door het OS gebruikt om de scherm-editor te implementeren. Voor gegevens omtrent de werking van de editor zie par. 6.2.7.

CHSNS

Adres:	009C
Invoer:	geen
Uitvoer:	Z-vlag
Verandert:	AF

Deze routine test de toetsenbordbuffer op aanwezigheid van enige tekens. Bij het verlaten van de routine is de Z-vlag hoog als er nog tekens in de buffer staan, anders laag.

CHGET

Adres:	009F
Invoer:	geen
Uitvoer:	A
Verandert:	AF

Deze routine leest een teken uit de toetsenbordbuffer. Als de buffer leeg is, wacht de

routine net zolang tot er iets in staat. Bij het verlaten van de routine staat in A de ASCII-code van het teken dat via de buffer ingevoerd is.
Het indrukken van de STOP-toets tijdens het uitvoeren van deze routine wordt onderdrukt.

CHPUT

Adres:	00A2
Invoer:	A
Uitvoer:	geen
Verandert:	geen

Deze routine schrijft een ASCII-code naar het scherm. Als het een besturingscode is (kleiner dan &H20), wordt de besturingsactie uitgevoerd die bij de code hoort. Als het geen besturingscode is, wordt het teken dat bij de code hoort op het scherm gezet, door de code op de juiste plaats in de naamtabel te zetten.
De volgende besturingscodes (in hexadecimale representatie opgegeven) worden door CHPUT herkend en leiden tot acties:

01	grafische extensie-kopbyte
07	piepje door aanroep van BEEP (00C0)
08	cursor naar links, niet naar vorige regel
09	tabulatorsprong
0A	cursor een regel naar beneden, evt. met scroll
0B	cursor naar linker bovenhoek
0C	scherm schoon en cursor naar linksboven
0D	cursor naar begin volgende regel
1B	begin escape-sequence
1C	cursor 1 positie naar rechts
1D	cursor 1 positie naar links
1E	cursor 1 positie naar boven
1F	cursor 1 positie naar beneden

Het verschil tussen codes 08 en 1D is, dat bij het uitvoeren van code 1D de cursor naar de laatste positie van de vorige regel gaat als hij in de eerste kolom staat. Het verschil tussen codes 0A en 1F is, dat bij het uitvoeren van code 0A het scherm naar boven schuift (scrollt) als de cursor op de onderste regel staat. Besturingscodes die niet op deze lijst voorkomen, worden door CHPUT genegeerd.

Codes 01 en 1B verwachten beide opvolgende codes, die het eigenlijke effect van deze codes bepalen. Code 01 verwacht een code tussen 40 en 5F (de ASCII-codes van '@' t/m '_'). Zo'n code resulteert dan in een teken uit het grafische extensie-deel van de ASCII-tabel van de MSX-computer. Als er een code volgt die buiten dit bereik valt, heeft code 01 geen effect. Code 1B (ESC) verwacht een escape-sequence. (Dit is een serie tekens die samen een speciaal effect bewerkstelligen. Sommige van deze effecten kunnen ook op andere manieren bereikt worden, andere zijn alleen via een escape-sequence te bereiken.)

De escape-sequences waarvoor een effect gedefinieerd is, zijn de volgende (codes zijn hexadecimaal gerepresenteerd):

41 cursor een regel omhoog
42 cursor een regel omlaag
43 cursor een positie naar rechts
44 cursor een positie naar links
45 scherm schoon en cursor naar links boven
48 cursor naar linker bovenhoek
4A verwijder tekst tot einde van scherm
4B verwijder tekst tot einde van regel
4C voeg een regel tussen
4D verwijder een regel
59 cursor naar opgegeven positie
6A scherm schoon en cursor naar links boven
78 schakel instellingen in
79 schakel instellingen uit

Code 4C voegt op het scherm een lege regel tussen die op de regel komt waar de cursor stond. Hierbij schuiven alle regels die onder de betreffende nieuwe regel stonden een positie omlaag. Door aan het begin van het scherm een regel tussen te voegen, kan men het scherm dus omlaag laten scrollen.

Code 4D doet het tegengestelde van 4C: er wordt een regel verwijderd. Alle tekst die op die regel stond, verdwijnt van het scherm. Regels die onder de betreffende regel op het scherm stonden, schuiven een positie naar boven. Op deze manier kan kunstmatig omhoog gescrolld worden.

Code 59 zet de cursor op een willekeurige positie op het scherm. Deze positie moet worden opgegeven in de twee codes die op 59 volgen. Eerst wordt het kolomnummer verwacht, vervolgens het regelnummer waar de cursor terecht moet komen. Met deze twee getallen in resp. H en L wordt POSIT (00C6) aangeroepen.

Codes 78 en 79 zijn elkaars complement. Beide verwachten nog een opvolgende code die aangeeft wat er in- of uitgeschakeld moet worden. Deze code kan twee waarden hebben (hex):

24 schakel CSTYLE (FCAA)=onderstreep-cursor in/uit
25 schakel INSFLG (FCA8)=tussenvoegen in/uit

Elke andere waarde wordt genegeerd. Het tussenvoegen wordt alleen gebruikt binnen de scherm-editor (Zie par. 6.2.7).

LPTOUT

Adres:	00A5
Invoer:	A
Uitvoer:	C-vlag
Verandert:	F

Deze routine stuurt een ASCII-code naar de printer, indien mogelijk. Bij aanroep van de routine staat de te versturen code in A. Bij het verlaten van de routine is de C-vlag hoog als het versturen mislukt is (omdat de printer niet klaar stond), anders laag.

LPTSST

Adres: 00A8
Invoer: geen
Uitvoer: A, Z-vlag
Verandert: AF

Deze routine test op het aangesloten zijn en klaar staan van de printer. Bij het verlaten van de routine staat 255 in A en is de Z-vlag laag als de printer klaar is voor uitvoer, anders is A=0 en de Z-vlag hoog.

CNVCHR

Adres: 00AB
Invoer: A
Uitvoer: A, C-vlag, Z-vlag
Verandert: AF, GRPHED (FCA6)

Bij herhaald aanroepen voor verschillende ASCII-codes test deze routine de opgegeven codes op het voorkomen van een grafische extensie (een extensie-kopbyte gevolgd door de ASCII-code van een grafische extensie).
Bij aanroep van de routine staat in A de te onderzoeken code. Bij het verlaten van de routine geven de C- en Z-vlag informatie over de code, en bevat A de bedoelde extensie-code als er sprake is van een grafische extensie.

Mogelijke waarden voor de C- en Z-vlag zijn:

C-vlag	*Z-vlag*	*betekenis*	*A bij verlaten*
laag		was extensie-kopbyte	1
hoog	laag	was gewone ASCII-code	ASCII-code
hoog	hoog	was grafische extensie	extensie-code

De geheugenplaats GRPHED geeft aan of de ASCII-code die bij de vorige aanroep door deze routine werd onderzocht, een extensie-kopbyte (ASCII-code 1) was.
De ASCII-codes van de grafische extensies zijn &H40-&H5F. De extensie-codes die hierbij horen, zijn &H00-&H1F.

PINLIN

Adres: 00AE
Invoer: geen
Uitvoer: HL, C-vlag
Verandert: alle registers, FSTPOS (FBCA), BUF (F55E)

Deze routine vertegenwoordigt de versie van de scherm-editor die gebruikt wordt in de directe modus, dus tijdens het schrijven of wijzigen van een BASIC-programma. Het verschil met INLIN (00B1) is, dat een regel in directe modus altijd begint in de eerste schermkolom; dit heeft gevolgen voor de waarde van FSTPOS, aangezien het kolomnummer daarin gelijk wordt gemaakt aan 1. Dit geldt echter niet als de AUTO-instructie werkzaam is.
Nadat FSTPOS op grond van deze uitzonderingen aangepast is, wordt INLIN aangeroepen.

Bij het verlaten van de routine bevat HL het adres van BUFMIN, ofte wel F55D. Dit is het adres direct vóór het begin van BUF. De C-vlag is hoog als de invoer beëindigd werd door CTRL+STOP, en laag als de invoer beëindigd werd door RETURN.

Binnen de routine PINLIN wordt als eerste actie de RAM-haak H.PINL (FDDB) aangeroepen.

INLIN

Adres:	00B1
Invoer:	geen
Uitvoer:	HL, C-vlag
Verandert:	alle registers, FSTPOS (FBCA), BUF (F55E)

Deze routine vormt het hart van de scherm-editor. De routine accepteert invoer vanaf het toetsenbord en reageert op de invoer door het zetten van tekens op het scherm of het verplaatsen van de cursor. De routine wordt beëindigd als er RETURN of CTRL+STOP vanaf het toetsenbord wordt ingetypt.
Bij beëindiging van de routine door een RETURN wordt de inhoud van de logische regel waar de cursor op dat moment staat, vanaf het scherm in de buffer (BUF) gezet.
Voordat er invoer geaccepteerd wordt, wordt de positie die de cursor heeft bij aanroep van de routine, opgeslagen in FSTPOS. Als de positie in FSTPOS binnen de logische regel valt die wordt ingevoerd, wordt het gedeelte van die regel vóór FSTPOS niet tot de invoer gerekend.
Het lezen van tekens uit de toetsenbordbuffer gebeurt met CHGET (009F). Het schrijven van tekens naar het scherm gebeurt met CHPUT (00A2). Naast de besturingscodes die door CHPUT ondersteund worden, kent INLIN nog een paar codes. Bovendien hebben sommige codes een iets andere betekenis gekregen. De volledige lijst van besturingscodes die door de scherm-editor herkend worden, is te vinden in par. 6.2.7.

Bij het verlaten van de routine bevat HL het adres van BUFMIN, ofte wel F55D. Dit is het adres direct vóór het begin van BUF. De C-vlag is hoog als de invoer beëindigd werd door CTRL+STOP, en laag als de invoer beëindigd werd door RETURN.
Binnen de routine INLIN wordt als eerste actie de RAM-haak H.INLI (FDE5) aangeroepen.

QINLIN

Adres:	00B4
Invoer:	geen
Uitvoer:	HL, C-vlag
Verandert:	alle registers, FSTPOS (FBCA), BUF (F55E)

Deze routine zet eerst een vraagteken gevolgd door een spatie op het scherm, en roept vervolgens INLIN (00B1) aan.
Bij het verlaten van de routine bevat HL het adres van BUFMIN, ofte wel F55D. Dit is het adres direct vóór het begin van BUF. De C-vlag is hoog als de invoer beëindigd werd door CTRL+STOP, en laag als de invoer beëindigd werd door RETURN.
Binnen de routine QINLIN wordt als eerste actie de RAM-haak H.QINL (FDE0) aangeroepen.

BREAKX

Adres:	00B7
Invoer:	geen
Uitvoer:	C-vlag
Verandert:	AF

Deze routine test op het indrukken van CTRL+STOP in een situatie waarin interrupts eventueel zijn uitgeschakeld.
Bij het verlaten van de routine geeft de C-vlag aan of CTRL+STOP ingedrukt was: de vlag is hoog als dit het geval is.

ISCNTC

Adres:	00BA
Invoer:	INTFLG (FC9B)
Uitvoer:	geen
Verandert:	AF

Deze routine test op het indrukken van STOP of CTRL+STOP, en reageert op het indrukken hiervan.
Als het BASIC-programma dat uitgevoerd wordt zich in ROM bevindt – te zien aan de waarde van BASROM (FBB1) – dan wordt de test op het indrukken van STOP of CTRL+STOP niet uitgevoerd.
In andere gevallen wordt aan de hand van de waarde van INTFLG bepaald of er iets interessants is gebeurd. De waarde van INTFLG kan zijn:

0 niets gebeurd
3 CTRL+STOP ingedrukt
4 STOP ingedrukt

Als STOP is ingedrukt, wordt er binnen ISCNTC gewacht tot er opnieuw STOP wordt ingedrukt; daarna wordt de routine verlaten. Als CTRL+STOP is ingedrukt, wordt er op de STOP-interrupt-faciliteit getest; als die niet ingeschakeld is, wordt het programma onderbroken.

Het voornaamste verschil met BREAKX (00B7) is, dat ISCNTC gebaseerd is op het feit dat het interrupt-systeem het eigenlijke indrukken van de toetsen opmerkt, en dat ISCNTC daarop reageert. BREAKX doet het opmerken zelf.

CKCNTC

Adres:	00BD
Invoer:	geen
Uitvoer:	geen
Verandert:	AF

Deze routine doet hetzelfde als ISCNTC (00BA), maar wordt gebruikt door de BASIC-interpreter.

BEEP

Adres:	00C0
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers

Het aanroepen van deze routine resulteert in een piepje. Dit is het piepje dat hoorbaar is (mits ingeschakeld) bij het optreden van een fout in een BASIC-programma. Tevens wordt het geluids-subsysteem opnieuw geïnitialiseerd door een aanroep van GICINI (0090).

CLS

Adres:	00C3
Invoer:	BAKCLR (F3EA), Z-vlag
Uitvoer:	geen
Verandert:	AF, BC, DE

Deze routine maakt het scherm leeg. In een tekstmodus komt dit erop neer dat op alle schermposities spaties komen te staan (de naamtabel wordt met waarden &H20 gevuld); in een grafische modus worden alle pixels teruggezet naar de achtergrondkleur – de kleur waarvan de code in BAKCLR staat – door de patroontabel en kleurentabel te veranderen.
Bij aanroep van de routine moet de Z-vlag laag staan.

POSIT

Adres:	00C6
Invoer:	H, L
Uitvoer:	geen
Verandert:	AF, CSRX (F3DD), CSRY (F3DC)

Deze routine verplaatst de cursor naar een opgegeven positie op het scherm.

Bij aanroep van de routine staat in L het regelnummer en in H het kolomnummer

van een schermpositie. De regels en kolommen worden geteld van de bovenste regel resp. de meest linkse kolom, die in dit geval nummer 1 hebben.

Bij het verlaten van de routine staat de cursor op de positie die door L en H aangeduid werd. CSRX en CSRY hebben de waarde van respectievelijk H en L gekregen.

FNKSB

Adres:	00C9
Invoer:	CNSDFG (F3DE)
Uitvoer:	geen
Verandert:	alle registers

Deze routine vernieuwt de functietoets-definities die op de onderste schermregel getoond worden. De routine test of de waarde van CNSDFG op dit moment groter is dan 0. Als dat het geval is, wordt DSPFNK (00CF) aangeroepen. Dit heeft tot gevolg dat de onderste schermregel wordt beschreven met de huidige (evt. vernieuwde) waarde van de functietoetsen.

ERAFNK

Adres:	00CC
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers, CNSDFG (F3DE)

Deze routine schakelt het tonen van de functietoetsen uit. Dit gebeurt door aan CNSDFG de waarde 0 te geven. Als het scherm in tekstmodus staat, wordt tevens de onderste regel van het scherm schoongemaakt.

DSPFNK

Adres:	00CF
Invoer:	geen
Uitvoer:	geen
Verandert:	alle registers, CNSDFG (F3DE)

Deze routine schakelt het tonen van de functietoetsen in. Dat gebeurt door aan CNSDFG een waarde groter dan 0 te geven. Als het scherm in tekstmodus staat worden de definities nu op de onderste regel getoond.

TOTEXT

Adres:	00D2
Invoer:	SCRMOD (FCAF), OLDSCR (FCB0)
Uitvoer:	geen
Verandert:	alle registers, SCRMOD (FCAF)

Deze routine zet de schermmodus op een tekstmodus als de huidige modus een grafische modus is.

De huidige schermmodus staat in SCRMOD. Als SCRMOD de waarde 2 of 3 heeft, moet de schermmodus veranderd worden in een tekstmodus. De tekstmodus die daarvoor gekozen wordt, staat in OLDSCR.
Het veranderen van de schermmodus gebeurt door een aanroep van INITXT (006C) of INIT32 (006F).

In TOTEXT vindt als eerste actie (voordat de schermmodus veranderd wordt) een aanroep plaats van de RAM-haak H.TOTE (FDBD).

Spelapparatuur
Via de spelingangen van de PSG kunnen verschillende soorten spelapparatuur aangesloten worden. De bekendste zijn de joysticks, maar er bestaat ook de mogelijkheid voor een paddle controller of een 'touch pad'.
De routines in deze categorie dienen om al deze apparaten uit te lezen.

GTSTCK
Adres: 00D5
Invoer: A
Uitvoer: A
Verandert: alle registers

Deze routine leest de huidige richting van een joystick. Bij aanroep van de routine bevat A het nummer van de te gebruiken joystick. De volgende nummers zijn mogelijk:

0 toetsenbord
1 joystick 1
2 joystick 2

Bij het verlaten van de routine bevat A de richting die de joystick aangeeft. De codering van de richting is als volgt:

0 geen richting
1 naar boven
2 rechts boven
3 naar rechts
4 rechts onder
5 naar onder
6 links onder
7 naar links
8 links boven

GTTRIG
Adres: 00D8
Invoer: A
Uitvoer: A
Verandert: alle registers

Deze routine leest de huidige status van de vuurknop van een joystick. Bij het aanroepen van de routine staat in A het nummer van de vuurknop die uitgelezen moet worden. De mogelijke waarden voor dit nummer zijn:

0 spatiebalk
1 eerste vuurknop joystick 1
2 eerste vuurknop joystick 2
3 tweede vuurknop joystick 1
4 tweede vuurknop joystick 2

Bij het verlaten van de routine staat in A de status van de geselecteerde vuurknop. A kan als waarde hebben:

00 niet ingedrukt
FF wel ingedrukt

GTPAD

Adres:	00DB
Invoer:	A
Uitvoer:	A
Verandert:	alle registers, PADX (FC9C), PADY (FC9D)

Deze routine leest de toestand van de touch pad. De waarde in A bij aanroep van de routine geeft aan wat er precies gelezen moet worden. A kan als waarden hebben:

0 lees touch pad 1
1 bepaal x-coördinaat pad 1
2 bepaal y-coördinaat pad 1
3 bepaal schakelaarstand pad 1
4 lees touch pad 2
5 bepaal x-coördinaat pad 2
6 bepaal y-coördinaat pad 2
7 bepaal schakelaarstand pad 2

Bij het verlaten van de routine bevat A de gelezen waarde. Wat deze waarde voorstelt hangt af van de waarde van A bij aanroep (zie boven).

Bij een aanroep van GTPAD met A=0 of A=4, worden de coördinaten van touch pad 1 of 2 vastgesteld, en de stand van de schakelaar gelezen. De waarde van de coördinaten worden opgeslagen in PADX en PADY. Bij het verlaten van de routine bevat A de schakelaarstand; deze kan de volgende waarden hebben:

00 niet aangeraakt
FF wel aangeraakt

Bij het aanroepen van GTPAD met in A de waarde 1, 2, 5 of 6 wordt PADX of PADY uitgelezen. In dat geval wordt niet de huidige waarde van de touch pad opnieuw bepaald! Twee keer dezelfde aanroep, zonder tussenliggende aanroep met A=0 of A=4, heeft dus altijd hetzelfde resultaat tot gevolg.

Bij een aanroep met A=3 of A=7 wordt de stand van de schakelaar opnieuw bepaald en in A opgeleverd. De mogelijke waarden van A zijn dezelfde als bij een aanroep met A=0 of A=4.

GTPDL

Adres:	00DE
Invoer:	A
Uitvoer:	A
Verandert:	alle registers

Deze routine leest een paddle controller uit. Aan elke ingang kunnen maximaal zes paddle controller aangesloten worden. Deze duiden we aan met A}F. De waarde van A geeft aan welke paddle controller uitgelezen moet worden. A kan als waarde hebben:

1 paddle controller A van ingang 1
2 paddle controller A van ingang 2
3 paddle controller B van ingang 1
4 paddle controller B van ingang 2
5 paddle controller C van ingang 1
6 paddle controller C van ingang 2
7 paddle controller D van ingang 1
8 paddle controller D van ingang 2
9 paddle controller E van ingang 1
10 paddle controller E van ingang 2
11 paddle controller F van ingang 1
12 paddle controller F van ingang 2

Bij het verlaten van de routine bevat A de stand van de geselecteerde paddle controller.

Routines van het cassettesysteem

De volgende routines dienen om het cassettesysteem te gebruiken. Voor details over dit systeem zie par. 6.2.5.
De routines ondersteunen het cassettesysteem op een tamelijk laag niveau: het niveau van geschreven en gelezen bytes.

TAPION

Adres:	00E1
Invoer:	geen
Uitvoer:	C-vlag
Verandert:	alle registers, LOWLIM (FCA4), WINWID (FCA5)

Deze routine maakt het cassettesysteem klaar om in te gaan lezen. Dit gebeurt door een kopblok van cassette te lezen. Aan de hand van het kopblok wordt de snelheid

bepaald die gebruikt is bij het schrijven. Gegevens over deze snelheid komen te staan in LOWLIM en WINWID.
Als bij het verlaten van de routine de C-vlag is gezet, is het inlezen van het kopblok onderbroken door het indrukken van CTRL+STOP. LOWLIM en WINWID hebben in dat geval geen geldige waarden gekregen; de motor van de cassetterecorder loopt echter nog.

TAPIN

Adres:	00E4
Invoer:	LOWLIM (FCA4), WINWID (FCA5)
Uitvoer:	A, C-vlag
Verandert:	alle registers

Deze routine leest een byte van cassette. Bij aanroep van de routine moet de motor van de cassetterecorder lopen, en moet er een kopblok zijn ingelezen, zodat WINWID en LOWLIM goede waarden hebben.
De waarde van het gelezen byte staat bij het verlaten van de routine in A.

De motor van de cassetterecorder blijft tijdens en na het inlezen doorlopen. Als er dus bytes direct na elkaar op cassette zijn geschreven, moeten ze ook weer direct na elkaar worden ingelezen, omdat ze anders de recorderkop gepasseerd zijn.
Als bij het verlaten van de routine de C-vlag is gezet, is het inlezen van het byte onderbroken door het indrukken van CTRL+STOP, en heeft A geen geldige waarde.

TAPIOF

Adres:	00E7
Invoer:	geen
Uitvoer:	geen
Verandert:	geen

Deze routine beëindigt het lezen van cassette. Dit gebeurt door de motor van de recorder stop te zetten.

TAPOON

Adres:	00EA
Invoer:	A, LOW (F406), HIGH (F408), HEADER (F40A)
Uitvoer:	C-vlag
Verandert:	alle registers

Deze routine maakt het cassettesysteem klaar voor een schrijfactie. De motor van de recorder wordt aangezet, en er wordt een kopblok naar de cassette geschreven. De waarde van A bij aanroep van de routine bepaalt de lengte van het kopblok: als A=0 dan wordt een kort kopblok geschreven, anders een lang kopblok.
Bij het schrijven van het kopblok worden de waarden van LOW, HIGH en HEADER gebruikt om de duur van hoge en lage uitvoer te bepalen.

Als bij het verlaten van de routine de C-vlag is gezet, is het schrijven van het kopblok onderbroken door het indrukken van CTRL+STOP, en is het schrijven van het kopblok mislukt. De motor van de cassetterecorder blijft echter doorlopen.

TAPOUT

Adres:	00ED
Invoer:	A, LOW (F406), HIGH (F408), HEADER (F40A)
Uitvoer:	C-vlag
Verandert:	alle registers

Deze routine schrijft een byte naar cassette. De motor moet lopen bij het uitvoeren van de routine, en de recorder moet op 'opname' staan.
De waarde van het byte dat weg moet worden geschreven, staat bij aanroep van de routine in A. Bij het verlaten van de routine geeft de C-vlag aan of alles naar wens is verlopen: de vlag is gezet als het schrijven onderbroken is (door het indrukken van CTRL+STOP).
Bij het schrijven van het kopblok worden de waarden van LOW, HIGH en HEADER gebruikt om de duur van hoge en lage niveaus bij de uitvoer te bepalen.

TAPOOF

Adres:	00F0
Invoer:	BC
Uitvoer:	geen
Verandert:	geen

Deze routine beëindigt het schrijven naar cassette. Dit gebeurt door de waarde van BC af te tellen en de motor van de cassetterecorder te stoppen zodra 0 bereikt is.

STMOTR

Adres:	00F3
Invoer:	A
Uitvoer:	geen
Verandert:	AF

Deze routine start of stopt de motor van de cassetterecorder. De keuze tussen starten of stoppen wordt gemaakt aan de hand van de waarde van A bij aanroep van de routine. Als A=0
0 dan wordt de motor gestopt, als A=1 dan wordt de motor gestart, en als A=255 dan wordt de toestand van de motor omgedraaid.

Routines van het queue-systeem

De hier volgende ingangen worden gebruikt voor het queue-systeem; zie par. 6.2.3. Het queue-systeem is maar gedeeltelijk via OS-ingangen te bereiken: er kunnen door de programmeur alleen waarden ingestopt worden, niet uitgehaald. Verder kan het aantal bytes dat op een gegeven moment in een queue staat worden bepaald.

LFTQ

Adres:	00F6
Invoer:	A, QUEUES (F3F2)
Uitvoer:	A
Verandert:	AF, BC, HL

Deze routine telt het aantal bytes dat ten tijde van de aanroep in een queue staat. Het nummer van de queue die onderzocht moet worden staat bij aanroep in A; in QUEUES staat het beginadres van de queue-descriptoren. Bij het verlaten van de routine staat het aantal bytes in A.

PUTQ

Adres:	00F9
Invoer:	A, E, QUEUES (F3F3)
Uitvoer:	Z-vlag
Verandert:	AF, BC, HL

Deze routine maakt deel uit van het queue-systeem (zie par. 6.2.3). De routine zet een nieuwe waarde in een queue.

Bij aanroep van de routine staat in A het nummer van de queue waarin een nieuwe waarde gezet moet worden, in E de waarde die in de queue moet komen te staan, en in QUEUES het beginadres van de tabel waarin de queue-descriptoren staan.

Bij het verlaten van de routine geeft de Z-vlag aan of de waarde inderdaad in de queue gezet is; als de Z-vlag hoog is, is dit niet gebeurd, omdat de queue vol is.

Grafische routines

De volgende ingangen zijn voor routines die worden gebruikt door het grafische systeem van de MSX-computer; zie par. 6.2.8. Met deze routines kan op het niveau van pixels getekend worden. Verder is het mogelijk één enkele lijn van het scherm in te kleuren. Routines voor ingewikkelde grafische acties zijn niet opgenomen.

RIGHTC

Adres:	00FC
Invoer:	CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer:	geen
Verandert:	AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar rechts. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct rechts ligt van het punt dat bij aanroep aangegeven werd.
Als het oude punt precies op de rechter rand van een lijn lag, wordt dat niet gesignaleerd, maar ligt het nieuwe punt op de linker rand van de lijn eronder.

LEFTC

Adres: 00FF
Invoer: CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer: geen
Verandert: AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar links. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct links ligt van het punt dat bij aanroep aangegeven werd.
Als het oude punt precies op de linker rand van een lijn lag, wordt dat niet gesignaleerd, maar ligt het nieuwe punt op de rechter rand van de lijn erboven.

UPC

Adres: 0102
Invoer: CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer: geen
Verandert: AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar boven. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct boven het punt ligt dat bij aanroep aangegeven werd.
Als het oude punt precies op de bovenste lijn lag, wordt dat niet gesignaleerd, maar krijgt CLOC een waarde die buiten de patroontabel valt.

TUPC

Adres: 0105
Invoer: CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer: C-vlag
Verandert: AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar boven. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct boven het punt ligt dat bij aanroep aangegeven werd.
Als het bepalen van het nieuwe punt zonder problemen verlopen is, staat de C-vlag laag bij het verlaten van de routine. Als het oude punt precies op de bovenste lijn lag, wordt de grafische accumulator niet van waarde veranderd, en is de C-vlag hoog bij het verlaten van de routine.

DOWNC

Adres: 0108
Invoer: CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer: geen
Verandert: AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar beneden. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct onder het punt ligt dat bij aanroep aangegeven werd.
Als het oude punt precies op de onderste lijn lag, wordt dat niet gesignaleerd, maar krijgt CLOC een waarde die buiten de patroontabel valt.

TDOWNC

Adres:	010B
Invoer:	CLOC (F92A), CMASK (F92C), SCRMOD (FCAF)
Uitvoer:	C-vlag
Verandert:	AF

Deze routine beweegt de grafische accumulator (de waarde van CLOC, CMASK) één pixel of modus 3-blok naar beneden. Met andere woorden, bij het verlaten van de routine geven CLOC, CMASK samen een punt aan dat direct onder het punt ligt dat bij aanroep aangegeven werd.
Als het bepalen van het nieuwe punt zonder problemen verlopen is, staat de C-vlag laag bij het verlaten van de routine. Als het oude punt precies op de onderste lijn lag, wordt de grafische accumulator niet van waarde veranderd, en is de C-vlag hoog bij het verlaten van de routine.

SCALXY

Adres:	010E
Invoer:	BC, DE
Uitvoer:	BC, DE, C-vlag
Verandert:	AF, BC, DE

Deze routine test of een coördinatenpaar binnen de grenzen van het scherm valt, en past de coördinaten aan als dit niet het geval is.
Bij aanroep bevat BC de x-coördinaat en DE de y-coördinaat. Bij terugkomst is de C-vlag hoog als het coördinatenpaar binnen de schermgrenzen lag, en anders laag. Bovendien bevatten BC en DE de respectievelijke coördinaten, zo getransformeerd dat de nieuwe waarde binnen de grenzen (aan de rand) ligt, zelfs als de oude waarde er buiten lag.

Als het scherm in modus 3 staat, zijn bij terugkomst de coördinaten bovendien omgezet zodat ze verwijzen naar modus 3-blokken; ofte wel, de coördinaten zijn door 4 gedeeld.

MAPXYC

Adres:	0111
Invoer:	BC, DE
Uitvoer:	A, HL, CLOC (F92A), CMASK (F92C)
Verandert:	AF, HL

Deze routine zet een coördinatenpaar om naar een adres in het videogeheugen en een bitmasker die samen het pixel of het modus 3-blok adresseren dat door de coördinaten wordt aangeduid.
Bij het aanroepen van de routine bevat BC de x-coördinaat en DE de y-coördinaat van het pixel of modus 3-blok. Aangenomen wordt dat de waarden van deze coördinaten binnen de schermgrenzen liggen en dat de coördinaten al omgezet zijn als het scherm in modus 3 staat. Dit kan gedaan worden door aanroep van de routine SCALXY (010E).

Bij het verlaten van de routine bevat CLOC het adres (binnen de patroontabel in het videogeheugen) van het byte waarin de informatie over het bedoelde pixel staat. HL is gelijk aan de waarde van CLOC. CMASK is een masker voor het byte dat door CLOC geadresseerd wordt, zodat alleen de relevante bits van dat byte geselecteerd worden. A is gelijk aan de waarde van CMASK.

De waarden van CLOC en CMASK samen worden wel aangeduid met de grafische accumulator.

FETCHC

Adres:	0114
Invoer:	CLOC (F92A), CMASK (F92C)
Uitvoer:	A, HL
Verandert:	A, HL

Deze routine haalt de waarde van de grafische accumulator op (het adres en masker die het laatst bereikte pixel of modus 3-blok aanduiden). Bij het aanroepen van de routine staan deze waarden in CLOC resp. CMASK; bij het verlaten van de routine zijn ze gekopieerd in HL en A.

STOREC

Adres:	0117
Invoer:	A, HL
Uitvoer:	geen
Verandert:	CLOC (F92A), CMASK (F92C)

Deze routine geeft een nieuwe waarde aan het adres en het masker die het laatst bereikte punt aanduiden (de grafische accumulator). De nieuwe waarden staan bij aanroep van de routine in HL en A; bij het verlaten van de routine zijn deze waarden in CLOC en CMASK gekopieerd.

SETATR

Adres:	011A
Invoer:	A
Uitvoer:	C-vlag, ATRBYT (F3F2)
Verandert:	F, ATRBYT (F3F2)

Deze routine verandert de werkkleur voor grafische akties. Deze werkkleur wordt bewaard in de lokatie ATRBYT.
Bij het aanroepen van de routine moet de code voor de nieuwe werkkleur in A staan. Bij het verlaten van de routine is de C-vlag hoog als de code voor de werkkleur geen legale kleurcode was (van 0 t/m 15), anders laag. Als de C-vlag laag is, bevat ATRBYT naderhand de nieuwe code.

READC

Adres:	011D
Invoer:	CLOC (F92A), CMASK (F92C)
Uitvoer:	A
Verandert:	AF

Deze routine bepaalt de kleur van het pixel of modus 3-blok dat aangeduid wordt door CLOC en CMASK. Bij het verlaten van de routine staat de code voor de kleur in A.

SETC

Adres:	0120
Invoer:	ATRBYT (F3F2), CLOC (F92A), CMASK (F92C)
Uitvoer:	geen
Verandert:	AF

Deze routine verandert de kleur van het pixel of modus 3-blok dat aangeduid wordt door CLOC en CMASK in de kleur waarvan de code in ATRBYT staat.

NSETCX

Adres:	0123
Invoer:	HL, ATRBYT (F3F2), CLOC (F92A), CMASK (F92C)
Uitvoer:	geen
Verandert:	alle registers, CLOC (F92A), CMASK (F92C)

Het aanroepen van deze routine heeft hetzelfde effect als HL keer achter elkaar SETC, RIGHTC aanroepen. Deze routine werkt echter sneller.
Bij aanroep duiden CLOC en CMASK het eerste pixel of modus 3-blok aan dat van kleur veranderd moet worden. De routine trekt vanaf dit punt een horizontale lijn naar rechts ter lengte van HL pixels of modus 3-blokken in de kleur waarvan de code in ATRBYT staat.
Bij het verlaten van de routine duiden CLOC en CMASK het punt aan direct rechts van het laatst gekleurde pixel of modus 3-blok.

GTASPC

Adres:	0126
Invoer:	ASPCT1 (F40B), ASPCT2 (F40D)
Uitvoer:	DE, HL
Verandert:	DE, HL

Deze routine haalt de hoogte/breedte-verhoudingen voor een cirkel uit de geheugenplaatsen ASPCT1 en ASPCT2. De waarde van ASPCT1 komt in DE, de waarde van ASPCT2 in HL.
GTASPC wordt gebruikt in het BASIC-statement CIRCLE.

PNTINI

Adres:	0129
Invoer:	A, ATRBYT (F3F2)
Uitvoer:	A, C-vlag, BRDATR (FCB2)
Verandert:	AF, BRDATR (FCB2)

Deze routine bepaalt de grenskleur voor vlakvulling. Bij aanroep bevat A de gewenste grenskleur. ATRBYT bevat de werkkleur, d.w.z. de kleur waarmee het vlak gevuld zal worden. Bij het verlaten van de routine staat de te gebruiken grenskleur in BRDATR en in het A-register.
In schermmodus 2 mag de grenskleur geen andere zijn dan de werkkleur. In dat geval worden BRDATR en A gelijk gemaakt aan ATRBYT. In modus 3 wordt BRDATR gelijk gemaakt aan A.
Als de code voor de werkelijke grenskleur geen legale kleurcode is (van 0 t/m 15) dan is bij het verlaten van de routine de C-vlag hoog, anders laag. Dit kan alleen voorkomen in modus 3, aangezien in modus 2 de waarde van ATRBYT gebruikt wordt, en dit altijd een legale waarde is.
PNTINI wordt gebruikt in het BASIC-statement PAINT.

SCANR

Adres:	012C
Invoer:	B, DE, ATRBYT (F3F2), CLOC (F92A), CMASK (F92C), BRDATR (FCB2)
Uitvoer:	C, DE, HL, CSAVEA (F942), CSAVEM (F944)
Verandert:	alle registers, CLOC, CMASK, CSAVEA, CSAVEM

Deze routine kan worden gebruikt om een lijn op het grafische scherm in te kleuren tussen grenspunten met een opgegeven kleur.
Bij het aanroepen van de routine geven CLOC en CMASK het beginpunt aan van waaraf het inkleuren moet starten. ATRBYT is de kleur waarmee geschilderd moet worden; BRDATR de grenskleur. DE is het aantal punten of modus 3-blokken die maximaal onderzocht moeten worden. B is een vlag die aangeeft of er werkelijk gekleurd moet worden, dan wel alleen onderzocht op mogelijke inkleuring. B heeft als mogelijke waarden:

0	onderzoeken
>0	inkleuren

Vanaf het beginpunt wordt er naar rechts gegaan tot er een punt wordt gevonden met een andere kleur dan de grenskleur, of tot DE=0, of tot de rand van het scherm bereikt is. In CSAVEA en CSAVEM wordt het eerste punt bewaard met een andere

kleur dan de grenskleur, als er zo'n punt gevonden is. Het registerpaar DE geeft aan of dat het geval is: DE=0 als alle onderzochte punten de grenskleur hadden.
Vanaf het punt in CSAVEA en CSAVEM (indien gevonden), wordt verder naar rechts gezocht tot het eerste punt dat weer de grenskleur heeft, of tot de rand van het scherm. In HL wordt bijgehouden hoeveel punten er in dit stuk gevonden worden.
Als B0, dan worden de punten vanaf CSAVEA, CSAVEM ingekleurd met de kleur die in ATRBYT staat.
Bij het verlaten van de routine adresseren CLOC en CMASK het laatst onderzochte (en niet ingekleurde) pixel of modus 3-blok. De waarde van C geeft aan of in het ingekleurde stuk alle pixels misschien al de werkkleur (kleur van ATRBYT) hadden: C>0 als dit het geval is.
SCANR wordt gebruikt in het BASIC-statement PAINT.

SCANL

Adres:	012F
Invoer:	B, DE, ATRBYT (F3F2), CLOC (F92A), CMASK (F92C), BRDATR (FCB2)
Uitvoer:	C, DE, HL, CSAVEA (F942), CSAVEM (F944)
Verandert:	alle registers, CLOC, CMASK, CSAVEA, CSAVEM

Deze routine doet precies hetzelfde als SCANR (012C), behalve dat de richting van het onderzoek naar links is in plaats van naar rechts.
SCANL wordt gebruikt in het BASIC-statement PAINT.

Allerlei

De nu volgende OS-ingangen zijn niet in één categorie onder te verdelen. Gedeeltelijk vormen de routines aanvullingen op al besproken categorieën; gedeeltelijk zijn ze in geen enkele categorie onder te brengen.

CHGCAP

Adres:	0132
Invoer:	A
Uitvoer:	geen
Verandert:	AF

Deze routine wordt gebruikt om de toestand van het lichtje bij de CAPS-toets te veranderen. Als bij het aanroepen van de routine A = 0, dan wordt het licht uitgezet, anders aan.
De aansturing van het lichtje van de CAPS-toets verloopt via poort C van de PPI.

CHGSND

Adres:	0135
Invoer:	A
Uitvoer:	geen
Verandert:	AF

Deze routine wordt gebruikt om de toestand van de 1-bits geluidsuitgang van de PPI te veranderen. Als bij het aanroepen van de routine A=0, dan wordt het bit van de geluidsuitgang laag gezet, anders hoog.
De aansturing van de geluidsuitgang verloopt via poort C van de PPI. In het MSX-systeem wordt de uitgang gebruikt voor de toetsenbordklik.

RSLREG

Adres: 0138
Invoer: geen
Uitvoer: A
Verandert: A

Deze routine leest de huidige waarde van poort A van de PPI, ofte wel het gleuf-selectieregister. De waarde van dit register komt in A te staan.
De bits van het gleuf-selectieregister hebben de volgende betekenis:

Bit 7-6	geselect. gleuf bij pag. 0	bereik: 0-3
Bit 5-4	geselect. gleuf bij pag. 1	bereik: 0-3
Bit 3-2	geselect. gleuf bij pag. 2	bereik: 0-3
Bit 1-0	geselect. gleuf bij pag. 3	bereik: 0-3

WSLREG

Adres: 013B
Invoer: A
Uitvoer: geen
Verandert: geen

Deze routine schrijft een nieuwe waarde naar het gleuf-selectieregister – poort A van de PPI. De waarde die naar het register geschreven wordt, staat bij aanroep van de routine in A.
De betekenis van de bits van het register is vermeld bij RSLREG (0138).

RDVDP

Adres: 013E
Invoer: geen
Uitvoer: A
Verandert: A

Deze routine leest de huidige waarde van het statusregister van de VDP; register 8. De waarde van het register staat bij het verlaten van de routine in A.
Voor de betekenis van de bits van het register zie par. 3.3.

SNSMAT

Adres: 0141
Invoer: A
Uitvoer: A
Verandert: AF

Deze routine bepaalt de huidige toestand van een bepaalde rij van de toetsenbordmatrix.
Bij het aanroepen van de routine staat in A het nummer van de rij die onderzocht moet worden. Bij het verlaten van de routine bevat A de toestand van deze rij. Elk kolomelement in de rij wordt vertegenwoordigd door een bit van A, dat hoog is als de toets los is gelaten, en laag als de toets op dat moment is ingedrukt.
Meer informatie over de toetsenbordmatrix en het scannen van het toetsenbord is te vinden in par. 5.2.

PHYDIO

Adres:	0144
Invoer:	geen
Uitvoer:	geen
Verandert:	geen

Deze routine wordt gebruikt bij het lezen en schrijven van diskettes, om de fysieke I/O plaats te laten vinden.
In het huidige MSX-systeem wordt in de routine PHYDIO alleen de RAM-haak H.PHYD (FFA7) aangeroepen.

FORMAT

Adres:	0147
Invoer:	geen
Uitvoer:	geen
Verandert:	geen

Deze routine verzorgt het initialiseren (formatteren) van I/O-apparaten zoals diskettestation.
In het huidige MSX-systeem wordt in de routine FORMAT alleen de RAM-haak H.FORM (FFAC) aangeroepen.

ISFLIO

Adres:	014A
Invoer:	PTRFIL (F864)
Uitvoer:	vlaggen
Verandert:	AF

Deze routine test of er op dit moment I/O plaatsvindt. Deze test gebeurt door de waarde van PTRFIL te vergelijken met 0. Het resultaat van de vergelijking wordt bij terugkeer uit de routine weergegeven door de vlaggen van het F-register.

OUTDLP

Adres:	014D
Invoer:	A, NTMSXP (F417)
Uitvoer:	geen
Verandert:	F, LPTPOS (F415)

Deze routine verzorgt het printen van een teken uit de ASCII-tabel van de MSX-computer. De ASCII-code van het teken dat geprint moet worden, staat in A.

Vergeleken met LPTOUT (00A5) is deze routine uitgebreider. De routine zet codes van sommige tekens om naar spaties, in plaats van ze direct naar de printer te sturen. Voor het uiteindelijke versturen van een code naar de printer maakt OUTDLP weer gebruik van LPTOUT.
De code voor een TAB (9) wordt omgezet naar spaties die naar de printer worden gestuurd in plaats van TAB. Er worden spaties gestuurd tot de waarde van LPTPOS een veelvoud is van 8.
Als de printer geen MSX-printer is (te zien aan de waarde van NTMSXP) dan worden grafische tekens (code 1 gevolgd door een code tussen &H41 en &H60) veranderd in spaties.

Als de printer niet on-line is, dan wordt de BASIC-foutmelding 'Device I/O error' gegeven.

GETVCP

Adres:	0150
Invoer:	A
Uitvoer:	HL
Verandert:	A, HL

Deze routine zoekt het adres op van het veld QLENGX binnen het VCB van een van de stemmen van het geluids-subsysteem.
Bij het aanroepen van de routine staat het nummer van de stem waarvan QLENGX opgezocht moet worden in A. Bij het verlaten van de routine staat het gewenste adres in HL.
Voor de betekenis van QLENGX en meer informatie over het geluids-subsysteem zie par. 6.2.4.

GETVC2

Adres:	0153
Invoer:	L, QUEUEN (FB3E)
Uitvoer:	HL
Verandert:	A, HL

Deze routine zoekt het adres op van een veld binnen het VCB van een van de stemmen van het geluids-subsysteem.
Het nummer van de stem waarvan het VCB opgezocht moet worden, staat bij aanroep van de routine in QUEUEN. De verplaatsing (ten opzichte van het begin van het VCB) van het veld dat gezocht wordt staat in L. Bij het verlaten van de routine staat het gewenste adres in HL.
Voor meer informatie over de VCB's en het geluids-subsysteem zie par. 6.2.4.

KILBUF

Adres:	0156
Invoer:	geen
Uitvoer:	geen
Verandert:	HL, GETPNT (F3FA)

Deze routine maakt de toetsenbordbuffer leeg door de waarde van GETPNT gelijk te maken aan de waarde van PUTPNT (F3F8). De buffer zelf wordt hierdoor niet aangetast, maar wel de bufferadministratie, zodat de buffer leeg lijkt te zijn.

CALBAS

Adres:	0159
Invoer:	IX

Deze routine roept de BASIC-interpreter aan op het adres dat in IX staat. Daartoe wordt eventueel eerst de BASIC-gleuf geselecteerd als op dit moment een andere gleuf geselecteerd is voor het adresgebied 4000-BFFF.

Ruimte voor uitbreidingen

Het gebied voor OS-ingangen eindigt met een ongebruikt gebied van 80 bytes lang. Dit gebied is gereserveerd voor uitbreiding van de hoeveelheid beschikbare ingangen in toekomstige versies van het MSX-systeem.

6.4 OS geheugenbeheer

Om zijn taken te kunnen vervullen, heeft het Operating System RAM-geheugen nodig, als werkgeheugen en als bewaarplaats voor allerlei instelbare waarden.
In de MSX-computer is voor het OS geheugen gereserveerd aan de 'bovenkant' van het geheugen, van geheugenadres FFFE naar beneden. In deze paragraaf wordt verteld op welke manier dit geheugen ingedeeld is en op welke manier er van het geheugen gebruik gemaakt wordt.

De algemene indeling van het Operating System-geheugen is weergegeven in afbeelding 6.2. Het geheugen is opgedeeld in gebieden, die elk een taak toegewezen krijgen. We zullen deze gebieden één voor één de revue laten passeren. Adressen in deze gebieden die een speciale betekenis hebben, hebben een naam van zes letters gekregen.

6.4.1 Hulproutine-gebied

Het eerste geheugengebied bevat enkele machinetaalroutines die door het OS gebruikt worden voor het bereiken (lezen, schrjven, aanroepen) van andere gleuven dan de huidige. Deze routines worden bij het aanzetten van de computer in RAM gezet. De routines worden aangeroepen vanuit ROM als de te bereiken pagina zich in hetzelfde adresbereik bevindt als de ROM.

F380 – F3AD	hulproutine – gebied
F3AE – F3F2	schermparametergebied
F3F3 – F418	toetsenbord – en cassette – parameters
F419 \ F85E	werkruimte BASIC – interpreter
F85F \ F91E	werkgeheugen filesysteem
F91F – F91E	werkgeheugen schermroutines
F959 \ FBAF	werkgeheugen queue-en geluidssysteem
FBB0 \ FCAD	instellingen scherm – editor en interrupt-systeem
FCAE – FCC0	allerlei
FCC1 \ FD99	opslag gleuf – informatie
FD9A \ FFCF	RAM – haken
FFD0 – FFFE	ongebruikt
FFFF	selectieregister secundaire gleuf

Afb. 6.2. Algemene indeling van het RAM-geheugen

F380-F384 RDPRIM
Begin van een machinetaalroutine die een waarde uit een pagina van een andere gleuf leest.

F385-F38B WRPRIM
Begin van een machinetaalroutine die een nieuwe waarde naar een pagina van een andere gleuf schrijft.

F38C-F399 CLPRIM
Begin van een machinetaalroutine waarin een aanroep van een routine in een pagina van een andere gleuf geïmplementeerd is.

F39A-F3AD USRTAB
Werkgebied voor het DEF USR-statement. USRTAB is een tabel van beginadressen van machinetaalroutines. De adressen in deze tabel zijn de adressen waarnaar de USR-functies in BASIC verwijzen.
De adressen worden voor alle USR-functies geïnitialiseerd op 475A. Een sprong naar dat adres levert namelijk de foutmelding 'Syntax error' op. Zolang de adressen niet van een nieuwe waarde zijn voorzien, blijven de USR-functies dus een foutmelding opleveren.

6.4.2 Schermparameter-gebied

In dit gebied worden allerlei gegevens omtrent het OS-videosysteem opgeslagen. Door veranderen van deze waarden kunnen soms interessante resultaten worden bereikt.

F3AE LINL40

Het aantal gebruikte posities op een regel van schermmodus 0. De waarde van LINL40 wordt naar LINLEN geschreven als met een SCREEN-statement modus 0 wordt ingesteld.
(Beginwaarde: 39)

F3AF LINL32

Het aantal gebruikte posities op een regel van schermmodus 1. De waarde van LINL32 wordt naar LINLEN geschreven als met een SCREEN-statement modus 1 wordt ingesteld.
(Beginwaarde: 29)

F3B0 LINLEN

Het aantal gebruikte posities op een regel in de huidige schermmodus. Een regel wordt zó gerekend dat de gebruikte posities zich in het midden van het schermgebied bevinden. Posities die niet worden gebruikt, scrollen niet met het scherm mee, en kunnen niet worden beschreven.
Deze waarde kan vanuit BASIC veranderd worden met een WIDTH-statement.
(Beginwaarde: 39)

F3B1 CRTCNT

Het aantal gebruikte regels van het scherm. Dit aantal wordt gerekend vanaf de bovenste schermregel. Alleen op de aangeduide regels kan worden geschreven, en alleen deze regels scrollen.
(Beginwaarde: 24)

F3B2 CLMLST

Het aantal posities van een tabulator-kolom. De tabulator-stops bevinden zich op veelvouden van deze waarde.
(Beginwaarde: 14)

F3B3-F3B4 TXTNAM

Het beginadres van de naamtabel in schermmodus 0. Dit adres wordt gebruikt als schermmodus 0 aangeroepen wordt om de waarde van NAMBAS (F922) te veranderen.
(Beginwaarde: &H0000)

F3B5-F3B6 TXTCOL
Het beginadres van de kleurentabel in schermmodus 0. De kleurentabel is in schermmodus 0 ongebruikt.
(Beginwaarde: &H0000)

F3B7-F3B8 TXTCGP
Het beginadres van de patroontabel in schermmodus 0. Dit adres wordt gebruikt als schermmodus 0 aangeroepen wordt om de waarde van CGPBAS (F924) te veranderen.
(Beginwaarde: &H0800)

F3B9-F3BA TXTATR
Het beginadres van de sprite-plaatstabel in schermmodus 0. Dit adres wordt gebruikt als schermmodus 0 aangeroepen wordt om de waarde van ATRBAS (F928) te veranderen. De sprite-plaatstabel is in schermmodus 0 ongebruikt.
(Beginwaarde: &H0000)

F3BB-F3BC TXTPAT
Het beginadres van de sprite-patroontabel in schermmodus 0. Dit adres wordt gebruikt als schermmodus 0 aangeroepen wordt om de waarde van PATBAS (F926) te veranderen.
De sprite-patroontabel is in schermmodus 0 ongebruikt.
(Beginwaarde: &H0000)

F3BD-F3BE T32NAM
Het beginadres van de naamtabel in schermmodus 1. Dit adres wordt gebruikt als schermmodus 1 aangeroepen wordt om de waarde van NAMBAS (F922) te veranderen.
(Beginwaarde: &H1800)

F3BF-F3C0 T32COL
Het beginadres van de kleurentabel in schermmodus 1.
(Beginwaarde: &H2000)

F3C1-F3C2 T32CGP
Het beginadres van de patroontabel in schermmodus 1. Dit adres wordt gebruikt als schermmodus 1 aangeroepen wordt om de waarde van CGPBAS (F924) te veranderen.
(Beginwaarde: &H0000)

F3C3-F3C4 T32ATR
Het beginadres van de sprite-plaatstabel in schermmodus 1. Dit adres wordt gebruikt als schermmodus 1 aangeroepen wordt om de waarde van ATRBAS (F928) te veranderen.
(Beginwaarde: &H1B00)

F3C5-F3C6 T32PAT
Het beginadres van de sprite-patroontabel in schermmodus 1. Dit adres wordt gebruikt als schermmodus 1 aangeroepen wordt om de waarde van PATBAS (F926) te veranderen.
(Beginwaarde: &H800)

F3C7-F3C8 GRPNAM
Het beginadres van de naamtabel in schermmodus 2. Dit adres wordt gebruikt als schermmodus 2 aangeroepen wordt om de waarde van NAMBAS (F922) te veranderen.
(Beginwaarde: &H1800)

F3C9-F3CA GRPCOL
Het beginadres van de kleurentabel in schermmodus 2.
(Beginwaarde: &H2000)

F3CB-F3CC GRPCGP
Het beginadres van de patroontabel in schermmodus 2. Dit adres wordt gebruikt als schermmodus 2 aangeroepen wordt om de waarde van CGPBAS (F924) te veranderen.
(Beginwaarde: &H0000)

F3CD-F3CE GRPATR
Het beginadres van de sprite-plaatstabel in schermmodus 2. Dit adres wordt gebruikt als schermmodus 2 aangeroepen wordt om de waarde van ATRBAS (F928) te veranderen.
(Beginwaarde: &H1B00)

F3CF-F3D0 GRPPAT
Het beginadres van de sprite-patroontabel in schermmodus 2. Dit adres wordt gebruikt als schermmodus 2 aangeroepen wordt om de waarde van PATBAS (F926) te veranderen.
(Beginwaarde: &H800)

F3D1-F3D2 MLTNAM
Het beginadres van de naamtabel in schermmodus 3. Dit adres wordt gebruikt als schermmodus 3 aangeroepen wordt om de waarde van NAMBAS (F922) te veranderen.
(Beginwaarde: &H0800)

F3D3-F3D4 MLTCOL
Het beginadres van de kleurentabel in schermmodus 3. De kleurentabel is in schermmodus 3 ongebruikt.
(Beginwaarde: &H0000)

F3D5-F3D6 MLTCGP
Het beginadres van de patroontabel in schermmodus 3. Dit adres wordt gebruikt als schermmodus 3 aangeroepen wordt om de waarde van CGPBAS (F924) te veranderen.
(Beginwaarde: &H0000)

F3D7-F3D8 MLTATR
Het beginadres van de sprite-plaatstabel in schermmodus 3. Dit adres wordt gebruikt als schermmodus 3 aangeroepen wordt om de waarde van ATRBAS (F928) te veranderen.
(Beginwaarde: &H1B00)

F3D9-F3DA MLTPAT
Het beginadres van de sprite-patroontabel in schermmodus 3. Dit adres wordt gebruikt als schermmodus 3 aangeroepen wordt om de waarde van PATBAS (F926) te veranderen.
(Beginwaarde: &H3800)

F3DB CLIKSW
Een waarde die aangeeft of er bij het indrukken van een toets een geluid (klik) moet worden gegeven. Mogelijke waarden zijn:

0	geen klik
1	wel een klik

(Beginwaarde: 1)

F3DC CSRY
Het nummer van de regel waarop de cursor zich bevindt, geteld vanaf de bovenste schermregel. De bovenste regel heeft nummer 1.

F3DD CSRX
Het nummer van de kolom waar de cursor zich bevindt, geteld van de meest linkse geldige kolom. De meest linkse kolom heeft nummer 1.

F3DE CNSDFG
Een switch die bepaalt of de definitie van de functietoetsen op de onderste schermregel moet worden getoond. Mogelijke waarden zijn:

0	niet tonen
>0	wel tonen

(Beginwaarde: 1)

F3DF RG0SAV
Voor het opslaan van de waarde van VDP-register 0.

Deze en volgende geheugenplaatsen worden gebruikt om er waarden van VDP-registers in op te bergen. De opgeborgen waarden worden gebruikt wanneer één van de registerwaarden wordt veranderd om alle registers opnieuw van de juiste waarde te voorzien.

F3E0 RGSAV
Voor het opslaan van de waarde van VDP-register 1. Zie RG0SAV (F3DF).

F3E1 RG2SAV
Voor het opslaan van de waarde van VDP-register 2. Zie RG0SAV (F3DF).

F3E2 RG3SAV
Voor het opslaan van de waarde van VDP-register 3. Zie RG0SAV (F3DF).

F3E3 RG4SAV
Voor het opslaan van de waarde van VDP-register 4. Zie RG0SAV (F3DF).

F3E4 RG5SAV
Voor het opslaan van de waarde van VDP-register 5. Zie RG0SAV (F3DF).

F3E5 RG6SAV
Voor het opslaan van de waarde van VDP-register 6. Zie RG0SAV (F3DF).

F3E6 RG7SAV
Voor het opslaan van de waarde van VDP-register 7. Zie RG0SAV (F3DF).

F3E7 STATFL
Laatst uitgelezen waarde van VDP-register 8.

F3E8 TRGFLG
Bevat informatie over de joystick-vuurknoppen en de spatiebalk. De betekenis van de bits van TRGFLG is als volgt:

Bit 7	- stick 2 knop 2 (vuurknop 4)	1=los
Bit 6	- stick 2 knop 1 (vuurknop 2)	1=los
Bit 5	- stick 1 knop 2 (vuurknop 3)	1=los
Bit 4	- stick 1 knop 1 (vuurknop 1)	1=los
Bit 3-1	- ongebruikt	
Bit 0	- spatiebalk (vuurknop 0)	1=los

F3E9 FORCLR
De code voor de standaard voorgrondkleur, die wordt gebruikt bij grafische acties en veranderingen van de schermmodus om de vereiste voorgrondkleur te bepalen. (Beginwaarde: 15)

F3EA BAKCLR
De code voor de standaard achtergrondkleur, die gebruikt wordt bij grafische acties en veranderingen van de schermmodus om de vereiste achtergrondkleur te bepalen.
(Beginwaarde: 4)

F3EB BDRCLR
D code voor de randkleur, die gebruikt wordt bij veranderingen van de schermmodus om de vereiste randkleur te bepalen.
(Beginwaarde: 7)

F3EC-F3EE MAXUPD
Toepassing onbekend.

F3EF-F3F1 MINUPD
Toepassing onbekend.

F3F2 ATRBYT
De code voor de werkkleur. Bij alle grafische acties bevat ATRBYT de kleur waarmee in werkelijkheid getekend wordt. Deze kleur is gelijk aan de voorgrondkleur, tenzij anders gespecificeerd.
(Beginwaarde: 15)

F3F3-F3F4 QUEUES
Beginadres van de queue-tabel. De queue-tabel bevat de beschrijving van 4 queues: 3 geluids-queues en de RS232-queue.
(Beginwaarde: QUETAB (F9F5))

F3F5 FRCNEW
Toepassing onbekend.

F3F6 SCNCNT
Een teller die bijhoudt of bij een VDP-interrupt tevens het toetsenbord gescand moet worden. De teller wordt aangestuurd door de VDP-interrupt. De waarde van SCNCNT loopt van 3 naar 0; als 0 bereikt is, wordt het toetsenbord gescand en wordt SCNCNT weer op 3 gezet.

F3F7 REPCNT
Een teller die bijhoudt of een bepaalde toetscombinatie al zo lang onophoudelijk is ingedrukt dat de toetsen herhalend worden.
De teller loopt van 13 naar 0. REPCNT wordt tegelijk met SCNCNT (F3F6) aangepast, en wel als de ingedrukte toetsen nog dezelfde zijn als bij de vorige toetsenbordscan, oftewel OLDKEY (vanaf FBDA) in zijn geheel gelijk is aan NEWKEY (vanaf FBE5). Zodra deze gebieden niet meer gelijk zijn, wordt REPCNT weer op 13 teruggezet. Wanneer REPCNT 0 bereikt, gaan de ingedrukte toetsen herhalen.
(Beginwaarde: 13)

F3F8-F3F9 PUTPNT

Adres van eerste vrije plaats in de invoerbuffer van het toetsenbord. Elke keer dat een teken aan de invoerbuffer wordt toegevoegd, wordt PUTPNT opgehoogd. Als PUTPNT gelijk is aan GETPNT, is de invoerbuffer vol.
De waarde van PUTPNT ligt altijd binnen het gebied van KEYBUF (FBF0-FC17).

F3FA-F3FB GETPNT

Adres van het eerste teken in de invoerbuffer van het toetsenbord dat nog niet uitgelezen is. Wordt opgehoogd wanneer er een teken uit de invoerbuffer wordt gelezen.
De waarde van GETPNT ligt altijd binnen het gebied van KEYBUF (FBF0-FC17).

F3FC-F405 CS120

Geheugengebied dat wordt gebruikt als bewaarplaats van een paar parameters voor het cassettesysteem. Het gebied valt uiteen in twee gelijkvormige delen: 5 bytes voor de instelling van 1200 baud, en 5 bytes voor 2400 baud.
Instellingen voor een snelheid van 1200 baud:

adres	*betekenis*		*beginwaarde*
F3FC	duur van laag signaal bij uitvoer	0	83
F3FD	duur van hoog signaal bij uitvoer	0	92
F3FE	duur van laag signaal bij uitvoer	1	38
F3FF	duur van hoog signaal bij uitvoer	1	45
F400	duur van synchronisatieblok		15

Instellingen voor een snelheid van 2400 baud:

adres	*betekenis*		*beginwaarde*
F401	duur van laag signaal bij uitvoer	0	37
F402	duur van hoog signaal bij uitvoer	0	45
F403	duur van laag signaal bij uitvoer	1	14
F404	duur van hoog signaal bij uitvoer	1	22
F405	duur van synchronisatieblok		31

F406-F407 LOW

In deze twee bytes staat de instelling van de signaalduur bij uitvoer 0, voor de huidige snelheid van het cassette-systeem. De waarde van LOW is óf gelijk aan de waarde van F3FC-F3FD óf aan de waarde van F401-F402.
(Beginwaarden: 83, 92)

F408-F409 HIGH

In deze twee bytes staat de instelling van de signaalduur bij uitvoer 1, voor de huidige snelheid van het cassette-systeem. De waarde van HIGH is óf gelijk aan de waarde van F3FE-F3FF óf aan de waarde van F403-F404.
(Beginwaarden: 38, 45)

F40A HEADER
De lengte van het synchronisatieblok voor de huidige snelheid van het cassettesysteem. De waarde van HEADER is óf gelijk aan de waarde van F400 óf aan de waarde van F405.
(Beginwaarde: 15)

F40B-F40C ASPCT1
Standaard-instelling van de verhouding hoogte/breedte bij het BASIC-statement CIRCLE. De waarde van F40C wordt niet gebruikt. Als ASPCT2 (F40D) groter is dan &H00FF is de waarde van F40B het aantal horizontale punten per &H100 verticale punten van de straal.
(Beginwaarde: &H0100)

F40D-F40E ASPCT2
Standaard-instelling van de verhouding hoogte/breedte bij het BASIC-statement CIRCLE. Als ASPCT2<&H200 dan is ASPCT 2 het aantal verticale punten per &H100 horizontale punten van de straal.
(Beginwaarde: &H0100)

F40F-F413 ENDPRG
Werkruimte voor het BASIC-statement RESUME. ENDPRG bevat een namaakeinde van het BASIC-programma ten behoeve van RESUME NEXT.

F414 ERRFLG
Foutnummer van de laatst opgetreden fout bij het uitvoeren van een BASIC-programma.
(Beginwaarde: 0)

F415 LPTPOS
Aantal tekens in de uitvoerbuffer van de printer die nog niet geprint zijn.

F416 PRTFLG
Switch die aangeeft of uitvoer naar de printer of naar het scherm gestuurd moet worden. Mogelijke waarden zijn:

0	naar het scherm
1	naar de printer

(Beginwaarde: 0)

F417 NTMSXP
Switch die aangeeft of de aangesloten printer een MSX-printer is of niet. Mogelijke waarden zijn:

0	MSX-printer
1	geen MSX-printer

Als de printer geen MSX-printer is, wordt de code van tekens die niet in de standaard ASCII-tabel staan, veranderd in de code voor een spatie voordat ze naar de printer worden gestuurd.
(Beginwaarde: 0)

F418 RAWPRT
Switch die aangeeft of de uitvoer naar de printer in zgn. 'raw-mode' moet gebeuren, dat wil zeggen dat speciale tekens zoals tabulatorsprongen en speciale MSX-tekens niet eerst omgezet worden in spaties of anderszins aangepast. Mogelijke waarden zijn:

0	geen raw-mode
1	wel raw-mode

N.B. als RAWPRT=1 dan heeft de instelling van NTMSXP (F417) geen effect meer.
(Beginwaarde: 0)

F419-F41A VLZADR
De werkruimte voor de BASIC-functie VAL. Adres van een teken dat door VAL tijdelijk door een O is vervangen.

6.4.3 Werkruimte voor de BASIC-interpreter
De hier volgende geheugenplaatsen worden door de BASIC-interpreter gebruikt om het programmaverloop te besturen. Dat wil zeggen dat het programma ontleed wordt in regels, en de regels in statements, en dat voor het uitvoeren van de statements de juiste parameters worden berekend.

F41B VLZDAT
Werkruimte voor de BASIC-functie VAL. De juiste waarde van het teken dat door VLZADR (F419) wordt aangewezen.

F41C-F41D CURLIN
Regelnummer van de BASIC-regel die net uitgevoerd wordt. In directe modus (als er geen programma uitgevoerd wordt) is CURLIN gelijk aan &HFFFF.
(Beginwaarde: &HFFFF)

F41F-F55C KBUF
Werkgeheugen voor het coderen van regels die in directe modus ingetypt zijn. Dergelijke regels worden opgevat als BASIC-statements of programmaregels. In beide gevallen moet de inhoud van de regel gecodeerd worden volgens de BASIC-codering (zie par. 7.2). De code komt in KBUF te staan.

F55D BUFMIN
Deze geheugenplaats wordt bij INPUT-statements opgevat als eerste byte van BUF (F55E). Zodoende kan een INPUT-statement altijd rekenen op een komma als eerste teken van elk veld van de invoer.
(Beginwaarde: 44 (ASCII-waarde van ','))

F55E-F65F BUF
Wanneer in directe modus of bij een INPUT- of LINE INPUT-statement een RETURN gegeven wordt, komen de ASCII-codes van de tekst van de ingevoerde regel in BUF te staan.

F660 ENDBUF
Laatste bruikbare byte van BUF.

F661 TTYPOS
Positie (nummer van kolom) van het laatste teken dat naar het scherm geschreven is.

F662 DIMFLG
Switch die aangeeft of het opzoeken van een variabele in het rijvariabelengebied al of niet gebeurt ten behoeve van een DIM-statement.

F663 VALTYP
Werkruimte bij het evalueren van uitdrukkingen. VALTYP bevat het type van de laatst geëvalueerde uitdrukking. De waarde van de uitdrukking bevindt zich in DAC (F7F6). Mogelijke waarden van VALTYP zijn:

2	integer
3	string
4	enkele precisie-real
8	dubbele precisie-real

F664 DORES
Werkruimte voor het coderen van BASIC-statements. Switch die aangeeft of sleutelwoorden gecodeerd moeten worden of niet. In het veld van een DATA-statement mogen sleutelwoorden niet gecodeerd worden. Mogelijke waarden van DORES zijn:

0	wel coderen
1	niet coderen

F665 DONUM
Werkruimte voor het coderen van BASIC-statements. Switch die aangeeft of getallen op dit moment gecodeerd moeten worden. Mogelijke waarden van DONUM zijn:

00	getal coderen als constante
01	getal coderen als regelnummer
FF	getal niet coderen

F666-F667 CONTXT
Werkruimte bij het evalueren van uitdrukkingen. CONTXT bevat het adres van het eerste teken na de code van de laatst geëvalueerde constante.

F668 CONSAV
Werkruimte bij het evalueren van uitdrukkingen. CONSAV bevat het informatiebyte dat de coderingswijze aangeeft van de laatst geëvalueerde getalconstante. De waarde van deze constante staat in CONLO (F66A).
Voor de mogelijke waarden van CONSAV en coderingswijzen van constanten zie par. 7.2.1.

F669 CONTYP
Werkruimte bij het evalueren van uitdrukkingen. CONTYP geeft het type van de laatst geëvalueerde getalconstante. De waarde van deze constante staat in CONLO (F66A).
Voor de mogelijke waarden van CONTYP zie VALTYP (F663). Aangezien CONTYP alleen betrekking heeft op getalconstanten, kan de waarde 3 niet voorkomen.

F66A-F671 CONLO
Werkruimte bij het evalueren van uitdrukkingen. CONLO bevat de waarde van de laatst geëvalueerde getalconstante. De waarde begint op adres F66A, en beslaat zoveel bytes als nodig.

F672-F673 MEMSIZ
De bovengrens van de string-ruimte. MEMSIZ bevat het hoogste adres van een geheugenplaats die nog wel gebruikt wordt als string-ruimte.

F674-F675 STKTOP
De bovengrens van de stapel. STKTOP bevat het hoogste adres van een geheugenplaats die nog bij de stapel hoort. Tevens is dit het adres van het eerste byte onder de string-ruimte.

F676-F677 TXTTAB
Beginadres van de code van het huidige BASIC-programma. Wordt bij initialisatie ingesteld en niet door het OS veranderd.
(Beginwaarde: &H8001)

F678-F679 TEMPPT
Adres van de eerste ongebruikte string-descriptor in TEMPST (F67A).
(Beginwaarde; &HF67A (TEMPST))

F67A-F697 TEMPST
Werkruimte voor evaluatie van string-uitdrukkingen. In het gebied van TEMPST is ruimte voor 10 string-descriptoren (van elk 3 bytes). Deze kunnen worden gebruikt voor tijdelijke opslag van string-tussenresultaten.

F698-F69A DSCTMP
Werkruimte voor evaluatie van string-uitdrukkingen. DSCTMP bevat de string-descriptor van het tot nog toe bereikte resultaat.

F69B-F69C FRETOP
Het hoogste adres van een geheugenplaats binnen de string-ruimte die nog vrij is voor string-opslag. Bij het evalueren en opslaan van nieuwe string-waarden wordt FRETOP alleen kleiner; bij een garbage collection wordt FRETOP groter. (Beginwaarde: gelijk aan de waarde van MEMSIZ (F672)).

F69D-F69E TEMP3
Tijdelijke opslagruimte voor administratie van de interpreter.

F69F-F6A0 TEMP8
Tijdelijke opslagruimte voor 'garbage' collection.

F6A1-F6A2 ENDFOR
Adres van het eerste byte in de BASIC-code na het laatst uitgevoerde FOR-statement.

F6A3-F6A4 DATLIN
Nummer van de regel die het laatst onderzocht is op een DATA-statement.

F6A5 SUBFLG
Switch die aangeeft of een variabele ook een element van een rij mag zijn. Dit mag bijvoorbeeld niet bij de lusvariabele van een FOR-statement. SUBFLG heeft als mogelijke waarden:

0	rij toegestaan
1	rij niet toegestaan

F6A6 FLGINP
Switch die aangeeft of er op dit moment een READ- of INPUT-statement wordt uitgevoerd.

F6A7-F6A8 TEMP
Tijdelijke opslagruimte voor administratie van de interpreter.

F6A9 PTRFLG
Switch die aangeeft of er op dit moment nog regelnummer-constanten in de code voor het BASIC-programma staan die omgezet zijn in wijzer-formaat (zie par. 7.2.1). Mogelijke waarden van PTRFLG zijn:

00	geen wijzer-formaten
0D	wel wijzer-formaten

F6AA AUTFLG
Switch die aangeeft of er op dit moment een AUTO-statement wordt uitgevoerd. Mogelijke waarden zijn:

0	geen AUTO-statement
1	wel een AUTO-statement

F6AB-F6AC AUTLIN
Bevat het laatst gegenereerde regelnummer van een AUTO-statement.

F6AD-F6AE AUTINC
Bevat de laatst gebruikte regelnummerverhoging van een AUTO-statement.

F6AF-F6B0 SAVTXT
Werkruimte voor het fout-opvangsysteem. SAVTXT bevat het adres van het eerste byte van het statement dat op dit moment uitgevoerd wordt.

F6B1-F6B2 SAVSTK
Werkruimte voor het fout-opvangsysteem. SAVSTK bevat de waarde die de stapelwijzer had voordat de uitvoering van het huidige statement begon.

F6B3-F6B4 ERRLIN
Bevat het nummer van de regel waar voor het laatst een fout opgetreden is.

F6B5-F6B6 DOT
Bevat het nummer van de laatst behandelde (veranderde, geliste, ingevoegde) regel.

F6B7-F6B8 ERRTXT
Werkruimte voor het fout-opvangsysteem. ERRTXT bevat het adres van het eerste byte van het statement waar het laatst een fout opgetreden is. Wordt bij het optreden van een fout gelijk gesteld aan SAVTXT (F6AF).

F6B9-F6BA ONELIN
Werkruimte voor het fout-opvangsysteem. ONELIN bevat het nummer van de regel waar naar toe moet worden gesprongen bij het optreden van een fout tijdens uitvoering van het huidige BASIC-programma.

F6BB ONEFLG
Werkruimte voor het fout-opvangsysteem. ONEFLG is een switch die aangeeft of de interpreter op dit moment bezig is met een niet afgeronde fout-opvangroutine. Mogelijke waarden zijn:

00	niet ingeschakeld
FF	wel ingeschakeld

F6BC-F6BD TEMP2
Tijdelijke opslagruimte voor administratie van de interpreter.

F6BE-F6BF OLDLIN
Nummer van de regel waar het laatst een programma-onderbreking heeft plaatsgevonden. Wordt op 0 gezet bij een verandering aan het programma.

F6C0-F6C1 OLDTXT
Adres van het eerste statement dat als gevolg van een programma-onderbreking niet is uitgevoerd.

F6C2-F6C3 VARTAB
Beginadres van het opslaggebied voor BASIC-variabelen en functie-descriptoren; (zie par. 7.2.1). Wordt bijgesteld wanneer het BASIC-programma in lengte verandert.

F6C4-F6C5 ARYTAB
Beginadres van het opslaggebied voor rijvariabelen (zie par. 7.2.1). Wordt bijgesteld wanneer het BASIC-programma in lengte verandert, en wanneer de variabelen-opslagruimte vergroot wordt.

F6C6-F6C7 STREND
Adres van het eerste vrije byte, d.w.z het eerste byte dat niet voor opslag van programmacode of variabelen gebruikt wordt.
(Beginwaarde: &H8003)

F6C8-F6C9 DATPTR
Adres van waaraf naar data gezocht wordt bij het volgende READ-statement. Wordt bijgesteld bij een READ- of RESTORE-statement.

F6CA-F6E3 DEFTBL
Een tabel met standaard variabeletypen. DEFTBL heeft 26 elementen. Elk element komt overeen met een letter. Als in een variabelenaam in een BASIC-programma geen type-aanduiding (%, !, #, $) gegeven is, geeft het element van DEFTBL dat overeenkomt met de beginletter van de naam het type van de variabele aan.

Elk element van DEFTBL heeft als mogelijke waarden:

2	integer
3	string
4	enkele precisie-real
8	dubbele precisie-real

F6E4-F6E5 PRMSTK
Werkruimte voor uitvoering van zelf gedefinieerde functies. Adres van het vorige parameterblok op de stapel, ten behoeve van garbage collection.

F6E6-F6E7 PRMLEN
Werkruimte voor uitvoering van zelf gedefinieerde functies. Aantal geldige bytes in het gebied van PARM1 (F6E8).

F6E8-F74B PARM1
Werkruimte voor uitvoering van zelf-gedefinieerde functies. Gebied waarin de actuele definities staan van variabelen die voorkomen in de parameterlijst van de functie die op dit moment geëvalueerd wordt. De manier waarop de definities opgeslagen zijn is precies gelijk aan de opslag van variabelen in het variabelengebied (zie par. 7.2.1).

F74C-F74D PRMPRV
Werkruimte voor uitvoering van zelf gedefinieerde functies. Vorige waarde van PRMSTK (F6E4).

F74E-F74F PRMLN2
Werkruimte voor uitvoering van zelf gedefinieerde functies. Aantal geldige bytes in het gebied van PARM2 (F750).

F750-F7B3 PARM2
Werkruimte voor uitvoering van zelf gedefinieerde functies. Gebied dat gebruikt wordt voor het berekenen van de waarden die in PARM1 (F6E8) komen te staan.

F7B4 PRMFLG
Werkruimte voor uitvoering van zelf gedefinieerde functies. Switch die aangeeft of bij het zoeken naar een variabelenaam het parameterblok in PARM1 (F6E8) al doorzocht is. Mogelijke waarden van PRMFLG zijn:

0	nog niet doorzocht
1	wel doorzocht

F7B5-F7B6 ARYTA2
Adres van het eerste byte waar niet meer gezocht hoeft te worden bij het opzoeken van een variabelenaam. ARYTA2 is gelijk aan de waarde van ARYTAB (F6C4) als het gewone variabelengebied doorzocht wordt, en gelijk aan PARM1 (F6E8) + de waarde van PRMLEN (F6E6) als het parameterblok in PARM1 doorzocht wordt.

F7B7 NOFUNS
Switch die aangeeft of er zich op dit moment een geldig parameterblok in PARM1 (F6E8) bevindt. Mogelijke waarden van NOFUNS zijn:

0	geen geldig blok
1	wel een geldig blok

F7B8-F7B9 TEMP9
Hulpgeheugen voor het doorlopen van parameterblokken op de stapel bij een garbage collection.

F7BA-F7BB FUNACT
Teller van de nestingsdiepte van de functie die op dit moment wordt geëvalueerd.

F7BC-F7C3 SWPTMP
Werkgeheugen bij het uitvoeren van een SWAP-statement. SWPTMP dient als bewaarplaats voor de waarde van de eerste variabele in een SWAP-statement.

F7C4 TRCFLG
Switch die aangeeft of er een TRON werkzaam is (dus de trace-functie is ingeschakeld). Mogelijke waarden van TRCFLG zijn:

0	niet ingeschakeld
>0	wel ingeschakeld

F7C5-F7EF FBUFFR
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F7F0-F7F1 DECTMP
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F7F2-F7F3 DECTM2
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F7F4 DECCNT
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F7F6-F805 DAC
Werkgeheugen bij het uitvoeren van numerieke operatoren. Plaats waar een bereikt tussenresultaat wordt opgeslagen.

Tevens plaats voor parameteroverdracht bij gebruik van een USR-functie in BASIC. Afhankelijk van het type van het meegegeven argument is de waarde in DAC als volgt opgeslagen:

type	*code*	*opslag*
integer	2	waarde in F7F8-F7F9
string	3	adres string-descriptor in F7F8-F7F9
enkele precisie	4	waarde in F7F6-F7F9
dubbele precisie	8	waarde in F7F6-F7FD

Het veld 'code' geeft de codering van het type. Deze codering is opgeslagen in VALTYP (F663).
Bij terugkeer uit de USR-functie (door een RET-instructie) is de opslag van het type en de waarde van het resultaat van de functie precies gelijk aan de opslag van het argument.

F806-F835 HOLD8
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F836-F83D HOLD2
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F83E-F846 HOLD
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F847-F856 ARG
Werkgeheugen bij het uitvoeren van numerieke operatoren.

F857-F85E RNDX
Laatst opgeleverde random-getal in dubbele precisie.

6.4.4 Werkgeheugen voor het file-systeem

De volgende adressen worden gebruikt om allerlei gegevens van het file-systeem op te slaan. Meer informatie hierover is te vinden in par. 6.4.

F85F MAXFIL
Het aantal file-buffers dat momenteel gereserveerd is min één (er blijft altijd een file-buffer gereserveerd voor laden en wegschrijven van files). Tevens het maximum aantal files die tegelijk open mogen zijn. De geldige file-nummers lopen van 1 tot MAXFIL.

F860-F861 FILTAB
Bevat het beginadres van de file-informatietabel. Deze tabel bevat de beginadressen van alle FCB's.

F862-F863 NULBUF
Bevat het beginadres van de eerste file-buffer (de tekst van de FCB van de file met nummer 0).

F864-F865 PTRFIL
Bevat tijdens het uitvoeren van file-invoer/uitvoer het beginadres van de FCB van de file die gebruikt wordt.

F866 RUNFLG
Switch die aangeeft of een file die geladen wordt een BASIC-programma bevat dat direct na het laden uitgevoerd moet worden. Mogelijke waarden van RUNFLG zijn:

00	niet uitvoeren
FF	wel uitvoeren

RUNFLG wordt ook gebruikt als werkruimte bij het uitlezen van aangesloten peddels en bij het inkleuren van een grafisch scherm.
N.B. RUNFLG overlapt het gebied FILNAM (zie verderop) dat de naam van de ingelezen file bevat.

F866-F870 FILNAM
Bevat de naam van de file waarop een of andere actie wordt uitgevoerd. De eerste acht tekens van FILNAM zijn de file-naam, de laatste drie de extensie.

F871-F87B FILNM2
Bevat een tweede file-naam voor acties die twee namen tegelijk nodig hebben, zoals het BASIC-statement NAME.

F87C NLONLY
Switch die aangeeft of er op dit moment een BASIC-programma geladen wordt. Mogelijke waarden zijn:

0	geen BASIC-programma
1	wel een BASIC-programma

F87D-F87E SAVEND
Werkruimte voor BLOAD- en BSAVE-statements. Wanneer er een gedeelte van het gewone geheugen wordt weggeschreven, bevat SAVEND het eindadres van het weggeschreven gedeelte. Als er een gedeelte van het videogeheugen wordt weggeschreven, bevat SAVEND &H4BE8 + het beginadres van het weggeschreven gedeelte.

F87F-F91E FNKSTR
Opslagruimte voor de huidige tekst van de functietoetsen. Per functietoets zijn er 16 bytes beschikbaar. De tekst die bij een functietoets hoort, is gecodeerd volgens de ASCII-tabel, en moet eindigen op een byte met waarde 0.

6.4.5 Werkgeheugen voor schermroutine

In de volgende geheugenplaatsen worden gegevens bewaard over de instelling van de VDP, en is ruimte gereserveerd voor tijdelijke opslag van allerlei parameters voor grafische routines.

F91F-F921 CGPNT

Beginadres van de standaard ASCII-patroontabel. Deze tabel wordt bij elke wisseling van het scherm naar een tekstmodus in de patroontabel van de VDP gezet.
F91F bevat de gleufaanduiding van de gleuf waarin de ASCII-patroontabel zich bevindt; F920-F921 bevat het beginadres van de ASCII-patroontabel.
(Beginwaarde (hexadecimaal):

gleufnummer:	00
beginadres:	1BBF)

F922-F923 NAMBAS

Beginadres van de naamtabel in het videogeheugen. Wordt bij verandering van schermmodus gelijk gesteld aan de waarde van TXTNAM (F3B3), T32NAM (F3BD), GRPNAM (F3C7) of MLTNAM (F3D1), afhankelijk van de nieuwe modus.

F924-F925 CGPBAS

Beginadres van de patroontabel in het videogeheugen. Wordt bij verandering van schermmodus gelijk gesteld aan de waarde van TXTCGP (F3B7), T32CGP (F3C1), GRPCGP (F3CB) of MLTCGP (F3D5), afhankelijk van de nieuwe modus.

F926-F927 PATBAS

Beginadres van de sprite-patroontabel in het videogeheugen. Wordt bij verandering van schermmodus gelijk gesteld aan de waarde van TXTPAT (F3BB), T32PAT (F3C5), GRPPAT F3CF) of MLTPAT (F3D9), afhankelijk van de nieuwe modus.

F928-F929 ATRBAS

Beginadres van de sprite-plaatstabel in het videogeheugen. Wordt bij verandering van schermmodus gelijk gesteld aan de waarde van TXTATR (F3B9), T32ATR (F3C3), GRPATR (F3CD) of MLTATR (F3D7), afhankelijk van de nieuwe modus.

F92A-F92B CLOC

Adres in het videogeheugen van het patroon van het pixel dat overeenkomt met de huidige positie op het grafische scherm. Wordt ingesteld op de linkerbovenhoek bij verandering van schermmodus naar een grafische modus.

F92C CMASK

Masker voor het byte van het videogeheugen dat wordt aangeduid door CLOC (F92A). Selecteert uit dat byte van CLOC de bits van het pixel dat overeenkomt met de huidige positie.

F92D-F92E MINDEL
Werkruimte voor grafische routines.

F92F-F930 MAXDEL
Werkruimte voor grafische routines.

F931-F932 ASPECT
Werkruimte voor het uitvoeren van het CIRCLE-statement. ASPECT is het verhoudingsgetal tussen het aantal punten van de straal in horizontale en verticale richting.

Als ASPECT=&H0100, dan worden ASPCT1 (F40B) en ASPCT2 (F40D) gebruikt voor de verhouding. Als ASPECT<&H0100, dan geeft de waarde van ASPECT het aantal punten in de ene richting, per &H100 punten in de andere richting. Wat de 'ene' en 'andere' richting zijn, wordt in dat geval gegeven door CSCLXY (F941).

F933-F934 CENCNT
Werkruimte voor het uitvoeren van het CIRCLE-statement. Afstand, gemeten in aantal punten vanaf de oorsprong, van het verste eindpunt van het cirkeldeel dat getekend moet worden. De waarde van CENCNT ligt tussen de waarde van CSTCNT (F93F) en de waarde van CNPNTS (F936).

F935 CLINEF
Werkruimte voor het uitvoeren van het CIRCLE-statement. Switch die aangeeft of het begin- en/of eindpunt van de getekende cirkel met het middelpunt moeten worden verbonden. De bits van CLINEF hebben de volgende betekenis:

Bit 7	- eindpunt wel/niet verbinden	1=wel
Bit 6-1	- ongebruikt	
Bit 0	- beginpunt wel/niet verbinden	1=wel

F936-F937 CNPNTS
Werkruimte voor het uitvoeren van het CIRCLE-statement.

F938 CPLOTF
Werkruimte voor het uitvoeren van het CIRCLE-statement. CPLOTF geeft aan of de cirkel getekend moet worden vanaf het punt aangegeven door CSTCNT (F93F) tot het punt aangegeven door CENCNT (F933) dan wel andersom. Mogelijke waarden zijn:

00	van CSTCNT naar CENCNT
FF	van CENCNT naar CSTCNT

F939-F93A CPCNT
Werkruimte voor het uitvoeren van het CIRCLE-statement.

F93B-F93C CPCNT8
Werkruimte voor het uitvoeren van het CIRCLE-statement. Bevat het totale aantal punten in de cirkel (dus niet alleen van het cirkeldeel dat getekend wordt).

F93D-F93E CRCSUM
Werkruimte voor het uitvoeren van het CIRCLE-statement.

F93F-F940 CSTCNT
Werkruimte voor het uitvoeren van het CIRCLE-statement. Afstand, gemeten in aantal punten vanaf de oorsprong, van het dichtstbijzijnde eindpunt van het cirkeldeel dat getekend moet worden. De waarde van CSTCNT ligt tussen 0 en de waarde van CENCNT (F933).

F941 CSCLXY
Werkruimte voor het uitvoeren van het CIRCLE-statement. Switch die aangeeft of de x- dan wel de y-coördinaat omgezet moet worden. Mogelijke waarden van CSCLXY zijn:

0	y-coördinaat
1	x-coördinaat

De waarde van CSCLXY heeft alleen betekenis als ASPECT (F931) ongelijk is aan &H0100.

F942-F943 CSAVEA
Bewaarplaats voor waarde van CLOC (F92A). Wordt ook gebruikt bij inkleuren (BASIC-statement PAINT).

F944 CSAVEM
Bewaarplaats voor waarde van CMASK (F92C). Wordt ook gebruikt bij inkleuren (BASIC-statement PAINT).

F945-F946 CXOFF
Werkruimte voor het uitvoeren van het CIRCLE-statement. Bewaarplaats voor de horizontale afstand tot het middelpunt van de cirkel.

F947-F948 CYOFF
Werkruimte voor het uitvoeren van het CIRCLE-statement. Bewaarplaats voor de verticale afstand tot het middelpunt van de cirkel.

F949 LOHMSK
Werkruimte voor het uitvoeren van het PAINT-statement.

F94A LOHDIR
Werkruimte voor het uitvoeren van het PAINT-statement.

F94B-F94C LOHADR
Werkruimte voor het uitvoeren van het PAINT-statement.

F94D-F94E LOHCNT
Werkruimte voor het uitvoeren van het PAINT-statement.

F94F-F950 SKPCNT
Werkruimte voor het uitvoeren van het PAINT-statement.

F951-F952 MOVCNT
Werkruimte voor het uitvoeren van het PAINT-statement.

F953 PDIREC
Werkruimte voor het uitvoeren van het PAINT-statement.

F954 LFPROG
Werkruimte voor het uitvoeren van het PAINT-statement.

F955 RTPROG
Werkruimte voor het uitvoeren van het PAINT-statement.

F956-F957 MCLTAB
Beginadres van een sprongtabel voor subcommando's die in een string verwerkt zijn.

De BASIC-interpreter gebruikt dezelfde routine voor het ontleden van een string van het PLAY-commando als voor het DRAW-commando. Deze routine maakt gebruik van een sprongtabel waarin elk subcommando door een element vertegenwoordigd is. Aangezien de tabel voor DRAW een andere is dan voor PLAY, wordt in MCLTAB bijgehouden waar de tabel zich bevindt die op dit moment gebruikt wordt.

F958 MCLFLG
Switch die aangeeft of de waarde van MCLTAB (F956) betrekking heeft op de tabel voor het PLAY-statement dan wel van het DRAW-statement. Mogelijke waarden van MCLFLG zijn:

00	DRAW-statement
FF	PLAY-statement

6.4.6 Werkgeheugen voor het queue- en geluids-subsysteem
De hier volgende geheugenplaatsen worden gebruikt om de queues op te slaan, en om allerlei parameters van het geluids-subsysteem te bewaren.

F959-F970 QUETAB
Queue-tabel met informatie over de bestaande queues. Per queue bevat QUETAB zes bytes. De betekenis van deze bytes is als volgt:

bytenr	*betekenis*
0	- offset van de eerste vrije plaats in de queue
1	- offset van het eerste ongelezen byte
2	- vlag die aangeeft of er een byte teruggezet is
3	- aantal plaatsen in de queue
4-5	- beginadres van de queue

Er zijn vier queues: drie voor het geluids-subsysteem (zie par. 6.2.4) en een voor de RS232-interface.

F971-F974 QUEBAK
Tabel van bytewaarden die vooraan in een queue zijn teruggezet.

F975-F9F4 VOICAQ
Queue voor stem A van het geluids-subsysteem. Voor meer details omtrent het geluids-subsysteem zie par. 6.2.4.

F9F5-FA74 VOICBQ
Queue voor stem B van het geluids-subsysteem.

FA75-FAF4 VOICCQ
Queue voor stem C van het geluids-subsysteem.

FAF5-FB34 RS2IQ
Queue voor de RS232. Voor meer details omtrent de RS232 zie par. 5.8.

FB35 PRSCNT
Werkruimte voor het BASIC-statement PLAY. PRSCNT houdt de administratie bij van de strings van een PLAY-statement die ontleed zijn. De bits van PRSCNT hebben de volgende betekenis:

Bit 7	- nog maar 1 keer ontleed	1 = nee
Bit 6-2	- ongebruikt	
Bit 1-0	- aantal ontlede strings	bereik: 0-3

FB36-FB37 SAVSP
Werkruimte voor het BASIC-statement PLAY. Bewaarplaats voor de stapelwijzer tijdens het ontleden van de strings van PLAY.

FB38 VOICEN
Werkruimte voor het BASIC-statement PLAY. Nummer van de stem waarvoor op dit moment een string ontleed wordt. Bereik: 0-2 voor stem A-C.

FB39-FB3A SAVVOL
Werkruimte voor het BASIC-statement PLAY. Bewaarplaats voor het volume van een stem terwijl die stem rust heeft.

FB3B MCLLEN
Lengte van de string die op dit moment onder handen wordt genomen door de string-ontleedroutine van PLAY of DRAW.

FB3C-FB3D MCLPTR
Beginadres van de string die op dit moment onder handen wordt genomen door de string-ontleedroutine van PLAY of DRAW.

FB3E QUEUEN
Tijdelijke bewaarplaats voor het nummer van de geluids-queue die op dit moment bekeken wordt. Bereik: 0-2 voor stem A-C.

FB3F MUSICF
Vlag die aangeeft of er iets te doen is in een van de geluids-queues. De bits van MUSICF hebben de volgende betekenis:

Bit 7-3	- ongebruikt	
Bit 2	- queue 2, stem C	1 = bezig
Bit 1	- queue 1, stem B	1 = bezig
Bit 0	- queue 0, stem A	1 = bezig

FB40 PLYCNT
Teller voor het aantal PLAY-statements waarvan de strings ontleed zijn en in de geluids-queues gezet, maar nog niet uitgevoerd.

FB41-FB65 VCBA
VCB (Voice Control Block) van stem A van de PSG. Wordt gebruikt door het geluidssysteem en door het statement PLAY.

Het gebied van VCBA is onderverdeeld in verschillende deelgebieden, die elk een eigen doel dienen. Deze gebieden zijn:

offset	*naam*	*lengte*	*doel*
0	METREX	2	interrupt-teller
2	VCXLEN	1	MCLLEN voor deze stem
3	VCXPTR	2	MCLPTR voor deze stem
5	VCXSTP	2	stapelwijzer
7	QLENGX	1	aantal bytes voor queue
8	NTICSX	2	nieuwe tellerwaarde
10	TONPRX	2	toonhoogte-instelling
12	AMPLTX	1	amplitude-instelling

13	ENVPRX	2	snelheid van volumeverloop
15	OCTAVX	1	octaaf-instelling
16	NOTELX	1	nootlengte-instelling
17	TEMPOX	1	tempo-instelling
18	VOLUMX	1	volume
19	ENVLPX	1	vorm van volumeverloop
33	MCLSTX		ruimte om stapel te bewaren
36	MCLSEX		beginstapel

De stapel waarvan hierboven gesproken wordt, is de plaats waar de bytewaarden worden klaargemaakt die op de queue van deze stem gezet gaan worden.

FB66-FB8A VCBB
VCB (Voice Control Block) van stem B van de PSG. Wordt gebruikt door het geluidssysteem en door het statement PLAY. De indeling van VCBB is net als die van VCBA.

FB8B-FBAF VCBC
VCB (Voice Control Block) van stem C van de PSG. Wordt gebruikt door het geluidssysteem en door het statement PLAY. De indeling van VCBC is net als die van VCBA.

6.4.7 Instellingen voor de scherm-editor en het interrupt-systeem

Door middel van allerlei schakelaars kunnen er een aantal mogelijkheden in- of uitgeschakeld worden. In de nu volgende geheugenplaatsen wordt de huidige stand van een aantal schakelaars van de scherm-editor en het interrupt-systeem bijgehouden.

FBB0 ENSTOP
Switch die aangeeft of de mogelijkheid van de software-reset ingeschakeld is. Mogelijke waarden van ENSTOP zijn:

0	niet ingeschakeld
1	ingeschakeld

Als de software-reset ingeschakeld is, heeft het tegelijk indrukken van de CTRL-, SHIFT-, CODE- en GRAPH-toetsen tot gevolg dat BASIC opnieuw opgestart wordt. Het huidige BASIC-programma gaat hierdoor niet verloren.
Beginwaarde: 0

FBB1 BASROM
Switch die aangeeft of het huidige BASIC-programma in ROM staat. Mogelijke waarden zijn:

0	niet in ROM
1	wel in ROM

FBB2-FBC9 LINTTB
Tabel die per regel van het scherm onthoudt of de tekst op die regel overloopt in de volgende regel. Deze informatie wordt gebruikt in de scherm-editor.
Elk byte in LINTAB correspondeert met een schermregel. Elk van de bytes heeft als mogelijke waarden:

0	loopt over
>0	loopt niet over

FBCA-FBCB FSTPOS
Bewaarplaats voor de coördinaten van de cursor op het moment dat de OS-routines INLIN (ingang 00B1) en QINLIN (ingang 00B4) de besturing overdragen aan de scherm-editor. De waarde op dat moment van CSRY (F3DC) komt in FBCA te staan en de waarde van CSRX (F3DD) in FBCB.

FBCC CODSAV
ASCII-code van het teken dat op dit moment bedekt wordt door de cursor.

FBCD FNKSWI
Switch die aangeeft welke van de functietoetsen op het scherm worden getoond. Mogelijke waarden van FNKSWI zijn:

0	functietoetsen 6 t/m 10
1	functietoetsen 1 t/m 5

Overigens is de waarde van FNKSWI alleen geldig als het tonen van functietoetsen ingeschakeld is d.m.v. CNSDFG (F3DE).

FBCE-FBD7 FNKFLG
Tabel van switches die aangeven of voor een functietoets de interrupt-faciliteit is ingeschakeld. Per functietoets is er een switch. Elk van de switches heeft als mogelijke waarden:

0	interrupt uitgeschakeld
1	interrupt ingeschakeld

FBD8 ONGSBF
Teller van het aantal interrupts die gegenereerd zijn voor een of andere gebeurtenis, maar waarop nog niet gereageerd is d.m.v. een ON-interrupt GOSUB-statement.

FBD9 CLIKFL
Vlag die aangeeft of er voor de laatste toetsaanslag al een klik gegenereerd is, om te voorkomen dat voor een toets die twee ASCII-codes genereert tweemaal een klik gegeven wordt. Mogelijke waarden zijn:

00	geen klik
0F	wel een klik

FBDA-FBE4 OLDKEY
Gebied waarin de oude toestand van de toetsenbordmatrix wordt bewaard, ten behoeven van het zich herhalen van lang ingedrukte toetscombinaties. Als voor een voldoende lange tijd de toestand van OLDKEY gelijk is aan die van NEWKEY (FBE5) dan gaan de toetsen zich herhalen.

FBE5-FBEF NEWKEY
De huidige toestand van de toetsenbordmatrix. Elk bit in het gebied van NEWKEY komt overeen met een toets: als het bit 1 is, is de corresponderende toets ingedrukt, anders is de toets losgelaten.

FBF0-FC17KEYBUF
Toetsenbordbuffer. Elk teken dat via het toetsenbord is ingevoerd, komt vanzelf in deze buffer te staan. De buffer wordt uitgelezen door de OS-routine CHGET. Elke keer dat invoer van het toetsenbord verwacht wordt, wordt deze routine aangeroepen.

FC18-FC3F LINWRK
Werkgeheugen waarin de laatst ingetypte regel wordt verwerkt.

FC40-FC47 PATWRK
Bewaarplaats voor een patroon van een ASCII-teken. Wordt gebruikt bij het schrijven van een ASCII-teken naar een scherm in grafische modus.

FC48-FC49 BOTTOM
Bevat het laagste adres van het RAM-geheugen. Dit adres wordt ingesteld bij initialisatie en daarna niet meer veranderd.

FC4A-FC4B HIMEM
Bevat het hoogste adres van het RAM-geheugen dat niet door het OS gereserveerd is. String-ruimte, file-buffers en de stapel bevinden zich nog onder dit adres. Dit adres wordt ingesteld bij initialisatie en daarna niet meer veranderd.

FC4C-FC99 TRPTBL
Tabel met gegevens ten behoeve van de interrupt-faciliteiten van de MSX-computer. De elementen van TRPTBL zijn drie bytes groot. Er is een element voor elke mogelijke interrupt.
Het eerste byte van elk element van TRPTBL geeft de huidige instelling van de interrupt-faciliteit behorende bij dat element. De betekenis van de bits van dat byte zijn:

Bit 7-3	- ongebruikt	
Bit 2	- interrupt opgetreden ja/nee	1 = ja
Bit 1	- interrupt STOP ja/nee	1 = ja
Bit 0	- interrupt OFF ja/nee	1 = nee

De volgende twee bytes van elk element van TRPTBL bevatten het beginadres van de regel in het BASIC-programma waarnaar toe moet worden gesprongen in geval dat de interrupt optreedt.

De tabel heeft een capaciteit van 26 elementen. Het verband tussen het elementnummer en de bijbehorende interrupt is als volgt:

elementnr.	*beginadr*	*interrupt*
0	FC4C	functietoets 1
1	FC4F	functietoets 2
2	FC52	functietoets 3
3	FC55	functietoets 4
4	FC58	functietoets 5
5	FC5B	functietoets 6
6	FC5E	functietoets 7
7	FC61	functietoets 8
8	FC64	functietoets 9
9	FC67	functietoets 10
10	FC6A	STOP-toets
11	FC6D	sprite-botsing
12	FC70	spatiebalk (vuurknop 0)
13	FC73	joystick 1 knop 1 (vuurknop 1)
14	FC76	joystick 2 knop 1 (vuurknop 2)
15	FC79	joystick 1 knop 2 (vuurknop 3)
16	FC7C	joystick 2 knop 2 (vuurknop 4)
17	FC7F	interval

De elementen met nummers 18-25 zijn in het huidige MSX-systeem ongebruikt.

FC9A RTYCNT
Toepassing onbekend.

FC9B INTFLG
Byte waarin wordt bijgehouden of de STOP-toets is ingedrukt, eventueel samen met de CTRL-toets. INTFLG kan de volgende waarden hebben:

0	niets ingedrukt
3	CTRL+STOP ingedrukt
4	STOP ingedrukt

Het gevolg van het indrukken van STOP of CTRL+STOP hangt sterk af van het moment waarop dat gebeurt, en de instelling van het interrupt-systeem. Zie ook de OS-ingang ISCNTC (00BA), par. 6.3.

FC9C PADY
Bewaarplaats voor de laatst uitgelezen y-coördinaat van een touch pad.

FC9D PADX
Bewaarplaats voor de laatst uitgelezen x-coördinaat van een touch pad.

FC9E-FC9F JIFFY
Waarde van de software-klok, die bij elke interrupt van de VDP met één wordt opgehoogd.

FCA0-FCA1 INTVAL
Beginwaarde voor INTCNT (FCA2), de teller van de interval-interrupt-faciliteit. Wordt gebruikt om de teller opnieuw te initialiseren wanneer deze de waarde 0 heeft bereikt.

FCA2-FCA3 INTCNT
Teller van de interval-interrupt-faciliteit. Wordt verlaagd bij elke VDP-interrupt. Wordt op de waarde van INTVAL (FCA0) teruggezet zodra 0 bereikt is. Als de interrupt-faciliteit is ingeschakeld, wordt tevens een interrupt gegeven zodra 0 bereikt is.

FCA4 LOWLIM
Parameter voor invoer van het cassettesysteem. Wordt van een waarde voorzien bij het inlezen van een kopblok van cassette.

FCA5 WINWID
Parameter voor invoer van het cassettesysteem. Wordt van een waarde voorzien bij het inlezen van een kopblok van cassette.

FCA6 GRPHED
Vlag die aangeeft of het vorige teken dat naar het scherm geschreven werd, het extensieteken voor grafische tekens was (ASCII-code 1). Mogelijke waarden:

0	was geen extensieteken
1	was extensieteken

FCA7 ESCCNT
Vlag die aangeeft of het vorige teken dat naar het scherm geschreven werd, het escape-teken was (ASCII-code 27). Mogelijke waarden:

00	was geen escape-teken
FF	was escape-teken

FCA8 INSFLG
Switch die aangeeft of tekens die naar het scherm geschreven worden, tussen de andere tekens op de regel gevoegd moeten worden, of de andere tekens moeten overschrijven. Deze switch wordt alleen gebruikt door de scherm-editor.

INSFLG heeft als mogelijke waarden:

00	overschrijven
FF	tussenvoegen

Bij het indrukken van de INS-toets wordt de instelling van INSFLG omgedraaid.

FCA9 CSRSW
Switch die aangeeft of de cursor op het scherm getoond moet worden of niet. CSRSW heeft als mogelijke waarden:

0	niet tonen
1	wel tonen

De instelling van CSRSW kan ook worden verzorgd met de escape-sequences 'x5' en 'y5'.

FCAA CSTYLE
Switch die aangeeft in welke vorm de cursor wordt getoond. Er zijn twee mogelijke vormen: blok-cursor en onderstreep-cursor. Mogelijke waarden van CSTYLE zijn:

0	blok-cursor
1	onderstreep-cursor

CSTYLE heeft alleen effect als de cursor inderdaad getoond wordt; dit is afhankelijk van de instelling van CSRSW (FCA9).

Door het indrukken van de INS-toets wordt de instelling van CSTYLE omgedraaid. Bovendien kan de instelling van CSTYLE worden verzorgd met de escape-sequences 'x4' en 'y4'.

FCAB CAPST
Switch die aangeeft of de CAPS-LOCK actief is. Mogelijke waarden van CAPST zijn:

00	niet actief
FF	wel actief

Door het indrukken van de CAPS-toets wordt de instelling van CAPST omgedraaid.

FCAC KANAST
Wordt gebruikt om het indrukken van de zogenaamde 'dode toets' te registreren. De dode toets is de toets op het toetsenbord zonder tekens erop. Deze toets kan gebruikt worden om geaccentueerde versies van klinkers te bereiken.
Wanneer de dode toets wordt ingedrukt, eventueel gecombineerd met de SHIFT- of CODE-toets, krijgt KANAST een waarde ongelijk aan 0. Als direct daarna een klinker wordt ingetypt (eventueel een hoofdletter), wordt van die klinker de geaccentueerde versie genomen, mits deze in de ASCII-tabel van de MSX-computer voorkomt.

Bij het indrukken van elke andere toets dan de dode toets wordt KANAST weer op 0 gezet.
KANAST kan de volgende waarden hebben:

waarde	*betekenis*	*versie*
0	geen dode toets	normaal
1	dode toets	accent grave
2	SHIFT + dode toets	accent aigu
3	CODE + dode toets	accent circumflex
4	SHIFT + CODE + dode toets	trema

Zie ook par. 6.2.6.

N.B. In de Japanse versie van de MSX-computer wordt KANAST gebruikt om de status van de KANA-LOCK in op te bergen; dit is een omschakeling naar een ander soort letterteken.

FCAD KANAMD
Wordt niet gebruikt in niet-Japanse versies van de MSX-computer. In de Japanse versie bepaalt de waarde van KANAMD het lettertype.
Beginwaarde: &H40

6.4.8 Allerlei
De nu volgende geheugenplaatsen horen in feite thuis in categorie n die al eerder aan de orde geweest zijn.

FCAE FLBMEM
Switch die tijdens het laden van een file aangeeft of de file een BASIC-programma is, zodat het huidige programma uit het geheugen verwijderd moet worden. FLBMEM heeft als mogelijke waarden:

0	is een BASIC-programma
>0	is geen BASIC-programma

FCAF SCRMOD
De huidige schermmodus. Geldige waarden: 0-3.

FCB0 OLDSCR
De tekst-schermmodus waarin het scherm automatisch wordt teruggezet na afloop van grafische acties. Geldige waarden: 0, 1.

FCB1 CASPRV
Wordt gebruikt bij het lezen van een invoer-file van cassette. CASPRV is de plaats waar eerst gekeken wordt voordat een teken uit de buffer gelezen wordt. Als CASPRV geen 0 bevat, wordt de inhoud ervan als eerste byte van de invoer beschouwd.

FCB2 BRDATR
Code voor de grenskleur bij een PAINT-statement. Wordt in schermmodus 2 altijd gelijk gemaakt aan de kleur waarmee getekend wordt, bewaard in ATRBYT (F3F2).

FCB3-FCB4 GXPOS
Bewaarplaats voor de x-coördinaat van de laatst bereikte plaats op het grafische scherm bij het uitvoeren van een grafisch commando. Wordt dikwijls gebruikt als startpunt voor een volgend commando.

FCB5-FCB6 GYPOS
Bewaarplaats voor de y-coördinaat van de laatst bereikte plaats op het grafische scherm bij het uitvoeren van een grafisch commando. Wordt dikwijls gebruikt als startpunt voor een volgend commando.

FCB7-FCB8 GRPACX
Werkruimte waarin bij het uitvoeren van grafische commando's de x-coördinaat samengesteld wordt.

FCB9-FCBA GRPACY
Werkruimte waarin bij het uitvoeren van grafische commando's de y-coördinaat samengesteld wordt.

FCBB DRWFLG
Werkruimte voor het uitvoeren van een DRAW-statement. Switch die informatie geeft over de manier waarop het volgende DRAW-subcommando uitgevoerd moet worden. De bits van DRWFLG hebben de volgende betekenis:

Bit 7	- lijn tekenen ja/nee	1 = nee
Bit 6	- cursor bewegen ja/nee	1 = nee
Bit 5-0	- ongebruikt	

Bit 7 wordt ingesteld met het grafische subcommando N; bit 6 wordt ingesteld met B.

FCBC DRWSCL
Werkruimte voor het uitvoeren van een DRAW-statement. Schaalverhouding tussen opgegeven en werkelijke lengte. Bij het uitvoeren van DRAW-subcommando's worden alle opgegeven lengtes vermenigvuldigd met DRWSCL/4. Als DRWSCL=0, dan wordt de lengte niet omgezet (hetzelfde effect als DRWSCL=4). DRWSCL wordt ingesteld met het grafische subcommando S.

FCBD DRWANG
Werkruimte voor het uitvoeren van een DRAW-statement. Rotatiehoek voor alle DRAW-subcommando's waarin een richting is opgegeven. De richting wordt geroteerd om een hoek die afhankelijk is van de waarde van DRWANG. Mogelijke waarden zijn:

0	geen rotatie
1	90° rotatie
2	180° rotatie
3	270° rotatie

De rotaties zijn opgegeven tegen de klok in.
DRWANG wordt ingesteld met het grafische subcommando A.

FCBE RUNBNF
Werkruimte voor het uitvoeren van een BLOAD-statement. Switch die aangeeft of na het uitvoeren van een BLOAD-statement de inhoud van de geladen file moet worden beschouwd als een machinetaalprogramma dat direct na het laden aangeroepen moet worden. RUNBF kan als waarden hebben:

0	niet uitvoeren
>0	wel uitvoeren

Het uitvoeren gebeurt door het executie-adres aan te roepen dat met de file mee is weggeschreven.

FCBF-FCC0 SAVENT
Werkruimte voor de BASIC-statements BLOAD en BSAVE. Als er een gedeelte van het videogeheugen wordt ingelezen of weggeschreven, bevat SAVENT de lengte in bytes van het betreffende gedeelte. Als er een gedeelte van het gewone geheugen wordt geladen of weggeschreven, bevat SAVENT het beginadres van het betreffende gedeelte.

6.4.9 Opslag van gleufinformatie
De volgende geheugenplaatsen worden gebruikt om het gleufsysteem te beheren. Daartoe worden allerlei instellingen van het gleufsysteem opgeslagen, en wordt per gleuf werkgeheugen gereserveerd.

FCC1-FCC4 EXPTBL
Aanduiding welke van de primaire gleuven uitgebreid is. Per primaire gleuf is een byte aanwezig, met mogelijke waarden (hexadecimaal):

00	niet uitgebreid
80	wel uitgebreid

Het verband tussen geheugenadres van het byte en nummer van de bijbehorende primaire gleuf is als volgt:

nummer	*adres*
0	FCC1
1	FCC2
2	FCC3
3	FCC4

FCC5-FCC8 SLTTBL

Aanduiding per primaire gleuf welke secundaire gleuf op het moment geselecteerd is, aangenomen dat de primaire gleuf uitgebreid is.
Per primaire gleuf is een byte aanwezig; de waarde daarvan is het nummer van de geselecteerde secundaire gleuf. De waarden van de bytes zijn alleen geldig als het overeenkomstige byte van EXPTBL (FCC1) als waarde &H80 heeft.
Het verband tussen geheugenadres van het byte en nummer van de bijbehorende primaire gleuf is als volgt:

nummer	*adres*
0	FCC5
1	FCC6
2	FCC7
3	FCC8

FCC9-FD08 SLTATR

Aanduiding van de functie van elke aanwezige pagina in het systeem.
Per mogelijke pagina is een byte gereserveerd. De bits van deze bytes hebben de volgende betekenis:

Bit 7	- BASIC-programma ja/nee	1 = ja
Bit 6	- apparaat-uitbreiding ja/nee	1 = ja
Bit 5	- statement-uitbreiding ja/nee	1 = ja
Bit 4-0	- ongebruikt	

Er zijn 4 primaire gleuven met maximaal 4 secundaire gleuven met elk maximaal 4 pagina's, totaal 4×4×4=64 pagina's. Het adres van het byte behorende bij een zekere pagina volgt uit de formule:

adres=&HFCC9+16xprim. gleufnr+4xsec. gleufnr+pagnr

Als de primaire gleuf niet uitgebreid is, moet voor het secundaire gleufnummer 0 worden ingevuld.

Het paginanummer volgt uit het adresbereik van de pagina volgens de volgende tabel:

adresbereik	*pagnr*
0000-3FFF	0
4000-7FFF	1
8000-BFFF	2
C000-FFFF	3

FD09-FD88 SLTWRK

Per mogelijk geïnstalleerde pagina zijn twee bytes werkgeheugen gereserveerd. Het gebruik van dit geheugen is geheel vrij. Het adres van het werkgeheugen van een zekere pagina volgt uit de volgende formule:

adres=&HFD09+32xprim. gleufnr+8xsec. gleufnr+2xpagnr

Voor deze formule gelden dezelfde opmerkingen als bij SLTATR (FCC9).

6.4.10 Parameter-opslag voor aanroepen van ROM-pagina's

Bij het aanroepen van ROM-pagina's die uitbreidingen voor de MSX bevatten – hetzij apparaatuitbreidingen, hetzij statement-uitbreidingen – moeten in enkele gevallen parameters worden overgedragen. Voor dit doel zijn de volgende geheugenplaatsen gereserveerd.

FD89-FD98 PROCNM

Opslagplaats voor statement-naam of apparaatnaam bij het aanroepen van een ROM-pagina die een statement-uitbreiding of een apparaatuitbreiding bevat.

FD99 DEVICE

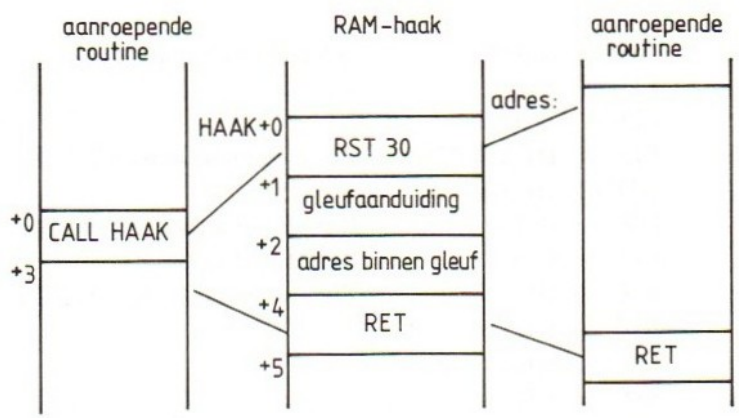
Opslagplaats voor het apparaatnummer van het bedoelde apparaat bij het aanroepen van een ROM-pagina die een apparaatuitbreiding bevat. Geldige waarden: 0-3.

6.5 RAM-haken

Naast werkgeheugen wordt er door het OS ook geheugen gereserveerd voor de zgn. RAM-haken (Engels: RAM-hooks). Deze RAM-haken vormen een uitbreidingsmogelijkheid voor MSX BASIC en het OS. Ze worden zogezegd gebruikt om nieuwe toepassingen aan de MSX 'vast te haken'.

Een RAM-haak is een gebied van vijf bytes, gereserveerd voor een zeer korte machinetaalroutine die over het algemeen zal bestaan uit een sprong naar een bepaald adres waar een afhandelingsroutine staat. Elke RAM-haak wordt aangeroepen op bepaalde momenten (verschillend per haak), waarop het mogelijk is het OS te onderscheppen en bepaalde zaken zelf anders te implementeren.
Over het algemeen zal deze eigen implementatie verzorgd zijn in een pagina in een andere gleuf. De vijf bytes die per haak beschikbaar zijn, zijn dan net voldoende voor een intergleuf-aanroep van de eigen routine via de OS-ingang CALLF (0030). Deze ingang kan namelijk aangeroepen worden door een RST &H30-instructie, gevolgd door één byte voor een gleufaanduiding en twee bytes voor een geheugenadres (zie ook par. 6.3). Het laatste byte is dan een RET-instructie.

Bij het initialiseren van het RAM-geheugen worden alle RAM-haken door het OS gevuld met vijf RET-instructies, zodat het aanroepen van een haak geen effect heeft. Als een bepaalde routine in een bepaalde ROM geïmplementeerd is, kan de initialisatieroutine van die ROM (zie par 6.2.1) de benodigde RAM-haken vullen met een aanroep van CALLF die naar de implementatie verwijst.



Afb. 6.3. Het gebruik van de RAM-haken

Het systeem van gebruik van de RAM-haken is nog eens weergegeven in afbeelding 6.3.

Om een CALLF-intergleuf-aanroep te kunnen gebruiken, moet de gleuf van be-

stemming bekend zijn. Voordat een RAM-haak kan worden ingevuld, moet de ROM-initialisatieroutine weten in welke gleuf hij zich bevindt. De gleuf van bestemming is als byte gecodeerd, waarin de bits de volgende betekenis hebben:

Bit 7	gleuf is wel/niet uitgebreid	1 = wel
Bit 6-4	ongebruikt	
Bit 3-2	nummer van secundaire gleuf	bereik: 0-3
Bit 1-0	nummer van primaire gleuf	bereik: 0-3

De volgende machinetaalroutine bepaalt de gleuf waarin hij zich bevindt, en codeert deze informatie volgens bovenstaand patroon. Bij verlaten van deze routine staat de codering in het A-register.

```
RSLREG  EQU     &H138           ;OS-ingang voor bepalen van
                                ;primaire gleuf
EXPTBL  EQU     FCC1            ;Tabel met informatie of gleuf
                                ;uitgebreid is
B8000   EQU     1               ;Zet dit op 1 als de routine zich
                                ;bevindt in bereik 8000-BFFF

        CALL    RSLREG          ;Ingeschakelde primaire gleuven
        RRC                     ;Zoek juiste element op binnen
        RRC                     ;de ingeschakelde gleuven
IF      B8000                   ;Macro, wordt alleen geassembleerd
        RRC                     ;als routine in bereik 8000-BFFF
        RRC                     ;Neem in dat geval volgende element
ENDIF

        AND     &B11            ;Selecteer relevante deel
        LD      C,A
        LD      B,0
        LD      HL,EXPTBL       ;Zoek element binnen EXPTBL op
        ADD     HL,BC
        OR      (HL)            ;Bit 7 hoog indien uitgebreid
        LD      C,A             ;Berg tijdelijk op
        INC     HL              ;Zoek element binnen SLTTBL op
        INC     HL
        INC     HL
        INC     HL
        LD      A,(HL)          ;Huidig ingeschakelde secundaire gleuf
IF      B8000                   ;Macro, geassembleerd als B8000=1
        RRC                     ;Schuif juiste element in bit 2,3
        RRC
ENDIF   AND     &B1100          ;Selecteer relevante deel
        OR      C               ;Combineer met reeds gevonden
                                ;informatie
        RET                     ;Klaar
```

De RAM-haken zijn ruwweg in drie groepen in te delen:
- OS RAM-haken. Dit zijn RAM-haken die vanuit het OS aangeroepen worden.
- BASIC-uitbreiding RAM-haken. Dit zijn haken die horen bij een BASIC sleutelwoord dat in standaard MSX BASIC niet geïmplementeerd is. Met gebruik van deze haken kunnen deze sleutelwoorden benut worden in de implementatie van DISK BASIC.
- DISK BASIC aanroepen. Deze RAM-haken dienen enkel en alleen voor het implementeren van DISK BASIC. De RAM-haken in deze categorie horen echter niet speciaal bij een sleutelwoord, zoals de hierboven genoemde haken.
- BASIC interpreter RAM-haken. Dit zijn haken die vanuit de BASIC interpreter aangeroepen worden. De momenten en omstandigheden van aanroep zijn over het algemeen nogal obscuur, en deze haken zijn in feite niet bruikbaar zonder grondige kennis van de interpreter.

Van alle RAM-haken wordt de indeling in een van deze groepen vermeld. Van de OS RAM-haken is bovendien het moment van aanroep en het doel van de haak gegeven. Van de BASIC uitbreidingshaken is vermeld bij welk sleutelwoord ze horen. De BASIC-interpreter-haken en DISK BASIC-haken zijn over het algemeen verder niet gedocumenteerd.

FD9A H.KEYI

Groep	OS
Aanroep	aan het begin van de interrupt-afhandelingsroutine, voordat getest is of de interrupt van de VDP afkomstig is.
Doel	om andere interrupts dan de VDP-interrupt te kunnen toelaten, bijvoorbeeld via een cartridge-aansluiting. De RS232 maakt hiervan gebruik. De routine moet zelf testen of de interrupt wel van de juiste bron afkomstig is.

FD9F H.TIMI

Groep	OS
Aanroep	aan het begin van de interrupt-afhandelingsroutine, op het moment dat vast staat dat de interrupt van de VDP afkomstig is.
Doel	om de VDP-interrupt voor andere doeleinden te kunnen gebruiken.

FDA4 H.CHPU

Groep	OS
Aanroep	aan het begin van CHPUT (00A2), de routine die een teken op het scherm zet.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDA9 H.DSPC

Groep	OS
Aanroep	aan het begin van de routine die de cursor op het scherm zichtbaar maakt.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDAE H.ERAC

Groep	OS
Aanroep	aan het begin van de routine die de cursor van het scherm verwijdert.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDB3 H.DSPF

Groep	OS
Aanroep	aan het begin van DSPFNK (00CF), de routine die de tekst van de functietoetsen op het scherm zet.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDB8 H.ERAF

Groep	OS
Aanroep	aan het begin van ERAFNK (00CC), de routine die de tekst van de functietoetsen van het scherm verwijdert.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDBD H.TOTE

Groep	OS
Aanroep	aan het begin van TOTEXT (00D2), de routine die het scherm geforceerd in een tekstmodus zet.
Doel	om andere uitvoerapparaten dan het beeldscherm als standaard uitvoer te kunnen gebruiken.

FDC2 H.CHGE

Groep	OS
Aanroep	aan het begin van CHGET (009F), de routine die een teken uit de toetsenbordbuffer leest.
Doel	om andere invoerapparaten dan het toetsenbord als standaard invoer te kunnen gebruiken.

FDC7 H.INIP

Groep	OS
Aanroep	aan het begin van de OS-routine die bij initialisatie van een tekstmodus de patroontabel vult met de patronen van standaard (ASCII-)tekens.
Doel	om andere tekens dan die in de ASCII-tabel als standaard te gebruiken.

FDCC H.KEYC

Groep	OS
Aanroep	aan het begin van de routine die samengestelde toetsaanslagen (de SHIFT-, CODE-, GRAPH- en/of CTRL-toets samen met een andere) ontcijfert en er een ASCII-code bij zoekt, die in de toetsenbordbuffer wordt gezet.
Doel	om andere toetsenbordconstructies mogelijk te maken.

FDD1 H.KYEA

Groep	OS
Aanroep	aan het begin van de routine die enkelvoudige toetsaanslagen verwerkt en er een ASCII-code aan toewijst die in de toetsenbordbuffer wordt gezet.
Doel	om andere toetsenbordconstructies mogelijk te maken.

FDD6 H.NMI

Groep	OS
Aanroep	vanaf ingang NMI (0066).
Doel	om het gebruik van NMI-interrupts mogelijk te maken.

FDDB H.PINL

Groep	OS
Aanroep	aan het begin van PINLIN (00AE), de routine die de versie van de scherm-editor vertegenwoordigt die in directe programmeermodus gebruikt wordt.
Doel	om een andere implementatie van de scherm-editor of een ander standaard invoerapparaat mogelijk te maken.

FDE0 H.QINL

Groep	OS
Aanroep	aan het begin van QINLIN (00B4), de routine die na het afdrukken van een vraagteken en een spatie de scherm-editor aanroept.
Doel	om een andere implementatie van de scherm-editor of een ander standaard invoerapparaat mogelijk te maken.

FDE5 H.INLI

Groep	OS
Aanroep	aan het begin van INLIN (00B1), de routine waarin de scherm-editor geïmplementeerd is.
Doel	om een andere implementatie van de scherm-editor of een ander standaard invoerapparaat mogelijk te maken.

FDEA H.ONGO

Groep	BASIC-interpreter.

FDEF H.DSKO

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie DSKO$.

FDF4 H.SETS

Groep	BASIC-sleutelwoord.
Doel	het implementeren van een SET-statement.

FDF9 H.NAME

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het NAME-statement.

FDFE H.KILL

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het KILL-statement.

FE03 H.IPL

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het IPL-statement.

FE08 H.COPY

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het COPY-statement.

FE0D H.CMD

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het CMD-statement.

FE12 H.DSKF

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie DSKF.

FE17 H.DSKI

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het DSKI$-statement.

FE1C H.ATTR

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie ATTR$.

FE21 H.LSET

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het LSET-statement.

FE26 H.RSET

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het RSET-statement.

FE2B H.FIEL

Groep	BASIC-sleutelwoord.
Doel	het implementeren van het FIELD-statement.

FE30H.MKI$

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie MKI$.

FE35 H.MKS$

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie MKS$.

FE3A H.MKD$

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie MKD$.

FE3F H.CVI

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie CVI.

FE44 H.CVS

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie CVS.

FE49 H.CVD

Groep	BASIC-sleutelwoord.
Doel	het implementeren van de functie CVD.

FE4E H.GETP

Groep	DISK-BASIC aanroep.
Aanroep	bij het opzoeken van de waarde van PTRFIL (F864).

FE53 H.SETF

Groep	DISK-BASIC aanroep.
Aanroep	wanneer aan PTRFIL (F864) een nieuwe waarde wordt gegeven.

FE58 H.NOFO

Groep	DISK-BASIC aanroep.
Aanroep	bij een OPEN-statement zonder FOR-gedeelte.

FE5D H.NULO

Groep	DISK-BASIC aanroep.
Aanroep	bij een operatie via file-buffer 0.

FE62 H.NTFL

Groep	DISK-BASIC aanroep.
Aanroep	bij een operatie via een file-buffer met een ander nummer dan 0.

FE67 H.MERG

Groep	DISK-BASIC aanroep.
Aanroep	bij het uitvoeren van een MERGE-statement voor diskette.

FE6C H.SAVE

Groep	DISK-BASIC aanroep.
Aanroep	bij het uitvoeren van een SAVE-statement voor diskette.

FE71 H.BINS

Groep	DISK-BASIC aanroep.
Aanroep	bij het uitvoeren van een BSAVE-statement voor diskette.

FE76 H.BINL

Groep	DIS-BASIC aanroep.
Aanroep	bij het uitvoeren van een BLOAD-statement voor diskette.

FE7B H.FILE

Groep	BASIC-uitbreiding.
Doel	het implementeren van het FILES-statement.

FE80 H.DGET

Groep	DISK-BASIC-aanroep.

FE85 H.FILO

Groep	DISK-BASIC-aanroep.

FE8A H.INDS

Groep	DISK-BASIC-aanroep.

FE8F H.RSLF

Groep	DISK-BASIC-aanroep.
Aanroep	voor het opnieuw selecteren van het vorige diskettestation voor diskette-I/O.

FE94 H.SAVD

Groep	DISK BASIC aanroep.
Aanroep	voor het onthouden van het huidig geselecteerde diskettestation.

FE99 H.LOC

Groep	BASIC-sleutelwoord.
Doel	om de functie LOC te implementeren.

FE9E H.LOF

Groep	BASIC-sleutelwoord.
Doel	om de functie LOF te implementeren.

FEA3 H.EOF

Groep	DISK BASIC-aanroep.
Aanroep	bij het uitvoeren van de functie EOF voor diskette.

FEA8 H.FPOS

Groep	BASIC-sleutelwoord.
Doel	om de functie FPOS te implementeren.

FEAD H.BAKU

Groep	DISK-BASIC-aanroep.

FEB2 H.PARD

Groep	BASIC-interpreter.
Aanroep	bij het ontleden van een apparaatnaam in een file-aanduiding.

FEB7 H.NODE

Groep	BASIC-interpreter.
Aanroep	wanneer een file-aanduiding zonder apparaatnaam wordt aangetroffen.

FEBC H.POSD

Groep	DISK-BASIC-aanroep.

FEC1 H.DEVN

Groep	BASIC-interpreter.
Aanroep	bij het opzoeken van een apparaat met een bepaalde naam.

FEC6 H.GEND

Groep	BASIC-interpreter.

FECB H.RUNC

Groep	BASIC-interpreter.
Aanroep	bij het terugzetten van variabele waarden tijdens het voorbereidende werk in een RUN-statement.

FED0 H.CLEA

Groep	BASIC-interpreter.
Aanroep	bij het uitvoeren van een CLEAR-statement.

FED5 H.LOPD

Groep	BASIC-interpreter.

FEDA H.STKE

Groep	BASIC-interpreter.

FEDF H.ISFL

Groep	OS
Aanroep	aan het begin van de routine ISFLIO (014A) waarin getest wordt of er op dit moment file-I/O plaatsvindt.
Doel	om nieuwe in- en uitvoerapparaten te implementeren.

FEE4 H.OUTD

Groep	OS
Aanroep	aan het begin van de routine OUTDO (0018) die een teken wegschrijft naar scherm, printer of ander uitvoerapparaat.
Doel	om nieuwe uitvoerapparaten te implementeren.

FEE9 H.CRDO

Groep	BASIC-interpreter.

FEEE H.DSKC

Groep	BASIC-interpreter.

FEF3 H.DOGR

Groep	BASIC-interpreter.
Aanroep	vóór het uitvoeren van elke grafische actie.

FEF8 H.PRGE

Groep	BASIC-interpreter.
Aanroep	bij het beëindigen van een BASIC-programma.

FEFD H.ERRP

Groep	BASIC-interpreter.

FF02 H.ERRF

Groep	BASIC-interpreter.

FF07 H.READ

Groep	BASIC-interpreter.

FF0C H.MAIN

Groep	BASIC-interpreter.

FF11 H.DIRD

Groep	BASIC-interpreter.
Aanroep	bij het uitvoeren van een statement in directe modus.

FF16 H.FINI

Groep	BASIC-interpreter.

FF1B H.FINE

Groep BASIC-interpreter.

FF20 H.CRUN

Groep BASIC-interpreter.
Aanroep bij het coderen van een zojuist ingetypt BASIC-statement.

FF25 H.CRUS

Groep BASIC-interpreter.
Aanroep bij het coderen van een zojuist ingetypt BASIC-statement.

FF2A H.ISRE

Groep BASIC-interpreter.
Aanroep wanneer een sleutelwoord is gevonden bij het coderen van een zojuist ingetypt BASIC-statement.

FF2F H.NTFN

Groep BASIC-interpreter.
Aanroep wanneer een functie is gevonden bij het coderen van een zojuist ingetypt BASIC-statement.

FF34 H.NOTR

Groep BASIC-interpreter.
Aanroep wanneer een niet-sleutelwoord is gevonden bij het coderen van een zojuist ingetypt BASIC-statement.

FF39 H.SNGF

Groep BASIC-interpreter.

FF3E H.NEWS

Groep BASIC-interpreter.

FF43 H.GONE

Groep BASIC-interpreter.

FF48 H.CHRG

Groep BASIC-interpreter.

FF4D H.RETU

Groep BASIC-interpreter.

FF52 H.PRTF

Groep BASIC-interpreter.

FF57 H.COMP

Groep BASIC-interpreter.

FF5C H.FINP

Groep BASIC-interpreter.

FF61 H.TRMN

Groep BASIC-interpreter.

FF66 H.FRME

Groep BASIC-interpreter.

FF6B H.NTPL

Groep BASIC-interpreter.

FF70 H.EVAL

Groep BASIC-interpreter.

Aanroep bij het berekenen van de waarde van een uitdrukking in een BASIC-statement.

FF75 H.OKNO

Groep BASIC-interpreter.

FF7A H.FING

Groep BASIC-interpreter.

FF7F H.ISMI

Groep BASIC-interpreter.

Aanroep bij een toewijzing van een waarde aan een substring (zie par. 7.3. MID$).

FF84 H.WIDT

Groep BASIC-interpreter.

Aanroep bij het uitvoeren van een WIDTH-statement.

FF89 H.LIST

Groep BASIC-interpreter.

Aanroep bij het uitvoeren van een LIST-statement.

FF8E H.BUFL

Groep BASIC-interpreter.

FF93 H.FRQI

Groep BASIC-interpreter.

FF98 H.SCNE

Groep	BASIC-interpreter.

FF9D H.FRET

Groep	BASIC-interpreter.

FFA2 H.PTRG

Groep	BASIC-interpreter.
Aanroep	bij het opzoeken van een variabele.

FFA7 H.PHYD

Groep	OS
Aanroep	vanuit de routine PHYDIO (0144), die operaties moet uitvoeren op opslagapparaten zoals een diskettestation.
Doel	om PHYDIO te implementeren.

FFAC H.FORM

Groep	OS
Aanroep	vanuit de routine FORMAT (0147) die opslagapparaten zoals een diskettestation moet initialiseren.
Doel	om FORMAT te implementeren.

FFB1 H.ERRO

Groep	BASIC-interpreter.
Aanroep	vanuit de fout-afhandelingsroutine.

FFB6 H.LPTO

Groep	OS
Aanroep	aan het begin van LPTOUT (00A5), de routine die een teken naar de printer stuurt.
Doel	om speciale printers te kunnen aansturen (eventueel met andersoortige aansluitingen).

FFBB H.LPTS

Groep	OS
Aanroep	aan het begin van LPTSTT (00A8), de routine die de instelling van de printer onderzoekt.
Doel	om speciale printers te kunnen aansturen (eventueel met andersoortige aansluitingen).

FFC0 H.SCRE

Groep	BASIC-interpreter.
Aanroep	bij het uitvoeren van het SCREEN-statement.

FFC5 H.PLAY

Groep	BASIC-interpreter.
Aanroep	bij het uitvoeren van het PLAY-statement.

FFCA H.BEXT

Groep	OS
Doel	het installeren van apparaat-uitbreidingen die extra OS-routines bevatten. (Zie par. 6.6.2).

6.6 File-systeem

Het OS van de MSX-computer is bij lange na niet een volwaardig besturingssysteem. Daarvoor ontbreken teveel elementen. In de officiële MSX-computer documentatie wordt wat in dit boek OS wordt genoemd, dan ook aangeduid met BIOS: BASIC Input Output System (BASIC in- en uitvoersysteem).

Het belangrijkste element dat ontbreekt in het OS is een behoorlijk file-afhandelingssysteem (Engels: filing system). Het enige dat het OS op het gebied van files biedt zijn een paar cassetteroutines op laag niveau. (Zie par. 6.2.5.)

Het begrip 'file' bestaat echter wel degelijk in het MSX-systeem, aangezien er in BASIC volop van gebruik wordt gemaakt. De routines die het gebruik van files mogelijk maken zijn echter ondergebracht in het programma van de BASIC-interpreter in plaats van in het OS.
Omdat geen bespreking van een OS volledig is zonder melding van het file-systeem, zelfs als dit er niet is, zullen we in deze paragraaf ingaan op de manier waarop files door de BASIC-interpreter worden behandeld.

6.6.1 Files: algemeen

In principe is een file een verzameling waarden, gerepresenteerd door bytes. Deze waarden worden bij elkaar bewaard, en zijn alleen bereikbaar door middel van file-operaties. Files worden dan ook over het algemeen gebruikt om bij elkaar horende gegevens (gerepresenteerd door middel van bytewaarden) ook bij elkaar te houden.

Een file heeft een specificatie. Deze specificatie is een unieke aanduiding van de file. Door het toepassen van de file-specificatie kan de file worden opgezocht, en kunnen er operaties op worden uitgevoerd.

Aan een file kunnen dan de volgende elementen worden onderscheiden:

- de file-specificatie: een aanduiding voor de file met behulp waarvan deze gevonden kan worden.
- de inhoud van de file: de waarden die binnen de file gegroepeerd zijn tot een logisch geheel.
- een opslagplaats of -apparaat voor de file, waar de inhoud en andere file-gegevens bewaard worden.

- besturingsgegevens, zoals de lengte (het aantal waarden) van de file, de plaats waar deze waarden te vinden zijn, de manier waarop de waarden gecodeerd zijn, en allerlei andere gegevens over de file die van pas kunnen komen bij het uitvoeren van file-operaties.

De file-specificatie

Zoals gezegd vormt de file-specificatie de aanduiding waarmee een file kan worden opgezocht en zich onderscheidt van andere files.
Voor het opzoeken van de file is het onder meer nodig dat men weet met welk apparaat de file verbonden is. Voorbeelden van mogelijke apparaten zijn: cassetterecorder, diskettestation, maar ook printer en toetsenbord. Hierop komen we terug in par 6.6.2.

De file-specificatie bevat dus als onderdeel een aanduiding van een apparaat, de apparaatnaam. Een file-specificatie heeft de volgende vorm:

apparaatnaam:file-naam

Hierin is apparaatnaam de aanduiding van het apparaat; file-naam is een naam die eventueel nodig is om de file te onderscheiden van andere files die met hetzelfde apparaat verbonden zijn. Dit is niet op alle apparaten van toepassing.

File-operaties

File-operaties zijn bewerkingen die worden uitgevoerd op een file of op de inhoud van de file. Deze bewerkingen lopen uiteen van het hernoemen van een file – waarbij de naam van de file dus veranderd wordt – tot het opzoeken van een bepaalde waarde binnen de file.

Waarden die in een file staan, kunnen alleen worden bereikt door middel van file-operaties. Dit is zowel een voordeel als een nadeel. Een voordeel omdat de gegevens dan niet zomaar verloren kunnen gaan, aangezien daarvoor een file-operatie nodig is; een nadeel omdat de file-operaties soms een omslachtige en langzame manier vormen om eenvoudige bewerkingen uit te voeren.
Om dit nadeel te verminderen, zijn er vele manieren bedacht voor het handig opslaan van de waarden waaruit een file bestaat, en zijn er vele soorten file-operaties ontwikkeld.

File-operaties vallen uiteen in twee groepen: operaties die een file als een geheel beschouwen en daarmee een bewerking verrichten – bijvoorbeeld het hernoemen van de file, het kopiëren van de file, het vernietigen van de file – en operaties die toegang geven tot de inhoud van de file. Deze tweede soort operaties zullen we nader toelichten.
Alvorens de inhoud van een file bewerkt kan worden, moet de file eerst aan het systeem gemeld worden. Dit gebeurt door het openen van de file. Het openen houdt in dat de file wordt opgezocht, en dat er geheugenruimte wordt gereserveerd om besturingsgegevens over de file te bewaren.

Bij het openen wordt opgegeven op welke manier de inhoud van de file behandeld moet worden. Deze manier van behandelen bepaalt welke file-operaties na het openen op de file uitgevoerd mogen worden.

De mogelijke manieren voor het openen van een file zijn:
- sequentiële invoer: de waarden in de file worden gelezen in de volgorde waarin ze in de file staan.
- sequentiële uitvoer: naar de file worden waarden geschreven, die in de file komen te staan in de volgorde van schrijven.
- random access (vertaling: willekeurig toegankelijk): de inhoud van de file is geordend in de vorm van blokken gegevens met dezelfde lengte, records genaamd; deze blokken kunnen in willekeurige volgorde worden ingelezen, of er kunnen blokken worden veranderd of toegevoegd.

De file-operaties die na het openen van een file op de inhoud ervan kunnen worden uitgevoerd, zijn in te delen in twee categorieën: lezen en schrijven. Met het lezen van een file wordt bedoeld dat een waarde uit de file wordt gehaald; met schrijven wordt bedoeld dat er een waarde in de file wordt veranderd of dat er een waarde aan wordt toegevoegd.
Op een sequentiële invoer-file kunnen geen schrijf-operaties worden uitgevoerd; op een sequentiële uitvoer-file geen leesoperaties.
Wanneer men klaar is met het bewerken van de inhoud van een file, wordt de file weer gesloten. Deze operatie is tegengesteld aan het openen van de file. De geheugenruimte die bij het openen gereserveerd was wordt weer vrijgegeven. Na het sluiten van de file is voor het systeem de toestand weer gelijk aan voor het openen van de file, behalve dat de file-inhoud veranderd kan zijn.

6.6.2 Apparaatuitbreiding

In het hierna volgende bedoelen we met 'apparaat' één van de invoer- of uitvoerapparaten die in het MSX-systeem kunnen worden gebruikt in file-operaties.
Welke operaties er op een file kunnen worden uitgevoerd, hangt sterk af van het apparaat waarmee de file verbonden is. Men kan een indeling van apparaten maken in opslag- en niet-opslagapparaten. Opslagapparaten zetten de waarden van de files die eraan verbonden zijn op een of ander medium (bijvoorbeeld cassette of diskette). Niet-opslagapparaten beschouwen hun files als eenmalig, en verzenden de waarden (RS232, apparaatnaam COM) of zetten ze als uitvoer op het scherm (apparaatnaam CGP of CRT) of de printer (apparaatnaam LPT).

Opslagapparaten kunnen altijd gebruikt worden voor in- én uitvoer; in ieder geval sequentieel, en in het geval van diskette ook nog random access. Niet-opslagapparaten kunnen in ieder geval alleen maar voor sequentiële file-operaties gebruikt worden, en in de meeste gevallen maar één kant op.
Een apparaat kan óf standaard aan de MSX bekend zijn, óf toegevoegd zijn via een ROM-pagina met een apparaat-uitbreiding (zie par. 6.2.1). Standaardapparaten worden geïmplementeerd door routines in de BASIC interpreter. De namen van deze apparaten zijn:

CAS	de cassetterecorder
LPT	de printer
CRT	het tekstscherm
CGP	het grafische scherm

De vaste apparaatnaam is CAS; als er in een file-specificatie geen apparaatnaam voorkomt, wordt CAS gekozen.
Als een apparaat is geïmplementeerd in een ROM-pagina, verzorgt die pagina de routines voor het gebruik van het apparaat in file-operaties. Elke pagina kan routines voor maximaal vier apparaten bevatten. Deze apparaten worden genummerd van 0 t/m 3.
Er zijn twee manieren om de routines van een apparaat in een ROM-pagina te bereiken: via de DEVICE-ingang van de pagina, en via de RAM-haak H.BEXT (FFCA).

De DEVICE-ingang
Een pagina kan worden aangeroepen via het adres in DEVICE met een waarde in register A die aangeeft wat voor soort actie er ondernomen moet worden. Deze waarde is een van de volgende lijst:

00	open een file
02	sluit een file
04	doe random I/O
06	doe sequentiële uitvoer
08	doe sequentiële invoer
0A	voer de functie LOC uit
0C	voer de functie LOF uit
0E	voer de functie EOF uit
10	voer de functie FPOS uit
12	ga een teken terug
FF	geef het nummer dat bij de naam hoort

Als A kleiner is dan &HFF staat het nummer van de apparaatnaam in het geheugenadres DEVICE (FD99). Als A=&HFF dan staat de tekst van de apparaatnaam in PROCNM (vanaf FD89). In dat geval moet bij het verlaten van de routine het nummer van het apparaat in A staan, en de C-vlag laag zijn. Als de naam niet bekend is, moet de C-vlag bij het verlaten hoog zijn.

Eventuele andere in- of uitvoerparameters, zoals te lezen of te schrijven tekens, zijn te vinden in het geheugengebied van de BASIC-interpreter: het type staat in VALTYP (F663), de waarde in DAC (vanaf F7F6). Zie par. 6.4. voor meer details.

De RAM-haak H.BEXT
Een tweede manier om toegang te krijgen tot de routines in een pagina met een apparaatuitbreiding is het aanroepen van de RAM-haak H.BEXT (FFCA) met in D het apparaatnummer. Voor het verkrijgen van dit nummer zie verderop. In E staat

een getal dat de gewenste reactie van de apparaatuitbreiding aanduidt. De betekenis van de verschillende waarden van E verschilt per apparaat.
Een apparaatuitbreiding in een ROM-pagina moet er bij initialisatie (aanroep via de ingang INIT, zie par 6.3) zorg voor dragen dat een aanroep van H.BEXT in deze pagina belandt via een netjes geneste serie van aanroepen.
Bij een aanroep van H.BEXT zijn de registers op de volgende manier gedefinieerd:

B – gleufnummer van resultaattabel
D – apparaatnummer
E – aanduiding gewenste functie
HL – beginadres van resultaattabel

Het apparaatnummer in D is een getal dat uniek moet zijn voor elke geïnstalleerde apparaatuitbreiding. Dit apparaatnummer wordt gevonden door H.BEXT aan te roepen met D=0 en E=0 (zie verderop).
De aanduiding van de gewenste functie in E is een waarde die wordt geïnterpreteerd door de apparaatuitbreiding waar het apparaatnummer in D bij hoort. De betekenis van de waarden van E hangt af van de apparaatuitbreiding, behalve dat voor E=0 een sprongtabel naar routines in de ROM-pagina moet worden opgeleverd.
De resultaattabel die wordt aangeduid door B en HL is een gebied in het RAM-geheugen dat door de apparaatuitbreiding kan worden gebruikt om eventueel opgeleverde waarden op te slaan. B bevat de gleufaanduiding, gecodeerd op de volgende wijze:

Bit 7	- gleuf wel/niet uitgebreid	1 = wel
Bit 6-4	- ongebruikt	
Bit 3-2	- secundair gleufnummer	bereik: 0-3
Bit 1-0	- primair gleufnummer	bereik: 0-3

HL bevat het adres van de eerste geheugenplaats die gebruikt mag worden binnen de gleuf van B. Na terugkeer uit H.BEXT bevat HL het eerste adres na de opgeleverde waarde.
Bij aanroep van H.BEXT met D=00 of D=FF moet elke apparaat-uitbreiding op een voorgeschreven manier reageren. In de volgende tabel zijn de voorgeschreven reacties vermeld:

D=	*E=*	*reactie*
00	0	lever apparaatnummer
00	1	reserveer elementen van TRBTBL
00	2	zet interrupts af
00	3	zet interrupts aan
FF	0	lever adres van sprongtabel

D=00, E=0: lever apparaatnummer. Elke aangesloten apparaat-uitbreiding moet in de resultaattabel twee bytes achterlaten, met de volgende betekenis:

Byte 0 - apparaatnummer
Byte 1 - niet gedefinieerd

Het apparaatnummer in byte 0 is het nummer waardoor de apparaat-uitbreiding geïdentificeerd wordt. Wanneer H.BEXT later wordt aangeroepen met in D dit nummer, is de aanroep bestemd voor die apparaatuitbreiding.

D=00, E=1: reserveer elementen in TRPTBL (vanaf FC4C). Dit is de tabel waarin alle software-interrupts geregistreerd worden. (zie par 6.2.2). Bij aanroep staat in A het aantal elementen van TRPTBL dat tot nu toe gebruikt is. Bij terugkeer moet dit aantal opgehoogd zijn met het aantal dat de apparaatuitbreiding wil gebruiken. De resultaattabel wordt niet gebruikt.

D=00, E=2: zet interrupts af. Wordt aangeroepen vlak voor een DI-instructie. Als de apparaatuitbreiding interrupts genereert, wordt hij in staat gesteld dit achterwege te laten voor de tijdsduur dat ze toch niet door de processor worden opgemerkt. De resultaattabel wordt niet gebruikt.

D=00, E=3: zet interrupts aan. Het tegengestelde van D=00, E=2.

D=FF, E=0: levert sprongtabel op. In de resultaattabel komen de volgende waarden te staan:

Byte 0 - gleufaanduiding
Byte 1-2 - adres sprongtabel
Byte 3 - fabrikant-identificatienummer
Byte 4 - ongebruikt

De gleufaanduiding: aanduiding van de gleuf waarin de apparaatuitbreiding zich bevindt, gecodeerd zoals hiervoor aangegeven.

Adres sprongtabel: adres van tabel met beginadressen van routines die in de apparaatuitbreiding geïmplementeerd zijn.
Fabrikant-identificatienummer: een nummer dat uniek is per fabrikant van MSX-onderdelen. Fabrikant-identificatienummers kunnen worden aangevraagd bij Microsoft.

6.6.3 BASIC-file-systeem

In deze paragraaf komt het files-systeem aan de orde dat in de BASIC-interpreter is ingebouwd en vanuit BASIC gebruikt kan worden. Alleen de file-operaties die werken op de inhoud van een file worden in ogenschouw genomen.

Bij het openen van een file wordt aan de hand van de file-specificatie de file opgezocht. De file-specificatie bestaat uit een apparaatnaam en/of een file-naam; zie par. 66.1. Met behulp van de apparaatnaam wordt het apparaat opgezocht volgens de methode die in par. 6.6.2 is uitgelegd. Het opzoeken van de file aan de hand van de file-naam wordt aan het apparaat overgelaten.
De structuur waarmee in het BASIC-file-systeem een file geassocieerd wordt wanneer deze geopend is, heet file-buffer. Elke file krijgt bij het openen een file-buffer toegewezen.

Er wordt een tabel bijgehouden van de bestaande file-buffers. Het beginadres van deze tabel staat in FILTAB (F860), het aantal elementen van de tabel in MAXFIL (F85F). De tabel bestaat uit beginadressen van file-buffers. Elke file-buffer heeft een nummer dat gelijk is aan de index in FILTAB. De eerste file-buffer in de tabel heeft nummer 0. Dit nummer wordt in veel file-operaties gebruikt om de buffer aan te duiden.
De structuur van een file-buffer bestaat uit 265 bytes. De eerste 9 bytes bevatten besturingsinformatie, de volgende 256 bytes bevatten een kopie van een gedeelte van de file-inhoud. De precieze indeling van een file-buffer is:

offset	*naam*	*functie*
0	FL.MOD	manier van openen van file
1-2	FL.FCA	adres van FCB indien file op diskette staat
3	FL.LSA	teruggezet teken
4	FL.DSK	apparaatnummer
5	FL.SLB	niet gedocumenteerd
6	FL.BPS	positie in bufferruimte
7	FL.FLG	allerlei instellingen
8	FL.OPS	positie van leeskop
9 e.v	FL.BUF	256 bytes opslagruimte

FL.MOD: geeft de manier aan waarop de file geopend is (zie par. 6.6.1). Het byte kan de volgende waarden aannemen:

0 niet geopend
1 geopend als sequentiële invoer-file
2 geopend als sequentiële uitvoer-file
4 geopend als random access file

FL.FCA: beginadres van het FCB (File Control Block = file-besturingsblok) dat bij de file hoort indien de file op diskette staat. Dit FCB wordt bijgehouden door DISK BASIC. We gaan hier verder niet op in.

FL.LSA: op sommige apparaten bestaat er een operatie 'schuif terug' die het mogelijk maakt een inleesactie van een sequentiële invoer-file ongedaan te maken. Het laatst ingelezen en 'teruggezette' byte wordt dan hier bewaard.

FL.DSK: het identificerende nummer van het apparaat waarmee de file verbonden is; zie (par. 6.6.2). Bestaande apparaatnummers zijn:

00 standaard apparaat (geen apparaatnaam)
01 diskettestation A:
02 diskettestation B:
FC grafisch scherm GRP:
FD tekstscherm CRT:
FE printer LPT:
FF cassetterecorder CAS:

FL.BPS: de plaats in de bufferruimte (vanaf FL.BUF, zie verderop) waar de laatste file-operatie opgehouden is. Deze waarde wordt alleen gebruikt bij sequentiële files. De plaats wordt opgeslagen als offset t.o.v. het begin van de bufferruimte.

FL.BUF: het eerste byte van de bufferruimte van de file-buffer. Deze bufferruimte kan worden gebruikt om een kopie te bewaren van dat gedeelte van de file-inhoud waar op een bepaald moment operaties op worden uitgevoerd. Als de bufferruimte wordt gebruikt – dit is niet voor alle apparaten het geval – wordt daarbij de volgende methode toegepast:

Bij sequentiële invoer wordt de inhoud van de file telkens in stukken van 256 bytes in de bufferruimte gezet; vanuit daar wordt de ingelezen waarde opgeleverd. De positie in de buffer waar men met inlezen is gekomen wordt bewaard in FL.BPS (zie hiervoor). Als deze waarde dreigt de 255 te overschrijden, wordt het volgende stuk van de inhoud van de file ingelezen.

Bij sequentiële uitvoer worden de geschreven waarden in de bufferruimte verzameld en naar de file geschreven zodra er 256 bytes bij elkaar zijn geschreven.

Bij random access-files wordt telkens de inhoud van één record van de file in de buffer geschreven, of andersom.

File-buffer 0 speelt in de MSX een speciale rol. Deze buffer wordt namelijk gebruikt bij het inlezen en wegschrijven van hele programma-files – dit in tegenstelling tot gegevens-files die binnen een programma als in – of uitvoer gebruikt worden.
Vanwege de speciale plaats die file-buffer 0 inneemt, wordt het adres ervan bijgehouden in een daarvoor gereserveerde geheugenplaats: NULBUF (F862).

Wanneer er een file gebruikt wordt voor in- of uitvoer wordt het adres van de buffer van deze file opgeslagen in PTRFIL (F864). Wanneer er geen file-operatie aan de gang is, is de waarde van PTRFIL gelijk aan 0. De waarde van deze geheugenplaats wordt tevens gebruikt om te testen of er een file-operatie gaande is, namelijk in de OS-routine ISFLIO (ingang 014A).

6.6.4 MSX-DOS

Wie een MSX-computer heeft die van een diskettestation en 64kbyte RAM is voorzien, heeft de mogelijkheid om MSX DOS aan te schaffen. Dit is een programma dat op diskette aangeleverd wordt en gebruik maakt van de 32kbyte extra geheugen om een volwaardig OS op de MSX te implementeren
Het programma dat MSX-DOS is, komt terecht in de 32kbyte RAM met hetzelfde adresbereik als de oorspronkelijke OS- en BASIC-ROM's: 0000-7FFF.

MSX DOS vormt een 'laag' tussen de computer – met name het diskettestation – en de BASIC-interpreter. BASIC wordt aangeroepen als programma vanuit DOS. Ook andere programma's kunnen op deze manier aangeroepen worden. De gebruiker van MSX DOS hoeft zich niet te bekommeren om de organisatie van files of afzonderlijke file-operaties, aangezien deze door DOS worden verzorgd.

Anderzijds kunnen BASIC- en machinetaal-programma's gebruik maken van routines die in DOS geïmplementeerd zijn. Deze routines hebben over het algemeen betrekking op files: DOS biedt een aantal routines voor file-operaties die door elke machinetaal-programmeur aangeroepen kunnen worden. Dit in tegenstelling tot de situatie in een systeem zonder MSX DOS, waar de routines van de file-operaties verborgen zijn in de BASIC interpreter.
Een gedetailleerde beschrijving van de routines van MSX DOS valt buiten het bereik van dit boek.

7. BASIC

In dit hoofdstuk komt de taal BASIC aan de orde. Eerst wordt de plaats van BASIC binnen het MSX-systeem bepaald; vervolgens wordt uitgelegd hoe een BASIC-programma er op machinetaalniveau uitziet; ten slotte worden alle BASIC sleutelwoorden besproken.

7.1 Wat is BASIC?

Natuurlijk, BASIC is een computertaal waarmee op een betrekkelijk eenvoudige manier op een MSX-huiscomputer geprogrammeerd kan worden. Daar gaat het in deze paragraaf niet om. Het gaat om de plaats van de taal BASIC binnen het MSX-systeem. Hoe kunnen we BASIC inpassen in het geheel van apparatuur (processor, geheugen, randapparaten) en programmatuur (het OS) dat in de voorgaande hoofdstukken aan de orde is geweest?

Het antwoord is, dat BASIC in feite zelf een programma is, namelijk het programma van de BASIC-interpreter (of vertolker; we zullen hier alleen het meer bekende woord interpreter gebruiken). Deze interpreter is in machinetaal geschreven en bevindt zich, samen met het OS, in het ROM-geheugen dat standaard met de computer wordt mee geleverd.
De interpreter behandelt een BASIC-programma als invoer die aangeeft wat voor acties ondernomen moeten worden. Hoe deze invoer eruit ziet wordt beschreven in par. 7.2.

Wanneer er op een MSX-computer een BASIC-programma wordt geschreven, gebeurt dat met behulp van een speciale routine, de scherm-editor, besproken in par. 6.2.7. Deze routine levert de tekst van ingevoerde regels als resultaat op. De interpreter bekijkt de ingevoerde regel, en codeert hem volgens de codering in par. 7.2. Daarna wordt de regel indien mogelijk uitgevoerd, of, als de regel genummerd was, toegevoegd aan het programma in het geheugen.
Bij het uitvoeren van ingetypte commando's, of van regels uit een programma, wordt dikwijls gebruik gemaakt van OS-routines. Dit gebeurt dan door middel van het aanroepen van een van de OS-ingangen die in het vorige hoofdstuk aan de orde zijn geweest. De manier waarop dat gebeurt, en het moment waarop het gebeurt, blijven echter voor de BASIC-programmeur verborgen. Deze verborgenheid wordt vaak aangeduid met het woord 'transparantheid': het OS is transparant voor de BASIC-gebruiker, hij kan het niet zien.

Over het algemeen is transparantheid een positieve zaak, omdat het het leven van een programmeur aanzienlijk kan vereenvoudigen. De keerzijde van de medaille is echter, dat veel BASIC-programmeurs als gevolg van deze situatie in het geheel niet op de hoogte zijn van het bestaan van het OS. Dit kan leiden tot een verminderd begrip van de werking en mogelijkheden van de computer.
Het is mogelijk om met behoud van het OS een nieuwe BASIC-interpreter te implementeren, of een vertaler voor een taal als PASCAL. Anderzijds is het mogelijk de BASIC onveranderd te laten, en het OS te veranderen. Dit laatste gebeurt bijvoorbeeld als men MSX-DOS aanschaft.

7.2 Opslagmethoden van BASIC

Als een BASIC-programma wordt ingetypt of geladen van cassette of diskette, wordt het in het geheugen opgeslagen. Deze opslag gebeurt volgens bepaalde regels, die zo opgesteld zijn dat het programma enerzijds niet te veel opslagruimte vergt, en anderzijds niet te langzaam wordt uitgevoerd. Deze twee vereisten zijn meestal tegenstrijdig.

Bij het uitvoeren van een BASIC-programma wordt er geheugenruimte gebruikt voor het opslaan van variabelen, functies, tussenresultaten en wat dies meer zij. De opslag hiervan is eveneens een compromis tussen zuinigheid en snelheid van gebruik.
Voor opslag en uitvoering van een BASIC-programma kan het RAM-geheugen in

opslag plaats grens adres		gebruik geheugengebied
naam	adres	
BOTTOM	FC48	ongebruikt
TXTTAB	F676	BASIC-programmacode
VARTAB	F6C2	enkelvoudige variabelen en functie-descriptoren
ARYTAB	F6C4	rijvariabelen
STREND STKTOP	F6C6 F674	vrij geheugen ; wordt ook gebruikt voor de stapel
MEMSIZ	F672	string-constanten
		ongebruikt
FILTAB	F860	file-buffers
HIMEM	FC4A	systeem-werkgeheugen

Afb. 7.1. Indeling van het RAM-geheugen door BASIC

het adresgebied 8000-FFFF worden gebruikt, uitgezonderd dat gedeelte dat door het OS in beslag wordt genomen. In afbeelding 7.1 is een overzicht van het RAM-geheugen gegeven, waarin is aangegeven hoe het geheugen opgedeeld is.

In deze paragraaf wordt de codering van een BASIC-programma en de bijbehorende variabelen uitgelegd. Deze informatie kan zeer van pas komen bij het implementeren van machinetaal-procedures die moeten samenwerken met een BASIC-programma, of bij het uitbuiten van sommige mogelijkheden van BASIC.

7.2.1 Opslag van variabelen

Onder de variabelen van BASIC verstaan we hier zowel niet-samengestelde variabelen als rijvariabelen en functie-descriptoren. Naast de methoden voor opslag van deze zaken zullen we tevens de opslag en administratie van string-resultaten bespreken.

Variabelen (in bovenstaande brede interpretatie) hebben altijd een naam en een type.

De ***naam*** van een variabele is een aanduiding voor de variabele, bestaande uit een of twee hoofdletters of een hoofdletter gevolgd door een cijfer. (Een naam kan wel uit meer dan twee tekens bestaan, maar alleen de eerste twee worden gebruikt, de overige worden genegeerd).
Aangezien variabelen binnen een BASIC-programma altijd worden aangeduid met een naam, moet er een administratie zijn die het mogelijk maakt om een variabele op te zoeken met behulp van de naam.

Het ***type*** van een variabele is een aanduiding van het soort waarde dat de variabele heeft (of oplevert, als het om een functie-descriptor gaat) en impliciet een aanduiding voor de opslagmethode voor deze waarde.
Het type van een variabele kent vier mogelijkheden: string, integer, enkele precisie-real en dubbele precisie-real. Bij de administratie van elke variabele wordt het type met behulp van een code aangegeven. De typen worden als volgt gecodeerd:

type	*codering*
string	03
integer	02
enkele precisie-real	04
dubbele precisie-real	08

Zoals zal blijken zijn de getallen van de coderingen van het type precies gelijk aan het aantal bytes dat de codering van de waarde in beslag neemt.

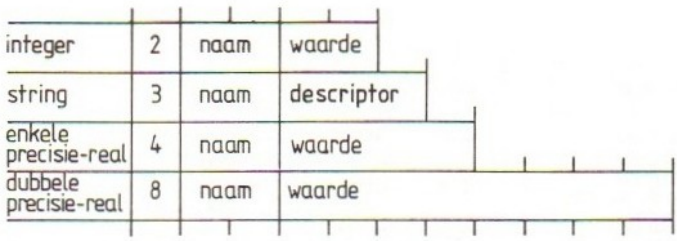
Voor de administratie van de variabelen en de opslag van de waarden zijn door de BASIC-interpreter drie geheugengebieden gereserveerd: één voor enkelvoudige variabelen en functie-descriptoren, één voor rijvariabelen en één voor string-waarden. De plaats van deze gebieden binnen het RAM-geheugen is aangegeven in afbeelding 7.1. De grenzen van de gebieden worden bewaard in enkele OS-adressen (zie ook par. 6.4.1):

F676	TXTTAB	– begin van het BASIC-programma
F6C2	VARTAB	– begin van gebied voor enkelvoudige variabelen en functie-descriptoren
F6C4	ARYTAB	– begin van het gebied voor rijvariabelen
F6C6	STREND	– begin van het vrije geheugengebied
F674	STKTOP	– einde van het gebied voor de stapel
F860	FILTAB	– begin van het gebied voor file-buffers

Enkelvoudige variabelen

Met enkelvoudige variabelen (niet-samengestelde of 'gewone' variabelen) bedoelen we variabelen die precies één waarde hebben – hetzij getalwaarde, hetzij stringwaarde. De enkelvoudige variabelen worden gewoonlijk aangeduid met alleen 'variabele'.

De codering van een enkelvoudige variabele hangt af van het type van de variabele. In afbeelding 7.2 zijn de verschillende coderingen weergegeven.



Afb. 7.2. Codering van variabelen

De codering voor de waarde van een variabele is als volgt:

type	*codering*
string	string-descriptor
integer	2-complement
real	floating point

Een string-descriptor bestaat uit een lengte-aanduiding en een beginadres. De floating point codering van een real wordt in meer detail besproken in par. 7.2.5.

Een variabele-codering wordt gemaakt als de variabele voor de eerste keer wordt aangetroffen bij de uitvoering van een BASIC-programma. De waarde van de variabele wordt initieel op 0 gesteld voor getalvariabelen, en op de lege string voor stringvariabelen.

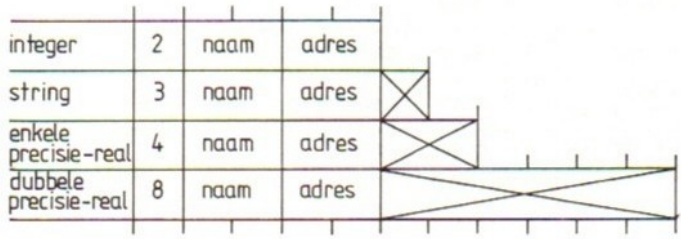
In het gebied voor enkelvoudige variabelen volgen de coderingen van de variabelen elkaar direct op. De administratie van variabelen is impliciet gegeven: een variabele kan gevonden worden door de lijst van opgeslagen variabelen af te lopen.

Functie-descriptoren

Een functie-descriptor is een beschrijving van een BASIC-functie die in het programma is gedefinieerd (dit in tegenstelling tot een standaard BASIC-functie of een machinetaalfunctie). De functie-descriptor wordt gemaakt wanneer een functiedefinitie wordt aangetroffen, en wordt gebruikt bij aanroep van een functie.
Een functie-descriptor wordt opgeslagen in het gebied van de enkelvoudige variabelen. De codering van een descriptor is bijna gelijk aan die van een variabele. Een verschil is dat op de plaats van de waarde van een variabele nu een adres staat dat verwijst naar de definitie van de functie. Bovendien wordt bit 7 van het byte van de eerste letter van de naam van de functie hoog gezet; dit garandeert dat functiedescriptoren nooit worden opgevat als variabelen, en vice versa.

Het adres in het waardeveld van de descriptor wijst naar het eerste byte van het BASIC-programma dat geen deel uitmaakt van de naam van de functie.

In afbeelding 7.3 is de codering voor functie-descriptoren van verschillende typen weergegeven.

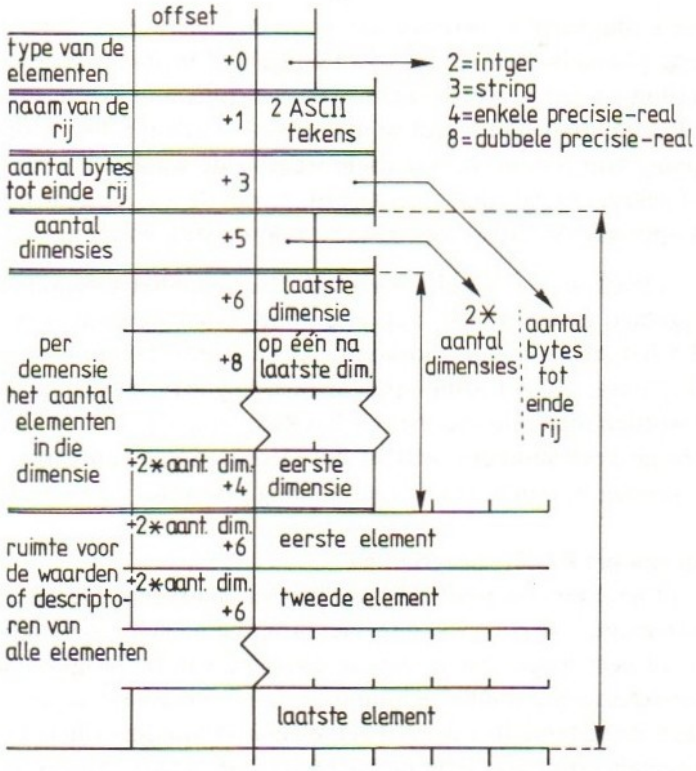


Afb. 7.3. Codering voor functie-descriptoren

Rijvariabelen

De codering van een rijvariabele (of gewoon rij, de Engelse term voor array) wordt opgeslagen in een speciaal daarvoor gereserveerd gebied. De codes voor verschillende rijvariabelen volgen direct op elkaar, net als bij de enkelvoudige variabelen, zodat een gegeven rij kan worden opgezocht door alle bekende rijen af te lopen.
Behalve een naam en een type kent een rijvariabele dimensies. Het aantal dimensies kan verschillen van rij tot rij, net als het aantal elementen in elke dimensie. Bij de code van een rij moet dus ook het aantal dimensies en voor elke dimensie het aantal elementen worden opgeslagen. In afbeelding 7.4 is de codering van een rij weergegeven.

De codering voor een rijvariabele wordt gemaakt bij een DIM-statement waarin de variabele voorkomt, of wanneer de rijvariabele in een programma gebruikt wordt zonder gedimensioneerd te zijn. In het laatste geval wordt het aantal dimensies gelijk gesteld aan het aantal dimensies waarin de rij gebruikt wordt, en het aantal elementen in elke dimensie wordt gelijk gesteld aan 11. Door een ERASE-statement wordt een rijcodering weer verwijderd.



Afb. 7.4. Codering van een rij

String-waarden

De waarde van string-variabelen wordt niet bij de variabele zelf opgeslagen, zoals dat gebeurt voor integer- en real-variabelen. In plaats daarvan wordt een string-descriptor opgeslagen.

De reden hiervoor is dat de lengte van een string-waarde niet vastligt. De waarde van een string-variabele kan een lengte hebben die varieert tussen 0 en 255. Het zou inefficiënt zijn om voor elke string-variabele 255 bytes te reserveren.

De organisatie van strings maakt gebruik van een speciaal gebied waar string-waarden worden opgeslagen. De descriptor van een string-variabele bevat twee elementen: de lengte van de string, en een wijzer naar het beginadres van de string, dat wil zeggen het adres van het eerste teken van de string.

Wanneer aan een string-variabele een nieuwe waarde wordt toegewezen die is berekend uit een string-uitdrukking, wordt de string-waarde opgeslagen in het gebied voor strings, en de descriptor van de variabele gaat naar het beginadres van de waarde wijzen. Als de nieuwe waarde echter een string-constante is, gaat het wijzerblok wijzen naar de representatie van de constante binnen de codering van het BASIC-programma.

Als er nieuwe string-waarden worden berekend en toegewezen, komen die telkens op een nieuwe plaats binnen het string-opslaggebied te staan. Wanneer er in dat gebied geen ruimte meer is, wordt er een 'garbage collection' (vuilnisophaal) op het opslaggebied uitgevoerd. Dit houdt in dat string-waarden in het opslaggebied die niet meer geldig zijn (omdat de variabele waarbij de waarde hoorde een nieuwe waarde heeft gekregen), worden opgespoord, en dat de waarden die nog wel geldig zijn worden opgeschoven tot ze een aaneengesloten stuk vormen.

Een garbage collection kost tamelijk veel tijd. Naarmate het gebruikte deel van het string-opslaggebied groter wordt, zal de garbage collection vaker worden uitgevoerd, omdat het steeds vaker voorkomt dat er ruimte tekort is. Daarom is het verstandig de grootte van het string-opslaggebied aan de ruime kant te houden (deze grootte kan worden ingesteld met het CLEAR-statement).
Als er op een gegeven moment zelfs na het uitvoeren van een garbage collection geen ruimte genoeg is, wordt er een foutmelding gegeven.

7.2.2 Opslag van het BASIC-programma

Een BASIC-programma is opgebouwd uit programmaregels. De regels zijn opgebouwd uit statements. De code waarmee het programma in het geheugen is opgeslagen, bestaat uit een opeenvolging van de codering van de programmaregels. De regels zijn gecodeerd met hun regelnummer, het beginadres van de code van de volgende regel, de statements van de regel, en een aanduiding van het einde van de regel. De statements zijn gecodeerd met behulp van de code van hun samenstellende onderdelen.

Het BASIC-programma is opgeslagen tussen de adressen die opgeslagen zijn in TXTTAB (F676) en VARTAB (F6C2). De codering van het programma begint met een 0, en eindigt met drie keer 0. Het laatste is voor de interpreter het signaal dat het programma afgelopen is.

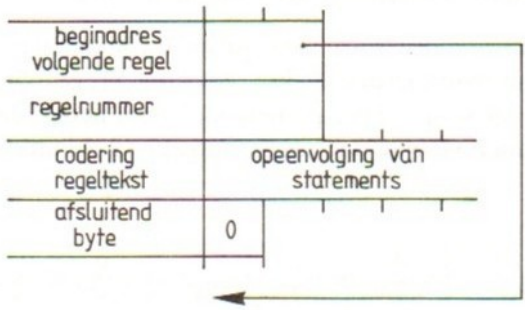
Programmaregels

De regels van een BASIC-programma delen het programma op in stukken, zodat het geheel overzichtelijk blijft. Bovendien zijn programmaregels de enige delen van een programma waar vanuit een statement naar verwezen kan worden (namelijk met behulp van het nummer van de regel).
De code van een programmaregel bestaat uit stuurinformatie en een statement-code. De stuurinformatie wordt gebruikt om het uitvoeren van het programma zo snel mogelijk te laten verlopen. De statement-code bevat de codering van de BASIC-statements waaruit de programmaregel bestaat, inclusief scheidingstekens (dubbele punten).

In afbeelding 7.5 is aangegeven hoe een programmaregel er in gecodeerde vorm uit ziet.

Binnen een programma wordt vanuit sommige (sprong- en andere) statements naar programmaregels verwezen. Deze verwijzingen kunnen op twee manieren worden

gecodeerd: als regelnummer, of als adreswijzer naar het begin van de regel. Voor de manier van coderen zie par. 7.2.5. Deze coderingen kunnen in elkaar worden omgezet. Bij het uitvoeren van een BASIC-programma wordt elke regelverwijzing in de vorm van een regelnummer omgezet in een adreswijzer. Worden er wijzigingen in het programma aangebracht, dan worden alle adreswijzers weer omgezet naar regelnummers.



Afb. 7.5. Een programmaregel in gecodeerde vorm

Bij een RENUM-statement worden alle constanten in een BASIC-programma die gecodeerd zijn in de vorm van een regelnummer of regeladres aangepast aan de veranderde nummering.

Statements

Een BASIC-statement is een bij elkaar horende groep sleutelwoorden, identifiers en uitdrukkingen, die samen een actie aangeven. Deze actie kan bijvoorbeeld het maken van een geluid zijn, of het toewijzen van een waarde aan een variabele.

Elk BASIC-statement is onder te verdelen in elementen. De voornaamste elementen zijn sleutelwoorden en uitdrukkingen. De codering van een BASIC-statement is een samenstelling van de codering van de elementen.
Alle delen van een statement die geen sleutelwoorden of uitdrukkingen zijn, worden gecodeerd op de manier waarop ze zijn gerepresenteerd; dat wil zeggen dat de code bestaat uit de ASCII-code van de letters van de representatie. Sleutelwoorden en uitdrukkingen kennen een speciale codering. Deze coderingen zullen in de volgende paragrafen aan de orde komen.

Een statement wordt afgesloten door een dubbele punt, mits deze geen deel uitmaakt van een string-constante of REM-statement, of door de code voor het regeleinde (zie hiervoor).

7.2.3 BASIC sleutelwoorden

Sleutelwoorden (Engels: keywords) zijn woorden die binnen BASIC een speciale betekenis hebben. De codering van sleutelwoorden is meestal anders dan van andere woorden. Om die reden mogen variabelenamen geen sleutelwoorden bevatten.

De meeste sleutelwoorden worden gecodeerd met behulp van byte-waarden groter dan &H80. Deze waarden kunnen nooit deel uitmaken van gewone uitdrukkingen, woorden of variabelenamen, aangezien alle tekens die in dergelijke elementen kunnen voorkomen, gecodeerd worden met byte-waarden kleiner dan &H80 (volgens de ASCII-code). Waar dus in de programmacode een byte-waarde groter dan &H80 voorkomt (behalve in string-constanten) is het zeker dat er een sleutelwoord gecodeerd is. Dit maak het uitvoeren van het programma sneller.

De code voor een sleutelwoord wordt vaak 'token' (symbool of teken) genoemd. De gebruikte tokens zijn bytewaarden 81-FC (hex), en FF gevolgd door 81-B0. De waarde FF dient als een soort 'ontsnappingscode': door het gebruik van FF op deze manier wordt het aantal beschikbare tokens uitgebreid. In afbeelding 7.6 zijn alle tokens vastgelegd.

	8	9	A	B	C	D	E	F
0		STOP INP	WIDTH CDBL	OPEN MKD$	BEEP	BSAVE	NOT	<
1	END LEFT$	PRINT POS	ELSE FIX	FIELD	PLAY	DSKO$	ERL	+
2	FOR RIGHT$	CLEAR LEN	TRON STICK	GET	PSET	SET	ERR	−
3	NEXT MID$	LIST STR$	TROFF STRIG	PUT	PRESET	NAME	STRING$	*
4	DATA SGN	NEW VAL	SWAP PDL	CLOSE	SOUND	KILL	USING	/
5	INPUT INT	ON ASC	ERASE PAD	LOAD	SCREEN	IPL	INSRT	^
6	DIM ABS	WAIT CHR$	ERROR DSKF	MERGE	VPOKE	COPY		AND
7	READ SQR	DEF PEEK	RESUME FPOS	FILES	SPRITE	CMD	VARPTR	OR
8	LET RND	POKE VPEEK	DELETE CVI	LSET	VDP	LOCATE	CSRLIN	XOR
9	GOTO SIN	CONT SPACE$	AUTO CVS	RSET	BASE	TO	ATTR$	EQV
A	RUN LOG	CSAVE OCT$	RENUM CVD	SAVE	CALL	THEN	DSKI$	IMP
B	IF EXP	CLOAD HEX$	DEFSTR EOF	LFILES	TIME	TABC	OFF	MOD
C	RESTORE COS	OUT LPOS	DEFINT LOC	CIRCLE	KEY	STEP	INKEY$	\
D	COSUB TAN	LPRINT BIN$	DEFSNG LOF	COLOR	MAX	USR	POINT	
E	RETURN ATN	LLIST CINT	DEFDBL MKI$	DRAW	MOTOR	FN	>	
F	REM FRE	CLS CSNG	LINE MK$	PAINT	BLOAD	SPCL	=	esc. code

Afb. 7.6. Overzicht van alle tokens

Zoals uit afbeelding 7.6 blijkt, zijn er ook tokens voor de operatoren die gerepresenteerd worden door speciale tekens in plaats van woorden, zoals de optelling, vermenigvuldiging etc. Dit is om de evaluatie van uitdrukkingen waarin de operatoren voorkomen algemeen te houden. Alle voorkomens van tekens die dienen als representatie van operatoren, worden als token gecodeerd, en niet als ASCII-code. Zie verder par. 7.2.4.

Voor de meeste sleutelwoorden geldt dat de codering door middel van een token minder bytes vergt dan een codering via de ASCII-code. Als gevolg daarvan heeft de token-codering als aardige bijkomstigheid dat de hoeveelheid code erdoor verminderd wordt.

7.2.4 Uitdrukkingen in BASIC

Een uitdrukking (of expressie) in een BASIC-programma is elke groep tekens en woorden binnen een BASIC-statement waarmee een waarde voorgesteld wordt of die een waarde oplevert.
Uitdrukkingen kunnen uit één element bestaan of samengesteld zijn. Uitdrukkingen die uit één element bestaan, worden wel atomaire uitdrukkingen genoemd. Hieronder vallen constanten, variabelen, functie-aanroepen en ook subexpressies, dat wil zeggen uitdrukkingen tussen haakjes. De uitdrukkingen in de laatste categorie kunnen weer samengesteld zijn zodra de haakjes worden weggenomen.

Samengestelde uitdrukkingen bestaan uit operatoren en operanden. Als operand kunnen atomaire uitdrukkingen optreden, of andere samengestelde uitdrukkingen met een hogere prioriteit. Operatoren verwerken de waarden van de operanden die er bij horen, en leveren weer een waarde op. De waarde van een samengestelde uitdrukking (of: de waarde die een uitdrukking oplevert) is de waarde die opgeleverd wordt door de laatst uitgevoerde operator van de uitdrukking.

Operatoren

Een operator verwerkt de waarden van de operanden die er bij horen, en levert aan de hand daarvan weer een waarde op.
Op twee na hebben alle operatoren twee operanden. Aan beide kanten van de code voor de operator staat de codering van een operand. De enige uitzonderingen zijn de operator NOT en de omkering (berekening van het tegengestelde): deze hebben maar één operand, waarvan de codering achter de code voor de operator staat.

Op basis van het type van de operanden die bij een operator horen en de manier waarop de waarden van de operanden verwerkt worden, kunnen de operatoren worden opgedeeld in verschillende categorieën. De categorieën zijn: rekenkundig, string, vergelijking en logisch.

Als een uitdrukking samengesteld is uit een heleboel operanden gescheiden door operatoren, moet er een of andere regeling zijn om te bepalen in welke volgorde de operatoren uitgevoerd moeten worden. Deze volgorde is van invloed op de waarde die de uitdrukking oplevert.

In MSX-BASIC wordt de volgorde geregeld door de prioriteit van de operatoren. Elke operator heeft een prioriteit, uitgedrukt door een getal; bij evaluatie van een uitdrukking worden operatoren met hogere prioriteit uitgevoerd vóór operatoren met lagere prioriteit. Operatoren met gelijke prioriteit worden van links naar rechts uitgevoerd.

In de volgende tabel worden de operatoren opgesomd met representatie, code, categorie en prioriteit.

categorie operator		*prioriteit*	*representatie*	*code*
rekenkundig	machtsverheffen	11	^	F5
	vermenigvuldigen	10	*	F3
	delen	10	/	F4
	tegengestelde	10	−	F2
	integer deling	9	\	FC
	modulus	8	MOD	FB
	optellen	7	+	F1
	aftrekken	7	−	F2
string	concateneren	7	+	F1
vergelijking	gelijk	6	=	EE
	ongelijk	6	< >	F0+EE
	kleiner	6	<	F0
	kleiner of gelijk	6	<=	F0+EF
	groter	6	>	EE
	groter of gelijk	6	>=	EE+EF
logisch	not	5	NOT	E0
	and	4	AND	F6
	or	3	OR	F7
	exclusive or	2	XOR	F8
	equivalentie	2	EQV	F9
	implicatie	1	IMP	FA

Het veld 'code' geeft de byte-waarden waarmee de operator binnen de codering van een BASIC-programma aangeduid wordt.

Rekenkundige operatoren hebben getaluitdrukkingen als operand, en leveren zelf ook een getal op. De meeste rekenkundige operatoren zijn overbekend. Het zij opgemerkt dat er twee betekenissen zijn van het minteken (−): tegengestelde-berekening en aftrekken. Er kan echter geen verwarring ontstaan, aangezien de berekening van het tegengestelde maar één operator vraagt, en de aftrekking twee.

Het type van de waarde die door een rekenkundige operator opgeleverd wordt, is afhankelijk van het type van de operanden. Als beide operanden hetzelfde type hebben, zal de opgeleverde waarde ook dat type hebben, tenzij de operanden type integer hebben en het resultaat van de operatie te groot is om met dat type te kunnen worden gerepresenteerd: in dat geval zal het resultaat het type enkele precisie hebben.
Als de operanden een verschillend type hebben, heeft de opgeleverde waarde het type met de grootste nauwkeurigheid. De nauwkeurigheid van een type is evenredig met het aantal bytes die de code van het type beslaat.

String-operatoren vergen twee string-uitdrukkingen als operanden, en leveren een string-waarde op. In het MSX-systeem is er maar een string-operator: concatenatie, ofte wel samenknopen van twee strings tot één.

Vergelijkings-operatoren vergen twee getaluitdrukkingen óf twee string-uitdrukkingen als operatoren, en leveren 0 of –1 op; deze waarden kunnen worden geïnterpreteerd als 'onwaar' respectievelijk 'waar'. De opgeleverde waarde is gecodeerd als integer.

Logische operatoren vergen twee integer-uitdrukkingen (of één in het geval van NOT) als operatoren en leveren een getal op waarvan elk bit afgeleid wordt uit de overeenkomstige bits van de operanden. De wijze van afleiding verschilt per operator. Als een operand niet als integer gecodeerd is, wordt de codering eerst omgezet naar 2-complement. Als dat niet mogelijk is, omdat de waarde te groot is, wordt er een foutmelding gegeven.
Voor een meer gedetailleerde uitleg van de operatoren zie par. 7.3.

Operanden
De operatoren in samengestelde uitdrukkingen werken op operanden. De operanden kunnen atomaire uitdrukkingen zijn, of samengestelde uitdrukkingen met een hogere prioriteit, of een samengestelde uitdrukking met gelijke prioriteit waarvan de codering volgt op de operator (de prioriteit van een samengestelde uitdrukking is bepaald door de prioriteit van de operator).

Een atomaire uitdrukking is een constante, een variabele, een functie-aanroep of een sub-expressie, dat wil zeggen een (eventueel samengestelde) nieuwe uitdrukking tussen haakjes. De codering van een sub-expressie bestaat uit de codering van de uitdrukking met daar omheen de ASCII-code voor de haakjes.
Constanten zijn representaties van getal- of string-waarden binnen het BASIC-programma. De codering van constanten wordt behandeld in par. 7.2.5.

Het gebruik van variabelen betekent het opvragen van de huidige waarde van de variabele, en het gebruik van die waarde in een uitdrukking. De manier van opslag van een variabele is in par. 7.2.1 uitgelegd. De waarde van een variabele wordt gevonden door de codering van de variabele op te zoeken in de opslagruimte van de variabelen, en daaruit de waarde te halen. Binnen een uitdrukking wordt een varia-

bele aangeduid met de naam van de variabele; deze naam wordt gecodeerd door de ASCII-code van de letter- en cijfertekens van de naam.

Er zijn drie soorten functies binnen een BASIC-programma: standaard BASIC-functies, zelf-gedefinieerde BASIC-functies en machinetaalfuncties. Standaard BASIC-functies zijn functies die in de interpreter ingebouwd zijn, en gecodeerd worden met behulp van tokens. Zelf-gedefinieerde functies vormen een methode om binnen een programma aan een uitdrukking een naam te geven, en verder de naam te gebruiken in plaats van de uitdrukking. Machinetaalfuncties zijn routines die in machinetaal geschreven zijn, en die in een programma worden gebruikt in de vorm van een functie-aanroep (zie par. 7.3, sleutelwoord USR).
In par. 7.2.6 wordt dieper ingegaan op functies in BASIC.

7.2.5 Constanten in BASIC

Onder een constante verstaan we een item in een BASIC-programma dat een waarde aanduidt. Constanten komen voor in uitdrukkingen. Een constante wordt meestal gecodeerd als een informatiebyte, waaruit het soort constante kan worden afgeleid, gevolgd door een codering van de waarde van de constante. Dit geldt echter niet voor string-constanten. Aan constanten onderscheiden we een type, een waarde, een codering en een representatie.

Het ***type*** van een constante is het soort waarde van de constante. De mogelijke typen van constanten zijn: integer, real, string of regelnummer.

De ***waarde*** van een constante is, afhankelijk van het type, een getalwaarde of een tekstwaarde (string-waarde). Een regelnummerconstante is een aanduiding van een regel in een BASIC-programma.

De ***codering*** van een constante is de manier waarop de waarde van de constante intern in het geheugen is weergegeven. Voor strings is de codering een uitgebreide versie van de ASCII-code. Integers kunnen binair gecodeerd zijn of in 2-complementvorm. Reals worden gecodeerd in floating-point. De waarde van een regelnummer is binair gecodeerd.

De ***representatie*** van een constante is de manier waarop de waarde van de constante door de programmeur wordt gezien of geschreven. De representatie van een constante kan variëren, afhankelijk van het type van de constante.

Integers

Een integer is een geheel getal, ofte wel een getal zonder decimaal deel. De codering van een integer is impliciet, binair in één of twee bytes, 2-complement in twee bytes of ASCII. De representatie van een integer is decimaal, binair, octaal of hexadecimaal.

Decimale representatie van een integer is de alledaagse vorm van een getal, bestaande uit cijfers van 0 tot 9, eventueel voorafgegaan door een minteken. Hexadecimale representatie bestaat uit de tekens &H, gevolgd door de hexadecimale cijfers (0-9 en

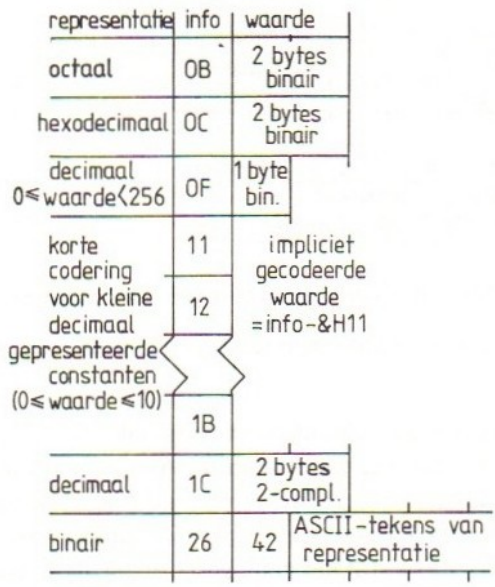
A-F) van het getal. Octale representatie bestaat uit de tekens &O en de octale cijfers (0-7); binaire representatie is &B gevolgd door de binaire cijfers (0 en 1).

Het verband tussen de waarde, de representatie en de codering van een integer constante is als volgt:

representatie	*waardebereik*	*coderingswijze*	*code*
dec	0 t/m 10	impliciet	11 t/m 1B
dec	11 t/m 255	binair	0F+1 byte
dec	–32768 t/m 32767	2-compl	1C+2 bytes
bin	00000000 t/m 11111111	ASCII	'&B'+n*'0'/'1'
oct	0 t/m 177777	binair	0B+2 bytes
hex	0 t/m FFFF	binair	0C+2 bytes

De kolom 'code' geeft aan hoe de waarde er gecodeerd uit ziet. De code begint altijd met een of ander byte dat aangeeft wat de coderingswijze en de representatie van de constante is. Dit byte noemen we het informatiebyte. Het informatiebyte wordt gevolgd door de codering van de waarde van de constante. De informatiebytes in de kolom 'code' zijn gegeven door middel van de hexadecimale representatie van hun waarde.

Bij de coderingswijzen zijn twee bijzondere methoden genoemd: impliciet en ASCII. Impliciete codering houdt in dat de waarde van de constante al in het informatiebyte ingesloten is. De bytewaarden 11 t/m 1B (hex) komen overeen met de integerwaarden 0 t/m 10 (decimaal).



Afb. 7.7. Coderingsvormen van integer-constanten

De ASCII-codering van binair gerepresenteerde constanten komt neer op een codering die precies gelijk is aan de representatie. In de code verschijnen als bytes de ASCII-waarden van de tekens van de representatie, dus &B gevolgd door de nullen en enen van het getal.

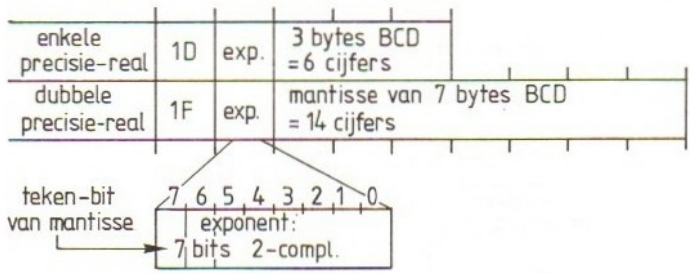
Bij de codering van een decimaal gerepresenteerde integer- constante in een BASIC-statement kiest de interpreter altijd voor de coderingswijze die de minste ruimte in beslag neemt.
In afbeelding 7.7 zijn de coderingsvormen van integerconstanten weergegeven.

Reals

Een real constante is een benadering van een reëel getal. Per definitie is een reëel getal niet met een eindig aantal cijfers te representeren; hoe meer cijfers echter gebruikt worden, hoe beter de benadering.

De codering van een real is als floating-point getal. De exponent is gecodeerd in 2-complement en beslaat zeven bits. De mantisse is gecodeerd in BCD en beslaat drie of zeven bytes; dat komt overeen met zes respectievelijk veertien cijfers. Voor het teken van de mantisse wordt één bit gebruikt; dit bit is 0 voor een positieve en 1 voor een negatieve mantisse.
Real-getallen met een mantisse van drie bytes worden aangeduid met enkele precisie-getallen; getallen met een mantisse van zeven bytes heten dubbele precisie-getallen.

In afbeelding 7.8 zijn de coderingsvormen van real constanten weergegeven.



Afb. 7.8. Coderingsvormen van real-constanten

Het informatie-byte geeft, net als bij integer-constanten, aan wat voor soort getalcodering er volgt.

De representatie van een real constante is óf decimaal (met een punt als het getal een decimale fractie heeft) óf in exponent-notatie. Als het getal decimaal gerepresenteerd wordt, zal de representatie worden gevolgd door een uitroepteken (!) voor enkele precisie-getallen en door een hekje (#) voor dubbele precisie-getallen. In het geval van exponent-notatie wordt het getal geschreven als een decimale representatie van de mantisse, gevolgd door de letter E en een decimale representatie van de exponent. Voor dubbele precisie-getallen kan de letter D worden gebruikt.

Strings
De waarde van een string-constante is een serie tekens. Het begrip 'tekens' moet hier ruim genomen worden: niet alleen letter- en cijfertekens vallen eronder, maar ook grafische symbolen, en zelfs speciale besturingstekens die niet gerepresenteerd worden (zie verderop).

Een string-constante wordt gecodeerd als een serie bytes waarvan elk byte de code voor een teken is. Het verband tussen de waarde van een byte en het bijbehorende teken is vastgelegd in een uitgebreide versie van de ASCII-code (American Standard Code for Information Interchange). In de officiële ASCII-code zijn alleen de bytewaarden 0 t/m 127 vastgelegd. In de MSX-standaard is ook de rest van de waarden gebruikt, voor speciale tekens en grafische symbolen.

ASCII-code 01 verdient nog enige uitleg. Deze code wordt wel het grafische extensie-kopbyte genoemd, omdat met behulp van deze code nog een aantal symbolen kunnen worden gerepresenteerd die niet in de ASCII-tabel staan: de grafische extensie-symbolen. De representatie hiervan gebeurt door code 01 gevolgd door een bytewaarde tussen 40 en 5F (hex). (Deze symbolen worden op het scherm gezet door gebruik van patronen 00-1F uit de VDP-patroontabel (zie par. 3.3.1).

De uitgebreide ASCII-code van de MSX-computer is weergegeven in Appendix A. De grafische extensie-symbolen zijn opgenomen onder de waarde waarmee ze in de VDP-naamtabel komen te staan.

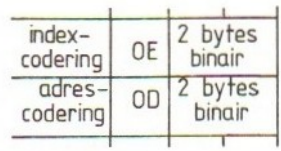
De representatie van een string-constante is de waarde, omsloten door aanhalingstekens ("). Als zich in de stringwaarde aanhalingstekens bevinden, kan de constante niet gerepresenteerd worden. Sommige van de tekens uit de ASCII-tabel kunnen helemaal niet gerepresenteerd worden, namelijk de besturingstekens, gecodeerd door middel van bytes met een waarde kleiner dan 20 (hex), uitgezonderd sommige voorkomens van het grafische extensie-kopbyte (code 01) (zie hiervoor).

Regelnummers
Een speciaal soort constanten vormen de regelnummers. Hoewel de representatie eruit ziet als van een integerconstante, worden regelnummers anders gecodeerd en door de BASIC-interpreter anders behandeld dan getalconstanten.

In feite moet een regelnummer niet worden beschouwd als een getal, maar als een aanduiding, namelijk van een BASIC-regel. Dat deze aanduiding toevallig de vorm van een getal heeft doet niet ter zake.
Regelnummers kunnen op twee manieren worden gecodeerd. Deze manieren zijn in afbeelding 7.10 weergegeven. In beide manieren bestaat de code uit een informatie-byte, gevolgd door twee bytes met daarin de waarde. Aan het informatie-byte is af te lezen welke manier van codering gebruikt is. De betekenis van de waarde verschilt nogal in beide manieren.

In beide coderingsvormen wordt de waarde van de constante binair gecodeerd.
In de codering als index (de term 'index' is niet helemaal juist, maar wordt hier

gebruikt om onderscheid te kunnen maken) is de waarde van de regelnummer constante, een getal dat overeenkomt met het getal waarmee de bedoelde regel in de codering van het BASIC-programma opgenomen is. De aanduiding van de regel gebeurt dus met een soort volgnummer – vandaar 'index'.



Afb. 7.9. Coderen van regelnummers

In de codering als adres duidt de regelnummer constante de regel aan door het adres binnen het programma te geven waar de codering van de regel begint.
De beide coderingen zijn gemakkelijk in elkaar over te zetten. Als bij een index-gecodeerd regelnummer het adres moet worden gevonden, kan het BASIC-programma vanaf het begin worden doorlopen tot de regel met de betreffende 'index' gevonden is. De index te vinden bij een adres-gecodeerd regelnummer is nog eenvoudiger: het adres wijst namelijk naar het begin van de code van de regel, en deze code begint nu juist met de index.

De beide coderingen van regelnummers worden op hun eigen manier gebruikt. Tijdens het uitvoeren van een statement wordt een index-gecodeerd regelnummer altijd omgezet in een adres. De volgende keer dat zo'n statement uitgevoerd wordt, is meteen dat adres bekend, hetgeen de uitvoering van het statement aanzienlijk versnelt.
Bij het schrijven of veranderen van een BASIC-programma is het juist nodig om de regelnummers met index te coderen, aangezien de adressen dan voortdurend veranderen. Zodra er een verandering wordt aangebracht in een BASIC-programma worden alle adres-gecodeerde regelnummers dan ook omgezet in een index.

7.2.6 Functie-aanroepen

Een functie is een mechanisme dat bij de meegegeven argumenten een waarde construeert en die waarde oplevert. De opgeleverde waarde is altijd op een bepaalde manier afhankelijk van de argumenten. De vorm van afhankelijkheid is de definitie of werking van de functie.

In BASIC zijn er drie soorten functies: standaard-BASIC-functies, zelf gedefinieerde BASIC-functies en machinetaal-functies.

Standaard BASIC-functies

In de BASIC- interpreter zijn een groot aantal functies igebouwd. Deze functies zijn op verschillende gebieden werkzaam: en zijn goniometrische functies, string-functies, systeemfuncties en andere.

Kenmerkend voor deze standaard BASIC-functies is dat ze gecodeerd worden door middel van tokens. In par. 7.2.3 is een lijst van alle tokens opgenomen; hierin zijn de functies terug te vinden. (Alle niet-systeemfuncties worden gecodeerd door FF plus een tweede code).

De standaard BASIC-functies zijn allemaal opgenomen in de beschrijving van sleutelwoorden in par. 7.3.

Zelf gedefinieerde BASIC-functies

Naast de standaard functies kunnen in een BASIC-programma functies zelf-gedefinieerd worden. Zo'n zelf gedefinieerde functie kan worden beschouwd als een naam voor een uitdrukking. De functie-definitie bestaat uit een functiekop en een uitdrukking die aangeeft hoe uit de argumenten van de functie (en eventueel nog andere variabelen) het resultaat van de functie berekend moet worden.

Bij aanroep van een zelf-gedefinieerde functie wordt voor de argumenten de waarden ingevuld die bij aanroep zijn meegegeven. Daarna wordt met gebruik van die waarden de uitdrukking berekend volgens welke de functie gedefinieerd is. Het resultaat is dan precies hetzelfde als wanneer in plaats van de functie-aanroep die uitdrukking was ingevuld.

Het precieze verloop van een functie-aanroep is als volgt:
- bij definitie van de functie is in het variabelengebied een functie-descriptor aangemaakt. In deze descriptor staat als waarde het adres van het eerste teken in het BASIC-programma dat niet meer tot de naam van de functie hoort;
- bij aanroep van de functie wordt de descriptor opgezocht. Als er bij de aangeroepen functie geen descriptor hoort, wordt er een foutmelding gegeven;
- uit de descriptor wordt het adres gehaald;
- als er argumenten over te dragen zijn, worden de waarden van de uitdrukkingen die in de aanroep staan, ingevuld in de variabelen die in de functiekop van de definitie staan;
- de uitdrukking in de functiedefinitie wordt berekend; het resultaat van deze berekening is de waarde die door de functie opgeleverd wordt.

Voor meer informatie over zelf-gedefinieerde BASIC-functies zie de sleutelwoorden DEF en FN in par. 7.3.

Machinetaalfuncties

Het enige aanknopingspunt dat MSX BASIC biedt met machinetaalroutines is een aanroep van een machinetaalfunctie.

Een machinetaalfunctie kan in een BASIC-programma gedefinieerd worden door een adres te koppelen aan een van de beschikbare USR-functies door middel van een DEF USR-statement. Het adres is het beginadres van de machinetaalroutine die aangeroepen moet worden.

Bij een aanroep van zo'n USR-functie roept de BASIC-interpreter de routine aan die begint op het adres dat bij de definitie is opgegeven.

De USR-functies hebben een parameter-overdrachtsmechanisme om het mogelijk te maken waarden die binnen het BASIC-programma berekend worden binnen de machinetaalroutine beschikbaar te hebben. Hetzelfde mechanisme wordt gebruikt om de functie een waarde te laten opleveren die binnen BASIC weer verder gebruikt kan worden.
Voor meer informatie over machinetaalfuncties zie de sleutelwoorden DEF en USR in par. 7.3.

7.3 Sleutelwoorden van BASIC

In het oorspronkelijke handboek van elke MSX-computer staat een lijst van BASIC-sleutelwoorden met uitleg. Het lijkt dan ook wellicht vrij zinloos om deze sleutelwoorden hier nogmaals op te nemen.
We hebben echter ondervonden dat de bestaande handleidingen op dit punt meestal noch volledig noch correct zijn. Afgezien daarvan is het zinvol om de sleutelwoorden vanuit een ander gezichtspunt te bespreken, namelijk vanuit de kennis die we al hebben van de opbouw van het MSX-systeem.

Naast de gewone BASIC-woorden zijn ook de DISK BASIC-sleutelwoorden opgenomen. Als een woord alleen in DISK BASIC voorkomt, is dit aangegeven.

Bij elk sleutelwoord wordt gegeven:
- Het token. Dit is de codering waarmee BASIC het woord intern opslaat. Het token wordt in hexadecimale vorm gegeven, met tussen haakjes de decimale waarde. Sommige tokens bestaan uit meer dan één byte. De bytewaarden zijn dan gescheiden door een plusteken.

- De categorie waarin het woord valt. Bekende categorieën zijn:

  Statement
  Deze categorie bevat alle eerste woorden van een statement. Voorbeelden: SAVE, PRINT, LET.
  Functie
  Een functie is een sleutelwoord dat een waarde oplevert, afhankelijk van de waarde van de argumenten. Zie par. 7.2.6. Voorbeelden: CSRLIN, INSTR, SIN.
  Operator
  Spreekt vanzelf. Voorbeelden: MOD, XOR.
  Hulpwoord
  In deze categorie zitten woorden die alleen gebruikt worden binnen een ander statement. Voorbeelden: AS, TO.
  Systeemvariabele
  Deze categorie bevat sleutelwoorden die kunnen worden gebruikt alsof het variabelen zijn. Er kan een waarde aan worden toegewezen, en van worden uitgelezen. Voorbeelden: BASE, VDP.

- Een verwijzing naar andere sleutelwoorden, die in doel of werking te maken hebben met het besproken woord.
- De vorm waarin het woord kan worden gebruikt. Wat er precies in de vorm van een sleutelwoord gegeven wordt, hangt van het soort woord af. Voor statements wordt de vorm van het hele statement gegeven, voor functies de aanroep, voor operatoren de toepassing op operanden, en voor hulpwoorden soms niets, en soms de vorm van het gedeelte van het statement waarin het hulpwoord voorkomt.

Kleine-letterwoorden in de vormbeschrijving duiden op waarden die zelf gekozen mogen worden; waaruit de keuze gedaan moet worden, wordt in de uitleg verteld.

- Een lijst van OS-routines en geheugenplaatsen die bij het uitvoeren van het statement worden gebruikt. Deze lijst bevat alleen geheugenplaatsen die specifiek zijn voor een statement; de geheugenplaatsen die worden gebruikt bij het evalueren van een uitdrukking worden nooit genoemd.

Gebruikte terminologie

In veel van de vormbeschrijvingen van sleutelwoorden komen velden of argumenten voor die bij gebruik van het woord moeten worden vervangen door uitdrukkingen. Aan de hand van het soort waarde die een uitdrukking mag hebben die voor een bepaald veld kan worden ingevuld, zijn uitdrukkingen opgedeeld in categorieën. In de vormbeschrijvingen komen de volgende categorieën van uitdrukkingen voor:

Numerieke uitdrukking: uitdrukking die een getalwaarde oplevert.

Integeruitdrukking: numerieke uitdrukking, die een geheel getal oplevert dat als integer gerepresenteerd kan worden. De codering van integers gebeurt door middel van twee bytes in 2-complement; dat houdt in dat de waarde van een integeruitdrukking moet liggen tussen –32768 en 32767 (inclusief grenzen). Wanneer een veld of argument waar een integeruitdrukking moet worden ingevuld een waarde buiten het integerbereik krijgt, resulteert dit in de foutboodschap 'Overflow'.
In feite hoeft een integeruitdrukking geen geheel getal op te leveren; echter, het resultaat van zo'n uitdrukking wordt altijd meteen omgezet (d.m.v. de functie CINT) naar een getal in integerrepresentatie; daarbij verdwijnt de eventuele decimale fractie van het getal.

Byte-uitdrukking: dit is een integeruitdrukking, waarvan de waarde in het bytebereik moet vallen, d.w.z tussen 0 en 255 (inclusief grenzen).

Kleuruitdrukking: dit is een integeruitdrukking, waarvan de waarde als code van een kleur opgevat wordt. Dat wil zeggen dat deze waarde tussen 0 en 15 (inclusief grenzen) moet liggen.
Het verband tussen de waarde van een kleuruitdrukking en de bijbehorende kleur is vastgelegd in de volgende tabel:

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

File-nummeruitdrukking: dit is een byte-uitdrukking waarvan de waarde verwijst naar een file.
Bij het openen van een file met een OPEN-statement wordt aan een file een nummer toegekend. Dit nummer moet kleiner dan of gelijk zijn aan het totaal aantal beschikbare file-buffers (in te stellen met de systeemvariabele MAXFILES). De waarde van een file-nummeruitdrukking moet gelijk zijn aan het nummer van een open file.

Regelnummerconstante: dit is een getal tussen 0 en 65529 dat een regelnummer voorstelt. Alle regelnummerconstanten worden intern gecodeerd als regelnummer of regeladres (zie par. 7.2.5), zodat ze mee worden hernummerd bij het uitvoeren van een RENUM-statement.

String-uitdrukking: deze uitdrukking levert een string-waarde op, d.w.z. een rij tekens gecodeerd volgens de uitgebreide ASCII-code van de MSX. De lengte van de string kan variëren van 0 tot 255 (inclusief grenzen).

Coördinaatuitdrukking: aanduiding van een coördinatenpaar. Deze aanduiding heeft de vorm: STEP(x,y). Het woord STEP mag eventueel weggelaten worden. Als STEP niet ingevuld is, duidt de uitdrukking het punt aan met coördinaten x en y. Als STEP wel ingevuld is, geven x en y de afstand tot het laatst bereikte punt, bewaard in GXPOS (FCB3) en GYPOS (FCB5).

File-uitdrukking: dit is een string-uitdrukking, waarvan de inhoud de vorm heeft van een file-aanduiding. De algemene vorm van een file-aanduiding is:

apparaatnaam:file-naam.extensie

Hierin is ***apparaatnaam*** (Engels: device) de naam van één van de mogelijke randapparaten die files gebruiken, bijvoorbeeld: cassetterecorder, diskettestation, toetsenbord, printer. Mogelijke apparaatnamen zijn:

CAS	Cassetterecorder
A	Diskettestation A
B	Diskettestation B
PT	Printer
KBD	Toetsenbord
GRP	Grafisch scherm
COMn	RS232 (n is het nummer van de RS232-ingang)

File-naam is de naam die de file op het apparaat heeft. Het gebruik van de file-naam verschilt. Van bovenstaande apparaten kunnen alleen de cassetterecorder en het diskettestation met file-namen werken. Voor de cassetterecorder mag een file-naam zes tekens lang zijn, voor het diskettestation acht tekens.

Extensie is een achtervoegsel voor de file-aanduiding. De extensie wordt alleen gebruikt voor diskettestation, en mag 3 tekens lang zijn.
Bij het gebruik van een diskettestation als apparaat kunnen er in de file-naam en extensie zgn. 'wildcards' opgenomen worden. Dit zijn tekens waarvoor een willekeurige invulling kan worden gedaan. De regels voor de wildcards zijn als volgt:

- het teken '?' is een één-letter-wildcard: voor elk voorkomen van dit teken kan iedere letter of cijfer worden ingevuld;
- het teken '*' aan het einde van de file-naam of de extensie geeft aan dat voor dat gedeelte een serie tekens van willekeurige lengte mag worden ingevuld.

ABS FF+86 (255+134)

Categorie functie
Zie ook SGN
Vorm ABS(x)

Het argument x is een numerieke uitdrukking.

ABS levert de absolute waarde van x op. Dat wil zeggen: als de waarde van x negatief is, levert de functie het tegengestelde van x op, anders x.

AND F6 (246)

Categorie operator
Zie ook EQV, IMP, NOT, OR, XOR
Vorm x AND y

De operanden x en y zijn integeruitdrukkingen.

De operanden worden gecodeerd volgens 2-complement. Het resultaat van AND is een getal, ook gecodeerd volgens 2-complement, waarvan elk bit direct afhankelijk is van het overeenkomstige bit in x en y.

AND neemt de logische bitsgewijze 'and'-functie van de operanden. Voor AND geldt het volgende diagram:

bit van x:	0	0	1	1
bit van y:	0	1	0	1
bit van AND:	0	0	0	1

AND is een logische operator met prioriteit 4 (zie par. 7.2.4).

APPEND 41+51+51+81 (65+65+71+129)

Categorie hulpwoord
Zie ook FOR, INPUT, OPEN, OUTPUT
Vorm FOR APPEND

FOR APPEND is één van de manieren waarop een file geopend kan worden. Vóór een file op deze wijze geopend kan worden, moet hij al bestaan. Er kan sequentieel naar geschreven worden; alle gegevens die naar de file geschreven worden, komen achter de bestaande inhoud.
In de BASIC-code is dit woord gecodeerd als A+P+P+END.

AS 41+54 (65+74)

Categorie hulpwoord
Zie ook FIELD, OPEN

AS wordt gebruikt als hulpwoord in allerlei statements.

Het woord AS heeft geen speciaal token, d.w.z. een coderings-byte met waarde 80. AS wordt gecodeerd als A+S. Dat houdt in dat, in tegenstelling tot wat in de meeste handleidingen wordt beweerd, AS wél als variabelenaam mag worden gebruikt.

ASC FF+95 (255+149)

Categorie functie
Zie ook CHR$
Vorm ASC(x)

Het argument x is een string-uitdrukking.
De functie levert de ASCII-code van het eerste teken van de waarde van x op.

ATN FF+8E (255+142)

Categorie functie
Zie ook SIN, COS, TAN
Vorm ATN(x)

Het argument x is een numerieke uitdrukking.
Het resultaat van de functie is de arctangens van de waarde van x, in radialen uitgedrukt.

ATTR$ E9 (233)

Categorie functie
Zie ook CMD, IPL, SET

Deze functie wordt in de huidige systemen niet gebruikt; het sleutelwoord is toegevoegd voor eventueel toekomstig gebruik.
In de functie ATTR$ is voorzien door middel van de RAM-haak H.ATTR (FE1C).

AUTO A9 (169)

Categorie statement
Zie ook DELETE, LIST, RENUM
Vorm AUTO *regnrs*

Bij het uitvoeren van AUTO worden regelnummers gegenereerd, waarachter regels van een BASIC-programma kunnen worden getypt. Telkens na het geven van RETURN wordt het volgende nummer gegenereerd. Dit houdt pas op bij het indrukken van CTRL+STOP of CTRL+C, of als het regelnummer dat gegenereerd zou moeten worden groter is dan 65529.

Het veld *regnrs* bepaalt welke regelnummers gegenereerd zullen worden. *Regnrs* kan de volgende vormen aannemen:

AUTO *nr,stap.*
Het eerste regelnummer is nr. Elk volgende regelnummer is stap groter dan het vorige.

AUTO *nr.*
Het eerste regelnummer is nr. Elk volgende nummer is een hoeveelheid groter, gelijk aan de vorige keer dat AUTO gebruikt werd. Die hoeveelheid is opgeslagen in AUTINC (F6AD). Deze vorm mag niet gebruikt worden de eerste keer dat AUTO gebruikt wordt.

AUTO *nr.*
Het eerste regelnummer is nr; elk volgende regelnummer is 10 groter dan het vorige.

AUTO *,stap.*
Het eerste regelnummer is 0. Elk volgende nummer is stap groter dan het vorige.

AUTO .
Het eerste regelnummer is 0. Elk volgende nummer is een hoeveelheid groter, gelijk aan de vorige keer dat AUTO gebruikt werd. Die hoeveelheid is opgeslagen in AUTINC (F6AD). Deze vorm mag niet gebruikt worden de eerste keer dat AUTO gebruikt wordt.

AUTO
Het eerste regelnummer is 10. Elk volgende nummer is 10 groter dan het vorige.

De velden *nr* en *stap* zijn regelnummerconstanten. De waarden van *nr* en *stap* worden bewaard in AUTLIN (F6AB) resp. AUTINC (F6AD).
Bij het uitvoeren van het AUTO-statement wordt gebruik gemaakt van de schermeditor (par. 6.2.7), en met name van de routine PINLIN (ingang 00AE, par. 6.3).

De volgende OS-routines worden door AUTO gebruikt:

0018	OUTDO	- voor het schrijven van het regelnummer
00AE	PINLIN	- voor het invoeren van een regel

De volgende geheugenplaatsen worden door AUTO gebruikt:

F6AA	AUTFLG	- geeft aan dat AUTO aan de gang is
F6AB	AUTLIN	- gegenereerd regelnummer
F6AD	AUTINC	- verschil tussen gegenereerde nummers

BASE C9 (201)

Categorie systeemvariabele
Zie ook VDP, VPOKE, VPEEK
Vorm BASE(nr)

Het argument nr is een byte-uitdrukking. De waarde ervan moet liggen tussen 0 en 19 (inclusief grenzen).
Het sleutelwoord BASE correspondeert met de beginadressen van de tabellen die gebruikt worden door de VDP in de verschillende schermmodi. Met de waarde van nr wordt de tabel gekozen. Voor de betekenis van de tabellen zie hoofdstuk 3 en par. 6.4.1.

Het verband tussen de waarde van nr, de gekozen tabel en de geheugenplaats waar het beginadres opgeslagen is, is als volgt:

nr	*tabel*	*modus*	*geheugenplaats*
0	Naamtabel	0	TXTNAM (F3B3)
1	Niet gebruikt		TXTCOL (F3B5)
2	Patroontabel	0	TXTCGP (F3B7)
3	Niet gebruikt		TXTATR (F3B9)
4	Niet gebruikt		TXTPAT (F3BB)
5	Naamtabel	1	T32NAM (F3BD)
6	Kleurentabel	1	T32COL (F3BF)
7	Patroontabel	1	T32CGP (F3C1)
8	Sprite-plaatstabel	1	T32ATR (F3C3)
9	Sprite-patroontabel	1	T32PAT (F3C5)
10	Naamtabel	2	GRPNAM (F3C7)
11	Kleurentabel	2	GRPCOL (F3C9)
12	Patroontabel	2	GRPCG
13	Sprite-plaatstabel	2	GRPATR (F3CD)
14	Sprite-patroontabel	2	GRPPAT (F3CF)
15	Naamtabel	3	MLTNAM (F3D1)
16	Kleurentabel	3	MLTCOL (F3D3)
17	Patroontabel	3	MLTCGP (F3D5)
18	Sprite-plaatstabel	3	MLTATR (F3D7)
19	Sprite-patroontabel	3	MLTPAT (F3D9)

Door een nieuwe waarde aan een van de adressen van BASE te geven, wordt de waarde van de bijbehorende geheugenplaats veranderd; daardoor wordt voortaan door het OS het nieuwe adres gebruikt als beginadres van de tabel.

BEEP C0 (192)

Categorie statement
Zie ook PLAY, SOUND
Vorm BEEP

Genereert een korte, zachte piep. Zet ook alle PSG-registers terug naar hun oorspronkelijke waarde, en reset het geluids-subsysteem (zie par. 6.2.4).

Het effect van dit statement komt overeen met het het effect van de OS-routine BEEP (ingang 00C0).

BEEP gebruikt de volgende OS-routine:

00C0	BEEP	- genereert een piep en reset geluidssysteem

BIN$ FF+9D (255+157)

Categorie functie
Zie ook HEX$, OCT$, STR$
Vorm BIN$(x)

Het argument x is een integeruitdrukking.
De functie levert een string op waarin de waarde van x in binaire representatie weergegeven is. Negatieve getallen worden in 2-complement-notatie gegeven.

BLOAD CF (207)

Categorie statement
Zie ook BSAVE
Vorm 1. BLOAD *fnaam,modus,off*
2. BLOAD *fnaam,off*

Het veld *fnaam* is een file-uitdrukking. *Modus* is de letter S of de letter R. *Off* is integeruitdrukking. *Off* mag eventueel weggelaten worden, mits de voorafgaande komma ook weggelaten wordt. In vorm 2 mag de uitdrukking *off* niet met de letter S of de letter R beginnen.

Dit commando leest een file in, met file-naam *fnaam*, die met BSAVE moet zijn weggeschreven. Als *modus*=S, dan wordt de inhoud van de file in het videogeheugen gezet. Als *modus*=R, dan wordt de file als machinetaalroutine beschouwd, en wordt na het inlezen van de file het executie-adres aangeroepen dat bij de file is weggeschreven (zie BSAVE).
De inhoud van de file komt terecht in het (video-)geheugen vanaf het adres waar

vandaan hij met BSAVE is weggehaald; als *off* is gespecificeerd, wordt bij dit adres echter eerst de waarde van *off* opgeteld.

BLOAD maakt gebruik van de volgende geheugenplaatsen:

F862	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- beginadres gebruikte file-buffer
F866	FILNAM	- file-naam (waarde van *fnaam*)
F87D	SAVEND	- informatie over eindadres geheugengebied
FCBE	RUNBNF	- geeft aan of file direct uitgevoerd moet
FCBF	SAVENT	- informatie over beginadres geheugengebied

BSAVE D0 (208)

Categorie statement
Zie ook BLOAD
Vorm 1. BSAVE *fnaam,begin,eind,exec*
2. BSAVE *fnaam,begin,eind,S*

Het veld *fnaam* is een file-uitdrukking. *Begin, eind* en *exec* zijn integer-uitdrukkingen. *Exec* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt. De uitdrukking *exec* mag niet met de letter S beginnen.

Dit statement schrijft een blok van het geheugen of videogeheugen weg onder filenaam *fnaam*. Het beginadres wordt gegeven door de waarde van *begin*, het eindadres door eind. Eind moet groter zijn dan *begin*.

Als vorm 1 gebruikt is, wordt het 'gewone' geheugen genomen. *Begin* en *eind* moeten in dat geval een waarde groter dan &H7FFF hebben. Bij de file wordt ook een executie-adres weggeschreven; dit is gelijk aan de waarde van *exec* als dat veld aanwezig is, anders gelijk aan *begin*. *Exec* hoeft niet tussen *begin* en *eind* te liggen.
Als vorm 2 gebruikt is, wordt een gedeelte van het videogeheugen weggeschreven.

BSAVE maakt gebruik van de volgende geheugenplaatsen:

F862	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- beginadres gebruikte file-buffer
F866	FILNAM	- file-naam (waarde van *fnaam*)
F87D	SAVEND	- informatie over eindadres geheugengebied
FCBE	RUNBNF	- geeft aan of file direct uitgevoerd moet
FCBF	SAVENT	- informatie over beginadres geheugengebied

CALL CA (202)

Categorie statement
Zie ook USR
Vorm CALL *ident(arg)*

Het veld *ident* is de naam van een statement-uitbreiding die door een pagina verzorgd wordt. *Arg* kan een reeks van string- of numerieke uitdrukkingen zijn, gescheiden door komma's, afhankelijk van de routine. Als er voor de routine geen argumenten nodig zijn, moeten *arg* en de omsluitende haakjes weggelaten worden.

Bij uitvoering van dit statement worden alle pagina's afgelopen die een BASIC-statement-uitbreiding bevatten (zie par. 6.2.1) op zoek naar een pagina die de naam *ident* kent. Wordt deze gevonden, dan wordt de besturing aan die pagina overgedragen. Het woord CALL kan altijd worden vervangen door een onderstreep (_).

CALL maakt gebruik van de volgende geheugenplaatsen:

FCC9	SLTATR	- tabel die statement-uitbreidingen aangeeft
FD89	PROCNM	- naam van statement (waarde van *ident*)

CDBL FF+A0 (255+160)

Categorie functie
Zie ook CINT, CSNG
Vorm CDBL(*x*)

Het veld *x* is een numerieke uitdrukking.
Het resultaat van CDBL is getal met dubbele precisie-formaat (8 bytes floating point) en dezelfde waarde als *x*.

CHR$ FF+96 (255+150)

Categorie functie
Zie ook ASC, STRING$
Vorm CHR$(*x*)

Het argument *x* is een byte-uitdrukking.
Het resultaat van CHR$ is een string met lengte 1, bestaande uit het teken met als ASCII-code de waarde van *x*.

CINT FF+9E (255+158)

Categorie functie
Zie ook CDBL, CSNG, FIX, INT
Vorm CINT(*x*)

Het veld *x* is een integer-uitdrukking.
Het resultaat van CINT is een getal met integerformaat (twee bytes gecodeerd volgens 2-complement). De waarde van het getal is de waarde van *x* zonder de decimale fractie.

CIRCLE BC (188)

Categorie statement
Zie ook DRAW, LINE, PAINT, PSET, PRESET
Vorm CIRCLE *midd,straal,kleur,begin,eind,verh*

Het veld *midd* is een coördinaatuitdrukking. *Straal* is een integeruitdrukking die een niet-negatieve waarde moet hebben. *Kleur* is een kleuruitdrukking. De velden *begin* t/m *verh* zijn numerieke uitdrukkingen. *Kleur* t/m *verh* mogen weggelaten worden. Als het statement daardoor eindigt op komma's, moeten die komma's ook weggelaten worden.

CIRCLE resulteert in een hele of gedeeltelijke cirkel of ellips op het scherm. *Midd* duidt het middelpunt aan; *straal* geeft de straal van de cirkel, *kleur* de kleur van de rand, *begin* en *eind* respectievelijk de begin- en eindhoek van de cirkelboog die getekend wordt, gerekend tegen de klok in vanaf de rechter x-as; *verh* geeft de verhouding in punten van de verticale en horizontale straal.
De waarde van *straal* geeft een afstand, uitgedrukt in coördinaatpunten. Bij *verh s* 1 is dit de afstand van het middelpunt tot de zijkanten, bij *verh r* 1 de afstand van het middelpunt tot de boven- en onderkant van de ellips; zie verderop.

Als *kleur* weggelaten wordt, wordt de huidige voorgrondkleur genomen. De kleur waarin getekend wordt (ofwel de waarde van *kleur* ofwel de voorgrondkleur, dus de waarde van FORCLR (F39E)) komt in ATRBYT (F3F2) te staan.

Begin en *eind* zijn hoekgrootheden, in radialen uitgedrukt. Als ze weggelaten worden, wordt als waarde 0 aangenomen. Als *begin* of *eind* een negatieve waarde heeft, wordt er een extra verbinding getekend tussen het begin- of eindpunt van de cirkelboog en het middelpunt van de cirkel.
Intern worden de waarden van *begin* en *eind* bijgehouden in CENCNT (F933) en CSTCNT (F93F). De waarde van CPLOTF (F938) geeft aan of CENCNT de waarde van *begin* dan wel *eind* representeert. Het negatief zijn van begin en/of eind wordt bijgehouden in CLINEF (F935).

Verh is een positief getal. Als *verh* gelijk is aan 1.4, wordt er een zuiver ronde cirkel getekend. Is verh kleiner, dan wordt de cirkel een platte ellips; *verh* > 1.4 resulteert in een smalle ellips. Intern wordt de waarde van *verh* bijgehouden in ASPECT (F931).
Als *verh* weggelaten wordt, wordt de gecombineerde waarde van ASPCT1 (F40B) en ASPCT2 (F40D) aangenomen. Standaard staan deze waarden zo ingesteld dat dat hetzelfde effect heeft als *verh*=1.

Het statement CIRCLE maakt gebruik van het grafische systeem en de meeste routines daarvan.

Speciale routines van CIRCLE:

0126 GTASPC - haalt waarden van ASPCT1 en ASPCT2 op

Geheugenplaatsen die gebruikt worden door CIRCLE:

F40B	ASPCT1	- standaard verhouding lengte/breedte
F40D	ASPCT2	- standaard verhouding lengte/breedte
F92A	CLOC	- adres in patroontabel van grafische cursor
F92C	CMASK	- bitmasker voor grafische cursor
F931	ASPECT	- verhoudingsgetal lengte/breedte
F933	CENCNT	- aantal punten tot eindpunt
F935	CLINEF	- vlag voor lijnen naar middelpunt
F936	CNPNTS	- toepassing onbekend
F938	CPLOTF	- richting van tekening
F939	CPCNT	- toepassing onbekend
F93B	CPCNT8	- aantal punten in cirkel
F93D	CRCSUM	- toepassing onbekend
F93F	CSTCNT	- aantal punten tot beginpunt
F941	CSCLXY	- vlag voor interpretatie van ASPECT
F942	CSAVEA	- opslagruimte voor CLOC
F944	CSAVEM	- opslagruimte voor CMASK
F945	CXOFF	- horizontale afstand tot middelpunt
F947	CYOFF	- verticale afstand tot middelpunt
FCB3	GXPOS	- x-coördinaat van de grafische cursor
FCB5	GYPOS	- y-coördinaat van de grafische cursor
FCB7	GRPACX	- constructie x-coördinaat nieuwe cursor
FCB9	GRPACY	- constructie y-coördinaat nieuwe cursor

N.B. CIRCLE werkt alleen in schermmodi 2 en 3.

CLEAR 92 (146)

Categorie statement
Zie ook FRE, MAXFILES
Vorm CLEAR *str,gebr*

Str en *gebr* zijn integeruitdrukkingen. Beide velden mogen weggelaten worden. Als *str* weggelaten wordt, moeten de komma en *gebr* ook weggelaten worden. Als *gebr* weggelaten wordt, moet ook de voorafgaande komma weggelaten worden.

CLEAR wordt gebruikt voor het reserveren van geheugen voor bepaalde doeleinden. Het veld *str* geeft aan hoeveel geheugen moet worden vrijgehouden voor strings. *Gebr* geeft het hoogste adres dat voor file-buffers gebruikt mag worden. Vanaf dat adres, tot aan het adres waar het werkgeheugen van het OS begint (de waarde van HIMEM (FC4A) bij het aanzetten van de computer) kan het geheugen gebruikt worden voor eigen doeleinden.

De waarde van *gebr* (indien opgegeven) wordt ingevuld in HIMEM (FCA4). De waarden van *gebr* en *str* zijn bepalend voor de waarde van de geheugenplaatsen MEMSIZ (F672), STKTOP (F674) en FILTAB (F860).

Als *str* niet wordt opgegeven, wordt de grootte van het string-opslaggebied niet veranderd. De beginwaarde voor de grootte van dat gebied is 200 bytes. Als *gebr* niet opgegeven wordt, blijft HIMEM onveranderd.

CLEAR heeft als neveneffect dat alle functies en variabelen die eerder in het programma zijn gebruikt of gedefinieerd, verloren gaan, en dat alle open files gesloten worden.

Geheugenplaatsen die door CLEAR kunnen worden veranderd:

F672	MEMSIZ	- hoogste adres van string-ruimte
F674	STKTOP	- hoogste adres van stapel
F860	FILTAB	- beginadres van file-tabel
F862	NULBUF	- adres file-buffer van file met nummer 0
FCA4	HIMEM	- beginadres werkgeheugen OS

CLOAD 9B (155)

Categorie statement
Zie ook CSAVE
Vorm 1. CLOAD *fnaam*
2. CLOAD? *fnaam*

Fnaam is een file-uitdrukking. De waarde van *fnaam* mag geen apparaatnaam of extensie bevatten. Als apparaat wordt altijd de cassetterecorder genomen. *Fnaam* mag weggelaten worden.
De cassette wordt afgezocht naar een file met naam *fnaam*. Als *fnaam* uit het statement wordt weggelaten, wordt de eerstvolgende file op de cassette genomen.
Terwijl gezocht wordt naar de juiste file, verschijnen de namen van files die overgeslagen worden, op het scherm met de boodschap 'Skip: naam'. Als de file gevonden is, wordt dat gemeld met 'Found: naam'.

Als vorm 1 is gebruikt, wordt de inhoud van de bedoelde file in het geheugen gezet, mits het een gecodeerd BASIC-programma is. Als vorm 2 is gebruikt, wordt de inhoud vergeleken met het BASIC-programma in het geheugen; zijn de twee niet gelijk, dan wordt een foutmelding gegeven.

De baudrate waarmee de file weggeschreven is (zie CSAVE) wordt door CLOAD automatisch vastgesteld, en de file wordt met de juiste snelheid ingelezen.

De volgende OS-routines worden door CLOAD gebruikt:

00E1	TAPION	- leest een kopblok in van cassette
00E4	TAPIN	- leest een byte van cassette
00E7	TAPIOF	- beëindigt het lezen van cassette

De volgende geheugenplaatsen worden door CLOAD gebruikt:

F860	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- beginadres van gebruikte file-buffer
F866	FILNAM	- naam van de file die ingelezen wordt
F87C	NLONLY	- geeft aan dat CLOAD gebruikt wordt
FCA4	LOWLIM	- wordt gebruikt door de OS-routines
FCA5	WINWID	- wordt gebruikt door de OS-routines

CLOSE B4 (180)

Categorie statement
Zie ook OPEN
Vorm CLOSE *nr,...*

Het veld *nr* is een file-nummeruitdrukking, eventueel voorafgegaan door een hekje (#). CLOSE kan willekeurig veel parameters meekrijgen; d.w.z. dat er 0 of meer velden *nr* kunnen zijn, gescheiden door komma's.

CLOSE sluit files die geopend zijn met OPEN (zie OPEN). Als er geen parameters gegeven zijn, worden alle open files gesloten, anders alleen die files waarvan het nummer voorkomt in de parameterlijst.
Het sluiten van een file houdt in dat de inhoud van de file-buffer aan de file wordt toegevoegd als de file een uitvoer-file of random access-file is, en dat de file wordt verwijderd uit de administratie van open files.

CLOSE gebruikt de volgende geheugenplaatsen:

F860	FILTAB	- tabel van beginadressen file-buffers
F864	PTRFIL	- beginadres gebruikte file-buffer

CLS 9F (159)

Categorie statement
Zie ook COLOR, SCREEN, WIDTH
Vorm CLS

Bij uitvoering van dit statement wordt het scherm schoongeveegd. Als het scherm in tekstmodus is (schermmodus 0 of 1) worden de logische regels tevens gelijk gemaakt aan de schermregels, en wordt de cursor in de linkerbovenhoek gezet.
Het schoonvegen van het scherm betekent in tekstmodi het vullen van het scherm (ofte wel de naamtabel) met spaties (ASCII-code &H20), en in grafische modi het terugzetten van alle patronen in de patroontabel op 'uit' en het veranderen van de kleuren in de kleurentabel in de huidige voor- en achtergrondkleur. Zie hoofdstuk 3.

Het effect van het statement CLS komt overeen met de OS-routine CLS (ingang 00C3). Deze routine wordt onder andere gebruikt in de scherm-editor (zie par. 6.2.7).

OS-routine waarmee CLS te maken heeft:

00C3	CLS	- schoonmaken van het scherm

Geheugenplaatsen die mogelijk door CLS aangepast worden:

F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)
FBB2	LINTTB	- administratie van logische regels

CMD D7 (215)

Categorie statement
Zie ook ATTR$, IPL, SET

Dit statement wordt in het huidige systeem niet gebruikt; het sleutelwoord is toegevoegd voor toekomstig gebruik.
In het statement CMD is voorzien door middel van de RAM-haak H.CMD (FE0D).

COLOR BD (189)

Categorie statement
Zie ook SCREEN
Vorm COLOR *voor,achter,rand*

De velden *voor, achter* en *rand* zijn kleuruitdrukkingen. Elk van deze velden mag weggelaten worden, maar niet alle drie tegelijk. Als het statement op komma's eindigt, moeten die ook weggelaten worden.

COLOR wordt gebruikt om de standaardkleuren waarmee gewerkt wordt, te veranderen. Als *voor* ingevuld is, wordt de waarde ervan de nieuwe voorgrondkleur en komt in FORCLR (F3E9) te staan; als *achter* ingevuld is, wordt dit de nieuwe achtergrondkleur en komt in BAKCLR (F3EA) te staan; als rand ingevuld is, wordt de waarde hiervan in schermmodi 1, 2 en 3 gebruikt voor de kleur van de rand, en komt in BDRCLR (F3EB) te staan.
In schermmodi 0 en 1 worden bij het uitvoeren van dit statement de kleuren op het scherm meteen veranderd; bij de grafische schermmodi wordt alleen de randkleur veranderd, en worden de andere nieuw ingestelde kleuren gebruikt als standaardkleur bij komende grafische statements.

In schermmodus 1 wordt bij het uitvoeren van COLOR de kleurentabel in het videogeheugen in zijn geheel aangepast aan de nieuwe kleuren (zie hoofdstuk 3).

De volgende OS-routine wordt door COLOR gebruikt:

0062	CHGCLR	- verandert de standaard kleuren

De volgende geheugenplaatsen worden door COLOR aangepast:

F3E9	FORCLR	- de voorgrondkleur
F3EA	BAKCLR	- de achtergrondkleur
F3EB	BDRCLR	- de randkleur
F3F2	ATRBYT	- de werkkleur

CONT 99 (153)

Categorie statement
Zie ook STOP, TROFF, TRON
Vorm CONT

Bij het uitvoeren van CONT gaat de computer verder met het uitvoeren van het BASIC-programma in het geheugen, op de plaats waar het de laatste keer onderbroken werd.

CONT werkt alleen als het programma onderbroken is door een foutmelding, door het statement STOP of door de toetsen CTRL+STOP, en er daarna geen veranderingen in het programma zijn aangebracht.
CONT wordt vaak gebruikt in combinatie met STOP om een programma op fouten te testen.

Geheugenplaatsen die door CONT gebruikt worden zijn:

F6BE	OLDLIN	- regelnummer van regel met onderbreking
F6C0	OLDTXT	- adres van volgend uit te voeren statement

COPY D6 (214)

Categorie statement
Zie ook LOAD, SAVE, FILES
Vorm COPY *fnaam1* TO *fnaam2*

Fnaam1 en *fnaam2* zijn file-uitdrukkingen. De enige apparaatnamen die in de waarden van *fnaam1* en *fnaam2* mogen voorkomen zijn de diskettestations A en B. Als er geen apparaatnaam is opgegeven, wordt het huidige diskettestation gekozen.

COPY maakt een kopie van de inhoud van de file met naam *fnaam1* – mits bestaand – in een file met naam *fnaam2*. *Fnaam2* mag niet op dezelfde file duiden als *fnaam1*. Als er al een file met naam *fnaam2* bestaat, wordt deze overschreven.
Als *fnaam2* alleen uit een apparaatnaam bestaat, wordt voor de file de naam uit *fnaam1* gekozen. Er mogen wildcards voorkomen in *fnaam1* en *fnaam2*, mits in beide namen op dezelfde plaats. In dat geval worden alle files gekopieerd met namen die voldoen aan de wildcards.

COPY maakt gebruik van de volgende geheugenplaatsen:

F860	FILTAB	- tabel van beginadressen file-buffers
F864	PTRFIL	- adres van gebruikte file-buffer
F866	FILNAM	- eerste file-naam (waarde van fnaam1)
F871	FILNM2	- tweede file-naam (waarde van fnaam2)

N.B. COPY werkt alleen in DISK-BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.COPY (FE08).

COS FF+8C (255+140)

Categorie functie
Zie ook ATN, SIN, TAN
Vorm COS(*x*)

Het argument *x* is een numerieke expressie.
De waarde van *x* wordt geïnterpreteerd als een hoekgrootte, in radialen uitgedrukt. De functie levert de cosinus van die hoek op.

CSAVE 9A (154)

Categorie statement
Zie ook CLOAD, SCREEN
Vorm CSAVE *fnaam,baud*

Fnaam is een file-uitdrukking. De waarde van *fnaam* mag geen apparaatnaam of extensie bevatten. *Baud* is een byte-uitdrukking die 1 of 2 als waarde heeft. *Baud* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.

CSAVE schrijft het BASIC-programma dat ten tijde van het statement in het geheugen staat, naar de cassette onder de file-naam die in *fnaam* gevonden wordt.

Baud verandert de snelheid van het wegschrijven. Als *baud=1*, dan wordt de snelheid ingesteld op 1200 baud (bits/sec); als *baud=2* dan wordt 2400 baud gebruikt. Als *baud* niet opgegeven is, wordt de huidig ingestelde snelheid gebruikt.
De verandering van snelheid wordt uitgevoerd door een gedeelte van CS120 (F3FC) te kopiëren naar LOW (F406), HIGH (F408) en HEADER (F40A). De standaard snelheid van wegschrijven kan ook veranderd worden door middel van het SCREEN-statement.

De volgende OS-routines worden door CSAVE gebruikt:

00EA	TAPOON	- schrijft een kopblok naar cassette
00ED	TAPOUT	- schrijft een byte naar cassette
00F0	TAPOOF	- beëindigt het schrijven naar cassette

De volgende geheugenplaatsen worden door CSAVE gebruikt:

F3FC	CS120	- signaallengten bij verschillende snelheden
F406	LOW	- duur van uitvoersignaal voor bit 0
F408	HIGH	- duur van uitvoersignaal voor bit 1
F40A	HEADER	- duur van kopblok
F862	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- adres van gebruikte file-buffer
F866	FILNAM	- opslag van file-naam (waarde van *fnaam*)

CSNG FF+9F (255+159)

Categorie functie
Zie ook CDBL, CINT
Vorm CSNG(x)

Het veld *x* is een numerieke uitdrukking.
De functie levert een getal op in enkele precisie-formaat (4 bytes floating point). De waarde van het getal is gelijk aan de waarde van *x*, eventueel afgerond als *x* in dubbele precisie was.

CSRLIN E8 (232)

Categorie functie
Zie ook LOCATE, POS
Vorm CSRLIN

Als het scherm in tekstmodus is (schermmodus 0 of 1), levert de functie het nummer op van de regel (verticale positie) waarop de cursor zich bevindt. De bovenste schermregel heeft nummer 0; elke volgende regel heeft een nummer dat 1 hoger is. Als het scherm in grafische modus is (schermmodus 2 of 3), levert de functie het nummer van de regel waar de cursor stond toen het scherm voor het laatst in tekstmodus was.

De opgeleverde waarde is de waarde van CSRY (F3DC) – 1.

CSRLIN gebruikt de geheugenplaats:

F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)

CVD FF+AA (255+170)

Categorie functie
Zie ook CVI, CVS, MKD$
Vorm CVD(*str*)

Het veld *str* is een string-uitdrukking. De lengte van de waarde van *str* is 8 bytes. Bij uitvoering van CVD wordt de waarde van *str* opgevat als een getal in dubbele

precisie-formaat (8 bytes floating point) waarvan alle bytes gecodeerd zijn als ASCII-tekens. Het resultaat van CVD is de waarde van dat getal in *str*.

N.B. CVD werkt alleen in DISK BASIC. De functie wordt geïmplementeerd via de RAM-haak H.CVD (FE49).

CVI FF+A8 (255+168)

Categorie functie
Zie ook CVD, CVS, MKI$
Vorm CVI(*str*)

Het veld *str* is een string-uitdrukking. De lengte van de waarde van *str* is 2 bytes.

Bij uitvoering van CVI wordt de waarde van *str* opgevat als een getal in integer-formaat (2 bytes in 2-complement) waarvan de bytes gecodeerd zijn als ASCII-tekens. Het resultaat van CVI is de waarde van dat getal in *str*.

N.B. CVI werkt alleen in DISK BASIC. De functie wordt geïmplementeerd via de RAM-haak H.CVI (FE3F).

CVS FF+A9 (255+169)

Categorie functie
Zie ook CVD, CVI, MKS$
Vorm CVS(*str*)

Het veld *str* is een string-uitdrukking. De lengte van de waarde van *str* is 4 bytes.

Bij uitvoering van CVS wordt de waarde van *str* opgevat als een getal in enkele precisie-formaat (4 bytes floating point) waarvan alle bytes gecodeerd zijn als ASCII-tekens. Het resultaat van CVS is de waarde van dat getal in *str*.

N.B. CVS werkt alleen in DISK-BASIC. De functie wordt geïmplementeerd via de RAM-haak H.CVS (FE44).

DATA 84 (132)

Categorie statement
Zie ook READ, RESTORE
Vorm DATA item,...

Het veld *item* is een constante string of getalwaarde. Als *item* een string is zonder komma, apostrof of dubbele punt en niet begint of eindigt met een spatie, hoeven er geen aanhalingstekens omheen te staan. DATA kan gevolgd worden door een willekeurig aantal (0 of meer) velden item.

Alle velden van DATA worden behandeld als invoergegevens voor eventuele READ-statements. Dat wil zeggen dat de waarde van *item* kan worden toegewezen aan een parameter van READ; zie READ.

Een DATA-statement hoeft niet te worden uitgevoerd alvorens gebruikt te worden. Dat houdt in, dat DATA-statements mogen staan *achter* READ-statements waarin de velden worden gebruikt.

DEF 97 (151)

1.

Categorie statement
Zie ook FN
Vorm DEF *FNident(arg)=uitdr*

Het veld *ident* is de naam van een functie. Deze naam is in vorm gelijk aan een variabelenaam. *Arg* is een lijst van variabelenamen, gescheiden door komma's. *Arg* mag weggelaten worden; dan moeten de omsluitende haakjes ook weggelaten worden. *Uitdr* is een uitdrukking van hetzelfde type (string of getal) als *ident.*

Uitvoering van dit statement heeft als gevolg dat voortaan de functie *FNident* bekend is, en overal in het programma gebruikt kan worden waar een uitdrukking gebruikt kan worden. Voor het gebruik van een dergelijke functie zie FN.
De functiedefinitie wordt onthouden door een descriptor van de functie op te slaan in het gebied voor variabelen. Als er al een functie met de naam *ident* bestond, wordt de oude definitie ongedaan gemaakt. Zie ook par. 7.2.1 en 7.2.6.

DEF FN gebruikt de geheugenplaatsen:

F6C2	VARTAB	- beginadres variabelen-opslaggebied
F6C4	ARYTAB	- beginadres rijvariabelen-opslaggebied
F6C6	STREND	- adres van de eerste vrije geheugenplaats

2.

Categorie statement
Zie ook USR
Vorm DEF *USRcijf=adres*

Cijf is een cijfer, dus een teken tussen 0 en 9 (inclusief grenzen). Als *cijf* weggelaten wordt, heeft dit hetzelfde resultaat als *cijf=0. Adres* is een integeruitdrukking.
Uitvoering van dit statement heeft tot gevolg dat voortaan de machinetaalroutine *USRcijf* bekend is. De waarde van *adres* is het geheugenadres waar deze routine begint. Voor het gebruik van de routine zie USR.

De waarde van *adres* wordt opgeslagen in USRTAB (vanaf F39A) in het element dat hoort bij de waarde van *cijf.*

DEF USR maakt gebruik van de volgende geheugenplaats:

F39A	USRTAB	- beginadressen van machinetaalroutines

DEFDBL AE (174)

Categorie statement
Zie ook DEFINT, DEFSNG, DEFSTR
Vorm DEFDBL *letlijst*

Het veld *letlijst* is een lijst van letters. In letlijst kunnen letters en letterbereiken voorkomen, gescheiden door komma's. Een letterbereik is van de vorm l1-l2, waarbij l1 en l2 losse letters zijn. Dit heeft hetzelfde effect als een lijst van alle letters tussen l1 en l2 (inclusief grenzen).

Na uitvoering van DEFDBL worden alle identifiers (variabele- en functienamen) die met één van de letters uit *letlijst* beginnen en niet eindigen op een speciaal teken dat het type aangeeft, opgevat als dubbele precisie-identifiers. Zo'n identifier is equivalent aan dezelfde identifier gevolgd door een hekje (#).
DEFDBL wordt uitgevoerd door aan de elementen van DEFTBL (vanaf F6CA) die horen bij de letters in letlijst de waarde 8 toe te kennen.

DEFDBL gebruikt de volgende geheugenplaatsen:

F6CA	DEFTBL	- tabel van standaardtypen voor variabelen

DEFINT AC (172)

Categorie statement
Zie ook DEFDBL, DEFSNG, DEFSTR
Vorm DEFINT *letlijst*

De betekenis van *letlijst,* en het effect van het statement, zijn gelijk aan DEFDBL. De identifiers die door *letlijst* aangeduid worden, zullen nu echter worden opgevat als integers, en zijn equivalent aan dezelfde identifier gevolgd door een procentteken (%).
DEFINT wordt uitgevoerd door aan de elementen van DEFTBL (vanaf F6CA) die horen bij de letters in letlijst de waarde 2 toe te kennen.

DEFINT gebruikt de volgende geheugenplaatsen:

F6CA	DEFTBL	- tabel van standaardtypen voor variabelen

DEFSNG AD (173)

Categorie statement
Zie ook DEFDBL, DEFINT, DEFSTR
Vorm DEFSNG *letlijst*

De betekenis van *letlijst,* en het effect van het statement, zijn gelijk aan DEFDBL. De identifiers die door *letlijst* worden aangeduid, worden nu echter opgevat als enkele precisie, en zijn equivalent aan dezelfde identifier gevolgd door een uitroepteken (!).

DEFSNG wordt uitgevoerd door aan de elementen van DEFTBL (vanaf F6CA) die horen bij de letters in letlijst de waarde 4 toe te kennen.

DEFSNG gebruikt de volgende geheugenplaatsen:

F6CA	DEFTBL	- tabel van standaardtypen voor variabelen

DEFSTR AB (171)

Categorie statement
Zie ook DEFDBL, DEFINT, DEFSNG
Vorm DEFSTR *letlijst*

De betekenis van *letlijst,* en het effect van het statement, zijn gelijk aan DEFDBL. De identifiers die door *letlijst* worden aangeduid, worden nu echter opgevat als strings, en zijn equivalent aan dezelfde identifier gevolgd door een dollarteken ($). DEFSTR wordt uitgevoerd door aan de elementen van DEFTBL (vanaf F6CA) die horen bij de letters in letlijst de waarde 3 toe te kennen.

DEFSTR gebruikt de volgende geheugenplaatsen:

F6CA	DEFTBL	- tabel van standaardtypen voor variabelen

DELETE A8 (168)

Categorie statement
Zie ook AUTO, LIST, RENUM
Vorm DELETE *regnrs*

DELETE verwijdert alle regels die door *regnrs* worden aangeduid uit het BASIC-programma dat ten tijde van het statement in het geheugen staat.
Regnrs geeft aan welke regels verwijderd moeten worden. Dit veld kan de volgende vormen aannemen:

DELETE *nr1-nr2*
Alle regels met nummers tussen *nr1* en *nr2* (inclusief grenzen) worden verwijderd.

DELETE *-nr*
Alle regels vanaf de eerste programmaregel tot en met nummer *nr* worden verwijderd.

DELETE *nr*
Alleen de regel met nummer *nr* wordt verwijderd. Het effect is hetzelfde als wanneer alleen *nr* wordt ingetypt (dus DELETE weggelaten).

DELETE .
De laatst behandelde (ingevoegde, veranderde, geliste) regel wordt verwijderd. Het nummer van deze regel wordt bewaard in DOT (F6B5).

De velden *nr, nr1* en *nr2* zijn regelnummerconstanten.

Bij het uitvoeren van het statement DELETE worden de opslaggebieden van variabelen en rijvariabelen vernietigd. Ook wordt het string-opslaggebied schoongemaakt, en wordt het uitvoeren van CONT onmogelijk gemaakt.

DELETE gebruikt de volgende geheugenplaatsen:

F69B	FRETOP	- eerste vrije adres in string-opslaggebied
F6B5	DOT	- nummer van laatst behandelde regel
FC60	OLDTXT	- adres van volgend uit te voeren statement
F6C2	VARTAB	- beginadres van variabelen-opslag
F6C4	ARYTAB	- beginadres van rijvariabelen-opslag
F6C6	STREND	- adres van eerste ongebruikte geheugenplaats

DIM 86 (134)

Categorie statement
Zie ook ERASE
Vorm DIM *ident(dimlijst),...*

Ident is een variabelenaam. *Dimlijst* is een lijst van integeruitdrukkingen (1 of meer), gescheiden door komma's. De uitdrukkingen in *dimlijst* mogen geen negatieve waarde opleveren
DIM kan een willekeurig aantal (1 of meer) parameters hebben. Na uitvoering van het statement zijn deze parameters gedeclareerd als rijvariabelen met als naam *ident* en als aantal dimensies het aantal uitdrukkingen in *dimlijst*. Elke dimensie heeft als aantal elementen: 1 + de waarde van de bijbehorende uitdrukking.

Er mag vóór het uitvoeren van het statement nog geen rijvariabele bestaan met naam *ident*.
Als ergens in een programma een rijvariabele wordt gebruikt die niet in een DIM-statement gedeclareerd is, wordt deze variabele alsnog gedimensioneerd, met het aantal dimensies waarmee hij gebruikt wordt, en in elke dimensie 11 elementen, genummerd van 0 tot 10 (inclusief grenzen).

Bij het dimensioneren van een rijvariabele wordt er in het opslaggebied voor rijvariabelen (vanaf ARYTAB (F6C4) tot STREND (F6C6)) ruimte gemaakt voor de variabele. Hoe deze ruimte georganiseerd is, staat in par. 7.2.1.

DIM gebruikt de volgende geheugenplaatsen:

F6C4	ARYTAB	- beginadres van rijvariabelen-opslag
F6C6	STREND	- adres van eerste vrije geheugenplaats

DRAW BE (190)

Categorie statement
Zie ook CIRCLE, LINE, PAINT, PRESET, PSET
Vorm DRAW *str*

De parameter *str* is een string-uitdrukking.
DRAW verzorgt het tekenen van figuren. Wat er precies getekend wordt, hangt af van de waarde van *str*.

De waarde van *str* is op te delen in substrings die een grafisch commando bevatten. Deze grafische commando's vormen een soort van mini-taaltje dat alleen geldig is binnen de parameters van DRAW.

Mogelijke grafische commando's zijn:

A *hoek*
Door het geven van dit commando worden alle volgende grafische subcommando's over een zekere hoek geroteerd. De grootte van de rotatiehoek wordt bepaald door de waarde van *hoek*:

hoek	rotatie
0	0°
1	*90°*
2	*180°*
3	*270°*

De beginwaarde van *hoek* is 0. De waarde van *hoek* wordt opgeslagen in DRWANG (FCBD).

B
Dit commando heeft tot gevolg dat bij het eerstvolgende commando waarin een lijn wordt getrokken, de lijn niet wordt getrokken, maar de grafische cursor wel verplaatst wordt. Het voorkomen van B wordt gesignaleerd in DRWFLG (FCBB).

C *kleur*
Door het geven van dit commando wordt de voorgrondkleur veranderd. Dit zal tot gevolg hebben dat alle volgende commando's in de *kleur*, aangeduid door kleur, worden uitgevoerd. De waarde van *kleur* wordt opgeslagen in ATRBYT (F3F2).

D *len*
Dit commando trekt een lijn in de voorgrondkleur naar beneden (270x t.o.v. de standaardhoek) ter lengte van *len* grafische punten.

E *len*
Dit commando trekt een lijn in de voorgrondkleur naar rechts boven (45x t.o.v. de standaardhoek) ter lengte van *len* grafische punten naar boven en *len* naar rechts.

F *len*
Dit commando trekt een lijn in de voorgrondkleur naar rechts onder (315x t.o.v. de standaardhoek) ter lengte van *len* grafische punten naar onder en *len* naar rechts.

G *len*
Dit commando trekt een lijn in de voorgrondkleur naar links onder (225x t.o.v. de standaardhoek) ter lengte van *len* grafische punten naar onder en *len* naar links.

H *len*
Dit commando trekt een lijn in de voorgrondkleur naar links boven (135x t.o.v. de standaardhoek) ter lengte van *len* grafische punten naar boven en *len* naar links.

L *len*
Dit commando trekt een lijn in de voorgrondkleur naar links (180x t.o.v. de standaardhoek) ter lengte van *len* grafische punten.

M *x,y*
Dit commando trekt een lijn in de voorgrondkleur vanaf het laatst bereikte punt naar het punt aangeduid door de coördinaten *x* en *y*. Als *x* vooraf wordt gegaan door een plus- of minteken worden *x* en *y* opgevat als verschil t.o.v. het laatst bereikte punt, en wordt er een lijn getekend ter lengte van *x* punten horizontaal en *y* punten verticaal.

N
Dit commando heeft tot gevolg dat het eerstvolgende grafische commando waarin een lijn wordt getekend, wel wordt uitgevoerd, maar dat het punt dat vóór het uitvoeren van het commando bereikt was, behouden blijft als laatst bereikte punt. Het voorkomen van *N* wordt gesignaleerd in DRWFLG (FCBB).

R *len*
Dit commando trekt een ijn in de voorgrondkleur naar rechts (in de richting van de standaardhoek) ter lengte van *len* grafische punten.

S *schaal*
Schaal is een getal tussen 0-255 (inclusief grenzen). Na het uitvoeren van dit commando worden lijnen waarvan de lengte in grafische punten uitgedrukt is, aangepast aan *schaal*. Een 'grafisch punt' wordt omgezet ter waarde van *schaal*/4; d.w.z. dat *schaal* = 4 de normale lengte aangeeft. *Schaal* = 0 heeft ook de standaard lengte tot gevolg. De waarde van *schaal* wordt opgeslagen in DRWSCL (FCBC).

U *len*
Dit commando trekt een lijn in de voorgrondkleur naar boven (90x t.o.v. de standaardhoek) ter lengte van *len* grafische punten.

X *ident*;
Ident is de naam van een string-variabele. De inhoud van die variabele wordt opgevat als verzameling grafische commando's, en als zodanig uitgevoerd. De puntkomma kan niet weggelaten worden.

Alle velden in deze grafische commando's zijn gehele getallen binnen het integerbereik, tenzij anders aangegeven. Grafische commando's binnen eenzelfde DRAW-statement kunnen gescheiden worden door spaties of puntkomma's.

Voor alle getalwaarden van alle commando's kan ook een variabele gebruikt worden. Dit gebeurt door in plaats van de getalwaarde de tekst '*=ident;*' in de string op te nemen. *Ident* is de naam van de (numerieke) variabele waaruit de waarde gehaald wordt. De puntkomma is nu verplicht.

DRAW maakt gebruik van het grafisch systeem en de ingangen die tot dat systeem toegang geven; zie par. 6.2.8.

DRAW maakt gebruik van de volgende geheugenplaatsen:

F92A	CLOC	- adres in patroontabel van grafische cursor
F92C	CMASK	- bit-masker van grafische cursor
F942	CSAVEA	- opslagruimte voor CLOC
F944	CSAVEM	- opslagruimte voor CMASK
F956	MCLTAB	- voor het ontleden van de commando-string str
F958	MCLFLG	- geeft aan of ontleden is voor PLAY of DRAW
FB3B	MCLLEN	- voor het ontleden van de commando-string str
FB3C	MCLPTR	- voor het ontleden van de commando-string str
FCB3	GXPOS	- x-coördinaat van grafische cursor
FCB5	GYPOS	- y-coördinaat van grafische cursor
FCBB	DRWFLG	- signaleert B- en N-commando's
FCBC	DRWSCL	- laatst ingestelde waarde van S-commando
FCBD	DRWANG	- laatst ingestelde waarde van A-commando

N.B. DRAW werkt alleen in schermmodi 2 en 3.

DSKF FF+A6 (255+166)

Categorie functie
Zie ook DSKI$, DSKO$
Vorm DSKF (*diskettestation*)

Het argument *diskettestation* is een byte-uitdrukking die als waarde 0, 1 of 2 heeft. DSKF levert een geheel getal op, dat het aantal ongebruikte clusters op een diskette voorstelt. Een cluster komt overeen met 1 kbyte opslagcapaciteit. De waarde van *diskettestation* geeft aan welk diskettestation onderzocht moet worden:

0 huidig *diskettestation*
1 *diskettestation* A:
2 *diskettestation* B:

N.B. DSKF werkt alleen in DISK BASIC. De functie wordt geïmplementeerd via de RAM-haak H.DSKF (FE12).

DSKI$ EA (234)

Categorie functie
Zie ook DSKF, DSKO$
Vorm DSKI$ (*diskettestation,sect*)

Diskettestation is een integeruitdrukking die 0, 1 of 2 oplevert. Sect is een integeruitdrukking met als waarde een getal tussen 0 en 719 (inclusief grenzen).

De functie DSKI$ levert de lege string op. Als 'neveneffect' wordt tevens de inhoud van een sector (1/2 kbytes bytes) van de diskette overgeheveld naar het geheugen.

Diskettestation geeft aan welk diskettestation gebruikt wordt, volgens de volgende tabel:

0 huidig *diskettestation*
1 *diskettestation* A:
2 *diskettestation* B:

Sect geeft het nummer van de sector die gelezen wordt. Het beginadres van het geheugen waarin de sectorinhoud terecht komt, staat in adressen F351-F352.

N.B. DSKI$ werkt alleen in DISK BASIC. De functie wordt geïmplementeerd via de RAM-haak H.DSKI (FE17).

DSKO$ D1 (209)

Categorie statement
Zie ook DSKF, DSKI$
Vorm DSKO$ *diskettestation,sect*

Diskettestation is een integeruitdrukking die 0, 1 of 2 oplevert. *Sect* is een integeruitdrukking met als waarde een getal tussen 0 en 719 (inclusief grenzen).

Dit statement heeft als resultaat dat een blok van het geheugen, ter grootte van één sector (1/2 kbytes bytes), naar diskette wordt geschreven. Het beginadres van het geheugen dat weggeschreven wordt, staat in geheugenadressen F351-F352. Het sectornummer, te vinden in *sect,* bepaalt waar op de diskette het resultaat terecht komt.

Diskettestation geeft aan welk diskettestation gebruikt wordt, volgens de volgende tabel:

0 huidige *diskettestation*
1 *diskettestation* A:
2 *diskettestation* B:

N.B. DSKO$ werkt alleen in DISK BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.DSKO (FDEF).

ELSE 3A+A1 (58+161)

Categorie hulpwoord
Zie ook IF, THEN

ELSE wordt gebruikt in een IF-statement. Als de uitdrukking achter IF gelijk is aan 0, worden de statements achter ELSE uitgevoerd. In geneste IF-statements hoort bij elke IF maximaal één ELSE; zie IF.
ELSE in een IF-statement mag gevolgd worden door een regelnummer, in plaats van een serie statements. Deze situatie wordt behandeld alsof er tussen ELSE en het regelnummer het woord GOTO staat.

END 81 (129)

1.

Categorie statement
Zie ook STOP
Vorm END

Bij END houdt de uitvoering van een BASIC-programma op. Tevens worden alle open files gesloten. De computer komt terug in directe modus.
Als er nog een foutverwerkingsroutine in uitvoering is (niet afgesloten door RESUME), dan wordt de melding 'No RESUME' gegeven. Dit wordt gesignaleerd aan de hand van ONEFLG (F6BB).
Het verschil met STOP is, dat na het uitvoeren van END geen CONT-statement meer gegeven kan worden.
END maakt gebruik van de volgende geheugenplaatsen:

F6BB	ONEFLG	- geeft aan of foutroutine in uitvoering is
F6BE	OLDLIN	- nummer van laatst uitgevoerde regel
F6C0	OLDTXT	- adres van laatst uitgevoerde statement

2.

Categorie hulpwoord
Zie ook APPEND
Vorm APPEND

END wordt gebruikt om het sleutelwoord APPEND te coderen. APPEND wordt gecodeerd als de letters A+P+P gevolgd door het woord END.

EOF FF+AB (255+171)

Categorie functie
Zie ook CLOSE, OPEN
Vorm EOF (*fnr*)

Het argument *fnr* is een file-nummeruitdrukking.

De file met nummer *fnr* moet geopend zijn in de INPUT-modus, of zonder modus op te geven; zie OPEN. EOF levert –1 (waar) op als het einde van de file bereikt is, anders 0 (onwaar).

EOF maakt gebruik van de volgende geheugenplaatsen:

F860	FILTAB	- tabel met beginadressen file-buffers
F864	PTRFIL	- Beginadres van gebruikte file-buffer

EQV F9 (249)

Categorie operator
Zie ook AND, IMP, NOT, OR, XOR
Vorm *x* EQV *y*

De velden *x* en *y* zijn integeruitdrukkingen.
De operanden worden gecodeerd volgens 2-complement. Het resultaat van EQV is een getal, ook gecodeerd volgens 2-complement, waarvan elk bit direct afhankelijk is van het overeenkomstige bit in *x* en *y*.

Voor EQV geldt het volgende diagram:

bit van *x*:	0	0	1	1
bit van *y*:	0	1	0	1
bit van EQV:	1	0	0	1

EQV is een logische operator met prioriteit 2 (zie par. 7.2.4).

ERASE A5 (165)

Categorie statement
Zie ook DIM
Vorm ERASE ident,...

Ident is een variabelenaam. Het veld *ident* mag willekeurig vaak (1 of meer keer) voorkomen.
ERASE maakt het effect van een DIM-statement ongedaan. De ruimte die de rijvariabelen *ident* in beslag nemen in het opslaggebied voor rijvariabelen, gereserveerd in een DIM-statement, wordt weer vrijgemaakt. De variabelen kunnen opnieuw worden gedimensioneerd. Het opslaggebied schuift aaneen om de ruimte beschikbaar te maken.

ERASE gebruikt de volgende geheugenplaatsen:

F6C4	ARYTAB	- beginadres opslaggebied rijvariabelen
F6C6	STREND	- adres eerste vrije geheugenplaats

ERL E1 (225)

Categorie functie
Zie ook ERR, ERROR, RESUME
Vorm ERL

Deze functie levert het regelnummer op van de regel waar het laatst een fout opgetreden is (of gesimuleerd, zie ERROR). Deze fout hoeft niet gemeld te zijn, maar kan opgevangen zijn door middel van ON ERROR.
Als de laatst opgetreden fout in directe modus optrad, in plaats van in een programmaregel, dan levert ERL 65535 op. Als er sinds het aanzetten van de computer nog geen enkele fout is opgetreden, levert ERL 0 op.

Wanneer ERL in een (vergelijkings-)expressie wordt gebruikt, worden alle constanten die in de expressie op het woord ERL volgend, gecodeerd als regelnummerconstanten.
De waarde die door ERL wordt opgeleverd, wordt bewaard in ERRLIN (F6B3).

ERL gebruikt de volgende geheugenplaatsen:

F6B3	ERRLIN	- regelnummer laatst opgetreden fout

ERR E2 (226)

Categorie functie
Zie ook ERL, ERROR, RESUME
Vorm ERR

Deze functie levert het nummer op van de laatst opgetreden of gesimuleerde fout (zie ERROR). Dit kan een fout zijn die niet gemeld is, maar door middel van ON ERROR opgevangen.
Als er nog geen enkele fout is opgetreden sinds het aanzetten van de computer, levert ERR 0 op.
De waarde die door ERR wordt opgeleverd, wordt bewaard in ERRFLG (F414).

ERR gebruikt de volgende geheugenplaatsen:

F414	ERRFLG	- foutnummer laatst opgetreden fout

ERROR A6 (166)

1.

Categorie statement
Zie ook ERL, ERR, RESUME
Vorm ERROR *nr*

Het veld *nr* is een byte-uitdrukking.
ERROR simuleert de fout die de waarde van *nr* als foutnummer heeft. De functie

ERR zal, zolang er geen andere fout is opgetreden of gesimuleerd, de waarde van *nr* opleveren.

De waarde van *nr* wordt opgeslagen in ERRFLG (F414). Het nummer van de regel waar het statement staat, wordt opgeslagen in ERRLIN (F6B3).
Als bij het uitvoeren van ERROR een ON ERROR werkzaam is, wordt die uitgevoerd. Als er geen ON ERROR werkzaam is, wordt er een foutboodschap gegeven: als *nr* een bestaand foutnummer is, wordt de bijbehorende boodschap gegeven, anders wordt de boodschap 'Unprintable error' gegeven.

ERROR gebruikt de volgende geheugenplaatsen:

F414	ERRFLG	- foutnummer laatst opgetreden fout
F6B3	ERRLIN	- regelnummer laatst opgetreden fout
F6BB	ONEFLG	- geeft aan of ON ERROR werkzaam is

2.
Categorie hulpwoord
Zie ook ON
Vorm ON ERROR GOTO *regnr*

In deze betekenis wordt ERROR gebruikt als hulpwoord in een ON ERROR-statement.

EXP FF+8B (255+139)

Categorie functie
Zie ook LOG, SQR
Vorm EXP(x)

Het veld *x* is een numerieke uitdrukking.
De functie levert de e-macht van de waarde van *x* op.

FIELD B1 (177)

Categorie statement
Zie ook GET, OPEN, PUT
Vorm FIELD *fnr,veld,...*

Het veld *fnr* is een file-nummeruitdrukking, eventueel voorafgegaan door een hekje (#).

Veld is een omschrijving van een veld van een record. Een FIELD-statement kan willekeurig veel (1 of meer) elementen *veld* bevatten. Elk element *veld* ziet eruit als: *lengte* AS *ident*. Hierin is *lengte* een byte-uitdrukking. *Ident* is de naam van een string-variabele.
FIELD geeft toegang tot de buffer van een random access-file. De file waarop het

statement betrekking heeft, heeft als file-nummer de waarde van fnr, en moet geopend zijn zonder modus op te geven (zie OPEN).

Na het uitvoeren van een FIELD-statement geven de variabelen *ident* die genoemd zijn in een element *veld* toegang tot de file-buffer. Dit gebeurt door descriptor van de string-variabelen te laten wijzen naar het gebied van de buffer. De eerste variabele wijst naar het eerste byte van de buffer; de volgende naar byte 1 plus de lengte van de eerste variabele; enz. De lengte van elke variabele wordt gegeven door *lengte*. De lengte van alle variabelen samen mag niet groter zijn dan de lengte van een record, opgegeven bij OPEN.
Door de string-variabelen te lezen, of er een nieuwe waarde in te zetten, wordt de inhoud van het record gelezen dat zich op dat moment in de buffer bevindt, of wordt er een nieuwe inhoud aan gegeven. Het record in de file-buffer kan van en naar diskette worden geschreven met GET en PUT.

Het toewijzen van nieuwe waarden aan een string-variabele van het FIELD-statement moet niet gebeuren met een 'gewone' (LET-) toewijzing; in plaats daarvan moeten LSET, RSET of MID$ gebruikt worden. Dit omdat bij een gewone toewijzing de nieuwe waarde in de string-opslagruimte wordt gezet, en de string-descriptor daarheen gaat wijzen.

De variabelen in het FIELD-statement zijn string-variabelen. Dat betekent dat getalwaarden niet zonder meer in een random access-file verwerkt kunnen worden, maar eerst van of naar een string moeten worden omgezet. Het omzetten van een getal naar een stringwaarde gebeurt met de functies MKD$, MDI$ en MKS$; de omgekeerde omzetting met CVD, CVI en CVS.
Voor dezelfde file kunnen verscheidene FIELD-statements worden uitgevoerd. Deze blijven dan tegelijkertijd werkzaam.

N.B. FIELD werkt alleen in DISK BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.FIEL (FE2B)

FILES B7 (183)

1.

Categorie statement
Zie ook LFILES
Vorm FILES *fnaam*

Het veld *fnaam* is een file-uitdrukking. De apparaatnaam in de waarde van *fnaam*, indien aanwezig, moet een diskettestation zijn. *Fnaam* mag weggelaten worden.
Dit statement geeft een overzicht van alle files die door fnaam aangeduid worden. Als *fnaam* weggelaten wordt, geeft FILES een overzicht van alle files op het huidig geselecteerde *diskettestation*.

Een overzicht bestaat uit de naam van alle betreffende files, gevolgd door een tab (ASCII-code 9).

N.B. FILES werkt alleen in DISK BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.FILE (FE7B)

FILES gebruikt de OS-routine:

0018	OUTDO	- schrijft een teken naar het scherm

FILES gebruikt de volgende geheugenplaatsen:

F3B0	LINLEN	- aantal tekens op een regel
F3B1	CRTCNT	- aantal regels van een scherm
F3B2	CLMLST	- aantal tekens tussen tabulatorstops
F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)

2.

Categorie hulpwoord
Zie ook MAX, MAXFILES
Vorm MAXFILES

FILES wordt gebruikt om het woord MAXFILES te helpen coderen. MAXFILES wordt gecodeerd als het woord MAX gevolgd door het woord FILES.

FIX FF+A1 (255+161)

Categorie functie
Zie ook CINT, INT
Vorm FIX(*x*)

Het argument *x* is een numerieke uitdrukking.
Het resultaat van de functie is de waarde van *x*, zonder de decimale fractie. FIX verschilt van INT in die mate dat voor negatieve getallen het resultaat groter of gelijk is aan de waarde van *x*, in plaats van kleiner of gelijk. Voor alle *x* geldt: FIX(*x*)=SGN(*x*)*INT(ABS(*x*))

FN DE (222)

1.

Categorie hulpwoord
Zie ook DEF

FN wordt gebruikt in de declaratie van een zelf-gedefinieerde BASIC-functie; zie DEF.

2.

Categorie hulpwoord
Zie ook DEF
Vorm *FNident(arg)*

Ident is een functienaam, in vorm gelijk aan een variabelenaam. *Arg* is een lijst van uitdrukkingen, gescheiden door komma's. *Arg* kan worden weggelaten, mits de omsluitende haakjes ook worden weggelaten.

De definitie van een functie is vastgelegd bij DEF. Een functie die met behulp van DEF gedefinieerd is, kan worden gebruikt op alle plaatsen waar een gewone uitdrukking kan worden gebruikt. Dit gebruik wordt het 'aanroepen' van een functie genoemd; zie par. 7.2.6.
Ident is de naam waaronder de functie is gedeclareerd; zie DEF. Het veld *arg* moet overeenkomen met het veld *arg* van de declaratie, in zoverre dat voor elke variabele in de declaratie een uitdrukking in de aanroep moet voorkomen, van hetzelfde type als de variabele (string- of getalwaarde).
De uitdrukkingen in *args* worden de parameters van de functie genoemd. Bij het aanroepen van de functie wordt aan elk van de variabelen in de definitie de waarde van de bijbehorende parameter in de aanroep toegekend. Daarna wordt de functie-uitdrukking uitgerekend, en het resultaat daarvan is het resultaat van de functie-aanroep.

Voor het uitvoeren van een functie-aanroep worden de volgende geheugenplaatsen gebruikt:

F6E4	PRMSTK	- adres van vorig parameterblok op de stapel
F6E6	PRMLEN	- aantal geldige bytes in PARM1
F6E8	PARM1	- huidig parameterblok
F74C	PRMPRV	- opslagruimte voor PRMSTK
F74E	PRMLN2	- aantal geldige bytes in PARM2
F750	PARM2	- parameterblok in opbouw
F7B4	PRMFLG	- wordt gebruikt bij opzoeken variabele
F7B5	ARYTA2	- gelijk aan ARYTAB (F6C4) of eind van PARM1
F7B7	NOFUNS	- geeft aan of er functies actief zijn
F7B8	TEMP9	- wordt gebruikt bij garbage collection
F7BA	FUNACT	- aantal actieve functies

FOR 82 (130)

1.

Categorie statement
Zie ook NEXT, STEP, TO
Vorm FOR *ident=ogr* TO *bgr* STEP *stap*

Ident is de naam van een numerieke variabele. *Ogr, bgr* en *stap* zijn numerieke uitdrukkingen. Stap mag weggelaten worden, mits het woord STEP ook weggelaten wordt.
Het FOR-statement vormt de helft van een FOR-NEXT besturingsstructuur. De variabele *ident* is de lusteller. Deze teller doorloopt een reeks van waarden, beginnend bij *ogr* en eindigend bij *bgr*.

Wanneer FOR in een BASIC-programma wordt gevonden, krijgt *ident* de waarde van *ogr,* en wordt op de stapel een descriptor van het statement gezet waarin het adres van de opslag van *ident,* de waarde van *bgr* en de waarde van *stap* zijn opgenomen.
Het programma gaat verder met het volgende statement. Wanneer de bijbehorende NEXT-instructie gevonden wordt, zal de waarde van *stap* (te vinden in de descriptor op de stapel) worden opgeteld bij *ident.* Als de resulterende waarde *bgr* heeft gepasseerd, gaat het programma verder met het eerste statement na NEXT, anders met het eerste statement na FOR.

Merk op dat dit inhoudt dat de statements tussen FOR en NEXT altijd tenminste één keer doorlopen worden, zelfs als dat gezien de waarden van *ogr, bgr* en *stap* niet zou moeten, zoals in het volgende voorbeeld:

```
FOR X=2 TO 1 STEP 2
```

Als *stap* weggelaten wordt, wordt voor de stapgrootte 1 genomen.

FOR maakt gebruik van de volgende geheugenplaatsen:

F6A1	ENDFOR	- adres van eerste statement na FOR

2.

Categorie hulpwoord
Zie ook APPEND, INPUT, OPEN, OUTPUT
Vorm FOR *modus*

In een OPEN-statement kan een veld FOR *modus* voorkomen. Dit geeft aan op welke manier de file gelezen en geschreven kan worden. *Modus* kan de volgende waarden hebben:

APPEND	- sequentiële uitvoer achter bestaande file
INPUT	- sequentiële invoer-file
OUTPUT	- sequentiële uitvoer-file

Als het veld FOR *modus* weggelaten wordt uit het OPEN-statement, betekent het dat de file een random access-file is.

FPOS FF+A7 (255+167)

Categorie functie
Vorm FPOS(*fnr*)

Het veld *fnr* is een file-nummeruitdrukking.
Het is waarschijnlijk dat deze functie bedoeld is om informatie omtrent een file op te leveren. In het huidige MSX-systeem levert de functie echter een 'Internal error' op. De functie kan dan ook nergens voor gebruikt worden.

In de functie FPOS is voorzien door de RAM-haak H.FPOS (FEA8).

FRE FF+8F (255+143)

Categorie functie
Zie ook CLEAR, MAXFILES
Vorm FRE(*x*)

Het argument *x* kan een string- of een numerieke uitdrukking zijn. De waarde van *x* doet er niet toe.
Als *x* een numerieke uitdrukking is, levert de functie het totaal aantal bytes op dat nog vrij is voor BASIC en variabelen. Dit aantal is gelijk aan het verschil tussen de waarde van STREND (F6C6) en de waarde van STKTOP (F674).

Als *x* een string-uitdrukking is, levert FRE het aantal bytes op dat nog vrij is voor het opslaan van string-waarden. Dit aantal is gelijk aan het verschil tussen de waarde van MEMSIZ (F672) en de waarde van STKTOP (F674).
Zolang er geen teksten zijn opgeslagen, is het aantal vrije string-bytes gelijk aan 200. Het aantal bytes dat in het begin vrij is voor BASIC hangt af van de totale hoeveelheid RAM in de computer en van de aangesloten randapparatuur.

FRE gebruikt de volgende geheugenplaatsen:

F672	MEMSIZ	- hoogste adres van het string-opslaggebied
F674	STKTOP	- hoogste adres van het stapelgebied
F69B	FRETOP	- eerste vrije plaats in string-opslaggebied
F6C6	STREND	- adres van de eerste vrije geheugenplaats

GET B2 (178)

Categorie statement
Zie ook FIELD, PUT
Vorm GET *fnr,recnr*

Het veld *fnr* is een file-nummeruitdrukking, eventueel voorafgegaan door een hekje (#). *Recnr* is een integeruitdrukking. *Recnr* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.

GET leest een record van de file met nummer *fnr*, en zet de inhoud ervan in de filebuffer. Deze buffer kan uitgelezen worden door gebruik te maken van de stringvariabelen die zijn opgegeven in een FIELD-instructie; zie FIELD.
De file moet geopend zijn als random access-file (zie OPEN). *Recnr* is het nummer van het record dat gelezen wordt. De plaats van het record binnen de file wordt bepaald door dit nummer met de lengte van een record (opgegeven bij OPEN) te vermenigvuldigen, en het resultaat te gebruiken als afstand vanaf het begin van de file.

Als *recnr* weggelaten wordt, zal het volgende record uit de file worden ingelezen. Het resultaat is, dat de file zich gedraagt als een sequentiële file.

GET maakt gebruik van de volgende geheugenplaatsen:

F860	FILTAB	- beginadres van file-tabel
F864	PTRFIL	- adres van gebruikte file-buffer

N.B. GET werkt alleen in DISK BASIC.

GOSUB 8D (141)

Categorie statement
Zie ook GOTO, RETURN
Vorm GOSUB *regnr*

Het veld *regnr* is een regelnummerconstante.
Na uitvoering van dit statement gaat de computer verder met het eerste statement op regel *regnr*. Het regelnummer en adres van het statement na GOSUB worden bewaard op de stapel, en gebruikt bij het uitvoeren van RETURN.

GOTO 89 (137)

Categorie statement
Zie ook GOSUB
Vorm GOTO *regnr*

Het veld *regnr* is een regelnummerconstante.
Na uitvoering van dit statement gaat de computer verder met het eerste statement op regel regnr.
Het woord GOTO kan overigens ook worden ingetypt als twee losse woorden: GO TO. In de listing van het programma zal het dan echter toch verschijnen als één woord.

HEX$ FF+9B (255+155)

Categorie functie
Zie ook BIN$, OCT$, STR$
Vorm HEX$($x$)

Het veld x is een integeruitdrukking.
HEX$ levert een string op die de hexadecimale representatie van de waarde van x bevat.

IF 8B (139)

Categorie statement
Zie ook THEN, ELSE
Vorm 1. IF *voorw* THEN *stats1* ELSE *stats2*
2. IF *voorw* GOTO *regnr* ELSE *stats2*

Voorw is een numerieke expressie. *Stats1* en *stats2* zijn series (0 of meer) statements, gescheiden door dubbele punten. Anderzijds kunnen *stats1* en *stats2* ook regelnummers zijn. Regnr is een regelnummer. Het ELSE-gedeelte (ELSE *stats2*) kan weggelaten worden.

Vorm 2 is precies equivalent aan vorm 1, waarbij *stats1* vervangen is door GOTO *regnr*.
Uitvoering van een IF-statement resulteert in uitvoering van *stats1* of uitvoering van *stats2*. Welke statements uitgevoerd worden, hangt af van *voorw:* als de waarde van deze uitdrukking 0 is, dan wordt *stats2* uitgevoerd, anders *stats1*.
Als het ELSE-gedeelte weggelaten wordt en *voorw* levert 0 op, gaat de computer verder met het eerste statement op de volgende regel in het programma.

Als *stats1* (of *stats2*) een regelnummer is, heeft dit precies hetzelfde effect als wanneer *stats1* (of *stats2*) gelijk zou zijn aan GOTO *stats1* (of GOTO *stats2*). Het regelnummer wordt in dat geval gecodeerd als regelnummerconstante, waardoor het onder andere mee wordt hernummerd bij een RENUM-statement. Zie ook par. 7.2.5.

Als zich tussen THEN en ELSE een ingebed ('genest') tweede IF-statement bevindt, moet dat statement in ieder geval een ELSE-gedeelte bevatten. Dit om te voorkomen dat een IF-statement voor twee interpretaties vatbaar zou zijn.

IMP FA (250)

Categorie operator
Zie ook AND, EQV, NOT, OR, XOR
Vorm *x* IMP *y*

De operatoren *x* en *y* zijn integeruitdrukkingen.
De operanden worden gecodeerd volgens 2-complement. Het resultaat van IMP is een getal, ook gecodeerd volgens 2-complement, waarvan elk bit direct afhankelijk is van het overeenkomstige bit in *x* en *y*.

Voor IMP geldt het volgende diagram:

bit van *x*:	0	0	1	1
bit van *y*:	0	1	0	1
bit van IMP:	1	1	0	1

IMP is een logische operator met prioriteit 1. Zie par. 7.2.4.

INKEY$ EC (236)

Categorie functie
Zie ook INPUT
Vorm INKEY$

De functie INKEY$ levert het eerste teken in de invoerbuffer van het toetsenbord op. Als de invoerbuffer leeg is, levert INKEY$ de lege string op.

INKEY$ maakt gebruik van de volgende OS-routines:

009C	CHSNS	- test of de toetsenbordbuffer leeg is
009F	CHGET	- haalt een code uit de toetsenbordbuffer

INKEY$ maakt gebruik van de volgende geheugenplaatsen:

F3F8	PUTPNT	- adres van eerste lege plaats in KEYBUF
F3FA	GETPNT	- adres van voorste teken in KEYBUF
FBF0	KEYBUF	- ruimte van toetsenbordbuffer

INP FF+90 (255+144)

Categorie functie
Zie ook OUT, WAIT
Vorm INP(x)

Het veld *x* is een byte-uitdrukking.
De functie INP leest de I/O-poort met poortadres *x*, en levert de gelezen waarde op.

INPUT 85 (133)

1.

Categorie statement
Zie ook PRINT
Vorm INPUT *tekst;ident,...*

Het veld *tekst* is een string-constante. *Ident* is een variabelenaam. Het veld ident kan een willekeurige aantal (minstens 1) keer voorkomen.
INPUT verzorgt het invoeren van gegevens vanaf het toetsenbord. Eerst wordt de waarde van *tekst* op het scherm gezet op de wijze van PRINT, gevolgd door een vraagteken en een spatie; daarna wordt de scherm-editor aangeroepen; zie par. 6.2.7. Aangezien de scherm-editor alleen in een tekstmodus kan werken, wordt eventueel eerst de schermmodus veranderd door middel van de OS-routine TOTEXT (ingang 00D2).

Na het uitvoeren van het INPUT-statement – dus wanneer de BASIC-interpreter weer werkzaam is, en verder gaat met het volgende statement – hebben alle variabelen *ident* die bij het statement stonden een waarde gekregen.
Tijdens het intypen van het antwoord kan van alle mogelijkheden van de scherm-editor gebruik worden gemaakt; zie par. 6.2.7.

De positie van de cursor bij het binnen gaan van de scherm-editor wordt opgeslagen in FSTPOS (FBCA). Als bij het geven van RETURN de cursor op een andere logische regel staat dan bij het begin van het INPUT-statement (te zien aan het verschil tussen CSRY en het eerste byte van FSTPOS), wordt de hele logische regel waarop de cursor staat als invoer genomen. Als de cursor nog op dezelfde logische regel staat, wordt de waarde van *tekst* van die invoer afgetrokken.

Uit de ingevoerde tekst die door de scherm-editor wordt opgeleverd, worden de nieuwe waarden voor de variabelen *ident* geconstrueerd. De waarde voor twee verschillende variabelen moet gescheiden zijn door een komma.
Als er een string-waarde wordt ingevoerd, hoeft deze waarde niet tussen aanhalingstekens te staan, tenzij er komma's in staan. Ingevoerde getalwaarden kunnen op elke manier gerepresenteerd worden. Spaties in een getalwaarde en aan het begin en eind van een string-waarde worden genegeerd.

Voor elke variabele *ident* in het INPUT-statement moet er een nieuwe waarde in de ingevoerde tekst staan. Als er niet genoeg antwoorden zijn gegeven (gescheiden door komma's) verschijnt er op het scherm een dubbel vraagteken, en moet alsnog de rest worden ingetypt. Als er teveel antwoorden zijn gegeven, wordt de waarschuwing '?Extra Ignored' gegeven, maar gaat de uitvoering van het programma verder. Als er string-waarden zijn ingevoerd waar getalwaarden werden verwacht, wordt de melding '?Redo from start' gegeven, en begint de uitvoering van het INPUT-statement van voren af aan.

INPUT maakt gebruikt van de volgende routines:

0018	OUTDO	- om *tekst* op het scherm te zetten
00B4	QINLIN	- om de scherm-editor aan te roepen
00D2	TOTEXT	- om het scherm in tekstmodus te zetten

INPUT maakt gebruik van de volgende geheugenadressen:

F3B0	LINLEN	- aantal kolommen in een regel
F3B1	CRTCNT	- aantal regels op het scherm
F3B2	CLMLST	- aantal posities tussen tabulatorstops
F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)
F6A6	INPFLG	- geeft aan dat INPUT aan de gang is
FBB2	LINTTB	- verband logische regels met schermregels
FBCA	FSTPOS	- cursorpositie bij aanroep QINLIN
FCAF	SCRMOD	- huidige schermmodus
FCB0	OLDSCR	- laatst gebruikte tekstmodus

2.
Categorie statement
Zie ook PRINT
Vorm INPUT *#fnr; ident,...*

Het veld *fnr* is een file-nummeruitdrukking. *Ident* is een variabelenaam. Het veld *ident* kan willekeurig vaak (minstens één) keer voorkomen; de voorkomens zijn dan gescheiden door komma's.
In deze vorm leest INPUT gegevens uit een file. De file is geopend onder nummer *fnr*, als sequentiële invoer-file (zie OPEN). INPUT leest alle tekens uit de file tot en met de eerstvolgende ASCII-code &H0D (RETURN). De gelezen tekens vormen

samen een string, waaruit de nieuwe waarden voor de variabelen *ident* bepaald worden op dezelfde manier als in de eerste betekenis van INPUT.

INPUT maakt gebruikt van de geheugenplaatsen:

F860	FILTAB	- tabel met beginadressen van file-buffers
F864	PTRFIL	- adres van gebruikte file-buffer

3.

Categorie hulpwoord
Zie ook INPUT$
Vorm INPUT$

INPUT wordt in deze betekenis gebruikt om het sleutelwoord INPUT$ te coderen.

4.

Categorie hulpwoord
Zie ook LINE

In deze betekenis is INPUT een hulpwoord bij het LINE INPUT-statement, dat overigens een variatie is op betekenis 1 van INPUT.

INPUT$ 85+24 (133+36)

Categorie functie
Zie ook INKEY$
Vorm 1. INPUT$(*len*)
2. INPUT$(*len,fnr*)

Het argument *len* is een byte-uitdrukking. Het argument *fnr* is een file-nummer-uitdrukking.
INPUT$ levert een string op met lengte *len*. Die string bestaat uit de eerste *len* bytes die ingelezen zijn van de toetsenbordbuffer (in vorm 1) of van de file met nummer *fnr* (in vorm 2).
In vorm 1 duurt de uitvoering van de functie net zolang tot er *len* tekens ingetypt zijn.

De file met nummer *fnr* in vorm 2 moet een sequentiële invoerfile zijn (zie OPEN). Als *len* groter is dan het aantal overgebleven bytes in de file, wordt de foutmelding 'Input past end' gegeven. Deze melding wordt ook gegeven als zich ergens in de invoer ASCII-code &H1A (26) bevindt.

INPUT$ maakt gebruik van de OS-routine:

009F	CHGET	- wacht op invoer in de toetsenbordbuffer
014A	ISFLIO	- test of invoer uit file moet komen (vorm 2)

INPUT$ maakt gebruik van de volgende geheugenplaatsen:

F3FA	GETPNT	- adres van eerste teken in KEYBUF
F860	FILTAB	- tabel met beginadressen van file-buffers
F864	PTRFIL	- adres van gebruikte file-buffer
FBF0	KEYBUF	- geheugengebied van toetsenbordbuffer

INSTR E5 (229)

Categorie functie
Zie ook LEFT$, LEN, MID$, RIGHT$
Vorm INSTR(*str,patr,pos*)

De velden *str* en *patr* zijn string-uitdrukkingen. *Pos* is een byte-uitdrukking. *Pos* mag weggelaten worden, mits de voorafgaande komma ook wordt weggelaten.

De functie levert een positief geheel getal op, dat de positie binnen *str* aangeeft waar *patr* begint. Dat wil zeggen, als de serie tekens van *patr* in zijn geheel binnen *str* voorkomt, levert INSTR nummer op van het teken in *str* dat overeenkomt met het eerste teken van *patr*. Als *patr* niet in *str* voorkomt, levert INSTR 0 op.
De waarde van pos geeft aan vanaf welke positie in *str* gezocht moet worden naar *patr*. Als *pos* weggelaten wordt, wordt er gezocht vanaf het eerste teken van *str*.

INT FF+85 (255+133)

1.

Categorie functie
Zie ook CINT, FIX
Vorm INT(*x*)

Het veld *x* is een numerieke expressie.
INT levert de waarde van het gehele deel van *x* op, ofte wel het grootste gehele getal dat kleiner is dan *x*. Dit betekent dat de INT van een negatief getal in absolute waarde groter is dan het getal zelf; dit in tegenstelling tot FIX.

2.

Categorie hulpwoord
Zie ook INTERVAL, VAL
Vorm INTERVAL

Het woord INT wordt gebruikt om het woord INTERVAL intern te coderen. INTERVAL wordt gecodeerd als het woord INT, gevolgd door de letters E+R, gevolgd door het woord VAL.

INTERVAL

FF+85+45+52+FF+94
(255+133+69+81+255+148)

1.

Categorie statement
Zie ook KEY, ON, SPRITE, STOP, STRIG
Vorm INTERVAL *modus*

Modus kan de volgende waarden aannemen: ON, OFF en STOP.
Dit statement regelt de reactie op het verstrijken van een tijdsduur. MSX-BASIC kent de mogelijkheid om na een van tevoren vastgestelde tijdsduur een (software-gestuurde) interrupt te genereren; deze interrupt kan gebruikt worden d.m.v. een ON INTERVAL-statement om naar een bepaalde plaats in een BASIC-programma te springen.

De interrupt wordt ingeschakeld met INTERVAL ON en uitgeschakeld met INTERVAL OFF. INTERVAL STOP zorgt ervoor dat de gebeurtenis van het verstrijken van de tijdsduur niet direct leidt tot een interrupt, maar onthouden wordt tot de interrupt weer ingeschakeld wordt met INTERVAL ON.
Voor meer gegevens over het interrupt-systeem zie par. 6.2.2.

INTERVAL gebruikt een element van het geheugengebied:

FC4C TRPTBL - instelling van de interruptfaciliteit

2.

Categorie hulpwoord
Zie ook ON

In deze betekenis wordt INTERVAL gebruikt als hulpwoord in het ON INTERVAL-statement; zie ON.

IPL D5 (213)

Categorie statement
Zie ook ATTR$, CMD, SET

Dit statement wordt in de huidige systemen niet gebruikt; het sleutelwoord is toegevoegd voor toekomstig gebruik.
In het statement IPL is voorzien door middel van de RAM-haak H.IPL (FE03).

KEY CC (204)

1.

Categorie statement
Vorm KEY modus

Het veld *modus* kan drie vormen aannemen: ON, OFF en LIST.
In deze betekenis kan KEY gebruikt worden om het overzicht van de definities van de functietoetsen te besturen. De betekenis van de verschillende vormen van *modus* is als volgt:

KEY ON
Deze vorm van het KEY-statement heeft precies hetzelfde effect als de OS-routine DSPFNK (00CF).

Als het scherm in tekstmodus is, verschijnt op de onderste schermregel een overzicht van de huidige definitie van de functietoetsen.
De onderste schermregel schuift niet mee als de tekst op het scherm een regel omhoog schuift. Ook kan de onderste regel niet met een PRINT-statement beschreven worden.

KEY OFF
Deze vorm van het KEY-statement heeft precies hetzelfde effect als de OS-routine ERAFNK (00CC).

Na het uitvoeren van KEY OFF is het overzicht op de onderste regel verdwenen, en gedraagt de onderste regel zich als alle andere schermregels.
Bij het aanzetten van de computer is het overzicht ingeschakeld.

KEY LIST
Het statement KEY LIST geeft op een andere manier een overzicht van de functietoetsen, namelijk door alle definities onder elkaar op het scherm te zetten. Deze vorm is handig om de definities te veranderen.

In deze vorm gebruikt KEY de volgende OS-routines:

00CC	ERAFNK	- verwijdert de definities van het scherm
00CF	DSPFNK	- zet de definities op het scherm

De volgende geheugenplaatsen worden gebruikt:

F3DE	CNSDFG	- geeft aan of definities ingeschakeld zijn
F87F	FNKSTR	- tekst van functietoets-definities
FCAF	SCRMOD	- huidige schermmodus

2.

Categorie statement
Vorm KEY *nr,str*

Nr is een byte-uitdrukking met waarde tussen 1 en 10 (inclusief grenzen). Str is een string-uitdrukking.
Na het uitvoeren van dit statement is de functietoets met nummer *nr* gedefinieerd volgens str. Dat wil zeggen dat bij het indrukken van deze functietoets de waarde van *str* op het scherm verschijnt.
Een functietoets-definitie kan maximaal 15 tekens lang zijn. Van de waarde van *str*

worden alleen de eerste 15 tekens gebruikt. Alle tekens die daarop volgen worden genegeerd.
In deze betekenis gebruikt KEY het geheugengebied:

F87F	FNKST	- tekst van de functietoets-definities

3.

Categorie statement
Zie ook INTERVAL, ON, SPRITE, STOP, STRIG
Vorm KEY *(nr) modus*

Nr is een integeruitdrukking met waarde tussen 1 en 10 (inclusief grenzen). *Modus* kan de volgende waarden aannemen: ON, OFF en STOP.
Dit statement regelt de reactie op het indrukken van een functietoets. MSX-BASIC kent de mogelijkheid om bij het indrukken van zo'n toets een (software-gestuurde) interrupt te genereren; deze interrupt kan gebruikt worden d.m.v. een ON KEY-statement om naar een bepaalde plaats in een BASIC-programma te springen.

Nr is het nummer van de bedoelde functietoets. De interrupt-faciliteit wordt ingeschakeld met KEY (*nr*) ON, en weer uitgeschakeld met KEY (*nr*) OFF. KEY (*nr*) STOP zorgt ervoor dat de gebeurtenis van het indrukken van de toets niet direct leidt tot een interrupt, maar onthouden wordt tot de faciliteit weer ingeschakeld wordt met KEY (nr) ON.
Als de faciliteit niet ingeschakeld is, heeft het indrukken van een functietoets het gewone effect: het genereren van tekst.

KEY gebruikt in deze betekenis de geheugengebieden:

FBCE	FNKFLG	- geeft aan of een toets interrupts genereert
FCAC	TRBTBL	- geeft modus van interrupt en sprongadres

4.

Categorie hulpwoord
Zie ook ON

In deze betekenis wordt KEY gebruikt als hulpwoord in het ON KEY-statement; zie ON.

KILL D4 (212)

Categorie statement
Zie ook FILES, LOAD, SAVE
Vorm KILL *fnaam*

Fnaam is een file-uitdrukking. De apparaatnaam in *fnaam,* indien aanwezig, moet de naam zijn van een diskettestation.
KILL verwijdert alle files die door *fnaam* worden aangeduid van diskette. Als *fnaam* geen apparaatnaam bevat, wordt het huidige diskettestation gekozen.

KILL gebruikt de volgende geheugenplaatsen:

F866	FILNAM	- file-naam (waarde van *fnaam*)

N.B. KILL werkt alleen in DISK BASIC. Het statement KILL is geïmplementeerd via de RAM-haak H.KILL (FDFE).

LEFT$ FF+81 (255+129)

Categorie functie
Zie ook INSTR, LEN, MID$, RIGHT$
Vorm LEFT$*(str,len)*

Het argument *str* is een string-uitdrukking. *Len* is een byte-uitdrukking.
De functie levert een substring van de waarde van *str* op. Deze substring bestaat uit de voorste (meest linkse) tekens van *str,* en heeft als lengte het minimum van de lengte van *str* en de waarde van *len.*

LEN FF+92 (255+146)

Categorie functie
Zie ook INSTR, LEFT$, MID$, RIGHT$
Vorm LEN*(str)*

Het argument *str* is een string-uitdrukking.

LEN levert de lengte (het aantal tekens) van *str* op.

LET 88 (136)

Categorie statement
Zie ook MID$, LSET, RSET, SWAP
Vorm LET *doel=uitdr*

Doel is een variabele-omschrijving; dit kan een gewone variabele zijn, een rij-element, een systeemvariabele of een substring (zie MID$). *Uitdr* is een uitdrukking die in type (getal of string) overeenstemt met het type van *doel.*
Het resultaat van LET is dat de waarde van *uitdr* wordt toegekend aan *doel.* Als *doel* een nog niet eerder gebruikte variabele is, wordt er eerst ruimte gereserveerd voor de variabele. Als *doel* een element van een nog niet eerder gebruikte rij is, wordt er ruimte voor de rij gereserveerd; zie DIM.

Als *uitdr* een string-uitdrukking is, wordt er ruimte voor de string vrijgemaakt in het string-opslaggebied, tenzij *doel* een substring is (zie MID$); in dat geval wordt de bestaande ruimte gebruikt. Als *doel* geen substring is en er is geen string-ruimte over, dan leidt dit tot een foutmelding.

LET kan de volgende geheugenplaatsen veranderen:

F69B	FRETOP	- eerste vrije plaats in string-opslaggebied
F6C4	ARYTAB	- beginadres van rijvariabelen-opslaggebied
F6C6	STREND	- adres van eerste vrije geheugenplaats

LFILES BB (187)

Categorie statement
Zie ook FILES
Vorm LFILES *fnaam*

Fnaam is een file-uitdrukking. *Fnaam* mag worden weggelaten. De apparaatnaam in *fnaam*, indien aanwezig, moet een diskettestation zijn.

Dit statement geeft een overzicht op de printer van alle files die door *fnaam* aangeduid worden. Dit gebeurt door de waarde van PRTFLG (F416) hoog te maken, zodat de OS-routine OUTDO (ingang 0018) naar de printer schrijft, met behulp van de OS-routine OUTDLP (ingang 014D), in plaats van naar het scherm. Voor het overige werkt LFILES precies als FILES.

LFILES maakt gebruik van de OS-routines:

0018	OUTDO	- schrijft een teken
014D	OUTDLP	- schrijft een teken naar de printer

LFILES gebruikt de volgende geheugenplaatsen:

F415	LPTPOS	- aantal geprinte tekens sinds vorige RETURN
F416	PRTFLG	- geeft aan dat uitvoer naar de printer gaat
F417	NTMSXP	- geeft aan of de printer een MSX-printer is
F418	RAWPRT	- geeft aan of uitvoer geïnterpreteerd moet
F866	FILNAM	- file-naam (waard van *fnaam*)

N.B. LFILES werkt alleen in DISK BASIC. Het statement is geïmplementeerd via de RAM-haak H.FILE (FE7B).

LINE AF (175)

1.

Categorie statement
Zie ook CIRCLE, DRAW, PAINT, PRESET, PSET
Vorm LINE *begin-eind,kleur,blok*

Begin en *eind* zijn coördinaatuitdrukkingen. *Kleur* is een kleuruitdrukking. *Blok* is een veld dat als waarde B of BF kan hebben. *Begin, kleur* en *blok* mogen weggelaten worden. Als door het weglaten van velden het statement op komma's eindigt, moeten deze komma's ook weggelaten worden.

LINE tekent een lijn of een rechthoek. *Begin* en *eind* geven het begin- en eindpunt van de lijn of twee schuin tegenover elkaar liggende hoekpunten van de rechthoek. Als *begin* weggelaten is, wordt er voor het beginpunt (of het ene hoekpunt) het laatst bereikte punt genomen, ofte wel het punt dat staat in de grafische cursorcoördinaten GXPOS (FCB3) en GYPOS (FCB5).
De kleur waarin de lijn of rechthoek getekend wordt, wordt bepaald door *kleur*. Als *kleur* weggelaten is, wordt de huidige voorgrondkleur genomen, bewaard in FORCLR (F3E9). De waarde van *kleur*, of de waarde van FORCLR, komt te staan in ATRBYT (F3F2).

Als *blok* is weggelaten wordt een lijn getekend. Als voor blok B is ingevuld, wordt de omtrek van een rechthoek getekend. Als er BF is ingevuld, wordt de rechthoek opgevuld in dezelfde kleur.
LINE maakt gebruik van het grafische systeem (par. 6.2.8) en de OS-routines die daarvan deel uitmaken.

De geheugenplaatsen die LINE gebruikt zijn:

F3E9	FORCLR	- de huidige voorgrondkleur
F3F2	ATRBYT	- de werkkleur
F92A	CLOC	- adres grafische cursor (in de patroontabel)
F92C	CMASK	- bitmasker grafische cursor
FCB3	GXPOS	- x-coördinaat grafische cursor
FCB5	GYPOS	- y-coördinaat grafische cursor
FCB7	GRPACX	- constructieplaats nieuwe x-coördinaat
FCB9	GRPACY	- constructieplaats nieuwe y-coördinaat

N.B. LINE werkt alleen in schermmodus 2 of 3.

2.

Categorie statement
Zie ook INPUT
Vorm LINE INPUT *tekst;ident*

Tekst is een string-constante. *Ident* is de naam van een string-variabele, of element van een rijvariabele van type string.
Het resultaat van LINE INPUT is, dat waarde van *tekst* op het scherm verschijnt, waarna de scherm-editor aangeroepen wordt door een aanroep van INLIN (ingang 00B1). Als het scherm in een grafische modus stond, wordt het eerst in een tekst-modus gezet met behulp van de OS-routine TOTEXT (ingang 00D2).
Bij het invoeren van het antwoord op het LINE INPUT-statement kan van alle mogelijkheden van de scherm-editor gebruik worden gemaakt; zie par. 6.2.7.
De tekst die de scherm-editor oplevert bij het verlaten van INLIN is de string-waarde die aan *ident* wordt toegewezen.

LINE gebruikt in deze betekenis de volgende OS-routines:

0018	OUTDO	- om tekst op het scherm te schrijven
00B1	INLIN	- de scherm-editor
00D2	TOTEXT	- om het scherm in tekstmodus te zetten

LINE maakt gebruik van de volgende geheugenadressen:

F3B0	LINLEN	- aantal kolommen in een regel
F3B1	CRTCNT	- aantal regels op het scherm
F3B2	CLMLST	- aantal posities tussen tabulatorstops
F3DC	CSRY	- regelnummercursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummercursor (linker kolom nr. 1)
FBB2	LINTTB	- verband logische regels met schermregels
FBCA	FSTPOS	- cursorpositie bij aanroep INLIN
FCAF	SCRMOD	- huidige schermmodus
FCB0	OLDSCR	- laatst gebruikte tekstmodus

3.

Categorie statement
Zie ook INPUT, OPEN, PRINT
Vorm LINE INPUT *#fnr,ident*

Het veld *fnr* is een file-nummeruitdrukking. *Ident* is de naam van een string-variabele of element van een rij van type string.
In deze vorm leest LINE INPUT uit een file tot en met de eerste RETURN (code &H0D). De ingelezen tekens, behalve de RETURN, vormen samen een string die aan *ident* wordt toegewezen.

De file waaruit gelezen wordt, heeft als nummer *fnr*. Deze file moet geopend zijn als sequentiële invoerfile; zie OPEN.

LINE gebruikt in deze betekenis de OS-routine:

014A	ISFLIO	- test of er uit file ingelezen wordt

LINE gebruikt in deze betekenis de geheugenplaatsen:

F860	FILTAB	- tabel met beginadressen van file-buffers
F864	PTRFIL	- beginadres van gebruikte file-buffers

LIST 93 (147)

1.

Categorie statement
Zie ook AUTO, DELETE, RENUM
Vorm LIST *regnrs*

LIST geeft een listing van een gedeelte van het BASIC-programma dat ten tijde van het statement in het geheugen staat. Welk gedeelte gelist wordt, is afhankelijk van de waarde van *regnrs*.
Het veld *regnrs* kan de volgende vormen aannemen:

LIST *nr1-nr2*
Alle regels van *nr1* tot *nr2* (inclusief grenzen) worden gelist.

LIST *-nr*
Alle regels vanaf de eerste regel in het programma tot en met *nr* worden gelist.

LIST *nr-*
Alle regels vanaf *nr* tot en met de laatste regel van het programma worden gelist.

LIST .
De regel die het laatst aan de beurt is geweest (het laatst ingetypt, gelist, veranderd etc) wordt gelist.

LIST – of LIST
Het hele programma wordt gelist.

De velden *nr, nr1* en *nr2* zijn regelnummerconstanten.
De listing wordt naar het scherm geschreven door middel van de OS-routine OUTDO (ingang 0018). Deze routine roept weer de routine CHPUT aan (ingang 00A2).

LIST gebruikt in deze betekenis de volgende OS-routines:

0018	OUTDO	- schrijft een teken
00A2	CHPUT	- schrijft een teken naar het scherm

LIST maakt gebruik van de volgende geheugenplaatsen:

F3B0	LINLEN	- aantal kolommen in een regel
F3B1	CRTCNT	- aantal regels op het scherm
F3DC	CSRY	- regelnummercursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummercursor (linker kolom nr. 1)

2.

Categorie	hulpwoord
Zie ook	KEY
Vorm	KEY LIST

In deze betekenis wordt LIST gebruikt als hulpwoord in een KEY-statement; zie KEY.

LLIST 9E (158)

Categorie statement
Zie ook LIST
Vorm LLIST *regnrs*

Het statement LLIST werkt precies zo als LIST, behalve dat de listing nu op de printer verschijnt. Dit gebeurt door de waarde van PRTFLG (F416) hoog te maken, zodat de OS-routine OUTDO (ingang 0018) naar de printer schrijft, met behulp van de OS-routine OUTDLP (ingang 014D), in plaats van naar het scherm.
Voor de mogelijke waarden en betekenissen van *regnrs* zie LIST.

LLIST maakt gebruik van de OS-routines:

0018	OUTDO	- schrijft een teken
014D	OUTDLP	- schrijft een teken naar de printer

LLIST gebruikt de volgende geheugenplaatsen:

F415	LPTPOS	- aantal geprinte tekens sinds vorige RETURN
F416	PRTFLG	- geeft aan dat uitvoer naar de printer gaat
F417	NTMSXP	- geeft aan of de printer een MSX-printer is
F418	RAWPRT	- geeft aan of uitvoer geïnterpreteerd moet

LOAD B5 (181)

Categorie statement
Zie ook MERGE, SAVE
Vorm LOAD *fnaam,R*

Het veld *fnaam* is een file-naamuitdrukking. Het veld R mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.
De file met naam *fnaam* moet een BASIC-programma bevatten. LOAD laadt het programma in het geheugen. Daarbij wordt een BASIC-programma dat eventueel al in het geheugen staat verwijderd. Als de file in ASCII-formaat is weggeschreven (zie SAVE), wordt dit formaat automatisch omgezet naar BASIC-code.

Als de *R* is ingevuld, wordt het programma na het laden direct uitgevoerd. In dat geval is het statement equivalent met RUN *fnaam*. Als er geen *R* is opgegeven, gebeurt dit niet. Het optreden van *R* wordt bewaard in de geheugenplaats RUNFLG (F866).

LOAD gebruikt de volgende geheugenplaatsen:

F862	NULBUF	- beginadres buffer van file met nummer 0
F864	PTRFIL	- beginadres gebruikte file-buffer
F866	RUNFLG	- geeft aan of direct uitgevoerd moet worden
F866	FILNAM	- naam van file die geladen wordt
F87C	NLONLY	- geeft aan dat er geladen wordt

LOC FF+AC (255+172)

Categorie functie
Zie ook OPEN
Vorm LOC(*fnr*)

Fnr is een file-nummeruitdrukking.
LOC levert van de file met nummer *fnr* het huidige recordnummer. Voor sequentiële files is dit nummer gelijk aan het aantal gelezen of geschreven bytes. Voor random access-files levert LOC het nummer op van het record dat op het moment van aanroep in de file-buffer staat.

LOCATE D8 (216)

Categorie statement
Zie ook CSRLIN, VPOS
Vorm LOCATE *hor,vert,curs*

De velden *hor, vert* en *curs* zijn byte-uitdrukkingen. Elk van de drie velden mogen worden weggelaten, echter niet alle drie tegelijkertijd. Als het statement door het weglaten van velden eindigt op komma's moeten deze komma's ook weggelaten worden.
LOCATE verplaatst de cursor naar de positie die aangegeven wordt door *hor* en *vert*. De waarden van deze velden geven respectievelijk het kolom- en het regelnummer van de gewenste positie, geteld vanaf de linker- en bovenrand. De randen zelf hebben nummer 0.
Het effect van LOCATE is precies gelijk aan het effect van OS-routine POSIT (ingang 00C6). De waarden van hor en vert, met 1 opgehoogd, worden opgeslagen in CSRX (F3DD) resp. CSRY (F3DC).

Curs geeft aan of tijdens niet-invoeracties de cursor op het scherm moet blijven staan of niet. Als de waarde van curs 0 is, dan blijft de cursor niet op het scherm staan, anders wel. De waarde van *curs* wordt opgeslagen in CSRSW (FCA9).
Als er velden uit het statement weggelaten zijn, wordt de huidige waarde van de geheugenplaats die bij dat veld hoort niet veranderd.

LOCATE maakt gebruik van de OS-routine:

00C6	POSIT	- verplaatst de cursor naar opgegeven positie

LOCATE gebruikt de volgende geheugenplaatsen:

F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)
FCA9	CSRSW	- geeft aan of cursor altijd aan moet zijn

LOF FF+AD (255+173)

Categorie functie
Zie ook OPEN
Vorm LOF(*fnr*)

Fnr is een file-nummeruitdrukking. De file die geopend is met nummer *fnr* moet zich op diskette bevinden.
LOF levert de huidige lengte in bytes op van de file met nummer *fnr*.

N.B. De functie LOF werkt alleen in DISK BASIC. De functie is geïmplementeerd via de RAM-haak H.LOF (FE9E).

LOG FF+8A (255+138)

Categorie functie
Zie ook EXP, SQR
Vorm LOG(*x*)

Het argument *x* is een numerieke uitdrukking met waarde>0.
LOG levert de e-logaritme van de waarde van *x*.

LPOS FF+9C (255+156)

Categorie functie
Zie ook POS
Vorm LPOS(*x*)

Het argument *x* is een numerieke uitdrukking. De waarde van *x* is niet van belang voor de werking van de functie.
LPOS levert het aantal tekens op dat naar de printer is gestuurd sinds de laatste keer dat een RETURN (code &H0D) naar de printer is gestuurd. Dit aantal wordt bijgehouden in de geheugenplaats LPTPOS (F415).

LPOS maakt gebruik van de geheugenplaats:

F415 LPTPOS - aantal geprinte tekens na de laatste RETURN

LPRINT 9D (157)

Categorie statement
Zie ook PRINT
Vorm LPRINT *tekst*

Tekst is een omschrijving van de tekst die geprint gaat worden. De vorm en mogelijke onderdelen van *tekst* zijn dezelfde als bij PRINT.
LPRINT werkt precies zoals PRINT, behalve dat de uitvoer naar de printer wordt gestuurd. Dit gebeurt door de waarde van PRTFLG (F416) hoog te maken, zodat de

OS-routine OUTDO (ingang 0018) naar de printer schrijft, met behulp van de OS-routine OUTDLP (ingang 014D), in plaats van naar het scherm.
De waarde van NTMSXP (F417) en RAWPRT (F418) bepalen de manier waarop de uitvoer behandeld wordt; zie par. 5.6 en OUTDLP (par. 6.3).

LPRINT maakt gebruik van de OS-routines:

0018	OUTDO	- schrijft een teken
014D	OUTDLP	- schrijft een teken naar de printer

LPRINT gebruikt de volgende geheugenplaatsen:

F415	LPTPOS	- aantal geprinte tekens sinds vorige RETURN
F416	PRTFLG	- geeft aan dat uitvoer naar de printer gaat
F417	NTMSXP	- geeft aan of de printer een MSX-printer is
F418	RAWPRT	- geeft aan of uitvoer geïnterpreteerd moet

LSET B8 (184)

Categorie statement
Zie ook FIELD, MID$, RSET
Vorm LSET *ident=uitdr*

Het veld *ident* is de naam van een string-variabele of element van een rijvariabele van type string. *Uitdr* is een string-uitdrukking.

Het resultaat van LSET is dat de waarde van *uitdr* terecht komt in de string-opslagruimte waar de vorige waarde van *ident* stond. Als de lengte van *uitdr* groter is dan de lengte van de vorige waarde van *ident,* wordt het einde van *uitdr* afgekapt. Als uitdr korter is, wordt de waarde aan het einde aangevuld met spaties.

LSET kan gebruikt worden om garbage collections te voorkomen door telkens dezelfde string-ruimte te gebruiken. Het voornaamste gebruik van LSET is echter een nieuwe waarde te geven aan de string-variabelen die wijzen in de file-buffer van een random access-file; zie FIELD.

N.B. LSET werkt alleen in DISK-BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.LSET (FE21).

MAX
CD (205)

Categorie hulpwoord
Zie ook FILES, MAXFILES
Vorm MAXFILES

Het sleutelwoord MAX wordt alleen gebruikt in de combinatie MAXFILES. Dit woord wordt gecodeerd als een combinatie van het woord MAX en het woord FILES: CD+B7.

MAXFILES

CD+B7 (205+183)

Categorie systeemvariabele
Zie ook CLEAR, FILES, MAX

MAXFILES wordt gebruikt om het aantal beschikbare file-buffers in te stellen. De nummers van de file-buffers lopen van 1 tot de toegewezen waarde.
Door het geven van een nieuwe waarde aan MAXFILES wordt de indeling van het RAM-geheugen veranderd, net als bij CLEAR. Tevens wordt het variabelen- en rijvariabelen-opslaggebied vernietigd.
De huidige waarde van MAXFILES kan niet uitgelezen worden. De waarde is echter opgeslagen in de geheugenplaats MAXFIL (adres F85F), en kan daaruit verkregen worden.

MAXFILES gebruikt of verandert de volgende geheugenplaatsen:

F672	MEMSIZ	- eindadres van string-opslaggebied
F674	STKTOP	- hoogste adres van de stapel
F69B	FRETOP	- eerste vrije plaats in string-opslaggebied
F6C4	ARYTAB	- beginadres van rijvariabelen-opslaggebied
F6C6	STREND	- adres van eerste vrije geheugenplaats
F85F	MAXFIL	- aantal beschikbare file-buffers
F860	FILTAB	- beginadres van de tabel met file-buffers
F862	NULBUF	- beginadres van buffer van file met nummer 0

MERGE B6 (182)

Categorie statement
Zie ook LOAD, SAVE
Vorm MERGE *fnaam*

Het veld *fnaam* is een file-uitdrukking. *Fnaam* mag geen wildcards bevatten.
De file die door *fnaam* wordt aangeduid moet een BASIC-programma bevatten, dat in ASCII-formaat is weggeschreven; zie SAVE. MERGE laadt deze file in het geheugen zonder eerst het aanwezige BASIC-programma te verwijderen; dit in tegenstelling tot LOAD. Op deze manier kunnen verschillende BASIC-programma's worden samengevoegd tot één programma.
Als in het programma in de file *fnaam* een regelnummer wordt gebruikt dat ook in het programma in het geheugen wordt gebruikt, dan wordt de regel in het geheugen overschreven door de regel in de file.

MERGE gebruikt de geheugenplaatsen:

F862	NULBUF	- adres van buffer van file met nummer 0
F864	PTRFIL	- adres van gebruikte file-buffer
F866	FILNAM	- file-naam (waarde van *fnaam*)
F87C	NLONLY	- geeft aan dat BASIC-programma geladen wordt

MID$ FF+83 (255+131)

1.

Categorie functie
Zie ook INSTR, LEFT$, LEN, RIGHT$
Vorm MID$(str,pos,len)

Het argument *str* is een string-uitdrukking. *Pos* en *len* zijn byte-uitdrukkingen. *Len* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.
MID$ levert een substring op van *str*. Deze substring wordt uit *str* afgeleid door, te beginnen bij het teken met nummer *pos*, de waarde van *str* te kopiëren tot een string bereikt is met lengte *len*, of tot het einde van *str* bereikt is.
Als het argument *len* weggelaten wordt, gaat het kopiëren altijd door tot het einde van *str*.

2.

Categorie hulpwoord
Zie ook LET, LSET, RSET
Vorm MID$*(ident,pos,len)=uitdr*

Het argument *ident* is de naam van een string-variabele of element van een rijvariabele van het type string. *Pos* en *len* zijn byte-uitdrukkingen. *Len* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt. *Uitdr* is een string-uitdrukking.

De waarde van *uitdr* komt terecht in de string-ruimte waar de waarde van *ident* staat, vanaf de positie die aangeduid wordt door *pos* en *len*. De betekenis van deze argumenten is dezelfde als in betekenis 1 van MID$.
Als *uitdr* meer tekens heeft is dan er ruimte is in het string-opslaggebied van *ident*, worden alleen de eerste tekens van *uitdr* gebruikt. Als *uitdr* minder tekens heeft, wordt het opslaggebied van *ident* alleen ter lengte van *uitdr* veranderd.

MKD$ FF+B0 (255+176)

Categorie functie
Zie ook CDBL, CVD, MKI$, MKS$
Vorm MKD$(*x*)

Het argument *x* is een numerieke expressie.

MKD$ zet de waarde van *x* om naar dubbele-precisie formaat (8 bytes floating point) en maakt van de resulterende 8 bytes een string van die lengte. Deze string kan worden gebruikt om velden van een record van een random-access-file een nieuwe waarde te geven; zie FIELD.

MKD$ heeft precies het omgekeerde effect van CVD. Zodoende geldt voor elke *x* de volgende formule:
CVD(MKD$(x))=*x*

N.B. MKD$ werkt alleen in DISK-BASIC. De functie wordt geïmplementeerd via de RAM-haak H.MKD$ (FE3A).

MKI$ FF+AE (255+174)

Categorie functie
Zie ook CINT, CVI, MKD$, MKS$
Vorm MKI$(*x*)

Het argument *x* is een numerieke expressie.
MKI$ zet de waarde van *x* om naar integer formaat (twee bytes in 2-complement) en maakt van de resulterende twee bytes een string van die lengte. Deze string kan worden gebruikt om velden van een record van een random-access-file een nieuwe waarde te geven; zie FIELD.
MKI$ heeft precies het omgekeerde effect van CVI. Zodoende geldt voor elke *x* de volgende formule: CVI(MKI$(*x*))=*x*

N.B. MKI$ werkt alleen in DISK-BASIC. De functie wordt geïmplementeerd via de RAM-haak H.MKI$ (FE30).

MKS$ FF+AF (255+175)

Categorie functie
Zie ook CSNG, CVS, MKD$, MKI$
Vorm MKS$(*x*)

Het argument *x* is een numerieke expressie.
MKS$ zet de waarde van *x* om naar dubbele-precisie formaat (4 bytes floating point) en maakt van de resulterende vier bytes een string van die lengte. Deze string kan gebruikt worden om velden van een record van een random-access-file een nieuwe waarde te geven; zie FIELD.
MKS$ heeft precies het omgekeerde effect van CVS. Zodoende geldt voor elke *x* de volgende formule: CVS(MKS$(*x*))=*x*

N.B. MKS$ werkt alleen in DISK-BASIC. De functie wordt geïmplementeerd via de RAM-haak H.MKS$ (FE35).

MOD FB (251)

Categorie operator
Vorm *x* MOD *y*

De operanden *x* en *y* zijn integeruitdrukkingen.
MOD berekent de rest na deling van de waarden van *x* en *y*. Zodoende geldt altijd de volgende formule: *x* MOD *y* = *x* – (*x**y*)**y*

MOD is een rekenkundige operator met prioriteit 8; zie par. 7.2.4.

N.B. MOD berekent niet de (wiskundige) modulus van de waarden van *x* en *y*. Deze modulus is namelijk altijd een positief getal, terwijl bijvoorbeeld –1 MOD 2 = –1.

MOTOR CE (206)

Categorie statement
Vorm MOTOR *modus*

Het veld *modus* kan de waarden ON en OFF aannemen, of kan weggelaten worden. MOTOR bestuurt de motor van de cassetterecorder via de remote control-ingang. MOTOR ON zet de motor aan; MOTOR OFF zet hem uit. MOTOR op zich zelf keert de status van de motor om (zet hem aan als hij uit is en vice versa).

MOTOR gebruikt de volgende OS-routine:

00F3	STMOTR	- zet de cassettemotor aan, uit of andersom

NAME D3 (211)

Categorie statement
Zie ook COPY
Vorm NAME *fnaam1* AS *fnaam2*

Fnaam1 en *fnaam2* zijn file-uitdrukkingen. Apparaatnamen in *fnaam1* en *fnaam2*, indien aanwezig, moeten diskettestation zijn.

NAME geeft een bestaande file een nieuwe file-naam. Na het uitvoeren van het statement heeft de file waarop *fnaam1* betrekking heeft, als naam de waarde van *fnaam2*.

De apparaten van *fnaam1* en *fnaam2* moeten hetzelfde diskettestation zijn. *Fnaam1* mag wildcards bevatten; *fnaam2* alleen op dezelfde plaatsen als *fnaam1*. Er mag vóór het uitvoeren van het statement nog geen file bestaan met file-naam *fnaam2*.

NAME gebruikt de volgende geheugenplaatsen:

F866	FILNAM	- naam van eerste file *(fnaam1)*
F871	FILNM2	- naam van tweede file *(fnaam2)*

N.B. NAME werkt alleen in DISK BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.NAME (FDF9).

NEW 94 (148)

Categorie statement
Zie ook CLEAR, MAXFILES
Vorm NEW

Dit statement heeft als gevolg dat het BASIC-programma in het geheugen wordt 'vergeten', ofte wel dat alle adreswijzers die aangeven waar het programma staat, teruggezet worden naar hun beginwaarde.
Tevens wordt de volg- (Engels: trace-) faciliteit indien nodig uitgeschakeld. Het effect van TRON wordt hiermee ongedaan gemaakt; zie TRON.
Anders dan gewoonlijk beweerd wordt, wist NEW het geheugen niet. Het programma is er nog steeds, maar wordt genegeerd.

NEW gebruikt de volgende geheugenplaatsen:

F6C0	OLDTXT	- adres van het eerstvolgende statement
F6C2	VARTAB	- beginadres variabelen-opslaggebied
F6C4	ARYTAB	- beginadres rijvariabelen-opslaggebied
F6C6	STREND	- adres van de eerste vrije geheugenplaats
F7C4	TRCFLG	- geeft aan of trace-faciliteit aan staat

NEXT 83 (131)

Categorie statement
Zie ook FOR
Vorm NEXT *ident*

Het veld *ident* is de naam van een numerieke variabele. *Ident* mag worden weggelaten.
NEXT is de afsluiting van een FOR-NEXT besturingsstructuur. Bij elk NEXT-statement wordt een FOR-statement gezocht waarvan de beschrijving op de stapel staat. De variabele *ident* is de lusvariabele van dat FOR-statement. Als *ident* weggelaten is, wordt als bijbehorend FOR-statement het eerste FOR-statement op de stapel genomen; dit is het laatste niet-afgesloten statement.

NEXT verhoogt de waarde van *ident* met de stapgrootte die gegeven is in het STEP-gedeelte van het bijbehorende FOR-statement, en voert vervolgens een test uit op het overschrijden van de grenswaarde die gegeven is in het TO-gedeelte van het bijbehorende FOR-statement.
Als de test mislukt, springt de computer naar het eerste statement achter het FOR-statement. Als de test lukt, wordt het beschrijvingsblok van het FOR-statement van de stapel gehaald, en gaat het programma verder met het statement na het NEXT-statement.

NOT E0 (224)

Categorie operator
Zie ook AND, EQV, IMP, OR, XOR
Vorm NOT *x*

De operand *x* is een integeruitdrukking.
De operand wordt eerst gecodeerd volgens 2-complement. NOT levert het 1-complement van de waarde van *x* op.
NOT is een logische operator met prioriteit 5; zie par. 7.2.4.

OCT$ FF+9A (255+154)

Categorie functie
Zie ook BIN$, HEX$, STR$
Vorm OCT$(*x*)

Het argument *x* is een integeruitdrukking.
OCT$ levert een string op die de representatie in het achttallig octaal stelsel van de waarde van *x* bevat.

OFF EB (235)

1.

Categorie hulpwoord
Zie ook KEY, MOTOR, ON

OFF wordt in deze betekenis als hulpwoord gebruikt om faciliteiten uit te schakelen. In het KEY-statement wordt het overzicht van de functietoetsen uitgeschakeld, in het MOTOR-statement de cassettemotor.

2.

Categorie hulpwoord
Zie ook INTERVAL, KEY, ON, SPRITE, STOP, STRIG

OFF wordt in deze betekenis als hulpwoord gebruikt om allerlei interrupt-faciliteiten uit te schakelen.

ON 95 (149)

1.

Categorie hulpwoord
Zie ook KEY, MOTOR, OFF

ON wordt in deze betekenis als hulpwoord gebruikt om faciliteiten uit te schakelen.

De dingen die met OFF kunnen worden uitgeschakeld, kunnen met ON weer ingeschakeld worden.

2.

Categorie hulpwoord
Zie ook INTERVAL, KEY, OFF, SPRITE, STOP, STRIG

ON wordt in deze betekenis als hulpwoord gebruikt om allerlei interrupt-faciliteiten in te schakelen.

3.

Categorie statement
Zie ook GOSUB
Vorm ON *x* GOSUB *regnr,...*

Het veld *x* is een byte-uitdrukking. *Regnr* is een regelnummerconstante. Het veld *regnr* mag een willekeurig aantal keren (minstens één) voorkomen, gescheiden door komma's.

ON GOSUB voert een subroutine-aanroep uit naar één van de regelnummers *regnr*, mits x dat toelaat. De waarde van *x* is een teller van de velden *regnr*.
Als *x* gelijk is aan 0 of groter is dan het aantal voorkomens van *regnr*, dan wordt meteen verder gegaan met het volgende statement. Anders wordt een subroutine-aanroep uitgevoerd naar het regelnummer dat hoort bij *x*. De uitvoering van de subroutine-aanroep verloopt precies als een gewoon GOSUB-statement. Na terugkeer uit de subroutine wordt dan verder gegaan met het volgende statement.

4.

Categorie statement
Zie ook GOTO
Vorm ON *x* GOTO *regnr,...*

Het veld *x* is een byte-uitdrukking. *Regnr* is een regelnummerconstante. Het veld *regnr* mag een willekeurig aantal keren (minstens één) voorkomen, gescheiden door komma's.

De uitvoering van het ON GOTO-statement verloopt precies als het ON GOSUB-statement, behalve dat er in plaats van een subroutine-aanroep een gewone sprong wordt uitgevoerd naar het regelnummer dat hoort bij *x*.
De velden *regnr* worden gecodeerd als regelnummerconstante. Dat houdt onder andere in, dat de velden mee worden hernummerd bij een RENUM-statement.

5.

Categorie statement
Zie ook: INTERVAL
Vorm ON INTERVAL=*duur* GOSUB *regnr*

Het veld *duur* is een integeruitdrukking. *Regnr* is een regelnummerconstante.
Het effect van dit statement is, dat de interrupt-faciliteit van de interval-klok van een sprongadres wordt voorzien. Als nu de faciliteit wordt ingeschakeld (met INTERVAL ON), dan wordt bij elk optreden van de interrupt een subroutine-aanroep verricht naar de regel met nummer *regnr*. Zie ook par. 6.2.2.
De waarde van *regnr* wordt opgeslagen in het element van TRPTBL (vanaf FCAC) dat hoort bij deze interrupt. De waarde van *duur* wordt opgeslagen in INTVAL (FCA0).

ON INTERVAL gebruikt de volgende geheugenplaatsen:

FC4C	TRPTBL	- administratietabel interrupt-faciliteiten
FCA0	INTVAL	- tijdsduur van interrupt

6.

Categorie statement
Zie ook KEY
Vorm ON KEY GOSUB *regnr1,...,regnr10*

De velden *regnr1* t/m *regnr10* zijn regelnummerconstanten. Elk van de velden *regnr1* t/m *regnr10* mag weggelaten worden. Als door het weglaten van velden het statement eindigt op komma's, mogen die komma's ook weggelaten worden.
Elk van de regelnummers hoort bij de functietoets met hetzelfde nummer. Het effect van dit statement is dat de interrupt-faciliteit van de functietoetsen waarvoor het regelnummer is ingevuld, van een sprongadres wordt voorzien. Als nu de faciliteit wordt ingeschakeld (met KEY (nr) ON), dan wordt bij elk optreden van een functietoets-interrupt een subroutine-aanroep verricht naar de bijbehorende regel. Zie ook par. 6.2.2.
De waarden van *regnr1* t/m *regnr10* worden opgeslagen in de elementen van TRPTBL (vanaf FCAC) die horen bij de functietoets-interrupts.

ON KEY gebruikt het volgende geheugengebied:

FC4C	TRPTBL	- administratietabel interrupt-faciliteiten

7.

Categorie statement
Zie ook SPRITE
Vorm ON SPRITE GOSUB *regnr*

Het veld *regnr* is een regelnummerconstante.
Het effect van dit statement is, dat de interrupt-faciliteit van de sprite-botsing van een sprongadres wordt voorzien. Als nu de faciliteit wordt ingeschakeld (met SPRITE ON), dan wordt bij elk optreden van de interrupt een subroutine-aanroep verricht naar de regel met nummer *regnr*. Zie ook par. 6.2.2.
De waarde van *regnr* wordt opgeslagen in het element van TRPTBL (vanaf FCAC) dat hoort bij deze interrupt.

ON SPRITE gebruikt de volgende geheugenplaatsen:

F3E7	STATFL	- geeft aan of sprite-botsing plaatsvond
FC4C	TRPTBL	- administratietabel interrupt-faciliteiten

8.

Categorie statement
Zie ook STOP
Vorm ON STOP GOSUB *regnr*

Het veld *regnr* is een regelnummerconstante.
Het effect van dit statement is, dat de interrupt-faciliteit van de CTRL+STOP-combinatie van een sprongadres wordt voorzien. Als nu de faciliteit wordt ingeschakeld (met STOP ON), dan wordt bij elk optreden van de interrupt een subroutine-aanroep verricht naar de regel met nummer *regnr*. Zie ook par. 6.2.2.
De waarde van *regnr* wordt opgeslagen in het element van TRPTBL (vanaf FCAC) dat hoort bij deze interrupt.

ON STOP gebruikt de volgende geheugenplaatsen:

FC4C	TRPTBL	- administratietabel interrupt-faciliteiten

9.

Categorie statement
Zie ook STRIG
Vorm ON STRIG GOSUB *regnr1,...,regnr5*

De velden *regnr1* t/m *regnr5* zijn regelnummerconstanten. Elk van de velden *regnr1* t/m *regnr5* mag weggelaten worden. Als door het weglaten van velden het statement eindigt op komma's, mogen die komma's ook weggelaten worden.
Elk van de regelnummers hoort bij de vuurknop met hetzelfde nummer. Het effect van dit statement is dat de interrupt-faciliteit van de vuurknoppen waarvoor een regelnummer is ingevuld, van een sprongadres wordt voorzien. Als nu de faciliteit wordt ingeschakeld (met STRIG (nr) ON), wordt bij elk optreden van een vuurknop-interrupt een subroutine-aanroep verricht naar de bijbehorende regel. Zie ook par. 6.2.2.

De waarden van *regnr1* t/m *regnr5* wordt opgeslagen in de elementen van TRPTBL (vanaf FCAC) die horen bij deze interrupt.

ON STRIG gebruikt de volgende geheugenplaatsen:

F3E8	TRGFLG	- geeft aan of vuurknoppen ingedrukt zijn
FC4C	TRPTBL	- administratietabel interrupt-faciliteiten

OPEN B0 (176)

Categorie statement
Zie ook CLOSE
Vorm 1. OPEN *fnaam* FOR *modus* AS *fnr*
2. OPEN *fnaam* AS *fnr* LEN=*len*

Fnaam is een file-uitdrukking. *Modus* kan als waarde INPUT, OUTPUT of APPEND hebben. *Fnr* is een file-nummeruitdrukking, eventueel voorafgegaan door een hekje (#). *Len* is een byte-uitdrukking. Het veld *len* mag worden weggelaten mits het voorafgaande deel LEN= ook weggelaten wordt.
OPEN heeft als resultaat dat de file die wordt aangeduid door *fnaam* wordt klaargemaakt om gebruikt te worden. Dit klaarmaken wordt gewoonlijk aangeduid met 'openen'.
De waarde van *fnr* is het nummer van de file-buffer die gebruikt wordt. Bij alle verdere acties op de file moet dit nummer gebruikt worden om de file aan te duiden. Er mag geen andere file geopend zijn met nummer *fnr*.

Het veld *modus* in vorm 1 bepaalt de manier waarop de file gebruikt kan worden. De mogelijke waarden van *modus* zijn:

INPUT
De file is een sequentiële invoer-file. Dit betekent dat de inhoud van de file alleen kan worden gelezen in de volgorde waarin hij in de file staat. Het lezen gebeurt met INPUT, LINE INPUT of INPUT$.
Een file die geopend wordt met modus INPUT moet al bestaan op het moment dat OPEN uitgevoerd wordt. Het openen houdt in dat de file-buffer met nummer *fnr* gereed wordt gemaakt, en het eerste blok uit de file in de buffer wordt gelezen.

OUTPUT
De file is een sequentiële uitvoer-file. Dat betekent dat er alleen in kan worden geschreven met PRINT. De gegevens die naar de file worden geschreven, komen in de file te staan in de volgorde waarin ze er naar toe zijn geschreven.
Het openen van een file met modus OUTPUT houdt in dat de file-buffer met nummer *fnr* wordt klaargemaakt, en dat een bestaande file met file-naam *fnaam* eerst verwijderd wordt.

APPEND
De file is een sequentiële uitvoer-file. De mogelijkheden zijn precies gelijk als bij OUTPUT; echter, de file moet al bestaan bij het openen, en nieuwe gegevens komen achter de al bestaande file. Deze modus werkt alleen voor files op diskette.
Het openen van een file met modus APPEND houdt in dat de file-buffer met nummer *fnr* wordt klaargemaakt, en dat het laatste (onvolledige) blok van de file in de buffer wordt gezet, zodat elke uitvoer naar de file daarachter komt te staan.

Als OPEN gebruikt wordt in vorm 2, wordt de file geopend als random access-file. Deze mogelijkheid bestaat alleen in DISK BASIC. De apparaatnaam in *fnaam*, indien aanwezig, moet de naam van een diskettestation zijn.

Een random access-file is opgedeeld in records, met een vaste lengte. De recordlengte wordt opgegeven door het veld *len*. Als dit veld weggelaten is, wordt de recordlengte vastgesteld op 256. De records van een random access-file kunnen in willekeurige volgorde worden gelezen en veranderd. Dit gebeurt met behulp van de instructies GET en PUT. De records van de file kunnen wordt bereikt met behulp van het statement FIELD.

Het openen van een random access-file bestaat uit het klaarmaken van de file-buffer, en het inlezen van het eerste record.

OPEN maakt gebruik van de volgende geheugenadressen:

F85F	MAXFIL	- hoogste toegestane file-nummer
F860	FILTAB	- tabel van file-buffers
F864	PTRFIL	- beginadres van gebruikte file-buffer
F866	FILNAM	- opslagruimte voor file-naam

OR F7 (247)

Categorie operator
Zie ook AND, EQV, IMP, NOT, XOR
Vorm *x* OR *y*

De operanden *x* en *y* zijn integeruitdrukkingen
De operanden worden gecodeerd volgens 2-complement. Het resultaat van OR is een getal, ook gecodeerd volgens 2-complement, waarvan elk bit direct afhankelijk is van het overeenkomstige bit in *x* en *y*.
OR neemt de logische bitsgewijze 'or'-functie van de operanden. Voor OR geldt het volgende diagram:

bit van *x*:	0 0 1 1

De velden *regnr* worden gecodeerd als regelnummerconstante. Dat wil onder meer zeggen dat ze mee worden hernummerd bij een RENUM-statement.

bit van *y*:	0 1 0 1
bit van OR:	0 1 1 1

OR is een logische operator met prioriteit 3; zie par. 7.2.4.

OUT 9C (156)

1.

Categorie statement
Zie ook INP, WAIT
Vorm OUT *adr,x*

De velden *adr* en *x* zijn byte-uitdrukkingen.
OUT schrijft de waarde van *x* naar de poort met poortadres *adr*.

2.

Categorie hulpwoord
Zie ook OUTPUT, PUT
Vorm OUTPUT

In deze betekenis wordt OUT gebruikt om het woord OUTPUT te coderen. OUTPUT wordt gecodeerd als OUT+PUT.

OUTPUT 9C+B3 (156+179)

Categorie hulpwoord
Zie ook APPEND, INPUT, OPEN
Vorm FOR OUTPUT

OUTPUT is een modus waaronder een file geopend kan worden; zie OPEN. Een file die met modus OUTPUT is geopend, is een sequentiële uitvoer-file. Er kan alleen naar geschreven worden met behulp van PRINT.

PAD FF+A5 (255+165)

Categorie functie
Zie ook PDL, STICK
Vorm PAD*(x)*

Het argument *x* is een byte-uitdrukking.
Het statement PAD leest een touch pad uit die aangesloten is op een van de spelingangen van de MSX-computer. De waarde van *x* geeft aan welke ingang bedoeld wordt, en wat er precies ingelezen moet worden. *X* kan de volgende waarden aannemen:

0 lees touch pad 1
1 bepaal x-coördinaat pad 1
2 bepaal y-coördinaat pad 1
3 bepaal schakelaarstand pad 1
4 lees touch pad 2
5 bepaal x-coördinaat pad 2
6 bepaal y-coördinaat pad 2
7 bepaal schakelaarstand pad 2

De werking van PAD is precies gelijk aan de werking van de OS-routine GTPAD (00DB).

PAD gebruikt de volgende OS-routine:

00DB GTPAD - leest toestand van touch pad

PAD gebruikt de volgende geheugenplaatsen:

FC9C	PADY	- y-coördinaat touch pad
FC9D	PADX	- x-coördinaat touch pad

PAINT BF (191)

Categorie statement
Zie ook CIRCLE, DRAW, LINE, PSET, PRESET
Vorm PAINT *begin,verf,grens*

Begin is een coördinaatuitdrukking; *verf* en *grens* zijn kleuruitdrukkingen. *Verf* en *grens* mogen weggelaten worden; als daardoor het statement eindigt op komma's, moeten die ook weggelaten worden.

PAINT verzorgt het inkleuren van een vlak in een grafische modus. *Begin* duidt het punt aan van waaruit het kleuren moet beginnen. Het inkleuren gebeurt in de kleur aangeduid door *verf*. Als *verf* niet ingevuld is wordt de standaard voorgrondkleur gebruikt. De verfkleur wordt opgeslagen in ATRBYT (F3F2).
Het in te kleuren gebied wordt begrensd door randen van de kleur *grens*. In modus 2 wordt het veld *grens* genegeerd, en kan dus weggelaten worden. Voor de grenskleur wordt dan de kleur van *verf* genomen. De grenskleur wordt opgeslagen in BRDATR (FCB2).

PAINT maakt gebruik van het grafische systeem en de routines daaruit. Verder maakt PAINT gebruik van de volgende OS-routine:

0129	PNTINI	- initialisatie voor PAINT-statement

PAINT maakt gebruik van de volgende geheugenplaatsen:

F3E9	FORCLR	- standaard voorgrondkleur
F3F2	ATRBYT	- werkkleur (kleur waarmee ingekleurd wordt)
F92A	CLOC	- adres van grafische cursor in patroontabel
F92V	CMASK	- bitmasker van grafische cursor
F924	CSAVEA	- tijdelijke opslagplaats voor CLOC
F926	CSAVEM	- tijdelijke opslagplaats voor CMASK
F949	LOHMSK	- tot en met
F955	RTPROG	- toepassing onbekend
FCAF	SCRMOD	- huidige schermmodus
FCB2	BRDATR	- grenskleur
FCB3	GXPOS	- x-coördinaat van grafische cursor
FCB5	GYPOS	- y-coördinaat van grafische cursor
FCB7	GRPACX	- plaats waar x-coördinaat samengesteld wordt
FCB9	GRPACY	- plaats waar y-coördinaat samengesteld wordt

N.B. PAINT werkt alleen in schermmodi 2 en 3.

PDL FF+A4 (255+164)

Categorie functie
Zie ook PAD, STICK
Vorm PDL*(x)*

Het argument *x* is een byte-uitdrukking met een waarde tussen 1 en 12 (inclusief grenzen).
PDL dient om een paddle controller uit te lezen die op een van de spelingangen is aangesloten. De waarde van *x* bepaalt welke paddle controller uitgelezen wordt. De functie levert een waarde op tussen 0 en 255 (inclusief grenzen), afhankelijk van de toestand waarin de paddle controller verkeert.
Het veld *x* kan de volgende waarden aannemen:

1	paddle controller A van ingang 1
2	paddle controller A van ingang 2
3	paddle controller B van ingang 1
4	paddle controller B van ingang 2
5	paddle controller C van ingang 1
6	paddle controller C van ingang 2
7	paddle controller D van ingang 1
8	paddle controller D van ingang 2
9	paddle controller E van ingang 1
10	paddle controller E van ingang 2
11	paddle controller F van ingang 1
12	paddle controller F van ingang 2

De werking van PDL is precies gelijk aan de werking van de OS-routine GTPDL (00DE).
PDL maakt gebruik van de OS-routine:

00DE	GTPDL	- leest toestand van paddle controller

PEEK FF+97 (255+151)

Categorie functie
Zie ook POKE, VPEEK, VPOKE
Vorm PEEK*(adr)*

Het argument *adr* is een integeruitdrukking. De waarde van adr mag liggen in het bereik &H8000-&HFFFF.
PEEK levert de waarde op van de geheugenplaats met adres *adr*. Voor het verband tussen adres en geadresseerd geheugen zie hoofdstuk 2.
PEEK wordt, samen met het bijbehorende POKE, gebruikt om het geheugen direct te kunnen aanspreken. Dit komt bijvoorbeeld van pas bij het gebruiken van ingewikkelde, zelf gedefinieerde gegevensstructuren, of in gevallen waarin parameters moeten worden overgedragen bij gebruik van machinetaalroutines (zie USR).

PLAY C1 (193)

1.

Categorie	statement
Zie ook	BEEP, SOUND
Vorm	PLAY *stem1, stem2, stem3*

De velden *stem1, stem2* en *stem3* zijn string-uitdrukkingen. Alle drie de velden mogen weggelaten worden, maar niet tegelijkertijd. Als door het weglaten van velden het statement op komma's eindigt, moeten deze komma's ook weggelaten worden. PLAY maakt het gebruik van het geluidssubsysteem binnen BASIC mogelijk; zie par. 6.2.4. De geluiden die geproduceerd worden, zijn afhankelijk van de waarde van de velden *stem1, stem2* en *stem3*.

Elk van de velden correspondeert met een stem van de PSG. De waarden van de velden zijn op te delen in substrings, die elk een geluidscommando voorstellen. De geluidscommando's vormen een soort mini-taal, die alleen geldig is binnen de velden van PLAY. De commando's mogen gescheiden zijn door spaties of puntkomma's. De bestaande geluidscommando's zijn:

A-G
Deze commando's brengen een toon voort die in frequentie gelijk is aan de noot met naam A t/m G. Het octaaf waaruit deze noot afkomstig is, kan worden ingesteld met het commando O. De toonduur kan worden ingesteld met het commando L.

De nootnamen kunnen gevolgd worden door een –, ten teken dat de noot een halve toonafstand verlaagd moet worden (mol), of door een + of #, ten teken dat de noot verhoogd moet worden (kruis).

Elke noot kan tenslotte worden gevolgd door een reeks punten (.). Deze punten geven een verlenging van de toonduur aan. Elke punt verlengt de toonduur met de helft van de vorige verlenging (dus een noot met één punt heeft 1½ keer de oorspronkelijke lengte, met twee punten 1¾ van de lengte enz).

Het verband tussen het octaafnummer, de nootnaam en het frequentiegetal (zoals dat ingesteld wordt in de PSG) is vastgelegd in afbeelding 7.7.

L *len*
Met dit commando wordt de toonduur ingesteld. Len geeft het omgekeerde van de duur. De waarde van len moet liggen tussen 1 en 64.

M *frqu*
Met dit commando wordt de snelheid van het volumeverloop geregeld. *Frqu* is een frequentiegetal, zoals dat ingevuld wordt in registers 11-12 van de PSG (zie par. 4.2.5). Frqu kan een waarde hebben tussen 1 en 65535.

N *nr*
Met dit commando kunnen noten met nummer worden genoemd in plaats van met naam. Het veld *nr* is het nummer van de noot. De waarde ligt tussen 0 en 95. De waarde 0 staat voor een pauze; met 1-95 worden echte noten aangeduid.
De noten die worden geproduceerd met het N-commando kunnen weer verlengd worden door er punten achter te zetten, net als bij A-G.

O *nr*
Met dit commando wordt het octaafnummer ingesteld. Nr is een getal tussen 1 en 8 (inclusief grenzen). Het nummer dat op deze manier wordt ingesteld, wordt gebruikt bij commando's A-G, om het octaaf te bepalen waaruit een noot gekozen moet worden.

R *len*
Dit commando resulteert in een pauze (rust) met een lengte die aangegeven wordt door len. De betekenis van dit veld is hetzelfde als bij het commando L.

S *nr*
Met dit commando wordt een nieuwe waarde voor het volume-verloop ingesteld. Het veld *nr* is het nummer van het nieuwe verloop. De waarde van *nr* ligt tussen 0 en 15.

T *tmp*
Met dit commando wordt de snelheid van spelen ingesteld. De waarde van *tmp* is een waarde tussen 32 en 255, die overeenkomt met het aantal kwartnoten (L4) per minuut.

V *vol*
Met dit commando wordt het volume ingesteld. De waarde van *vol* ligt tussen 0 en 15. Deze waarde komt overeen met het volume dat naar registers 8-10 van de PSG geschreven worden; zie par. 4.2.4.

X *id;*
Met dit commando wordt een nieuwe string ingevoegd in de huidige string, en wordt ook die string geïnterpreteerd als geluidscommando-string. *Id* is de naam van de string-variabele waaruit de string verkregen wordt. De puntkomma mag niet weggelaten worden.

In elk getalveld kan een variabele worden gebruikt, door in plaats van het getal de tekst '=id;' op te nemen, waarbij id de naam is van een numerieke variabele. De puntkomma is in dat geval verplicht.

PLAY maakt gebruik van de routines en geheugenplaatsen van het geluidssysteem; zie par. 6.2.4. Daarnaast gebruikt PLAY de volgende geheugenplaatsen:

F956	MCLTAB	- voor het ontleden van de commando-strings
F958	MCLFLG	- geeft aan of ontleden is voor PLAY of DRAW
FB35	PRSCNT	- aantal ontlede strings; eerste of tweede keer
FB36	SAVSP	- opslagplaats voor stapelwijzer
FB38	VOICEN	- nummer van stem waarvan veld ontleed wordt
FB39	SAVVOL	- opslagplaats voor volume tijdens rust
FB3B	MCLLEN	- voor het ontleden van de commando-strings
FB3C	MCLPTR	- voor het ontleden van de commando-strings

2.

Categorie functie
Vorm PLAY*(stem)*

Stem is een byte-uitdrukking die een waarde oplevert tussen 0 en 3 (inclusief grenzen).
PLAY onderzoekt of de stem van de PSG, aangeduid door de waarde van *stem*, geluid voortbrengt op het moment dat de functie aangeroepen wordt. De functie levert –1 (waar) op als de stem geluid voortbrengt, anders 0 (onwaar).
Het verband tussen de waarde van *stem* en de onderzochte stem van de PSG is als volgt:

0	alle stemmen van de PSG
1	stem A
2	stem B
3	stem C

Als *stem=0* levert PLAY alleen 0 op als geen van de stemmen van de PSG geluid voortbrengt.

PLAY maakt in deze betekenis gebruik van de geheugenplaats:

FB40	MUSICF	- geeft aan welke van de stemmen nog speelt

POINT ED (237)

Categorie functie
Zie ook PSET, PRESET
Vorm POINT *punt*

Punt is een coördinaatuitdrukking.
Bij aanroep in een grafische schermmodus levert de functie POINT de kleurcode op van het punt dat wordt aangeduid door punt. Als dat punt buiten het scherm valt door te grote of te kleine coördinaten, levert POINT –1 op.

Bij aanroep in een tekstschermmodus levert de functie een ongedefinieerde waarde op. Er wordt echter geen foutmelding gegeven.

POINT maakt gebruik van de volgende OS-routines:

010E	SCALXY	- test coördinaten en zet ze eventueel om
0111	MAPXY	- zet coördinaten om in adres en bitmasker
011D	READC	- bepaalt kleur van punt onder cursor

POINT maakt gebruik van de volgende geheugenadressen:

F92A	CLOC	- adres van grafische cursor in patroontabel
F92C	CMASK	- bitmasker grafische cursor
FCB3	GXPOS	- x-coördinaat van grafische cursor
FCB5	GYPOS	- y-coördinaat van grafische cursor
FCB7	GRPACX	- plaats waar x-coördinaat samengesteld wordt
FCB9	GRPACY	- plaats waar y-coördinaat samengesteld wordt

POKE 98 (152)

Categorie statement
Zie ook PEEK, VPEEK, VPOKE
Vorm POKE *adr,x*

Het veld *adr* is een integeruitdrukking. Het veld *x* is een byte-uitdrukking.
POKE zet de waarde van *x* in de geheugenplaats met adres *adr*. Voor het verband tussen adres en geheugenplaats zie hoofdstuk 2.

POS FF+91 (255+145)

Categorie functie
Zie ook CSRLIN, LOCATE
Vorm POS(*x*)

Het veld *x* is een numerieke uitdrukking. De waarde van x heeft geen invloed op de werking van de functie.
In een tekstmodus levert POS het nummer van de kolom op waar de cursor zich bevindt. De meest linkse kolom heeft nummer 0. In grafische modus levert POS de kolom op waar de cursor zich bevond toen het scherm voor het laatst in tekstmodus was.
De waarde die POS oplevert, is 1 kleiner dan de waarde van geheugenplaats CSRX (F3DD).

POS gebruikt de volgende geheugenplaats:

F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)

PRESET C3 (195)

Categorie statement
Zie ook CIRCLE, DRAW, LINE, PAINT, POINT, PSET
Vorm PRESET *punt,kleur*

Het veld *punt* is een coördinaatuitdrukking. *Kleur* is een kleuruitdrukking. *Kleur* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.

PRESET verandert de kleur van het punt dat wordt aangeduid door *punt*. De nieuwe kleur voor dat punt is gegeven in *kleur*. Als *kleur* weggelaten is, wordt voor de nieuwe kleur de achtergrondkleur gekozen (de waarde van BAKCLR (F3EA)). De kleur die het punt krijgt wordt bewaard in ATRBYT (F3F2).

PRESET gebruikt de volgende OS-routines:

010E	SCALXY	- test coördinaten en zet ze om
0111	MAPXY	- zet coördinaten om in adres en bitmasker
011A	SETATR	- geeft ATRBYT een nieuwe waarde
0120	SETC	- geeft punt onder cursor een nieuwe kleur

PRESET gebruikt de volgende geheugenplaatsen:

F3EA	BAKCLR	- standaard achtergrondkleur
F3F2	ATRBYT	- werkkleur (nieuwe kleur voor punt)
F92A	CLOC	- adres van grafische cursor in patroontabel
F92C	CMASK	- bitmasker grafische cursor
FCAF	SCRMOD	- huidige schermmodus
FCB3	GXPOS	- x-coördinaat van grafische cursor
FCB5	GYPOS	- y-coördinaat van grafische cursor
FCB7	GRPACX	- plaats waar x-coördinaat samengesteld wordt
FCB9	GRPACY	- plaats waar y-coördinaat samengesteld wordt

N.B. PRESET werkt alleen in schermmodi 2 en 3.

PRINT 91 (145)

1.

Categorie statement
Zie ook INPUT, LPRINT, SPC(, TAB(, USING
Vorm PRINT *tekst*

Dit statement zet tekst op het scherm. Het veld *tekst* bepaalt wat er precies op het scherm komt te staan. *Tekst* kan de volgende elementen bevatten: uitdrukkingen, SPC(-elementen, TAB(-elementen en maximaal één USING-element. Het effect van SPC(, TAB(en USING is vastgelegd bij die sleutelwoorden zelf. Uitdrukkingen die voorkomen in *tekst* worden berekend, en de waarde wordt op het scherm gezet.

Getalwaarden worden decimaal gerepresenteerd, tenzij ze teveel cijfers in beslag gaan nemen; in dat geval worden ze gerepresenteerd in exponent-notatie. De eerste positie wordt in beslag genomen door het teken (positief of negatief) van het getal; dit is een spatie bij een positief getal, een minteken bij een negatief getal. Het getal wordt gevolgd door een spatie.
Uitdrukkingen kunnen worden gescheiden door komma's, puntkomma's of niets. Bij een komma worden zoveel spaties geschreven tot de cursor op de eerstvolgende tabulatorstop staat; bij een puntkomma of niets wordt de cursor niet verplaatst.
Na elk PRINT-statement wordt een RETURN gegeven, behalve als tekst eindigde op een komma, puntkomma, SPC(of TAB(.

PRINT gebruikt de volgende OS-routines:

0018	OUTDO	- schrijft een teken
00A2	CHPUT	- schrijft een teken naar het scherm

PRINT gebruikt de volgende geheugenplaatsen:

F3B0	LINLEN	- aantal kolommen van een schermregel
F3B1	CRTCNT	- aantal regels van een scherm
F3B2	CLMLST	- aantal kolommen van tussen tabulatorstops
F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker regel nr. 1)
FCA6	GRPHED	- geeft aan of grafisch kopbyte geweest is
FCA7	ESCCNT	- geeft aan of escape-sequence aan de gang is

2.

Categorie statement
Zie ook INPUT, OPEN
Vorm PRINT *#fnr,tekst*

Het veld *fnr* is een file-nummeruitdrukking. *Tekst* is een omschrijving van de tekst die geschreven wordt. De vorm van *tekst* is dezelfde als bij betekenis 1 van PRINT.

Dit statement schrijft naar een file. De file is geopend met nummer *fnr*, als sequentiële uitvoer-file (modus OUTPUT of APPEND). Het veld *tekst* geeft aan wat er geschreven wordt. De betekenis van de elementen van *tekst* is hetzelfde als bij betekenis 1 van PRINT, waar naar het scherm geschreven wordt in plaats van naar een file. Het verschil in uitvoering met de eerste betekenis van PRINT is, dat eerst aan PTRFIL (F864) het adres van de file-buffer met nummer *fnr* wordt gegeven. Zodoende resulteert het aanroepen van de OS-routine OUTDO (ingang 0018) in het schrijven naar file in plaats van naar het scherm.

PRINT maakt in deze betekenis gebruik van de routine:

0018	OUTDO	- schrijft een teken (nl. naar file)

PRINT gebruikt in deze betekenis de geheugenplaatsen:

F860	FILTAB	- tabel met beginadressen file-buffers
F864	PTRFIL	- beginadres van gebruikte file-buffer

PSET C2 (194)

Categorie statement
Zie ook CIRCLE, DRAW, LINE, PAINT, POINT, PRESET
Vorm PSET *punt, kleur*

Het veld *punt* is een coördinaat-uitdrukking. *Kleur* is een kleuruitdrukking. *Kleur* mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt. PSET verandert de kleur van het punt dat wordt aangeduid door *punt.* De nieuwe kleur wordt aangeduid door *kleur.* Als *kleur* weggelaten is, wordt de nieuwe kleur van het punt gegeven door de voorgrondkleur, bewaard in FORCLR (F3E9).

PSET gebruikt de volgende OS-routines:

010E	SCALXY	- test coördinaten en zet ze om
0111	MAPXY	- zet coördinaten om in adres en bitmasker
011A	SETATR	- geeft ATRBYT een nieuwe waarde
0120	SETC	- geeft punt onder cursor een nieuwe kleur

PSET gebruikt de volgende geheugenplaatsen:

F3E9	FORCLR	- standaard voorgrondkleur
F3F2	ATRBYT	- werkkleur (nieuwe kleur voor punt)
F92A	CLOC	- adres van grafische cursor in patroontabel
F92C	CMASK	- bitmasker grafische cursor
FCAF	SCRMOD	- huidige schermmodus
FCB3	GXPOS	- x-coördinaat van grafische cursor
FCB5	GYPOS	- y-coördinaat van grafische cursor
FCB7	GRPACX	- plaats waar x-coördinaat samengesteld wordt
FCB9	GRPACY	- plaats waar y-coördinaat samengesteld wordt

N.B. PSET werkt alleen in schermmodi 2 en 3.

PUT B3 (179)

1.

Categorie statement
Zie ook FIELD, GET, OPEN
Vorm PUT *fnr,recnr*

Het veld *fnr* is een file-nummeruitdrukking, eventueel voorafgegaan door een hekje (#). *Recnr* is een integeruitdrukking. *Recnr* mag worden weggelaten, mits de voorafgaande komma ook wordt weggelaten.

De file met nummer *fnr* moet geopend zijn als random access-file, dus zonder modus (zie OPEN). PUT schrijft het record dat staat in de file-buffer met nummer *fnr* naar de bijbehorende file op diskette.

In *recnr* staat het nummer van het record in de file dat moet worden overschreven door de inhoud van de file-buffer. Als *recnr* weggelaten wordt, wordt hiervoor het eerstvolgende record in de file genomen. De file gedraagt zich in dat geval als een soort sequentiële uitvoer-file.
Uit het recordnummer wordt met behulp van de recordlengte de plaats van het betreffende record in de file berekend. De lengte van een record is bij het openen van de file opgegeven; zie OPEN.

PUT maakt gebruik van de volgende geheugenplaatsen:

F860	FILTAB	- beginadres van file-tabel
F864	PTRFIL	- adres van gebruikte file-buffer

N.B. PUT werkt alleen in DISK-BASIC.

2.

Categorie	statement
Zie ook	SPRITE$
Vorm	PUT SPRITE *vlaknr,plaats,kleur,sprnr*

Vlaknr is een byte-uitdrukking met een waarde tussen 0 en 31 (inclusief grenzen). *Plaats* is een coördinaatuitdrukking. Kleur is een kleuruitdrukking. *Sprnr* is een niet-negatieve integeruitdrukking. *Kleur* en *sprnr* en worden mogen weggelaten. Als daardoor het statement op komma's eindigt, moeten die ook weggelaten worden. *Plaats* mag weggelaten worden, mits *kleur* of *sprnr* wél opgegeven zijn.

PUT SPRITE verandert een element van de sprite-vlaktabel (zie par. 3.2.1). Het resultaat is dat het patroon van een of andere sprite op het scherm verschijnt, verdwijnt of van plaats of vorm verandert.

Vlaknr is het sprite-vlaknummer, ofte wel het elementnummer in de sprite-vlaktabel.

Plaats duidt het punt aan waar de linkerbovenhoek van de sprite komt te staan. De waarde van de x- en y-coördinaten in *plaats* worden in de sprite-vlaktabel gezet. Als plaats weggelaten wordt, wordt de plaats van de sprite niet veranderd.
De x-coördinaat in *plaats* kan een effectieve waarde hebben tussen –32 en 255. Als de waarde groter of kleiner is, wordt alleen het minst significante byte van de waarde gebruikt.
Van de waarde van de y-coördinaat in *plaats* wordt alleen het minst significante byte gebruikt. De y-coördinaat kan de volgende waarden hebben:

0 t/m 191	- lijn met dat nummer
192 t/m 207	- sprite niet op scherm
208	- deze en volgende sprites niet op scherm
209 t/m 233	- sprite niet op scherm
234 t/m 255	- lijn met nummer 256 lager

Kleur is de kleur waarin de sprite op het scherm verschijnt. De waarde van *kleur* komt ook in de sprite-vlaktabel te staan. Als *kleur* niet wordt opgegeven, wordt de waarde van de kleur in de sprite-vlaktabel niet veranderd.
Sprnr is het nummer van de sprite, ofte wel het elementnummer in de sprite-patroontabel van het sprite-patroon. Het sprite-nummer wordt in de sprite-vlaktabel ingevuld. Als *sprnr* weggelaten wordt, wordt de waarde van het sprite-nummer in de sprite-vlaktabel niet veranderd.

PUT SPRITE maakt gebruik van de volgende OS-routines:

004D	WRTVRM	- schrijft naar videogeheugen
0087	CALATR	- berekent beginadres van spritevlakelement

N.B. PUT SPRITE werkt alleen in schermmodi 1, 2 en 3.

3.

Categorie	hulpwoord
Zie ook	OUT, OUTPUT
Vorm	OUTPUT

In deze betekenis wordt PUT gebruikt om het woord OUTPUT te coderen. OUTPUT wordt gecodeerd als OUT+PUT.

READ 87 (135)

Categorie	statement
Zie ook	DATA, RESTORE
Vorm	READ *ident,...*

Ident is de naam van een variabele of element van een rij. Het veld *ident* kan willekeurig vaak (1 of meer keer) voorkomen.
Het effect van READ is dat de waarde van het eerstvolgende veld van DATA toegewezen wordt aan *ident,* mits van het juiste type. Met het eerstvolgende veld wordt de eerste constante in een DATA-statement bedoeld met een hoger adres dan de waarde van DATPTR (F6C8).

Nadat de waarde van die constante aan *ident* is toegewezen, wordt het adres ervan aan DATPTR toegewezen. Vervolgens wordt het hele proces herhaald voor het volgende veld *ident* in het READ-statement.

READ maakt gebruik van het geheugenadres:

F6A3	DATLIN	- regelnummer van laatst gevonden DATA-veld
F6A6	FLGINP	- geeft aan of er INPUT of READ plaatsvindt
F6C2	VARPTR	- beginadres van variabelen-opslaggebied
F6C8	DATPTR	- beginadres voor speurtocht naar DATA-veld

REM 8F (143)

Categorie statement
Vorm REM *tekst*

tekst bestaat uit alle tekens tussen het woord REM en het eind van de regel.
Het REM-statement doet niets. In *tekst* kan commentaar staan omtrent de werking of opzet van het programma. Dit commentaar wordt door de computer niet behandeld. Het woord REM vertelt in feite dat wat er na dat woord volgt, tot het einde van de regel, niet bekeken hoeft te worden.
Het woord REM kan vervangen worden door een apostrof (').

RENUM AA (170)

Categorie statement
Zie ook AUTO, DELETE, LIST
Vorm RENUM *nieuw,oud,stap*

De velden *nieuw, oud* en *stap* zijn regelnummerconstanten. Alle drie de velden mogen worden weggelaten. Als door het weglaten van velden het statement op komma's eindigt, moeten die komma's ook worden weggelaten. Voor elk van de drie velden mag een punt worden ingevuld; deze staat voor het nummer van de laatst behandelde regel.

RENUM geeft de regels in een BASIC-programma nieuwe nummers. Tevens worden alle verwijzingen naar die regels door regelnummerconstanten – d.w.z. constanten die gecodeerd zijn als regelnummer, zie par. 7.2.5 – aangepast aan de nieuwe nummering.
Het hernummeren begint bij de regel met nummer *oud*. Als *oud* niet ingevuld is, begint het hernummeren bij de eerste regel van het programma.
De eerste hernummerde regel krijgt nummer *nieuw*. Als *nieuw* weggelaten is, krijgt de eerste regel nummer 10.
Het verschil in regelnummers van twee opvolgende regels is na het hernummeren gelijk aan *stap*. Als *stap* niet is opgegeven, is het verschil 10.

Als een regelnummer in een GOTO-, GOSUB- of dergelijk statement geen bestaand regelnummer is, wordt bij het hernummeren een foutmelding van die strekking gegeven.

RENUM maakt gebruik van de volgende geheugenplaatsen:

F676	TXTTAB	- beginadres van het BASIC-programma
F6A9	PTRFLG	- geeft aan of nrs als wijzer gecodeerd zijn
F6B5	DOT	- regelnr dat voor de punt ingevuld wordt
F6C2	VARTAB	- beginadres variabelen-opslaggebied

RESTORE 8C (140)

Categorie	statement
Zie ook	DATA, READ
Vorm	RESTORE *regnr*

Het veld *regnr* is een regelnummerconstante. *Regnr* mag weggelaten worden.
RESTORE zet de datawijzer (zie READ), bewaard in DATPTR (F6C8) op het eerste statement van de regel met nummer *regnr*. Als *regnr* weggelaten is, wordt de datawijzer aan het begin van het programma gezet.
RESTORE dient om DATA-statements meerdere malen te kunnen gebruiken. Na een RESTORE-statement wordt bij een READ-actie naar DATA-velden gezocht vanaf regel *regnr* (of het begin van het programma).
RESTORE maakt gebruik van de volgende geheugenplaatsen:

F676	TXTTAB	- beginadres van BASIC-programma
F6C2	VARTAB	- beginadres van variabelen-opslaggebied
F6C8	DATPTR	- beginadres voor speurtocht naar DATA-veld

RESUME A7 (167)

Categorie	statement
Zie ook	ON, ERROR, NEXT
Vorm	RESUME *regel*

Het veld *regel* kan een regelnummerconstante zijn of het sleutelwoord NEXT. *Regel* kan weggelaten worden.
Een foutverwerkingsroutine, binnengekomen door ON ERROR GOTO, moet altijd eindigen met RESUME. Zolang er geen NEXT, RESUME of RETURN is uitgevoerd, blijft de opgetreden fout bewaard.
Als *regel* het woord NEXT is, gaat het programma verder met het eerste statement na het statement waarin de fout is opgetreden.
Als *regel* een nummer is, groter dan 0, dan gaat het programma verder met het eerste statement op de programmaregel met nummer *regel*.
Als *regel* weggelaten is, of *regel* is het getal 0, dan gaat het programma verder met (een herhaling van) het statement waarin de fout opgetreden is.
Als RESUME in het programma voorkomt zonder dat er een ON ERROR uitgevoerd is (dus in een stuk programma dat niet bij de foutafhandelingsroutine hoort), levert dit een foutmelding op.

RESUME gebruikt de volgende geheugenplaatsen:

F6AF	SAVTXT	- tijdelijk opslaggebied voor statement-adres
F6B1	SAVSTK	- tijdelijk opslaggebied voor stapelwijzer
F6B3	ERRLIN	- nummer van regel waar fout is opgetreden
F6B7	ERRTXT	- adres van statement waar fout is opgetreden
F6BB	ONEFLG	- geeft aan of foutroutine in uitvoering is

RETURN 8E (142)

Categorie statement
Zie ook GOSUB, RESUME
Vorm RETURN *regnr*

Het veld *regnr* is een regelnummerconstante. *Regnr* mag weggelaten worden.
Het statement RETURN is de afsluiting van een subroutine. Elke subroutine (d.w.z. een stuk programma dat bereikt is door een GOSUB-statement) moet afgesloten worden door een RETURN-statement.
Bij het uitvoeren van RETURN gaat het programma verder met het eerste statement op regel *regnr*. Als *regnr* niet ingevuld is, gaat het programma verder met het eerste statement na het GOSUB-statement waarmee de subroutine aangeroepen werd. Het regelnummer en adres van dit statement zijn op de stapel bewaard.

RIGHT$ FF+82 (255+130)

Categorie functie
Zie ook INSTR, LEFT$, LEN, MID$
Vorm RIGHT$(*str,len*)

Het argument *str* is een string-uitdrukking. *Len* is een byte-uitdrukking.

De functie levert een substring op van de waarde van *str*. Deze substring bestaat uit de meest rechtse tekens van *str*, ter lengte van *len*. Als de waarde van *len* groter is dan de lengte van *str*, levert RIGHT$ de hele waarde van *str* op.

RND FF+88 (255+136)

Categorie functie
Vorm RND(*x*)

Het argument *x* is een numerieke uitdrukking.

De functie RND verzorgt de random-getalgenerator van de MSX-computer. Deze generator levert bij elke aanroep het volgende getal uit een getallenreeks op, die is vastgelegd door het eerste getal van de reeks. Elk getal uit deze reeks ligt tussen 0 en 1 (inclusief 0, exclusief 1).
De getallenreeks wordt vastgelegd door het eerste getal uit de reeks. Dit eerste getal

wordt bij het gebruiken van RUN, CLEAR of MAXFILES opnieuw ingesteld op een vaste beginwaarde. Zodoende levert RND binnen een BASIC-programma, wanneer er geen tegenmaatregelen genomen worden, bij elke aanroep van dat programma dezelfde reeks random-getallen op.
Door het argument *x* kan de reeks worden beïnvloed. Als *x* positief is, wordt gewoonweg het volgende getal uit de reeks opgeleverd, ongeacht de waarde van *x*. Als *x* nul is, levert RND het laatst berekende getal van de reeks opnieuw op. Als *x* negatief is, wordt er een nieuwe getallenreeks begonnen, waarvan het eerste getal afhankelijk is van de waarde van *x*.
Door aan het begin van een programma RND aan te roepen met een negatief getal waarin de waarde van TIME verwerkt is, levert RND bij elke aanroep van het programma een andere getalreeks op.

RND gebruikt de volgende geheugenplaats:

F857	RNDX	- laatst opgeleverde getal (dubbele precisie)

RSET B9 (185)

Categorie statement
Zie ook LET, LSET, MID$
Vorm RSET *ident=uitdr*

Het veld *ident* is de naam van een string-variabele of element van een rij van type string. *Uitdr* is een string-uitdrukking.
Het resultaat van RSET is, dat de waarde van *uitdr* terecht komt in de string-opslagruimte waar de vorige waarde van *ident* stond. Als de lengte van *uitdr* groter is dan de lengte van de vorige waarde van *ident,* wordt het einde van uitdr afgekapt. Als *uitdr* korter is, wordt de waarde aan het begin aangevuld met spaties.
RSET kan gebruikt worden om garbage collections te voorkomen door telkens dezelfde string-ruimte te gebruiken. Het voornaamste gebruik van RSET is echter, een nieuwe waarde te geven aan de string-variabelen die wijzen in de file-buffer van een random access-file; zie FIELD.

N.B. RSET werkt alleen in DISK-BASIC. Het statement wordt geïmplementeerd via de RAM-haak H.RSET (FE26).

RUN 8A (138)

Categorie statement
Zie ook CLEAR, MAXFILES
Vorm 1. RUN *regnr*
2. RUN *fnaam*

Vorm 1:

Regnr is een regelnummerconstante. *Regnr* kan worden weggelaten.

RUN heeft tot gevolg dat eerst alle variabelen worden vernietigd en alle functie-definities ongedaan worden gemaakt; vervolgens wordt het BASIC-programma in het geheugen uitgevoerd, te beginnen met de regel met nummer *regnr.* Als *regnr* wordt weggelaten, begint de programma-uitvoering met de eerste regel van het programma.

RUN gebruikt in deze betekenis de volgende geheugenplaatsen:

F69B	FRETOP	- eerste vrije plaats in string-opslaggebied
F672	MEMSIZ	- eindadres van string-opslaggebied
F676	TXTTAB	- beginadres van BASIC-programma
F6C4	ARYTAB	- beginadres van rijvariabelen-opslaggebied
F6C6	STREND	- adres van eerste vrije geheugenplaats
F6C8	DATPTR	- beginadres voor speurtocht naar DATA-veld

Vorm 2:

Fnaam is een file-uitdrukking. De file die wordt aangeduid door *fnaam* moet een BASIC-programma bevatten.
De file met naam *fnaam* wordt geladen, en een gewone RUN wordt uitgevoerd. Het effect is precies gelijk aan het effect van LOAD *fnaam,R.*

Als het apparaat in *fnaam* de cassetterecorder is, moet de file in ASCII-formaat weggeschreven zijn, d.w.z. met SAVE *fnaam,A*; zie SAVE.

RUN gebruikt in deze betekenis de volgende geheugenplaatsen:

F69B	FRETOP	- eerste vrije plaats in string-opslaggebied
F672	MEMSIZ	- eindadres van string-opslaggebied
F676	TXTTAB	- beginadres van BASIC-programma
F6C4	ARYTAB	- beginadres van rijvariabelen-opslaggebied
F6C6	STREND	- adres van eerste vrije geheugenplaats
F6C8	DATPTR	- beginadres voor speurtocht naar DATA-veld
F862	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- beginadres van gebruikte file-buffer
F866	RUNFLG	- geeft aan dat file direct wordt uitgevoerd
F866	FILNAM	- file-naam (waarde van *fnaam*)
F87C	NLONLY	- geeft aan dat BASIC-programma in file staat

SAVE BA (186)

Categorie statement
Zie ook LOAD, KILL, RUN
Vorm SAVE *fnaam,A*

Fnaam is een file-uitdrukking. Het veld A mag weggelaten worden, mits de voorafgaande komma ook weggelaten wordt.

SAVE schrijft het BASIC-programma in het geheugen weg onder file-naam *fnaam*. Als het veld *A* opgegeven is, wordt de file weggeschreven in ASCII-formaat. Als het apparaat in *fnaam* een cassetterecorder is, wordt de file in ASCII-formaat weggeschreven, of het veld *A* nu wel of niet is opgegeven.

SAVE maakt gebruik van de volgende geheugenplaatsen:

F862	NULBUF	- beginadres van file-buffer met nummer 0
F864	PTRFIL	- beginadres van gebruikte file-buffer
F866	FILNAM	- file-naam (waarde van *fnaam)*

SCREEN C5 (197)

Categorie statement
Zie ook CLS, COLOR
Vorm SCREEN *modus,spr,klik,baud,print*

Modus en *spr* zijn byte-uitdrukkingen met waarde 0-3. *Klik* en *print* zijn byte-uitdrukkingen met waarde 0 of 1. *Baud* is een byte-uitdrukking met waarde 1 of 2. Alle velden van het statement SCREEN kunnen worden weggelaten. Als door het weglaten van velden het statement eindigt op komma's moeten die ook weggelaten worden.

SCREEN verandert de schermmodus, en wordt bovendien gebruikt om allerlei 'losse' zaken in te stellen. Voor alle instellingen geldt: als het desbetreffende veld is weggelaten, wordt de instelling niet veranderd t.o.v. de huidige waarde.
Met *modus* wordt de schermmodus ingesteld. De waarde van modus is gelijk aan de in te stellen schermmodus. Bij het instellen van de schermmodus worden de VDP-registers aangepast en de naam-, patroon-, kleuren-, sprite-patroon- en sprite-vlaktabel geïnitialiseerd. De waarde van *modus* wordt bewaard in SCRMOD (FCAF), en in OLDSCR (FCB0) als het een schermmodus is.

Met *spr* wordt de sprite-grootte en -vergroting ingesteld. Met andere woorden, met het veld *spr* worden bits 0 en 1 van register 1 van de VDP ingesteld. De relatie tussen de waarde van *spr* en de grootte en vergroting van de sprites is als volgt:

spr	*grootte*	*vergroot*
0	8×8	nee
1	8×8	ja
2	16×16	nee
3	16×16	ja

Met het veld *klik* kan het geluid (de 'klik') dat klinkt als er een toets aangeslagen wordt aan of uit worden gezet. Als *klik*=0 wordt het geluid uitgeschakeld, als *klik*=1 wordt het ingeschakeld. Samen met de toetsenbord-klik wordt ook het piepje bij een foutmelding in- of uitgeschakeld.

De instelling van *klik* wordt bewaard in CLIKSW (F3DB). Bij het aanzetten van de computer is de *klik* ingeschakeld.

Het veld *baud* bepaalt de standaard snelheid waarmee files naar cassette worden geschreven. Normaal is deze standaard snelheid 1200 baud. De snelheid kan worden ingesteld op 2400 baud; echter, niet alle cassetterecorders zijn in staat om files die met deze snelheid zijn weggeschreven goed weer te geven.

Als *baud=1*, dan wordt de standaard snelheid 1200 baud; als *baud=2*, wordt de snelheid 2400 baud. De standaard snelheid kan ook veranderd worden in het statement CSAVE; zie daar.
De instelling van de standaard snelheid wordt veranderd door het deel van het geheugengebied CS120 (vanaf F3FC) dat de gegevens over de gewenste snelheid bevat te kopiëren naar LOW (F406), HIGH (F408) en HEADER (F40A).

Met het veld *print* kan informatie worden gegeven over het soort printer dat op de computer is aangesloten. Als print=0, is er een MSX-printer aangesloten, die alle speciale tekens van de MSX kan weergeven. Als *print*=1, is de aangesloten printer een niet-MSX-printer; in dat geval worden in de geprinte teksten alle speciale MSX-tekens vervangen door spaties.
De instelling van *print* wordt bewaard in NTMSX (F417) en RAWPRT (F418). Standaard staat *print* ingesteld op 0 (MSX-printer).

SCREEN maakt gebruik van de volgende OS-routine:

005F	CHGMOD	- verandert de schermmodus

SCREEN maakt gebruik van de volgende geheugenplaatsen:

F3AE	LINL32	- aantal tekens op een regel in modus 0
F3AF	LINL40	- aantal tekens op een regel in modus 1
F3B0	LINLEN	- aantal tekens in huidige schermmodus
F3DB	CLIKSW	- geeft aan of toetsenbordklik aan staat
F3DC	CSRY	- regelnummer cursor (bovenste regel nr. 1)
F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)
F3DE	CNSDFG	- geeft aan of functietoetsen getoond worden
F3FC	CS120	- gebied met gegevens voor schrijfsnelheden
F406	LOW	- gegevens voor uitvoer per cassette bit 0
F408	HIGH	- gegevens voor uitvoer per cassette bit 1
F40A	HEADER	- gegevens voor uitvoer kopblok
F417	NTMSXP	- geeft aan of de printer een MSX-printer is
FBB2	LINTTB	- tabel met gegevens over logische regels
FCAF	SCRMOD	- huidige schermmodus
FCB0	OLDSCR	- laatst gebruikte tekstmodus

SET D2 (210)

Categorie statement
Zie ook ATTR$

Dit statement wordt in het huidige systeem niet gebruikt, en is toegevoegd voor toekomstig gebruik.
In het statement SET is voorzien door middel van een RAM-haak: H.SETS (FDF4).

SGN FF+84 (255+132)

Categorie functie
Zie ook ABS
Vorm SGN(x)

Het argument x is een numerieke uitdrukking.
De functie SGN geeft informatie over het rekenkundige teken van de waarde van x.
Voor het resultaat van de functie geldt:

$x<0$	SGN(x) = –1
$x=0$	SGN(x) = 0
$x>0$	SGN(x) = +1

Hieruit volgt dat te allen tijde geldt: SGN(x)*ABX(x)=x

SIN FF+89 (255+137)

Categorie functie
Zie ook ATN, COS, TAN
Vorm SIN(x)

Het argument x is een numerieke expressie.
De waarde van x wordt geïnterpreteerd als een hoekgrootte, in radialen uitgedrukt.
De functie levert de sinus van die hoek op.

SOUND C4 (196)

Categorie statement
Zie ook BEEP, PLAY
Vorm SOUND *reg,x*

Reg is een byte-uitdrukking met een waarde tussen 0 en 13 (inclusief grenzen). Het veld x is een byte-uitdrukking.
SOUND schrijft de waarde van x naar het register van de PSG met nummer *reg*.
Door deze directe aansturing van de PSG-registers kunnen meer geluidseffecten verkregen worden dan door het gebruik van het statement PLAY; bijvoorbeeld kan er ruis gegenereerd worden.

Door het gebruik van SOUND werkt PLAY soms niet meer naar behoren. Dit komt doordat SOUND het OS-geluidssysteem omzeilt; zie par. 6.2.4. Het geluidssysteem kan weer ingeschakeld worden door het statement BEEP, of door het gebruik van de OS-routine GICINI (ingang 0090).
De betekenis van de registers van de PSG, en de wijze van gebruik, zijn in hoofdstuk 4 beschreven.

SOUND maakt gebruik van de OS-routine:

0093	WRTPSG	- schrijft waarde naar PSG-register

SPACE$ FF+99 (255+153)

Categorie functie
Zie ook SPC, STRING$
Vorm SPACE$*(len)*

Het argument *len* is een byte-uitdrukking.
De functie SPACE$ levert een string op die uit alleen spaties bestaat, en een lengte heeft die gelijk is aan de waarde van *len*.

SPC(DF (223)

Categorie hulpwoord
Zie ook LPRINT, PRINT, SPACE$, TAB(
Vorm SPC(*len*)

Het argument *len* is een byte-uitdrukking.
SPC(treedt op als hulpwoord in een PRINT- of LPRINT-statement. Het geeft aan dat er een serie spaties ter lengte *len* naar het scherm of de printer moet worden gestuurd.
Het verschil met SPACE$ is dat SPC(niet een echte functie is die iets oplevert; SPC(kan alleen als hulpwoord worden gebruikt.
Als een PRINT- of LPRINT-statement eindigt op het hulpwoord SPC(, dan wordt het statement niet afgesloten door RETURN.

N.B. 'Het haakje' openen hoort bij het sleutelwoord!

SPRITE C7 (199)

1.

Categorie statement
Zie ook INTERVAL, KEY, ON, STOP, STRIG
Vorm SPRITE *modus*

Modus kan de volgende waarden aannemen: ON, OFF en STOP.

Dit statement regelt de reactie op een sprite-botsing. MSX-BASIC kent de mogelijkheid om bij zo'n botsing een software-gestuurde interrupt te genereren; deze interrupt kan worden gebruikt d.m.v. een ON SPRITE-statement om naar een bepaalde plaats in een BASIC-programma te springen.

De interrupt wordt ingeschakeld met SPRITE ON en uitgeschakeld met SPRITE OFF. SPRITE STOP zorgt ervoor dat de gebeurtenis van een sprite-botsing niet direct leidt tot een interrupt, maar onthouden wordt tot de interrupt weer wordt ingeschakeld met SPRITE ON.

Voor meer gegevens over het interrupt-systeem zie par. 6.2.2.

SPRITE gebruikt een element van het geheugengebied:

FC4C	TRPTBL	- instelling van de interrupt-faciliteit

2.

Categorie hulpwoord
Zie ook ON

In deze betekenis is SPRITE een hulpwoord in een ON SPRITE-statement; zie daar.

3.

Categorie hulpwoord
Zie ook PUT

In deze betekenis is SPRITE een hulpwoord in een PUT SPRITE-statement; zie daar.

4.

Categorie hulpwoord
Zie ook SPRITE$
Vorm SPRITE$

In deze betekenis wordt SPRITE gebruikt om het woord SPRITE$ te coderen. SPRITE$ wordt gecodeerd als SPRITE+$.

SPRITE$ C7+24 (199+36)

Categorie systeemvariabele
Zie ook PUT, SPRITE
Vorm SPRITE$(*nr*)

Het veld *nr* is een byte-uitdrukking met waarde tussen 0 en 63 of tussen 0 en 255 (inclusief grenzen).
SPRITE$(nr) correspondeert met het element van de sprite-patroontabel met nummer *nr*. SPRITE$ kan worden uitgelezen, of er kan een nieuwe waarde aan worden

toegekend. Het verband tussen de waarde van SPRITE$ en de vorm van het patroon is vastgelegd in par. 3.1.4.

Welke waarden van *nr* toegestaan zijn, hangt af van het formaat van de sprites; zie SCREEN. Als het formaat van de sprites 8×8 is, mag *nr* liggen tussen 0 en 255, en is de waarde van SPRITE$ 8 bytes lang; anders mag *nr* niet groter zijn dan 63, en is de waarde van SPRITE$ 32 bytes lang.

SPRITE$ maakt gebruik van de OS-routines:

004A	RDVRM	- leest uit het videogeheugen
004D	WRTVRM	- schrijft naar het videogeheugen
0084	CALPAT	- levert beginadres van patroontabel-element
008A	GRPSIZ	- levert het formaat van de sprites op

SQR FF+87 (255+135)

Categorie functie
Zie ook EXP, LOG
Vorm SQR(*x*)

Het argument *x* is een numerieke uitdrukking met een niet-negatieve waarde.
De functie SQR levert de tweedemachtswortel van de waarde van *x* op.

STEP DC (220)

Categorie hulpwoord
Zie ook FOR

STEP kan gebruikt worden in een FOR-statement om de grootte van de verhoging van de waarde van de lusteller in te stellen.

2.

Categorie hulpwoord

STEP kan optreden als woord in elke coördinaatuitdrukking. Als STEP in zo'n uitdrukking voorkomt, betekent het dat de coördinaten die van die uitdrukking deel uitmaken niet als absolute waarde moeten worden opgevat, maar relatief moeten worden genomen ten opzichte van het laatst bereikte punt. De coördinaten van dit punt staan in GXPOS (FCB3) en GYPOS (FCB5).

STICK FF+A2 (266+162)

Categorie functie
Zie ook PAD, PDL
Vorm STICK(*x*)

Het argument *x* is een byte-uitdrukking met waarde 0, 1 of 2.

De functie STICK levert een waarde op die aangeeft welke richting het geselecteerde 'richtingssysteem' aangeeft. Het richtingssysteem wordt geselecteerd door de waarde van *x*, volgens de volgende tabel:

0 de cursortoetsen
1 de joystick aangesloten op ingang 1
2 de joystick aangesloten op ingang 2

De waarde die STICK oplevert ligt tussen 0 en 8 (inclusief grenzen). Deze waarde staat in verband met de richting die op het moment van aanroep wordt aangegeven door het geselecteerde systeem, en wel op de volgende wijze:

0 geen richting
1 naar boven
2 naar rechts boven
3 naar rechts
4 naar rechts onder
5 naar beneden
6 naar links onder
7 naar links
8 naar links boven

Het is duidelijk dat alle drie de richtingssystemen al deze richtingen kunnen aangeven.

STICK maakt gebruik van de OS-routine:

00D5	GTSTCK	- bepaalt toestand van richtingssysteem

STOP 90 (144)

1.

Categorie statement
Zie ook CONT, END
Vorm STOP

Het statement STOP heeft tot gevolg dat de uitvoering van het BASIC-programma stopt en dat er een melding wordt gegeven. Het programma kan voortgezet worden met het statement CONT.
Het verschil met END is, dat STOP het programma niet werkelijk beëindigt: files blijven open, terugkeeradressen van subroutines blijven bewaard, kortom, de hele situatie blijft gelijk.

STOP maakt gebruik van de volgende geheugenplaatsen:

F6BE	OLDLIN	- nummer van laatst uitgevoerde regel
F6C0	OLDTXT	- adres van volgend uit te voeren statement

2.

Categorie statement
Zie ook INTERVAL, KEY, ON, SPRITE, STRIG
Vorm STOP *modus*

Modus kan de volgende waarden aannemen: ON, OFF en STOP.
Dit statement regelt de reactie op het indrukken van CTRL+STOP. MSX-BASIC kent de mogelijkheid om bij het indrukken van deze toetscombinatie een softwaregestuurde interrupt te genereren; deze interrupt kan gebruikt worden d.m.v. een ON STOP-statement om naar een bepaalde plaats in een BASIC-programma te springen.
Normaal gesproken wordt bij het indrukken van CTRL+STOP het programma onderbroken.

De interrupt wordt ingeschakeld met STOP ON en uitgeschakeld met STOP OFF. STOP STOP zorgt ervoor dat de gebeurtenis van het indrukken van CTRL+STOP niet direct leidt tot een interrupt, maar onthouden wordt tot de interrupt weer ingeschakeld wordt met STOP ON.
Voor meer gegevens over het interrupt-systeem zie par. 6.2.2.

STOP gebruikt een element van het geheugengebied:

FC4C TRPTBL - instelling van de interrupt-faciliteit

3.

Categorie hulpwoord
Zie ook ON

In deze betekenis wordt STOP gebruikt als hulpwoord in een ON STOP-statement.

4.

Categorie hulpwoord
Zie ook OFF, ON

In deze betekenis wordt STOP gebruikt om allerlei interrupt-faciliteiten tijdelijk uit te schakelen. STOP komt voor als modus in de interrupt-instelling.

STR$ FF+93 (255+147)

Categorie functie
Zie ook BIN$, HEX$, OCT$, VAL
Vorm STR$(*x*)

Het argument *x* is een numerieke uitdrukking.
De functie STR$ levert een string op die de decimale representatie van de waarde van *x* bevat. Het eerste teken van de string is een spatie, of een minteken als de

waarde van *x* negatief is. Daarna volgen de cijfers van *x*, met eventueel een decimale punt. Als de waarde van *x* te groot is, wordt de waarde gerepresenteerd in exponentiële notatie.

STR$ doet het tegengestelde van VAL: er geldt altijd VAL(STR$(*x*))=*x*

STRIG FF+A3 (255+163)

1.

Categorie statement
Zie ook INTERVAL, KEY, ON, SPRITE, STOP
Vorm STRIG *(nr) modus*

Nr is een byte-uitdrukking met waarde tussen 1 en 5 (inclusief grenzen). *Modus* kan de volgende waarden aannemen: ON, OFF en STOP.

Dit statement regelt de reactie op het indrukken van een vuurknop. MSX-BASIC kent de mogelijkheid om bij het indrukken van zo'n knop een (software-gestuurde) interrupt te genereren; deze interrupt kan gebruikt worden d.m.v. een ON STRIG-statement om naar een bepaalde plaats in een BASIC-programma te springen.
Nr is het nummer van de bedoelde vuurknop. De interrupt-faciliteit wordt ingeschakeld met STRIG (*nr*) ON, en weer uitgeschakeld met STRIG (*nr*) OFF. STRIG (*nr*) STOP zorgt ervoor dat de gebeurtenis van het indrukken van de knop niet direct leidt tot een interrupt, maar onthouden wordt tot de faciliteit weer ingeschakeld wordt met STRIG (*nr*) ON.

STRIG gebruikt een element van het geheugengebied:
FCAC TRBTBL - geeft modus van interrupt en sprongadres

2.

Categorie hulpwoord
Zie ook ON

In deze betekenis wordt STRIG gebruikt als hulpwoord in een ON STRIG-statement; zie ON.

STRING$ E3 (227)

Categorie functie
Zie ook SPACE$, CHR$
Vorm 1. STRING$(*len,asc*)
2. STRING$(*len,str*)

De argumenten *len* en *asc* zijn byte-uitdrukkingen. *Str* is een string-uitdrukking. De functie STRING$ levert een string op die bestaat uit een serie van dezelfde

tekens, ter lengte van de waarde van *len*. Het teken waaruit de serie opgebouwd is wordt bepaald door *asc* in vorm 1 en *str* in vorm 2.

In vorm 1 is de waarde van *asc* de ASCII-code van het teken waaruit de serie bestaat. In vorm 2 is het eerste teken van de waarde van *str* het teken dat gebruikt wordt voor de serie. Zodoende geldt voor elke waarde van *asc*: STRING$(*len,asc*)=STRING$(*len*,CHR$(*asc*))

SWAP A4 (164)

Categorie statement
Zie ook LET
Vorm SWAP *ident1,ident2*

De velden *ident1* en *ident2* zijn variabelenamen of rij-elementen. *Ident1* en *ident2* moeten van hetzelfde type (string- of numerieke variabele) zijn.

Het resultaat van SWAP is dat de waarde van *ident1* aan *ident2* wordt toegewezen, en vice versa. Het gebruik van dit statement bij het uitwisselen van waarden maakt een hulpvariabele voor het bewaren van een van beide waarden overbodig.

SWAP gebruikt de volgende geheugenplaats:

F7BC	SWPTMP	- tijdelijke opslagplaats van waarde

TAB(DB (219)

Categorie hulpwoord
Zie ook LPRINT, PRINT
Vorm TAB(*pos*)

Het argument *pos* is een byte-uitdrukking.
TAB(treedt op als hulpwoord in een PRINT- of LPRINT-statement. Het geeft aan dat er net zo lang spaties geschreven moeten worden tot de cursor of printerkop de positie aangeduid door de waarde van *pos* heeft bereikt. Als vóór het uitvoeren van TAB(deze positie al gepasseerd is, heeft TAB(geen effect.

Als een PRINT- of LPRINT-statement eindigt op het hulpwoord TAB(, dan wordt het statement niet afgesloten door RETURN.

TAB(maakt gebruik van de volgende geheugenadressen:

F3DD	CSRX	- kolomnummer cursor (linker kolom nr. 1)
F415	LPTPOS	- positie printerkop

N.B. Het haakje openen hoort bij het sleutelwoord!

TAN FF+8D (255+141)

Categorie functie
Zie ook ATN, COS, SIN
Vorm TAN(*x*)

Het argument *x* is een numerieke uitdrukking.
De waarde van *x* wordt geïnterpreteerd als een hoekgrootte, in radialen uitgedrukt.
De functie levert de tangens van die hoek op.

THEN DA (218)

Categorie hulpwoord
Zie ook IF, ELSE

THEN wordt gebruikt in een IF-statement. Als de uitdrukking achter IF ongelijk is aan 0, worden de statements achter THEN uitgevoerd.
THEN in een IF-statement mag gevolgd worden door een regelnummer, in plaats van een serie statements. Deze situatie wordt behandeld alsof er tussen THEN en het regelnummer het woord GOTO staat.

TIME CB (203)

Categorie systeemvariabele
Vorm TIME

TIME staat in verbinding met de interne klok. Deze klok doorloopt het positieve integerbereik, en gaat terug naar 0 als de waarde groter wordt dan 65535.
De waarde van TIME wordt bij elke VDP-interrupt (50 maal per seconde) 1 opgehoogd. Aan TIME kan een nieuwe waarde toegewezen worden die binnen het integerbereik moet liggen.
De waarde van TIME wordt bewaard in JIFFY (FC9E).

TO D9 (217)

1.

Categorie hulpwoord
Zie ook FOR

In een FOR-statement wordt TO gebruikt om de eindwaarde van de lusvariabele aan te geven.

2.

Categorie hulpwoord
Zie ook COPY

In een COPY-statement wordt TO gebruikt als scheiding tussen file-uitdrukkingen.

TROFF A3 (163)

Categorie statement
Zie ook CONT, NEW, STOP, TRON
Vorm TROFF

Met dit statement wordt de volg- (Engels: trace-) faciliteit uitgeschakeld. Het effect van TRON wordt ongedaan gemaakt; zie TRON. De trace-faciliteit wordt ook uitgeschakeld met het statement NEW.
In de geheugenplaats TRCFLG (F7C4) wordt bijgehouden of de trace-faciliteit is ingeschakeld.

TROFF maakt gebruik van de volgende geheugenplaats:

F7C4	TRCFLG	- geeft aan of trace ingeschakeld is

TRON A2 (162)

Categorie statement
Zie ook CONT, STOP, TROFF
Vorm TRON

Het statement TRON schakelt de trace-faciliteit in. Dit heeft tot gevolg dat tijdens het uitvoeren van een BASIC-programma elke keer bij het bereiken van een nieuwe programmaregel het regelnummer tussen vierkante haken op het scherm gezet wordt, mits het scherm in tekstmodus staat.
De trace-faciliteit kan zeer van pas komen bij het foutloos maken van een programma. De faciliteit wordt uitgeschakeld door het statement TROFF of door het statement NEW.
In de geheugenplaats TRCFLG (F7C4) wordt bijgehouden of de trace-faciliteit is ingeschakeld.

TRON maakt gebruik van de volgende geheugenplaats:

F7C4	TRCFLG	- geeft aan of trace ingeschakeld is

USING E4 (228)

Categorie hulpwoord
Zie ook LPRINT, PRINT
Vorm USING *vorm;uitdr,...*

Het veld *vorm* is een string-uitdrukking. *Uitdr* is een uitdrukking van een of ander type. Het mogelijke aantal velden *uitdr* is onbeperkt. Deze velden worden gescheiden door komma's of puntkomma's.
USING treedt op als element (hulpwoord) in een PRINT- of LPRINT-statement. Het USING-element wordt gebruikt om nette uitvoer in een van te voren vastgelegde vorm te verkrijgen.

De vorm van de uitvoer is vastgelegd door de waarde van *vorm*. Het resultaat van het USING-element is dat de tekst van *vorm* naar het scherm of de printer wordt geschreven. In deze tekst worden de velden *uitdr* als variabelen ingevuld.

Alle elementen die volgen op een USING-element worden opgevat als uitdrukkingen die in *vorm* moeten worden ingevuld. Er kan dus nooit een SPC(- of TAB(-element volgen op een USING-element in een PRINT- of LPRINT-statement.
De waarde van vorm bevat velden waarin de waarden van de verschillende voorkomens van *uitdr* als variabelen worden ingevuld. Deze velden worden op de volgende manier aangegeven:

!
Op de plaats van dit veld wordt het eerste teken van een string-waarde ingevuld.

\spaties
Op de plaats van dit veld worden de eerste tekens van een string-waarde ingevuld, ter lengte van het aantal spaties +2.

&
Op de plaats van dit veld wordt een komplete string-waarde ingevuld.

\# (hekjes, één of meer)
Op de plaats van dit veld wordt een getalwaarde ingevuld. Als het aantal cijfers van het getal kleiner is dan het aantal hekjes, wordt het aan de voorkant aangevuld met spaties.

Een getalveld, bestaande uit hekjes, kan met de volgende tekens uitgebreid worden:

. (punt)
Als tussen de hekjes een punt staat, komt de decimale punt van de in te vullen getalwaarde op die plaats te staan.

, (komma)
Als direct voor een punt (zie hiervoor) een komma staat, wordt in het gehele deel van het getal tussen elk drietal cijfers een komma gezet.

\+ (plusteken)
Als een getalveld voorafgegaan wordt door een plusteken, komt er voor het getal in de uitvoer een plus- of minteken te staan. Als het plusteken achter het getalveld staat, wordt het plus- of minteken ook achter het getal gezet.

\- (minteken)
Als er achter een getalveld een minteken staat, wordt het eventuele minteken van het getal in de uitvoer niet voor maar achter het getal gezet.

** (twee sterretjes)
Als er voor een getalveld twee sterretjes staan, wordt het getal aan de voorkant aangevuld met sterretjes in plaats van met spaties.

$$ (twee dollartekens)
Als er voor een getalveld twee dollartekens staan, wordt er voor het getal in de uitvoer een dollarteken gezet. Het eventuele plus- of minteken van het getal komt dan nog vóór het dollarteken.

**$ (twee sterretjes en een dollarteken)
Dit is een combinatie van de twee voorgaande mogelijkheden. Direct voor het getal komt in de uitvoer een dollarteken te staan, en verder wordt het veld opgevuld met sterretjes.

^ ^ ^ ^ (vier dakjes)
Wanneer er vier dakjes achter een getalveld staan, wordt het getal in de uitvoer gerepresenteerd in exponent-notatie.

Alle tekens in *vorm* die niet tot een van de bovenstaande veldbeschrijvingen horen, verschijnen in de uitvoer precies zoals ze in *vorm* staan.

De voorkomens van *uitdr* moeten van hetzelfde type zijn als het bijbehorende veld in de waarde van *vorm*. Als er meer voorkomens van *uitdr* zijn dan velden in de waarde van *vorm*, wordt *vorm* nogmaals toegepast. Als er minder voorkomens van *uitdr* zijn dan velden in *vorm*, dan houdt de uitvoer op bij het laatste voorkomen van *uitdr*.

USR DD (221)

1.

Categorie hulpwoord
Zie ook DEF

USR wordt gebruikt als hulpwoord in de definitie van een machinetaalroutine.

2.

Categorie hulpwoord
Zie ook DEF
Vorm USR*cijf(arg)*

Het veld *cijf* is een cijfer, dus een teken tussen 0 en 9 (inclusief grenzen). *Cijf* mag weggelaten worden; dit heeft hetzelfde effect als *cijf* = 0. *Arg* is een uitdrukking van een of ander type.
Door USR*cijf* te gebruiken, wordt een machinetaalroutine aangeroepen. USR*cijf* moet vóór gebruik gedefinieerd zijn in een DEF USR-statement; zie DEF. In dat statement is het beginadres van de routine van USR*cijf* vastgelegd. Het beginadres wordt opgeslagen in het geheugengebied USRTAB (vanaf F39A).

Bij het uitvoeren van USR*cijf* wordt allereerst het veld *arg* verwerkt. Dit gebeurt door informatie over het type van *arg* (string, integer, enkele of dubbele precisie) op te slaan in VALTYP (F663), en de waarde van *arg* op te slaan binnen DAC (tussen

F7F6 en F7FD). Vervolgens wordt een JSR uitgevoerd naar het beginadres van de routine.
De routine moet worden beëindigd met een RET-instructie. De waarde die de functie USR*cijf* oplevert, staat op dezelfde manier als *arg* gecodeerd in VALTYP en DAC.
De waarde van *arg* en het resultaat van de functie USR*cijf* zijn op de volgende manier gecodeerd:

type	*VALTYP*	*opslag*
integer	2	waarde in F7F8-F7F9
string	3	adres van descriptor in F7F8-F7F9
enkele precisie	4	waarde in F7F6-F7F9
dubbele precisie	8	waarde in F7F6-F7FD

USR maakt in deze betekenis gebruik van de geheugenplaatsen:

F39A	USRTAB	- tabel van beginadressen USR-routines
F663	VALTYP	- type van uitdrukking
F7F6	DAC	- waarde van uitdrukking

VAL FF+94 (255+148)

1.

Categorie functie
Zie ook STR$
Vorm VAL(*str*)

Het argument *str* is een string-uitdrukking.
VAL zet de waarde van *str* om naar een getalwaarde. De waarde van str moet de representatie van een getalconstante zijn. Deze representatie mag alle mogelijke vormen aannemen. VAL levert de waarde van de gerepresenteerde constante op.

VAL maakt gebruik van de volgende geheugenplaatsen:

F419	VLZADR	- adres van teken dat door VAL vervangen is
F41B	VLZDAT	- teken dat op adres VLZADR hoort te staan

2.

Categorie hulpwoord
Zie ook INT, INTERVAL
Vorm INTERVAL

In deze betekenis wordt VAL gebruikt om het woord INTERVAL te coderen; zie INTERVAL.Het woord wordt gecodeerd als INT+E+R+VAL.

VARPTR E7 (231)

1.

Categorie functie
Vorm VARPTR(*ident*)

Het argument *ident* is de naam van een variabele of een rijelement van een willekeurig type.
De functie VARPTR levert het beginadres op van de waarde van ident.

VARPTR gebruikt in deze betekenis de geheugenadressen:

F6C2	VARTAB	- beginadres variabelen-opslaggebied
F6C4	ARYTAB	- beginadres rijvariabelen-opslaggebied
F6C6	STREND	- adres eerste vrije geheugenplaats
F7B5	ARYTA2	- eindadres voor variabelen-zoekprocedure

2.

Categorie functie
Vorm VARPTR(*fnr*)

Het argument *fnr* is een file-nummeruitdrukking, voorafgegaan door een hekje (#).
De functie VARPTR levert het beginadres op van de file-buffer met nummer *fnr*. Er hoeft geen file geopend te zijn met nummer *fnr*.

VARPTR gebruikt in deze betekenis de geheugenplaatsen:

F85F	MAXFIL	- hoogst toegestane file-nummer (aantal buffers)
F860	FILTAB	- beginadres tabel file-buffers

VDP C8 (200)

Categorie systeemvariabele
Zie ook BASE, VPEEK, VPOKE
Vorm VDP(*nr*)

Het argument *nr* is een byte-uitdrukking met waarde tussen 0 en 8 (inclusief grenzen).
De systeemvariabele VDP is verbonden met de VDP-registers. Het registernummer wordt bepaald door de waarde van *nr*. Met behulp van VDP kunnen de registers van de VDP worden uitgelezen of van nieuwe waarden worden voorzien. Alleen VDP(*8*) kan niet van een nieuwe waarde worden voorzien.

De VDP-registers kunnen niet direct worden uitgelezen. Om die reden worden de waarden die het laatst naar de registers zijn geschreven bewaard in geheugenplaatsen RG0SAV t/m RG7SAV (F3DF-F3E6). De waarde die opgeleverd wordt bij het opvragen van een VDP-variabele is dan ook in feite de waarde van een van deze geheugenplaatsen.

De VDP-registers en het gebruik ervan zijn beschreven in par. 3.2.1.

VDP gebruikt de volgende OS-routine:

0047	WRTVDP	- schrijft een nieuwe waarde naar een VDP-reg

VDP gebruikt de volgende geheugenplaatsen:

F3DF	RG0SAV	- laatst bepaalde waarde van VDP-register 0
F3E0	RG1SAV	- laatst bepaalde waarde van VDP-register 1
F3E1	RG2SAV	- laatst bepaalde waarde van VDP-register 2
F3E2	RG3SAV	- laatst bepaalde waarde van VDP-register 3
F3E3	RG4SAV	- laatst bepaalde waarde van VDP-register 4
F3E4	RG5SAV	- laatst bepaalde waarde van VDP-register 5
F3E5	RG6SAV	- laatst bepaalde waarde van VDP-register 6
F3E6	RG7SAV	- laatst bepaalde waarde van VDP-register 7
F3E7	STATFL	- laatst ingelezen waarde van VDP-register 8

VPEEK FF+98 (255+152)

Categorie functie
Zie ook BASE, VDP, VPOKE
Vorm VPEEK(*adr*)

Het argument *adr* is een integeruitdrukking.
VPEEK leest de inhoud van een byte van het videogeheugen. Welk byte wordt gelezen, wordt bepaald door de waarde van *adr*.

VPEEK maakt gebruik van de volgende OS-routine:

004A	RDVRM	- leest waarde uit het videogeheugen

VPOKE C6 (198)

Categorie statement
Zie ook BASE, VDP, VPEEK
Vorm VPOKE *adr,x*

Het veld *adr* is een integeruitdrukking. Het veld x is een byte-uitdrukking.
VPOKE zet de waarde van *x* in het videogeheugen, op de geheugenplaats met adres *adr*.

VPOKE maakt gebruik van de volgende OS-routine:

004D	WRTVRM	- zet een waarde in het videogeheugen

WAIT 96 (150)

Categorie statement
Zie ook INP, OUT
Vorm WAIT *nr,x,y*

De velden *nr, x* en *y* zijn byte-uitdrukkingen. Het veld *y* mag worden weggelaten, mits de voorafgaande komma ook wordt weggelaten.
WAIT test bij voortduring de waarde van de I/O-poort met poortadres *nr*. Het statement wordt pas beëindigd als deze test slaagt. De test is afhankelijk van de waarden van x en y volgens de volgende formule:

((INP(*nr*) XOR *y*) AND *x*) 0

Door de keuze van *x* worden bits van de waarde van de poort geselecteerd. Alleen op die bits wordt getest. Door de keuze van *y* wordt van die bits getest of ze hoog dan wel laag zijn. De test is geslaagd als minstens één van de geteste bits aan de voorwaarde voldoet.
Als *y* wordt weggelaten, wordt voor de waarde van *y* nul ingevuld.

WIDTH A0 (160)

Categorie statement
Zie ook SCREEN
Vorm WIDTH *x*

Het veld *x* is een byte-uitdrukking.
Dit statement verandert het aantal gebruikte kolommen van een schermregel in een tekstmodus. Dit aantal wordt bewaard in LINL32 (F3AE) voor tekstmodus 0 en in LINL40 (F3AF) voor tekstmodus 1.
Als het scherm in een tekstmodus staat, verandert WIDTH het aantal kolommen van die tekstmodus; anders verandert het aantal kolommen van de tekstmodus die het laatst gebruikt is, bewaard in OLDSCR (FCB0).
Als het scherm in een tekstmodus staat en de waarde van *x* verschilt van het huidige aantal kolommen, bewaard in LINLEN (FB30), dan maakt WIDTH tevens het scherm schoon door middel van de OS-routine CLS (ingang 00C3).

WIDTH gebruikt de volgende OS-routine:

00C3	CLS	- maakt het scherm schoon

WIDTH maakt gebruik van de volgende geheugenplaatsen:

F3AE	LINL40	- standaard aantal kolommen in schermmodus 0
F3AF	LINL32	- standaard aantal kolommen in schermmodus 1
F3B0	LINLEN	- aantal kolommen in huidige modus
FCAF	SCRMOD	- huidige schermmodus
FCB0	OLDSCR	- laatst gebruikte schermmodus

XOR F8 (248)

Categorie operator
Zie ook AND, EQV, IMP, NOT, OR
Vorm *x* XOR *y*

De operatoren *x* en *y* zijn integerexpressies.
De operanden worden gecodeerd volgens 2-complement. Het resultaat van XOR is een getal, ook gecodeerd volgens 2-complement, waarvan elk bit direct afhankelijk is van het overeenkomstige bit in *x* en *y*.

Voor XOR geldt het volgende diagram:

bit van *x*:	0	0	1	1
bit van *y*:	0	1	0	1
bit van XOR:	0	1	1	0

XOR is een logische operator met prioriteit 2; zie par. 7.2.4.

Appendix A Byte-coderingen

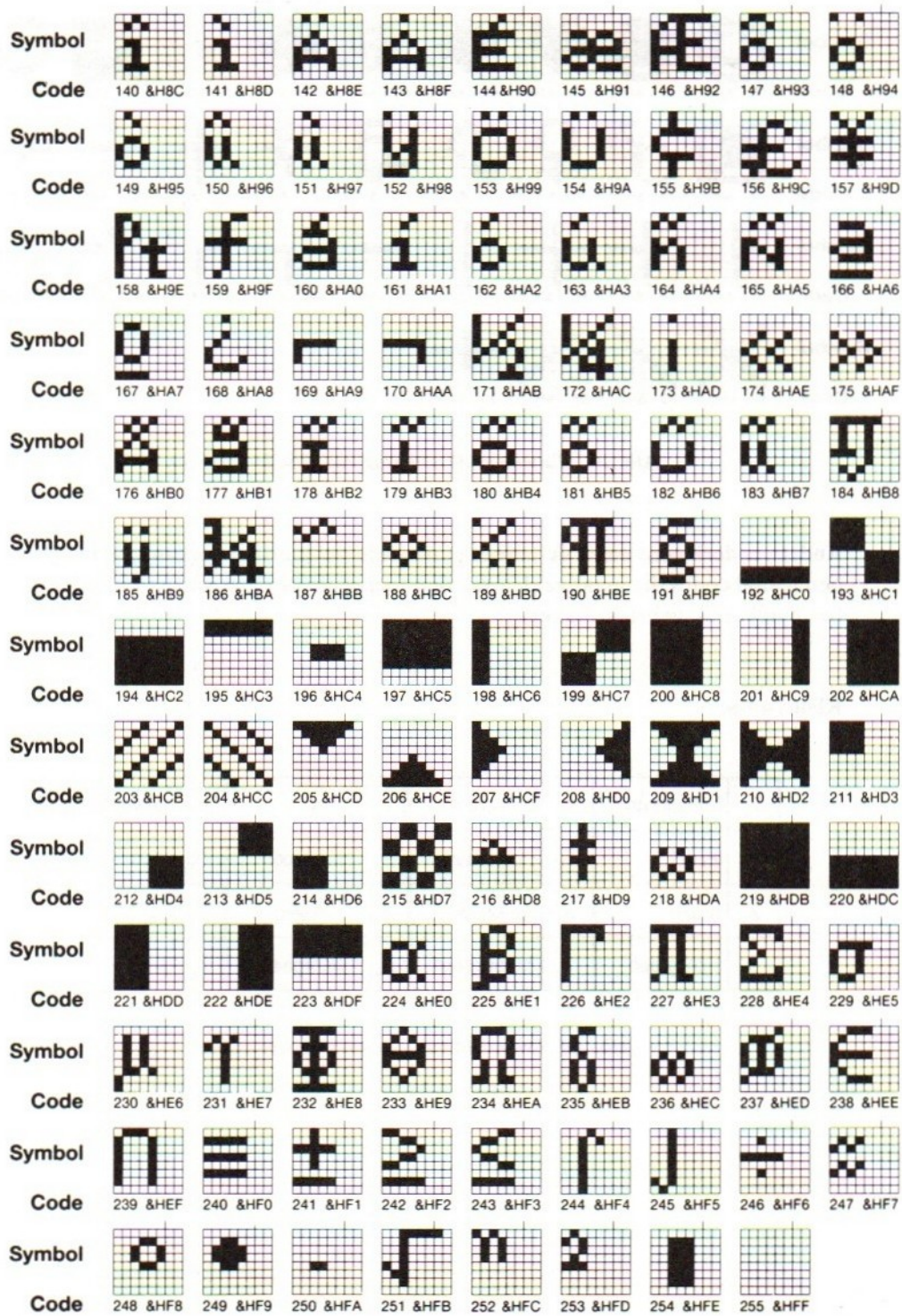
Het waardebereik van één byte (0-255) wordt in een heleboel verschillende situaties gebruikt om verschillende dingen te coderen. Een voorbeeld van zo'n codering is de BASIC-tabel.

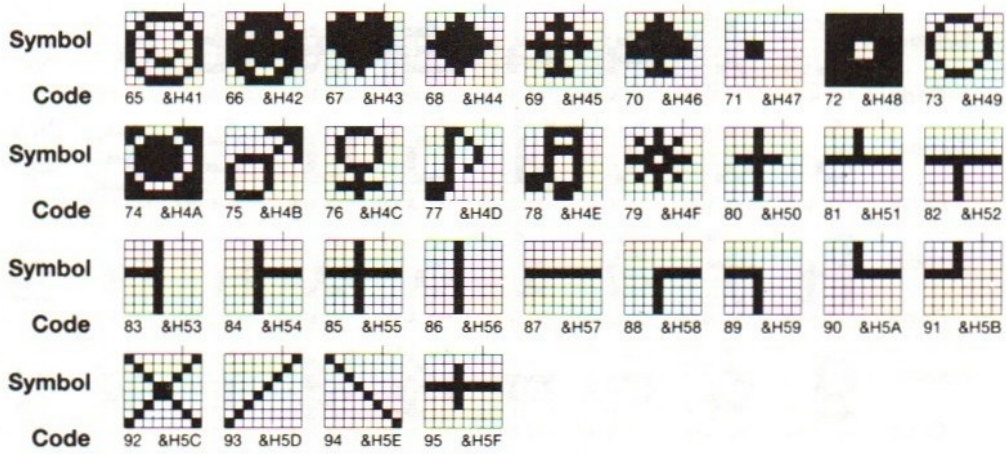
	0_{000}	1_{016}	2_{032}	3_{048}	4_{064}	5_{080}	6_{096}	7_{112}	8_{128}	9_{144}	A_{160}	B_{176}	C_{192}	D_{208}	E_{224}	F_{240}
0_0	einde BASIC-regel		spatie	0	@***	P	`***	p***		STOP INP*	WIDTH CDBL*	OPEN MKDS*	BEEP	BSAVE	NOT	<**
1_1		const. waarde 0	!	1	A	Q	a***	q***	END LEFT$*	PRINT POS*	ELSE FIX*	FIELD	PLAY	DSKO$	ERL	+**
2_2		const. waarde 1	"	2***	B	R	b***	r***	FOR RIGHT$*	CLEAR LEN$*	TRON STICK*	GET	PSET	SET	ERR	-**
3_3		const. waarde 2	#	3***	C	S	c***	s***	NEXT MID$*	LIST STR$	TROFF STRIG*	PUT	PRESET	NAME	STRING$	***
4_4		const. waarde 3	$	4***	D	T	d***	t***	DATA SGN*	NEW VAL*	SWAP PDL*	CLOSE	SOUND	KILL	USING	/**
5_5		const. waarde 4	%	5***	E	U	e***	u***	INPUT INT*	ON ASC*	ERASE PAD*	LOAD	SCREEN	IPL	INSTR	^**
6_6		const. waarde 5	&	6***	F	V	f***	v***	DIM ABS*	WAIT CHR$*	ERROR DSKF*	MERGE	VPOKE	COPY	**	AND
7_7		const. waarde 6	'	7***	G	W	g***	w***	READ SQR*	DEF PEEK*	RESUME FPOS*	FILES	SPRITE	CMD	VARPTR	OR
8_8		const. waarde 7	(	8***	H	X	h***	x***	LET RND*	POKE VPEEK*	DELETE CVI*	LSET	VDP	LOCATE	CSRLIN	XOR
9_9		const. waarde 8	)	9***	I	Y	i***	y***	GOTO SIN*	CONT SPACE$*	AUTO CVS*	RSET	BASE	TO	ATTR$	EQV
A_{10}		const. waarde 9	***	:	J	Z	j***	z***	RUN LOG*	CSAVE OCT$*	RENUM CVD*	SAVE	CALL	THEN	DSKI$	IMP
B_{11}	info-byte oct.repr.	const. waarde 10	+***	;	K	[***	k***	{***	IF EXP*	CLOAD HEX$*	DEFSTR EOF*	LFILES	TIME	TAB(	OFF	MOD
C_{12}	info-byte hex.repr.	info-byte integer	,	<***	L	***	l***	|***	RESTORE COS*	OUT LPOS*	DEFINT LOC*	CIRCLE	KEY	STEP	INKEY$	**
D_{13}	info-byte regelwijzer	info-byte enkele prec.real	_***	=***	M	]***	m***	}***	GOSUB TAN*	LPRINT BIN$*	DEFSNG LOF*	COLOR	MAX	USR	POINT	
E_{14}	info-byte regelindex		.***	>***	N	^***	n***	~***	RETURN ATN*	LLIST CINT*	DEFDBL MKI$*	DRAW	MOTOR	FN	>**	
F_{15}	info-byte byte const.	info-byte dubbele prec.real	/***	?***	O	_***	o***	△***	REM FRE*	CLS CSNG*	LINE MKS$	PAINT	BLOAD	SPCL	=**	begin code 2e helft

indien voorafgegaan door byte &HFF
*sleutelwoord. Heeft een andere betekenis dan het ASCII-teken
**komt normaal gesproken niet voor

Afb. A1. Tabel van BASIC-codes

ASCII-patronen: in de patroontabel van het videogeheugen staan in de basis-instelling van de VDP de ASCII-tekens opgeslagen volgens de volgende codering:


Symbol
Code
140 &H8C
141 &H8D
142 &H8E
143 &H8F
144 &H90
145 &H91
146 &H92
147 &H93
148 &H94
Symbol
Code
149 &H95
150 &H96
151 &H97
152 &H98
153 &H99
154 &H9A
155 &H9B
156 &H9C
157 &H9D
Symbol
Code
158 &H9E
159 &H9F
160 &HA0
161 &HA1
162 &HA2
163 &HA3
164 &HA4
165 &HA5
166 &HA6
Symbol
Code
167 &HA7
168 &HA8
169 &HA9
170 &HAA
171 &HAB
172 &HAC
173 &HAD
174 &HAE
175 &HAF
Symbol
Code
176 &HB0
177 &HB1
178 &HB2
179 &HB3
180 &HB4
181 &HB5
182 &HB6
183 &HB7
184 &HB8
Symbol
Code
185 &HB9
186 &HBA
187 &HBB
188 &HBC
189 &HBD
190 &HBE
191 &HBF
192 &HC0
193 &HC1
Symbol
Code
194 &HC2
195 &HC3
196 &HC4
197 &HC5
198 &HC6
199 &HC7
200 &HC8
201 &HC9
202 &HCA
Symbol
Code
203 &HCB
204 &HCC
205 &HCD
206 &HCE
207 &HCF
208 &HD0
209 &HD1
210 &HD2
211 &HD3
Symbol
Code
212 &HD4
213 &HD5
214 &HD6
215 &HD7
216 &HD8
217 &HD9
218 &HDA
219 &HDB
220 &HDC
Symbol
Code
221 &HDD
222 &HDE
223 &HDF
224 &HE0
225 &HE1
226 &HE2
227 &HE3
228 &HE4
229 &HE5
Symbol
Code
230 &HE6
231 &HE7
232 &HE8
233 &HE9
234 &HEA
235 &HEB
236 &HEC
237 &HED
238 &HEE
Symbol
Code
239 &HEF
240 &HF0
241 &HF1
242 &HF2
243 &HF3
244 &HF4
245 &HF5
246 &HF6
247 &HF7
Symbol
Code
248 &HF8
249 &HF9
250 &HFA
251 &HFB
252 &HFC
253 &HFD
254 &HFE
255 &HFF



Afb. A3. Tabel van de codering van ASCII-patronen

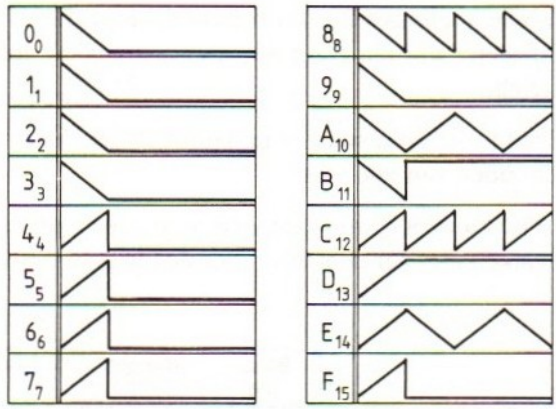
Andere coderingen: naast de eerder genoemde belangrijke coderingswijzen zijn er een aantal codes die maar zelden worden toegepast of maar van een klein deel van het bytebereik gebruik maken.

Kleurcodes:

0_0	transparant	4_4	donkerblauw	8_8	rood	C_{12}	donkergroen
1_1	zwart	5_5	lichtblauw	9_9	lichtrood	D_{13}	paars
2_2	groen	6_6	donkerrood	A_{10}	donkergeel	E_{14}	grijs
3_3	lichtgroen	7_7	blauwgroen	B_{11}	lichtgeel	F_{15}	wit

Afb. A4. Overzicht van de codering van kleuren

Codes voor volumeverloop:



Afb. A5. Overzicht van volumeverloop-codes

Besturingscodes voor de printer: (alle niet-vermelde codes hebben dezelfde betekenis als in string-waarden):

00		08		10		18	
01	grafische extensie	09		11		19	
02		0A	regel verder	12		1A	
03		0B		13		1B	ESC *
04		0C	pagina verder	14		1C	
05		0D	naar begin regel	15		1D	
06		0E		16		1E	
07		0F		17		1F	

* 1B+41=6 regels/inch of lege regel
1B+42=8 regels/inch of geen lege regel
1B+53=grafische modus moet gevolgd worden door 3 cyfers in ASCII die het aantal bitpatronen bepalen

Afb. A6. Overzicht van printer-besturingscodes

Appendix B File-opslag

In par. 6.6 is het file-afhandelingssysteem van het OS besproken. De manier waarop files op cassette of diskette worden gezet is echter niet aan de orde gekomen. Hier volgt een kort overzicht.

Files op cassette: elke file die op cassette staat vormt een eenheid. Er wordt nergens een overzicht bijgehouden van alle files.

De files op een cassette kunnen in drie categorieën worden onderverdeeld. De categorieën hebben te maken met het doel waarvoor de file gebruikt wordt. De categorieën zijn:

		BASIC file	geheugen dump	datafile
file descriptor blok	lang kopblok	ja	ja	ja
	file kategorie-aanduiding	10x &HD3	10x &HD0	10x &HEA
	filenaam	6 bytes ASCII	6 bytes ASCII	6 bytes ASCII
file-inhoud	kort kopblok	ja	ja	ja
	hoofddeel	gecodeerde BASIC tekst	2 bytes beginadres	blokken van 256 bytes gescheiden door korte kopblokken
			2 bytes eindadres	
			2 bytes aanroepadres	
			waarde van geheugen blok	
	beëindiging	7x &H00	impliciet in begin- en eindadres	laatste blok bevat &H1A

Afb. A8. Overzicht van file-formaten op cassette

BASIC-file: een file die de codering van een BASIC-programma bevat. Deze file wordt gemaakt door met CSAVE een programma weg te schrijven. De file kan alleen met CLOAD of RUN weer worden ingelezen.

Geheugendump: een file waarin een kopie van de waarden van een stuk video- of gewoon geheugen. Het weggeschreven blok zal over het algemeen een machinetaal-programma of een plaatje bevatten dat dan later weer kan worden ingelezen.

Een geheugendump kan worden gemaakt met het statement BSAVE, en kan alleen weer worden ingelezen met BLOAD.

Data-file: files die gebruikt worden om gegevens die binnen een programma nodig zijn weg te schrijven en weer in te lezen. Ook BASIC-programma's die in ASCII-formaat zijn weggeschreven horen in deze categorie.

Een data-file kan worden gemaakt door SAVE, en worden geladen door LOAD of MERGE. Verder kunnen het OPEN-statement en alle bijbehorende statements gebruikt worden voor data-files.

In het volgende overzicht is weergegeven hoe de files van de verschillende categorieën op cassette gezet worden.

Files op diskette: op een diskette wordt naast de file-inhoud een hoeveelheid systeeminformatie opgeslagen. Deze informatie betreft de plaats van de files op diskette, en het huidige gebruik van de sectoren van de diskette.

Van elke geformatteerde diskette worden een aantal sectoren gereserveerd voor het opslaan van systeeminformatie. Welke sectoren dit zijn, hangt af van het soort diskettestation.

De systeeminformatie valt uiteen in drie delen: bootstrap info, directory (overzicht van files) en FAT (File Allocation Table), ofte wel file-toewijzingstabel).

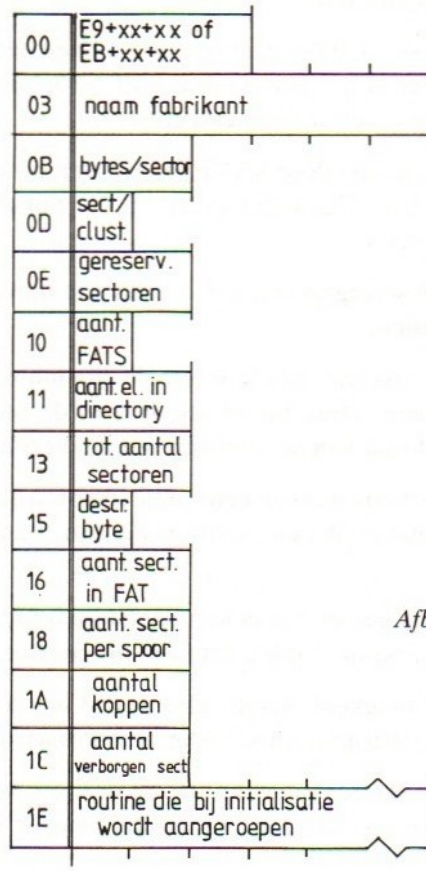
Bootstrap info: 'bootstrap' betekent 'veter'. Het woord wordt hier gebruikt in de betekenis 'opstart-'; een bootstrap-routine is dus een opstartroutine.

De bootstrap-informatie bevat een gedeelte dat de diskette beschrijft – het soort diskette, het aantal sectoren op een spoor, het aantal bytes in een sector, de fabrikant, etc.

Een ander gedeelte van de bootstrap-informatie vormt een opstartroutine voor de diskette.

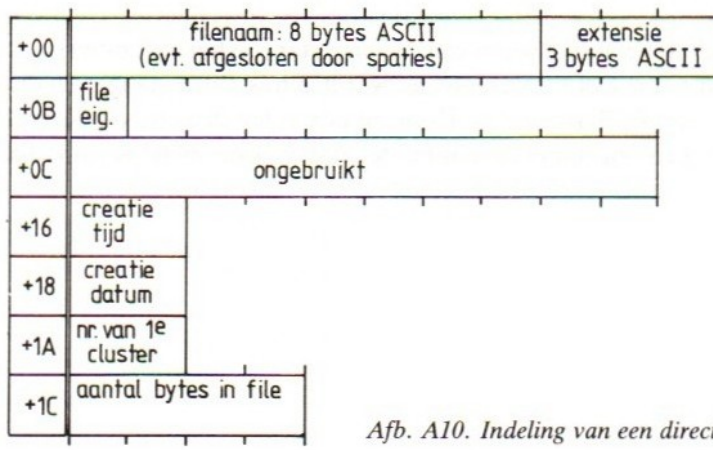
De bootstrap-informatie van het diskettestation staat op sector 0, spoor 0 van de diskette. Deze sector wordt ook wel de boot-sector genoemd. De inhoud van deze sector wordt bij het initialiseren van de computer (bij het aanzetten) ingelezen in het adresgebied C000-C0FF. Vervolgens wordt adres C0IE aangeroepen. Hier staat gewoonlijk een RET-instructie. Door op een eigen diskette op deze plaats van de boot-sector een machinetaalroutine te zetten kan men de initialisatie onderscheppen.

De boot-sector is als volgt ingedeeld:



Afb. A9. De boot-sector

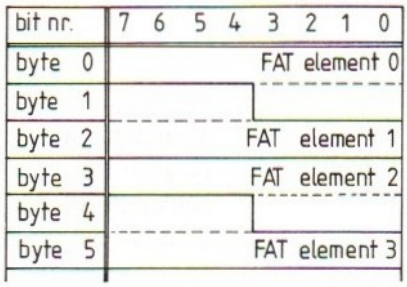
Directory: de directory is een overzicht van de files die er op de diskette staan. Elke file heeft een element in de directory. Zo'n element is als volgt ingedeeld:



Afb. A10. Indeling van een directory-element

Voor de betekenis van het eerste clusternummer zie onder.

FAT: de FAT geeft aan welke clusters van de diskette op dit moment gebruikt worden voor file-opslag. Een cluster is gelijk aan 1kbyte opslagruimte.



Afb. A11. FAT

Elk cluster heeft een element in de FAT. Zo'n element beslaat 1½ bytes. De elementen van de FAT sluiten direct op elkaar aan. Dat betekent dat de FAT er als volgt uit ziet:

Een FAT-element bevat het nummer van het cluster dat volgt op het cluster waarbij het element hoort. De inhoud van een file wordt opgeslagen in een serie clusters. Door nu de FAT te bestuderen kan uit elk cluster het volgende worden bepaald. Bij de file-opslaginformatie hoeft nu alleen het nummer van het eerste cluster worden opgeslagen, aangezien de volgende clusters met behulp van de FAT op te zoeken zijn.

Wanneer een cluster op dit moment niet wordt gebruikt voor file-opslag, bevat het bijbehorende FAT-element 0.

INDEX

Algemeen

BASIC-functies en -commando's

RAM-haken

MSX-computers hebben inmiddels een zeer groot aandeel in de Nederlandse computermarkt verworven. Als gevolg hiervan stijgt de behoefte aan goede, diepgaande boeken over deze computers.

Het MSX-handboek voor gevorderden biedt doorgewinterde MSX-gebruikers en -programmeurs een antwoord op alle MSX-problemen.

De zeer gedetailleerde omschrijvingen van alle BASIC-statements, aangevuld met een uitgebreid BIOS-overzicht en een bespreking van de specifieke MSX-chips, zijn voorzien van talrijke verhelderende illustraties.

Het is niet de bedoeling van dit boek om een uitleg te geven van de basisgegevens van machinetaal, bytes of bits. Deze kennis veronderstellen we bij de lezers reeds aanwezig. Wél maken we gebruik van deze basisgegevens om de speciale eigenschappen en mogelijkheden van de MSX-computer duidelijk te maken.

Sommige gegevens in dit boek zijn wellicht reeds bekend uit de MSX-gebruiksaanwijzing of uit andere bronnen. De meeste onderwerpen zijn echter volkomen nieuw en zijn betrokken uit nog niet eerder gepubliceerde MSX-informatie.

ISBN 90 201 1925 7