De SHARP MZ-800 ten voeten uit

geschreven door **ARJAN HABING**
mmv **MARK** de **ROVER**

NEPTUNES PRODUCTIONS

Geachte SHARP MZ-800 bezitter,

Hierbij ontvangt U, als bezitter van het boek "De SHARP MZ-800 ten voeten uit", aanvullende informatie met betrekking tot het genoemde boek.
Achteraf is gebleken dat er enkele fouten in het boek zijn geslopen. Op die fouten en op enkele andere dingen wil ik hier graag verder ingaan en U eventueel de juiste versie geven.

Tijdens het drukken van het boek zijn enkele interne veranderingen opgetreden. De juiste gegevens van Neptunes Productions zijn nu de volgende:

Neptunes Productions
Prakkestraat 19
7741 CK Coevorden
tel:05240-14646 (gaarne alleen in het weekend!)
KvK Meppel 38490
Bank: Bondspaarbank Vlaardingen/Schiedam 977600130
Giro: 234247

Voor specifieke vragen verzoek ik U zich tot dit adres te wenden en niet aan het in het boek vermelde adres.

Het adres van de CPM-man (Ton de Pan) is anders dan in het boek vermeld. Het juiste adres is:

Ton de Pan
Lengweg 42
Hoogvliet
tel:010-4161033

<u>Enkele kleine foutjes in het boek:</u>

blz 72 - POKE NR.14 Oorspronkelijke POKE moet worden:
POKE $6079,$CD,$9B,$61

blz 150 - Voorbeeld 1 adres 04E8 moet zijn : adres 048E
Eveneens: Voer eerst M048E in

Op enkele kleine informatie-punten in het boek is een aanvulling/verbetering nodig. Deze punten zijn niet echt van belang, maar voor de critici misschien toch wel storend.

blz 8 - Het zelf afstellen van de cassetterecorder wordt afgeraden. Er schijnt een programma te bestaan (helaas in het buitenland) dat gebruikt kan worden voor de afstelling van de cassette-recorder kop. Op hardwarematige manier is het ook mogelijk, daarvoor kunt U eventueel dhr Gans bellen (zie blz. 241).

blz 9 - Niet alleen matrixprinters, maar ook thermische, inktjet-, laser- en margrietwielprinters kunnen op de SHARP MZ-800 gebruikt worden.

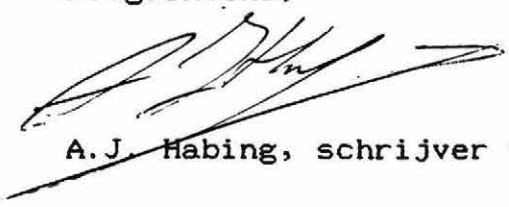
blz 8 - Over RGB-standaarden bestaan grote onduidelijkheden. Zeker is dat er meer dan één standaard is. Voor eventuele vragen hierover kunt U eveneens terecht bij dhr Gans of bij dhr Smits.

blz ? - UNI-gg-BASIC. Dat het boek alleen op de standaard-BASIC is gericht, is iets dat het hele boek tegen spreekt.
De SHARP MZ-gg klaagt dat het boek veel te weinig op deze BASIC in gaat. Zoals U zelf heeft kunnen constateren, is slechts 1-2% van het boek specifiek voor de standaard BASIC-800 bedoeld en kan de rest ook voor UNI-gg-BASIC gebruikt worden.

blz 238 - SHARP Computer Club Edam.
In de tussentijd heb ik tot tweemaal toe een bezoek af kunnen leggen bij deze club. Mijn opgedane ervaring wil ik U niet onthouden.
Van de drie clubs die er zijn heeft SCCEdam de kleinste clubruimte, die echter tot in alle hoeken gevuld is. Het is in geen enkel opzicht te vergelijken met de twee andere clubs die er zijn.
De sfeer die er heerst is erg gezellig. Een verloting wordt gehouden om de clubkas te spekken, maar daar staan heel aardige prijsjes tegenover.
Voor wie iets wil weten is genoeg deskundigheid aanwezig. Leuke demonstraties moeten een extra dimensie aan de bijeenkomsten geven. Je kunt bijvoorbeeld zien hoe een SHARP en een PC aan elkaar gekoppeld zijn of iemand een eigengemaakte scanner op de SHARP laat werken!!
Ik kan U zeker aanraden een keer op bezoek te gaan bij deze club. Mocht U dan genoeg hebben van al die SHARPS dan gaat U toch gewoon een wandelingetje maken door het leuke toeristenplaatsje Edam of het vlakbij gelegen Volendam.

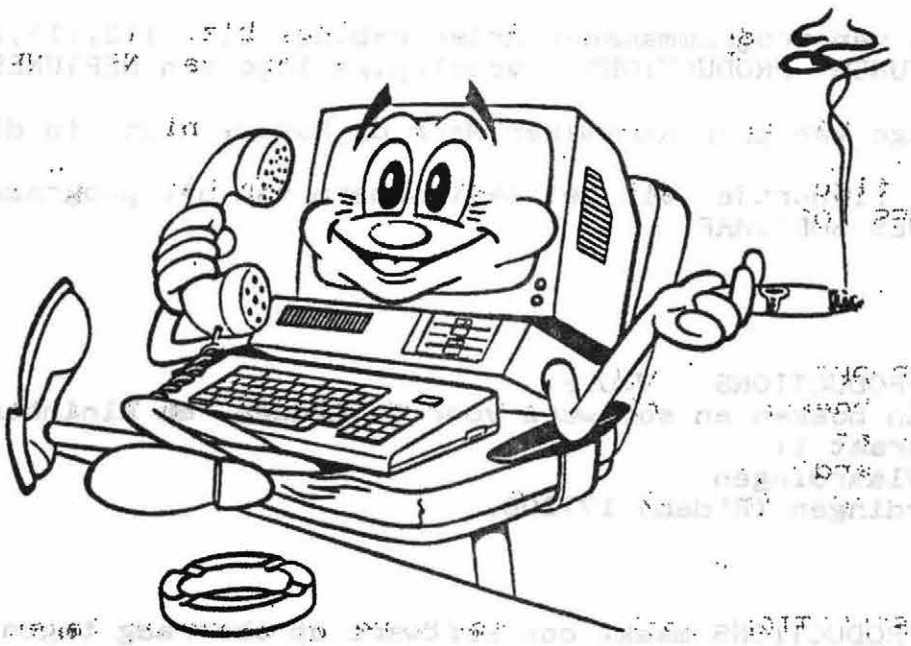
Voor zover wat betreft de aanvullende informatie. Mij rest alleen nog U veel plezier te wensen met het boek en hopelijk kunt U er nog heel veel van leren.
Met vriendelijke groeten.

Hoogachtend,



A.J. Habing, schrijver van het boek "De SHARP MZ-800 ten voeten uit".

De SHARP MZ-800
ten voeten uit



Eerste druk
Oplage 200 stuks

Geschreven door
Arjan Habing ism Mark de Rover

Vormgeving
Arjan Habing

Tekeningen op omslag:

BABY, logo van programmamaker Arjan Habing, blz. 112,113,114.
LOGO NEPTUNES PRODUCTIONS, voorlopige logo van NEPTUNES PRODUCTIONS, blz. 112.
WIZARD, logo van programmamaker Mark de Rover, niet in dit boek gepubliceerd.
VLIEGTUIG, figuurtje uit het beginscherm van het programma PICTURESHOW van NEPTUNES SOFTWARE.

NEPTUNES PRODUCTIONS V.O.F.
verkoop van boeken en software voor de midden- en kleinhandel
Geraniumstraat 11
3135 XE Vlaardingen
KvK Vlaardingen (R'dam) 177306.

NEPTUNES PRODUCTIONS maakt ook software op aanvraag tegen lage prijzen.
Bel of schrijf voor meer informatie.

INLEIDING.

Veel mensen zullen er waarschijnlijk op gewacht hebben; eindelijk is het er dan, een compleet boek waarin bijna alles over en voor Uw SHARP MZ-800 staat. Een boek waardoor U in staat zult zijn om zelf (betere) programma's te maken, zowel in BASIC als machinetaal. Tevens zult U door dit boek beter leren begrijpen hoe Uw SHARP MZ-800 in elkaar zit. Het boek is in eerste instantie bedoeld voor de standaard configuratie van een SHARP MZ-800 met uitsluitend een cassetterecorder. Natuurlijk zal de extra randapparatuur (QD, FD, RAM-kaart, printer/plotter en HARD-DISK) ook aan de orde komen. Tevens zal voor bepaalde dingen uit dit boek de VIDEO-RAM uitbreiding nodig zijn. Er wordt in dit boek uitgegaan van de standaard cassette-BASIC (1Z016 - BASIC 800 en 1Z013 - BASIC 700) en de standaard QD-BASIC (5Z-009).
Voor het tot stand brengen van dit boek zijn heel wat voorbereidingen nodig geweest. Allereerst moest een onderzoek gehouden worden naar de afzetmogelijkheden en moest gekeken worden waar de mensen belangstelling voor hebben. Daarna hebben wij nog verschillende personen moeten benaderen om alle informatie bij elkaar te krijgen. Daarnaast moest nog behoorlijk wat geregeld worden, maar dat laten we verder achterwege.
Zonder hulp van bepaalde personen en clubs zou het moeilijker geweest zijn om dit boek op de markt te brengen, daarom willen wij ze in deze inleiding daar nog eens speciaal voor bedanken. Hier volgen de namen:

SHARP Computer Club Elst en met name
- -de gehele redactie.
- -het bestuur.
- -en de vaste medewerkers.

De MZ gebruikersgroep van de HCC en met name
- -Dhr C. Gans.
- -Dhr H. Niessen.

SHARP Computer Club Edam.

Een speciaal dankwoord gaat uit naar Han Bleeker, voorzitter en tevens hoofdredacteur van SHARP Computer Club Elst. Hij bedacht voor ons de naam van dit boek.

Na deze korte dankbetuiging nog wat meer informatie over dit boek. Het boek is gemaakt voor iedereen, van beginner tot vergevorderde programmeur. Dat was alleen mogelijk door bepaalde informatie bij de beste programmeurs van ons land weg te halen. Sommige dingen die U in dit boek vindt zijn tot nu toe nog maar bij 1 of 2 mensen bekend. In dit boek zult U dus menigmaal een primeur tegenkomen.
Ook al is het boek behoorlijk dik en staat er een heleboel in; het is onmogelijk om alles wat er voor en over de SHARP MZ-800 bekend is in dit boek te zetten, vandaar dat er een redelijke kans bestaat dat er aanvullingen op dit boek zullen komen. Die aanvullingen zullen dan ongeveer 30 bladzijden tellen. Of er aanvullingen zullen komen, hangt af van de belangstelling voor dit boek en hoe het zich verder zal ontwikkelen binnen ons wereldje. Het is in ieder geval zeker dat wij software (zowel BASIC-800 als machinetaal programma's) op de markt

zullen brengen. Bij dit boek is waarschijnlijk een papiertje gevoegd met daarop de programma's die wij verwachten (binnenkort) op de markt te kunnen brengen.
Wij zijn momenteel ook bezig met het ontwikkelen van een eigen BASIC, een GAME-BASIC. In deze BASIC zullen een heleboel instructies ontbreken die niet nodig zijn bij het maken van spelprogramma's en er zullen een aantal nieuwe instructies bijkomen die de kwaliteit van de programma's alleen maar ten goede zullen komen. Door deze opzet zal het mogelijk zijn om meer geheugen te gebruiken, nog iets waardoor de kwaliteit van de programma's beter kan worden.
Dit boek is gemaakt met de tekstverwerker WordPerfect versie 4.2 op een OLIVETTI M240 computer en uitgeprint op een OLIVETTI DM-100 printer.
Verder rest ons alleen nog ter afsluiting van deze inleiding, die tevens als voorwoord bedoeld was, U te bedanken voor Uw belangstelling voor dit boek en hopelijk zal ons boek U verder helpen in de wereld van de SHARP MZ-800. Met vriendelijke groeten,

Arjan Habing
en
Mark de Rover.

INHOUDSOPGAVE.

HOOFDSTUK 0: WETENSWAARDIGHEDEN. blz. 7

0.1: Het begin. blz. 7
0.2: Verder met Uw SHARP. blz. 11

HOOFDSTUK 1: BASIC 800 (1Z016 / 5Z-009). blz. 14

1.1: Optimaal gebruik van BASIC-800. blz. 14
1.2: 'Extra' BASIC-instructies. blz. 19
1.3: Extra BASIC-instructies. blz. 21
1.4: Handige BASIC-(sub)routines. blz. 24

HOOFDSTUK 2: MACHINETAAL OP DE SHARP MZ-800. blz. 29

2.1: Nuttige en leuke machinetaal(sub)routines. blz. 29
2.2: Machinetaal binnen BASIC-800. blz. 38
2.3: Calls in de ROM-monitor. blz. 43

HOOFDSTUK 3: DE VIDEO-RAM. blz. 51

3.1: Scrollen in de VIDEO-RAM. blz. 51
3.2: Andere leuke geintjes in de VIDEO-RAM. blz. 61
3.3: VIDEO-RAM aansturing via de ROM-monitor. blz. 66

HOOFDSTUK 4: POKES, PEEKS, TRUCS, TIPS en allerlei andere fratsen. blz. 69

4.1: Pokes en Peeks. blz. 69
4.2: IN's en OUT's. blz. 75
4.3: Trucs en Tips. blz. 78
4.4: CALLS in BASIC. blz. 81

HOOFDSTUK 5: MUZIKALE TOEPASSINGEN. blz. 83

5.1: De poort $F2. blz. 83
5.2: BASIC-instructies voor geluid en muziek. blz. 89
5.3: Muziek via machinetaal. blz. 93
5.4: Enkele muziekstukken. blz. 97

HOOFDSTUK 6: GRAFISCHE TOEPASSINGEN. blz.103

6.1: Leuke wiskundige figuren. blz.103
6.2: Leuke toepassingen met kleuren. blz.108
6.3: Leuke figuurtjes. blz.112
6.4: Zakelijke toepassingen. blz.117
6.5: Animaties. blz.122

HOOFDSTUK 7: Andere BASICS. blz.125

7.1: NEPTUNES-BASIC. blz.126
7.2: BASIC-700. blz.128
7.3: UNI-GG-BASIC. blz.137

HOOFDSTUK 8: HARDWARE. blz.142

8.1: Uitleg over de verschillende stukken hardware. blz.142
8.2: Enkele programma's voor de hardware. blz.148
8.3: Kleine hardware-aanpassingen. blz.159

HOOFDSTUK 9: ASSEMBLEREN. blz.163

9.1: Assemblerlisting van programma's in dit boek. blz.163
9.2: Uitleg van enkele assemblerinstructies en specifieke SHARP zaken. blz.190

HOOFDSTUK 10: PROGRAMMA'S. blz.203

APPENDICES. blz.225

APPENDIX A: Z80-A mnemonics. blz.225
APPENDIX B: Woordenboek. blz.230
APPENDIX C: BASIC-INSTRUCTIES EN AFKORTINGEN. blz.234
APPENDIX D: SHARP-CLUBS EN ADRESSEN. blz.237

MEER OVER NEPTUNES PRODUCTIONS. blz.242

SLOTWOORD. blz.243

HOOFDSTUK 0: WETENSWAARDIGHEDEN.

0.1: Het begin.

Alle begin is moeilijk, vandaar dit hoofdstuk dat speciaal bedoeld is voor de mensen die niet goed overweg kunnen met het handboek en nog niet helemaal weten hoe ze hun SHARP-computer moeten bedienen. De meesten van U zullen natuurlijk zeggen dat alles om fatsoenlijk met de SHARP MZ-800 aan de gang te kunnen in het handboek staat. Het is echter gebleken dat er nog redelijk veel mensen zijn die weinig tot niets van het handboek begrijpen.
Dit hoofdstuk zal U zoveel over Uw SHARP vertellen dat U in staat zult zijn de rest van dit boek toe te passen. Het zal waarschijnlijk niet zo zijn dat U alles in dit boek zult begrijpen, dat komt omdat het boek voor iedereen bedoeld is; zelfs de beste programmeurs moeten er nog van kunnen leren.
Laten we dan nu eens beginnen. Normaal wordt er bij het begin begonnen, maar dat zullen wij niet doen. Hoe U Uw SHARP uit moet pakken en aan moet sluiten zal iedereen wel duidelijk zijn, maar na het aansluiten komen de eerste problemen al om de hoek kijken. Met welke randapparatuur wilt U bijvoorbeeld werken, in welke mode wilt U werken en waarvoor wilt U de computer gaan gebruiken?
Allereerst iets over de randapparatuur. U kunt heel veel randapparatuur op Uw SHARP aansluiten, de belangrijkste daarvan zijn wel: een cassetterecorder, een QUICK-DISK, een TV of MONITOR, een JOYSTICK, een PRINTER of PLOTTER, 1 of meer diskdrives en een harde schijf. Hieronder zal deze apparatuur nader toegelicht worden.

- De cassetterecorder: Bij de meesten van U zat de cassetterecorder bij aankoop in de computer ingebouwd. Bij enkelen kan het geweest zijn dat een QUICK-DISK al ingebouwd was en de cassetterecorder achter op de QUICK-DISK aangesloten moest worden. In feite maakt het niets uit of de cassetterecorder of een QUICK-DISK ingebouwd is, het is gewoon een kwestie van voorkeur.
  In het handboek staat duidelijk beschreven hoe U eventueel de cassetterecorder en de QUICK-DISK om kunt wisselen.
  De cassetterecorder wordt gebruikt voor het bewaren en 'terughalen van programma's en/of gegevens.
  De cassetterecorder kan door bepaalde instructies aangestuurd of 'aangesproken' worden. Welke instructies dat precies zijn zal later in dit hoofdstuk besproken worden.
  Een cassetterecorder maakt gebruik van cassettes om de informatie op te slaan. Het maakt weinig uit welke cassettes gebruikt worden, alleen het gebruik van C90-cassettes en te goedkope cassettes (F0,95 ps) wordt afgeraden. Het is natuurlijk ook niet nodig om hele dure cassettes te gebruiken, dat is gewoon zonde van Uw geld.

in de cassetterecorder zitten koppen. Wanneer deze koppen verkeerd zijn afgesteld, kan het zijn dat bepaalde programma's niet geladen kunnen worden. Dan moeten de koppen opnieuw afgesteld worden. Dat kunt U doen door met een kleine schroevendraaier of iets anders aan het schroefje boven de PLAY-knop te draaien en dan de programma's opnieuw proberen te laden.

De QUICK-DISK: Naast de cassetterecorder is de QUICK-DISK een veelgebruikt medium voor het opslaan of opvragen van programma's en/of gegevens. Kon de cassetterecorder nog gewoon op de computer aangesloten worden, voor de QUICK-DISK is een aparte interface nodig om hem te kunnen gebruiken.
Een interface zorgt ervoor dat bepaalde randapparatuur, die voor verschillende computers gebruikt kan worden, de specifieke gegevens krijgt om zich volledig geschikt te maken voor gebruik op een bepaalde computer. Eigenlijk zorgt een interface er dus voor dat randapparatuur volledig wordt of aangepast wordt, zodat het op een bepaalde computer gebruikt kan worden. Het is dan ook wel begrijpelijk dat voor iedere computer van een ander type een andere interface gebruikt moet worden.
Na even de term "interface" behandeld te hebben gaan we weer terug naar de QUICK-DISK.
Voor de QUICK-DISK worden speciale schijfjes gebruikt, de zogenaamde 2.8-inch schijfjes. Deze schijfjes hebben een opslagcapaciteit van 128K en zijn tamelijk duur in aanschaf, vandaar dat het veelvoudig gebruik van deze schijfjes een dure aangelegenheid wordt.
De aansturingscommando's voor de QD zullen eveneens later in dit hoofdstuk besproken worden.

De TV/MONITOR: Rechts (van voor de computer gezien) van de computer zitten achter vier gaten. Twee daarvan zijn bedoeld voor een eventuele tweede cassetterecorder, daaronder staan de woorden READ en WRITE. Op deze twee gaten wordt niet verder ingegaan, maar wel op de andere twee.
Bij de ene staat RF vermeld en bij de andere VIDEO. Tussen de twee gaten bevindt zich ook nog een schakelaar die op B/W of op COLOR gezet kan worden. B/W staat voor BLACK and WHITE oftewel ZWART en WIT.
Wanneer U een "tweekleurige" monitor (monochrome monitor) of een zwart/wit TV gebruikt, moet U de schakelaar op B/W zetten. Wanneer U een kleuren-TV of kleurenbeeldscherm gebruikt, moet de schakelaar op COLOR (kleur) gezet worden.
Naast het gat waar COLOR bij staat bevindt zich ook nog een ingang waar RGB bij staat. Deze ingang is bedoeld voor een RGB-monitor. RGB is een bepaalde standaard, evenals er een CENTRONICS- en een PARALLEL-poort is. Voor de rest is dit niet van belang, want de meeste monitoren voor HOME-computers gebruiken de RGB-standaard.

Dan is er nog een klein zwart schroefje dat zich ook tussen de twee gaten bevindt. Dit is een knopje dat hetzelfde doel heeft als het knopje op Uw TV om een bepaalde zender op te zoeken en scherp te stellen. Mocht U geen goed beeld krijgen bij het scherpstellen van het scherm dan kan dit schroefje misschien van nut zijn. Overigens kan elke TV met een VIDEO-aansluiting op Uw SHARP aangesloten worden.
De poort waar RF bij staat is voor de aansluiting op een TV en de aansluiting waar VIDEO bij staat is voor de aansluiting op een MONITOR.

De JOYSTICK: Op Uw SHARP zitten twee joystickpoorten. Hierin kunnen alle joysticks die ATARI-compatible zijn. Welke joysticks zijn nu ATARI-compatible zult U dan vragen. Meestal staat dat vermeld op de verpakking. Staat het niet vermeld, vraagt U het dan aan de verkoper.
Boven de poorten staat vermeld wat poort 1 is en wat poort 2.
Ook de instructie voor het uitlezen van een joystick zal later in dit hoofdstuk volgen.

De PRINTER/PLOTTER: Eigenlijk zijn dit twee geheel verschillende dingen, maar vanwege hun grote overeenkomst in functie worden ze gelijk besproken. Ze zijn namelijk beide bedoeld voor het uitdraaien van een tekst of een (grafische) voorstelling. De manier waarop ze dit doen is weer geheel verschillend, maar daar zal later meer over verteld worden. Allereerst zal de aansluiting besproken worden.
De speciale SHARP-plotter kan zo op de SHARP aangesloten worden. Onder de aansluiting voor de netspanning zit de aansluiting voor de plotter. Daar staat bij: POWER for MZ-1P16.
De aansluiting voor de printer is beschermd door een ijzeren plaatje. Boven de aansluiting staat het woord PRINTER, dat kan dus bijna niet missen. Tussen de printer en de computer komt een kabeltje dat zorgt voor de verbinding.
Overigens past lang niet iedere printer op Uw SHARP. Koop dus niet zomaar een printer, maar controleer eerst of hij op Uw SHARP aangesloten en gebruikt kan worden. De verschillende clubs die er zijn kunnen U daar eventueel bij helpen.
Omdat de printer/plotter toch niet zo heel erg vaak gebruikt wordt, zal er haast niet verder op de besturingscommando's ingegaan worden. Wel zal er wat aanvullende informatie gegeven worden.
Dan nog even iets over de manier van "uitdraaien". Een printer heeft een zogenaamde printerkop met daarop allemaal naaldjes. Doordat de naaldjes willekeurig bewogen kunnen worden, kunnen er allerlei tekens gevormd worden. Als een naaldje tegen het inktlint aan komt, komt er een zwart puntje op papier. Alle puntjes bij elkaar vormen

dan een bepaalde "uitdraai".
Een plotter tekent alles. Bij een plotter gaat een pennetje over het papier heen. Hierdoor ontstaat een mooier resultaat dan op de printer. Er zijn echter twee nadelen. Ten eerste is de plotter veel en veel langzamer en ten tweede is de SHARP-plotter er alleen in klein formaat.

De DISKDRIVE: Meestal wordt er niet 1, maar worden er twee diskdrives gebruikt. Dat werkt namelijk veel sneller dan met 1 diskdrive.
De diskdrive is voornamelijk bedoeld om te gebruiken bij CPM. Wat CPM precies inhoudt en wat U er precies mee kunt doen, wordt hier niet besproken. Met CPM kunt U op een professionelere wijze gebruik maken van Uw SHARP.
Het gebruik van diskdrives is stukken goedkoper dan het gebruik van een QUICK-DISK. Dit komt omdat de schijfjes veel goedkoper zijn en ook een veel grotere opslagcapaciteit hebben. Het enige nadeel is dat de diskdrives in aanschaf een stuk duurder zijn.
Nog enkele voordelen van een diskdrive:
-Het werkt sneller.
-Langere programma's kunnen gebruikt worden.
-De schijven zijn overal te koop.
-De programma's zijn gemakkelijk te wissen.
-De programma's zijn gemakkelijk te veranderen.
-Enz.
Ook al zijn er veel mensen die in het bezit zijn van 1 of meer diskdrives, toch zal er weinig over diskdrives in dit boek staan. Dit is eenvoudigweg gedaan omdat er zo ontzettend veel utilities en andere dingen voor de diskdrive zijn dat er een heel boek mee te vullen is en daar komt ook nog eens bij dat de mensen die 1 of meer diskdrives hebben vaak wel weten hoe ze er mee om moeten gaan; vandaar.

De HARDDISK: U zult natuurlijk denken: wat raar, daar heb ik nog niet van gehoord. Dat kan kloppen, want er is maar 1 persoon in heel Nederland die een harde schijf op zijn SHARP heeft laten draaien. Verderop in dit boek kunt U daar veel meer over lezen en ook over andere ontwikkelingen die op hardwaregebied voor de SHARP staan te wachten.

Dit waren de belangrijkste randapparaten die U op Uw SHARP aan kunt sluiten. Er zijn natuurlijk nog veel meer dingen die er op aangesloten kunnen worden, zoals een RAM-kaart, allerlei electronica-projecten (zoals ROBOT, TREIN, enz.) en ga zo maar door.
Het zal niet altijd even gemakkelijk zijn om iets op Uw SHARP aan te sluiten. Voor sommige dingen moet de SHARP helemaal omgebouwd worden of moet er een aparte kaart komen.
Het is jammer dat zich maar zo weinig mensen op dit gebied bezighouden, maar we kunnen U nog wel een aantal hele leuke dingen beloven die U in de toekomst op Uw SHARP aan zult kunnen sluiten.

0.2: Verder met Uw SHARP.

Nu zal behandeld worden welke handelingen U uit moet voeren om gebruik te kunnen maken van de randapparatuur en andere zaken die voor U van belang zijn om met de computer aan de gang te kunnen en de rest van het boek toe te kunnen passen.
Allereerst is het van belang om te weten hoe U de dipswitsches (de vier kleine zwarte knopjes achter op Uw SHARP) het beste in kunt stellen. Daarvoor zult U eerst de computer om moeten draaien zodat U tegen de achterkant aan kijkt.
Het meest linker knopje komt omhoog te staan. Dat betekent wel dat de computer in de 700-mode komt te staan, maar dat betekent ook dat alle programma's geladen kunnen worden en ook lopen. Als BASIC-800 geladen wordt, schakelt de computer uit zichzelf over naar de 800-mode.
Wanneer U een printer gebruikt moeten het tweede en derde knopje allebei in dezelfde stand komen te staan. Heeft U een MZ-printer dan moeten beide knopjes omhoog staan. Heeft U een Centronics-printer dan moeten beide knopjes naar beneden. Mocht de printer het niet doen dan moet 1 van de 2 knopjes omhoog gezet worden.
Het vierde knopje heeft geen nut, het maakt dus niet uit hoe die staat.
Nu we de computer toch omgekeerd hebben kunnen we gelijk twee andere knopjes achter op de computer uitleggen en dat zijn de volumeknop en de resetknop die vlak naast elkaar zitten. Bij beide knopjes staat waarvoor ze bedoeld zijn.
Wanneer het geluid uit Uw SHARP U te hard of te zacht klinkt kunt U met de volumeknop het juiste volume regelen.
Wanneer Uw BASIC eens een keer vastloopt of U wilt een ander machinetaalprogramma laden kunt U gebruik maken van de resetknop om het beginscherm van de SHARP te krijgen. Dan hoeft U tenminste niet elke keer op de AAN/UIT-knop te drukken, waardoor Uw computer sneller kapot zal gaan.
Wanneer U Uw computer aanzet en U heeft een QD in de QUICK-DISK-drive dan zal de QD-drive automatisch gaan draaien en als er dan een OBJ-file als eerste programma op de QD staat zal dat geladen worden. Wanneer er meer OBJ-files op die QD staat, dan is dat natuurlijk heel erg lastig.
Hoe moet U die andere programma's nu laden? Dat gaat als volgt:
-U stopt de QD in de QD-drive, maar doet de QD-drive nog NIET dicht.
-U zet de computer aan en doet daarna de QD-drive dicht.
-U drukt op M (MONITOR).
-U typt QD <cr> in.
-U ziet nu alle namen van de programma's op het scherm.
-U typt QL <cr> in.
-U ziet nu het woord FILENAME? verschijnen.
-U typt letterlijk de naam in van het programma dat U wilt laden en drukt daarna op <cr>.
-Heeft U de naam goed ingetypt dan wordt vervolgens het door U gekozen programma geladen.
-Waarschuwing: Voor het programma dat U wilt laden moet wel het woord OBJ staan anders laadt de computer het programma wel, maar loopt na het laden meestal compleet vast.

Dit was in het kort een bespreking van hoe U een OBJ-file van QD kunt laden. Er zijn nog veel meer instructies die in de MONITOR uit te voeren zijn, maar die zullen besproken worden wanneer U ze pas echt nodig heeft.

Laten we nu de belangrijkste BASIC-instructies bespreken die U nodig kunt hebben bij het aansturen van randapparatuur.

Het invoeren van een programma en het vervolg.

Hoe U een BASIC-programma in moet voeren zal wel duidelijk zijn. Eerst de regel intypen en daarna op <cr> drukken.
U kunt naderhand of tussendoor bekijken wat U in heeft getypt met behulp van de instructie LIST.
U kunt alles bekijken door simpel LIST <cr> in te typen. Past de hele listing niet op 1 scherm dan zal gewoon de hele listing getoond worden. U kunt het LISTEN stoppen door op SHIFT/BREAK te drukken of tijdelijk stoppen door op de spatiebalk te drukken (verder laten gaan door op een willekeurige toets te drukken) of door op de BREAK-toets te drukken en deze ingedrukt te houden. Wanneer U de BREAK-toets loslaat gaat het LISTEN weer verder.
Wanneer U klaar bent met intypen of wanneer U gewoon tussendoor het programma op QD of cassette wilt zetten kunt U dat doen met de instructie SAVE.
Allereerst moet U weten waar het programma op komt te staan, op QD of op cassette. Heeft U alleen een cassetterecorder dan heeft U met het volgende niets te maken!
U kunt het programma bewaren door het volgende in te typen:
SAVE"CMT:xxxxxx" <cr> - voor cassette
of
SAVE"QD:xxxxxx" <cr> - voor QD
Hierbij kan xxxxxx vervangen worden door iedere willekeurige naam zolang het maar niet langer is dan 16 tekens.
U kunt ook gebruik maken van de instructie DEFAULT.
Wanneer U intypt DEFAULT"QD:" <cr> dan wordt alles automatisch alleen naar QD geschreven of van QD geladen.
Wanneer U intypt DEFAULT"CMT:" <cr> dan wordt alles automatisch alleen naar cassette geschreven of van cassette geladen.
Wanneer U DEFAULT heeft gebruikt of alleen een cassette heeft hoeft U alleen SAVE"xxxxxx" <cr> in te typen.
Wanneer U het programma dan weer van cassette of QD wilt halen kunt U dat doen met de instructie LOAD. Dit gaat precies hetzelfde als bij SAVE. Overal waar SAVE staat, moet U dan LOAD invullen.
Heeft U het complete programma ingetypt dan kunt U het programma starten met RUN <cr>, maar dat zal iedereen wel weten.
Wilt U een programma helemaal uit het geheugen halen dan typt U NEW <cr> in. Wanneer U daarna LIST <cr> intypt, zult U alleen de melding Ready (In sommige gevallen Klaar.) krijgen.
Voor zover genoeg over het SAVEN, LADEN, RUNnen en LISTen. Voor het schrijven van Uw eigen programma's zal het ook nodig zijn dat U weet hoe U in een programma kunt kijken welke toetsen er ingedrukt zijn en hoe U een JOYSTICK uit kunt lezen en hoe U een printer aan kunt sturen en ga zo maar door.
Allereerst iets over het programmeren zelf. Met behulp van deze uitleg zult U nog niet direct een programma kunnen schrijven, maar U zou wel in bestaande programma's veranderingen aan kunnen brengen zodat U bijvoorbeeld iets uit kunt printen of een spelletje dat alleen met het toetsenbord gespeeld kan worden zo veranderen dat het ook met de joystick gespeeld kan worden. Dat is vaak gemakkelijker dan U in eerste instantie zult denken.

STICK(n).

Wanneer de pijltjestoetsen of een joystick worden gebruikt in een BASIC-spel, komt U deze instructie vaak tegen. Wanneer er staat:
-STICK(0) wordt er gebruik gemaakt van het toetsenbord.
-STICK(1) wordt er gebruik gemaakt van joystickpoort 1.
-STICK(2) wordt er gebruik gemaakt van joystickpoort 2.
Soms gaat dat gepaard met de instructie STRIG waar dan meestal hetzelfde getal tussen haakjes achter staat.
Wilt U nu het spel zo veranderen dat het bijvoorbeeld met de joystick ipv met het toetsenbord gespeeld kan worden dan verandert U simpelweg de 0 in een 1 of een 2. Andersom is het precies hetzelfde. Wanneer er STRIG bij staat, moet U dat natuurlijk ook veranderen in datzelfde getal. STRIG is overigens bedoeld voor de spatiebalk of de vuurknop op de joystick.
Hoe weet U nu of er STICK en/of STRIG in een programma staat? Dat kunt U opzoeken met de instructie SEARCH. U typt in:
SEARCH"STICK" <cr>
en automatisch worden alle regels opgezocht waar deze instructie in staat.
Met STRIG gaat het precies hetzelfde daarvoor typt U in:
SEARCH"STRIG" <cr>
Helaas is het niet altijd zo dat het toetsenbord of de joystick met de instructie STICK uitgelezen wordt. Verderop in het boek staat nog een andere mogelijkheid voor het uitlezen van de joystick. Dat zal echter voor de echte leken nog iets te moelijk zijn om te begrijpen, vandaar dat het hier nog niet besproken wordt.

GET.

Met de GET-instructie kan het toetsenbord ook uitgelezen worden. Dat heeft echter nogal wat nadelen. Het voordeel is dat niet alleen de pijltjestoetsen, maar ook bijna alle andere toetsen uitgelezen kunnen worden. Wanneer U bijvoorbeeld in een programma ziet:
10 GET A$:IF A$="" THEN 10
dan betekent dat dat er pas verder gegaan wordt met het programma wanneer er een willekeurige toets is ingedrukt.
Zo zijn er nog wat meer mogelijkheden met GET, maar dat valt ook onder de categorie te moeilijk voor leken en daar komt ook nog eens bij dat het ook uitgebreid in het handboek staat; vandaar dat er niet verder op ingegaan zal worden.

PRINTER-aansturing.

Bij het aansturen van een printer moet de printer altijd eerst met INIT ingesteld worden. Bij elke printer kan deze INIT anders zijn. Vaak is het al bekend wat er moet staan; vandaar dat er niet verder op ingegaan wordt.
Met PRINT/P kunt U daarna een tekst naar de printer sturen, bijvoorbeeld: PRINT/P"NEPTUNES."
Ook kan er een afdruk van alles wat op het scherm staat op papier gemaakt worden. Dat gaat helaas niet met de instructie HCOPY zoals veel mensen denken, daar is een speciaal programma voor nodig. Dat programma staat bekend als een HCOPY-programma. Er zijn in totaal zeker 4 HCOPY-programma's voor alle verschillende soorten printers.

HOOFDSTUK 1: BASIC-800 (1Z016 / 5Z-009).

Dit hoofdstuk gaat over de taal BASIC-800. Er zal o.a. verteld worden hoe U deze BASIC zo optimaal mogelijk kunt gebruiken en welke instructies niet in het handboek voorkomen, maar wel te gebruiken zijn. Tevens komt U in dit hoofdstuk kleine programmaatjes tegen die extra instructies aan de BASIC toe zullen voegen. Daarbij zal ook een uitleg van de nieuwe instructies gegeven worden. Hoe deze programmaatjes precies werken zal in een ander hoofdstuk worden uitgelegd. Als laatste treft U in dit hoofdstuk enkele zeer interessante BASIC-(sub)routines aan die U in Uw eigen programma's kunt gebruiken. Daarbij zal ook menigmaal een nadere uitleg staan.
Tussen de cassette-BASIC (1Z016) en de QD-BASIC (5Z-009) zit nog een redelijk verschil, daarom zal onderscheid gemaakt worden tussen deze BASICS en zal, indien nodig, bij elk programmaatje vermeld staan of het in cassette-BASIC of in QD-BASIC werkt of in allebei en eventueel de aanpassingen die nodig zijn om het in de andere BASIC te laten werken.

1.1 : Optimaal gebruik van BASIC-800.

U weet allemaal dat er een redelijk beperkt aantal BYTES vrij zijn om Uw programma's in te programmeren. Met alle informatie die U hierbij gegeven zal worden zult U optimaal gebruik kunnen maken van die BYTES die U ter beschikking staan.
Laten we U eerst eens vertellen hoe U kunt bekijken hoeveel BYTES U nog tot Uw beschikking heeft. De meesten zullen het wel weten, maar voor die paar mensen die het nog niet weten volgt hier de instructie die U in moet voeren om te bekijken hoeveel BYTES er nog over zijn:

```
PRINT SIZE <cr>
```

Hierbij betekent <cr> dat U de CR-toets in moet drukken en niet dat U letterlijk <cr> in moet typen. U zult dit vaker tegenkomen in dit boek, weh er dus maar alvast aan.
Wanneer U gaat beginnen met een programma moet U beginnen bij een bepaald regelnummer, bijv. regelnummer 10. U kunt ook beginnen bij regelnummer 10000. U zult natuurlijk denken dat het getal 10000 meer geheugenruimte in beslag zal nemen dan het getal 10 omdat er drie cijfers meer in voor komen, dat is echter NIET het geval. Beide getallen nemen evenveel geheugenruimte in beslag, namelijk 2 BYTES. U hoeft dus niet met alle geweld bij een zo laag mogelijk regelnummer te beginnen.
Daarna komen we bij het programma zelf. U wilt iets invoeren. Aan het begin van het programma komen dan meestal eerst een paar REM-regels te staan waarin bijvoorbeeld de naam en het adres van de maker van het programma staan. Meestal wordt dan niet het woord REM gebruikt, maar het teken ('). Ten eerste oogt dit veel beter en ten tweede LIJKT het minder geheugen in beslag te nemen, omdat het slechts 1 teken is en REM uit 3 letters bestaat. Het tegengestelde is echter waar. Het woord REM neemt slechts 1 BYTE in beslag en het teken (') neemt er twee in beslag. Een waarschuwing: Denk nooit dat elke letter of elk teken minimaal 1 BYTE in beslag neemt, want dat is absoluut niet waar. Alle BASIC-instructies worden namelijk in 1 code omgezet en nemen zodoende slechts 1 BYTE in beslag. Getallen daarentegen nemen veel meer geheugen in beslag, maar dat wordt later besproken.

Na de REM-regels wordt meestal eerst de waarde van een aantal variabelen vastgelegd. Daarbij is enorm veel geheugenruimte te besparen. Wat is namelijk het geval: Wanneer U een variabele een waarde geeft kunt U dat doen door bijvoorbeeld A=10 in te voeren. Wat U nu niet weet is dat het getal 10 maar liefst 6 BYTES in beslag neemt. Hieronder volgen een aantal voorbeelden hoe U dat veel beter aan kunt pakken en waardoor U veel geheugen zult besparen. Het eerste voorbeeld is altijd het voorbeeld dat het meeste aantal BYTES in beslag neemt. Wilt U het niet geloven, bekijkt U het dan eens door elke keer PRINT SIZE in te typen.

Opmerking: Het π-teken dat U af en toe tegenkomt kunt U op Uw SHARP krijgen door op SHIFT en 0 te drukken.

Voorbeeld 1:

```
10 B=0
```

kan vervangen worden door:

```
10 B=π-π
```

Voorbeeld 2:

```
10 B=10:C=10
```

kan vervangen worden door:

```
10 B=10:C=B
```

Voorbeeld 3:

```
10 A=10:B=15:C=24:D=2:E=90
```

kan vervangen worden door:

```
10 DATA 10,15,24,2,90:READ A,B,C,D,E
```

In voorbeeld drie zag U al dat de waarden van variabelen ook in DATA opgeslagen kunnen worden. Mocht het nu zo zijn dat U veel DATA heeft dan kunt U deze ook opslaan in een OBJ-file en deze hoog in het geheugen laden. Hoe dat allemaal precies in elkaar zit, is te ingewikkeld om in dit hoofdstuk uit te leggen. In het laatste deel van dit hoofdstuk staan nog wel twee programmaatjes die laten zien hoe U bepaalde gegevens kunt LADEN en SAVEN, maar dat heeft weinig te maken met wat hierboven gezegd is.
Nieuwsgierige mensen die toch willen weten hoe het laden en saven van zo'n OBJ-file in elkaar zit en in het bezit zijn van het programma PICTURESHOW moeten dat programma maar eens goed doorkijken, want daarin wordt gebruik gemaakt van deze methode.
Het programma PICTURESHOW is overigens bij ons te bestellen of zal binnenkort te bestellen zijn. U kunt het bestellen door ons te bellen (telefoonummers in APPENDIX C).

Dan zijn we aanbeland bij een aantal dingen die eigenlijk overal in het programma voor kunnen komen. Allereerst wordt het vaak overbodige gebruik van GOTO besproken. Twee voorbeelden:

Voorbeeld 1:

```
100 IFA=10THENGOTO10
```

kan vervangen worden door:

```
100 IFA=10THEN10
```

Voorbeeld 2:

```
100 IFB=10THENGOTO100ELSEGOTO100
```

kan vervangen worden door:

```
100 IFB=10THEN100ELSE100
```

Ook hierbij is het eerste voorbeeld het voorbeeld dat het meeste aantal BYTES in beslag neemt.

Een veelvoorkomende geheugenverslindende uitleesroutine voor het toetsenbord kan ook vervangen worden door een veel korter en tevens minder geheugenverslindend regeltje. Een voorbeeld:

```
10 A=0
20 CURSOR0,0:PRINTA:S=STICK(0):IFS=3 AND
A<100THENA=A+1
30 IFS=7 ANDA>0THENA=A-1
40 GOTO20
```

kan vervangen worden door:

```
10 A=0
20 CURSOR0,0:PRINTA:S=STICK(0):A=A+(1 AN
DS=3 ANDA<100)-(1 ANDS=7 ANDA>0):GOTO20
```

Natuurlijk kunt U A=0 nog vervangen door A=π-π en CURSOR0,0 vervangen door CURSOR π-π,π-π , maar dat is al eerder besproken.

Misschien is het U opgevallen, maar in het bovenstaande voorbeeld is nog iets geslopen dat geheugenbesparend is. Mocht U het nog niet gezien hebben, dan volgt hier de oplossing:

```
10 IFSTICK(0)=3THENEND
20 IFSTICK(0)=5THENEND
```

Neemt meer geheugen in beslag dan:

```
10 S=STICK(0):IFS=3THENEND
20 IFS=5THENEND
```

Nog enkele geheugenbesparende oplossingen:

Voorbeeld 1:

```
10 S=0
20 S=S+1:IFS=50THENCLS:END
30 GOTO20
```

neemt meer geheugen in beslag dan:

```
10 S=0
20 S=S+1:IFS<50THEN20ELSECLS:END
```

Voorbeeld 2:

Nog een voorbeeld met variabelen.

```
10 A=10:B=25:C=15
```

neemt meer geheugen in beslag dan:

```
10 A=10:B=25:C=B-A
```

Zo kunt U ook vermenigvuldigingen, delingen, etc. uitvoeren. U kunt maximaal drie variabelen tegelijk gebruiken, wil het nog geheugenbesparend blijven.

Als laatste in deze serie komt het begrip recursief programmeren aan de orde. Aan de hand van recursief programmeren zult U leren dat bij elke GOSUB ook een RETURN moet volgen, anders raakt het hele geheugen op den duur vol. Wat houdt recursief programmeren nu precies in?
Eerst wordt iets uitgevoerd, waarna het in omgekeerde volgorde weer terug gaat naar het startpunt. Het is te vergelijken met een auto die eerst 100 meter vooruit rijdt en daarna diezelfde 100 meter weer achteruit rijdt, zodat hij weer bij het beginpunt uitkomt.
Hieronder volgen enkele voorbeelden van hoe het wel en hoe het niet moet met GOSUB.

Het goede voorbeeld:

```
10 GOSUB20:GOTO10
20 CURSOR0,0:PRINTSIZE:RETURN
```

Het verkeerde voorbeeld:

```
10 GOSUB20
20 CURSOR0,0:PRINTSIZE:GOTO10
```

Recursief programmeren:

Dit voorbeeld laat dmv recursief programmeren zien dat het noodzakelijk is om evenveel RETURN's als GOSUB's te geven. Als kleinigheidje tussendoor: De pythagorasboom is ook opgebouwd mbv recursief programmeren.

```
10 CLS:A=1
20 GOSUB30
30 CURSOR0,0:PRINT"AANTAL BYTES VRIJ:";S
IZE:WAIT50:A=A+1:IFA<51THEN20
40 IFA=51THENWAIT2000
50 IFA<101THENRETURN
```

In dit programmaatje worden evenveel RETURN's als GOSUB's gegeven. U ziet het geheugen eerst afnemen, omdat de GOSUB's niet beantwoord worden met een RETURN. Later worden alle RETURN's afgewerkt en ziet U het geheugen weer oplopen.

Na enkele geheugenbesparende handelingen besproken te hebben volgen nu een aantal handelingen die het intypen van een programma vereenvoudigen, waardoor het ook sneller gaat.

Zoals al eerder gezegd, begint U bij elk programma met een bepaald regelnummer. Om nu niet elke keer eerst een regelnummer in te hoeven typen is een instructie aanwezig, dat is de instructie AUTO. Met de instructie AUTO krijgt U automatisch regelnummers op het scherm, die hoeft U dan niet meer in te typen. Er is één bezwaar bij het gebruik van AUTO. De regelnummers lopen gelijkmatig op. Enkele voorbeelden van AUTO:

```
AUTO <cr>      10 \
               20 ] De regelnummers komen met sprongen van 10.
               30 /
               enz.

AUTO1000,100 <cr>    1000 \ De regelnummers komen met sprongen van 100,
                     1100 ] beginnend bij regelnummer 1000.
                     1200 /
                     enz.

AUTO50,1 <cr>     50 \ De regelnummers komen nu met sprongen van 1,
                  51 ] beginnend bij regelnummer 50.
                  52 /
```

Hopelijk is de werking van AUTO zo een beetje duidelijk geworden. Kijk anders nog eens in het handboek.

Enkele tips om de invoersnelheid te verhogen:

-Ipv PRINT kunt U ook ? intypen. Wanneer U dan de listing opvraagt staat er gewoon PRINT.
-U kunt ook bijna iedere instructie afkorten. In APPENDIX C vindt U alle BASIC-instructies en hun afkortingen. Wanneer U de afkorting intypt en daarna de listing opvraagt krijgt U de hele instructie te zien.
-U kunt ook een bepaalde veelgebruikte instructie onder een functie-toets vastleggen en dan door 1 toets in te drukken de instructie op het scherm toveren. Probeer maar eens uit:
DEFKEY(3)="DATA" <cr>
en druk daarna maar eens op de functie-toets F3.

1.2: 'Extra' BASIC-instructies.

Extra is tussen aanhalingstekens geplaatst omdat het eigenlijk geen extra instructies zijn, maar gewoon instructies die niet in het handboek vermeld staan. Die instructies zullen hieronder nader uitgelegd worden.

BEEP.

Bij het invoeren van deze instructie zal een korte toon van 440 Hz te horen zijn.

EDIT.

Bij het invoeren van deze instructie zal de laatste regel waar U handelingen in uitgevoerd heeft, op het scherm verschijnen. Deze instructie is bedoeld om veranderingen uit te voeren in die regel of die regel nog eens te bekijken.
Het is ook mogelijk om een willekeurige regel op te roepen met EDIT, bijv. EDIT 10. Het regelnummer volgt dus na de EDIT-instructie. EDIT heeft ongeveer dezelfde functie als LIST (gevolgd door één regelnummer, dus niet meer dan één programmaregel) met als grootste verschil dat de cursor bij EDIT meteen na het regelnummer komt te staan.

FRAC.

Met deze instructie kunt U het deel van een getal dat na de komma volgt, naar voren halen. Probeer het volgende maar eens uit:

```
PRINT π
```
en daarna
```
PRINT FRAC(π)
```

Ziet U het verschil? Hopelijk zal met dit voorbeeld de werking van FRAC duidelijk zijn.

HEX$(x).

Met deze instructie kunt U een decimaal getal in een hexadecimaal getal veranderen. Probeer maar eens uit:

```
PRINT HEX$(15)
```

Het omgekeerde (van HEX. naar DEC.) wordt bereikt door gewoon PRINT voor een hexadecimaal getal te zetten. Probeer maar eens uit:

```
PRINT $A0
```

MOD.

Met MOD kunt U de rest van een deling opvragen. De uitkomst is wel in gehele getallen! Een voorbeeld:

```
PRINT (26 MOD 4)
```
geeft als uitkomst: 2 (26:4=6 rest 2)

SPACE$.

Met deze instructie is het mogelijk om een bepaald aantal spaties achter elkaar neer te zetten tot maximaal 255 spaties. Enkele voorbeelden:

```
PRINT SPACE$(20);"SHARP"
```

Zet eerst twintig spaties en daarna het woord "SHARP"

```
PRINT SPACE$(80);"SHARP"
```

Zet eerst tachtig spaties en daarna het woord "SHARP"

enz.
Hiermee zal de bedoeling van SPACE$ wel duidelijk zijn. Het lijkt veel op TAB, maar bij TAB wordt niet alles van het scherm ge'veegd' wat binnen het bedoelde gebied staat en dat is bij SPACE$ wel het geval.

LOAD ALL.

Met deze instructie worden alle programma's van QD op de RAM-kaart geladen mits er een RAM-kaart in Uw SHARP aanwezig is, anders volgt er een Unformat error. Eenmaal op de RAM-kaart aanwezig, kunnen de programma's eventueel verandert worden met de instructies RENAME of DELETE.
Aangezien er maar weinig mensen zijn die een RAM-kaart bij hun SHARP gebruiken, zal dit een van de weinige keren zijn dat er ingegaan wordt op de RAM-kaart.

SAVE ALL.

Alle programma's die op de RAM-kaart staan worden nu naar QD weggeschreven. Deze instructie doet dus precies het omgekeerde van de instructie LOAD ALL.
Met behulp van deze twee instructies is het mogelijk om programma's op de QD een andere naam te geven en eventueel 1 of meer programma's van de QD te wissen. Het is dan wel noodzakelijk dat U de QD eerst opnieuw formateert. Deze mogelijkheid bestaat alleen wanneer U in het bezit bent van een RAM-kaart, anders moet U gebruik maken van het programma DELETE.

In het volgende deel van dit hoofdstuk zullen nog enkele instructies volgen die U zelf aan de BASIC toe kunt voegen. De werking en bedoeling van de nieuwe instructies volgen ook in het volgende deel van dit hoofdstuk. U moet er echter wel rekening meehouden dat het toevoegen van de nieuwe instructies inhoudt dat dat ten koste van een aantal andere instructies zal gaan.
Uit het onderzoek dat voorafgaande aan dit boek gehouden is, is gebleken dat er maar weinig mensen zijn die intensief gebruik maken van een plotter. Vandaar dat de meeste nieuwe instructies in plaats zullen komen van een aantal instructies die specifiek voor de plotter bedoeld zijn. Er zal bij de nieuwe instructies vermeld staan welke oude instructies zijn verdwenen.

1.3: Extra BASIC-instructies.

In dit deel van hoofdstuk 1 treft U wel echt extra BASIC-instructies aan, vandaar dat de aanhalingstekens rond extra weggelaten zijn. In dit deel staan niet alleen programma's in staan, maar ook een uitleg hoe de nieuwe instructies werken. Het is jammer dat we U nog niet de allernieuwste instructies kunnen laten zien die in de speciale GAME-BASIC voor zullen komen. U zult het dus moeten doen met instructies die bij een behoorlijk aantal mensen bekend zullen zijn. Verderop in dit boek zal van sommige programma's uitgelegd worden hoe de programma's precies werken. Hier zullen we ons beperken tot de uitleg van de instructies.

INVERSE[ON].

Voor QD-BASIC:

```
10 POKE $A015,$CD,$64,$1,$9D,$28,$8,$11,
$23,$D9,$ED,$53,$EC,$5,$C9,$11,$DF,$42,$
18,$F6
20 POKE $A028,$23,$D9,$EE,$FF,$C9
30 POKE $200,$28,$A0 :POKE $7460,$41 :PO
KE $9FF7,$A3
40 POKE $59EC,$49,$4E,$56,$45,$52,$53,$C
5 :POKE $9FF1,$79
```

Helaas hebben we geen cassetteversie voor U.
INVERSE zegt eigenlijk zelf al wat de bedoeling van deze instructie is. Met deze instructie kunt U de tekst inverse op het scherm krijgen. Inverse houdt in dat alle zwarte delen wit worden en alle witte delen zwart. Wanneer de werkkleur en de achtergrondkleur anders zijn, dan zullen natuurlijk die kleuren gelden. De werking van INVERSE:

```
INVERSEON - De kleuren worden omgekeerd.
INVERSE   - De kleuren worden weer normaal.
```

Een voorbeeld:

```
INVERSEON :PRINT"NEPTUNES SOFTWARE"; :IN
VERSE :PRINT"-The best there is."
```

De instructies PMODE en PSKIP zijn onbruikbaar geworden.

CLIST.

Deze listing geldt voor de QD-BASIC. De aanpassingen voor cassette-BASIC staan achter de regels die verandert moeten worden.
De instructie XOPEN is niet meer te gebruiken.

```
10 FOR A=$0 TO $43 :READ A$ :POKE $FAB+A
,VAL("$"+A$) :NEXT A :POKE $5FF0,$AB,$F
20 FOR A=0 TO 29 :READ A$ :POKE $3BDF+A,
VAL("$"+A$) :NEXT A :POKE $5CFD,$DF,$3B
```

```
30 POKE $FEF,$43,$98,$92,$9F,$A9,$3A :PO
KE $5AAA,$43,$4C,$49,$53,$D4
40 DATA ED,5B,4D,9E,C4,D3,5E,D5,EB,CD
50 DATA 78,5E,D1,23,23,23,23,30,03,21
60 DATA A6,00,EB,D5,22,4D,9E,E1,11,A4
70 DATA 11,D5,CD,01,84,D1,21,00,00,1A
80 DATA 4F,06,00,FE,00,28,06,13,1A,47
90 DATA 09,18,F2,11,F5,0F,DF,16,11,EF
100 DATA 0F,DF,05,DF,01,C3,81,58
110 DATA CD,51,85,E5,7B,FE,01,28,0D,FE
120 DATA 00,C2,6A,63,21,81,58,22,F0,5F
130 DATA E1,C9,21,AB,0F,22,F0,5F,E1,C9
```

Aanpassingen voor cass.BASIC:

```
40 DATA ED,5B,D6,9D,C4,D3,5E,
D5,EB,CD
60 DATA A6,00,EB,D5,22,D6,9D,
E1,11,A4
70 DATA 11,D5,CD,8A,83,D1,21,
00,00,1A
110 DATA CD,DA,84,E5,7B,FE,01
,28,0D,FE
```

Werking van CLIST:

CLIST1 - het checklist commando aanzetten.
CLIST0 - het checklist commando uitzetten.

Als het checklist commando aangezet is, volgt na het intypen van een regel een checknummer. Wanneer U nu een listing uit een boek in typt waarachter deze checknummers staan en de nummers overeenkomen met de nummers in het boek, weet U dat U de regels goed ingetypt heeft. U kunt het natuurlijk ook gebruiken wanneer U een lsiting van U zelf op papier zet.

BACK.

Deze listing geldt voor zowel de QD- als de cassette-BASIC.

```
10 DATA 2A,70,10,23,23,23,23,7E,B7,C9,CD
,00,57,28,13,21,70,10
20 DATA 11,44,57,01,12,00,ED,B0,2A,70,10
,01,10,00,ED,B0,C3,8E
30 DATA 61,CD,00,57,C2,8E,63,21,44,57,11
,70,10,01,12,00,ED,B0
40 DATA ED,5B,70,10,01,10,00,ED,B0,CD,3E
,00,AF,C9
50 FOR A=$5700 TO $5743 :READ A$ :POKE A
,VAL("$"+A$)
60 NEXT A :POKE $5B17,$42,$41,$43,$CB :P
OKE $5C99,$A,$57 :POKE $5D1D,$25,$57
```

De werking van BACK:

Wanneer U NEW heeft gegeven en nog niets nieuws geladen of ingetypt heeft, krijgt U na het invoeren van BACK de listing terug.
Mocht er wel iets nieuws in het geheugen staan dan krijgt U een instruction error.

BOR.

Allereerst volgt de cassette-BASIC versie en daarna de QD versie. Waarschijnlijk zullen de meesten van U de instructie al kennen.

cassette-versie:

```
10 POKE $59E9,$42,$4F,$D2
20 POKE $55B0,$CD,$DA,$84,$7B,$1,$CF,$6,
$ED,$79,$C9
30 POKE $5C9D,$B0,$55
```

diskette-versie:

```
10 POKE $59E9,$42,$4F,$D2
20 POKE $5599,$CD,$51,$85,$7B,$1,$CF,$6,
$ED,$79,$C9
30 POKE $5C9D,$99,$55
```

Werking van de BOR-instructie:

BOR x (0 =< x =< 65535) verandert de randkleur in de kleur die bepaald wordt door x.

De instructie OFF is na invoer van dit programma niet meer te gebruiken.

BCOLOR.

Allereerst de QD-versie en daarna de aanpassing voor cassette-BASIC.

```
10 POKE $5600,$CD,$51,$85,$7B,$32,$15,$5
6,$C9
20 POKE $5608,$3A,$9B,$10,$32,$1F,$56,$C
3,$D9,$5
30 POKE $5611,$23,$D9,$77,$3E,$0,$F6,$40
,$D3,$CC,$7E,$EE,$FF
40 POKE $561D,$77,$3E,$7,$F6,$80,$D3,$CC
,$C3,$EF,$5
50 POKE $5B21,$42 :POKE $5D6,$C3,$8,$56
60 POKE $5EC,$C3,$11,$56 :POKE $5D21,$0,
$56
```

Voor cassette-BASIC moet U bij regel 10 het volgende intypen:

```
10 POKE $5600,$CD,$DA,$84,$7B,$32,$15,$5
6,$C9
```

De werking van BCOLOR:

BCOLOR x (0 =< x =< 65535) zorgt dat de achtergrondkleur van de tekst die op het scherm komt verandert in de kleur die met x overeen komt. Een voorbeeld:

```
BCOLOR 1 :PRINT"NEPTUNES SOFTWARE"; :BCO
LOR 2 :PRINT"-The best there is."
```

Dit was de laatste extra instructie. Later volgt meer hierover. De instructie CCOLOR kan overigens niet meer gebruikt worden.

1.4: Handige BASIC-(sub)routines.

Als laatste in dit hoofdstuk zullen U enkele handige en soms ook leuke (sub)routines voorgeschoteld worden. Tevens zal bij enkele programma's een nadere uitleg gegeven worden, zodat U ook nog begrijpt hoe U de programma's het beste kunt gebruiken en hoe ze in elkaar zitten.

```
5 REM Soort achtervolging
10 INIT"CRT:M1"
20 XP=0 :YP=0
30 A=INT(RND*40) :B=INT(RND*24)
40 CURSOR A,B :PRINT"O"
50 CURSOR XP,YP :PRINT CHR$(104);
60 IF XP=A AND YP=B THEN 120
70 WAIT 150
80 CURSOR XP,YP :PRINT" ";
90 IF XP<A THEN XP=XP+1
100 IF YP<B THEN YP=YP+1
110 GOTO 50
120 CURSOR 0,0 :PRINT"GOTCHA!"
130 GET A$ :IF A$="" THEN 130 ELSE 10
```

Wat U in dit programma ziet is waarchijnlijk wel bekend. In het overbekende spel PAC-MAN gebeurt precies hetzelfde, daarin achtervolgt een spookje de hoofdpersoon. In dit programma wordt er niets achtervolgd, maar gaat het figuurtje via de kortste weg op het doel af. Als het doel nu zou bewegen, zou precies hetzelfde gebeuren.
Aan het programma zelf valt weinig uit te leggen. In de regels 90 en 100 wordt het figuurtje richting het doel gestuurd. Als het doel nu zou bewegen, zouden er ook nog twee regels moeten worden toegevoegd voor het geval dat het figuurtje zich onder of rechts van het doel zou bevinden. Zoekt U zelf eens uit hoe dat gaat.

```
5 REM Tijd constant in beeld
10 INIT"CRT:M1"
20 INPUT"WAT IS DE TIJD? (HHMMSS)";A$
30 IF LEN(A$)<>6 THEN 10
40 TI$=A$
50 CLS
60 CONSOLE 1,24
70 CURSOR 29,0 :PRINT"TIJD:";TI$
80 CURSOR INT(RND*39),24 :PRINT"*"
90 GOTO 70
```

Alles in dit programma draait eigenlijk om de instructie CONSOLE. Voordat deze instructie uitgelegd wordt, moet U eerst iets anders weten. Het beeldscherm is opgebouwd uit 25 rijen. De eerste rij heeft het nummer 0 en de laatste rij heeft het nummer 24. Daar is de instructie CONSOLE ook op gebaseerd. Het eerste getal dat volgt na de instructie CONSOLE verwijst naar de eerste rij waar de instructie zijn uitwerking op heeft en het tweede getal verwijst naar het aantal rijen dat ook gewoon gebruikt kan worden. Met CONSOLE wordt namelijk een bepaald aantal rijen afgescheiden waarbinnen een aantal instructies geen invloed meer hebben. Het gebied dat aangegeven wordt door CONSOLE is gewoon nog door iedere instructie te bewerken. Het gebied daarbuiten

is slechts nog door een paar instructies te bewerken.
Een paar voorbeelden van CONSOLE:

CONSOLE 1,24 - Rij 1 t/m 24 zijn nog door alle instructies te bewerken.
CONSOLE 5,10 - Rij 5 t/m 14 zijn nog door alle instructies te bewerken.
enz.

Voor mensen die de werking van deze instructie nog niet helemaal begrijpen: Kijk eens goed in het handboek, daarin staat ook een goede uitleg van alle instructies; ook van de instructie CONSOLE.

```
1 REM Leuke manier om tekst op scherm te
zetten.
2 DATA 46,3,46,5,45,5,47,3,47,5,45,3
3 INIT"CRT:M1"
4 PAL 3,14 :PAL 1,5
5 FOR A=1 TO 6 :READ X,Y :SYMBOL X,Y,"NE
PTUNES",3,3 :NEXT A
6 SYMBOL [2]46,4,"NEPTUNES",3,3
7 FOR A=0 TO 7 :LINE [1]46+A*14,30+A*3,2
37-A*14,30+A*3 :NEXT A
8 GET A$ :IF A$="" THEN 8 ELSE CLS :END
```

Bij dit programmaatje hoeft geen verdere toelichting, het is gewoon een leuk klein programmaatje.

```
5 REM REM-regels in programma gebruiken.
10 ''''''''''''''''''''''''''''''''''''''
20 '            INSTRUCTIES             '
30 ''''''''''''''''''''''''''''''''''''''
31 ' Dit programmaatje laat zien hoe '
32 ' het mogelijk is om instructies  '
33 ' die in REM-regels staan ook in  '
34 ' het programma te laten zien     '
35 ' zonder dat de regelnummers      '
36 ' zichtbaar zijn.                 '
37 ''''''''''''''''''''''''''''''''''''''
50 INIT"CRT:M1"
60 PAL 3,0
70 LIST 10-37
80 BOX [0]0,0,20,199,0
90 PAL 3,15
100 GET A$ :IF A$="" THEN 100 ELSE CLS :
END
```

De beschrijving van dit programmaatje vindt U in het programmaatje zelf, daar wordt dus niet verder op ingegaan. De werking van het programma is ook niet moeilijk te begrijpen.

```
5 REM BASIC score-routine
10 INIT"CRT:M1"
20 BOX 135,85,185,115
30 CURSOR 17,11 :PRINT"SCORE:"
40 SYMBOL 70,130,"SCORE:",2,2
50 S$="00000" :SC=0
```

```
60 SYMBOL [0]165,130,S$,2,2
70 SC$=STR$(SC) :S$=MID$((S$),1,6-LEN(SC
$))+SC$
80 CURSOR 17,13 :PRINT S$
90 SYMBOL 165,130,S$,2,2
100 WAIT 250
110 SC=SC+10 :GOTO 60
```

In dit programmaatje wordt op een behoorlijk ingewikkelde manier de score opgebouwd. Hoe dat precies in elkaar zit is ook moeilijk uit te leggen, vandaar dat U het uitsluitend en alleen met de listing moet doen.

```
10 REM Letter voor letter
20 DIM A$(2)
30 A$(1)="Dit is een voorbeeld."
40 A$(2)="(C) by NEPTUNES SOFTWARE."
50 INIT"CRT:M1"
60 FOR A=1 TO 2 :FOR B=1 TO LEN(A$(A))
70 CURSOR B-1,A-1 :PRINT [2]CHR$(200)
80 WAIT 100 :BEEP
90 CURSOR B-1,A-1 :PRINT MID$(A$(A),B,1)
 :NEXT :NEXT
```

In principe een niet zo heel erg ingewikkeld programma, vandaar dat er ook geen nadere uitleg bij staat.

```
5 REM Sorteren
10 INIT"CRT:M1"
20 DIM A$(10)
30 PRINT"VOER 10 NAMEN IN. (IN HOOFDLETT
ERS)"
40 FOR T=1 TO 10 :INPUT A$(T)
50 NEXT T
60 CLS
70 PRINT"HET PROGRAMMA IS NU AAN HET SOR
TEREN."
80 FOR E=1 TO 10 :FOR D=1 TO 10
90 IF A$(E)<A$(D) THEN GOSUB "SORTEER"
100 NEXT D,E
110 CLS
120 PRINT"GESORTEERD KOMEN DEZE NAMEN IN
 DEZE     VOLGORDE:" :PRINT
130 FOR T=1 TO 10 :PRINT A$(T) :NEXT T
140 GET B$ :IF B$="" THEN 140 ELSE CLS :
END
150 LABEL "SORTEER"
160 U$=A$(E) :A$(E)=A$(D) :A$(D)=U$
170 RETURN
```

Het sorteren van bepaalde gegevens wordt in veel programma's gebruikt, vandaar dat het hier ook besproken wordt. Allereerst een korte uitleg over de werking van het programma.
Waarom alleen hoofdletters, zult U natuurlijk vragen. De mensen die de characterset van de SHARP kennen weten dat de hoofdletters keurig op

alfabetische volgorde in de characterset achter elkaar staan. Dat is niet het geval met de kleine letters, die staan kriskras door elkaar in de characterset.
Aan de hand van de characterset wordt gesorteerd. De letter die het eerste in de characterset staat komt als eerste. Als het nu zo is dat de kleine 'a' verderop in de characterset staat dan de kleine 'e', wordt bij het sorteren de 'e' eerder genomen dan de 'a'.
Door een klein stukje machinetaal te gebruiken is dit wel te veranderen, maar daar houden we ons momenteel niet mee bezig.
Sorteren wordt gebruikt in allerlei bestanden en verschillende soortgelijke programma's. In die programma's wordt meestal niet alleen op naam gesorteerd, maar ook op adres, postcode, woonplaats en ga zo maar door. In die programma's zit het sorteren heel wat ingewikkelder in elkaar dan op de vorige bladzijde beschreven is. Met dit programma kunt U in eerste instantie aan de gang, maar voor het toepassen in een groot bestand is het niet geschikt, daarvoor zult U eerst verder moeten experimenteren met sorteren.

```
5 REM Saven van data.
10 INIT"CRT:M1"
20 DIM A(10)
30 FOR B=1 TO 10
40 PRINT"GETAL";B :INPUT A(B)
50 NEXT B
60 PRINT"DRUK OP REC/PLAY"
70 INP@$D2,A :IF A<>152 THEN 70
80 INIT"CRT:M1"
90 DEFAULT"CMT:"
100 WOPEN #2,"10 GETALLEN"
110 FOR B=1 TO 10
120 PRINT #2,A(B)
130 NEXT B
140 CLOSE #2
150 END
```

Dit programma schrijft tien data naar een cassette. Het wegschrijven naar cassette gaat via de instructie PRINT #2. Op die manier kunt U allerlei data naar cassette schrijven, ook namen en dergelijke.
U kunt niet alleen naar cassette schrijven, maar ook naar QD, FD, RS232 en ga zo maar door. In het handboek staat wat U dan moet veranderen in het programma.
WOPEN staat voor WRITE OPEN, oftewel maak het mogelijk om data weg te schrijven. Deze instructie gaat (altijd) vooraf aan de instructie PRINT. Aan het einde moet U niet vergeten om af te sluiten met CLOSE!
Nu wilt U natuurlijk ook weten hoe U de data, die net weggeschreven zijn, weer kunt laden. Hier volgt het programma:

```
5 REM Laden van data.
10 INIT"CRT:M1"
20 DIM A(10)
30 PRINT"DRUK OP PLAY"
40 INP@$D2,A :IF A<>152 THEN 40
50 INIT"CRT:M1"
60 DEFAULT"CMT:"
70 ROPEN#2,"10 GETALLEN"
```

```
80 FOR B=1 TO 10
90 INPUT #2,A(B)
100 NEXT B
110 CLOSE #2
120 FOR B=1 TO 10 :PRINT"GETAL";B;":";A(
B)
130 NEXT B
140 END
```

Spoelt U natuurlijk eerst de cassette terug, voordat U dit programma RUNt!
U ziet dat dit programma veel weg heeft van het vorige programma. Het grootste verschil is dat hier ROPEN #2 staat ipv WOPEN #2 en dat er INPUT ipv PRINT staat. ROPEN staat voor READ OPEN, oftewel maak het mogelijk om data in te lezen. INPUT staat normaal ook voor invoer en PRINT voor uitvoer. Het zit dus allemaal behoorlijk logisch in elkaar. Vergeet hierbij ook niet om af te sluiten met CLOSE #2.
Er zijn nog enkele instructies die te gebruiken zijn, zoals EOF en dergelijke, maar die instructies zijn momenteel nog niet relevant. U heeft in eerste instantie meer dan voldoende aan de bovenstaande twee programma's. De andere instructies worden toch zelden tot nooit gebruikt.

```
5 REM Tafels
10 INIT"CRT:M1"
20 INPUT"Welke tafel?? ";T
30 FOR A=1 TO 10
40 PRINT USING"##";A;
50 PRINT" x";
60 PRINT T;" = ";
70 PRINT USING"###";A*T
80 NEXT
90 WAIT 3000
100 GOTO 10
110 'Probeer maar eens uit wat
120 'dit programma doet als je op
130 'regel 40 en 70 een # weghaalt
140 'of toevoegt, als je bijvoorbeeld
150 'de tafel van 100 of 555 doet!!!
```

```
10 REM Omgekeerd
50 INIT"CRT:M1"
60 LIST
70 FOR A=0 TO 24 :A$="" :DIM B$(40)
80 FOR T=0 TO 39 :B$(40-T)=CHR$(PEEK($20
00+A*40+T)) :NEXT T
90 FOR T=1 TO 40 :A$=A$+B$(T) :NEXT T
100 BOX [0]0,A*8,319,A*8+7,0
110 SYMBOL 0,A*8,A$,1,1
120 NEXT A
130 GET P$ :IF P$="" THEN 130
140 END
```

Ter afsluiting van dit hoofdstuk waren dit twee kleine routines die waarschijnlijk geen nadere uitleg nodig hebben.

HOOFDSTUK 2: MACHINETAAL OP DE SHARP MZ-800.

In dit hoofdstuk zullen, aan de hand van veel kleine machinetaal-programmaatjes, de mogelijkheden in machinetaal op de SHARP duidelijk gemaakt worden. Niet alleen pure machinetaalprogramma's maar ook BASIC-programma's met machinetaal zullen getoond worden.
De uitleg van een aantal programma's zal later in dit boek volgen. Wanneer U trouwens wilt weten wat alle DATA precies betekenen, oftewel wat de assemblerinstructie is, kunt U dat opzoeken in APPENDIX A, daarin staan namelijk alle codes met hun bijbehorende assemblerinstructie.

2.1: Nuttige en leuke machinetaal(sub)routines.

Allereerst zal de pure machinetaal aan de orde komen. Om te beginnen zal uitgelegd worden hoe U de programma's in moet voeren.

-Allereerst zet U de computer aan.
-Daarna drukt U op M. (Monitor.)
-Daarna voert U M2000 in als het eerste adres van de listing 2000 is.
M1200 in als het eerste adres van de listing 1200 is.
-Wanneer op papier 1200 0D staat, hoeft U alleen 0D op het toetsenbord in te typen (uiteraard gevolgd door <cr>), want op het scherm staat dan al 1200 wanneer U het goed gedaan heeft. Na <cr> ingedrukt te hebben verschijnt er dan 1201 op het scherm en dan typt U de code in die op papier naast het adres 1201 staat en zo gaat U door tot U alles ingetypt heeft.
-Aan het einde drukt U op SHIFT/BREAK.
-Daarna kunt U het programma SAVEN naar cassette of QD.
-Voor cassette voert U dan S in.
-Voor QD voert U dan QS in.
-Daarna volgen de woorden Filename, Top adrs, End adrs en Exc adrs, allemaal gevolgd door een ? Bij elk programmaatje staat apart vermeld wat U daar in moet voeren.
-U kunt daarna het programma starten door Gxxxx in te typen. Bij elk programma staat ook vermeld wat U voor xxxx in moet typen.
-Het programma stopt automatisch of U moet op de resetknop of een bepaalde toets drukken, dat zal ook bij ieder programma apart vermeld worden.
-U kunt een machinetaalprogramma van cassette laden door simpelweg de computer aan te zetten en daarna op C te drukken.
-Hoe U een machinetaalprogramma van QD kunt laden is reeds in hoofdstuk 0 besproken.

Helaas is er lang niet van elk programma een assemblerlisting. De assemblerlisting van enkele programma's kunt U verderop in dit boek vinden. Daar staat ook een redelijk uitgebreide uitleg bij. Het zal niet zo zijn dat deze uitleg zo uitgebreid is dat iemand die een leek is op het gebied van de machinetaal gelijk in machinetaal kan programmeren, maar het is wel zo dat mensen die in een beginnend stadium staan ermee verder kunnen en ook mensen die al redelijk ver in de machinetaal verdiept zijn er nieuwe ideeën door opdoen.

Voor mensen die machinetaal willen leren zal waarschijnlijk in 1990 een boek uitkomen bij SCCE of bij ons.

Het beeldscherm scrollen naar links.

```
1200 21   1205 D0   120A 06   120F 13   1214 23   1219 0E   121E 20
1201 00   1206 0E   120B 27   1210 10   1215 13   121A 20   121F FB
1202 D0   1207 19   120C 1A   1211 FA   1216 0D   121B 10   1220 C3
1203 11   1208 7E   120D 77   1212 08   1217 20   121C FE   1221 00
1204 01   1209 08   120E 23   1213 77   1218 EF   121D 0D   1222 12
```

```
Filename? LINKS-SCROLLEN
Top adrs? 1200
End adrs? 1222
Exc adrs? 1200
Opstarten met G1200.
Dit programma is alleen te stoppen door op de resetknop te drukken.
```

Het programma laat het hele scherm naar links bewegen. Alles wat aan de linker kant verdwijnt komt er rechts weer bij.

Het beeldscherm scrollen naar rechts.

```
1200 21   1205 D3   120A 06   120F 1B   1214 2B   1219 0E   121E 20
1201 E7   1206 0E   120B 27   1210 10   1215 1B   121A 20   121F FB
1202 D3   1207 19   120C 1A   1211 FA   1216 0D   121B 10   1220 C3
1203 11   1208 7E   120D 77   1212 08   1217 20   121C FE   1221 00
1204 E6   1209 08   120E 2B   1213 77   1218 EF   121D 0D   1222 12
```

```
Filename? RECHTS-SCROLLEN
Top adrs? 1200
End adrs? 1222
Exc adrs? 1200
Opstarten met G1200.
Dit programma is alleen te stoppen door op de resetknop te drukken.
```

Het programma laat het hele scherm naar rechts bewegen. Alles wat aan de rechter kant verdwijnt komt er links weer bij.

Het beeldscherm scrollen naar boven.

```
1200 21   1207 1A   120E 10   1215 CF   121C 77   1223 10   122A 00
1201 D8   1208 06   120F FA   1216 21   121D 23   1224 FE   122B C0
1202 CF   1209 27   1210 0D   1217 C0   121E 13   1225 0D   122C C3
1203 11   120A 1A   1211 20   1218 D3   121F 10   1226 20   122D 00
1204 00   120B 77   1212 F5   1219 06   1220 FA   1227 FB   122E 12
1205 D0   120C 23   1213 11   121A 28   1221 0E   1228 CD
1206 0E   120D 13   1214 D8   121B 1A   1222 20   1229 1B
```

```
Filename? BOVEN-SCROLLEN.
Top adrs? 1200
End adrs? 122E
Exc adrs? 1200
Opstarten met G1200.
Dit programma is te stoppen door op de resetknop of een willekeurige
toets te drukken.
```

Het beeldscherm scrollen naar beneden.

```
1200 21   1207 1A   120E 10   1215 D3   121C 77   1223 10   122A 00
1201 0F   1208 06   120F FA   1216 21   121D 23   1224 FE   122B C0
1202 D4   1209 27   1210 0D   1217 00   121E 13   1225 0D   122C C3
1203 11   120A 1A   1211 20   1218 D0   121F 10   1226 20   122D 00
1204 E7   120B 77   1212 F5   1219 06   1220 FA   1227 FB   122E 12
1205 D3   120C 2B   1213 11   121A 28   1221 0E   1228 CD
1206 0E   120D 1B   1214 E8   121B 1A   1222 20   1229 1B
```

```
Filename? BENEDEN-SCROLLEN
Top adrs? 1200
End adrs? 122E
Exc adrs? 1200
Opstarten met G1200.
Dit programma is te stoppen door op de resetknop of een willekeurige
toets te drukken.
```

Het programma hiervoor liet het hele beeldscherm naar boven bewegen en alles wat bovenaan verdween kwam er onderaan weer bij. Dit programma doet precies het omgekeerde.

2 Charactersets tonen.

```
1200 3E   1206 00   120C FF   1212 52   1218 D1   121E F0   1224 C9
1201 C6   1207 08   120D 70   1213 23   1219 06   121F ED
1202 CD   1208 21   120E 19   1214 10   121A FF   1220 52
1203 DC   1209 00   120F 36   1215 F7   121B 70   1221 23
1204 0D   120A D0   1210 70   1216 21   121C 19   1222 10
1205 11   120B 06   1211 ED   1217 18   121D 36   1223 F7
```

```
Filename? 2 CHARACTERSETS
Top adrs? 1200
End adrs? 1224
Exc adrs? 1200
Opstarten met G1200.
Dit programma stopt vanzelf.
```

Het programma toont de twee charactersets die in de SHARP zitten. In totaal staan U dus 512 tekens ter beschikking en die tekens kunnen ook veranderd worden in elk ander willekeurig teken, maar dat zal in een ander programma in dit hoofdstuk aan de orde komen.

De borderkleur veranderen met de joystick.

```
2000 3E   2008 32   2010 FD   2018 CC   2020 FE   2028 0C   2030 0C
2001 5F   2009 01   2011 25   2019 27   2021 EF   2029 20   2031 20
2002 01   200A 20   2012 20   201A 20   2022 C8   202A 2D   2032 2C
2003 CF   200B 21   2013 FA   201B FE   2023 00   202B 22   2033 C3
2004 06   200C 20   2014 DB   201C FD   2024 C3   202C 0C   2034 2B
2005 ED   200D 03   2015 F0   201D CC   2025 00   202D 20   2035 20
2006 79   200E 2D   2016 FE   201E 2F   2026 20   202E C9
2007 3D   200F 20   2017 FE   201F 20   2027 2A   202F 2A
```

```
Filename? BORDER-JOY
Top adrs? 2000
End adrs? 2035
Exc adrs? 2000
Opstarten met G2000.
```
Dit programma is te stoppen door op de resetknop of op de vuurknop van joystick 1 te drukken.

Met behulp van de joystick (in poort 1) kunt U de border met een andere snelheid van kleur doen veranderen. In het begin ziet U een aantal kleuren die heel langzaam naar boven bewegen.
Normaal kunt U maar 1 kleur aan de BORDER geven en nu ziet U er meer. Dat komt omdat het menselijk oog gemiddeld maar 21 beelden per seconde waar kan nemen en dat er veel sneller een andere kleur aan de border gegeven wordt. Dit verklaart ook waarom U de border ziet flikkeren.
Het is niet de bedoeling om U precies uit te leggen hoe dat nu kan, want dan komt er natuurkunde en een klein beetje biologie aan te pas.

Random iets op het scherm zetten.

```
1200 21   1208 16   1210 12   1218 23   1220 23   1228 0E   1230 00
1201 BF   1209 09   1211 3D   1219 0D   1221 36   1229 20   1231 12
1202 D3   120A 06   1212 10   121A 20   1222 6B   122A 10   1232 4F
1203 3A   120B 26   1213 F8   121B FC   1223 3E   122B FE   1233 C3
1204 05   120C FE   1214 15   121C 36   1224 C0   122C 0D   1234 0A
1205 E0   120D 00   1215 C2   121D 6B   1225 CD   122D 20   1235 12
1206 07   120E CA   1216 32   121E 23   1226 DC   122E FB
1207 4F   120F 18   1217 12   121F 23   1227 0D   122F C3
```

```
Filename? RANDOM
Top adrs? 1200
End adrs? 1235
Exc adrs? 1200
Opstarten met G1200.
```
Dit programma is alleen te stoppen door op de resetknop te drukken.

Dit programma zet op een 'willekeurige' plaats twee sterretjes onder aan het scherm en laat deze naar boven bewegen en herhaalt daarna deze cyclus. Waarom staat willekeurig tussen aanhalingstekens?
Een computer kan alleen logisch denken. Het kiezen van een willekeurig getal is iets onlogisch, vandaar dat dat moeilijk gaat. In dit geval wordt het getal ook nog uit de klok weggehaald en een klok loopt als het goed is erg regelmatig ook al is dit een heel speciale klok. Om nu goed een willekeurig getal te krijgen is dus heel erg moeilijk.

Achter- en voorgrondkleur veranderen.

```
2000 21   2006 28   200C 20   2012 20   2018 00
2001 00   2007 73   200D F7   2013 FD   2019 C0
2002 D8   2008 23   200E 1D   2014 10   201A C3
2003 0E   2009 10   200F 06   2015 FB   201B 00
2004 19   200A FC   2010 50   2016 CD   201C 20
2005 06   200B 0D   2011 0D   2017 1B
```

```
Filename? KLEUREN
Top adrs? 2000
End adrs? 201C
Exc adrs? 2000
Opstarten met G2000.
```
Dit programma is te stoppen door op de resetknop of een willekeurige toets te drukken.

Dit programma laat de voor- en achtergrondkleur veranderen. Af en toe ziet U ook de characterset veranderen. U kunt dus een andere characterset gebruiken door een andere kleur te kiezen. Dat werd ook al in een ander programma uit dit hoofdstuk getoond. Wanneer de code voor de kleur kleiner is dan 80H, wordt de eerste characterset getoond.

SCORE-routine.

```
2000 3E   200C 23   2018 D1   2024 36   2030 20   203C 20   2048 C3
2001 C6   200D 10   2019 7E   2025 20   2031 36   203D 20   2049 20
2002 CD   200E FB   201A 3C   2026 2B   2032 20   203E 36   204A 20
2003 DC   200F 06   201B FE   2027 7E   2033 2B   203F 20   204B 36
2004 0D   2010 01   201C 2A   2028 3C   2034 7E   2040 2B   204C 20
2005 21   2011 0D   201D CA   2029 FE   2035 3C   2041 7E   204D 2B
2006 F2   2012 20   201E 24   202A 2A   2036 FE   2042 3C   204E 36
2007 D1   2013 FD   201F 20   202B CA   2037 2A   2043 FE   204F 21
2008 06   2014 10   2020 77   202C 31   2038 CA   2044 2A   2050 C9
2009 04   2015 FB   2021 C3   202D 20   2039 3E   2045 CA
200A 36   2016 21   2022 0F   202E C3   203A 20   2046 4B
200B 20   2017 F5   2023 20   202F 20   203B C3   2047 20
```

```
Filename? SCORE-ROUTINE
Top adrs? 2000
End adrs? 2050
Exc adrs? 2000
Opstarten met G2000
```
Dit programma stopt vanzelf.

U ziet in het midden van het scherm de score van 0 tot en met 10000 oplopen. Dit gebeurt wel met een vertraging, anders staat de score op 10000 voordat U iets heeft kunnen zien. U kunt het natuurlijk ook voor iets anders dan de score gebruiken.

Character veranderen.

```
A000 D3   A00A 11   A014 36   A01E FF   A028 23   A032 01   A03C C9
A001 E4   A00B 00   A015 FF   A01F 23   A029 36   A033 00
A002 DB   A00C A8   A016 23   A020 36   A02A 00   A034 10
A003 E0   A00D ED   A017 36   A021 FF   A02B D3   A035 11
A004 21   A00E B0   A018 FF   A022 23   A02C E4   A036 00
A005 00   A00F DB   A019 23   A023 36   A02D DB   A037 C0
A006 C0   A010 E1   A01A 36   A024 FF   A02E E0   A038 ED
A007 01   A011 21   A01B FF   A025 23   A02F 21   A039 B0
A008 00   A012 00   A01C 23   A026 36   A030 00   A03A DB
A009 10   A013 A8   A01D 36   A027 FF   A031 A8   A03B E1
```

```
Filename? CHAR-CHANGE
Top adrs? A000
End adrs? A03C
Exc adrs? A000
Opstarten met GA000
Dit programma stopt vanzelf.
```

Een leuk resultaat. Wat is nu het geval? Het teken van de spatie is in een blokje verandert en overal waar spaties op het scherm stonden, staan nu dus blokjes. Zo kunt U alle 256 tekens van de eerste characterset en tevens alle 256 tekens van de tweede characterset veranderen in ieder willekeurig teken.
Een nadere toelichting:

De data van de verschillende characters liggen allemaal opgeslagen in het schaduwgeheugen. Via bankswitsching kunt U in dit schaduwgeheugen komen en alle data overzetten naar het gewone geheugen. In dit geval komen alle data vanaf A800 in het geheugen te staan. Elke character heeft zijn eigen code, zo heeft de spatie de code 00. Aangezien de data opgeslagen zijn naar volgorde van de codes, komt de spatie dus als eerste.
Elke character bestaat uit 8 data. De data van de spatie liggen dus vanaf A000 t/m A007 opgeslagen. Wanneer deze data nu veranderd worden en daarna alle data vanaf A800 weer naar het schaduwgeheugen geschreven worden, zal het teken van de spatie veranderd zijn en U ziet wat daarvan het gevolg is.
Wanneer U dus een bepaalde character wilt veranderen, moet U de code daarvan opzoeken en deze met 8 vermenigvuldigen en dat bij A000 optellen. (De codes kunt U overigens niet in het handboek vinden, U moet ze zelf uitzoeken). Vanaf dat adres tot 7 adressen daarna staan de data van de character opgeslagen. Hoe het programma verder in elkaar zit zal later in dit boek besproken worden.

HIGH-RESOLUTION Scroll.

```
6000 DB   600C D3   6018 78   6024 F0   6030 D8   603C 0B   6048 06
6001 E0   600D CC   6019 B1   6025 21   6031 7F   603D 78   6049 28
6002 3E   600E 21   601A C2   6026 F0   6032 11   603E B1   604A 1A
6003 00   600F 00   601B 14   6027 8F   6033 00   603F C2   604B 77
6004 D3   6010 80   601C 60   6028 06   6034 80   6040 38   604C 23
6005 CE   6011 01   601D 3E   6029 28   6035 01   6041 60   604D 13
6006 3E   6012 00   601E 32   602A 36   6036 40   6042 21   604E 10
6007 00   6013 20   601F D3   602B FF   6037 1F   6043 18   604F FA
6008 D3   6014 AF   6020 F0   602C 23   6038 1A   6044 9F   6050 C3
6009 F0   6015 77   6021 3E   602D 10   6039 77   6045 11   6051 2F
600A 3E   6016 23   6022 C3   602E FB   603A 23   6046 D8   6052 60
600B 03   6017 0B   6023 D3   602F 21   603B 13   6047 7F
```

```
Filename? HR-SCROLLEN
Top adrs? 6000
End adrs? 6052
Exc adrs? 6000
Opstarten met G6000.
Dit programma is alleen te stoppen door op de resetknop te drukken.
```

Dit programma scrollt een rode lijn langzaam naar boven. Wanneer de lijn boven verdwijnt, komt hij beneden weer terug. Het verschil met het eerder getoonde scrollen is dat het nu veel geleidelijker gaat. Dit heeft te maken met het feit dat dit programma gebruik maakt van de VIDEO-RAM en die is specifiek voor de MZ-800, terwijl de andere scroll-programma's ook op de MZ-700 draaien.
Hoe het programmeren in de VIDEO-RAM precies in elkaar zit zal in een ander hoofdstuk, dat speciaal over de VIDEO-RAM gaat, besproken worden. Vanuit de BASIC kunt U namelijk veel meer met de VIDEO-RAM doen en er ook beter inkomen.

Randscroll.

```
A000 BF   A018 00   A030 23   A048 D0   A060 11   A078 07   A090 4E
A001 A1   A019 A8   A031 A0   A049 11   A061 00   A079 77   A091 11
A002 AD   A01A ED   A032 21   A04A 27   A062 C6   A07A 23   A092 EE
A003 A5   A01B B0   A033 00   A04B 00   A063 ED   A07B 10   A093 AE
A004 A5   A01C DB   A034 D0   A04C 06   A064 B0   A07C FA   A094 06
A005 BD   A01D E1   A035 06   A04D 17   A065 DB   A07D 21   A095 07
A006 81   A01E 21   A036 28   A04E 36   A066 E1   A07E E0   A096 1A
A007 FF   A01F E0   A037 36   A04F DD   A067 21   A07F AE   A097 77
A008 3E   A020 AE   A038 DF   A050 19   A068 F8   A080 11   A098 2B
A009 C6   A021 0E   A039 23   A051 36   A069 AE   A081 E1   A099 1B
A00A CD   A022 04   A03A 10   A052 DC   A06A 06   A082 AE   A09A 10
A00B DC   A023 11   A03B FB   A053 23   A06B 08   A083 4E   A09B FA
A00C 0D   A024 00   A03C 21   A054 10   A06C 7E   A084 06   A09C 71
A00D D3   A025 A0   A03D C0   A055 F8   A06D 0F   A085 07   A09D 0E
A00E E4   A026 06   A03E D3   A056 D3   A06E 77   A086 1A   A09E 10
A00F DB   A027 08   A03F 06   A057 E4   A06F 23   A087 77   A09F 06
A010 E0   A028 1A   A040 28   A058 DB   A070 10   A088 13   A0A0 FF
A011 21   A029 77   A041 36   A059 E0   A071 FA   A089 23   A0A1 10
A012 00   A02A 23   A042 DE   A05A 21   A072 21   A08A 10   A0A2 FE
A013 C0   A02B 13   A043 23   A05B 00   A073 F0   A08B FA   A0A3 0D
A014 01   A02C 10   A044 10   A05C AE   A074 AE   A08C 71   A0A4 20
A015 00   A02D FA   A045 FB   A05D 01   A075 06   A08D 21   A0A5 F9
A016 10   A02E 0D   A046 21   A05E 00   A076 08   A08E EF   A0A6 C3
A017 11   A02F C2   A047 28   A05F 01   A077 7E   A08F AE   A0A7 56
                                                           A0A8 A0
```

```
Filename? RANDSCROLL
Top adrs? A000
End adrs? A0A8
Exc adrs? A008
Opstarten met GA008.
```
Dit programma is alleen te stoppen door op de resetknop te drukken.

Een leuk effect. Het is bijvoorbeeld te gebruiken bij een beginscherm. In het midden komen dan bijvoorbeeld het copyright en de naam van de maker van het programma.
Ook hier worden tekens van de characterset verandert. Door die tekens op het scherm te zetten, krijgt U de speciale figuurtjes te zien.
Het lijkt of alles naar links, rechts, boven of beneden beweegt, dat is echter niet het geval. Het teken wordt gewoon geroteerd en dan wordt dit effect al bereikt.

Randtekst.

Dit programma bestaat uit twee delen. Het eerste deel is het hoofdprogramma en het tweede deel zijn de bijbehorende data. Dit programma is leuk om te gebruiken aan het begin van een door U zelf gemaakt machinetaalprogramma. In het midden kunt U desnoods dan de naam van het programma zetten. De mensen die het programma BREAK-OUT van SFF-SOFT (clubcassette drie van SCCE) kennen begrijpen meteen wat er met het voorgaande bedoeld wordt, want in dat programma wordt dit ook toegepast.

deel 1:

```
2000 CD   2015 11   202A 21   203F 00   2054 00   2069 7E   207E C9
2001 87   2016 59   202B 01   2040 21   2055 ED   206A FE   207F AF
2002 20   2017 FD   202C 28   2041 21   2056 B0   206B 7F   2080 32
2003 0E   2018 19   202D 00   2042 00   2057 CD   206C CC   2081 29
2004 08   2019 10   202E ED   2043 D0   2058 1B   206D 7F   2082 20
2005 06   201A EC   202F B0   2044 06   2059 00   206E 20   2083 32
2006 28   201B 0D   2030 1B   2045 18   205A FE   206F 0E   2084 3F
2007 36   201C 20   2031 06   2046 1A   205B 00   2070 A0   2085 20
2008 02   201D E7   2032 18   2047 77   205C C2   2071 10   2086 C9
2009 11   201E 06   2033 0E   2048 13   205D 79   2072 FE   2087 CD
200A 68   201F 28   2034 28   2049 C5   205E 20   2073 0D   2088 79
200B 01   2020 36   2035 13   204A 01   205F 3A   2074 20   2089 20
200C 19   2021 07   2036 0D   204B 28   2060 29   2075 FB   208A CD
200D 36   2022 23   2037 20   204C 00   2061 20   2076 C3   208B 7F
200E 07   2023 10   2038 FC   204D 09   2062 3C   2077 25   208C 20
200F 11   2024 FB   2039 7E   204E C1   2063 32   2078 20   208D 21
2010 40   2025 11   203A 12   204F 10   2064 29   2079 3E   208E 00
2011 01   2026 00   203B 23   2050 F5   2065 20   207A C6   208F D8
2012 19   2027 D0   203C 10   2051 EB   2066 32   207B CD   2090 C9
2013 36   2028 21   203D F5   2052 01   2067 3F   207C DC
2014 09   2029 00   203E 11   2053 28   2068 20   207D 0D
```

deel 2:

```
2100 6B   2112 6B   2124 6B   2136 6B   2148 00   215A 04   216C 00
2101 6B   2113 6B   2125 6B   2137 6B   2149 05   215B 00   216D 10
2102 6B   2114 6B   2126 6B   2138 6B   214A 05   215C 0F   216E 01
2103 6B   2115 6B   2127 6B   2139 6B   214B 0E   215D 0D   216F 13
2104 6B   2116 6B   2128 6B   213A 6B   214C 00   215E 00   2170 13
2105 6B   2117 6B   2129 6B   213B 6B   214D 0C   215F 14   2171 05
2106 6B   2118 6B   212A 6B   213C 6B   214E 05   2160 0F   2172 0E
2107 6B   2119 6B   212B 6B   213D 6B   214F 15   2161 05   2173 00
2108 6B   211A 6B   212C 6B   213E 6B   2150 0B   2162 00   2174 09
2109 6B   211B 6B   212D 6B   213F 6B   2151 00   2163 14   2175 0E
210A 6B   211C 6B   212E 6B   2140 00   2152 16   2164 05   2176 00
210B 6B   211D 6B   212F 6B   2141 00   2153 0F   2165 00   2177 15
210C 6B   211E 6B   2130 6B   2142 04   2154 0F   2166 0B   2178 17
210D 6B   211F 6B   2131 6B   2143 09   2155 12   2167 15   2179 00
210E 6B   2120 6B   2132 6B   2144 14   2156 02   2168 0E   217A 05
210F 6B   2121 6B   2133 6B   2145 00   2157 05   2169 0E   217B 09
2110 6B   2122 6B   2134 6B   2146 09   2158 05   216A 05   217C 07
2111 6B   2123 6B   2135 6B   2147 13   2159 0C   216B 0E   217D 05
```

```
217E 0E   2191 00   21A4 15   21B7 07   21CA 6B   21DD 6B   21F0 6B
217F 00   2192 0F   21A5 0B   21B8 12   21CB 6B   21DE 6B   21F1 6B
2180 10   2193 10   21A6 0B   21B9 01   21CC 6B   21DF 6B   21F2 6B
2181 12   2194 00   21A7 05   21BA 0D   21CD 6B   21E0 6B   21F3 6B
2182 0F   2195 05   21A8 0E   21BB 0D   21CE 6B   21E1 6B   21F4 6B
2183 07   2196 05   21A9 00   21BC 01   21CF 6B   21E2 6B   21F5 6B
2184 12   2197 0E   21AA 13   21BD 2E   21D0 6B   21E3 6B   21F6 6B
2185 01   2198 00   21AB 14   21BE 00   21D1 6B   21E4 6B   21F7 6B
2186 0D   2199 14   21AC 0F   21BF 00   21D2 6B   21E5 6B   21F8 6B
2187 0D   219A 0F   21AD 10   21C0 6B   21D3 6B   21E6 6B   21F9 6B
2188 01   219B 05   21AE 14   21C1 6B   21D4 6B   21E7 6B   21FA 6B
2189 67   219C 14   21AF 00   21C2 6B   21D5 6B   21E8 6B   21FB 6B
218A 13   219D 13   21B0 04   21C3 6B   21D6 6B   21E9 6B   21FC 6B
218B 2E   219E 00   21B1 09   21C4 6B   21D7 6B   21EA 6B   21FD 6B
218C 00   219F 14   21B2 14   21C5 6B   21D8 6B   21EB 6B   21FE 6B
218D 04   21A0 05   21B3 00   21C6 6B   21D9 6B   21EC 6B   21FF 7F
218E 0F   21A1 00   21B4 10   21C7 6B   21DA 6B   21ED 6B
218F 0F   21A2 04   21B5 12   21C8 6B   21DB 6B   21EE 6B
2190 12   21A3 12   21B6 0F   21C9 6B   21DC 6B   21EF 6B
```

```
Filename? RANDTEKST
Top adrs? 2000
End adrs? 2200
Exc adrs? 2000
Opstarten met G2000.
```
Dit programma stopt automatisch bij het indrukken van een willekeurige toets.

Willekeurige kleuren.

Zoals al eerder gezegd, klopt het woord willekeurig niet helemaal. Op deze computer is het onmogelijk om willekeurige getallen te kiezen, omdat een computer nu eenmaal een logisch apparaat is en het kiezen van willekeurige getallen is iets onlogisch.
In dit programma wordt het A-register geroteerd en wordt een vertraging uitgevoerd om de willekeurigheid iets meer te benaderen.
Het (willekeurige) getal wordt uit de klok gehaald. Dit is niet een gewone klok die om de zoveel tijd verspringt, maar een klok die bijhoudt hoeveel tijd-cycli er voor een aantal instructies doorlopen moeten worden.

```
2000 3E   2005 21   200A 06   200F 77   2014 15   2019 F8   201E EB
2001 C6   2006 00   200B 28   2010 23   2015 20   201A 10   201F C9
2002 CD   2007 D8   200C 3A   2011 0F   2016 FD   201B F0
2003 DC   2008 0E   200D 05   2012 16   2017 3D   201C 0D
2004 0D   2009 19   200E E0   2013 20   2018 20   201D 20
```

```
Filename? WILL. KLEUREN
Top adrs? 2000
End adrs? 201F
Exc adrs? 2000
Opstarten met G2000
```
Dit programma stopt automatisch.

2.2: Machinetaal binnen BASIC-800.

De mogelijkheid om machinetaal te gebruiken wanneer U met BASIC-800 bezig bent bestaat natuurlijk ook. Een goed voorbeeld hiervan is hoofdstuk 3 dat de hele VIDEO-RAM behandeld. In hoofdstuk 3 staan hoofdzakelijk BASIC-programma's die zorgen dat er iets in machinetaal uitgevoerd wordt. In dit deel van hoofdstuk twee wordt precies hetzelfde gedaan, alleen hebben deze programma's (bijna) niets met de VIDEO-RAM te maken. Alle programma's zijn door iedereen te gebruiken. U heeft de extra ic's dus niet nodig.

Programma 1: BORDER-verandering bij zien van knipperende cursor.

Zolang de knipperende cursor te zien is, verandert de kleur van de BORDER. Bij INPUT verandert de BORDER dus ook, omdat er dan ook een knipperende cursor te zien is.

```
10 DATA $21,$2C,$09,$36,$20,$23,$36,$F0,
$21,$3C,$09,$36,$20,$23,$36,$F0,$C9
20 FOR A=0 TO 16 :READ B :POKE $F000+A,B
 :NEXT
30 DATA $C5,$F5,$DB,$D5,$01,$CF,$06,$ED,
$79,$F1,$C1,$CD,$BA,$00,$C9
40 FOR A=0 TO 14 :READ B :POKE $F020+A,B
 :NEXT
50 USR($F000)
```

Om het knipperen te stoppen, typt U het volgende in:

```
POKE $F025,$0 - knipperen uit.
POKE $F025,$CF - knipperen aan.
```

Programma 2: 20 functietoetsen.

De mogelijkheid bestaat om, met behulp van de CTRL-toets, 20 functietoetsen te definiëren en te gebruiken.

```
F1  t/m F5  = direct indrukken
F6  t/m F10 = +SHIFT indrukken
F11 t/m F15 = +CTRL  indrukken
F16 t/m F20 = +SHIFT+CTRL indrukken

10 DATA 87,87,87,87,21,5F,57,C3,58,1,!0F
11 DATA FE,14,D2,6A,63,37,3F,DE,A,E5,!03
12 DATA CD,90,55,C3,5C,6C,!40
13 DATA E,0,DF,8,79,C6,31,57,1E,31,!0B
14 DATA FE,3A,20,3,11,32,30,ED,53,20,!39
15 DATA 6C,11,18,6C,DF,B,79,CD,90,55,!4F
16 DATA 46,23,C5,CD,BB,6B,C1,C,79,FE,!B4
17 DATA A,20,D7,DF,8,E1,C9,!46
18 DATA CB,77,C2,7F,C,3A,30,D,FE,FF,!03
19 DATA CA,3B,C,3A,31,D,CB,47,21,AF,!6E
20 DATA 57,28,3,21,5F,57,1,10,0,3A,!12
```

```
21 DATA 30,D,CB,7F,28,3,9,CB,77,28,!37
22 DATA 3,9,CB,6F,28,3,9,CB,67,28,!0B
23 DATA 6,9,CB,5F,3A,31,D,C2,3B,C,!C5
24 DATA C3,99,C,C,49,4E,49,54,22,43,!D2
25 DATA 52,54,3A,4D,32,D,0,0,0,8,!46
26 DATA 43,4F,4E,53,4F,4C,45,D,0,0,!66
27 DATA 0,0,0,0,0,6,53,59,4D,42,!A7
28 DATA 4F,4C,0,0,0,0,0,0,0,0,!42
29 DATA 0,6,4C,41,42,45,4C,22,0,0,!CA
30 DATA 0,0,0,0,0,0,0,9,4B,45,!63
31 DATA 59,20,4C,49,53,54,D,0,0,0,!25
32 DATA 0,0,0,6,50,4F,49,4E,54,28,!DD
33 DATA 0,0,0,0,0,0,0,0,0,7,!E4
34 DATA 52,45,53,54,4F,52,45,28,0,0,!30
35 DATA 0,0,0,0,0,B,4F,4E,45,52,!6F
36 DATA 52,4F,52,47,4F,54,4F,0,0,0,!9B
37 DATA 0,9,4C,49,4D,49,54,4D,41,58,!09
38 DATA D,0,0,0,0,0,0,6,53,54,!C3
39 DATA 49,43,4B,28,0,0,0,0,0,0,!C2
40 DATA 0,0,0,0,!C2
100 AD=$5590 :RG=10
110 READ A$ :IF LEFT$(A$,1)="!" THEN 140
120 POKE AD,VAL("$"+A$) :AD=AD+1 :CK=CK+
VAL("$"+A$) :IF CK>255 THEN CK=CK-256
130 GOTO 110
140 KC=VAL("$"+RIGHT$(A$,2)) :IF KC<>CK
THEN PRINT"Fout in DATA regel";RG :END
150 RG=RG+1 :IF RG<13 THEN 110
200 AD=$55C1 :RG=13 :CK=0
210 READ A$ :IF LEFT$(A$,1)="!" THEN 240
220 POKE AD,VAL("$"+A$) :AD=AD+1 :CK=CK+
VAL("$"+A$) :IF CK>255 THEN CK=CK-256
230 GOTO 210
240 KC=VAL("$"+RIGHT$(A$,2)) :IF KC<>CK
THEN PRINT"Fout in DATA regel";RG :END
250 RG=RG+1 :IF RG<18 THEN 210
300 AD=$5720 :RG=18 :CK=0
310 READ A$ :IF LEFT$(A$,1)="!" THEN 340
320 POKE AD,VAL("$"+A$) :AD=AD+1 :CK=CK+
VAL("$"+A$) :IF CK>255 THEN CK=CK-256
330 GOTO 310
340 KC=VAL("$"+RIGHT$(A$,2)) :IF KC<>CK
THEN PRINT"Fout in DATA regel";RG :END
350 RG=RG+1 :IF RG<41 THEN 310
400 POKE $6C56,$9A,$55 :POKE $6BB7,$C3,$
C1,$55 :POKE $C7B,$C3,$20,$57
410 CLS :KEY LIST
```

Natuurlijk kunt U ook andere instructies onder de extra functietoetsen zetten. Een instructie als INIT"CRT:M2" heeft bijvoorbeeld geen enkele zin wanneer U de extra ic's niet in Uw computer heeft.
U kunt nu met de DEF KEY instructie alle 20 functietoetsen naar eigen keuze definiëren, U hoeft dus geen DATA te gaan veranderen.
Met de CTRL-toets kan trouwens nog meer gedefinieerd worden, dat zal in een ander programma getoond worden.

Programma 3: 'Sprite' bewegen met joystick.

Dit programma laat een 'sprite' door middel van de joystick (via poort 1) over het scherm bewegen. Het is niet 100% volledig een sprite en het is ook (bijna?) niet mogelijk om een volledige sprite te maken. In mode M2 bestaat ook de mogelijkheid om de sprite voor of achter iets langs te laten bewegen. In mode M1 bestaat deze mogelijkheid helaas niet. Om iedereen te kunnen laten zien hoe er met een sprite gewerkt wordt, is dit programma in mode M1 geschreven. Na dit programma zullen enkele toevoegingen komen, zodat U in mode M2 voor en achter iets langs kunt bewegen. Dit alles betekent dus dat U de extra ic's niet nodig heeft voor dit programma, maar wel voor de toevoegingen.

```
10 LIMIT $FC00
20 INIT"CRT:M1" :PAL 2,0 :PAL 0,14
30 A$=CHR$(239,111,95,96,31,120,74,120,9
5,223,39,190,167,167,51,25,95,86,182,103
,111,95,111,239)
40 B$=CHR$(66,231,255,60,165,165,165,165
,165,165,36,255,0,195,255,128,127,170,14
8,127,128,255,247,65)
50 C$=CHR$(247,246,250,6,248,30,82,30,25
0,244,116,228,142,226,242,126,174,174,10
2,246,250,254,237)
60 D$=A$+B$+C$
70 POSITION 0,1 :PATTERN [1]-24,D$
80 LINE [2]0,0,319,0
90 LINE [2]0,199,319,199
100 DATA $3E,$00,$DB,$E0,$3E,$01,$D3,$CD
,$D3,$CC,$21,$28,$80,$DB,$F0,$FE,$F7,$CC
,$26,$FC,$FE,$FB,$CC,$4B,$FC,$FE,$FE,$CC
,$74,$FC,$FE,$FD,$CC,$AB,$FC,$C3,$04,$FC
110 DATA $3A,$01,$FC,$3C,$FE,$08,$CA,$00
,$FD,$32,$01,$FC,$11,$24,$00,$3E,$01,$D3
,$CD,$0E,$18,$37,$3F,$06,$04,$7E,$17,$77
,$23,$10,$FA,$19,$0D,$20,$F2,$AF,$C9
120 DATA $3A,$01,$FC,$3D,$FE,$FF,$CA,$22
,$FD,$32,$01,$FC,$11,$03,$00,$19,$11,$2C
,$00,$3E,$01,$D3,$CD,$0E,$18,$37,$3F,$06
,$04,$7E,$1F,$77,$2B,$10,$FA,$19,$0D,$20
,$F2,$AF,$C9
130 DATA $E5,$EB,$D5,$E1,$11,$D8,$FF,$19
,$3E,$02,$D3,$CD,$7E,$FE,$00,$C2,$A8,$FC
,$22,$0B,$FC,$D1,$3E,$01,$D3,$CD,$0E,$19
,$06,$04,$1A,$77,$23,$13,$10,$FA,$1B,$1B
,$1B,$1B,$D5,$11,$4C,$00,$19,$D1,$EB,$0D
,$20,$EA,$AF,$C9,$D1,$AF,$C9
140 DATA $11,$9B,$03,$19,$E5,$EB,$E1,$D5
,$11,$28,$00,$19,$3E,$02,$D3,$CD,$7E,$FE
,$00,$C2,$A8,$FC,$E5,$2A,$0B,$FC,$11,$28
,$00,$19,$22,$0B,$FC,$E1,$3E,$01,$D3,$CD
,$D1,$0E,$19,$06,$04,$1A,$77,$2B,$1B,$10
,$FA
150 DATA $13,$13,$13,$13,$D5,$11,$B4,$FF
,$19,$D1,$EB,$0D,$20,$EA,$AF,$C9
```

```
160 FOR T=0 TO 235 :READ A :POKE $FC00+T
,A :NEXT
170 DATA $11,$9C,$03,$19,$3E,$02,$D3,$CD
,$7E,$FE,$00,$C2,$49,$FC,$11,$64,$FC,$19
,$CD,$32,$FC,$2A,$0B,$FC,$23,$22,$0B,$FC
,$32,$01,$FC,$C3,$49,$FC
180 DATA $2B,$3E,$02,$D3,$CD,$7E,$FE,$00
,$C2,$72,$FC,$22,$0B,$FC,$3E,$07,$C3,$54
,$FC
190 FOR T=0 TO 52 :READ A :POKE $FD00+T,
A :NEXT
200 USR($FC00)
```

Dit programma is alleen te stoppen door op CTRL/reset te drukken.

In mode M2 de sprite voor een project langs laten bewegen kan gerealiseerd worden door de volgende regels te veranderen, cq toe te voegen:

```
20 INIT"CRT:M2" :PAL 0,14 :PAL 3,1 :PAL 2
,5
80 LINE [8]0,0,319,0
90 LINE [8]0,199,319,199
95 CIRCLE [2]160,100,50 :PAINT [2]160,10
0,2
195 POKE $FC7D,$08 :POKE $FCB8,$08 :POKE
 $FD05,$08 :POKE $FD24,$08
```

Om de sprite achterlangs te laten bewegen, hoeft U alleen PAL 3,1 in regel 20 te veranderen in PAL 3,5.

Eigenlijk gaat dit programma ook in op de VIDEO-RAM, maar omdat er al genoeg in hoofdstuk 3 staat en dit programma ook uitstekend in dit hoofdstuk past, staat het in dit hoofdstuk.

Programma 4: Characters veranderen.

Ook in BASIC bestaat de mogelijkheid om een character te veranderen. Het nadeel is echter wel dat dit 4K aan extra geheugen in beslag neemt. De data van de characters worden opgeslagen van $E000 t/m $EFFF. U kunt de data van een character veranderen door de ASCII-waarde van de character met 8 te vermenigvuldigen en dat bij $E000 op te tellen. Vanaf dat adres staan de 8 data van de character en door die te veranderen, verandert U de character.
Allereerst het programma, daarna zult U op de volgende bladzijde een voorbeeld om een character te veranderen kunnen vinden.

```
10 LIMIT $E000
20 INIT"CRT:M1"
30 POKE $FC00,$DB,$E0,$21,$0,$10,$11,$0,
$E0,$1,$0,$10,$ED,$B0,$DB,$E1,$C9
40 USR ($FC00)
50 POKE $FD00,$29,$CB,$EC,$CB,$F4,$CB,$F
C,$C3,$E2,$5
60 POKE $5DF,$C3,$0,$FD
```

Een voorbeeld:

U wilt bijvoorbeeld de letter 'B' veranderen. Dan zoekt U eerst uit wat de ASCII-waarde van deze character is en dat is 02. U vermenigvuldigt dit met 8 en krijgt dan als uitkomst 16. De hexadecimale waarde van 16 is $10. U telt deze $10 bij $E000 op en komt dan op $E010.
We maken voor het gemak een blokje van de letter 'B' en dat gaat als volgt:

```
POKE $E010,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$
FF
```

Wanneer U nu de letter 'B' in typt, krijgt U een blokje te zien. Met andere characters gaat dat precies hetzelfde.
U kunt ook de tweede characterset veranderen door $E800 ipv $E000 te gebruiken.
U kunt ten alle tijde de originele characterset terug krijgen door het volgende in te typen:

```
POKE $5DF,$29,$CB,$E4
```

Programma 5: 2e characterset onder CTRL-K.

Zoals al eerder gezegd, bestaat de mogelijkheid om met CTRL veel meer toetsen te definiëren. In dit programma kunt U met CTRL-K of PRINT CHR$(11) de tweede characterset oproepen of vanuit de tweede characterset terug gaan naar de eerste. Zo zijn er nog veel meer toetsen in combinatie met CTRL te gebruiken voor speciale zaken. Het probleem hierbij is dat U voor elke toets weer de bijbehorende adressen moet weten. Om het niet al te moeilijk te maken, het gaat hier tenslotte alleen om programmaatjes en niet om hoe de BASIC in elkaar zit, zal daar niet verder op ingegaan worden.

```
10 POKE $55ED,$DF,$20,$3A,$D4,$5,$FE,$0,
$28,$2,$3E,$FF,$3C,$32,$D4,$5,$C9
20 POKE $71,$ED,$55
30 INIT"CRT:M1"
40 PRINT"NEPTUNES PRODUCTIONS"
50 PRINT CHR$(11)"NEPTUNES PRODUCTIONS"
60 PRINT CHR$(11)
```

Er zijn nog veel meer programmaatjes te bedenken om in dit deel van hoofdstuk 2 te plaatsen, maar daarvoor is het niet bedoeld. Dit deel van hoofdstuk 2 is uitsluitend bedoeld voor programma's die niet in andere hoofdstukken in te delen zijn.
Ook programma's die de BASIC drastisch veranderen komen niet voor in dit hoofdstuk. Dit is gedaan om de BASIC niet te snel vast te laten lopen. Wanneer U te veel veranderingen in de BASIC aanbrengt, bestaat de mogelijkheid dat de BASIC vast loopt. Dit kan gebeuren omdat U bijvoorbeeld op 1 plaats in het geheugen twee verschillende zaken neer wilt zetten, bijvoorbeeld een extra BASIC-instructie en het veranderen van de BORDER tijdens het knipperen van de cursor. De computer kan op zo'n moment vast lopen en dan moet de BASIC opnieuw geladen worden.

2.3: Calls in de ROM-monitor.

In de ROM-monitor zijn een aantal standaardroutines al vastgelegd, zoals routines voor het uitlezen van het toetsenbord, voor het opvragen van de Directory, voor het omhoog scrollen van het scherm en ga zo maar door.
Door één CALL zijn deze routines aan te spreken. Zo kan met 3 bytes de directory van de QD opgevraagd worden en nog veel meer. Af en toe moeten bepaalde registers een zelf gekozen waarde bevatten, voordat U een CALL geeft. Om bijvoorbeeld de klok te starten moet in het register A (de accumulator) de waarde staan die aangeeft af het AM (voor de middag) of PM (na de middag) is en in het adres DE moeten het aantal seconden komen te staan (maximaal 43200), daarna kan met CALL 0033H de klok gestart worden. In dit deel van hoofdstuk 2 staat een programma dat dit heel duidelijk zal laten zien.
Het omgekeerde kan ook het geval zijn. U geeft eerst een CALL en daarna worden bepaalde registers met een bepaalde waarde geladen. Om het voorbeeld van de klok aan te houden: U voert eerst in CALL 003BH en daarna wordt het aantal seconden in DE gezet en de aanduiding voor welk deel van de dag het is komt in A te staan.
Om alles duidelijk uit te kunnen leggen, zullen de meeste CALLS in een programma verwerkt worden en zal tevens de Assemblerlisting er bij komen te staan om alles zo duidelijk mogelijk uit te kunnen leggen.
Eigenlijk horen de assemblerlistings in een ander hoofdstuk thuis, maar voor dit geval worden ze bij de machinetaallisting zelf geplaatst om het geheel niet te moeilijk te maken.

Programma 1: Tijd in beeld.

Zoals al gezegd, is er een interne klok in de SHARP aanwezig. Door CALL 0033H in te voeren is de klok te starten en door CALL 003BH in te geven is de tijd in seconden uit te lezen. Door dit alles in een ordelijk programma te verwerken, bestaat de mogelijkheid om een gewone digitale klok te laten lopen en dat is in dit programma gedaan.
Helaas start de klok elke keer bij 00:00:00 uur en gaat hij bij 12:00:00 uur gewoon door. Het is bijvoorbeeld leuk om in Uw eigen programma's toe te kunnen passen, maar dan met een willekeurig te kiezen begintijd. Om het programma niet al te lang te maken, is dat hier echter niet toegepast.
Allereerst volgt het machinetaalprogramma, dan gegevens over het saven van het programma, daarna volgt de assemblerlisting en als laatste komt er ook nog eens een hele lading uitleg.

```
1200 3E   120B DC   1216 43   1221 34   122C 77   1237 26   1242 3C
1201 00   120C 0D   1217 3A   1222 21   122D C3   1238 CC   1243 FE
1202 11   120D 11   1218 03   1223 0E   122E 13   1239 3C   1244 2A
1203 00   120E 68   1219 12   1224 D0   122F 12   123A 12   1245 CC
1204 00   120F 12   121A B8   1225 7E   1230 3E   123B C9   1246 49
1205 CD   1210 CD   121B CA   1226 3C   1231 20   123C 3E   1247 12
1206 33   1211 15   121C 13   1227 FE   1232 77   123D 20   1248 C9
1207 00   1212 00   121D 12   1228 2A   1233 2B   123E 77   1249 3E
1208 3E   1213 CD   121E 21   1229 CC   1234 7E   123F 2B   124A 20
1209 C6   1214 3B   121F 03   122A 30   1235 3C   1240 2B   124B 77
120A CD   1215 00   1220 12   122B 12   1236 FE   1241 7E   124C 2B
```

```
124D 7E   1254 C9   125B 3C   1262 3E   1269 49   1270 30   1277 0D
124E 3C   1255 3E   125C FE   1263 20   126A 4A   1271 3A
124F FE   1256 20   125D 2A   1264 77   126B 44   1272 30
1250 26   1257 77   125E CC   1265 2B   126C 20   1273 30
1251 CC   1258 2B   125F 62   1266 34   126D 3D   1274 3A
1252 55   1259 2B   1260 12   1267 C9   126E 20   1275 30
1253 12   125A 7E   1261 C9   1268 54   126F 30   1276 30
```

```
Filename? KLOK
Top adrs? 1200
End adrs? 1277
Exc adrs? 1200
Opstarten met G1200.
Dit programma is alleen te stoppen door op de resetknop te drukken.
```

Assemblerlisting van programma 1:

```
1200 ORG 1200H
1200 LD A,00H
1202 STTIJD:  LD DE,0000H
1205 CALL 0033H
1208 LD A,C6H
120A CALL 0DDCH
120D LD DE,TEKST
1210 CALL 0015H
1213 TIJD:  CALL 003BH
1216 LD B,E
1217 LD A,(STTIJD+1)
121A CP B
121B JP Z,TIJD
121E LD HL,STTIJD+1
1221 INC (HL)
1222 LD HL,D00DH
1225 LD A,(HL)
1226 INC A
1227 CP 2AH
1229 CALL Z,PA
122C LD (HL),A
122D JP TIJD
1230 PA:  LD A,20H
1232 LD (HL),A
1233 DEC HL
1234 LD A,(HL)
1235 INC A
1236 CP 26H
1238 CALL Z,PB
123B RET
123C PB:  LD A,20H
123E LD (HL),A
123F DEC HL
1240 DEC HL
1241 LD A,(HL)
1242 INC A
1243 CP 2AH
1245 CALL Z,PC
1248 RET
1249 PC:  LD A,20H
124B LD (HL),A
124C DEC HL
124D LD A,(HL)
124E INC A
124F CP 26H
1251 CALL Z,PD
1254 RET
1255 PD:  LD A,20H
1257 LD (HL),A
1258 DEC HL
1259 DEC HL
125A LD A,(HL)
125B INC A
125C CP 2AH
125E CALL Z,PE
1261 RET
1262 PE:  LD A,20H
1264 LD (HL),A
1265 DEC HL
1266 INC (HL)
1267 RET
1268 TEKST:  DEFB 54H
1269 DEFB 49H  I
126A DEFB 4AH  J
126B DEFB 44H  D
126C DEFB 3DH  =
126D DEFB 20H
126E DEFB 30H  0
126F DEFB 30H  0
1270 DEFB 3AH  :
1271 DEFB 30H  0
1272 DEFB 30H  0
1273 DEFB 3AH  :
1274 DEFB 30H  0
1275 DEFB 30H  0
1276 DEFB 0DH  CR
```

Iemand die goed opgelet heeft, heeft kunnen constateren dat er een heel klein verschil is tussen de machinetaallisting en de assemblerlisting. Dit is gedaan om het programma precies op de bladzijde te laten passen. Voor het programma maakt het heel weinig uit.

De uitleg van programma 1:

In dit programma komen in totaal 4 verschillende CALLS voor. Deze CALLS zullen hieronder nader besproken worden.

CALL 0033H - Deze CALL start de interne klok. In DE is het aantal seconden en in A is de waarde 0 (AM, oftewel voor de middag) of de waarde 1 (PM, oftewel na de middag). U moet de tijd dus in seconden omrekenen.

CALL 003BH - Deze CALL leest de klok uit. In DE komt het aantal seconden en in A komt het getal 0 of 1. Om de exacte tijd te weten moet U nu dus het aantal seconden in DE omrekenen naar uren, minuten en seconden en U weet de tijd.

CALL 0DDCH - Deze CALL voert iets met het scherm of met de cursorpositie uit. De waarde van het register A bepaalt wat er precies moet gebeuren. In dit geval bevat het register A de waarde C6H en dat betekent dat het beeldscherm schoon gemaakt wordt en in HL de waarde van de eerste cursorpositie komt.

CALL 0015H - Deze CALL print een bepaalde tekst op het scherm vanaf de cursorpostie. (Dat is dus het hokje links boven aan het scherm, omdat eerst de CALL hierboven is gegeven.)
De tekst is opgeslagen vanaf het adres dat door het register DE bepaald wordt en het einde van de tekst moet altijd aangegeven worden door een CODE 0DH.

Vanaf adres 1216 (LD B,E) t/m adres 122B wordt gekeken of de klok al 1 seconde vooruit is gegaan. Is dat niet het geval dan wordt dit herhaald. Dat gaat net zo lang door tot de klok wel een seconde vooruit is gegaan en dan wordt de cyclus van 122C t/m 1267 doorlopen om de klok op het beeldscherm met 1 seconde vooruit te laten gaan.
Aangezien de klok op het scherm altijd bij 00:00:00 uur begint, maakt het niet uit welke waarde het DE-register heeft bij het starten van de klok omdat die waarde niet op het scherm komt.

Hier heeft U kunnen zien hoe een programma besproken wordt. Zo zal het gaan met alle programma's. Ook in het hoofdstuk over assembleren zal de uitleg van deze aard zijn. In dat hoofdstuk zal er misschien nog iets extra bijkomen, maar dan hebben we het ook wel gehad.
U ziet dat het hier dus duidelijk niet de bedoeling is om U machinetaal te leren, daarvoor zijn namelijk al genoeg boeken geschreven.

Programma 2: Toon met veranderbare frequentie.

Dit is een heel erg aardig programmaatje dat gebruik maakt van een CALL om een toon te laten horen met een zelf te bepalen frequentie. Met de pijltjestoetsen voor omhoog en naar beneden kunt U de frequentie veranderen en dat geeft een heel erg leuk resultaat.

```
1200 21   1208 CD   1210 CC   1218 FE   1220 C3   1228 A1   1230 A1
1201 A1   1209 44   1211 23   1219 64   1221 08   1229 11   1231 11
1202 11   120A 00   1212 12   121A CA   1222 12   122A C9   1232 C9
1203 36   120B CD   1213 FE   121B 33   1223 2A   122B 2A   1233 CD
1204 00   120C 1B   1214 11   121C 12   1224 A1   122C A1   1234 47
1205 23   120D 00   1215 CC   121D 00   1225 11   122D 11   1235 00
1206 36   120E FE   1216 2B   121E 00   1226 23   122E 2B   1236 C9
1207 05   120F 12   1217 12   121F 00   1227 22   122F 22
```

```
Filename? TOON <FREQ>
Top adrs? 1200
End adrs? 1236
Exc adrs? 1200
Opstarten met G1200.
```

Dit programma is te stoppen door op SHIFT/BREAK te drukken. De toon hoort U dan ook niet meer.

Assemblerlisting van programma 2:

```
1200 LD HL,11A1H
1203 LD (HL),00H
1205 INC HL
1206 LD (HL),05H
1208 TOON:  CALL 0044H
120B CALL 001BH
120E CP 12H
1210 CALL Z,HOOG
1213 CP 11H
1215 CALL Z,LAAG
1218 CP 64H
121A JP Z,EINDE
121D DEFB 00H
121E DEFB 00H
121F DEFB 00H
1220 JP TOON
1223 HOOG:  LD HL,(11A1H)
1226 INC HL
1227 LD (11A1H),HL
122A RET
122B LAAG:  LD HL,(11A1H)
122E DEC HL
122F LD (11A1H),HL
1232 RET
1233 EINDE:  CALL 0047H
1236 RET
```

Uitleg van programma 2:

In dit programma komen 3 verschillende CALLS voor.

CALL 0044H - Deze CALL zorgt er voor dat een toon met een bepaalde frequentie opgewekt wordt. De frequentie wordt bepaald door de waardes van de adressen 11A1 en 11A2.

CALL 0047H - Deze CALL zorgt er voor dat de toon niet meer te horen is.

CALL 001BH - Deze CALL leest het toetsenbord uit en de waarde van de toets(en) die ingedrukt is/zijn komt/komen in het register A te staan. Is de pijltjestoets omhoog ingedrukt dan bevat A de waarde 12H. Is de pijltjestoets naar beneden ingedrukt dan bevat A de waarde 11H. Zijn de toetsen SHIFT en BREAK ingedrukt dan bevat A de waarde 64H.

Het programma zit zo eenvoudig in elkaar dat een verdere uitleg niet nodig is. Een leuk geintje kunt U wel uithalen door de adressen 121D t/m 121F te veranderen in CD,47,00 en daarna het programma te starten, U krijgt dan een ander soort toon te horen. In de assemblerlisting komt dan te staan:

```
121D CALL 0047H
```

Programma 3: 20 keer een zelf gekozen tekst tonen.

Bij dit programma moet U zelf eerst een tekst van maximaal 80 tekens invoeren en daarna op <cr> drukken. Vervolgens zult U deze tekst 20 keer achter elkaar op het beeldscherm zien verschijnen.

```
1200 11   1205 00   120A 0D   120F 00   1214 CD   1219 00   121E 00
1201 00   1206 3E   120B 06   1210 10   1215 15   121A 10
1202 20   1207 C6   120C 19   1211 FB   1216 00   121B F8
1203 CD   1208 CD   120D CD   1212 06   1217 CD   121C CD
1204 03   1209 DC   120E 06   1213 14   1218 06   121D AD
```

```
Filename? TEKST*20
Top adrs? 1200
End adrs? 121E
Exc adrs? 1200
```

Opstarten met G1200. Na opstarten is dit programma alleen nog op te starten met J1200. Verderop zal dit nader toegelicht worden.
Dit programma stopt vanzelf.

Assemblerlisting van programma 3:

```
1200 LD DE,2000H                1210 DJNZ HERHAAL
1203 CALL 0003H                 1212 LD B,14H
1206 LD A,C6H                   1214 ?TEKST:  CALL 0015H
1208 CALL 0DDCH                 1217 CALL 0006H
120B LD B,19H                   121A DJNZ ?TEKST
120D HERHAAL:  CALL 0006H       121C CALL 00ADH
```

Uitleg van programma 3:

In dit programma komen drie nieuwe CALLS voor. De CALLS CALL 0DDCH en CALL 0015H zijn al bekend en zullen hier dus niet uitgelegd worden.

CALL 0003H - Deze CALL zorgt er voor dat U een tekst van maximaal 80 tekens op het toetsenbord in kunt voeren. Deze tekst wordt vanaf het adres aangegeven door register DE in het geheugen opgeslagen en wordt gevolgd door een code 0DH of een break-code afgesloten.

CALL 0006H - Deze CALL werkt precies hetzelfde als de <cr>-toets. De cursor gaat namelijk naar de eerste positie van de volgende regel wanneer deze CALL uitgevoerd wordt.

CALL 00ADH - Deze CALL zorgt er voor dat de computer in de werkelijke ROM-monitor springt. Deze monitor heeft niet de mogelijkheid om de instructies voor de QD te gebruiken en gebruikt de J ipv de G om een CALL rechtstreeks uit te laten voeren. Tevens bestaat de mogelijkheid om met het teken # gevolgd door <cr> terug te gaan naar de BASIC, wanneer deze zich tenminste in de computer bevindt.

Over de laatste CALL nog een opmerking:
Wanneer U een machinetaalprogramma van QD laadt en met een RET-instructie teruggesprongen moet worden naar de ROM-monitor, doet de computer dat niet. Start U dat programma vanuit de ROM-monitor, zonder het eerst van QD te halen, dan wordt er wel gevolg gegeven aan RET.

Deze narigheid kunt U op twee manieren oplossen. De eerste oplossing is dat U het programma met QC laadt ipv met QL, dat kan alleen bij programma's die beginnen bij het adres 1200. Na het programma met QC geladen te hebben, drukt U op 'N' en kunt U het programma opstarten en zal er wel gevolg gegeven worden aan een RET-instructie.
Programma's die niet bij 1200 beginnen kunnen meestal niet op deze manier geladen en gestart worden. Het is namelijk zo dat bij QC het programma altijd vanaf 1200 in het geheugen gezet wordt.
De tweede oplossing is om, ipv een RET-instructie, CALL 00ADH te gebruiken. De computer springt dan altijd terug naar de ROM-monitor, ook al wordt het programma van QD gestart. Het enige nadeel is dat in deze echte ROM-monitor een aantal instructies niet meer gebruikt kunnen worden, zoals al eerder verteld is.

Programma 4: Kaderomlijning.

Een 'gewoon' programma om een kaderomlijning te maken is korter en ook sneller dan een programma met CALLS. Dit programma is dus uitsluitend bedoeld om een voorbeeld te geven van de gebruikte CALLS en niet om een efficiënter programma te maken.

```
1200 3E   1208 C8   1210 3E   1218 10   1220 00   1228 12   1230 C4
1201 C6   1209 CD   1211 C8   1219 F6   1221 10   1229 00   1231 CD
1202 CD   120A 12   1212 CD   121A 06   1222 F9   122A 3E   1232 DC
1203 DC   120B 00   1213 12   121B 27   1223 06   122B C2   1233 0D
1204 0D   120C 10   1214 00   121C 3E   1224 17   122C CD   1234 10
1205 06   120D F9   1215 CD   121D C8   1225 3E   122D DC   1235 EF
1206 28   120E 06   1216 06   121E CD   1226 C8   122E 0D   1236 C9
1207 3E   120F 16   1217 00   121F 12   1227 CD   122F 3E
```

Assemblerlisting van programma 4:

```
1200 LD A,C6H
1202 CALL 0DDCH
1205 LD B,28H
1207 LEEN:  LD A,C8H
1209 CALL 0012H
120C DJNZ LEEN
120E LD B,16H
1210 LTWEE:  LD A,C8H
1212 CALL 0012H
1215 CALL 0006H
1218 DJNZ LTWEE
121A LD B,27H
121C LDRIE:  LD A,C8H
121E CALL 0012H
1221 DJNZ LDRIE
1223 LD B,17H
1225 LVIER:  LD A,C8H
1227 CALL 0012H
122A LD A,C2H
122C CALL 0DDCH
122F LD A,C4H
1231 CALL 0DDCH
1234 DJNZ LVIER
1236 RET
```

Uitleg van programma 4:

In dit programma komt slechts 1 nieuwe CALL en twee bekende CALLS voor. CALL 0DDCH is wel iets veranderd, het register A is namelijk een keer met C2H en C4H geladen ipv de waarde C6H. Bij die waarden gebeurt het volgende:
Bij een waarde van C2H gaat de cursor 1 regel omhoog.
Bij een waarde van C4H gaat de cursor 1 positie naar links.

CALL 0012H - Bij deze CALL wordt de waarde van het register A omgezet in het overeenkomende teken en dat teken wordt op de cursorpositie neergezet en de cursorpositie schuift vervolgens één plaats naar rechts.

In het begin wordt eerst het beeldscherm schoongemaakt en komt de cursor op de positie links-boven (1200 - 1204). Daarna worden veertig blokjes naast elkaar neergezet en belandt de cursor op de eerste regel aan de linkerkant van het scherm (1205 - 120D). Vervolgens wordt aan het begin van elke regel een blokje neergezet en komt de cursor aan de linkerkant van de éénnalaatste regel (120E - 1219). Daaropvolgend worden er op de éénnalaatste regel 39 blokjes neergezet en belandt de cursor aan de rechterkant van de éénnalaatste regel (121A - 1221). Als laatste wordt op de laatste positie van iedere regel ook een blokje gezet en is de omlijning helemaal rond.
Waarom gaat de onderste regel niet helemaal tot de laatste regel?
Wanneer U op de laatste regel aan het eind iets neer gaat zetten met deze CALLS, wordt het beeldscherm 1 regel omhoog gescrolld en verdwijnt de bovenste regel.

Er zijn natuurlijk veel meer CALLS, de belangrijkste daarvan zullen hieronder los uitgelegd worden. De CALLS kunt U natuurlijk rechtstreeks uitproberen door G te gebruiken. Een voorbeeld: U kunt CALL 000FH uitproberen door G000F in te voeren.

CALL 000FH - Deze CALL zorgt er voor dat er 10 spaties achter elkaar gezet worden. Deze CALL heeft dus dezelfde functie als de TAB-toets.
CALL 001BH - Deze CALL is al besproken, maar hier willen wij U verwijzen naar de juiste blz. in het handboek waar U meer informatie over deze CALL kunt vinden en dat is blz. A-15.
CALL 0030H - Speelt een melody die opgeslagen is in data vanaf het adres dat door register DE bepaald wordt. De data moeten altijd afgesloten worden met 0DH. In het hoofdstuk over muziek kunt U hier veel meer over lezen.
CALL 003EH - Deze CALL doet hetzelfde als BEEP in BASIC.
CALL 0041H - Met deze CALL kan het TEMPO van de te spelen melodie bepaald worden. In het register A moet de waarde van het tempo staan. (1 <= A <= 7)
CALL 00F7H - Nà deze CALL zal bij het indrukken van een toets een korte BEEP te horen zijn. Bij het intypen van B gevolgd door <cr> zal hetzelfde resultaat bereikt worden. Om deze functie uit te schakelen, moet U nog een keer dezelfde CALL invoeren.
CALL 018FH - Het teken, waarvan de waarde zich in het A-register bevindt, wordt naar de printer of de plotter gestuurd.
CALL 01A5H - De tekst die opgeslagen is vanaf het adres dat door het DE-register aangegeven wordt wordt naar de printer of de plotter gestuurd. De tekst moet beëindigd worden met een code 0DH.
CALL 038DH - Bij deze CALL wordt de klok op 00.00.00 gezet en wordt AM verandert in PM of andersom.
CALL 069FH - De cassettemotor wordt gestart.
CALL 0700H - De cassettemotor stopt.
CALL 0DDCH - Over deze CALL vindt U meer op blz. A-16 van het handboek.

CALL 0FD8H - Vanaf het adres dat door HL aangegeven is, wordt op ieder adres de code 00H neergezet. Het aantal adressen dat met 00H geladen moet worden, wordt door het B-register aangegeven.

Op de vorige bladzijden vond U enkele CALLS uit het lage deel van de ROM monitor. Nu volgen enkele CALLS uit het hoge deel van de ROM-monitor.

CALL E090H Formateert de QD.
CALL E29BH - De QD-motor wordt gestart.
CALL E2E8H - De QD-motor stopt.
CALL E807H - Een programma wordt van cassette geladen en vanaf adres 1200H neergezet.
CALL E80AH - Een programma wordt naar cassette geschreven. Het programma moet vanaf adres 1200H in het geheugen staan.
CALL EFEFH - Directory van de QD wordt gegeven.

Er zijn zo nog veel meer CALLS voor de QD en de cassette. Tevens staan er van adres E000H t/m FFFFH CALLS voor de Floppy Disk. Het heeft weinig zin om CALLS voor deze apparaten te publiceren, omdat het meeste bereikt kan worden door een simpele instructie als QX e.d.

Nu volgt nog een overzicht van de belangrijkste adressen uit de serie van 1000H t/m 1200H. Op veel van deze adressen staat belangrijke informatie over o.a. de programma's die geladen zijn. De belangrijkste adressen volgen hieronder.

10F0H - Op dit adres is te bekijken wat voor een soort programma geladen is. Is de waarde 01H dan is het een OBJ-file. Is de waarde 05H dan is het een BTX-file en ga zo maar door.
10F1H - Vanaf dit adres t/m adres 1101H staat de programmanaam opgeslagen. De naam moet altijd met een code 0DH afgesloten worden.
1102H - Op dit adres en de volgende 5 adressen staan respectievelijk de lengte, het beginadres en het startadres van een programma opgeslagen. Voor de QD gelden de adressen 1104H t/m 1109H.
1171H - De X-positie van de cursor.
1172H - de Y-positie van de cursor.
1192H - De code van het teken dat op de cursorpositie staat.
119BH - Wordt gebruikt voor de klok. AM=0, PM=1.
119CH - Indien F0H dan loopt de interne klok.
119DH - Geeft aan of een BEEP te horen is bij een toetsaanslag of niet. De waarde 00H geeft aan dat er een BEEP te horen zal zijn bij iedere toetsaanslag.
119EH - Hier wordt de waarde van het muziektempo opgeslagen.
119FH - Hier wordt de waarde van de toonlengte opgeslagen.
11A0H - Hier wordt de waarde van het oktaafnummer opgeslagen.
11A1H - Dit adres is al eerder besproken.

Dit was een greep uit de vele CALLS en handige adressen die de ROM-monitor van de SHARP kent. Om U niet te belasten met CALLS waar U toch weinig aan heeft, zijn alleen de meest nuttige CALLS geplaatst. Wanneer U echter een QD-index programma of iets dergelijks wilt gaan schrijven, heeft U meer nodig. Voor meer CALLS kunt U ons altijd schrijven.

HOOFDSTUK 3: DE VIDEO-RAM.

In dit hoofdstuk zal verder ingegaan worden op de VIDEO-RAM.
Wat houdt VIDEO-RAM nu precies in? Bij verkoop wordt vaak gezegd dat de SHARP ook nog eens 16K VIDEO-RAM ter beschikking heeft. Wat wordt daar nu precies mee bedoeld?
U weet wat de gewone RAM inhoudt, daarin worden alle programma's opgeslagen. Zo bestaat er ook een opslaggedeelte dat enkel en alleen voor alle handelingen op het scherm bedoeld is, dat gedeelte wordt de VIDEO-RAM genoemd.
Het aansturen van de VIDEO-RAM gaat niet zo gemakkelijk als het aansturen van de gewone RAM. Allereerst moet er gebankswitscht worden alvorens U in de VIDEO-RAM iets uit kunt voeren. Wat houdt bankswitsching nu precies in?
Met bankswitsching kunt U omschakelen naar een ander geheugen. Vanwege de beperktheid van de Z80-A microprocessor, die maximaal 64K aan geheugen aan kan, is het noodzakelijk om op een andere manier meer geheugen te kunnen gebruiken, dat kan dus via bankswitsching. Een slimme denker is er dus ook al achter gekomen dat via bankswitsching meer dan 64K geheugen te gebruiken is. Het is inderdaad mogelijk om het geheugen uit te breiden met zeker 512K en dat extra geheugen is aan te sturen via bankswitsching. Dat is echter allemaal toekomstmuziek, het gaat nu om de VIDEO-RAM en die is maar 16K groot en geen 512K.
Hoe ziet bankswitsching er nu precies uit? U kunt het herkennen als U in het machinetaalgedeelte van een programma de instructies DB en E0 tegen komt. (In mnemonics: IN A,(E0H)). Het VIDEO-RAM geheugen is dan te vinden vanaf adres 8000. Het terugschakelen vanuit de VIDEO-RAM naar de gewone RAM is te herkennen aan de instructies DB en E1.
Nu genoeg over bankswitsching, de meesten zullen toch niet begrijpen hoe het nu precies in elkaar zit. Het belangrijkste is dat U niet helemaal in het duister blijft tasten naar het begrip bankswitsching.
Verderop in het boek zullen enkele programma's uit dit hoofdstuk gelicht worden en tevens nader besproken worden. Dan zal voor menigeen duidelijk worden hoe de aansturing van de VIDEO-RAM nu precies in elkaar zit.
Dit hoofdstuk bestaat hoofdzakelijk uit kleine programmaatjes die laten zien wat er zoal mogelijk is met de VIDEO-RAM en dat is meer dan de meesten van U zouden verwachten.
Nog even iets over de uitbreidings-ic's die het mogelijk maken om met 16 kleuren te werken op Uw SHARP. Bij elk programma zal vermeld staan of het alleen met die ic's werkt of ook met de standaard SHARP. Eventueel staan er aanpassingen bij om het resultaat leuker te maken.

3.1: Scrollen in de VIDEO-RAM.

Het scrollen op de SHARP kan op twee manieren; software- en hardwarematig. Sofwarematig houdt in dat het scrollen via een programma gebeurt en hardwarematig houdt in dat het via de ingebouwde hardware gebeurt, er is dan wel een klein programmaatje nodig om de hardware aan te kunnen sturen. Het voordeel van de hardwarematige oplossing is dat het veel sneller gaat. Het nadeel is dat U gebonden bent aan bepaalde rijen en dat de kolommen niet gevarieerd kunnen worden. Tevens kan het opzij-scrollen via de hardwarematige oplossing alleen met grote blokken en sprongen.

Omdat het opzijscrollen alleen met grote sprongen tegelijk kan, zal er verder niet op deze manier van scrollen op de hardwarematige manier ingegaan worden.
Nu volgt een programma dat laat zien hoe U gebruik kunt maken van de hardware om (een deel van) het beeldscherm te scrollen. Bij dit programma vindt U tevens enkele aanpassingen om in beide richtingen te kunnen scrollen en er wordt uitgelegd hoe U sneller kunt scrollen.

Scrollen naar boven.

Onderstaand programma zal (een deel van) het beeldscherm omhoog scrollen. Alles wat bovenaan verdwijnt, ziet U onderaan weer terugkomen. Het programma is zo ingesteld dat, wanneer U het beeldscherm 1 keer laat scrollen, het scherm na die ene keer precies weer in de beginstand komt te staan.
Er zal verder weinig uitleg gegeven worden bij dit programma. Hoe de hardware nu precies werkt, is een zaak voor de makers van de hardware. Voor U is van belang om te weten hoe U er gebruik van kunt maken en aan de hand van dit programma en een klein beetje uitleg zal dat hopelijk duidelijk worden.

```
10 INIT"CRT:M1" :INPUT"VANAF WELKE REGEL
 WILT U SCROLLEN?      (MAX. 24) ";A
20 IF A<0 OR A>24 OR A<>INT(A) THEN 10
30 INPUT"TOT WELKE REGEL WILT U SCROLLEN
?       (MAX. 25) ";B
40 IF B<=A THEN BEEP :PRINT :PRINT"KUNT
U AUB EEN HOGER REGELNUMMER OPGEVENDAN D
E BEGINREGEL?" :PRINT :GOTO 30
50 IF B>25 OR B<>INT(B) THEN PRINT :GOTO
 30
60 PRINT :INPUT"HOEVEEL KEER WILT U HET
SCHERM HELEMAAL ROND SCROLLEN?  (MAX. 20
) ";MA
70 IF MA<1 OR MA>20 OR MA<>INT(MA) THEN
BEEP :PRINT :GOTO 60
80 LIMIT $FE00
90 POKE $FE00,$DB,$CE,$E6,$40,$20,$FA,$E
D,$5B,$33,$FE,$2A,$2E,$FE,$19,$E5,$ED,$5
B,$37,$FE,$B7,$ED,$52,$E1,$20,$3,$2A,$39
,$FE,$22,$2E,$FE,$1,$CF,$6,$21,$32,$FE,$
ED,$BB,$DB,$CE,$E6,$40,$28,$FA,$C9
100 AR=(B-A)*5 :'AR=AANTAL REGELS
110 BE=A*5     :'BE=BEGINREGEL
120 EI=B*5     :'EI=EINDREGEL
130 SO=(EI-BE)*8
140 S1=INT(SO/256) :S2=SO-S1*256
150 POKE $FE2E,0,0,AR,BE,EI,5,0,0,0,S2,S
1,0,0
160 CLS :FOR KL=8 TO 32 :CURSOR KL-8,KL-
8 :PRINT [KL/8.1]"NEPTUNES"; :NEXT KL :C
URSOR 0,0 :PRINT [1]"NEPTUNES";
170 WAIT 1000 :FOR F=1 TO AR/5*8*MA :WAI
T 50/(MA*2.5) :USR($FE00) :NEXT F :WAIT
2000 :INIT"CRT:M1"
```

Het programma werkt helaas niet in alle vier de modes. Het werkt in ieder geval in de modes M1 en M2. Voor de modes M3 en M4 zijn kleine aanpassingen nodig die niet zo heel erg moeilijk te vinden zijn. Het aanpassen voor de modes M3 en M4 laten we dus aan U over.
Het omlaag scrollen kan via een korte aanpassing met hetzelfde programma gedaan worden. U verandert daarvoor de volgende regel als volgt:

```
150 POKE $FE2E,S2,S1,AR,BE,EI,$FB,$FF,0,
0,0,0,S2,S1
```

U heeft aan de hand van het programma kunnen zien dat het scrollen via de hardwarematige oplossing behoorlijk snel gaat. Het kan echter nog sneller. Dat kan gedaan worden door niet rij voor rij te scrollen, maar bijvoorbeeld twee of drie rijen tegelijk te scrollen. Het effect blijft ongeveer hetzelfde. Het ziet er nog behoorlijk regelmatig uit, maar het gaat een stukje sneller. Hoe kunt U nu twee of drie regels tegelijk scrollen, dat gaat als volgt:

Om twee regels tegelijk omhoog te scrollen, verandert U de volgende regels als volgt:

```
75 MA=MA/2
150 POKE $FE2E,0,0,AR,BE,EI,10,0,0,0,S2,
S1,0,0
```

Om drie regels tegelijk omhoog te scrollen, verandert U de volgende regels als volgt:

```
75 MA=MA/3
150 POKE $FE2E,0,0,AR,BE,EI,20,0,0,0,S2,
S1,0,0
```

Om twee regels tegelijk omlaag te scrollen, verandert U de volgende regels als volgt:

```
75 MA=MA/2
150 POKE $FE2E,S2,S1,AR,BE,EI,$F6,$FF,0,
0,0,0,S2,S1
```

Om drie regels tegelijk omlaag te scrollen, verandert U de volgende regels als volgt:

```
75 MA=MA/3
150 POKE $FE2E,S2,S1,AR,BE,EI,$EC,$FF,0,
0,0,0,S2,S1
```

Zo kunt U natuurlijk oneindig doorgaan, maar op een gegeven moment is het effect niet meer zo geweldig. Dan gaat het scherm met grote blokken naar beneden of boven bewegen en dat is de bedoeling natuurlijk niet.
Op de volgende bladzijden zal de softwarematige oplossing voor het scrollen besproken worden. Daarbij zijn veel meer mogelijkheden denkbaar en die zullen ook allemaal getoond worden. U zult dan gelijk zien dat het scrollen op die manier veel minder aan bepaalde grenzen gebonden is, maar wel een stuk langzamer gaat.

Scrollen op de softwarematige manier.

Op de softwarematige manier zijn er tal van mogelijkheden om te scrollen. U kunt naar boven, beneden, links en rechts scrollen en daarbij zijn tal van variaties mogelijk. Zo kunt U het beeld aan het einde laten verdwijnen, maar U kunt het ook aan het begin weer terug laten komen. Zo kunt U ook iets in het beeld laten scrollen dat ergens in het geheugen opgeslagen staat. U kunt een deel van de kleuren scrollen en de rest laten staan, maar ook alle kleuren tegelijk scrollen. Zo zijn er nog tal van mogelijkheden en die zullen allemaal besproken worden.

Programma 1: Naar links scrollen met 1 byte tegelijk in 1 kleur.

De eenvoudigste manier van scrollen is het scrollen van het hele scherm in 1 kleur, in 1 richting en met blokken van 8 bits (1 byte) tegelijk. Na het programma zal nog iets meer verteld worden.

```
10 LIMIT $FE00
20 DATA $DB,$E0,$3E,$83,$D3,$CC,$D3,$CD,
$21,$00,$80,$11,$01,$80,$0E,$C8,$7E,$08,
$06,$27,$1A,$77,$23,$13,$10,$FA,$08,$77,
$23,$13,$0D,$20,$EF,$DB,$E1,$C9
30 FOR A=0 TO 35 :READ B :POKE $FE00+A,B
 :NEXT A
40 INIT"CRT:M1"
50 LIST :LIST
60 FOR A=1 TO 80 :USR($FE00) :NEXT A
```

Bij dit programma komt alles wat aan de linkerkant verdwijnt aan de rechterkant weer tevoorschijn. Het programma kan dus nog eenvoudiger door alles gewoon aan de linkerkant te laten verdwijnen, maar dan is het effect minder leuk.
Het is heel eenvoudig om nu bijvoorbeeld de kleur blauw of rood te scrollen, waardoor de andere kleur achterblijft. Het is namelijk zo dat de kleur wit uit alle andere kleuren is opgebouwd. Voeg de volgende regel maar eens toe en kijk dan nog eens wat het effect is:

```
35 POKE $FE03,$1
```

U zult dan zien dat de kleur blauw gescrolld wordt en dat de kleur rood achterblijft en op de plaatsen waar rood en blauw over elkaar heenkomen is de kleur wit weer te zien. Het omgekeerde (rood scrollen en blauw laten staan) kan gedaan worden door regel 35 te veranderen in:

```
35 POKE $FE03,$2
```

Dit verschijnsel zal later in dit hoofdstuk nog een aantal malen terugkeren. Dan worden ook de mogelijkheden in MODE M2 bekeken, daarin zijn 16 kleuren tegelijk te gebruiken. Door nu te gaan scrollen in MODE M2, kan aangetoond worden dat de 16 kleuren opgebouwd zijn uit 4 hoofdkleuren. Dat brengt met zich mee dat alle 16 kleuren gescrolld worden, wanneer de 4 hoofdkleuren gescrolld worden. Dat zal verderop uitvoerig besproken worden.

Programma 2: Naar boven scrollen in 1 kleur.

Het programma voor het naar boven scrollen is korter dan het programma voor het opzij scrollen en tevens zit het programma iets gemakkelijker in elkaar.

```
10 INIT"CRT:M1" :LIMIT $FE00
20 DATA $DB,$E0,$3E,$83,$D3,$CC,$D3,$CD,
$21,$00,$80,$11,$D8,$7F,$01,$40,$1F,$ED,
$B0,$11,$18,$9F,$21,$D8,$7F,$01,$28,$00,
$ED,$B0,$DB,$E1,$C9
30 FOR A=0 TO 32 :READ B :POKE $FE00+A,B
 :NEXT A
40 LIST :LIST :LIST
50 FOR T=1 TO 200 :USR($FE00) :NEXT T
60 GET A$ :IF A$="" THEN 60
70 CLS :END
```

Uit deze twee programmaatjes is heel gemakkelijk af te leiden hoe het scrollen naar rechts en beneden gaat, daarvoor zijn slechts een paar kleine veranderingen nodig.
Natuurlijk is het niet noodzakelijk om elke keer het hele scherm te scrollen. Er kan ook voldaan worden met een deel van het scherm. Aan het eind van dit deel van hoofdstuk 3 zullen enkele voorbeelden volgen die een deel van het scherm scrollen.

Programma 3: Mengen van rood en blauw in mode M1.

Om aan te tonen dat in mode M1 de kleuren blauw en rood samen de kleur wit vormen, volgt hier een programma dat dat heel duidelijk aantoont. Het programma beweegt twee blokken in de kleuren blauw en rood naar elkaar toe en laat af en toe een deel van deze twee blokken elkaar overlappen. Op dat overlappingsgebied wordt de kleur dan wit.

```
10 LIMIT $FD00
20 DATA $DB,$E0,$3E,$01,$D3,$CC,$D3,$CD,
$21,$3F,$9F,$11,$3E,$9F,$0E,$C8,$7E,$08,
$06,$27,$1A,$77,$2B,$1B,$10,$FA,$08,$77,
$2B,$1B,$0D,$20,$EF,$DB,$E1,$C9
30 DATA $DB,$E0,$3E,$02,$D3,$CC,$D3,$CD,
$21,$00,$80,$11,$01,$80,$0E,$C8,$7E,$08,
$06,$27,$1A,$77,$23,$13,$10,$FA,$08,$77,
$23,$13,$0D,$20,$EF,$DB,$E1,$C9
40 FOR A=0 TO 35 :READ B :POKE $FD00+A,B
 :NEXT A
50 FOR A=0 TO 35 :READ B :POKE $FD30+A,B
 :NEXT A
60 INIT"CRT:M1"
70 BOX [1]0,0,100,125,1
80 BOX [2]219,75,319,199,2
90 FOR A=1 TO 80
100 USR($FD00) :USR($FD30) :NEXT A
110 GET A$ :IF A$="" THEN 110
120 CLS :END
```

Programma 4: Alle 16 kleuren in MODE M2 apart scrollen.

De voorgaande drie programma's werken op alle SHARPS. Voor dit programma zijn echter de uitbreidings-ic's nodig.
Zoals bekend, kunt U in de MODE M2 zestien kleuren tegelijk op het scherm gebruiken. Het is mogelijk om ieder van deze kleuren apart te scrollen. Het nadeel daarbij is dat de andere kleuren dan wel verdwijnen.

```
10 INIT"CRT:M2" :LIMIT $FE00
20 DATA $DB,$E0,$3E,$81,$D3,$CC,$D3,$CD,
$21,$00,$80,$11,$D8,$7F,$01,$40,$1F,$ED,
$B0,$11,$18,$9F,$21,$D8,$7F,$01,$28,$00,
$ED,$B0,$DB,$E1,$C9
30 FOR A=0 TO 32 :READ B :POKE $FE00+A,B
 :NEXT A
40 FOR D=1 TO 15
50 FOR A=1 TO 15 :BOX [A](A-1)*20,92,A*2
0-2,108,A :NEXT A
60 WAIT 2000 :POKE $FE03,160+D
70 FOR T=1 TO 200 :USR($FE00) :NEXT T,D
80 GET A$ :IF A$="" THEN 80
90 CLS :END
```

Programma 5: Alle vier de hoofdkleuren apart scrollen in mode M2.

Het is ook mogelijk om één van de vier hoofdkleuren te scrollen en dat kan door een kleine aanpassing in programma 4. Nu blijven wel alle andere kleuren staan. Tevens zijn hierbij ook de extra ic's nodig.
De volgende aanpassingen in programma 4 zijn daar voor nodig:

```
40 D=1
60 POKE $FE03,D
70 FOR T=1 TO 200 :USR($FE00) :NEXT T
75 IF D<8 THEN D=D*2 :GOTO 60
```

Aan de hand van dit programma kunt U precies zien uit welke hoofdkleuren de 16 kleuren opgebouwd zijn.

Programma 6: Alle vier de hoofdkleuren tegelijk scrollen in mode M2.

Het is natuurlijk ook mogelijk om alle vier de hoofdkleuren tegelijk te scrollen. Voor het gemak hebben we dat deze keer maar in het BASIC-gedeelte van het programma verwerkt. Het gaat natuurlijk een stukje sneller wanneer het in het machinetaalgedeelte verwerkt wordt, dat zal later nog een keer gedemonstreerd worden.
De volgende aanpassingen in programma 4 zijn daarvoor nodig:

```
DELETE 40.
60 FOR T=1 TO 200 :D=1
70 POKE $FE03,D :USR($FE00) :D=D*2 :IF D
<16 THEN 70
75 NEXT T
```

Programma 7: Een leuke demo.

In dit programma wordt nog eens duidelijk gemaakt dat er op een redelijk eenvoudige manier hele leuke dingen te doen zijn met het scrollen.
In dit programma wordt eerst een scherm in het geheugen ingelezen en daarna komt dat scherm stukje voor beetje weer te voorschijn op een ander scherm en verdwijnt daarna eveneens weer op een leuke manier.
Voor dit programma is het echter wel weer noodzakelijk dat U de extra ic's in Uw computer heeft.

```
10 LIMIT $DE00
20 DATA $DB,$E0,$21,$00,$80,$11,$00,$DE,
$01,$00,$20,$ED,$B0,$DB,$E1,$C9
30 DATA $DB,$E0,$3E,$85,$D3,$CC,$D3,$CD,
$21,$F0,$9E,$11,$00,$DE,$0E,$01,$06,$50,
$1A,$77,$23,$13,$10,$FA,$0D,$20,$F5,$DB,
$E1,$C9
40 DATA $DB,$E0,$21,$00,$80,$11,$01,$80,
$0E,$32,$06,$50,$1A,$77,$23,$13,$10,$FA,
$36,$00,$3E,$50,$13,$23,$3D,$20,$FB,$0D,
$20,$EC,$DB,$E1,$C9
50 DATA $DB,$E0,$21,$3F,$9F,$11,$3E,$9F,
$0E,$32,$06,$50,$1A,$77,$2B,$1B,$10,$FA,
$36,$00,$3E,$50,$1B,$2B,$3D,$20,$FB,$0D,
$20,$EC,$DB,$E1,$C9
60 FOR A=0 TO 15 :READ B :POKE $FE00+A,B
 :NEXT
70 FOR A=0 TO 29 :READ B :POKE $FE80+A,B
 :NEXT
80 FOR A=0 TO 32 :READ B :POKE $FEA0+A,B
 :NEXT
90 FOR A=0 TO 32 :READ B :POKE $FED0+A,B
 :NEXT
100 INIT"CRT:M4" :CURSOR 0,24 :PRINT" ";
 :PAL 3,0 :GH=0
110 SYMBOL 98,0,"DIT IS OOK EEN",4,2 :SY
MBOL 66,16,"MOGELIJKHEID MET",4,2
120 SYMBOL 194,32,"SCROLLEN",4,2 :SYMBOL
 130,48,"gemaakt door",4,2
130 SYMBOL 86,66,"NEPTUNES PRODUCTIONS",
3,4 :CURSOR 0,0 :PRINT" "
140 USR($FE00) :INIT"CRT:M4" :PAL 0,1 :P
AL 1,0 :PAL 2,7 :PAL 3,14
150 BOX [2]0,100,639,199,2 :B=3 :A=100
160 LINE [1]0,A,640,A :A=A+B :B=B+1 :IF
A<200 THEN 160
170 B=0 :FOR A=160 TO 0 STEP -10 :LINE [
1]A*2,100,2*(A-B),200
180 LINE [1]640-2*A,100,640-2*A+2*B,200
:B=B+30 :NEXT A
190 F=40688 :C=2
200 USR($FE80) :GH=GH+1 :F=F-80 :H=INT(F
/256) :G=((F/256)-H)*256
```

(Vervolg op de volgende bladzijde.)

```
210 POKE $FE8A,H :POKE $FE89,G :POKE $FE
8F,C :C=C+1 :IF C=101 THEN 230
220 WAIT 100-C :GOTO 200
230 FOR A=1 TO 80 :USR($FEA0) :USR($FED0
) :GH=GH+1 :NEXT A
240 GET A$ :IF A$="" THEN 240
250 INIT"CRT:M1" :END
```

Ook dit programma kan natuurlijk sneller en eventueel korter door bepaalde stukjes BASIC om te zetten in machinetaal en de machinetaal korter te maken, want het gebeurt allemaal een beetje omslachtig. Het voordeel van deze omslachtigheid is dat het machinetaalgedeelte overzichtelijker is en daardoor ook beter te begrijpen, omdat er weinig ingewikkelde machinetaalinstructies aan te pas komen.

Programma 8: Bit voor bit naar links scrollen met 1 hoofdkleur.

In principe is deze manier van scrollen precies gelijk aan het byte voor byte scrollen. Er is echter 1 verschil en dat is dat het bit voor bit scrollen veel geleidelijker gaat en daardoor ook een mooier resultaat geeft.

```
10 INIT"CRT:M1"
20 FOR A=1 TO 3 :BOX [A]140,A*30,180,(A+
1)*30-4,A :NEXT A :WAIT 2000
30 POKE $FD00,$DB,$E0,$3E,$1,$D3,$CC,$D3
,$CD,$21,$17,$9F,$D9,$21,$F0,$9E,$E,$C7,
$7E,$1F,$D9,$6,$28,$CB,$1E,$2B,$10,$FB,$
D9,$11,$D8,$FF,$19,$D,$20,$EE,$DB,$E1,$C
9
40 FOR A=1 TO 320 :USR($FD00) :NEXT A
50 GET A$ :IF A$="" THEN 50
60 CLS :END
```

Natuurlijk zijn de kleuren ook weer om te draaien door de volgende regel toe te voegen:

```
35 POKE $FD03,$2
```

Het maakt overigens niets uit wanneer U de kleur van de paletten 1 of 2 verandert, want de nieuwe kleur wordt gelijk een hoofdkleur.
Een gekke situatie wordt bereikt wanneer U de paletten 1 en 2 dezelfde kleur geeft, probeert U het maar eens uit door bijvoorbeeld de volgende regel toe te voegen:

```
15 PAL 1,6 :PAL 2,6
```

U ziet dan dat wit uit twee dezelfde kleuren opgebouwd is. U begrijpt dan misschien ook dat het mengen van kleuren in MODE M1 wel op een hele rare manier gaat die wij U helaas niet uit kunnen leggen.
In dit programma heeft U ook kunnen zien dat het bit voor bit scrollen langzamer gaat dan het byte voor byte scrollen en dat het programma iets ingewikkelder in elkaar zit, maar daar staat tegenover dat het een mooier resultaat geeft.

Programma 9: Alle kleuren tegelijk scrollen.

Dit programma werkt alleen met de uitbreidings-ic's. Het scrollen in MODE M1 gaat op een soortgelijke manier. In MODE M1 hoeven alleen de eerste twee hoofdkleuren gescrolld te worden en in MODE M2 moeten alle vier de hoofdkleuren gescrolld worden.
Het scrollen van de vier hoofdkleuren achter elkaar gebeurt in dit programma volledig in machinetaal en dat 320 maal achter elkaar (om alles precies weer op de oude plaats terug te krijgen), waardoor het iets sneller zal werken.
Ook bij bit voor bit scrollen is het natuurlijk mogelijk om 1 van de 16 kleuren apart te scrollen of 1 hoofdkleur tegelijk te scrollen, daarvoor zijn maar een paar kleine aanpassingen in dit programma nodig en die zult U langzamerhand wel zelf kunnen bedenken aan de hand van de voorbeelden die tot nu toe gegeven zijn.

```
10 INIT"CRT:M2"
20 CLS :KH=2 :FOR T=0 TO 7 :FOR U=0 TO 9
 :CURSOR U*4,T*3 :PRINT [KH]CHR$(200); :
KH=KH+1 :IF KH=16 THEN KH=1
30 NEXT U,T
40 POKE $FD00,$DB,$E0,$3E,$1,$D3,$CC,$D3
,$CD,$21,$17,$9F,$D9,$21,$F0,$9E,$E,$C7,
$7E,$1F,$D9,$6,$28
50 POKE $FD16,$CB,$1E,$2B,$10,$FB,$D9,$1
1,$D8,$FF,$19,$D,$20,$EE,$3A,$3,$FD,$87,
$32
60 POKE $FD28,$3,$FD,$FE,$10,$C2,$2,$FD,
$3E,$1,$32,$3,$FD,$1,$40,$2,$D,$ED,$43,$
35,$FD
70 POKE $FD3C,$20,$C4,$5,$ED,$43,$35,$FD
,$20,$BD,$DB,$E1,$C9
80 USR($FD00)
90 GET A$ :IF A$="" THEN 90
100 CLS :END
```

Programma 10: Tunnel.

In dit programma zal slechts een deel van het scherm gescrolld worden en wel op zo'n manier dat er een tunnel ontstaat. Dit kan natuurlijk gemakkelijk in een spel toegepast worden. Wij passen het in ieder geval toe in het spel TUNNEL.
Dit programma werkt op iedere SHARP MZ-800.

```
10 INIT"CRT:M1" :Y=81
20 BOX 88,59,231,122,3
30 POKE $FD00,$DB,$E0,$3E,$2,$D3,$CC,$D3
,$CD,$21,$2C,$93,$E,$40,$37,$3F,$6,$12
40 POKE $FD11,$CB,$1E,$2B,$10,$FB,$11,$E
A,$FF,$19,$D,$20,$F0,$DB,$E1,$C9
50 LINE [3]231,Y,231,Y+20 :USR($FD00)
60 IF RND>=.5 AND Y<100 THEN Y=Y+1
70 IF RND<.5 AND Y>62 THEN Y=Y-1
80 GOTO 50
```

Programma 11: Een deel omhoog scrollen.

Het omhoog scrollen van een deel van het scherm is niet zo gemakkelijk als het hele scherm omhoog scrollen. Het machinetaalgedeelte is even iets langer, maar komt in principe op hetzelfde neer.
Het deel dat gescrolld wordt krijgt een andere kleur dan de omgeving om het scrollen duidelijk te laten verlopen. Er is ook een kleine vertraging, omdat het scrollen anders te snel zou gaan en het resultaat daardoor minder zou worden.

```
10 INIT"CRT:M1" :LIMIT $FE00 :PAL 2,9
20 DATA $DB,$E0,$3E,$82,$D3,$CC,$3E,$02,
$D3,$CD,$21,$D8,$7F,$11,$00,$88,$06,$14,
$1A,$77,$23,$13,$10,$FA,$21,$00,$88,$11,
$28,$88,$0E,$30,$06,$14,$1A,$77,$23,$13,
$10,$FA,$06,$14,$23,$13,$10,$FC,$0D,$20,
$EF
30 DATA $11,$D8,$7F,$06,$14,$1A,$77,$23,
$13,$10,$FA,$DB,$E1,$C9
40 FOR A=0 TO 62 :READ B :POKE $FE00+A,B
 :NEXT A
50 LIST :LIST :WAIT 2000
60 FOR T=1 TO 147 :USR($FE00) :WAIT 10 :
NEXT T :WAIT 1000 :PAL 2,15
70 GET A$ :IF A$="" THEN 70
80 CLS :END
```

Dit waren twee programma's die duidelijk moeten maken hoe een deel van het scherm gescrolld kan worden. Tevens wordt in het tweede programma de mogelijkheid getoond om het te scrollen gedeelte een andere kleur te geven dan de omgeving. Nu volgt nog een leuk programmaatje dat laat zien wat het effect is als het hele scherm omhoog gescrolld wordt en de onderste helft ook nog eens naar links gescrolld wordt. Dit is tevens het laatste programma over het scrollen in de VIDEO-RAM. Hoe de VIDEO-RAM aansturing vanuit de ROM-monitor geregeld wordt, kunt U verderop in dit hoofdstuk lezen.

Programma 12: Leuk grapje.

```
10 INIT"CRT:M1" :LIMIT $FD00
20 POKE $FD00,$DB,$E0,$3E,$83,$D3,$CC,$D
3,$CD,$21,$17,$9F,$D9,$21,$F0,$9E,$E,$63
,$7E,$1F,$D9,$6,$28,$CB,$1E,$2B,$10,$FB,
$D9,$11,$D8,$FF,$19,$D,$20,$EE,$DB,$E1,$
C9
30 DATA $DB,$E0,$3E,$83,$D3,$CC,$D3,$CD,
$21,$00,$80,$11,$D8,$7F,$01,$40,$1F,$ED,
$B0,$11,$18,$9F,$21,$D8,$7F,$01,$28,$00,
$ED,$B0,$DB,$E1,$C9
40 FOR A=0 TO 32 :READ B :POKE $FE00+A,B
 :NEXT A
50 LIST :LIST
60 FOR A=1 TO 200 :USR($FD00) :USR($FE00
) :NEXT A
```

3.2 Andere leuke geintjes in de VIDEO-RAM.

In dit deel van hoofdstuk 3 zullen nog enkele programma's getoond worden die laten zien wat er zoal mogelijk is met de VIDEO-RAM. Tevens zal bij de programma's het BASIC-programma getoond worden, zodat U de snelheid van machinetaal eens goed kunt vergelijken met de snelheid van BASIC.

Programma 1: Inverse.

Onderstaand programma zet het hele beeldscherm om in inverse kleuren. Alles wat zwart is, wordt wit en omgekeerd en ga zo maar door.

```
10 POKE $F800,$DB,$E0,$21,$0,$80,$E,$C8,
$6,$28,$16,$4,$3E,$1,$D3,$CC,$D3,$CD,$87
,$32,$C,$F8,$7E,$2F,$77,$15,$20,$F0,$23,
$3E,$1,$32,$C,$F8,$10,$E6,$D,$20,$E1,$DB
,$E1,$C9
20 USR($F800)
```

Inverse in BASIC:

```
10 FOR Y=0 TO 199
20 FOR X=0 TO 319
30 SET [3-POINT(X,Y)]X,Y
40 NEXT X,Y
```

Wanneer U de uitbreidings-ic's heeft en in mode M2 het programma RUNt, kunt U regel 30 veranderen in:

```
30 SET [15-POINT(X,Y)]X,Y
```

Programma 2: Regen.

Dit programma laat alles wat op het scherm staat verdwijnen door op een willekeurige plaats op het scherm elke keer een stukje te laten verdwijnen. Hierdoor lijkt het er een klein beetje op of het regent, vandaar dat het programma 'Regen' wordt genoemd.

```
10 COLOR 0 :CURSOR 0,0 :PRINT CHR$(200);
20 DATA $DB,$E0,$11,$00,$00,$21,$00,$80,
$3E,$3C,$32,$30,$D0,$E5,$DB,$D5,$5F,$19,
$36,$FF,$E1,$11,$80,$00,$19,$3A,$30,$D0,
$3D,$20,$EB,$DB,$E1,$C9
30 FOR T=0 TO 33 :READ A :POKE $D000+T,A
 :NEXT T
40 FOR A=1 TO 2000 :USR($D000) :NEXT A
50 INIT"CRT:M1"
```

Regen in BASIC:

```
10 INIT"CRT:M1" :LIST :LIST :LIST
20 FOR T=1 TO 100000
30 X=INT(RND*40) :Y=INT(RND*200) :LINE [
0]X*8,Y,X*8+7,Y :NEXT T
```

Programma 3: Om de verticale as omgekeerd.

Dit programma keert het hele beeldscherm om. Het beeldscherm wordt eigenlijk gespiegeld in de verticale as die over het midden van het scherm loopt. Alles komt dus in spiegelbeeld op het scherm te staan.

```
10 DATA $DB,$E0,$21,$00,$80,$11,$27,$80,
$0E,$C7,$06,$14,$C5,$3E,$01,$D3,$CC,$D3,
$CD,$7E,$CD,$46,$FC,$08,$1A,$CD,$46,$FC,
$77,$08,$12,$3A,$0E,$FC,$87,$32,$0E,$FC,
$FE,$10,$20,$E3,$3E,$01,$32,$0E,$FC
20 DATA $23,$1B,$C1,$10,$D8,$06,$14,$23,
$13,$13,$13,$10,$FA,$00,$00,$00,$00,$0D,
$20,$C7,$DB,$E1,$C9,$CB,$47,$28,$2E,$CB,
$F8,$CB,$4F,$28,$2D,$CB,$F0,$CB,$57,$28,
$2C,$CB,$E8,$CB,$5F,$28,$2B,$CB,$E0,$CB,
$67,$28,$2A,$CB,$D8,$CB,$6F,$28,$29,$CB,
$D0,$CB,$77,$28
30 DATA $28,$CB,$C8,$CB,$7F,$28,$27,$CB,
$C0,$78,$C9,$CB,$B8,$C3,$4C,$FC,$CB,$B0,
$C3,$52,$FC,$CB,$A8,$C3,$58,$FC,$CB,$A0,
$C3,$5E,$FC,$CB,$98,$C3,$64,$FC,$CB,$90,
$C3,$6A,$FC,$CB,$88,$C3,$70,$FC,$CB,$80,
$C3,$76,$FC
40 FOR T=0 TO 159 :READ A :POKE $FC00+T,
A :NEXT T
50 USR($FC00)
```

Verticaal omgekeerd in BASIC:

```
10 INIT"CRT:M1"
20 COLOR 2 :LIST
30 COLOR 3 :LIST
40 FOR Y=0 TO 199
50 FOR X=0 TO 159
60 A=POINT(X,Y) :B=POINT(319-X,Y)
70 SET [B]X,Y :SET [A]319-X,Y
80 NEXT X,Y
```

Programma 4: Om de horizontale as omgekeerd:

In dit programma wordt het beeldscherm gespiegeld in de horizontale as die midden over het scherm loopt. Een leuk effect wordt bereikt wanneer dit programma gecombineerd wordt met programma 3.

```
10 DATA $DB,$E0,$11,$18,$9F,$21,$00,$80,
$0E,$64,$06,$28,$C5,$3E,$01,$D3,$CC,$D3,
$CD,$7E,$08,$1A,$08,$12,$08,$77,$3A,$0E,
$FC,$87,$32,$0E,$FC,$FE,$10,$20,$E8,$3E,
$01,$32,$0E,$FC,$23,$13,$C1,$10,$DD,$06,
$50,$1B,$10,$FD,$0D,$20,$D3,$DB,$E1,$C9
40 FOR T=0 TO 57 :READ A :POKE $FC00+T,A
 :NEXT
50 USR($FC00)
60 CURSOR 0,23
```

Horizontaal omgekeerd in BASIC:

```
10 INIT"CRT:M1"
20 COLOR 2 :LIST
30 COLOR 3 :LIST
40 FOR Y=0 TO 99
50 FOR X=0 TO 319
60 A=POINT(X,Y) :B=POINT(X,199-Y)
70 SET [B]X,Y :SET [A]X,199-Y
80 NEXT X,Y
```

Programma 5: (Willekeurige) kleuren.

Dit programma geeft elke byte een min of meer willekeurig gekozen kleur. De kleur is niet helemaal willekeurig, omdat een computer een logisch apparaat is en daardoor niet iets willekeurig kan kiezen. Door iets extra's in het programma te zetten, wordt de willekeurigheid iets meer benaderd dan zonder dat extra's. Voeg de volgende regel maar eens toe en bekijk het verschil:

```
35 POKE $D010,0,0,0
```

U ziet duidelijk dat de willekeurigheid voor het grootste deel verdwenen is.
Dit programma draait overigens alleen met de extra ic's.

```
10 INIT"CRT:M1"
20 DATA $DB,$E0,$21,$00,$80,$0E,$C8,$06,
$28,$DB,$D5,$D3,$CC,$36,$FF,$23,$3D,$20,
$FD,$10,$F4,$0D,$20,$EF,$DB,$E1,$C9
30 FOR T=0 TO 26 :READ A :POKE $D000+T,A
 :NEXT T
40 USR($D000)
```

Willekeurige kleuren in BASIC.

```
10 INIT"CRT:M2"
20 FOR Y=0 TO 199
30 FOR X=0 TO 319 STEP 8
40 LINE [RND*16]X,Y,X+7,Y
50 NEXT X,Y
```

Vooruitlopend op programma 7, willen wij U het volgende alvast laten zien:
Het is mogelijk om in mode M1 13 kleuren te gebruiken, dat kan in dit programma door de volgende regels toe te voegen aan het machinetaalprogramma:

```
15 INIT"CRT:M1" :OUT@$CE,2 :CLS
```

Om weer terug te springen naar 4 kleuren, voert U het volgende in:

```
OUT@$CE,0
```

In de volgende programma's gaan we verder hierop in.

Mensen die de extra VIDEO-RAM ic's in hun computer hebben kunnen 16K extra geheugen gebruiken in de modes M1 en M3. Dat is gebaseerd op het feit dat er in die modes twee FRAMES zijn voor de kleuren. In beide FRAMES zit(ten) 2, respectievelijk 1 hoofdkleur(en). Omdat er in deze modes slechts 1 FRAME gebruikt en gezien wordt, kan het andere FRAME gebruikt worden om iets in op te slaan. Het tweede FRAME kan zo geheel gebruikt worden, zonder dat er iets van te zien is op het beeldscherm. Er zal nu een programma volgen dat gebruik maakt van deze methode.

Programma 6: Regen met terugkeer van het scherm.

Dit programma doet hetzelfde als programma 2, met als verschil dat nu het hele verdwenen scherm weer terugkomt. In het begin wordt het hele scherm eerst in het andere FRAME opgeslagen. Daarna gaat het scherm verregenen en daarna wordt alles van het tweede FRAME naar het eerste FRAME overgezet, waardoor het oorspronkelijke scherm weer te zien is. Het voorbeeld van schermopvulling (PRINT"NEPTUNES PRODUCTIONS") is uiteraard willekeurig gekozen. Het maakt niets uit wat er op het scherm staat, alles komt weer in de oorspronkelijke staat terug en er wordt GEEN extra normaal geheugen gebruikt alleen een stuk ongebruikt VIDEO-RAM geheugen.

```
10 INIT"CRT:M1"
20 DATA $DB,$E0,$3E,$80,$D3,$CC,$11,$00,
$00,$21,$00,$80,$3E,$3C,$32,$30,$D0,$E5,
$DB,$D5,$5F,$19,$36,$FF,$E1,$11,$80,$00,
$19,$3A,$30,$D0,$3D,$20,$EB,$DB,$E1,$C9
30 FOR T=0 TO 37 :READ A :POKE $D000+T,A
 :NEXT T
40 CURSOR 0,9
50 COLOR 1 :PRINT"   ****** NEPTUNES PRO
DUCTIONS ******   "
60 COLOR 2 :PRINT"   ****** NEPTUNES PRO
DUCTIONS ******   "
70 COLOR 3 :PRINT"   ****** NEPTUNES PRO
DUCTIONS ******   "
80 DATA $DB,$E0,$3E,$04,$D3,$CC,$3E,$01,
$D3,$CD,$21,$00,$80,$11,$00,$80,$01,$40,
$1F,$ED,$B0,$DB,$E1,$C9
90 DATA $DB,$E0,$3E,$08,$D3,$CC,$3E,$02,
$D3,$CD,$21,$00,$80,$11,$00,$80,$01,$40,
$1F,$ED,$B0,$DB,$E1,$C9
100 FOR A=0 TO 23 :READ B :POKE $FD00+A,
B :NEXT A
110 FOR A=0 TO 23 :READ B :POKE $FD20+A,
B :NEXT A
120 USR($FD00) :USR($FD20)
130 FOR A=1 TO 2000 :USR($D000) :NEXT
140 POKE $FD03,1 :POKE $FD07,4 :POKE $FD
23,2 :POKE $FD27,8 :USR($FD00) :USR($FD2
0)
```

Zo bestaat ook de mogelijkheid om 13 kleuren te gebruiken in mode M1. Dat wordt gedaan door zowel het eerste als het tweede FRAME zichtbaar te maken. Hoe dat gaat wordt in het volgende programma getoond.

Programma 7: 13 kleuren in mode M1.

Zoals al twee keer eerder gememoreerd, bestaat de mogelijkheid om in mode M1 13 verschillende kleuren te tonen, dat is mogelijk door FRAME A en FRAME B tegelijk te laten zien in deze mode. Eigenlijk houdt U de computer voor de gek. Alles werkt zoals in mode M1 behalve de VIDEO-RAM, die bevindt zich in mode M2.
Waarom zijn er maar 13 kleuren en geen 16 verschillende kleuren? Dat heeft te maken met het feit dat in mode M1 de kleuren rood en blauw de kleur wit opleveren (ook na de verandering) en in mode M2 levert deze kleurcombinatie de kleur paars op. In mode M1 leveren ook alle vier de hoofdkleuren bij elkaar de kleur wit op, evenals in mode M2. Alles bij elkaar is dit behoorlijk ingewikkeld. Hopelijk begrijpt U er iets van.
Om het programma niet al te lang te maken, wordt het veld alvast opgebouwd in mode M2 en daarna is het in mode M1 weer te zien. Ook zijn niet alle 13 kleuren te zien. Met een klein stukje machinetaal zijn wel alle 13 mogelijkheden zichtbaar te maken. Dat doen wij hier niet, omdat het alleen de bedoeling is om te laten zien dat er meer dan 4 kleuren te gebruiken zijn in mode M1.
Uiteraard werkt dit programma ook alleen met de extra ic's.

```
10 INIT"CRT:M2"
20 FOR A=0 TO 15
30 BOX [A]A*20,A*12,A*20+15,A*12+8,A
40 NEXT A
50 INIT"CRT:M1"
60 FOR A=1 TO 15 STEP 2
70 BOX [1]A*20,A*12,A*20+15,A*12+8,1
80 NEXT A
90 FOR A=2 TO 14 STEP 4
100 BOX [2]A*20,A*12,A*20+15,A*12+8,2
110 OUT@$CE,$2
120 NEXT A
```

In regel 110 ziet U de instructie OUT@$CE,$2. Met deze instructie kunt U nog veel meer leuke effecten bereiken. Het is namelijk de instructie die bepaalt hoe het geheugen en ook het VIDEO-RAM gedeelte ingedeeld worden. Een leuk effect wordt bijvoorbeeld bereikt wanneer de computer in mode M3 of M4 staat en er met OUT@$CE,n een VIDEO-RAM indeling wordt gemaakt in mode M1 of M2. Probeert U het volgende maar eens uit:

```
INIT"CRT:M3" :OUT@$CE,0  - Dit kan ook wanneer U de extra ic's niet
                           heeft.
```

of

```
INIT"CRT:M4" :OUT@$CE,2 - Dit kan alleen wanneer U de extra ic's heeft.
```

U ziet dan dat de computer naar de 40-koloms mode gaat en de letters een keer zo lang worden. Het probleem is echter dat bij het intypen een regel nog wel steeds 80 tekens kan bevatten. (Probeert U het eens uit.) Een ander probleem is dat het geheugen anders ingedeeld wordt en U daardoor U listing kwijt kunt raken. Het zou natuurlijk heel erg leuk zijn als op deze manier op een scherm gewerkt kan worden, maar dat is helaas (bijna) onmogelijk.

3.3: VIDEO-RAM aansturing via de ROM-monitor.

De aansturing van de VIDEO-RAM via de ROM-monitor gaat ook via bankswitsching. Het scherm verandert wel wanneer U gaat bankswitschen, omdat er van de 700-mode naar een 800-mode gesprongen wordt en daardoor de beeldschermindeling ook geheel anders wordt. Dat is niet het geval wanneer U vanuit de BASIC gaat bankswitschen, omdat de computer dan al in een 800-mode staat.
In hoofdstuk 2 heeft U al twee programma's kunnen zien die iets in de VIDEO-RAM doen via de ROM-monitor.
In dit deel van hoofdstuk 3 zullen nog enkele programma's volgen die iets in de VIDEO-RAM doen en tevens zal de assemblerlisting met uitleg er bij staan, omdat het dan veel overzichtelijker is en het hoofdstuk over assembleren al vol genoeg staat.

Programma 1: Alfabet-scroll.

Dit programma laat het hele alfabet van onder naar boven over het beeldscherm gaan.

```
6000 DB   600B 03   6016 23   6021 00   602C 77   6037 11   6042 23
6001 E0   600C D3   6017 0B   6022 00   602D 00   6038 00   6043 13
6002 3E   600D CC   6018 78   6023 00   602E 13   6039 80   6044 0B
6003 00   600E 21   6019 B1   6024 00   602F 00   603A 21   6045 78
6004 D3   600F 00   601A C2   6025 11   6030 00   603B D8   6046 B1
6005 CE   6010 80   601B 14   6026 30   6031 00   603C 7F   6047 C2
6006 3E   6011 01   601C 60   6027 5F   6032 00   603D 01   6048 40
6007 00   6012 40   601D 3E   6028 21   6033 7B   603E 40   6049 60
6008 D3   6013 1F   601E 37   6029 19   6034 32   603F 1F   604A C3
6009 F0   6014 AF   601F D3   602A 9F   6035 26   6040 1A   604B 25
600A 3E   6015 77   6020 F0   602B 1A   6036 60   6041 77   604C 60
```

```
Filename? A-Z SCROLL
Top adrs? 6000
End adrs? 604C
Exc adrs? 6000
Opstarten met G6000.
Dit programma is alleen te stoppen door op de resetknop te drukken.
```

Assemblerlisting van programma 1:

```
6000 IN A,(E0H)
6002 LD A,00H
6004 OUT (CEH),A
6006 LD A,00H
6008 OUT (F0H),A
600A LD A,03H
600C OUT (CCH),A
600E LD HL,8000H
6011 LD BC,1F40H
6014 EEN:  XOR A
6015 LD (HL),A
6016 INC HL
6017 DEC BC
6018 LD A,B
6019 OR C
601A JP NZ,EEN
601D LD A,37H
601F OUT (F0H),A
6021 LD A,00H
6023 LD A,00H
6025 TWEE:  LD DE,5F30H
6028 LD HL,9F19H
602B LD A,(DE)
602C LD (HL),A
602D NOP
602E INC DE
```

```
602F NOP
6030 NOP
6031 NOP
6032 NOP
6033 LD A,E
6034 LD (6026H),A
6037 LD DE,8000H
603A LD HL,7FD8H
603D LD BC,1F40H
```

```
6040 DRIE:  LD A,(DE)
6041 LD (HL),A
6042 INC HL
6043 INC DE
6044 DEC BC
6045 LD A,B
6046 OR C
6047 JP NZ,DRIE
604A JP TWEE
```

Uitleg van programma 1:

Allereerst wordt er gebankswitschd op adres 6000.
Daarna wordt de computer in één van de 800-modes gezet op de adressen 6002 t/m 6005.
Vervolgens wordt palet 0 met de kleur 0 geladen. Palet 0 wordt dus zwart. Via de poort F0H kunt U de kleur van een palet bepalen. In het adres A komt zowel de palet als de kleur te staan. Het eerste getal in A is het palet en het tweede getal is de kleur die het palet krijgt. Dit alles gebeurt op de adressen 6006 t/m 6009.
Op de adressen 600A t/m 600D wordt de palet bepaald waarmee gewerkt zal worden en dat is de palet 3.
Vanaf 600E t/m 601C wordt het hele scherm schoongemaakt.
Vervolgens wordt palet 3 met de waarde 7 (licht-grijs) geladen op de adressen 601D t/m 6020.
Daarna volgt een loos stukje.
Vanaf 6025 t/m 6034 wordt onderaan het scherm 1/8 deel van een letter getekend en vanaf 6035 wordt het scherm 1 regel omhoog gescrolld. Deze cyclus wordt herhaald tot er op de resetknop wordt gedrukt.
In het programma zitten ook een aantal loze stukjes die te herkennen zijn aan de instructie NOP. Het programma stond al op papier toen we er achter kwamen dat deze stukjes niet nodig waren, vandaar dat er loze stukjes in zitten.
In het hoofdstuk over assembleren gaan we nog iets dieper in op de VIDEO-RAM.

Programma 2: 13 rijtjes van 16 kleuren.

Dit programma trekt 16 lijntjes in 16 verschillende kleuren achter elkaar en herhaald dat 13 keer. Dit programma werkt dus eigenlijk alleen met de extra ic's. Wanneer U echter geen ic's heeft, werkt dit programma ook. U ziet dan slechts 4 kleuren ipv 16.
U begrijpt nu waarschijnlijk ook waarom veld drie van Tut-Ench-Amun met ic's wel te zien is en zonder ic's niet. Dit heeft te maken met het feit dat in veld 3 meer dan 4 kleuren gebruikt worden.

```
2000 DB   2008 D3   2010 AF   2018 20   2020 D3   2028 FB   2030 ED
2001 E0   2009 CC   2011 77   2019 21   2021 CC   2029 3C   2031 C3
2002 3E   200A 21   2012 23   201A 00   2022 06   202A FE   2032 31
2003 02   200B 00   2013 0B   201B 80   2023 28   202B 90   2033 20
2004 D3   200C 80   2014 78   201C 0E   2024 36   202C 20
2005 CE   200D 01   2015 B1   201D 0D   2025 FF   202D F2
2006 3E   200E 40   2016 C2   201E 3E   2026 23   202E 0D
2007 03   200F 1F   2017 10   201F 80   2027 10   202F 20
```

```
Filename? KLEURRIJEN
Top adrs? 2000
End adrs? 2033
Exc adrs? 2000
Opstarten met G2000.
```
Dit programma is alleen te stoppen door op de resetknop te drukken.

Assemblerlisting van programma 2:

```
2000 IN A,(E0H)
2002 LD A,02H
2004 OUT (CEH),A
2006 LD A,03H
2008 OUT (CCH),A
200A LD HL,8000H
200D LD BC,1F40H
2010 EEN:  XOR A
2011 LD (HL),A
2012 INC HL
2013 DEC BC
2014 LD A,B
2015 OR C
2016 JP NZ,EEN
2019 LD HL,8000H
201C LD E,0DH
201E TWEE:  LD A,80H
2020 DRIE:  OUT (CCH),A
2022 LD B,28H
2024 VIER:  LD (HL),FFH
2026 INC HL
2027 DJNZ VIER
2029 INC A
202A CP 90H
202C JR NZ,DRIE
202E DEC C
202F JR NZ,TWEE
2031 VIJF:  JP VIJF
```

Uitleg van programma 2:

In het begin wordt eerst gebankswitscht en vervolgens wordt er omgeschakeld naar een 800-mode en wel de mode waarin 16 kleuren te gebruiken zijn.
Daarna wordt het beeldscherm 'schoongemaakt'. Eigenlijk heeft dit helemaal geen nut, maar dat is niet van belang.
Daarna worden 14 keer 16 rijtjes met verschillende kleuren getrokken.

Mogelijkheden via de poort CEH.

U heeft kunnen zien dat via de poort CEH bepaald kan worden hoe het beeldscherm ingedeeld wordt en hoeveel kleuren er te gebruiken zijn. Hieronder zullen alle mogelijkheden via de poort CEH besproken worden. Alle mogelijke waardes en de bijbehorende schermindeling worden getoond. U moet de waardes dus uit poort CEH sturen om dat effect te bereiken.

00H - De schermindeling wordt 320*200 en U kunt 4 kleuren gebruiken.
01H - De schermindeling wordt 320*200 en U kunt 4 kleuren gebruiken, maar nu worden, indien aanwezig, de extra ic's gebruikt.
02H - De schermindeling wordt 320*200 en U kunt 16 kleuren gebruiken. Dit geldt alleen wanneer U de extra ic's heeft.
04H - De schermindeling wordt 640*200 en U kunt 1 kleur gebruiken.
05H - De schermindeling wordt 640*200 en U kunt 1 kleur gebruiken, maar nu worden, indien aanwezig, de extra ic's gebruikt.
06H - De schermindeling wordt 640*200 en U kunt 4 kleuren gebruiken. Dit geldt alleen wanneer U de extra ic's heeft.
08H - De schermindeling wordt 40*25 en U kunt 8 kleuren gebruiken. Dit is dus de 700-mode.

HOOFDSTUK 4: POKES, PEEKS, TRUCS, TIPS en allerlei andere fratsen.

In dit hoofdstuk zal getoond worden dat er allerlei leuke geintjes met de computer uit te halen zijn en die geintjes kunt U vaak op een niet al te moeilijke wijze zelf ontdekken.

4.1: Pokes en Peeks.

Met de POKE-instructie kan de waarde van een bepaald adres of bepaalde adressen in de BASIC verandert worden. U kunt dus in principe ieder adres POKEN, dat betekent dat er in totaal 65535 POKES uitgevoerd kunnen worden. Helaas heeft 99% van deze POKES een effect op de computer dat niet de moeite van het vermelden waard is. Zo bestaan er bijvoorbeeld een paar duizend POKES om de BASIC vast te laten lopen, maar wat heeft U daar nu aan? Ook kunnen alle BASIC-instructies uitgeschakeld worden door op het juiste adres de waarde C9 te POKEN, maar wat heeft dat voor nut?
Via de BASIC-monitor kunt U zelf een paar leuke POKES ontdekken. U moet dan wel weten waar U moet zoeken en waarnaar U moet zoeken. Zo zou U er bijvoorbeeld achter kunnen komen dat de schermopbouw gebeurt via o.a. het B-register en het DE-register. 1 regel bevat 40 (28 hexadecimaal) tekens en dat aantal wordt opgeslagen in het DE-register. Er staat dan in het geheugen 11 28 00. Waar dat precies staat, kunt U opzoeken met F (Find). Verder moet U nog weten dat het grootste gedeelte van de directe handelingen van de computer in het gebied van 0000H t/m 1000H plaatsvinden. In dat gebied gaan we dus zoeken naar 11 28 00 en dat gaat als volgt:

```
BYE <cr>  (om in de BASIC-MONITOR te komen)
*F00001000112800
```

U krijgt dan het volgende te zien:

```
:05E5-11 28 00 /.(.
:071F-11 28 00 /.(.
:07C2-11 28 00 /.(.
:0829-11 28 00 /.(.
:0897-11 28 00 /.(.
```

Nu gaan we een beetje experimenteren en dat doen we bijvoorbeeld door 28 te vervangen door 50 en te kijken of er iets verandert. Verandert er niets dan zetten we de 50 weer terug op 28.
Veranderen we de eerste 28 in 50 en typen daarna een paar letters in dan zien we dat de letters op dubbele grootte op het scherm komen en kijk daar: we hebben al een leuke POKE gevonden. Nu moeten we het alleen nog omzetten in een programma en dat kan er dan bijvoorbeeld zo uit komen te zien:

```
10 INIT"CRT:M1"
20 PRINT"NORMALE GROOTTE."
30 POKE $5E6,$50
40 PRINT
50 PRINT"DUBBELE GROOTTE."
60 POKE $5E6,$28 :CURSOR 0,6
```

U kunt op een dergelijke manier natuurlijk nog veel meer POKES vinden, maar dat idee hebben wij als makers van dit boek ook al gehad en hebben op die manier de meeste leuke POKES al gevonden. De absolute knaller die we gevonden hebben is het volgende:
Met twee POKES is het mogelijk om het beeldscherm naar boven te scrollen zoals dat ook bij de hardwarematige oplossing voor het scrollen in de VIDEO-RAM gebeurt. Om de werking van deze POKES te kunnen demonstreren zijn ze in een programma verwerkt.

```
10 INIT"CRT:M1"
20 LIST :LIST
30 WAIT 2000
40 CURSOR 0,24
50 POKE $898,$5
60 POKE $97E,$01
70 FOR T=1 TO 200 :PRINT :NEXT T
80 POKE $898,$28
90 POKE $97E,$00
100 WAIT 2000 :INIT"CRT:M1"
```

Deze POKES werken zowel in QD-BASIC als in cassette-BASIC, dat is ook het geval met de POKE van de vorige bladzijde. Om elke keer verwarringen te voorkomen, zal bij de rest van de POKES in dit hoofdstuk zowel de POKE in QD-BASIC als de overeenkomende POKE in cassette-BASIC vermeld staan ook al zijn deze twee gelijk.

```
POKE NR.1 : QD-BASIC       :POKE $109B,x   (0 =< x =< 255)
            cassette-BASIC :POKE $109B,x
            Omschrijving   :Is gelijk aan COLOR x.
            Oorspronkelijk :POKE $109B,$3   ($F in mode M2)

POKE NR.2 : QD-BASIC       :POKE $5E6,$D8,$FF
            cassette-BASIC :POKE $5E6,$D8,$FF
            Omschrijving   :Omkeren van de tekst.
            Oorspronkelijk :POKE $5E6,$28,$0
            Toepassing     : 10 INIT"CRT:M1"
                             20 PRINT"DEZE TEKST ZIET ER OP Z'N KOP A
                             LS VOLGT UIT:"
                             30 PRINT :PRINT
                             40 POKE $5E6,$D8,$FF
                             50 PRINT"DEZE TEKST ZIET ER OP Z'N KOP A
                             LS VOLGT UIT:"
                             60 POKE $5E6,$28,$0

POKE NR.3 : QD-BASIC       :POKE $5E4,x   (1 =< x =< 8)
            cassette-BASIC :POKE $5E4,x
            Omschrijving   :Deel van tekst wordt getoond.
            Oorspronkelijk :POKE $5E4,$8
            Toepassing     : 10 INIT"CRT:M1"
                             20 FOR G=1 TO 8
                             30 CURSOR 0,0 :POKE $5E4,G
                             40 PRINT"DIT IS EEN VOORBEELD"
                             50 PRINT"VAN NEPTUNES SOFTWARE."
                             60 WAIT 1000
                             70 NEXT G
```

```
POKE NR.4 : QD-BASIC        :POKE $3919,$1
            cassette-BASIC :POKE $3919,$1
            Omschrijving   :Programma wordt slechts 1 keer gesaved,
                            zodat het saven sneller gaat.
            Oorspronkelijk :POKE $3919,$2

POKE NR.5 : QD-BASIC        :POKE $5D4,$1
            cassette-BASIC :POKE $5D4,$1
            Omschrijving   :Omschakelen naar tweede characterset.
            Oorspronkelijk :POKE $5D4,$0

POKE NR.6 : QD-BASIC        :POKE $4DD0,$1
            cassette-BASIC :POKE $4DCF,$1
            Omschrijving   :Omschakelen naar 2e ch.set bij SYMBOL.
            Oorspronkelijk :POKE $4DD0,$0 / POKE $4DCF,$0
            Toepassing     :POKE $4DD0,$1 :SYMBOL 0,99,"NEPTUNES",3,3
                            :POKE $4DD0,$0
                            POKE $4DCF,$1 :SYMBOL 0,99,"NEPTUNES",3,3
                            :POKE $4DCF,$0

POKE NR.7 : QD-BASIC        :POKE $55B0,$CD,$3E,$0,$11,$F0,$63,$C9 :
                            POKE $587A,$CD,$B0,$55
            cassette-BASIC :POKE $55B0,$CD,$3E,$0,$11,$F0,$63,$C9 :
                            POKE $587A,$CD,$B0,$55
            Omschrijving   :BEEP-toon bij Ready-melding.
            Oorspronkelijk :POKE $55B0,$0,$0,$0,$0,$0,$0,$0 :
                            POKE $587A,$11,$F0,$63

POKE NR.8 : QD-BASIC        :POKE $B34,x   (0 <= x <= 100)
            cassette-BASIC :POKE $B34,x
            Omschrijving   :Cursorsnelheid veranderen.
            Oorspronkelijk :POKE $B34,$10

POKE NR.9 : QD-BASIC        :POKE $6E35,0,0
            cassette-BASIC :POKE $6E35,0,0
            Omschrijving   :INPUT zonder vraagteken.
            Oorspronkelijk :POKE $6E35,$3F,$20
            Toepassing     :POKE $6E35,0,0 :INPUT A :POKE $6E35,$3F,
                            $20

POKE NR.10 : QD-BASIC        :POKE $1099,$1
             cassette-BASIC :POKE $1099,$1
             Omschrijving   :Wanneer U geen uitbreidings-IC's in Uw
                             computer heeft, kunt U nu wel INIT"CRT:M2"
                             en INIT"CRT:M4" invoeren. Wanneer U de
                             IC's heeft, heeft deze POKE dus geen nut.
             Oorspronkelijk :POKE $1099,$0 - Wanneer U geen IC's heeft.
                             POKE $1099,$1 - Wanneer U wel IC's heeft.

POKE NR.11 : QD-BASIC        :POKE $6879,$0 :POKE $688E,$0
             cassette-BASIC :POKE $6879,$0 :POKE $688E,$0
             Omschrijving   :met GET herhaaldelijk uitlezen.
             Oorspronkelijk :POKE $6879,$FF :POKE $688E,$FF
             Toepassing     :10 POKE $6879,$0 :POKE $688E,$0
                             20 GET A$ :PRINT A$ :GOTO 20
```

```
POKE NR.12 : QD-BASIC        :POKE $4678,$80
             cassette-BASIC  :POKE $4678,$80
             Omschrijving    :Bij SYMBOL wordt de achtergrond niet
                              gewist.
             Oorspronkelijk  :POKE $4678,$C0
             Toepassing      : 10 INIT"CRT:M1"
                               20 A$="NEPTUNES SOFTWARE."
                               30 POKE $4678,$80 :FOR A=319 TO -1130 ST
                               EP -8 :SYMBOL [2]A,92,A$,8,2
                               40 WAIT 25 :NEXT A
                               50 POKE $4678,$C0

POKE NR.13 : QD-BASIC        :POKE $6F29,$89
             cassette-BASIC  :POKE $6F29,$89
             Omschrijving    :PRINT wordt PRINT/P.
             Oorspronkelijk  :POKE $6F29,$88

POKE NR.14 : QD-BASIC        :POKE $6079,$0,$0,$0
             cassette-BASIC  :POKE $6079,$0,$0,$0
             Omschrijving    :Variabelen worden niet meer gewist bij
                              RUN.
             Oorspronkelijk  :POKE $6079,$D,$9B,$61

POKE NR.15 : QD-BASIC        :POKE $4E9,$0,$0,$0
             cassette-BASIC  :POKE $4E9,$0,$0,$0
             Omschrijving    :Geen CLS bij CONSOLE.
             Oorspronkelijk  :POKE $4E9,$CD,$52,$6

POKE NR.16 : QD-BASIC        :POKE $5592,$3A,$83,$6B,$CD,$6,$0,$C9 :
                              POKE $6B20,$CD,$92,$55
             cassette-BASIC  :POKE $55BA,$3A,$83,$6B,$CD,$6,$0,$C9 :
                              POKE $6B20,$CD,$BA,$55
             Omschrijving    :SEARCH en LIST met vrije tussenregel.
             Oorspronkelijk  :POKE $5592,$0,$0,$0,$0,$0,$0,$0 :
                              POKE $6B20,$3A,$83,$6B
                              POKE $55BA,$0,$0,$0,$0,$0,$0,$0 :
                              POKE $6B20,$3A,$83,$6B
```

Zo zijn er nog veel en veel meer POKES. Hieronder en op de volgende bladzijden zullen nog een paar volgen. Daarna zal ook nog een lijst van nutteloze POKES gegeven worden. De POKES die hier besproken zijn, zijn zo'n beetje de leukste en handigste POKES die momenteel bekend zijn. Natuurlijk zijn er veel meer, maar ze zijn nog niet allemaal bekend of niet de moeite van het vermelden waard. Zo zijn er bijvoorbeeld de ANTI-BREAK pokes die heel erg aardig zijn, maar verder totaal geen nut hebben omdat een BASIC-programma niet voor 100% te beveiligen is. (Tenzij het programma op QD staat. Dan is het wel te beveiligen tegen overname, maar dat wordt een dure aangelegenheid.)

POKES om de save- en laadsnelheid van de cassette op te voeren:
(Deze POKES werken zowel in QD- als in cassette-BASIC.)

```
POKE $3B91,$23 :POKE $3B97,$B :POKE $3B9B,$31
Oorspronkelijk:
POKE $3B91,$4C :POKE $3B97,$18 :POKE $3B9B,$69
```

POKES voor CURSOR X,Y:

Onderstaande POKES werken zowel in QD- als cassette-BASIC en laten zien dat met twee POKES hetzelfde effect bereikt kan worden als met CURSOR X,Y.

```
10 INIT"CRT:M1"
20 FOR B=0 TO 23
30 FOR A=0 TO 39
40 WAIT 25
50 POKE 4226,A :POKE 4227,B :PRINT"  NEP
TUNES." :NEXT A,B
```

POKES om teken van cursor te veranderen:
(Geldt voor beide BASICS.)

```
POKE $1391,$FF,$81,$81,$81,$81,$81,$81,$FF
```
Oorspronkelijk:
```
POKE $1391,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

POKES om aantal sprongen bij indrukken van TAB-toets te veranderen:
(Geldt voor beide BASICS.)

```
10 INIT"CRT:M1"
20 INPUT"WAT WORDT DE NIEUWE TAB-GROOTTE
?  (1-39)";A
30 IF A<1 OR A>39 THEN 20
40 POKE $A2D,A :POKE $A32,A :POKE $A49,A
 :POKE $A5F,A :POKE $A63,A
50 PRINT :PRINT"DRUK NU EENS OP DE TAB-T
OETS!"
```

<u>Een aantal nutteloze POKES:</u>

ANTI-BREAK POKES:
(Gelden voor beide BASICS.)

```
POKE $F4C,$6E,$60 :POKE $6403,$C3,$0,$0
```
Oorspronkelijk:
```
POKE $F4C,$DA,$0 :POKE $6403,$ED,$7B,$64
```

Andere nutteloze pokes voor beide BASICS:

```
POKE $6A9D,$C9 - LIST uitgeschakeld.          Normaal: POKE $6A9D,$3E
POKE $6A5C,$C9 - BYE uitgeschakeld.           Normaal: POKE $6A5C,$DF
POKE $A24,$C9  - TAB-toets uitgeschakeld.     Normaal: POKE $A24,$06
POKE $6BE,$C9  - DEL-toets uitgeschakeld.     Normaal: POKE $6BE,$D9
POKE $A1E,$C9  - GRAPH-toets uitgeschakeld.   Normaal: POKE $A1E,$3E
POKE $D28,$C9  - BREAK-toets uitgeschakeld.   Normaal: POKE $D28,$D1
```
Zo bestaan er nog veel meer POKES die allemaal even nutteloos zijn en daarom worden ze niet gepubliceerd.

Na een hele rij POKES gegeven te hebben, zijn we nu aanbeland bij de PEEKS. Met PEEK is het mogelijk om de inhoud van een bepaald adres uit te lezen. Bij PEEK geldt dus precies hetzelfde als bij POKE, alle adressen zijn te PEKEN.
Er zijn maar heel weinig PEEKS die echt nuttig zijn en de belangrijksten daarvan zult U op deze bladzijde kunnen vinden.

PEEKS om uit te lezen met welk DEVICE U werkt:
(Deze PEEKS zijn eigenlijk alleen van belang wanneer U ook een QD op Uw SHARP heeft aangesloten.)

```
10 IF PEEK($106A)=$5A AND PEEK($106B)=$3
7 THEN PRINT"U WERKT MOMENTEEL MET DE CA
SSETTE."
20 IF PEEK($106A)=$D0 AND PEEK($106B)=$2
F THEN PRINT"U WERKT MOMENTEEL MET DE QD
."
```

PEEKS om uit te lezen of de uitbreidings-ic's in Uw computer aanwezig zijn:
(Geldt voor beide BASICS.)

```
PRINT PEEK($1099)
```

1 wanneer de ic's wel aanwezig zijn.
0 wanneer de ic's niet aanwezig zijn.

PEEKS om uit te lezen wat er op het scherm staat:
(Geldt voor beide BASICS en geldt alleen wanneer er gewone tekst of tekens op het scherm staat/staan.)

Vanaf adres $2000 is opgeslagen wat er op het scherm staat. Wanneer U bijvoorbeeld wilt weten of er iets staat op de eerste positie van de tweede rij, voert U in: PRINT PEEK($2000+1*40+0) en wanneer er dan iets staat, krijgt U de ASC-II code van het overeenkomstige teken te zien.

PEEKS om uit te lezen hoeveel BYTES een programma op QD in beslag neemt en hoeveel BYTES er nog over zijn:
(Alleen van belang voor bezitters van een QD.)

```
10 DIR :NO=$27D0 :P=0 :CLS :PRINT"DIR QD
:" :PRINT"-------"
30 N=NO+1 :IF PEEK(NO)=0 THEN PRINT TAB(
23);CHR$(17);65454-P;" Bytes free" :END
40 IF PEEK(NO)=1 THEN PRINT"OBJ  "; :ELS
E PRINT"BTX  ";
50 IF PEEK(N)<>13 THEN PRINT CHR$(PEEK(N
));  :N=N+1 :GOTO 50
60 B=PEEK(NO+21)*256+PEEK(NO+20) :PRINT
TAB(23);B;"  Bytes" :NO=NO+32 :P=P+B :GO
TO 30
```

4.2: IN's en OUT's.

Met de instructies INP@ en OUT@ kunt U een bepaalde poort uitlezen of een waarde tussen 0 en 255 naar een poort toe sturen. Met poorten worden o.a. de aansluiting voor de cassetterecorder, de aansluitingen voor de joysticks, maar ook de poort voor het geluid bedoeld. Natuurlijk zijn er nog meer poorten en de meest nuttige daarvan zullen uitgelezen of aangestuurd worden.

De twee JOYSTICK-poorten.

Er zijn twee joystickpoorten, dat zijn de poorten $F0 en $F1. Het aansturen van deze poorten heeft geen enkel nut, maar ze kunnen wel uitgelezen worden en wel als volgt:

```
10 INIT"CRT:M1"
20 INP@$F0,A
30 PRINT A
40 GOTO 20
```

Wanneer U een joystick aansluit op poort 1 en U gaat deze joystick bewegen, zult U zien dat de waarde van A verandert. Dit kan ook gedaan worden met poort 2, maar dan moet U regel 20 veranderen in:

```
20 INP@$F1,A
```

De cassetterecorder.

Via een bepaalde poort (poort $D2) kan uitgelezen worden of een toets (uitgezonderd de EJECT-toets) op de cassetterecorder is ingedrukt en dat gaat als volgt:

```
10 INIT"CRT:M1"
20 INP@$D2,A :IF A=152 THEN PRINT"ER IS
EEN KNOP VAN DE CASSETTERECORDER  INGEDR
UKT." :END ELSE RUN
```

Wanneer U dit programma RUNt en daarna een knop op de cassetterecorder indrukt, ziet U de melding daarvan op het scherm.

Random getallen.

Er is ook een poort om random-getallen (willekeurige getallen) uit te lezen en dat is de poort $D5. Een voorbeeld:

```
10 INIT"CRT:M1"
20 INP@$D5,A
30 WAIT 100
40 CURSOR A/255*39,24 :PRINT"O" :GOTO 20
```

Er is een wachtlus ingebracht, omdat anders de 'regelmatigheid' te groot is. Wanneer U goed kijkt, zult U zien dat er toch een bepaalde regelmatigheid in de zogenaamde willekeurige reeks zit.

80 bij 50.

Een verdeling van 80 kolommen bij 50 regels is mogelijk, alleen wordt het scherm dan wel in vier stukken verdeelt. Hoe kan dat nu precies? Dat komt omdat de poort $CE de poort is die bepaald met welk oplossend vermogen er gewerkt wordt en in welke mode (700 of 800 mode). Door deze poort aan te sturen terwijl er BASIC in de computer zit, kunnen leuke effecten ontstaan. De 80 bij 50 oplossing krijgt U door het volgende in te typen:

```
OUT@$CE,$6
```

CTRL/SHIFT toetsen uitlezen.

Het is niet mogelijk om met GET of een dergelijke instructie de SHIFT- of de CTRL-toets uit te lezen. Door een bepaalde poort (poort $D1) uit te lezen kan gecontroleerd worden of 1 van deze beide toetsen is ingedrukt en wel als volgt:

```
10 INIT"CRT:M1"
20 INP@$D1,A
30 IF A=191 THEN PRINT"DE CTRL-TOETS IS
INGEDRUKT." :END
40 IF A=254 THEN PRINT"DE SHIFT-TOETS IS
INGEDRUKT." :END
50 RUN
```

Geluid.

Er is ook een poort (poort $F2) die het mogelijk maakt om allerlei verschillende geluidjes voort te brengen. Omdat in een apart hoofdstuk deze poort uitgebreid aan de orde zal komen, wordt hij hier niet besproken.

Palet veranderingen.

Normaal wordt een kleur van een bepaald PALET verandert met de instructie PAL, maar het kan ook door een bepaalde poort (poort $F0) aan te sturen en daarbij stelt het eerste cijfer het PALET voor dat veranderd wordt en het tweede cijfer de kleur voor die het PALET gaat krijgen. Bijvoorbeeld: OUT@$F0,$39 , daarbij gaat het om het getal $39. De 3 slelt het PALET voor (PALET 3 dus) en de 9 de nieuwe kleur die het PALET gaat krijgen. In programmavorm:

```
10 INIT"CRT:M1"
20 SYMBOL [1]100,0,"NEPTUNES",2,2
30 SYMBOL [2]100,20,"NEPTUNES",2,2
40 SYMBOL 100,40,"NEPTUNES",2,2
50 FOR D=$3F TO $0 STEP -1
60 OUT@$F0,D
70 WAIT 100 :NEXT D
80 PAL 0,0 :PAL 1,1 :PAL 2,2 :PAL 3,15
```

De computer laten vastlopen.

Er is één poort waarbij de computer vastloopt wanneer die poort uitgelezen wordt en dat is de poort $E0. Dit heeft weer te maken met het bankswitsching dat in hoofdstuk 3 al aan de orde is gekomen. Er wordt naar een ander geheugen overgeschakeld en daardoor loopt de computer vast. Probeert U het maar eens uit:

```
INP@$E0,A
```

en U zult zien dat de computer vastgelopen is en de BASIC opnieuw geladen moet worden of U drukt op de CTRL-toets en de resetknop.

Hoe vindt U Uw eigen IN's en OUT's.

Omdat er maar 255 poorten zijn, is het niet zo heel erg moeilijk om zelf nuttige IN's en OUT's te zoeken. De aanwezigheid van 255 poorten betekent ook dat er niet zoveel nuttige IN's en OUT's zijn.
Het kan best zo zijn dat er nog enkele nuttige IN's en OUT's zijn, maar dat ze nog niet ontdekt zijn en misschien kunt U ze zelf wel ontdekken.
Ook bij het zoeken naar nuttige IN's en OUT's geldt dat U eerst moet weten waarnaar U zoekt en dat U dan pas gaat zoeken en wel als volgt:

Laten we het voorbeeld van de cassetterecorder nemen. We zoeken dus een poort die controleert of er een toets van de cassetterecorder ingedrukt is. Het kan best zijn dat zo'n poort niet bestaat. Dan zoekt U dus helemaal voor niets, maar daar komt U dan vanzelf wel achter.
We beginnen er allereerst mee om de hoogste poort uit te lezen en dat is poort $FF en dat doen we als volgt:

```
10 INP@$FF,A
20 PRINT A :GOTO 10
```

Tijdens het lopen van dit programma drukken we bijvoorbeeld op de PLAY-toets en kijken of er een bepaalde verandering in de getallenreeks komt. Is dat niet het geval, dan kijken we een poort lager (poort $FE). Op een gegeven moment komt U dan bij poort $D2 en dan zal het U opvallen dat er een verandering komt in de getallenreeks en zie daar: We hebben de poort gevonden die controleert of er een toets op de cassetterecorder is ingedrukt.
U zult natuurlijk met de vraag zitten waarom er bij poort $FF begonnen is en niet bij poort $00. Dit heeft te maken met het feit dat de meeste poorten die gebruikt worden voor dergelijke zaken hoge waardes hebben. Kijkt U maar naar alle IN's en OUT's die gegeven zijn, alle poorten hebben hoge waardes.
Op een soortgelijke manier als bij IN gaat het ook bij OUT. U begint bij poort $FF en stuurt uit deze poort alle waardes van 0 t/m 255 en wacht af of er toevallig iets gebeurt. Daarbij zult U bijvoorbeeld op een gegeven moment bij poort $F2 tegen komen dat er geluid uit Uw SHARP komt. In programmavorm komt het aansturen van poorten er als volgt uit te zien:

```
10 FOR A=0 TO 255
20 OUT@$F2,A :NEXT A
```

4.3: Trucs'en Tips.

Ook sommige trucjes kunt U zelf vinden of bedenken. Het is moeilijk om uit te leggen hoe ze zijn te vinden. Vaak is het zo dat U ze bij toeval ontdekt. Hier volgen enkele nuttige trucs en tips.

CHECK-SUM ERROR bij laden, wat nu?

Het komt af en toe wel eens voor dat een programma niet goed van cassette geladen kan worden en er dan een CHECK-SUM ERROR melding verschijnt. Er is een oplossing om deze melding er in de meeste gevallen uit te krijgen en dat is door het programma met TRANS in te laden en daarna opnieuw te SAVEN.

Onduidelijk scherm, wat nu?

Het komt in de BASIC wel eens voor dat de achtergrondkleur en de voorgrondkleur dezelfde waarde hebben of dat de computer nergens meer op reageert. In dat geval kunt U proberen om een fatsoenlijke kleurcombinatie (zwart en wit) te krijgen door de CRTL-toets in combinatie met de resetknop in te drukken. Mocht dit niet helpen dan moet de BASIC opnieuw geladen worden.

Programma van QD naar cassette.

Er bestaat in de ROM-monitor geen aparte instructie om een programma van QD naar cassette te schrijven. Dat komt er dus op neer dat U gebruik moet gaan maken van het programma TRANS en dat wordt elke keer behoorlijk omslachtig. Er is echter ook een oplossing om het via de ROM-monitor te doen en die ziet er zo uit:

-Zet de computer aan.
-Druk op M om in de ROM-monitor te komen.
-Stop de QD met het programma in de QD-drive.
-Voer QC in en daarna de FILENAME. Mocht U de naam van het programma niet weten, voert U dan eerst QD in om alle programma's op de QD te kunnen bekijken.
-Druk op N.
-Voer in: M1104.
-Druk op <cr> tot U alle adressen t/m 1109 in beeld heeft.
-Druk op SHIFT/BREAK.
-Voer in: M1102.
-Voer nu het hexadecimale getal in dat bij adres 1104 staat.
-Doe hetzelfde met de adressen 1103 en 1105. De hexadecimale waarde die bij 1105 staat voert U bij 1103 in.
-Gaat U zo door tot het hexadecimale getal van adres 1109 op het adres 1107 staat.
-Druk daarna weer op SHIFT/BREAK.
-Voer daarna in: GE80A.
-Als het goed gebeurt is, staat het programma nu op cassette. Controleert U het even voor de zekerheid.
-Vergeet niet om elke keer op <cr> te drukken!

Controle van QUICK-DISK vanuit een programma.

```
10 ON ERROR GOTO "ERROR"
20 LOAD"XXXXXX"
30 END
40 LABEL "ERROR"
50 IF ERN=50 THEN PRINT"DE QUICK-DISK IS
 NIET KLAAR."
60 IF ERN=40 THEN PRINT"DIT PROGRAMMA ST
AAT NIET OP DEZE QUICK- DISK."
70 END
```

De naam "XXXXXX" in regel 20 is een willekeurig gekozen naam.

KEYWORDS voor het spel MOTY.

1- 6	MAJOR	51- 56	SHARK	101-106	PERKY	151-156	SHOOT
6-11	6MOTY	56- 61	RODEO	106-111	E002H	156-161	TRACK
11-16	LAKAI	61- 66	ALIEN	111-116	SONIC	161-166	TRACE
16-21	TUTOR	66- 71	PIO80	116-121	PYROL	166-171	VAULT
21-26	JEANS	71- 76	ASCII	121-126	GYROS	171-176	PAPUA
26-31	ROHAN	76- 81	METRO	126-131	04163	176-181	RELAX
31-36	SUM5Y	81- 86	BIMBO	131-136	SOLID	181-186	LOAVR
36-41	EAGLE	86- 91	NOOSE	136-141	SPARK	186-191	JAMOI
41-46	RSB55	91- 96	ERROR	141-146	STDAR	191-196	55296
46-51	LEICA	96-101	TASTE	146-151	TRAMP	196-200	NP2IJ

Repeated GET via PEEK.

Eigenlijk had deze PEEK in het eerste deel van dit hoofdstuk moeten staan, maar U zal er waarschijnlijk geen bezwaar tegen hebben dat het nu pas wordt vermeld.

```
10 GET A$ :PRINT CHR$(PEEK(4965)) :GOTO 10
```

Rechtstreeks in ROM-monitor.

Wanneer U een QD in de QD-drive heeft en U zet de computer aan, begint de computer automatisch te zoeken naar het eerste programma op die QD. Het kan best zijn dat U dat helemaal niet wilt, dat U bijvoorbeeld een programma van cassette wilt hebben of dat U in de ROM-monitor wilt werken. Dat kunt U oplossen door de QD-drive open te zetten of door eerst de M of de C in te drukken en ingedrukt te houden en daarna op de resetknop te drukken of de computer aan te zetten.

Van ROM-monitor terug naar de BASIC.

Bovenstaande kan op drie manieren. Ten eerste door op CTRL en reset te drukken. Ten tweede door in de ROM-monitor G005A in te voeren of door eerst G00AD in te voeren en daarna # in te typen. (Vergeet niet om elke keer op <cr> te drukken). De BASIC moet wel in het geheugen zitten!!

Alle BASIC-instructies.

Uit programmaatje laat alle BASIC-instructies zien, uitgezonderd de instructies voor de printer. (bv PRINT/P)

```
10 INIT"CRT:M1" :B=0
20 FOR A=$5973 TO $5C59
30 IF PEEK(A)>$40 AND PEEK(A)<$5B THEN P
RINT CHR$(PEEK(A));
40 IF PEEK(A)>$BF AND PEEK(A)<$DB THEN P
RINT CHR$(PEEK(A)-$80), :B=B+1
50 IF PEEK(A)=$A4 THEN PRINT"$", :B=B+1
60 IF B=84 THEN PRINT :PRINT"DRUK OP RET
URN VOOR VERVOLG." :GOSUB 90
70 NEXT A
80 GET A$ :IF A$="" THEN 80 ELSE END
90 GET A$ :IF ASC(A$)=13 THEN B=0 :CLS :
RETURN ELSE GOTO 90
```

Eenvoudiger een programma laden.

Wanneer er meer programma's op één QD staan en U wilt graag een machinetaalprogramma van deze QD laden en het is niet het eerste machinetaalprogramma op deze QD, kunt U dat doen door QL in te typen en daarna de naam van het programma, maar het kan ook op een andere manier. (Dit is vooral bedoeld voor programma's met lange of ingewikkelde namen.) Dat gaat als volgt:

-U drukt op M en zet de computer aan.
-Daarna voert U in: QD.
-U ziet nu alle programma's die op de QD staan.
-Daarna voert U in: QL.
-U gaat met de cursortoetsen omhoog tot de cursor aanbeland is bij het programma dat U wilt laden en daarna drukt U op <cr>.
-Als alles goed gebeurd is, wordt nu het door U gekozen programma geladen.
-Het kan zijn dat er 32 programma's op de QD staan en dat U net één van de eerste programma's wilt hebben. Dan moet U gewoon de 'oude' methode toepassen.

Een QD met zowel OBJ als BTX programma's.

Het komt regelmatig voor dat er zowel BTX als OBJ programma's op één diskette staan. Wanneer het eerste programma een BTX programma is en U wilt graag het eerste OBJ programma laden, moet U normaal dat doen wat hiervoor besproken is. Het kan ook eenvoudiger en wel als volgt:

-U drukt op M en zet de computer aan.
-U voert QL in.
-Bij FILENAME? drukt U gewoon op <cr>.
-Nu wordt automatisch het eerste OBJ-programma op deze QD geladen ook al is het eerste programma een BTX-programma.
-Deze tip werkt alleen wanneer het gaat om het eerste OBJ-programma!

4.4: CALLS in BASIC.

In hoofdstuk 2 zijn een aantal CALLS in machinetaal behandeld. Een soortgelijke instructie bestaat ook in BASIC en dat is de instructie USR($xxxx). Deze instructie is ongeveer vergelijkbaar met de instructie GOTO xx of RUN xx alleen wordt er niet naar een bepaald regelnummer gesprongen, maar naar een bepaald adres in het geheugen en alles wat er vanaf die geheugenplaats staat wordt dan uitgevoerd. Wanneer een RET-code wordt bereikt, zal er weer naar de BASIC teruggesprongen worden. Het kan ook zo zijn dat de computer vastloopt. Dan helpt CTRL/reset vaak, maar af en toe moet de gehele BASIC opnieuw geladen worden.
Bij USR is het weer precies als bij POKE en PEEK. Er kan naar alle adressen gesprongen worden, maar het grootste deel is van geen enkel nut. Het vinden van nuttige USR's gaat niet zo gemakkelijk als het vinden van POKES. Het is bij de meeste USR's zo dat ze bij toeval ontdekt worden of door de hele BASIC uit te gaan spitten.
De nuttige USR's zouden bijvoorbeeld ook gebruikt kunnen worden in een klein machinetaalprogrammaatje. Een voorbeeld daarvan zult U straks ook zien, maar eerst zal een rij met USR's gegeven worden waarvan sommige echt nuttig kunnen zijn en andere er gewoon voor de gein zijn.

USR($3B68) - Zet de cassettemotor stop.

USR($38C5) - Start de cassettemotor.

A$="D000" :USR($16C6,A$) - HEX-DUMP vanaf het adres D000. Het adres kan heel eenvoudig verandert worden door gewoon A$ te veranderen. U heeft hierbij als voordeel dat naar de BASIC teruggegaan wordt.

A$="A000" :USR($1738,A$) - MA000. Hierbij kan natuurlijk ook weer een ander adres genomen worden door A$ te veranderen.

USR($0419) - CLS.

USR($0A1B) - Kleinschriftmode wordt ingesteld.

USR($0A19) - Terug naar normale mode.

USR($0A1E) - Graphicmode wordt ingesteld.

USR($0992) - Cursor naar linker bovenhoek.

A$="NEPTUNES SOFTWARE." :USR($0015,A$) - Print de tekst NEPTUNES SOFTWARE op de cursorpositie.

```
10 USR($0003,A$)
20 CLS
30 USR($0015,A$)
```
- Allereerst moet U tekst invoeren en deze tekst wordt in A$ opgeslagen en eventueel uitgeprint.

USR($003E) - BEEP.

USR($0006) - Sla een regel over.

CALL om een punt in een bepaalde kleur op het scherm te zetten:
Deze CALL kan helaas niet gegeven worden met de instructie USR. Deze CALL moet in een programma verwerkt worden om de werking ervan te kunnen laten zien. Allereerst het programma, daarna zal een korte uitleg gegeven worden.

```
10 INIT"CRT:M1"
20 POKE $FD00,$6,$0,$21,$0,$0,$11,$0,$0,
$3E,$2,$32,$9F,$10,$CD,$94,$46,$2A,$6,$F
D,$23,$22,$6,$FD,$22,$3,$FD,$21,$1,$ED,$
35,$20,$E2,$C9
30 USR($FD00)
```

De CALL waar het hier om gaat (CALL 4694H) zet een punt op de plaats die aangegeven wordt door de registers DE en HL en in de kleur die bepaald wordt door het adres 109FH.
In DE komt de x-positie.
In HL komt de y-positie.
Op het adres 109FH komt de waarde van het palet of de kleur (dit geldt alleen voor de MODE M2).
Wanneer het HL-register en het DE-register nu elke keer met 1 vermeerderd worden, zoals hier gebeurt, krijgt U een schuine lijn.
Eigenlijk hoort dit programma thuis in het hoofdstuk over de VIDEO-RAM. Omdat de CALL het belangrijkste deel uit dit programma is, is dit programma in dit hoofdstuk geplaatst.

CALL om het toetsenbord uit te lezen en het ingetypte teken op het scherm te laten zien:
Allereerst het programma, daarna volgt de uitleg.

```
10 INIT"CRT:M1"
20 PAL 3,0 :PRINT"Het teken ";
30 POKE $FD00,$CD,$1B,$0,$FE,$0,$C2,$12,
$0,$C3,$0,$FD :USR($FD00)
40 PAL 3,15 :PRINT" heeft U ingetoetst."
```

Deze twee CALLS moesten ook in een programma verwerkt worden, omdat alleen op die manier de werking duidelijk te maken is.

CALL 001BH - Deze CALL zorgt er voor dat het toetsenbord uitgelezen wordt en dat de waarde van een ingedrukte toets in het A-register terecht komt.
CALL 0012H - Deze CALL zorgt er voor dat het teken waarvan de waarde in het A-register staat op de cursorpositie op het scherm komt te staan.

Wanneer er een toets ingedrukt wordt, wordt naar 0012H gesprongen en wordt de toets die ingedrukt is op de cursorpositie weergegeven.

Er zijn natuurlijk nog veel meer CALLS in BASIC, maar ze zijn lang niet allemaal zo leuk en zo handig als de gepubliceerde CALLS. De CALL voor het kleuren van een punt kan bijvoorbeeld een programma sneller maken, zoals in het programma PICTURESHOW is gedaan.

HOOFDSTUK 5: MUZIKALE TOEPASSINGEN.

In dit hoofdstuk zullen de mogelijkheden van de SHARP op muzikaal gebied besproken worden. De SHARP kent in totaal 3 akkoorden en 6 oktaven, er zijn dus genoeg mogelijkheden. Alle instructies die te maken hebben met het voortbrengen van geluid of het maken van muziek zullen in dit hoofdstuk besproken worden. Een instructie als BEEP zal dus ook weer aan de orde komen, omdat die instructie ook een kortstondig 'piepje' produceert.

5.1: De poort $F2.

De poort waar het eigenlijk allemaal om draait en die er voor zorgt dat er geluid of muziek uit Uw SHARP komt is poort $F2. Dit is de zogeheten PSG-poort en PSG staat voor Programmable Sound Generator. Deze generator bestaat uit drie gewone, afzonderlijk te bedienen generatoren en een ruisgenerator. Tevens bevat elke generator een verzwakker met een bereik van 28 dB.
Op de bladzijden A-7 t/m A-9 van het handboek vindt U ook een klein beetje informatie over deze PSG. De manier waarop in het handboek de PSG besproken wordt is niet de meest eenvoudige, vandaar dat wij in dit hoofdstuk deze poort heel uitvoerig zullen behandelen.
Veel boeken voor de SHARP MZ-800 behandelen de PSG. Wij hebben deze informatie ontleend aan het handboek, Alles über den MZ-800 van BBG, het boek Programmeren in de Toongenerator van SCCE en het Technical Reference Manual van SHARP zelf.
Het aansturen van de poort $F2 is niet bepaald een eenvoudige materie. Er komt vaak wel wat rekenwerk aan te pas om deze poort goed aan te kunnen sturen.

Elk van de drie toongeneratoren bevat een 10-bit brede teller die software-matig aan te sturen is en gebruikt wordt als een variabele frequentiedeler.
Met een formule is de frequentie om te rekeken naar getallen die voor de PSG te gebruiken zijn. Uit de formule komen in totaal 2 waardes. De ene waarde moet bij een getal opgeteld worden en dan uit poort $F2 gestuurd worden en de andere waarde moet gewoon uit poort $F2 gestuurd worden.
Voor elke toongenerator is een beginwaarde waarbij de uitkomst van de formule opgeteld moet worden, deze beginwaardes zijn:

128 voor toongenerator 1.
160 voor toongenerator 2.
192 voor toongenerator 3.

Nu volgt een programma waarmee U de frequentie om kunt rekenen naar waardes die voor de PSG te gebruiken zijn. Dit programma is afkomstig uit het boek Alles über den MZ-800 van BBG.

```
10 CLS
20 INPUT"Frequentie in Hertz ?";FR
30 F0=11094/(FR/10)
40 B6=INT(F0/16)
50 B4=FRAC(F0/16)*16
```

```
60 IF FRAC(B4)>=.5 THEN B4=INT(B4)+1 :EL
SE B4=INT(B4)
70 IF FRAC(F0)>=.5 THEN F0=INT(F0)+1 :EL
SE F0=INT(F0)
80 PRINT"6 Bit deel :";B6
90 PRINT"4 Bit deel :";B4
100 PRINT"F0 :";F0
110 PRINT :PRINT :GOTO 20
```

U ziet dat er een 4 bit en een 6 bit deel wordt berekend. 4 plus 6 is 10 en daar heeft U de 10 bits teller. De waarde van het 4 bits deel wordt opgeteld bij de beginwaarde van een toongenerator en de waarde van het 6 bits deel wordt apart uit de poort $F2 gestuurd.
Een voorbeeld:

U wilt een frequentie van 442 Hz uit toongenerator 1 laten komen. U berekend met het programma het 4 bits en het 6 bits deel. Vervolgens stuurt U de berekende waardes op de juiste manier uit poort $F2 en U hoort een toon met een frequentie van 442 Hz.
Wanneer U het goed heeft gedaan, komt het 4 bits deel op 11 en het 6 bits deel op 15.
U stuurt deze waardes daarna op de volgende manier uit poort $F2:

```
OUT@$F2,160+11 :OUT@$F2,15
```

U heeft nu een toon met een frequentie van 442 Hz uit toongenerator 2 gestuurd, maar U hoort nog niets. Dit heeft te maken met het feit dat U de geluidssterkte nog in moet voeren. Voor elke toongenerator is daar een apart rijtje voor.

```
Toongenerator 1 : 144-159
Toongenerator 2 : 176-191
Toongenerator 3 : 208-223
Ruisgenerator   : 240-255
```

Hierbij is de laagste waarde voor een bepaalde generator de grootste geluidssterkte. Wilt U nu bijvoorbeeld de toon van 442 Hz zo hard mogelijk horen dan voert U in:

```
OUT@$F2,176
```

Wilt U de toon helemaal niet meer horen dan voert U in:

```
OUT@$F2,191
```

U kunt de toon van 442 Hz dus van hard naar zacht laten lopen op de volgende manier:

```
OUT@$F2,160+11 :OUT@$F2,15 :FOR A=176 TO
 191 :OUT@$F2,A :WAIT 150 :NEXT A
```

Zo kunt U ook een golvende toon laten horen door de geluidssterkte eerst van hard naar zacht en vervolgens van zacht naar hard te laten lopen en dat de hele tijd te herhalen. In dit hoofdstuk zult U dat nog een keer tegenkomen.

Er is nog iets belangrijks dat U moet weten en dat is dat er maximaal 3 generatoren tegelijk gebruikt kunnen worden. U kunt dus maximaal 3 tonen laten horen of 2 tonen en een ruis.
Met de instructie NOISE kunt U slechts 1 toon en 1 ruis gebruiken. Door de poort $F2 rechtstreeks aan te sturen kunt U dus meer bereiken dan met een BASIC-instructie.
U ziet dat het aansturen van poort $F2 misschien toch iets eenvoudiger is dan in eerste instantie werd verondersteld. Om een muziekstuk te laten spelen door poort $F2 rechtstreeks aan te sturen is heel erg moeilijk, daarvoor is het gebruik van de BASIC-instructie MUSIC of een machinetaalroutine veel geschikter. Zowel de MUSIC-instructie als de oplossing in machinetaal zullen verderop in dit hoofdstuk aan de orde komen.
Nu zullen eerst een legio aan mogelijkheden via poort $F2 besproken worden. U kunt er echt van alles mee doen, van geluidjes van een laserpistool tot spraak. Kan de SHARP praten dan? Natuurlijk kan de SHARP dat. Het is alleen een kwestie van het goed aansturen van poort $F2 en dat is niet eenvoudig. Wij zullen dan ook niet verder ingaan op de spraakmogelijkheden van de SHARP. Mensen die de SHARP willen horen praten, verwijzen wij naar programma's als SPRACHUHR e.d.
Nu zullen een aantal voorbeelden volgen van allerlei geluidjes die op de SHARP mogelijk zijn. Soms zal er een korte toelichting bij staan.

<u>Voorbeeld 1: Lancering.</u>

Dit programmaatje laat het geluid van een lancering van een raket horen.

```
10 ' LANCERING
20 INIT"CRT:M1" :PAL 0,1 :CURSOR 0,24
30 IF PEEK($4DCF)>1 THEN PO=$4DDO ELSE P
O=$4DCF
40 POKE PO,$1 :SYMBOL 152,184,CHR$(60),2
,2 :POKE PO,$0
50 P=242 :' P=poort $F2
60 GOSUB "WISSEN"
70 FOR L=0 TO 15
80 OUT@P,231 :OUT@P,240
90 PRINT
100 FOR A=192 TO 207
110 OUT@P,A :OUT@P,L
120 NEXT A :PRINT
130 OUT@P,(L+240)
140 NEXT L
150 END
160 LABEL "WISSEN"
170 OUT@P,159 :OUT@P,191
180 OUT@P,223 :OUT@P,255
190 RETURN
```

In regel 80 wordt de ruis bepaald die gebruikt wordt. U ziet dat in dit programma gebruik wordt gemaakt van een toon die steeds lager wordt in frequentie en tevens een ruis die steeds lager wordt in frequentie en daardoor wordt dit effect bereikt.

In de subroutine die aangegeven wordt door LABEL "WISSEN" worden alle geluidssterktes op 0 gezet. Na het aanspreken van deze routine zal er dus niets meer te horen zijn. Een zelfde effect wordt bereikt met USR(0071). Na gebruik van deze USR zal er ook niets meer te horen zijn.

Voorbeeld 2: Tik van een benzinepomp.

```
10 ' TIK
20 INIT"CRT:M1"
30 P=242 :' P=poort $F2
40 GOSUB "WISSEN"
50 OUT@P,131 :OUT@P,10
60 OUT@P,175 :OUT@P,9
70 FOR T=0 TO 11 :CURSOR 0,0 :PRINT T
80 FOR A=144 TO 159
90 OUT@P,A :OUT@P,(A+32)
100 WAIT 60
110 NEXT A
120 NEXT T
130 END
140 LABEL "WISSEN"
150 OUT@P,159 :OUT@P,191
160 OUT@P,223 :OUT@P,255
170 RETURN
```

In dit programma wordt gebruik gemaakt van twee toongeneratoren en wordt geen gebruik gemaakt van de ruis.

Voorbeeld 3: Twee tonen tegen elkaar in.

Dit programma laat het effect horen van twee tonen die tegengesteld aan elkaar werken. De frequentie van de ene toon wordt steeds hoger, terwijl de frequentie van de andere toon steeds lager wordt.
Op een gegeven moment zullen beide frequenties precies gelijk zijn.

```
10 INIT"CRT:M1"
20 USR(71) :P=242
30 FOR A=15 TO 0 STEP -1
40 FOR B=175 TO 160 STEP -1
50 OUT@P,B :OUT@P,A
60 OUT@P,367-B :OUT@P,15-A
70 OUT@P,176 :OUT@P,208
80 NEXT B,A
90 USR(71)
```

Voorbeeld 4: Steeds sneller lopende golf.

Dit programma laat een toon van hard naar zacht lopen en daarna van zacht naar hard en dat gaat steeds sneller.

```
10 INIT"CRT:M1" :USR(71) :P=242 :OUT@P,1
76 :OUT@P,171 :OUT@P,5
```

```
20 FOR A=25 TO 0 STEP -1
30 FOR B=177 TO 190
40 OUT@P,B :WAIT A*4
50 NEXT B
60 FOR B=189 TO 176 STEP -1
70 OUT@P,B :WAIT A*4
80 NEXT B
90 NEXT A
100 USR(71)
```

Voorbeeld 5: Explosie.

```
1 INIT"CRT:M1" :X=20 :C=3
2 USR(71) :OUT@$F2,228
3 PAL 0,7 :CURSOR X,12 :PRINT [2]CHR$(20
0) :WAIT 2000
4 IF PEEK($4DCF)>1 THEN PO=$4DDO ELSE PO
=$4DCF
5 POKE PO,1 :CURSOR X,12 :PRINT [0]CHR$(
200) :SYMBOL [1]X*8,96,CHR$(141),1,1 :SY
MBOL [1]X*8-8,88,CHR$(188),3,3
6 OUT@$F2,$C8 :FOR A=$F0 TO $FF :OUT@$F2
,A :WAIT 200 :GOSUB 8 :NEXT A
7 POKE PO,0 :END
8 C=C+2 :CLS
9 SYMBOL [1]X*8-4*(C-1),88-4*(C-3),CHR$(
188),C,C
10 SYMBOL [1]X*8-2*(C-3),96-2*(C-3),CHR$
(141),C/2,C/2 :RETURN
```

Voorbeeld 6: 3 variabele tonen.

Dit programma laat twee tonen en een ruistoon horen en die tonen kunt U met de toetsen 1-6 en A-F zelf veranderen. U kunt met de toetsen 1-6 de toonhoogte veranderen en met de toetsen A-F kunt U de demping veranderen. Een demping van 30 dB betekent dat U de toon niet meer hoort.
Door op de CR-toets te drukken kunt U het programma beëindigen.

```
1 USR(71)
2 INIT"CRT:M1" :PAL 1,12 :PAL 2,14
3 PRINT"1=TOONHOOGTE POORT 1 OMHOOG"
4 PRINT"2=TOONHOOGTE POORT 1 OMLAAG"
5 PRINT"3=TOONHOOGTE POORT 2 OMHOOG"
6 PRINT"4=TOONHOOGTE POORT 2 OMLAAG"
7 PRINT"5=TOONHOOGTE RUISPOORT OMHOOG"
8 PRINT"6=TOONHOOGTE RUITPOORT OMLAAG"
9 PRINT :PRINT"A=GELUIDSSTERKTE POORT 1
OMHOOG"
10 PRINT"B=GELUIDSSTERKTE POORT 1 OMLAAG
"
11 PRINT"C=GELUIDSSTERKTE POORT 2 OMHOOG
"
```

```
12 PRINT"D=GELUIDSSTERKTE POORT 2 OMLAAG
"
13 PRINT"E=GELUIDSSTERKTE RUISPOORT OMHO
OG"
14 PRINT"F=GELUIDSSTERKTE RUISPOORT OMLA
AG"
15 COLOR 1
16 PRINT :PRINT :PRINT TAB(13)"freq. (Hz
)"; :PRINT TAB(27)"demping (dB)" :PRINT
17 PRINT" POORT 1"
18 PRINT :PRINT" POORT 2"
19 PRINT :PRINT" RUISPOORT"
20 COLOR 2
21 BOX 0,132,319,147
22 BOX 0,147,319,162
23 BOX 0,162,319,177
24 BOX 88,132,204,177
25 COLOR 3
26 CURSOR 0,24 :PRINT"CR=EINDE  (C)1990
by NEPTUNES SOFTWARE.";
27 F1=400 :D1=144
28 F2=500 :D2=176
29 FR=600 :DR=240
30 CURSOR 12,17 :PRINT F1;" " :PRINT :PR
INT TAB(12)F2;" " :PRINT :PRINT TAB(12)F
R;" "
31 CURSOR 26,17 :PRINT (D1-144)*2;" " :P
RINT :PRINT TAB(26)(D2-176)*2;" " :PRINT
 :PRINT TAB(26)(DR-240)*2;" "
32 FQ=F1 :DE=D1 :GOSUB "BEREKEN"
33 FQ=F2 :DE=D2 :GOSUB "BEREKEN"
34 FQ=FR :DE=DR :GOSUB "BEREKEN"
35 GET A$ :A=PEEK(4965) :IF A=13 THEN US
R(71) :CLS :END
36 IF A<49 OR A>70 THEN 35
37 IF A>54 AND A<65 THEN 35
38 F1=F1+(5 AND A=49 AND F1<10000)-(5 AN
D A=50 AND F1>30)
39 F2=F2+(5 AND A=51 AND F2<10000)-(5 AN
D A=52 AND F2>30)
40 FR=FR+(5 AND A=53 AND FR<10000)-(5 AN
D A=54 AND FR>30)
41 D1=D1+(1 AND A=66 AND D1<159)-(1 AND
A=65 AND D1>144) :D2=D2+(1 AND A=68 AND
D2<191)-(1 AND A=67 AND D2>176)
42 DR=DR+(1 AND A=70 AND DR<255)-(1 AND
A=69 AND DR>240) :GOTO 30
43 LABEL "BEREKEN" :FO=11094/(FQ/10) :B6
=INT(FO/16) :B4=FRAC(FO/16)*16
44 IF FRAC(B4)>.5 THEN B4=INT(B4)+1 :ELS
E B4=INT(B4)
45 IF FRAC(FO)>.5 THEN FO=INT(FO)+1 :ELS
E FO=INT(FO)
46 OUT@$F2,DE-16+B4 :OUT@$F2,B6
47 OUT@$F2,DE :RETURN
```

5.2: BASIC-instructies voor geluid en muziek.

In dit deel van hoofdstuk 5 zullen de BASIC-instructies besproken worden die er voor zorgen dat de SHARP geluiden of muziek produceert. Op de bladzijden 6-68 t/m 6-72 van het handboek vindt U ook informatie over de zogenaamde muziekbesturings statements. Omdat deze uitleg ruim voldoende is om met de meeste instructies te kunnen werken, zal dit deel van hoofdstuk 5 hoofdzakelijk voorbeelden van de verschillende instructies behandelen.

BEEP - Met de instructie BEEP is een korte toon van 440 Hz op te wekken. Een voorbeeld:

```
10 GET A$ :IF A$<>"" THEN BEEP
20 GOTO 10
```

Dit programmaatje laat bij iedere toetsaanslag een korte toon horen.

SOUND - Met deze instructie kunt U een toon met een variabele toonhoogte en een variabele lengte opwekken. De instructie ziet er als volgt uit: SOUND (x,y) waarbij x de toonhoogte is en y de lengte. Op blz. 6-71 van het handboek vindt U een tabel voor de instructie SOUND.

NOISE - Met deze instructie kunt U een "heldere noise" opwekken. In tegenstelling tot wat er in het handboek staat, is het niet mogelijk om twee delen "heldere noise" tegelijk op te wekken.

MUSIC:

Dan zijn we aanbeland bij de instructie waar het eigenlijk allemaal om draait en dat is de instructie MUSIC. Voorafgaand of aansluitend aan deze instructie kunnen de volgende instructies voor komen:

TEMPO - Bepaald het tempo waarmee een muziekstuk gespeeld moet worden. Hierbij is TEMPO 1 het laagste tempo en TEMPO 7 het hoogste tempo.

MUSIC STOP - Stopt alle geluidsopwekkingen die op dat moment bezig zijn, dus ook geluiden die opgewekt zijn door poort $F2 rechtstreeks aan te sturen.

MUSIC WAIT - Deze instructie zorgt er voor dat de hele serie noten uitgespeeld wordt alvorens verder wordt gegaan met het programma.

MUSIC INIT - Deze instructie stelt de muziek en de NOISE-instructie in op een standaardpositie en dat is "O2V15L5T4S8M255"

Nu komen we dan echt bij de MUSIC-instructie zelf. Wat precies de mogelijkheden zijn met deze instructie en hoe U het moet gebruiken, kunt U keurig nalezen op de blz. 6-68 t/m 6-70 van het handboek. Op de volgende bladzijden worden enkele voorbeelden van de mogelijkheden van de MUSIC-instructie getoond.

Nu zullen een aantal voorbeelden van de mogelijkheden van de instructie MUSIC volgen. U kunt er van alles mee doen, van het geluid van een LASER-pistool tot de BOLERO of iets dergelijks. Een stuk van de BOLERO zult U in het laatste deel van dit hoofdstuk kunnen vinden.
De voorbeelden die hier gegeven worden zijn eenvoudige voorbeelden van geluidjes en korte stukjes van bepaalde muziekstukken.
Mocht een programma niet werken, probeert U dan eens de instructie MUSIC INIT.
In de programma's kunnen de tekens '■' en '▄' voor komen. U kunt deze tekens op de volgende manier krijgen:

■ kunt U krijgen door eerst op de GRAPH-toets te drukken en daarna op 2.
■ kunt U krijgen door eerst op de GRAPH-toets te drukken en daarna op SHIFT/2.

Met deze informatie kunt U aan de slag met de voorbeelden.

Voorbeeld 1: Een stukje uit When the Saint.

In dit voorbeeld wordt gebruik gemaakt van slechts 1 akkoord, dat zal in de meeste voorbeelden het geval zijn.

```
1 MUSIC INIT
2 TEMPO 7
3 MUSIC"C5E5F5G9G5C5E5F5G9G5C5E5F5"
4 MUSIC"G7E7C7E7D9D5E5E5D5"
5 MUSIC"C8R0C5E7G7R0G5F8F7E5F5"
6 MUSIC"G7E7C7D7C9C5R0C5E5F5"
7 MUSIC"G9G5C5E5F5G9G5C5E5F5"
8 MUSIC"G7E7C7E7D9D5E5R0E5D5"
9 MUSIC"C8R0C5E7G7R0G5F8F7E5F5"
10 MUSIC"G7E7C7D7C9"
11 MUSIC WAIT :CLS :END
```

Voorbeeld 2: Pac-man muziekje.

```
TEMPO 7 :MUSIC"C2GECGRC4#C2#GF#C#GR#C4C2
GECGRC4R0C2DEFGAB■C"
```

Voorbeeld 3: Laser-pistool.

```
TEMPO 7 :MUSIC"C■CBAGFEDC■CBAGFEDCRR"
```

Voorbeeld 4: Kort melodietje.

Veel voorbeelden bestaan uit zeer korte melodietjes. Soms komen ze uit grote muziekstukken en soms komen ze uit spelletjes. Andere melodietjes zijn weer volledige melodietjes.

```
TEMPO 4 :MUSIC"CRCEREGR0GRGR■C2"
```

Voorbeeld 5: Kort melodietje.

Dit is een kort stukje uit een bekende melodie.

```
TEMPO 4 :MUSIC"RR AD#CDEFEFGAARR"
```

Voorbeeld 6: Kort melodietje.

Dit is een kort stukje uit een bekende melodie.

```
TEMPO 4 :MUSIC"■CC0AARR0D■CAAR0AGFG0F0DD
D0R0AA0G0F0DD0RR"
```

Voorbeeld 7: Kort melodietje.

Weer een stukje uit een hele bekende melodie.

```
TEMPO 4 :MUSIC"D1#D1E1■C2E1■C2E1■C4■C1■C
1■D1■#D1■E1■C1■D1■E1■E1B1■D1■C4"
```

Voorbeeld 8: Kort melodietje.

Een kort stukje uit een bekende melodie.

```
TEMPO 6 :MUSIC" #D3 F4R1 F7 F1R1 #D3 F5
#G #A B3C BC5 #G3 F5 C3 #D3 F4R1 F7 F3"
```

Voorbeeld 9: Kort melodietje.

Een kort stukje uit een hele bekende melodie.

```
TEMPO 6 :MUSIC"R3 C5 D3 F4 GR3R3 G5 F3 D
4 CR3R3 C5 D3 F4 GR1 #G1 A2R0 A3 G0R G3
F1 F5RR3C5D3F4GR3R3G5F3D4CR3R3C5D3F4GR1#
G1A2R0A3G0RGG3F1F5R5"
```

Muziekstuk.

Dit programma laat een groter muziekstuk horen.

```
1 MUSIC INIT :TEMPO 5
2 A1$="■C2R0■C2G■C■E■C■E■G■C■#B■B■A■G6R1
"
3 A2$="■G2■F■DBGB■D"
4 A3$="■F■E■C■A■G■G"
5 A4$="■#B■A■F■CA■C■F■A■G■C■G■GR"
6 A5$="■GR1"
7 A6$="B■CG■E■C■C■CR"
8 A7$="B■CG■E■C■C■CRR"
9 A8$="RR"
```

```
10 MUSIC A1$,A2$,A3$,A5$,A1$,A2$,A6$,A4$
,A2$,A3$,A8$,A4$,A8$,A2$,A7$
11 MUSIC WAIT :CLS :END
```

Voorbeeld 10: Kort melodietje.

```
TEMPO 7 :MUSIC"+F5+G5+#D5#A5+#D7R7C5D5-#
A5-F5-#A7"
```

Voorbeeld 11: Het laatste korte melodietje.

Ook dit melodietje is een klein stukje uit een bekende melodie.

```
TEMPO 7 :MUSIC"F5+C7+D1+C1#A1+C7F5+C7+D1
+C1#A1+C7F5+E7+C5F5+E7+C5F5+E1+D1+E1+F7"
```

Dit waren een aantal voorbeelden van de mogelijkheden van de instructie MUSIC. Sommige melodietjes kunnen pas gespeeld worden na het gebruik van MUSIC INIT. Andere melodietjes komen echter niet tot hun recht wanneer MUSIC INIT is gebruikt.
Het voorbeeld van het laser-pistool komt alleen tot zijn recht wanneer de instructie MUSIC INIT nog niet is gebruikt.
U zag hier allemaal voorbeelden die gebruik maakten van slechts 1 akkoord, terwijl er 3 tegelijk gebruikt kunnen worden. In het laatste deel van dit hoofdstuk staan enkele melodietjes die gebruik maken van alle 3 de akkoorden. Het resultaat is dan uiteraard mooier, maar de programma's worden gelijk een stuk langer.
Deze melodietjes zijn ook allemaal op een andere manier ten gehore te brengen en dat is door gebruik te maken van machinetaal. In het volgende deel van dit hoofdstuk zal dit uitgebreid besproken worden.
De mogelijkheden in de 800-BASIC zijn niet helemaal gelijk aan de mogelijkheden in de 700-BASIC. De mogelijkheden in de 700-BASIC zullen gedeeltelijk in hoofdstuk 7 aan de orde komen.

5.3: Muziek via machinetaal.

De mogelijkheid bestaat om via een heel eenvoudig machinetaalprogrammaatje muziek te laten horen. Het enorme voordeel van deze methode is dat U tijdens het spelen iets anders kunt doen op de computer (U hoeft dus niet te wachten tot de muziek uitgespeeld is) en het muziekprogramma in machinetaal minder geheugen in beslag neemt.
In dit deel van hoofdstuk 5 zal uigelegd worden hoe U de data om kunt zetten naar data die voor het machinetaalprogramma te begrijpen zijn en zullen enkele voorbeelden gegeven worden van muziekstukjes die wij voor U om gaan zetten naar machinetaal.

De noten:

In de machinetaal van de BASIC (MB) kunt U niet letterlijk de noten invoeren die U wilt spelen. In MB moet U speciale codes invoeren.
Elke noot heeft een andere code en één enkele noot kan weer verschillen per oktaaf.
Onderstaand schema geeft aan welke codes U in MB moet gebruiken voor de noten en de verschillende oktaven:

OKTAAF / NOOT	0	1	2	3	4	5	6
C		12	24	36	48	60	72
#C		13	25	37	49	61	73
D		14	26	38	50	62	74
#D		15	27	39	51	63	75
E		16	28	40	52	64	76
F		17	29	41	53	65	77
#F		18	30	42	54	66	78
G		19	31	43	55	67	79
#G		20	32	44	56	68	80
A	9	21	33	45	57	69	81
#A	10	22	34	46	58	70	82
B	11	23	35	47	59	71	83

Enkele voorbeelden: MUSIC "O2#D" is in MB de code 27.
MUSIC "O4#G" is in MB de code 56.
MUSIC "O6A" is in MB de code 81.
enz.

Vervorming en vervormingssnelheid.

In de BASIC wordt de vervorming aangegeven door S en de vervormingssnelheid door M, bijvoorbeeld MUSIC "S2M7".
In MB zijn daar ook speciale codes voor en die codes moet U zelf berekenen en dat gaat als volgt:
Voor de vervorming geldt: 144 + vervorming. S2 wordt in MB dus de code 146.
Voor de vervormingssnelheid geldt: 160, snelheid. U geeft dus eerst de code 160 en daarna de code van de vervormingssnelheid. M7 wordt in MB dus 160,7 en M255 wordt 160,255.
Het bovenstaande voorbeeld komt er in MB dus als volgt uit te zien:
146,160,7

Tempo.

Bij het tempo wordt ook uitgegaan van een standaard code waarbij U het tempo op moet tellen. Voor het tempo geldt het volgende in MB:
TEMPO in MB is 105 + tempo. TEMPO 1 wordt in MB dus 106. TEMPO 5 wordt in MB 110, enz.

Lengte van de noten.

In de BASIC wordt de lengte van noten aangegeven door het getal dat er achter staat. In MB komt de lengte voor de noot te staan.
Een voorbeeld: MUSIC "O2A3" wordt in MB: 99,33. Bij de lengte geldt dus weer precies hetzelfde als bij het tempo en de vervorming. Er wordt uitgegaan van een standaard code waarbij U de lengte van een noot op moet tellen. Voor de lengte van een noot geldt het volgende:
De lengte wordt in MB aangegeven door de lengte bij 96 op te tellen.
Lengte 3 wordt dus 99. Lengte 6 wordt 102, enz.

Volume.

Ook bij het instellen van het volume gaan we uit van een standaard code waarbij het volume opgeteld moet worden.
Voor het volume geldt het volgende:
Het volume wordt in MB aangegeven door het volume op te tellen bij 128.
Volume 3 wordt dus 131. Volume 15 wordt 143, enz.

Aan de hand van de uitleg die tot nu toe gegeven is, kan een gewoon muziekprogramma omgezet worden naar een machinetaalprogramma. Er zijn echter ook bepaalde dingen die nog niet besproken zijn en dat zijn tekens als '+' '-' en dergelijke. In MB is het heel eenvoudig om deze tekens om te zetten in codes. Bij '+' en '■' neemt U gewoon een oktaaf hoger en bij '-' en ' ' neemt U gewoon een oktaaf lager.
Een voorbeeld: MUSIC "O2+A3" wordt in MB: 99,45.

Het machinetaalprogramma.

In het machinetaalprogramma dat het muziekje ten gehore moet brengen kunt U zelf bepalen hoeveel akkoorden U wilt gebruiken. Op een heel eenvoudige manier kunt U 1,2 of 3 akkoorden laten horen. Hoe het machinetaalprogramma er precies uit komt te zien, kunt U lezen op de volgende pagina's.

Het machinetaalprogramma kunnen we bijvoorbeeld vanaf het adres $F000 neerzetten. Het neemt dan natuurlijk wel een behoorlijk deel van het geheugen weg, maar het is toch slechts als een voorbeeld bedoeld.
Vanaf het adres $F000 zetten we alle muziekcodes neer. Van belang is dat de muziekstring in MB afgesloten wordt met een eindcode en dat is $FF (255 decimaal).

Een voorbeeld:
We willen MUSIC "V7T6O2A3S2M27#CA5" in MB omzetten. De muziekstring komt er in MB-code als volgt uit te zien:

```
V7          = 135       (128+7).
T6          = 111       (105+6)
O2,A3       = 99,33     (96+3,33)
S2          = 146       (144+2)
M27         = 160,27
(O2) #C     = 25
(O2) A5     = 101,33    (96+5,33)
```

Deze waardes gaan we vanaf adres $F000 in het geheugen zetten en afsluiten met een eindcode. Eerst wordt het geheugen vanaf $EFFF gelimiteerd om niet toevallig BASIC over het machinetaalprogramma heen te schrijven. Alles bij elkaar komt het er dus als volgt uit te zien:

```
LIMIT $EFFF :POKE $F000,135,111,99,33,14
6,160,27,25,101,33,$FF
```

Nu moet U natuurlijk ook nog een machinetaalprogramma hebben dat deze muziekstring kan laten horen. Hier volgt het programmaatje dat daar voor nodig is en dat programmaatje komt vanaf adres $F500 in het geheugen te staan:

```
POKE $F500,$6,$3,$DF,$23,$3E,$0,$11,$00,
$F0,$DF,$21,$6,$1,$DF,$23,$C9
```

(Hierbij zijn de onderstreepte waardes de waardes die echt van belang zijn.)

U kunt deze routine aanroepen door USR($F500) en wanneer U dat ingevoerd heeft, hoort U de muziekstring.

De assemblerlisting van het programma ziet er als volgt uit:

```
F500 ORG F500H
F500 LD B,03H
F502 RST 3
F503 DEFB 23H
F504 LD A,00H
F506 LD DE,F000H
F509 RST 3
F50A DEFB 21H
F50D LD B,01H
F50D RST 3
F50E DEFB 23H
F50F RET
```

De waarde in het A-register (in dit geval 0) geeft aan welk akkoord er gespeeld moet worden en de waarde in het DE-register geeft aan vanaf welk adres de codes te vinden zijn (in dit geval dus vanaf F000H).
De rest van het programma blijft ten alle tijden ongewijzigd, ook al worden er meer akkoorden tegelijk gespeeld.

Het A-register kan de volgende waardes bevatten:

A=0 - Akkoord 1.
A=1 - Akkoord 2.
A=2 - Akkoord 3.
A=3 - NOISE.

Een machinetaalprogramma om twee akkoorden en de NOISE tegelijk te laten horen komt er als volgt uit te zien:

```
POKE $6,$3,$DF,$23,$3E,$0,$11,$00,$F0,$D
F,$21,$3E,$1,$11,$4E,$F7,$DF,$21,$3E,$3,
$11,$34,$F7,$DF,$21,$6,$1,$DF,$23,$C9
```

Het onderstreepte gedeelte is er dus extra bij gekomen om akkoord 2 en de NOISE te laten horen.
De codes voor akkoord 2 komen vanaf het adres $F74E te staan.
De codes voor NOISE komen vanaf het adres $F734 te staan.

U ziet dat het dus helemaal niet zo moeilijk is om een muziekje van BASIC naar machinetaal om te zetten en wat helemaal gemakkelijk is dat U in het geheugen kunt kijken hoe een BASIC-muziekstring er in MB uit komt te zien, omdat een normale BASIC-muziekstring ook eerst in MB omgezet wordt alvorens het gespeeld wordt.
Als laatste volgt nu een programma dat drie muziekstrings vanaf adres $F000 in MB in het geheugen zal zetten en tevens aan zal geven vanaf welk adres de strings in het geheugen komen te staan.

```
10 LIMIT $EFFF :TEMPO 5
20 CLS: ADRES=$F000 :LEES=$2ED0 :STEM=1
30 A$="03V15S0M10+C5B1+C+D+CBA+C3+C1A+C5
B1+C"
40 B$="02V10S0M10E5D1CDCDFE3G1FE5D1C"
50 C$="01V11S0M5C3C1C0CC3C1C0CC3CCE1E0EE
3E1E0E" :MUSIC A$;B$;C$
60 FOR F=0 TO 2
70 PRINT"STEM";STEM;" BEGINT OP ADRES ";
HEX$(ADRES) :STEM=STEM+1
80 P=PEEK(LEES)
90 POKE ADRES,P :ADRES=ADRES+1 :LEES=LEE
S+1
100 IF P=255 AND PEEK(LEES-1)<>160 THEN
NEXT F :END
110 GOTO 80
```

De eerste regel van de BOLERO (een muziekstuk uit het het volgende deel van dit hoofdstuk) wordt omgezet in machinetaal. U ziet gelijk waar de verschillende stemmen (akkoorden) in het geheugen komen te staan. Door het volgende programmaatje kunt U dit stukje muziek dus ook laten horen:

```
POKE $F500,$6,$3,$DF,$23,$3E,$0,$11,$00,
$F0,$DF,$21,$3E,$1,$11,$18,$F0,$DF,$21,$
3E,$2,$11,$30,$F0,$DF,$21,$6,$1,$DF,$23,
$C9 :USR($F500)
```

5.4: Enkele muziekstukken.

In het laatste deel van dit hoofdstuk tonen wij U nog een aantal muziekstukken. Eén programma is afkomstig van SCCE en twee uit het boek Alles über den MZ-800 van BBG.

Programma 1: Anniversary song.

```
10 'Anniversary song
20 'Een programma van L.J. Roetman
30 '                   Rutgerskamp 20
40 '                   3853 ES Ermelo
50 '                   Tel. 03417-58678
60 '
70 'Dit programma draait op alle
80 'gangbare Basicvarianten
90 '
100 MUSIC INIT :TEMPO 7
110 IF I=0 THEN INIT"CRT:M1" :PAL 2,1 :P
AL 1,6 :ELSE 210
120 DIM X$(4,7) :DIM A(2)
130 GOSUB "IN"
140 BOX 8,8,312,88,2
150 BOX 25,50,160,70,2
160 SYMBOL 22,16,"ANNIVERSARY",3,3
170 SYMBOL 190,50,"SONG",3,3
180 SYMBOL [1]32,53,"Al Jolson",1,1
190 SYMBOL [1]32,61,"Saul Chaplin",1,1
200 SYMBOL [1]32,76,"Based on a theme by
 Ivanovici",1,1
210 RESTORE 690 :GOSUB 550
220 Y=116
230 SYMBOL 12,184,"LJR 6-1988",1,1
240 FOR I=0 TO 2
250 GET X$ :IF X$="" THEN 250
260 IF X$="E" THEN CLS :END
270 IF X$="1" THEN A(I)=1 :SYMBOL 12,Y,"
Sopraan",1,1
280 IF X$="2" THEN A(I)=2 :SYMBOL 12,Y,"
Alt",1,1
290 IF X$="3" THEN A(I)=3 :SYMBOL 12,Y,"
Tenor",1,1
300 IF X$="4" THEN A(I)=4 :SYMBOL 12,Y,"
Bas",1,1
310 IF X$="5" THEN RESTORE 700 :I=2
320 Y=Y+16 :NEXT I
330 WAIT 1000 :GOSUB 550
340 MUSIC X$(A(0),0);X$(A(1),0);X$(A(2),
0)
350 MUSIC X$(A(0),1);X$(A(1),1);X$(A(2),
1)
360 WAIT 5000 :GOSUB 550
370 MUSIC X$(A(0),0);X$(A(1),0);X$(A(2),
0)
```

```
380 MUSIC X$(A(0),7);X$(A(1),7);X$(A(2),
7)
390 WAIT 5000 :GOSUB 550
400 MUSIC X$(A(0),2);X$(A(1),2);X$(A(2),
2)
410 MUSIC X$(A(0),3);X$(A(1),3);X$(A(2),
3)
420 MUSIC X$(A(0),4);X$(A(1),4);X$(A(2),
4)
430 MUSIC X$(A(0),5);X$(A(1),5);X$(A(2),
5)
440 WAIT 2000 :GOSUB 550
450 MUSIC X$(A(0),0);X$(A(1),0);X$(A(2),
0)
460 MUSIC X$(A(0),1);X$(A(1),1);X$(A(2),
1)
470 WAIT 5000 :GOSUB 550
480 MUSIC X$(A(0),0);X$(A(1),0);X$(A(2),
0)
490 MUSIC X$(A(0),6);X$(A(1),6);X$(A(2),
6)
500 WAIT 7000
510 FOR I=0 TO 2 :A(I)=0 :NEXT I
520 GOTO 210
530 '
540 LABEL "TEKST"
550 BOX 8,100,100,192,2
560 READ X$
570 SYMBOL 8,107,X$,11,11
580 BOX 110,100,312,192,2
590 Y=110
600 FOR I=0 TO 10
610 READ X$ :IF X$="@" THEN I=10 :GOTO 6
20 :ELSE SYMBOL 118,Y,X$,1,1 :Y=Y+8
620 NEXT I
630 RETURN
640 LABEL "IN"
650 RESTORE 770 :FOR J=1 TO 4
660 FOR I=0 TO 7
670 READ X$(J,I) :NEXT I,J
680 RETURN
690 DATA,Keuze uit max.3 stemmen,,1 Sopr
aan,2 Alt,3 Tenor,4 Bas,,5 Start melodie
,,E Einde,@
700 DATA 1,Oh! how we danced,in the nigh
t,we were wed;,,We vowed our true love,t
hough a word wasn't,said,@
710 DATA 2,The world was in bloom,there
were stars in the,skies,,except for the
few,that were there in your,eyes,@
720 DATA R,Dear as I held you,so close i
n my arms,Angels were singing,a hymn to
your charms,Two hearts gently,beating we
re murmering,low; My darling,,I LOVE YOU
 so,@
```

```
730 DATA 3,The night seemed to fade,into
 blossoming dawn,,The sun shone a-new,bu
t the dance,lingered on,@
740 DATA 4,Could we but relive,that swee
t moment,sublime,,We'd find that our lov
e,is unaltered by time,@
750 '
760 'DATA Sopraan
770 DATA O3-B8 -B5R0#D5R0E5R0 #F8 #F5R0#
D5R0-B5R0 G8 G5R0#F5R0E5R0 B8B7R0
780 DATA B5 +C8 +C5R0B5R0A5R0 B8 B5R0A5R
0G5R0 #F8 #F5R0G5R0#F5R0 E8 E7R0-B5R0
790 DATA +E5R0+E5R0+E5R0 +E5R0+E5R0+E5R0
 +E5R0+D6R0+#C3R0 +D8R0
800 DATA +C5R0+C5R0+C5R0 +C5R0+C5R0+C5R0
 +C5R0B6R0A3R0 B7R0B5R0
810 DATA A5R0A5R0A5R0 A5R0A5R0A5R0 A5R0G
5R0#F5R0 E7R0+E5R0
820 DATA +C5R0A5R0#F5R0 G7R1#F5R0 E8 E5R
-BR0
830 DATA B5 +C8 +C5R0B5R0A5R0 B8 B5R0A5R
0G5R0 #F8 #F5R0G5R0#F5R0  E8 E8
840 DATA B5 +C8 +C5R0B5R0A5R0 B8 B5R0A5R
0G5R0 #F8 #F5R0G5R0#F5R0  E8 E7R5
850 '
860 'DATA Alt
870 DATA -B8 -B5R0-B5R0-B5R0 -B8 -B5R0-B
5R0-B5R0 E8 E5R0-B5R0-B5R0 E8E7R0
880 DATA E5 E8 E5R0E5R0E5R0 E8 E5R0E5R0E
5R0 -B8 -B5R0E5R0-B5R0 -B8 -B7R0-B5R0
890 DATA G5R0G5R0G5R0 #F5R0G5R0A5R0 G5R0
G6R0G3R0    G8R0
900 DATA G5R0G5R0G5R0 #F5R0G5R0A5R0 G5R0
G6R0G3R0   G7R0G5R0
910 DATA E5R0E5R0E5R0 #D5R0E5R0#F5R0 #F5
R0E5R0#D5R0 E7R0G5R0
920 DATA E5R0E5R0E5R0   E5R0#C5R0#D5R0 E
8 E5R-BR0
930 DATA E5 E8  E5R0E5R0E5R0 E8 E5R0E5R0
E5R0 -B8 -B5R0E5R0-B5R0 -B8 -B8
940 DATA E5 E8  E5R0E5R0E5R0 E8 E5R0E5R0
E5R0 -B8 -B5R0E5R0-B5R0 -B8 -B7R5
950 '
960 'DATA Tenor
970 DATA -A8 -A5R0-A5R0-A5R0 -A8 -A5R0-A
5R0-A5R0 -B8 -B5R0-A5R0-G5R0 -B8-B7R0
980 DATA -B5 C8 C5R0C5R0C5R0 -B8 -B5R0C5
R0-B5R0 -A8 -A5R0-#A5R0-A5R0 -G8-G7R0-G5
R0
990 DATA C5R0C5R0C5R0 C5R0C5R0C5R0 C5R0-
B6R0-#A3R0 -B8R0
1000 DATA E5R0E5R0E5R0 D5R0D5R0D5R0  E5R
0D6R0#C3R0  D7R0D5R0
1010 DATA C5R0C5R0C5R0 -B5R0-B5R0-B5R0 C
5R0-B5R0-A5R0 -G7R0-B5R0
```

```
1020 DATA -A5R0-#F5R0-A5R0 -B5R0-#A5R0-A
5R0 -G8-G5R-GR0
1030 DATA -B5 C8 C5R0C5R0C5R0-B8-B5R0C5R
0 -B5R0 -A8 -A5R0-#A5R0-A5R0 -G8-G8
1040 DATA -B5 C8 C5R0C5R0C5R0-B8-B5R0C5R
0 -B5R0 -A8 -A5R0-#A5R0-A5R0 -G8-G7R5
1050 '
1060 'DATA Bas
1070 DATA O1 #D8 D5R0-B5R0#C5R0 #D8 D5R0
-B5R0#D5R0 E8 E5R0E5R0E5R0 G8G5R0
1080 DATA G5 A8 A5R0G5R0#F5R0 G8 G5R0#F5
R0E5R0 #D8 D5R0#C5R0#D5R0 E8 E7R0E5R0
1090 DATA A5R0A5R0A5R0 D5R0E5R0#F5R0 G5R
0G6R0#G3R0 G8R0
1100 DATA A5R0A5R0A5R0 D5R0E5R0#F5R0 G5R
0G6R0G3R0 G7R0G5R0
1110 DATA #F5R0#F5R0#F5R0 -B5R0#C5R0#D5R
0 E5R0E5R0E5R0 E7R0E5R0
1120 DATA -A5R0C5R0C5R0 -B7R1-B5R0  E8 E
5RER0
1130 DATA G5 A8 A5R0G5R0#F5R0 G8 G5R0#F5
R0E5R0 #D8 D5R0#C5R0#D5R0 E8 E8
1140 DATA G5 A8 A5R0G5R0#F5R0 G8 G5R0#F5
R0E5R0 #D8 D5R0#C5R0#D5R0   E8 E7R5
```

Programma 2: De Bolero.

De volgende melodie is een deel van de Bolero. Dit programma komt, evenals het programma dat hierna volgt, uit het boek Alles über den MZ-800 van BBG.

```
10 TEMPO 5
20 DIM M$(3,10)
30 FOR J=1 TO 3
40 FOR I=1 TO 9
50 READ M$(J,I)
60 NEXT I,J :FOR I=1 TO 10
70 MUSIC M$(1,I);"O2"+M$(2,I);M$(3,I)
80 NEXT I
90 DATA O3V15S0M10
100 DATA +C5B1+C+D+CBA+C3+C1A+C5B1+C
110 DATA A1GEFG6R1FEDEFGAG6
120 DATA A1BAGFEDEDC5C1DM6E3FM10
130 DATA D5M20G9R4M10+D6+C1BAB+C+D+CB4
140 DATA +C1BA+C1BAF4M5F1FF3A+C1ABD
150 DATA F3F1FF3AM10B1GAF
160 DATA D3D1CD6M5D1DD3FM10A1FGED3D1C
170 DATA D6D1CD3E1FM15G7M10R1FEDC9
180 REM ----------------------------
190 DATA V10S0M10
200 DATA E5D1CDCDFE3G1FE5D1C
210 DATA F1ECDE6R1FEGCDEFE6
220 DATA F1DECDGBGBG5C1GM6A3GM10
230 DATA B5M20E9R4M10B6G1GFDEGEG4
```

```
240 DATA E1GFEGFA4M5D1DA3FE1FGG
250 DATA A3A1AC3FM10G1GFA
260 DATA G3F1EB6M5D1FG3AM10C1GECB3G1E
270 DATA B6G1ED3G1FM15E7M10R1DGGG9
280 REM ---------------------------------
290 DATA O1V11S0M5
300 DATA C3C1C0CC3C1C0CC3CCE1E0EE3E1E0E
310 DATA C3C1C0CC3C1C0CC3CCC1C0CC3C1C0CG
1G0GG1G0G
320 DATA C3C1C0CC3CCC1C0CG3G1G0GG1G0G
330 DATA C3C1C0CC3C1C0CG3GGG1C0CC3C1C0CC
3C1C0C3C1C0CC3C1C0CC3C1C0C
340 DATA G3G1G0GG3G1G0GC3CCC1C0C
350 DATA C3C1C0CC3CCC1C0C
360 DATA G3C1E0EE3E1E0GG3GCG1C0CC3C1C0C
370 DATA G3G1G0GG3G1G0GG3GCG1C0CC3C1C0CC
3C1C0CC3C1C0CC1C0C
```

Programma 3: Nog een leuke melodie van BBG.

De volgende melodie is ook een melodie uit het boek Alles über den MZ-800 van BBG.

```
10 TEMPO 6
20 FOR JT=1 TO 2
30 FOR I=1 TO 10
40 READ A$,B$,C$
50 MUSIC A$;B$;C$
60 NEXT I
70 RESTORE "ENTRY"
80 NEXT JT
90 END
100 DATA O3V15S0M10
110 DATA O2V13S0M10
120 DATA O2V11S0M10
130 '''''''''''''''''''''''''''''''''''''''
140 DATA R9
150 DATA D5E3#FA#F1EM4D3E
160 DATA R9
170 '''''''''''''''''''''''''''''''''''''''
180 DATA R9
190 DATA #F#FEEDR6
200 DATA R9
210 LABEL "ENTRY"
220 '''''''''''''''''''''''''''''''''''''''
230 DATA M10#F3AA5M20A7
240 DATA M6#F5M13A7M4+#C0+D2A3
250 DATA M6D5M20#F6M4R5
260 '''''''''''''''''''''''''''''''''''''''
270 DATA M10#F3AA5M20A7
280 DATA M6D5M20#F7M4R5
290 DATA M6#F5M13A7M4+#C0+D2A3
300 '''''''''''''''''''''''''''''''''''''''
310 DATA M10R3BAABBAA
```

```
320 DATA R3M10B3A5M4B3RAA
330 DATA V15-D5-D-D-D
340 ''''''''''''''''''''''''''''''''''''
350 DATA #F3ABGM18A7M10
360 DATA #F3ABGAM10D1E#F5
370 DATA -D-D-D-DV12
380 ''''''''''''''''''''''''''''''''''''
390 DATA R3#FAA#FAAA
400 DATA M5R3DM15#F5M10R3#F3AA
410 DATA M5R3M15#F5M10R3D#F#F
420 ''''''''''''''''''''''''''''''''''''
430 DATA B3B1+#C+D3EA7
440 DATA R5M20D5D7M10
450 DATA R5M20-B5-A7M10
460 ''''''''''''''''''''''''''''''''''''
470 DATA R3+DA#FEBBB
480 DATA M3R2+D5M10R0R3M20E7M10
490 DATA M3R1A5M10R4M20B7M10
500 ''''''''''''''''''''''''''''''''''''
510 DATA +D3A1#FE5M14#F7M10
520 DATA +D3R5#G0A2+D3+B+A+#F1+E
530 DATA A3R3R7
540 ''''''''''''''''''''''''''''''''''''
550 DATA R9
560 DATA O3M10D5E3#F3M4A3M10#F1ED3M4E3
570 DATA R9
580 ''''''''''''''''''''''''''''''''''''
590 DATA R8+D5
600 DATA M3#F3#FEEDR3+A5
610 DATA M4+D3+DAA#FR3+#F5
```

Programma 4: Leuk eindmuziekje.

Dit kleine melodietje is uitstekend te gebruiken als melodietje wanneer U dood gaat in een spel.

```
MUSIC"T601D6R0D5R0D2R0D6F6E5D3R0D5C3D5";
"T602S6M100D6R0D5R0D2R0D6F6E5D3R0D5C3D5"
```

Programma 5: Bekend Frans melodietje.

Als laatste melodietje het beroemde Franse Can Can melodietje. Bij dit melodietje wordt slechts 1 akkoord gebruikt.

```
10 MUSIC"T7O4C9D5F5E5D5G7G7G5A5E5F5"
20 MUSIC"D7R0D7D5F5E5D5C7R0C9"
30 MUSIC"D5F5E5D5G7R0G7R0G5A5E5F5"
40 MUSIC"R0D7R0D7R0D5F5E5D5C5G5D5E5C9"
50 MUSIC"C9D5F5E5D5G7G7G5A5E5F5"
60 MUSIC"D7D7D5F5E5D5C5+C5B5A5G5F5E5D5"
70 MUSIC"C9D5F5E5D5G7G7G5A5E5F5"
80 MUSIC"D7D7D5F5E5D5C5G5D5E5C9"
90 MUSIC WAIT :CLS :END
```

HOOFDSTUK 6: GRAFISCHE TOEPASSINGEN.

Wat wordt er nu precies verstaan onder grafische toepassingen? Worden daar grafieken, wiskundige figuren of tekeningen mee bedoeld?
Eigenlijk komen al deze dingen voor in dit hoofdstuk. Met grafische toepassingen wordt een heel ruim gebied bedoeld van wiskundige figuren tot animaties en natuurlijk ook een heel belangrijk onderdeel is de zakelijke toepassing aan de hand van twee- en driedimensionale taart- en staafdiagrammen. In het eerste deel van dit hoofdstuk worden een aantal programmaatjes getoond die leuke wiskundige figuren tekenen.

6.1: Leuke wiskundige figuren.

SPIROGRAAF.

```
1 'SPIROGRAAF
2 INIT"CRT:M1" :K=1.7 :PAL 3,3
3 Z=3.2 :M=50 :N=31 :P=600 :PAL 0,15
4 READ T,A,B,F,C :IF A>0 THEN 6
5 BEEP :COLOR 1 :END
6 S=(A+B)/2 :E=A-S :H=T/60*π :D=H :IF K=
3 THEN P=240
7 IF T=20 THEN P=240
8 IF T=30 THEN P=125
9 FOR I=T TO T+P :IF C=2 THEN X=M+SIN(H)
*S+SIN(D)*E :Y=N+COS(H)*S+COS(D)*E
10 H=I/60*π :D=(T+F*(I-T))/60*π
11 IF C=0 THEN X=M :Y=N
12 IF C=1 THEN X=M+SIN(H)*S :Y=N+COS(H)*
S
13 LINE [K]X*Z,Y*Z,(M+SIN(H)*S+SIN(D)*E)
*Z,(N+COS(H)*S+COS(D)*E)*Z
14 NEXT I
15 K=K+.7
16 GOTO 4
17 DATA 0,25,15,4.4,2,10,15,9,-2.4,1,20,
9,5,-2.5,0,30,30,20,1,1,0,0,0,0,0
```

Wanneer U de VIDEO-RAM uitbreiding heeft, kunt U het effect iets leuker maken door regel 2 en 14 te veranderen in:

```
2 INIT"CRT:M2" :K=1
14 K=K+1
```

Wave.

```
10 INIT"CRT:M1" :B=0
20 FOR A=1 TO 15 STEP .1
30 SET B*10,100+COS(A)*20
40 B=B+SIN(A)
50 IF SIN(A)<0 THEN B=B+1
60 IF SIN(A)>0 THEN B=B-.5
70 NEXT A
```

```
80 GET A$ :IF A$="" THEN 80
90 CLS :END
```

Mozaïek.

```
10 INIT"CRT:M1" :PAL 3,7 :B=.1
20 FOR A=0 TO 2*π STEP π/120
30 X=SIN(A+B*π)*70+160
40 Y=COS(A+B*π)*70+100
50 X2=SIN(A-B*π)*70+160
60 Y2=COS(A-B*π)*70+100
70 LINE X2,Y2,X,Y
80 B=B+.1 :NEXT A
90 GET A$ :IF A$="" THEN 90
100 CLS :END
```

Kussen.

```
10 INIT"CRT:M1"
20 FOR A=0 TO 2*π STEP π/3
30 FOR X=0 TO 2*π STEP .1
40 LINE (SIN(X+A)+COS(X+A))*100+160,SIN(
X)*80+100,(SIN(X+.1+A)+COS(X+.1+A))*100+
160,SIN(X+.1)*80+100
50 NEXT X,A
60 GET A$ :IF A$="" THEN 60
70 CLS :END
```

Hoed.

Eventjes wachten voordat de tekening klaar is. Het duurt slechts 28 minuten en 47 seconden.

```
10 INIT"CRT:M3 :PAL 1,0 :PAL 0,7
20 W=45 :K=.3125 :A=180 :K1=45
30 U=320 :V=80 :H=.5 :RD=4*ATN(1)/180
40 C=K*COS(W*RD) :S=K*SIN(W*RD)
50 DX=3 :DY=8 :AF=A/210
60 DIM H(320)
70 FOR L=0 TO 320
80 H(L)=1000
90 NEXT L
100 FOR YY=-210 TO 210 STEP DY
110 Y=YY*AF
120 FOR XX=-210 TO 210 STEP DX
130 X=XX*AF :R=SQR(X*X+Y*Y)*RD
140 Z=K1*(COS(R)-COS(3*R)/3+COS(5*R)/5-C
OS(7*R)/7)
150 XG=INT(U+XX+C*YY+H)
160 YG=INT(V-S*YY-Z+H)
170 IF XX>-210 THEN 210
180 F1=0 :L=INT(XG/DX)
```

```
190 IF YG<=H(L) THEN F1=1 :H(L)=YG
200 X1=XG :Y1=YG :GOTO 260
210 F2=0 :L=INT(XG/DX)
220 IF YG<=H(L) THEN F2=1 :H(L)=YG
230 X2=XG :Y2=YG :IF F1*F2<>1 THEN 250
240 LINE X1,Y1,X2,Y2
250 X1=X2 :Y1=Y2 :F1=F2
260 NEXT XX,YY
270 GET I$ :IF I$="" THEN 270 ELSE CLS :
END
```

3-D computerberg.

Dit programma duurt nog even iets langer dan het vorige programma, ongeveer het dubbele. Dit programma tekent een 3-D computerberg. De lijnen die U normaal niet ziet, ziet U hie wel.

```
10 INIT"CRT:M3" :A=0 :PAL 1,0 :PAL 0,7
20 FOR Y=-2.5 TO 2.5 STEP .1 :B=0
30 FOR X=-2 TO 2 STEP .025
40 Z=(3*X*X+Y*Y)*EXP(1-X*X-Y*Y)
50 SET X*120+250+A,195-Z*40-B
60 B=B+.5 :A=A+.02 :NEXT X,Y
70 B=0 :FOR X=-2 TO 2 STEP .1 :A=0
80 FOR Y=-2.5 TO 2.5 STEP .025
90 Z=(3*X*X+Y*Y)*EXP(1-X*X-Y*Y)
100 SET X*120+250+A,195-Z*40-B
110 A=A+.8 :NEXT Y :B=B+2 :NEXT X
120 GET A$ :IF A$="" THEN 120
130 CLS :END
```

Figuren zonder naam.

Nu volgen enkele wiskundige figuren die geen naam hebben. Ze zien er allemaal wel leuk uit, maar lijken nergens op; vandaar.

```
1 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
2 A=160 :B=100 :H=1/2 :RD=4*ATN(1)/180
3 FOR K=-40 TO 40 STEP 10 :X1=A+60 :Y1=B
4 FOR W=2 TO 360 STEP 2 :P=W*RD :R=60+K*
SIN(4*P) :X2=INT(A+R*COS(P)+H)
5 Y2=INT(B-R*SIN(P)+H)
6 LINE X1,Y1,X2,Y2 :X1=X2 :Y1=Y2
7 NEXT W,K
```

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
20 A=160 :B=100 :H=1/2 :RD=4*ATN(1)/180
30 FOR K=20 TO 100 STEP 10
40 P=0 :R=COS(4*SIN(2*P))
50 X1=A+K*R*COS(P)+H :Y1=B-K*R*SIN(P)+H
60 FOR W=2 TO 360 STEP 2
70 P=W*RD :R=COS(4*SIN(2*P))
80 X2=A+K*R*COS(P)+H
```

```
90 Y2=B-K*R*SIN(P)+H
100 LINE X1,Y1,X2,Y2
110 X1=X2 :Y1=Y2
120 NEXT W,K
```

Bloem.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
20 A=160 :B=100 :H=1/2 :RD=4*ATN(1)/180
30 N=4 :C=.25
40 FOR K=30 TO 80 STEP 10
50 X1=A+K :Y1=B
60 FOR W=3 TO 360 STEP 3
70 P=W*RD :R=K*(1+C*ABS(SIN(N*P)))
80 X2=A+R*COS(P)+H
90 Y2=B-R*SIN(P)+H
100 LINE X1,Y1,X2,Y2
110 X1=X2 :Y1=Y2
120 NEXT W,K
130 R=30 :P1=(180/N)*RD
140 FOR J=1 TO N
150 P=J*P1
160 X1=A+R*COS(P)+H
170 Y1=B-R*SIN(P)+H
180 X2=A+R*COS(P+4*ATN(1))+H
190 Y2=B-R*SIN(P+4*ATN(1))+H
200 LINE X1,Y1,X2,Y2
210 NEXT J
```

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
20 U=160 :V=100 :H=1/2 :RD=4*ATN(1)/180
30 KX=10 :KY=10
40 FOR F=1 TO 7 :READ A,B
50 FOR N=-3 TO 3
60 T=0 :GOSUB 150
70 X1=U+KX*X+H :Y1=V-KY*Y+H
80 FOR W=2 TO 360 STEP 2
90 T=W*RD :GOSUB 150
100 X2=U+KX*X+H :Y2=V-KY*Y+H
110 LINE X1,Y1,X2,Y2
120 X1=X2 :Y1=Y2
130 NEXT W,N
140 WAIT 5000 :CLS :NEXT F :END
150 X=(A+B)*COS(T)-N*B*COS((A+B)/B*T)
160 Y=(A+B)*SIN(T)-N*B*SIN((A+B)/B*T)
170 RETURN
180 DATA -6,1,-6,2,-8,2,4,1,4,2,6,1,4.5,
1.5
```

Op de volgende bladzijde treft U nog twee kleine programmaatjes die ook iets tekenen aan de hand van eenvoudige wiskundige formules. Daarbij zult U zien dat eenvoudigheid ook leuk kan zijn.

Zoals al aangekondigd op de vorige bladzijde, volgen nu nog twee kleine programmaatjes die iets leuks doen met wiskundige formules. Bij beide programma's wordt er ook met kleur gewerkt. Met een kleurenscherm of kleurentelevisie komt het effect dus het beste uit. Er worden ook meer dan vier kleuren gebruikt. Het is dus noodzakelijk om de uitbreidings-IC's in Uw computer te hebben om de programma's voor 100% te laten werken.

Kleurenbol.

```
10 INIT"CRT:M2" :PAL 0,7
20 R=50 :N=6 :P=π/N :W=3 :V=1
30 FOR I=0 TO N-1
40 K=P*I :A=R*COS(K) :B=R*SIN(K) :J=30*I
50 CIRCLE [V]160+A,B+100,R
60 V=V+1 :IF V=5 THEN V=1
70 CIRCLE [W]160-A,100-B,R
80 W=W+1 :IF W=5 THEN W=1
90 NEXT I
```

4-kleuren 'ster'.

```
10 INIT"CRT:M2" :PAL 0,7
20 X=240 :Y=200 :N=10
30 XX=120 :YY=100
40 LINE 0,100,X,100
50 LINE XX,YY+100,XX,-YY+100
60 FOR I=1 TO N
70 X1=XX+XX/N*I
80 Y2=YY-(YY/N*(I-1))
90 X3=XX-XX/N*I
100 LINE [1]X1,100,XX,Y2+100
110 LINE [2]XX,Y2+100,X3,100
120 LINE [3]X3,100,XX,-Y2+100
130 LINE [4]XX,-Y2+100,X1,100
140 NEXT I
```

Dit waren een aantal voorbeelden van wiskundige figuren. Er zaten heel gemakkelijke figuren bij, maar ook heel moeilijke. Voor iemand die weinig wiskunde heeft gehad is het moeilijk te begrijpen hoe de figuren precies gevormd worden. Zelfs voor mensen die goed in wiskunde zijn, is het regelmatig moeilijk te begrijpen hoe de formules nu precies in elkaar zitten.
De figuren die U hier ziet zijn ontleend aan eigenstudie, maar ook aan een aantal boeken. De programma's uit die boeken, die eigenlijk voor de PC geschreven zijn, zijn uitgepluisd en daarna herschreven voor de SHARP MZ-800. De boeken waar we het hier over hebben, zijn:

GRAPHICS VOOR MICROCOMPUTERS van KLUWER.
en
40 GRAFISCHE PROGRAMMA'S IN IBM- EN GW-BASIC van ACADEMIC SERVICE.

In die boeken staat ook een uitleg over hoe alles precies in elkaar zit. Wilt U dus meer weten dan kunt U het in die boeken opzoeken.

6.2: Leuke toepassingen met kleuren.

Door gebruik te maken van alle kleuren die tot Uw beschikking staan, kunnen heel leuke effecten bereikt worden en daarvoor heeft U geeneens bij alle programma's de extra ic's nodig. Wel is het noodzakelijk om een goede kleurenmonitor of kleurentelevisie te gebruiken om de effecten goed te kunnen zien.

Mengkleuren.

Onderstaand programma werkt met de 16 kleuren die de SHARP heeft. Door de lijnen van twee verschillende kleuren, gestuurd via PAL 1 en PAL 2, vlak naast elkaar te zetten, krijgt U een mengkleur te zien. Bovenaan kunt U zien welke twee kleuren gemengd worden.

```
10 INIT"CRT:M1" :B=0
20 CURSOR 1,0 :PRINT"PAL 1, 0    PAL 2,
0"
30 FOR T=0 TO 15
40 FOR Q=B TO 15
50 PAL 1,Q :PAL 2,T
60 CURSOR 7,0 :PRINT Q;" " :CURSOR 19,0
:PRINT T;" "
70 CURSOR 25,0 :PRINT [1]CHR$(200); :PRI
NT" + "; :PRINT [2]CHR$(200)
80 FOR A=8 TO 320 STEP 2 :LINE [2]A,8,A,
199 :LINE [1]A+1,8,A+1,199
90 NEXT A :WAIT 1000 :BOX [0]8,8,320,199
,0 :BEEP :NEXT Q :B=B+1 :NEXT T
100 CLS :END
```

Willekeurige boxen.

```
10 INIT"CRT:M1" :R=4
20 FOR A=1 TO 250
30 B=INT(RND(1)*R)
40 E=INT(RND(1)*320) :F=INT(RND(1)*200)
50 C=INT(RND(1)*320) :D=INT(RND(1)*200)
60 BOX [B]E,F,C,D,B
70 NEXT A
80 GET A$ :IF A$="" THEN 80
90 CLS :END
```

Wanneer U de extra ic's in Uw computer heeft, kunt U het effect iets leuker maken door regel 10 te veranderen in:

```
10 INIT"CRT:M2" :R=16
```

Willekeurige cirkels.

Op de volgende bladzijde treft U een soortgelijk programma als het vorige programma aan, alleen worden nu cirkels met een willekeurige grootte in een willekeurige kleur op het scherm gezet.

```
10 INIT"CRT:M1" :R=4
20 FOR A=1 TO 100
30 B=INT(RND(1)*R)
40 E=INT(RND(1)*320)
50 F=INT(RND(1)*200)
60 C=INT(RND(1)*100)
70 CIRCLE [B]E,F,C
80 PAINT [B]E,F,B
90 NEXT A
100 GET A$ :IF A$="" THEN 100
110 CLS :END
```

Ook bij dit programma geldt dat U het effect mooier kunt maken wanneer U in het bezit bent van de extra ic's, door regel 10 te veranderen in:

```
10 INIT"CRT:M2" :R=16
```

Willekeurige lijnmotieven.

Om in de sfeer van de willekeurigheid te blijven, volgt hier nog een programma dat willekeurig een kleur kiest, maar het patroon van de lijnen blijft elke keer hetzelfde. Ondanks het feit dat er maar drie kleuren worden gebruikt, lijkt het uit veel meer kleuren te bestaan.

```
10 INIT"CRT:M1" :Z=3
20 XM=319 :YM=199
30 XC=XM/2 :YC=YM/2
40 FOR X=0 TO XM
50 C=INT(RND(1)*Z)+1
60 COLOR C,0
70 LINE XC,YC,X,0
80 NEXT X
90 FOR Y=0 TO YM
100 C=INT(RND(1)*Z)+1
110 COLOR C,0
120 LINE XC,YC,XM,Y
130 NEXT Y
140 FOR X=XM TO 0 STEP -1
150 C=INT(RND(1)*Z)+1
160 COLOR C,0
170 LINE XC,YC,X,YM
180 NEXT X
190 FOR Y=YM TO 0 STEP -1
200 C=INT(RND(1)*Z)+1
210 COLOR C,0
220 LINE XC,YC,0,Y
230 NEXT Y
240 GET A$ :IF A$="" THEN 240
250 CLS :END
```

Wanneer U de extra ic's heeft, kunt U het effect van dit programma leuker maken door regel 10 te veranderen in:

```
10 INIT"CRT:M2" :Z=15
```

Alle 256 kleurcombinaties bij elkaar.

Dit programma gebruikt alle 16 kleuren tegelijk. U moet dus de extra ic's in Uw computer hebben om dit programma te laten draaien.

```
10 INIT"CRT:M2" :E=0 :F=0
20 FOR A=0 TO 127 STEP 8
30 FOR B=97 TO 224 STEP 8
40 FOR C=B TO B+7 STEP 2
50 LINE [E]C,A,C,A+7
60 LINE [F]C+1,A,C+1,A+7
70 NEXT C
80 E=E+1 :IF E=16 THEN E=0 :F=F+1
90 NEXT B,A
100 GET A$ :IF A$="" THEN 100
110 CLS :END
```

Gezichtsbedrog.

Ook bij dit programma zijn de extra ic's nodig. U hoeft geen kleurenscherm te gebruiken, omdat er maar twee kleuren gebruikt worden.

```
10 INIT"CRT:M2" :C=1 :PAL 3,7
20 SYMBOL 103,90,"Gezichtsbedrog",1,1
30 SYMBOL 128,99,"Welke rij",1,1
40 SYMBOL 140,107,"draait",1,1
50 SYMBOL 144,116,"welke",1,1
60 SYMBOL 132,125,"kant op?",1,1
70 CIRCLE 160,100,99
80 CIRCLE 160,100,58
90 FOR A=0 TO 2*π STEP π/24
100 X=SIN(A)*89+160
110 X1=SIN(2*π-A+π/48)*79+160
120 X2=SIN(A)*69+160
130 Y=COS(A)*89+100
140 Y1=COS(2*π-A+π/48)*79+100
150 Y2=COS(A)*69+100
160 CIRCLE [C]X,Y,4 :PAINT [C]X,Y,C
170 CIRCLE [C]X1,Y1,4 :PAINT [C]X1,Y1,C
180 CIRCLE [C]X2,Y2,4 :PAINT [C]X2,Y2,C
190 C=C+1 :IF C=4 THEN C=1
200 NEXT A
210 C=1 :FOR A=0 TO 200 STEP 3
220 BOX [C]0,A,30,A+2,C
230 BOX [C]31,200-A,60,200-A-2,C
240 BOX [C]260,200-A,290,200-A-2,C
250 BOX [C]291,A,320,A+2,C
260 C=C+1 :IF C=4 THEN C=1
270 NEXT A
280 FOR A=1 TO 100
290 PAL 1,0 :PAL 2,0 :PAL 3,7 :WAIT A*3
300 PAL 1,7 :PAL 2,0 :PAL 3,0 :WAIT A*3
310 PAL 1,0 :PAL 2,7 :PAL 3,0 :WAIT A*3
320 NEXT A
```

```
330 PAL 1,1 :PAL 2,2 :PAL 3,7
340 GET A$ :IF A$="" THEN 340
350 CLS :END
```

To the middle.

```
10 INIT"CRT:M1" :PAL 3,15 :PAL 0,8
20 A=0 :B=2 :C=1
30 BOX [0]-1,A,320,A+B,C
40 BOX [0]-1,200-A,320,200-A-B,C
50 A=A+B :B=B+.5
60 C=C+1 :IF C=4 THEN C=1
70 IF A<90 THEN 30
80 BOX [0]1,A+1,318,199-A,C
90 SYMBOL [0]80,A+5,"NEPTUNES PRODUCTION
S",1,1
100 FOR T=1 TO 50
110 WAIT 150 :PAL 1,15 :PAL 2,1 :PAL 3,2
120 WAIT 150 :PAL 1,2 :PAL 2,15 :PAL 3,1
130 WAIT 150 :PAL 1,1 :PAL 2,2 :PAL 3,15
140 NEXT T
150 SYMBOL 80,A+5,"NEPTUNES PRODUCTIONS"
,1,1
160 GET A$ :IF A$="" THEN 160
170 CLS :END
```

Kaleidoscoop.

```
10 INIT"CRT:M2"
20 A=15 :B=12 :C=0
30 FOR X=0 TO B-1
40 FOR Y=X TO B-1
50 Z=0
60 Z=C+X+Y :IF Z>31 THEN Z=Z-32
70 IF Z>15 THEN Z=Z-16
80 CURSOR A+X,Y+B :PRINT [Z]CHR$(200)
90 CURSOR A+X,-Y+B :PRINT [Z]CHR$(200)
100 CURSOR A-X,Y+B :PRINT [Z]CHR$(200)
110 CURSOR A-X,-Y+B :PRINT [Z]CHR$(200)
120 CURSOR A+Y,X+B :PRINT [Z]CHR$(200)
130 CURSOR A+Y,-X+B :PRINT [Z]CHR$(200)
140 CURSOR A-Y,X+B :PRINT [Z]CHR$(200)
150 CURSOR A-Y,-X+B :PRINT [Z]CHR$(200)
160 NEXT Y,X
170 C=C+1 :IF C=16 THEN C=0
180 GOTO 30
```

Dit programma stopt pas wanneer U op SHIFT/BREAK drukt.
Dit is gelijk het laatste programma in dit deel van hoofdstuk 6. Er zijn natuurlijk veel meer mogelijkheden met kleuren te bedenken, maar die creativiteit laten wij aan U over. Aan de hand van deze programma's heeft U al een aardig inzicht in de mogelijkheden op kleurengebied gekregen.

6.3: Leuke figuurtjes.

In dit deel van hoofdstuk 6 worden enkele leuke figuurtjes gedemonstreerd. De figuurtjes worden opgebouwd met de grafische instructies, zoals LINE, PATTERN en ga zo maar door.
Er zijn natuurlijk eindeloos veel mogelijkheden, waarvan hier een aantal getoond zullen worden. De figuurtjes zijn echter wel allemaal opgebouwd uit slechts 1 kleur, omdat de programma's anders veel te lang zouden worden. De eerste twee figuurtjes zijn twee figuurtjes die U wel vaker tegen zult komen, want ze worden door ons gebruikt bij het maken van programma's en boeken. Het zijn dus eigenlijk de voorlopige logo's van Neptunes Productions.

Figuur 1: De voorlopige logo van Neptunes Productions.

```
10 INIT"CRT:M1" :PAL 1,5
20 BOX [2]0,0,245,199,2
30 BOX [0]5,15,240,64,0
40 DATA 142,134,17,20,20,20,138,143,191,
133,17,20,20,20,20,135,140,153,130,17,20
,20,20,141,177
50 CURSOR 3,3 :FOR G=1 TO 25 :READ F :PR
INT CHR$(F); :NEXT G
60 LINE 8,39,56,39 :LINE 8,40,56,40
70 LINE 32,16,32,63 :LINE 31,16,31,63
80 CIRCLE 31,39,17 :CIRCLE 31,39,20 :PAI
NT 33,37 :PAINT 30,41
90 BLINE 50,39,49,39 :BLINE 50,40,49,40
100 BLINE 12,39,13,39 :BLINE 12,40,13,40
110 BLINE 31,20,31,21 :BLINE 32,20,32,21
120 BLINE 31,57,32,57 :BLINE 31,58,32,58
130 SYMBOL [1]89,20,"NEPTUNES",2,2
140 SYMBOL [1]65,45,"PRODUCTIONS",2,2
150 CURSOR 0,10
```

Figuur 2: De logo van programmamaker Arjan Habing.

```
10 DATA 0,22,31,0,18,21,32,37,0,15,17,38
,39,0,13,14,40,41,0,11,12,42,42,0,10,10,
43,43,0,9,9,27,27,40,40,44,44,0,8,8,44,4
4,0,8,8,10,13,23,23,25,26,38,38,44,44,0,
7,7,9,9,14,14,22,22,25,26,32,34,37,37,39
,40,43,43
20 DATA 0,7,8,11,12,15,15,30,31,35,35,40
,40,43,43,0,7,8,11,11,36,36,43,43,54,57,
0,7,7,9,9,12,12,27,27,44,44,52,53,57,60,
0,8,8,10,11,14,14,26,29,37,37,44,44,51,5
1,55,56,60,63
30 DATA 0,9,9,12,13,24,25,29,36,38,38,44
,44,50,51,54,54,57,59,64,65,0,10,11,17,1
7,32,34,43,44,50,50,52,53,56,56,61,61,65
,65,0,11,17,42,43,50,50,54,55,58,60,66,6
6
```

```
40 DATA 0,7,13,17,18,40,41,51,51,55,58,6
5,65,0,6,9,14,15,19,20,38,39,52,52,56,56
,64,64,0,4,5,10,12,16,16,20,25,36,38,41,
47,53,53,63,63
50 DATA 0,3,3,13,13,17,17,20,20,26,35,48
,49,54,54,57,57,62,62,0,2,2,7,10,14,14,1
6,16,18,20,50,51,54,54,61,61,0,2,2,10,12
,14,15,19,19,52,53,61,61
60 DATA 0,3,3,11,11,14,14,18,18,53,53,61
,61,0,4,4,13,13,15,17,45,45,52,52,62,62,
0,5,6,11,15,46,46,52,52,62,62,0,6,7,10,1
0,14,14,25,25,44,44,47,47,53,53,62,62
70 DATA 0,5,5,14,14,47,49,61,61,0,4,4,15
,15,19,19,30,37,47,47,50,52,59,60,0,3,3,
16,16,19,19,26,29,38,39,48,48,53,58,0,2,
2,16,16,19,19,25,25,40,40,49,49
80 DATA 0,2,2,15,15,19,19,25,25,41,41,49
,49,0,2,2,18,18,24,24,42,42,50,50,0,3,3,
17,17,23,23,42,42,51,51,61,66,0,4,6,13,1
6,23,23,43,43,53,53,56,60,67,69
90 DATA 0,7,13,22,22,27,27,43,43,50,55,7
0,70,0,13,13,22,22,27,27,43,43,46,49,52,
52,56,56,71,71,0,14,14,21,21,25,25,28,28
,43,45,49,49,51,51,56,56,71,71
100 DATA 0,14,14,17,20,26,28,42,42,46,46
,48,48,55,55,61,61,72,72,0,15,16,19,19,2
9,29,41,41,45,45,55,55,57,57,61,61,73,73
,84,85,0,15,15,18,18,30,30,41,41,54,54,5
7,57,60,60,74,74,82,83,86,86
110 DATA 0,14,14,18,18,31,32,40,40,52,53
,55,55,58,58,60,60,75,75,80,81,86,86,0,1
3,13,19,19,33,34,39,39,50,52,54,54,56,57
,59,60,75,75,78,79,85,85
120 DATA 0,13,13,19,19,35,35,39,39,49,51
,53,53,55,55,57,59,61,61,76,77,85,85,0,1
3,13,15,15,20,21,36,36,48,50,52,52,54,54
,56,56,58,58,62,63,84,84
130 DATA 0,14,14,22,23,37,37,49,51,53,53
,55,55,57,57,64,65,82,83,0,15,15,18,18,2
4,25,38,38,49,50,52,52,54,54,56,56,58,58
,66,67,80,81
140 DATA 0,15,15,17,17,21,21,23,23,25,27
,37,37,39,39,48,49,51,51,53,53,55,55,57,
58,68,69,79,79
150 DATA 0,16,16,19,19,22,22,24,24,26,26
,28,30,36,36,38,38,40,41,47,48,50,50,52,
52,54,56,69,69,79,79
160 DATA 0,17,19,21,21,23,23,25,25,27,27
,29,29,31,35,37,37,39,39,41,41,43,43,45,
45,47,47,49,49,51,53,70,70,78,78
170 DATA 0,21,23,26,26,28,28,30,30,32,32
,34,34,36,36,38,38,40,46,48,50,71,71,77,
78,0,24,25,27,31,33,39,73,76
180 INIT"CRT:M1" :PAL 3,0 :PAL 0,7
190 H=2 :FOR R=1 TO 380
```

```
200 READ C :IF C=0 THEN READ B :H=H+1 EL
SE B=C
210 READ A :BOX B,H,A,H :NEXT R
220 SYMBOL 75,17,"BABY-programmer",2,3 :
CURSOR 0,8 :END
```

Figuur 3: Een hondje.

Een hele leuke tekening van een hondje.

```
10 INIT"CRT:M1" :PAL 3,0 :PAL 0,7 :Y=1
20 CURSOR 0,20
30 FOR T=0 TO 375
40 READ X1,X2
50 IF X1=0 THEN X1=X2 :Y=Y+1 :READ X2
60 LINE X1+110,Y+80,X2+110,Y+80
70 NEXT T
80 DATA 16,32,39,44,0,13,15,33,35,37,38,
44,45,0,11,12,45,46,0,9,10,44,44,46,47,0
,5,8,34,34,45,45,47,48,0,4,4,7,7,35,35,4
4,44,46,46,48,48,0,2,3,6,6,18,18,21,21,2
4,24,35,35,45,45,47,48
90 DATA 0,1,1,6,6,15,15,19,19,22,22,24,2
4,34,34,44,44,46,47,0,1,1,5,5,13,13,17,1
7,34,34,43,43,45,45,47,48,75,76,0,1,1,4,
6,15,15,22,22,24,25,34,34,44,44,46,46,48
,49,76,78
100 DATA 0,2,2,7,8,21,21,25,27,35,35,47,
47,50,50,77,80,0,3,3,9,10,21,21,24,27,35
,35,50,50,78,78,81,83,0,3,3,11,11,19,19,
22,22,24,26,36,36,47,47,50,50,77,77,80,8
0,82,82,84,85
110 DATA 0,4,4,10,10,18,18,20,20,37,38,4
4,44,48,49,76,76,81,81,83,83,85,86,0,3,3
,8,9,17,17,19,19,21,21,23,23,38,40,45,45
,49,50,75,75,84,84,86,86
120 DATA 0,2,2,7,7,16,16,18,18,20,20,37,
38,41,42,46,46,48,49,75,75,85,85,87,87,0
,2,2,6,15,17,17,25,25,36,41,43,43,46,47,
49,53,76,76,84,84,86,87
130 DATA 0,3,3,7,13,24,24,34,36,40,40,44
,45,54,59,77,77,81,81,85,86,0,4,4,9,12,2
2,23,32,39,60,62,77,78,82,82,84,84,86,86
,0,5,8,12,13,20,21,28,31,35,37,63,66,76,
76,79,79,85,85,87,87
140 DATA 0,14,18,25,35,67,68,76,76,80,80
,84,84,87,87,0,18,26,30,32,69,73,77,78,8
6,86,0,25,29,74,76,79,80,87,87,0,26,27,7
7,79,81,81,88,88,0,25,25,80,81,88,88,0,2
4,24,82,82,88,88
150 DATA 0,13,19,24,24,35,35,87,87,0,12,
13,20,21,25,25,34,34,66,66,86,86,0,10,11
,14,16,22,25,33,33,64,64,86,86,0,9,9,17,
19,24,25,32,32,61,61,63,63,65,65,86,86
```

```
160 DATA 0,10,14,26,26,32,32,60,60,62,64
,87,87,0,10,10,15,17,26,28,31,32,61,61,6
3,63,87,87,0,11,11,29,31,45,45,60,60,62,
62,87,87,0,12,13,42,42,46,46,57,57,61,61
,86,86,0,14,16,43,43,47,47,56,56,58,58,6
0,60,85,85
170 DATA 0,13,18,40,40,44,47,53,53,55,55
,57,57,59,60,85,85,0,11,13,18,21,37,37,4
1,43,48,49,51,51,54,54,56,56,58,60,82,82
,85,85,0,9,11,19,20,22,23,27,27,33,33,38
,38,41,41,51,51,53,54,56,58,61,61,83,83,
86,86
180 DATA 0,8,9,15,18,20,20,22,22,24,24,2
8,28,34,34,37,37,39,40,61,62,79,79,83,83
,86,86,0,7,8,13,14,17,19,21,21,23,23,25,
27,29,29,33,36,62,64,80,80,84,85,0,7,8,1
2,12,15,20,22,22,30,32,60,61,64,66,81,82
,85,85
190 DATA 0,7,9,12,12,14,14,16,17,19,19,5
9,60,66,66,68,69,78,78,82,82,84,84,0,9,1
1,14,14,16,16,54,59,65,65,67,67,70,70,79
,80,83,83,0,49,53,64,64,66,66,68,68,70,7
0,80,81,83,83
200 DATA 0,47,48,55,57,61,61,63,65,67,69
,80,80,82,83,0,47,47,52,54,60,60,62,64,6
6,66,69,69,81,81,0,48,52,57,57,59,64,78,
78,82,82,0,53,54,56,56,58,61,67,67,79,81
,0,54,58,65,66,72,73,80,80
210 DATA 0,57,57,63,64,70,70,79,79,0,58,
59,62,63,68,70,77,78,0,60,61,64,64,66,66
,68,68,71,72,74,75
```

Figuur 4: Een uil.

```
10 DATA 0,0,2,0,4,0,6,0,8,0,10,0,12,0,14
,0,16,0,1,1,7,1,9,1,15,1
20 DATA 0,2,4,2,8,2,12,2,16,2,3,3,5,3,11
,3,13,3,0,4,4,4,12,4,16,4
30 DATA 1,5,7,5,9,5,15,5,0,6,3,6,8,6,14,
6,16,6,1,7,3,7,13,7
40 DATA 0,8,2,8,4,8,6,8,8,8,10,8,12,8,16
,8,1,9,3,9,5,9,7,9
50 DATA 0,10,2,10,4,10,6,10,8,10,16,10,1
,11,3,11,5,11,7,11
60 DATA 2,12,4,12,6,12,8,12,16,12,3,13,5
,13,7,13,9,13
70 DATA 4,14,6,14,8,14,10,14,16,14,5,15,
7,15,9,15,11,15
80 DATA 6,16,8,16,10,16,12,16,16,16,7,17
,11,17,13,17,6,18,10,18,14,18,16,18
90 DATA 1,19,3,19,5,19,7,19,9,19,11,19,1
5,19,16,20
100 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
```

(Vervolg op de volgende bladzijde.)

```
110 INPUT"GROOTTE (0-8)";GR :IF GR<0 OR
GR>8 THEN 110
120 G=GR+1 :X1=(320-G*17)/2 :Y1=(200-G*2
1)/2 :CLS
130 FOR A=0 TO 97 :READ B,C :BOX B*G+X1,
C*G+Y1,B*G+GR+X1,C*G+GR+Y1,3 :NEXT A
140 GET A$ :IF A$="" THEN 140 ELSE CLS :
END
```

Figuur 4: Rijdende tank.

Eigenlijk behoren de bewegende beelden bij het animatiegedeelte, maar je kunt een tank natuurlijk moeilijk stil laten staan, daarvoor is een tank niet bedoeld.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
20 A$=CHR$(0,24,60,63,61,36,61,63,62,60,
36,61,61,60,60,38,63,61,60,61,39,60,24,0
)
30 B$=CHR$(24,24,24,231,231,231,231,231,
36,102,195,66,66,195,102,60,255,255,255,
255,255,0,0,0)
40 C$=CHR$(0,24,60,252,188,36,188,252,12
4,60,36,188,188,60,60,100,252,188,60,188
,228,60,24,0)
50 D$=A$+B$+C$
60 FOR Y=0 TO 224 :POSITION 150,Y :PATTE
RN 24,D$ :NEXT Y
70 FOR Y=200 TO -24 STEP -1 :POSITION 15
0,Y :PATTERN -24,D$ :NEXT Y
```

Figuur 5: Hoofd van Bengashie.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
20 A$=CHR$(239,111,95,96,31,120,74,120,9
5,223,39,190,167,167,51,25,95,86,182,103
,111,95,111,239)
30 B$=CHR$(66,231,255,60,165,165,165,165
,165,165,36,255,0,195,255,128,127,170,14
8,127,128,255,247,65)
40 C$=CHR$(247,246,250,6,248,30,82,30,25
0,244,116,228,142,226,242,126,174,174,10
2,246,250,254,237)
50 D$=A$+B$+C$
60 POSITION 150,90 :PATTERN -24,D$
```

Zo zijn er natuurlijk nog veel meer mogelijkheden, maar U heeft kunnen zien dat de mooiste resultaten bereikt worden door programma's die langer zijn dan 1 bladzijde. Kunt U nagaan hoe lang een programma is dat ook nog met verschillende kleuren werkt.
Voor mooie kleurenplaatjes verwijs ik U naar het programma PICTURESHOW van Neptunes Software. In dit programma treft U een aantal hele leuke plaatjes, nog veel mooier dan de plaatjes uit dit hoofdstuk.

6.4: Zakelijke toepassingen.

Aan de hand van taart-, gewone staaf- en 3-d staafgrafieken kunnen bepaalde cijfers op een overzichtelijke manier weergegeven worden. In dit deel van hoofdstuk 6 zullen enkele programma's getoond worden die laten zien hoe cijfers in een grafiek uitgezet kunnen worden. Aan de hand van die programma's kunt U een programma voor U zelf schrijven.

Programma 1: Gewone 2-d staafgrafiek.

Met dit programma kunt U gegevens uitzetten in een staafgrafiek. U kunt maximaal 50 gegevens invoeren en het maximum mag niet hoger zijn dan 197 en het minimum mag niet lager zijn dan 0. Voor dit programma zijn de uitbreidings-ic's noodzakelijk.

```
10 DIM A(50) :INIT"CRT:M2" :PAL 0,1
20 INPUT"HOEVEEL GEGEVENS WILT U INVOERE
N? (1-50)";D
30 IF D<1 OR D>50 THEN 20
40 CLS :FOR T=1 TO D
50 PRINT"GEGEVEN";T; :INPUT":";A(T)
60 IF A(T)<0 OR A(T)>197 THEN 190
70 NEXT T
80 PAL 2,5 :PAL 0,7
90 CLS :FOR Y=-1 TO 199 STEP 10 :LINE [1
]0,Y,319,Y :NEXT Y
100 S=0 :S$="0" :FOR A=192 TO 2 STEP -10
 :SYMBOL [1]2,A,S$,1,1
110 S=S+10 :S$=STR$(S) :NEXT A
120 G=1 :K=1 :FOR T=30 TO 315 STEP 286/D
 :BOX [1]T,199,T+286/D,199-A(K),G :K=K+1
 :G=G+1 :IF G=16 THEN G=1
130 NEXT T
140 GET A$ :IF A$="" THEN 140 ELSE RUN
```

Programma 2: 3-d staafdiagram.

Gegevens kunnen ook in een 3-d staafdiagram uitgezet worden. Begrijpelijk is natuurlijk dat het programma iets ingewikkelder in elkaar zit. Bij dit programma zijn de gegevens trouwens gegeven en hoeft U ze dus niet meer in te voeren. U kunt ze natuurlijk wel zelf veranderen in het programma. Het kan dan wel zo zijn dat de grafieken niet helemaal meer goed opgebouwd worden, omdat de voorste en achterste grafieken gedeeltelijk over elkaar heenvallen.
Dit programma werkt op iedere SHARP MZ-800.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0 :PAL 1
,8 :PAL 2,14
20 SYMBOL 0,0,"Aantal inwoners (x 1000)"
,1,1
30 FOR A=10 TO 109 STEP 10
40 BOX 20,A,200,A+10
50 SYMBOL 5,A-3,STR$(110-A),1,1
```

```
60 LINE 200,A,200+SIN(1/3*π)*100,A+COS(1
/3*π)*100
70 NEXT A
80 LINE 200,110,200+SIN(1/3*π)*100,110+C
OS(1/3*π)*100
90 LINE 20,110,20+SIN(1/3*π)*100,110+COS
(1/3*π)*100
100 LINE 200+SIN(1/3*π)*100,10+COS(1/3*π
)*100,200+SIN(1/3*π)*100,110+COS(1/3*π)*
100
110 LINE 20+SIN(1/3*π)*100,110+COS(1/3*π
)*100,200+SIN(1/3*π)*100,110+COS(1/3*π)*
100
120 FOR J=1960 TO 1980 STEP 10
130 SYMBOL 115+(J-1960)*6,162,STR$(J),1,
1
140 NEXT J
150 DATA COMPUDORP,50,65,77,DISKSTAD,76,
87,83
160 DIM X(3) :DIM Y(3)
170 READ S1$,X(1),X(2),X(3),S2$,Y(1),Y(2
),Y(3)
180 CURSOR 14,22 :PRINT [3]CHR$(200);" -
 ";S1$
190 CURSOR 14,24 :PRINT [2]CHR$(200); :P
RINT" - ";S2$;
200 S1=SIN(1/3*π)
210 FOR T=0 TO 2 :COLOR 1
220 C1=30+T*60+S1*70 :C2=30+T*60+S1*90 :
C3=50+T*60+S1*90 :C4=50+T*60+S1*70
230 LINE C1,145,C2,155,C3,155
240 LINE C1,145-Y(T+1),C2,155-Y(T+1),C3,
155-Y(T+1),C4,145-Y(T+1),C1,145-Y(T+1)
250 LINE C1,145,C1,145-Y(T+1) :LINE C2,1
55,C2,155-Y(T+1) :LINE C3,155,C3,155-Y(T
+1)
260 COLOR 2 :PAINT C1+1,145,1 :PAINT C2+
2,154,1 :PAINT C1+3,146-Y(T+1),1
270 NEXT T
280 FOR T=0 TO 2
290 COLOR 2 :C1=30+T*60+S1*10 :C2=30+T*6
0+S1*30 :C3=50+T*60+S1*30 :C4=50+T*60+S1
*10
300 LINE C1,115,C2,125,C3,125
310 LINE C1,115-X(T+1),C2,125-X(T+1),C3,
125-X(T+1),C4,115-X(T+1),C1,115-X(T+1)
320 LINE C1,115,C1,115-X(T+1) :LINE C2,1
25,C2,125-X(T+1) :LINE C3,125,C3,125-X(T
+1)
330 COLOR 3
340 PAINT C1+1,117-X(T+1),2
350 PAINT C2+2,126-X(T+1),2
360 PAINT C1+3,116-X(T+1),2
370 NEXT T
380 GET F$ :IF F$="" THEN 380
```

Programma 3: Taartdiagram.

Ook met taartdiagrammen bestaat de mogelijkheid om het twee- of drie-dimensionaal weer te geven. Ook kan een bepaald deel natuurlijk uit de taart gehaald worden en zo zijn er nog veel meer mogelijkheden.
In dit programma zullen we ons beperken tot een 2-d taartdiagram met het ouderwetse voorbeeld van appels en peren.
Dit programma werkt op iedere SHARP MZ-800.

```
10 DATA APPELS,PEREN,BANANEN,SINAASAPPEL
S
20 INIT"CRT:M1" :PAL 0,7 :PAL 3,0
30 DIM AT(4) :TT=0 :FOR A=1 TO 4
40 READ A$ :PRINT"HOEVEEL ";A$;" HEEFT U
?" :INPUT AT(A)
50 TT=TT+AT(A) :NEXT A
60 CLS
70 PRINT"U HEEFT IN TOTAAL";TT;" STUKKEN
 FRUIT." :PRINT
80 RESTORE :FOR A=1 TO 4 :READ A$
90 PRINT"WAARVAN "; :PRINT USING"##.##";
AT(A)/TT*100; :PRINT"% BESTAAT UIT ";A$
100 PRINT :NEXT A
110 PRINT :PRINT"DRUK OP EEN TOETS VOOR
EEN VERDELING IN EEN TAARTDIAGRAM."
120 GET F$ :IF F$="" THEN 120
130 CLS :CIRCLE 100,100,90
140 LINE 100,100,100,10
150 R=π :FOR A=1 TO 4
160 R=R+AT(A)/TT*π*2
170 LINE 100,100,SIN(R)*90+100,COS(R)*90
+100
180 PAINT [A-1]SIN(R-.02)*85+100,COS(R-.
02)*85+100,3
190 NEXT A
200 CURSOR 22,2 :PRINT"WIT - APPELS"
210 CURSOR 24,4 :PRINT [1]CHR$(200)+" -
PEREN"
220 CURSOR 24,6 :PRINT [2]CHR$(200)+" -
BANANEN"
230 CURSOR 24,8 :PRINT [3]CHR$(200)+" -
SINAASAPPELS"
240 CURSOR 0,0
```

Programma 4: Lijndiagram.

Een lijndiagram kan het beste gewoon 2-d gedaan worden en dat doen wij dus ook.
In dit programma moet U de gegevens weer zelf invoeren. Het programma zet de omzet over de afgelopen tien jaar uit tegen de kosten. Het tussenliggende gebied is dus de winst, cq het verlies. Dat gebied zal gearceerd worden. Verlies en winst krijgen een verschillende kleur en zijn dus gemakkelijk te onderscheiden.
Dit programma werkt alleen met de extra ic's.

```
10 INIT"CRT:M2" :DIM OZ(10) :DIM KT(10)
20 FOR A=1980 TO 1989
30 PRINT"WAT WAS DE OMZET IN";A; :INPUT
OZ(A-1979)
40 IF OZ(A-1979)<0 THEN BEEP :PRINT :PRI
NT"OM DE ZAKEN NIET TE MOEILIJK TE MAKEN
,  VERZOEK IK U OM DE OMZET NIET LAGER T
E  MAKEN DAN 0. HET IS TOCH SLECHTS ALS
EENVOORBEELD BEDOELD." :PRINT :GOTO 30
50 IF OZ(A-1979)>250 THEN BEEP :PRINT :P
RINT"OM DE ZAKEN NIET TE MOEILIJK TE MAK
EN,  VERZOEK IK U OM DE OMZET NIET HOGER
 TE  MAKEN DAN 250. HET IS TOCH SLECHTS
ALS  EEN VOORBEELD BEDOELD." :PRINT :GOT
O 30
60 PRINT"WAT WAREN DE KOSTEN IN";A; :INP
UT KT(A-1979)
70 IF KT(A-1979)<0 THEN BEEP :PRINT :PRI
NT"OM DE ZAKEN NIET TE MOEILIJK TE MAKEN
,  VERZOEK IK U OM DE KOSTEN NIET LAGER
TE MAKEN DAN 0. HET IS TOCH SLECHTS ALS
EENVOORBEELD BEDOELD." :PRINT :GOTO 60
80 IF KT(A-1979)>250 THEN BEEP :PRINT :P
RINT"OM DE ZAKEN NIET TE MOEILIJK TE MAK
EN,  VERZOEK IK U OM DE KOSTEN NIET HOGE
R TE MAKEN DAN 250. HET IS TOCH SLECHTS
ALS  EEN VOORBEELD BEDOELD." :PRINT :GOT
O 60
90 OZ(A-1979)=OZ(A-1979)/2
100 KT(A-1979)=KT(A-1979)/2 :NEXT A
110 CLS :PAL 0,7
120 FOR X=25 TO 277 STEP 28
130 FOR Y=0 TO 130 STEP 10
140 BOX [1]X,Y,X+28,Y+10
150 NEXT Y,X
160 FOR A=0 TO 260 STEP 20
170 SYMBOL [1]0,134-A/2,STR$(A),1,1 :NEX
T
180 A$="" " :C=80 :FOR B=12 TO 264 STEP 2
8 :D$=A$+STR$(C)
190 SYMBOL [9]B,143,D$,1,1
200 C=C+1 :NEXT B
210 BOX 0,157,319,199,15
220 SYMBOL [4]0,160,"GROEN = WINST",1,1
230 SYMBOL [10]0,170,"ROOD = VERLIES",1,
1
240 SYMBOL [2]0,180,"RODE LIJN = OMZET",
1,1
250 SYMBOL [3]0,190,"ZWARTE LIJN = KOSTE
N",1,1
260 X=25 :PAL 3,0
270 FOR T=1 TO 10
280 SYMBOL [2]X-4,137-OZ(T),"+",1,1
290 SYMBOL [3]X-3,136-KT(T),"x",1,1
300 IF T=10 THEN 320
```

```
310 LINE [2]X,140-OZ(T),X+28,140-OZ(T+1)
 :LINE [3]X,140-KT(T),X+28,140-KT(T+1)
320 X=X+28 :NEXT T
330 T=1 :FOR X=25 TO 249 STEP 28 :D=1
340 FOR X1=X+4 TO X+28 STEP 4
350 A=OZ(T+1)-OZ(T)
360 A=A/7*D+OZ(T)
370 B=KT(T+1)-KT(T)
380 B=B/7*D+KT(T)
390 IF A>B THEN LINE [5]X1,140-A,X1,140-
B
400 IF A<B THEN LINE [10]X1,140-A,X1,140
-B
410 D=D+1 :NEXT X1 :T=T+1 :NEXT X
420 GET A$ :IF A$="" THEN 420
430 CLS :END
```

Naast zakelijke toepassingen zijn er natuurlijk tal van andere toepassingen mogelijk met grafieken, zoals educatieve toepassingen, toepassingen voor huishoudelijk gebruik en ga zo maar door.
Aan de hand van de programma's die hier gegeven zijn, hopen we dat U zelf een programma kunt ontwikkelen dat U zelf kunt gebruiken.
Wanneer U er helemaal niet meer uit kunt komen, kunnen wij voor U altijd nog een programma schrijven, maar dan moet U het programma wel echt nodig hebben en natuurlijk vragen wij ook een kleine vergoeding.
We kunnen ook wel echte zakelijke programma's in dit boek gaan zetten, maar dan staat het boek zo vol.
Tot slot volgt nog een klein programmaatje voor de liefhebbers die er nog geen genoeg van hebben.

Programma 5: Een stuk uit de taartdiagram lichten.

Natuurlijk kunt U niet veel uit dit programma halen, maar het helpt U in ieder geval al een stukje op weg en samen met programma 3 kunt U een heel leuk eigen programma met daarin een taartdiagram schrijven en kunt U ook willekeurig een stuk uit de taart lichten.

```
10 INIT"CRT:M1"
20 CIRCLE 90,100,90,,,1.5*π
30 CIRCLE 100,109,90,,1.5*π
40 LINE 0,100,90,100,90,10
50 LINE 90,190,90,100,180,100
60 LINE 100,199,100,109,190,109
70 PAINT [1]40,40,3 :PAINT [2]30,130,3 :
PAINT [3]130,130,3
80 CURSOR 25,2 :PRINT [1]CHR$(200); :PRI
NT" = SOORT 1"
90 CURSOR 25,4 :PRINT [2]CHR$(200); :PRI
NT" = SOORT 2"
100 CURSOR 25,6 :PRINT [3]CHR$(200); :PR
INT" = SOORT 3"
110 CURSOR 25,8 :PRINT "ZWART = SOORT 4"
120 GET A$ :IF A$="" THEN 120
130 CLS :END
```

6.5: Animaties.

Wat wordt er precies onder een animatie verstaan?
Een animatie is een weergave van de werkelijkheid op de computer en die werkelijkheid beweegt ook nog, anders zou het een model zijn.
Om een goede animatie te maken heeft U zeker machinetaal nodig en ook een veel groter geheugen dan de SHARP heeft. De eenvoudigste animaties kunnen wel op de SHARP, zoals bijvoorbeeld een draaiende bol, dat gaat als volgt:
Allereerst wordt de beginstand van de bol getekend en wordt dat gedeelte ergens in het geheugen opgeslagen. Zo worden bijvoorbeeld vijf verschillende draaistanden in het geheugen opgeslagen, maar dat is afhankelijk van de grootte van het geheugen. Wanneer U vindt dat de draaiïng geleidelijk genoeg zal verlopen, schrijft U een machinetaalprogrammaatje om de verschillende standen van de bol snel achter elkaar te laten zien zodat het net lijkt of de bol draait. Wanneer U de draaiïngen goed verdeelt heeft, komt de bol bij de zesde draaiïng weer in de beginstand en kan de hele cyclus herhaald worden tot U er genoeg van krijgt.
Helaas kunnen wij U deze methode van animatie niet tonen, omdat daar nogal behoorlijk veel werk voor nodig is. In dit hoofdstuk zullen we ons beperken tot nog eenvoudigere animaties die te maken zijn met de grafische instructies die de SHARP kent. In de vorm van programmaatjes zullen wij proberen duidelijk te maken wat er op het gebied van de animatie zoal mogelijk is binnen de BASIC.

Programma 1: Trein.

Een animatie van een 'rijdende' trein. Door iets met kleuren te doen, lijkt het of het rails onder de trein door gaat en de trein rijdt.

```
10 INIT"CRT:M1" :PAL 2,7 :PAL 3,7 :PAL 1
,0
20 FOR C=1 TO 39 :READ A,B,A1,B1 :LINE A
,B,A1,B1 :NEXT C
30 FOR C=1 TO 14 :READ A,B,A1,B1 :BOX A,
B,A1,B1 :NEXT C
40 FOR T=8 TO 296 STEP 32 :BOX [2]T,161,
T+12,165 :BOX [1]T+16,161,T+28,165 :NEXT
50 CIRCLE 115,147,9 :CIRCLE 150,147,9
60 BOX [2]104,144,160,152 :PAINT [0]105,
145,2 :BOX 104,144,160,152
70 OUT@$F2,$F5 :OUT@$F2,$D9 :T=10 :LINE
T,158,T,159 :S=-150
80 WAIT 50 :PAL 2,0 :PAL 1,7 :GOSUB 190
90 WAIT 50 :PAL 2,7 :PAL 1,0 :GOSUB 190
100 GOTO 80
110 DATA 73,32,319,32,64,36,80,32,40,80,
64,36,40,80,40,122,64,152,40,122,64,152,
96,152,96,152,102,136,102,136,319,136
120 DATA 168,136,182,152,182,152,319,152
,0,157,319,157,0,160,319,160,314,64,319,
64,314,88,319,88,314,64,314,88,314,72,31
9,72
```

```
130 DATA 71,96,71,136,96,96,96,136,53,56
,70,60,70,60,64,86,64,86,40,80,59,48,319
,48,199,48,211,64
140 DATA 230,88,264,136,238,48,256,72,26
8,88,304,136,255,48,266,64,285,88,320,13
6,295,48,313,72
150 DATA 152,104,160,104,160,104,165,107
,165,107,171,107,171,107,167,111,171,107
,168,104,167,111,160,111,160,111,157,108
,157,108,149,108,149,108,152,104,149,108
,152,111
160 DATA 144,64,184,88,200,64,240,88,256
,64,296,88,144,64,164,72,164,64,184,72,2
00,64,220,72,220,64,240,72
170 DATA 256,64,276,72,276,64,296,72,104
,64,136,136,107,68,118,96,123,68,134,96
180 DATA 80,72,88,96,48,102,53,112
190 LINE [0]T,158,T,159 :LINE [0]T+1,158
,T+1,159 :T=T+16 :IF T>319 THEN T=0
200 LINE T,158,T,159 :LINE T+1,158,T+1,1
59
210 LINE [0]S,158,S,159 :LINE [0]S+1,158
,S+1,159 :S=S+16 :IF S>319 THEN S=10
220 LINE S,158,S,159 :LINE S+1,158,S+1,1
59
230 RETURN
```

Programma 2: Rollend wiel.

Dit programma werkt met een stukje machinetaal om het wiel voort te bewegen. Dit had in het vorige programma ook toegepast kunnen worden, alleen beweegt een trein wel iets sneller dan een gewoon wiel en met machinetaal beweegt een trein niet bepaald snel en een draaiend wiel kan nog net in machinetaal.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0 :PAL 2
,0 :XV=1
20 POKE $FD00,$DB,$E0,$3E,$83,$D3,$CC,$D
3,$CD,$21,$20,$9C,$E,$14,$37,$3F,$6,$4,$
CB,$16,$23,$10,$FB,$11,$24,$0,$19,$D,$20
,$F0,$DB,$E1,$C9
30 CIRCLE 10,190,9
40 FOR S=1 TO 14
50 FOR T=0 TO π STEP π/20
60 X=COS(T)*9 :Y=SIN(T)*9
70 X1=COS(T+.5*π)*9 :Y1=SIN(T+.5*π)*9
80 USR($FD00)
90 LINE [2,1]10-X+XV,190-Y,10+X+XV,190+Y
100 LINE [2,1]10-X1+XV,190-Y1,10+X1+XV,1
90+Y1
110 IF XV>0 AND XV/8=INT(XV/8) THEN POKE
 $FD09,PEEK($FD09)+1
120 XV=XV+1
130 NEXT T,S
```

Programma 3: Lopende mannetjes.

Door verschillende bewegingen van een poppetje in verschillende kleuren naast elkaar te zetten en door daarna elke keer een andere kleur zichtbaar te maken, kan een aantal lopende poppetjes achter elkaar gesimuleerd worden.
Dit programma maakt geen gebruik van de extra ic's.

```
1 INIT"CRT:M1"
2 C=1 :FOR A=1 TO 24 :CIRCLE [C]A*12;160
,5 :C=C+1 :IF C=4 THEN C=1
3 NEXT A
4 C=1 :FOR B=1 TO 23 STEP 2 :LINE [C]B*1
2,165,B*12-3,180,B*12+3,190,B*12-6,197 :
LINE [C]B*12-3,180,B*12,190,B*12-2,199
5 LINE [C]B*12+1,178,B*12-4,175,B*12,165
,B*12+1,174,B*12+5,177 :C=C+2 :IF C>3 TH
EN C=C-3
6 NEXT B
7 C=2 :FOR B=2 TO 24 STEP 2 :LINE [C]B*1
2,165,B*12-5,192,B*12-15,189 :LINE [C]B*
12-3,180,B*12+10,186,B*12+7,199
8 LINE [C]B*12-5,176,B*12-9,172,B*12-1,1
67,B*12+4,175,B*12+11,172 :C=C+2 :IF C>3
 THEN C=C-3
9 NEXT B
10 WAIT 1000
11 PAL 1,0 :PAL 2,0 :PAL 3,0
12 PAL 1,0 :PAL 2,0 :PAL 3,15 :WAIT 100
13 PAL 1,15 :PAL 2,0 :PAL 3,0 :WAIT 100
14 PAL 1,0 :PAL 2,15 :PAL 3,0 :WAIT 100
15 GET A$ :IF A$="" THEN 12
16 PAL 1,1 :PAL 2,2 :PAL 3,15 :END
```

Programma 4: Stuiterende bal.

Door een sinusbeweging op en neer met een steeds kleiner wordende amplitude en door een gewone beweging opzij, is de beweging van een stuiterende bal na te bootsen.

```
1 INIT"CRT:M1" :M=319 :N=199 :H=200 :W=π
/40 :D=90*π/180 :K=.01 :LINE 0,N,M,N
2 FOR O=0 TO M-10 STEP 4 :P=H*SIN(W*O+D)
*EXP(-K*O) :P=N-ABS(P)-3
3 CIRCLE [0]X,Y-1,2 :CIRCLE O+10,P-1,2 :
X=O+10 :Y=P :IF Y>195 THEN NOISE"O1T7A1"
4 NEXT O
```

Zo zijn er nog veel meer bewegingen die je kunt simuleren, zoals een rijdende auto, een vliegtuigbeweging en ga zo maar door. Het probleem is echter dat de bewegingen ver buiten de werkelijkheid staan. De enige oplossing die er dan bestaat is een programma dat volledig in machinetaal is geschreven en zeker het hele geheugen in beslag neemt.

HOOFDSTUK 7: Andere BASICS.

Naast de standaard BASIC-800 (5Z009 en 1Z016) zijn er nog veel, heel veel andere BASICS, waarvan de bekendste de standaard BASIC-700, UNI-GG-BASIC en de FLOPPY-BASIC zijn.
Binnenkort zal ook Neptunes Productions een eigen BASIC uitgeven, de zogenaamde GAME-BASIC. Zoals al uit de naam blijkt, zal deze BASIC hoofdzakelijk bedoeld zijn voor spelen. Enkele bijzonderheden over de GAME-BASIC:
-De GAME-BASIC zal meer geheugen vrij hebben, omdat heel veel instructies die niet in spelletjes gebruikt worden weggelaten zijn.
-De GAME-BASIC zal een aantal heel leuke nieuwe commando's bevatten, zoals het SCROLL-commando, waarschijnlijk een SPRITE-commando en ga zo maar door.
-De GAME-BASIC kent een automatische start.
U ziet dat het eigenlijk een geheel nieuwe BASIC zal worden waarin weinig terug is te vinden van de oude BASIC.
GAME-BASIC wordt een aanvulling op de hele reeks BASICS die tot nu toe al gemaakt zijn. Aan GAME-BASIC is echter wel veel meer tijd en denkwerk besteed dan aan veel andere BASICS. Sommige BASICS bevatten af en toe alleen een paar aanvullingen, waardoor het geheugen alleen maar kleiner wordt.
Er zijn ook BASICS, zoals de UNI-GG-BASIC, die wel leuke nieuwe zaken bevatten. De UNI-GG-BASIC biedt de mogelijkheid om een deel van de VIDEO-RAM als gewone RAM te gebruiken. Het is dan wel noodzakelijk dat U de extra ic's in Uw computer heeft. UNI-GG-BASIC zal verderop in dit hoofdstuk uitvoerig besproken worden.
In dit stukje willen we graag ook even onze aandacht richten op de FLOPPY-BASICS.
Er zijn een aantal aardige FLOPPY-BASICS, maar een echte goede is er nog niet. Het probleem is namelijk dat er voor het aansturen van de floppy-drives extra geheugen nodig is. De BASIC moet dus helemaal omgegooid worden om minimaal evenveel geheugen over te houden en ook de andere devices nog aan te kunnen sturen.
Een mogelijkheid is het gedeelte dat bestemd is voor de plotter uit de BASIC te gooien en daar het FLOPPY-gedeelte neer te zetten. Een andere mogelijkheid is het opvullen van lege gedeeltes in het geheugen en die mogelijkheden worden vaak niet benut, een onnodig geheugenverlies is het gevolg.
Een leuk verschijnsel ontstond toen de SHARP MZ-gg een programmeerwedstrijd uitgeschreven had. Ze hadden namelijk niet gezegd in welke BASIC de programma's geschreven moesten zijn en kregen zodoende programma's, geschreven in allemaal verschillende BASICS, binnen. Daarna hebben ze ook nog de fout gemaakt om alle programma's onverandert op 1 cassette te zitten. Zo kreeg je een wirwar van BASICS door elkaar en niemand wist meer in welke BASIC een programma was geschreven.
Zulke verwarringen zullen wel vaker voor komen wanneer er zoveel BASICS blijven, daarom heeft o.a. de SHARP MZ-gg geprobeerd een standaard BASIC door te voeren. Aangezien deze BASIC ook zijn nadelen kent, zullen er toch weer nieuwe BASICS bij blijven komen. Een goed voorbeeld daarvan is de GAME-BASIC.
Naast de GAME-BASIC zullen we altijd ook nog de standaard BASIC aan blijven houden, want het blijkt toch altijd weer dat dit een uitstekende BASIC is en voor heel veel zaken uiterst geschikt.

7.1: NEPTUNES-BASIC.

Wij zitten nu wel de hele tijd te praten over allerlei BASICS, zoals de GAME-BASIC, de standaard 700- en 800-BASIC, de standaard FLOPPY-BASIC en UNI-GG-BASIC, maar wist U dat NEPTUNES SOFTWARE al eerder een BASIC voor de MZ-700 geschreven heeft?
In de tijd dat alleen MARK en JEROEN de ROVER zich met NEPTUNES SOFTWARE bezig hielden, hebben ze onder de naam NEPTUNES BASIC een BASIC voor de MZ-700 geschreven. Dit was echter gewoon een aanvulling op de normale MZ-700 BASIC en niet een echte nieuwe BASIC, toch zaten er leuke aardigheden in. Een aantal mensen hebben deze BASIC zeker nog thuis en omdat het ook door NEPTUNES SOFTWARE geschreven is, willen wij er niet helemaal aan voorbij gaan en hieronder nog een aantal POKES voor de NEPTUNES BASIC publiceren.
De opzet van de POKES is als volgt:
In het rijtje onder ADRES komt het adres te staan dat U kunt poken. In het rijtje onder PEEK komt te staan wat de originele waarde van dat adres is en in het rijtje onder POKE komt te staan welke waarden U moet/kunt poken. In het rijtje onder uitleg komt de uitleg van de poke, cq het adres te staan.

ADRES:	PEEK:	POKE:	UITLEG:
53248	?	?	Eerste beeldschermadres.
55296	?	?	Eerste kleurenbeeldschermadres.
2617	0	0-255	Soundgenerator I (hoog).
2618	0	0-255	Soundgenerator II (laag).
89	83	240	Repeated GET.
6452	113,32	54,25	SHIFT+BREAK-toets.
1203	216	201	BREAK-toets.
12629	2,65	254,32	LIST > Syntax error.
11029	76	0	LIST > Syntax error.
16642	175	201	LIST > Ready.
12721	157,47	254,32	SAVE > Syntax error.
17060	205	201	SAVE > Ready.
11194	83	0	SAVE > Syntax error.
11045	65	0	AUTO > Syntax error.
8661	17	201	AUTO > Ready.
8603	82	0	Ready > " ".
11105	78	0	NEW > Syntax error.
11034	68	0	DELETE > Syntax error.
11032	212	0	Alle commando's omgeschakeld.
3901	1	0	De letter 'A' > CHR$(0).
3926	26	0	De letter 'Z' > CHR$(0).
93	113	241	Print hierna 1 keer in 2e ch.set.
91	0	0-39	Beginpositie van CONSOLE (x-as).
92	39	0-39	Eindpositie van CONSOLE (x-as).
94	0	0-255	Teken van DELETE-toets.
11108	80	0	POKE > Syntax error.
11138	75	0	KEY > Syntax error.
10997	71	0	GOTO > Syntax error.
96	239	0-255	Alpha-teken.
2041	239	0-255	Alpha-teken permanent veranderen.
10003	63	0-255	INPUT-teken.

ADRES:	PEEK:	POKE:	UITLEG:
6452	113,32	54,25	\
8987	105,32	29,35	> Bij INPUT is (SHIFT+) BREAK niet meer
9056	105,32	98,32	> mogelijk.
1203	216	201	/
12919	181,101	254,32	PEEK > Syntax error.
26037	254	201	PEEK > Ready.
12410	0,0	205,249	\
12412	0,0	23,205	> Tussen de regels van de listing komt
12414	0,0	249,23	> een vrije regel.
12416	0	201	/
16781	249,23	122,48	LIST met of zonder vrije regel.
292	4	1	CR > pieptoon.
399	153	150	Pieptoon bij alle andere toetsen.
100	0	1	AM (0) of PM (1).
84	?	0-39	Bevat cursorpositie x-as.
85	?	0-24	Bevat cursorpositie y-as.
648	16	0-255	Snelheid van de cursor. 1=hoge snelheid. 255=lage snelheid.
4092	0-x	-	Bevat de laatst ingeladen filemode.
4110	x,x	-,-	PRINT PEEK(4110)+PEEK(4111)*256 geeft de lengte van een programma.
4112	x,x	-,-	PRINT PEEK(4112)+PEEK(4113)*256 geeft het opslagadres van een programma.
4093 t/m 4109	?	?	Hier staat de naam van het laatst ingeladen programma.
57345	255	?	Op dit adres zijn de volgende mogelijkheden: 254 - SHIFT. 191 - CTRL. 190 - SHIFT+CTRL.
21022	1	<>1	Schakelt automatische RUN uit.
1	218,0	110,28	Automatische RUN na reset.
77	0	1	Alles is of op het scherm of op de printer/plotter te zien.
3050	96	48	\
3063	95	48	\
3081	46	23	> SAVE in 2400 BAUD.
3091	43	22	/
3397	68	34	LOAD in 2400 BAUD.
5058	229	201	BYE > Ready.
9827	44	0	\
9789	34	0	> Bij INPUT een komma mogelijk.
4114	x,x	-,-	PRINT PEEK(4114)+PEEK(4115)*256 geeft het startadres van een programma.
7303	205,78,34	0,0,0	Bij RUN geen CLR.
79	x	255	Lengte BASIC-regel (max. 255)

Hopelijk begrijpt U waarvoor U de pokes of peeks kunt gebruiken. Ze zijn niet allemaal even nuttig, maar er zitten wel een aantal leuke tussen. Tevens kan het zo zijn dat een aantal pokes ook in de gewone 700-BASIC werken.

7.2: BASIC-700.

Er bestaan twee verschillende versies van de standaard 700-BASIC, één voor QD en één voor cassette. Tussen deze twee BASICS zit nog behoorlijk veel verschil, daarom zal er van één van deze twee BASICS uitgegaan worden in dit hoofdstuk en dat is de standaard 700-BASIC voor de cassette, dat is de BASIC 1Z013 (S-BASIC).

De 700-BASIC heeft voordelen, maar ook nadelen vergeleken met de 800-BASIC. Het grootste voordeel is wel dat er veel meer geheugenruimte is te gebruiken. De grootste nadelen zijn wel dat de hoogste schermoplossing slechts 25*40 is en dat er geen QD-instructies gebruikt kunnen worden.
Er zijn een aantal dingen die anders gaan dan in de 800-BASIC, zoals het opbouwen van een scherm of het maken van geluidjes. In dit hoofdstuk willen wij een aantal dingen laten zien die anders gaan dan in de 800-BASIC, dat zal gedaan worden aan de hand van een aantal kleine programmaatjes die kriskras door elkaar staan. Tevens zullen er enkele POKES en andere interessante dingen voor de 700-BASIC gegeven worden.

Voorbeeld 1: Het programma STERREN IN 700-BASIC.

In hoofdstuk 10 staat een spel met de naam STERREN. Hier volgt de 700-BASIC versie van dat programma, maar nu onder de naam STARS.

```
10 COLOR,,,0
20 CONSOLE 0,25,0,40 :CLS
30 PRINT CHR$(21);" STARS  SCORE:
  1990 NEPTUNES-SW"
40 FOR A=6 TO 39 :POKE $D800+A,$50 :NEXT
 A
50 FOR A=0 TO 5 :POKE $D800+A,$D0 :NEXT
A
60 POKE $D000+22,136 :POKE $D800+22,$D0
70 CONSOLE 2,22,1,38
80 B=2 :FOR A=1 TO 38 :SET B,3 :SET B+1,
3 :SET B,2 :SET B+1,2 :COLOR A,1,3 :SET
B,48 :SET B+1,48 :COLOR A,24,3 :B=B+2 :N
EXT A
90 B=2 :FOR A=1 TO 23 :SET 1,B :SET 1,B+
1 :COLOR 0,A,3 :SET 78,B :SET 78,B+1 :CO
LOR 39,A,3 :B=B+2 :NEXT A :SET 1,48 :SET
 78,48 :COLOR 0,24,3 :COLOR 39,24,3
100 A=20
110 TI$="000000"
120 CURSOR INT(RND(1)*38)+1,23 :PRINT"*"
130 IF PEEK($D000+80+A)<>0 THEN GOTO 200
140 CURSOR A,2 :PRINT CHR$(252)
150 CURSOR 14,0 :PRINT TI$
160 GET A$ :IF (A$<>"")*(ASC(A$)>18)*(AS
C(A$)<21) THEN D$=A$
170 IF (D$=CHR$(19))*(A<38) THEN A=A+1
180 IF (D$=CHR$(20))*(A>1) THEN A=A-1
```

```
190 GOTO 120
200 COLOR,,,1
210 CURSOR 9,12 :PRINT"NOG EEN KEER? (J/
N)"
220 GET A$ :IF A$="N" THEN GOTO 250
230 IF A$<>"J" THEN 220
240 RUN
250 CLS :PRINT"TOT ZIENS" :NEW
```

Om na dit programma het volledige scherm weer te kunnen gebruiken, moet U gewoon CONSOLE intypen.

<u>Voorbeeld 2: Nog een programma.</u>

Het volgende spel is het spel WORM. U moet het hele veld leeg eten en daarbij wordt U steeds langer. U moet uitkijken dat U Uzelf niet raakt. Besturing met de pijltjestoetsen.

```
10 COLOR,,6,0
20 CLS :INPUT"SNELHEID (1-100)";U :IF (U
<1)+(U>100) THEN GOTO 20
30 DIM G(1900) :DIM H(1900) :X=2 :L=2
40 CLS :CURSOR 1,0 :PRINT"WORM" :CURSOR
8,0 :PRINT"SCORE:";Y;"   (C)1990 NEPTUNE
S-SW"
50 FOR A=1 TO 5 :POKE $D800+A,$C0 :NEXT
A
60 CURSOR 0,1
70 PRINT"██████████████████████████████
███████"
80 PRINT"█··········███·········███····
······█"
90 PRINT"█·████·███·····███████·····███
·████·█"
100 PRINT"█······███·███·········███·██
█······█"
110 PRINT"█·████·███·███·███████·███·██
█·████·█"
120 PRINT"█····························
·······█"
130 PRINT"█·███·████████·█·███·█·██████
██·███·█"
140 PRINT"█·█···█········█·····█·······
·█···█·█"
150 PRINT"█·█·█·█·████████·███·████████
·█·█·█·█"
160 PRINT"█·█·█···█···················█
···█·█·█"
170 PRINT"█·█···█·█·███████·█·███████·█
·█···█·█"
180 PRINT"█···█·············█··········
···█···█"
190 PRINT"█·█···█·█·███████·█·███████·█
·█···█·█"
```

```
200 PRINT" ▌·▌·▌···▌·····················▌
···▌·▌·▌"
210 PRINT" ▌·▌·▌·▌·▌▌▌▌▌▌·▌▌·▌▌▌▌▌▌
·▌·▌·▌·▌"
220 PRINT" ▌·▌···▌·····▌·····▌·····
·▌···▌·▌"
230 PRINT" ▌·▌▌·▌▌▌▌▌·▌·▌▌·▌·▌▌▌▌
▌▌·▌▌▌·▌"
240 PRINT" ▌···························
·····▌"
250 PRINT" ▌·▌▌▌·▌▌·▌▌·▌▌▌▌▌·▌▌·▌▌
▌·▌▌▌·▌"
260 PRINT" ▌·····▌▌·▌▌·····▌▌·▌▌
▌·····▌"
270 PRINT" ▌·▌▌▌·▌▌·····▌▌▌▌▌·····▌▌
▌·▌▌▌·▌"
280 PRINT" ▌·····▌▌·····▌▌···
·····▌"
290 PRINT" ▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌
▌▌▌▌▌▌▌"
300 CURSOR 2,2 :PRINT CHR$(198)
310 GET A$ :IF (ASC(A$)<17)+(ASC(A$)>20)
 THEN GOTO 310
320 D$=A$ :A=2 :B=2 :C=A :D=B :E=A :F=B
:G(X)=A :H(X)=B
330 IF (ASC(A$)>16)*(ASC(A$)<21) THEN GO
SUB 450
340 CURSOR A,B :PRINT CHR$(Z) :CURSOR E,
F :PRINT" " :CURSOR G(X),H(X) :PRINT CHR
$(145)
350 E=G(INT(L)) :F=H(INT(L))
360 C=A :D=B
370 GET A$ :IF (A$<>"")*(ASC(A$)>16)*(AS
C(A$)<21) THEN D$=A$
380 A$=D$
390 IF ASC(D$)=18 THEN B=B-1 :Z=94
400 IF ASC(D$)=19 THEN A=A+1 :Z=198
410 IF ASC(D$)=17 THEN B=B+1 :Z=252
420 IF ASC(D$)=20 THEN A=A-1 :Z=95
430 FOR QW=0 TO U*1 :NEXT
440 GOTO 330
450 IF PEEK($D000+A+40*B)=67 THEN A=C :B
=D :S=0 :RETURN
460 IF PEEK($D000+A+40*B)=163 THEN FOR Q
W=1 TO 2000 :NEXT :GOTO 510
470 IF PEEK($D000+A+40*B)=46 THEN Y=Y+1
:CURSOR 14,0 :PRINT Y
480 IF Y=429 THEN FOR QW=1 TO 2000 :NEXT
 :GOTO 560
490 X=X+1 :G(INT(X))=C :H(INT(X))=D :L=L
+.9
500 RETURN
510 CLS :PRINT"                    HELAAS"
520 PRINT"          UW SCORE WAS";Y;" PUN
TEN"
```

```
530 PRINT"          DRUK OP DE SPATIEBAL
K"
540 GET A$ :IF A$<>" " THEN 540
550 RUN
560 CLS
570 PRINT"            GEFELICITEERD !!!"
580 PRINT"          DRUK OP DE SPATIEBAL
K"
590 GET A$ :IF A$<>" " THEN 590
600 RUN
```

Voorbeeld 3: Het derde en laatste spel.

REACT is een klein beetje een denkspel dat steeds moeilijker wordt. Het is de bedoeling dat U alle rondjes van het veld eet, daarbij mag U niets raken. U moet ook uitkijken dat U Uzelf niet insluit, want overal waar U bent geweest, kunt U daarna niet meer komen. Bij elk veld komen er meer rondjes, maar ook meer koppen en die koppen mag U ook absoluut niet raken.
Besturing met de pijltjestoetsen.

```
10 COLOR,,7,0 :CLS :INPUT"SNELHEID (1-10
0)";R :IF (R<1)+(R>100) THEN 10
20 CLS :L=3 :X=5 :Y=2 :SC=0
30 CLS :FOR A=0 TO 38 :CURSOR A,1 :PRINT
"■" :CURSOR A,23 :PRINT"■" :NEXT A
40 FOR A=1 TO 23 :CURSOR 0,A :PRINT"■" :
CURSOR 38,A :PRINT"■" :NEXT A :FOR B=1 T
O 12 :CURSOR B,12 :PRINT"■" :CURSOR B+25
,12 :PRINT"■" :NEXT B
50 FOR B=2 TO 8 :CURSOR 19,B :PRINT"■" :
CURSOR 19,B+14 :PRINT"■" :NEXT B
60 A=198 :B=2 :C=1
70 CURSOR 0,0 :PRINT"REACT" :CURSOR 7,0
:PRINT"SCORE:";SC :CURSOR 20,0 :PRINT"NE
P-SOFT  LEVENS:";L
80 POKE $D000+19,136 :POKE $D800+19,$F0
90 FOR Z=0 TO 4 :POKE $D800+Z,$D0 :NEXT
100 FOR D=1 TO X
110 W=INT(RND(1)*37+1) :Z=INT(RND(1)*22+
2)
120 IF PEEK($D000+W+Z*40)<>0 THEN GOTO 1
10
130 IF W=1 THEN GOTO 150
140 GOTO 160
150 IF Z=2 THEN GOTO 110
160 CURSOR W,Z :PRINT CHR$(104) :NEXT D
170 FOR D=1 TO Y
180 W=INT(RND(1)*37+1) :Z=INT(RND(1)*22+
2)
190 IF PEEK($D000+W+Z*40)<>0 THEN GOTO 1
80
200 IF W=1 THEN GOTO 220
210 GOTO 230
```

```
220 IF Z=2 THEN GOTO 180
230 CURSOR W,Z :PRINT"O" :NEXT D
240 V=B :Z=C
250 IF S>0 THEN CURSOR Z,V :PRINT CHR$(1
49) :Z=C :V=B
260 CURSOR C,B :PRINT CHR$(A)
270 GET A$ :S=PEEK(95)
280 IF S=18 THEN B=B-1 :A=94 :GOTO 330
290 IF S=19 THEN C=C+1 :A=198 :GOTO 330
300 IF S=17 THEN B=B+1 :A=252 :GOTO 330
310 IF S=20 THEN C=C-1 :A=95 :GOTO 330
320 S=0 :GOTO 390
330 W=PEEK($D000+C+B*40) :IF (W=67)+(W=2
07)+(W=166) THEN L=L-1 :CURSOR 37,0 :PRI
NT L :FOR QW=1 TO 1000 :NEXT :WE=150 :IF
 L=0 THEN GOTO 400
340 IF WE=150 THEN WE=0 :G=0 :GOTO 30
350 IF PEEK($D000+C+B*40)=15 THEN G=G+1
:SC=SC+1 :CURSOR 13,0 :PRINT SC
360 IF SC=0 THEN GOTO 380
370 IF SC/25=INT(SC/25) THEN L=L+1 :CURS
OR 37,0 :PRINT L :FOR QW=1 TO 250 :NEXT
:SC=SC+1 :CURSOR 13,0 :PRINT SC
380 IF G=Y THEN FOR QW=1 TO 500 :NEXT :G
=0 :X=X+5 :Y=Y+2 :GOTO 30
390 FOR QW=1 TO R*1.5 :NEXT :GOTO 250
400 FOR QW=1 TO 1000 :NEXT :CLS :CURSOR
0,11
410 PRINT"                    HELAAS"
420 PRINT"              UW SCORE WAS:";SC
430 GET A$ :IF A$="" THEN GOTO 430
440 RUN
```

Voorbeeld 4: Schaakbord.

Dit schaakbord kan eventueel ook in de 800-BASIC gemaakt worden, maar in 700-BASIC gaat het veel gemakkelijker.

```
10 COLOR,,2,2
20 CLS
30 DATA 42,213,43,214,82,215,83,216,46,1
93,47,194,86,195,87,196,50,201,51,202,90
,203,91,204,54,197,55,198,94,199,95,200,
58,205,59,206,98,207,99,208
40 DATA 62,201,63,202,102,203,103,204,66
,193,67,194,106,195,107,196,70,213,71,21
4,110,215,111,216
50 FOR A=1 TO 32 :READ C,D :POKE $D027+C
,D :POKE $D36F+C,D :NEXT A
60 FOR A=2 TO 30 STEP 4 :POKE $D0C7+A,20
9 :POKE $D0C8+A,210 :POKE $D0EF+A,211 :P
OKE $D0F0+A,212
70 POKE $D31F+A,209 :POKE $D320+A,210 :P
OKE $D347+A,211 :POKE $D348+A,212 :NEXT
```

```
80 R=0 :FOR U=1 TO 23 STEP 3 :R=R+1 :FOR
 T=0 TO 2 :FOR K=0 TO 30 STEP 4 :FOR L=0
 TO 3
90 IF U>12 THEN 130
100 IF INT(R/2)<>R/2 THEN POKE $D800+L+K
+40*U+40*T,$FC
110 IF INT(R/2)=R/2 THEN POKE $D800+L+K+
40*U+40*T,$F1
120 GOTO 150
130 IF INT(R/2)<>R/2 THEN POKE $D800+L+K
+40*U+40*T,$8C
140 IF INT(R/2)=R/2 THEN POKE $D800+L+K+
40*U+40*T,$81
150 NEXT L :R=R+1 :NEXT K;T,U
160 COLOR,,3,0
170 COLOR,,7,0 :PRINT CHR$(21);"SCHAAKBO
RD (C)1990 NEPTUNES-SOFT          "
180 GET A$ :IF A$="" THEN 180
190 CLS :END
```

Voorbeeld 5: Belangrijke adressen in de 700-BASIC.

Nu zullen een aantal belangrijke adressen uit de 700-BASIC getoond worden. Tevens zal aangegeven worden hoe U de adressen aan moet sturen, met POKE, USR of iets dergelijks, om het gewenste resultaat te bereiken.

`A$="NEPTUNES" :USR($0015,A$)` - Deze USR bent U al vaker tegengekomen. Ook hier print het de tekst in A$ op het scherm.

`M$="A7#C5" :USR($0030,M$)` - Deze USR speelt het muziekje dat in M$ is opgeslagen.

`USR($003E)` - Zelfde functie als BEEP in de 800-BASIC.

`POKE $004D,1` - Alles gaat naar de printer/plotter.
`POKE $004D,0` - Alles gaat naat het beeldscherm.

`POKE 89,240` - Repeated GET.
`POKE 89,83` - Normaal.

Voorbeeld:

```
10 POKE 89,240
20 GET A$ :PRINT A$ :GOTO 20
```

`PRINT PEEK($55)` - De y-positie van de cursor.

`POKE $55,x (0 =< x =< 24)` - Bepaal zelf de y-positie van de cursor.

```
10 ' CONSOLE VIA POKES
20 CLS
30 POKE 86,10 :'Beginrij
```

```
40 POKE 87,15 :'Laatste rij
50 POKE 91,10 :'Eerste kolom
60 POKE 92,30 :'Laatste kolom
70 LIST :LIST
```

Poke voor verandering achter- en voorgrondkleur van tekst:

```
POKE 93,$27 :PRINT"NEPTUNES PRODUCTIONS"
 :POKE 93,$71
```

Peek voor uitlezen toetsenbord:

```
10 PRINT CHR$(PEEK(95))
20 GOTO 10
```

POKE 96,73 - Verander teken van de cursor in een ?
POKE 96,239 - Normaal.

POKE 292,1 - Bij drukken van <cr> is elke keer een korte toon te horen.
POKE 292,4 - Normaal.

POKE 648,x - Hiermee kan de snelheid van de cursor veranderd worden.
POKE 648,16 - Normaal.

USR(1621) - Scrollt beeldscherm één regel omhoog.

U kunt heel leuke effecten bereiken (bijvoorbeeld het verdwenen scherm terug krijgen!!!) door tegelijk op SHIFT en de pijltjestoets omhoog of omlaag te drukken, dat is hetzelfde effect als bij de beide USR's.

USR(1661) - Scrollt beeldscherm één regel omlaag.

Toon met veranderende frequentie:

```
10 FOR A=1 TO 20
20 FOR B=1 TO 255
30 POKE 2617,B :POKE 2618,A
40 USR(68) :'Toongenerator AAN.
50 NEXT B,A
60 USR(71) :'Toongenerator UIT.
```

Zowel de voorgrond- als de achtergrondkleur kunnen over het hele scherm verandert worden zonder dat het beeldscherm schoongemaakt hoeft te worden, dat gaat als volgt:

```
COLOR,,6,1 :USR($72D)
```

Machinetaal in combinatie met BASIC.

Nu volgt een programma om het scherm naar boven te scrollen. Alles wat boven verdwijnt, komt er beneden weer bij.
U zult zien dat het scrollen via machinetaal in 700-BASIC via twee CALLS moet gaan, anders wil het niet. Dit maakt dat het scrollen in 700-BASIC iets lastiger is.

```
10 CLS :LIST :LIST :LIST
20 POKE $C000,$21,$0,$D0,$11,$D8,$CF,$E,
$19,$6,$28,$CD,$EA,$0,$EB,$CD,$F2,$0,$EB
,$23,$13,$10,$F4,$D,$20,$EF,$21,$D8,$CF,
$6,$28,$CD,$EA,$0,$EB,$CD,$F2,$0,$EB,$23
,$13,$10,$F4,$C9
30 FOR A=1 TO 50 :USR($C000) :NEXT A
```

Op een soortgelijke manier kunt U natuurlijk ook weer naar beneden, links en rechts scrollen. Aangezien dat al behandeld is, zal het hier niet nog een keer extra getoond worden.

In BASIC-700 kan via de adressen 2617 en 2618 een toon bepaald worden, die dan door USR(68) ten gehore is te brengen. Hieronder staan een aantal voorbeelden van verschillende tonen. Aan de hand van de voorbeelden die U hier te zien krijgt, kunt U zelf concluderen dat er op deze manier veel mooiere geluidjes uit de SHARP te halen zijn. Een klein beetje doorrekenen en U kunt het ook in de 800-BASIC doen.

Voorbeeld 1:

```
FOR A=30 TO 1 STEP -.6 :FOR B=0 TO 20 ST
EP .8 :POKE 2618,A+B :USR(68) :NEXT B,A
:USR(71)
```

Voorbeeld 2:

```
FOR A=20 TO 150 :POKE 2618,A :USR(68) :F
OR B=1 TO 10 :NEXT B :USR(69) :NEXT A
```

Voorbeeld 3:

```
FOR A=220 TO 20 STEP -10 :FOR B=A+20 TO
A-10 STEP -1 :POKE 2618,B :USR(68) :NEXT
 B,A :USR(69)
```

Voorbeeld 4:

```
FOR W=1 TO 5 :FOR T=19 TO 10 STEP -.1 :F
OR A=2 TO 1 STEP -.2 :POKE 2618,T*W*A :U
SR(68) :NEXT A,T,W :USR(69)
```

Voorbeeld 5:

```
A=10 :FOR B=30 TO A STEP -1 :FOR C=1 TO
B :POKE 2618,C :USR(68) :NEXT C,B :USR(6
9)
```

Voorbeeld 6:

```
FOR B=30 TO 15 STEP -1 :FOR C=1 TO B :PO
KE 2618,ABS((C-10)*(B-14)-99) :USR(68) :
NEXT C,B :USR(69)
```

Voorbeeld 7:

```
FOR A=10 TO 1 STEP -1 :POKE 2618,A :FOR
B=0 TO 255 STEP A :POKE 2617,B :USR(68)
:NEXT B,A :USR(69)
```

Op een dergelijke manier kunt U zelf natuurlijk nog genoeg leuke geluidjes bedenken.

Nog enkele POKES.

Toon bij Ready.

```
POKE 6256,129,48 :POKE 12417,175,205,62,
0,205,6,0,201
```

Toon bij error.

```
POKE 8561,137,48 :POKE 12425,205,81,0,22
9,17,149,48,205,48,0,225,201,45,65,13
```

LIST stoppen met spatiebalk.

Normaal is de list alleen tijdelijk te stoppen door de spatiebalk ingedrukt te houden. Met deze pokes kunt U de spatiebalk daarna weer los laten.

```
POKE 16786,77,36,204,83,2,0 :POKE 9293,2
05,27,0,254,32,201
```

LIST met vrije tussenregel.

```
POKE 12410,205,249,23,205,249,23,201
aan: POKE 16781,122,48
uit: POKE 16781,249,23
```

Enkele nutteloze pokes:

```
POKE 16642,201 - LIST uitgeschakeld.
POKE 16642,175 - LIST weer mogelijk.

POKE 17060,201 - SAVE uitgeschakeld.
POKE 17060,205 - SAVE weer mogelijk.

POKE 26037,201 - PEEK uitgeschakeld.
POKE 26037,254 - PEEK weer mogelijk.

POKE 5058,201  - BYE  uitgeschakeld.
POKE 5058,229  - BYE  weer mogelijk.
```

In dit korte stukje over de 700-BASIC hebben we geprobeerd een aantal leuke dingen uit de 700-BASIC te laten zien. Het had allemaal wel een rommelig karakter, maar wie werkt er nog met de 700-BASIC. De gepubliceerde voorbeelden zijn dus alleen om eens uit te proberen.

7.3: UNI-GG-BASIC.

In het rijtje van bekende BASICS komt ook UNI-GG-BASIC, de BASIC die hoofdzakelijk door de MZ-gg geschreven is. Ondanks het feit dat deze BASIC hoofdzakelijk dezelfde instructies en statements gebruikt als de standaard 800-BASIC, zijn er behoorlijke verschillen tussen de UNI-GG-BASIC en de standaard 800-BASIC.
Met het volgende programma zullen we eerst eens bekijken welke instructies en statements de UNI-GG-BASIC kent:
(Waarschuwing: Alles wat er in dit hoofdstuk staat over UNI-GG-BASIC is in versie 2.9 van de UNI-GG-BASIC gemaakt. Of het in de andere versies werkt is dus niet zeker.)

```
10 INIT"CRT:M1" :B=0
20 FOR A=$A8F0 TO $AC3F
30 IF PEEK(A)>$40 AND PEEK(A)<$5B THEN P
RINT CHR$(PEEK(A));
40 IF PEEK(A)>$BF AND PEEK(A)<$DB THEN P
RINT CHR$(PEEK(A)-$80), :B=B+1
50 IF PEEK(A)=$A4 THEN PRINT"$", :B=B+1
60 IF B=84 THEN PRINT :PRINT"DRUK OP RET
URN VOOR VERVOLG." :GOSUB 90
70 NEXT A
80 GET A$ :IF A$="" THEN 80 ELSE END
90 GET A$ :IF ASC(A$)=13 THEN B=0 :CLS :
RETURN ELSE GOTO 90
```

Aan de hand van dit programma heeft U kunnen zien dat er een aantal nieuwe instructies en statements in de BASIC zijn gekomen.
Welke instructies er precies bij gekomen zijn en wat U er mee kunt doen, kunt U zelf lezen in de handleiding van de UNI-GG-BASIC en daarvoor moet U bij de SHARP MZ-gg zijn.
Als het goed is, moet ieder lid van de SHARP MZ-gg in het bezit zijn van de UNI-GG-BASIC, want deze BASIC wordt gratis onder alle leden verspreid.
Omdat er steeds weer nieuwere versies van de UNI-GG-BASIC bij komen, krijg je ook onderling verschil tussen de BASICS. Programma's, gechreven in een nieuwere versie, draaien af en toe niet in een oudere versie. U moet dus zorgen dat U steeds de nieuwste versie in huis heeft.
Buiten de nieuwe instructies en statements om heeft de UNI-GG-BASIC nog een aantal aardige dingen in huis. Zo kunt U zowel cassette, QD als FLOPPY-DISK gebruiken. Dit heeft wel als nadeel dat U minder geheugen over houdt en de vrije geheugenruimte van de SHARP-BASIC is al zo klein, daar heeft de SHARP-MZ-gg echter een oplossing voor gevonden. In de nieuwste versies van de UNI-GG-BASIC kunt U, wanneer U de extra VIDEO-RAM ic's in Uw computer heeft, 16K aan extra geheugen gebruiken. De VIDEO-RAM wordt dan als gewoon geheugen gebruikt.
Het is zeker dat deze aanpassing niet door de SHARP MZ-gg zelf is gemaakt. Geruchten willen zelfs zeggen dat het uit het buitenland komt. Hopelijk is dit maar een gerucht, want er zit genoeg informatie voor de SHARP in ons eigen land en het zou bedroevend met de SHARP gesteld zijn als we de informatie al uit het buitenland moesten halen.
Een goed voorbeeld van informatie uit eigen land is dit boek en dit boek is dan nog informatie van hoofdzakelijk één persoon.

Eigenlijk dwaalden we een klein beetje af, maar nu gaan we dan weer terug naar de UNI-GG-BASIC.
Dat het een leuk alternatief is voor de standaard 800-BASIC, is zeker, maar dat het nu een veel betere BASIC is kan absoluut niet gezegd worden.
De bedoeling van de UNI-GG-BASIC is waarschijnlijk geweest om een BASIC te maken die in de richting komt van GW-BASIC voor PC's. U kunt dan heel gemakkelijk programma's van GW-BASIC naar de UNI-GG-BASIC omzetten.
UNI-GG-BASIC is gedurende de afgelopen jaren uitgegroeid tot een standaard-BASIC voor de MZ-gg. Alles wordt in de UNI-GG-BASIC geschreven.
Omdat de UNI-GG-BASIC alles binnen de BASIC omgegooid heeft, komt bijna geen enkel adres meer overeen met de standaard 800-BASIC. Dit heeft als nadeel dat bijna alle boeken, ook al waren het er niet veel, die het interne van de BASIC behandelen nagenoeg nutteloos zijn geworden.
Alle peeks, pokes e.d. moeten omgeschreven worden naar de UNI-GG-BASIC. De sprongtabellen voor de instructies en statements moeten omgeschreven worden naar de UNI-GG-BASIC en ga zo maar door. De ARGONAUT heeft dus nog genoeg om te publiceren.
Of de UNI-GG-BASIC werkelijk een verbetering is van de standaard 800-BASIC valt te betwijfelen. Er is in ieder geval een stap gedaan in de richting van een andere standaard. Mensen die echter niet bij de MZ-gg aangesloten zijn, gebruiken vaak toch liever de standaard 800-BASIC.
Voor de eerste cassette met software die wij uit zullen geven zal ook uitsluitend gebruik worden gemaakt van de standaard 800-BASIC.
Dit stukje over de UNI-GG-BASIC is bedoeld om deze BASIC eens van een andere kant te belichten. Op die manier komen de negatieve en natuurlijk ook de positieve punten beter naar voren. Hieronder zult U twee rijtjes aantreffen met positieve en negatieve punten:

Positief:	Negatief:
Alle devices (CMT, QD en FD) zijn te gebruiken.	Minder geheugenruimte vrij.
Wanneer U de extra ic's in Uw computer heeft, kunt U in de modes M1 en M3 16K extra aan geheugen gebruiken.	Oude en nieuwe versies niet 100% compatible.
De standaardcassettes van de MZ-gg bevatten alleen nog maar programma's geschreven in de UNI-GG-BASIC en niet in tien verschillende BASICS.	Weinig adressen komen overeen met de standaard 800-BASIC.
UNI-GG-BASIC is in zekere mate compatible met de standaard 800-BASIC.	Er zijn al weer te veel versies.
	NIET-leden van de MZ-gg kunnen in principe geen standaardcassettes van de MZ-gg kopen, omdat ze de UNI-GG-BASIC niet in huis hebben.
	Er zijn nu twee standaarden die niet in beide richtingen compatible zijn.

Dit waren zo'n beetje de belangrijkste positieve en negatieve punten van de UNI-GG-BASIC en als U die goed bekijkt, komt de UNI-GG-BASIC redelijk positief uit de bus.
Op de volgende bladzijden zullen we een aantal pokes en nog iets anders interessants voor de UNI-GG-BASIC tonen. Het is namelijk mogelijk, door op een speciale manier een aantal toetsen in te drukken, bepaalde kleuren te veranderen, daar kunt U verderop meer over lezen.

Pokes voor UNI-GG-BASIC.

Hieronder en op de volgende bladzijde vindt U een aantal pokes die ook al eerder gepubliceerd zijn in dit boek, maar die anders worden voor de UNI-GG-BASIC.

POKES voor geleidelijk omhoog scrollen.

```
10 INIT"CRT:M1
20 LIST :LIST
30 WAIT 2000
40 CURSOR 0,24
50 POKE $891,$5
60 POKE $976,$1
70 FOR T=1 TO 200 :PRINT :NEXT T
80 POKE $891,$28
90 POKE $976,$0
100 WAIT 2000 :INIT"CRT:M1"
```

POKE $109B,x (0 =< x =< 255)
Deze poke geeft de cursor een andere kleur.
Oorspronkelijk: POKE $109B,3

POKE om gewone letters dubbel zo groot te maken.

```
10 INIT"CRT:M1"
20 PRINT"NORMALE GROOTTE."
30 POKE $5DF,$50
40 PRINT
50 PRINT"DUBBELE GROOTTE."
60 POKE $5DF,$28 :CURSOR 0,6
```

Poke om tekst om te keren.

```
10 INIT"CRT:M1"
20 PRINT"DEZE TEKST ZIET ER OP Z'N KOP A
LS VOLGT UIT:"
30 PRINT :PRINT
40 POKE $5DF,$D8,$FF
50 PRINT"DEZE TEKST ZIET ER OP Z'N KOP A
LS VOLGT UIT:"
60 POKE $5DF,$28,$0
```

POKE om deel van tekst te tonen.

```
10 INIT"CRT:M1"
20 FOR G=1 TO 8
30 CURSOR 0,0 :POKE $5DD,G
40 PRINT"DIT IS EEN VOORBEELD"
50 PRINT"IN UNI-GG-BASIC."
60 WAIT 1000
70 NEXT G
```

Op de volgende bladzijde zult U nog een aantal pokes aantreffen.

POKE $5CF,3
Er wordt omgeschakeld naar de tweede characterset.
Oorspronkelijk: POKE $5CF,2

POKE $B27,x
Hiermee kunt U de snelheid van de cursor bepalen.
Oorspronkelijk: POKE $B27,$10

POKES voor CURSOR X,Y.

```
10 INIT"CRT:M1"
20 FOR B=0 TO 23
30 FOR A=0 TO 39
40 WAIT 25
50 POKE $1082,A :POKE $1083,B :PRINT"  N
EPTUNES." :NEXT A,B
```

POKE $77B9,0,0
INPUT zonder vraagteken.
Oorspronkelijk: POKE $77B9,$3F,$20

De pokes voor het veranderen van de cursor en voor het wel dan niet mogelijk maken van INIT"CRT:M2" zijn hetzelfde als in de standaard 800-BASIC.

POKES om TAB-sprong te veranderen.

```
10 INIT"CRT:M1"
20 INPUT"WAT WORDT DE NIEUWE TAB-GROOTTE
?  (1-39)";A
30 IF A<1 OR A>39 THEN 20
40 POKE $A24,A :POKE $A55,A :POKE $A29,A
 :POKE $A59,A :POKE $A40,A
50 PRINT :PRINT"DRUK NU EENS OP DE TAB-T
OETS!"
```

POKES om achtergrond bij SYMBOL te wissen.

```
10 INIT"CRT:M1"
20 A$="NEPTUNES SOFTWARE."
30 POKE $52A3,$80 :FOR A=319 TO -1130 ST
EP -8 :SYMBOL [2]A,92,A$,8,2
40 WAIT 25 :NEXT A
50 POKE $52A3,$C0
```

POKE $4E4,0,0,0
CONSOLE zonder CLS.
Oorspronkelijk: POKE $4E4,$CD,$4C,$6

Zo zijn in principe bijna alle pokes op een redelijk eenvoudige manier over te zetten van de standaard 800-BASIC naar de UNI-GG-BASIC. U kijkt eerst in de RAM-monitor van de standaard 800-BASIC hoe een POKE tot stand komt, U kijkt dus naar alles wat er omheen staat, en vervolgens gaat U met de instructie F in de RAM-monitor van de UNI-GG-BASIC hetzelfde opzoeken. Helaas is het echter niet in alle gevallen gelijk.

Een aantal leuke geintjes in UNI-GG-BASIC.

Hieronder volgen een paar leuke geintjes die U uit kunt halen in de UNI GG-BASIC dmv het toetsenbord. Alles is beproefd in versie 2.9 en werkte daarin uitstekend.

Geintje 1: De tekstkleur veranderen.

-U drukt eerst de Z-toets in en houdt deze ingedrukt.
-Daarna drukt U de SHIFT- en de BREAK-toets in en houdt deze eveneens ingedrukt.
-Daarna drukt U de INST-toets in en houdt deze ook ingedrukt.
-Als laatste drukt U de DEL-toets in.
-Na ongeveer 1 seconde laat U de DEL-toets los.
-Als de kleur eenmaal begint te veranderen, kunt U alle toetsen, behalve de Z-toets, loslaten.

Geintje 2: De borderkleur veranderen.

-U drukt de A-toets in en houdt deze ingedrukt.
-Daarna drukt U in dezelfde volgorde als bij de Z-toets de genoemde toetsen in.
-Hierna wordt het iets anders als bij de Z-toets, want nu moet U alle toetsen vasthouden.

Geintje 3: De achtergrondkleur veranderen.

-U drukt eerst de B-toets in en doet daarna precies hetzelfde als bij de A-toets.

Geintje 4: De tweede characterset.

-U drukt de I-toets in en doet daarna precies hetzelfde als bij de Z-toets. U kunt ook de I-toets loslaten.

Geintje 5: De computer vast laten lopen.

-U drukt allereerst de tweede toets naast de 0-toets in, dat is de toets met het pijltje en het golfje. Deze toets houdt U ingedrukt.
-Daarna drukt U op de BREAK-toets en houdt deze eveneens ingedrukt.
-Daarna drukt U op de INST-toets en houdt deze ook ingedrukt.
-Als laatste drukt U op de DEL-toets.
-Hierna kunt U de INST-toets of de DEL-toets loslaten en U zult zien dat de computer vastgelopen is. De enige oplossing is dan de BASIC opnieuw te laden.

Deze geintjes zijn mogelijk omdat UNI-GG-BASIC een heel andere BASIC is dan de standaard 800-BASIC en deze geintjes maken dat nog eens extra duidelijk.
Deze geintjes zijn ontdekt door op een saaie dag eens met het toetsenbord te gaan spelen, misschien zijn er dus nog veel meer. Heeft U dus ooit eens een saaie dag dan weet U wat U kunt doen, namelijk met het toetsenbord gaan spelen en misschien ontdekt U dan ook wat.

HOOFDSTUK 8: HARDWARE.

Wat wordt er precies onder hardware verstaan. In appendix B staat wel een omschrijving van het begrip hardware, maar die is veel te ruim-omvattend. De hardware waar we hier over gaan spreken zijn onderdelen van de computer en de bijbehorende randapparatuur.
In hoofdstuk 0 is al over randapparatuur gesproken, in dit hoofdstuk zal dat onder andere nog een keer gedaan worden, alleen dan op een andere manier. Er zullen technische begrippen aan te pas komen en weinig tot geen schema's, omdat die er al genoeg zijn. Zelfs in het handboek staan veel schema's. Voor de echte hardwareliefhebbers is er het Technical Reference Manual, daarin staan tal van interessante schema's en uitleg in het Engels.
De verschillende schema's zijn voor veel mensen abakadabra. In dit hoofdstuk zal geprobeerd worden de hardware in een eenvoudige, voor iedereen te begrijpen taal uit te leggen, namelijk het Nederlands.
Er zijn heel veel afzonderlijke stukken hardware die allemaal besproken kunnen worden, iedere ic kan bijvoorbeeld apart behandeld worden, maar dat wordt te omslachtig. De belangrijkste ic's komen alleen aan de orde evenals de belangrijkste randapparatuur.
Natuurlijk staan er in dit hoofdstuk ook verschillende programma's om de hardware aan te kunnen sturen of te kunnen manipuleren.
Voor dit hoofdstuk hebben we medewerking gekregen van de beste hardwarespecialist in ons land en dat is dhr C. Gans. Hij is ook opgenomen in ons servicebestand (Zie appendix D.) en U kunt hem dus ook bellen (Gelieve **NIET** op zondag.) voor vragen op het gebied van hardware.
Momenteel is dhr Gans bezig met een aantal leuke ontwikkelingen op het gebied van harware. Die ontwikkelingen zullen in dit hoofdstuk uitgebreid behandeld worden.
In het laatste deel van dit hoofdstuk zijn nog een aantal kleinere schema's te zien, eventueel met een kleine uitleg daarbij. Deze schema's zijn overzichten van eenvoudige stukjes hardware of van kleine aanpassingen die mogelijk zijn op de SHARP, zoals bijvoorbeeld een CGA-monitor op Uw SHARP.

8.1: Uitleg over de verchillende stukken hardware.

In dit deel van hoofdstuk 8 zullen verschillende stukken hardware, zoals de besturings-ic's, de HARD-DISK en nog veel meer behandeld worden. De programma's die op deze stukken hardware van toepassing zijn vindt U in het tweede deel van dit hoofdstuk.

EPROMS - In eproms kan informatie opgeslagen worden die er vast in op-opgeslagen blijft. Dit zijn dus de zogenaamde ROM-chips. Eproms zijn maar op één manier te programmeren en dat is door een epromprogrammer. Dit is een programma dat iets weg kan schrijven op een eprom.
De SHARP maakt ook gebruik van eproms voor de zogenaamde ROM-monitor. De programma's die bij het aanzetten van de SHARP dus op de adressen van 0000H t/m 1000H en E000H t/m FFFFH staan zijn opgeslagen op eproms. Deze eproms kunt U helaas niet zelf programmeren, maar wel vervangen door zelfgeprogrammeerde eproms. Dat wordt echter wel een dure aangelegenheid zoals op de volgende bladzijde zal blijken.

Mark en Jeroen de Rover hebben een complete nieuwe ROM-monitor gemaakt, maar stuitten op het probleem dat er tien eproms nodig zijn voor deze nieuwe ROM-monitor en één eprom kost bijna F100,=.

CPU De CPU is het hart van de computer, van daaruit worden alle handelingen gecontroleerd en wordt de snelheid van de handelingen bepaald. De CPU werkt natuurlijk samen met nog enkele andere delen, omdat de CPU niet alles zelf kan. Zo kennen we bijvoorbeeld de Memory controller, de I/O controller, de WAIT controller, controllers voor de VRAM, enz.
Hoe de verschillende controllers onderling in verbinding staan, wordt niet behandeld.

KLOKFREQUENTIE - De klokfrequentie van de SHARP is 3.547 MHz. Eigenlijk wordt er een frequentie van 17.734 MHz afgegeven door de oscilator. Dit wordt door een frequentiedeler gereduceerd tot 3.547 MHz en daarna aan de CPU doorgegeven. Niet alles werkt op deze 3.457 MHz. Voor sommige dingen is een andere frequentie nodig. Daarvoor worden andere frequentiedelers gebruikt.

DE VIDEO-RAM - De VIDEO-RAM kan hardwarematig aangestuurd worden, dat gaat via de poort CFH. Via het B-register kan bepaald worden wat er met de VIDEO-RAM moet gebeuren.

ANDERE IC'S - Er zijn veel ic's die enkel en alleen bedoeld zijn voor de aansturing van bepaalde randappararatuur of interne zaken, zoals de ic voor het geluid, de PSG. Hier volgen de nummers van de ic's en waar ze voor bedoeld zijn:
De 8255 wordt gebruikt voor het toetsenbord en de cassetterecorder. Aansturing van deze ic verloopt via de poorten $D0 t/m $D3.
De 8253 wordt gebruikt als Programmable Interval Timer.
enz.

Helaas konden we U niet heel erg veel vertellen over de verschillende ic's, omdat wij zelf helaas niet veel van hardware afweten. Over de randapparatuur waar we het nu over zullen gaan hebben kunnen we meer vertellen. Hoe de ic's precies werken en waarvoor ze bedoeld zijn is ook niet van belang voor U, omdat U alleen hoeft te weten hoe U er mee kunt werken en dat is bij de belangrijkste ic's al gedaan. In dit boek kunt U namelijk genoeg lezen over de VIDEO-RAM, de joystick, de PSG (voor geluid) en hieronder en op de volgende bladzijden zal daar nog veel meer bij komen.

DE QUICK-DISK - De QUICK-DISK is in hoofdstuk 0 eigenlijk al behandeld, maar zal hier nog eens behandeld worden.
Hoe kan een QUICK-DISK controleren of er een QD in de QUICK-DISK zit? Maakt U de QUICK-DISK open dan zult U links aan het begin een zwart staafje zien. Dat staafje zal naar beneden gedrukt worden als er een QD in de QUICK-DISK zit. Drukt U dit staafje dus naar beneden dan zal de computer in de veronderstelling zijn dat er een QD in de QUICK-DISK zit.

Op het klepje van de QUICK-DISK bevindt zich een ijzeren strookje en in de QUICK-DISK staan links en rechts koperkleurige staafjes. Maken deze staafjes contact met het strookje dan is de schijf beveiligd tegen formateren of wegschrijven. Dit komt omdat de lipjes dan uit de QD weg zijn en de koperkleurige staafjes door de gaatjes contact kunnen maken met het strookje.
De QD is via een interface aan te sluiten op de SHARP MZ-800. Dit houdt in dat de drive in principe dus niet speciaal voor de SHARP MZ-800 gemaakt is, anders zou er geen interface voor gemaakt zijn. Dat klopt ook wel, want de QUICK-DISK werd in eerste instantie voor MSX-computers gemaakt, zodoende dat er ook PHILIPS QD's voor de QUICK-DISK zijn.

DE CASSETTERECORDER - De cassetterecorder wordt nog door weinig mensen als enige medium bij de SHARP gebruikt. Dit heeft te maken met het feit dat de cassetterecorder veel te langzaam is en mensen meestal erg ongeduldig zijn. De cassetterecorder heeft echter als enorm voordeel dat het heel erg goedkoop in gebruik is, veel goedkoper dan de QUICK-DISK zelfs.
Een cassetterecorder wordt om deze reden vaak als tweede medium naast de QUICK-DISK gebruikt voor programma's die niet ieder moment gebruikt worden.
Een cassetterecorder heeft geen interface nodig, alleen een klein snoertje om het op de SHARP aan te sluiten.
Eventueel kan de cassetterecorder achter op de QUICK-DISK aangesloten worden.
De cassetterecorder kan veel sneller gebruikt worden dan meestal gedaan wordt. De cassette-recorder is namelijk ingesteld op een tamelijk lage overdrachtssnelheid en die snelheid kan drastisch opgevoerd worden. Het enige nadeel is dan dat bandjes op den duur sneller data kunnen verliezen en dat C90 bandjes niet zo goed te gebruiken zijn.
Een programma wordt altijd twee keer naar cassette geschreven en dat is ook absoluut niet nodig. In deel twee van dit hoofdstuk staat een aanpassing om in de ROM-monitor het geheugen zo te veranderen dat het programma slechts één keer weggeschreven zal worden.

DE RAM-DISK - De RAM-DISK wordt niet meer zo veel gebruikt als vroeger. Tegenwoordig hebben steeds meer mensen een FLOPPY-DISK bij hun SHARP staan.
De RAM-DISK wordt gebruikt voor opslag van programma's en is meestal 64K groot. Er zijn ook grotere RAM-DISKS, maar die worden zelden tot nooit gebruikt. Ook het gebruik van de vrije ruimte in de VRAM verjaagt de RAM-DISK.

Niet alleen het afnemen van het gebruik van de RAM-DISK heeft er toe geleid dat er in dit boek weinig over de RAM-DISK verteld wordt, ook het feit dat we zelf niet in het bezit zijn van een RAM-DISK en daarom niets kunnen controleren heeft er toe geleid dat de RAM-DISK op een tweede plan komt.

DE FLOPPY-DISK - Evenals bij de RAM-DISK geldt hier ook dat wij niet in het bezit zijn van een dergelijk apparaat en zodoende ook niet kunnen controleren of iets juist is. De FLOPPY-DISK heeft veel voordelen ten opzichte van de cassetterecorder en de QUICK-DISK. Het werkt veel sneller, is goedkoper in gebruik en er kunnen professionelere programma's mee gedraaid worden. De meeste programma's voor de FLOPPY zijn de zogenaamde CPM programma's. Wij kunnen U helaas niet veel over CPM vertellen, maar hebben daarvoor in de plaats een man in de servicegroep (Zie appendix D.) zitten die meer van CPM af weet. Hetzelfde geldt voor de RAM-DISK en de FLOPPY-DISK. Voor aansturing en andere zaken aangaande deze twee media hebben we ook een servicelijn beschikbaar en dat is de lijn van Hans Smits. Hij weet veel over de FLOPPY-DISK en de RAM-DISK. Veel FLOPPY gebruikers zullen het schitterende programma HS-INDEX van Hans Smits wel kennen.
Voor meer over het inwendige van de FLOPPY-DISK en de RAM-DISK kunt U terecht bij dhr C. Gans.

U ziet dat we helaas niet veel over de FLOPPY-DISK en de RAM-DISK in dit boek hebben staan, hetzelfde geldt voor de plotter. Daar staat echter tegenover dat we speciale servicelijnen hebben waar U altijd naar toe kunt bellen of schrijven en U krijgt ten alle tijden antwoord terug. Bij schriftelijke vragen is het dan wel noodzakelijk dat U een retourenvelop met postzegel bijsluit.
Nu zullen besprekingen volgen van enkele nieuwe ontwikkelingen op het gebied van de SHARP MZ-800. Hier is een uitgebreid gesprek met dhr C. Gans aan vooraf gegaan.

MEER GEHEUGEN - De mogelijkheid bestaat om het geheugen van de SHARP op te voeren naar het geheugen van een PC, dat is dus 512K. Dit extra geheugen is aan te sturen via bankswitsching. Er kan dan een blok van 16K geheugen per keer aangesproken worden. Er moet één blok van 16K zijn dat elke keer rechtstreeks aan te sturen blijft en waarin het besturinggedeelte komt. Vanuit dat blok kunnen dan alle andere blokken van 16K aangestuurd worden.
Een echte PC zal de SHARP daardoor niet direct worden, maar de mogelijkheden worden er enorm mee uitgebreid.
Een gewoon BASIC-programma zal echter niet langer gemaakt kunnen worden, omdat een BASIC-programma zich binnen één blok van 64K moet bevinden, omdat anders door het switschen tussen blokken enorm veel tijdverlies optreedt en het BASIC-programma dus veel te langzaam wordt.

Een probleem dat bij zo'n groot geheugen om de hoek komt kijken is het refreshen. Hoe moet dat nu gedaan worden. Achteraf is echter gebleken dat dit probleem veel minder groot is dan in eerste instantie werd verondersteld. Er bleken namelijk geen problemen te ontstaan bij het refreshen van het volledige geheugen. In twee seconden kon het hele geheugen gerefreshed worden en er bleken geen missing bytes te zijn. Als een byte namelijk niet gerefreshed wordt, kan dataverlies optreden.
Het geheugen kon zo snel gerefreshed worden, omdat het refreshen met blokken van 256 bytes tegelijk gaat.
Dhr C. Gans weet momenteel niet of deze aanpassing nog op de markt zal komen, omdat de SHARP MZ-gg er niet achter staat. De aanpassing om het geheugen uit te breiden wordt namelijk veel te duur als het op kleine schaal gebeurt. Tevens bestaat er nog geen software die gebruik maakt van deze geheugenuitbreiding.
De aansluiting van een expansion-board om de geheugen-ic's op vast te maken en de veranderingen in de SHARP zijn niet zo heel erg duur. Bij elkaar kost dat nog geen F100,=, maar dan komen de geheugen-ic's er nog bij. U hoeft niet gelijk het geheugen op te voeren tot 512K, een uitbreiding van 64K of 128K is in eerste instantie ook voldoende. Het probleem is echter dat de geheugen-ic's tamelijk duur zijn vanwege de enorme vraag ernaar. Er zijn er namelijk veel te weinig.
Mochten de ic's echter nog goedkoper worden dan zal de SHARP voor een heel erg schappelijke prijs uit te breiden zijn in geheugen. Dan bestaat alleen het probleem nog dat er geen software voor is, maar die zal er ook wel komen als veel mensen de extra geheugenuitbreiding willen hebben.
Heeft U belangstelling voor de geheugenuitbreiding en weet U zeker dat U nog een aantal jaren met Uw SHARP wilt werken dan kunt U ons een briefje sturen (Graag inclusief een retourenvelop met postzegel en aan Uzelf geadresseerd.) met daarop de grootte waarmee U het geheugen uit wilt breiden en wat U er voor over heeft. Mocht het blijken dat er genoeg belangstelling is en het financieel haalbaar is, zult U ongeveer in december 1990 bericht van ons krijgen.

DE HARDDISK - Wie naar de HCC-dagen in 1988 is geweest heeft bij een stand van de SHARP MZ-gg een HARD-DISK op de SHARP aangesloten kunnen zien.
U zult zich afvragen of het dan zomaar mogelijk is om een HARD-DISK op de SHARP aan te sluiten. Daar zijn twee antwoorden op mogelijk: Ja en Nee.
De HARD-DISK die U op de HCC-dagen heeft kunnen zien was een SCSI HARD-DISK. SCSI is een bepaalde standaard. Deze HARD-DISK is een zogenaamde intelligente HARD-DISK. Het is heel gemakkelijk om deze HARD-DISK aan te sturen, omdat er heel veel in de HARD-DISK is ingebakken. Er is dus ook geen speciale controller nodig.

Een SCSI HARD-DISK is door zijn ingebakken intelligentie wel een stuk duurder dan een gewone HARD-DISK, namelijk tussen de F1400,= en F1500,=. Een gewone HARD-DISK kost slechts de helft, maar daar staat wel tegenover dat een normale HARD-DISK een controller nodig heeft en die is er helaas niet voor de SHARP. Mocht er dus ooit nog een HD-controller voor de SHARP komen dan zal de aansluiting van een HD op de SHARP net zo duur worden als de aansluiting van een HD op een PC.
Een ander voordeel van de SCSI HARD-DISK is dat er maar een heel kort besturingsprogramma nodig is.
Of er ooit een HARD-DISK op Uw SHARP aangesloten kan worden is nog een groot vraagteken, omdat een SCSI HARD-DISK te duur is voor de normale SHARP-bezitter. Nu maar hopen dat er een controller komt voor een gewone HARD-DISK.

DE KLOKFREQUENTIE - De klokfrequentie is al eerder behandeld, maar er valt veel meer over te vertellen. Op de MSX-computer is het namelijk mogelijk om de klokfrequentie te verdubbelen door een simpele hardware-aanpassing. De vraag is nu of dat ook op de SHARP mogelijk is. In principe is het antwoord ja, maar dan zou een ic vervangen moeten worden en dat is niet één van de meest goedkope ic's.
De bedoelde ic is de ic die de klokfrequentie van de oscilator door vijf deelt en dan naar de CPU stuurt.

EEN MUIS - Dit bestaat in Nederland eigenlijk nog niet voor de SHARP. Er zijn wel muizen voor de SHARP, maar die worden gewoon op de joystickpoort aangesloten en werken dus als een gewone joystick, alleen dan in een ander jasje.
Een echte muis zal via een aparte aansluiting en controller aangesloten moeten worden.
De mogelijkheid is er natuurlijk wel, maar of iemand weet hoe een muiscontroller voor de SHARP gemaakt moet worden is nog maar even afwachten.

EEN LICHTPEN - Met een lichtpen kunnen verschillende kleuren herkend worden. U kunt dus bijvoorbeeld vijftien blokken in vijftien verschillende kleuren op het scherm zetten en in elk blok een bepaalde opdracht. Als door de lichtpen een bepaalde kleur herkend wordt, zal de opdracht in dat blok uitgevoerd worden.
De aansluiting van een lichtpen op de SHARP is niet heel erg moeilijk. We willen er zelf één ontwikkelen en eigenlijk had het ontwerp in dit boek moeten staan, maar door andere bijkomstigheden zijn we er nog niet aan toegekomen. Misschien zijn anderen er momenteel ook mee bezig, want een lichtpen is gemakkelijk te gebruiken, vergemakkelijkt een aantal handelingen, is gemakkelijk aan te sturen en is redelijk goedkoop. Het probleem is dat weinig mensen bezig zijn op hardwaregebied en zodoende komen er weinig nieuwe dingen bij.

8.2: Enkele programma's voor de hardware.

In het eerste deel zijn een aantal stukken hardware besproken. In dit deel zullen enkele programma's en instructies voor de hardware getoond worden. Ook staan er enkele dingen tussen voor hardware die niet in het eerste deel van dit hoofdstuk vermeld is.

HCOPY voor EPSON-printer.

In BASIC bestaat de instructie HCOPY om een afdruk van het scherm op de printer te maken. Deze instructie is echter alleen bedoeld voor specifieke SHARP-printers. Voor andere printers is een speciaal HCOPY-programma nodig. Helaas kunnen we niet het HCOPY-programma voor alle printers behandelen. Om het kort te houden plaatsen we hier het HCOPY-programma voor EPSON en EPSON-compatible printers, dat zijn namelijk de meest gebruikte printers. U kunt een HCOPY van het scherm maken door op CTRL-A te drukken of HCOPY in te typen.

```
10 DATA 24,30,67,46,83,67,67,69,56,56,46
,72,75,183,158,189,77,90,45,56,48,48
20 DATA 32,32,46,72,66,46,46,46,00,01,21
9,224,58,31,253,211,205,58,30,253,60
30 DATA 61,32,32,33,64,1,34,110,254,33,6
4,31,34,112,254,33,40,0,34,114,254
40 DATA 33,48,51,34,98,254,33,50,48,34,1
00,254,24,30,33,128,2,34,110,254
50 DATA 33,128,62,34,112,254,33,80,0,34,
114,254,33,48,54,34,98,254,33,52
60 DATA 48,34,100,254,33,87,254,126,254,
255,40,6,35,205,39,254,24,245,43
70 DATA 126,254,27,32,250,229,33,0,128,2
37,91,112,254,25,43,235,175,33,0
80 DATA 128,237,75,110,254,237,66,237,75
,114,254,9,221,33,116,254,237,75
90 DATA 110,254,9,237,75,114,254,175,237
,66,43,175,237,82,202,18,254,25
100 DATA 237,75,114,254,3,197,253,225,35
,253,45,40,57,175,55,23,56,246
110 DATA 245,62,7,237,75,114,254,245,8,2
41,47,7,7,7,61,50,222,253,241,245
120 DATA 166,40,4,221,203,0,254,8,9,61,2
42,203,253,175,237,75,110,254,237
130 DATA 66,58,116,254,205,39,254,50,116
,254,241,24,201,62,13,205,39,254
140 DATA 229,253,225,225,229,126,254,255
,40,6,35,205,39,254,24,245,253,229
150 DATA 225,24,140,225,42,110,254,175,2
05,39,254,43,124,181,32,247,62,13
160 DATA 205,39,254,219,225,201,245,219,
254,31,56,251,241,197,6,20,0,16,253
170 DATA 211,255,58,2,253,254,67,194,0,0
,62,128,211,254,58,12,253,254,75
180 DATA 194,0,0,58,16,253,254,77,194,0,
0,175,211,254,193,201,27,78,27,62
```

```
190 DATA 27,84,49,54,13,27,83,48,54,52,4
8,255,0,0,0,0,0,0,0,128,2,128,62
200 DATA 80,0,0,205,39
210 LIMIT $FCFF
220 POKE $14AE,205,0,253 :POKE $5D,0,253
230 FOR X=0 TO 374
240 READ A
250 POKE $FD00+X,A
260 NEXT X
270 FOR X=0 TO 15
280 READ A
290 POKE $FE57+X,A
300 NEXT X
310 DATA 27,65,8,0,0,0,0,0,13,27,75,64,1
,255,255,255
320 POKE $FD41,75,64,34,97
330 POKE $FD47,1,255
340 POKE $FD4A,99
350 POKE $FD61,76,128,34,97
360 POKE $FD67,2,255,34,99
370 POKE $FDCE,7,7,7,198,198
380 INIT"CRT:M1"
390 PRINT"[1] 40-KOLOMS."
400 PRINT"[2] 80-KOLOMS."
410 GET A$ :IF A$<"1" OR A$>"2" THEN 410
420 IF A$="1" THEN POKE $FD1E,0 ELSE POK
E $FD1E,1
```

Omdat de aanpassing voor andere printers achteraf toch weinig ruimte in beslag blijkt te nemen, volgt hier de aanpassing voor enkele andere printers:

Bij de meeste andere printers moet U het volgende invoeren:

```
POKE $FE5A,27,50
```

Helaas kunnen we geen assemblerlisting van het HCOPY programma geven, omdat het daar veel te lang voor is.

Veranderingen in de ROM-monitor.

Met een heel klein machinetaalprogrammaatje is de ROM-monitor van 0000H t/m 1000H om te toveren in RAM, U kunt er dan dus vrij veranderingen in aanbrengen.
In de Technical Reference Manual staat de assemblerlisting, inclusief uitleg van de hele ROM-monitor. Met dit boek bij de hand kunt U dus pecies zien hoe U leuke resultaten kunt bereiken met het veranderen van de ROM-monitor.
Het probleem is echter dat de meeste mensen niet in het bezit zijn van het TRM, daarom zullen wij hier de leukste veranderingen laten zien.

Het programma.

Op de volgende bladzijde vindt U eerst het benodigde programma.

```
1200 21   1204 00   1208 10   120C ED   1210 D1   1214 B0
1201 00   1205 20   1209 E5   120D B0   1211 D3   1215 C9
1202 00   1206 01   120A C5   120E E1   1212 E0
1203 11   1207 00   120B D5   120F C1   1213 ED
```

Dit programma kunt U opstarten met G1200.

Assemblerlisting van bovenstaand programma.

```
1200 LD HL,0000H          120E POP HL
1203 LD DE,2000H          120F POP BC
1206 LD BC,1000H          1210 POP DE
1209 PUSH HL              1211 OUT (E0H),A
120A PUSH BC              1213 LDIR
120B PUSH DE              1215 RET
120C LDIR
```

Nu kunt U het geheugen van 0000H t/m 1000H vrij veranderen. Hieronder en op de volgende bladzijden zullen een aantal voorbeelden gegeven worden van leuke veranderingen die U uit kunt voeren.

Voorbeeld 1: Een programma maar één keer wegschrijven.

Een programma wordt normaal gesproken altijd twee keer weggeschreven om zeker te weten dat het goed gebeurt. De verandering gaat als volgt:

U verandert adres 04E8 in 01 (oorspronkelijke waarde 02) en dat doet U als volgt:

-Voer eerst M04E8 in.
-Voer daarna de waarde 01 in en druk op <cr>.
-Druk hierna op SHIFT-BREAK en de aanpassing zit in de monitor.

Voorbeeld 2: Knipperen van cursor uitschakelen.

De cursor knippert constant. Wanneer U dit uit wilt schakelen moet U adres 057E in 00 veranderen. Oorspronkelijk heeft dit adres de waarde CD.

Voorbeeld 3: Teken van cursor veranderen.

Het teken van de cursor kan als volgt veranderd worden:
U verandert eerst de waarde van adres 09EE van 11 in 00.
Vervolgens verandert U de waarde van adres 09ED in de waarde van het teken dat U graag wilt hebben. (Oorspronkelijk 92.)
Als laatste verandert U adres 09EC van 3A in 3E.

Tekens veranderen in andere tekens.

Vanaf adres 0A92 staan de verschillende display codes van de tekens opgeslagen. Wanneer U hier veranderingen in aan gaat brengen, kan van een A bijvoorbeeld een I of iets anders gemaakt worden. Zo zijn er nog tal van mogelijkheden. Een leuk voorbeeld is als U de spatie gaat veranderen in iets anders. De spatie staat opgeslagen op adres 0AB2. Verandert U dat adres maar eens in bijvoorbeeld 3E.

Voorbeeld 4: Bovenste regel laten staan bij scrollen.

Normaal gaan alle regels omhoog en verdwijnt de bovenste regel uit het scherm als de onderkant van het scherm wordt bereikt. Het is heel gemakkelijk om bij dit scrollen de bovenste regel te laten staan en dat gaat als volgt:

-Voer op adres 0E6E de waarde 98 in. (Oorspronkelijk C0.)
-Voer op adres 0E71 de waarde 28 in. (Oorspronkelijk 00.)
-Voer op adres 0E74 de waarde 50 in. (Oorspronkelijk 28.)
-Voer op adres 0E7C de waarde 28 in. (Oorspronkelijk 00.)
-Voer op adres 0E7F de waarde 50 in. (Oorspronkelijk 28.)

Voorbeeld 5: Andere voor- en achtergrondkleur.

U kunt de voor en achtergrondkleur zelf bepalen en die kleur blijft ook staan! Dat gaat op de volgende manier:

-Op adres 0E87 voert U de waarde van de nieuwe kleur in. (Oorspronkelijk 71.)
-Op adres 0E49 voert U dezelfde waarde in. (Oorspronkelijk ook 71.)
-Daarna drukt U op SHIFT/INST en de verandering is doorgevoerd.

Voorbeeld 6: Functie van enkele toetsen veranderen.

Normaal gesproken heeft het drukken van DEL de functie om iets te corrigeren. Hier kan heel gemakkelijk een andere functie aan gegeven worden door de sprongtabel iets te veranderen.
We willen van de DEL-toets een toets maken die het scherm één regel omhoog scrollt, dat gaat als volgt:

-Verander het adres 0EB8 in de waarde 6D. (Oorspronkelijk F8.)
-Druk hierna eens op de DEL-toets.

Voorbeeld 7: Bij CLS cursor naar rechts onder laten gaan.

Normaal gaat de cursor naar de linker bovenhoek wanneer er op de toetsen SHIFT en INST (CLS) wordt gedrukt. U kunt de cursor echter op iedere willekeurige plaats op het scherm laten komen. Wanneer U de cursor in de rechter onderhoek wilt laten verschijnen dan gaat dat als volgt:

-Verander het adres 0E4E in 27. (Oorspronkelijk 00.)
-Verander het adres 0E4F in 18. (Oorspronkelijk 00.)
-Druk hierna eens op SHIFT/INST.

Voorbeeld 8: Teken van cursor bij knipperen veranderen.

Normaal ziet U bij het knipperen van de cursor de cursor verdwijnen en vervolgens weer terugkomen. Bij het verdwijnen van de cursor kunt U zelf een willekeurig teken op het scherm laten komen, bijvoorbeeld een vol blokje. Dat gaat als volgt:

-Op het adres 097D komt de waarde 00. (Oorspronkelijk 11.)
-Op het adres 097C komt de waarde 43. (Oorspronkelijk 8E.)

-Op het adres 097B komt de waarde 3E. (Oorspronkelijk 3A.)

Zo zijn er natuurlijk nog tal van voorbeelden, waarvan de leukste allemaal gepubliceerd zijn.

Joystickprogramma.

Navolgend programma is een programma voor de joystick in poort 1. In het midden van het scherm wordt een soort joystick getekend. Wanneer de joystick in een bepaalde richting beweegt, zal dat te zien zijn op het scherm. Dit programma heeft echter niet alle mogelijkheden in zich. Het gelijktijdig drukken van de vuurknop en het bewegen van de joystick wordt niet aangegeven. U kunt dit eventueel zelf heel gemakkelijk inbouwen. Elders in dit boek staat namelijk een overzicht van alle mogelijkheden met de joystick en de bijbehorende waardes.
Aan de hand van de assemblerlisting kunt U eventueel zelf aan de gang.

```
2000 3E   2010 23   2020 D9   2030 58   2040 20   2050 C3   2060 19
2001 C6   2011 23   2021 DB   2031 20   2041 21   2051 1E   2061 36
2002 CD   2012 36   2022 F0   2032 FE   2042 00   2052 20   2062 01
2003 DC   2013 43   2023 FE   2033 FE   2043 D8   2053 23   2063 C9
2004 0D   2014 23   2024 EF   2034 CC   2044 0E   2054 23   2064 11
2005 21   2015 23   2025 CC   2035 5D   2045 19   2055 36   2065 50
2006 A3   2016 36   2026 6B   2036 20   2046 06   2056 01   2066 00
2007 D1   2017 5A   2027 20   2037 FE   2047 28   2057 C9   2067 19
2008 36   2018 11   2028 FE   2038 FD   2048 36   2058 2B   2068 36
2009 50   2019 4E   2029 F7   2039 CC   2049 71   2059 2B   2069 01
200A 11   201A 00   202A CC   203A 64   204A 23   205A 36   206A C9
200B 4E   201B 19   202B 53   203B 20   204B 10   205B 01   206B 36
200C 00   201C 36   202C 20   203C FE   204C FB   205C C9   206C 01
200D 19   201D 58   202D FE   203D FF   204D 0D   205D 11   206D C9
200E 36   201E 21   202E FB   203E C2   204E 20   205E B0
200F 45   201F F3   202F CC   203F 1E   204F F6   205F FF
```

```
Filename? JOYSTICK
Top adrs? 2000
End adrs? 206D
Exc adrs? 2000
Opstarten met G2000.
```
Dit programma is alleen te stoppen door op de resetknop te drukken.

Assemblerlisting van het programma JOYSTICK.

```
2000 LD A,C6H
2002 CALL 0DDCH
2005 LD HL,D1A3H
2008 LD (HL),50H
200A LD DE,004EH
200D ADD HL,DE
200E LD (HL),45H
2010 INC HL
2011 INC HL
2012 LD (HL),43H
2014 INC HL
2015 INC HL
2016 LD (HL),5AH
2018 LD DE,004EH
201B ADD HL,DE
201C LD (HL),58H
201E BEGIN:  LD HL,D9F3H
2021 IN A,(F0H)
2023 CP EFH
2025 CALL Z,VKNOP
2028 CP F7H
202A CALL Z,RECHTS
202D CP FBH
202F CALL Z,LINKS
```

```
2032 CP FEH
2034 CALL Z,BOVEN
2037 CP FDH
2039 CALL Z,ONDER
203C CP FFH
203E JP NZ,BEGIN
2041 LD HL,D800H
2044 LD C,19H
2046 CLEAR:  LD B,28H
2048 LOOP:  LD (HL),71H
204A INC HL
204B DJNZ LOOP
204D DEC C
204E JR NZ,CLEAR
2050 JP BEGIN
2053 RECHTS:  INC HL
2054 INC HL
2055 LD (HL),01H
2057 RET
2058 LINKS:  DEC HL
2059 DEC HL
205A LD (HL),01H
205C RET
205D BOVEN:  LD DE,FFB0H
2060 ADD HL,DE
2061 LD (HL),01H
2063 RET
2064 ONDER:  LD DE,0050H
2067 ADD HL,DE
2068 LD (HL),01H
206A RET
206B VKNOP:  LD (HL),01H
206D RET
```

Uitleg van het programma JOYSTICK:

Het programma zit heel eenvoudig in elkaar, een uitgebreide uitleg zal dus waarschijnlijk niet nodig zijn.
In het begin wordt eerst het beeldscherm schoongemaakt.
Vervolgens wordt de 'joystick' op het scherm getekend.
Daarna wordt de joystick uitgelezen en wordt naar het overeenkomende adres gesprongen wanneer de joystick bewogen is of de vuurknop ingedrukt is.
Is de joystick niet aangeraakt dan wordt vanaf adres 2041 het beeldscherm weer in de originele kleuren hersteld.
De methode die hier gebruikt wordt heeft een klein nadeel. Probeert U de joystick eens als roerstaaf uit dan zult U zien dat alle hokjes (uitgezonderd die van de vuurknop) zwart worden. Zolang de joystick bewogen wordt kan de routine die op adres 2041 begint niet doorlopen worden en zullen de hokjes zwart blijven.

Programma's voor en meer informatie over de QUICK-DISK.

In de ROM-monitor zitten een aantal vaste routines voor de QUICK-DISK, zoals het laden en saven van programma's, het opvragen van de Directory, het formateren van een QD en nog een aantal van zulke zaken.
De hele besturing van de QUICK-DISK begint op adres E010. Op adres 1130 moet de CODE staan die bepaald wat er met de QD moet gebeuren.
De volgende CODES kunnen voor komen:

01 - READY CHECK: Controleert of de QD inzetbaar is. Zo niet dan zal de C-vlag op 1 gezet worden.
02 - FORMAT: Formateert een QD.
03 - READ: Leest een datablok in vanaf de Headpoint.
04 - SAVE: Schrijft een datablok naar QD vanaf de Headpoint.
05 - HEADPOINTCLEAR: Initialiseert de Headpoint.
06 - Motor stop.

Wanneer U bijvoorbeeld op het adres 1130 de waarde 02 neer zet en daarna GE010 in voert, zal de QD (indien aanwezig) geformateerd worden.

Het spel ADVOKA overzetten naar QD.

Het spel ADVOKA, dat bij de meeste mensen thuis ligt, bestaat alleen als cassette-versie. ADVOKA bestaat uit twee delen. Deel 1 verandert de characters en laadt vervolgens deel twee van cassette. Wanneer we nu deze laadroutine voor cassette veranderen in een laadroutine voor QD, kunnen we dat spel spel vanaf QD draaien. Om ADVOKA nu van cassette naar QD over te zetten moet U het volgende doen:

-Spoel eerst de cassette terug naar het begin en doe een QD met voldoende ruimte in de QUICK-DISK.
-Ga vervolgens in de MONITOR en voer QX in.
-Druk op PLAY.
-Als het programma geladen is moet U de cassetterecorder stoppen. Spoel de cassette echter NIET terug.
-Druk op SHIFT/BREAK.
-Voer M1A1C in.
-Voer op de volgende adressen de volgende waarden in:
1A1C CD
1A1D 00
1A1E 3C
-Druk weer op SHIFT/BREAK.
-Voer M1B00 in.
-Voer op de volgende adressen de volgende waarden in:

1B00	11	1B08	1A	1B10	F2	1B18	34	1B20	21	1B28	E0	1B30	FF
1B01	32	1B09	77	1B11	CD	1B19	11	1B21	03	1B29	DA	1B31	FF
1B02	3C	1B0A	23	1B12	F7	1B1A	2A	1B22	01	1B2A	00	1B32	41
1B03	21	1B0B	13	1B13	EE	1B1B	06	1B23	22	1B2B	00	1B33	0D
1B04	A3	1B0C	10	1B14	2A	1B1C	11	1B24	30	1B2C	CD		
1B05	11	1B0D	FA	1B15	04	1B1D	22	1B25	11	1B2D	E8		
1B06	06	1B0E	CD	1B16	11	1B1E	32	1B26	CD	1B2E	E2		
1B07	02	1B0F	5F	1B17	22	1B1F	11	1B27	10	1B2F	C9		

-Voer M10F0 in en voer op dat adres de waarde 01 in.
-Voer M1104 in.
-Voer op de volgende adressen de volgende waarden in:
1104 34 1107 33
1105 09 1108 00
1106 00 1109 3B
-Voer GF168 in.
-Geef als antwoord Y op Y/N.
-Voer QX in.
-Druk op PLAY.
-Geef als Filename : A
-Nu staat alles op QD. Opstarten met het programma Advoka.
-Op de vraag RESTART Y/N Y antwoorden.

Assemblerlisting vanaf adres 1B00 t/m 1B33.

```
1B00 LD DE,3C32H
1B03 LD HL,11A3H
1B06 LD B,02H
1B08 FNAME:  LD A,(DE)
1B09 LD (HL),A
1B0A INC HL
1B0B INC DE
1B0C DJNZ FNAME
1B0E CALL F25FH
1B11 CALL EEF7H
1B14 LD HL,(1104H)
1B17 LD (1134H),HL
1B1A LD HL,(1106H)
1B1D LD (1132H),HL
```

```
1B20 LD HL,0103H          1B2F RET
1B23 LD (1130H),HL        1B30 DEFB FFH
1B26 CALL E010H           1B31 DEFB FFH
1B29 JP C,0000H           1B32 DEFB 41H
1B2C CALL E2E8H           1B33 DEFB 0DH
```

Uitleg van bovenstaand programma en alle aanpassingen:

Eerst moet U het volgende weten: De veranderingen die hier doorgevoerd worden staan vanaf adres 1B00. In werkelijkheid is dit adres 3C00. Dit heeft te maken met het feit dat ieder programma dat met QX ingeladen wordt vanaf adres 1200 in het geheugen komt te staan, ongeacht waar het programma in werkelijkheid begint.
Op de adressen 1B00 t/m 1B0D wordt de Filename A op de juiste plaats in het geheugen gezet, zodat het programma uiteindelijk onder deze filename het volgende programma zal laden.
U kunt de filename gemakkelijk anders maken, want de data staan opgeslagen op de adressen 1B32 en 1B33. De code 0D geeft het eind van de filename aan.
CALL F25FH zorgt er voor dat de Headpoint geïnitialiseerd wordt.
CALL EEF7H zorgt er voor dat er naar het bedoelde programma op QD gezocht gaat worden.
Op de adressen 1134 en 1135 komt de lengte van het programma te staan. Na een programma van QD geladen te hebben, staat deze waarde altijd op de adressen 1104 en 1105.
Op de adressen 1132 en 1133 komt het startadres van het programma te staan. Na een programma van QD geladen te hebben, staat deze waarde altijd op de adressen 1106 en 1107.
Op de adressen 1130 en 1131 komen de codes te staan voor de juiste aansturing van de QD en dat zijn in dit geval de codes 03 en 01. De code 03 controleert of de QD gebruiksgereed is, zoals al eerder verteld is. Is de QD niet gereed dan zal de C-vlag op 1 gezet worden en zal vervolgens de computer gereset worden door naar adres 0000H te springen.
CALL E2E8 zorgt er voor dat de QUICK DISK stopt.

Uitleg over de veranderingen:

Op de adressen 1A1C t/m 1A1E wordt aangegeven dat de QD-laadroutine aangeroepen moet worden. In het originele programma staat hier een CALL die er voor zorgt dat het eerstvolgende programma van cassette geladen wordt.
Op het adres 10F0 staat origineel een waarde die aangeeft dat het programma geen OBJ, BTX of wat voor soort programma dan ook is. Wanneer hier de code 01 ingevoerd wordt, betekent dat dat het om een OBJ-file gaat.
Op de adressen 1104 en 1105 komt de lengte van het eerste programma te staan.
Op de adressen 1106 en 1107 komt het beginadres van het eerste programma te staan.
Op de adressen 1108 en 1109 komt het startadres van het eerste programma te staan.
GF168 zorgt er voor dat het programma naar QD weggeschreven wordt. Dit is een routine die normaal gebruikt wordt bij de instructie QC. Wanneer het programma geladen is, wordt bij adres F168 verder gegaan.

Een FORMAT-routine.

U kunt zelf heel eenvoudig een FORMAT-routine schrijven. Die kunt U bijvoorbeeld vanaf adres 2000 neerzetten. De routine komt er dan als volgt uit te zien:

```
2000 CD   2006 3E   200C 10   2012 11   2018 C9   201E 45   2024 45
2001 27   2007 02   200D E0   2013 19   2019 51   201F 54   2025 45
2002 EF   2008 32   200E DA   2014 20   201A 44   2020 20   2026 44
2003 DA   2009 30   200F 12   2015 CD   201B 20   2021 47   2027 0D
2004 12   200A 11   2010 20   2016 15   201C 4E   2022 45
2005 20   200B CD   2011 C9   2017 00   201D 49   2023 52
```

Assemblerlisting van bovenstaand programma.

```
2000 CALL EF27H
2003 JP C,ERROR
2006 LD A,02H
2008 LD (1130H),A
200B CALL E010H
200E JP C,ERROR
2011 RET
2012 ERROR:  LD DE,TEKST
2015 CALL 0015H
2018 RET
2019 TEKST:  DEFB 51H
201A DEFB 44H
201B DEFB 20H
201C DEFB 4EH
201D DEFB 49H
201E DEFB 45H
201F DEFB 54H
2020 DEFB 20H
2021 DEFB 47H
2022 DEFB 45H
2023 DEFB 52H
2024 DEFB 45H
2025 DEFB 45H
2026 DEFB 44H
2027 DEFB 0DH
```

Uitleg van het formateerprogramma:

Met CALL EF27H wordt gecontroleerd of de QD gereed is. Is dat niet het geval dan zal de C-vlag op 1 gezet worden en zal er naar de routine ERROR gesprongen worden en zal de tekst QD NIET GEREED op het scherm verschijnen.
Vervolgens wordt de code 02 op adres 1130 gezet en de routine vanaf E010 aangesproken. Dit betekent dat de QD geformateerd wordt. Is de QD echter schrijfbeveiligd of is er iets anders met de QD waardoor hij niet geformateerd kan worden dan zal er ook naar de routine ERROR gesprongen worden. De tekst zal in dat geval niet de juiste zijn. U weet door deze tekst in ieder geval dat de schijf niet geformateerd is. CALL 0015H wordt elders in dit boek uitvoerig besproken.

Enkele programma's voor de RAM-DISK.

Programma 1: Iets naar de RAM-DISK schrijven.

In de ROM-monitor bestaan helaas geen routines voor de RAM-DISK, daarom volgt hier een programma dat een bepaald geheugenblok naar de RAM-DISK schrijft. In dit programma schrijven we het blok van 1200 t/m 3000 naar de RAM-DISK.

```
4000 21   4002 12   4004 00   4006 01   4008 1E   400A E5   400C F5
4001 00   4003 11   4005 00   4007 00   4009 1A   400B D5   400D C5
```

```
400E 6F   4013 79   4018 E1   401D 78   4022 C3
400F 44   4014 C1   4019 D1   401E B1   4023 AD
4010 0E   4015 F1   401A 23   401F C2   4024 00
4011 EB   4016 D3   401B 0B   4020 09
4012 ED   4017 EA   401C 13   4021 40

Filename? RAM-DISK-SAVE
Top adrs? 4000
End adrs? 4024
Exc adrs? 4000
Opstarten met G4000.
Dit programma stopt vanzelf.
```

Assemblerlisting van het programma RAM-DISK-SAVE.

```
4000 LD HL,1200H
4003 LD DE,0000H
4006 LD BC,1E00H
4009 BEGIN:  LD A,(HL)
400A PUSH HL
400B PUSH DE
400C PUSH AF
400D PUSH BC
400E LD A,L
400F LD B,H
4010 LD C,EBH
4012 OUT (C),A
4014 POP BC
4015 POP AF
4016 OUT (EAH),A
4018 POP HL
4019 POP DE
401A INC HL
401B DEC BC
401C INC DE
401D LD A,B
401E OR C
401F JP NZ,BEGIN
4022 JP 00ADH
```

Uitleg van bovenstaand programma:

In HL komt het beginadres van het gedeelte dat U naar RAM-DISK wilt schrijven.
In DE komt het beginadres vanwaar op de RAM-DISK geschreven gaat worden.
In BC komt de lengte van het datablok dat naar de RAM-DISK geschreven moet worden.
Daarna worden alle bytes stuk voor stuk naar de RAM-DISK gestuurd.

Programma 2: Datablok van de RAM-DISK naar het gewone geheugen sturen.

```
5000 11   5006 01   500C C5   5012 79   5018 D1   501E B1   5024 00
5001 00   5007 00   500D 6F   5013 C1   5019 12   501F C2
5002 60   5008 1E   500E 44   5014 F1   501A 23   5020 09
5003 21   5009 E5   500F 0E   5015 DB   501B 13   5021 20
5004 00   500A D5   5010 EB   5016 EA   501C 0B   5022 C3
5005 00   500B F5   5011 ED   5017 E1   501D 78   5023 AD

Filename? RAM-DISK-LOAD
Top adrs? 5000
End adrs? 5024
Exc adrs? 5000
Opstarten met G5000.
Dit programma stopt vanzelf.
```

Assemblerlisting van het programma RAM-DISK-LOAD.

```
5000 LD DE,6000H
5003 LD HL,0000H
5006 LD BC,1E00H
5009 BEGIN:  PUSH HL
500A PUSH DE
500B PUSH AF
500C PUSH BC
500D LD A,L
500E LD B,H
500F LD C,EBH
5011 OUT (C),A
5013 POP BC
5014 POP AF
5015 IN A,(EAH)
5017 POP HL
5018 POP DE
5019 LD (DE),A
501A INC HL
501B INC DE
501C DEC BC
501D LD A,B
501E OR C
501F JP NZ,BEGIN
5022 JP 00ADH
```

Uitleg van bovenstaand programma:

Nu staat in DE het beginadres voor het geheugenblok waar de informatie van de RAM-DISK in moet komen.
In HL staat het adres van de RAM-DISK waar het te transformeren geheugenblok begint.
In BC staat weer de lengte van het geheugenblok.
Nu worden alle data van RAM-DISK naar het gewone geheugen geschreven.

Voor- en nadelen van een RAM-kaart.

(Onderstaand artikel is een uitreksel van een artikel van Hans Smits.)
Er zijn een aantal verschillende RAM-kaarten te koop. De formaten zijn 64K, 256K, 512K en 1024K. De kleinste versie wordt het meest gebruikt. Dit komt onder andere door de hoge prijzen die er voor grotere RAM-kaarten wordt gevraagd. Omdat de RAM-kaart niet een geheugenuitbreiding is, maar een opslagmedium van tijdelijke aard, is het ook niet noodzakelijk om met een grote RAM-kaart te werken.
Vanuit de BASIC is de RAM-kaart heel gemakkelijk aan te sturen. Er zijn een aantal heel erg handige instructies aanwezig die meestal in combinatie met de QD gebruikt worden. Normaal moeten met het programma DELETE programma's van QD gehaald worden of programma's een andere naam gegeven worden. Met de RAM-kaart kan dat rechtstreeks in de BASIC.

Voordelen RAM-kaart:

-Snel opslagmedium voor o.a. bestanden en CHAIN-programma's.
-Informatie blijft behouden na een reset.
-Zeer geschikt in combinatie met QD. (LOADALL/SAVEALL)
-Het is mogelijk meer bestanden tegelijk te openen.
-Programma's kunnen er afzonderlijk afgehaald worden met DELETE.
-In BASIC is de RAM-kaart te gebruiken als printerbuffer.
-Er is minder slijtage van de QUICK-DISK en QD's.
-Is te gebruiken om tussentijds programma's te saven.
-Is gemakkelijk toegankelijk in machinetaal.

Nadelen RAM-kaart:

-Informatie gaat verloren bij uitzetten computer.
-64 Kb kaart is niet interessant voor CPM-gebruikers.

8.3: Kleine hardware-aanpassingen

Hieronder en op de volgende bladzijden zullen een aantal voorbeelden volgen van kleine stukjes hardware die nodig zijn om iets speciaals op de SHARP te doen. Tevens zal er een zeer uitgebreide uitleg bij staan. Op blz.162 vindt U de schemaatjes die bij de aanpassingen horen.

Aanpassing 1: CGA-monitor op de SHARP.

De beeldschermen die voor PC's gebruikt worden zijn vaak veel beter dan een gewone tv of zelfs de SHARP monitor.
Veel mensen willen later de SHARP weg doen en een PC kopen, waarom dan alles specifiek voor de SHARP kopen en later alles weer verkopen, terwijl U ook PC-spullen op de SHARP aan kan sluiten. Een CGA-monitor is daar een goed voorbeeld van.
Helaas is een kleine aanpassing nodig om een CGA-monitor op de SHARP aan te sluiten en kan dat niet rechtstreeks. Hoe zit die aanpassing precies in elkaar?

De MZ-800 is voorzien van een **drietal** mogelijke aansluitingen voor een beeldscherm terminal. De **eerste** aansluiting geeft een gemoduleerd signaal dat bedoeld is voor aansluiting op de normale TV.
Op een zwart/wit televisie is redelijk te werken. De tekst is in de 80-koloms mode echter vaak slecht te lezen. Met kleurentelevisies is het vaak nog bedroefder gesteld. Hier is in de 80-koloms mode meestal helemaal niets te lezen. Dat gaat echter niet in alle gevallen op, want Arjan gebruikt een gewone SHARP-televisie op zijn SHARP en heeft daarop een perfect beeld, zowel in de 40- als 80-koloms mode. Het beeld is bijna gelijk aan dat van een SHARP-monitor.
Het schakelaartje in het midden moet natuurlijk wel in de juiste stand gezet worden. Deze moet op B/W (Black/White) staan wanneer U met een zwart/wit televisie werkt. Bij kleurentelevisies komt deze schakelaar uiteraard op COLOR.
De **tweede** aansluiting geeft een samengesteld videosignaal. Dit signaal is bedoeld voor monitoren en voor TV's die een zogenaamde SCART-plug hebben. Ook hier geldt natuurlijk dat de schakelaar in de goede stand moet komen te staan.
De **derde** aansluiting is de RGB-aansluiting, een 8-polige aansluiting die o.a. voor de SHARP kleurenmonitor wordt gebruikt. Deze aansluiting is voor ons van belang en daarom zullen we hem eens nader bekijken.
Deze plug heeft de volgende signalen tot zijn beschikking:
-De kleursignalen RGB/I, waaruit de 16 kleurmogelijkheden samengesteld kunnen worden.
-Een massa-aansluiting.
-Twee synchronisatiesignalen: Hsync en Vsync.
Alle signalen hebben TTL niveau, dwz een logische '0' is 0 Volt en een logische '1' is 5 volt.
Het videobeeld is opgebouwd uit een aantal lijnen die achtereenvolgens van links naar rechts over het scherm getekend worden. Eerst worden alle oneven lijnen en vervolgens alle even lijnen getekend. De Hsync zorgt er voor dat elke lijn op het juiste moment 'gestart' wordt. Het aantal lijnen dat per seconde getekend wordt is 15625 (lijnfrequentie). Evenals het begin van elke lijn wordt ook het begin van elke beeldschermopbouw gesynchroniseerd. Vijfentwintig keer per seconde wordt een compleet nieuw beeld getekend.

Er wordt vaak gesproken over de rasterfrequentie van een TV. De rasterfrequentie van de SHARP is 50 Hz, omdat er vijftig keer per seconde gesynchroniseerd wordt. (Oneven en even lijnen!)
Dan kijken we als laatste nog even naar het oplossend vermogen van de SHARP, dat is een oplossing van 25 bij 80 regels. Dit houdt in: 200 (verticaal) bij 640 (horizontaal) beeldpunten.
Er is een vrij groot ongebruikt gedeelte op het scherm, de zogenaamde border. In verband hiermee is het handig om een monitor met een groter oplossend vermogen te gebruiken.
De VIDEO-kaarten voor PC's bevatten dezelfde eigenschappen als hier beschreven zijn. De CGA-kaart zorgt ook voor een RGB/I kleursignaal en een Hsync en een Vsync. Hierbij wordt er echter gesynchroniseerd op de positieve puls en bij de SHARP is dat op de negatieve puls.
De lijnfrequentie is 15,6 kHz en de rasterfrequentie is 60 Hz (zie figuur 3.)
Bij de EGA-kaart is de lijnfrequentie 21.8 kHz en de rasterfrequentie 60 Hz.

<u>De oplossing.</u>

De specificaties van de meeste monitoren zijn tegenwoordig zo dat zowel rasterfrequenties van 50 Hz als 60 Hz geaccepteerd worden. CGA monitoren zijn dus in principe op de MZ-800 aan te sluiten. Er rest alleen nog het probleem van de onjuiste polariteit van de beide synchronisatiesignalen. Dit is gemakkelijk op te lossen door beide signalen te inverteren. Hiervoor moet U een kleine schakeling toevoegen. De voeding voor de kleine schakeling kan van de voedingsaansluiting voor de plotter gehaald worden. In figuur 4 ziet U het schema.
Nog even een kleine waarschuwing.
Er zijn twee soorten 8-polige pluggen en die lijken veel op elkaar. In figuur 5 kunt U zowel de onjuiste als de juiste plug zien.
Met deze aanpassing zult U meer kijkplezier hebben op Uw SHARP.
Succes bij het proberen.

<u>Aanpassing 2: Meeluisterschakeling voor de MZ-800.</u>

Bij deze aanpassing horen twee kleine tekeningen/schema's en die vindt U weer op blz.162.
Het bedoelde ic is de 8443E0
M74LS14P
Deze ic zit onder het toetsenbord.
De aan- en uitschakelaar komt aan de buitenkant van de computer.
De schakelaar moet verbonden worden met pootje 2 van het bedoelde ic.

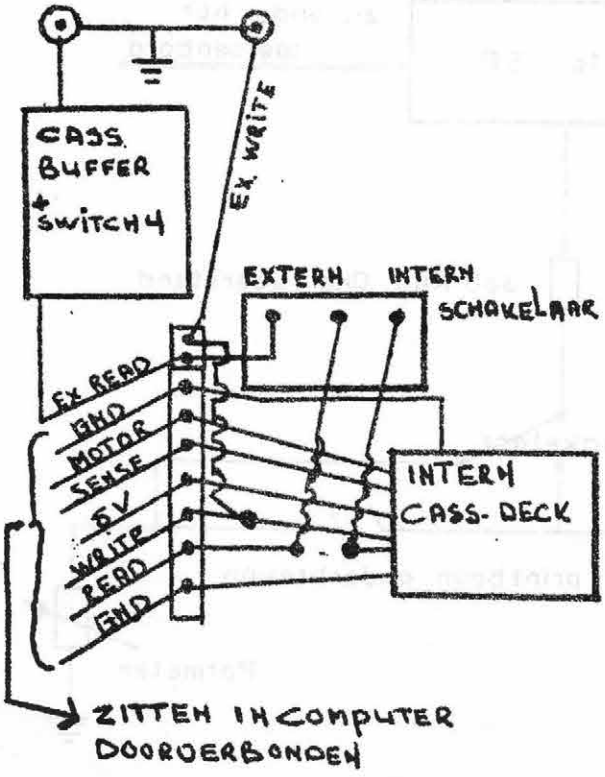
<u>Aanpassing 3: Joystick op de SHARP.</u>

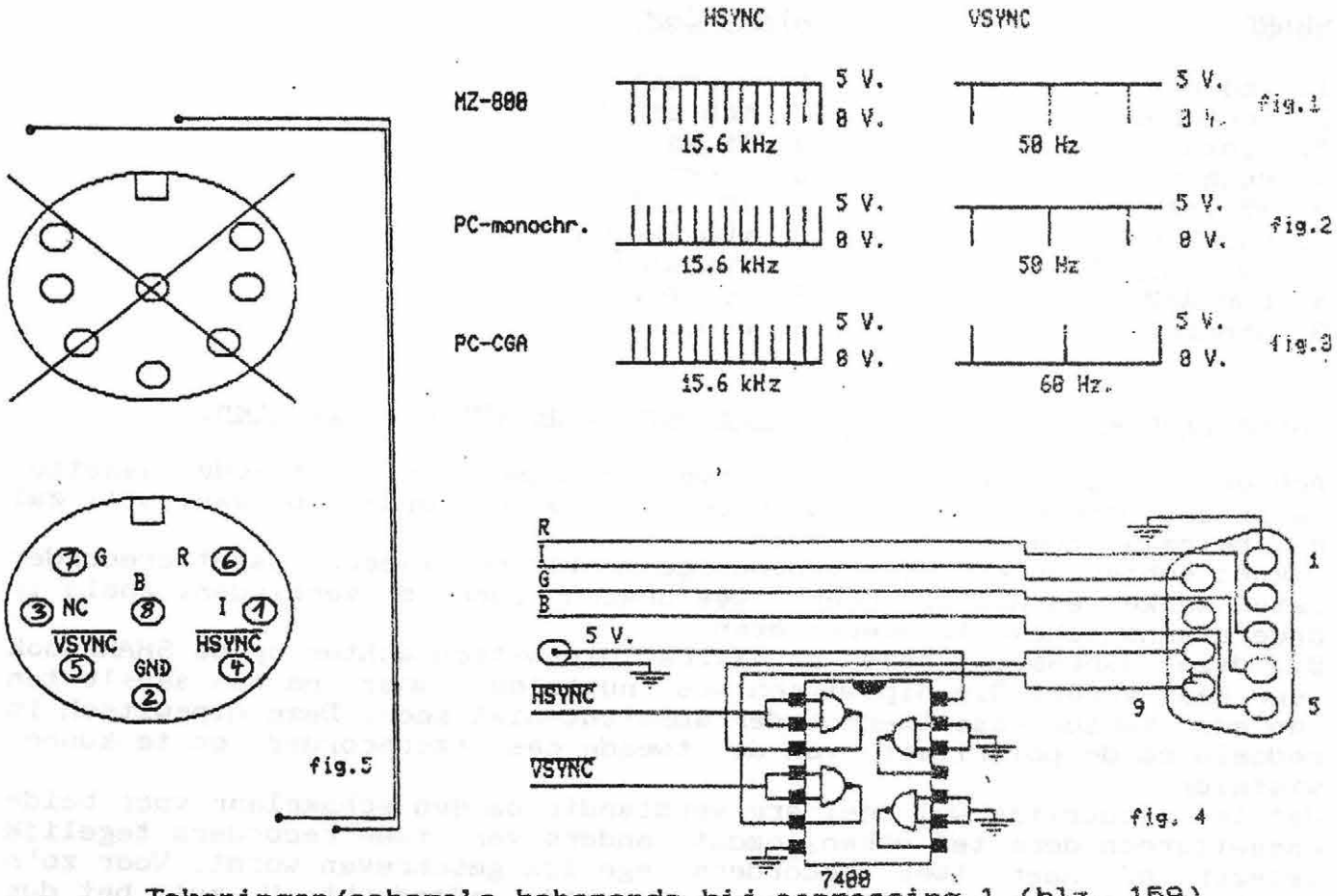
Er zijn een aantal joysticks die niet helemaal voor 100% werken op de SHARP. In de meeste gevallen gaat het om de vuurknop die niet werkt. De oplossing hiervoor is heel erg simpel. U hoeft alleen 4 draadjes om te wisselen.
Op bladzijde 7-11 van het handboek vindt U het overzicht van de joystick. Daarop is de tabel op de volgende bladzijde gebaseerd.

SHARP:	ATARI/CBM:
1. vooruit	1. vooruit
2. achteruit	2. achteruit
3. links	3. links
4. rechts	4. rechts
5. +5 volt	7. +5 volt
6. vuurknop 1	6. vuurknop 1
7. vuurknop 2	5. vuurknop 2
8. com A/B	9. com A/B
9. minpool	8. minpool

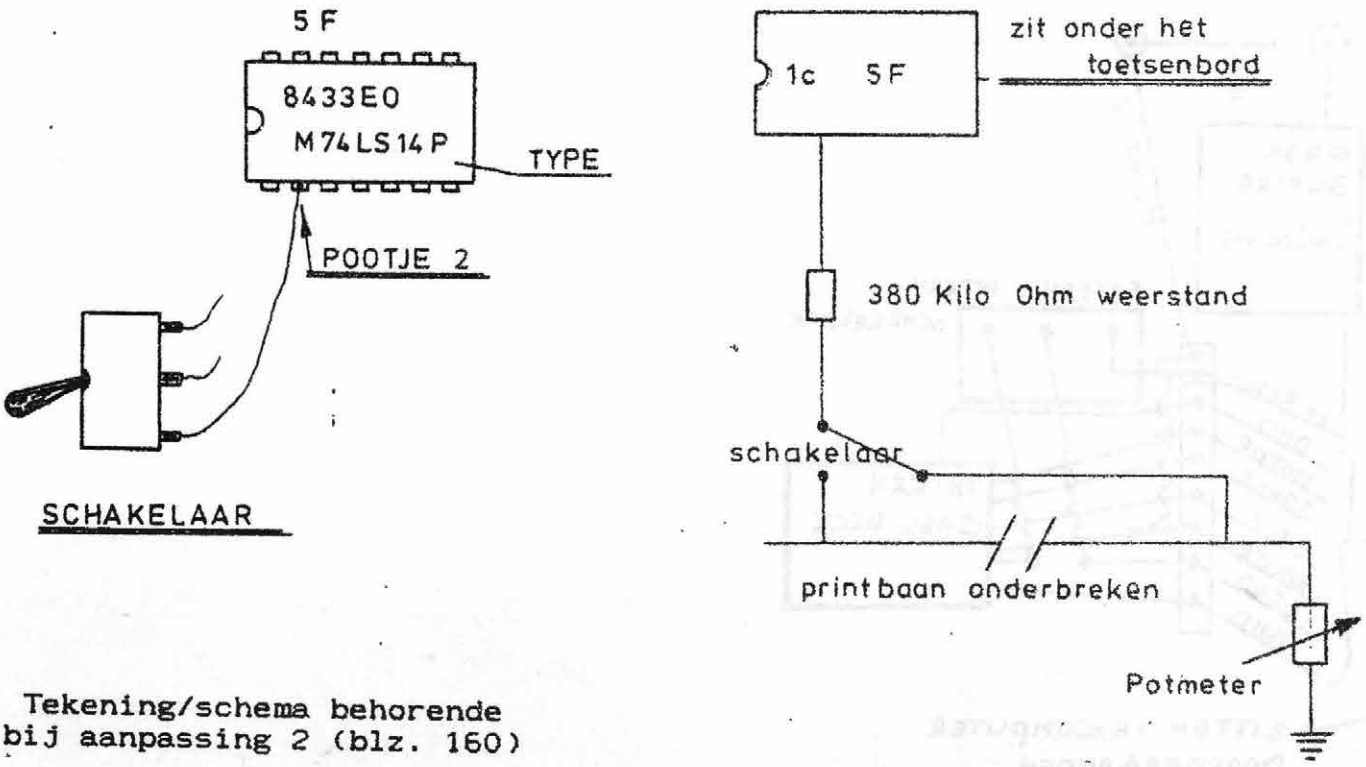
Aanpassing 4: Tweede cassetterecorder op de SHARP aansluiten.

Achter op de SHARP ziet U twee pluggen voor een tweede cassetterecorder. Wanneer U daar echter een cassetterecorder op aansluit, zal dat helemaal geen effect hebben.
U kunt echter op een heel eenvoudige manier een tweede cassetterecorder laten werken en dat kan door twee draden door te verbinden, zoals in onderstaand schema is weergegeven.
Bij deze aanpassing komt de vierde dipswitsch achter op de SHARP ook weer naar voren. Die dipswitsch was nutteloos, maar na het aansluiten van een tweede cassetterecorder absoluut niet meer. Deze dipswitsch is bedoeld om de polariteit van de tweede cassetterecorder om te kunnen wisselen.
Het is natuurlijk wel heel erg verstandig om een schakelaar voor beide cassetterecorders te maken, omdat anders van twee recorders tegelijk gelezen of naar twee recorders tegelijk geschreven wordt. Voor zo'n schakelaar is hier echter geen schemaatje afgedrukt, U zult het dus zelf moeten ontwikkelen.





Tekeningen/schema's behorende bij aanpassing 1 (blz. 159)



Tekening/schema behorende bij aanpassing 2 (blz. 160)

HOOFDSTUK 9: ASSEMBLEREN.

Dit is het grootste hoofdstuk in dit boek. Dit heeft te maken met het feit dat machinetaal de enige mogelijkheid is om meer te beraiken met Uw SHARP dan U voorheen gekund heeft. Dit hoofdstuk zal alle hoofdstukken waar machinetaal in voor komt, behandelen op volgorde van hoofdstuk. We beginnen dus bij hoofdstuk 1 en eindigen bij hoofdstuk 8. We zullen proberen zo veel mogelijk machinetaalprogramma's te behandelen. Ze kunnen echter niet allemaal behandeld worden, omdat sommige programma's daarvoor te lang zijn.
In dit hoofdstuk zal de assemblerlisting van de meeste machinetaal-programma's getoond worden en vervolgens zal een uitgebreide uitleg volgen.
In het tweede deel van dit hoofdstuk zal een deel van de assembler-instructies behandeld worden, ook al zijn daar al genoeg boeken voor. De specifieke SHARP-instructies zullen ook uitgebreid behandeld worden en tevens zult U hier en daar een tabel tegen komen.
In hoofdstuk 2 en 3 staan ook een aantal machinetaalprogramma's met de bijbehorende assemblerlisting. De uitleg bij die programma's is niet bepaald uitgebreid. In dit hoofdstuk zal die uitleg uitgebreider zijn dan in de hoofdstukken 2 en 3.
In dit boek is ook al eerder gezegd dat in dit hoofdstuk uitgelegd zal worden hoe U zelf een BASIC-instructie aan de BASIC toe kunt voegen. Van één van de programma's uit hoofdstuk 1 zal uitgelegd worden hoe dat precies in elkaar zit. Daarmee zult U nog niet direct een BASIC-instructie aan de BASIC toe kunnen voegen en dat is vaak ook niet nodig, omdat een gewoon machinetaalprogramma vaak net zo gemakkelijk te gebruiken is als een extra BASIC-instructie. Voor de BASIC zelf is het ook beter dat U gewoon kleine machinetaalprogrammaatjes gebruikt en die aanroept met USR, want met extra BASIC-instructies gaat U zitten te knoeien in de originele BASIC en moet U ook een andere instructie inleveren en in het ergste geval verliest U ook nog geheugen.
In dit hoofdstuk vindt U een tabel waarin precies staat naar welk adres in cassette- en QD-BASIC gesprongen wordt, wanneer U een bepaalde instructie gebruikt.

9.1: Assemblerlistings van programma's in dit boek.

Assemblerlisting van het programma BOR. (Hfdst.1.3 - blz.22-23.)

```
59E9 DEFB 42H                    59EB DEFB D2H
59EA DEFB 4FH

55B0 CALL 84DAH                  55B7 OUT (C),A
55B3 LD A,E                      55B9 RET
55B4 LD BC,06CFH

5C9D DEFB B0H                    5C9E DEFB 55H
```

De assemblerlisting die U hier ziet is de assemblerlisting van de cassette-versie. De QD-versie ziet er als volgt uit:

```
59E9 DEFB 42H                    59EB DEFB D2H
59EA DEFB 4FH
```

```
5599 CALL 8551H                 55A0 OUT (C),A
559C LD A,E                     55A2 RET
559D LD BC,06CFH

5C9D DEFB 99H                   5C9E DEFB 55H
```

U ziet dat dit niet een assemblerlisting is die U letterlijk in een assembler in kunt typen. In een assembler bepaald U met de instructie ORG vanaf welk adres de daaropvolgende instructies komen te staan.
Hoe U precies met een assembler om moet gaan zal hier verder niet behandeld worden, omdat daarvoor handleidingen bestaan. De assemblerlistings die hier gegeven worden zijn uitsluitend bedoeld om een goed overzicht te krijgen van een machinetaalprogramma.

Uitleg van het programma BOR:

Op de adressen 59E9 t/m 59EB wordt de tekst BOR neergezet. Bij de code van de letter R wordt 80H opgeteld en die volgt als laatste. Bij de code van de laatste letter van een instructie moet dus 80H opgeteld worden.
De instructie OFF is niet meer te gebuiken. De tekst OFF stond dus eerst op de adressen 59E9 t/m 59EB.
Als OFF gebruikt wordt, wordt normaal naar adres 6364 gesprongen. Dat blijkt o.a. uit de tabel die U na deze uitleg aan zult treffen. U kunt via de BASIC-monitor met behulp van F (Find) opzoeken waar U in het geheugen de waardes 64H en 63H kunt vinden. Wanneer U het volgende in voert:

```
F500060006463
```

zult U een heleboel adressen te zien krijgen. Op deze manier gaat het dus niet.
U gaat opnieuw in de tabel kijken en ziet dan dat de instructie POKE voorafgaat aan de instructie OFF. Dit betekent dat het adres waar naar toe gesprongen wordt wanneer POKE gebruikt wordt voor het adres staat waar naar toe gesprongen wordt wanneer OFF gebruikt wordt. Uit de tabel volgt dat het adres waar naar toe gesprongen wordt, wanneer POKE gebruikt wordt, het adres 68A1 is.
U zoekt nu op waar in het geheugen de codes A1 en 68 staan en dat doet U als volgt:

```
F50006000A168
```

en U krijgt te zien dat deze waardes op de adressen 5C9B en 5C9C staan. Dat betekent dat het adres voor OFF op de adressen 5C9D en 5C9E staat.
U ziet dat het adres waarnaar gesprongen wordt altijd in twee codes opgeslagen is in het gebied vanaf adres 5C5B t/m 5E16. Door deze twee codes te veranderen kunt U een instructie iets anders laten doen dan normaal.
Wanneer een instructie naar 6364 springt, betekent dat normaal gesproken dat er een Syntax error gegeven wordt.
Bij de BOR-instructie zetten we de waardes B0H en 55H neer en wordt er dus naar adres 55B0 gesprongen wanneer de BOR-instructie gebruikt wordt. Wat er vanaf adres 55B0 precies gebeurt zal op de volgende bladzijde besproken worden en daarna zal de tabel volgen.

Voordat we verder gaan met de uitleg, willen wij U er nog op wijzen dat we nu dus de cassette-versie aan het bespreken zijn.
Wat gebeurt er nu precies vanaf adres 55B0?

Allereerst wordt er een CALL gegeven die kijkt of U na de instructie BOR ook een getal of een variabele ingevoerd heeft. Is dit niet het geval dan wordt er een Syntax error gegeven. De waarde die eventueel achter de instructie BOR staat wordt in het adres E geladen en na terugkeer van deze CALL wordt het A-register met de waarde van het E-register geladen.
Vervolgens wordt BC geladen met de waarde 06CFH. Bij het aansturen van de border moet register BC altijd met deze waarde geladen zijn. Register B kan namelijk ook een andere waarde dan 06H hebben, maar dan gebeurt er iets anders. Via de poort CFH wordt de waarde van de kleur naar de border gestuurd. De border krijgt dus de kleur die door het A-register bepaald wordt.

U ziet dat voor het toevoegen van een eenvoudige instructie vaak nogal wat denkwerk nodig is. Dit was dan nog een heel eenvoudige instructie, maar er zijn ook veel ingewikkeldere instructies en die kunt U beter via een klein machinetaalprogrammaatje laten werken.

Nu volgt de tabel die aangeeft naar welk adres in de cassette-BASIC en de QD-BASIC gesprongen wordt bij gebruik van een bepaalde instructie. Iedere instructie heeft zijn eigen TOKEN, dat is 1 getal of dat zijn 2 getallen die een bepaalde instructie aangeven. In het geheugen komt een TOKEN te staan en niet de volledige instructie, zodat iedere instructie slechts 1 of 2 bytes nodig heeft. Een aantal tokens zijn nog vrij en zijn dus eigenlijk ook te gebruiken om een extra instructie aan de BASIC toe te voegen, maar dat is veel ingewikkelder dan een nieuwe instructie over een oude instructie heen schrijven.

Instr.:	TOKEN:	CASS:	QD:	Instr.:	TOKEN:	CASS:	QD:
GOTO	128	69AF	69AF	READ	149	6DCD	6DCD
GOSUB	129	692A	692A	DIM	150	8D55	8DCC
vrij	130	6364	6364	REM	151	67FE	67FE
RUN	131	606E	606E	END	152	60B9	60B9
RETURN	132	68F9	68F9	STOP	153	6468	6468
RESTORE	133	6E9E	6E9E	CONT	154	6499	6499
RESUME	134	64B6	64B6	CLS	155	78FB	7972
LIST	135	6A9D	6A9D	vrij	156	6364	6364
vrij	136	6364	6364	ON	157	6952	6952
DELETE	137	620C	620C	LET	158	650D	650D
RENUM	138	62AB	62AB	NEW	159	618E	618E
AUTO	139	60CA	60CA	POKE	160	68A1	68A1
EDIT	140	6125	6125	OFF	161	6364	6364
FOR	141	6602	6602	PMODE	162	9F9E	A015
NEXT	142	6699	6699	PSKIP	163	A003	A07A
PRINT	143	6C9F	6C9F	PLOT	164	A3B1	A428
vrij	144	6364	6364	PLINE	165	A045	A0BC
INPUT	145	6DD4	6DD4	RLINE	166	A048	A0BF
vrij	146	6364	6364	PMOVE	167	A04B	A0C2
IF	147	69C2	69C2	RMOVE	168	A04E	A0C5
DATA	148	67FE	67FE	TRON	169	61CC	61CC

Instr.:	TOKEN:	CASS:	QD:	Instr.:	TOKEN:	CASS:	QD:
TROFF	170	61DF	61DF	vrij	223	6364	6364
INP	171	681F	681F	vrij	254 128	6364	6364
DEFAULT	172	6F41	6F41	CSET	254 129	6364	6364
GET	173	686C	686C	CRESET	254 130	6364	6364
PCOLOR	174	A0B3	A12A	CCOLOR	254 131	6364	6364
PHOME	175	A0E8	A15F	vrij	254 132	6364	6364
HSET	176	A0EB	A162	vrij	254 133	6364	6364
GPRINT	177	A0F6	A16D	vrij	254 134	6364	6364
KEY	178	6B84	6B84	vrij	254 135	6364	6364
AXIS	179	A15E	A1D5	vrij	254 136	6364	6364
LOAD	180	6FFD	6FFD	vrij	254 137	6364	6364
SAVE	181	733F	73C0	SOUND	254 138	9B35	9BAC
MERGE	182	7068	7068	vrij	254 139	6364	6364
CHAIN	183	7053	7053	NOISE	254 140	9B6C	9BE3
CONSOLE	184	6A61	6A61	BEEP	254 141	6A56	6A56
SEARCH	185	6A9B	6A9B	vrij	254 142	6364	6364
OUT	186	680A	680A	vrij	254 143	6364	6364
PCIRCLE	187	A19E	A215	COLOR	254 144	760B	7675
PTEST	188	A36C	A3E3	vrij	254 145	6364	6364
PAGE	189	A37A	A3F1	SET	254 146	7528	7592
WAIT	190	67EE	67EE	RESET	254 147	7529	7593
SWAP	191	7425	74A6	LINE	254 148	7537	75A1
vrij	192	6364	6364	BLINE	254 149	7538	75A2
ERROR	193	63F8	63F8	PAL	254 150	75AF	7619
ELSE	194	69E4	69E4	CIRCLE	254 151	76DE	774C
USR	195	67A4	67A4	BOX	254 152	75DE	7648
BYE	196	6A59	6A59	PAINT	254 153	768F	76FD
vrij	197	6364	6364	POSITION	254 154	758B	75F5
vrij	198	6364	6364	PATTERN	254 155	755A	75C4
DEF	199	6C31	6C31	HCOPY	254 156	78F1	7968
vrij	200	6364	6364	vrij	254 157	6364	6364
vrij	201	6364	6364	vrij	254 158	6364	6364
LABEL	202	67FE	67FE	vrij	254 159	6364	6364
vrij	203	6364	6364	SYMBOL	254 160	78A1	790F
vrij	204	6364	6364	vrij	254 161	6364	6364
vrij	205	6364	6364	MUSIC	254 162	9B6F	9BE6
WOPEN	206	6F59	6F59	TEMPO	254 163	9B62	9BD9
CLOSE	207	6F8A	6F8A	CURSOR	254 164	683B	683B
ROPEN	208	6F56	6F56	VERIFY	254 165	7328	73A9
XOPEN	209	6F5C	6F5C	CLR	254 166	619B	619B
vrij	210	6364	6364	LIMIT	254 167	68B5	68B5
vrij	211	6364	6364	vrij	254 168	6364	6364
vrij	212	6364	6364	vrij	254 169	6364	6364
DIR	213	73A4	742A	vrij	254 170	6364	6364
vrij	214	6364	6364	vrij	254 171	6364	6364
vrij	215	6364	6364	vrij	254 172	6364	6364
RENAME	216	73CA	744B	vrij	254 173	6364	6364
KILL	217	6F8B	6F8B	BOOT	254 174	6A95	6A95
LOCK	218	7397	7418	INT	255 128	954B	95C2
UNLOCK	219	7395	7416	ABS	255 129	9546	95BD
INIT	220	6F47	6F47	SIN	255 130	9692	9709
vrij	221	6364	6364	COS	255 131	9682	96F9
vrij	222	6364	6364	TAN	255 132	9771	97E8

Instr.:	TOKEN:	CASS:	QD:	Instr.:	TOKEN:	CASS:	QD:
LN	255 133	99D8	9A4F	vrij	255 170	6364	6364
EXP	255 134	98B4	992B	ASC	255 171	891B	8992
SQR	255 135	95A8	961F	LEN	255 172	8925	899C
RND	255 136	986F	98E6	VAL	255 173	892D	89A4
PEEK	255 137	9863	98DA	vrij	255 174	6364	6364
ATN	255 138	95C6	963D	vrij	255 175	6364	6364
SGN	255 139	9829	98A0	vrij	255 176	6364	6364
LOG	255 140	99CC	9A43	vrij	255 177	6364	6364
FRAC	255 141	7B49	7BC0	vrij	255 178	6364	6364
PAI	255 142	9841	98B8	ERN	255 179	8803	887A
RAD	255 143	983C	98B3	ERL	255 180	8811	8888
vrij	255 144	6364	6364	SIZE	255 181	87DB	8852
vrij	255 145	6364	6364	CSRH	255 182	87ED	8864
vrij	255 146	6364	6364	CSRV	255 183	87F2	8869
vrij	255 147	6364	6364	POSH	255 184	87F7	886E
vrij	255 148	6364	6364	POSV	255 185	87FD	8874
vrij	255 149	6364	6364	LEFT$	255 186	893F	89B6
vrij	255 150	6364	6364	RIGHT$	255 187	895C	89D3
vrij	255 151	6364	6364	MID$	255 188	8979	89F0
vrij	255 152	6364	6364	vrij	255 189	6364	6364
vrij	255 153	6364	6364	vrij	255 190	6364	6364
vrij	255 154	6364	6364	vrij	255 191	6364	6364
vrij	255 155	6364	6364	vrij	255 192	6364	6364
STICK	255 156	8FE9	9060	vrij	255 193	6364	6364
STRIG	255 157	9008	907F	vrij	255 194	6364	6364
vrij	255 158	6364	6364	vrij	255 195	6364	6364
vrij	255 159	6364	6364	TI$	255 196	89CF	8A46
CHR$	255 160	88BA	889F	POINT	255 197	88F5	896C
STR$	255 161	8879	8931	EOF	255 198	88C8	893F
HEX$	255 162	88F0	88F0	FN	255 199	8F48	8FBF
vrij	255 163	6364	6364	vrij	255 200	6364	6364
vrij	255 164	6364	6364	vrij	255 201	6364	6364
vrij	255 165	6364	6364	vrij	255 202	6364	6364
vrij	255 166	6364	6364	vrij	255 203	6364	6364
vrij	255 167	6364	6364	vrij	255 204	6364	6364
SPACE$	255 168	889C	8913	vrij	255 205	6364	6364
vrij	255 169	6364	6364	vrij	255 206	6364	6364

Assemblerlisting van het programma LINKS-SCROLLEN (Hfdst.2.1 - blz.30.)

```
1200 START:  LD HL,D000H
1203 LD DE,D001H
1206 LD C,19H
1208 BEGIN:  LD A,(HL)
1209 EX AF,AF'
120A LD B,27H
120C SCROLL:  LD A,(DE)
120D LD (HL),A
120E INC HL
120F INC DE
1210 DJNZ SCROLL
1212 EX AF,AF'
1213 LD (HL),A
1214 INC HL
1215 INC DE
1216 DEC C
1217 JR NZ,BEGIN
1219 LD C,20H
121B VERTR:  DJNZ VERTR
121D DEC C
121E JR NZ,VERTR
1220 JP START
```

Uitleg van het programma LINKS-SCROLLEN:

Dit is een tamelijk eenvoudig programma, er zal dus ook niet veel uitleg nodig zijn.
Het eerste wat U moet weten is dat in de 700-mode, waarin we hier werken, het beeldscherm opgeslagen is vanaf adres D000H. Het bovenste linker hokje komt overeen met het adres D000H. Deze waarde wordt eerst in het register HL gezet. In DE komt de waarde van het tweede hokje.
De inhoud van het adres D000H wordt eerst ingelezen en bewaard in het schaduwregister AF'. Daarna wordt 39 keer de inhoud van het DE-register uitgelezen en deze wordt op het adres gezet dat door HL aangegeven wordt. Elke keer schuiven zowel het HL- als het DE-register één plaats naar rechts. Er wordt dus precies één regel één hokje naar links gescrolld. Het HL-register bevat nu de waarde van het laatste hokje aan het eind van een regel en daarop wordt de waarde van het schaduwregister AF' gezet. Het teken dat eerst aan het begin van een regel stond staat nu dus aan het eind van een regel. Op deze manier worden alle 25 regels afgewerkt en het beeldscherm zal dus nooit verdwijnen, omdat het begin altijd aan het eind weer terug komt.
Aan het eind wordt ook een vertraging doorlopen om het scrollen niet te snel te laten verlopen, want anders is het haast niet meer bij te houden.
Hopelijk is deze uitleg voldoende om zowel het scrollen naar links als naar rechts te begrijpen.

Assemblerlisting van 't programma BOVEN-SCROLLEN. (Hfdst.2.1 - blz.30.)

```
1200 START:  LD HL,CFD8H
1203 LD DE,D000H
1206 LD C,1AH
1208 BEGIN:  LD B,27H
120A SCROLL:  LD A,(DE)
120B LD (HL),A
120C INC HL
120D INC DE
120E DJNZ SCROLL
1210 DEC C
1211 JR NZ,BEGIN
1213 LD DE,CFD8H
1216 LD HL,D3C0H
1219 LD B,28H
121B ONDER:  LD A,(DE)
121C LD (HL),A
121D INC HL
121E INC DE
121F DJNZ ONDER
1221 LD C,20H
1223 VERTR:  DJNZ VERTR
1225 DEC C
1226 JR NZ,VERTR
1228 CALL 001BH
122B RET NZ
122C JP START
```

Uitleg van het programma BOVEN-SCROLLEN:

Dit programma verschilt eigenlijk niet zoveel van het programma LINKS-SCROLLEN. In dit programma worden alle 25 regels één regel omhoog gescrolld en in het begin komt de bovenste regel in een gebied dat niet meer tot het beeldscherm behoort. Aan het eind wordt deze regel er onderaan weer bij gezet. Voor de rest is het programma eigenlijk precies hetzelfde als het programma LINKS-SCROLLEN, op één ding na en dat is het afvragen van het toetsenbord met CALL 001BH. Mocht er een toets ingedrukt worden dan wordt het A-register geladen met de waarde van die toets en wordt vervolgens teruggesprongen in de ROM-monitor, omdat A niet meer gelijk is aan nul. Wanneer het programma van cass. of QD gestart wordt, zal het programma niet terugspringen in de ROM-mon.

Waarom wordt er dan niet teruggesprongen in de ROM-monitor?
Dit heeft te maken met het feit dat het programma vanuit een routine die een programma laadt van cass. of QD, opgestart wordt en niet vanuit de ROM-monitor. Bij een RET-opdracht zal er dus teruggesprongen worden naar die routine en niet naar de ROM-monitor. Dit kan opgelost worden door met een CALL het programma automatisch in de ROM-monitor te laten springen en dat is CALL 00ADH. In hoofdstuk 2 wordt deze CALL iets uitvoeriger uitgelegd.
Het programma BENEDEN-SCROLLEN is bijna gelijk aan het programma BOVEN-SCROLLEN. Een aparte uitleg van dat programma is dus niet nodig.

Assemblerlisting van het programma 2 CH.SETS. (Hfdst.2.1 - blz.31.)

```
1200 LD A,C6H
1202 CALL 0DDCH
1205 LD DE,0800H
1208 LD HL,D000H
120B LD B,FFH
120D PA:  LD (HL),B
120E ADD HL,DE
120F LD (HL),70H
1211 SBC HL,DE
1213 INC HL
1214 DJNZ PA
1216 LD HL,D118H
1219 LD B,FFH
121B PB:  LD (HL),B
121C ADD HL,DE
121D LD (HL),F0H
121F SBC HL,DE
1221 INC HL
1222 DJNZ PB
1224 RET
```

Uitleg van het programma 2 CH.SETS:

Met CALL 0DDCH wordt eerst het beeldscherm schoongemaakt. In hoofdstuk 2 wordt deze CALL uitvoerig besproken.
Voordat we verder gaan nog even het volgende: Het gewone beeldscherm begint bij adres D000H en het kleurengedeelte begint bij D800H. De kleuren 00H t/m 7FH geven een character in de eerste characterset weer en de kleuren 80H t/m FFH geven een character in de tweede characterset weer.
Er wordt eerst een rijtje met characters neergezet die gelijk een kleur krijgen beneden de waarde van 7FH. Deze characters zijn dus gewoon in de eerste characterset te zien. Vervolgens wordt daaronder nog een rijtje met characters neergezet die allemaal een kleur krijgen boven de waarde van 80H. Deze characters zijn dus allemaal in de tweede characterset te zien.
Het programma is verder heel gemakkelijk te begrijpen. Een verdere uitleg is dus waarschijnlijk niet nodig.

Assemblerlisting van het programma BORDER-JOY. (Hfdst.2.1 - blz.31-32.)

```
2000 START:  LD A,5FH
2002 LD BC,06CFH
2005 OUT (C),A
2007 DEC A
2008 LD (START+1),A
200B LD HL,0320H
200E VERTR:  DEC L
200F JR NZ,VERTR
2011 DEC H
2012 JR NZ,VERTR
2014 IN A,F0H
2016 CP FEH
2018 CALL Z,SNEL
201B CP FDH
201D CALL Z,LANGZ
2020 CP EFH
2022 RET Z
2023 NOP
```

```
2024 JP START
2027 SNEL:  LD HL,(200CH)
202A DEC L
202B SA:  LD (200CH),HL
202E RET
202F LANGZ:  LD HL,(200CH)
2032 INC L
2033 JP SA
```

Uitleg van het programma BORDER-JOY:

De border wordt in het begin geladen met de kleur die door de waarde van het A-register bepaald wordt. De waarde van het A-register wordt elke keer met 1 verlaagd. De kleur is dus elke keer anders.
Na het bepalen van de kleur wordt een vertraging doorlopen. Deze vertraging is in eerste instantie zo ingesteld dat U de border als een kleurenbalk ziet die heel langzaam beweegt.
U ziet een kleurenbalk omdat de border heel erg snel van kleur verandert, sneller dan het menselijk oog kan waarnemen. De televisie kan de frequentie waarmee de border verandert wordt ook niet aan, zodoende krijgt U elke keer slechts een deel van een kleur te zien.
Door met de joystick, die op poort 1 aangesloten moet worden, naar boven of beneden te bewegen kunt U de vertraging kleiner of groter maken, daardoor zal de mooie kleurenbalk op een gegeven moment vervagen en zult U een flikkerende border krijgen.
Omdat de vertraging na 256 bewegingen in dezelfde richting, of dit nu vooruit of achteruit is, altijd weer op dezelfde vertraging terecht komt, zult U elke keer ongeveer in dezelfde situatie terecht komen.
Om de kleurenbalk in precies dezelfde stand terug te krijgen is niet gemakkelijk, omdat bij een kleine beweging van de joystick de vertraging gelijk met sprongen van 5 of zo groter of kleiner wordt.
Er is ook een stukje ingebouwd dat er voor zorgt dat er in de ROM-monitor terug gesprongen wordt wanneer er op de vuurknop gedrukt wordt.
De instructies die in dit programma gebruikt worden zijn van algemene aard en zijn niet moeilijk te begrijpen. Een verdere uitleg zal dus hier waarschijnlijk ook niet nodig zijn.

Assemblerlisting van het programma RANDOM. (Hfdst.2.1 - blz.32.)

```
1200 START:  LD HL,D3BFH
1203 LD A,(E005H)
1206 RLCA
1207 LD C,A
1208 LD D,09H
120A BEGIN:  LD B,26H
120C RANDOM:  CP 00H
120E JP Z,PRINT
1211 DEC A
1212 DJNZ RANDOM
1214 DEC D
1215 JP NZ,ROUT
1218 POS:  INC HL
1219 DEC C
121A JR NZ,POS
121C LD (HL),6BH
121E INC HL
121F INC HL
1220 INC HL
1221 LD (HL),6BH
1223 LD A,C0H
1225 CALL 0DDCH
1228 LD E,20H
122A VERTR:  DJNZ VERTR
122C DEC C
122D JR NZ,VERTR
122F JP START
1232 ROUT:  LD C,A
1233 JP BEGIN
```

Uitleg van het programma RANDOM:

Zie verder op de volgende bladzijde.

Allereerst wordt het HL-register geladen met de waarde van het laatste hokje op de éénnalaatste regel. Vervolgens wordt het A-register geladen met een 'willekeurig' getal. Dit getal wordt geroteerd om de nauwkeurigheid iets meer te benaderen.
In het C-register wordt de waarde van het willekeurige getal bewaard.
Daarna wordt het willekeurige getal, dat tussen 0 en 255 ligt, gereduceerd tot een willekeurig getal tussen 0 en 38, dat wordt op de volgende manier gedaan:
Er wordt gekeken of het getal tussen 0 en 38 ligt. Is dat niet het geval dan wordt er 38 van het getal afgetrokken en dat gaat net zo lang door tot het getal tussen 0 en 38 ligt.
Ligt het getal éénmaal tussen 0 en 38 dan wordt het getal bij het HL-register opgeteld en op die plaats komt een sterretje te staan en drie plaatsen naar rechts komt er nog één te staan.
Hierna wordt het hele beeldscherm één regel omhoog gescrolld en wordt er een vertraging doorlopen. Daarna begint alles weer van voren af aan.

Assemblerlisting van het programma SCORE. (Hfdst.2.1 - blz.33.)

```
2000 LD A,C6H
2002 CALL 0DDCH
2005 LD HL,D1F2H
2008 LD B,04H
200A NULLEN:  LD (HL),20H
200C INC HL
200D DJNZ NULLEN
200F BEGIN:  LD B,01H
2011 VERTR:  DEC C
2012 JR NZ,VERTR
2014 DJNZ VERTR
2016 LD HL,D1F5H
2019 LD A,(HL)
201A INC A
201B CP 2AH
201D JP Z,SA
2020 PRINT:  LD (HL),A
2021 JP BEGIN
2024 SA:  LD (HL),20H
2026 DEC HL
2027 LD A,(HL)
2028 INC A
2029 CP 2AH
202B JP Z,SB
202E JP PRINT
2031 SB:  LD (HL),20H
2033 DEC HL
2034 LD A,(HL)
2035 INC A
2036 CP 2AH
2038 JP Z,SC
203B JP PRINT
203E SC:  LD (HL),20H
2040 DEC HL
2041 LD A,(HL)
2042 INC A
2043 CP 2AH
2045 JP Z,SD
2048 JP PRINT
204B SD:  LD (HL),20H
204D DEC HL
204E LD (HL),21H
2050 RET
```

Uitleg van het programma SCORE:

Dit programma zit tamelijk eenvoudig in elkaar. Allereerst wordt het beeldscherm schoongemaakt en vervolgens worden er vier nullen in het midden van het scherm gezet. Daarna wordt een vertraging doorlopen om de score niet te snel op te laten lopen, want zonder vertraging gaat het echt veel te snel.
Dan volgt de eigenlijke score-routine. Allereerst wordt het laatste getal ingelezen en wordt deze met 1 verhoogd. Is dit getal groter dan 9 (Is de waarde 2AH?) dan wordt naar het volgende getal links van dit getal gekeken. Bij dat getal gebeurt precies hetzelfde als bij het eerste getal en zo gaat dat bij alle getallen.

Is het meest linker getal ook groter dan 9 dan wordt alles op 0 gezet en wordt aan het begin een 1 neergezet. Er is dan dus tot 10000 geteld.
Nog even iets dat bijna vergeten was: Als een getal groter dan 9 is, wordt dat getal weer op 0 gezet.
De waardes van 0 t/m 9 zien er in codes als volgt uit:
0 = 20H
1 = 21H
en dat gaat zo door tot en met
9 = 29H

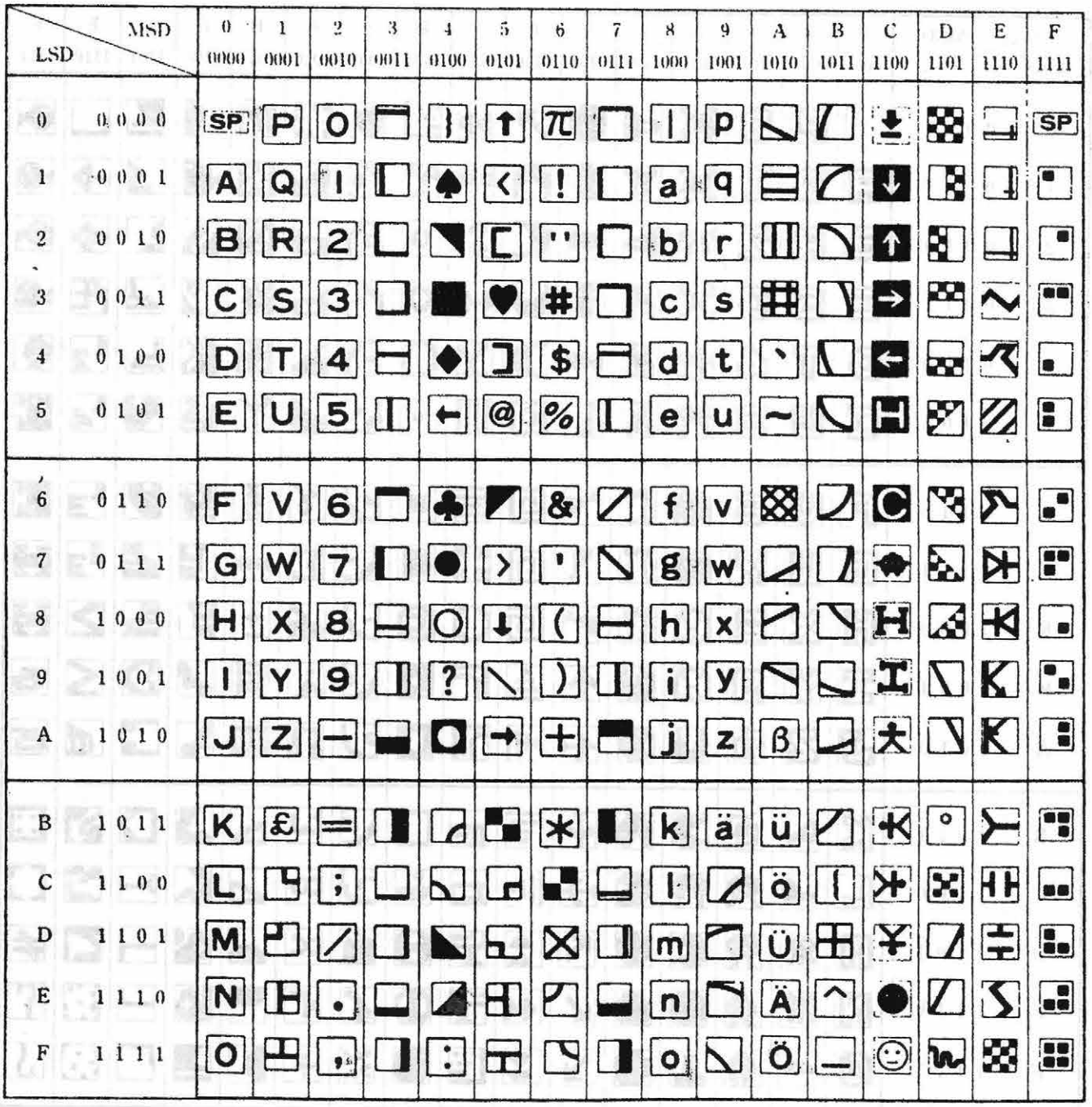
Assemblerlisting van het programma CH-CHANGE. (Hfdst.2.1 - blz.33-34.)

```
A000 OUT (E4H),A
A002 IN A,(E0H)
A004 LD HL,C000H
A007 LD BC,1000H
A00A LD DE,A800H
A00D LDIR
A00F IN A,(E1H)
A011 LD HL,A800H
A014 LD (HL),FFH
A016 INC HL
A017 LD (HL),FFH
A019 INC HL
A01A LD (HL),FFH
A01C INC HL
A01D LD (HL),FFH
A01F INC HL
A020 LD (HL),FFH
A022 INC HL
A023 LD (HL),FFH
A025 INC HL
A026 LD (HL),FFH
A028 INC HL
A029 LD (HL),00H
A02B OUT (E4H),A
A02D IN A,(E0H)
A02F LD HL,A800H
A032 LD BC,1000H
A035 LD DE,C000H
A038 LDIR
A03A IN A,(E1H)
A03C RET
```

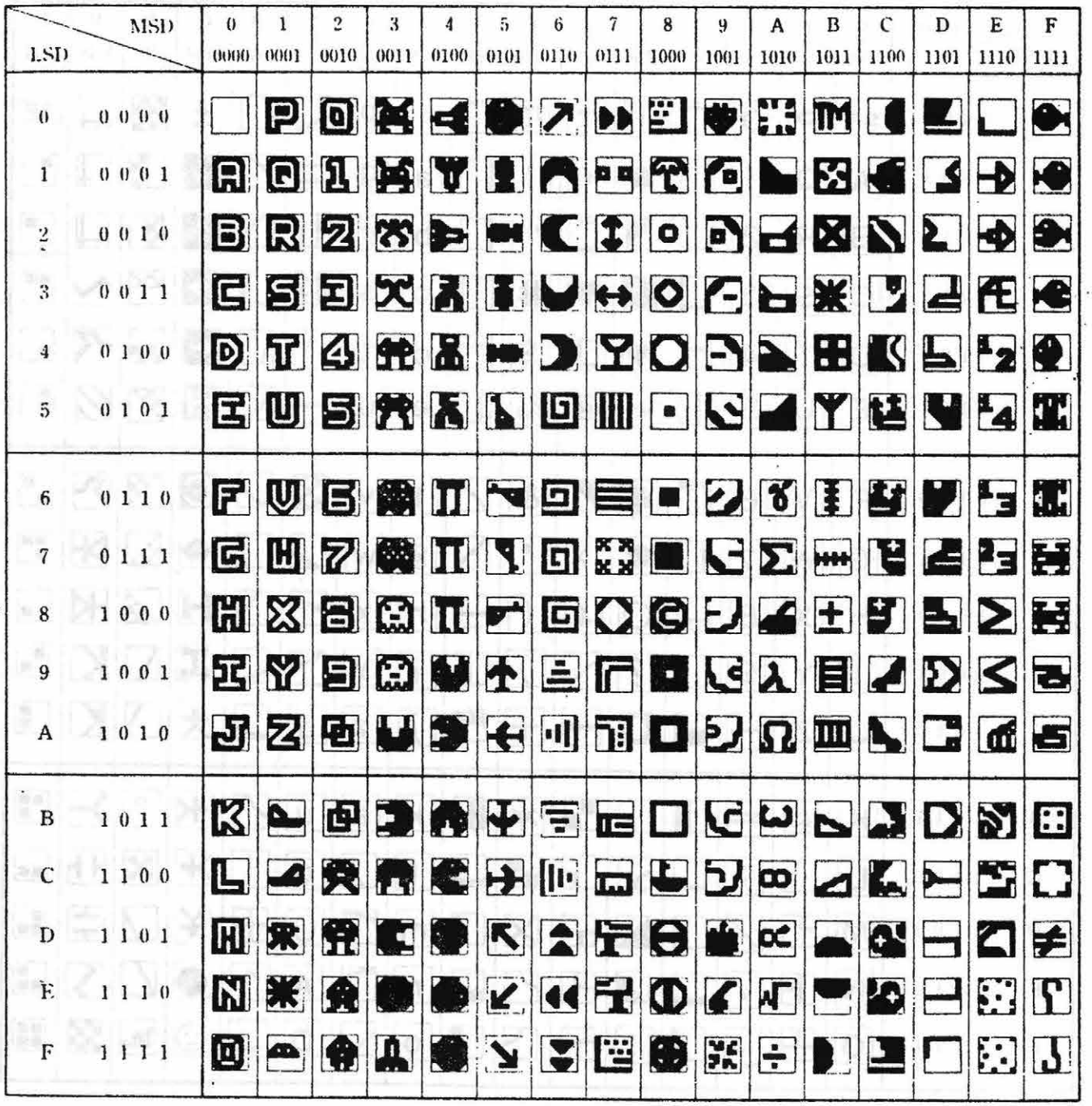
Uitleg van het programma CH-CHANGE:

Dit programma zit misschien niet zo heel erg moeilijk in elkaar, maar hoe weet U nu waar welke tekens in het geheugen staan? Er worden wel termen gebruikt als de ASC-II code, maar weet U wel wat dat is?
Om U niet met ingewikkelde verhalen op te schepen, staan op de volgende twee bladzijden tabellen van de tekens en hun bijbehorende waarden afgedrukt en een korte uitleg over berekening van de waardes. Zowel voor de eerste als de tweede characterset staat er een tabel.
Nu volgt eerst nog een kleine uitleg van het programma.

Allereerst wordt het geheugen zo ingedeeld dat de data van de verschillende tekens vanaf adres C000H in het geheugen komen te staan. Daarna wordt het gebied van C000H t/m D000H naar het gebied van A800H t/m B800H getransformeerd. De data van de tekens komen dus vanaf adres A800H in het geheugen te staan.
Daarna gaan we iets veranderen in de data op de adressen A800H t/m A807H en daar zijn de data van het spatieteken opgeslagen.
Zijn de data van de spatie verandert dan worden alle data van A800H t/m B800H weer naar het gebied van C000H t/m D000H getransformeerd. De data van de spatie zijn nu dus in werkelijkheid veranderd.
Wat doet de instructie LDIR? Deze instructie transformeert een blok ten grootte van de waarde van het BC-register van het gebied beginnend bij het adres in DE, naar het gebied beginnend bij het adres in HL.



Op deze en de volgende bladzijde staan de tabellen van de ASC-II waardes van alle 512 tekens die de SHARP kent.
In de horizontale richting staan de 16-tallen.
In de verticale richting staan de gewone getallen die bij de 16-tallen opgeteld moeten worden.
Gemakkelijker gezegd: De hexadecimale waarde van een teken bestaat uit twee 'getallen'.
Het eerste 'getal' van de hex. waarde staat in de horizontale richting.
Het tweede 'getal' van de hex. waarde staat in de verticale richting.
(Zie verder onderaan de volgende bladzijde.)



Tevens kan de binaire waarde afgelezen worden.
Een voorbeeld:

Het teken '3' heeft de hexadecimale waarde 23 en de binaire waarde 00100011.
De 2 uit de waarde 23 kunt U in de horizontale richting aflezen en de 3 uit de waarde 23 kunt U in de verticale richting aflezen. Zo gaat dat ook met de binaire waarde.
Deze informatie geldt natuurlijk voor beide tabellen. Hoe U de tweede characterset kunt krijgen is al eerder uitgelegd.

Assemblerlisting van het programma HR-SCROLLEN. (Hfdst.2.1 - blz.34.)

```
6000 IN A,(E0H)
6002 LD A,00H
6004 OUT (CEH),A
6006 LD A,00H
6008 OUT (F0H),A
600A LD A,03H
600C OUT (CCH),A
600E LD HL,8000H
6011 LD BC,2000H
6014 CLS:  XOR A
6015 LD (HL),A
6016 INC HL
6017 DEC BC
6018 LD A,B
6019 OR C
601A JP NZ,CLS
601D LD A,32H
601F OUT (F0H),A
6021 LD A,C3H
6023 OUT (F0H),A
6025 LD HL,8FF0H
6028 LD B,28H
602A LIJN:  LD (HL),FFH
602C INC HL
602D DJNZ LIJN
602F BEGIN:  LD HL,7FD8H
6032 LD DE,8000H
6035 LD BC,1F40H
6038 SCROLL:  LD A,(DE)
6039 LD (HL),A
603A INC HL
603B INC DE
603C DEC BC
603D LD A,B
603E OR C
603F JP NZ,SCROLL
6042 LD HL,9F18H
6045 LD DE,7FD8H
6048 LD B,28H
604A BOON:  LD A,(DE)
604B LD (HL),A
604C INC HL
604D INC DE
604E DJNZ BOON
6050 JP BEGIN
```

Uitleg van het programma HR-SCROLLEN:

Aangezien verderop in dit hoofdstuk nog genoeg over de VIDEO-RAM verteld wordt, zal dit programma niet al te uitvoerig behandeld worden. In het begin wordt er eerst gebankswitscht. Vervolgens wordt het VIDEO-RAM geheugen ingedeeld op 320*200 met 4 kleuren.
Via poort F0H kunnen de 4 paletten van kleur veranderd worden. Het eerste getal in het A-register geeft de palet aan en het tweede getal geeft de kleur aan. Wanneer het A-register bijvoorbeeld de waarde 37H bevat, staat de 3 voor het palet en de 7 (licht-grijs) voor de nieuwe kleur van dat palet.
De poort CCH bepaald in welke kleur er geschreven wordt. Bij bespreking van programma's uit hoofdstuk 3 zal dit nog heel uitvoerig aan de orde komen.
Daarna wordt over het hele beeldscherm alles van het scherm gehaald. Het beeldscherm wordt dus schoongemaakt (CLS). Vervolgens wordt er een rood lijntje in het midden van het scherm getekend en daarna wordt dat lijntje omhoog gescrolld. Is het lijntje boven verdwenen dan komt het er onderaan weer bij.

Assemblerlisting van het programma RANDSCROLL. (Hfdst.2.1 - blz.35.)

```
A000 DEFB BFH
A001 DEFB A1H
A002 DEFB ADH
A003 DEFB A5H
A004 DEFB A5H
A005 DEFB BDH
A006 DEFB 81H
A007 DEFB FFH
A008 START:  LD A,C6H
A00A CALL 0DDCH
A00D OUT (E4H),A
A00F IN A,(E0H)
```

```
A011 LD HL,C000H
A014 LD BC,1000H
A017 LD DE,A800H
A01A LDIR
A01C IN A,(E1H)
A01E LD HL,AEE0H
A021 LD C,04H
A023 VIER:  LD DE,A000H
A026 LD B,08H
A028 CHCH:  LD A,(DE)
A029 LD (HL),A
A02A INC HL
A02B INC DE
A02C DJNZ CHCH
A02E DEC C
A02F JP NZ,VIER
A032 LD HL,D000H
A035 LD B,28H
A037 BRIJ:  LD (HL),DFH
A039 INC HL
A03A DJNZ BRIJ
A03C LD HL,D3C0H
A03F LD B,28H
A041 ORIJ:  LD (HL),DEH
A043 INC HL
A044 DJNZ ORIJ
A046 LD HL,D028H
A049 LD DE,0027H
A04C LD B,17H
A04E LRRIJ:  LD (HL),DDH
A050 ADD HL,DE
A051 LD (HL),DCH
A053 INC HL
A054 DJNZ LRRIJ
A056 BEGIN:  OUT (E4H),A
A058 IN A,(E0H)
A05A LD HL,AE00H
A05D LD BC,0100H
A060 LD DE,C600H
A063 LDIR
A065 IN A,(E1H)
A067 LD HL,AEF8H
A06A LD B,08H
A06C RROT:  LD A,(HL)
A06D RRCA
A06E LD (HL),A
A06F INC HL
A070 DJNZ RROT
A072 LD HL,AEF0H
A075 LD B,08H
A077 LROT:  LD A,(HL)
A078 RLCA
A079 LD (HL),A
A07A INC HL
A07B DJNZ LROT
A07D LD HL,AEE0H
A080 LD DE,AEE1H
A083 LD C,(HL)
A084 LD B,07H
A086 BROT:  LD A,(DE)
A087 LD (HL),A
 A088 INC DE
A089 INC HL
A08A DJNZ BROT
A08C LD (HL),C
A08D LD HL,AEEFH
A090 LD C,(HL)
A091 LD DE,AEEEH
A094 LD B,07H
A096 OROT:  LD A,(DE)
A097 LD (HL),A
A098 DEC HL
A099 DEC DE
A09A DJNZ OROT
A09C LD (HL),C
A09D LD C,10H
A09F VERTR:  LD B,FFH
A0A1 LOOP:  DJNZ LOOP
A0A3 DEC C
A0A4 JR NZ,VERTR
A0A6 JP BEGIN
```

Uitleg van het programma RANDSCROLL:

Het programma zit in principe heel gemakkelijk in elkaar, er zal dus ook niet veel uitleg nodig zijn.
Dit programma verandert 4 tekens van de normale characterset in een teken waarvan de data opgeslagen liggen op de adressen A000-A007. Door de vier nieuwe tekens, die in het begin dus allemaal gelijk zijn, naar links, rechts, boven en onder te roteren en daarna op dezelfde plaats op het scherm te zetten lijkt het of iedere rij een bepaalde kant op scrollt. In de praktijk wordt een teken alleen opgepakt, geroteerd in een bepaalde richting en daarna weer terug gezet. Het teken komt dus niet van zijn plaats. Toch lijkt het of de tekens opzij, omhoog of omlaag schuiven.

De programma's RANDTEKST en WILL. KLEUREN zijn ook niet al te moeilijk te begrijpen en zullen dus ook niet besproken worden.

Assemblerlisting van het programma SPRITE. (Hfdst.2.2 - blz.40-41.)

```
FC00 ORG FC00H
FC00 LD A,00H
FC02 IN A,(E0H)
FC04 BEGIN:  LD A,01H
FC06 OUT (CDH),A
FC08 OUT (CCH),A
FC0A LD HL,8028H
FC0D IN A,(F0H)
FC0F CP F7H
FC11 CALL Z,RECHTS
FC14 CP FBH
FC16 CALL Z,LINKS
FC19 CP FEH
FC1B CALL Z,BOVEN
FC1E CP FDH
FC20 CALL Z,ONDER
FC23 JP BEGIN
FC26 RECHTS: LD A,(FC01H)
FC29 INC A
FC2A CP 08H
FC2C JP Z,RBYTE
FC2F LD (FC01H),A
FC32 RCALL:  LD DE,0024H
FC35 LD A,01H
FC37 OUT (CDH),A
FC39 LD C,18H
FC3B RSPRITE:  SCF
FC3C CCF
FC3D LD B,04H
FC3F RBEW:  LD A,(HL)
FC40 RLA
FC41 LD (HL),A
FC42 INC HL
FC43 DJNZ RBEW
FC45 ADD HL,DE
FC46 DEC C
FC47 JR NZ,RSPRITE
FC49 RRET:  XOR A
FC4A RET
FC4B LINKS:  LD A,(FC01H)
FC4E DEC A
FC4F CP FFH
FC51 JP Z,LBYTE
FC54 LTERUG:  LD (FC01H),A
FC57 LD DE,0003H
FC5A ADD HL,DE
FC5B LD DE,002CH
FC5E LD A,01H
FC60 OUT (CDH),A
FC62 LD C,18H
FC64 LSPRITE:  SCF
FC65 CCF
FC66 LD B,04H
FC68 LBEW:  LD A,(HL)
FC69 RRA
FC6A LD (HL),A
FC6B DEC HL
FC6C DJNZ LBEW
FC6E ADD HL,DE
FC6F DEC C
FC70 JR NZ,LSPRITE
FC72 LRET:  XOR A
FC73 RET
FC74 BOVEN:  PUSH HL
FC75 EX DE,HL
FC76 PUSH DE
FC77 POP HL
FC78 LD DE,FFD8H
FC7B ADD HL,DE
FC7C LD A,02H
FC7E OUT (CDH),A
FC80 LD A,(HL)
FC81 CP 00H
FC83 JP NZ,BLIJN
FC86 LD HL,(FC0BH)
FC89 POP DE
FC8A LD A,01H
FC8C OUT (CDH),A
FC8E LD C,19H
FC90 BSPRITE:  LD B,04H
FC92 BBEW:  LD A,(DE)
FC93 LD (HL),A
FC94 INC HL
FC95 INC DE
FC96 DJNZ BBEW
FC98 DEC DE
FC99 DEC DE
FC9A DEC DE
FC9B DEC DE
FC9C PUSH DE
FC9D LD DE,004CH
FCA0 ADD HL,DE
FCA1 POP DE
FCA2 EX DE,HL
FCA3 DEC C
FCA4 JR NZ,BSPRITE
FCA6 XOR A
FCA7 RET
```

```
FCA8 BLIJN:   POP DE
FCA9 XOR A
FCAA RET
FCAB ONDER:   LD DE,039BH
FCAE ADD HL,DE
FCAF PUSH HL
FCB0 EX DE,HL
FCB1 POP HL
FCB2 PUSH DE
FCB3 LD DE,0028H
FCB6 ADD HL,DE
FCB7 LD A,02H
FCB9 OUT (CDH),A
FCBB LD A,(HL)
FCBD CP 00H
FCBF JP NZ,BLIJN
FCC2 PUSH HL
FCC3 LD HL,(FC0BH)
FCC6 LD DE,0028H
FCC9 ADD HL,DE
FCCA LD (FC0BH),HL
FCCD POP HL
FCCE LD A,01H
FCD0 OUT (CDH),A
FCD2 POP DE
FCD3 LD C,19H
FCD5 OSPRITE:   LD B,04H
FCD7 OBEW:   LD A,(DE)
FCD8 LD (HL),A
FCD9 DEC HL
FCDA DEC DE
FCDB DJNZ OBEW
FCDD INC DE
FCDE INC DE
FCDF INC DE
FCE0 INC DE
FCE1 PUSH DE
FCE2 LD DE,FFB4H
FCE5 ADD HL,DE
FCE6 POP DE
FCE7 EX DE,HL
FCE8 DEC C
FCE9 JR NZ,OSPRITE
FCEB XOR A
FCEC RET

FD00 ORG FD00H
FD00 RBYTE:   LD DE,039CH
FD03 ADD HL,DE
FD04 LD A,02H
FD06 OUT (CDH),A
FD08 LD A,(HL)
FD09 CP 00H
FD0B JP NZ,RRET
FD0E LD DE,FC64H
FD11 ADD HL,DE
FD12 CALL RCALL
FD15 LD HL,(FC0BH)
FD18 INC HL
FD19 LD (FC0BH),HL
FD1C LD A,(FC01H)
FD1F JP RRET
FD22 LBYTE: DEC HL
FD23 LD A,02H
FD25 OUT (CDH),A
FD27 LD A,(HL)
FD28 CP 00H
FD2A JP NZ,LRET
FD2D LD HL,(FC0BH)
FD30 LD A,07H
FD32 JP LTERUG
```

Uitleg van het programma SPRITE:

Dit is een behoorlijk lang programma en U denkt misschien dat daar ook een behoorlijk lange uitleg bij hoort, maar dat valt wel mee omdat dit programma veel in de VIDEO-RAM doet en dat wordt later in dit hoofdstuk uitvoerig besproken.
Een groffe beschrijving van het programma:
De sprite wordt getekend in kleur 1 (blauw), dat is de kleur die palet 1 heeft. Boven en onder aan het scherm wordt een lijntje in de kleur zwart getekend en dat is de kleur die palet 2 heeft.
In het programma wordt gecontroleerd of de sprite bij het bewegen aanbeland is bij het zwarte lijntje onder- of bovenaan het scherm. Is dat het geval dan zal de sprite niet meer verder in die richting bewegen.
Hoe kan nu gecontroleerd worden of het zwarte lijntje is geraakt? Er is een poort die bepaalt welk palet er gelezen kan worden en dat is poort CDH. Zetten we deze poort op 02H dan wordt alleen palet 2 gelezen. Komt het programma deze palet tegen dan is één van de lijntjes bereikt.

Aangezien alleen deze twee lijntjes uit palet 2 zijn opgebouwd, zal de sprite nergens anders op het scherm stoppen.
Gelijk in het begin wordt de joystick al uitgelezen. Is de waarde van poort F0H gelijk aan

F7H dan is de joystick naar rechts bewogen.
FBH dan is de joystick naar links bewogen.
FEH dan is de joystick omhoog bewogen.
FDH dan is de joystick omlaag bewogen.

De sprite wordt bewogen door het gebied van 4 bij 24 bytes waarin de sprite zich bevindt in de gewenste richting te scrollen. Alles wat er in de kleur van palet 1 op het scherm staat, verdwijnt als de sprite er overheen gaat. In mode M1 van dit programma staat er niets op het scherm in blauw afgebeeld, maar in de aanpassingen voor M2 staat er wel iets in de kleur van palet 1. U zult zich misschien afvragen waar dan? Wat is er precies aan de hand. De cirkel in het midden van het scherm is in kleur 3 getekend en dat is de mengkleur van palet 1 en 2. Gaat U met de sprite over de cirkel heen dan zal dus de kleur van palet 1 uit de cirkel verdwijnen en zal alleen de kleur van palet 2 achterblijven. Is de kleur van palet 2 gelijk aan de kleur van palet 1 dan zal de sprite voor de cirkel te zien zijn, omdat alleen palet 2 eigenlijk maar te zien is. Is de kleur van palet 2 gelijk aan de kleur van de cirkel dan zal de sprite juist niet te zien zijn.
Dit begrijpt U misschien niet zo goed, maar U zult het waarschijnlijk beter leren begrijpen als de programma's uit hoofdstuk 3 nader besproken zijn.

Assemblerlisting van programma 1 uit hoofdst.3.1 - blz.54.

```
FE00 IN A,(E0H)
FE02 LD A,83H
FE04 OUT (CCH),A
FE06 OUT (CDH),A
FE08 LD HL,8000H
FE0B LD DE,8001H
FE0E LD C,C8H
FE10 SCROLL:  LD A,(HL)
FE11 EX AF,AF'
FE12 LD B,27H
FE14 LOOP:  LD A,(DE)
FE15 LD (HL),A
FE16 INC HL
FE17 INC DE
FE18 DJNZ LOOP
FE1A EX AF,AF'
FE1B LD (HL),A
FE1C INC HL
FE1D INC DE
FE1E DEC C
FE1F JR NZ,SCROLL
FE21 IN A,(E1H)
FE23 RET
```

Uitleg van programma 1:

Dit programma zal heel uitgebreid uitgelegd worden.
Allereerst wordt op adres FE00H gebankswitscht. Het geheugen wordt zo ingedeeld dat het VIDEO-RAM vanaf adres 8000H te vinden is.
Via de poorten CCH en CDH kan bepaald worden in welke kleur er geschreven wordt en in welke kleur er gelezen wordt.
In mode M1 worden twee paletten gebruikt en dat zijn de paletten 1 en 2. Hiermee kunnen drie verschillende kleuren gemaakt worden. Allereerst kunnen zowel palet 1 als palet 2 afzonderlijk een kleur krijgen. Daarnaast kan uit deze twee paletten een mengkleur gemaakt worden.

Met de BASIC-instructie PAL 1,x kunt U de kleur van palet 1 bepalen.
Met de BASIC-instructie PAL 2,x kunt U de kleur van palet 2 bepalen.
Met de BASIC-instructie PAL 3,x kunt U de 'mengkleur' van de paletten 1 en 2 bepalen. Hoe het mengen precies in elkaar zit kunnen we U helaas niet uitleggen. Verderop zal duidelijk aangetoond worden dat PAL 3,x een combinatie is van palet 1 en 2.
Terug naar de beide poorten. U kunt alle waarden van 00H t/m FFH uit deze poorten sturen, maar het meest worden de waarden tussen 00H en 0FH en de waarden tussen 80H en 8FH gebruikt. Wat doen deze waarden?
De waarden van 00H t/m 0FH lezen of schrijven alleen een bepaald palet. Bij het schrijven blijft alles wat er op het scherm stond ook staan. Wanneer U dus in kleur 2 iets over kleur 1 heen schrijft, krijgt U een mengkleur te zien op de plekken waar deze twee kleuren elkaar overlappen.
In mode M1 is de waarde van palet 1 gelijk aan 01H en de waarde van palet 2 gelijk aan 02H.
In mode M2 kunt U vier paletten tegelijk gebruiken en daar zijn in totaal 15 kleurcombinaties mee te maken. In deze mode is de waarde van palet 1 gelijk aan 01H, de waarde van palet 2 gelijk aan 02H, de waarde van palet 3 gelijk aan 04H en de waarde van palet 4 gelijk aan 08H.
Wanneer U nu alle kleuren wilt scrollen, moet U in mode M1 twee paletten en in mode M2 vier paletten scrollen. Wanneer U dus slechts één palet of één kleur wilt scrollen gaat dat een stuk sneller.
De waarden van 80H t/m 8FH lezen of schrijven alleen een bepaalde kleur. In mode M1 zijn dat 3 kleuren (81H t/m 83H) en in mode M2 zijn dat 15 kleuren (81H t/m 8FH). Bij het schrijven verdwijnt alles wat er origineel op die plaats stond.
Dit programma leest alles wat er in kleur 3 (wit) op het scherm staat en schrijft ook enkel en alleen in kleur 3.
Het HL-register wordt geladen met het adres van het meest linker byte bovenaan het scherm. De waarde van dit adres wordt ingelezen in de accumulator en daarna bewaard in het schaduwregister (EX AF,AF') van de accumulator.
Het DE-register wordt geladen met het adres van het tweede byte op de bovenste rij.
In het gedeelte dat LOOP is genoemd wordt eerst de waarde van het adres aangegeven door het DE-register, in de accumulator geladen en vervolgens wordt de waarde van de accumulator op het adres aangegeven door het HL-register, geladen en HL en DE schuiven vervolgens één plaats naar rechts en hetzelfde principe wordt herhaald. Dat gaat zo door tot er 39 bytes (op één byte na dus precies één regel) één plaats naar links geschoven zijn. Aan het eind van een regel komt dan de waarde te staan van het schaduwregister van de accumulator. Hierdoor komt het begin van het scherm er dus achteraan weer bij.
Er is nu één regel naar links gescrolld en zo wordt dat met alle 200 regels gedaan. Aan het eind wordt er dan naar de BASIC teruggesprongen na het geheugen weer op de oorspronkelijke manier (IN A,(E1H)) ingedeeld te hebben.
Het scrollen op zich is al een keer eerder besproken, maar het werken met de poorten CCH en CDH nog niet. Aan deze poorten zal veel aandacht geschonken worden, omdat daar eigenlijk alles om draait bij de verschillende programma's. Door deze poorten anders aan te sturen kan het kleureffect bij het scrollen heel anders worden.
Op de volgende bladzijde volgt eerst een overzicht van de opbouw van verschillende kleuren. Uit welke paletten zijn ze opgebouwd?

In mode M1 zijn de kleuren als volgt opgebouwd:

kleur 1 - oorspronkelijk blauw - is alleen opgebouwd uit palet 1.
kleur 2 - oorspronkelijk rood - is alleen opgebouwd uit palet 2.
kleur 3 - oorspronkelijk wit - is opgebouwd uit de paletten 1 en 2.

In mode M2 zijn de kleuren als volgt opgebouwd:

kleur 1 - oorspr. blauw - alleen opgebouwd uit palet 1.
kleur 2 - oorspr. rood - alleen opgebouwd uit palet 2.
kleur 3 - oorspr. helrood - opgebouwd uit de paletten 1 en 2.
kleur 4 - oorspr. groen - alleen opgebouwd uit palet 3.
kleur 5 - oorspr. cyaan - opgebouwd uit de paletten 1 en 3.
kleur 6 - oorspr. geel - opgebouwd uit de paletten 2 en 3.
kleur 7 - oorspr. wit - opgebouwd uit de paletten 1, 2 en 3.
kleur 8 - oorspr. grijs - alleen opgebouwd uit palet 4.
kleur 9 - oorspr. lichtblauw - opgebouwd uit de paletten 1 en 4.
kleur 10 - oorspr. lichtrood - opgebouwd uit de paletten 2 en 4.
kleur 11 - oorspr. licht helrood - opgebouwd uit de paletten 1, 2 en 4.
kleur 12 - oorspr. lichtgroen - opgebouwd uit de paletten 3 en 4.
kleur 13 - oorspr. licht cyaan - opgebouwd uit de paletten 1, 3 en 4.
kleur 14 - oorspr. lichtgeel - opgebouwd uit de paletten 2, 3 en 4.
kleur 15 - oorspr. lichtwit - opgebouwd uit alle paletten.

Deze opbouw is heel mooi te zien in programma 5 uit hoofdst.3.1.

Assemblerlisting van programma 2 uit hoofdst.3.1 - blz.55.

```
FE00 IN A,(E0H)
FE02 LD A,83H
FE04 OUT (CCH),A
FE06 OUT (CDH),A
FE08 LD HL,8000H
FE0B LD DE,7FD8H
FE0E LD BC,1F40H
FE12 LDIR
FE14 LD DE,9F18H
FE17 LD HL,7FD8H
FE1A LD BC,0028H
FE1D LDIR
FE1F IN A,(E1H)
FE21 RET
```

Uitleg van programma 2:

Eigenlijk is alles uit dit programma al eens eerder besproken. Een uitleg lijkt ons dan ook niet nodig.

Toelichting op programma 3 uit hoofdst.3.1 - blz.55.

In dit programma wordt de kleur in palet 1 naar rechts en de kleur in palet 2 naar links gescrolld. In het gebied waar deze twee kleuren elkaar overlappen is de kleur te zien die samengesteld wordt uit de paletten 1 en 2.
Omdat de waarden die uit de poorten CCH en CDH gestuurd worden tussen 00H en 0FH liggen, wordt het overlappingseffect bereikt. Dit zou niet het geval zijn geweest als de waarden tussen 80H en 8FH hadden gelegen, want dan zou bij de eerste scroll (USR($FD00)) de kleur in palet 2 en bij de tweede scroll (USR($FD30)) de kleur in palet 1 verdwenen zijn.

Toelichting op programma 4 uit hoofdst.3.1 - blz.56.

Dit programma werkt in mode M2 en stuurt waarden tussen 80H en 8FH uit de poorten CCH en CDH. Dit betekent dat er dus met één kleur gewerkt wordt en dat de andere kleuren verdwijnen als deze overlapt worden. U ziet in dit programma duidelijk dat dat klopt.

Toelichting op programma 5 uit hoofdst.3.1 - blz.56.

In dit programma kunt U zien uit welke hoofdkleuren de 16 kleuren in mode M2 zijn opgebouwd. U kunt dan zien dat het rijtje van de vorige bladzijde klopt.

Toelichting op programma 6 uit hoofdst.3.1 - blz.56.

Door eerst alle kleuren die uit palet 1 zijn opgebouwd te scrollen en dat vervolgens ook met de andere drie paletten te doen, worden alle kleuren gescrolld. Zo ziet U ook dat de kleuren uit maximaal vier hoofdkleuren zijn opgebouwd.

Assemblerlisting van programma 7 uit hoofdst.3.1 - blz.57 - 58.

```
FE00 ORG FE00H
FE00 IN A,(E0H)
FE02 LD HL,8000H
FE05 LD DE,DE00H
FE08 LD BC,2000H
FE0B LDIR
FE0D IN A,(E1H)
FE0F RET

FE80 ORG FE80H
FE80 IN A,(E0H)
FE82 LD A,85H
FE84 OUT (CCH),A
FE86 OUT (CDH),A
FE88 LD HL,9EF0H
FE8B LD DE,DE00H
FE8E LD C,01H
FE90 SCROLL:  LD B,50H
FE92 LOOP:  LD A,(DE)
FE93 LD (HL),A
FE94 INC HL
FE95 INC DE
FE96 DJNZ LOOP
FE98 DEC C
FE99 JR NZ,SCROLL
FE9B IN A,(E1H)
FE9D RET

FEA0 ORG FEA0H
FEA0 IN A,(E0H)
FEA2 LD HL,8000H
FEA5 LD DE,8001H
FEA8 LD C,32H
FEAA LINKS:  LD B,50H
FEAC LR:  LD A,(DE)
FEAD LD (HL),A
FEAE INC HL
FEAF INC DE
FEB0 DJNZ LR
FEB2 LD (HL),00H
FEB4 LD A,50H
FEB6 REGEL:  DEC DE
FEB7 DEC HL
FEB8 DEC A
FEB9 JR NZ,REGEL
FEBB DEC C
FEBC JR NZ,LINKS
FEBE IN A,(E1H)
FEC0 RET

FED0 ORG FED0H
FED0 IN A,(E0H)
FED2 LD HL,9F3FH
FED5 LD DE,9F3EH
FED8 LD C,32H
FEDA RECHTS:  LD B,50H
FEDC RR:  LD A,(DE)
FEDD LD (HL),A
FEDE DEC HL
FEDF DEC DE
```

```
FEE0 DJNZ RR
FEE2 LD (HL),00H
FEE4 LD A,50H
FEE6 TREGEL:  DEC DE
FEE7 DEC HL
FEE8 DEC A
FEE9 JR NZ,TREGEL
FEEB DEC C
FEEC JR NZ,RECHTS
FEEE IN A,(E1H)
FEF0 RET
```

Uitleg van programma 7:

Dit programma bestaat uit vier stukken. Het eerste stuk is bedoeld om een scherm van het VIDEO-RAM geheugen in het normale geheugen te zetten. Het tweede stuk is bedoeld om het scherm dat ergens in het geheugen is opgeslagen, langzaam weer in het VIDEO-RAM geheugen zichtbaar te maken. Het derde stuk scrollt alle even rijen naar links en het vierde stuk scrollt alle oneven rijen naar rechts.
Het eerste stuk:
Er wordt een scherm ingelezen en vanaf DE00H in het geheugen gezet.
De instructies uit dit stuk zijn allemaal wel bekend.
Het tweede stuk:
Hier wordt het weggeschreven scherm langzaam weer zichtbaar gemaakt op het beeldscherm. Het BASIC-programma zorgt er voor dat het scherm elke keer een regel omhoog gaat.
De kleur die gebruikt wordt is kleur 5. Alle andere zaken verdwijnen van het scherm, omdat de waarde van de poorten CCH en CDH tussen 80H en 8FH ligt.
In het BASIC-programma wordt het HL-register elke keer met 80 verlaagd. (Eén regel omhoog dus, omdat er in de 80-koloms mode wordt gewerkt.) Tevens wordt het C-register met 1 verhoogd om één regel meer zichtbaar te maken.
Omdat er steeds meer regels zichtbaar gemaakt moeten worden, wordt het machinetaalprogramma steeds langzamer. Dit wordt opgevangen door de BASIC-instructie WAIT. In het begin wordt er wat langer gewacht dan aan het eind.
Het derde stuk:
Hier worden alle even regels naar links gescrolld. Allereerst wordt regel 0 naar links gescrolld. Daarna wordt één regel overgeslagen en wordt regel 2 naar links gescrolld. Zo gaat dat met alle even regels beneden de 100. Na regel 100 staat het lijnenpatroon. Als er dus na regel 100 gescrolld gaat worden, zal het lijnenpatroon verdwijnen.
Het vierde stuk:
Hier worden alle oneven regels beneden de 100 naar rechts gescrolld. Samen met het derde stuk zorgt dit stuk er dus voor dat de tekst naar twee kanten uit elkaar wordt getrokken.
Er is voor een 80-koloms mode gekozen, omdat het effect anders minder mooi was geweest. Of er had bit voor bit gescrolld moeten worden, maar dat is weer te langzaam.

Assemblerlisting van programma 8 uit hoofdst.3.1 - blz.58.

```
FD00 IN A,(E0H)
FD02 LD A,01H
FD04 OUT (CCH),A
FD06 OUT (CDH),A
FD08 LD HL,9F17H
FD0B EXX
FD0C LD HL,9EF0H
FD0F LD C,C7H
FD11 BEGIN:  LD A,(HL)
FD12 RRA
```

```
FD14 EXX
FD15 LD B,28H
FD17 ROT:  RR (HL)
FD19 DEC HL
FD1A DJNZ ROT
FD1C EXX
FD1D LD DE,FFD8H
FD20 ADD HL,DE
FD21 DEC C
FD22 JR NZ,BEGIN
FD24 IN A,(E1H)
FD26 RET
```

Uitleg van programma 8:

Het bit voor bit scrollen gaat niet zo gemakkelijk als het scrollen van een hele byte in één keer. Het voordeel is wel dat de byte alleen geroteerd hoeft te worden en niet ook nog eens verplaatst moet worden.
Eerst wordt er natuurlijk gebankswitscht. Vervolgens wordt het palet voor het lezen en schrijven bepaald en dat is palet 1. Alleen de kleur blauw zal dus gescrolld worden en de rest zal blijven staan.
Het HL-register wordt geladen met het adres van de meest rechter positie van de onderste rij. Vervolgens worden alle registers bewaard in hun schaduwregisters (EXX). Het HL-register wordt vervolgens geladen met het adres van de eerste positie van de onderste rij. Het C-register wordt met het aantal rijen (199) geladen en daarna wordt de inhoud van het HL-register in de accumulator geladen en de accumulator wordt naar rechts geroteerd (RRA). De waarde van het meest rechter bit komt nu in de C-vlag.
Het B-register wordt geladen met het aantal hokjes per rij (40) en daarna worden alle 40 hokjes op de onderste rij één plaats naar rechts geroteerd.
Hoe gaat dat roteren nu precies?
Een byte bestaat uit 8 bits, zoals U allemaal zult weten. Wordt nu de instructie RRA gegeven dan worden alle 8 bits in de accumulator één plaats naar rechts geschoven. Maar wat gebeurt er nu precies met het eerste en het laatste bit? Het eerste bit wordt geladen met de waarde van de C-vlag en de waarde van het laatste bit wordt in de C-vlag geladen.
Omdat eerst het laatste byte in een rij geroteerd wordt, zal de waarde van het laatste bit in de C-vlag komen en aangezien vervolgens het eerste byte geroteerd wordt, zal de waarde van de C-vlag (daarin bevindt zich dus het laatste bit in een rij) aan het eerste bit op een rij gegeven worden. Zo komt dus alles wat aan het eind verdwijnt aan het begin weer terug.
Door deze methode van scrollen is het ook mogelijk om het laatste bit van iedere byte door te geven aan het eerste bit van de volgende byte.
Op deze manier worden alle rijen afgewerkt en is het scherm dus 1 bit naar links geschoven.
In een overzicht komt RRA er als volgt uit te zien:
(We nemen aan dat de C-vlag 0 is.)

01100101 komt er na RRA als volgt uit te zien: C-vlag > 00110010.
De C-vlag zal dus nu de waarde 1 van het laatste bit bevatten.

Waarom beweegt alles naar links, terwijl U naar rechts aan het roteren bent?
Dit heeft te maken met het feit dat de SHARP op het scherm een BYTE omkeert. Ziet een BYTE er bijvoorbeeld uit als 00010010 dan zal op het scherm 01001000 komen te staan. Denkt U hier aan als U bijvoorbeeld een character wilt gaan veranderen.

Assemblerlisting van programma 9 uit hoofdst.3.1 - blz.59.

```
FD00 IN A,(E0H)
FD02 BEGIN:  LD A,01H
FD04 OUT (CCH),A
FD06 OUT (CDH),A
FD08 LD HL,9F17H
FD0B EXX
FD0C LD HL,9EF0H
FD0F LD C,C7H
FD11 RIJ:  LD A,(HL)
FD12 RRA
FD13 EXX
FD14 LD B,28H
FD16 ROT:  RR (HL)
FD18 DEC HL
FD19 DJNZ ROT
FD1B EXX
FD1C LD DE,FFD8H
FD1F ADD HL,DE
FD20 DEC C
FD21 JR NZ,RIJ
FD23 LD A,(BEGIN+1)
FD26 ADD A,A
FD27 LD (BEGIN+1),A
FD2A CP 10H
FD2C JP NZ,BEGIN
FD2F LD A,01H
FD31 LD (BEGIN+1),A
FD34 REG:  LD BC,0240H
FD37 DEC C
FD38 LD (REG+1),BC
FD3B JR NZ,BEGIN
FD3D DEC B
FD3E LD (REG+1),BC
FD41 JR NZ,BEGIN
FD43 IN A,(E1H)
FD45 RET
```

Uitleg van programma 9:

Dit programma scrollt het hele scherm tot het weer in de beginpositie is gekomen. Dit gaat volledig in machinetaal, het programma is dus niet met SHIFT/BREAK te onderbreken. Het programma lijkt veel op de voorgaande programma's, veel uitleg zal dus niet nodig zijn.
Natuurlijk wordt er eerst weer gebankswitscht.
Allereerst wordt palet 1 gescrolld. Vervolgens wordt palet 2 (ADD A,A oftewel A=A+A) gescrolld. Daarna wordt palet 3 gescrolld. U weet hopelijk nog dat daarvoor de waarde 04H gebruikt moet worden. Als laatste wordt palet 4 gescrolld.
Door de accumulator dus elke keer met zichzelf te vermeerderen, worden achtereenvolgens de waarden 01H, 02H, 04H en 08H uit de poorten CCH en CDH gestuurd. Deze waarden staan precies voor de verschillende paletten. Dat betekent dus dat alle paletten gescrolld worden. Is de waarde van de accumulator gelijk aan 10H dan zal de inhoud van de accumulator weer 1 worden.
In het BC-register aan het eind wordt bijgehouden of het scherm al 320 keer gescrolld is. Is dat het geval dan zal het geheugen eerst weer op de normale manier ingedeeld worden en zal vervolgens naar de BASIC teruggesprongen worden.

Assemblerlisting van programma 10 uit hoofdst.3.1 - blz.59.

```
FD00 IN A,(E0H)
FD02 LD A,02H
FD04 OUT (CCH),A
FD06 OUT (CDH),A
FD08 LD HL,932CH
FD0B LD C,40H
FD0D RIJEN:  SCF
FD0E CCF
FD0F LD B,12H
FD11 ROT:  RR (HL)
FD13 DEC HL
FD14 DJNZ ROT
FD16 LD DE,FFEAH
FD19 ADD HL,DE
FD1A DEC C
FD1B JR NZ,RIJEN
FD1D IN A,(E1H)
FD1F RET
```

Korte uitleg bij programma 8:

Dit programma zit bijna hetzelfde in elkaar als enkele voorgaande programma's, er is echter één verschil en dat is het zetten van de C-vlag aan de hand van de instructies SCF en CCF. Wanneer deze instructies in deze volgorde gebruikt worden, zal de C-vlag op nul gezet worden. Wat doen de instructies SCF en CCF precies?
De instructie SCF (Set Carry Flag) zet het bit in de C-vlag op één.
De instructie CCF (Complement Carry Flag) zet het bit in de C-vlag op nul als het één is≠ en op één als het nul is. Het uiteindelijke resultaat is dus dat de C-vlag op nul gezet wordt.
Als de C-vlag niet gelijk aan nul zou zijn aan het begin van een regel, zou een misvormd patroon ontstaan, omdat de inhoud van de C-vlag in het eerste bit komt.

Toelichting bij programma 12 uit hoofdst.3.1 - blz.60.

Dit programma scrollt het hele scherm omhoog en scrollt de onderste helft ook nog eens naar links. Dit resulteert in een scrollrichting naar links boven. U ziet dat dit een heel leuk effect op kan leveren. In een programma van H. Smits wordt dit ook gebruikt.

Assemblerlisting van programma 1 uit hoofdst.3.2 - blz.61.

```
F800 IN A,(E0H)
F802 LD HL,8000H
F805 LD C,C8H
F807 RIJEN:  LD B,28H
F809 BYTE:   LD D,04H
F80B REG:    LD A,01H
F80D OUT (CCH),A
F80F OUT (CDH),A
F811 ADD A,A
F812 LD (REG+1),A
F815 LD A,(HL)
F816 CPL
F817 LD (HL),A
F818 DEC D
F819 JR NZ,REG
F81B INC HL
F81C LD A,01H
F81D LD (REG+1),A
F820 DJNZ BYTE
F822 DEC C
F823 JR NZ,RIJEN
F825 IN A,(E1H)
F827 RET
```

Uitleg van programma 1:

Alle bytes worden stuk voor stuk voor alle vier de paletten uitgelezen en geïnverteerd. Dit gebeurt door de instructie CPL. Deze instructie zorgt er voor dat alle bits in de accumulator die nul zijn, één worden en alle bits die één zijn, nul worden. Alle kleuren op het scherm krijgen hun tegengestelde kleur. Kleur 0 wordt kleur 15, kleur 1 wordt kleur 14, kleur 2 wordt kleur 13, enz.
Bekijkt U de tabel van de opbouw van de verschillende kleuren dan zult U zien dat dat ook klopt. Als iets zwart is, staat er dus geen enkele kleur en zullen alle paletten dus op die plaats komen te staan en alle vier de paletten bij elkaar vormen de kleur wit.
Als ergens de kleur blauw staat (palet 1 dus), staan daar niet de paletten 2, 3 en 4 en die komen er na dit programma wel te staan en palet 1 verdwijnt. De paletten 2, 3 en 4 vormen bij elkaar de kleur lichtgeel. Zo gaat dat met alle kleuren.

Verbetering van programma 3 uit hoofdst.3.1 - blz.62.

Achteraf is gebleken dat het verticaal omkeren van het scherm veel eenvoudiger kan. De methode die op blz.62 wordt getoond leest het scherm bit voor bit uit en dat is echt veel te omlachtig. Het kan door gewoon te roteren. Daarbij worden de accumulator, het B-register en de C-vlag gebruikt. Het programma komt er dan als volgt uit te zien:

```
10 LIMIT$FC00
20 DATA $DB,$E0,$21,$00,$80,$11,$27,$80,
$0E,$C8,$06,$14,$C5,$3E,$01,$D3,$CC,$D3,
$CD,$7E,$0E,$08,$1F,$CB,$10,$0D,$20,$FA,
$78,$08,$1A,$0E,$08,$1F,$CB,$10,$0D
30 DATA $20,$FA,$78,$77,$08,$12,$3A,$0E,
$FC,$87,$32,$0E,$FC,$FE,$10,$20,$D7,$3E,
$01,$32,$0E,$FC,$C1,$23,$1B,$10,$CC,$D5,
$11,$3B,$00,$19,$D1,$EB,$D5,$11,$15,$00
40 DATA $19,$D1,$0D,$20,$BA,$DB,$E1,$C9
50 FOR A=0 TO 82 :READ B :POKE $FC00+A,B
 :NEXT A
60 USR($FC00)
```

Dit programma is ongeveer maar de helft van het programma op blz.62 en het doet precies hetzelfde. Het werkt bovendien ook nog iets sneller. Het kan eventueel nog korter, maar daardoor zal het niet sneller worden.

Assemblerlisting van het bovenstaand programma.

```
FC00 IN A,(E0H)
FC02 LD HL,8000H
FC05 LD DE,8027H
FC08 LD C,C8H
FC0A RIJEN:  LD B,14H
FC0C BEGIN:  PUSH BC
FC0D REG:  LD A,01H
FC0F OUT (CCH),A
FC11 OUT (CDH),A
FC13 LD A,(HL)
FC14 LD C,08H
FC16 AROT: RRA
FC17 RL B
FC19 DEC C
FC1A JR NZ,AROT
FC1C LD A,B
FC1D EX AF,AF'
FC1E LD A,(DE)
FC1F LD C,08H
FC21 BROT:  RRA
FC22 RL B
FC24 DEC C
FC25 JR NZ,BROT
FC27 LD A,B
FC28 LD (HL),A
FC29 EX AF,AF'
FC2A LD (DE),A
FC2B LD A,(REG+1)
FC2E ADD A,A
FC2F LD (REG+1),A
FC32 CP 10H
FC34 JR NZ,REG
FC36 LD A,01H
FC38 LD (REG+1),A
FC3B POP BC
FC3C INC HL
FC3D DEC DE
FC3E DJNZ BEGIN
FC40 PUSH DE
FC41 LD DE,003BH
FC44 ADD HL,DE
FC45 POP DE
FC46 EX DE,HL
FC47 PUSH DE
FC48 LD DE,0015H
FC4B ADD HL,DE
FC4C POP DE
FC4D DEC C
FC4E JR NZ,RIJEN
FC50 IN A,(E1H)
FC52 RET
```

Uitleg van programma 3:

HL bevat het adres van de meest linker positie van de bovenste regel en DE bevat het adres van de meest rechter positie van de bovenste regel. A wordt geladen met de waarde die HL bevat. Deze wordt naar rechts geroteerd (Op het scherm in werkelijkheid dus naar links!) en de waarde van het meest rechter bit komt in de C-vlag. Vervolgens wordt B naar links geroteerd en komt de waarde van de C-vlag bij het eerste bit aan de rechterkant bij B binnen. Zo worden alle 8 bits in A en B afgewerkt. De inhoud van A staat daarna omgekeerd in B. A wordt geladen met de waarde van B en wordt vervolgens verwisseld met het schaduwregister en het roteren op bovenstaande manier wordt uitgevoerd met de inhoud van DE. Daarna wordt A weer geladen met de waarde van B en deze waarde wordt in HL gezet. De waarde van het schaduwregister van A wordt daarna in DE gezet.
Het eerste en laatste byte in een rij staan nu omgekeerd aan de andere kant van het scherm. Zo worden alle bytes in een rij voor alle paletten afgewerkt en zullen HL en DE elkaar op een gegeven moment in het midden tegenkomen. Dan wordt er naar de volgende regel gesprongen en wordt hetzelfde gedaan en zo wordt het hele scherm omgekeerd.

Toelichting bij programma 6 uit hoofdst.3.2 - blz.64.

In dit programma wordt de vrije geheugenruimte in de VIDEO-RAM gebruikt. Dat zit als volgt in elkaar:
U weet dat er maar twee paletten in mode M1 worden gebruikt en dat zijn de paletten 1 en 2. De geheugenruimte in de VIDEO-RAM die gebruikt wordt voor de paletten 3 en 4 om in mode M2 te kunnen werken, is dus nog vrij. Aangezien er per palet 8K aan VIDEO-RAM-geheugen gebruikt kan worden, is er dus nog 8*2=16K aan geheugenruimte in de VIDEO-RAM over.
Als er in mode M1 gewerkt wordt met de paletten 3 en 4, zal daar niets van te zien zijn op het scherm. U kunt deze paletten dus gebruiken om eventueel iets in op te slaan, data bijvoorbeeld.
Er is één manier om de paletten 3 en 4 zichtbaar te maken en dat is door OUT@$CE,2 in te voeren. Om het weer te laten verdwijnen typt U OUT@$CE,0 in. Alle informatie staat er dan nog in.
Alle informatie zal echter verdwijnen wanneer U naar mode M2 of M4 omschakelt.
In de UNI-GG-BASIC wordt deze mogelijkheid van extra geheugenopslag gebruikt als gewoon werkgeheugen. Er is dan echter wel een nadeel en dat is dat U niet naar de modes M2 en M4 kunt. Voor deze aanpassing is een ingrijpende aanpassing van de BASIC nodig geweest.
In het programma van blz.44 wordt eerst iets in mode M1 op het scherm getekend. Alles wat in palet 1 op het scherm staat wordt in palet 3 in het vrije VIDEO-RAM gedeelte gezet. Alles wat in palet 2 op het scherm staat wordt in palet 4 in het vrije VIDEO-RAM gedeelte gezet. Vervolgens verdwijnt het hele scherm. Een copy van het scherm staat echter nog in het vrije VIDEO-RAM gedeelte opgeslagen. Door na het verdwijnen van het hele scherm het copy weer in de paletten 1 en 2 op het scherm te zetten, is het oorspronkelijke scherm weer te zien.
Bij animaties, bestandsprogramma's, kaartenbakken en ga zo maar door is het ook heel gemakkelijk om het vrije gedeelte in het VIDEO-RAM gedeelte te gebruiken als extra werkgeheugen. U zou er eventueel ook een programma van maximaal 16K in op kunnen slaan.

Assemblerlisting van programma 2 uit hoofdst.6.5 - blz.123.

```
FD00 IN A,(E0H)
FD02 LD A,83H
FD04 OUT (CCH),A
FD06 OUT (CDH),A
FD08 LD HL,9C20H
FD0B LD C,14H
FD0D RECHTS:  SCF
FD0E CCF
FD0F LD B,04H
FD11 LOOP:  RL (HL)
FD13 INC HL
FD14 DJNZ LOOP
FD16 LD DE,0024H
FD19 ADD HL,DE
FD1A DEC C
FD1B JR NZ,RECHTS
FD1D IN A,(E1H)
FD1F RET
```

Uitleg van programma 2:

Dit programma zit niet zo heel erg moeilijk in elkaar. Wat doet het programma?
Het programma scrollt het gedeelte waarin het wiel zich bevindt. Aangezien in het machinetaalprogramma bepaald is dat alleen de kleur 3 gelezen en geschreven wordt en de andere kleuren zullen verdwijnen bij het scrollen, zal alleen de buitenkant van het wiel nog zichtbaar zijn na het aanroepen van de routine. De spaken verdwijnen dus automatisch na een scroll en zo gaat het beter en sneller dan in BASIC.

Assemblerlisting van het MB-programma uit hoofdst.7.2 - blz.134 - 135.

```
C000 LD HL,D000H
C003 LD DE,CFD8H
C006 LC C,19H
C008 SCROLL:  LD B,28H
C00A LOOP:  CALL 00EAH
C00D EX DE,HL
C00E CALL 00F2H
C011 EX DE,HL
C012 INC HL
C013 INC DE
C014 DJNZ LOOP
C016 DEC C
C017 JR NZ,SCROLL
C019 LD HL,CFD8H
C01C LD B,28H
C01E BO:  CALL 00EAH
C021 EX DE,HL
C022 CALL 00F2H
C025 EX DE,HL
C026 INC HL
C027 INC DE
C028 DJNZ BO
C02A RET
```

Uitleg van bovenstaand programma:

Er komen in dit programma twee CALLS voor en daar draait het om, want de rest is alleen een eenvoudige scrollroutine om het beeldscherm omhoog te scrollen.

CALL 00EAH - Deze CALL zorgt er voor dat in het gebied vanaf D000H (het beeldschermgedeelte dus) een PEEK uitgevoerd kan worden. In HL moet het adres staan dat gepeekt moet worden en in A komt, na deze CALL, de waarde van dat adres.

CALL 00F2H - Deze CALL zorgt er voor dat in het gebied vanaf D000H een POKE uitgevoerd kan worden. In HL komt het adres dat gepoked moet worden en in A komt, voor deze CALL, de waarde voor dat adres.

U kunt het beeldscherm in machinetaal dus niet rechtstreeks poken of peeken. U moet eerst één van de CALLS geven.

9.2: Uitleg van enkele assemblerinstructies en specifieke SHARP zaken.

In dit deel van hoofdstuk 9 zullen een aantal assemblerinstructies nader uitgelegd worden en zullen bepaalde SHARP zaken, zoals bijvoorbeeld de schermindeling, besproken worden.

Registers en de instructie LD.

Een register is een variabele in machinetaal. Registers kunnen waarden tussen 0 en 65535 bevatten. Er zijn enkele en dubbele registers. In een enkel register kan een waarde tussen 0 en 255 en in een dubbel register kan een waarde tussen 0 en 65535 opgeslagen worden.
De registers die wij kennen:
-Het A-register, een enkel register.
-Het BC-register, een dubbel register dat ook is op te splitsen in twee enkele registers B en C.
-Het DE-register, ook een dubbel register dat op te splitsen is in twee enkele registers D en E.
-Het HL-register, ook een dubbel register dat op te splitsen is in twee enkele registers H en L.
-De schaduwregisters van alle bovengenoemde registers.
Een register kan met een waarde geladen worden met behulp van de machinetaalinstructie LD, bijvoorbeeld LD A,20H. Let wel dat er meestal gebruik wordt gemaakt van hexadecimale waarden. Het register A is dus geladen met de waarde 20H, oftewel 32 decimaal.
Er zijn ook andere manieren om LD te gebruiken, zoals LD (HL),C8H. Deze instructie is eigenlijk gelijk aan de BASIC-instructie POKE. De waarde van HL geeft een bepaald adres in het geheugen aan. Dat adres wordt met de waarde C8H (200 decimaal) geladen. In BASIC zou dan POKE HL,200 komen te staan.
Op een soortgelijke manier kan ook een adres gepeeked worden. We willen bijvoorbeeld de inhoud van het geheugenadres D000H hebben en die inhoud moet in het A-register komen te staan, dat gaat als volgt: LD A,(D000H) en daarna zal de inhoud van het adres D000H zich in het A-register bevinden. In BASIC zou A=PEEK($D000) gebruikt worden.
Er kunnen ook twee adressen gelijk gepeeked of gepoked worden en dat kan alleen met de dubbele registers en het A-register en dat gaat bijvoorbeeld als volgt:
LD HL,(F000H) en het HL-register zal geladen worden met de waarden op de adressen F000H en F001H. Op adres F000H staat bijvoorbeeld de waarde 50H en op adres F001H staat de waarde 10H. HL zal dan geladen worden met 1050H. (Het tweede adres komt dus als eerste en het eerste adres als tweede in HL.)
Omgekeerd gaat het precies hetzelfde. Als HL de waarde 4067H bevat en de instructie LD (F000H),HL wordt gegeven, wordt adres F000H met 67H en adres F001H met 40H geladen.
Registers worden meestal als teller gebruikt.

De instructies INC en DEC.

Deze instructies worden altijd gebruikt in combinatie met een bepaald register of de inhoud van een bepaald adres dat door een register wordt aangegeven, dat is bij de meeste instructies het geval.
De instructie DEC vermindert de inhoud van een register met 1 en de instructie INC vermeerdert de inhoud met 1.

Een instructie als DEC A zal dus de inhoud van het A-register met 1 verminderen en een instructie als INC (HL) zal de inhoud van het adres dat door het HL-register aangegeven wordt, met 1 vermeerderen.
Een voorbeeld:
HL bevat de waarde D000H en de waarde van adres D000H is 80H. Vervolgens wordt de instructie INC (HL) gegeven en de inhoud van D000H zal met 1 verhoogd worden tot 81H.
Bij adressen en enkele registers geldt dat, als de waarde boven de FFH (255 decimaal) komt, het adres of register bij 0 verder gaat. De waarde 0 is dus in principe gelijk aan 256.
Bij dubbele registers geldt dat, als de waarde boven de FFFFH (65535 decimaal) komt, het register bij 0 verder gaat. De waarde 0 is hier dus in principe gelijk aan 65536.

Vlaggen.

Eigenlijk hadden de vlaggen meteen na het begrip register besproken moeten worden, maar dat is niet gedaan omdat de instructies INC en DEC ook prima aansluiten bij het begrip register.
We kennen de C-vlag, de H-vlag, de S-vlag, de Z-vlag, de P/V-vlag en de N-vlag. Elke vlag geeft 1 bit aan. In het 8-bit vlag-register waar de vlaggen in opgeslagen staan, staan de vlaggen in deze volgorde:

[S][Z][X][H][X][P/V][N][C]

De beide X-en zijn ongebruikte bits in het vlag-register.
Een vlag kan dus de waarde nul of één bevatten, dat is afhankelijk van de assemblerinstructie die uitgevoerd wordt. De meeste instructies of het resultaat er van beïnvloeden een bepaalde vlag. Aan de hand daarvan kan iets bekeken worden.
Aangezien het behandelen van de functies van de verschillende vlaggen en alles wat er nog meer met vlaggen te maken heeft heel veel ruimte in beslag neemt, zal er niet verder op de vlaggen ingegaan worden.

Instructies om de inhoud van een register of adres te controleren.

De instructies die hier besproken worden sluiten weer voor een groot deel aan bij de instructies die hiervoor al besproken zijn.
De instructies die hier besproken zullen worden, zijn DJNZ xx ; JR xx ; JR NZ,xx ; JR Z,xx ; JR NC,xx ; JR C,xx ; CALL NZ,xxxx ; enz.
Omdat de meeste instructies op hetzelfde neer komen, zal een algemene uitleg het beste zijn.
De allereerste instructie, DJNZ xx, heeft enkel en alleen betrekking op het B-register. Deze instructie trekt 1 van het B-register af, controleert vervolgens of B al gelijk aan nul is en is dat niet het geval dan zal naar het adres gesprongen worden dat door xx bepaald wordt.
Als xx tussen 80H en FFH ligt, zal naar een adres vóór dit adres gesprongen worden. Als xx tussen 01H en 7FH ligt, zal naar een adres na dit adres gesprongen worden. Hierbij wordt met dit adres het adres bedoeld waarop de instructie DJNZ xx staat.
Met xx wordt aangegeven hoeveel adressen er terug of vooruit gesprongen moet worden. Is de waarde bijvoorbeeld F0H dan zal er 14 adressen (De twee adressen die nodig zijn voor de instructie worden er dus afgetrokken.) teruggesprongen worden.

Is de waarde bijvoorbeeld 10H dan zal er 18 adressen vooruit gesprongen worden. De twee adressen die nodig zijn voor de instructie worden er nu dus bij opgeteld.
De instructies waar xxxx bij staat kunnen naar een bepaald adres, aangegeven door xxxx, springen. Wanneer xxxx bijvoorbeeld 2015H is dan zal er naar dat adres gesprongen worden.
In assemblertaal staat er voor xx en xxxx meestal geen waarde, maar de naam van een bepaald adres. In assemblertaal worden labels gegeven aan bepaalde adressen of aan het begin van een routine. Wanneer er bijvoorbeeld staat EEN: ,begint daar de routine EEN. Staat er nu ergens DJNZ EEN dan zal naar het begin van deze routine gesprongen worden.
Er zijn veel verschillende instructies om de inhoud van registers of vlaggen te controleren en aan de hand daarvan iets uit te laten voeren. Hier volgen alle instructies:
DJNZ xx ; JR NZ,xx ; JR Z,xx ; JR NC,xx ; JR C,xx ; RET NZ ; JP NZ,xxxx ; CALL NZ,xxxx ; JP Z,xxxx ; CALL Z,xxxx ; RET NC ; JP NC,xxxx ; CALL NC,xxxx ; JP C,xxxx ; CALL C,xxxx ; RET PO ; RET PE ; CALL PE,xxxx ; RET P ; CALL P,xxxx ; RET M ; JP M,xxxx en CALL M,xxxx.
De eerste instructie is al uitgelegd, de rest lijkt allemaal veel op elkaar.
JR Z staat voor Jump Relative Zero en dat is wanneer een register naar een bepaalde verandering gelijk aan 0 is. Deze instructie heeft betrekking op het laatste register dat verandert is vóór deze instructie. Een voorbeeld:

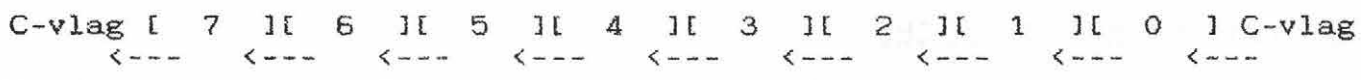
```
LD C,10H
EEN:  DEC C
JR Z,TWEE
JR EEN
TWEE:  RET
```

Zolang het C-register ongelijk aan 0 is zal naar routine EEN gesprongen worden, dat gebeurt dmv de instructie JR EEN. Deze instructie zegt gewoon dat er naar routine EEN gesprongen moet worden. In de BASIC wordt daar de instructie GOTO voor gebruikt.
De instructie RET zegt dat er teruggeprongen moet worden naar de plaats vanwaar deze routine aangesproken is, dat kan een hoofdprogramma, de BASIC, de ROM- of RAM-monitor of iets dergelijks zijn. In BASIC wordt daar de instructie RETURN voor gebruikt.
JR NZ,xx zit precies omgekeerd in elkaar. Hierbij staat NZ voor Not Zero, oftewel niet gelijk aan 0.
JR C,xx staat voor Jump Relative Carry (wanneer Carry-vlag 1 is dan ..)
JR NC,xx zit dus weer precies omgekeerd in elkaar. Hierbij staat NC voor Not Carry.
CALL staat voor spring naar. De uitdrukkingen die daarna kunnen volgen zijn hetzelfde als hiervoor. In BASIC wordt hier de instructie GOSUB voor gebruikt. xxxx staat natuurlijk weer voor een zelf te kiezen adres.
RET kan ook gebruikt worden als bijvoorbeeld RET NZ. Er wordt in dat geval dus alleen teruggesprongen als het register ongelijk is aan nul.
Dan zijn er ook nog toevoegingen als PO, PE, P en M. De P staat voor de Pariteitsvlag. PO (Parity Odd) staat voor oneven pariteit en PE voor even pariteit (Parity Even). Wat pariteiten precies in houden en waarvoor ze dienen zullen we niet uit leggen, omdat het in de besproken programma's toch niet gebruikt wordt.

De instructies met de toevoeging M worden ook niet in dit boek gebruikt, ze zullen daarom ook niet nader uitgelegd worden.
Er werd gezegd dat de getoonde instructies alle instructies in die richting zijn, maar dat is niet helemaal waar. Er zijn nog meer instructies, maar die worden ook niet gebruikt in dit boek. Een nadere uitleg is dus in principe zinloos.

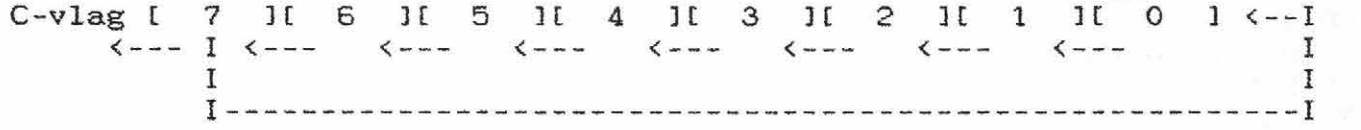
Roteerinstructies.

In hoofdstuk 3 wordt heel veel gebruik gemaakt van roteerinstructies, zoals bijvoorbeeld RLA. De instructies roteren de bits binnen een bepaalde byte op een eigen manier.
De instructie RL x (Rotate Left), waarbij x ieder willekeurig enkel register of de inhoud van HL kan zijn, roteert een byte op de volgende manier:
De inhoud van de C-vlag komt in het laatste bit (dat is dus bit 0) en de inhoud van het eerste bit komt in de C-vlag. De bits schuiven allemaal een plaats naar links op. In een overzicht komt dat er als volgt uit te zien:



U ziet dat bit 7 van een byte dus als eerste komt en op de SHARP is dat precies andersom. Op de SHARP komt namelijk bit 0 als eerste.
Bij RR x (Rotate Left) gaat het natuurlijk precies de andere richting uit.

De instructie RLC x, waarbij x weer ieder willekeurig enkel register voor kan stellen, roteert een byte zo dat het eerste bit zowel in de C-vlag als in het laatste bit komt. In een overzicht ziet dat er als volgt uit:



Bij RRC x gaat het natuurlijk weer precies de andere richting uit.

IN en OUT instructies.

Veelgebruikte instructies zijn ook de instructies IN en OUT. Deze instructies doen precies hetzelfde als de BASIC-instructies IN@ en OUT@, daarom zal slechts een korte uitleg nodig zijn.
IN wordt gebruikt om een bepaalde poort uit te lezen en de waarde daarvan in een register te zetten.
OUT wordt gebruikt om een bepaalde waarde uit een poort te sturen. De waarde kan rechtstreeks of via een register aangegeven worden.

De instructies EX AF,AF' ; EXX ; PUSH xx en POP xx.

De mogelijkheid bestaat om de inhoud van bepaalde registers ergens te bewaren of om te wisselen, omdat U het register bijvoorbeeld nodig heeft voor andere doeleinden.

Ieder dubbel register en het A-register (de accumulator) hebben een schaduwregister. Door de instructie EXX worden alle waarden van de gewone registers met de waarden van de shàduwregisters omgewisseld. Er bestaat ook een dergelijke instructie die enkel en alleen bedoeld is voor de accumulator, omdat die heel veel gebruikt wordt, en dat is de instructie EX AF,AF'. Deze instructie wisselt dus de waarde van de accumulator om met de waarde van het shaduwregister.
Er bestaat ook een dergelijke instructie om de waarden van het DE-register en het HL-register om te wisselen en dat is de instructie
EX DE,HL.
Er zijn nog een aantal instructies die iets dergelijks doen, maar die zijn niet relevant omdat ze niet in dit boek voor komen.
Er is ook een instructie om de waarden van de registers alleen te bewaren en dat is de instructie PUSH xx, hierbij kan xx AF, HL, BC of DE zijn. Deze instructie bewaart de inhoud van een bepaald register. Met de instructie POP xx kan de waarde weer teruggehaald worden. Het verschil met de instructie EX is dat hierbij de waarde van een register na de instructie PUSH niet is veranderd en bij EX is de waarde verwisseld met de waarde van het schaduwregister.

De instructies ADD en CP.

Allereerst komt de instructie CP aan de orde. Deze instructie heeft alleen betrekking op de accumulator. De instructie CP xx, waarbij xx een getal, een enkel register of de inhoud van HL voor kan stellen, trekt de waarde van xx af van de accumulator en daarna wordt met bijvoorbeeld de instructie CALL Z,xxxx gekeken of de uitkomst van deze rekensom gelijk aan nul (of in andere gevallen: ongelijk aan nul) is. Bij de instructie CP xx verandert de inhoud van de accumulator echter niet!!! Er wordt alleen naar de uitkomst van de rekensom gekeken.
Een voorbeeld:

```
START:  LD HL,D000H
LD B,FFH
LOOP:   LD A,(HL)
CP FFH
RET Z
INC HL
DJNZ LOOP
JP START
```

Dit programmaatje zal oneindig door lopen als er tussen de adressen D000H t/m D0FFH geen enkel adres is dat de waarde FFH bevat.
Het programma laadt stuk voor stuk de inhoud van de adressen D000H t/m D0FFH in de accumulator en controleert aan de hand van CP FFH of de inhoud gelijk is aan FFH. Is dat het geval dan zal er teruggesprongen worden naar de plaats vanwaar deze routine is aangesproken.
Is de waarde van een adres niet gelijk aan FFH dan zal het volgende adres bekeken worden.
Is geen van de adressen gelijk aan FFH dan zal er weer vanaf het begin af aan begonnen worden.
Een gespecificeerde uitleg van de verschillende instructies is waarschijnlijk niet nodig, omdat alle instructies die hier gebruikt zijn allemaal al afzonderlijk besproken zijn. Om alle instructies te begrijpen hoeft U dus alleen maar even terug te bladeren.

Dan zijn we aanbeland bij de instructie ADD. Dit is een instructie om een optelling te doen. Zo bestaat er ook een instructie SBC die een aftrekking doet. We bespreken echter alleen de instructie ADD, omdat de instructie SBC precies het omgekeerde doet.
Met ADD kan een bepaalde waarde of de waarde van een bepaald register bij de waarde van een ander register opgeteld worden. Het register waarbij de waarde opgeteld wordt zal na de optelsom de uitkomst van de optelsom bevatten. Een tweetal voorbeelden:

```
LD HL,2040H
LD DE,1020H
ADD HL,DE
```

Het HL-register zal na uitvoer van dit stukje de waarde 3060H (2040H + 1020H) bevatten. De inhoud van het DE-register wordt dus bij het eerstgenoemde register, het HL-register, opgeteld en de uitkomst komt daarna ook in het eerstgenoemde register.
In BASIC zou deze optelling er als volgt uit komen te zien:

```
HL=$2040 :DE=$1020 :HL=HL+DE
```

Wanneer U hierna PRINT HEX$(HL) invoert, zult U zien dat de waarde 3060 aangegeven zal worden.
Met de instructie ADD kunnen zowel dubbele registers als enkele registers bij elkaar opgeteld worden.
Een voorbeeld met een enkel register:

```
LD A,90H
ADD A,A
```

De accumulator zal na uitvoer van dit stukje de waarde 20H bevatten. 90H+90H levert normaal gesproken de waarde 120H op, een enkel register kan echter maximaal de waarde FFH bevatten en zodoende zal bij waardes hoger dan FFH weer bij 00H begonnen worden en zo komen we aan de uitkomst van 20H.
Bij dubbele registers geldt precies hetzelfde. Wanneer de uitkomst van een optelsom boven FFFFH uit komt, zal er vanaf 0000H doorgeteld worden. Zo kan bijvoorbeeld ook iets van een register afgetrokken worden. Wilt U bijvoorbeeld 40 (28H) van een dubbel register aftrekken dan gaat dat als volgt:

```
LD HL,D050H
LD DE,FFD8H
ADD HL,DE
```

Het HL-register zal hierna de waarde D028H bevatten. Er is dus 28H van het HL-register afgetrokken.
Natuurlijk kan een dergelijke aftreksom ook op de volgende manier:

```
LD HL,D050H
LD DE,0028H
SBC HL,DE
```

Er is echter een speciale reden waarom dit minder vaak door ons gebruikt wordt en daar willen wij U hier niet mee lastig vallen.

De instructie DEFB xx.

De instructie DEFB is ongeveer gelijk aan de instructie DATA in BASIC. Deze instructie wordt gevolgd door maximaal één getal tussen 00H en FFH. Op het adres waar deze instructie staat komt dan de aangegeven waarde te staan.

De instructie LDIR.

Er zijn ook een aantal instructies die meer dan één instructie herbergen, zoals ook al het geval was bij de instructie DJNZ. De instructie LDIR is een goed voorbeeld van meer instructies die samengevoegd zijn tot één instructie.
Wat doet de instructie LDIR precies?
De instructie LDIR laadt de inhoud van het adres dat aangegeven wordt door HL op het adres dat aangegeven wordt door DE. Misschien is dit een beetje omslachtig gezegd. Wat er precies mee bedoeld wordt, wordt in dit voorbeeld duidelijk aangegeven:

```
LD HL,2000H
LD DE,D000H
```

Wanneer hierna het effect van LDIR op dit stukje wordt toegepast, zal de inhoud van adres 2000H geladen worden op het adres D000H. De inhoud van D000H zal dan dus gelijk zijn aan de inhoud van 2000H.
Wat doet de instructie LDIR nog meer.
Deze instructie verhoogt, na de inhoudsverplaatsing, de registers DE en HL met 1. Daarna wordt er van register BC 1 afgetrokken en zolang het BC-register niet gelijk aan nul is, zal de hele cyclus van LDIR van voren af aan herhaald worden. Bedenk wel dat de registers DE en HL in de tussentijd dus met 1 verhoogd zijn.
Deze instructie leent zich dus uitstekend om een geheugenblok te verplaatsen naar een ander geheugenblok. Een voorbeeld:

```
LD HL,0000H
LD DE,D000H
LD BC,0100H
LDIR
RET
```

Na dit korte programmaatje zal op de adressen D000H t/m D100H precies hetzelfde staan als op de adressen 0000H t/m 0100H, omdat het blok van 0000H t/m 0100H verplaatst is naar het eerstgenoemde blok via de instructie LDIR.

De instructies XOR A en OR.

Er is maar één ding dat eigenlijk over de instructie XOR A verteld hoeft te worden en dat is dat deze instructie de accumulator op 0 zet. Hoe dat precies in elkaar steekt is niet van belang. Het enige dat van belang is, is dat deze instructie 1 byte minder geheugenruimte in beslag neemt dan de instructie LD A,00H
De instructie OR, die er ook in voor komt, wordt in de programma's ook een keer gebruikt als OR C en dat in combinatie met de instructie LD A,B die er aan voorafgaat.

De instructie OR vergelijkt twee inhouden. Is één van beide of zijn ze allebei één dan zal de instructie OR als eindresultaat ook één geven. Bij gebruik van LD A,B ; OR C en vervolgens bijvoorbeeld CALL Z,xxxx zal er dus alleen aan de CALL voldaan worden als zowel de inhoud van het register B als van het register C gelijk aan nul is.
In tabelvorm komt OR er als volgt uit te zien:

```
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

Er zijn ook instructies als AND en NOT die ook iets dergelijks doen, maar die worden niet in de programma's gebruikt en zullen dus ook niet nader uitgelegd worden.

Speciale assemblerinstructies.

Er zijn een aantal speciale assemblerinstructies die iets doen met de assemblerlisting. De belangrijksten daarvan zijn ORG en SKP. Er zijn er natuurlijk veel meer, maar hier laten we het bij.
Met de instructie ORG kunt U bepalen waar U een programma(deel) in het geheugen wilt zetten. Wilt U dat het programma vanaf 1200H in het geheugen komt dan geeft U ORG 1200H aan het begin.
De instructie SKP slaat een bepaald aantal regels over in de assemblerlisting. Wanneer U bijvoorbeeld een aantal deelprogramma's in de listing heeft, kunt U ze overzichtelijk uit elkaar houden door de instructie SKP te geven. Achter SKP komt het aantal regels te staan dat U over wilt slaan.
Een voorbeeld van de instructies ORG en SKP.

```
ORG 2000H
LD HL,D000H
LD (HL),C8H
JP TWEE
SKP 2


ORG 2100H
TWEE:  INC HL
LD (HL),C8H
RET
```

In het geheugen komt dan het volgende te staan:

```
2000 21                    2100 23
2001 00                    2101 36
2002 D0                    2102 C8
2003 36                    2103 C9
2004 C8
2005 C3
2006 00
2007 21
```

Deze waardes komen uit de tabel in APPENDIX A.

De instructie CPL.

De laatste instructie die in de programma's voor komt, maar eigenlijk al besproken is, is de instructie CPL. Wat doet deze instructie precies?
De instructie CPL heeft alleen betrekking op de accumulator en keert de waarden van alle bits binnen de accumulator om. Alle bits die nul zijn, worden één en alle bits die één zijn, worden nul.
Een voorbeeld:

A = 00010011
Na de instructie CPL komt A er als volgt uit te zien:
A = 11101100

Een zelfde effect heeft de instructie NEG. Met de instructie AND kan ook iets dergelijks gedaan worden, maar die instructie neemt twee bytes geheugenruimte in beslag.

Enkele instructies die niet in de programma's voor komen.

De instructies BIT, RES en SET.

Dit zijn allemaal instructies die betrekking hebben op bits.
Met de instructie BIT x,y kan gecontroleerd worden of een bit nul of één is. Is een bit één dan zal de Zero-vlag (de vlag die gebruikt wordt in instructies als JR Z,xx e.d.) op 0 gezet worden. Is een bit nul dan zal de Zero-vlag op één gezet worden.
De x in de instructie BIT x,y stelt het te onderzoeken bit voor en de y in de instructie BIT x,y stelt het te onderzoeken register voor.
Een voorbeeld:

BIT 3,A controleert of het derde bit van de accumulator gelijk is aan nul of niet.

Met de instructie SET x,y kan een bepaald bit op één gezet worden en met de instructie RES x,y kan een bepaald bit op nul gezet worden.
De x stelt weer het bit voor en de y stelt weer het register voor.
SET 2,C zal dus bit 2 in het C-register op één zetten.

Ook bij al deze instructies kan de inhoud van het HL-register gebruikt worden.
RES 5,(HL) zal dus bit 5 van de inhoud van het HL-register op nul zetten.

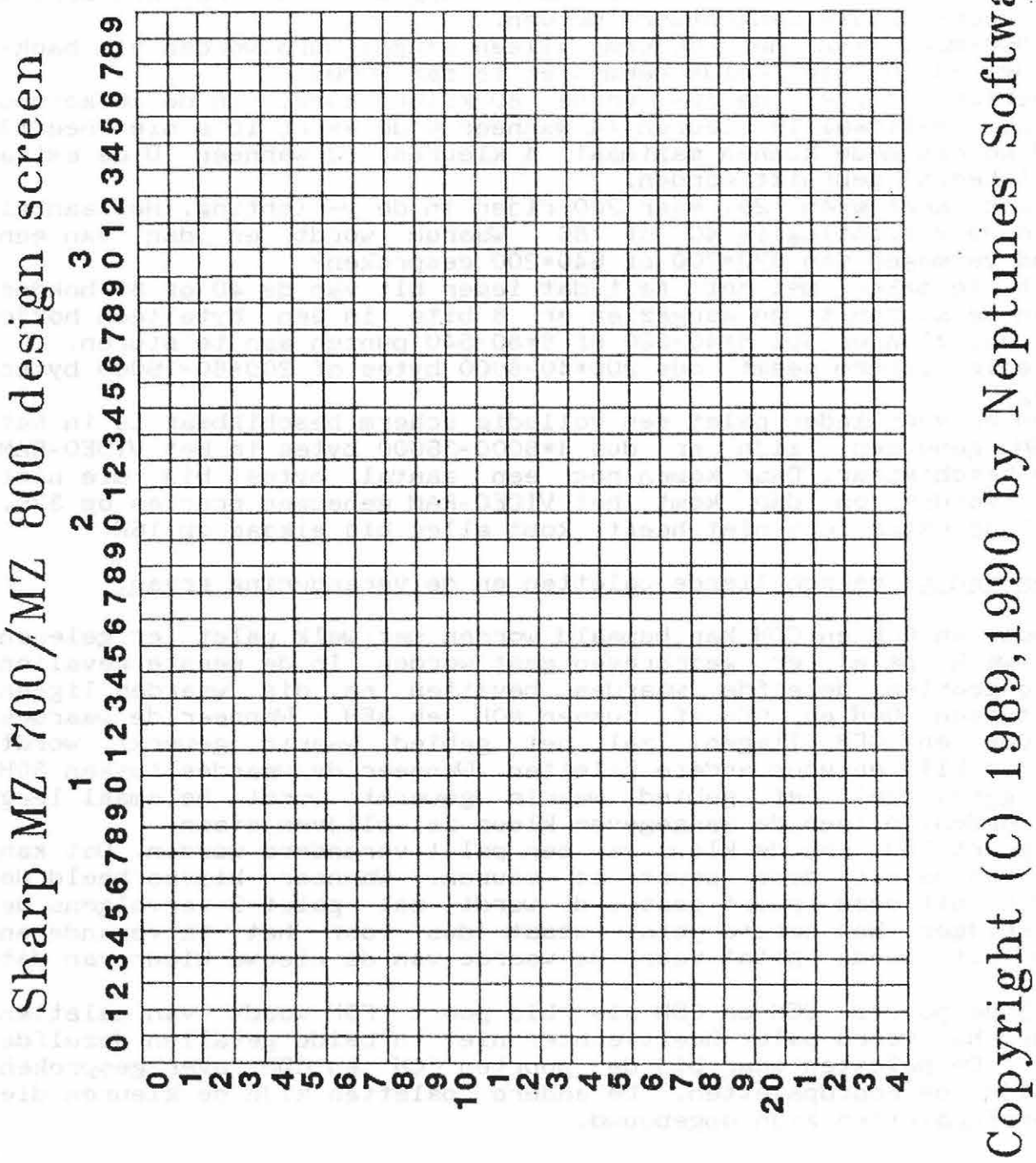
De instructie NOP.

Er wordt wel eens gezegd:"Dat was ook voor noppes." waarmee wordt bedoeld dat het helemaal voor niets was. De instructie NOP doet ook niets. Het wordt echter nog wel eens gebruikt in wachtlussen. Dit heeft te maken met het aantal kloktijden dat deze instructie gebruikt.
Het is namelijk zo dat iedere instructie een aantal kloktijden gebruikt. Kloktijden zijn de tijden die een bepaalde instructie nodig heeft. De instructie NOP gebruikt 14 kloktijden.

Specifieke SHARP-zaken.

Op de volgende pagina's zullen specifieke SHARP-zaken, zoals de beeldschermindeling en de belangrijkste poorten uitgelegd worden. Het grootste deel zal dus een herhaling zijn. Dit deel is dan ook uitsluitend bedoeld om alles overzichtelijk bij elkaar te hebben.

De beeldschermindeling in de 700-MODE.



De tekening van de vorige bladzijde geeft duidelijk aan hoe het beeldscherm in de 700-mode is opgebouwd. Bovenaan staat de x-as en aan de zijkant de y-as. De positie x,y komt overeen met het adres D000H+x+40*y in de ROM-monitor.
De positie 0,0 komt dus overeen met adres D000H+0+40*0 = D000H.
De laatste positie (39,24) op het scherm komt overeen met adres D000H+39+24*40 = D3E7H.
In totaal zijn er dus 25*40=1000 schermposities, dat komt overeen met bijna 1K aan geheugenruimte.

De schermindeling in de 800-mode.

In de 800-mode is het scherm iets anders ingedeeld en kan het scherm ook niet rechtstreeks aangestuurd worden.
In de 800-mode kan het scherm alleen aangestuurd worden via bank-switsching. Het eerste beeldschermadres is dan 8000H.
Er is verschil tussen de 40- en de 80-koloms mode. In de 40-koloms mode kunnen maximaal 16 kleuren (4 wanneer U de extra ic's niet heeft) en de 80-koloms mode kunnen maximaal 4 kleuren (2 wanneer U de extra ic's niet heeft) gebruikt worden.
De 800-mode kent geen 25, maar 200 rijen in de y-richting. Het aantal hokjes in de x-richting is 40 of 80. Waarom wordt er dan van een oplossend vermogen van 320*200 of 640*200 gesproken?
Dit heeft te maken met het feit dat ieder bit van de 40 of 80 hokjes apart aan te sturen is en aangezien er 8 bits in één byte (één hokje dus) zitten, zijn er dus 8*40=320 of 8*80=640 punten aan te sturen.
Een volledig scherm neemt dus 200*40=8000 bytes of 200*80=16000 bytes in beslag.
Aangezien er voor ieder palet een volledig scherm beschikbaar is in het VIDEO-RAM geheugen, zijn er dus 4*8000=36000 bytes in het VIDEO-RAM geheugen beschikbaar. Daar komen nog een aantal bytes bij die niet gebruikt worden en dan komt het VIDEO-RAM geheugen precies op 32K. Wanneer U de extra ic's niet heeft, komt alles bij elkaar op 16K.

Aansturen van de verschillende paletten en de verandering ervan.

Via de poorten CCH en CDH kan bepaald worden met welk palet er gelezen en met welk palet er geschreven gaat worden. In de meeste gevallen zullen de poorten dezelfde waardes bevatten en die waardes liggen meestal tussen 00H en 0FH of tussen 80H en 8FH. Wanneer de waardes tussen 00H en 0FH liggen, zal het gebied waarin gewerkt wordt onveranderd blijven voor andere paletten. Wanneer de waardes tussen 80H en 8FH liggen, zal het gebied waarin gewerkt wordt helemaal leeg gemaakt worden. Alleen de aangegeven kleur zal blijven staan.
Met de poort F0H kan de kleur van een palet veranderd worden. Dat kan door een waarde uit deze poort te sturen. Wanneer bijvoorbeeld de waarde 28H uit deze poort gestuurd wordt, zal palet 2 vervolgens de kleur 8 krijgen. Het eerste getal staat dus voor het te veranderen palet en het tweede getal voor de waarde van de nieuwe kleur van dat palet.
Zowel bij de poorten CCH en CDH als bij poort F0H wordt van paletten gesproken. Het woord palet heeft echter niet in beide gevallen dezelfde betekenis. De paletten waar bij de poorten CCH en CDH over gesproken wordt, zijn de hoofdpaletten. De andere paletten zijn de kleuren die uit de hoofdpaletten zijn opgebouwd.

De joystickpoorten.

Poort F0H wordt ook gebruikt als poort voor joystick 1. Deze poort kan uitgelezen worden en zo kan gekeken worden of er iets met de joystick gedaan is.
Poort F1H wordt voor joystick 2 gebruikt.
De volgende waardes kunnen voor komen:

E5H - Joystick is schuin naar rechts onder bewogen en de vuurknop is ingedrukt.
E6H - Joystick is schuin naar rechts boven bewogen en de vuurknop is ingedrukt.
E7H - Joystick is naar rechts bewogen en de vuurknop is ingedrukt.
E9H - Joystick is schuin naar links onder bewogen en de vuurknop is ingedrukt.
EAH - Joystick is schuin naar links boven bewogen en de vuurknop is ingedrukt.
EBH - Joystick is naar links bewogen en de vuurknop is ingedrukt.
EDH - Joystick is naar onder bewogen en de vuurknop is ingedrukt.
EEH - Joystick is naar boven bewogen en de vuurknop is ingedrukt.
EFH - Alleen de vuurknop is ingedrukt.
F5H - Joystick is schuin naar rechts onder bewogen.
F6H - Joystick is schuin naar rechts boven bewogen.
F7H - Joystick is naar rechts bewogen.
F9H - Joystick is schuin naar links onder bewogen.
FAH - Joystick is schuin naar links boven bewogen.
FBH - Joystick is naar links bewogen.
FDH - Joystick is naar onder bewogen.
FEH - Joystick is naar boven bewogen.
FFH - Joystick is niet aangeraakt.

Allerlei andere poorten.

Poort D2 wordt gebruikt voor de cassetterecorder. Indien de waarde van deze poort 152 is, is een toets op de cassetterecorder ingedrukt.

Poort CE wordt gebruikt voor de geheugen- en schermindeling. De waarde 0 komt overeen met mode M1. De waarde 2 komt overeen met mode M2.

Poort D1 kan gebruikt worden om te controleren of de SHIFT- of CTRL-toets is ingedrukt. Is de waarde 191 dan is de CTRL-toets ingedrukt. Is de waarde 254 dan is de SHIFT-toets ingedrukt.

Poort F2 is de poort voor het geluid.

Poort E0 en E1 zijn bedoeld voor bankswitsching.

Pijltjestoetsen.

Er zijn weinig computers die pijltjestoetsen op de computer hebben zitten als vervanging voor de joystick. De SHARP MZ-800 heeft deze pijltjestoetsen wel en die zijn uit te lezen met behulp van de instructie STICK. Met deze instructie zijn ook de beide joysticks in BASIC uit te lezen. Op de volgende bladzijde vindt U een voorbeeldprogramma voor de pijltjestoetsen.

```
10 INIT"CRT:M1" :A=0
20 CURSOR 0,0 :PRINT A
30 S=STICK(0) :A=A+(1 AND S=3 AND A<250)
-(1 AND S=7 AND A>0)
40 GOTO 20
```

Wanneer U na het RUNnen van dit programma de pijltjestoets naar links of naar rechts drukt, zal de waarde boven in beeld veranderen.

QUICK-DISK.

Er zijn ook slechts weinig computers die gebruik maken van een QUICK-DISK. SHARP maakt er echter wel gebruik van. In de QUICK-DISK passen alleen de zogenaamde 2.8 inch schijfjes, de QD's. Deze zijn er in de merken MAXELL, ROLAND, PHILIPS en nog een ander merk en kosten gemiddeld tussen de F7,50 en F10,= per stuk.

VIDEO-RAM uitbreiding.

Veel computers kunnen uitgebreid worden met een aantal ic's. Bij SHARP bestaat ook die mogelijkheid. Er zijn namelijk twee ic-poortjes open gelaten, waar twee extra VIDEO-RAM ic's in passen. De volgende types ic's kunnen daar voor gebruikt worden:

De TMS4416-N15
De TMS4464-N15

In het handboek staan ook wel types vermeld, maar die zijn vaak stukken duurder dan bovenstaande types.
Bij kleine electronicazaken vindt U af en toe nog wel eens een aantal ic's voor een hele schappelijke prijs die heel wat lager ligt dan de clubprijzen. Voor twee ic's betaalt U bij sommige electronicazaken ongeveer F12,= en bij de clubs ligt dat tussen de F20,= en F25,=.

Plotter.

Er bestaat een speciale vier-kleurenplotter voor de SHARP MZ-800, de MZ-1P16. Deze plotter kan via monitorroutines aangestuurd worden. In de ROM-monitor zijn dus al programma's opgenomen voor de plotter. Er zijn twee hoofdroutines in de ROM-monitor en die beginnen op de adres 018F en 01A5. Via CALL kunnen de routines, die op deze adressen beginnen, aangesproken worden. Bij CALL 018FH moet in de accumulator de waarde staan die U naar de plotter wilt sturen. De waarde van de accumulator kan de waarde van een gewoon teken, maar kan ook een speciale stuurcode voor de plotter zijn. Hieronder staan enkele stuurcodes:

01 - De plotter gaat naar de character mode.
02 - De plotter gaat naar de grafische mode.
04 - Plottertest.
0B - Van 40 naar 26 tekens per regel.
0C - Van 26 naar 40 tekens per regel.
1D - Wisselen van kleur.
09,09,09 - Van 40 naar 80 tekens per regel.
09,09,0B - Van 80 naar 40 tekens per regel.
0E - Backspace.

HOOFDSTUK 10: PROGRAMMA'S.

In dit hoofdstuk staan een aantal leuke programma's van verschillende aard. Het grootste deel bestaat uit spelletjes, maar ook educatieve en andere soorten programma's komen aan de orde. Er zijn niet alleen BASIC-programma's, er is ook een machinetaalprogramma en natuurlijk gecombineerde programma's met zowel BASIC als machinetaal.

Programma 1: STERREN.

Als eerste programma gelijk het gemakkelijkste spel dat te maken is op Uw SHARP. Het is de bedoeling dat U met het vliegtuig tussen de sterren door vliegt. U kunt sturen met de pijltjestoetsen. Met de pijltjestoetsen kunt U ook de snelheid laten toe- en afnemen.
VEEL SUCCES.

```
10 ''''''''''''''''''''''''''''
20 '      ** STERREN **       '
30 '       (C) 1989 door      '
40 '    NEPTUNES SOFTWARE     '
50 ''''''''''''''''''''''''''''
60 INIT"CRT:M1" :CONSOLE 1,24 :TI$="0000
00" :T=0 :B=20 :CURSOR 0,0 :PRINT"SCORE:
" :A$="*  *" :OUT@$F2,$E2 :OUT@$F2,$F1
70 IF PEEK($2028+B)<>0 THEN 100
80 CURSOR 6,0 :PRINT TI$ :CURSOR B,1 :PO
KE $5D4,1 :PRINT CHR$(223) :POKE $5D4,0
:IF STICK(0)<>0 THEN S=STICK(0)
90 B=B+(1 AND S=3 AND B<39)-(1 AND S=7 A
ND B>0) :WAIT T :T=T+(1 AND S=5 AND T<25
)-(1 AND S=1 AND T>0) :CURSOR RND*37,24
:PRINT A$ :GOTO 70
100 CLS :OUT@$F2,$FF :PRINT"JE SCORE WAS
:";VAL(TI$);" PUNTEN" :IF VAL(TI$)>Z THE
N Z=VAL(TI$)
110 PRINT"DE HOOGSTE SCORE IS:";Z;" PUNT
EN" :WAIT 2000
120 GET A$ :IF A$="" THEN 120 ELSE 60
```

Programma 2: REKENEN.

Nadere uitleg staat in het programma zelf.

```
10 ''''''''''''''''''''''''''''
20 '          REKENEN         '
30 '       (C) 1989 door      '
40 '    NEPTUNES SOFTWARE     '
50 ''''''''''''''''''''''''''''
60 INIT"CRT:M1" :PAL 2,6 :SYMBOL [2]47,0
,"REKENEN",4,3 :BOX [1]47,24,273,25 :SYM
BOL 52,32,"(C)1989 door",2,2 :SYMBOL 24,
52,"NEPTUNES SOFTWARE",2,3
70 CURSOR 0,10 :PRINT [2]"Dit is een rek
enprogramma in zijn een-  voudigste vorm
. Je kunt OPTELLEN,       AFTREKKEN, DEL
```

```
EN, VERMENIGVULDIGEN en   MACHTSVERHEFFE
N."
80 PRINT [2]"Een antwoord kan goed of fo
ut zijn, dat geeft de computer voor U aa
n. Er worden verder geen tabellen van go
ede of foute antwoorden bijgehouden. Het
 programma   is dus eigenlijk bedoeld om
 Uw reken-"
90 PRINT [2]"vaardigheid op peil te houd
en.          Mocht U het antwoord niet w
eten, druk   dan op 'W', dan krijgt U he
t antwoord.  Bij dit programma hoeft U g
een gebruik  te maken van de CR-toets. A
lles gaat    vanzelf."; :WAIT 24000
100 E=ASC("=") :CLS :PAL 0,6 :PAL 3,0
110 PRINT"MENU"
120 PRINT :PRINT"(1) OPTELLEN" :PRINT :P
RINT"(2) AFTREKKEN" :PRINT :PRINT"(3) DE
LEN" :PRINT :PRINT"(4) VERMENIGVULDIGEN"
 :PRINT :PRINT"(5) MACHTSVERHEFFEN"
130 GET A$ :IF A$="" OR A$<"1" OR A$>"5"
 THEN 130
140 CLS :A=ASC(A$)-48 :ON A GOSUB 150,19
0,230,270,300
150 A=INT(RND*99+1) :B=INT(RND*99+1) :D=
ASC("+") :C=A+B :GOSUB 340
160 IF B$="W" THEN 180
170 GOSUB 480
180 CLS :GOSUB 440 :GOTO 150
190 A=INT(RND*99+1) :B=INT(RND*99+1) :IF
 B>A THEN 190
200 D=ASC("-") :C=A-B :GOSUB 340 :IF B$=
"W" THEN 220
210 GOSUB 480
220 CLS :GOSUB 440 :GOTO 190
230 B=INT(RND*20+1) :A=INT(RND*20+1)*B :
IF A>=100 THEN 230
240 D=ASC("/") :C=A/B :GOSUB 340 :IF B$=
"W" THEN 260
250 GOSUB 480
260 CLS :GOSUB 440 :GOTO 230
270 B=INT(RND*20+1) :A=INT(RND*20+1) :D=
ASC("*") :C=A*B :GOSUB 340 :IF B$="W" TH
EN 290
280 GOSUB 480
290 CLS :GOSUB 440 :GOTO 270
300 A=INT(RND*10+1) :B=INT(RND*4+1) :IF
A↑B>999 THEN 300
310 D=ASC("↑") :C=A↑B :GOSUB 340 :IF B$=
"W" THEN 330
320 GOSUB 480
330 CLS :GOSUB 440 :GOTO 300
340 SYMBOL 20,90,STR$(A),4,4 :SYMBOL 80,
90,CHR$(D),4,4 :SYMBOL 110,90,STR$(B),4,
4 :SYMBOL 170,90,CHR$(E),4,4
```

```
350 C$=STR$(C) :F=0 :D$=""
360 FOR T=1 TO LEN(C$)
370 GET A$ :IF A$="" THEN 370
380 B$=A$ :IF T>1 THEN 400
390 IF B$="W" THEN A$=STR$(C) :SYMBOL 20
0,90,A$,4,4 :WAIT 2500 :RETURN
400 IF A$<"0" OR A$>"9" THEN 370
410 D$=D$+A$ :SYMBOL 200+F,90,A$,4,4 :F=
F+30 :NEXT :WAIT 1000
420 IF D$<>C$ THEN CLS :SYMBOL 40,90,"FO
UT !!!",4,4,0 :WAIT 1000 :CLS :GOTO 340
430 RETURN
440 CLS :SYMBOL 35,95,"NOG EEN KEER ?",2
,2
450 GET A$ :IF A$="" THEN 450
460 IF A$="J" THEN CLS :RETURN
470 IF A$<>"N" THEN 450 ELSE 100
480 CLS :SYMBOL 40,90,"GOED !!!",4,4 :WA
IT 1000 :RETURN
```

Programma 3: FLYING.

Bij dit spel is het de bedoeling dat U met Uw 'raket' probeert te voorkomen dat er vier bommen op één plaats door de bodem gaan. U kunt zowel de bommen als de vliegtuigen kapot schieten. U kunt schieten met de spatiebalk en bewegen met de cursortoetsen.
VEEL PLEZIER.

```
10 IF PEEK($4DCF)>1 THEN PK=$4DD0 ELSE P
K=$4DCF
20 INIT"CRT:M1" :INPUT"SNELHEID (1-100)"
;WI :IF WI>100 OR WI<1 THEN 20
30 PAL 2,1 :PAL 1,3 :PAL 0,7 :PAL 3,0
40 A$=CHR$(198)+" " :SC=0
50 POKE $5D4,1 :CURSOR 0,0 :PRINT [1]"FL
YING    "; :PRINT CHR$(152); :POKE $5D4,0
 :PRINT"1989/1990 by N-S  SCORE:";SC
60 LINE [2]0,8,319,8
70 FOR T=21 TO 23 :FOR T1=0 TO 39 :CURSO
R T1,T :PRINT [2]CHR$(200); :NEXT T1 :LI
NE [0]0,T*8+7,319,T*8+7 :NEXT T
80 PAINT [3]0,195,2 :POKE $5D4,1
90 X=19 :X1=19 :V=39 :K=1 :P=0
100 D=INT(RND(1)*38+1) :J=0
110 W=INT(RND(1)*13+2)
120 CURSOR X1,20 :PRINT" "; :X1=X
130 CURSOR X,20 :PRINT CHR$(202);
140 CURSOR V,W :PRINT A$;
150 IF K>1 THEN H=H+1 :GOSUB 310
160 S=STICK(0)
170 X=X-(S=3 AND X<39)+(S=7 AND X>0)
180 SR=STRIG(0)
190 IF SR=0 THEN 240 ELSE E=X1*8+3
200 IF X1=V AND P=0 THEN LINE [1]E,158,E
,W*8+9 :GOTO 530
```

```
210 IF X1=D AND P=0 AND K=2 THEN LINE [1
]E,158,E,H*8+9 :GOSUB 560 :GOTO 230
220 IF P=0 THEN LINE [1]E,158,E,10 :WAIT
 10 :LINE [0]E,158,E,10
230 IF P=0 THEN P=1
240 IF P>0 THEN P=P+1 :IF P=4 THEN P=0
250 V=V-1 :IF V=-1 THEN CURSOR 0,W :PRIN
T" ";  :V=39 :WI=WI-.5 :GOSUB 280
260 IF V=D AND J=0 THEN K=2 :H=W+1 :J=1
270 WAIT WI :IF X1<>X THEN 120 ELSE 130
280 IF WI<0 THEN WI=0
290 IF J=1 AND K=1 THEN J=0
300 IF K=1 THEN 100 ELSE 110
310 IF H=24 THEN POKE PK,1 :GOTO 520
320 CURSOR D,H-1 :PRINT" ";
330 YT=PEEK($2000+D+H*40)
340 IF YT=200 THEN K=1 :CURSOR D,H :PRIN
T" ";  :RETURN
350 IF YT=202 THEN 370
360 CURSOR D,H :PRINT CHR$(91);  :RETURN
370 CURSOR X,20 :PRINT [1]CHR$(141);
380 POKE PK,1 :SYMBOL [1]X*8-8,20*8-8,CH
R$(188),3,3
390 OUT@$F2,$C8 :FOR A=$F0 TO $FF :OUT@$
F2,A :WAIT 200 :NEXT A
400 FOR A=1 TO 250 :PAL 0,INT(RND(1)*16)
 :PAL 2,INT(RND(1)*16) :NEXT A
410 PAL 0,7 :PAL 2,1
420 BOX 107,70,213,130,1
430 SYMBOL [3]113,75,"GAME",3,3
440 SYMBOL [3]113,102,"OVER",3,3
450 PAINT [0]108,71,3 :PAINT [0]123,116,
3
460 POKE PK,0 :BOX 102,135,218,145,2
470 SYMBOL [0]105,137,"NOG EEN KEER ?",1
,1
480 POKE $5D4,0
490 GET A$ :IF A$="J" THEN RUN
500 IF A$<>"N" THEN 490
510 CLS :PRINT"TOT ZIENS" :PRINT" GOODBY
E" :PRINT"  AUF WIEDERSEHEN" :PRINT"   A
U REVOIR" :END
520 CURSOR D,H-1 :PRINT" ";  :CURSOR D,H
:PRINT CHR$(91);  :GOTO 400
530 POKE $5D4,0 :SC=SC+10 :CURSOR 34,0 :
PRINT SC :POKE $5D4,1
540 IF J=1 AND K=1 THEN J=0
550 CURSOR V,W :PRINT CHR$(141);  :WAIT 9
99 :CURSOR V,W :PRINT" ";  :LINE [0]E,160
,E,W*8+8 :V=39 :IF J=0 THEN 100 ELSE 110
560 POKE $5D4,0 :SC=SC+5 :CURSOR 34,0 :P
RINT SC :POKE $5D4,1
570 CURSOR D,H :PRINT CHR$(141);  :WAIT 9
99 :CURSOR D,H :PRINT" ";  :LINE [0]E,160
,E,H*8+8 :J=1 :K=1 :RETURN
```

Programma 4: Speciale characterset.

In eerste instantie was dit programma gemaakt voor SHARP Computer Club Elst en is gemaakt onder de oude naam SFF-SOFT. (Tegenwoordig dus Neptunes Software.)
Omdat het programma er zo aardig uit zag zijn er geen veranderingen aangebracht in het originele programma en wordt het originele programma hieronder volledig getoond.
Het is wel een klein beetje typwerk, maar het resultaat is heel aardig. Alle hoofdletters worden schuine letters en er worden twee logo's gevormd, één van SFF-SOFT en één van SCCE.
Ook hierbij kunt U ten alle tijden de oude characterset terug krijgen door het volgende in te voeren:

```
POKE $5DF,$29,$CB,$E4
```

```
10 'SPECIALE SCCE/SFF-SOFT CHARACTERSET
20 '@1988 door SFF-SOFT
30 ' 0000000000
40 ' 0000000000
50 ' 0000000000
60 ' 0000000000
70 ' 0000
80 LIMIT $E000 :POKE $F000,$1,$0,$10,$11
,$0,$E0,$21,$0,$10,$DB,$E0,$ED,$B0,$DB,$
E1
90 POKE $F00F,$3E,$C3,$21,$1B,$F0,$32,$D.
F,$5,$22,$E0,$5,$C9,$29,$CB,$EC,$CB,$F4,
$CB,$FC,$C3,$E2,$5
100 USR($F000)
110 POKE $4DD6,$E0
120 'A :DATA 120,68,68,62,34,17,17,0
130 'B :DATA 56,68,68,62,66,33,31,0
140 'C :DATA 56,68,4,2,2,17,15,0
150 'D :DATA 56,68,68,34,34,17,15,0
160 'E :DATA 120,4,4,30,2,1,15,0
170 'F :DATA 120,4,4,30,2,1,1,0
180 'G :DATA 56,68,4,50,34,17,15,0
190 'H :DATA 72,36,36,30,18,9,9,0
200 'I :DATA 120,16,16,8,8,4,15,0
210 'J :DATA 112,32,16,16,8,9,6,0
220 'K :DATA 72,36,20,14,10,9,17,0
230 'L :DATA 16,8,8,4,4,2,62,0
240 'M :DATA 72,52,36,18,18,9,9,0
250 'N :DATA 72,36,44,26,18,9,9,0
260 'O :DATA 120,68,34,34,17,17,15,0
270 'P :DATA 56,68,68,62,2,1,1,0
280 'Q :DATA 120,68,34,42,25,25,47,0
290 'R :DATA 56,68,68,62,10,17,17,0
300 'S :DATA 56,72,4,12,16,17,14,0
310 'T :DATA 124,8,8,4,4,2,2,0
320 'U :DATA 68,68,34,34,17,17,15,0
330 'V :DATA 65,33,19,9,5,3,1,0
340 'W :DATA 65,65,33,33,21,27,9,0
350 'X :DATA 72,40,24,8,12,10,9,0
```

```
360 'Y :DATA 72,40,24,8,4,2,1,0
370 'Z :DATA 124,32,16,8,4,2,31,0
380 FOR A=1 TO 26*8
390 READ B
400 POKE A+$E007,B
410 NEXT A
420 '@ :DATA $3C,$42,$99,$85,$85,$99,$42
,$3C
430 FOR A=1 TO 8
440 READ B
450 POKE A+$E2A7,B
460 NEXT A
470 '0 :DATA 120,100,50,42,21,19,15,0
480 '1 :DATA 48,40,16,16,8,8,30,0
490 '2 :DATA 24,36,32,16,8,4,30,0
500 '3 :DATA 56,68,32,56,16,17,14,0
510 '4 :DATA 16,40,36,126,16,8,8,0
520 '5 :DATA 124,4,2,12,16,8,7,0
530 '6 :DATA 96,16,8,12,18,17,14,0
540 '7 :DATA 124,32,16,62,4,2,1,0
550 '8 :DATA 120,68,68,62,34,17,15,0
560 '9 :DATA 56,68,36,24,8,4,3,0
570 FOR A=1 TO 10*8
580 READ B
590 POKE A+$E0FF,B
600 NEXT A
610 'SFF-SOFT EMBLEEM
620 DATA 255,1,249,13,5,229,21,21
630 DATA 255,0,255,0,0,124,5,4
640 DATA 255,0,255,0,0,31,1,1
650 DATA 255,128,159,176,160,160,160,160
660 DATA 229,5,21,229,5,69,165,21
670 DATA 28,5,5,4,0,0,0,1
680 DATA 199,1,1,1,0,0,0,255
690 DATA 167,160,160,160,160,160,160,160
700 DATA 21,21,165,69,5,245,21,21
710 DATA 193,33,160,144,144,17,16,16
720 DATA 0,0,195,66,195,0,0,36
730 DATA 163,164,165,169,169,168,168,168
740 DATA 117,21,21,21,5,245,69,69
750 DATA 16,144,16,32,32,193,0,0
760 DATA 0,0,129,126,0,0,255,0
770 DATA 168,169,168,164,164,163,160,160
780 DATA 69,69,69,69,13,249,1,255
790 DATA 0,0,0,0,0,255,0,255
800 DATA 160,160,160,160,176,159,128,255
810 FOR A=1 TO 19*8
820 READ B
830 POKE A+$E63F,B
840 NEXT A
850 'SCCE EMBLEEM
860 DATA 192,48,8,12,18,34,65,129
870 DATA 255,0,192,32,32,192,0,32
880 DATA 255,0,1,2,0,1,2,2
890 DATA 3,12,16,43,72,68,130,129
```

```
900 DATA 1,1,1,1,249,9,9,57
910 DATA 193,2,252,172,84,172,84,172
920 DATA 129,64,63,42,53,42,53,42
930 DATA 128,128,128,128,142,145,129,129
940 DATA 9,9,249,1,1,1,1,1
950 DATA 84,172,84,172,84,252,2,193
960 DATA 53,42,53,42,53,63,64,129
970 DATA 129,145,142,128,128,128,128,128
980 DATA 129,65,34,18,12,8,48,192
990 DATA 32,32,32,32,32,192,0,255
1000 DATA 2,0,0,0,2,1,0,255
1010 DATA 129,130,68,72,48,16,12,3
1020 FOR A=1 TO 16*8
1030 READ B
1040 POKE A+$E537,B
1050 NEXT A
1060 'SFF-SOFT EMBLEEM
1070 CLS :PRINT :PRINT TAB(18)CHR$(97,98
,99,100)
1080 PRINT TAB(18)CHR$(101,102,103,104)
1090 PRINT TAB(18)CHR$(112,113,114,115)
1100 PRINT TAB(18)CHR$(116,117,118,119)
1110 PRINT TAB(18)CHR$(120,121,121,122)
1120 WAIT 2000 :PAL 0,7 :PAL 3,0 :CLS
1130 SYMBOL 112,10,CHR$(97,98,99,100),3,
3
1140 SYMBOL 112,34,CHR$(101,102,103,104)
,3,3
1150 SYMBOL 112,58,CHR$(112,113,114,115)
,3,3
1160 SYMBOL 112,82,CHR$(116,117,118,119)
,3,3
1170 SYMBOL 112,106,CHR$(120,121,121,122
),3,3
1180 'SCCE EMBLEEM
1190 WAIT 2000 :CLS :PAL 0,0 :PAL 3,7
1200 PRINT TAB(18)CHR$(180,181,182,174)
1210 PRINT TAB(18)CHR$(173,186,178,185)
1220 PRINT TAB(18)CHR$(168,177,131,136)
1230 PRINT TAB(18)CHR$(141,134,132,137)
1240 WAIT 2000 :PAL 0,7 :PAL 3,0 :CLS
1250 SYMBOL 112,10,CHR$(180,181,182,174)
,3,3
1260 SYMBOL 112,34,CHR$(173,186,178,185)
,3,3
1270 SYMBOL 112,58,CHR$(168,177,131,136)
,3,3
1280 SYMBOL 112,82,CHR$(141,134,132,137)
,3,3
1290 WAIT 2000 :CLS
1300 ' POKE EMBLEEMS IN REM-REGELS:
1310 POKE $A4BC,$61,$62,$63,$64,$20,$20,
$B4,$B5,$B6,$AE
1320 POKE $A4CE,$65,$66,$67,$68,$20,$20,
$AD,$BA,$B2,$B9
```

```
1330 POKE $A4E0,$70,$71,$72,$73,$20,$20,
$A8,$B1,$83,$88
1340 POKE $A4F2,$74,$75,$76,$77,$20,$20,
$8D,$86,$84,$89
1350 POKE $A504,$78,$79,$79,$7A
1360 LIST -70
```

Om dit programma in te typen, hebben we ongeveer 50 minuten nodig gehad. Wanneer U wat minder snel kunt typen, zult U er iets meer tijd voor nodig hebben.

Programma 5: Klok.

Twee klokken in één. De tijd wordt op twee manieren weergegeven: Met een klok en ook nog in cijfers.

```
10 INIT"CRT:M1" :INPUT"TIJD (HHMMSS)";A$
 :IF LEN(A$)<>6 THEN 10
20 AB=VAL(MID$(A$,1,2)) :CD=VAL(MID$(A$,
3,2)) : ' AB=UREN , CD=MINUTEN
30 EF=VAL(MID$(A$,5,2)) :U=2*π-π/6*AB-π/
360*CD : ' EF=SECONDEN
40 CLS :PAL 0,1 :CIRCLE 100,100,95 :CIRC
LE 100,100,5
50 SYMBOL 92,14,"12",1,1 :SYMBOL 179,97,
"3",1,1
60 SYMBOL 96,179,"6",1,1 :SYMBOL 14,97,"
9",1,1
70 P=π/30 :FOR Z=2*π TO P STEP -P :Y=SIN
(Z)*95 :X=COS(Z)*95 :W=SIN(Z)*92
80 V=COS(Z)*92 :R=R+1 :IF (R-1)/5=INT((R
-1)/5) THEN W=SIN(Z)*88 :V=COS(Z)*88
90 LINE 100-Y,100-X,100-W,100-V :NEXT Z
:TI$=A$
100 T$=MID$(TI$,1,2)+":"+MID$(TI$,3,2)+"
:"+MID$(TI$,5,2) :A$=T$ :B$=TI$
110 FOR F=2*π-P*CD TO π/60 STEP -P :FOR
A=2*π-P*EF TO π/60 STEP -P
120 BEEP :CIRCLE 100,100,5 :LINE [0]100,
100,100-D,100-E
130 B=SIN(A)*75 :C=COS(A)*75 :LINE 100,1
00,100-B,100-C :D=B :E=C
140 G=SIN(F)*50 :H=COS(F)*50 :LINE 100,1
00,100-G,100-H :I=G :J=H
150 K=SIN(U)*30 :L=COS(U)*30 :LINE 100,1
00,100-K,100-L :M=K :N=L
160 IF Q=0 THEN 190
170 IF A<>2*π THEN 190
180 U=U-π/360 :LINE [0]100,100,100-M,100
-L
190 Q=1 :GOTO 230
200 T$=MID$(TI$,1,2)+":"+MID$(TI$,3,2)+"
:"+MID$(TI$,5,2)
210 SYMBOL [0]190,5,A$,2,2 :A$=T$ :B$=TI
$ :SYMBOL 190,5,T$,2,2 :NEXT A
```

```
220 EF=0 :LINE [0]100,100,100-I,100-J :N
EXT F :CD=0 :GOTO 110
230 IF B$<>TI$ THEN 200 ELSE 230
```

Programma 6: SHOOT DOWN.

Een heel eenvoudig spel waarbij het de bedoeling is dat alle blokken verdwenen zijn, voordat U bij de blokken komt. U kunt blokken vernietigen door er recht boven een bom te laten vallen en dat kunt U doen door op de spatiebalk te drukken.

```
10 INIT"CRT:M1" :CURSOR 1,0 :PRINT"SCORE
:00000   SHOOT DOWN   (C)1989 N-S"
20 W=4 :T=0 :Z=0 :FOR A=1 TO 38 :H=INT(R
ND(1)*7+1) :T=T+H+1 :FOR B=23 TO 23-H ST
EP -1
30 CURSOR A,B :PRINT CHR$(204); :NEXT B,
A :A=0 :B=1
40 IF A>0 THEN CURSOR A-1,B :PRINT" "
50 CURSOR A,B :PRINT">" :IF Z=1 THEN 70
60 IF STRIG(0)=1 THEN K=A :L=B+1 :Z=1
70 IF POINT(K*8,L*8+3)=3 THEN S=S+1 :T=T
-1 :W=W-1 :IF W=0 THEN W=4 :Z=0 :CURSOR
K,L :PRINT" "; :CURSOR K,L-1 :PRINT" "
80 IF Z=1 THEN CURSOR K,L-1 :PRINT" " :C
URSOR K,L :PRINT"*"; :L=L+1 :SOUND 20,2
90 IF T=0 THEN K=0 :L=0 :GOTO 10
100 IF L=24 THEN Z=0 :CURSOR K,L-1 :PRIN
T" " :W=4
110 WAIT 50 :IF B=24 THEN 150
120 A=A+1 :IF A=39 THEN CURSOR A-1,B :PR
INT" " :B=B+1 :A=0
130 IF POINT(A*8,B*8+3)=3 THEN 150
140 S$=STR$(S) :CURSOR 12-LEN(S$),0 :PRI
NT S$; :GOTO 40
150 FOR B=0 TO 3 :FOR A=0 TO 15 :PAL 3,A
 :PAL 0,15-A :WAIT 150 :NEXT A,B
160 INIT"CRT:M1" :SYMBOL 10,5,"UW SCORE
WAS",3,3,0 :S$=S$+" PUNTEN" :SYMBOL 30,4
0,S$,3,3,0
170 IF STRIG(0)=0 THEN 170
180 S=0 :K=0 :L=0 :RUN
```

Programma 7: DOBBELSTENEN.

Dit programma simuleert het werpen met 1 dobbelsteen. Hoe vaak een getal gegooid is is achteraf af te lezen in een grafiek.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0 :PRINT
"DIT PROGRAMMA VRAAGT U EERST HOEVEEL
 KEER U WILT GOOIEN. DAARNA GOOIT DE COM
-PUTER ZOVEEL KEER MET 1 DOBBELSTEEN EN"
20 PRINT"MAAKT EEN GRAFIEK VAN HET AANTA
L KEER   DAT EEN BEPAALD AANTAL OGEN IS
GEGOOID."
```

```
30 PRINT"DE LIJN DIE U ZIET IS DE LIJN V
AN HET   GEMIDDELDE DAT GEGOOID IS. DEZE
 LIJN    LIGT DUS ALTIJD TUSSEN 1 EN 6."
 :PRINT :PRINT :PRINT"          DRUK OP
EEN TOETS."
40 GET A$ :IF A$="" THEN 40
50 DIM K(100,6)
60 CLS :INPUT"HOEVEEL KEER WILT U GOOIEN
? (1-100)";A :IF A>100 OR A<1 THEN 60
70 CLS :BOX 80,30,230,150 :BOX 75,26,235
,154 :PAINT 79,29 :FOR T=1 TO A :X=INT(R
ND(1)*6+1) :K(T,X)=1 :CURSOR 14,20 :PRIN
T"WORP";T;":";X :ON X GOSUB 130,140,150,
160,170,180
80 WAIT 1500 :BOX [0]81,31,229,149,0 :NE
XT
90 CLS :PAL 2,4 :FOR E=190 TO 0 STEP -5
:LINE 0,E,5,E :LINE 314,E,319,E :NEXT E
:LINE 0,190,319,190 :F=1 :FOR T=32 TO 31
5 STEP 50 :SYMBOL T,192,STR$(F),1,1,0 :F
=F+1 :NEXT T
100 L=1 :Y=0 :O=0 :FOR G=10 TO 309 STEP
50 :R=0
110 FOR H=1 TO A :R=R+K(H,L) :Y=Y+L*K(H,
L) :O=O+K(H,L) :NEXT H :BOX G,190-R*5,G+
50,190,2 :L=L+1 :NEXT G :P=190-Y/O*5 :LI
NE 10,P,309,P
120 GET A$ :IF A$="" THEN 120 ELSE 60
130 BOX 150,85,160,95,3 :RETURN
140 BOX 110,52,120,62,3 :BOX 190,117,200
,127,3 :RETURN
150 BOX 110,52,120,62,3 :BOX 150,85,160,
95,3 :BOX 190,117,200,127,3 :RETURN
160 BOX 110,52,120,62,3 :BOX 190,52,200,
62,3 :BOX 190,117,200,127,3 :BOX 110,117
,120,127,3 :RETURN
170 BOX 150,85,160,95,3 :BOX 110,52,120,
62,3 :BOX 190,52,200,62,3 :BOX 190,117,2
00,127,3 :BOX 110,117,120,127,3 :RETURN
180 BOX 110,52,120,62,3 :BOX 190,52,200,
62,3 :BOX 190,85,200,95,3 :BOX 190,117,2
00,127,3 :BOX 110,117,120,127,3 :BOX 110
,85,120,95,3 :RETURN
```

Programma 8: MASTERMIND.

Dit spel is bij iedereen bekend. Verdere instructies vindt U in het programma zelf.

```
10 INIT"CRT:M1" :PAL 0,7 :PAL 3,0 :SYMBO
L 25,90,"INSTRUCTIES (J/N)",2,2,0
20 GET A$ :IF A$="J" THEN 500
30 IF A$<>"N" THEN 20
40 DIM A(4) :DIM D(4)
50 FOR T=1 TO 4 :A(T)=T/10 :NEXT T
```

```
60 CLS
70 FOR T=1 TO 4
80 A(T)=INT(RND(1)*9+1) :IF A(4)=A(1) OR
 A(3)=A(1) OR A(2)=A(1) OR A(4)=A(2) OR
A(3)=A(2) OR A(4)=A(3) THEN 210
90 NEXT T
100 PAL 2,3 :FOR X=0 TO 8 :LINE [2]104,X
,184,X :NEXT X :X=2 :FOR T=1 TO 11 :Y=13
 :FOR P=1 TO 4 :CURSOR Y,X :PRINT"."
110 GET A$ :IF A$<"1" OR A$>"9" THEN GOT
O 110
120 D(P)=ASC(A$)
130 IF D(4)=D(1) OR D(3)=D(1) OR D(2)=D(
1) OR D(4)=D(2) OR D(3)=D(2) OR D(4)=D(3
) THEN 230
140 CURSOR Y,X :PRINT A$ :Y=Y+3
150 NEXT P
160 GOSUB 270
170 X=X+2 :FOR R=1 TO 4 :D(R)=0 :NEXT R,
T
180 CURSOR 12,0 :FOR T=1 TO 4 :PRINT A(T
);" "; :WAIT 1000 :NEXT T
190 IF W=1 THEN RETURN
200 WAIT 4000 :RUN 40
210 IF A(T)<>T/10 THEN A(T)=T/10 :GOTO 8
0
220 GOTO 90
230 IF P=1 THEN 140
240 IF D(1)=D(2) THEN 110
250 IF D(3)>0 THEN 110
260 GOTO 140
270 K=0 :L=0
280 IF D(1)-48=A(1) THEN K=K+1
290 IF D(2)-48=A(2) THEN K=K+1
300 IF D(3)-48=A(3) THEN K=K+1
310 IF D(4)-48=A(4) THEN K=K+1
320 IF D(1)-48=A(2) THEN L=L+1
330 IF D(3)-48=A(2) THEN L=L+1
340 IF D(4)-48=A(2) THEN L=L+1
350 IF D(1)-48=A(3) THEN L=L+1
360 IF D(2)-48=A(3) THEN L=L+1
370 IF D(4)-48=A(3) THEN L=L+1
380 IF D(2)-48=A(1) THEN L=L+1
390 IF D(3)-48=A(1) THEN L=L+1
400 IF D(4)-48=A(1) THEN L=L+1
410 IF D(1)-48=A(4) THEN L=L+1
420 IF D(2)-48=A(4) THEN L=L+1
430 IF D(3)-48=A(4) THEN L=L+1
440 Y=Y+3 :CURSOR Y,X
450 IF K>0 THEN FOR I=1 TO K :PRINT"O";
:PAINT [3]Y*8+4,X*8+5,3 :Y=Y+1 :NEXT I
460 IF L>0 THEN FOR I=1 TO L :PRINT"O";
:NEXT I
470 IF K=4 THEN 490
480 RETURN
```

```
490 PAL 1,1 :S$="U HEEFT HET IN "+STR$(T
)+" BEURTEN GERADEN." :W=1 :GOSUB 180 :S
YMBOL [1]25,190,S$,1,1,0 :WAIT 4000 :RUN
 40
500 INIT"CRT:M1" :SYMBOL 45,10,"INSTRUCT
IES:",2,2,0 :LINE 44,26,236,26
510 CURSOR 0,4 :PRINT"DE COMPUTER KIEST
VIER WILLEKEURIGE GE- TALLEN. HET IS DE
BEDOELING DAT U DIE   VIER GETALLEN IN M
AXIMAAL 11 BEURTEN    RAADT. "
520 PRINT"GEEN VAN DE VIER GETALLEN IS H
ETZELFDE. KIES DUS VERSCHILLENDE GETALLE
N." :PRINT"BETEKENIS VAN DE RONDJES:"
530 PRINT"EEN GESLOTEN RONDJE BETEKENT D
AT 1 VAN  DE VIER GETALLEN OP DE GOEDE P
LAATS     STAAT."
540 PRINT"EEN OPEN RONDJE BETEKENT DAT 1
 VAN DE   VIER GETALLEN GOED IS, MAAR NI
ET OP DE  GOEDE PLAATS STAAT."
550 PRINT :PRINT"                !!VEEL SUCC
ES!!                           DRUK OP EEN
TOETS"
560 GET Z$ :IF Z$="" THEN 560 ELSE 40
```

Programma 9: TYPY.

Een spel dat bedoeld is om Uw typvaardigheid in combinatie met Uw reactievermogen te verhogen. De instructies krijgt U te zien wanneer U het programma runt.

```
1 INIT"CRT:M1" :BOX [2]90,135,237,151 :P
AINT [2]91,136 :SYMBOL 96,140,"DRUK OP E
EN TOETS",1,1 :SYMBOL 100,10,"TYPY",4,4
2 FOR T=41 TO 43 :LINE [1]100,T,230,T :N
EXT
3 CURSOR 0,7 :PRINT"ER KOMT LANGZAAM EEN
 WOORD IN BEELD. HETIS DE BEDOELING DAT
U DIT WOORD INTYPT  OP DE COMPUTER, VOOR
DAT HET BEGIN VAN   HET WOORD DE LIJN RA
AKT." :T=0
4 PRINT"ALS DE EERSTE LETTER INGETYPT IS
, VER-  DWIJNT DEZE LETTER. DE VOLGENDE
LETTER  WORDT DAN DE EERSTE LETTER."
5 PRINT"N.B. NA ELKE TIEN WOORDEN KOMT D
E STREEPJETS DICHTER BIJ HET EIND!"
6 T=T+1 :IF T=50 THEN PAL 2,0
7 IF T=100 THEN PAL 2,2 :T=0
8 GET F$ :IF F$="" THEN 6
9 DIM A$(81,1) :FOR T=1 TO 81 :READ A$(T
,1) :NEXT T
10 F=230 :SC=-1 :AT=-1
11 INIT"CRT:M1" :PAL 1,6 :PAL 2,0
12 FOR A=0 TO 195 STEP 10 :LINE [1]F,A,F
,A+4 :LINE [2]F,A+5,F,A+9 :LINE [2]F+1,A
,F+1,A+4 :LINE [1]F+1,A+5,F+1,A+9 :NEXT
```

```
13 X=INT(RND(1)*81+1) :A$=A$(X,1) :AT=AT
+1 :IF AT=10 THEN AT=-1 :F=F+5 :SC=SC+1
:GOTO 11
14 SC=SC+1 :A=1 :B=A :C=0 :C$=A$
15 IF C=0 THEN PAL 1,0 :PAL 2,6 :C=1 :GO
TO 17
16 PAL 1,6 :PAL 2,0 :C=0
17 SYMBOL [0]319-B,30,LEFT$(C$,B/8),1,1
18 IF A$="" THEN 13
19 C$=A$ :B=A
20 SYMBOL 319-A,30,LEFT$(A$,A/8),1,1
21 GET B$ :IF B$=LEFT$(A$,1) THEN A$=MID
$(A$,2,LEN(A$)-1) :A=A-8 :T=T+1
22 A=A+1 :IF 319-A=F+1 THEN 24
23 GOTO 15
24 FOR A=0 TO 200 :PAL 0,INT(RND(1)*16)
:PAL 1,INT(RND(1)*16) :PAL 2,INT(RND(1)*
16) :PAL 3,INT(RND(1)*16) :NEXT A
25 WAIT 1000 :INIT"CRT:M1" :SYMBOL 15,10
,"UW SCORE WAS",3,3 :SYMBOL 160-12*LEN(S
TR$(SC)),40,STR$(SC),3,3
26 BOX [2]90,115,237,131
27 PAINT [2]91,116 :SYMBOL 96,120,"DRUK
OP EEN TOETS",1,1 :T=0
28 T=T+1 :IF T=50 THEN PAL 2,0
29 IF T=100 THEN PAL 2,2 :T=0
30 GET F$ :IF F$="" THEN 28
31 RUN 10
32 DATA COMPUTER,DISKETTE,TYPLES,TOPOGRA
FIE,LOGICA,STENO,GELUKKIG,POLITIE,LESSEN
AAR,TELEVISIE,BRANDWEER
33 DATA SATELIET,DRIEHOEK,KLIMREK,INJECT
IE,FIETS,HULPPOST,TROUWDAG,BRUILOFT,PROG
RAMMA,DIRECT,PROJECTOR,GETAL,GENETICA,TR
AMPOLINE,SPITSUUR,AUTOBAND,UITTREDEN,PRO
EFWERK
34 DATA WASTAFEL,IDIOOM,KAMERDEUR,ZAKDOE
K,FABRIEK,OORBEL,HORLOGE,KALENDER,PRINTE
R,SPORTTAS,TELEFOON,THEETAFEL,ZIEKTE,PUI
ST,KLEDING,AURA,MACHINE
35 DATA ZONNEBANK,CIJFER,AFDELING,PENNED
OP,HONDEHOK,KIPPEREN,DOELPAAL,VOETBAL,HA
NDBAL,VOLLEYBAL,TENNIS,GOLFBAL,ZWEMLES,D
UIKPLANK,ZWEMVEST,REDDING,KEEPER
36 DATA WISKUNDE,BIOLOGIE,ECONOMIE,WELVA
ART,WELZIJN,ZWAAIEN,RETURN,JODELAAR,PRIK
TOL,KNIKKER,SHAMPOO,SCHELP,POTLOOD,NATIO
NAAL,ALCOHOL,CIGARET,SIGAAR,PIJPTABAK
```

Programma 10: POINTER.

De instructies staan in het programma.

```
1 INIT"CRT:M1" :SYMBOL 10,90,"INSTRUCTIE
S? (J/N)",2,2
```

```
2 GET A$ :IF A$="J" THEN "INSTRUCTIES"
3 IF A$<>"N" THEN 2
4 LABEL "START"
5 INIT"CRT:M1" :PAL 1,7 :PAL 2,8 :BOX 25
0,0,319,199 :PAINT [2]251,1,3 :COLOR 1 :
SYMBOL 257,5,"POINTER",1,1 :LINE [3]257,
13,312,13
6 SYMBOL 257,20,"(C)1989",1,1 :SYMBOL 26
2,30," door",1,1 :SYMBOL 257,41,"  N-S",
1,1 :SYMBOL 261,90,"SCORE:",1,1
7 T$="000000" :SYMBOL [3]260,99,T$,1,1 :
BOX [2]0,0,249,199
8 T$="000000" :A=0 :COLOR 3
9 X=1 :Y=1 :X1=1 :Y1=1 :TI=300
10 FOR T=1 TO 5 :SET [1]INT(RND(1)*249)+
1,INT(RND(1)*197+2) :NEXT T :AT=5
11 RESET X1,Y1 :SET X,Y :X1=X :Y1=Y
12 TI=TI-1 :IF TI=0 THEN "EINDE"
13 S=STICK(0) :IF S=3 THEN X=X+1
14 IF S=5 THEN Y=Y+1
15 IF S=7 THEN X=X-1
16 IF S=1 THEN Y=Y-1
17 IF POINT(X,Y)=2 THEN X=X1 :Y=Y1 :GOTO
 11
18 IF POINT(X,Y)=1 THEN GOSUB 22
19 IF AT=0 THEN RESET X1,Y1 :RESET X,Y :
GOTO 9
20 IF S<>0 THEN SOUND 10,1
21 GOTO 11
22 A=A+10 :A$=STR$(A)
23 S$=T$ :T$=LEFT$(T$,6-LEN(A$))+A$ :SYM
BOL [2]260,99,S$,1,1 :SYMBOL [3]260,99,T
$,1,1
24 AT=AT-1 :TI=TI+150
25 FOR G=1 TO 100 :SOUND INT(RND(1)*70)+
10,1 :PAL 0,INT(RND(1)*16) :NEXT G :PAL
0,0
26 RETURN
27 LABEL "INSTRUCTIES"
28 CLS :SYMBOL 100,5,"POINTER",2,2 :FOR
A=22 TO 24 :LINE [2]100,A,215,A :NEXT A
29 CURSOR 0,5 :PRINT"HET IS DE BEDOELING
 DAT U ALLE PUNTJES  OP HET BEELDSCHERM
PAKT. HIERVOOR HEEFT U MAAR EEN BEPERKTE
 TIJD. DEZE TIJD"
30 PRINT"STAAT NIET AANGEGEVEN OM HET SP
EL ZO    SNEL MOGELIJK TE LATEN ZIJN."
31 PRINT :PRINT"BESTURING MET DE PIJLTJE
STOETSEN." :PRINT :PRINT :PRINT :PRINT"
        *** VEEL SUCCES ***"
32 PAL 1,2 :BOX [1]75,155,221,172 :PAINT
 [1]76,156,1 :SYMBOL 80,160,"DRUK OP EEN
 TOETS",1,1 :T=0
33 T=T+1 :IF T=50 THEN PAL 1,0
34 IF T=100 THEN PAL 1,2 :T=0
```

```
35 GET A$ :IF A$="" THEN 33 ELSE "START"
36 LABEL "EINDE"
37 FOR A=0 TO 250 :PAL 0,INT(RND(1)*16)
:PAL 3,INT(RND(1)*16) :NEXT A :PAL 3,15
:PAL 0,0
38 BOX [2]50,30,200,105 :PAINT [1]51,31,
2
39 SYMBOL [0]58,35,"GAME",4,4 :SYMBOL [0
]58,70,"OVER",4,4 :BOX [2]47,120,203,135
 :PAINT [1]51,121,2 :SYMBOL [0]51,124,"N
OG EEN KEER? (J/N)",1,1
40 GET A$ :IF A$="J" THEN "START"
41 IF A$<>"N" THEN 40
42 INIT"CRT:M1" :NEW
```

Programma 11: JUMPMAN.

In dit spel moet U proberen helemaal naar boven te springen. De besturing gaat volledig met de pijltjestoetsen. Dit betekent dat U kunt springen door op de pijltjestoets naar boven te drukken.
U mag natuurlijk niets raken bij het springen. Kijk ook uit voor het 'spook'.

```
1 INIT"CRT:M1" :INPUT"SNELHEID (1-100)";
SN
2 IF SN<1 OR SN>100 THEN 1
3 CLS :BOX [1]0,192,319,199,1 :PRINT CHR
$(21)
4 Q=215 :P=32 :A$=CHR$(Q,Q,Q,Q,P,P,Q,Q,Q
,Q,P,P,Q,Q,Q,Q,Q,P,Q,P,Q,Q,Q,Q,Q,P,P,Q,Q
,Q,Q,P,P,Q,Q,Q,Q,Q,P,P) :A$=A$+A$ :B$=A$
5 FOR A=0 TO 23 STEP 3 :CURSOR 0,A :PRIN
T MID$(A$,1,40) :NEXT :X=20 :Y=23 :E=1 :
F=21 :X1=X :Y1=Y :R=0 :SC=0 :ZZ=0
6 SYMBOL 2,192,"* JUMPMAN *",1,1 :SYMBOL
 105,192,"(C)1989/1990 N-S",1,1 :SYMBOL
248,192,"SCORE:"+STR$(SC),1,1
7 I=INT(RND(1)*7)*3+2 :H=INT(RND(1)*40)
:CURSOR H,I :PRINT CHR$(104)
8 IF PEEK($2000+X1+40*Y1)<>215 THEN CURS
OR X1,Y1 :PRINT" "
9 X1=X :Y1=Y
10 IF PEEK($2000+X+40*Y)=215 THEN 29
11 CURSOR X,Y :PRINT"O"; :CURSOR 0,F :PR
INT MID$(A$,E,40); :IF R=3 THEN R=0 :G=0
 :GOSUB 26
12 S=STICK(0) :IF S=3 AND X<38 THEN X=X+
1 :SOUND 10,1
13 IF S=7 AND X>0 THEN X=X-1 :SOUND 10,1
14 IF S=1 THEN G=1
15 IF G=1 THEN Y=Y-1 :R=R+1 :S=1 :SOUND
10,1
16 IF Y=-1 THEN Y=23 :GOSUB 35 :GOTO 7
17 WAIT SN
18 IF PEEK($2000+X+Y*40)=104 THEN 29
```

```
19 IF F+2=I THEN CURSOR H,I :PRINT" " :H
=H-1 :IF H=-1 THEN H=39
20 IF F+2=I THEN CURSOR H,I :PRINT CHR$(
104)
21 IF ZZ=0 AND F+5=I THEN CURSOR H,I :PR
INT" " :ZZ=1
22 IF PEEK($2000+X+Y*40)=104 THEN 29
23 E=E+1 :IF E=41 THEN E=1
24 IF (S-1)/2=INT((S-1)/2) THEN 8
25 GOTO 11
26 E=1 :SYMBOL [1]296,192,STR$(SC),1,1 :
SC=SC+1 :SYMBOL 296,192,STR$(SC),1,1
27 CURSOR 0,F :PRINT MID$(A$,1,40) :F=F-
3 :IF F=-3 THEN F=21
28 RETURN
29 CURSOR X1,Y1 :PRINT" " :CURSOR X,Y :P
RINT"*" :WAIT 1000 :FOR A=0 TO 250 :PAL
3,INT(RND(1)*16) :PAL 0,INT(RND(1)*16) :
NEXT A
30 PAL 0,7 :PAL 3,0 :PAL 1,5 :BOX [2]106
,45,214,100,1 :BOX [2]107,46,213,99 :SYM
BOL 112,50,"GAME",3,3 :SYMBOL 112,76,"OV
ER",3,3
31 BOX [2]105,110,215,121,1 :SYMBOL 109,
113,"NOG EEN KEER?",1,1
32 GET A$ :IF A$="N" THEN INIT"CRT:M1" :
NEW
33 IF A$<>"J" THEN 32
34 INIT"CRT:M1" :CLR :GOTO 1
35 E=1 :CURSOR H,I :PRINT" " :SYMBOL [1]
296,192,STR$(SC),1,1 :SC=SC+10 :SYMBOL 2
96,192,STR$(SC),1,1 :RETURN
```

Programma 12: VIER OP EEN RIJ voor twee personen.

Het overbekende spel vier op een rij voor twee personen. De besturing gaat hierbij ook weer volledig met de pijltjestoetsen. Beide spelers gebruiken dezelfde toetsen. De speler die de beurt heeft kan een fiche laten vallen door op de pijltjestoets naar beneden te drukken.
Met een pijltje (>) wordt aangegeven wie er aan de beurt is.
Speler 1 speelt met de gesloten rondjes, speler 2 met de open rondjes.
Degene die als eerste vier op een rij heeft is winnaar.
Wanneer iemand gewonnen heeft en er wordt daarna nog een spelletje gespeeld, wordt speler 1 automatisch speler 2 en wordt speler 2 automatisch speler 1.

```
1 DIM H(81) :FOR A=3 TO 83 :H(A-2)=A :NE
XT A :M=73 :DIM X(9) :DIM Y(9)
2 FOR A=1 TO 9 :X(A)=50+A*20 :NEXT A
3 FOR A=1 TO 9 :Y(A)=186 :NEXT A
4 INIT"CRT:M1" :CONSOLE 14,11 :A$="VIER
OP EEN RIJ" :B$=" (C)1989 door" :C$="NEP
TUNES SOFTWARE" :FOR A=1 TO 15 :CURSOR 4
0-A,10 :PRINT MID$(A$,1,A) :SOUND A+16,1
 :NEXT A
```

```
5 FOR A=16 TO 27 :CURSOR 40-A,10 :PRINT
A$;" " :SOUND A+16,1 :NEXT A :FOR A=1 TO
 14 :CURSOR 0,12 :PRINT MID$(B$,15-A,A)
:SOUND (24-A)+16,1 :NEXT A :FOR A=15 TO
27 :CURSOR A-15,12 :PRINT" ";B$ :SOUND (
24-A)+16,1 :NEXT A
6 FOR A=23 TO 15 STEP -1 :CURSOR 12,A :P
RINT C$ :CURSOR 0,24 :PRINT :SOUND 36-A,
1 :NEXT A :FOR A=1 TO 3 :FOR B=0 TO 15 :
PAL 3,B :WAIT 100 :NEXT B :FOR C=15 TO 0
 STEP -1 :PAL 3,C :WAIT 100 :NEXT C,A :P
AL 3,7 :WAIT 1000
7 PAL 2,3 :B=0 :FOR A=0 TO 160 :LINE [2]
A,B,A,200-B :LINE [2]320-A,B,320-A,200-B
 :LINE [2]A,B,320-A,B :LINE [2]A,200-B,3
20-A,200-B :B=B+100/160 :NEXT A :PAL 3,1
2
8 A$="SPELER 1:" :B$="SPELER 2:"
9 FOR T=1 TO 9 :CURSOR 0,0 :PRINT MID$(A
$,1,T); :BEEP :WAIT 100 :NEXT T :INPUT A
1$ :IF LEN(A1$)>13 THEN CURSOR 10,0 :FOR
 T=1 TO LEN(A1$)+1 :PRINT" "; :NEXT T :P
AINT [2]0,0,2 :GOTO 9
10 FOR T=1 TO 9 :CURSOR 0,2 :PRINT MID$(
B$,1,T); :BEEP :WAIT 100 :NEXT T :INPUT
A2$ :IF LEN(A2$)>13 THEN CURSOR 10,2 :FO
R T=1 TO LEN(A2$)+1 :PRINT" "; :NEXT T :
PAINT [2]0,17,2 :GOTO 10
11 INIT"CRT:M1"
12 FOR A=60 TO 240 STEP 20 :LINE A,38,A,
195 :NEXT A :FOR A=55 TO 195 STEP 140/8
:LINE 60,A,240,A :NEXT A :PAL 2,5 :CURSO
R 8,0 :PRINT"SPELER 1:";A1$ :CURSOR 8,1
:PRINT"SPELER 2:";A2$
13 A=0 :B=62 :R=1 :E=0 :PAL 1,15
14 CURSOR 7,A :PRINT">";
15 SYMBOL B,21,CHR$(252),2,2
16 S=STICK(0) :IF B=222 THEN 19
17 IF S=3 THEN B=B+20 :FOR C=21 TO 37 :L
INE [0]0,C,320,C :NEXT C :M=M+1 :R=R+1
18 IF B=62 THEN 20
19 IF S=7 THEN B=B-20 :FOR C=21 TO 37 :L
INE [0]0,C,320,C :NEXT C :M=M-1 :R=R-1
20 IF S=5 THEN GOSUB 23
21 IF E=81 THEN 27
22 GOTO 14
23 IF Y(R)=28.5 THEN RETURN
24 G=-((Y(R)-28.5)/17.5)+9 :G=G*9 :G=M-G
25 CIRCLE [1]X(R),Y(R),5,1 :Y(R)=Y(R)-17
.5 :IF A=0 THEN CURSOR 7,A :PRINT" "; :A
=1 :PAINT [1]X(R)+3,Y(R)+20.5,1 :E=E+1 :
H(G)=1 :GOSUB 28 :RETURN
26 CURSOR 7,A :PRINT" "; :A=0 :E=E+1 :H(
G)=2 :GOSUB 28 :RETURN
27 INIT"CRT:M1"
```

```
28 FOR Z=1 TO 78 :IF H(Z)=H(Z+1) THEN GO
SUB 36
29 NEXT Z
30 FOR Z=1 TO 54 :IF H(Z)=H(Z+9) THEN GO
SUB 39
31 NEXT Z
32 FOR Z=1 TO 51 :IF H(Z)=H(Z+10) THEN G
OSUB 42
33 NEXT Z
34 FOR Z=1 TO 54 :IF H(Z)=H(Z+8) THEN GO
SUB 45
35 NEXT Z :RETURN
36 IF H(Z+1)<>H(Z+2) THEN RETURN
37 IF H(Z+2)<>H(Z+3) THEN RETURN
38 GOTO 47
39 IF H(Z+9)<>H(Z+18) THEN RETURN
40 IF H(Z+18)<>H(Z+27) THEN RETURN
41 GOTO 47
42 IF H(Z+10)<>H(Z+20) THEN RETURN
43 IF H(Z+20)<>H(Z+30) THEN RETURN
44 GOTO 47
45 IF H(Z+8)<>H(Z+16) THEN RETURN
46 IF H(Z+16)<>H(Z+24) THEN RETURN
47 W=H(Z) :FOR A=1 TO 5 :FOR Z=0 TO 15 :
PAL 0,Z :PAL 3,15-Z :WAIT 50 :NEXT Z,A
48 PAL 0,0 :PAL 3,15 :WAIT 1000 :CLS
49 IF W=1 THEN S$=A1$+" HEEFT GEWONNEN."
 :A$=A1$ :GOTO 51
50 S$=A2$+" HEEFT GEWONNEN." :A$=A2$
51 SYMBOL 75-3*LEN(A$),92,S$,1,2
52 SYMBOL 100,150,"NOG EEN KEER",1,2
53 GET Z$ :IF Z$="J" THEN 56
54 IF Z$="N" THEN INIT"CRT:M1" :NEW
55 GOTO 53
56 FOR A=3 TO 83 :H(A-2)=A :NEXT A :M=73
57 FOR A=1 TO 9 :X(A)=50+A*20 :NEXT A
58 FOR A=1 TO 9 :Y(A)=186 :NEXT A
59 A3$=A1$ :A1$=A2$ :A2$=A3$ :GOTO 11
```

Programma 13: JAGER, 16 kleurentekening.

Onderstaand programma tekent een jager in zestien kleuren. Dit is één van de vele figuurtjes uit het programma PICTURESHOW. Om het programma zo kort mogelijk te houden, is een speciale vorm van data gebruikt. Het intypen is daardoor wel iets lastiger.

```
1 INIT"CRT:M2" :CURSOR 0,22 :BOX [9]0,0,
195,160,9 :B=1 :Y=0 :A=1 :DIM A$(12)
2 A$(1)="GA2933C3438GA273283334C3542GA25
3083133C3444GA242983031C3246GA232882930C
314744044GA222782829C304743845GA21268272
8C294843746GA202682727C284843647GA192582
627C284943547GA192582626C274943948GA1824
82526C27494333744148GA182482525C2649H"
```

```
3 A$(2)="43748GA172382425C264944048GA172
382424C254044148GA161881923C243443547G81
618B192342435C3646G81616B1724C2548GB1626
C274933744GB153733846C4749GB153962626A29
3134047C4850GB144062626A282834148C4950GB
144062525A2828833964141342480495OH"
4 A$(3)="F5152F5960GB143962525A272784040
6414134248C4950F5153F575976061GB14396252
5A2727840406414134248C494975050F5153F566
275960GB1438639413424875050F5152F5460757
5876162GB143862930639413424774961F5154F5
759F6262GB153762829F3030638403414 5H"
5 A$(4)="74859F5057F6062GB153662728F2930
63740F393974857F4955F5861GB1335A15166273
102828F293063637F383974848F4960GB1234817
186263102829F30306353603737F3839F4759749
4975151GB1225A1415618188191962635027 29F3
03003637F383864747F485674951757 58H"
6 A$(5)="GB1225A151681719618186293162632
02728F2930A333503637F3838B464664747F4857
75151GB12358171761818A2532730300363 6F373
7C4444B454664748F49547515275556G61317B18
38A3030A3636C4344B454664749F505175254GA1
517B1839A2931A35367424344444B4545H"
7 A$(6)="64647F5050GB183982627A2829A3233
83435C41434444464546GB193882425227280343
443946C41436454 5G82020B21372293343846C40
43G819200212162222B23356323343645C3943G8
1920021216222 4B2534A293083132435 5C3843G
C194382026B273243536GC17424181882132H"
8 A$(7)="43637GC163941818823324363744041
GA0507C1540419218283243536GA0410C1437420
208293143435438 39G80306A0712C13364182783
030432334373 8G80211C123541620F2122428318
303043637G80210C113441418119207212 1F2223
430314353 6G80109C10344171711819F2023H"
9 A$(8)="7212143030E31324353 5G80109C1035
F17239242643030E3132G80109C103541430F162
232325126290363 6G80110C113341414A1524F16
223252812932034 36GC111241314A1527F162282
324328311323503636GC121241314A153071616F
172382427331341353 8G4131571616F1724H"
10 A$(9)="82530A31333343713842G41616C171
7F18230242782833A343633739140450555 5G416
16C172502627E2830831353364214349054 55G41
737C1929C31368333433845146530545 5G41737C
2029C3135C38383404814959G41736C2228C3234
C37383435115259057 58G41635C363934654H"
11 A$(10)="1556005659G41534C2327C3539349
5415560056 59G41534C1823C3540352551566005
758G41437C1719C384035555156599272 9G41423
C172143135C3639A4040G41423C1821A32404343
8GA142341522A323934040GA142333341A3537G3
070931422A1821334390373704041344 47H"
```

```
12 A$(11)="04849G3051100606014163172233 4
50037400494 9G3042100606016173335003839 4
949G3042100606016173335103940048 9G30320
00708014150212103348334393434534951G3032
00091402121033453344234651G3031901616020
20033513343733950G3041401520033513415 H"
13 A$(12)="G00514306120161903437040503 2
49G00512042501"
14 C$=MID$(A$(B),A,1) :C=VAL(C$)
15 IF C$=>"A" THEN C=ASC(C$)-55
16 IF C$="H" THEN B=B+1 :A=1 :GOTO 14
17 IF C$="I" THEN END
18 IF C$="G" THEN A=A+1 :Y=Y+1 :GOTO 14
19 X$=MID$(A$(B),A+1,2) :X1$=MID$(A$(B),
A+3,2) :X=VAL(X$) :X1=VAL(X1$) :BOX [C]X
*2,Y*2,X1*2+1,Y*2+1 :A=A+5 :GOTO 14
```

Programma 14: LOOK OUT.

Als laatste programma uit dit hoofdstuk volgt hier een volledig machinetaalprogramma zoals dat oorspronkelijk is geschreven. Er staat dus geen Neptunes Software in het programma, maar A. Habing. Tevens zitten er nog dingen in die korter kunnen. Het programma loopt prima en daar gaat het om.
Het is de bedoeling dat U de aarde beschermt tegen de ufo's. U kunt ze kapot schieten door op de spatiebalk te drukken. U kunt ook heen en weer bewegen met behulp van de pijltjestoetsen.
Mocht een ufo op Uw verdedigingwapen komen of mocht een ufo door de drie lagen dikke beschermlaag komen dan bent U af.
Wanneer U tien ufo's kapot geschoten heeft zal het spel iets sneller worden. Bij elkaar duurt het dan toch wel even voordat het spel echt veel sneller wordt.
Wanneer een ufo de zijkant raakt, zal de ufo verdwijnen.
Het programma kunt U opstarten met G124E.
S=Pauze C=doorgaan. VEEL PLEZIER

1200 0C	1212 2E	1224 20	1236 20	1248 0A	125A 27	126C 60
1201 0F	1213 08	1225 20	1237 20	1249 2D	125B 1A	126D 21
1202 0F	1214 01	1226 20	1238 20	124A 0E	125C 77	126E 20
1203 0B	1215 02	1227 0A	1239 20	124B 69	125D 23	126F D8
1204 00	1216 09	1228 05	123A 0E	124C 00	125E 13	1270 36
1205 0F	1217 0E	1229 00	123B 0F	124D 02	125F 10	1271 0F
1206 15	1218 07	122A 13	123C 07	124E 3E	1260 FA	1272 21
1207 14	1219 00	122B 03	123D 00	124F C6	1261 21	1273 12
1208 00	121A 00	122C 0F	123E 05	1250 CD	1262 00	1274 D8
1209 00	121B 13	122D 12	123F 05	1251 DC	1263 D8	1275 36
120A 88	121C 03	122E 05	1240 0E	1252 0D	1264 06	1276 0F
120B 00	121D 0F	122F 00	1241 00	1253 21	1265 27	1277 21
120C 21	121E 12	1230 17	1242 0B	1254 00	1266 36	1278 28
120D 29	121F 05	1231 01	1243 05	1255 D0	1267 8F	1279 D8
120E 28	1220 4F	1232 13	1244 05	1256 11	1268 23	127A 06
120F 29	1221 20	1233 4F	1245 12	1257 00	1269 10	127B 18
1210 00	1222 20	1234 20	1246 00	1258 12	126A FB	127C 0E
1211 01	1223 20	1235 20	1247 68	1259 06	126B 36	127D 28

127E 36
127F 60
1280 23
1281 0D
1282 C2
1283 7E
1284 12
1285 10
1286 F5
1287 21
1288 70
1289 DB
128A 06
128B 27
128C 36
128D 07
128E 23
128F 10
1290 FB
1291 23
1292 06
1293 27
1294 36
1295 01
1296 23
1297 10
1298 FB
1299 23
129A 06
129B 27
129C 36
129D 05
129E 23
129F 10
12A0 FB
12A1 21
12A2 27
12A3 D0
12A4 3A
12A5 05
12A6 E0
12A7 4F
12A8 06
12A9 27
12AA FE
12AB 01
12AC CA
12AD B5
12AE 12
12AF 3D
12B0 10
12B1 F8
12B2 C3
12B3 A7
12B4 12
12B5 23
12B6 0D
12B7 C2
12B8 B5
12B9 12
12BA 36
12BB C7
12BC 7D
12BD 32
12BE 24
12BF 13
12C0 ED
12C1 5F
12C2 FE
12C3 00
12C4 CA
12C5 D5
12C6 12
12C7 FE
12C8 2A
12C9 CA
12CA DD
12CB 12
12CC FE
12CD 55
12CE CA
12CF E2
12D0 12
12D1 3D
12D2 C3
12D3 C2
12D4 12
12D5 3E
12D6 28
12D7 32
12D8 27
12D9 13
12DA C3
12DB E7
12DC 12
12DD 3E
12DE 27
12DF C3
12E0 D7
12E1 12
12E2 3E
12E3 29
12E4 C3
12E5 D7
12E6 12
12E7 21
12E8 4F
12E9 D8
12EA 11
12EB 28
12EC 00
12ED 06
12EE 18
12EF 36
12F0 80
12F1 19
12F2 10
12F3 FB
12F4 21
12F5 5B
12F6 D3
12F7 36
12F8 3F
12F9 11
12FA 00
12FB 08
12FC 19
12FD 36
12FE D0
12FF 0E
1300 80
1301 06
1302 FF
1303 10
1304 FE
1305 0D
1306 C2
1307 01
1308 13
1309 3A
130A 4C
130B 12
130C FE
130D 00
130E CA
130F 15
1310 13
1311 3D
1312 32
1313 4C
1314 12
1315 3A
1316 4D
1317 12
1318 3D
1319 FE
131A 00
131B C2
131C 35
131D 14
131E 3E
131F 02
1320 32
1321 4D
1322 12
1323 21
1324 00
1325 D0
1326 11
1327 28
1328 00
1329 36
132A 00
132B 19
132C 7E
132D FE
132E 3F
132F CA
1330 6A
1331 14
1332 7C
1333 FE
1334 D4
1335 CA
1336 6A
1337 14
1338 36
1339 C7
133A 7C
133B 32
133C 25
133D 13
133E 7D
133F 32
1340 24
1341 13
1342 11
1343 00
1344 08
1345 19
1346 7E
1347 FE
1348 80
1349 CA
134A 28
134B 14
134C FE
134D 60
134E C2
134F 66
1350 13
1351 CD
1352 E5
1353 14
1354 FE
1355 13
1356 CA
1357 71
1358 13
1359 FE
135A 14
135B CA
135C 95
135D 13
135E FE
135F 20
1360 CA
1361 B1
1362 13
1363 C3
1364 F4
1365 12
1366 36
1367 60
1368 11
1369 00
136A F8
136B 19
136C 36
136D 00
136E C3
136F 28
1370 14
1371 3A
1372 F6
1373 12
1374 67
1375 3A
1376 F5
1377 12
1378 6F
1379 7D
137A FE
137B 6E
137C CA
137D F4
137E 12
137F 36
1380 00
1381 11
1382 00
1383 08
1384 19
1385 36
1386 60
1387 ED
1388 52
1389 23
138A 7C
138B 32
138C F6
138D 12
138E 7D
138F 32
1390 F5
1391 12
1392 C3
1393 F4
1394 12
1395 3A
1396 F6
1397 12
1398 67
1399 3A
139A F5
139B 12
139C 6F
139D 7D
139E FE
139F 48
13A0 CA
13A1 F4
13A2 12
13A3 36
13A4 00
13A5 11
13A6 00
13A7 08
13A8 19
13A9 36
13AA 60
13AB ED
13AC 52
13AD 2B
13AE C3
13AF 8A
13B0 13
13B1 3A
13B2 4C
13B3 12
13B4 FE
13B5 00
13B6 C2
13B7 F4
13B8 12
13B9 3E
13BA 05
13BB 32
13BC 4C
13BD 12
13BE 11
13BF 28
13C0 00
13C1 3A
13C2 F5
13C3 12
13C4 6F
13C5 3A
13C6 F6
13C7 12
13C8 67
13C9 ED
13CA 52
13CB 0E
13CC 14
13CD 7E
13CE FE
13CF C7
13D0 CA
13D1 ED
13D2 13
13D3 36
13D4 80
13D5 11
13D6 D8
13D7 FF
13D8 19
13D9 11
13DA 28
13DB 00
13DC 10
13DD FE
13DE 0D
13DF C2
13E0 CD
13E1 13
13E2 19
13E3 06
13E4 14
13E5 36
13E6 00
13E7 19
13E8 10
13E9 FB
13EA C3
13EB F4
13EC 12
13ED 36
13EE EF
13EF 11
13F0 00
13F1 08
13F2 19
13F3 36
13F4 D0
13F5 3E
13F6 04
13F7 0E
13F8 FF
13F9 06
13FA FF
13FB 10
13FC FE
13FD 0D
13FE C2

```
13FF F9   1423 2A   1447 3C   146B 34   148F D8   14B3 1A   14D7 13
1400 13   1424 CC   1448 FE   146C 12   1490 0E   14B4 77   14D8 3E
1401 3D   1425 3B   1449 2A   146D 11   1491 19   14B5 23   14D9 5B
1402 C2   1426 14   144A CC   146E 21   1492 06   14B6 13   14DA 32
1403 F7   1427 77   144B 4E   146F D0   1493 28   14B7 10   14DB F5
1404 13   1428 3E   144C 14   1470 06   1494 36   14B8 FA   14DC 12
1405 21   1429 02   144D C9   1471 06   1495 70   14B9 CD   14DD 3E
1406 28   142A 32   144E 36   1472 1A   1496 23   14BA 1B   14DE D0
1407 D0   142B 4D   144F 20   1473 77   1497 10   14BB 00   14DF 32
1408 11   142C 12   1450 2B   1474 23   1498 FB   14BC FE   14E0 25
1409 00   142D 3E   1451 7E   1475 13   1499 0D   14BD 4A   14E1 13
140A 08   142E D0   1452 3C   1476 10   149A C2   14BE CA   14E2 C3
140B 0E   142F 32   1453 FE   1477 FA   149B 92   14BF C9   14E3 4E
140C 14   1430 25   1454 2A   1478 3E   149C 14   14C0 14   14E4 12
140D 06   1431 13   1455 CC   1479 08   149D 21   14C1 FE   14E5 CD
140E 28   1432 C3   1456 59   147A 0E   149E 9C   14C2 4E   14E6 1B
140F 36   1433 A1   1457 14   147B FF   149F D1   14C3 CA   14E7 00
1410 00   1434 12   1458 C9   147C 06   14A0 11   14C4 00   14E8 FE
1411 19   1435 32   1459 36   147D FF   14A1 27   14C5 00   14E9 53
1412 36   1436 4D   145A 20   147E 10   14A2 12   14C6 C3   14EA CA
1413 60   1437 12   145B 2B   147F FE   14A3 06   14C7 B9   14EB EE
1414 ED   1438 C3   145C 7E   1480 0D   14A4 13   14C8 14   14EC 14
1415 52   1439 51   145D 3C   1481 C2   14A5 1A   14C9 3E   14ED C9
1416 23   143A 13   145E FE   1482 7C   14A6 77   14CA 00   14EE CD
1417 10   143B 36   145F 2A   1483 14   14A7 13   14CB 32   14EF 1B
1418 F6   143C 20   1460 CC   1484 3D   14A8 23   14CC 4C   14F0 00
1419 0D   143D 3A   1461 64   1485 C2   14A9 10   14CD 12   14F1 FE
141A C2   143E 00   1462 14   1486 7A   14AA FA   14CE 3E   14F2 43
141B 0D   143F 13   1463 C9   1487 14   14AB 21   14CF 02   14F3 C2
141C 14   1440 D6   1464 36   1488 3E   14AC EC   14D0 32   14F4 EE
141D 21   1441 05   1465 20   1489 C6   14AD D1   14D1 4D   14F5 14
141E 25   1442 32   1466 2B   148A CD   14AE 11   14D2 12   14F6 CD
141F D0   1443 00   1467 7E   148B DC   14AF 3A   14D3 3E   14F7 1B
1420 7E   1444 13   1468 3C   148C 0D   14B0 12   14D4 80   14F8 00
1421 3C   1445 2B   1469 C9   148D 21   14B1 06   14D5 32   14F9 C9
1422 FE   1446 7E   146A 21   148E 00   14B2 12   14D6 00
```

Dit was het laatste programma uit dit hoofdstuk en het is tevens het laatste programma uit dit boek. Hierna vindt U nog een aantal appendices met daarin nog enkele zaken die U nodig heeft bij het gebruiken van dit boek.
Dit boek wordt afgesloten met een slotwoord. Wij zouden het erg op prijs stellen wanneer U dit slotwoord ook leest. In dit slotwoord proberen we duidelijk te maken waarom het boek op deze manier opgezet is en er in principe maar weinig uitleg in dit boek staat.
Ook willen wij U graag verwijzen naar appendix D. In appendix D vindt U o.a. onze adressen en informatie over SHARP-clubs in ons land.
Mochten er nog aanvullingen op dit boek komen, en die kans is behoorlijk groot, dan zullen er zeker ook nog een paar leuke programma's gepubliceerd worden. Vooral aan muziekstukken hebben we weinig aandacht besteed. In eventuele aanvullingen hopen we veel meer muziekstukken te kunnen plaatsen.
Hopelijk heeft U iets aan deze programma's en kunt U er iets van leren of iets leuks mee doen.

APPENDIX A.

Z80-A mnemonics.

Hieronder vindt U de hele lijst van mnemonics terug. De opbouw is iets anders dan U waarschijnlijk gewend zult zijn van andere boeken. Er valt verder weinig aan uit te leggen.

Decimaal:	Hexadecimaal:	Assemblertaal:	Na code CB:	Na code ED:
0	00	NOP	RLC B	-
1	01	LD BC,NN	RLC C	-
2	02	LD (BC),A	RLC D	-
3	03	INC BC	RLC E	-
4	04	INC B	RLC H	-
5	05	DEC B	RLC L	-
6	06	LD B,N	RLC (HL)	-
7	07	RLCA	RLC A	-
8	08	EX AF,AF'	RRC B	-
9	09	ADD HL,BC	RRC C	-
10	0A	LD A,(BC)	RRC D	-
11	0B	DEC BC	RRC E	-
12	0C	INC C	RRC H	-
13	0D	DEC C	RRC L	-
14	0E	LD C,N	RRC (HL)	-
15	0F	RRCA	RRC A	-
16	10	DJNZ DIS	RL B	-
17	11	LD DE,NN	RL C	-
18	12	LD (DE),A	RL D	-
19	13	INC DE	RL E	-
20	14	INC D	RL H	-
21	15	DEC D	RL L	-
22	16	LD D,N	RL (HL)	-
23	17	RLA	RL A	-
24	18	JR DIS	RR B	-
25	19	ADD HL,DE	RR C	-
26	1A	LD A,(DE)	RR D	-
27	1B	DEC DE	RR E	-
28	1C	INC E	RR H	-
29	1D	DEC E	RR L	-
30	1E	LD E,N	RR (HL)	-
31	1F	RRA	RR A	-
32	20	JR NZ,DIS	SLA B	-
33	21	LD HL,NN	SLA C	-
34	22	LD (NN),HL	SLA D	-
35	23	INC HL	SLA E	-
36	24	INC H	SLA H	-
37	25	DEC H	SLA L	-
38	26	LD H,N	SLA (HL)	-
39	27	DAA	SLA A	-
40	28	JR Z,DIS	SRA B	-
41	29	ADD HL,HL	SRA C	-
42	2A	LD HL,(NN)	SRA D	-
43	2B	DEC HL	SRA E	-

Decimaal:	Hexadecimaal:	Assemblertaal:	Na code CB:	Na code ED:
44	2C	INC L	SRA H	-
45	2D	DEC L	SRA L	-
46	2E	LD L,N	SRA (HL)	-
47	2F	CPL	SRA A	-
48	30	JR NC,DIS	-	-
49	31	LD SP,NN	-	-
50	32	LD (NN),A	-	-
51	33	INC SP	-	-
52	34	INC (HL)	-	-
53	35	DEC (HL)	-	-
54	36	LD (HL),N	-	-
55	37	SCF	-	-
56	38	JR C,DIS	SRL B	-
57	39	ADD HL,SP	SRL C	-
58	3A	LD A,(NN)	SRL D	-
59	3B	DEC SP	SRL E	-
60	3C	INC A	SRL H	-
61	3D	DEC A	SRL L	-
62	3E	LD A,N	SRL (HL)	-
63	3F	CCF	SRL A	-
64	40	LD B,B	BIT 0,B	IN B,(C)
65	41	LD B,C	BIT 0,C	OUT (C),B
66	42	LD B,D	BIT 0,D	SBC HL,BC
67	43	LD B,E	BIT 0,E	LD (NN),BC
68	44	LD B,H	BIT 0,H	NEG
69	45	LD B,L	BIT 0,L	RETN
70	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	LD B,A	BIT 0,A	LD I,A
72	48	LD C,B	BIT 1,B	IN C,(C)
73	49	LD C,C	BIT 1,C	OUT (C),C
74	4A	LD C,D	BIT 1,D	ADC HL,BC
75	4B	LD C,E	BIT 1,E	LD BC,(NN)
76	4C	LD C,H	BIT 1,H	-
77	4D	LD C,L	BIT 1,L	RETI
78	4E	LD C,(HL)	BIT 1,(HL)	-
79	4F	LD C,A	BIT 1,A	LD R,A
80	50	LD D,B	BIT 2,B	IN D,(C)
81	51	LD D,C	BIT 2,C	OUT (C),D
82	52	LD D,D	BIT 2,D	SBC HL,DE
83	53	LD D,E	BIT 2,E	LD (NN),DE
84	54	LD D,H	BIT 2,H	-
85	55	LD D,L	BIT 2,L	-
86	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	LD D,A	BIT 2,A	LD A,I
88	58	LD E,B	BIT 3,B	IN E,(C)
89	59	LD E,C	BIT 3,C	OUT (C),E
90	5A	LD E,D	BIT 3,D	ADC HL,DE
91	5B	LD E,E	BIT 3,E	LD DE,(NN)
92	5C	LD E,H	BIT 3,H	-
93	5D	LD E,L	BIT 3,L	-
94	5E	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F	LD E,A	BIT 3,A	LD A,R
96	60	LD H,B	BIT 4,B	IN H,(C)

Decimaal:	Hexadecimaal:	Assemblertaal:	Na code CB:	Na code ED:
97	61	LD H,C	BIT 4,C	OUT (C),H
98	62	LD H,D	BIT 4,D	SBC HL,HL
99	63	LD H,E	BIT 4,E	LD (NN),HL
100	64	LD H,H	BIT 4,H	-
101	65	LD H,L	BIT 4,L	-
102	66	LD H,(HL)	BIT 4,(HL)	-
103	67	LD H,A	BIT 4,A	RRD
104	68	LD L,B	BIT 5,B	IN L,(C)
105	69	LD L,C	BIT 5,C	OUT (C),L
106	6A	LD L,D	BIT 5,D	ADC HL,HL
107	6B	LD L,E	BIT 5,E	LD DE,(NN)
108	6C	LD L,H	BIT 5,H	-
109	6D	LD L,L	BIT 5,L	-
110	6E	LD L,(HL)	BIT 5,(HL)	-
111	6F	LD L,A	BIT 5,A	RLD
112	70	LD (HL),B	BIT 6,B	-
113	71	LD (HL),C	BIT 6,C	-
114	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	74	LD (HL),H	BIT 6,H	-
117	75	LD (HL),L	BIT 6,L	-
118	76	HALT	BIT 6,(HL)	-
119	77	LD (HL),A	BIT 6,A	-
120	78	LD A,B	BIT 7,B	IN A,(C)
121	79	LD A,C	BIT 7,C	OUT (C),A
122	7A	LD A,D	BIT 7,D	ADC HL,SP
123	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	7C	LD A,H	BIT 7,H	-
125	7D	LD A,L	BIT 7,L	-
126	7E	LD A,(HL)	BIT 7,(HL)	-
127	7F	LD A,A	BIT 7,A	-
128	80	ADD A,B	RES 0,B	-
129	81	ADD A,C	RES 0,C	-
130	82	ADD A,D	RES 0,D	-
131	83	ADD A,E	RES 0,E	-
132	84	ADD A,H	RES 0,H	-
133	85	ADD A,L	RES 0,L	-
134	86	ADD A,(HL)	RES 0,(HL)	-
135	87	ADD A,A	RES 0,A	-
136	88	ADC A,B	RES 1,B	-
137	89	ADC A,C	RES 1,C	-
138	8A	ADC A,D	RES 1,D	-
139	8B	ADC A,E	RES 1,E	-
140	8C	ADC A,H	RES 1,H	-
141	8D	ADC A,L	RES 1,L	-
142	8E	ADC A,(HL)	RES 1,(HL)	-
143	8F	ADC A,A	RES 1,A	-
144	90	SUB B	RES 2,B	-
145	91	SUB C	RES 2,C	-
146	92	SUB D	RES 2,D	-
147	93	SUB E	RES 2,E	-
148	94	SUB H	RES 2,H	-
149	95	SUB L	RES 2,L	-

Decimaal:	Hexadecimaal:	Assemblertaal:	Na code CB:	Na code ED:
150	96	SUB (HL)	RES 2,(HL)	-
151	97	SUB A	RES 2,A	-
152	98	SBC A,B	RES 3,B	-
153	99	SBC A,C	RES 3,C	-
154	9A	SBC A,D	RES 3,D	-
155	9B	SBC A,E	RES 3,E	-
156	9C	SBC A,H	RES 3,H	-
157	9D	SBC A,L	RES 3,L	-
158	9E	SBC A,(HL)	RES 3,(HL)	-
159	9F	SBC A,A	RES 3,A	-
160	A0	AND B	RES 4,B	LDI
161	A1	AND C	RES 4,C	CPI
162	A2	AND D	RES 4,D	INI
163	A3	AND E	RES 4,E	OUTI
164	A4	AND H	RES 4,H	-
165	A5	AND L	RES 4,L	-
166	A6	AND (HL)	RES 4,(HL)	-
167	A7	AND A	RES 4,A	-
168	A8	XOR B	RES 5,B	LDD
169	A9	XOR C	RES 5,C	CPD
170	AA	XOR D	RES 5,D	IND
171	AB	XOR E	RES 5,E	OUTD
172	AC	XOR H	RES 5,H	-
173	AD	XOR L	RES 5,L	-
174	AE	XOR (HL)	RES 5,(HL)	-
175	AF	XOR A	RES 5,A	-
176	B0	OR B	RES 6,B	LDIR
177	B1	OR C	RES 6,C	CPIR
178	B2	OR D	RES 6,D	INIR
179	B3	OR E	RES 6,E	OTIR
180	B4	OR H	RES 6,H	-
181	B5	OR L	RES 6,L	-
182	B6	OR (HL)	RES 6,(HL)	-
183	B7	OR A	RES 6,A	-
184	B8	CP B	RES 7,B	LDDR
185	B9	CP C	RES 7,C	CPDR
186	BA	CP D	RES 7,D	INDR
187	BB	CP E	RES 7,E	OTDR
188	BC	CP H	RES 7,H	-
189	BD	CP L	RES 7,L	-
190	BE	CP (HL)	RES 7,(HL)	-
191	BF	CP A	RES 7,A	-
192	C0	RET NZ	SET 0,B	-
193	C1	POP BC	SET 0,C	-
194	C2	JP NZ,NN	SET 0,D	-
195	C3	JP NN	SET 0,E	-
196	C4	CALL NZ,NN	SET 0,H	-
197	C5	PUSH BC	SET 0,L	-
198	C6	ADD A,N	SET 0,(HL)	-
199	C7	RST 0	SET 0,A	-
200	C8	RET Z	SET 1,B	-
201	C9	RET	SET 1,C	-
202	CA	JP Z,NN	SET 1,D	-

Decimaal:	Hexadecimaal:	Assemblertaal:	Na code CB:	Na code ED:
203	CB	-	SET 1,E	-
204	CC	CALL Z,NN	SET 1,H	-
205	CD	CALL NN	SET 1,L	-
206	CE	ADC A,N	SET 1,(HL)	-
207	CF	RST 8	SET 1,A	-
208	D0	RET NC	SET 2,B	-
209	D1	POP DE	SET 2,C	-
210	D2	JP NC,NN	SET 2,D	-
211	D3	OUT (N),A	SET 2,E	-
212	D4	CALL NC,NN	SET 2,H	-
213	D5	PUSH DE	SET 2,L	-
214	D6	SUB N	SET 2,(HL)	-
215	D7	RST 16	SET 2,A	-
216	D8	RET C	SET 3,B	-
217	D9	EXX	SET 3,C	-
218	DA	JP C,NN	SET 3,D	-
219	DB	IN A,(N)	SET 3,E	-
220	DC	CALL C,NN	SET 3,H	-
221	DD	by gebruik IX	SET 3,L	-
222	DE	SBC A,N	SET 3,(HL)	-
223	DF	RST 24	SET 3,A	-
224	E0	RET PO	SET 4,B	-
225	E1	POP HL	SET 4,C	-
226	E2	JP PO,NN	SET 4,D	-
227	E3	EX (SP),HL	SET 4,E	-
228	E4	CALL PO,NN	SET 4,H	-
229	E5	PUSH HL	SET 4,L	-
230	E6	AND N	SET 4,(HL)	-
231	E7	RST 32	SET 4,A	-
232	E8	RET PE	SET 5,B	-
233	E9	JP (HL)	SET 5,C	-
234	EA	JP PE,NN	SET 5,D	-
235	EB	EX DE,HL	SET 5,E	-
236	EC	CALL PE,NN	SET 5,H	-
237	ED	-	SET 5,L	-
238	EE	XOR N	SET 5,(HL)	-
239	EF	RST 40	SET 5,A	-
240	F0	RET P	SET 6,B	-
241	F1	POP AF	SET 6,C	-
242	F2	JP P,NN	SET 6,D	-
243	F3	DI	SET 6,E	-
244	F4	CALL P,NN	SET 6,H	-
245	F5	PUSH AF	SET 6,L	-
246	F6	OR N	SET 6,(HL)	-
247	F7	RST 48	SET 6,A	-
248	F8	RET M	SET 7,B	-
249	F9	LD SP,HL	SET 7,C	-
250	FA	JP M,NN	SET 7,D	-
251	FB	EI	SET 7,E	-
252	FC	CALL M,NN	SET 7,H	-
253	FD	by gebruik IY	SET 7,L	-
254	FE	CP N	SET 7,(HL)	-
255	FF	RST 56	SET 7,A	-

APPENDIX B.

Woordenboek.

Hieronder vindt U nadere uitleg van een aantal woorden die min of meer met computers verbonden zijn. Niet alle woorden komen in dit boek voor, maar worden wel regelmatig gebruikt wanneer over computers gesproken wordt.

ADRES - Een adres is normaal gesproken de plaats waar iemand woont en wordt aangegeven met een straatnaam, een nummer en de woonplaats. Bij computers komt een adres neer op een geheugenplaats waar iets neergezet kan worden. Een adres wordt aangegeven door een getal, bijvoorbeeld 15000. De SHARP bezit in totaal 65535 adressen.

ASSEMBLEREN - Dit woord is beter te begrijpen wanneer U kijkt naar APPENDIX A, waarin alle mnemonics getoond worden. Assembleren houdt in dat de assemblercodes omgezet worden naar een hexadecimaal getal. NOP wordt bijvoorbeeld 00H. Het omgekeerde van assembleren is disassembleren, daarbij wordt een hexadecimaal getal omgezet in een assemblercode. 07H wordt bijvoorbeeld RLCA.

BANKSWITSCHING - De SHARP MZ-800 heeft meer dan één geheugen. Denkt U bijvoorbeeld eens aan de 16K VIDEO-RAM die alleen te bereiken is door bankswitsching. Bankswitsching houdt in dat er overgeschakeld wordt van de ene geheugenbank naar de andere. Er wordt dus gewoon van het ene geheugen naar het andere gesprongen.

BASIC - Beginners All purpose Symbolic Instruction Code oftewel een taal voor beginners die voor alle doeleinden bedoeld is. Dat dit niet helemaal klopt weet U misschien al. Het is niet direct een taal voor beginners en ook niet een taal voor alle doeleinden. Wel is het een gemakkelijk te leren taal waar veel mee gedaan kan worden.

BINAIR - Een getal kan op drie manieren weergegeven worden, namelijk decimaal, hexadecimaal en binair. Een binair getal kan uit 8, 16 of 32 cijfercombinaties bestaan. Een binair getal is opgebouwd uit nullen en enen. Bij een cijfercombinatie bestaande uit 8 cijfers kunnen maximaal 256 combinaties gemaakt worden. Een voorbeeld van een binair getal:
00001100 (12 decimaal en 0CH hexadecimaal)

BIT - Wat een bit precies is kan het beste uitgelegd worden aan de hand van het woord dat hiervoor besproken is. Een binair getal is namelijk uit bits opgebouwd. Elk cijfertje dat in een binair getal voor komt is een bit. Een bit kan 0 of 1 zijn en is de kleinste eenheid die binnen het getallenstelsel van de SHARP voor komt.
8 Bits bij elkaar vormen samen een byte en een byte is precies de ruimte op één geheugenplaats, oftewel de ruimte op één adres.

BTX - U komt deze 'uitdrukking' erg vaak tegen en dat niet alleen bij de SHARP MZ-800, maar ook bijvoorbeeld bij de PC. Een BTX-file houdt bij de SHARP in dat het programma geschreven is in BASIC-800 of in BASIC-700. Zo kunt U bijvoorbeeld ook BSD-files aantreffen. Die files horen meestal bij bestanden of tekstverwerkers. Wat een OBJ-file is, wordt verderop besproken.

BYTE - Zoals bij BIT al gezegd is, is een byte een eenheid die uit 8 bits opgebouwd is en een byte is precies de inhoud van één geheugenplaats of adres.
Naast de eenheid byte bestaat ook nog de eenheid kbyte oftewel kilobyte. 1 Kilobyte is gelijk aan duizend bytes. Zo zijn er ook eenheden van miljoen bytes (megabyte) en miljard bytes (gigabyte), maar die worden zelden tot nooit genoemd in combinatie met de SHARP MZ-800.

CURSORTOETSEN - De cursortoetsen of pijltjestoetsen worden meestal gebruikt bij spelletjes en bevinden zich helemaal rechts op het toetsenbord. Het voordeel van deze toetsen is dat ze gemakkelijk gebruikt kunnen worden ipv een joystick. Mensen die dus geen joystick hebben kunnen op die manier ten alle tijden een spelletje spelen.

DECIMAAL - Zoals al gezegd bij binair, kan een getal op drie manieren weergegeven worden en decimale weergave behoort daar ook bij.
De decimale weergave is de weergave zoals het getal normaal ook geschreven wordt. 20 blijft decimaal dus gewoon 20 en 900.356 blijft decimaal 900.356.
Naast de decimale weergave staat de hexadecimale weergave. Hexadecimale weergave werkt met een getallenstelsel van 16 (van 0 t/m F). In appendix A kunt U goed zien wat de hexadecimale waarde van de getallen van 0 t/m 255 is.

DISKETTE - Een diskette is een opslagmedium voor computerinformatie en komt voor in drie verschillende formaten, waarvan 5 1/4 inch en 3 1/2 inch de meest gangbare zijn. Bij de SHARP wordt alleen het eerstgenoemde formaat gebruikt.

FUNCTIETOETSEN - De naam zegt het al, deze toetsen hebben een bepaalde functie en zijn te vinden links bovenaan het toetsenbord. Wat voor functie de toetsen hebben kunt U zelf bepalen aan de hand van een bepaalde instructie. Door een druk op een functietoets verschijnt er automatisch een woord in beeld en als dat bijvoorbeeld RUN is, kan dat ook nog automatisch uitgevoerd worden. Het voordeel is dat U niet het hele woord meer in hoeft te typen, maar alleen een toets in hoeft te drukken.

GEHEUGEN - In een computergeheugen kan informatie gezet worden, maar er staat ook informatie in. Die vaste informatie (ROM) blijft er altijd in staan, maar die andere informatie (RAM) verdwijnt bij het uitzetten van de computer.

HARDWARE - Een computer en alles wat er bij hoort is simpelweg op te delen in software en hardware. Om het heel gemakkelijk te zeggen, kun je zeggen dat alles wat breekbaar is bij de software hoort en alles wat niet breekbaar is bij de hardware hoort. Zo gemakkelijk zit het echter niet in elkaar. Een betere verdeling is om te zeggen dat alle programma's en hun opslagmedia (cassette, floppy's, enz.) tot de software behoren en al het andere tot de hardware. Deze verdeling is ook niet helemaal juist, maar komt toch een aardig eind in de richting.

IC - Een ander woord voor een ic is chip. De ic's zijn de belangrijkste onderdeeltjes van een computer. Een computer bestaat hoofdzakelijk alleen uit ic's die via een plaat met elkaar in verbinding staan. Door de ontwikkelingen van de laatste jaren worden er steeds betere ic's gemaakt, waardoor de computers sneller, compacter en ook beter worden.

KAART - Een kaart is een plaat met daarop een aantal ic's die ervoor zorgt dat randapparatuur of iets dergelijks op een bepaalde computer kan draaien. Soms is randapparatuur helemaal op zo'n kaart zelf gemaakt, kijk bijvoorbeeld maar eens naar de RAM-kaart en de kaartmodem.

KABEL - Een kabel verbindt randapparatuur met de computer net als een telefoonkabel een telefoon met de centrale verbindt.

MACHINETAAL - Machinetaal is de taal die de computer begrijpt. In de microprocessor staan allerlei codes opgeslagen die de computer iets laten doen als een code aangeroepen wordt. Machinetaal houdt in dat deze codes rechtstreeks ingevoerd worden in de computer, zodat de computer razendsnel kan werken. Bij BASIC is het zo dat alles eerst omgezet moet worden in machinetaal voordat de computer iets doet, vandaar dat BASIC niet zo snel werkt.

MICROPROCESSOR - De microprocessor bepaalt welke soort machinetaal ingevoerd moet worden. Er zijn heel veel verschillende microprocessoren die vaak met verschillende snelheden werken. De snelheid waarmee ze werken bepaalt voor een groot deel de snelheid waarmee de hele computer werkt. Het is dus belangrijk dat de microprocessor snel werkt om een snelle computer te krijgen.
De microprocessor van de SHARP is de bekende Z80-A microprocessor die in de meeste home-computers gebruikt wordt.

MNEMONICS - Mnemonics zijn de machinetaalcodes die al eerder genoemd zijn bij uitleg van het woord machinetaal. In appendix A staan alle mnemonics die de Z80-A microprocessor kent. Daarbij ziet U ook dat er op een behoorlijk aantal plaatsen niets staat. Oorspronkelijk waren die plaatsen ook bedoeld voor een bepaalde mnemonic, maar staat er geen mnemonic. In werkelijkheid is er dus wel iets op die lege plaatsen, maar heeft dat geen invloed op de computer bij gebruik.

MONITOR - Bij de SHARP kennen we twee verschillende monitoren, namelijk het gewone beeldscherm als monitor en de monitor in de betekenis van een scherm om in het geheugen van de SHARP te kunnen werken, de bekende BASIC- en ROM-monitor.

OBJ Een OBJ-file is een programma dat in machinetaal geschreven is of het zijn data die in de vorm van een machinetaalprogramma gesaved zijn. Nog even een opmerking tussendoor: Het gaat niet altijd op dat een machinetaalprogramma een OBJ-file is en dat een BASIC-programma een BTX-file is. Dit heeft te maken met het feit dat op een eenvoudige manier een BASIC- of machinetaalprogramma op elke manier gesaved kan worden, dus ook als BSD-file.

PROGRAMMEREN - Programmeren kan het maken van een klein programmaatje inhouden, maar ook het maken van programma's als DBASE e.d. behoort tot het programmeren. Programmeren is niets anders dan het maken van een programma dmv het op logische volgorde zetten van allerlei instructies. Dit geldt voor elke computertaal of het nu PASCAL, C of BASIC is. Wanneer de instructies niet in de goede volgorde staan of er ontbreekt iets, zal het programma niet voor 100% werken. Om goed te kunnen programmeren is het noodzakelijk om een goed inzicht te hebben.

QUICK-DISK - Na de cassetterecorder komt de QUICK-DISK als meest gebruikte opslagmedium voor de SHARP MZ-800. In het begin van dit boek wordt de QUICK-DISK uitvoerig besproken.

RAM - Read Access Memory oftewel het deel van het geheugen dat vrij toegangelijk is en vrij te veranderen is, maar waarvan de inhoud geheel leeg raakt bij het uitzetten van de computer.

ROM - Read Only Memory oftewel het deel van het geheugen dat niet vrij toegangelijk is en niet zomaar te veranderen is. Dit deel van het geheugen is ook altijd hetzelfde wanneer de computer aangezet wordt ook al worden er veranderingen in uitgevoerd via een omweg.

ROTEREN - Doordraaien is een woord dat ongeveer neer komt op roteren. In werkelijkheid is roteren niets anders dan het verschuiven van de bits binnen een byte naar links of naar rechts. Daarbij is het afhankelijk van de soort rotatie wat er met het eerste en/of het laatste bit gebeurt.

SOFTWARE - Bij hardware is al min of meer uitgelegd wat software precies inhoudt.

TOETSENBORD - Het woord toetsenbord is al een aantal keren gevallen bij het bespreken van een aantal woorden. Het toetsenbord is het geheel van alle toetsen bij elkaar en het gebied waarin de toetsen zich bevinden.
Bij een aantal computers, zoals de meeste PC's, is het zo dat het toetsenbord zich los van de rest van de computer bevindt. Bij de SHARP zit het toetsenbord aan de rest van de computer vastgeschroefd en is in principe dus ook van de rest van de computer te verwijderen.

APPENDIX C: BASIC-INSTRUCTIES EN AFKORTINGEN.

In deze appendix zullen alle BASIC-instructies met hun bijbehorende afkorting getoond worden. Uiteraard is er niet voor iedere instructie een afkorting.
Wanneer U een afkorting intypt, geeft dat hetzelfde resultaat als de hele instructie.
Ook zullen enkele vaste variabelen besproken worden.
We beginnen met deze variabelen die allemaal een vaste naam hebben.
Wilt U de inhoud van een variabele zien dan kunt U dat doen door er PRINT voor te zetten.

TI$ - In deze variabele bevindt zich de tijd. De tijd begint bij 000000 wanneer de computer aangezet wordt. De eerste twee getallen zijn het aantal uren. Het derde en vierde getal zijn samen het aantal minuten en het vijfde en zesde getal vormen het aantal seconden. Deze variabele kunt U zelf veranderen door bijvoorbeeld TI$="123540" in te voeren.
SIZE - In deze variabele is de vrije geheugenruimte opgeslagen.
ERN - Wanneer er een fout in een programma(regel) zit, bevat deze variabele het overeenkomstige foutnummer.
ERL - Wanneer er een fout in een programma(regel) zit, bevat deze variabele het overeenkomstige regelnummer.
CSRH - Deze variabele geeft de kolompositie van de cursor aan.
CSRV - Deze variabele geeft het nummer van de rij waarop de cursor is.
POSH - Deze variabele bevat de X-coördinaat van de grafische plaatsaanduiding.
POSV - Deze variabele bevat de Y-coördinaat van de grafische plaatsaanduiding.

Voor alle functies, zowel numeriek als functies voor charactertoepassingen verwijzen wij U naar de bladzijden 5-10 t/m 5-15 van het handboek. Op deze bladzijden zult U ook nog enkele andere zaken vinden.

Dan volgen nu alle SHARP-instructies en statements.

AUTO - Afkorting A.
DELETE - Afkorting D.
LIST - Afkorting L.
SEARCH - Afkorting SE.
RENUM - Afkorting REN.
NEW - Geen afkorting.
NEW ON - Afkorting NEW O.
CLR - Geen afkorting.
CONT - Afkorting C.
RUN - Afkorting R.
CLS - Afkorting CL.
CONSOLE - Afkorting CONS.
CURSOR - Afkorting CU.
REM - Geen afkorting.
LET - Afkorting LE.
STOP - Afkorting S.
END - Afkorting EN.
FOR~TO~NEXT - Afkorting F.~TO~N.
LABEL - Afkorting LA.

GOTO - Afkorting G.
ON Afkorting O.
GOSUB - Afkorting GOS.
RETURN - Afkorting RE.
IF~THEN~ELSE - Afkorting IF~TH.~EL.
PRINT Afkorting ? of P.
PRINT USING - Afkorting ?USI.
INPUT - Afkorting I.
GET - Afkorting GE.
DIM - Afkorting DI.
READ~DATA - Afkorting REA.~DA.
RESTORE - Afkorting RES.
DEF FN - Geen afkorting.
TRON - Afkorting T.
TROFF - Afkorting TROF.
DEF KEY - Afkorting DEF K.
KEY LIST - Afkorting K.L.
INIT - Afkorting INI.
BYE - Afkorting B.
BOOT - Afkorting BOO.
WAIT - Afkorting W.
DIR - Geen afkorting.
RUN - Afkorting R.
LOAD - Afkorting LO.
SAVE - Afkorting SA.
VERIFY - Afkorting V.
RENAME - Afkorting RENA.
CHAIN - Afkorting CH.
MERGE - Afkorting M.
WOPEN# - Afkorting WO.#
PRINT# - Afkorting ?#
ROPEN# - Afkorting RO.#
XOPEN# - Afkorting X.
INPUT# - Afkorting I.#
EOF# - Afkorting EO.#
CLOSE# - Afkorting CLO.#
KILL# - Afkorting KI.#
DEFAULT - Afkorting DEF.
COLOR - Afkorting COL.
PAL - Geen afkorting.
SET - Geen afkorting.
RESET - Afkorting RESE.
LINE - Afkorting LIN.
BLINE - Afkorting BL.
BOX - Afkorting BO.
CIRCLE - Afkorting CI.
PAINT - Afkorting PAI.
PATTERN - Afkorting PAT.
POSITION - Afkorting POS.
SYMBOL - Afkorting SY.
POINT - Afkorting POI.
MUSIC - Afkorting MU.
TEMPO - Afkorting TE.
SOUND - Afkorting SO.
NOISE - Afkorting NO.

PTEST - Afkorting PT.
PMODE Afkorting PM.
PCOLOR - Afkorting PC.
PSKIP - Afkorting PS.
PAGE - Afkorting PA.
PLINE PLI.
RLINE Afkorting RL.
PMOVE - Afkorting PMOV.
RMOVE Afkorting RM.
PHOME Afkorting PH.
HSET Afkorting H.
GPRINT Afkorting GP.
AXIS - Afkorting AX.
PCIRCLE - Afkorting PCI.
HCOPY Afkorting HC.
PLOT - Afkorting PL.

Veel instructies kunnen ook in combinatie met /P gebruikt worden om deze instructies voor de plotter of de printer te gebruiken. In het handboek kunt U precies zien welke instructies dit zijn.
Voor verdere details over alles wat met BASIC te maken heeft verwijzen wij U eveneens naar het handboek. Het handboek geeft U meer dan voldoende uitleg over alle BASIC-instructies en al het andere wat met BASIC te maken heeft. Dat is natuurlijk alleen voor de mensen die die uitleg begrijpen en dat is iets wat vaak niet het geval is.
U ziet dat hier dus duidelijk geen BASIC-cursus geschreven is. Deze appendix is alleen bedoeld om een overzicht te hebben van alle instructies en statements en hun afkortingen. Het zal ook weinig nut hebben om een BASIC-cursus te schrijven, omdat het toch in grote lijnen overeen zal komen met het bestaande handboek en dit boek is duidelijk niet bedoeld om U bepaalde dingen uit andere boeken nog' eens te laten zien, maar het boek is bedoeld om U hoofdzakelijk nieuwe dingen te laten zien.
Een complete BASIC-cursus heeft waarschijnlijk ook weinig nut, omdat U altijd met bepaalde vragen blijft zitten. Om het ontbreken van een complete uitleg van de BASIC op te vangen hebben wij het volgende voorstel. Mensen die graag iets willen weten over de BASIC en die daar niet uit kunnen komen door in het handboek te kijken stellen wij in staat om schriftelijk vragen over de BASIC te stellen en dat kunt U doen bij Arjan Habing (zie appendix D voor adres e.d.). Zolang de vragen nog binnen de perken blijven (U moet bijvoorbeeld niet om een uitgebreid programma vragen) en U een retourenvelop met postzegel bijsluit, krijgt U binnen een maand tijd uitgebreide uitleg van Arjan teruggestuurd. Telefonische vragen over de BASIC en vragen over de plotter kunnen helaas niet worden beantwoord. Dit heeft te maken met het feit dat Arjan thuis helaas geen plotter heeft en de programma's dus ook niet uit kan proberen. Hetzelfde geldt voor de RAM-kaart en de FLOPPY-DISK.
In appendix D zal vermeld staan voor welke informatie U nog meer bij ons terecht kunt. In ieder geval niet voor vragen over de plotter, de RAM-kaart en de FLOPPY-DISK. Deze apparaten zullen in aanvullingen op dit boek, wanneer die tenminste zullen komen, uitgebreid behandeld worden. Helaas konden we dat in dit boek nog niet doen en zullen wij proberen zo snel mogelijk deze apparaten in huis te krijgen om U er meer over te kunnen vertellen.

APPENDIX D: SHARP-CLUBS EN ADRESSEN.

In deze appendix vint U informatie over de drie grootste clubs voor SHARP MZ-800 bezitters en tevens zullen enkele namen en adressen vermeld worden van mensen waar U eventueel terecht kunt voor vragen over Uw SHARP MZ-800.

Club 1: SHARP Computer Club Elst.

Deze club is opgezet door een aantal fanatieke SHARP gebruikers uit de omgeving van Elst (Gelderland) die graag iets meer wilden doen met hun SHARP dan alleen de spelletjes ADVOKA en HEAD DRIVER spelen.
Binnen een aantal maanden is dit clubje uitgegroeid van een roekeloze computerclub met een wild clubblad tot een volwassen club met een professioneel clubblad.
Momenteel heeft deze club ongeveer 140 leden, waarvan het grootste deel in het bezit is van een SHARP MZ-800. Deze leden komen uit alle windstreken van het land en zelfs uit het buitenland (België). Ook de makers van dit boek zijn aangesloten bij deze club, maar daar volgt later meer over.
Wat heeft SCCE U eigenlijk allemaal te bieden en hoe hoog (of misschien beter gezegd: hoe laag) is het lidmaatschapsgeld?

-SCCE houdt 6 bijeenkomsten per jaar in het gebouw Onder de Toren te Elst. Op die bijeenkomsten staan elke keer zeker 10-15 SHARPS en nog een tiental andere computers en er komen elke keer zeker 50-75 bezoekers en deze aantallen worden elke keer weer hoger.
Op de bijeenkomsten worden programma's, ideeën en nog veel meer andere zaken uitgewisseld. Tevens worden er clubartikelen, zoals clubcassettes, QD's en andere zaken verkocht.
Er bestaat ook een idee om in de toekomst misschien regionale bijeenkomsten door het hele land te gaan houden. Daar is natuurlijk geld voor nodig en er moeten nog een aantal leden bij komen. Op dat gebied bestaat er goede hoop. Wie weet is er volgend jaar ook wel een bijeenkomst van SCCE bij U in de buurt.

-SCCE geeft IEDERE MAAND een clubblad uit dat ieder lid GRATIS thuisgestuurd krijgt. In dit clubblad staat zeer nuttige informatie over en voor Uw SHARP, zoals POKES, programma's, trucs en tips, een machinetaalcursus, een cursus over grafische toepassingen en ga zo maar door. Als U weet dat de makers van dit boek ook vaste medewerkers aan het clubblad zijn, kunt U wel nagaan dat de inhoud wel goed moet zijn. (We gaan er natuurlijk wel vanuit dat U de inhoud van dit boek waardeert, anders klopt de voorgaande uitspraak niet helemaal.)

-SCCE geeft clubcassettes uit met op elke cassette ongeveer 22 programma's. Die programma's variëren van spelletjes tot uitgebreide bestanden en zijn de moeite van het aanbevelen waard. De prijs van een clubcassette bedraagt slechts F15,= en daarvoor krijgt U hem ook nog thuisgestuurd. Ook aan deze clubcassettes hebben de makers van dit boek hun medewerking verleend in de vorm van het aanbieden van zeer nuttige tot zeer leuke programma's zowel in BASIC als in MACHINETAAL.

-SCCE onderhoudt contacten met verschillende bedrijven op computergebied en kan op die manier regelmatig een aanbieding doen aan de leden door een printer bijvoorbeeld F50,= goedkoper aan te bieden of QD's 10% goedkoper aan te bieden. Zo kunt U bij de aanschaf van computertoebehoren altijd nog een paar gulden besparen.

We kunnen op deze manier nog wel een tijdje doorgaan over SCCE, maar dan blijft er geen ruimte meer over voor de andere clubs en die moeten natuurlijk niet vergeten worden.
Wat U nog wel moet weten is het bedrag dat U betaald voor een lidmaatschap en dat is slechts F35,= voor een heel jaar en als U nagaat dat daar al F24,= van wordt uitgegeven aan het maken en verzenden van het clubblad, kunt U wel nagaan dat U Uw geld er zeker wel weer uithaalt in de vorm van nuttige informatie en allerlei aanbiedingen en natuurlijk niet te vergeten: De gezellige bijeenkomsten waar iedereen heel gemoedelijk met elkaar omgaat. Niet voor niets is het motto van SCCE: Voel je thuis binnen je eigen club.
Als makers van dit boek kunnen wij U sterk aanraden lid te worden van deze club, U zult er waarschijnlijk geen spijt van krijgen. Achteraf hebben wij zelfs nog veel kunnen leren bij deze club.
Veel mensen zitten echter met het probleem dat ze niet bepaald dicht in de buurt van Elst wonen en daardoor hoge reiskosten moeten maken. Dat probleem is echter minder groot dan U denkt. Er is altijd wel iemand bij U in de buurt die ook lid is van SHARP Computer Club Elst en waarmee U vast en zeker wel mee kunt rijden en dan wordt de rit al een stuk minder duur. Het zou natuurlijk ideaal zijn als er vier mensen mee zouden rijden, want dan worden de kosten minimaal. Misschien is het zelfs helemaal niet nodig om helemaal naar Elst te rijden, want de kans zit er in dat er binnenkort regionale bijeenkomsten zullen komen.
Wacht U niet af tot het zover is, maar wordt meteen lid. U kunt zich opgeven als lid bij de penningmeester. U kunt natuurlijk ook de boot eerst nog even afwachten en eerst nog wat informatie opvragen bij de voorzitter of de penningmeester, zij willen U graag van dienst zijn.
Misschien tot ziens bij SHARP Computer Club Elst.

De voorzitter: Han Bleeker
De Zuiling 91
6662 RB Elst
tel:08819-76380
(18.00-20.00 uur)

De penningmeester: Jos Bos
Lijsterstraat 1
6991 ES Rheden
tel:08309-54483
(19.00-21.00 uur)

Club 2: SHARP Computer Club Edam.

Om verwarringen te voorkomen, zal deze club afgekort worden met SCCEd.
Deze club bestaat eigenlijk al lang, maar is pas laat een zelfstandige club geworden. In eerste instantie was SCCEd nog een afdeling van de SHARP MZ-gg van de HCC, maar door ontevredenheid over de gang van zaken binnen de HCC MZ-gg is deze club zelfstandig geworden en draait nu beter dan voorheen.
SCCEd is een klein beetje te vergelijken met SCCE. Het grootste verschil is echter dat SCCEd minder vaak een clubblad uitbrengt en dat voor een lidmaatschapsprijs die even hoog is als die van de SHARP club uit Elst. Er worden zes bijeenkomsten per jaar gehouden (op zondag) en voor elke bijeenkomst komt een clubblad, SHARPIE genaamd, uit. Het clubblad van SCCEd is, evenals het clubblad van SCCE, professioneel samengesteld en zeer de moeite waard.
SCCEd heeft voor de leden een grote bibliotheek met boeken en software om te huren of te kopen. Een nadeel hierbij is, is dat voor het huren van een boek altijd nog een geringe vergoeding, die naar onze mening te hoog is, wordt gevraagd.

Wanneer we SCCEd in het geheel bekijken, heeft deze club minder te bieden dan SCCE. Daar staat wel tegenover dat de club alles serieuser aanpakt in de vorm van werkgroepen en een boeken- en software-bibliotheek. Daarin is toch nog een klein beetje de verbondenheid met de HCC SHARP MZ-gg weer te vinden.
Als makers van dit boek hebben we tot nu toe weinig te maken gehad met deze club, maar dat zal wel veranderen door het uitkomen van dit boek. Dan krijgen we gelijk een beter inzicht in deze club en kunnen we er betere uitspraken over doen.
De bijeenkomsten worden goed bezocht, maar de sfeer is net iets minder dan bij SHARP Computer Club Elst.
SCCEd telt ongeveer 120 leden en werkt ook op kleine schaal samen met SCCE. Misschien zal de samenwerking zich in de toekomst uitbreiden. Hopelijk zal de samenwerking met de SHARP MZ-gg ook beter worden. Momenteel is daar absoluut geen sprake van.
Voor meer informatie kunt U bellen of schrijven naar:

De voorzitter: Joop Veenstra
tel:02993-67170

De secretaris: Jan Rutte
tel:02550-33796

Club 3: De SHARP MZ-gebruikers groep.

De vorige twee clubs waren zelfstandige clubs, maar de SHARP MZ-gg is aangesloten bij de grootste computerclub van ons land en is tevens ook de grootste SHARP club van ons land.
Door het dumpen van de SHARP MZ-800 bij KWANTUM kreeg de SHARP MZ-gg enkele jaren terug te maken met een enorme stijging van het aantal leden. De mensen konden vaak geen informatie en programma's voor de SHARP vinden en zochten toen hun heil bij de SHARP MZ-gg. Waarom keken ze dan niet naar andere clubs? Ten eerste waren die er toen nog niet en ten tweede stond het adres van de SHARP MZ-gg in het meegeleverde handboek.
De afgelopen twee jaar zijn er weer redelijk veel mensen vertrokken bij deze club. Waar 6000 mensen zijn, zijn 6000 verschillende ideeën en de interesses liggen ook behoorlijk verdeeld. Omdat niet aan alle interesses kon worden voldaan, hebben veel mensen hun lidmaatschap opgezegd. Nu zit de SHARP MZ-gg met veel regionale bijeenkomsten die vaak heel mager bezocht worden, simpel en alleen omdat er teveel bijeenkomsten zijn en er door de terugval weinig te beleven valt. Er zijn dan ook een aantal regio's die gaan vervallen om meer bezoekers naar de andere bijeenkomsten te trekken.
U moet ook niet schrikken van het bedrag dat U moet betalen om lid te kunnen worden van de HCC en ook het eigen blad van de SHARP MZ-gg te kunnen ontvangen. U betaalt namelijk F60,= voor een lidmaatschap op de HCC (en ook nog eens F10,= voor een lidmaatschap op de SHARP MZ-gg) en nog eens F15,= om 6 nummers van het SHARP-blad ARGONAUT en een LEDENBRIEF te kunnen ontvangen. Bij elkaar komt alles op F85,=, een aardige investering. Als U ziet wat U daar voor terug krijgt, is dat bedrag misschien toch iets aan de hoge kant.
Wat U er allemaal voor terug krijgt kunt U lezen op de volgende bladzijde.

-U krijgt maandelijks de HCC-Nieuwsbrief, een professioneel computer-blad dat zeker de moeite waard is. Het probleem is echter dat er niets voor de SHARP in staat en dat is toch uiteindelijk waar het om draait.
-Er worden maandelijks regionale bijeenkomsten gehouden, behalve in de vakantiemaanden. Er zal zeker ook bij U in de buurt wel een regionale bijeenkomst gehouden worden.
-De SHARP MZ-gg biedt U de mogelijkheid om tegen een laag bedrag QD's aan te schaffen. Tevens kunnen andere SHARP artikelen vaak beneden de winkelprijs aangeschaft worden.
-Er bestaat ook een mogelijkheid om clubcassettes te kopen voor F25,= per stuk. De kwaliteit van de cassettes is misschien niet erg geweldig, maar de variatie in programma's is enorm groot. Onder de 20 programma's die op één cassette staan, bevindt/bevinden zich meestal wel één of meer programma's die voor U interessant zijn.
-De SHARP MZ-gg kent ook de opsplitsing in werkgroepen. Voor specifieke problemen kunt U dus altijd bij zo'n werkgroep terecht. Daar komt gelijk ook het probleem van de grote club om de hoek kijken. Iedereen stelt specifieke vragen aan die werkgroepen, daardoor is het soms niet mogelijk dat de werkgroep een probleem voor U op kan lossen, maar dat komt niet erg vaak voor.
Alleen SCCE kent deze opsplitsing in werkgroepen niet, maar op het moment van het uitkomen van dit boek is er bij die club zeker ook een dergelijke indeling gemaakt.

Natuurlijk heeft de SHARP MZ-gg U nog wel meer te bieden, maar dit was zo'n beetje het belangrijkste. Het is de SHARP-club die U het meeste te bieden heeft en ook een professioneel blad heeft. Het is echter ook een club met minder goede punten, zoals het hoge lidmaatschapsbedrag
We zijn zelf momenteel niet meer aangesloten bij de SHARP MZ-gg, maar daar zal snel verandering in komen. De SHARP MZ-gg kan ieder lid gebruiken, omdat het geld voor deze club uit de verkoop van SHARP artikelen moet komen. Van het lidmaatschapsbedrag voor de HCC gaat slechts F5,= naar de SHARP MZ-gg. U begrijpt nu ook waarom de cassettes misschien iets meer kosten dan normaal.
Om mensen toch volledig te kunnen laten profiteren van alle artikelen die de SHARP MZ-gg te bieden heeft zonder daarvoor een lidmaatschap op de HCC te hebben, bestaat tegenwoordig de mogelijkheid om als niet HCC-lid alle clubartikelen van de SHARP MZ-gg te kopen. Ook bestaat voor die mensen de mogelijkheid om een abonnement op de ARGONAUT te nemen.
Wij kunnen U van harte aanbevelen om van deze mogelijkheid gebruik te maken, zodat de SHARP MZ-gg meer geld in kas zal krijgen en zo nog meer voor U kan doen.
U denkt misschien dat we nu een spelletje in ons eigen voordeel spelen door zowel met de SHARP MZ-gg als SCCE vriendjes te blijven en ze allebei te promoten. Dat is misschien wel een beetje waar, maar we zullen in de toekomst ook informatie en programma's aan deze clubs doorspelen. Daar kunt U weer van profiteren en daar gaat het toch om.
Voor meer informatie over de SHARP MZ-gg kunt U terecht op de regionale of landelijke bijeenkomsten of U kunt contact opnemen met:

De voorzitter: Hans Niessen 033-945286 De penningmeester: Albert vd Pol 02993-61823

De secretaris :Jan Bancken, Lis 14, 1273 CD Huizen (schriftelijk)

Na wat meer informatie over de verschillende SHARP-clubs gegeven te hebben volgen nu een aantal adressen waar U terecht kunt voor Uw vragen of opmerkingen:

Voor alles over het boek, over Neptunes Software, allerlei tabellen of aanvullende informatie kunt U terecht bij de makers van dit boek en tevens de makers van Neptunes Software.

Probeert U in eerste instantie: Arjan Habing
Prakkestraat 19
7741 CK Coevorden
05240-14646
(Het liefst schriftelijk. U ontvangt dan binnen een maand bericht. U moet natuurlijk wel een retourenvelop met postzegel bijsluiten.)

Mocht Arjan niet te bereiken zijn dan kunt U het proberen bij:

Mark de Rover
Geraniumstraat 11
3135 XE Vlaardingen
010-4352722

(Ook hier het liefst schriftelijke vragen. Vergeet U niet de retourenvelop met postzegel bij te sluiten!)

Voor vragen over alles wat met hardware te maken heeft kunt U terecht bij:

C. Gans
Ketelmeer 10
1509 HJ Zaandam
075-172050

(Het liefst telefonische vragen. U kunt bellen tussen 20.00u en 22.00u, uitgezonderd op zondag.)

Voor vragen over o.a. de BASIC-interpreter en andere zaken in die richting kunt U terecht bij:

H. Smits
Zweepslag 7
6852 ED Huissen
085-250006

(Ook hier het liefst telefonische vragen.)

Voor vragen over CPM kunt U terecht bij: Ton de Pan
Venezuelaweg 10C
3193 BA Hoogvliet
010-4161033

U moet niet verwachten dat al deze mensen ieder moment voor U klaar kunnen staan. Het is een dienst van hun om buiten hun normale werkzaamheden om ook nog diensten te verlenen aan kopers van dit boek. De woorden:" Helaas, ik heb het te druk." kunt U dus ook verwachten.

MEER OVER NEPTUNES PRODUCTIONS.

NEPTUNES PRODUCTIONS is een VOF met als doelstelling het verkopen van software en computerboeken voor de midden- en kleinhandel. Daar behoort natuurlijk ook U, de kleingebruiker, bij.

Om iedereen software naar eigen wens te laten kopen bestaat de mogelijkheid dat wij zelf software voor U maken.
Daarvoor bestelt U via een formulier een programma en dat programma maken wij helemaal zoals U dat wilt. Uiteraard moet het wel binnen de grenzen van de SHARP MZ-800 blijven.
Wanneer U een specifiek programma voor U zelf wilt hebben, stuurt U dan even een briefje of briefkaartje. U krijgt dan zo spoedig mogelijk bericht van ons terug over de prijs e.d.

Momenteel hebben we al een aantal programma's die voor de verkoop geschikt zijn en dat zijn:

WORM V2.0 In dit multi-screen (5 schermen) spel is het de bedoeling dat U, als worm zijnde, een heel veld leeg eet. Er is echter een probleem en dat is dat U langer wordt door het eten. Aangezien U Uzelf absoluut niet mag raken, is dat langer worden een hinderlijk probleem en dat hinderlijke probleem wordt steeds erger naarmate het spel vordert.

PICTURESHOW Een heel mooi programma dat ongeveer 25 plaatjes tekent en daarbij alle 16 kleuren gebruikt. Dit programma is veel mooier dan alle voorgaande programma's waarin plaatjes worden vertoond.

TONGO Een spel voor twee personen naar aanleiding van het televisie-spel LINGO. In totaal wordt er uit 1000 woorden gekozen.

CASTLE DUNGEON Onder in een donker kasteel moet U met een klein zaklampje het hele gewelf doorzoeken naar tijdbommen. Er zijn veel deuren en verschillende vertrekken in het kasteel. Het zal dus absoluut geen gemakkelijke opgave worden.

Helaas zijn dit nog niet voldoende programma's om een cassette mee te vullen, want we willen toch ongeveer 15 programma's op één cassette en die cassette zal dan rond de F25,= gaan kosten. Op de cassette staan alleen maar kwaliteitsprogramma's, de prijs is dus heel erg laag.

Hopelijk zullen wij ook nog meer boeken gaan maken en verkopen. Er zijn in ieder geval ideeën voor de volgende boeken:

-**EDUCATIEVE TOEPASSINGEN** - Een boek voor leerdoeleinden.
-**MACHINETAAL OP DE SHARP MZ-800** - De titel zegt het al.
-**UITBREIDINGEN VAN DIT BOEK** - o.a. meer over FLOPPY, RAM-DISK en PLOTTER.

U ziet dat er goede hoop is voor de toekomst. De SHARP begint een nieuw leven met NEPTUNES PRODUCTIONS. Wij willen via deze VOF proberen de SHARP weer op te laten bloeien, want er zijn altijd nog veertigduizend SHARP MZ-800 modellen in ons land verkocht en die kunnen in een redelijk kort tijdbestek toch niet allemaal op de schroothoop beland zijn.

SLOTWOORD.

Ter afsluiting van dit boek willen wij U graag nog het volgende zeggen: In dit boek zijn we heel erg openlijk geweest. Wisten wij iets niet of was iets te moeilijk voor ons om uit te leggen dan zeiden wij dat ook. Wij zijn tenslotte ook maar mensen en kunnen ook niet alles weten.
Een opzet die waarschijnlijk niet bij iedereen goed over zal zijn gekomen, omdat het toch een beetje argwaan kan wekken. Wij zijn zelf van mening dat het boek genoeg informatie bevat voor iedereen en dat de uitleg vaak ook meer dan voldoende is.
In een boek, waarin zovéél onderwerpen worden behandeld, kan niet bij ieder onderwerp een uitgebreide uitleg komen. Om deze reden hebben wij ook gezegd dat U ons altijd kunt bellen of schrijven wanneer U vragen heeft.
Sommige informatie uit dit boek zal in het jaar 1991 geheel verouderd zijn. Wij zijn er van uit gegaan dat het grootste deel van de verkoop in het jaar 1990 plaats zal vinden en hebben de informatie op dat jaar afgestemd.
Ondanks de vele verwijzingen naar het handboek en soms naar andere SHARP-boeken en het vrij losbandige taalgebruik, hopen wij toch dat U veel heeft kunnen of kunt leren van dit boek, want daar gaat het uiteindelijk om.
Af en toe staan er ook overbodige dingen of staat er nutteloze tekst om ieder hoofdstuk en deel van een hoofdstuk precies aan het einde van een bladzijde uit te laten komen. Zoals U heeft kunnen merken, bevat iedere bladzijde hetzelfde aantal gebruikte regels. Alleen bij de Inleiding, de Inhoudsopgave en het Slotwoord is dat niet het geval.
In dit boek wordt ook constant gesproken over "wij", terwijl het boek volledig door Arjan is geschreven. Een aantal artikelen uit dit boek zijn afkomstig van Mark en wij verkopen samen dit boek, vandaar dat er constant het woordje "wij" wordt gebruikt.
Dan rest ons U nogmaals te bedanken voor Uw belangstelling en begrip te hebben voor de gekozen opzet. Met vriendelijke groeten,

Arjan Habing
en
Mark de Rover

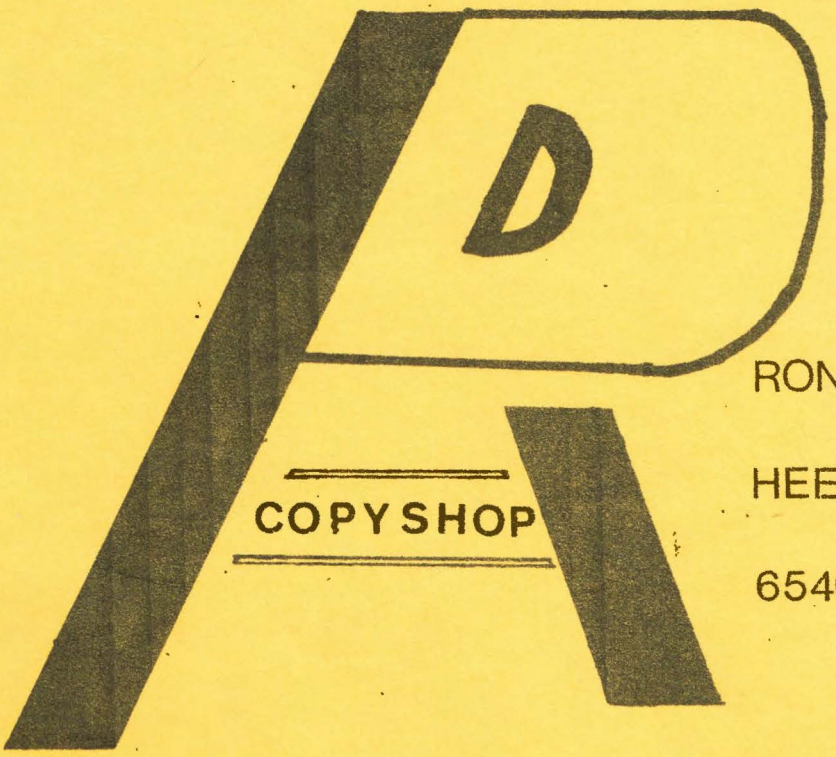
Geachte bezitter van het boek "De SHARP MZ-800 ten voeten uit",

Hierbij wil ik gaarne nog wijzen op enkele onjuistheden die in het boek voor kunnen komen door enkele foutjes die opgetreden zijn bij de drukker/binder.
Het kan zijn dat in het boek enkele bladzijden ontbreken of slecht leesbaar zijn. Die bladzijden zijn dan afzonderlijk bijgesloten.
Het kan ook zijn dat een aantal bladzijden omgekeerd zijn, zoals in veel gevallen bij de bladzijden 119/120 en 133/134 het geval is.
In enkele gevallen komt ook een dubbele perforatie voor.
Hierbij mijn excuses voor de ongemakken die hierdoor kunnen ontstaan, ook al ligt de schuld hiervan bij de drukker.
Voor eventuele nieuwe boeken voor de MZ-800 beloof ik verbetering op dit gebied, in dit geval is er echter niets aan te doen, omdat de eerste 200 exemplaren al helemaal gereed zijn en eerst verkocht moeten worden.
In enkele gevallen kunnen de onjuistheden in het boek dermate groot zijn dat U eventueel korting kunt krijgen op dat boek. Dit alleen in overleg met Neptunes Productions.
Neptunes Productions distansieerd zich van eventuele latere aanspraken die gemaakt worden aan de hand van de onjuistheden in boeken die gekocht zijn en dit schrijven bevatten.
Ondanks de ongemakken hoop ik voor U dat U veel kunt leren van dit boek en dat U er veel plezier van zult hebben.
Met vriendelijke groeten.

Hoogachtend,

A.J. Habing, Neptunes Productions.

blz 205-210 omgekeerd.



RON DIBBETS

HEESKESACKER 13-36

6546 JJ NIJMEGEN