master your zx microdrive

programs, machine code and networking

andrew pennell

Werken met de ZX Microdrive

Andrew Pennell

Werken met de ZX Microdrive

KLUWER TECHNISCHE BOEKEN B.V.
DEVENTER - ANTWERPEN

Vertaald door: G. J. van der Veen
Omslag: W. Niessink

ISBN 90 201 1856 0
D/1985 /0108/205

Oorspronkelijke titel: Master your ZX microdrive

1e druk 1985

Inhoud

Inleiding

De bedoeling van dit boek is te laten zien hoe men optimaal gebruik kan maken van de ZX-interface I, in het bijzonder verwijzend naar de ZX-Microdrive. De interface verruimt de mogelijkheden van de ZX Spectrum aanzienlijk, en de Microdrive biedt zeer snel toegankelijke geheugenopslag.
De standaardmogelijkheden worden duidelijk uiteengezet, maar zijn voorzien van veel verborgen aanvullende mogelijkheden die de interface nog veel nuttiger maken. Er zijn verschillende machinetaalroutines in het boek opgenomen, die het potentieel aanmerkelijk vergroten en voor een ieder zijn te begrijpen en te gebruiken, zelfs zonder enige kennis van machinecode. Voor die lezers, die wel over dergelijke kennis beschikken, is er een paragraaf opgenomen met interface ROM-routines en het gebruik daarvan, evenals assemblerlijsten van alle machinetaalroutines in dit boek.
De machinetaalprogramma's worden gepresenteerd in de vorm van DATA-regels en een FOR...NEXT-lus wordt gebruikt om de waarden op de gewenste adressen te POKEn. Waar mogelijk, is de machinecode positie-onafhankelijk. Dit betekent dat ze op iedere willekeurige plaats in het geheugen kan worden gepositioneerd. De meest geschikte plaats hiervoor is boven RAMTOP (zie pagina 179 in de Spectrum handleiding), die omlaag kan worden gebracht via CLEAR. Andere gebieden waar men de routines kan plaatsen, zijn het UDG gebied (168 bytes) en de printerbuffer (256 bytes). REM regels dienen om technische redenen te worden vermeden als opslag voor machinecode die de interface gebruikt. Voor het gemak is iedere routine voorzien van twee aanbevolen geheugenadressen (voor 16K en 48K machines) die het mogelijk maken om alle routines tegelijkertijd aanwezig te hebben. Worden zij gebruikt, dan dient eerst RAMTOP voldoende omlaag te worden gebracht – voor alle routines reserveert, CLEAR 32009 voor 16K machine, CLEAR 64779 voor 48K machine, voldoende ruimte. Daar sommige routines uit zeer veel nummers bestaan, is een controlesom opgenomen ter beveiliging tegen fouten. Is het totaal van gePOKEte waarden ongelijk aan de controlesom, dan stopt de routine met een passende foutboodschap. De gegevens dienen dan te worden gecontroleerd op verschillen en gecorrigeerd indien nodig.
Er zijn ook paragrafen opgenomen ten behoeve van programmabeveiliging en het creëren van aanvullende BASIC-commando's. De appendix bevat meer technische details en geeft informatie over bekende problemen (''bugs'') bij het gebruik van de interface.
Dit boek laat zien hoe men de Spectrum als serieuze machine kan gebruiken – tijdens het schrijven van dit boek is gebruik gemaakt van een 48K Spectrum voorzien van een groot toetsenbord en interface I, samen met een tekstverwerker en een EPSON RX 80 dot/matrix afdrukeenheid.

1. Streams en channels

Voordat we de vele extra mogelijkheden gaan bespreken die de ZX-interface 1 toevoegt aan de Spectrum, is het nodig de grondregels van streams en channels te begrijpen, omdat deze van essentieel belang zijn voor een optimaal gebruik van Microdrives, de RS232 en netwerken.
Een *channel* is een onderdeel van een computersysteem waar gegevens heen en vandaan kunnen worden gezonden, zoals het geval is bij het toetsenbord, dat gegevens invoert en het beeldscherm dat gegevens uitvoert. De wegen waarlangs de gegevens worden getransporteerd, worden *streams* genoemd.
Op de Spectrum zijn zestien streams beschikbaar in de BASIC, waarvan er vier initieel zijn toegekend aan specifieke channels, zoals weergegeven in onderstaande tabel:

stream nr.	*channel specificatie*	*uitvoer naar*	*invoer van*
#0	"K"	onderkant scherm	toetsenbord
#1	"K"	onderkant scherm	toetsenbord
#2	"S"	bovenkant scherm	n.v.t.
#3	"P"	ZX printer	n.v.t.

De streams 0 en 1 zijn identiek en toegekend aan het "K"-channel. Dit zendt tekens naar de onderkant van het beeldscherm (waar foutboodschappen en INPUTs verschijnen) en voert ze in van het toetsenbord. Stream 2 is het "S"-channel, dat tekens afdrukt op de bovenkant van het scherm en stream 3 is het "P"-channel, dat tekens uitvoert naar de ZX-printer. Het Spectrum-systeem verbiedt invoer via streams 2 en 3. Als bijvoorbeeld de BASIC-instructie INPUT #3; a$ wordt geprobeerd, produceert het systeem de foutboodschap `Invalid I/O device`.
Het symbool voor een stream is het hekje-teken (#, symbol shift 3), dat kan worden toegevoegd aan het PRINT, INPUT en enkele andere commando's. Om bijvoorbeeld iets af te drukken op de onderste twee regels van het beeldscherm, die normaal niet toegankelijk zijn, kunt u streams 0 of 1 gebruiken, zoals:

```
PRINT #0; AT 0,0;"Line 23"; AT 1,0;"Line 24";: PAUSE 0
```

Het pauzecommando dient ter voorkoming van het "OK" bericht dat de tekst zou overschrijven. Het afdrukken op channel "K" kan onverwachts de onderkant van het scherm in de bovenkant van het scherm doen overlopen. Dit kan worden voorkomen door het gebruik van het AT-commando en door tekst te beëindigen met een puntkomma.
De instructie PRINT #2; is gelijk aan de gebruikelijke PRINT-opdracht, die tekens verstuurt naar channel "S", dus de bovenkant scherm. PRINT #3; is identiek aan LPRINT, dat tekens afdrukt op de ZX-printer (channel "P"). Een voordeel van het gebruik van streams 2 en 3 in programma's is, dat op eenvoudige wijze een variabele kan worden gebruikt om te bepalen of uitvoer moet worden gestuurd naar het

scherm of de printer. Het volgende programma demonstreert het gebruik van deze techniek:

```
1000 INPUT "Printer (J/N)";a$
1010 LET c=2; IF a$="j" OR a$="J" THEN LET c=3
1020 PRINT #c; "Dit is een ZX-Spectrum"
```

Het is ook mogelijk om streams te mixen in dezelfde PRINT-opdracht;

```
2000 PRINT #2; "Het scherm"; #3;"De ZX-Printer"
```

De opdrachten INPUT en LIST kunnen ook worden gebruikt met streams. Voor een Spectrum zonder interface is alleen INPUT #0; en INPUT #1; mogelijk en zijn equivalent aan de gebruikelijke INPUT-opdracht. Echter, de INPUT #-opdracht is buitengewoon nuttig wanneer men beschikt over een interface, zoals zal blijken uit de latere hoofdstukken.
De LIST #n-opdracht, waarin n een streamnummer voorstelt, kan ook worden gebruikt en stuurt een listing van een BASIC-programma naar de geselecteerde stream. LIST #3 is equivalent n aan het LLIST-commando, dat een listing produceert op de ZX-printer en LIST #0 produceert een interessant, doch volkomen onbruikbaar effect op het scherm.
De INKEY$ #n-opdracht kan ook worden gebruikt met streams, maar heeft geen enkel nut indien gebruikt met de standaard channels. INKEY$ #0 en INKEY$ #1 produceren gewoonlijk nullen als resultaat en #2 en #3 zijn niet toegestaan. In latere hoofdstukken zal blijken hoe dit kan worden gebruikt in combinatie met extra interface channels.

1.1 OPEN # en CLOSE

Het OPEN #-commando heeft beperkte mogelijkheden op een standaard Spectrum. De bedoeling ervan is, een channel aan een stream toe te kennen en het algemene formaat is

```
OPEN #n,f$
```

waarin n het streamnummer en f$ het channel voorstelt. Merk op dat het #-teken onderdeel is van het OPEN # sleutelwoord, dat wordt verkregen via extended mode symbol shift 4. De waarde van n moet liggen tussen 0 en 15 (zoniet dan volgt een foutboodschap `Invalid stream`) en de channel-aanduiding moet een enkele letter zijn en een bestaand channel voorstellen (zowel met de groteletter- als de kleinelettertoets), bijv. "K", "k", "S", "s", "P" of "p". Indien onjuist, dan volgt een foutboodschap `Invalid filename`. Met de interface 1 aangesloten, worden de channel-aanduidingen "M", "N", "B" en "T" en de bijbehorende kleine letters ook geaccepteerd en stellen achtereenvolgens de Microdrive, het netwerk, binair RS232 en tekst RS232 channels voor. Deze additionele channel-aanduidingen staan ook het gebruik toe van een punt-komma als scheidingssymbool in de OPEN #-opdracht, evenals een komma en sommige hebben aanvullende gegevens nodig. Meer details worden gegeven in de relevante hoofdstukken.
Het voorbeeld

```
OPEN #3;"s"
```

zorgt ervoor dat alle stream 3 uitvoer die normaal naar de ZX printer gaat, nu naar channel "s", de bovenkant scherm gaat, zodat LLIST een listing produceert op het scherm. Dit kan handig zijn bij het oplossen van fouten in programma's die gebruik maken van de ZX-printer, zonder dat deze is aangesloten, of om papier te besparen. Het tegenovergestelde hiervan is

```
OPEN #2,"p"
```

dat ervoor zorgt dat alle uitvoer naar de printer wordt gestuurd in plaats van naar het scherm. Wanneer deze commando's gelijktijdig worden uitgevoerd kan programma-uitvoer nogal verwarrend zijn, zachtjes uitgedrukt!
U kunt ook OPEN # gebruiken om aanvullende streams te activeren. Gewoonlijk hebben de streams 4 tot 15 geen channels en iedere poging om ze te gebruiken (bijv. PRINT #6;"6";) resulteert in het `Invalid stream` bericht. Echter, deze extra streams kunnen worden geOPENd op de normale wijze - bijvoorbeeld:

```
OPEN #4,"s"
PRINT #4;"Hallo daar!"
```

produceert de woorden "Hallo daar!" op het scherm, channel "S". De streams 4 tot 15 worden uitvoerig gebruikt door de extra Sinclair randapparatuur die de interface gebruiken.
Om een OPEN #n te de-activeren, wordt zijn complement, het CLOSE #n-commando gebruikt, waarin n naar het streamnummer verwijst. Een CLOSE van streams 0 tot 3 heeft tot gevolg dat de standaard channels worden toegekend, terwijl een CLOSE van de streams 4 tot 15 ieder channel verwijdert.
BELANGRIJK: Wees voorzichtig met het CLOSE #-commando samen met streams 4 tot 15 indien geen interface is aangesloten – tengevolge van een fout in de BASIC zullen bij reeds gesloten streams vreemde dingen gebeuren. Gewoonlijk wordt een vreemde boodschap gegenereerd, maar soms moet de computer even uit worden geschakeld! Het probleem doet zich niet voor indien de interface is aangesloten, omdat deze de fout herstelt.

1.2 CLEAR # en MOVE

Er is nóg een opdracht die kan worden gebruikt met channels – namelijk `CLEAR #` (let op het hekje-teken "#"). Deze voegt niet veel belangwekkends toe aan de standaard CLEAR-opdracht – zijn doel is om een CLOSE uit te voeren voor streams 4 tot 15 en om streams 0 tot 3 te voorzien van de standaard channels. Voorzichtigheid is geboden in het gebruik ervan met de extra interface channel typen. Het laatste commando dat gebruik maakt van channels is MOVE. Dit kent verschillende syntax-formaten, maar het standaardformaat is

```
MOVE #a TO #b
```

waarin **a** en **b** de streamnummers voorstellen (**TO** is een sleutelwoord). Wat het doet, is inlezen van een teken van stream a en afdrukken ervan op stream b. Dit gaat zo door tot aan een bepaalde conditie wordt voldaan – deze conditie hangt af van het type channel waarmee stream a is verbonden. Het is een krachtig commando en de verschillende toepassingen worden uiteengezet in de volgende hoofdstukken. Het is mogelijk om met machinetaalroutines speciale streams te creëren voor speci-

fieke toepassingen. Bijvoorbeeld, het BASIC-besturingssysteem gebruikt intern channel "R" op stream – 1 (welke niet beschikbaar is voor BASIC-programma's) om tekens af te drukken in het interne werkgeheugen. Het is ook mogelijk om bestaande channels te wijzigen – alle onafhankelijk geproduceerde Centronics-type printer interfaces wijzigen channel "P", zodat alle stream 3-uitvoer (bijv. LPRINT) tekens zendt naar een externe printer via zijn eigen interface, in plaats van de ZX-printer.
Wanneer verscheidene streams tegelijkertijd worden aangewend, is het vaak lastig bij te houden welk channel is toegekend aan welke stream en welke streams niet worden gebruikt. Het volgende programma produceert een tabel die alle relevante gegevens toont van iedere stream, met inbegrip van de streams #3 tot #1, welke beschikbaar zijn voor machinetaalprogramma's. Als het programma wordt gedraaid, drukt het een beschrijving af van het gebruik van iedere stream en of ze gebruikt kunnen worden voor invoer of uitvoer of beide. Als de beschrijving klaar is, wordt een streamnummer gevraagd. Als u deze invoert, wordt de desbetreffende stream meer in detail beschreven. Het programma zoals hier afgedrukt, verschaft alleen informatie over de standaard channels. Meer mogelijkheden zullen worden toegevoegd in de relevante hoofdstukken van dit boek.

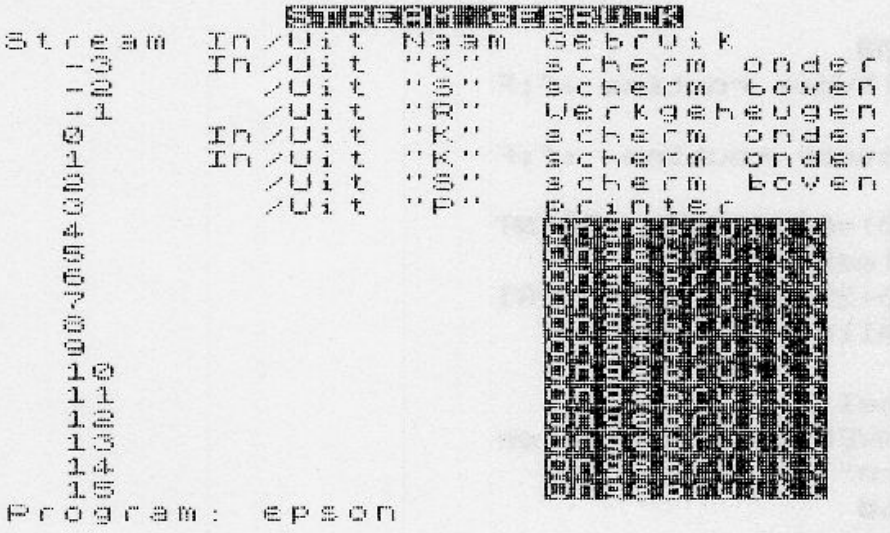
Streamoverzichtprogramma

```
1000 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
1005 CLS : PRINT TAB 10; INVERSE
 1;"STREAM GEBRUIK"
1010 PRINT "Stream";TAB 7;"In/Ui
t";TAB 14;"Naam";TAB 19;"Gebruik
"
1020 FOR s=-3 TO 15
1025 PRINT TAB 2;s;
1030 LET d=FN p(s*2+23566+8)
1040 IF d=0 THEN  PRINT TAB 19;
INVERSE 1;"Ongebruikt": GO TO 12
00
1050 LET d=d+FN p(23631)-1
1060 IF FN p(d+2)<>5572  THEN  P
RINT TAB 7;"In";
1070 PRINT TAB 9;"/";
1080 IF FN p(d)<>5572 THEN  PRIN
T TAB 10;"Uit";
1090 LET f$=CHR$ PEEK (d+4)
1100 PRINT TAB 14;"""";f$;"""";T
AB 19;
1110 IF f$="K" THEN  PRINT "sche
rm onder": GO TO 1200
1120 IF f$="S" THEN  PRINT "sche
rm boven": GO TO 1200
1130 IF f$="P" THEN  PRINT "prin
ter": GO TO 1200
1140 IF f$="M" THEN  PRINT "Micr
odrive": GO TO 1200
1150 IF f$="N" THEN  PRINT "Netw
erk": GO TO 1200
```

```
1160 IF f$="T" THEN  PRINT "RS23
2": GO TO 1200
1170 IF f$="R" THEN  PRINT "Werk
geheugen": GO TO 1200
1190 PRINT  FLASH 1;"Ongebruikt"
1200 NEXT s
1210 PRINT AT 0,0;
1500 INPUT "Geef streamnummer of
 stop met ENTER"; LINE S$
1505 IF S$="" THEN  STOP
1510 CLS
1515 LET S=VAL S$
1520 PRINT TAB 10; INVERSE 1;"ST
REAMNUMMER ";S''
1530 LET d=FN P(S*2+23566+8)
1540 IF D=0 THEN  PRINT "CLOSEd
stream": GO TO 1500
1550 LET D=D+FN P(23631)-1
1560 LET F$=CHR$ PEEK (D+4)
1570 PRINT "Channel aanduiding:"
;F$
1580 REM controleer ieder type
1600 IF F$="K" THEN  GO TO 2000
1610 IF F$="S" THEN  GO TO 2020
1620 IF F$="P" THEN  GO TO 2040
1630 IF F$="R" THEN  GO TO 2050
1670 PRINT  FLASH 1;"Onbekende a
anduiding"
1700 GO TO 1500
1800 PRINT "Uitvoer routine :";F
N P(D)
1810 PRINT "Invoer routine  :";F
N P(D+2)
1820 IF FN P(D)=5572 THEN  PRINT
  FLASH 1;"Alleen Invoer"
1830 IF FN P(D+2)=5572 THEN  PRI
NT  FLASH 1;"Alleen Uitvoer"
1870 RETURN
1999 REM channel K
2000 PRINT  INVERSE 1;"Scherm on
der/toetsenbord"
2010 GO TO 2060
2019 REM channel S
2020 PRINT  INVERSE 1;"Scherm bo
ven"
2030 GO TO 2060
2039 REM channel P
2040 PRINT  INVERSE 1;"ZX Printe
r"
2045 GO TO 2060
2049 REM channel R
2050 PRINT  INVERSE 1;"Werkgeheu
gen"
2060 GO SUB 1800
2070 GO TO 1500
```

Het programma leest verschillende geheugenvelden met behulp van het PEEK-commando – lees alleen verder indien dit u interesseert. De **s** FOR...NEXT-lus (regel 1020) wordt doorlopen voor iedere stream. Daarbij wordt de variabele **d** steeds aangepast opdat deze verwijst binnen het STRMS geheugengebied (1030) naar het bijbehorende streamveld, dat nul aangeeft indien de stream is gesloten (1040). De waarde van d wordt toegevoegd aan de systeemvariabele CHANS (1050), met d verwijzend naar enkele bytes in het CHANS-geheugengebied. De gecreëerde channels (dus K,S,P en R) gebruiken ieder een minimum van vijf bytes binnen het CHANS-gebied (totaal dus 20 bytes). In ieder gedeelte van vijf bytes verwijzen de eerste twee bytes naar de "tekenuitvoer"-routine, de volgende twee bytes naar de "tekeninvoer"-routine en de vijfde byte stelt de channel-aanduiding voor in hoofdletterformaat. Positie 5572 is de `Invalid I/O device` foutaanroep (probeer RANDOMIZE USR 5572 ter controle) en deze wordt gecontroleerd in regel 1060 en 1080. De channel-aanduiding wordt afgedrukt in regel 1100, vervolgens wordt een controle uitgevoerd op bestaande channels in de regels 1110-1180 en daarna kan het van toepassing zijnde in/uitvoer medium worden afgedrukt. Na het overzicht kan de gebruiker een streamnummer inbrengen (1500) om meer gegevens te verkrijgen. Voor de huidige channels worden alleen de invoer- en uitvoerlocaties getoond, door gebruik te maken van een subroutine vanaf regel 1800, maar dit wordt later uitgebreid.
Onderstaand programma toont een typisch voorbeeld van het resultaat.

Uitvoer van streamoverzichtprogramma



1.3 Stream 14-z$-routine

Het volgende machinetaalprogramma creëert een stream 14, zodanig dat dit tekens "afdrukt" binnenin een BASIC-stringvariabele. Indien de routine wordt aangeroepen via GOSUB 8000 moet de variabele st het adres bevatten waar de routine naar toe moet. De lengte is 101 bytes en de aanbevolen locaties zijn 65260 (48 K) en 32490 (16 K). Nadat de routine is aangeroepen wordt alle stream 14-uitvoer toegevoegd aan het eind van de stringvariabele z$ (welke geen DIMensie mag hebben). Als z$ niet bestaat, of een DIMensie heeft, resulteert dit in de foutmelding `Variable not found`. Het is mogelijk om PRINT #14; en LIST #14 uit te voeren en, wellicht

zeer bruikbaar, wordt het gebruikt in het volgende hoofdstuk om een CATalogue van een Microdrive te genereren binnenin de string. Er is echter een beperking – opdrachten die rechtstreeks strings afdrukken zullen niet correct werken – bijvoorbeeld:

```
PRINT #14; "string"
```

zal foutievelijk "sssss" toevoegen aan het eind van string z$. Dit kan verholpen worden door of letter voor letter af te drukken (`PRINT #14; "s"; "t"; "r"; "i"; "h"; "g"`), of door gebruik te maken van een andere stringvariabele (`LET a$="string":PRINT #14;a$`). Voor gevorderde programmeurs – de reden ervan is dat de routinegedeelten van het geheugengebied verschuift om de letters in te voegen en ook het werkgebied mee verschuift, waarin de tijdelijke string is opgeborgen, waardoor steeds de eerste letter wordt toegevoegd.

Stream z$-14 programmalijst

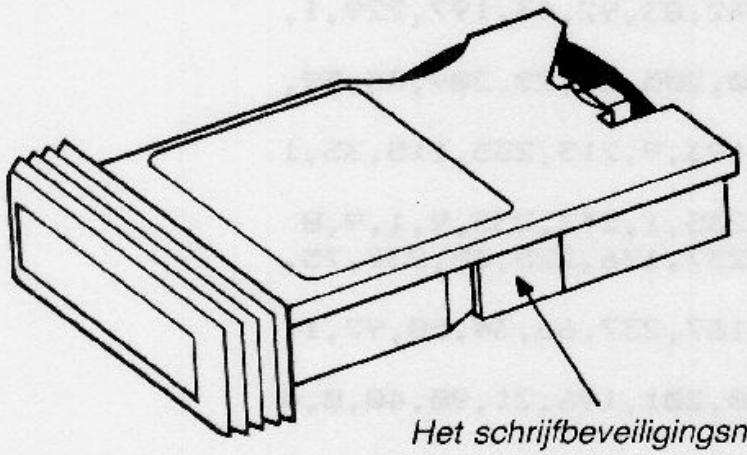
```
7995 REM ************************
7996 REM * stream14-z$ routine *
7997 REM ************************
7998 REM z$ moet bestaan,st= sta
rt adres
7999 REM  aanbevolen: 65260 / 32
490
8000  RESTORE 8070
8005 LET c=0
8010 FOR i=st TO st+100
8020 READ a: POKE i,a: LET c=c+a
8030 NEXT i
8035 IF c<>10557 THEN  PRINT "Co
ntrolesom fout": STOP
8040 CLOSE #14
8050 RANDOMIZE USR st
8060 RETURN
8070 DATA 42,83,92,43,197,229,1,
11
8075 DATA 0,205,85,22,209,33,59,
0
8080 DATA 193,9,213,235,115,35,1
14,35
8085 DATA 235,1,247,255,9,1,9,0
8090 DATA 237,176,225,35,237,75,
79,92
8095 DATA 167,237,66,34,50,92,1,
0
8100 DATA 0,201,196,21,90,40,0,4
0
8105 DATA 0,11,0,245,42,75,92,12
6
8110 DATA 254,90,40,11,254,128,2
02,112
8115 DATA 6,205,184,25,235,24,24
0,35
8120 DATA 78,35,70,3,197,229,9,2
05
8125 DATA 82,22,35,235,225,193,1
12,43
8130 DATA 113,241,18,167,201
```

2. De werking van Microdrives

Een Microdrive cartridge bevat een doorlopende, zeer dunne magnetische tape van bijna vijf meter lang, waarvan de kwaliteit aanzienlijk hoger is dan die, welke gebruikt wordt voor audio-cassettes. De Microdrive zelf bestaat voornamelijk uit een motor, die de tape rondtrekt en een opneem/afspeelkop zoals gebruikt in een cassette-recorder. De tape wordt met zeer hoge snelheid langs de kop getrokken en kan gegevens opslaan via het opnemen of gelezen worden via het afspelen. Het is echt een zeer snel cassettesysteem, zonder de problemen van terugspoelen enz.
Het doorlopende tapesysteem heeft ook nadelen – daarvan is de belangrijkste de toegangstijd. Is het gedeelte van de tape waar uw programma op staat net voorbij de kop getrokken en wilt u dit programma laden, dan moet de volle lengte van de tape worden gelezen, omdat er niet kan worden teruggespoeld. Dit betekent dat het soms zeven seconden kan duren, alvorens een programma of een groep gegevens is gevonden op de tape.
Het wegschrijven van gegevens naar tape kan ook traag zijn. Is de cartridge nog nieuw en staan er weinig programma's op, dan zal het voor de computer eenvoudig zijn om een vrij gebied te vinden om gegevens op te slaan. Echter, indien de cartridge grotendeels is volgeschreven, dan zal de computer de tape moeten afzoeken en op verschillende plaatsen op tape kleine beetjes gegevens opslaan, hetgeen ook enige tijd kan duren. Dit veroorzaakt ook dat het laden van de gegevens trager wordt.

2.1 Beveiliging tegen abusievelijk wissen

Audiocassettes hebben twee nokjes aan de rugzijde, die, indien eruit gewipt, voorkomen dat u iets nieuws opneemt op een cassette die al iets anders bevat. Eenzelfde systeem wordt toegepast op de Sinclair Microdrive cartridges.



Het schrijfbeveiligingsnokje

Zoals de afbeelding laat zien, heeft elke cartridge een klein plastic nokje aan een zijde. Indien u deze eruit wipt, bijvoorbeeld met een schroevedraaier of gelijkend voorwerp, voorkomt dit dat u, of iemand anders, gegevens wegschrijft op de cartridge. Dit is zeer belangrijk voor commerciële programma's en kan zeer handig zijn voor uw eigen programma's die u niet wilt wissen. Wilt u later, nadat u het nokje heeft

verwijderd, toch weer gebruik maken van de cartridge, dan kunt u dit bereiken door kleefband te plakken over de uitsparing. De kleefband moet smal genoeg zijn, opdat de onder- of bovenzijde van de cartridge niet bedekt worden, waardoor de cartridge niet meer in de microdrive zou passen.

2.2 Cartridges

De tape binnen in de cartridge is zeer kwetsbaar. U mag deze nimmer aanraken met uw vingers of iets anders en u moet de cartridge daarom direct na gebruik terugsteken in het doosje ter bescherming.
Bij het insteken van de cartridge in de microdrive, dient u erop te letten, dat de juiste zijde boven komt (het grote etiket boven) en dat hij stevig wordt aangedrukt. Eenmaal ingestoken, moet u zich ervan overtuigen dat de cartridge zover als mogelijk is ingedrukt, omdat anders fouten kunnen ontstaan gedurende het gebruik ervan. De volgende twee belangrijke regels dienen te worden aangehouden, wanneer cartridges worden gebruikt:

1. *Verwijder een cartridge NOOIT wanneer de rode LED brandt.*
2. *Laat nooit een cartridge in de microdrive zitten, terwijl de stroom aan of uit wordt geschakeld.*

Wordt een van deze regels niet nagevolgd, dan kan blijvende schade worden veroorzaakt aan de cartridge of aan de microdrive zelf.
Cartridges gaan niet zo lang mee als audiocassettes, vanwege het intensieve gebruik ervan. Bewaar daarom altijd kopieën van belangrijke programma's of gegevens op een andere cartridge, of, nog altijd beter, op een cassette.

2.3 Het ingebruikstellen van een cartridge door middel van FORMAT

Indien u een nieuwe cartridge heeft gekocht, zal de tape willekeurige magnetische informatie bevatten. Iedere poging om gebruik te maken van de cartridge resulteert in de foutboodschap `Microdrive not present`. Om het mogelijk te maken gebruik te maken van de cartridge, moet deze geïnitialiseerd worden door middel van het FORMAT-commando. Voor microdrives heeft dit commando de volgende algemene vorm:

```
FORMAT "m";d;"naam"
```

waarin d het drivenummer voorstelt (van 1 tot 8) en "naam" de titel is die permanent wordt toegekend aan de cartridge. De lengte van deze naam mag hooguit tien letters zijn. Het commando heeft tot gevolg dat de computer 30 seconden actief is met het indelen van de cartridge, het toekennen van de titel en het diverse keren controleren van de tape. Blijkt een gebied op de tape onbruikbaar te zijn, bijvoorbeeld omdat dit is beschadigd, dan zal deze ruimte worden gemarkeerd om toekomstig gebruik ervan te voorkomen. Gedurende het proces flitst de rand van het beeldscherm. Voor het uitvoeren van FORMAT, dient u zich ervan te overtuigen dat de cartridge correct in de Microdrive is geduwd. De eerste cartridge waarmee ik een FORMAT uitvoerde was niet voldoende in de Microdrive geduwd, waardoor de computer dacht dat

slechts eenderde bruikbaar was! Door het commando te herhalen, na de cartridge zover mogelijk te hebben ingeduwd, werd de cartridge correct behandeld.
Met splinternieuwe cartridges doet u er goed aan twee kort op elkaar volgende FORMAT-commando's uit te laten voeren.
Het FORMAT-commando zal elk programma aanwezig op de cartridge wissen, dus controleer de cartridges altijd zorgvuldig, alvorens het commando uit te geven. Indien het schrijfbeveiligingsnokje is verwijderd, zal een FORMAT-commando resulteren in een foutboodschap.
Na een FORMAT kunt u het proces volgen door middel van **CAT d** (waarin **d** het drivenummer voorstelt), dat na ongeveer zeven seconden de titel afdrukt, gevolgd door een getal. Dit getal is een aantal dat aangeeft hoeveel kilobytes (K) van de cartridge gebruikt kunnen worden. Na een FORMAT, moet dit altijd minstens 85 K zijn. Het hoogste dat ik ooit heb bereikt met een cartridge is 92 K.

2.4 Programma's en de Microdrives

Een van de belangrijkste voordelen van de microdrive is de hoge snelheid waarmee geschreven en geladen wordt. Om hier gebruik van te maken zijn dezelfde commando's voorhanden als die voor de cassetterecorder. Wilt u bijvoorbeeld het streamoverzichtprogramma op een cassettebandje bewaren, dan kunt u gebruik maken van

```
SAVE "streams"
```

Om deze commando's te gebruiken met de Microdrives moeten verschillende dingen worden toegevoegd in de regel, achter het commando. Ten eerste moet er een sterretje (*) worden toegevoegd, daarna **"m";** om aan te geven dat het een Microdrive-opdracht betreft en vervolgens **d;** om de desbetreffende drive te adresseren. Om bijvoorbeeld het streamoverzichtprogramma te bewaren op een cartridge in drive nummer 2, kunt u opgeven

```
SAVE *"m";2;"streams"
```

(zowel "m" als "M" worden geaccepteerd – er wordt geen onderscheid tussen gemaakt.) De aanvullende commando's zullen niet worden geaccepteerd door de syntaxcontrole van een Spectrum zonder interface.
Zodra een Microdrive-opdracht wordt uitgevoerd, zal de motor gaan draaien en de rode LED oplichten. Wordt een drive aangeroepen zonder dat er een cartridge is aangebracht, dan volgt een hoge pieptoon en op het scherm verschijnt de foutboodschap `Microdrive not present`. Of een drive/cartridge combinatie correct functioneert, blijkt meestal uit het geluid dat wordt gemaakt. Onder normale omstandigheden komt er een zacht gezoem vanuit de drive. Is er echter iets mis, zoals bijvoorbeeld een niet volledig ingestoken cartridge, dan volgt meestal een ratelend geluid. Zodra dit gebeurt, dient u de cartridge alsnog stevig aan te drukken. Als het geratel aanhoudt druk dan op BREAK (door CAPS SPACE in te drukken), onderzoek de cartridge en vervang deze om het opnieuw te proberen. Indien de drive met cartridge een signaal produceert en een foutboodschap toont, betekent dit meestal dat de cartridge fout is, waardoor de tape niet langer kan draaien. Hij moet dan worden vervangen. Als alles in orde is, dan duurt een SAVE hooguit zeven seconden; de cartridge wordt gelezen, op zoek naar de opgegeven programmanaam. Als deze

wordt gevonden verschijnt de foutboodschap `Writing to a "read" file.` De rand van het beeldscherm flitst op ieder moment waarop gegevens daadwerkelijk naar de cartridge worden geschreven.
Het is mogelijk, evenals bij cassettes, gebruik te maken van de LINE-functie binnenin een SAVE-opdracht, waardoor een BASIC-programma automatisch start zodra is is geladen. Om bijvoorbeeld het eerder bedoelde programma automatisch te laten starten vanaf regel 1000, kunt u gebruik maken van

```
SAVE *"m";2;"stream"LINE 1000
```

Na een SAVE, kunt u controleren of de opdracht correct is uitgevoerd door de apparatuur, door middel van de verify-functie

```
VERIFY *"m";2;"stream"
```

Indien een verschil optreedt, verschijnt de foutboodschap `Verification failed` op het scherm. Microdrive-programma's dienen altijd een perfect VERIFY op te leveren. Doen ze dat niet, controleer dan zowel de cartridge als de Microdrive grondig. Indien de fout blijft, vervang dan een of beide.
Om een programma te laden van een cartridge, wordt het LOAD-commando gebruikt met de volgende syntax

```
LOAD *"m";d;"naam"
```

waarin d het drivenummer voorstelt en ''naam'' de benodigde programmanaam aangeeft. Anders dan bij cassettes, is het niet mogelijk om het eerstvolgende programma te laden door middel van een lege string als programmanaam. Bijvoorbeeld, de poging om

```
LOAD *"m";1;""
```

uit te voeren, resulteert in de foutboodschap `Invalid filename.` Kan een programmanaam niet worden gevonden op een cartridge, dan volgt het bericht `File not found.`
De samenvoegfunctie MERGE werkt ook op de microdrives en heeft eenzelfde syntax als de andere commando's namelijk:

```
MERGE *"m";d;"naam"
```

Dit zal programma ''naam'' laden van drive **d** en deze toevoegen aan het reeds geladen programma evenals de variabelen, waarbij de reeds aanwezige variabelen worden vervangen door die op de cartridge.
Anders dan bij het cassettecommando, is het niet mogelijk om een MERGE uit te voeren bij auto-startprogramma's, die bewaard zijn met LINE X. Iedere poging hiertoe leidt tot het bericht `Merge error`
Een aanvullend commando, niet nodig bij cassettebestanden, is ERASE. Dit commando heeft tot doel bepaalde bestanden op de cartridge te wissen. De syntax is

```
ERASE "m";d;"naam"
```

ERASE heeft geen sterretje nodig, in tegenstelling tot de andere microdrive-commando's. Het is niet toegestaan om de bestandsnaam te laten volgen door andere parameters zoals LINE 2000.
Indien u een BREAK uitvoert gedurende een SAVE-actie, zal het bestand incom-

pleet, en daardoor niet te laden zijn. Gebruik in dat geval het ERASE-commando. ERASE ieder bestand dat tengevolge van onderbrekingen gedurende een SAVE slechts voor een gedeelte is weggeschreven. Probeert u een programma weg te schrijven dat reeds aanwezig is op de cartridge, dan verschijnt het bericht `Writing to a "read" file`. Het oude programma moet gewist worden alvorens de nieuwe versie weg te schrijven.

2.5 Koppelen van programma's

Door gebruik te maken van de SAVE *''m'' ...LINE-mogelijkheid, kan een programma automatisch worden gestart en daardoor een ander programma, misschien een machinetaalprogramma, worden geladen.
Een probleem is dat gekoppelde programma's alleen mogelijk zijn met een bepaalde drive – om dit te omzeilen, moet het eerste programma de volgende regel bevatten

```
LET d=PEEK 23766
```

waarin d het drivenummer voorstelt waar het programma van wordt geladen. Heet het tweede programma ''prog2'' en wilt u dit laden via het eerste programma, dan kunt u gebruik maken van

```
100 LOAD *"m";d;"prog2"
```

De twee programma's zullen dan draaien onafhankelijk van de drive waarin ze zich bevinden. Iets als

```
100 LOAD *"m";PEEK 23766;"prog2"
```

zal *NIET* werken om technische redenen. Gebruik altijd een variabele.

2.6 Arrays en machinetaal op een cartridge

Evenals het opslaan van programma's op een cartridge, is het mogelijk om arrays, machinetaalroutines en schermen op te slaan door het toevoegen van bepaalde parameters achter de programmanaam in de relevante commando's. Om arrays op te slaan, voegt u DATA toe achter de programmanaam, vervolgens de arraynaam, numeriek of als string. Bijvoorbeeld, om de array b ( ) op drive 2 op te slaan onder de naam van ''lonen'', gebruikt u

```
SAVE *"m";2;"lonen"DATA b()
```

Met cassettes was het mogelijk om eenvoudige stringvariabelen zonder dimensie naar tape weg te schrijven, maar indien ze terug werden geladen werden ze overschreven. Dit is voorkomen op de Microdrive – om dit aan te tonen, tikt u in

```
LET a$="test":SAVE *"m";1;"string"DATA a$()
```

hetgeen leidt tot het bericht `"Nonsense in BASIC"`.
Machinetaal en bytes kunnen worden opgeslagen door het toevoegen van de parameter CODE gevolgd door twee getallen gescheiden door een komma. Het eerste getal is het startadres en het tweede getal geeft het aantal bytes aan. Beide getallen zijn optioneel in LOAD- en VERIFY-opdrachten. Plaatjes van het scherm kunnen

ook worden opgeslagen, door gebruik te maken van SCREEN$. Anders dan bij cassettes is het mogelijk een VERIFY uit te voeren voor deze plaatjes.
Wordt geprobeerd een bestand te laden met het verkeerde commando, dan verschijnt het bericht `Wrong filetype` zoals bijvoorbeeld bij

```
LOAD *"m";1;"test"CODE
```

als "test" een programma is.

2.7 CATalogue

Om te zien welke bestanden zich op een cartridge bevinden, kunt u het CAT-commando gebruiken. Het formaat is

```
CAT d
```

waarin **d** het drivenummer voorstelt. Dit commando genereert op het beeldscherm de titel van de cartridge, een blanco regel, vervolgens een alfabetische lijst van alle bestanden aanwezig op de cartridge en ten slotte een getal. Het getal geeft de hoeveelheid ongebruikte bytes aan in kilobytes (K) en moet minstens 85 K aangeven voor nieuwe lege cartridges. Een CAT-overzicht kan ook naar andere channels worden gezonden dan het beeldscherm met behulp van

```
CAT #n,d
```

waarin n het streamnummer voorstelt. Bijvoorbeeld:

```
CAT #3,2
```

genereert een overzicht van drive 2 via stream 3, gewoonlijk de ZX-Printer. Alleen de eerste 50 bestanden worden getoond. Heeft u dus meer dan 50 bestanden op de cartridge staan, dan toont CAT dus niet alle bestanden.
Wat gebeurt er wanneer u in een programma wilt weten welke bestanden op de cartridge staan. Een grove en langzame manier is, om met het CAT-commando een overzicht te tonen op het scherm, en vervolgens iedere letter te lezen via de SCREEN$-functie. Deze methode is nogal omslachtig, en bovendien zinloos indien meer dan 20 bestanden op de cartridge staan, omdat het scherm door zal scrollen en daardoor de eerste namen zal verliezen. De beste methode is het gebruikmaken van de stream14-z$-routine in het vorige hoofdstuk, die stream 14 zodanig opzet dat hij tekens toevoegt aan de variabele z$. Het CAT-overzicht wordt als volgt aan z$ toegevoegd

```
LET z$="":CAT #14,d
```

zodat u met de variabele z$ kunt beschikken over het overzicht. Om deze informatie in z$ te kunnen gebruiken, moet het formaat bekend zijn; dit is als volgt:

Ten eerste vindt u 10 tekens die de titel van de cartridge voorstellen, gevolgd door twee newline codes (CHR$ 13). Daarachter staat iedere bestandsnaam alfabetisch opgeslagen, elk bestaande uit 10 tekens en daarna volgt een newline. Ten slotte volgt nog een newline, met een of twee tekens waarmee het restant vrije kilobytes wordt aangegeven en uiteindelijk staat er nog een newline. Dit formaat wordt duidelijker door de volgende routine, die een keurig overzicht produceert op het scherm.

Het stream 14-z$-machinetaalprogramma moet eerst worden geladen en moet voor de GOSUB 2000-opdracht komen te staan.

Mooie CATalogue programmalijst

```
1999 REM Handige catalogue
2000 LET z$="": CAT #14,d
2010 CLS
2020 PRINT "Cartridge titel:"; I
NVERSE 1;z$( TO 10)
2025 PRINT
2030 LET z$=z$(13 TO )
2040 LET f=0
2050 IF LEN z$<10 THEN  GO TO 21
00
2060 LET f=f+1
2070 PRINT f;". ";z$( TO 10),
2080 LET z$=z$(12 TO )
2090 GO TO 2050
2100 PRINT ''f;" Vrije bytes ";z
$(2 TO LEN z$-1);"K"
2110 RETURN
```

2.8 Opslaan van variabelen

Indien u een lang programma heeft, dat nogal wat arrays en kritieke variabelen bevat, kan het veel tijd kosten om zowel het programma als de variabelen weg te schrijven (met de standaard SAVE-opdracht), of door iedere array met een andere naam weg te schrijven (d.m.v. SAVE ... DATA). De volgende subroutines maken het mogelijk alle variabelen van een programma naar een cartridge weg te schrijven, om ze misschien later weer terug te laden in een ander programma. De variabelen worden opgeslagen op drive 1 onder de naam "var1" en "var2", maar dit kan naar eigen inzicht worden veranderd door het aanpassen van de regels 4040, 4050, 4110 en 4130. Het is niet mogelijk om de bestandsnamen en drivenummers in variabelen op te slaan vanwege de toegepaste techniek om ze te bewaren. Dat de regels 4010 en 4015 gelijk zijn is geen drukfout, maar pure noodzaak. De SAVE-routine kan worden aangeroepen door middel van GOSUB 4000, maar de LOAD-routine wordt gestart met GO TO 4100 – GOSUB kan niet worden gebruikt omdat deze een CLEAR-opdracht bevat, welke alle vorige GOSUB's wegveegt; hierdoor moet regel 4140 worden veranderd om terug te springen naar de juiste regel in het aanroepende programma.

SAVE/LOAD variabelen programmalijst

```
3999 REM bewaar de variabelen
4000 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
4010 LET l=FN p(23641)-FN p(2362
7)
4015 LET l=FN p(23641)-FN p(2362
7)
4020 POKE 23565,l-256*INT (l/256
)
4030 POKE 23566,INT (l/256)
4040 SAVE *"m";1;"var1"CODE 2356
5,2
4050 SAVE *"m";1;"var2"CODE FN p
(23627),l
4060 RETURN
4099 REM laden van de variabelen
4100 CLEAR
4110 LOAD *"m";1;"var1"CODE 2356
5
4120 DIM a$(FN p(23565)-7)
4130 LOAD *"m";1;"var2"CODE FN p
(23627)
4140 GO TO xxxx: REM rest v h pr
ogramma
```

2.9 Meervoudige kopieën

Zoals eerder werd vermeld is het raadzaam om altijd kopieën te bewaren van belangrijke programma's en gegevens op andere cartridges of cassettes. Echter, er zit in de Spectrum een ongedocumenteerde mogelijkheid ingebouwd, om een bestand een willekeurig aantal malen op dezelfde cartridge op te slaan. Indien u dit via de BASIC wilt bereiken door tweemaal een SAVE uit te voeren, volgt een foutboodschap. Om meervoudige kopieën van een programma of ander bestandstype weg te schrijven, voert u eerst een POKE 23791,x uit, waarin x het aantal kopieën voorstelt dat kan variëren van 1 tot 255, maar een waarde 2 is meestal voldoende. Daarna pas wordt de SAVE gedaan, zodat het bestand x maal wordt opgeslagen. Het aantal kopieën wordt na de SAVE teruggezet op 1. Voor u blijft het resultaat onzichtbaar, omdat CAT een bestandsnaam slechts één keer afdrukt. Het voordeel van deze methode is, dat als iemand per ongeluk een bestand wist met het ERASE-commando, nog altijd een kopie overblijft. De resterende kopieën kunnen altijd op de gebruikelijke wijze worden geladen. Maakt u x kopieën van een bestand, dan moet u, om het gehele bestand te wissen, x maal een ERASE uitvoeren. Het is duidelijk dat meervoudige kopieën x maal zoveel ruimte gebruiken op een cartridge. Merk op dat meervoudige kopieën niet bestand zijn tegen een FORMAT "m"-commando, dat, zoals reeds beschreven, alle kopieën in één keer wist.

3. Microdrive-bestandsmanipulatie

3.1 Seriële bestanden

Een bestand is een blok gegevens waaraan gegevens kunnen worden toegevoegd, maar ook waarvan gegevens kunnen worden gelezen. Met *seriële* bestanden moet u gegevens in een bepaalde volgorde lezen en wegschrijven. Bijvoorbeeld, als u het tiende item wilt hebben, moet u eerst de eerste negen items met gegevens lezen voordat u de tiende kunt lezen. Met de Spectrum worden bestanden gewoonlijk op een cartridge opgeslagen, en wel volgens de seriële methode. (De andere methode, die *random access* wordt genoemd, is niet beschikbaar vanuit BASIC.)
Hoofdstuk 1 heeft uiteengezet hoe streams werken en behandelde de channels "K", "S" en "P". Om gegevens op een microdrive op te slaan, met een andere methode dan het SAVEn van programma's of arrays, wordt van een andere channel-identificatie gebruik gemaakt – "m" (of "M"); dit geeft Microdrive aan.
Op een identieke wijze als reeds is beschreven, moet het OPEN #-commando worden toegepast om een stream toe te kennen aan het "m"-channel. De syntax wijkt iets af, omdat meer informatie benodigd is voor de Spectrum. Deze syntax is als volgt:

```
OPEN #n;"m";d;"naam"
```

en creëert een nieuw channel met de identificatie "m" op drive **d** met de naam "naam" en kent hieraan het streamnummer **n** toe. Omdat het bestand met de naam "naam" zojuist is gecreëerd, staan er geen gegevens in en is het voorbestemd als *schrijf* bestand.

3.2 Schrijfbestanden

Dit zijn bestanden die nog niet bestonden voordat het OPEN #-commando werd uitgevoerd, maar daardoor werden aangemaakt. Het hoofdcommando om gegevens naar een Microdrive te zenden is de PRINT #n;-opdracht, die evenals het bekende PRINT-commando tekens afdrukt, maar in dit geval op een cartridge. Als eenvoudig voorbeeld, kunt u het volgende programma starten. Let vooral op wanneer de Microdrive actief is:

```
100 OPEN #14;"m";1;"testbestand"
110 FOR i=1 TO 500
120 PRINT i;" ";
130 PRINT #4;i
140 NEXT i
150 CLOSE #4
```

U zou misschien hebben verwacht, dat de Microdrive continu actief zou zijn, omdat ieder teken moet worden afgedrukt; integendeel! Na het OPEN #-commando, is de drive even actief omdat deze op zoek is naar een bestand met dezelfde naam, dat niet gevonden mag worden, gezien het feit dat het om een schrijfbestand gaat. Het OPEN #-commando creëert ook een geheugengebied, een *buffer,* waar tijdelijk tekens in worden opgeslagen.

Wordt er een PRINT #-opdracht uitgevoerd, dan worden de tekens opgeslagen in de buffer, maar ze worden niet naar de cartridge gestuurd, tenzij de buffer vol is. Is de buffer vol, dan is de Microdrive gedurende een seconde actief, omdat de 512 tekens vanuit de buffer naar de cartridge worden getransporteerd. Daarna wordt de buffer schoongemaakt, gereed voor nog meer tekentransport.
Dit is de reden dat de Microdrive slechts enkele malen even snel actief is, want de buffer wordt vier keer gevuld. Een CLOSE #-opdracht is absoluut noodzakelijk, wanneer met schrijfbestanden wordt gewerkt, omdat met een gedeeltelijk gevulde buffer zonder CLOSE #-opdracht, de laatste gegevens, waarvan u dacht dat ze via de PRINT #-opdracht verstuurd zouden worden, de cartridge nooit zullen bereiken, waardoor het bestand incompleet raakt. Een CLOSE #-opdracht stuurt datgene wat nog resteert in de buffer naar de cartridge en verwijdert vervolgens de buffer uit het geheugen. Indien u een CLEAR # uitvoert terwijl er een schrijfstream is geopend, zal het bestand incompleet raken en daardoor onbruikbaar zijn. Dergelijke gedeeltelijke bestanden zouden direct moeten worden gewist. Wilt u een OPEN #-uitvoeren voor een schrijfbestand dat reeds is geopend, dan volgt de boodschap `Reading from a "write" file.`

3.3 Leesbestanden

Zodra een schrijfbestand is gecreëerd via een OPEN #-commando, en is voorzien van gegevens en dit is afgesloten door middel van een CLOSE #-commando, spreken we van een leesbestand en is het zichtbaar op de cartridge via het CAT-commando. Om nu gegevens te lezen vanuit dit bestand, moet het worden geopend met dezelfde OPEN #-opdracht als eerder gebruikt. Het eigenlijke lezen van gegevens wordt gewoonlijk uitgevoerd met behulp van het INPUT #-commando. Om het bestand dat we hebben gecreëerd door middel van het vorige programma te lezen, brengt u het volgende in:

```
100 OPEN #4;"m";1;"testbestand"
110 INPUT #4;i
120 PRINT i;" ";
130 GO TO 110
```

Dit creëert eerst een leesbestand-channel op stream 4. De variabele **i** wordt vervolgens gelezen vanuit het bestand (regel 110) en afgedrukt op het beeldscherm. Het programma springt steeds terug, totdat het stopt met het bericht `End of file`. We zullen nog zien hoe we deze end-of-file-situatie kunnen onderkennen, voordat het bericht verschijnt.) Zoals u wellicht heeft geconstateerd, gedurende de uitvoering van het programma, wordt voor leesbestanden ook gebruik gemaakt van een buffer, ook weer van 512 tekens lengte. Zodra een INPUT #-commando wordt uitgevoerd wordt een blok van 512 tekens (of minder, indien dit het laatste blok betreft) van de cartridge gelezen en in de buffer opgeslagen. Indien een teken benodigd is voor de INPUT-routine (of INKEY$ #) wordt dit uit de buffer genomen. Wordt het laatste teken in de buffer gelezen, dan wordt een volgend blok van 512 ingelezen vanaf de cartridge, totdat er geen blokken meer over zijn in het bestand, zodat een `End of file` bericht verschijnt.
Er is een andere manier om tekens uit een bestand in te lezen, los van het gebruik van de INPUT #-opdracht – nl. de INKEY$ #-functie, die al even genoemd werd in

hoofdstuk 2. Anders dan bij de INPUT-opdracht, die een hele reeks van tekens behandelt tot aan het einde van een regel en deze in een variabele kopieert, leest INKEY$ # slechts een enkel teken. Anders dan bij de gewone INKEY$-functie, "wacht" deze op een teken – INKEY$ # met een Microdrive stream zal nooit met een lege string terugkeren (" "). Als u klaar bent met het lezen van een bestand, moet u afsluiten met het CLOSE #-commando. Ook al is dit niet zo belangrijk als bij schrijfbestanden, uit oogpunt van netheid is het toch aan te bevelen en bovendien bespaart het geheugenruimte, vanwege de onnodige buffers. Het is prima om een CLEAR # uit te voeren bij geopende leesbestanden.
Bij het gebruik van de INPUT #-opdracht wordt de teller, die gebruikt wordt door de Spectrum om het "scroll?" bericht te genereren, teruggezet, zodat gedurende het lezen van een bestand en het afdrukken ervan op het scherm, continu wordt gescrolled. Indien dit niet gewenst is in het programma, gebruik dan de volgende regel:

```
LET sc=PEEK 23692:INPUT # (iets):POKE 23692,sc
```

De reden hiervoor is dat iedere INPUT-opdracht de teller terugzet, wat gewoonlijk niet wordt opgemerkt. In feite is een simpele manier om het "scroll"-bericht tegen te houden, de volgende opdracht.

```
INPUT ""
```

Opmerkingen bij PRINT en INPUT

U dient erg voorzichtig te zijn in het gebruik met PRINT # en INPUT # bij Microdrives, gezien het effect van de scheidingstekens die worden gebruikt tussen de argumenten. Wanneer u op het scherm of de printer afdrukt, brengt een komma (,) u naar de volgende helft van de regel en een apostrof (') veroorzaakt een nieuwe regel. Echter, een microdrive kent geen regels en dus betekent het afdrukken van een komma in een bestand dat alleen de controlecode van een komma, dit is CHR$6, wordt verstuurd. Evenzo zal bij de apostrof CHR$13 (het nieuwe-regel-teken) worden gestuurd. Tekencode 6 kan de Spectrum in verwarring brengen wanneer u gebruik wilt maken van INPUT # (maar niet INKEY #). Ter illustratie, moet u het volgende programma eens proberen:

```
10 OPEN #4;"m";1;"fouttest"
20 LET a$="eerste";LET b$="tweede"
30 PRINT #4;a$,b$
40 CLOSE #4
50 OPEN #4;"m";1;"fouttest"
60 INPUT #4;a$
70 INPUT #4;b$
80 CLOSE #4
```

U zult de boodschap `End of file` op regel 70 niet verwachten, maar het PRINT #-commando op regel 30 stuurt de tekst "eerste" naar de cartridge gevolgd door een komma-teken (CHR$ 6), daarna de tekst "tweede" en ten slotte een nieuwe-regel-teken (CHR$ 13). Een INPUT #-opdracht verwacht dat variabelen in een bestand zijn gescheiden door een CHR$ 13, dus las dit a$ in als "eerste" + CHR$ 6 + "tweede", en maakte het bestand leeg, waardoor de poging om b$ daarna in te lezen resulteerde in de foutboodschap.

Scheidingstekens in INPUT #-opdrachten kunnen onverwachte problemen veroorzaken. Als u probeert

```
10 OPEN $4;"m";1;"testbestand"
20 INPUT #4;a$,b$
```

(waarbij ''testbestand'' wordt verondersteld te bestaan) zal de foutboodschap `Writing to a "read" file` verschijnen. Dit wordt veroorzaakt, doordat de komma in regel 20 een poging doet om een CHR$ 6 naar een leesbestand te versturen. Als quote-tekens worden opgenomen in gegevensbestanden, kan het lezen van strings, door gebruik te maken van INPUT #, problemen veroorzaken. U moet daarom altijd gebruik maken van PRINT # ...LINE.
De gouden regels voor het gebruik van PRINT # en INPUT # zijn:

1. *Scheid variabelen altijd met behulp van apostrofs, wanneer u gebruik maakt van* `PRINT #` *of gebruik aparte regels bijv.* `PRINT #4;a$'b$` *of* `PRINT #4;a$:PRINT;#4b$`
2. *Gebruik altijd het punt-kommateken als scheidingsteken in INPUT-opdrachten en gebruik LINE voor stringvariabelen bijv.* `INPUT #4; LINE a$; LINE b$`

3.4 Het gebruik van MOVE met Microdrive-bestanden

Het MOVE-commando kent vele toepassingen, maar alleen die, welke direct beschikbaar zijn voor de Microdrive-bestanden worden hier behandeld. Heeft u een gegevensbestand en wilt u zien wat er in staat, zonder daarbij het bestand te openen en dit teken voor teken te lezen, dan kunt u het volgende doen

```
MOVE "m";d;"naam" TO #2
```

(hierin is TO een parameter), dat ervoor zorgt dat de inhoud van het aangegeven bestand op stream 2 wordt afgedrukt, dit is het scherm. (Indien u een MOVE probeert met een ander bestandstype, zoals een programma, dan volgt de foutboodschap ''Wrong file type''.) Wilt u een afdruk van het bestand op de printer, dan kunt u gebruik maken van

```
MOVE "m";d;"naam" TO #3
```

hetgeen een afdruk stuurt naar stream 3, dit is de ZX printer. U kunt MOVE ook gebruiken om gegevensbestanden te kopiëren.

```
MOVE "m";d1;"bestand1" TO "m";d2; "bestand2"
```

dit kopieert de gegevens vanuit ''bestand1'' op drive d1 naar drive d2 onder de naam ''bestand2''.

3.5 Toevoegen aan bestanden

Veronderstel dat u een bestand heeft vol met gegevens, zoals een lijst met programmanamen en u wilt enkele gegevens toevoegen aan het einde van deze lijst. Omdat u niet kunt afdrukken in een leesbestand, moet u een nieuw bestand creëren, hierin vervolgens de inhoud van het oude bestand kopiëren en, terwijl het bestand nog steeds is geopend, de nieuwe gegevens eraan toevoegen. Een manier is, om tel-

kens een groep gegevens te lezen vanaf het oude bestand en dan vervolgens af te drukken op het nieuwe bestand, maar dit is een zeer trage en inefficiënte methode en zou niet eens voor alle soorten gegevens werken.
De aangewezen methode hiervoor is, gebruik te maken van het MOVE-commando, zoals het volgende voorbeeld laat zien, waarin de strings "Space Invaders" en "25 december 1984" moeten worden toegevoegd aan het bestand "pgmnaam":

```
10 OPEN #;"m";1;"pgmnaam2"
20 MOVE "m";1;"pgmnaam" TO #4
30 PRINT #;"Space Invaders" "25 december 1984"
40 CLOSE #4
```

Indien u het nieuwe bestand dezelfde naam wilt geven als het oude, kunt u de volgende regels toevoegen

```
50 ERASE "m";1;"pgmnaam"
60 MOVE "m";1;"pgmnaam2" TO "m";1;"pgmnaam"
70 ERASE "m";1;"pgmnaam2"
```

Het bovenstaande programma zal sneller draaien indien de twee bestanden staan opgeslagen op verschillende Microdrives.
Het MOVE-commando is ook zeer nuttig wanneer twee gegevensbestanden aan elkaar moeten worden toegevoegd, bijvoorbeeld als de bestanden "een" en "twee" moeten worden samengevoegd tot "drie"

```
10 OPEN #4;"m";1;"drie"
20 MOVE "m";1;"een" TO #4
30 MOVE "m";1;"twee" TO #4
40 CLOSE #4
```

De snelheid van dit programma kan aanzienlijk worden verbeterd, wanneer twee of drie Microdrives worden gebruikt, ieder voorzien van een bestand.

3.6 Het gebruik van LIST

Ook al is PRINT # de gebruikelijke methode om gegevens naar een cartridge te schrijven, LIST # kan ook voor dit doel worden toegepast. Het produceert een BASIC programma-listing op een stream, welke een Microdrive channel kan zijn. Wanneer u het bestand terugleest, zullen de instructies (zoals STOP) een teken lang zijn (CHR$ 226 voor STOP) en geen individuele tekens bevatten met spaties tussen de parameters. Het converteren van een programma in tekenformaat, door middel van LIST #, kan zeer handig zijn wanneer u een BASIC-programma wilt behandelen of doorzoeken, misschien met een tekstverwerker.
Het volgende programma doorzoekt een gegevensbestand dat is gecreëerd via OPEN #, LIST # en daarna een CLOSE #-opdracht, voor iedere reeks en drukt iedere regel die deze bevat op het scherm af. Dit is zeer nuttig voor het opzoeken van de namen van variabelen, of bepaalde GO ... TO-opdrachten. Het kan worden gebruikt voor het doorzoeken van ieder soort bestand en dus niet alleen programmabestanden.

Het programma stopt uiteindelijk met het bericht "End of file".

```
999 REM Zoek-Programma
1000 INPUT "Drive nr.?";d,"Bestandsnaam?";LINE f$
1005 CLOSE #4: OPEN #4;"m";d;f$
1010 INPUT "Gezochte string?";LINE b$
1020 INPUT #4; LINE a$
1030 FOR i=1 TO LEN a$-LEN b$
1035 IF a$(i TO i+LEN b$-1)=b$ THEN PRINT a$:
PAUSE 50:GO TO 1050
1040 NEXT i
1050 GO TO 1020
```

Indien u naar specifieke instructies wilt zoeken (bijv. GO TO) moet u THEN intikken, dit bewerkstelligt, dat de Spectrum in de K-mode wordt gezet, vervolgens op de relevante toets drukken (G voor GO TO) en vervolgens THEN weer verwijderen.

3.7 End of file (EOF)

Wanneer u een bestand leest, zult u uiteindelijk het einde bereiken. Probeert u daarna meer te lezen, dan volgt een boodschap "End of file". Er zijn echter diverse methoden om deze EOF-situatie eerder te onderkennen dan het verschijnen van de boodschap, hetgeen programma's netter en efficiënter maakt.
De eenvoudigste methode om EOF te detecteren is, aan het einde van het bestand een speciaal teken te plaatsen, dat normaal niet wordt gebruikt, vlak voor de CLOSE van het bestand. Zelf geef ik de voorkeur aan de string CHR$ 0 + CHR$ 0, die ik nooit zou gebruiken voor andere doeleinden. Dit is de beste methode voor de meeste programma's, maar kan niet worden gebruikt voor programma's die zijn ontworpen voor gebruik van verschillende soorten bestanden, of die programma- of machinetaalbestanden kunnen lezen (door gebruik te maken van een methode die later wordt uiteengezet).
Andere, op disk gebaseerde, BASIC-programma's, beschikken over een functie ON EOF GO TO of, meer algemeen, ON ERROR GO TO, welke een sprong maakt naar een specifieke regel, indien een EOF (of een andere foutconditie) optreedt. Helaas biedt de Spectrum een dergelijke functie niet, maar een paar regels BASIC detecteert 99% van de EOFs. Als u de volgende regels toevoegt aan een programma (zoveel mogelijk aan het begin ervan) dan zal de functie FN E(x), waarin x het streamnummer voorstelt, een 1 geven in geval van een leeg bestand en een 0 in alle andere gevallen.

```
10 DEF FN E(A)=((INT (PEEK (FN D(a)+67)/2)-2*INT
(PEEK (FN D(A)+67)/4)AND (FN P(FN D(A)+11) >=FN
P(FN D(A)+69))))
11 DEF FN P(P)=PEEK P+256*PEEK (P+1)
12 DEF FN D(D)=FN P(D*2+23574)+FN P (23631)-1
```

(Indien stream **x** niet open is of niet aan het "M"-channel is toegekend, dan is het resultaat van geen enkele betekenis.) De 1% waarin het fout gaat, geldt voor die gevallen, waarin het aantal tekens in de buffer precies een veelvoud is van 512, het-

geen gelukkig zelden voorkomt. Gedurende het creëren van een schrijfbestand, kunt u voorkomen dat dit gebeurt door middel van de functie

```
FN P(FN D(X)+11)
```

vlak voor de CLOSE van stream x. Is het resultaat 0, druk dan van te voren een extra teken af op de stream.
De volgende machinetaalroutine biedt een alternatief en kan worden gebruikt als ON EOF GOTO functie. De routine is 67 bytes lang en dynamisch. Zet eerst de variabele **eof** op de juiste positie en **line** op het benodigde regelnummer (dat aanwezig moet zijn) en vervolgens een GO SUB 8150. De routine doet een POKE van iedere code en voert deze vervolgens uit. Als daarna een EOF optreedt, zal geen foutboodschap worden geproduceerd – er wordt naar de aangegeven regel gesprongen. De functie wordt verwijderd indien een fout optreedt, inclusief een EOF. Wilt u het regelnummer wijzigen, waar in geval van EOF naartoe wordt gesprongen, pas dan **line** aan en doe een GO SUB 8220. Zodra een EOF optreedt, vindt u de regel waarin deze optrad, via

```
LET ftregel=PEEK 23753+256*PEEK 23754
```

Doe dit voordat een microdrive-actie wordt uitgevoerd, omdat het anders een foutief resultaat oplevert.

ON EOF GOTO programmalijst

```
8145 REM **********************
8146 REM * ON EOF GOTO       *
8147 REM **********************
8148 REM eof=startpositie,line=f
outroutine
8149 REM aanbevolen:65190/32490
8150 RESTORE 8250
8160 LET c=0
8170 FOR i=eof TO eof+66
8180 READ a: LET c=c+a
8190 IF a<>260 THEN  POKE i,a
8200 NEXT i
8210 IF c<>6456 THEN  PRINT "Con
trolesom fout": STOP
8220 POKE eof+49,line-256*INT (l
ine/256)
8225 POKE eof+50,INT (line/256)
8230 RANDOMIZE USR eof
8240 RETURN
8250 DATA 33,17,0,9,235,42,61,92
8260 DATA 115,35,114,207,49,1,0,
0
8270 DATA 201,42,61,92,58,58,92,
254
8280 DATA 7,194,3,19,94,35,86,21
3
8290 DATA 205,176,22,253,203,55,
174,205
8300 DATA 110,13,42,69,92,34,201
,92
```

```
8310 DATA 17,260,260,33,66,92,11
5,35
8320 DATA 114,35,54,1,253,54,0,2
55
8330 DATA 195,125,27
```

3.8 Het gebruik van programma's als gegevensbestanden

Gewoonlijk, indien u met behulp van OPEN een Microdrive-channel opent met een programmanaam, of een naam van een ander type niet-gegevensbestand, verschijnt de foutboodschap `Wrong file type`. Het kan zeer handig zijn om dergelijke bestanden toch te kunnen lezen vanuit de BASIC, dit bereikt u met de volgende machinetaalroutine, waarmee u naar believen een OPEN # kunt uitvoeren. De routine is vrijwel gelijk aan de OPEN #-opdracht, maar opent ieder bestand, onafhankelijk van de soort. Het vertelt u ook of een bestand bestaat of niet, hetgeen op zichzelf al zeer handig is. Zet eerst de variabele **open** op de waarde van de positie, waar u de routine wilt hebben (aanbevolen wordt gebruik te maken van adres 32320 voor de 16K en 65090 voor de 48K) en doe vervolgens GO SUB 8350. De GO SUB doet een POKE voor iedere code, maar voert deze niet uit.

OPEN – willekeurig programmalijst

```
8344 REM ************************
8345 REM *OPEN # ieder bestand *
8346 REM ************************
8347 REM open=startpositie
8348 REM aanbevolen:65090/32320
8350 RESTORE 8400: LET c=0
8360 FOR i=open TO open+93
8370 READ a: LET c=c+a
8375 POKE i,a
8380 NEXT i
8385 IF c<>10725 THEN  PRINT "Co
ntrolesom fout": STOP
8390 RETURN
8400 DATA 58,216,92,205,39,23,33
,17
8410 DATA 0,175,237,66,1,0,0,216
8420 DATA 50,215,92,33,10,0,34,2
18
8430 DATA 92,42,123,92,34,220,92
,58
8440 DATA 216,92,135,33,22,92,95
,22
8450 DATA 0,25,217,229,217,229,2
07,34
8460 DATA 221,203,24,70,40,13,17
5,207
8470 DATA 33,207,44,225,217,225,
217,1
8480 DATA 1,0,201,221,203,4,190,
175
```

```
8490 DATA 229,207,33,209,225,115
,35,114
8500 DATA 221,126,67,230,4,198,2
,79
8510 DATA 6,0,217,225,217,201
```

Wilt u er gebruik van maken, dan moet u opgeven in welke drive, stream en bestand u geïnteresseerd bent. Hiervoor dient u het drivenummer in positie 23766 te POKEn, het streamnummer in positie 23768 en de naam van het bestand in de eerste 10 posities van de user-defined graphics. Bijvoorbeeld, als u wilt uitvoeren `OPEN #6;"m";2;"test"`("test" kan ieder type bestand zijn) dan horen daar de volgende instructies bij:

```
3500 POKE 23766,2:REM drivenummer
3510 POKE 23768,6:REM streamnummer
3520 LET a$="test"
3530 FOR i= 1 TO 10
3540 IF i > LEN a$ THEN POKE USR "a" + i-1,32: GO
TO 3560
3550 POKE USR "a"+i-1,CODE a$(i)
3560 NEXT i
3570 LET a = USR open: REM roept de routine aan
```

Let erop, als de bestandsnaam minder dan 10 tekens lang is, dat u de resterende posities opvult met spaties (CHR$ 32s). De waarde die de **USR open** functie oplevert (**a** in het voorbeeld) kan de volgende zijn:

0 – stream reeds geopend
1 – bestand niet gevonden
2 – gegevensbestand
6 – geen gegevensbestand

Indien het bestand niet blijkt te bestaan, dan is de stream gesloten en het bestand niet gecreëerd. Daarom is de routine alleen geschikt voor leesbestanden – schrijfbestanden worden er niet mee gecreëerd, hetgeen anders is dan bij de gebruikelijke OPEN #-opdracht.
Is de teruggegeven waarde 6, hetgeen dus aangeeft dat het een niet-gegevensbestand betreft, hoe komt u dan te weten om welk type bestand het gaat? Het antwoord vindt u in de eerste negen tekens die van het bestand zijn ingelezen. Deze tekens bevatten alle attributen voor het bestand, inclusief het type. Het eerste teken bepaalt het type, dus:

0 – BASIC-programma
1 – numerieke array
2 – string array
3 – bytes

De daaropvolgende acht bytes bevatten gegevens omtrent lengte, start, regelnummer enz. en worden eigenlijk overgenomen van de systeemvariabelen HD-00 tot HD-11 (zie hiervoor Appendix A voor precieze details). Na deze eerste negen bytes, volgt het eigenlijke bestand. Het volgende programma, Ware CATalogue, is een

zeer verbeterde versie van de CAT-functie. Het geeft behalve de bestandsnaam, ook informatie over het bestandstype, de lengte, auto-start regelnummer enz. Het maakt gebruik van de machinetaalroutines stream14-z$ en OPEN #-ieder-bestand. Het is langzamer dan de CAT-functie, maar geeft desondanks een grote hoeveelheid bruikbare informatie.

Ware CATalogue programmalijst

```
4997 REM ************************
4998 REM * ware CATalogue       *
4999 REM ************************
5000 LET z$="": CAT  #14,d
5010 CLS : POKE 23766,d: POKE 23
768,15
5020 PRINT  INVERSE 1;"Titel:";z
$( TO 10)
5030 LET z$=z$(13 TO )
5040 IF LEN z$<10 THEN  GO TO 53
30
5050 FOR i=1 TO 10
5060 POKE USR "a"+i-1,CODE z$(i)
5070 NEXT i
5080 CLOSE #15: LET z=USR open
5090 PRINT z$( TO 10);"  ";
5100 IF z=2 THEN  PRINT "Gegeven
sbestand": GO TO 5300
5110 LET a$="": FOR i=1 TO 9
5120 LET a$=a$+INKEY$#15: NEXT i
5130 LET z=CODE a$(1)
5140 GO TO 5150+z*40
5149 REM z=0 Programma
5150 PRINT "Programma  LINE ";CO
DE a$(8)+256*CODE a$(9)
5160 GO TO 5300
5189 REM z=1 Nummerieke array
5190 PRINT "Array  ";CHR$ (CODE
a$(6)-32);"()"
5230 PRINT "Array  ";CHR$ (CODE
a$(6)-96);"$()"
5240 GO TO 5300
5269 REM z=3 Code
5270 PRINT "Code  ";CODE a$(4)+2
56*CODE a$(5)
5280 PRINT ",";CODE a$(2)+256*CO
DE a$(3)
5300 CLOSE #15
5310 LET z$=z$(12 TO )
5320 GO TO 5040
5330 PRINT 'z$(2 TO LEN z$-1);"K
 vrije bytes"
5340 RETURN
```

3.9 Het aanmaken van niet-gegevensbestanden

Evenals de mogelijkheid om niet-gegevensbestanden te lezen van een cartridge, is het mogelijk om ieder type bestand te creëren vanuit de BASIC. Er is geen machinetaal voor nodig, maar alleen een POKE-commando. Het gebruik van deze faciliteit kan zeer moeilijk zijn, en beginners wordt het gebruik afgeraden omdat fouten in de toepassing ervan tot een systeemcrash kunnen leiden.
De methode om dit te bereiken is om een OPEN #-commando te volgen dat een bestand creëert met regels als

```
100 DEF FN p(p)=PEEK p + 256*PEEK(p+1)
110 POKE FNp(s*2+23566+8)+FN p(23631)+66,4
```

waarin **s** het streamnummer voorstelt in het OPEN-commando. Dit misleidt het systeem tot de aanmaak van niet-gegevensbestanden, zodra gegevens via de stream naar de cartridge worden geschreven.
Na de POKE, zou u eigenlijk negen tekens via PRINT op de Stream moeten afdrukken. Het eerste teken geeft het bestandstype aan en de volgende acht bepalen de parameters van het bestand en zijn gelijk aan eerder genoemde velden, voor het lezen van bestanden – de systeemvariabelen HD-00 tot HD-11 (zie ook Appendix A).
Als voorbeeld schrijft het volgende programma alle BASIC-variabelen in een programma, evenals de routine in hoofdstuk 2, maar met een groot verschil – in plaats van ze te bewaren als twee CODE-bestanden, worden ze hier bewaard als een programmabestand, zodat ze in andere programma's kunnen worden geMERGEd. Dit programma kan veel langzamer zijn dan de andere methode, maar is veel flexibeler.

SAVE variabelen programmalijst

```
3999 REM SAVE variabelen als pro
gramma
4000 OPEN #4;"m";1;"variabelen"
4010 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
4020 LET d=FN p(4*2+23566+8)+FN
p(23631)-1
4030 IF PEEK (d+4)<>CODE "M" THE
N  PRINT "Fout": STOP
4040 POKE d+67,4: REM SAVE
4045 FOR i=1 TO 2: NEXT i
4050 LET z=FN p(23641)-FN p(2362
7)-1
4055 LET z=FN p(23641)-FN p(2362
7)-1
4060 PRINT #4;CHR$ 0;: REM Progr
amma identificatie
4070 PRINT #4;CHR$ (z-256*INT (z
/256));CHR$ INT (z/256);
4080 PRINT #4;CHR$ PEEK 23627;CH
R$ PEEK 23628;: REM start
4090 PRINT #4;CHR$ 0;CHR$ 0;: RE
M Programma lengte
```

```
4100 PRINT #4;CHR$ 255;CHR$ 255;
: REM "regelnummer"
4110 FOR i=FN p(23627) TO FN p(2
3627)+z-1
4120 PRINT #4;CHR$ PEEK i;
4130 NEXT i
4150 CLOSE #4
4160 RETURN
```

Regel 4000 creëert het schrijfbestand, daarna doet regel 4040 de benodigde POKE-opdracht. Regels 4045 en 4050 lijken overbodig, maar zijn beide van vitaal belang. De variabele **z** bevat dan het aantal bytes in het variabelengeheugen (regel 4055), waarna regel 4060 het eerste teken afdrukt, daarmee aangevend dat het om een programmabestand gaat. Regel 4070 draagt zorg voor het opgeven van de lengte (2 bytes) van het bestand en regel 4080 drukt twee tekens af, welke het startadres aangeven van het variabelengeheugen. Regel 4090 zet de programmalengte op 0 en regel 4100 zet 65535 op als auto-start regelnummer (dit is geen auto-start). De lus van 4110 tot 4130 stuurt vervolgens iedere byte van de variabelen naar het bestand, voordat regel 4150 deze sluit (CLOSE).

Andere manieren om gegevens te lezen
Ondanks het feit dat INPUT # en INKEY$ # de hoofdcommando's zijn voor het lezen van bestanden, kan het MOVE commando veel nuttiger zijn. Als de stream 14-z$-routine actief is, zal de regel

```
LET z$="":MOVE "m";1;"bestand"TO #14
```

het gehele gegevensbestand in een keer inlezen vanaf de cartridge en deze opslaan in de variabele z$, voorzover het geheugen hiervoor ruimte biedt. Het bestand kan dan worden gemanipuleerd en gemodificeerd indien nodig en later worden teruggeschreven naar een ander bestand, via een enkele PRINT-opdracht bijvoorbeeld:

```
OPEN #4;"m";1;"bestand2";PRINT #4;z$;:CLOSE #4
```

Deze methode heeft een minimale invloed op de slijtage-onderhevigheid van cartridges, hetgeen een belangrijke factor kan zijn, indien u te maken heeft met zeer lange gegevensbestanden, die regelmatig worden geraadpleegd. De drive hoeft nu slechts tweemaal actief te zijn – eenmaal voor het lezen en eenmaal om te schrijven.

3.10 Statusroutine

De volgende routine, statusroutine, stelt een programma in staat, te bepalen of een Microdrive is aangesloten of niet, of dat een cartridge aanwezig is of niet en of deze beveiligd is tegen ongewenst wissen. Zet de variabele **stat** op de gewenste geheugenlocatie en voer dan een GO SUB 8550 uit.

STATUS programmalijst

```
8545 REM ************************
8546 REM *  S T A T U S         *
8547 REM ************************
```

```
8548 REM stat=start adres
8549 REM aanbevolen:64960/32190
8550 RESTORE 8630: LET c=0
8560 LET x2=INT ((stat+100)/256)
: LET x1=stat-256*x2+100
8570 FOR i=stat TO stat+128
8580 READ a: LET c=c+a
8590 POKE i,a
8600 NEXT i
8610 IF c<>14341+4*(x1+x2) THEN
 PRINT "Controlesom fout": STOP
8620 RETURN
8630 DATA 58,214,92,243,24,33,33
,136
8640 DATA 19,43,125,180,32,251,3
3,136
8650 DATA 19,6,6,219,239,230,4,3
2
8660 DATA 4,16,248,24,84,43,124,
181
8670 DATA 32,239,1,0,0,24,84,17
8680 DATA 0,1,237,68,198,9,79,6
8690 DATA 8,13,32,20,122,50,247,
0
8700 DATA 62,238,211,239,205,x1,
x2,62
8710 DATA 236,211,239,205,x1,x2,
24,17
8720 DATA 62,239,211,239,123,211
,247,205
8730 DATA x1,x2,62,237,211,239,2
05,x1
8740 DATA x2,16,214,122,211,247,
62,238
8750 DATA 211,239,24,162,197,245
,1,135
8760 DATA 0,11,120,177,32,251,24
1,193
8770 DATA 201,219,239,230,1,1,1,
0
8780 DATA 32,1,3,197,175,207,33,
193
8790 DATA 201
```

De GO SUB plaatst de machinecode op de juiste plaats, maar voert deze niet uit. Voor het gebruik, moet u een POKE 23766 uitvoeren om het drivenummer mee te geven. Gebruik daarna enkele van de volgende regels:

```
2500 LET a=USR stat
2510 IF a=0 THEN PRINT "Microdrive niet
aangesloten"
2520 IF a=1 THEN PRINT "Cartridge aanwezig"
2530 IF a=2 THEN PRINT "Cartridge beveiligd"
```

3.11 Kleurcommando's

In de Sinclair interface-handleiding wordt zijdelings gemeld dat kleurcommando's mogelijkerwijs niet werken na het gebruik van andere channels dan "K", "S" of "P". In feite gaat het om een serieus probleem, waar u zich zeer bewust van dient te zijn, wanneer u met bestanden manipuleert.
Het probleem is, dat na gebruik van de interface channels voor uitvoer, de permanente kleurcommando's (bijv. PAPER 3) ogenschijnlijk geen effect hebben. Bovendien worden er gegevens gezonden naar schrijfbestanden. Om dit te corrigeren, voert u een nep **PRINT**;-opdracht uit, voordat u de kleuren zet. Om de fout te illustreren, kunt u het volgende uitproberen:

```
10 OPEN #4;"m";1;"kltest"
20 PRINT #4;"Eerste regel"
30 FLASH 1: PAPER 4: CLS
40 PRINT #4;"Tweede regel"
50 CLOSE #4
60 MOVE "m";1;"kltest" TO #2
```

Regel 10 creëert een schrijfbestand "kltest", en regel 20 stuurt een regel gegevens naar dit bestand. Regel 30 zou ervoor moeten zorgen dat het gehele scherm groen flitst, maar in de plaats daarvan worden kleurcodes naar de cartridge weggeschreven. Regels 40 en 50 sturen een andere regel gegevens naar het bestand en sluiten deze. Ten slotte drukt regel 60 het gehele bestand af op het scherm, om hiermee de ongewilde maar ingevoegde kleurcodes te openbaren. Om dit op te lossen, voegt u toe

```
25 PRINT;
```

Microdrive-aanvullingen voor het streamoverzicht
De volgende regels kunnen worden toegevoegd aan het streamoverzichtprogramma, zodat meer details worden gegeven over Microdrive-streams.

Streamoverzicht programmalijst-aanvullingen

```
1630 IF f$="M" THEN  GO TO 2500
1840 IF FN p(d)<>8 AND FN p(d+2)
<>8 THEN  RETURN
1850 PRINT "Schaduw ROM uitvoer:
";FN p(d+5)
1860 PRINT "Schaduw ROM invoer :
";FN p(d+7)
2499 REM Channel M
2500 PRINT  INVERSE 1;"MICRODRIV
E"
2510 GO SUB 1800
2520 PRINT "Drivenummer          :
";PEEK (d+25)
2530 LET m=FN p(d+26)
2540 PRINT "Tape loop map:"
2550 FOR i=0 TO 31: FOR j=1 TO 6
2560 POKE 16384+2048+2*32+j*256+
31-i,PEEK (m+i)
```

```
2570 NEXT j: NEXT i
2575 PLOT 0,95: DRAW 255,0: PLOT
 0,88: DRAW 255,0
2580 PRINT '"Map gebied
:";m;"-";m+31
2590 PRINT "Cartridge naam     :
";
2600 FOR i=d+44 TO d+53
2610 PRINT CHR$ PEEK i;
2620 NEXT i: PRINT
2630 PRINT "Bestandsnaam       :
";
2640 FOR i=d+14 TO d+23
2650 PRINT CHR$ PEEK i;
2660 NEXT i: PRINT '"Vrije ruimt
e       :";
2670 LET f=0
2680 FOR i=0 TO 255
2690 LET f=f+NOT POINT (i,90)
2700 NEXT i
2710 PRINT f/2;"Kbytes"
2720 GO TO 1500
```

De extra gegevens zijn het drivenummer, het cartridgenummer, de cartridgenaam en de bestandsnaam en de vrije ruimte op de gekozen cartridge. Ook wordt een zgn. "tape loop map" getoond; dit is een soort van strepencode die een grafische weergave toont van de gebruikte sectoren op tape. Onderstaande afbeelding toont het resultaat van een gloednieuwe cartridge. De zwarte strepen geven de gebieden aan die in gebruik zijn of niet-bestaand zijn. Het grote zwarte gebied helemaal links toont een sector van de tape die niet bestaat, maar geeft aan dat het systeem met nog langere tapes overweg zou kunnen. De strepen in het midden geven aan waar de tape-einden aan elkaar zijn gelast. De streep helemaal rechts geeft de start aan van de eerste sector, die nooit wordt gebruikt.

Uitvoervoorbeeld voor een cartridge van streamoverzichtprogramma

```
Channel aanduiding : M
Uitvoer routine  : 0
Invoer routine   : 0
Schaduw ROM uitvoer : 4566
Schaduw ROM invoer  : 4366
Drivenummer         : 1
Tape loop map :
Map gebied          : 20792-20823
Cartridge naam      : Demo
Bestandsnaam        : test
Vrije ruimte        : 19Kbytes
```

De eerste belangrijke toevoeging aan het streamoverzicht bestaat uit de regels 1840-1860.
Zij controleren op de aanwezigheid van soorten interface channels – zijn deze aanwezig, dan worden de schaduw ROM invoer- en uitvoerroutineadressen afgedrukt. Deze extra regels worden ook gebruikt voor de RS232 en netwerk-toevoegingen aan het programma. De regels 2500-2720 drukken diverse details af over de stream.

De "tape loop map" wordt gecreëerd door middel van het direct POKEn van de gegevens uit het geheugen op het scherm; regels 2530-2575. De vrije ruimte wordt berekend door het tellen van de hoeveelheid witte strepen op het scherm (2670-2710).

3.12 Hoogste scoreroutine voor spelen

Om een toepassing te demonstreren van gegevensbestanden volgen nu enkele routines die een hoogste score kunnen toevoegen aan een spel, door de tabel op te slaan in een bestand. Ze worden gegeven in de vorm van twee subroutines – regels 9000 – het inlezen van de tabel, die gebruikt moet worden voordat het spel start en regels 9100 – die de tabel muteert indien een nieuwe hoogste score wordt bereikt.

Hoge score routine

```
8996 REM ************************
8997 REM * Hoogste score        *
8998 REM ************************
8999 REM Lees de gegevens in
9000 LET ns=5: DIM s(ns): DIM s$
(ns,10)
9010 CLOSE #4: OPEN #4;"m";1;"ho
gescore"
9020 FOR i=1 TO ns
9030 INPUT #4;s(i); LINE s$(i)
9040 NEXT i: CLOSE #4
9050 GO TO 9300
9099 REM vervang bestand igv hoo
gste score
9100 LET ns=5: IF sc<s(ns) THEN
 RETURN
9110 PRINT ' FLASH 1;"       Een
nieuwe hoogste score !"
9120 INPUT "Geef uw naam op (max
 10 letters)"; LINE a$
9130 FOR i=1 TO ns
9140 IF sc<=s(i) THEN  NEXT i
9150 ERASE "m";1;"hogescore"
9152 CLOSE #4
9155 OPEN #4;"m";1;"hogescore"
9160 FOR j=1 TO ns
9170 IF j<1 THEN  PRINT #4;s(j)'
s$(j)
9180 IF j=1 THEN  PRINT #4;sc'a$
9190 IF j>1 THEN  PRINT #4;s(j-1
)'s$(j-1)
9200 NEXT j
9210 CLOSE #4
9220 RETURN
9299 REM druk scoretabel af
9300 CLS
9310 PRINT  FLASH 1; INK 7; PAPE
R 0;TAB 8;"HOOGSTE SCORES";TAB 3
1;" "
```

```
9320 FOR i=1 TO ns
9330 PRINT  PAPER i; INK 9,,TAB
6;i;" ";s$(i);TAB 20;s(i),,,
9340 NEXT i
9350 RETURN
9989 REM creeer nieuw bestand
9990 OPEN #4;"m";1;"hogescore"
9992 FOR i=1 TO 5
9994 PRINT #4;0'" "
9996 NEXT i: CLOSE #4
```

De tabel wordt opgeslagen, bestaande uit 10 elementen in de volgorde eerste score, eerste naam, tweede score, tweede naam enz. De regels 9900 – en verder moeten alleen worden uitgevoerd als de tabel op nullen moet worden gezet. Om de tabel in te lezen, worden de regels 9000-9050 gebruikt. Regel 9000 zet het aantal items in de tabel (in **ns**), en de arrays s(), voor iedere score en s$() voor iedere naam. Regel 9010 opent het bestand om dit te kunnen lezen en de regels 9020-9040 lezen iedere score en naam en plaatsen deze in de arrays. Het bestand wordt afgesloten door middel van een CLOSE # en de tabel wordt afgedrukt vanaf regel 9300 –. Nadat het spel is beëindigd, moet de score worden opgenomen in de variabele **sc** en moet een GO SUB 9100 worden uitgevoerd. Regel 9100 kijkt of de score minder is dan de laagste in de tabel. Als dit zo is, wordt een RETURN uitgevoerd omdat de speler zich niet heeft gekwalificeerd voor een plaats in de tabel. Heeft de speler zich wel gekwalificeerd, dan brengt hij zijn naam in (regel 9120) en wordt zijn nieuwe positie in de tabel berekend door de regels 9130-9140 en geplaatst in de variabele **i.** Regel 9150 wist het oude bestand en regel 9155 creëert een nieuw schrijfbestand. De lus van 9160 tot 9200 drukt dan de nieuwe tabel af in het nieuwe bestand, door het invoegen van de nieuwe score en naam en het opschuiven van de gegevens daaronder met één plaats. Ten slotte voert regel 9210 een CLOSE # uit.

4. Unifile

Het programma in dit hoofdstuk is hoofdzakelijk een database-programma, dat gebruik maakt van de Microdrives en indien nodig, van een printer (ZX Printer of RS232). Het is voor een groot gedeelte gebaseerd op het programma unifile 1 van David Lawrence, in zijn boek *The Working Spectrum,* met zijn toestemming.
Het programma is bedoeld voor 48K-bezitters, doch later volgen enkele gegevens voor een aangepaste versie voor 16K-bezitters.

UNIFILE programmalijst

```
1000 PAPER 7: CLS : BORDER 7: IN
K 6: PAPER 0: PRINT  PAPER 2;"
 UNIFILE ";F$;TAB 31;" "
1010 PRINT "KEUZE MOGELIJKHEDEN:
"
1020 PRINT '"      1)CREEER NIEUW
 BESTAND"
1030 PRINT '"      2)VOER GEGEVEN
S IN"
1040 PRINT '"      3)ZOEK/TOON/MU
TEER"
1050 PRINT '"      4)STOP"
1055 PRINT '"      5)SAVE/LOAD/CA
T"
1060 PRINT ''"MAAK EEN KEUZE."
1070 PAUSE 0: LET z$=INKEY$
1080 CLS
1090 IF z$="1" THEN  GO SUB 1210
1100 IF z$="2" THEN  GO SUB 1440
1110 IF z$="3" THEN  GO SUB 2180
1120 IF z$="4" THEN  GO SUB 1150
1125 IF z$="5" THEN  GO TO 3500
1130 CLS
1140 GO TO 1000
1150 PRINT AT 10,2; INK 7; PAPER
 2;"DATABASE SYSTEEM AFGESLOTEN"
1160 BEEP 2,2
1180 INPUT "Heeft u gegevens ing
ebracht die u wilt bewaren ? (J/
N)";q$: IF q$="N" THEN  STOP
1190 SAVE "UNIFILE": PRINT "Spoe
l de tape terug, druk daarna op
een willekeurige toets om een VE
RIFY te starten": PAUSE 0: VERIF
Y "UNIFILE": STOP
1200 REM ************************
1210 REM BESTANDSOPBOUW
1220 REM ************************
1230 PRINT  PAPER 2;"     BESTAN
DSSTRUCTUUR          "
1235 GO SUB 4200
1240 PRINT '"HOEVEEL ONDERDELEN
IN EEN RECORD ?"
```

```
1250 INPUT x
1260 CLS
1270 DIM a$(x,20)
1280 PRINT  PAPER 2;"     NAAM PE
R ONDERDEEL              "
1290 FOR i=1 TO x
1300 PRINT "ONDERDEEL ";i;":";
1310 GO SUB 2780
1320 PRINT q$(2 TO )
1330 LET a$(i)=q$
1340 NEXT i
1350 DIM b$(28000)
1360 LET b$(1 TO 4)=CHR$ 2+CHR$
0+CHR$ 2+CHR$ 255
1370 DEF FN a()=256*CODE y$(2*s-
1)+CODE y$(2*s)
1380 DEF FN a$()=b$(c TO c+CODE
b$(c)-1)
1390 LET p=5
1400 LET y$=CHR$ 0+CHR$ 1+CHR$ 0
+CHR$ 3
1410 LET n=2
1420 RETURN
1430 REM ************************
1440 REM STANDAARD INVOER
1450 REM ************************
1460 LET r$=""
1470 PRINT  PAPER 2;"          RE
CORDS                    "
1480 PRINT '"BESCHIKBARE COMMAND
O'S:"
1490 PRINT '">VOER AANGEGEVEN ON
DERDEEL IN"''">""ZZZ"" OM TE STO
PPEN"
1500 PRINT "**********************
************"
1510 PRINT "BESTANDSGROOTTE:";p-
1;"/";LEN b$
1520 FOR i=1 TO x
1530 GO SUB 2810
1540 GO SUB 2780
1580 PRINT q$(2 TO )
1590 IF q$(2 TO )="ZZZ" THEN  RE
TURN
1600 LET r$=r$+q$
1610 NEXT i
1620 CLS
1630 GO SUB 1660
1640 GO TO 1440
1650 REM ************************
1660 REM PLAATS GEGEVENS
1661 REM IN BESTAND
1670 REM ************************
1680 IF p+LEN r$-1<LEN b$ THEN
GO TO 1730
```

```
1690 PRINT AT 14,10;"BESTAND IS
NU VOL"
1700 PRINT '''"  Druk op een will
ekeurige toets om verder te gaan
"
1710 PAUSE 0
1720 RETURN
1730 LET POWER=INT (LN (n-1)/LN
2)
1740 LET s=2^POWER
1750 LET t$=r$(2 TO CODE r$(1))
1760 FOR k=POWER-1 TO 0 STEP -1
1770 LET c=FN a()
1780 LET u$=FN a$()(2 TO )
1790 LET s=s+(2^k)*(t$>u$)-(2^k)
*(t$<u$)
1810 IF s>n-1 THEN  LET s=n-1
1820 IF s<2 THEN  LET s=2
1830 NEXT k
1840 LET c=FN a()
1850 LET u$=FN a$()(2 TO )
1860 IF t$<u$ THEN  LET s=s-1
1870 LET b$(p TO p+LEN r$-1)=r$
1880 LET n=n+1
1890 LET y$=y$(1 TO 2*s)+CHR$ IN
T (p/256)+CHR$ (p-256*INT (p/256
))+y$(2*(s+1)-1 TO )
1900 LET p=p+LEN r$
1910 RETURN
1920 REM ************************
1930 REM MUTATIE
1940 REM ************************
1950 LET s=s-1
1960 LET c=FN a()
1970 LET r$=""
1980 PRINT "ENTRY ";s-1;":-"
1990 FOR i=1 TO x
2000 GO SUB 2810
2010 GO SUB 2830
2020 PRINT AT 17,0; PAPER 2;"
     MUTEER                  "
2030 PRINT "BESCHIKBARE COMMANDO
'S:"
2040 PRINT ">""ENTER"" NEGEER MU
TATIE"'">""ZZZ"" VERWIJDEREN"'">
 GEEF VERVANGENDE GEGEVENS"
2050 GO SUB 2780
2060 IF LEN q$=1 THEN  LET r$=r$
+b$(c TO c+CODE b$(c)-1)
2070 LET c=c+CODE b$(c)
2080 CLS
2090 IF LEN q$=1 THEN  GO TO 212
0
2100 IF q$(2 TO )="ZZZ" THEN  GO
 TO 2130
```

```
2110 LET r$=r$+q$
2120 NEXT i
2130 GO SUB 3130
2140 IF q$(2 TO )="ZZZ" THEN  RE
TURN
2150 GO SUB 1660
2160 RETURN
2170 REM ************************
2180 REM ZOEKEN
2190 REM ************************
2200 LET s=2
2205 LET st=2
2210 PRINT  PAPER 2;"
  ZOEKEN                "
2220 PRINT ''"BESCHIKBARE COMMAN
DO'S:"
2230 PRINT ">VOER STRING IN VOOR
"'" NORMALE ZOEK"'">PREFIX ""SSS
"" VOOR SPECIALE ZOEK"'">PREFIX"
"III"" VOOR ZOEK"'" NAAR EERSTE
LETTER VAN STRING"'">""ENTER"" V
OOR EERSTE RECORD "'" VAN BESTAN
D"
2235 PRINT ">""PPP"" VOOR";"PRIN
TER" AND st=2;"SCHERM" AND st=3;
" UITVOER"
2240 PRINT "**********************
************"
2250 PRINT ''"GEEF ZOEKSTRING:";
2260 GO SUB 2780
2265 IF q$=CHR$ 4+"PPP" THEN  LE
T st=5-st: CLS : GO TO 2210
2270 PRINT q$(2 TO )
2280 LET s$=q$
2290 IF LEN s$=1 THEN  GO TO 251
0
2300 LET c=FN a()
2310 IF LEN s$<5 THEN  GO TO 243
0
2320 IF s$(2 TO 4)<>"III" THEN
GO TO 2390
2330 FOR i=s TO n
2340 LET s=i
2350 LET c=FN a()
2360 IF b$(c+1)=s$(5) THEN  GO T
O 2510
2370 NEXT i
2380 RETURN
2390 IF s$(2 TO 4)<>"SSS" THEN
GO TO 2430
2400 GO SUB 2920
2410 IF c4=1 THEN  GO TO 2510
2420 RETURN
2430 FOR i=1 TO x
2440 IF FN a$()=s$ THEN  GO TO 2
510
```

```
2450 IF FN a$()=CHR$ 2+CHR$ 255
THEN  RETURN
2460 LET c=c+CODE b$(c)
2470 NEXT i
2480 LET s=s+1
2490 LET c=FN a()
2500 GO TO 2430
2510 LET c=FN a()
2520 LET c4=0
2530 IF FN a$()=CHR$ 2+CHR$ 255
THEN  RETURN
2540 CLS
2545 IF st<>2 THEN  GO TO 4300
2550 PRINT "ENTRY ";s-1;":-"
2560 GO SUB 2850
2570 LET s=s+1
2580 PRINT AT 16,0; PAPER 2;"
       ZOEK                    "
2590 PRINT "BESCHIKBARE COMMANDO
'S:"
2600 PRINT ">""ENTER"" TOONT VOL
GEND RECORD"'">""ZZZ"" OM TE STO
PPEN"'">""AAA"" OM TE MUTEREN"'"
>""CCC"" OM DOOR TE ZOEKEN"
2610 INPUT p$
2620 CLS
2630 IF p$="CCC" THEN  GO TO 230
02
2640 IF p$="" THEN  GO TO 2510
2650 IF p$<>"AAA" THEN  GO TO 27
10
2660 LET c=FN a()
2670 CLS
2680 GO SUB 1930
2710 IF p$="ZZZ" THEN  RETURN
2720 IF p$="AAA" THEN  RETURN
2730 CLS
2740 GO TO 2260
2750 REM ************************
2760 REM FUNKTIONELE SUBROUTINES
2770 REM ************************
2780 INPUT q$
2790 LET q$=CHR$ (LEN q$+1)+q$
2800 RETURN
2810 PRINT a$(i,2 TO CODE a$(i,1
));":";
2820 RETURN
2830 PRINT FN a$()(2 TO )
2840 RETURN
2850 FOR i=1 TO x
2860 GO SUB 2810
2870 GO SUB 2830
2880 LET c=c+CODE b$(c)
2890 NEXT i
2900 RETURN
```

```
2910 REM ************************
2920 REM SPECIALE ZOEK
2930 REM ************************
2940 LET c4=0
2950 FOR h=s TO n-1
2960 LET s=h
2970 LET c=FN a()
2980 LET c1=c
2990 FOR i=1 TO x
3000 LET c1=c1+CODE b$(c1)
3010 NEXT i
3020 FOR j=c+1 TO c1-LEN s$+5
3030 IF b$(j TO j+LEN s$-5)<>s$(
5 TO ) THEN  GO TO 3060
3040 LET c4=1
3050 RETURN
3060 NEXT j
3070 NEXT h
3080 LET c4=0
3090 RETURN
3100 REM ************************
3110 REM SCHUIF BESTAND INEEN
3120 REM ************************
3130 LET c=FN a()
3140 LET shift=1000
3150 LET c1=c
3160 LET c3=c
3170 FOR i=1 TO x
3180 LET c1=c1+CODE b$(c1)
3190 NEXT i
3200 LET c2=c1-c
3210 FOR i=c1 TO LEN b$-1 STEP s
hift
3220 IF LEN b$-i+1<shift THEN  L
ET shift=LEN b$-i+1
3230 LET s$=b$(i TO i+shift-1)
3240 LET b$(c TO c+shift-1)=s$
3250 LET c=c+shift
3260 NEXT i
3270 LET y$=y$(1 TO 2*(s-1))+y$(
2*(s+1)-1 TO )
3280 FOR i=1 TO n-1
3290 LET s=i
3300 LET c=FN a()
3310 IF c<=c3 THEN  GO TO 3350
3320 LET c=c-c2
3330 LET y$(2*i-1)=CHR$ INT (c/2
56)
3340 LET y$(2*i)=CHR$ (c-256*INT
 (c/256))
3350 NEXT i
3360 LET p=p-c2
3370 LET n=n-1
3380 RETURN
```

```
3390 FOR i=1 TO 20: PRINT CODE y
$(i): NEXT i
3500 CLS
3510 PRINT "UNIFILE - MICRODRIVE
-ROUTINES"
3520 PRINT '"  1) SAVE "; INVERS
E 1;f$
3530 PRINT '"  2) LOAD nieuwe ge
gevens"
3540 PRINT '"  3) CAT van cartri
dge"
3545 PRINT '"  4) TERUG naar hoo
fdmenu"
3550 PAUSE 0: LET z$=INKEY$
3560 IF z$="1" THEN  GO SUB 3900
3570 IF z$="2" THEN  GO TO 3800
3580 IF z$="3" THEN  GO SUB 3700
3590 IF z$="4" THEN  GO TO 1000
3600 GO TO 3500
3670 REM ************************
3680 REM CAT ROUTINE
3690 REM ************************
3700 CLS : PRINT "CARTRIDGE CAT:
"
3710 INPUT "DRIVE NUMMER? (0 tot
 exit)";d
3720 IF d=0 OR d>8 THEN  RETURN
3730 CAT d
3740 PRINT #0;AT 0,0; FLASH 1;"
Druk op een willekeurige toets "
;
3750 PAUSE 0
3760 RETURN
3770 REM ************************
3780 REM LOAD ROUTINE
3790 REM ************************
3800 PRINT '' FLASH 1;"WAARSCHUW
ING:LADEN VAN NIEUWE GEGEVENS WI
ST HUIDIGE GEGEVENS"
3805 PRINT '"Druk op C om verder
 te gaan, of een andere toets om
 te stoppen."
3810 PAUSE 0: LET z$=INKEY$: IF
z$<>"C" AND z$<>"c" THEN  GO TO
3500
3820 GO SUB 4200
3830 INPUT "DRIVENUMMER (1-8)?";
d
3840 IF d<0 OR d>8 THEN  GO TO 3
830
3850 LOAD *"M";d;f$+CHR$ 128
3860 GO TO 1000
3870 REM ************************
3880 REM SAVE ROUTINE
3890 REM ************************
```

```
3900 PRINT '''(BESTAANDE NAAM=";
f$;")"
3905 GO SUB 4200
3910 INPUT "DRIVENUMMER? ";d
3920 IF d<0 OR d>8 THEN  RETURN
3930 CLS
3940 PRINT "   UNIFILE - ";f$
3950 PRINT AT 15,0;"Wilt u ";f$'
"op drive ";d;" wissen ? (J/N)"
3960 INPUT  LINE z$
3965 PRINT AT 15,0;,,,,
3970 IF z$<>"J" AND z$<>"j" THEN
  GO TO 4000
3980 ERASE "M";d;f$+CHR$ 128
3990 PRINT '"BESTAND GEWIST"
4000 SAVE *"M";d;f$+CHR$ 128 LIN
E 1000
4010 PRINT "GEGEVENS WEGGESCHREV
EN - VERIFY GESTART"
4020 VERIFY *"M";d;f$+CHR$ 128 L
INE 1000
4030 RETURN
4170 REM ************************
4180 REM VOER BESTANDSNAAM OP
4190 REM ************************
4200 INPUT "BESTANDSNAAM ?"; LIN
E f$
4210 IF LEN f$<10 AND LEN f$>0 T
HEN  RETURN
4220 INPUT "(MAX LENGTE 9) "; LI
NE f$
4230 GO TO 4210
4270 REM ************************
4280 REM PRINTER UITVOER
4290 REM ************************
4300 LPRINT "ENTRY ";s-1;":-"
4310 FOR i=1 TO x
4320 LPRINT a$(i,2 TO CODE a$(i,
1));":";
4330 LPRINT FN a$()(2 TO )
4340 LET c=c+CODE b$(c)
4350 NEXT i
4360 GO TO 2570
```

Als u klaar bent met het inbrengen van de regels, tikt u het volgende in:

```
CLEAR
LET F$=""
```

en SAVE het programma vervolgens met LINE 1000. Om het programma voor de eerste keer te starten, doet u een `GO TO 1` – gebruik nooit RUN, omdat dan alle gegevens worden gewist. Ieder gegevensbestand kan maximaal 28000 tekens bevatten, die elk bestaan uit een aantal onderdelen, welke worden opgezet wanneer het bestand wordt aangemaakt. Een voorbeeld zou kunnen zijn een adressenboek, bestaande uit drie onderdelen – naam, adres en telefoonnummer.

Mogelijkheid 1 verzorgt de aanmaak van een nieuw bestand en vraagt een naam voor het bestand op te geven.
Mogelijkheid 2 stelt u in staat om gegevens in het bestand in te voeren.
Mogelijkheid 3 is de meest nuttige toepassing – deze stelt u in staat het gehele bestand te doorzoeken, om vervolgens de gevonden mogelijkheden af te drukken. Deze speciale zoekmogelijkheid doorzoekt ieder gebied van het bestand op de aanwezigheid van de opgegeven string, maar is niet erg snel. Bijvoorbeeld, als u heeft opgegeven **SSS-KENT** met het adresboekbestand, dan wordt iedere bewoner van KENT, maar ook iedereen met de naam KENT, afgedrukt. Een snellere mogelijkheid biedt de normale zoekroutine, die alleen het eerste teken controleert.
Mogelijkheid 4 verzorgt het wegschrijven van de gegevens naar een cassette voor back-up doeleinden.
Mogelijkheid 5 kan de huidige gegevens bewaren, of andere gegevens laden, of een CAT uitvoeren voor een cartridge. Bestanden worden op de cartridge bewaard, voorzien van een CHR$ 128 aan het einde ervan, om ze te kunnen onderscheiden van de gewone programmabestanden. Dit extra teken wordt niet opgemerkt door de CAT-functie, want deze drukt hiervoor een spatie af.
Ik zal geen poging doen de werking van het programma uit te leggen – deze wordt volledig gedekt door het boek van David Lawrence. Ik zal echter wel uitleggen, hoe de printerroutine werkt, opdat gebruikers deze naar eigen wens kunnen aanpassen.
De routine bevindt zich tussen de regels 4300-4350. **S-1** is de positie binnen het bestand en **X** is het aantal onderdelen. Regel 4320 drukt de titel van elk onderdeel af. Regel 4330 drukt de eigenlijke gegevens af. Regel 4340 verhoogt een pointer. Bijvoorbeeld, als uw bestand was opgezet zoals het hierboven genoemde adresboek en u wilt de gegevens afdrukken in de vorm van adresetiketten, zou u dat als volgt kunnen doen:

```
4300 LPRINT "Ref.nr.";S-1
4310 LPRINT FN A$ () (2 TO): REM de naam
4320 LET C=C-CODE B$(C)
4330 LPRINT FN A$ () (2 TO): REM het adres
4340 LPRINT
4350 LET C=C-CODE B$(C)
4360 LPRINT "Telefoon:"; FN A$() (2 TO):REM het
     nummer
4370 LET C=C-CODE B$(C)
4380 GO TO 2570
```

Verdere aanvullingen

Er zijn nog veel meer mogelijkheden die kunnen worden toegevoegd. Een daarvan is, een CAT te maken van alleen unifile-bestanden, door gebruik te maken van het stream14-z$-programma, zodat de variabele z$ kan worden gecontroleerd op de aanwezigheid van CHR$ 128. Een andere verbetering zou kunnen zijn, de regels 3130-3380 te converteren naar machinetaal, hetgeen de snelheid van het schrappen of muteren van gegevens aanzienlijk vergroot.
16K-versie: Een beperkte unifile-toepassing kan worden gebruikt op een 16K-Spectrum met behulp van de volgende wijzigingen: Verwijder alle REM-regels en de regels 3540, 3580 en 3700-3760. Verander de regels:

```
1350 DIM B$(1000)
3140 LET SHIFT=100
3375 LET S$=" "
```

5. Programmabeveiliging

Wanneer u een zeer goed programma heeft geschreven en dit wilt verkopen, dan wilt u zeker niet dat iedere Piet of Klaas in staat is net zoveel kopieën van uw meesterwerk te maken als zij willen, om aan hun vrienden te geven. Ondanks het feit dat geen enkel programma 100% is te beveiligen, zijn er toch veel eenvoudige technieken, om kopiëren zeer moeilijk te maken. Er is een aantal ingebouwde mogelijkheden die hier worden behandeld, evenals een aantal andere, nog slimmere methodes. Het moet nog eens worden gezegd, dat geen enkele methode onverslaanbaar is – een zelfverzekerde machinetaalprogrammeur zal altijd in staat zijn de beveiligingen te passeren van een programma, afhankelijk van de tijd en de kennis.

5.1 Automatische uitvoering

De eenvoudigste methode om het kopiëren te bemoeilijken, is het programma automatisch te starten, met behulp van SAVE * ...LINE. Het kan dan niet worden geMERGEd, dus zal altijd direct na het laden met de uitvoering beginnen. Als u een programma de naam "run" (kleine letters!) geeft, dan zal, na het aanschakelen van de Spectrum of na een NEW, het commando RUN dit programma automatisch laden en starten (indien het was opgeslagen met LINE).
Het is ook erg handig om programmanamen te laten beginnen met CHR$ 0, zodat ze niet verschijnen in het CAT-overzicht. Wanneer u het "run"-programma het onzichtbare programma laat laden, dan zal de potentiële kopieerder niet in staat zijn de naam van het hoofdprogramma te weten te komen. Een andere methode is de naam van het programma van alleen maar spaties te voorzien – de meeste gebruikers zal dit niet opvallen.

Crash POKE

Wat voorkomen dient te worden, is het BREAKen van een programma zodra dit is gestart, of tijdens een LOAD. In bovenstaande situatie zou dit de naam van het onzichtbare bestand beschikbaar maken, door het bestuderen van de programmalijst.
Een manier om dit te bereiken, zij het niet subtiel maar wel erg mooi, is door de eerste regel als volgt te laten zijn:

```
POKE 23659,0
```

Dit leidt ertoe dat de Spectrum wordt misleid, zodat hij denkt dat er geen regels in het onderste gedeelte van het scherm aanwezig zijn. Zodra foutberichten verschijnen (bijvoorbeeld `Break into program`) zal het systeem gaan "hangen", omdat nergens een foutbericht kan worden afgedrukt. De gebruiker zal de stekker uit de computer moeten trekken, om het systeem terug te zetten. Zodoende wordt voorkomen dat de programmalijst kan worden onderzocht of gekopieerd.
Let goed op bij het gebruik van deze POKE: een CLS-commando, of iedere andere poging die een bericht op de onderste regels van het scherm tot gevolg heeft (bijv. INPUT) zal het systeem laten "hangen". De POKE is eigenlijk alleen nuttig voor machinetaal-programma's gedurende het laden ervan – voor BASIC-programma's is de methode te beperkt.

Een alternatief verkrijgt men met de volgende regel:

```
LET p=PEEK 23613+256*PEEK 23614:POKE p,0:POKE
p+1,0
```

die ook het systeem opknoopt zodra een foutbericht verschijnt, maar toch afdrukken op de onderste regels van het scherm toestaat.

5.2 On Error GOTO

Verschillende computers beschikken over een On Error GOTO functie, die een sprong forceert naar een bepaalde regel, in geval van een fout. Helaas ontbreekt een dergelijke functie op de Spectrum, maar de volgende machinecoderoutine biedt deze functie:

ON ERROR GOTO programmalijst

```
8794 REM ************************
8795 REM * ON ERROR GOTO        *
8796 REM ************************
8797 REM err=start adres, line=f
outsprong
8798 REM aanbevolen:64780/32010
8800 RESTORE 8900: LET c=0
8810 FOR i=err TO err+170
8820 READ a: LET c=c+a
8830 IF a<256 THEN  POKE i,a
8840 NEXT i
8850 IF c<>17439 THEN  PRINT "Co
ntrolesom fout": STOP
8860 POKE err+149,line-256*INT (
line/256)
8870 POKE err+150,INT (line/256)
8875 RETURN
8880 RANDOMIZE USR err
8885 POKE 23734,0: RETURN
8890 RANDOMIZE USR err
8895 POKE 23734,4: RETURN
8900 DATA 33,17,0,9,235,42,61,92
8905 DATA 115,35,114,207,49,1,0,
0
8910 DATA 201,34,201,92,42,61,92
,17
8915 DATA 3,19,213,205,176,22,25
3,203
8920 DATA 55,174,205,110,13,33,1
85,23
8925 DATA 34,237,92,58,58,92,50,
211
8930 DATA 92,245,207,50,33,129,0
,237
8935 DATA 91,201,92,241,167,237,
82,32
8940 DATA 20,253,203,124,70,32,1
4,254
```

```
8945 DATA 7,40,10,62,100,50,211,
92
8950 DATA 62,63,215,24,21,60,71,
254
8955 DATA 10,56,2,198,7,205,239,
21
8960 DATA 62,32,215,120,17,145,1
9,205
8965 DATA 10,12,175,17,54,21,205
,10
8970 DATA 12,237,75,69,92,237,67
,201
8975 DATA 92,205,27,26,62,58,215
,253
8980 DATA 78,13,6,0,205,27,26,33
8985 DATA 59,92,203,174,251,203,
110,40
8990 DATA 252,33,66,92,17,260,27
0,115
8994 DATA 35,114,35,54,1,253,54,
0
8995 DATA 255,253,54,124,0,205,1
10,13
8996 DATA 195,125,27
```

Het programma is 171 bytes groot en onafhankelijk van de positie in het geheugen. Zet eerst de variabele **err** op de gewenste geheugenlocatie en **line** op het regelnummer waar in geval van een fout naar moet worden gesprongen. Voor de gewone opdrachten (niet-interface-opdrachten), wordt de routine geactiveerd via een GO SUB 8880. Zodra een fout optreedt, zal de daarbij behorende foutboodschap normaal worden afgedrukt. Vervolgens wordt gewacht op het indrukken van een toets, waarna het scherm wordt schoongemaakt en een sprongopdracht wordt uitgevoerd naar het aangegeven regelnummer. De routine wordt gedeactiveerd na het optreden van de fout. Het regelnummer waarin de fout ontstond, kan men lezen door uit te voeren:

```
PEEK 23753 + 256*PEEK 23754
```

het foutnummer verkrijgt men door

```
PEEK 23763
```

Voor interface-opdrachten is een GO SUB 8890 nodig om de routine te activeren. Let op dat zodra een interfacefout optreedt (een zonder foutcode) de desbetreffende boodschap niet wordt afgedrukt – er wordt daarvoor in de plaats een vraagteken afgedrukt en positie 23763 zal 100 bevatten.
Na een willekeurige interface-opdracht, is de routine gedeactiveerd tengevolge van die opdracht.
Wilt u zelf de routine uitschakelen, doe dan

```
LET p=PEEK 23613 + 256*PEEK 23614: POKE p,3: POKE
p+1,19: POKE 23734,0
```

Deze routine is niet te gebruiken samen met de On Error GOTO routine uit het vorige hoofdstuk. Om op EOF te controleren zodra een fout optreedt, moet PEEK 23763 een 7 bevatten in geval van EOF.
Gedurende het testen, moet u oppassen voor oneindige lussen. Zou u regel 100 hebben gekozen als de foutregel en deze zou bevatten:

```
100 GO SUB 8880
```

dan zou u niet meer in staat zijn om het programma te BREAKen. Plaats daarom altijd de POKE's om de routine te deactiveren in uw programma.
Om commerciële programma's van de routine te voorzien, doet u er goed aan de volgende regel toe te voegen:

```
8855 POKE err +23,94:POKE err+24,35: POKE
err+25,86
```

Deze POKE's stellen de routine zo bij, dat deze niet wordt uitgeschakeld zodra een fout optreedt.
Wilt u gebruik maken van deze routine, of van andere machinetaalroutines in dit boek, in een commercieel programma, dan kan dit **ALLEEN** met de toestemming van de auteur, voordat ze op de markt worden gebracht, omdat er sprake is van copyright.
Zoals u heeft gezien, kunnen machinecode en een paar POKE's zeer effectief worden toegepast om programma's te beveiligen, maar het tegenovergestelde is ook waar – machinecode kan ook worden gebruikt om programma's te kraken, door de beveiligingen te doorkruisen.
Om een beveiliging te kraken, moet de programmeur weten welke methode is toegepast. Dit is, waarom het voor een vastberaden machinetaalcodeur relatief eenvoudig is om bijvoorbeeld een CAT te verkrijgen van onzichtbare bestanden, of om autostart programma's toch te MERGEn. Het is een ongelukkig bijprodukt van de overvloedige beschikbaarheid van technische gegevens over alle Sinclair-produkten.

6. Het gebruik van de RS232-interface

Een andere mogelijkheid die de interface I biedt, is het kunnen verzenden en ontvangen van gegevens via de RS232. De RS232 is een (bijna) algemene standaard voor seriële verzending en er bestaat een scala van randapparaten, evenals andere computers, die kunnen communiceren via RS232. Voor het gebruik ervan met de Spectrum, is de meest gangbare toepassing, die van het aansturen van normale-breedte, normaal papier printers, voor het produceren van programmalijsten en mooie programma-uitvoer.
De snelheid waarmee gegevens worden verstuurd en ontvangen, wordt uitgedrukt in *BAUD*-snelheid en komt vrijwel overeen met tien keer de verzonden hoeveelheid bytes per seconde. Er bestaan negen verschillende standaarden BAUD-snelheden, namelijk 50,110,300,600,1200,2400,4800,9600 en 19200. Telexverkeer maakt gebruik van een langzame snelheid van 110 of 300 BAUD, doch met CRT (Cathode Ray Tube) beeldschermen worden snelheden toegepast van 2400 BAUD en hoger.
Evenals de snelheden variëren, bestaan er verschillende formaten waarmee gegevens tussen verschillende apparaten kunnen worden uitgewisseld. Op de Spectrum is het formaat vast nl.:

- geen pariteit;
- acht gegevensbits;
- een stopbit.

Om gebruik te maken van de RS232-mogelijkheid op de Spectrum, zijn nog twee channels extra nodig – de "t"- en "b"-channels. Deze streams verschillen van uitvoering in de manier waarop ze de tekens behandelen.

6.1 Tekst channel "t"

Het "t"-channel is het meest geschikt voor gebruik met printers, omdat dit verschillende dingen doet met elk teken. Met verwijzing naar de karakterset in Appendix A van het Spectrum handboek, drukt het "t"-channel het volgende af:

0– 12 (controletekens) worden genegeerd
13 (newline) zendt een einde-regel-teken (13) en vervolgens een nieuwe-regel-teken (11)
14– 31 (controletekens) worden genegeerd
32–127 (ASCII-tekens) worden onveranderd verstuurd
128–164 (Graphics-tekens) worden verstuurd als "?"
165–255 (instructiesymbolen) worden verstuurd als gedecodeerde enkele tekens

Deze tekenconversie is ideaal voor het sturen van programmalijsten en programma-uitvoer, doch veel printers maken gebruik van de controletekens onder de 32, om speciale effecten te bereiken – bijvoorbeeld, de Epson-serie maakt gebruik van CHR$ 14 om vergrote tekst af te drukken, maar dergelijke tekens kunnen niet worden verstuurd via het "t"-channel. Daarbij komt nog, dat de buitengewoon handige TAB-functie wordt genegeerd, hetgeen mooie programma-uitvoer een stuk moeilijker maakt. Een alternatieve methode biedt het "b"-channel.

6.2 Binair channel "b"

Het binaire channel is geschikt voor gebruik met apparaten anders dan printers, omdat geen tekenconversie wordt gedaan. Elk teken wordt onveranderd verstuurd en daardoor is het ideaal voor de verzending van ruwe gegevens en programma's. Het wordt ook gebruikt indien speciale controletekens moeten worden verstuurd naar printers, omdat deze niet kunnen worden verstuurd met het "t"-channel.
Voordat uw Spectrum gegevens kan versturen of ontvangen via de RS232, moet bekend worden gemaakt met welke BAUD-snelheid dit gaat gebeuren. Dit kan worden gedaan met behulp van het FORMAT "b"-commando, gevolgd door de BAUD-snelheid. Om bijvoorbeeld een BAUD-snelheid van 110 te bereiken, doet u

```
FORMAT "b";110
```

(U kunt ook opgeven FORMAT "t";110 – zij hebben hetzelfde effect.) Indien u een onbekende BAUD-snelheid opgeeft, krijgt u geen foutboodschap, doch neemt de Spectrum de erop volgende hogere BAUD-snelheid aan. Om de op een bepaald moment toegepaste snelheid te weten te komen, doet u

```
PRINT INT (350000/(( PEEK 23747 + 256* PEEK 23748
+2)*26))
```

(Voor de hoogste snelheden kan dit enkele BAUDs afwijken.) Wordt geen BAUD-snelheid opgegeven, dan heeft het systeem daarvan niets in de gaten, zodat vreemde verplaatssnelheden kunnen optreden. De initiële waarde van de BAUD-snelheid staat op 9600.

6.3 Aansturen van een RS232-printer

Vóór een RS232-printer voor de eerste keer aan de Spectrum wordt aangesloten moet deze worden opgezet om te voldoen aan het gegevensformaat van de Spectrum.
Gewoonlijk wordt dit bereikt door het aanpassen van een aantal interne schakelaars van de printer en dit behoeft slechts eenmaal te worden uitgevoerd. De hoogst mogelijke BAUD-snelheid moet worden opgegeven (schrijf dit op voordat u het vergeet!) en het formaat moet zijn resp. geen pariteit, acht tekenbits en een stopbit. Biedt de printer de mogelijkheid van een automatische regelvoeding na een einde-regel-teken (auto CR na LF in jargon) dan moet deze mogelijkheid onklaar worden gemaakt, omdat de Spectrum hier reeds in voorziet.
Zijn alle benodigde mogelijkheden gekozen, dan kan de printer worden aangesloten via een passende kabel aan de 9-pin D-type stekker van de interface.
Maakt u gebruik van een RS232-printer, dan is het aan te bevelen om zowel het "t"-channel als het "b"-channel tegelijkertijd open te hebben – het eerste ten behoeve van tekst en lijsten en het laatste voor controletekens. Het verdient de voorkeur om stream 3 te gebruiken voor printer "t"-channels, om hierdoor gebruik te kunnen maken van LLIST en LPRINT in bestaande programma's bijvoorbeeld:

```
10 FORMAT "t"; 1200: REM BAUD-snelheid
20 OPEN #3;"t": REM tekst via stream 3
30 OPEN #15;"b"; REM binair via stream 15
40 LLIST : REM lijst via channel "t"
```

```
50 LPRINT "Normale grootte"
60 PRINT #15; CHR$ 14; #3;"Dubbele grootte"
70 CLOSE #3: CLOSE #15
```

In bovenstaand voorbeeld wordt stream 15 gebruikt als een "b"-channel voor het sturen van controletekens en wordt stream 3 gebruikt als een "t"-channel voor tekst en lijsten.
Gedurende het versturen via de RS232 wordt de rand van het beeldscherm zwart, als het ontvangende apparaat "actief" is, en dus geen gegevens kan ontvangen. De Spectrum wacht, totdat het apparaat gereed is voor ontvangst of totdat de BREAK-toets wordt ingedrukt. Als een RS232-stream gesloten is, wordt een teken 13 verstuurd naar de lege printerbuffers of soortgelijke buffers. Een teken 13 wordt niet verstuurd indien het CLEAR # wordt gebruikt.
Het is niet mogelijk om meerdere RS232-apparaten aangesloten te hebben, ondanks het feit dat verschillende streams kunnen worden aangesloten op hetzelfde channel. Het is ook niet mogelijk om met verschillende BAUD-snelheden te werken op verschillende streams of channels.
Het enige hoofdprobleem dat blijft wanneer gebruik wordt gemaakt van RS232-printers, is het gebrek van het TAB-commando. Het volgende machinetaalprogramma opent stream 3 als een "t"-channel, maar met een kleine aanpassing. Het telt de hoeveelheid te versturen tekens en maakt daarom het gebruik van de TAB-functie volledig mogelijk. Je kunt je afvragen, waarom een dergelijke aanpassing niet is meegenomen in de oorspronkelijke machine. Ook wordt de fout van dubbele spaties verholpen, die ontstonden in programmalijsten (bijv. THEN PRINT). De routine is zodanig geschreven, dat hij zich bevindt in de ZX-printer buffer van het geheugen, die nooit wordt gebruikt als stream 3 is hergedefinieerd. Let op dat noch een CLOSE #3 noch een CLEAR # stream 3 toekent aan de ZX-printer en dat ook geen nieuwe-regel-teken wordt verstuurd.

TAB-routine

```
8997 REM ************************
8998 REM *  RS232 TAB           *
8999 REM ************************
9000 RESTORE 9100: LET c=0
9010 FOR i=23296 TO 23467
9020 READ a: LET c=c+a
9030 POKE i,a
9040 NEXT i
9050 POKE 23540,80: REM printer
breedte
9055 IF c<>19420 THEN  PRINT "Co
ntrolesom fout": STOP
9060 RANDOMIZE USR 23296
9070 RETURN
9100 DATA 42,79,92,1,15,0,9,17
9110 DATA 23,91,115,35,114,1,0,0
9120 DATA 33,245,91,112,35,112,2
01,254
9130 DATA 32,48,93,254,13,32,26,
33
9140 DATA 246,91,203,70,203,134,
192,33
```

```
9145 DATA 246,91,203,134,43,54,0
,62
9150 DATA 13,205,169,91,62,10,19
5,169
9160 DATA 91,254,23,63,192,17,71
,91
9170 DATA 42,81,92,115,35,114,20
1,50
9180 DATA 15,92,17,79,91,24,241,
17
9190 DATA 23,91,205,64,91,58,15,
92
9200 DATA 87,33,244,91,150,210,1
08,4
9210 DATA 35,122,150,213,220,39,
91,209
9220 DATA 122,253,150,187,200,71
,62,32
9230 DATA 197,217,215,217,193,16
,247,201
9240 DATA 254,165,56,5,214,165,1
95,16
9250 DATA 12,253,203,188,134,33,
59,92
9260 DATA 203,134,254,32,32,2,20
3,198
9270 DATA 254,128,56,2,62,63,205
,169
9280 DATA 91,33,245,91,52,126,43
,190
9290 DATA 192,205,39,91,253,203,
188,198
9300 DATA 201,207,30,201
```

6.4 Beeldscherm afdrukken

Een andere zinnige toepassing bij een brede printer is een afdruk te maken van het beeldscherm. In zijn eenvoudigste vorm bestaat dit uit het volgende kleine programma dat werkt voor iedere printer (met ten minste 32 tekens per regel) en dat ieder teken leest en naar de printer verstuurt.

```
100 FORMAT "T"; BAUD: REM passende waarde
110 OPEN #4;"T"
120 FOR Y=0 TO 21
130 FOR X = TO 31
140 LET A$=SCREEN$(Y,X)
150 IF A$= "THEN LET A $="
160 PRINT #4;A$;
170 NEXT X
180 PRINT #4
190 NEXT Y
```

Het programma doorzoekt de tekenposities op het beeldscherm en drukt deze af, of, indien onherkenbaar, drukt een spatie af (regel 150). Na iedere regel van 32 tekens

wordt een nieuwe-regel-teken verstuurd (regel 180). Ondanks het feit dat dit een grove manier is, kan het nog steeds een zeer handige subroutine zijn, vooral omdat het een printer-onafhankelijke routine is.
Wilt u een afdruk van een high resolution scherm, of van een programmalijst met inverse en grafische tekens, hoe kunt u dit dan bereiken?
Beschikt u over een passende printer, dan kunt u daarmee getrouwe kopieën maken, zoals met het COPY-commando bij de ZX-printer. In BASIC is dit echter een trage aangelegenheid, maar de resultaten zijn het wel waard. Ik bied hier twee versies – een voor de Epson-printers (met een RS232-interface), en een voor de Seikosha GP100.

EPSON Hi-Res afdruk-routine

```
1000 FORMAT "b";1200: REM aanpas
sen voor interface
1010 OPEN #3;"b"
1020 LPRINT CHR$ 27;"A";CHR$ 8;
1030 FOR y=175 TO 0 STEP -8
1040 LPRINT CHR$ 27;"K";CHR$ 0;C
HR$ 1;
1050 FOR x=0 TO 255
1060 LPRINT CHR$ (128*POINT (x,y
)+64*POINT (x,y-1)+32*POINT (x,y
-2)+16.*POINT (x,y-3)+8*POINT (x
,y-4)+4*POINT (x,y-5)+2*POINT (x
,y-6)+POINT (x,y-7));
1070 NEXT x
1080 LPRINT CHR$ 13;CHR$ 10;
1090 NEXT y
1100 LPRINT CHR$ 27;"A";CHR$ 12
```

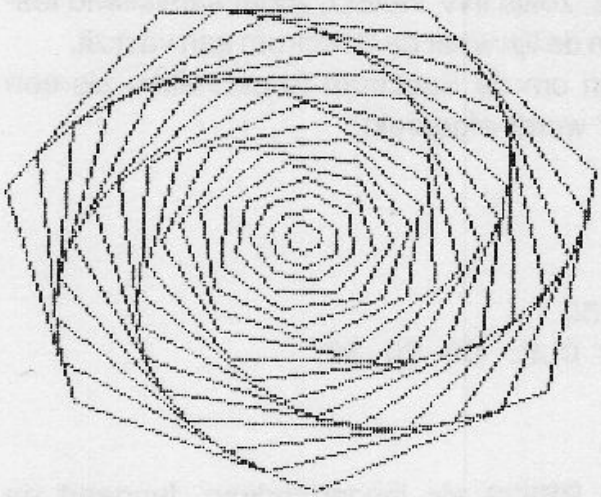
SEIKOSHA Hi-Res afdruk-routine

```
1000 FORMAT "b";1200: REM aanpas
sen voor interface
1010 OPEN #3;"b"
1020 LPRINT CHR$ 18
1030 FOR y=174 TO 0 STEP -7
1050 FOR x=0 TO 255
1060 LPRINT CHR$ (POINT (x,y)+2*
POINT (x,y-1)+4*POINT (x,y-2)+8*
POINT (x,y-3)+16*POINT (x,y-4)+3
2*POINT (x,y-5)+64*POINT (x,y-6)
+128);
1070 NEXT x
1080 LPRINT CHR$ 13;CHR$ 10;
1090 NEXT y
1100 LPRINT CHR$ 30
1500 REM ************************
1501 REM ************************
```

Beide printers maken gebruik van het POINT-commando om een binair getal te berekenen dat wordt verstuurd naar de printer. Merk op dat het ''b''-channel wordt gebruikt, omdat geen tekenconversie mag optreden.

Het onderstaande voorbeeld toont de uitvoer van de Epson-routine.

Voorbeeld van een EPSON afdruk



Andere toepassingen van de RS232

Channel "b" kan ook worden gebruikt voor het versturen van programma's, gegevens, machinecode en beeldschermen, op een vergelijkbare wijze als bij "m"-channels, doch een bestandsnaam is niet nodig. Dit kan erg handig zijn bij het versturen van programma's over de telefoonlijn, bijvoorbeeld, wanneer men over een RS232-modem beschikt.
Om met de relevante hardware een programma via een 300 BAUD modem naar een vriend aan het andere eind van de lijn te sturen, moet hij eerst intikken

```
FORMAT "b";300: LOAD *"b"
```

zodat de Spectrum wordt geïnstalleerd om op het programma te wachten. U zult dan moeten intikken

```
FORMAT "b";300: SAVE *"b"
```

Channel "t" moet alleen worden gebruikt voor de versturing van lijsten of tekst. De commando's LOAD *, SAVE *, MERGE * en VERIFY * kunnen allen worden gebruikt met de RS232, evenals OPEN # en CLOSE #. Bestandsnamen en daarop volgende nummers, zijn niet benodigd bij RS232-channels, ook al worden deze wel geaccepteerd door de Spectrum, die ze verder negeert. Bijv., SAVE *"b";5;"naam" wordt geaccepteerd, maar de extra details worden genegeerd.
Een andere toepassing kan zijn, de Spectrum te beschouwen als een RS232-eindstation, dat ingetoetste tekens verstuurt naar een andere computer:

```
10 FORMAT "b";BAUD
20 OPEN #4;"b"
30 PAUSE 0:LETa$=INKEY$
40 IF a$>CHR$127 THEN GO TO 30
50 IF a$=CHR$6 THEN POKE 23658,8-PEEK 23658: GO TO
30
60 PRINT #4;a$
70 GO TO 30
```

De routine tast het toetsenbord af (regel 30), negeert commandosymbolen (bijv.

STOP) die niet standaard zijn (40) en gebruikt CAPS LOCK om van C en L mode te verwisselen (regel 50). Als alles in orde is, worden de gegevens verzonden naar stream 4, dit is de ontvangende computer. Sommige computers kunnen vreemd reageren op enkele van de Spectrum-tekens, zoals INV VIDEO, zodat aanvullend testen noodzakelijk kan zijn aan het einde van de lijn waar de Spectrum aan vast zit.
Een verdere toepassing zou kunnen zijn om de Spectrum te gebruiken als een beeldscherm, waarop alle RS232-invoer wordt afgedrukt:

```
10 FORMAT "b";BAUD
20 OPEN #4;"b"
30 LET a$=INKEY$ #4
40 IFa$="" THEN GOTO 30
50 IF a$=CHR$ 12 THEN CLS: GO TO 30
60 PRINT a$
70 GO TO 30
```

Indien gebruik wordt gemaakt van de RS232 als invoermedium, fungeert de SPACE-toets als **BREAK**functie – CAPS SHIFT is niet nodig.
Het is mogelijk om twee Spectrums te koppelen via de RS232 en een daarvoor geschikte kabel, maar er zijn enkele voordelen indien gebruik wordt gemaakt van het netwerk, dat eenvoudiger te bewerkstelligen is, omdat het gelijksoortige activiteiten sneller doet en de RSR232-poort vrij laat voor andere randapparaten.

RS232-aanvullingen voor het streamoverzicht

De volgende regels zouden moeten worden toegevoegd aan het streamoverzichtprogramma, om ook RS232 aan te kunnen:

Streamoverzicht programmalijst-aanvullingen

```
1502 REM ************************
1640 IF F$="T" OR F$="B" THEN  G
O TO 3000
2999 REM channel T/B
3000 PRINT  INVERSE 1;"RS232  ";
3010 IF FN P(D+5)=3130 OR FN P(D
+5)=3132 THEN  PRINT "Tekst": GO
 TO 3040
3020 IF FN P(D+5)=3335 OR FN P(D
+5)=3162 THEN  PRINT "Binair": G
O TO 3040
3030 PRINT "(Onbekend)"
3040 GO SUB 1800
3050 PRINT "Baud waarde  :";INT
(3500000/((FN P(23747)+2)*26));"
 (ongev.)"
3060 GO TO 1500
```

Beide typen RS232-channels hebben dezelfde identificatie in het geheugen – "T", zodat regels 3010 en 3020 uitzoeken of dit tekst of binair is. De BAUD-snelheid wordt ook getoond.

7. Gebruik van het netwerk

Onder een netwerk wordt verstaan, een methode waarbij computers aan elkaar worden gekoppeld, zodat zij onderling zeer snel kunnen communiceren. Bij de Spectrum is sprake van een snelheid van boven de 3K bytes per seconde, hetgeen aanzienlijk meer is dan de maximale BAUD-snelheid van 19200. Er kunnen hooguit 64 Spectrums aan elkaar worden gekoppeld en er bestaan verschillende toepassingen van deze mogelijkheid – in de eerste plaats kunnen ze worden opgesteld in een klaslokaal, zodanig dat ze gezamenlijk gebruik kunnen maken van de randapparatuur, zoals printers en microdrives. In de tweede plaats kunnen de Spectrums van bijv. twee vrienden zodanig worden gekoppeld, dat onderling snel gegevens en programma's kunnen worden uitgewisseld.
Een andere toepassing wordt gebruikt bij het ontwikkelen van programma's, met name machinetaalprogramma's. Een Spectrum bevat dan de assembler en de toolkit-programma's en schiet dan een ontwikkeld programma door naar een andere Spectrum, waar het vervolgens kan worden uitgetest. Het voordeel hiervan is, dat als het machinetaalprogramma stuk loopt, snel een aangepaste versie geladen kan worden van de eerste Spectrum.
Om een netwerk te gebruiken zijn streams en channels nodig, met de identificatie "N" (of "n"), die staat voor netwerk. Voor u hiervan gebruik wilt maken, dient u uw Spectrum te voorzien van een nummer (een *stations*nummer). Dit bereikt u met het volgende commando

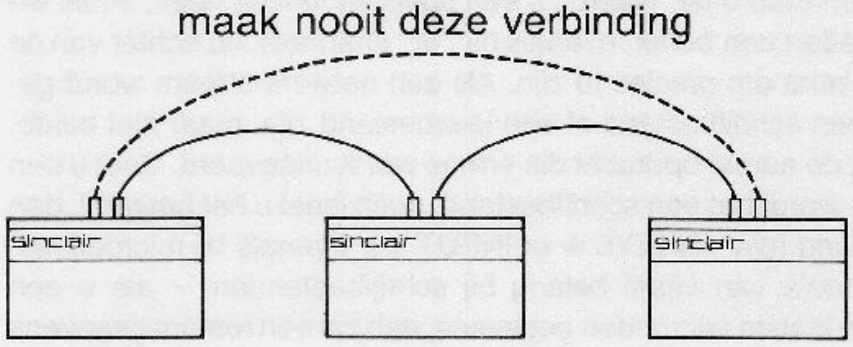
```
FORMAT "n";t
```

waarin "t" het gewenste stationsnummer voorstelt, van 1 tot 64. Als de Spectrum wordt aangeschakeld, dan wordt deze waarde op 1 gezet. Indien slechts twee Spectrums binnen een netwerk aanwezig zijn, is het FORMAT-commando overbodig, omdat ze beide nummer 1 mogen hebben. Weet u niet meer welk nummer u heeft toegekend aan uw Spectrum, dan kunt u dit aan de weet komen met

```
PEEK 23749
```

Om de Spectrums met elkaar te verbinden, gebruikt u de bij de interface meegeleverde kabels, welke u in de aansluiting steekt onder de cassette-aansluiting, zo ontstaat een kettingvorm. Er moet altijd een aansluiting open blijven, omdat er geen lus mag ontstaan. De volgende afbeelding toont de opstelling in geval van drie stations.

Een netwerk-voorbeeld



7.1. Het verzenden van programma's via het netwerk

Evenals bij de microdrives en de RS232, kunnen programma's en gegevens worden verzonden via het netwerk, door gebruik te maken van commando's die bijna gelijk zijn aan die, welke gebruikt worden bij de RS232, maar nu met de "N"-identificatie, gevolgd door een stationsnummer. Als u bijvoorbeeld bij station 10 zit, en uw vriend, die graag een programma van u wil ontvangen, bij station 20, zou u moeten intikken

```
SAVE *"n";20
```

waarna de rand van het beeldscherm zwart wordt en wacht op uw vriend, die vervolgens het programma dient te accepteren. Uw vriend moet dan intikken

```
LOAD *"n";10
```

Zodra hij dit doet zullen de randen van de beeldschermen van beide Spectrums flitsen, gedurende het transport van het programma. Had uw vriend als eerste LOAD ingetikt, dan zou zijn Spectrum gewacht hebben op het door u in te tikken SAVE-commando en ook dan zou de verzending prima zijn verlopen. Om bepaalde redenen die we later zullen bespreken, is het een goede gewoonte om de ontvanger als eerste LOAD te laten intikken en pas daarna de verzender SAVE.
Wil uw vriend zich ervan overtuigen dat de verzending goed is verlopen, dan kan hij intikken

```
VERIFY *"n";10
```

en u moet weer intikken

```
SAVE *"n";20
```

ook nu weer doet de volgorde er niet toe. Alle gebruikelijke functies kunnen worden toegepast, zoals

```
SAVE *"n";10 LINE 10
LOAD *"n";20 SCREEN$
```

7.2 Het versturen van gegevens via het netwerk

Om gegevens via het netwerk te versturen naar een ander station, worden streams en channel "n" gebruikt. Om een stream op te zetten, wordt het commando

```
OPEN #s;"n";x
```

gebruikt, waarin **s** het streamnummer voorstelt (van 0-15) en x het stationsnummer van de andere Spectrum (van 0-64, waarbij 0 een speciale functie heeft, zoals we later zullen zien). Dit creëert een *buffer,* evenals bij "M" channels, nu echter van de halve capaciteit-255 tekens om precies te zijn. Als een netwerk-stream wordt geopend, dan kan dit of een schrijfbestand of een leesbestand zijn, maar niet beide. Wat het is, bepaalt u bij de eerste opdracht die ermee wordt uitgevoerd. Doet u een PRINT#-opdracht, dan wordt het een schrijfbestand, doch leest u het bestand, dan wordt het een leesbestand (bijv. INKEY$ # of INPUT #). Evenals bij microdrives, zijn CLOSE#-commando's van vitaal belang bij schrijfbestanden – als u een CLOSE # vergeet na de laatste verzonden gegevens, dan zal een restant gegevens

(dat in de buffer zit) niet worden verstuurd, maar, en dit is veel erger, het ontvangende station zal voor altijd blijven wachten.
Gedurende de uitvoering van netwerkcommunicatie (als de rand van het beeldscherm flitst) is de CAPS SHIFT toets niet nodig om een **BREAK** te veroorzaken. Het intoetsen van SPACE is reeds voldoende hiervoor. Een Spectrum mag nooit worden uitgeschakeld gedurende de netwerkcommunicatie.

7.3 Station 0-uitzending

Gewoonlijk wordt gedurende het proces van versturen en ontvangen van gegevens, van een ander station het zgn. "handshaking" toegepast – dit betekent dat indien het ontvangende station niet gereed is, het versturende station blijft wachten tot dit wel het geval is, opdat geen gegevens verloren gaan. Echter, de interface biedt een zeer handige mogelijkheid –"*uitzending*", door gebruik te maken van stationnummer 0, dat zich niet stoort aan eerdergenoemde regels.
Zodra u gegevens verstuurt naar "station 0" worden deze tegelijkertijd naar alle aangesloten Spectrums uitgezonden, maar er wordt niet gewacht of ze gereed zijn. Indien ze wachten op gegevens van station 0, dan zullen ze deze ontvangen, zoniet dan gaan de gegevens verloren. Gedurende het verzenden van programma's via het netwerk, moet ieder ontvangend station het volgende intikken:

```
LOAD *"n";0
```

waarna de randen van de beeldschermen zwart zullen worden, omdat ze wachten op ontvangst van het programma. Daarna zal het station dat het programme verstuurt, moeten intikken

```
SAVE *"n";0
```

waarna ieder "luisterend" station het programma laadt. Als u gegevens uitzendt, is er geen manier om uit te vinden wie er iets ontvangen heeft, als dit het geval is. Heeft u echter iets gelezen via het netwerk, dat kunt u met

```
PEEK 23759
```

te weten komen, welk station dit heeft verstuurd.

7.4 Het printer- en microdrive-bedieningsprogramma

Een groot voordeel van een netwerk is, dat meerdere Spectrums gebruik kunnen maken van de randapparatuur, zoals printers en Microdrives. Er volgen twee programma's om ieder "slave"-station (bijv. van een leerling) in staat te stellen gebruik te maken van de printer en Microdrives bij het "master"-station (bijv. de leraar). Ze zijn zodanig geschreven, dat ze het gebruik van randapparatuur eenvoudig maken, met commando's als SAVE en LPRINT. Eén programma is voor de "slave", het ander is voor de "master". Als eerste moet ieder station worden voorzien van het "slave"-programma, hetgeen het eenvoudigst kan door het "master"-station dit programma te laten uitzenden naar alle andere stations. Het "master"-programma moet vervolgens worden geladen in de Spectrum waaraan de microdrives en printers zijn gekoppeld. Vervolgens wordt dit programma gestart. Als de machinetaal is gePOKEd, wordt de rand van het beeldscherm zwart en is het systeem gereed.

Ieder ''slave''-station kan een FORMAT voor zichzelf uitvoeren om elk stationsnummer te verkrijgen, behalve 32 en 64. Er kunnen programma's worden geschreven en getest op de ''slave''-stations, zolang de regelnummers onder de 9000 worden gehouden, om het ''slave''-programma niet te overschrijven. Indien een ''slave''-station wenst te beschikken over één van de randapparaten, moet GO TO 9000 worden uitgevoerd. Er wordt dan een menu getoond van waaruit de gebruiker een keuze kan maken om gebruik te maken van de printer, om programma's te laden en weg te schrijven, of om een CAT van de programma's uit te voeren.
Wordt mogelijkheid 1 gekozen, dan wordt hij aangesloten op de printer, via het ''master''-station – dit kan zijn de ZX-printer of een type RS232-printer. Er kan nu gebruik worden gemaakt van LPRINT en LLIST enz. totdat men daarmee klaar is. Via CLOSE #3 wordt er dan afgesloten. Mogelijkheid 2 stelt de gebruiker vervolgens in staat om programma's van ieder soort bestand (bijv. SCREEN$) weg te schrijven naar een cartridge bij het ''master''-station. Mogelijkheid 3 kan bestanden laden, die zijn weggeschreven via mogelijkheid 2, naar de cartridge van het ''master''-station. Mogelijkheid 4 drukt een CAT af van alle gebruikersprogramma's, en mogelijkheid 5 beëindigt het bedieningsprogramma.

Slave programmalijst

```
9000 CLS
9010 PRINT '"  1. Gebruik printe
r"
9020 PRINT '"  2. SAVE programma
"
9030 PRINT '"  3. LOAD programma
"
9040 PRINT '"  4. CAT cartridge"
9050 PRINT '"  5. beeindigen"
9060 PAUSE 0: LET a$=INKEY$
9070 IF a$="1" THEN  GO TO 9150
9080 IF a$="2" THEN  GO TO 9300
9090 IF a$="3" THEN  GO TO 9400
9100 IF a$="4" THEN  GO TO 9500
9110 IF a$<>"5" THEN  GO TO 9060
9120 GO TO 9000
9147 REM ************************
9148 REM *  printer uitvoer     *
9149 REM ************************
9150 LET a$="print": GO SUB 9200
9160 CLOSE #3: OPEN #3;"n";64
9170 PRINT "Printer is gereed. G
ebruik LPRINT, LLIST etc. Bent u
 klaar doe dan CLOSE #3"
9180 GO TO 16000
9197 REM ************************
9198 REM *uitzenden a$ & wacht *
9199 REM ************************
9200 PRINT  INVERSE 1;"Systeem i
s bezig kontakt te     ";"maken
 met Master station       ."
9205 CLOSE #4: OPEN #4;"n";0
9210 PRINT #4;a$
```

```
9220 CLOSE #4
9230 OPEN #4;"n";64
9240 IF INKEY$#4="" THEN  GO TO
9205
9250 CLOSE #4
9260 PRINT "Contact gemaakt."
9270 RETURN
9297 REM ************************
9298 REM * SAVE programma       *
9299 REM ************************
9300 LET a$="SAVE": GO SUB 9200
9310 INPUT "Bestandsnaam voor SA
VE:"; LINE f$
9320 OPEN #4;"n";64
9330 PRINT #4;f$
9340 CLOSE #4
9350 PRINT "Microdrive staat ger
eed. Doe SAVE*""n"";64 enz."
9360 GO TO 16000
9397 REM ************************
9398 REM *  LOAD programma      *
9399 REM ************************
9400 LET a$="LOAD": GO SUB 9200
9410 INPUT "Bestandsnaam voor LO
AD:"; LINE f$
9420 OPEN #4;"n";64
9430 PRINT #4;f$
9440 CLOSE #4
9450 OPEN #4;"n";64
9460 INPUT #4;st
9470 CLOSE #4
9480 IF st<>6 THEN  PRINT "Besta
nd niet gevonden": STOP
9490 PRINT "Microdrive staat ger
eed. Doe   LOAD*""n"";64 etc."
9495 GO TO 16000
9497 REM  ************************
9498 REM  *  CATalogus           *
9499 REM  ************************
9500 LET a$="CAT": GO SUB 9200
9510 CLS
9520 PRINT "CATALOGUS:"
9530 MOVE "n";64 TO #2
9540 PRINT ' FLASH 1;"Druk op ee
n willekeurige toets om door te
gaan"
9550 PAUSE 0
9560 GO TO 9000
```

Master programmalijst

```
  1 REM ***********************
  2 REM *Master station prog*
  3 REM ***********************
 10 CLEAR 65089
 20 LET st=65260: GO SUB 8000
```

```
  30 LET open=65090: GO SUB 8350
  40 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
 100 FORMAT "n";64
 110 CLOSE #4
 120 OPEN #4;"n";0
 130 INPUT #4; LINE a$
 140 CLOSE #4
 150 LET n=PEEK 23759
 160 IF a$="PRINT" OR a$="SAVE"
OR a$="LOAD" OR a$="CAT" THEN  G
O TO 180
 170 GO TO 120
 180 OPEN #4;"n";n
 190 PRINT #4
 200 CLOSE #4
 210 IF a$="SAVE" THEN  GO TO 30
0
 220 IF a$="LOAD" THEN  GO TO 60
0
 230 IF a$="CAT" THEN  GO TO 800
 237 REM ************************
 238 REM * printer nodig       *
 239 REM ************************
 240 PRINT "Printer in gebruik d
oor station ";n
 250 MOVE "n";n TO #3
 260 LPRINT ''''''''
 270 PRINT "(Printer klaar)"
 280 GO TO 110
 297 REM ************************
 298 REM * SAVE nodig          *
 299 REM ************************
 300 OPEN #4;"n";n
 310 PRINT "SAVE voor station ";
n
 320 DIM f$(10): INPUT #4; LINE
f$
 330 CLOSE #4
 340 GO SUB 500: LET f$(10)=CHR$
 n
 350 FOR i=1 TO 10
 360 POKE USR "a"+i-1,CODE f$(i)
 370 NEXT i
 380 POKE 23766,d: POKE 23768,5
 390 LET a=USR open: CLOSE #5
 400 IF a<>1 THEN  ERASE "m";d;f
$
 410 OPEN #5;"m";d;f$
 420 POKE FN p(5*2+23566+8)+FN p
(23631)-1+67,4
 430 MOVE "n";n TO #5
 440 CLOSE #5
 450 PRINT "(Programma SAVE klaa
r)"
 460 GO TO 110
```

```
 497 REM ************************
 498 REM *  bereken d uit n     *
 499 REM ************************
 500 LET d=1
 510 RETURN
 597 REM ************************
 598 REM * LOAD programma       *
 599 REM ************************
 600 OPEN #4;"n";n
 610 DIM f$(10): INPUT #4; LINE
f$
 620 CLOSE #4: LET f$(10)=CHR$ n
 630 PRINT "LOAD voor station ";
n
 640 GO SUB 500
 670 FOR i=1 TO 10
 680 POKE USR "a"+i-1,CODE f$(i)
 690 NEXT i
 700 POKE 23766,d: POKE 23768,5
 710 LET a=USR open
 720 OPEN #4;"n";n
 730 PRINT #4;a
 740 CLOSE #4
 750 IF a<>6 THEN  GO TO 770
 760 MOVE #5 TO "n";n
 770 CLOSE #5
 780 PRINT "(Programma geladen)"
 790 GO TO 110
 797 REM ************************
 798 REM * CAT  cartridge       *
 799 REM ************************
 800 PRINT "CATalogus voor stati
on ";n
 805 GO SUB 500
 810 LET z$="": CAT #14,d
 820 OPEN #4;"n";n
 830 LET z$=z$(13 TO )
 840 IF LEN z$<10 THEN  GO TO 88
0
 850 IF z$(10)=CHR$ n THEN  PRIN
T #4;z$( TO 9)
 860 LET z$=z$(12 TO )
 870 GO TO 840
 880 PRINT #4: CLOSE #4
 890 PRINT "(CAT klaar)"
 900 GO TO 110
 910 REM
```

Let op dat ook de OPEN #-routine en Stream 14-z$-routine aanwezig moeten zijn tussen de regels 7995-8510.
Ik zal beide programma's toelichten, opdat u kunt zien wat er zich bij beide stations afspeelt:
Het eerste wat de ''master'' doet, is het opzetten van de machinetaalroutines – 16K ''master''-stations moeten de volgende routine overnemen

```
10 CLEAR 32489
20 LET st=32490: GO SUB 8000
30 LET open=32490: GO SUB 8350
```

Het "master"-station moet nummer 64 hebben; hier zorgt regel 100 voor.
Een "slave" moet eerst contact maken met de "master", dit gebeurt door het uitzenden van een woord over het netwerk dat de "master" herkent en bevestigt. De bestanden van de "slave" staan opgeslagen op de cartridge, met negen posities voor de naam en de tiende geeft het stationsnummer aan van de "slave" als CHR$, waarin N het "slave"-stationsnummer is.
Dit is gedaan om de programma's van de "slaves" gescheiden te houden.
Als een "slave" gebruik wenst te maken van de printer, dan verstuurt hij herhaaldelijk het woord "PRINT", totdat een teken wordt ontvangen van station 64. Dit bewerkstelligen van het contact wordt uitgevoerd door de subroutine vanaf regel 9200, die stream 4 opent als uitzender, een **a$** verstuurt (9210) en ten slotte de stream weer sluit. Vervolgens kan worden geconstateerd of er een antwoord is gearriveerd van station 64 (9230-9240). Is er geen antwoord, dan wordt opnieuw een a$ verstuurd. Is er wel antwoord, dan wordt de stream gesloten (9250). Zodra het contact is gelegd, opent regel 9160 stream 3 opdat gegevens naar het "master"-station worden verstuurd.
Het `GOTO 1600` op regel 9180 en ook andere regels, dienen om het programma te stoppen met de boodschap `Program finished`.
Het "master"-station leest continu strings via de zendstream, regels 120-140. Vervolgens wordt getest of het verzoek van een "slave" afkomstig is (regel 160). Zo niet, dan wordt het lezen voortgezet (170), anders wordt het stationsnummer van de "slave" berekend (150) en wordt er een antwoord naar deze "slave" verstuurd (180-200). Ieder woord dat iets verzoekt, wordt getest, en de bijbehorende GO TO wordt uitgevoerd (210-230). Is het woord "PRINT", dan wordt een passend bericht getoond op het beeldscherm van de "master" en wordt het krachtige MOVE-commando gebruikt om de gegevens te verplaatsen van de "slave" naar het randapparaat dat is aangesloten op stream 3 (regel 250). Voert de "slave" een CLOSE #3 uit, dan stopt het MOVE-commando en wordt weer een bericht afgedrukt. Er wordt weer teruggekeerd naar de routine die het zendchannel leest.

Als een "slave" mogelijkheid 2 kiest om zijn programma te bewaren (SAVE), dan wordt het woord "SAVE" verzonden, totdat de "master" antwoordt (regel 9300). Er wordt vervolgens gevraagd naar de naam van het bestand, die dan wordt opgestuurd naar de "master" (9310-9340). Er wordt dan een bericht afgedrukt waaruit blijkt dat een SAVE *"n";64 kan worden uitgevoerd, gevolgd door de voor het bestandstype geschikte parameters. Indien alleen programma's mogen worden weggeschreven, plaats dan de volgende regel

```
9350 SAVE *"n";64: GO TO 9000
```

Zodra de "master" het woord "SAVE" leest van een station via het zendchannel, opent hij een stream naar de "slave" en drukt een boodschap af (300-310). Regel 340 gebruikt dan de subroutine op regel 500 om te berekenen welke microdrive correspondeert met deze "slave". In de huidige versie, wordt drive 1 altijd geselecteerd, maar zijn er erg veel "slave"-stations dan kan dit onpraktisch zijn. Bijvoorbeeld, indien er 63 "slaves" zijn, en de "master" beschikt over acht microdrives,

dan zou iedere drive door acht stations kunnen worden gebruikt en zou regel 500 kunnen worden

```
500 LET d= 1 + INT(n/8)
```

Is het drivenummer berekend, dan wordt het tiende teken in de bestandsnaam hiervan voorzien (340) en wordt gePOKEd in de "users graphics area" (350-370). Dit, samen met regel 380, zorgt ervoor dat alles klaar staat voor de OPEN#-routine van stream 5, in regel 390. Als de benodigde bestandsnaam reeds bestaat, wordt deze eerst gewist (met ERASE regel 400). Regel 410 opent een schrijfbestand met een passende naam en regel 420 misleidt het systeem door het bestand als niet-gegevensbestand voor te stellen. Het programma wordt verstuurd vanaf de "slave" in het bestand, weer met behulp van MOVE (430). Is de SAVE klaar, dan wordt de stream gesloten en een bericht afgedrukt (440-450).
Als de "slave" mogelijkheid 3 kiest, het laden van een programma, dan wordt het woord "LOAD" uitgezonden en wordt er weer gewacht op een antwoord (9400). Er wordt om een bestandsnaam gevraagd die wordt verstuurd naar de "master" (9450-9470), waarna wordt gecontroleerd of het bestand aanwezig is. Zo niet, dan wordt een passende boodschap afgedrukt (9480). Bestaat het bestand wel, dan wordt dit geladen met

```
LOAD *"n";64
```

gevolgd door de benodigde attributen.
Indien alleen programma's zijn toegestaan, verander dan de volgende regel

```
9490 LOAD *"n";64:GO TO 9000
```

Als de "master" het woord "LOAD" ontvangt (220) dan leest deze de bestandsnaam van de "slave" (600-620) in en plaatst een stationsnummer in positie 10, om de bron te identificeren. Het drivenummer wordt berekend (640) en de bestandsnaam wordt in de UDG area gePOKEd. Regel 700 voert meer POKEs uit, met behulp van de OPEN #-routine (710) om te controleren op de aanwezigheid van het gevraagde programma. Het geretourneerde nummer wordt verstuurd naar de "slave" met behulp van MOVE (760).
De microdrive stream wordt dan afgesloten en een bericht wordt afgedrukt (770-780).
Wordt mogelijkheid 4 gekozen om zijn programma's via een CAT-overzicht te bekijken, dan wordt het woord "CAT" verstuurd totdat een bevestiging van de "master" wordt ontvangen (9500). Het scherm wordt gewist en de titel wordt afgedrukt, waarna alle invoer vanaf het "master"-station (de bestandsnamen) wordt afgedrukt, met behulp van MOVE (9530).
Als het "master"-station het woord "CAT" ontvangt (230), wordt een bericht afgedrukt en het drivenummer berekend (800-805). Met behulp van de stream 14-z$-routine, wordt het CAT-overzicht in de variabele z$ geplaatst (810), waarna de stream naar de "slave" wordt geopend (820). De titel wordt van de string verwijderd (830) en als er een cartridgenaam volgt, wordt het tiende teken getest. Blijkt dit te corresponderen met het aanroepende "slave"-station, dan wordt de bijbehorende naam naar de "slave" opgestuurd (850). Zijn er verder geen namen aanwezig, dan wordt de stream afgesloten en een boodschap afgedrukt (800-890).

Verdere verbeteringen

Sommige gebruikers onder u zouden graag meer mogelijkheden willen zien in dit programma. Eén daarvan zou kunnen zijn een lijst van toegangswoorden op te nemen, met voor ieder aangesloten station een toegangswoord (bijv. in een array of een gegevensbestand) opdat het ene station nooit de gegevens van een ander station kan vernietigen. Als de namen van iedere gebruiker per station ook worden opgenomen, dan kunnen de uitvoerresultaten worden voorzien van deze namen, om ze zo te kunnen onderscheiden.

Netwerkaanvullingen bij het streamoverzicht-programma

De volgende regels dienen te worden toegevoegd aan het streamoverzicht-programma om ook netwerk-streams te onderkennen. Dit drukt het stationsnummer en bestemmingsnummer af en geeft aan of gebruik wordt gemaakt van de uitzendmogelijkheid.

Streamoverzicht programmalijst-aanvullingen

```
1650 IF f$="N" THEN  GO TO 3500
3499 REM Channel N
3500 PRINT  INVERSE 1;"NETWERK"
3510 GO SUB 1800
3520 PRINT "Stationsnr. :";PEEK
23749
3530 PRINT "zendt naar:";PEEK (D
+11);" (uitzending)" AND PEEK (d
+11)=0
3540 GO TO 1500
```

8. Machinetaal en de interface

Dit hoofdstuk is geschreven voor die lezers die kennis hebben van Z80-machinetaal, maar ook de lezers die niet over deze kennis beschikken, vinden mogelijk nuttige onderdelen.

8.1 De schaduw 8K ROM

Als aanvulling op de 16K BASIC ROM in de Spectrum, heeft de interface 1 een 8K ROM die in hetzelfde geheugengebied wordt gezet, dus van 0000-1FFFH (nummers die gevolgd worden door H, staan in hexadecimaal formaat). Het zeer vernuftige ontwerp van dit zogenaamde "paging"-systeem maakt het gebruik van de extra routines in de schaduw ROM, door het Spectrum-besturingssysteem zeer eenvoudig, maar bemoeilijkt het leven van machinetaalprogrammeurs, die deze ook willen gebruiken. Voordat we ons verder verdiepen in het besturingssysteem en de ROM, moet het vernuftige "paging"-systeem worden uitgelegd. Om verwarring te voorkomen, worden de schaduw ROM-adressen in dit boek voorzien van het prefix "X" en waar adressen afwijken binnen de twee ROMs wordt de oude positie eerst getoond, gevolgd door het nieuwe adres, gescheiden door middel van een schuine streep.

8.2 Het "paging"-systeem

Met één uitzondering, wordt de schaduw ROM inge"paged" indien de programmateller de waarde 0008 bereikt, dus als bijvoorbeeld een RST 8 optreedt. In de 16K ROM wordt deze herstart gevolgd door een byte die bepaalt welke foutboodschap moet worden gegenereerd terwijl het BASIC-programma draait (of dat een flitsend vraagtekend wordt gegeven, indien de syntax van een regel fout is). Wordt er een poging ondernomen om één van de aanvullende commando's (bijv. CAT) of syntax (bijv. LOAD *) te gebruiken, dan genereert de 16K BASIC een foutboodschap, met behulp van RST 8. Is echter de interface 1 aangesloten dan wordt de BASIC uitge"paged" en de schaduw ROM inge"paged". Er wordt dan uitgerekend wat de fout veroorzaakte en of het een uitgebreide BASIC-opdracht betreft. Zo niet, dan wordt de normale foutafhandeling gebruikt (in de 16K ROM). Betreft het een uitgebreide BASIC-opdracht, dan reageert de schaduw ROM hierop (voert deze uit, of controleert de syntax) en "paged" zichzelf uit na de foutindicatie te verwijderen die het "inpagen" heeft veroorzaakt. Het moet worden erkend, dat dit een briljante methode is om commando's toe te voegen zonder ook maar één byte te wijzigen in de 16K ROM van de Spectrum, maar het bemoeilijkt het gebruik van de ROM-routines in eigen programmatuur.
Om u te helpen bij het ontwijken van dit obstakel, hebben de schaduw ROM-auteurs gebruik gemaakt van zogenaamde "hook codes" die een routine in staat stellen, toegang te verkrijgen tot bepaalde schaduw ROM-routines. Deze "hook codes" zijn gegevensbytes die volgen op een RST8-commando en kunnen variëren van 1BH tot en met 32H (1B t/m 34H in de nieuwe ROM). De codes FFH tot 1AH produceren de gewone foutboodschappen, terwijl de codes boven 32H (of 34H op de nieuwe ROM) een `Hook code error` produceren, welke niet wordt beschreven in de handleiding. Sommige van deze "hook codes" zijn zeer nuttig, andere weer niet, maar de pro-

grammeur moet zich bewust zijn van de status van de Maskable Interrupts bij het aanroepen en beëindigen van sommige routines. Indien een terugkeer naar de BASIC zal optreden (na een USR-functie) dan moet het H''L'' registerpaar altijd worden bewaard wanneer microdrive-routines worden gebruikt. Wordt H''L'' niet bewaard, dan kan de drijvende-komma-calculator stuklopen en vervolgens het systeem.
Hook code 32H is misschien de meest bruikbare code, ondanks het feit dat deze officieel is ontworpen als toekomstige uitbreiding, **Reserved by Sinclair Research Ltd.** Deze stelt een gebruikersroutine in staat om de meeste schaduw ROM-routines aan te roepen, evenals het naar believen in- en uit''pagen'' van de schaduw ROM.
Sommige lezers zullen geïnteresseerd zijn in een disassembly van de 8K ROM, maar natuurlijk heeft een PEEK of een LD A,(HL) tot gevolg dat de 16K ROM wordt gelezen. Indien een volledige disassembly benodigd is, volgt hier de manier waarop ik het op een 48K Spectrum deed:

```
10 CLEAR 32767
20 SAVE *"m";1;"8K ROM" CODE 0,8192
30 LOAD *"m";1;"8K ROM" CODE 32768
```

Wanneer de SAVE* wordt uitgevoerd, is de schaduw ROM actief, zodat een kopie ervan wordt bewaard op de cartridge. Regel 30 laadt deze weer terug, maar nu 8000H boven de oorspronkelijke locatie. Vervolgens kan worden gedisassembleerd vanaf locatie 8000H, maar vergeet niet 8000H af te trekken van alle absolute adressen.
Behalve RST-8, wordt de schaduw ROM inge''paged'' indien PC=1708H, dit is het midden van de CLOSE #-commandoroutine. Deze wordt veroorzaakt door een fatale fout en is vrijwel niet in staat om de extra channelsoorten die de interface gebruikt, te behandelen.

De oude en nieuwe schaduw ROM
In maart 1984 heeft Sinclair Research de schaduw ROM van de interface 1 veranderd, zoals eerder beschreven in dit boek. Ook al zijn de verschillen bijna onzichtbaar voor de BASIC-programmeur, ze zijn het zeker niet voor de machinetaal-programmeur. De nieuwe ROM heeft de meeste fouten aanwezig in de oude gecorrigeerd (zie ook Appendix C) en heeft de meeste microdrive-routines versneld, met name CAT en FORMAT. Helaas bevinden de meeste routines zich op verschillende locaties in iedere ROM, waardoor het gebruik ervan iets verschilt. Om te zien welke ROM is aangesloten, terwijl deze is inge''paged'' (zie later hoe dit gebeurt) dient u locatie X16DA te testen. In de oude ROM is dit een ongebruikt gedeelte en bevat FFH, maar in de nieuwe ROM bevindt zich hier de start van een copyright-bericht en bevat 7FH. In de volgende notities wordt zoveel mogelijk gebruik gemaakt van ROM-onafhankelijke routines.

8.3 Hoe streams en channels werken

Streams en channels zijn voor zowel de BASIC- als voor machinetaalprogramma's beschikbaar. Streams maken gebruik van het STRMS-geheugengebied en worden gekoppeld aan de channels die gebruik maken van het CHANS-gebied. Het STRMS-gebied ligt vast en bevindt zich in het gebied van 5C10H tot en met 5C35H,

het bestaat uit 19 paren bytes, één paar per stream van FDH tot 0FH. Ieder paar vormt een 16-bit verplaatsing, die 0000 is indien de stream geen channel heeft. Is deze niet nul, dan wordt het gebruikt als een index binnen het CHANS-gebied. Zodra geïnitialiseerd, bestaat het CHANS-gebied uit 20 bytes, vier groepen van vijf, met vijf bytes per channel. Het CHANS-gebied start op de locatie geadresseerd door de systeemvariabele CHANS op positie 5C4FH en eindigt twee bytes voor PROG.
In het begin bevat de channelinformatie gegevens over vier channels, met vijf bytes per channel, in de volgorde "K", "S", "R" en "P". De eerste zestien bytes van het STRMS-gebied bevatten de bytes zoals getoond in de volgende tabel, samen met het toegekende channel:

Streamnr.	*Streamloc.*	*Channel ind.*	*Channel*
FD	5C10	0001	"K"
FE	5C12	0006	"S"
FF	5C14	000B	"R"
00	5C16	0001	"K"
01	5C18	0001	"K"
02	5C1A	0006	"S"
03	5C1C	0010	"P"
04	5C1E	0000	n.v.t.

Om de relevante channelgegevens van een stream te vinden, wordt de channel-index opgesteld bij CHANS (nadat deze gecontroleerd is op nullen) en vervolgens verlaagd met één. Het resultaat wijst dan naar de start van de relevante gegevens, die uit ten minste vijf bytes bestaan.
Het eerste paar bytes wijst naar de ROM-uitvoerroutines, het tweede paar naar de invoerroutines en de vijfde byte bevat de channelidentificatie met een hoofdletter. Channel "S" wordt bijvoorbeeld als volgt opgeslagen:

```
CHANS+5 DEFW 09F4H
        DEFW 15C4H
        DEFM "S"
```

In feite wordt 09F4H gebruikt als uitvoerroutine voor de channels "K", "S" en "P", doch op basis van de channelidentificatie worden verschillende acties ondernomen.

8.4 De interface channels

Om van de interfacemogelijkheden gebruik te maken, heeft het systeem 58 bytes extra nodig voor systeemvariabelen, toegevoegd vanaf positie 5CB6H. Dit gebied moet aanwezig zijn, voordat de schaduw ROM of hookcoderoutines worden aangeroepen en voordat bepaalde extra channeltypen worden gecreëerd (zoals bijv. de Stream14-z$ routine). Om dit te bereiken, maakt u gebruik van de volgende instructies:

```
CF RST 08
31 DEFB 31H
```

Indien de interface niet is aangesloten, dan wordt de niet-bestaande foutboodschap "i" gegeven. Is hij wel aangesloten, dan worden de 58 extra bytes aangemaakt en wordt geen foutboodschap gegeven. De variabelen worden in Appendix A beschreven.
De extra systeemvariabelen worden normaal aangemaakt wanneer

a. er een fout optreedt (in de syntax of tijdens de uitvoering);
b. een extended BASIC-commando wordt gegeven of uitgevoerd;
c. een CLOSE #-commando wordt uitgevoerd.

De variabele VECTOR op adres 5CB7H wordt initieel op 01F0H gezet, maar dit kan zodanig worden veranderd, dat extra commando's kunnen worden toegevoegd aan de BASIC. Deze eigenschap van de interface zou in een apart boek kunnen worden beschreven, doch enige gegevens hieromtrent vindt u in het volgende hoofdstuk.
De variabele IOBORD op adres 5CC6H wordt gebruikt om de BORDER-kleur te bepalen gedurende bepaalde I/O-opdrachten, namelijk RS232, netwerken en gedurende het schrijven naar een cartridge. Indien u een bepaalde BORDER-kleur wenst, kunt u POKEn op adres 23750 met de gewenste kleur. Wanneer u echter liever geen flitsende BORDER heeft, kunt u doen:

```
POKE 23750, INT (PEEK 23624/8)
```

die de I/O-kleur gelijk maakt aan die van de normale BORDER-kleur.
De extra interface channels "M", "N", "T" en "B" maken gebruik van een uitgebreid formaat om hun gegevens op te slaan in het CHANS-gebied.

"M"-channel
Dit gebruikt 595 bytes in het CHANS-gebied en wordt geadresseerd via het IX-register in de schaduw ROM. De lay-out is als volgt (alle posities in decimaal):

0	DEFW 8	"uitvoer" routine
2	DEFW 8	"invoer" routine
4	DEFM "M"	channelidentificatie
5	DEFW 11D8H/12B3H	schaduw ROM-uitvoer
7	DEFW 1122H/11FDH	schaduw ROM-invoer
9	DEFW 595	lengte van CHANS-gebied
11	CHBYTE	huidige bufferpositie (0-512)
13	CHREC	positie van record in bestand (0-255)
14	CHNAME	10 posities bestandsnaam
24	CHFLAG	bit 0 – uit – leesbestand aan – schrijfbestand bit 1 – 7 – ongebruikt (alle aan)
25	CHDRIVE	drivenummer (1–8)
26	CHMAP	drive MAP positie
28	HPREAM	12 bytes headerinleiding
40	HDFLAG	bit 0 – aan om header te signaleren bit 1–7 – ongebruikt (ongedefinieerd)
41	HDNUM	sectornummer (0-255)
42		ongebruikt (ongedefinieerd)
44	HDNAME	10 bytes cartridgenaam

54	HDCHK	header controlesom
55	DPREAM	12 bytes headerinleiding
67	RECFLG	bit 0 – uit om aan te geven "geen header"
		bit 1 – uit – geen EOF
		aan – EOF
		bit 2 – uit – een PRINT-bestand
		aan – geen PRINT-bestand
		bit 3–7 – ongebruikt (alle uit)
68	RECNUM	recordnummer (0-255)
69	RECLEN	aantal bytes in record (0-512)
71	RECNAM	10 bytes bestandsnaam
81	DESCHK	controlesom van RECFLG tot DESCHK-1
82	CHDATA	eerste byte van 512 bytes buffer
593		laatste byte van buffer
594	DCHK	controlesom van buffer

Iedere Microdrive die in gebruik is, heeft ook nog 32 bytes gereserveerd in het Microdrive maps-gebied, van 5CH tot CHANS-1. Ieder blok van 32 bytes is een bit-map van iedere sector op de cartridge. Indien een bit aan staat, is de sector onbruikbaar omdat deze gegevens bevat, of beschadigd is, of omdat hij niet bestaat.

"N"-channel

Dit gebruikt 276 bytes in het CHANS-gebied en wordt geadresseerd via het IX-register in de schaduw ROM. De lay-out is als volgt:

0	DEFW 8	
2	DEFW 8	
4	DEFM "N"	channelidentificatie
5	DEFW 0D6CH/0EA9H	schaduw ROM uitvoerroutine
7	DEFW 0D0CH/0DA9H	schaduw ROM invoerroutine
9	DEFW 276	lengte van dit CHANS-gebied
11	NCIRIS	nummer eindstation (0-64)
12	NCSELF	nummer van dit station (0-64)
13	NCNUMB	bloknummer (0-65535)
15	NCTYPE	pakketsoort - 0-gegevens, 1-EOF
16	NCOBL	bytes in gegevensblok (of 0 indien uitvoer)
17	NCDCS	gegevens controlesom
18	NCHCS	header controlesom
19	NCCUR	invoerbuffer positie
20	NCIBL	aantal bytes in invoerbuffer (0 indien uitvoer)
21	NCB	begin van 225 bytes invoerbuffer

"T"- en "B"-channels

Deze gebruiken elk 11 bytes in het CHANS-gebied, wat het minimum is voor van de standaard afwijkende channeltypen. De lay-out is als volgt:

0	DEFB 8	
2	DEFB 8	
4	DEFB "T" (oude ROM)/"T" of "B" (nieuwe ROM)	identificatie
5	DEFW 0C3CH/0C3AH (T) of 0C5AH/0D07H (B)	schaduw uitvoer

7	DEFW 0B6FH/0B76H (T) of 0B75H/0B7CH (B)	schaduw invoer
9	DEFW 11	lengte van dit CHANS-gebied

Indien gebruik wordt gemaakt van SAVE * enz., MOVE, FORMAT en ERASE vanuit BASIC of machinecode, worden tijdelijke channels aangemaakt. Zij zijn identiek aan de bovenstaande channels, maar met drie belangrijke verschillen:

a. de identificatie-byte heeft de hoogste bit aan staan;
b. zij worden nooit toegekend aan een stream;
c. zij worden verwijderd zodra het commando is beëindigd, of indien een fout optreedt;
d. zij sluiten aan op het einde van het CHANS-gebied – een permanent channel mag nooit hoger liggen dan een willekeurig tijdelijk channel.

Bij van de standaard afwijkende channels, is het minimum aantal bytes in het CHANS-gebied 11, zoals dat van toepassing is voor "T"- en "B"-channels. De eerste twee paren bytes, gewoonlijk de invoer- en uitvoerroutine-adressen, staan beide op 0008, en veroorzaken dat de schaduw-ROM wordt ingePAGEd. Deze onderkent dat een channelroutine nodig is en gebruikt de bytes die vijf bytes verderop in het CHANS-gebied staan (dit is de schaduw ROM-routine). Zodra een routine eindigt, wordt de schaduw ROM vanzelf uitgePAGEd.

Het paar bytes op de 9de en 10de positie vormt een 16-bit nummer, dat het systeem laat weten hoeveel bytes het channel gebruikt in het CHANS-gebied. Zij vormen een zeer belangrijk paar bytes, omdat zij worden gebruikt in geval van fouten, om door het CHANS-gebied heen te stappen op zoek naar channels die moeten worden gesloten. Wordt dit overgeslagen, of incorrect uitgevoerd, dan zal het systeem stuklopen bij de eerste fout die optreedt.

Gebruik van streams

Om een stream te gebruiken voor invoer, wordt de STRM_OPEN routine gebruikt op positie 1601H. (Deze is in geen enkel opzicht gelijk aan de BASIC OPEN #-opdracht.) Tijdens de aanroep moet de accumulator een correct streamnummer bevatten (van FDH tot 0FH). Hiervan wordt de relevante locatie in STRMS afgeleid, tenzij deze nul is, wat de foutboodschap `Invalid stream` tot gevolg heeft. Is deze niet nul, dan wordt de relevante locatie in CHANS berekend en deze waarde opgeslagen in de systeemvariabele CURCHL op positie 5C51H.
Om tekens uit te voeren naar de huidige stream, laadt u de code in de accumulator en roept u de OUTPUT-routine op positie 0010H aan met behulp van een RST 10 instructie. De registerparen BC, DE en HL blijven onveranderd.
Om tekens in te voeren via de huidige stream, voer dan uit

```
SET 4,(IY+124)
```

en roep de INPUT-routine op positie 15E6H aan. De registerparen BC, DE en HL blijven ongewijzigd. Tijdens de terugkeer, wordt de carry flag gezet indien geen teken werd gelezen, anders worden zowel carry als zero flag op nul gezet in geval van een **EOF**-situatie.

Gebruik van channels
Er is helaas geen eenvoudig machinecode-equivalent van de OPEN #-opdracht. Voor experimenteren, kunnen de benodigde streams worden geopend vanuit de BASIC en gebruikt in de machinecode, maar dit is wellicht onvoldoende voor uiteindelijk gebruik.
De machinecode-equivalenten voor ieder type OPEN #-opdracht worden later in dit hoofdstuk gegeven, in de relevante sectie. Het machinecode-equivalent van CLOSE # is als volgt:

```
LD A,stream
RES 1,(IY+124)  ; geef aan "niet CLEAR #"
LD HL,1718H     ; (schaduw ROM CLOSE routine)
LD (HD#11),HL
RST 8
DEFB 32H
RET
```

Deze routine gebruikt hook code 32H om de schaduw ROM CLOSE #-routine aan te roepen.

8.5 ROM-routines

Er volgt nu een lijst van schaduw ROM-routines om gebruik te kunnen maken van de interface-mogelijkheden. Er wordt uitgelegd hoe de routine te gebruiken (gewoonlijk via hook-codes) en de aanroep- en terugkeercondities van registers en interrupts worden gegeven. Ook worden de registers getoond die worden beïnvloed door de routine, evenals de eigenlijke locaties van de routines.

8.6 Microdrive-routines

OPEN-M- Creëer een tijdelijk "M"-channel in het CHANS-gebied

gebruik:	RST 8
	DEFB 22H
aanroep:	INTs aan
terugkeer:	IX=start van gebied, HL=streamverplaatsing
	(=IX-CHANS+1), INTs uit, motor aan
Registers:	AF,BC,DE,HL,B'C',D'E',H'L',IX
Locatie:	X1B29H/1B05H

Uitvoering: Voordat deze routine wordt gebruikt, moet de systeemvariabele D_STR1 het drivenummer bevatten, N_STR1 de lengte van de bestandsnaam en T_STR1 moet geadresseerd staan aan het begin van de bestandsnaam. Een gebied van 595 bytes wordt aangemaakt aan het eind van het CHANS-gebied (indien het geheugen daar nog ruimte voor biedt) en de relevante attributen worden erin gekopieerd. Indien een map-gebied niet aanwezig is voor de gekozen drive, wordt er een aangemaakt en gevuld met FFen. De drive wordt dan geactiveerd en de cartridge wordt gelezen om het bestand te vinden. Wordt deze gevonden, dan wordt de eerste sector geladen en bit 0 van CHFLAG afgezet, zo niet, dan wordt deze gezet om aan te geven dat het een schrijfbestand betreft. Wanneer het een leesbestand betreft,

wordt bit 0 van IX+25 (CHFLAG) afgezet als het een PRINT-bestand betreft, anders wordt deze aangezet (voor programma's, bytes enz.). Let er op dat het schrijfbeveiligingsnokje niet wordt gecontroleerd. Om dit wél te controleren, kunt u het volgende doen na de routine

```
IN (A),(EFH)
AND 1    ; aan indien beveiligd
```

Let er op dat de drivemotor niet wordt uitgeschakeld door deze routine en dat het gecreëerde channel van tijdelijke aard is. Om dit permanent te maken en toe te kennen aan een stream doet u het volgende:

```
LD A,stream
ADD A,A
LD HL,5C16H
LD E,A
LD D,0
ADD HL,DE
PUSH HL      ; bewaar stream locatie
RST 8
DEFB OPEN-M
PUSH HL      ; bezwaar stream verplaatsing
XOR A
RST 8
DEFB MOTOR   ; schakel motor uit
POP DE
POP HL
LD (HL),E
INC HL
LD (HL),D    ; sla nieuwe gegevens op in STRMS
RES7,(IX+4) ; maak het permanent
RET
```

MAKE-M- creëer een 595 bytes groot "M" gebied in CHANS

Gebruik

```
: Oude ROM          Nieuwe ROM
LD (HL),0FE8H       RST 8
LD (HD_11),HL       DEFB 2BH
RST 8
DEFB 32H
```

Aanroep: INTs aan
Terugkeer: IX=begin van gebied, HL=IX-CHANS+1, INTs aan
Registers: AF,BC,DE,HL,B'C',D'E',H'L',IX
Locatie: XOFE8H/10A5H

Uitvoering: Voordat de routine wordt gebruikt, dient D_STR1 het drivenummer te bevatten, N_STR1 de lengte van de bestandsnaam en T_STR1 het begin. Een 595 bytes groot gebied wordt toegevoegd aan het eind van CHANS (indien het geheugen dit toelaat) en voorzien van alle relevante attributen.
Wanneer nog geen map bestaat voor de drive dan wordt er een aangemaakt en

gevuld met FFen. Bij de oude ROM kan hook-code 2BH niet worden gebruikt, omdat deze foutievelijk hetzelfde doet als 22H.

MOTOR – Activeer een drivemotor of schakel alle motoren uit.

Gebruik:	RST 8
	DEFB 21H
Aanroep:	A=1 tot 8 om de drive aan te schakelen, A=0 om alle motoren uit
Terugkeer:	INTs uit indien A<>0, aan indien A=0
Registers:	AF,BC,DE,HL
Locatie:	X17F7H/1532H

Uitvoering: Wanneer de accumulator niet nul is, wordt de relevante Microdrivemotor aangeschakeld en alle andere uit en interrupts worden gedeactiveerd. Indien A nul is, dan worden alle drives uitgeschakeld en de interrupts weer geactiveerd.
Indien de gekozen drive niet is aangesloten, of als er een cartridge ontbreekt of niet is ingedeeld (FORMAT) dan wordt de foutboodschap `Microdrive not present` gegeven.

CLOSE-M – sluit een "M"-channel

Gebruik:	RST 8
	DEFB 23H
Aanroep:	IX=begin van "M"-channel-gebied
Terugkeer:	INTs aan
Registers:	AF,BC,DE,HL
Locatie:	X12A9H/138EH

Uitvoering: Indien het gekozen channel een schrijfbestand was, dan worden de resterende bytes in de buffer verzonden. De drivemotor wordt uitgeschakeld en het 595 bytes gebied teruggegeven. Het relevante map-gebied wordt ook vrijgegeven, indien dit niet door een ander channel wordt gebruikt.

ERASE-M – wis een Microdrivebestand

Gebruik:	RST 8
	DEFB 24H
Aanroep:	n.v.t.
Terugkeer:	INTs aan
Registers:	AF,BC,DE,HL,B'C',D'E',H'L',IX
Locatie:	X1D6EH/1D79H

Uitvoering: Voordat deze routine wordt gebruikt, moeten de systeemvariabelen D-STR1, T-STR1 en N-STR1 worden ingevuld met het benodigde drivenummer en de string van het te wissen bestand. Een tijdelijk "M"-channel wordt aangemaakt en de cartridge wordt doorzocht op zoek naar het bestand. Indien het wordt gevonden dan wordt het gewist (doch slechts een kopie wordt gewist indien er meerdere kopieën aanwezig zijn) op voorwaarde dat het schrijfbeveiligingsnokje aanwezig is. Is het niet aanwezig, dan treedt er een fout op. Zodra de routine klaar is, wordt de motor uitgeschakeld en het tijdelijke channel verwijderd.

RECLAIM-M – reclameer een 595 bytes groot "M"-gebied

Gebruik: RST 8
DEFB 2CH
Aanroep: IX=begin van het "M"-channel-gebied
Terugkeer: n.v.t.
Registers: AF,BC,DE,HL
Locatie: X10C4H/119FH

Uitvoering: Ten eerste wordt het 595 bytes gebied vrijgegeven, en de ernaar verwijzende streams worden gesloten. De 32 bytes map wordt alleen vrijgegeven indien het door geen ander channel wordt gebruikt.

CAT – geef een overzicht van een Microdrive cartridge

```
Gebruik:    LD HL,CATANY
            LD (HD_11),HL
            RST 8
            DEFB 32H
              .
              .
              .
CATANY      LD HL, (04B0H)
            INC HL
            LD E, (HL)
            INC HL
            LD D,(HL)
            EX DE,HL
            JP (HL)
```

Aanroep: INTs aan
Terugkeer: n.v.t.
Registers: AF,BC,DE,HL,A'F',B'C',D'E',H'L',IX
Locatie: X1C58H/1C52H

Uitvoering: Voordat de routine wordt gebruikt, moet S_STR1 het streamnummer bevatten voor de CAT en D_STR1 het drivenummer. Omdat er geen hook-code bestaat voor CAT, wordt code 32H gebruikt samen met HD_11 om de routine CATANY aan te roepen, waarbij de schaduw ROM is ingePAGEd. CATANY berekent de positie van de CAT-routine door het bestuderen van de syntax-routine die op dezelfde plaats staat in beide ROMs. Ten eerste wordt de gekozen stream geopend (via 1601H) en daarna wordt een tijdelijk "M"-channel gecreëerd. Iedere header op de cartridge wordt gelezen en de bestandsnaam opgeslagen in de 512 bytes buffer in het CHANS-gebied, tenzij deze al aanwezig is, of voorzien is van het prefix CHR$ 0. De naam wordt in alfabetische volgorde toegevoegd aan de buffer.
Als de gehele cartridge is gelezen, of wanneer meer dan 50 namen zijn gevonden, wordt de drive uitgeschakeld en de naam van de cartridge afgedrukt op de gekozen stream. Iedere bestandsnaam wordt vervolgens opgenomen uit de buffer en afgedrukt, ten slotte wordt de hoeveelheid vrije kilobytes afgedrukt. Dit laatste aantal wordt berekend uit de helft van de hoeveelheid vrije bits in het relevante map-gebied. Uiteindelijk wordt het "M"-gebied gereclameerd.

Cartridge gegevensindeling

Voordat de meer ingewikkelde microdrive-routines worden besproken, moet de layout van de gegevens op tape worden uitgelegd. Een cartridge is verdeeld in ongeveer 180 sectoren, die ieder 512 bytes bevatten. De routines zijn eigenlijk alleen geschikt voor 256 sectoren, maar de sectoren 0 en die boven 180 bestaan niet, of worden niet gebruikt. Daarbij komt nog dat twee of drie sectoren onbruikbaar zijn, omdat zij bij de las van de tape liggen.

Voorafgaand aan iedere sector ligt een header, bestaande uit een 12 bytes inleiding (dit zijn tien nullen en twee FFen) en daarna 15 bytes met de variabelen HDFLAG-HDCHK. Iedere header bevat een verschillend nummer (HDNUMB) van 1 tot ongeveer 180, sequentieel opgeslagen op de cartridge. Volgend op de header bevindt zich de sector zelf, bestaande uit een 12 bytes inleiding (zoals eerder), de 15 bytes variabelen (RECFLG tot DESCHK), daarna de 512 bytes gegevens en ten slotte een controlesombyte (DCHK). Een sector is ongebruikt indien bit 1 van RECFLG en bit 1 van RECLENhi (IX+70) uit staan.

Omdat de meeste bestanden te groot zijn om in een sector te worden opgeslagen, worden ze verdeeld in porties van 512 bytes records en wordt ieder record opeenvolgend genummerd te beginnen bij nul. Indien een sector minder dan 512 bytes bevat, dan wordt dit via het aanzetten bit 1 van RECFLG gemarkeerd als laatste record. Niet-PRINT-typebestanden, zoals programma's, slaan hun variabelen op in de eerste negen bytes van record nummer 0. Deze attributen zijn HD_00 tot HD_11, die worden verklaard in Appendix A.

De resterende hook-codes zijn als volgt:

READ-P – lees een record van een PRINT-bestand

Gebruik:	RST 8
	DEFB 27H
Aanroep:	IX=begin van "M"-channel-gebied
Terugkeer:	INTs uit, motor aan
Registers:	AF,BC,DE,HL
Locatie:	X1A17H/1F0BH

Uitvoering: Voordat deze routine wordt gebruikt moet CHDRIV het drivenummer bevatten, CHREC het gewenste recordnummer en CHNAME de bestandsnaam. Als eerste wordt de aangegeven drive geactiveerd en SECTOR gezet op 04FB (=5 volledige omwentelingen). De cartridge wordt vervolgens doorzocht, op zoek naar het gekozen recordnummer van het gekozen bestand. Wordt dit gevonden en is het een PRINT-bestand dan wordt de sector ingelezen in de buffer en de controle teruggegeven. Betreft het geen PRINT-bestand, dan wordt het gebied gereclameerd en een foutboodschap `Wrong file type` gegenereerd. Wordt het bestandsrecord na 5 zoekacties op de tape niet gevonden, dan treedt de foutboodschap `File not found` op en wordt het gebied vrijgegeven indien dit van tijdelijk aard was.

READ-NP – lees het volgende record van een PRINT-bestand

Gebruik:	RST 8
	DEFB 25H
Aanroep:	IX=begin van "M"-gebied

Terugkeer: INTs uit, motor aan
Registers: AF,BC,DE,HL
Locatie: X1A09H/1EFDH

Uitvoering: Deze routine is identiek aan READ-P, CHDRIV moet het drivenummer bevatten en CHNAME de bestandsnaam. Ten eerste wordt bit 1 van RECFLEG gecontroleerd om te zien of het huidige record in het geheugen de laatste is – is dit het geval dan verschijnt een `End of file` bericht. Is dit niet het geval dan wordt de inhoud van CHREC verhoogd en de controle wordt overgedragen aan READ-P.

READ-S – lees volgende sector

Gebruik: RST 8
DEFB 29H
Aanroep: IX=start van "M"-gebied, INTs uit, motor aan
Terugkeer: INTs uit, motor aan
Registers: AF,BC,DE,HL
Locatie: X1AB6H/1F7AH

Uitvoering: De volgende header en sector op de tape worden ingelezen in het channelgebied. Indien het PRINT-bestand betreft, wordt de controle teruggegeven met de carry flag uit. Is het geen PRINT-bestand, of klopt de controlesom niet, dan wordt de inhoud van de buffer uitgewist en de controle teruggegeven met de carry flag aan.

READ-R – lees een willekeurige sector

Gebruik: RST 8
0DEFB 28H
Aanroep: IX=begin van "M"-gebied, INTs uit, motor aan
Terugkeer: INTs uit, motor aan
Registers: AF,BC,DE,HL
Locatie: X1FE4/1F3FH

Uitvoering: De cartridge wordt doorzocht op zoek naar het sectornummer in CHREC. Wordt dit gevonden, dan wordt de buffer gevuld. Indien het een PRINT-bestand betreft wordt de carry flag uitgezet en de controle teruggegeven. Betreft het geen PRINT-bestand dan wordt de buffer uitgewist en de carry flag aangezet. Wordt de relevante sector na een omwenteling niet gevonden, dan wordt de foutboodschap `File not found` gegeven.

SCAN-M – lees een header van een cartridge (alleen nieuwe ROM)

Gebruik: RST 8
DEFB 33H
Aanroep: IX=begin van "M"-gebied, INTs uit, motor aan
Terugkeer: INTs uit, motor aan
Registers: AF,BC,DE,HL
Locatie: X1FE4 (alleen nieuwe ROM)

Uitvoering: De cartridgetape die de kop van de Microdrive passeert, wordt gelezen totdat een correcte header wordt gevonden, welke wordt opgeslagen in het gebied van HDFLAG tot en met DESCHK. Klopt de controlesom niet, of is het prefix van de

bij de sector behorende bestandsnaam gelijk aan CHR 0, dan wordt het gehele gebied gewist en de carry flag aangezet; in het andere geval wordt deze uitgezet. Om opeenvolgende sectorheaders te lezen, mag de codering tussen twee aanroepen niet langer duren dan 30 ms.

WRITE-S – schrijf een sector serieel weg

Gebruik:	RST 8
	DEFB 26H
Aanroep:	IX=begin van "M"-gebied
Terugkeer:	INTs aan, motor aan
Registers:	AF,BC,DE,HL
Locatie:	X11FFH/12DAH

Uitvoering: Voor de aanroep moeten de variabelen CHBYTE, CHREC, CHNAME, CHDRIVE en het buffergebied correct worden ingevuld. Als eerste wordt de motor geactiveerd en gecontroleerd of de cartridge vol is. De bestandsnaam wordt gekopieerd van CHNAME naar RECNAM, en CHBYTE en CHREC gekopieerd naar RECLEN en RECNUM. De controlesommen DESCHK en DCHK worden berekend, en het gebied RECFLG tot DCHK wordt weggeschreven naar de eerste ongebruikte sector op de cartridge (op voorwaarde dat deze niet is beveiligd tegen overschrijven). De relevante bit in map wordt aangezet, CHBYTE wordt op nullen gezet en CHREC verhoogd. Ten slotte wordt de motor uitgeschakeld.

WRITE-R – schrijf een sector willekeurig weg

Gebruik:	RST 8
	DEFB 2AH
Aanroep:	IX=begin van "M"-gebied, INTs uit, motor aan
Terugkeer:	INTs uit, motor aan
Registers:	AF,BC,DE,HL
Locatie:	X1A91H/1F85H

Uitvoering: Voor de aanroep, moet CHREC op de gewenste waarde worden gezet en CHNAME de naam bevatten van de sector die moet worden weggeschreven. De cartridge wordt eenmaal doorzocht op sectornummer CHREC. Wordt dit niet gevonden, dan verschijnt de foutboodschap `File not found`. Wordt dit wel gevonden en is de schrijfbeveiliging uitgeschakeld, dan worden de sector- en gegevensbytes (RECFLG-DCHK) naar de cartridge weggeschreven en de relevante bit in map aangezet. Let er op dat de map niet wordt gecontroleerd voor de schrijfactie, zodat, indien dit wel gewenst is, het moet worden uitgevoerd door de gebruiker.

8.7 RS232-routines

Voordat de RS232-routines worden gebruikt, moeten de extra systeemvariabelen aanwezig zijn en moet een passende BAUD-waarde worden gekozen. Het eerste kan worden bereikt met behulp van hook-code 31H, het laatste door het aanpassen van BAUD op positie 5CC3H.

232IN – lees een byte in van de RS232

Gebruik:	RST 8
	DEFB 1DH
Aanroep:	n.v.t.
Terugkeer:	de carry flag wordt aangezet indien een byte is gelezen, A=byte, INTs aan
Registers:	AF,BC,DE,HL
Locatie:	X0B81H/0B88H

Uitvoering: Leest een byte vanuit de RS232-poort, tenzij er te lang moet worden gewacht of indien SPACE (oude ROM) of BREAK (nieuwe ROM) wordt ingedrukt (opdat de boodschap `Break into program` verschijnt). De carry flag wordt aangezet wanneer een byte is ingelezen.

232OUT – verzendt een byte van de RS232

Gebruik:	RST 8
	DEFB 1EH
Aanroep:	INTs aan
Terugkeer:	A=byte code
Registers:	AF,BC,DE,HL
Locatie:	X0C5AH/0D07H

Uitvoering: Een byte wordt verzonden via de poort, met de juiste snelheid en de benodigde "handshaking". Wordt er op BREAK gedrukt dan treedt er een fout op.

Indien u gebruik wenst te maken van de equivalent van het "T"-channel voor RS232-uitvoer, gebruik dan hook-code 32H om X0C3CH (in de oude ROM) of X0C3AH (in de nieuwe ROM) waarbij register A een tekencode bevat. Let erop dat de nieuwe ROM twee extra variabelen gebruikt wanneer met het "T"-channel wordt gebruikt – 5CB0H voor de cursorpositie en 5CB1 voor de printerbreedte. Initieel staan deze op resp. 0 en 80.

OPEN-R – open een RS232-channel

Gebruik:	oude ROM	nieuwe ROM
	LD HL,0B13H	LD HL,0B17H
	LD (HD-11),HL	LD (HD-11),HL
	RST 8	RST 8
	DEFB 32H	DEFB 32H
Aanroep:	INTs aan	
Terugkeer:	DE=begin van nieuw CHANS-gebied	
Registers:	AF,BC,DE,HL	
Locatie:	X0B13H/0B17H	

Uitvoering: Deze routine creëert een 11 bytes gebied aan het einde van CHANS voor een "B"- of "T"-channel. Gewoonlijk wordt er een "T"-channel gecreëerd, tenzij L-STR1 een "B" bevat, zodat een "B"-channel wordt aangemaakt. Op de nieuwe ROM zit een additionele hook-code OPEN-B, 34H die een "B"-channel opent.

8.8 Netwerkroutines

Netwerk is geïmplementeerd op de Spectrum door een continue aaneensluiting tussen de poorten van ieder station. Er kan eenmaal communicatie per keer plaats vinden en alle gegevens worden serieel verzonden. De gegevens worden verdeeld in genummerde pakketten, ieder met een maximum van 255 bytes. Een pakket wordt als volgt verzonden: Ten eerste controleert de verzender of het netwerk reeds actief is – is dit het geval, dan wordt er gewacht. Is het netwerk niet actief, dan worden acht bytes van een header via de kabel verzonden, welke zijn opgebouwd uit de systeemvariabelen NTDEST tot NTCHS. Deze header bevat gevarieerde informatie over de gegevens die erop volgen, en in het bijzonder de nummers van zender en ontvanger, evenals de twee controlesommen. Tenzij het een uitzending betreft, wacht de verzender op een bevestiging van de ontvanger in de vorm van een enkele byte met het nummer van het ontvangende station. Wordt deze byte niet ontvangen, dan wordt opnieuw een header verzonden. In het geval van een uitzending, wordt er geen controle uitgevoerd op de confirmatiebyte, en worden de gegevens zonder meer verzonden. De gegevenssectie wordt niet verzonden als er geen bytes in staan (dit is als NTLEN=0), anders wordt het relevante aantal bytes verzonden. Een andere antwoordbyte wordt daarna gecontroleerd (tenzij het een uitzending betreft) en indien deze niet wordt ontvangen wordt opnieuw een header verzonden. Er bestaan vier hook-codes om gebruik te maken van het netwerk, doch een daarvan is helaas onbruikbaar op de oude ROM.

OPEN-N – open een tijdelijk "N"-channel

Gebruik:	RST 8
	DEFB 2DH
Aanroep:	n.v.t.
Terugkeer:	IX=DE=(CURCHL)=begin van "N"-gebied in CHANS
Registers:	AF,BC,DE,HL
Locatie:	X0EA9H/0F46H

Uitvoering: Voordat de routine wordt aangeroepen, moet D-STR1 het bestemmingsnummer bevatten en NSTAT het eigen nummer van de Spectrum. Er wordt een 265 bytes gebied gecreëerd aan het einde van het CHANS-gebied en er worden gegevens in opgeslagen. Het bestemmingsnummer wordt gekopieerd van D-STR1 naar NCIRIS, het stationsnummer van NSTAT naar NCSELF en het gebied van NCNUMB tot het laatste teken in de buffer wordt voorzien van nullen. Het channel wordt gemarkeerd als zijnde tijdelijk via het zetten van bit 7 in de channelidentificatie. De variabele CURCHL wordt gezet op het adres dat wijst naar het begin van het nieuw gecreëerde gebied. Om een permanent netwerk-channel te creëren, kunt u de volgende codering gebruiken (na het zetten van D-STR1 en NTSTAT);

```
        LD A,stream
        ADD A,A
        LD HL,5C16H
        LD E,A
        LD D,0
        ADD HL,DE       ; HL passende locatie in STRMS
        PUSH HL         ; bewaar het
```

```
RST 8
DEFB OPEN-N   ; open tijdelijk "N"
RES 7, (IX+4) ; maak het permanent
LD HL,(CHANS)
EX DE,HL
AND A
SBC HL,DE
INC HL        ; HL=stream verplaatsing
POP DE        ; DE=STRMS locatie
EX DE,HL
LD (HL),E     ; bewaar verplaatsing in STRMS
INC HL
LD (HL),D
RET
```

CLOSE-N – sluit een "N"-channel

Gebruik: RST 8
DEFB 2EH
Aanroep: (CURCHL)=begin van "N"-gebied in CHANS
Terugkeer: INTs aan
Registers: AF,BC,DE,HL,IX
Locatie: X1A24H/1F18H

Uitvoering: Indien het channel een schrijfbestand is (dit is als NCOBL<>0) dan wordt de resterende bufferinhoud in een pakket verzonden met de indicatie EOF. Het 256 bytes gebied wordt daarna gereclameerd, doch iedere aangesloten stream wordt niet gesloten. Het "N"-gebied moet of tijdelijk, of het laatste channel in CHANS zijn, anders worden andere streams of channels overschreven.

WRITE-N – zendt een pakket via het netwerk

Gebruik: RST 8
DEFB 30H
Aanroep: A=0 voor gegevens, 1 voor EOF, IX=begin van "N"-gebied
Terugkeer: INTs aan, A=bestemmingsnummer (Z indien uitzending)
Registers: AF,BC,DE,HL
Locatie: X0DB2H/0E4FH

Uitvoering: Voordat de routine wordt aangeroepen, moeten de variabelen NCOBL en NCNUMB, de inhoud van de buffer en het headergebied (exclusief de controlesom) van de juiste waarden zijn voorzien. Bij aanroep, zou de accumulator normaal 0 moeten bevatten, of 1 wanneer het een laatste pakket betreft van een reeks (dit is een EOF-indicatie). De waarde wordt opgeslagen in NCTYPE en de kleur van BORDER wordt gezet op IOBORD. Een controlesom van de bufferinhoud wordt opgeslagen in NCDCS en een controlesom van NCIRIS tot NCDCS wordt opgeslagen in NCHCS. De interrupt wordt onmogelijk gemaakt en het pakket wordt verzonden. Indien het pakket is ontvangen (en het betreft geen uitzending) dan wordt NCNUMB opgehoogd, de kleur van BORDER teruggezet en de interrupt weer mogelijk gemaakt.

READ-N – lees een pakket van het netwerk (alleen nieuwe ROM)

Gebruik: RST 8
DEFB 2FH
Aanroep: IX=begin van ''N''-gebied
Terugkeer: INTs aan
Registers: AF,BC,DE,HL
Locatie: (X1A31H)/1F25H

Uitvoering: Deze hook-code leest een pakket bytes vanuit het netwerk. De carry flag wordt uitgezet, indien het pakket correct is ingelezen, anders wordt deze aangezet, maar in de oude ROM wordt deze overschreven.
De alternatieve methode om tekens in te lezen vanuit het netwerk, gebeurt via de 16K ROM routine INPUT, op positie 15E6H zoals eerder beschreven. Onthoud goed dat CURCHL wordt bewaard wanneer u andere channels gebruikt tussen meerdere aanroepen van de routine.

Andere hook-codes

PAUSE – wacht op een toetsaanslag (of toetsherhaling)

Gebruik: RST 8
DEFB 1BH
Aanroep: n.v.t.
Terugkeer: INTs aan, A=toetscode
Registers: AF, BC,DE,HL
Locatie: X19D9H/1ECDH

Uitvoering: De interrupt wordt geactiveerd en ieder 50ste deel van een seconde wordt de toetsenbord-leesroutine in de 16K ROM aangeroepen, totdat een toets wordt aangeslagen. De toetscode wordt gelezen vanuit LAST-K.

PRINT – Druk een teken af op het scherm

Gebruik: RST 8
DEFB 1CH
Aanroep: A=tekencode, INTs aan
Terugkeer: n.v.t.
Registers: AF,BC,DE,HL,A'F',B'C',D'E'
Locatie: X19ECH/1EEOH

Uitvoering: De variabele SCR-CT wordt gezet op FF om een automatische scroll mogelijk te maken, stream FEH (channel ''S'') wordt geopend en het teken afgedrukt.

LPRINT – Druk een teken af op de ZX printer

Gebruik: RST 8
DEFB 1FH
Aanroep: A=tekencode, INTs aan
Terugkeer: n.v.t.
Registers: AF,BC,DE,HL,A'F',B'C',D'E'

Locatie: X19FCH/1EFOH

Uitvoering: Stream 3 (gewoonlijk channel "P") wordt geopend en het teken afgedrukt.

SCAN-K – tast de toetsenbordmatrix af

Gebruik: RST 8
DEFB 20H
Aanroep: n.v.t.
Terugkeer: NZ (Non-Zero) indien toets is ingedrukt
Registers: AF,BC,DE,HL
Locatie: X1A01H/1EF5H

Uitvoering: Het toetsenbord wordt afgetast en wanneer een willekeurige toets wordt aangeslagen (inclusief de SHIFT-toetsen) dan wordt de Zero flag afgezet. De interrupt is niet nodig.

NEWVARS – creëer de extra systeemvariabelen

Gebruik: RST 8
DEFB 31H
Aanroep: n.v.t.
Terugkeer: n.v.t.
Registers: AF,BC,DE,HL
Locatie: X19A8H/1E98H

Uitvoering: Indien de extra 58 systeemvariabelen (in Appendix A) nog niet aanwezig zijn, dan worden zij gecreëerd, wanneer daarvoor voldoende geheugenruimte aanwezig is. (In feite bestaat de routine uit alleen de instructie RET – de extra variabelen worden al gecreëerd door RST 8 zelf.)

SHADOW – roep een willekeurige schaduw ROM-routine aan

Gebruik: RST 8
DEFB 32H
Aanroep: (HD-11)=locatie
Terugkeer: n.v.t.
Registers: Ongespecificeerd (hangt van routine af)
Locatie: X19AH/1E94H

Uitvoering: De schaduw ROM-routine geadresseerd via HD-11 wordt aangeroepen. De officieel als Reserved by Sinclair Research Ltd ontworpen hook-code is de meest krachtige in de ROM. Alleen de waarde van het A-register hoeft te worden overgedragen, terwijl alle waarden door de routine worden teruggegeven. Hij kan zelfs worden gebruikt om de BASIC ROM uit te PAGEn, via de volgende codering:

```
        LD HL,PAGOUT
        LD (HD-11),HL
        RST 8
        DEFB 32H
PAGOUT: POP HL          ; verwijder 0700H van de stack
```

```
        POP HL          ; verwijder PAGOUT van de stack
        .               ; rest van het programma
        .
```

Wilt u de 16K ROM terugPAGEn, voer dan een CALL 0700H uit.

8.9 Overzicht hook-codes

Dit is een overzicht van iedere hook-code, de naam, schaduw ROM-locatie en functie.

Code	*Naam*	*Locatie*	*Functie*
1B	PAUSE	19D9/1ECD	Wacht op toetsaanslag – waarde in A
1C	PRINT	19EC/1EE0	druk CHR$ A af op stream FE (het scherm)
1D	232IN	0B81/0B88	RS232 invoer in A – carry indien o.k.
1E	232OUT	0C5A/0D07	RS232 uitvoer – code in A
1F	LPRINT	19FC/1EF0	druk CHR$ A af op stream 3 (printer)
20	SCAN-K	1A01/1EF5	tast toetsenbordmatrix af – NZ indien aanslag
21	MOTOR	17F7/1532	schakel Microdrive-motor aan of uit
22	OPEN-M	1B29/1B05	open een tijdelijk ''M''-channel
23	CLOSE-M	12A9/138E	sluit ''M''-channel
24	ERASE-M	1D6E/1D79	wis een bestand van een cartridge
25	READ-NP	1A09/1EFD	lees volgende PRINT-buffer in
26	WRITE-S	11FF/12DA	zend de ''M''-buffer naar een cartridge
27	READ-P	1A17/1F0B	lees een PRINT-buffer in
28	READ-R	1A4B/1F3F	zoek een willekeurig bestand
29	READ-S	1A86/1F7A	zoek een serieel bestand
2A	WRITE-R	1A91/1F85	schrijf een blok willekeurig
2B*	MAKE-M	----/10A5	creëer ''M''-channel
2C	RECLAIM	10C4/119F	reclameer een 595 bytes ''M''-gebied
2D	OPEN-N	0EA9/0F46	open een tijdelijk ''N''-channel
2E	CLOSE-N	1A24/1F18	sluit een ''N''-channel
2F*	READ-N	----/1F25	lees een netwerkpakket
30	WRITE-N	0DB2/0E4F	schrijf een netwerkpakket
31	NEWVARS	19A8/1E98	creëer de extra systeemvariabelen indien nog niet aanwezig
32	SHADOW	19A4/1E94	roep de schaduw ROM-routine aan geadresseerd via HD-11
33*	SCAN-M	----/1FE4	lees volgende cartridge header
34*	OPEN-B	----/)FF6	open ''B''-channel

* Alleen van toepassing op nieuwe ROM.

Opmerkingen

Er zijn bepaalde dingen waar een machinetaalprogrammeur zich van bewust moet zijn tijdens het gebruik van de interface:

a. Gebruik alleen de USR-functie in LET- of RANDOMIZE-opdrachten. Wordt USR gebruikt in de uitgebreide BASIC-opdrachten, dan zal een fout optreden en kan de schaduw ROM het systeem ophangen.

b. Wees uzelf steeds bewust van de status van de interrupt bij aanroep en terugkeer van enkele schaduw ROM-routines. Interrupts moeten actief zijn, voordat wordt teruggekeerd naar de BASIC.
c. Behoud altijd de waarde van het H''L'' registerpaar, wanneer gebruik wordt gemaakt van microdriveroutines, zodra wordt teruggekeerd naar de BASIC.
d. Gebruik nooit belangrijke cartridges bij het uittesten van microdriveroutines. Het kan zeer teleurstellend zijn als blijkt dat een ongewilde FORMAT al uw brontaal heeft uitgewist. Schrijfbeveiligingsnokjes zijn niet altijd bestand tegen wilde machinecodefouten.
e. Overtuig u ervan dat de extra systeemvariabelen zijn aangemaakt, via hookcode 31H.
f. Gebruik nooit de REM-instructie voor interface-routines. De aanmaak van het CHANS-gebied veroorzaakt een verschuiving van de routines in het geheugengebied, gedurende de uitvoering ervan en zullen het systeem ophangen.

9. Het toevoegen van BASIC-commando's

Evenals de uitgebreide BASIC-commando's zijn toegevoegd aan de Spectrum door de interface I, deze stelt ook de machinetaalprogrammeur in staat zijn eigen commando's toe te voegen. Dit is een zeer bruikbare mogelijkheid, aanwezig op de meeste andere computers, maar altijd afwezig bij de Sinclair-serie. De belangrijkste reden hiervoor is het feit dat, tot nu toe, alle commando's bepaalde sleutelwoorden moesten zijn en de tekenset biedt hiervoor geen ruimte. Wat de interface I doet is het toevoegen van een mogelijkheid om naar een toegevoegde machinecoderoutine te springen in geval van een syntaxfout in een instructieregel, indien deze wordt ingebracht, of wordt uitgevoerd. Dit maakt het mogelijk de syntax te wijzigen van de meeste bestaande commando's, om op deze wijze hun functie te veranderen, de interface-commando's echter, kunnen gewoonlijk niet worden veranderd. Voor niet-interface-commando's, bestaan verschillende manieren om functies toe te voegen:

a. voeg tekens toe op vreemde plaatsen bijv. `COPY #`
b. verander de parametersoorten bijv. `POKE 30000,"x"`
c. voeg parameters toe bijv. `PLOT x,y,"PLOF"`
d. laat parameters weg bijv. `CIRCLE 10,30`
e. gebruik E-mode functies bijv. `LINE 10,30`

Ook kunt u uw eigen commando's gebruiken, voluit getypt, maar voorafgegaan door een niet-alfabetisch teken (bijv. **!REPEAT**). Het niet-alfabetische teken is nodig om de Spectrum uit de K-mode te halen gedurende het intikken van de regel.
De sleutelwoorden waarvan de syntax gewoonlijk niet kan worden gewijzigd, zijn LOAD, SAVE, MERGE, VERIFY, CAT, FORMAT, OPEN, CLOSE, CLS, CLEAR en MOVE. Ik zeg gewoonlijk, omdat ze eigenlijk wel gewijzigd kunnen worden wanneer u het niet erg vindt om vingeroefeningen toe te passen als u ze wilt intikken! De manier om ze te wijzigen is, ze vooraf te laten gaan van een SHIFT-teken, zoals ''*''. Voor de sleutelwoorden betekent dit het intikken van het woord, cursor terugbrengen naar links, intikken van ''*'', cursor naar rechts brengen en vervolgens de rest intikken!
Om uw eigen commando's te creëren, moet u over een behoorlijke dosis machinetaalkennis beschikken en van het Spectrum besturingssysteem op de hoogte zijn, voor deze lezers beschrijft de tweede helft van dit hoofdstuk de bruikbare methodes. De eerste helft bevat een programma dat de syntax wijzigt van de meest gebruikelijke microdriveroutines, dat door iedere lezer kan worden ingebracht en gebruikt.
Om de interface te informeren over het feit dat nieuwe commando's zijn toegevoegd, moet u twee POKE-opdrachten uitvoeren, welke moeten worden uitgevoerd in een multi-opdrachtregel. Zij mogen niet worden uitgevoerd indien er interface-opdrachten zijn uitgevoerd na een NEW. De eenvoudigste manier is een `CLOSE #` te doen, wat in feite een betekenisloze opdracht is. Om de interface terug te laten keren naar alleen de normale commando's, moet u doen

```
POKE 23735,240:POKE 23736,1
```

opnieuw in een enkele regel.
Ik moet toegeven dat ik geen grote fan ben van de gekozen syntax om Microdrive-programma's weg te schrijven en te laden. Volgens mij is het onnodig uitgebreid en

vereist een onnoemelijk aantal toetsaanslagen, zodat ik het volgende programma heb geschreven om eenvoudiger commando's toe te voegen. Zij zijn in de vorm van een sterretje gevolgd door de relevante commandoletter (grote of kleine letter) en details. Om een programma genaamd *"test"* van drive 2 te laden, zou het nieuwe commando er als volgt uitzien: `*L"2test"`, hetgeen equivalent is aan: `LOAD *"m";1;"test"`. Zo geldt dit eveneens voor `SAVE *"m"`, VERIFY en MERGE die worden vervangen door resp. `*S`, `*V`, `*M` gevolgd door een enkele string. Het eerste teken in de string moet altijd het drivenummer zijn. Een ander toegevoegd commando is `*C`, dat equivalent is met `CAT 1` en `*C` gevolgd door een nummer voor de andere drives. Het laatste nieuwe commando is `*run` die het programma met de naam *run* van drive 1 laadt, hetgeen equivalent is aan een `NEW` gevolgd door RUN.

Extra commando's programmalijst

```
  10 REM ************************
  20 REM *Uitgebreide M syntax*
  30 REM ************************
  40 REM                  voor 16K
  50 CLEAR 65169:  REM 32401
  60 LET ex=65170: REM 32402
  65 LET x2=INT ((ex+124)/256):
LET x1=ex+124-256*x2
  70 RESTORE 200: LET c=0
  80 FOR i=ex TO ex+194
  90 READ a: LET c=c+a
 100 POKE i,a
 110 NEXT i
 115 IF c<>22003+2*(x1+x2) THEN
 PRINT "Kontrolesom fout": STOP
 120 CLOSE #0
 130 POKE 23735,ex-256*INT (ex/2
56): POKE 23736,INT (ex/256)
 140 PRINT "Nieuwe commando's"
 150 PRINT "*L LOAD"'"*S SAVE"'"
*M MERGE"'"*V VERIFY"
 160 PRINT "*C CAT"'"*RUN"
 170 STOP
 200 DATA 254,92,194,240,1,215,3
2,0
 210 DATA 246,32,254,115,40,22,2
54,108
 220 DATA 40,28,254,118,40,34,25
4,109
 230 DATA 40,36,254,99,40,38,254
,114
 240 DATA 40,62,24,222,253,203,1
24,238
 250 DATA 205,x1,x2,195,54,8,253
,203
 260 DATA 124,230,205,x1,x2,195,
165,8
 270 DATA 253,203,124,254,24,244
,253,203
```

```
 280 DATA 124,246,24,238,33,214,
92,54
 290 DATA 1,35,54,0,35,54,2,215
 300 DATA 32,0,254,13,40,7,254,5
8
 310 DATA 40,3,205,30,6,195,169,
4
 320 DATA 215,32,0,246,32,254,11
7,194
 330 DATA 40,0,215,32,0,246,32,2
54
 340 DATA 110,32,244,215,32,0,20
5,183
 350 DATA 5,195,149,10,62,77,50,
217
 360 DATA 92,215,32,0,215,140,28
,215
 370 DATA 24,0,223,202,35,7,215,
241
 380 DATA 43,33,11,0,167,237,66,
218
 390 DATA 76,6,120,177, 202,76,6
,26
 400 DATA 214,48,254,1,56,27,254
,9
 410 DATA 48,23,50,214,92,175,50
,215
 420 DATA 92,11,19,237,67,218,92
,237
 430 DATA 83,220,92,215,24,0,195
,35
 440 DATA 7,231,4
```

Ik vind deze nieuwe commando's veel eenvoudiger en sneller in het gebruik en heb het programma bewaard onder de naam "run" op mijn belangrijkste programma-ontwikkelings-cartridge. Wanneer u het programma intikt, verder dan de regels 50 en 60 om RAMTOP van een passende waarde te voorzien en om aan te geven waar het programma moet worden geplaatst. De aanbevolen adressen zijn niet in overeenstemming met de eerder in dit boek beschreven routines. Wenst u dat ze wel overeenstemmen, voeg dan de volgende regels toe:

```
50 CLEAR 64579: REM 31809 voor de 16K
60 LET mc=64580: REM 31810 voor de 16K
```

Ieder later toegevoegd nieuw commando zal worden gedeactiveerd na een NEW, zodat u een leeg programmabestand moet aanmaken op een cartridge genaamd "new". Daarna, om een bestaand programma te wissen en de commando's te behouden, doet u

```
*L "1new"
```

9.1 Hoe commando's worden toegevoegd

Deze helft van het hoofdstuk is voor machinetaalprogrammeurs. Enige kennis van het Spectrum-besturingssysteem is ook zeer bruikbaar – voor dit doel beveel ik van harte aan *''The Complete Spectrum ROM Disassembly''* van Dr Ian Logan, waarvan ik de 16K ROM-routine adreslabels in dit boek heb gebruikt.
Van cruciaal belang voor de extra commando's is de interface variabele VECTOR op positie 5CB7H. Deze bevat normaal 01F0H, wat een schaduw ROM-locatie is die de standaard-foutafhandelingroutine uitvoert (na het kopiëren van CHADD terug naar CH-ADD).
Om uit te leggen hoe VECTOR moet worden veranderd, moeten de acties van de schaduw ROM in geval van een fout worden verklaard. De waarden tussen haakjes verwijzen naar de ROM-posities in hex.
Ten eerste wordt HL voorzien van het terugkeeradres (0013) (van de RST 8) en wordt gekeken of deze 0000 of 15FE is. Is deze 0000 dan wordt een 16K ROM-routine aangeroepen vanuit de schaduw ROM, of als deze 15FE is, wordt een interface channel aangevraagd. Was het geen van beide, dan worden de interface variabelen opgezet, als deze nog niet aanwezig zijn en bepaalde signalen worden verzonden aan de ULA. Vervolgens wordt het foutnummer gevonden (00C7) (vanuit de byte die volgt op de RST 8) en nagezien wordt, of dit een hook-code betreft. Zo niet, dan wordt gekeken of de fout `Nonsense in BASIC`, `Invalid filename` of `Invalid stream` was. Was het geen van deze, dan wordt de foutafhandeling in de oude ROM gebruikt (00F8). Was het wel een van de genoemde fouten, dan wordt de inhoud van CH-ADD opgeslagen in CHADD- (wat is er een verwarrende mnemonics gekozen!). Bit 3 van FLAGS3 wordt gecontroleerd – is deze aan, dan wordt de normale foutafhandeling aangeroepen. Na het berekenen van het eerste teken in de regel waar de fout optrad (0126-01E9), wordt een vergelijking gemaakt met de interface commando's (01B5-01E9). Wordt geen gelijkheid gevonden, dan wordt gesprongen naar de routine geadresseerd door VECTOR. Gewoonlijk wordt dan een fout gegenereerd, maar als VECTOR is veranderd, kunt u commando's toevoegen die afwijken van de normale syntax.
Dus, wanneer u VECTOR laat verwijzen naar uw eigen toegevoegde routines, wat moet de routine dan doen? Ten eerste moet het A-register worden gecontroleerd op het eerste teken of commando van de door uzelf gekozen commando's, hetgeen altijd minder moet zijn dan 206. Wordt er geen overeenkomst gevonden, dan verzorgt JP 01F0H de foutboodschap. Is er wel een gelijkheid gevonden, dan moet u de routine laten springen naar de bijbehorende syntaxcontrole, de benodigde parameters laten evalueren en de opdracht uitvoeren (de laatste twee acties alleen gedurende de uitvoering). Er is een aantal dingen dat u moet weten wanneer u een dergelijke routine schrijft:

1. De schaduw ROM is ingePAGEd, niet de 16K BASIC.
2. Alle Z80 restarts zijn verschillend.
3. Doe geen poging om hook-codes te gebruiken – roep de routines direct aan.
4. Ook al is de interrupt actief, het toetsenbord wordt niet afgetast en FRAMES wordt niet opgehoogd.

De schaduw restarts doen het volgende:

RST 0 – Zet FLAGS3 af en keert terug naar 16K ROM

RST 8 – NOOIT GEBRUIKEN

RST 10 – Roept 16K ROM-routines aan – gevolgd door twee gegevensbytes met de locatie.

RST 19 – BIT 7, (FLAGS) – Z indien syntaxcontrole
NZ indien in uitvoering

RST 20 – Geef een schaduw ROM-fout. Gevolgd door een gegevensbyte:

FF	Program finished	00E7
00	Nonsense in BASIC	0139
01	Invalid stream number	0663
02	Invalid device expression	062D
03	Invalid name	064C
04	Invalid drive number	0681
05	Invalid station number	05F6
06	Missing name	068D
07	Missing station number	06A1
08	Missing drive number	0683
09	Missing baud rate	06B7
0A	Header mismatch error	(nooit gebruikt in ROM)
0B	Stream already open	052F
0C	Writing to a "read" file	0D78
0D	Reading a "write" file	0D1C
0E	Drive "write" protected	128D
0F	Microdrive full	1219
10	Microdrive not present	1828
11	File not found	11A3
12	Hook code error	1985
13	CODE error	092E
14	MERGE error	07D8
15	Verification has failed	0930
16	Wrong file type	0902

(De bovenstaande hex locaties geven aan waarheen gesprongen moet worden om iedere fout te geven.)

RST 28 – Geef een 16K ROM-fout. Sla in ERRNR het benodigde boodschapnummer op, voordat de RST wordt gegeven.

RST 30 – Creëer de systeemvariabelen indien nog niet aanwezig.

RST 38 – Activeer de interrupt (hierdoor wordt het toetsenbord niet afgetast).

9.2 Routines om een regel af te tasten

Welke commando's u ook toevoegt, het is nodig om de BASIC-regel te controleren op syntax of uitvoering. Uw commandoroutine wordt tweemaal aangeroepen voor iedere regel – eenmaal voor de syntaxcontrole en eenmaal gedurende de uitvoering. Om te weten te komen welke het is, kunt u gebruik maken van RST 10. Er zijn veel routines in de oude ROM die kunnen worden gebruikt om de regel af te tasten, met behulp van RST 10. Om de regel teken voor teken af te tasten, kunt u de volgende oude ROM-routines gebruiken:

GET-CHAR 0018H – A-register bevat huidige byte in de regel.
NEXT-CHAR 0020H – A-register bevat de volgende byte in de regel en negeert controletekens en spaties.

De belangrijkste voor regelevaluatie zijn:

Class-06 1C82H – Controleer/evalueer een numerieke parameter en plaats de waarde op de calculatorstack in geval van uitvoering; in het andere geval worden onzichtbare bytes in de regel geplaatst.
CLASS-0A 1C8CH – Controleer/evalueer een parameterstring en plaats de waarde op de calculatorstack in geval van uitvoering.

Bij de aanroep van de laatste twee routines, moet CH-ADD verwijzen naar het eerste teken van de uitdrukking. Na terugkeer verwijst deze naar het eerste "niet op zijn plaats"-teken, terwijl A daarvan de code bevat (bijv. ",").
Er zijn ook enkele evaluatieroutines in de schaduw ROM die direct kunnen worden aangeroepen.

PARAMS#0701 – De routine die controleert op *string; nummer<;string;parameters>* dit zijn de tekens die volgen op LOAD,SAVE enz. Alle parameters worden overgedragen naar D-STR1, N-STR1, S-STR1, L-STR1 en HD-00 tot HD-11 gedurende de uitvoering.
EVALBC#061E – Controleert het einde van een instructie.
Indien niet aanwezig, dan volgt een fout. Is het wel aanwezig, dan wordt een RET uitgevoerd in geval van uitvoering, anders wordt het terugkeeradres verwijderd en de controle overgedragen aan de 16K ROM via 05C1.
CHKDRV 066D – Controleert de inhoud van D-STR1 op een waarde liggend tussen 1-8. Zo niet, dan verschijnt invalid drive nummer.
CHKST$ #062F – Evalueer een string – in het geval van uitvoering wordt gecontroleerd op minder dan 11 tekens en de parameters worden opgeslagen in N-STR1 en T-STR1.

Zodra de gehele syntax is gecontroleerd en correct bevonden, dient er gesprongen te worden naar 05C1H (tenzij een indirecte terugkeer plaatsvindt via CHKEND). Indien een commando succesvol is uitgevoerd, moet eenzelfde sprong worden gemaakt. De stack en foutcode worden gewist en controle wordt overgedragen aan de 16K ROM.
Zoals blijkt, is het toevoegen van commando's nogal moeilijk, maar zeer de moeite waard. Het opent de mogelijkheid de Spectrum van allerlei toolkit-type programma's te voorzien, evenals de mogelijkheid gebruik te maken van de BASIC regel-bewerker om regels van andere talen of assemblers in te brengen.
Hier volgt het brontaal overzicht (assembly) van de in de eerste helft van dit hoofdstuk genoemde toegevoegde commando's. De routine is niet positie-onafhankelijk of dynamisch – het BASIC laadprogramma verandert de CALL bytes automatisch tijdens het inPOKEn.

```
;Vector voor wijzigen van syntax
        ORG   40000     ; OORSPRONKELIJK ADRES
        ENT   $
        CP    #5C       ; TEST VOOR "*"-206
ERROR   JP    NZ,#01F0  ; GEEF FOUT ALS ONGELIJK
        RST   #10
        DEFW  #0020     ; NEXT-CHAR
        OR    #20       ; MAAK KLEINE LETTER
        CP    "s"
        JR    Z,SAVE    ; SPRING IGV    "s" of "S"
        CP    "l"
        JR    Z,LOAD    ; SPRING IGV    "l" OF "L"
        CP    "v"
        JR    Z,VERIF   ; SPRING IGV    "v" OF "V"
        CP    "m"
        JR    Z,MERGE   ; SPRING IGV    "m" OF "M"
        CP    "c"
        JR    Z,CAT     ; SPRING IGV    "c" OF "C"
        CP    "r"
        JR    Z,RUN     ; SPRING IGV    "r" OF "R"
        JR    ERROR     ; ANDERS FOUT
SAVE    SET   5,(IY+124)    ;GEEF AAN SAVE
        CALL  VARS      ; EVALUEER STRING
        JP    #0B36     ; VOER SAVE UIT
LOAD    SET   4,(IY+124)    ;GEEF AAN LOAD
DOIT    CALL  VARS      ;EVALUEER STRING
        JP    #08A5     ; DOE LOAD/VERIFY/MERGE
VERIF   SET   7,(IY+124)    ; GEEF AAN VERIFY
        JR    DOIT      ; VOER UIT
MERGE   SET   6,(IY+124)    ; GEEF AAN MERGE
        JR    DOIT      ; VOER UIT
CAT     LD    HL,#5CD6  ; HL=D_STR1
        LD    (HL),1    ; GEEF AAN DRIVE 1
        INC   HL
        LD    (HL),0    ; MAAK D_STR1 HI NUL
        INC   HL
        LD    (HL),2    ; MAAK STREAM 2
        RST   #10
        DEFW  #0020     ; NEXT-CHAR
        CP    13
        JR    Z,CAT2    ; SPRING IGV NEWLINE
        CP    ":"
        JR    Z,CAT2    ; SPRING IGV SCHEIDINGSTEKEN
        CALL  #016E     ; EVALUEER NUMMER
CAT2    JP    #04A9     ; VOER CAT UIT
RUN     RST   #10
        DEFW  #0020     ; NEXT-CHAR
        OR    #20       ; MAAK KLEINE LETTER
        CP    "u"
ERR2    JP    NZ,#0028  ; FOUT INDIEN NIET "u"
        RST   #10
        DEFW  #0020     ; NEXT-CHAR
        OR    #20
        CP    "n"
        JR    NZ,ERR2   ; FOUT INDIEN NIET "n"
        RST   #10
```

```
570         DEFW #0020     ; NEXT-CHAR
580         CALL #05B7     ; CHKEND
590         JP   #0A95     ; LAAD "RUN" PROGRAMMA
600 VARS    LD   A,"M"
610         LD   (#5CD9),A    ;MAAK L_STR1="M"
620         RST  #10
630         DEFW #0020     ; NEXT-CHAR
640         RST  #10
650         DEFW #1C8C     ; CLASS_0A (STRING)
660         RST  #10
670         DEFW #0018     ; NEXT-CHAR
680         RST  #18       ; KONTROLEER SYNTAX
690         JP   Z,#0723   ; SLA OVER IGV SYNTAX
700         RST  #10
710         DEFW #2BF1     ; STK-FETCH (BC=LENGTE, DE=START)
720         LD   HL,11
730         AND  A
740         SBC  HL,BC
750         JP   C,#064C   ; FOUT IGV MEER DAN 11 TEKENS
760         LD   A,B
770         OR   C
780         JP   Z,#064C   ; FOUT IGV LEGE STRING
790         LD   A,(DE)    ; A=EERSTE TEKENCODE
800         SUB  "0"       ; A=DRIVENUMMER
810         CP   1
820         JR   C,INVDR   ; FOUT INDIEN < 1
830         CP   9
840         JR   NC,INVDR  ; FOUT INDIEN > 8
850         LD   (#5CD6),A    ;SLA WAARDE OP IN D_STR1
860         XOR  A
870         LD   (#5CD7),A    ; WIS D_STR1 HI UIT
880         DEC  BC        ; VERLAAG LENGTE
890         INC  DE        ; STARTPOSITIE
900         LD   (#5CDA),BC   ; SLA LENGTE OP
910         LD   (#5CDC),DE   ; SLA BEGIN OP
920         RST  #10
930         DEFW #0018     ; GET_CHAR
940         JP   #0723     ; KONTROLEER ARGUMENTEN, DAARNA RET
950 INVDR   RST  #20       ; GEEF FOUT
960         DEFB 4         ; BYTE VOOR "INVALID DRIVE NO"
```

Het blijkt, dat ik naar nogal obscure schaduw ROM-locaties spring. Deze locaties mogen niet worden aangeroepen met behulp van hook-code 32H, omdat zij gebruik maken van routines die de regel aftasten en terugkeren naar de BASIC, niet naar de aanroepende machinecode.

APPENDIX A

De interface-systeemvariabelen

Behalve de bestaande systeemvariabelen, voegt de interface de volgende 58 bytes toe aan het geheugengebied, met de onderstaande functies:

DECIMAAL	*HEX*	*NAAM*	*GEBRUIK*
23734	5CB6	FLAGS3	bit 0 – aan gedurende uitvoering van uitgebreid BASIC-commando bit 1 – aan gedurende CLEAR # bit 2 – aan indien ERRSP verandert bit 3 – aan gedurende netwerkactie bit 4 – aan gedurende LOAD bit 5 – aan gedurende SAVE bit 6 – aan gedurende MERGE bit 7 – aan gedurende VERIFY
23735	5CB7	VECTOR	Wordt gebruikt om de evaluator uit te breiden – Normaal 01F0

23737	5CB9	SBRT	Routine die gebruikt wordt door schaduw ROM om 16K routines aan te roepen, in de vorm:
			LD HL, value
			CALL routine
			LD (5CBAH), HL
			RET

23747	5CC3	BAUD	RS232 baudwaarde – initieel 000CH, 19200 – ruwweg (3500000/(26*waarde))–2
23749	5CC5	NTSTAT	Stationsnummer van netwerk (1-64)
23750	5CC6	IOBORD	BORDER-kleur gedurende I/O-opdrachten
23751	5CC7	SER-FL	RSR232 werkruimte – eerste byte is een indicator, de tweede een invoerteken
23753	5CC9	SECTOR	Microdrive werkruimte – gewoonlijk om sectoren te tellen, te beginnen vanaf FFH of 04FBH
23755	5CCB	CHADD-	Tijdelijke opslag voor CH-ADD gedurende evaluatie van uitgebreide syntax in BASIC-regels
23757	5CCD	NTRESP	Netwerk antwoordcode – nummer van ontvangende station – het begin van een 8 bytes confirmatieheader in een netwerk
23758	5CCE	NTDEST	Netwerk bestemmingsstationsnummer
23759	5CCF	NTSRCE	Netwerk bronstationsnummer
23760	5CD0	NTNUMB	Netwerk bloknummer (0-65535)
23762	5CD2	NTTYPE	Code van netwerkheadertype – 0 gegevens, 1 EOF
23763	5CD3	NTLEN	Netwerk bloklengte (0-255)
23764	5CD4	NTDCS	Netwerk gegevensblok controlesom
23765	5CD5	NTHCS	Netwerk header controlesom (van NTDEST-NTDCS)

23766	5CD6	D-STR1	Begin van eerste 8 bytes bestandsidentificatie: 2 bytes drivenummer (1-8) of netwerk bestemmingsnummer of baud-waarde
23768	5CD8	S-STR1	Streamnummer (0-15)
23769	5CD9	L-STR1	Channelidentificatie (Grote letter)
23770	5CDA	N-STR1	Lengte van bestandsnaam
23772	5CDC	T-STR1	Begin van bestandsnaam (gewoonlijk in werkruimte)
23774	5CDE	D-STR2	Begin van tweede 8 bytes bestandsidentificatie
23782	5CE6	HD-00	Bestandstype: 0 – programma 1 – numerieke array 2 – string array 3 – bytes
23783	5CE7	HD-0B	Lengte van gegevens
23785	5CE9	HD-0D	Begin van de gegevens
23787	5CEB	HD-0F	Programmalengte
23789	5CED	HD-11	Auto-start regelnummer (ook gebruikt door hook-code 32H)
23791	5CEF	COPIES	Aantal veelvoudige kopieën bij SAVE – teruggezet op 1 na een SAVE

Opmerking: FLAGS3 wordt normaal op nullen gezet zodra de schaduw ROM uit-PAGE-t. Het kan nuttig worden geadresseerd door IY+124.
De parameters in de routine SBRT worden ge-POKE-d door de schaduw ROM. De RETurn gaat naar positie 8, maar er staat altijd een 0000 achter op de stack, om onderscheid te maken bij een fout.
De namen van de variabelen HD-00 tot HD-11 zijn rechtstreeks overgenomen van hun equivalent die worden gebruikt door de 16K ROM cassetteroutines, geadresseerd via IX+0 tot IX+11H.

APPENDIX B

Assemblerlijsten

Deze appendix bevat de Z80 assemblerlijsten van de routines die in dit boek zijn gebruikt. Zij zijn gemaakt met behulp van de HISOFT GENS3 assembler/editor. Hexadecimale waarden worden voorzien van ''#''. De regelnummers zijn alleen van belang voor de assembler en hebben hier geen betekenis. Toelichting is gegeven aan de rechterzijde van de lijsten. De gekozen ORGs zijn willekeurig, daar de meeste routines positie-onafhankelijk zijn. De routines die dynamisch zijn, gebruiken de mogelijkheid dat bij aanroep het BBC-register is geladen met hun startadres.

```
STREAM 14-Z$
```

Deze routine creëert een nieuw channel ''Z'' en kent deze toe aan stream 14, die verondersteld wordt te zijn gesloten. Aangenomen wordt ook dat de extra 58 systeemvariabelen aanwezig zijn. Om deze reden is aan de BASIC-lader het commando CLOSE # toegevoegd, daar dit in beide behoeften voorziet.

```
 10 ;        ;     "DRUK AF IN STRING ROUTINE"
 20 ;        ;     "ROUTINE OM #14 AAN TE MAKEN"
 30 ;        ;     "VOEG TEKENS TOE AAN Z$"
 40 ;        ;     "(C) A.PENNELL 1983"
 50 PROG     EQU   #5C53
 55 VARS     EQU   #5C4B
 60          ORG   40000
 65          ENT   $
 70 SETUP    LD    HL,(PROG)
 80          DEC   HL          ; HL=PROG-1=EIND VAN CHANS
 90          PUSH  BC          ; BEWAAR BEGINADRES
100          PUSH  HL          ; BEWAAR BEGIN NIEUWE GEGEVENS
110
120
130          LD    BC,11       ; 11 BYTES NODIG
140          CALL  #1655       ; MAKE_ROOM
150          POP   DE          ; DE=BEGIN NIEUWE GEGEVENS
160          LD    HL,OUTCH-SETUP
170          POP   BC          ; HAAL SETUP TERUG
180          ADD   HL,BC       ; HL=OUTCH
190
200          PUSH  DE          ; BEWAAR BEGIN GEGEVENS
210          EX    DE,HL
220          LD    (HL),E      ; BEWAAR OUTCH IN NIEUW GEBIED
230          INC   HL
240          LD    (HL),D
250          INC   HL
260          EX    DE,HL
270          LD    BC,LABEL-OUTCH
280          ADD   HL,BC       ; HL=LABEL
290          LD    BC,9        ; 9 EXTRA BYTES GEGEVENS
300          LDIR  ; KOPIEER GEGEVENS NAAR CHANS
310          POP   HL          ; HL=BEGIN VAN GEGEVENS
320          INC   HL
330          LD    BC,(#5C4F)      ; BC=CHANS
```

```
340          AND  A
350          SBC  HL,BC
360          LD   (#5C32),HL    ; BEWAAR STRMS VERPLAATSING
370 ;                           ; IN STRMS VOOR #14
380          LD   BC,0
390          RET  ; TERUG NAAR BASIC MET 0
392 ;                           ; RESTERENDE GEGEVENS NAAR CHANS:
398 LABEL    DEFW #15C4         ; INVOERROUTINE
400          DEFW "Z"           ; NAAM V CHANNEL
410          DEFW #28           ; SCHADUW ROM UITVOER
420          DEFW #28           ; SCHADUW ROM INVOER
430          DEFW 11            ; AANTAL BYTES
435 ;        UITVOER            ROUTINE:
440 OUTCH    PUSH AF            ; BEWAAR TEKENCODE
450          LD   HL,(VARS)
460 L1       LD   A,(HL)
470          CP   #5A
480          JR   Z,FOUND       ; SPRING INDIEN Z$ GEVONDEN
490          CP   #80
500          JP   Z,#0670       ; "VAR NOT FOUND" INDIEN NIET MEER
510          CALL #19B8         ; NEXT_ONE
520          EX   DE,HL         ; HL=BEGIN VOLGENDE VARIABELE
530          JR   L1            ; EINDE LUS
540 FOUND    INC  HL            ; HL=LENGTE STRING LO
550          LD   C,(HL)
560          INC  HL            ; HL=LENGTE STRING HI
570          LD   B,(HL)        ; BC=LENGTE STRING
580          INC  BC            ; VERHOOG LENGTE
590          PUSH BC            ; BEWAAR NIEUWE LENGTE
600          PUSH HL            ; BEWAAR LOKATIE 1STE TEKEN
610          ADD  HL,BC         ; HL=EINDE STRING
620          CALL #1652         ; ONE_SPACE (MAAK RUIMTE VOOR TEKEN)
630          INC  HL            ; HL=NIEUWE RUIMTE
640          EX   DE,HL         ; DE=NIEUWE RUIMTE
650          POP  HL            ; HL=POSITIE 1STE TEKEN
660          POP  BC            ; BC=NIEUWE LENGTE
670          LD   (HL),B        ; BEWAAR NIEUWE LENGTE
680          DEC  HL
690          LD   (HL),C
700          POP  AF            ; PLAATS TEKECODE TERUG
710          LD   (DE),A        ; BEWAAR TEKEN IN STRING
720          AND  A             ; WIS CARRY (VOORKOM FOUTEN)
730          RET  ; TERUG NAAR O/S
```

ON EOF GO TO

Deze routine wijzigt ERRSP zodat het verwijst naar een geschikte foutafhandelings-routine.

```
10 ;         ;"ON EOF GOTO"
20 ;         ;"PIC CODE"
30
40           ORG  40000
50 SETUP     LD   HL,START-SETUP        ; VERANDER ERRSP:
60           ADD  HL,BC     ; HL=START
70           EX   DE,HL     ; DE=START
```

```
 80          LD    HL,(ERRSP)    ;HL=ERRSP
 90          LD    (HL),E     ; BEWAAR NIEUWE FOUTROUTINE
100          INC   HL         ; IN DE RELEVANTE STACK
110          LD    (HL),D     ; POSITIE
120          RST   8          ; KONTROLEER OF INTERFACE
130          DEFB  #31        ; VARIABELEN AANWEZIG ZIJN
140          LD    BC,0
150          RET   ; TERUG NAAR BASIC FOUTAFHANDELING
160 ;                         ; MET 0:
170 START    LD    HL,(ERRSP)     ; HL=STACKPOSITIE
175          LD    A,(#5C3A)      ; A=FOUTCODE
180          CP    EOF        ; WAS HET EEN EOF ?
190          JP    NZ,#1303   ; SPRING NAAR ROM AFHANDELING ALS NIET EOF
200          LD    E,(HL)     ; ANDERS DE=START
210          INC   HL
220          LD    D,(HL)
230          PUSH  DE         ; ZET START OP BODEM STACK
240          CALL  #16B0      ; MAAK WERKVELDEN SCHOON
250          RES   5,(IY-#37)    ; GEEF AAN GEREED VOOR NIEUWE SLEUTEL
260          CALL  #0D6E      ; WIS ONDERKANT SCHERM/ OPEN STREAM 0
270          LD    HL,(#5C45)     ; HL=HUIDIG REGELNUMMER
280          LD    (#5CC9),HL     ; BEWAAR IN SECTOR
290          LD    DE,LINE    ; DE=REGEL OM NAARTOE TE SPRINGEN
300          LD    HL,#5C42   ; HL=NEWPPC
310          LD    (HL),E     ; BEWAAR NIEUW REGELNUMMER
320          INC   HL
330          LD    (HL),D
340          INC   HL
350          LD    (HL),1     ; ZET NSPPC=1 D.I. 1STE OPDRACHT
360          JP    #1B7D      ; SPRONG IN ROM
370
380 ERRSP    EQU   #5C3D
390 EOF      EQU   7          ; EOF FOUTCODE
400 LINE     EQU   1000       ; FOUTREGELNUMMER
410          END
```

OPEN # willekeurige streamroutine

Deze routine opent een meegegeven stream naar een Microdrive channel, maar het bestandstype is niet beperkt tot gegevensbestanden. Het is in principe de OPEN #-opdracht met gewijzigde foutcontroles.

```
 10 ;        ;"OPEN EEN WILLEKEURIGE STREAM
 20 ;
 30 ;
 40 OPENM    EQU   #22        ; HOOK CODES
 50 CLOSM    EQU   #23
 60 MOTOR    EQU   #21
 70          ORG   50000
 75          ENT   $
 80 START    LD    A,(#5CD8)     ; A=S_STR1=STREAMNUMMER
 90          CALL  #1727     ; CALL STR_DATA1
100          LD    HL,17
110          XOR   A
120          SBC   HL,BC
```

```
130         LD   BC,0
140         RET  C              ; TERUG NAAR BASIC MET 0
150 ;                           ; INDIEN STREAM REEDS OPEN
160         LD   (#5CD7),A        ; MAAK D_STR1 HI NUL
170         LD   HL,10
180         LD   (#5CDA),HL       ; LENGTE BESTANDSNAAM=10
190         LD   HL,(#5C7B)
200         LD   (#5CDC),HL       ; BEGIN NAAM= UDG GEBIED
210         LD   A,(#5CD8)        ; A=STREAMNUMMER
220         ADD  A,A            ; BEREKEN RELEVANTE STRM
230         LD   HL,#5C16
240         LD   E,A
250         LD   D,0
260         ADD  HL,DE          ; HL=STREAMLOKATIE
270         EXX
280         PUSH HL             ; BEWAAR H'L'
290         EXX
300         PUSH HL             ; BEWAAR STRM LOKATIE
310         RST  8              ; OPEN TIJDELIJK "M" CHANNEL
320         DEFB OPENM
330         BIT  0,(IX+24)
340         JR   Z,READ         ; SPRING IGV LEESBESTAND
350         XOR  A              ; BESTAND NIET GEVONDEN
360         RST  8              ; ALLE MOTORS UIT
370         DEFB MOTOR
380         RST  8              ; REKLAMEER GEBIED
390         DEFB #2C
400         POP  HL             ; PLAATS STACK TERUG
410         EXX
412         POP  HL             ; ZET H'L' TERUG
414         EXX
420         LD   BC,1
430         RET
440 READ    RES  7,(IX+4) ; MAAK PERMANENT
450         XOR  A
460         PUSH HL             ; BEWAAR STREAM VERPLAATSING
470
480         RST  8              ; ZET ALLE MOTORS UIT
490         DEFB MOTOR
500         POP  DE             ; DE=VERPLAATSING
510         POP  HL             ; HL=LOKATIE
520         LD   (HL),E         ; BEWAAR NIEUWE VERPLAATSING
530         INC  HL             ; IN STRMS GEBIED
540         LD   (HL),D
550         LD   A,(IX+67)        ; A=RECFLG
560         AND  4              ; MASKEER PRINTBESTAND BIT
570         ADD  A,2            ; PLUS 2
580         LD   C,A            ; ZET IN BC
590         LD   B,0
600         EXX
610         POP  HL             ; ZET H'L' TERUG
620         EXX
630         RET  ; KEER TERUG MET 2 IGV PRINT BESTAND
640 ;                           ; OF MET 6 ALS GEEN PRINTBESTAND
650         END
```

Statusroutine

Deze routine is gelijk aan de MOTOR-routine in de schaduw ROM, van adres X182A tot X1871, doch met gewijzigde foutafhandeling. De bedoeling is om te controleren of een Microdrive aanwezig is en of de cartridge beveiligd is tegen wissen. De genoemde labels zijn gebaseerd op de equivalente schaduw ROM-locaties. De ORG bevat de waarde die gebruikt is bij het uittesten – deze code is *niet* positie-onafhankelijk – de BASIC-lader positioneert alle vier de CALLs.
(Ik kan niet stellen dat ik precies weet wat iedere invoer/uitvoer-instructie doet – de opmerkingen zijn slechts giswerk!)

```
 10 ;       STATUS ROUTINE
 20 MOTOR   EQU   #21        ; MOTOR HOOK CODE
 30         ORG   50000      ; OORSPRONKELIJK ADRES
 40         LD    A,(#5CD6)     ; A=D_STR1=DRIVENUMMER
 50         DI    ; INTERRUPT UIT
 60         JR    X182A      ; SPRING HIEROVERHEEN
 70 X1806   LD    HL,#1388 ; HL=5000
 80 X1809   DEC   HL
 90         LD    A,L
100         OR    H
110         JR    NZ,X1809 ; WACHT EVEN
120         LD    HL,#1388
130 X1811   LD    B,6
140 X1813   IN    A,(#0EF) ; LEES POORT EF
150         AND   4          ; ALLEEN BIT 2
160         JR    NZ,X1820 ; SPRING ALS DRIVE NIET AANWEZIG
170         DJNZ  X1813      ; DOE 6 KEER
180         JR    PRESN      ; AANWEZIG
190 X1820   DEC   HL         ; VERLAAG TELLER
200         LD    A,H
210         OR    L
220         JR    NZ,X1811 ; PROBEER 5000 MAAL
230         LD    BC,0       ; KAN ZO NIET WORDEN AANGESLOTEN
240         JR    NOTPR      ; KEER TERUG MET 0
250 X182A   LD    DE,#0100 ; D=1,E=0
260         NEG   ; VERANDER TEKEN DRIVENUMMER
270         ADD   A,9        ; A=9-DRIVENUMMER
280         LD    C,A        ; C=GEKOZEN DRIVE
290         LD    B,8        ; B=DRIVETELLER
300 X1835   DEC   C          ; VERLAAG GEKOZEN DRIVE
310         JR    NZ,X184B ; SPRING IGV ONDOORZOCHTE DRIVE
320         LD    A,D
330         LD    (#0F7),A ; STUUR 1 NAAR POORT F7 - AAN
340         LD    A,#0EE
350         OUT   (#EF),A  ; STUUR EE NAAR POORT EF
360         CALL  X1867      ; WACHT EVEN
370         LD    A,#EC
380         OUT   (#EF),A  ; STUR EC NAAR POORT EF
390         CALL  X1867      ; WACHT EVEN
400         JR    X185C      ; DOE VOLGENDE
410 ;       ; SCHAKEL DRIVE UIT
420 X184B   LD    A,#EF
430         OUT   (#EF),A  ; STUUR EF NAAR POORT EF
440         LD    A,E
```

```
450          OUT   (#F7),A   ; STUR 0 NAAR PORT F7 - UIT
460          CALL  X1867     ; WACHT EVEN
470          LD    A,#ED
480          OUT   (#EF),A   ; STUUR ED NAAR POORT EF
490          CALL  X1867     ; WACHT EVEN
500 X185C    DJNZ  X1835     ; DOE VOOR IEDER VAN 8 DRIVES
510          LD    A,D
520          OUT   (#F7),A   ; STUR 1 NAAR POORT F7
530          LD    A,#EE
540          OUT   (#EF),A   ; STUR EE NAAR PORT EF
550          JR    X1806     ; KONTROLEER STATUS
560 X1867    PUSH  BC        ; VERTRAAG ROUTINE
570          PUSH  AF        ; BEWAAR REGISTERS
580          LD    BC,#87    ; BC=135
590 X18FB    DEC   BC
600          LD    A,B
610          OR    C
620          JR    NZ,X18FB  ; 135 MAAL
630          POP   AF        ; PLAATS REGISTERS TERUG
640          POP   BC
650          RET
660 PRESN    IN    A,(#EF)   ; DRIVE AANWEZIG DUS...
670          AND   1         ; KONTROLEER SCHRIJFBEVEILIGINGNOKJE
680          LD    BC,1
690          JR    NZ,NOTPR  ; SPRING MET 1 IGV NOKJE AANWEZIG
700          INC   BC        ; ANDERS 2
710 NOTPR    PUSH  BC        ; BEWAAR TERUGKEER WAARDE
720          XOR   A
730          RST   8
740          DEFB  MOTOR     ; SCHAKEL ALLE MOTORS UIT
750          POP   BC        ; PLAATS WAARDE TERUG
760          RET
770          END
```

ON ERROR GO TO

Deze routine wijzigt ERR-SP zodat het naar een nieuwe foutafhandeling verwijst. Dit alleen is niet voldoende voor interface-fouten – hiervoor dient bit 2 van FLAGS3 ook te worden aangezet, omdat anders de gewone foutafhandeling zou worden aangeroepen. Indien een fout optreedt met de schaduw ROM ingePAGEd, bevat HL de waarde #81.

```
 10 ;        ON ERROR GOTO
 20 ;        INTERFACE VERSIE
 30 ERRSP    EQU   #5C3D     ; ZET KONSTANTEN
 40 SECTR    EQU   #5CC9
 50 LINE     EQU   1000      ; REGELNUMMER WAAR IGV FOUT WORDT
 51 ;        ;     HEEN GESPRONGEN
 60          ORG   50000     ; OORSPRONKELIJK ADRES
 65 SETUP    LD    HL,START-SETUP
 70          ADD   HL,BC     ; HL=START
 80          EX    DE,HL     ; DE=START
 90          LD    HL,(ERRSP)    ; HL=FOUTSTACKPOSITIE
100          LD    (HL),E    ; BEWAAR NIEUWE FOUTAFHANDELING
110          INC   HL        ; OP DE STACK
120          LD    (HL),D
```

```
130         RST  8
140         DEFB #31        ; CREEER EXTRA VARIABELEN
150         LD   BC,0
165 ;       ;    DE NIEUWE FOUTAFHANDELING:
170 START   LD   (SECTR),HL    ; BEWAAR HL WAARDE
180         LD   HL,(ERRSP)
190         LD   DE,#1303 ; DE=OUDE ROM AFHANDELING
200         PUSH DE         ; PLAATS OP STACK OM VOLGENDE KEER
201 ;       ;    FUNKTIE ONMOGELIJK TE MAKEN
220         CALL #16B0      ; MAAK VERSCHILLENDE GEBIEDEN SCHOON
230         RES  5,(IY+#37)    ; ZET SLEUTELBIT AF IN FLAGS
240         CALL #0D6E      ; MAAK ONDERSTE SCHERM SCHOON
241 ;       ;    EN OPEN STREAM 0
250         LD   HL,#17B9 ; HL=SCHADUW ROUTINE OM ALLE
251 ;       ;    TIJDELIJKE CHANNELS TE REKLAMEREN
252 ;       ;    EN ALLE MOTORS UIT TE SCHAKELEN
260         LD   (#5CED),HL    ; BEWAAR IN HD_11
270         LD   A,(#5C3A)     ; A=FOUTNUMMER (OUDE FOUT)
280         LD   (23763),A     ; BEWAAR IN NTLEN
290         PUSH AF         ; BEWAAR FOUTCODE
300         RST  8
310         DEFB #32        ; ROEP AAN X17B9
320         LD   HL,#0081
330         LD   DE,(SECTR)    ; DE=WAARDE VAN HL NA FOUT
340         POP  AF         ; A= OUDE ROM FOUTCODE
350         AND  A          ; ZET CARRY UIT
360         SBC  HL,DE
370         JR   NZ,OLD     ; SPRING INDIEN HL <> #81 D.I.
372 ;       ;    OUDE ROM FOUT
380         BIT  0,(IY+124)    ; KONTROLEER FLAGS3
390         JR   NZ,OLD     ; NAAR EEN OUDE FOUT INDIEN
391 ;       ;    IN MIDDEN VAN EEN OPDRACHT
400         CP   7
410         JR   Z,OLD      ; SPRING IGV EOF
420         LD   A,100      ; INTERFACE FOUT DUS BEWAAR
430         LD   (23763),A     ; 100 IN NTLEN
440         LD   A,"?"
450         RST  #10        ; DRUK EEN "?" AF
460         JR   PRNTN      ; DRUK DAN REGELNUMMERS ENZ.
465 ;       ;    OUDE ROM FOUT
470         INC  A
480         LD   B,A        ; BEWAAR FOUTNUMER IN B
490         CP   10
500         JR   C,TWEE
510         ADD  A,7        ; MAAK BOODSCHAP TEKEN
520 TWEE    CALL #15EF      ; DRUK AF
530         LD   A," "
540         RST  #10        ; DRUK EEN SPATIE AF
550         LD   A,B
560         LD   DE,#1391
570         CALL #0C0A      ; DRUK FOUTBERICHT AF
580 PRNTN   XOR  A
590         LD   DE,#1536
600         CALL #0C0A      ; DRUK "," AF
610         LD   BC,(#5C45)    ; BC=PPC=HUIDIGE REGEL
```

```
620         LD   (SECTR),BC    ; BEWAAR IN SECTOR
630         CALL #1A1B     ; DRUK REGELNUMMER AF
640         LD   A,":"
650         RST  #10       ; DRUK ":" AF
660         LD   C,(IY+13)
670         LD   B,0
680         CALL #1A1B     ; DRUK OPDRACHTNUMMER AF
690         LD   HL,#5C3B ;HL=FLAGS
700         RES  5,(HL)
710         EI   ; ACTIVEER INTERRUPT
720 WAIT    BIT  5,(HL)
730         JR   Z,WAIT    ; WACHT OP TOETSAANSLAG
740         LD   HL,#5C42 ; HL=NEWPPC
750         LD   DE,LINE   ; DE=REGEL OM NA FOUT NAARTOE
            TE SPRINGEN
760         LD   (HL),E    ; BEWAAR REGEL
770         INC  HL
780         LD   (HL),D
790         INC  HL
800         LD   (HL),1    ; MAAK ER OPDRACHT 1 VAN
810         LD   (IY+0),255    ; WIS DE FOUT
820         LD   (IY+124),0    ; WIS FLAGS3
830         CALL #0D6E     ; MAAK ONDERSTE SCHERM SCHOON
840         JP   #1B7D     ; SPRING NAAR 16K ROM
850         END
```

RS232 TAB-routine

Deze routine verandert het "P"-channel opdat LPRINT ... enz. worden verzonden naar de RS232-poort, zoals dit het geval is bij het "T"-channel, maar voorzien van de TAB-functie. Drie eigen variabelen worden gebruikt – WIDTH = is breedte van de printerwagen, POSN = huidige positie van printkop en CONTR = een indicator. Als de TAB wordt geïnterpreteerd, wordt eerst een teken 23 verstuurd, daarna resp. LSB en MSB.

```
 10 ;       "RS232 TAB ROUTINE
 20         ORG  23296     ; LOKATIE IN PRINTER BUFFER
 30 SETUP   LD   HL,(23631)    ; HL=CHANS
 40         LD   BC,15
 50         ADD  HL,BC     ; HL= CHANNEL "P" GEBIED
 60         LD   DE,START  ; DE=NIEUWE UITVOERROUTINE
 70         LD   (HL),E    ; BEWAAR NIEUWE ROUTINE
 80         INC  HL        ; LOKATIE IN "P" GEBIED
 90         LD   (HL),D
100         LD   BC,0
110         LD   HL,POSN
120         LD   (HL),B    ; ZET POSN OP NULLEN
130         INC  HL
140         LD   (HL),B    ; ZET CONTR OP NULLEN
150         RET  ; KEER TERUG MET 0
155 ;                      ; DE UITVOERROUTINE: A=CODE
160 START   CP   " "
170         JR   NC,NORM   ; SPRING IGV >= " "
180         CP   13
190         JR   NZ,NNL    ; SPRING ALS GEEN NEWLINE
```

```
200         LD   HL,CONTR ; NEWLINE:
210         BIT  0,(HL)   ; KONTR. AUTO FLAG
220         RES  0,(HL)   ; WIS AUTOFLAG
230         RET  NZ       ; GEEN NEWLINE ALS AAN
240 NEWLI   LD   HL,CONTR
250         RES  0,(HL)   ; WIS AUTOFLAG
260         DEC  HL       ;  HL=POSN
270         LD   (HL),0   ; WIS POSN
280         LD   A,13
290         CALL OUTCH    ; STUUR N/L (13)
300         LD   A,10
310         JP   OUTCH    ; STUUR L/F (10)
320 NNL     CP   23       ; IS DIT EEN TAB TEKEN ?
330         CCF
340         RET  NZ       ; TERUG INDIEN GEEN TAB
350         LD   DE,TAB2  ; TAB KOMMANDO:
360 REDO    LD   HL,(#5C51)    ; HL=CURCHL
370         LD   (HL),E   ; BEWAAR DE IN CURCHL, D.I.
380         INC  HL       ; WIJZIG UITVOER ROUTINE NAAR DE
390         LD   (HL),D
400         RET  ; TERUG
402 ;           KOMT HIER MET LSB OF TAB POSITIE:
410 TAB2    LD   (#5C0F),A     ; BEWAAR LSB IN TVDATA2
420         LD   DE,TAB3
430         JR   REDO     ; MAAK UITVOERROUTINE TAB3
435 ;                     ; KOMT HIER MET MSB OF TAB POSITIE:
440 TAB3    LD   DE,START
450         CALL REDO     ; ZET UITVOER NAAR NORMAAL
460         LD   A,(#5C0F)     ; A=TABPOSITIE
470         LD   D,A      ; BEWAAR IN D
480 TAB4    LD   HL,WIDTH
490         SUB  (HL)     ; A=TAB-WIDTH
500         JP   NC,#046C ;"INT OUT OF RANGE" INIDIEN WAGEN NIET
505 ;                     ; BREED GENOEG
510         INC  HL       ; HL=POSN
520         LD   A,D      ; A=TAB
530         SUB  (HL)     ; A=TAB-POSN
540         PUSH DE       ; SAVE D
550         CALL C,NEWLI  ; ALS HET NIET PAST, N/L
560         POP  DE       ; ZET D TERUG
570         LD   A,D      ; A=TAB
580         SUB  (IY+POSY)  ;  ZET AUTO FLAG AAN
590         RET  Z        ; KEER TERUG ALS IN JUISTE PLAATS
600         LD   B,A      ; B=AANTAL BENODIGDE SPATIES
610 TABB    LD   A," "
620         PUSH BC       ; BEWAAR BC
630         EXX  ; VERWISSEL REGISTERS
640         RST  #10      ; DRUK SPATIE AF
650         EXX  ; VERWISSEL REGISTERS
660         POP  BC       ; ZET BC TERUG
670         DJNZ TABB     ; DRUK GEWENSTE AANTAL SPATIES AF
680         RET  ; TERUG INDIEN KLAAR
685 ;                     ; KOMT HIER MET 'NORMALE' CODE
690 NORM    CP   #A5
700         JR   C,NORM2  ; GEEN "OPDRACHT"
```

```
710         SUB   #A5
720         JP    #0C10      ; CONVERTEER OPDRACHT
730 NORM2   RES   0,(IY+CONY)   ; WIS AUTO FLAG
740         LD    HL,#5C3B ; HL=FLAGS
750         RES   0,(HL)     ; WIS 'VOORAFGAANDE SPATIE ' INDIKATIE
755 ;                        ; ROM DOET DIT NL NIET
760         CP    " "
770         JR    NZ,NSPAC ; LAAT ZO INDIEN GEEN SPATIE
780         SET   0,(HL)     ; ANDERS INDIKATIE AANZETTEN
790 NSPAC   CP    128
800         JR    C,OUTC2  ; DRUK TEKE AF INDIEN NIET GRAPHICS
810         LD    A,"?"      ; GRAPHICS TEKEN, DRUK "?" AF
820 OUTC2   CALL  OUTCH      ; STUUR BYTE NAAR PRINTER
830         LD    HL,POSN
840         INC   (HL)       ; VERHOOG POSN
850         LD    A,(HL)     ; A=NIEUW POSN
860         DEC   HL         ; HL=WIDTH
870         CP    (HL)       ; VERGELIJK MET WIDTH
880         RET   NZ         ; TERUG INDIEN ONGELIJK
890         CALL  NEWLI      ; EINDE REGEL, DUS DOE N/L
900         SET   0,(IY+CONY)   ; ZET AUTOFLAG AAN
910         RET
915 ;                        ; STUUR TEKEN NAAR RS232
920 OUTCH   RST   8          ; HOOKCODE
930         DEFB  #1E        ; 2320P BYTE
940         RET   ; KEER TERUG
945 ;                        ; KONSTANTEN
950 WIDTH   EQU   23540      ; PRINTER BUFFER
960 POSN    EQU   WIDTH+1
970 CONTR   EQU   WIDTH+2
980 POSY    EQU   -69        ; IY = VERPLAATSING
990 CONY    EQU   -68
1000        END
```

APPENDIX C

Interface-fouten

Met de interface aangesloten, wordt gebruik gemaakt van een aanvullende 8K ROM ten behoeve van de nieuwe mogelijkheden. Echter, men moet zich bewust zijn van bepaalde problemen die kunnen ontstaan tengevolge van fouten (zgn. 'bugs') met name in de oude ROM.

1. Syntaxcontrolefout
Bij het schrijven van de oorspronkelijke 16K BASIC ROM, is incorrect geanticipeerd op de syntax van de interface commando's. Helaas wordt deze incorrecte syntax geaccepteerd door de evaluator, maar kan niet worden uitgevoerd. De incorrecte syntaxformaten zijn:

`ERASE <string>`	bijv. ERASE "test"
`MOVE <string>,<string>`	bijv. MOVE "a", "b"
`FORMAT <string>`	bijv. FORMAT "test"
`CAT` (geen nummer)	

2. Kleurcommando's
Ook al is dit niet echt een fout die wordt veroorzaakt door de ROM in de interface, wordt het probleem genoemd in Hoofdstuk 3, met betrekking tot de kleurcommando's veroorzaakt door een slordig stuk codering in de 16K BASIC. Om dit te verhelpen, laat u ieder kleurcommando voorafgaan door PRINT;

3. Inconsequente BREAK
In oude ROM-machines bestaat een algemene inconsequentie in de wijze van het onderbreken van de interface-uitvoeringen. Gedurende Microdrive-opdrachten en RS232-uitvoer, moeten CAPS SHIFT en SPACE beide worden ingedrukt, doch gedurende netwerk-opdrachten en RS232-invoer is SPACE alleen al voldoende. In de nieuwe ROM moeten beide toetsen worden ingedrukt.

4. Cassette-uitvoeringen
Indien een bestandsnaam in een cassetteopdracht fout is op de een of andere manier (bijv. langer dan 10 tekens), dan wordt de foutboodschap (**Invalid filename**) overschreven door het zinloze **Nonsense in BASIC.**

5. Lange BASIC LOAD
Indien een poging wordt ondernomen om een lang BASIC-programma te laden waar onvoldoende geheugenruimte voor bestaat, kan het systeem blijven hangen, terwijl de motor van de Microdrive blijft draaien. Helaas bestaat dezelfde fout ook in de nieuwe ROM.
De volgende problemen gelden alleen voor de oude ROM – zij zijn alle verbeterd in de nieuwe ROM:

6. Machinecode hook-codes
Twee van de machinecode hook-codes werken helaas niet – READ-N is onbruik-

baar vanwege een overschreven carry flag en MAKE-M werkt hetzelfde als OPEN-M vanwege een incorrecte aanroep van de hook-code sprongentabel op positie X19C9H.

7. RS232 dubbele spaties
Wanneer gebruik wordt gemaakt van een "t"-channel om een programmalijst af te drukken, wordt een dubbele spatie afgedrukt tussen de sleutelwoorden bijv. PRINT PAPER of THEN PRINT. Dit wordt veroorzaakt door het falen in het gebruik van bit 0 in FLAGS.

8. CLOSE #-probleem
Indien u een BREAK doet tijdens een CLOSE #-opdracht die verwijst naar een interface channeltype, dan wordt het door de stream gebruikte geheugengebied niet gereclameerd. Dit kan grote hoeveelheden geheugen in beslag nemen en kan niet eenvoudig worden vrijgegeven. Een CLEAR # zal het niet vrijgeven, alleen een NEW. Het is gecorrigeerd op de niewe ROM door de instructie **SET 7, (HL)** op positie X1741H toe te voegen, zoals in de eerste Engelse editie van dit boek werd gesuggereerd.

9. Lage RAMTOP-probleem
Indien een SAVE *"m" of LOAD *"m" wordt geprobeerd met een te lage RAMTOP, dan kan het systeem gaan hangen terwijl de motor blijft draaien. Dit is gecorrigeerd in de nieuwe ROM-versie, door het verbeteren van de controle op vrij geheugen.

10. Netwerkprobleem
Indien een SAVE *"N" wordt gedaan en de vierde byte in de laatste "N"-buffer heeft toevallig de waarde 205, dan gaat de machine hangen. Dit wordt veroorzaakt door een verminkte waarde in het IX-register. Gelukkig is de kans dat dit optreedt erg klein.

BIBLIOGRAFIE

Sybex Inc.
Programming the Z80
Rodney Zaks

Melbourne House Ltd.
The Complete Spectrum ROM Disassembly
Dr. Ian Logan en Dr. Frank O'Hara

Kluwer Technische Boeken
ZX Spectrum
A. Sickler

Zakboekje voor de ZX Spectrum
W. Akkermans

Leren omgaan met de ZX Spectrum
B. Baarda/A. van Londen

ZX Spectrum hardware-boek
A. Dickens

BASIC-computerspellen voor de ZX Spectrum
M. Th. A. M. Vijftigschild

Praktijkboek voor de ZX Spectrum
L. Smeesters

Machinetaal voor de ZX Spectrum
W. Tang

BASIC-programma's voor de ZX Spectrum
P. Williams

ZX Spectrum machinetaalroutines
A. Hewson

Programmeercursus BASIC, ZX Spectrum
T. Mervyn

Avonturen in BASIC, ZX Spectrum
T. de Havas

Dit boek bevat alle informatie die benodigd is om optimaal met de ZX Microdive te kunnen werken. Dankzij de heldere uitleg en de vele voorbeelden is deze uitgave voor zowel de beginnende programmeur in BASIC als de doorgewinterde machinetaalprogrammeur zeer de moeite waard. De standaardmogelijkheden van de Interface I worden uiteengezet. Daarnaast wordt veel aanvullende informatie gegeven die de gebruiksmogelijkheden van deze interface aanzienlijk vergroot. Verder zijn er een aantal machinetaalroutines opgenomen die, ook zonder enige kennis van machinetaal, goed te gebruiken zijn. Voor degenen die wel over deze kennis beschikken is er een hoofdstuk over het gebruik van interface ROM-routines, evenals assembler-listings van alle machinetaalroutines in dit boek. Tenslotte wordt uitgebreid aandacht besteed aan de RS232-poort, het gebruik van een netwerk en het toevoegen van extra BASIC-commando's. Dit boek is hiermede onmisbaar geworden voor iedere Spectrum-programmeur die zijn of haar computer als serieuze machine wil gebruiken.

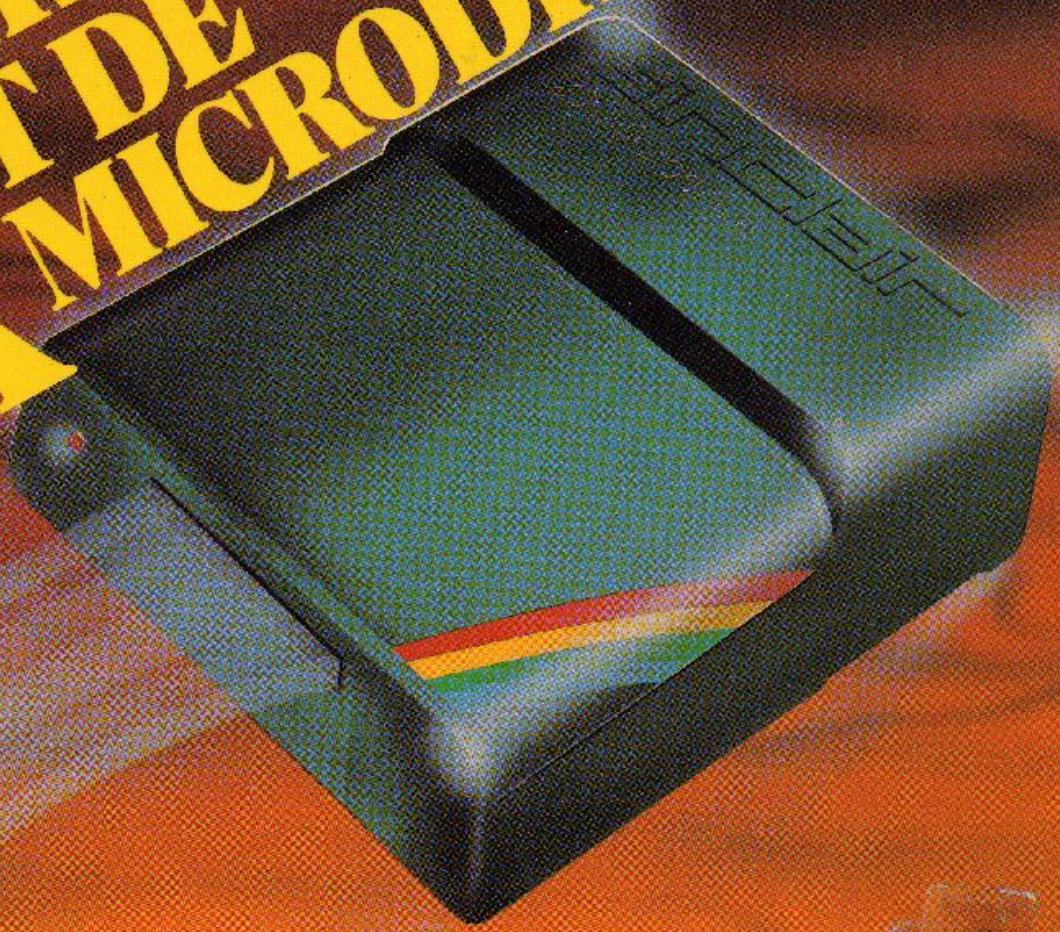
isbn 90 201 1856 0

WERKEN MET DE ZX MICRODRIVE

Andrew Pennel

Andrew Pennel

WERKEN MET DE ZX MICRODRIVE



Kluwer Software Reeks

WERKEN MET DE ZX MICRODRIVE

Andrew Pennel