Paulissen

Het cassette boek

voor de Commodore 64 en de VIC 20

Commodore Bibliotheek 11

DATA BECKER
NEDERLANDS*

Paulissen

Het cassette boek

voor de Commodore 64 en de VIC 20

Commodore Bibliotheek 11

DATA BECKER
NEDERLANDS

De keuze en opbouw van de programma's die in dit boek voorkomen, zijn gebaseerd op hun educatieve waarde. Alle programma's zijn getest op hun juiste werking. De uitgever kan geen enkele verantwoordelijkheid voor eventueel toch optredende fouten aanvaarden.

Uit deze uitgave mag niets worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door:
A.W. Bruna & Zoons Uitgeversmij b.v.,
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.
Antwerpsesteenweg 29a, 2630 Aartselaar.

Oorspronkelijke titel: Das Cassettenbuch zu Commodore 64 und VC-20
Copyright (c) 1984 DATA BECKER GmbH, Düsseldorf.
Voor de Nederlandse vertaling:
Copyright (c) 1985 Uitg. A.W. Bruna & Zoon, Utrecht
Vertaling: Henk Drost, Groningen
Redaktie: Casper Bosveld, Jan Walburg
Produktie: ProCread, Groningen

ISBN 90 229 3336 9

D/1985/0939/97

INHOUDSOPGAVE

1	**INLEIDING**	7
2	**COMMANDO'S VOOR HET BEDIENEN VAN DE RECORDER**	8
	2.1 SAVE	8
	2.2 LOAD	11
	2.3 VERIFY	12
	2.4 OPEN	12
	2.5 PRINT# en CLOSE	13
	2.6 INPUT#	14
	2.7 GET#	20
3	**HET SECUNDAIRE ADRES**	22
4	**DE STATUSVARIABELE**	24
	4.1 Programma's redden na een LOAD ERROR	25
5	**LADEN EN OPSLAAN VANUIT HET PROGRAMMA**	28
	5.1 OVERLAY-techniek	29
6	**HET OPSLAAN VAN MACHINETAALPROGRAMMA'S**	32
7	**DE CASSETTEBUFFER**	34
	7.1 De inhoudsopgave van een cassette maken	36
	7.2 Vermelding van de gevonden databestanden	38
	7.3 Zelfstartende programma's en programmabescherming	39
8	**HET FORMAT VAN DE OPSLAG OP CASSETTE**	44
9	**HET KOPPELEN VAN BASIC-PROGRAMMA'S**	46
10	**EEN PROGRAMMA BEDIENT DE CASSETTERECORDER**	49
11	**DE HARDWARE VAN DE CASSETTERECORDER**	55
	11.1 De verzorging van de cassetterecorder	55
	11.2 De keuze van en de zorg voor de cassettes	56
	11.3 De werkwijze van de cassetterecorder	56
	11.4 Een luidspreker voor de cassetterecorder	58
	11.5 Kopinstelling	59
	11.6 Andere cassetterecorders voor het opslaan van data	65
12	**EEN NIEUW GEBRUIKSSYSTEEM VOOR DE CASSETTERECORDER: "FAST-TAPE"**	65 67
	12.1 Programmabeschrijving	87
13	**DATAVERWERKING MET FAST-TAPE**	95
	13.1 Programmabeschrijving	114

14 CASSETTE-INHOUD EN CATALOGUS VOOR FAST-TAPE CASSETTES 120

15 BACK-UP VAN DISK OP CASSETTE EN OMGEKEERD 122
15.1 Back-up van cassette op disk 122
15.2 Programmabeschrijving back-up cassette-disk 128
15.3 Back-up van diskette naar cassette 129
15.4 Programmabeschrijving back-up disk naar cassette 150

BIJLAGE 151

1 INLEIDING

Als u een computer heeft gekocht, merkt u al gauw, dat het apparaat als u het uitschakelt alles vergeet, wat u het met zoveel moeite heeft bijgebracht. Maar u wilt natuurlijk de gegevens die u in de computer heeft ingevoerd, bewaren. Omdat het hele bestand dat de computer in het geheugen heeft, uit nullen en enen bestaat, is het eenvoudig te coderen en magnetisch op te slaan, bijvoorbeeld op een cassettebandje. Om de computer weer te kunnen laten lezen wat hij heeft opgeslagen, moet alles echter in een vast format worden vastgelegd. Het opgeslagen bestand moet bovendien van bepaalde hulpvariabelen zijn voorzien. Om dit te waarborgen heeft de computer een ingebouwd programma, dat dit allemaal regelt. Als u echter de handleiding bekijkt, zult u zien dat daar weinig in staat over de manier waarop de Commodore 64 en de VIC 20 met de cassetterecorder werken en hoe u deze moet bedienen. Dit cassetteboek wil u dan ook wegwijs maken in het werken met de computer en de cassetterecorder. Aan de hand van vele voorbeeldprogramma's wordt u duidelijk gemaakt dat u met de cassetterecorder aanzienlijk meer kunt beginnen, dan het handboek doet vermoeden. Als kroon op het werk vindt u in het tweede gedeelte van het boek een compleet nieuw besturingssysteem voor de data- en programma-opslag met de cassetterecorder. Dit systeem werkt aanzienlijk sneller dan het ingebouwde besturingssysteem en zelfs sneller dan de diskdrive. Dit programmapakket toont aan, dat voor een homecomputer het gebruik van een cassetterecorder voldoende is, mits u maar de juiste software gebruikt.

Een wenk voor het gebruik van dit boek:

Alle in dit boek gepubliceerde programma's zijn zowel op de Commodore 64 als op de VIC 20 te gebruiken. Omdat het echter niet mogelijk is bij alle programma's een programma te schrijven dat op beide computers bruikbaar is, zijn er steeds 2 versies afgedrukt. Bij machinetaal-programma's zijn altijd een assembler-listing, een BASIC-lader voor de Commodore 64 en een BASIC-lader voor de VIC 20 opgenomen om het gebruik te vergemakkelijken.

In de BASIC-programma's zijn alle besturingsaanwijzingen vertaald en tussen rechte haken geplaatst. Als u dus HOME,RVS ON leest, moet u op de CLR/HOME-toets drukken voor HOME en daarna op de CTRL-toets met toets 9 voor RVS ON.

2 COMMANDO'S VOOR HET BEDIENEN VAN DE RECORDER

Commodore-BASIC heeft voor het bedienen van de cassetterecorder 8 commando's beschikbaar.

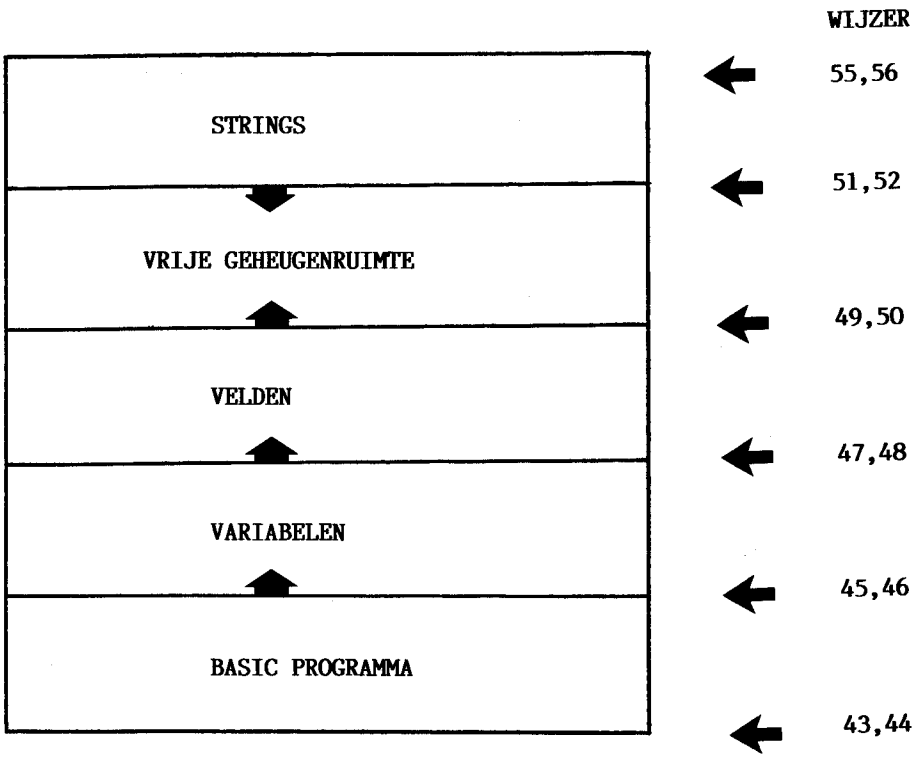
1. SAVE programma opslaan
2. LOAD programma laden
3. VERIFY programma testen
4. OPEN bestand openen
5. CLOSE bestand sluiten
6. PRINT# uitdrukking versturen
7. INPUT# uitdrukking inlezen
8. GET# enkele byte lezen

2.1 SAVE

Met dit commando laat u de computer het in het werkgeheugen aanwezige programma naar een extern geheugen overbrengen. Om te weten op welke plaats het programma zich in het werkgeheugen bevindt, zijn er in uw computer enige geheugenplaatsen waarin de adressen staan van het begin en einde van een programma.

De computer maakt namelijk direkt na het inschakelen een "kladblok". Dit kladblok telt vier pagina's (0-3), zogenaamde "pages", en zit vooraan in het geheugen. Door de geheugensturing van de processor worden er steeds 256 opeenvolgende geheugenplaatsen als pagina ingedeeld omdat iedere geheugenplaats of byte precies 256 (0-255) verschillende waarden kan aannemen. Op pagina nul, van 0 tot en met 255, legt de computer de waarden vast die hij vaak gebruikt.

BASIC RAM VERDELING EN WIJZERS



Afbeelding 1

Algemene opbouw en wijzer van het BASIC RAM

Op de plaatsen 43 en 44 staat steeds de start van het BASIC-geheugen en op de plaatsen 45 en 46 het einde van een BASIC-programma.

Kijkt u het maar na. Toets PRINT PEEK(43), PEEK(44), PEEK(45), PEEK(46) in, dan verschijnt er, als u geen programma heeft geladen, op uw beeldscherm:

1 8 3 8

De betekenis van de afzonderlijke wijzers kunt u in afbeelding 1 zien.

Omdat de computer 65535 geheugenplaatsen telt, worden adressen steeds door 2 bytes weergegeven. De eerste byte, de LSB (Least Significant Byte, byte met de laagste waarde), geeft de positie op een pagina aan; de tweede byte, de MSB (Most Significant Byte, byte met de hoogste waarde), het paginanummer.

Ons BASIC-bereik begint op pagina 8 op de eerste geheugenplaats, dus op geheugenplaats 8*256+1 = 2049.

U zult zich nu misschien afvragen, waarom in de geheugenplaatsen 45 en 46 de waarden 3 en 8 staan, terwijl er geen programma in het geheugen zit. De reden daarvan is, dat de computer altijd twee nullen toevoegt om aan te geven wanneer een programma afgelopen is. U heeft dus een programma in het geheugen dat alleen maar uit een aanduiding van het eind van het programma bestaat. U kunt dit gemakkelijk nagaan. Als u

```
PRINT PEEK(PEEK(45)+PEEK(46)*256-1)
```

intoetst, krijgt u 0. U voert nu een BASIC-regel in, bijvoorbeeld

```
10 PRINT "HALLO"
```

en sluit de invoer met RETURN. Als u nu de inhoud van de geheugenplaatsen 43, 44, 45, 46 opzoekt, krijgt u:

```
1       8       16       8
```

De computer heeft de aantekeningen in zijn kladblok veranderd volgens uw programma.

We keren nu naar het SAVE-commando terug. Door dit commando laten we de computer alles wat tussen de zojuist genoemde wijzers staat, op cassetteband opslaan. Het maakt voor de computer geen verschil of er al wat op de band staat. Net als bij een normale cassetterecorder worden de nieuwe gegevens over de oude heen opgenomen. Om het extern opgeslagen programma weer terug te kunnen vinden, moet u het een naam geven. De instructie en schrijfwijze hiervoor zijn:

```
SAVE "NAAM"
```

U kunt ook de string "NAAM" tevoren in een stringvariabele inlezen en een programma met de volgende syntaxis extern opslaan:

```
A$="NAAM"
SAVE A$
```

Deze mogelijkheid kan belangrijk zijn als u dit commando binnen programma's gebruikt. Maar hierover vindt u meer in hoofdstuk 5.

Na de naam kunt u ook nog een apparaatadres en een secundair adres laten volgen, steeds door komma's gescheiden. De precieze betekenis van het secundaire adres vindt u in hoofdstuk 3 beschreven. Het apparaatadres vertelt de computer welke randapparatuur hij moet kiezen. Het is als het ware het huisnummer van de apparatuur. Het nummer van de cassetterecorder is 1, dat van een diskdrive 8 of 9. Om u typewerk te besparen is uw computer zo geprogrammeerd, dat hij steeds de cassetterecorder kiest als er geen apparaatadres wordt opgegeven.

2.2 LOAD

Met dit commando kunt u extern opgeslagen gegevens weer in het werkgeheugen laden. U kunt bij dit commando net als bij SAVE een naam opgeven. Geeft u de computer geen naam op, dan laadt hij het eerste programma dat hij aantreft. Voor het apparaatadres en het secundaire adres geldt hetzelfde als bij het SAVE-commando.

Na elk LOAD-commando in direkte vorm voert de computer automatisch de aantekeningen in zijn notitieschrift in volgens het geladen programma. Als u bijvoorbeeld een machinetaalprogramma invoert dat aan het eind van het BASIC-geheugen ligt, worden ook de BASIC-eindvectoren aan het BASIC-RAM-eind geplaatst. Een verdere invoer wordt daarna met "?OUT OF MEMORY ERROR" door de computer becommentarieerd. Om verder gegevens te kunnen invoeren moet u of in de wijzers 45 en 46 de oude waarde POKEn of u geeft een NEW-commando om het programma in het BASIC-geheugen, maar niet het machinetaalprogramma, te wissen. U kunt het NEW-commando vermijden door het machinetaalprogramma in de programma-modus in te voeren. Hierover leest u meer in hoofdstuk 5.

2.3 VERIFY

Dit commando werkt net zo als het LOAD-commando. Het enige verschil is, dat door dit commando het programma niet in het externe geheugen wordt weggeschreven, maar byte voor byte met het programma in het externe geheugen wordt vergeleken. Daardoor kunt u testen of het SAVEn van een programma is gelukt. Was het SAVEn succesvol en bevatte de cassette geen fouten, dan verschijnt er OK op het beeldscherm. Wordt er bij het vergelijken een fout geconstateerd, dan krijgt u de mededeling "?VERIFY ERROR".

2.4 OPEN

Tot nu toe hebben we het alleen maar over het laden en opslaan van programma's gehad. U kunt echter ook data opslaan en laden. Om de computer te vertellen dat u data wilt laden en opslaan, moet u met het OPEN-commando een databestand openen. Met dit commando vertelt u de computer alle parameters die hij nodig heeft. Dit zijn: het apparaatnummer, de filenaam en het secundaire adres. Dit laatste vertelt of er moet worden geladen of opgeslagen. Om een databestand te openen moet u het volgende commando invoeren:

```
OPEN 1f,AA,SA,"NAAM"
```

Met "NAAM" kunt u het databestand net als bij SAVE en LOAD een naam geven.

'1f' staat voor logisch filenummer. Dit getal tussen 0 en 255 stelt de index van deze OPEN-aanwijzing voor. Om te voorkomen dat bij elk in- en uitvoercommando alle parameters mee moeten worden overgedragen, legt de computer een tabel aan waarin de parameters van de OPEN-aanwijzing onder dit indexnummer zijn vastgelegd.

SA staat voor secundair adres. Met dit getal deelt u de computer mee of u data wilt zenden (SA=1) of ontvangen (SA=0). Voert u een 2 in, dan wordt er na het SAVEn nog een EOT (End Of Tape) blok geschreven, zie hoofdstuk 3.

AA staat voor apparaatadres. Dit getal vertelt de computer met welk apparaat hij moet werken. Hier betekent

```
0        = toetsenbord
1        = cassette
3        = beeldscherm
4 of 5   = printer
8 of 9   = diskette
```

Nu we een databestand hebben geopend, zijn we aan de instructies voor de in- en uitvoer van data toe.

2.5 PRINT# en CLOSE

Met het PRINT#-commando is het mogelijk afzonderlijke data op de band te zetten. Dit wordt duidelijk als u het volgende programma intoetst.

```
10 OPEN 1,1,1,"GEGEVENSFILE"
20 PRINT "BESTAND IS GEOPEND"
30 PRINT:PRINT"VOER DE GEGEVENS IN"
40 INPUT"GEGEVENS";A$
50 PRINT# 1,A$
60 A=A+1
70 PRINT"ER ZIJN"A"AANTAL GEGEVENS GEBUF
FERD"
80 PRINT:PRINT"VERDERE GEGEVENS ?"
90 GET F$: IF F$="" THEN 90
100 IF F$="J" THEN 40
110 CLOSE 1
120 PRINT:PRINT"BESTAND IS GESLOTEN"
130 END
```

Stop eerst een lege cassette in de recorder en start vervolgens het programma met RUN. Dan verschijnt meteen de mededeling "PRESS RECORD & PLAY ON TAPE" op het beeldscherm. Als u deze mededeling opvolgt, wordt volgens de OPEN-aanwijzing in regel 10 een titel van een databestand op cassette gezet. Dan stopt de recorder en wordt u in regel 40 gevraagd data in te voeren.

In regel 50 wordt dan door het PRINT#-commando de ingevoerde string opgeslagen. Het nummer na de PRINT#-aanwijzing, het logische filenummer, stelt de verbinding met de OPEN-aanwijzing voor. Aan het stilstaan van de band kunt u zien dat de string niet direkt op de band wordt gezet. Pas wanneer u een hele reeks data heeft ingevoerd gaat de band lopen en worden alle ingevoerde data op de band gezet. Op deze tussenopslag, het gaat hier om de cassettebuffer, komen we in hoofdstuk 7 terug.

Als u de vraag in regel 100 met "J" beantwoordt, kunt u andere data invoeren. Zodra de buffer vol is, volgt een pauze en gaat de band korte tijd lopen. Nu worden de data uit de buffer op de band

gezet. De buffer wordt vervolgens met "SPACE" (CHR$(32) leeggemaakt om nieuwe data op te nemen.

Als u de invoer van data beëindigt, gaat de computer verder naar de CLOSE-instructie in regel 110. Het getal achter dit commando, een logisch filenummer, vertelt de computer dat hij het met het logische filenummer 1 geopende databestand moet sluiten. Dat doet hij door alle invoer die hij in zijn kladblok onder de index 1 heeft genoteerd, uit te wissen. De computer schrijft nu achter de laatste invoer een nulbyte om "einde data" aan te geven en zet de inhoud van de buffer op de band. Het afsluiten is belangrijk om te voorkomen dat de laatste data verloren gaan.

Tot nu toe heeft u alleen strings, of de inhoud van stringvariabelen opgeslagen. Hoe verwerkt de computer nu numerieke termen en variabelen? Dergelijke termen worden op dezelfde wijze als strings opgeslagen. Dat betekent, dat de computer bij elk PRINT#-commando automatisch een STR$- commando uitvoert. Het maakt dus geen verschil of u nu

```
PRINT#1,55
```

of

```
A$=STR$(55):PRINT#1,A$
```

schrijft. Daarom kunt u later elke numeriek opgeslagen waarde ook in een stringvariabele inlezen. Een ander belangrijk punt is het format van een PRINT#-opdracht, als u met een opdracht meer dan een enkele term wilt opslaan. Op dit probleem gaan we voor de duidelijkheid pas na de INPUT#-instructie in.

2.6 INPUT#

Dit commando is gelijk aan het "normale" INPUT-commando, alleen is hier het toetsenbord tevoren niet als vast ontvangstapparaat ingesteld. Als u bijvoorbeeld met

```
OPEN 1,0
```

(0=toetsenbord) het toetsenbord definieert, gedraagt zich INPUT#1,A$ net als INPUTA$, maar dan zonder dat er een vraagteken wordt weergegeven.

Hoe de INPUT#-instructie zich ten opzichte van de cassetterecorder gedraagt, ziet u het best aan de hand van een voorbeeldprogramma.

U spoelt nu uw cassette terug naar het punt waar u het databestand "DATAFILE" heeft opgeslagen en u voert het volgende programma in:

```
10 OPEN 1,1,0,"GEGEVENSFILE"
20 PRINT "BESTAND IS GEOPEND"
30 PRINT:PRINT"GEGEVENS WORDEN INGELEZEN
"
40 INPUT# 1,A$
50 PRINT A$
60 A=A+1
70 PRINT"GEGEVENS ZIJN INGELEZEN"
80 PRINT:PRINT"VERDERE GEGEVENS ?"
90 GET F$: IF F$="" THEN 90
100 IF F$="J" THEN 40
110 CLOSE 1
120 PRINT:PRINT"BESTAND IS GESLOTEN"
130 END
```

Als u vervolgens het programma start en overeenkomstig de opdracht de PLAY-toets indrukt, gaat de band enige tijd lopen en stopt dan weer. Uw computer heeft nu op grond van het OPEN-commando het databestand "DATAFILE" opgezocht. Tevens is daarna volgens de INPUT#-instructie in regel 40 het eerste datablok ingelezen. Tenslotte is ook de eerste string aan de variabele A$ doorgegeven die door regel 50 op het beeldscherm wordt getoond.

De afzonderlijke data staan nu net zo in de buffer als u ze bij het extern opslaan heeft ingetypt. Achter iedere string is door de PRINT#-opdracht een RETURN (CHR$(13)) geplaatst. Het PRINT#-commando schrijft de data precies zo in de buffer als een PRINT-commando ze op het beeldscherm zou schrijven. Het INPUT#-commando herkent net als het INPUT-commando door een komma of RETURN dat hier een string is afgelopen.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	...	27	28	29	30	31
Inhoud	S	T	R	1	CR	S	T	R	2	CR	S	T	...	R	5	CR	00	

U moet nu de vraag of er nog meer data moeten worden ingelezen, beantwoorden. In ons voorbeeldprogramma toetst u "J" in. Nu kunt u string voor string weer inlezen en op het beeldscherm laten verschijnen, tot de mededeling "?STRING TOO LONG ERROR" verschijnt. Deze mededeling verschijnt als u intussen de laatst opgeslagen string heeft ingelezen en u er nog een probeert in te lezen. De computer zoekt het einde van de string, dus een komma of RETURN (CR), maar vindt die niet in de eerstvolgende 80

geheugencellen (u kunt maximaal 80 tekens met een INPUT- respectievelijk INPUT#-opdracht inlezen). Daardoor concludeert de computer dat deze string meer dan 80 tekens bevat en geeft hij een foutmelding. Om echter te voorkomen, dat u bij een "normaal" databestand steeds deze foutmelding tegenkomt, kunt u met behulp van de statusvariabele het einde van het databestand herkennen. Hoe u dit moet programmeren vindt u in hoofdstuk 4.

Deze foutmelding krijgt u echter niet alleen als u verder dan het einde van het databestand wilt lezen, maar ook als de data in een verkeerd format worden opgeslagen.

Bij de bespreking van de PRINT#-instructie hebben we steeds alleen een term met een PRINT#-opdracht opgeslagen, waardoor er steeds automatisch een RETURN aan werd toegevoegd. U kunt echter ook meer termen met een PRINT#-opdracht opslaan. U moet er dan wel op letten dat de afzonderlijke termen worden gescheiden. Schrijft u bijvoorbeeld

```
PRINT#1,A$,B$
```

met A$="STRING1" en B$="STRING2", dan worden de beide termen precies zo in de buffer geschreven als wanneer ze met een PRINT-opdracht op het beeldscherm zouden worden geschreven, namelijk

```
PRINT#1,A$,B$
```

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inhoud	S	T	R	I	N	G	1				S	T	R	I	N	G	2	CR

Als u echter de volgende opdracht geeft:

```
T$=","
PRINT#1,A$;T$;B$
```

krijgt u

```
T$=","
PRINT#1,A$;T$;B$
```

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inhoud	S	T	R	I	N	G	1	,	S	T	R	I	N	G	2	CR		

Nu zijn de strings duidelijk van elkaar gescheiden en worden door een INPUT#1,A$,B$-opdracht correct ingelezen.

We moeten er nog op wijzen dat de komma weliswaar voldoende is om afzonderlijke strings aan verschillende variabelen toe te kennen, maar niet om de computer te vertellen dat hier een invoer is afgelopen en dat er vanaf de volgende byte een nieuwe begint. Hoewel dat hetzelfde klinkt, is het dat niet. Daarom het volgende. U kunt de computer vergelijken met een grote firma met vele afdelingen. Het verwerken van data bij het lezen en schrijven laat ze aan twee verschillende afdelingen over.

- Eerste afdeling - variabelenverwerking. Deze afdeling zorgt ervoor dat een term, die ergens in het geheugen staat (cassettebuffer, beeldschermgeheugen, programmageheugen enzovoort), aan de juiste variabele in het juiste format wordt toegekend.

- Tweede afdeling - geheugensturing. Deze afdeling vertelt de eerste afdeling telkens, vanaf welke geheugenplaats de voor haar van belang zijnde term te vinden is.

Elk van beide afdelingen heeft haar eigen manier om het einde van een term en daarmee het begin van een nieuwe te zoeken. De eerste afdeling is het om het even of zij een komma of een RETURN vindt. Bij beide weet ze dat hier een term is afgelopen en er daarna een nieuwe begint. De tweede afdeling trekt zich niets van een komma aan. Zij herkent alleen de RETURN als scheidingsteken. Dat betekent dat ze steeds als startpunt voor een nieuwe term de plaats na een RETURN aangeeft. Dit wordt duidelijk aan de hand van een voorbeeldprogramma. U voert de volgende regels in, stopt een lege cassette in uw recorder en start het programma met RUN.

```
10 OPEN 1,1,1,"TEST"
20 T$=","
30 A1$="STRING1" : A2$="STRING2"
40 A3$="STRING3" : A4$="STRING4"
50 PRINT#1,A1$;T$;A2$;T$;A3$
60 PRINT#1,A4$
70 CLOSE 1
80 END
100 OPEN 1,1,0,"TEST"
110 INPUT#1,A$
120 PRINT A$
130 INPUT#1,A$
140 PRINT A$
150 CLOSE 1
160 END
```

Nadat het databestand TEST is opgeslagen, spoelt u de cassette naar het begin van het databestand terug en start het programma met RUN100. Als de computer het databestand heeft ingelezen, staan de strings als volgt in het geheugen:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S	T	R	I	N	G	1	,	S	T	R	I	N	G	2	,	S	T	R	I

20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
N	G	3	CR	S	T	R	I	N	G	4	CR	00			

Op het beeldscherm verschijnt

```
STRING1
STRING4
```

Afdeling twee heeft bij de eerste INPUT# haar wijzer op de eerste string geplaatst en de eerste afdeling heeft hem aan A$ gekoppeld. Omdat er geen andere strings bij deze INPUT#-opdracht worden ingelezen, wordt de uitwerking weer aan afdeling twee overgedragen. Bij de volgende INPUT#-opdracht zoekt afdeling twee de volgende RETURN en vindt deze voor STRING4. Dit startpunt geeft ze aan afdeling een door, die de term "STRING4" weer aan A$ toekent.

U verandert nu regel 100 en 120 door ',B$,C$' toe te voegen. Als u het databestand nu opnieuw inleest, krijgt u op het beeldscherm:

```
STRING1   STRING2   STRING3
STRING4
```

Nu kent de eerste afdeling in regel 110 eerst de beide volgende termen aan de variabelen B$ en C$ toe, voordat ze de leiding aan afdeling twee terug geeft.

In het kort:

- De wijzer van afdeling 1 wordt door elke toekenning van variabelen op het volgende scheidingsteken gezet (komma of RETURN).

- De wijzer van afdeling 2 wordt door elke INPUT#-opdracht op het volgende afscheidingsteken gezet (RETURN).

Samenvattend kunnen we zeggen dat een komma als scheidingsteken voldoende is, zolang u de data in het zelfde format inleest als u ze extern heeft opgeslagen. Wilt u echter bij het inlezen niet afhankelijk zijn van het invoer-format, dan definieert u de scheidingsstring met RETURN.

```
20 T$=CHR$(13)
```

Bij data die zo zijn opgeslagen, krijgt u steeds hetzelfde resultaat. Hierbij maakt het niet uit of u met een INPUT#-opdracht een of meer variabelen inleest.

Tot nu toe hebben we het alleen maar over de scheiding van afzonderlijke termen gehad. Bij sommige toepassingen is het echter nuttig afzonderlijke termen niet te scheiden. Dit bereikt u, net als bij de PRINT-instructie op het beeldscherm, door een PRINT#-opdracht met een puntkomma af te sluiten. Probeert u dit eens met het kleine testprogramma van hierboven. U verandert de regels als volgt:

```
10 OPEN1,1,1,"TEST2"
50 PRINT#1,A1$;A2$;A3$;
100 OPEN1,1,0,"TEST2"
```

Daarna wist u de regels 130 en 140. Als u nu met RUN het databestand TEST2 opslaat en het met RUN100 opnieuw laadt, krijgt u

```
STRING1STRING2STRING3STRING4
```

Ontstaan er echter door deze opeenvolging strings die langer zijn dan 80 karakters, dan kunt u die niet meer met de INPUT#-opdracht inlezen, wel byte voor byte met de GET#-opdracht.

2.7 GET#

Ook deze instructie werkt analoog aan het GET-commando. Hiermee kunt u byte voor byte data inlezen. Voert u het volgende programma maar in

```
10 OPEN1,1,0,"TEST"
20 GET#1,A$
30 PRINTA$;
40 GETA$: IF A$="" THEN 40
50 GOTO20
```

U spoelt vervolgens naar het begin van het databestand TEST terug en start het programma. Nadat de computer het databestand heeft gevonden, verschijnt in letters het volgende beeld op het scherm:

```
STRING1   STRING2   STRING3
STRING4
```

Als er nog andere tekens worden ingevoerd, ziet u geen veranderingen op het beeldscherm omdat de buffer is gevuld met "SPACE". Met het volgende programma maakt u deze tekens zichtbaar:

```
30 PRINT A$,ASC(A$+CHR$(0))
```

Hierdoor bereikt u dat elk teken als letter en als ASCII-waarde wordt uitgedrukt. Staat de GET#-opdracht op een nulbyte, dan wordt die als een lege byte geïnterpreteerd. Om te verhinderen dat hierdoor bij de verandering in de ASCII-code een foutenmelding ontstaat, wordt bij A$ steeds CHR$(0) opgeteld. Als u terugspoelt en het programma opnieuw start, ziet u het volgende op uw beeldscherm:

```
S          83
T          84
R          82
I          76
N          78
G          71
1          49
                    44
S          83
.
.
.
N          78
G          71
4          52
```

```
13
0
32
.
.
.
```

De nul na STRING4 is het teken voor het einde van het databestand. Hierna volgt alleen nog "SPACE".

3 HET SECUNDAIRE ADRES

Er is behalve het apparaatadres ook een secundair adres. Dit adres wordt gebruikt om de randapparatuur of de computer als ontvanger meer informatie over de werkwijze te verstrekken. Eerst een paar opmerkingen over de twee verschillende programmasoorten. Allereerst zijn er BASIC-programma's. Deze zijn niet afhankelijk van het geheugenbereik waarin ze staan, ze zijn verschuifbaar. Bij de VIC 20 hangt bijvoorbeeld de plaats van het BASIC-geheugenbereik af van de mate waarin het is uitgebreid. In de basisversie start het BASIC-bereik in geheugencel 1025. Breidt u het met 8 kbytes uit, dan begint het BASIC-bereik bij 1609. Niettemin lopen programma's die in de basisversie zijn geschreven ook in de uitgebreide versie. Om deze programma's steeds aan het begin van het BASIC-bereik te laden, bezitten de VIC 20 en de Commodore 64 een laadprogramma dat automatisch alle BASIC-programma's bij de BASIC-start inleest.

De andere programmasoort is het machinetaalprogramma. Deze programma's lopen alleen in het geheugenbereik waar ze zijn genoteerd. Ze mogen niet relatief worden geladen.

Om BASIC-programma's van machinetaalprogramma's te kunnen onderscheiden wordt bij het cassettegebruik het secundaire adres benut. Deze secundaire adressen hebben volgende functies:

I. SAVE

0 — Het opgeslagen programma wordt als BASIC-programma gekenmerkt. Daarmee wordt bereikt dat dit programma met de instructie LOAD automatisch op de actuele BASIC-start, dus relatief, wordt geladen.

1 — Kenmerkt een programma als machinetaalprogramma. Als u een dergelijk extern opgeslagen programma met LOAD laadt, wordt het automatisch op die plaats geladen waarvandaan het werd weggeschreven.

2 — Slaat een programma als BASIC-programma op net als bij het secundaire adres 0. Bovendien schrijft de computer achter het programma nog een zogenaamd EOT (End Of Tape) blok. Dit blok vertelt de computer bij het lezen, dat hij hier met het zoeken naar verdere programma's moet

ophouden. Stuit de computer bij het zoeken naar een programma op zo'n blok zonder eerder een programma te hebben gevonden, dan komt hij met de mededeling "?FILE NOT FOUND ERROR" voor de dag.

3 Kenmerkt een programma als machinetaalprogramma en schrijft er een EOT blok achter.

II. LOAD

0 Het programma wordt geladen volgens de informatie die in de programmakop staat. Dat betekent dat programma's die als machinetaalprogramma's zijn opgeslagen, ook zo worden geladen. Hetzelfde geldt voor BASIC-programma's. Die worden relatief geladen.

1 Ieder programma, om het even hoe het is gekenmerkt, wordt absoluut, dus op dat adres geladen waarvandaan het werd weggeschreven.

III. OPEN

0 opent een databestand om te lezen.

1 opent een databestand om te schrijven.

2 opent een databestand om te schrijven en schrijft achter het databestand een EOT-blok.

4 DE STATUSVARIABELE

In het vorige hoofdstuk is u al verteld dat er in uw Commodore 64 een vast gedefinieerde variabele zit die u informatie geeft over het verloop van een cassetteproces. Deze zogenaamde statusvariabele wordt door de computer bij elk gebruik op cassette of andere randapparatuur gezet. Met deze variabele kunt u eventueel optredende fouten preciezer specificeren. Verder kunt u met dit hulpmiddel vaststellen wanneer een databestand is afgelopen. Hij bestaat uit acht afzonderlijke "flags" die overeenkomstig de ontstane fouten worden geplaatst. Met behulp van een AND-operatie kunt u elke afzonderlijke bit van deze variabele testen. In tabel 1 kunt u de precieze betekenis van de flags lezen.

Om de vierde bit van de statusvariabele te begrijpen moet u weten dat de computer alle data en programma's twee keer op de band schrijft en bij het inlezen beide versies met elkaar vergelijkt (zie ook hoofdstuk 8).

Tabel 1

ST-Bit	ST-decimaal equivalent	Verklaring
0	1	Geen betekenis voor de cassette
1	2	_ " _
2	4	Kort blok. Een gevonden blok is korter dan hij zou moeten zijn.
3	8	Lang blok. Een gevonden blok is langer dan hij zou moeten zijn.
4	16	Second Pass fout. De gegevens van de eerste Pass komen niet overeen met die van de tweede.
5	32	Proefsom-fout. De opgeslagen controlesom komt niet met de berekende overeen.
6	64	File-einde.
7	128	Band-einde. EOT werd gelezen.

Hoe moet u deze variabele nu gebruiken? Zoals u in de tabel kunt zien, vertelt de 6de bit u dat een databestand is afgelopen. Als u dus binnen uw programma bit 6 vraagt, kunt u het inlezen na de laatste string beëindigen. U voert het volgende programma in en vervolgens leest u daarmee het in hoofdstuk 2 opgeslagen databestand in. Nadat "STRING4" werd afgedrukt, verschijnen READY en de cursor.

```
10 OPEN1,1,0,"TEST"
20 INPUT#1,A$
30 PRINT A$
40 IF(ST AND 64) = 64 THEN60
50 GOTO20
60 CLOSE1
70 END
```

4.1 Programma's redden na een LOAD ERROR

Het is u vast wel eens overkomen dat uw computer zich na het laadproces met "?LOAD ERROR" meldde en het geladen programma totaal niet of slechts gedeeltelijk was ingelezen. De oorzaak is meestal dat de positie van de toonkop ten opzichte van de band bij het schrijven een andere was dan bij het lezen. Uitkomst biedt hier alleen een juiste plaatsing van de toonkop, zoals in paragraaf 11.5 wordt beschreven. Laat het programma zich na een nieuwe toonkop-instelling nog steeds niet laden, dan ligt de fout aan de band of aan het onjuist opslaan van het programma. Paniek is overbodig. Het is namelijk mogelijk het programma te redden, althans gedeeltelijk.

Als u met uw computer een programma opslaat, wordt het twee keer achter elkaar op de band geschreven. Bij het inlezen wordt de eerste versie in het geheugen geladen en vervolgens met de tweede vergeleken. Constateert de computer hierbij een fout, dan probeert hij deze eerst te corrigeren. Als dit niet lukt, zet hij bit 4 in de statusvariabele en geeft "?LOAD ERROR".

Het is echter ook mogelijk, dat hij afzonderlijke bits of byte-kenmerken niet kan lezen, zodat de synchronisatie van de leesroutine en de ingelezen informatie wordt verstoord. Dit heeft tot gevolg dat de bits 2 en/of 3 worden gezet.

Elke fout leidt ertoe dat de BASIC-vectoren 45 en 46 (programma-einde) niet op de juiste manier worden ingesteld.

Indien de fout pas aan het eind van het programma of in de tweede pass is opgetreden, kunt u het programma helemaal of gedeeltelijk listen. Dit listbare deel kunt u met een UNNEW-commando redden. Dit commando is niet in Commodore-BASIC 2.0 opgenomen. Bezit u geen BASIC-uitbreiding die dit commando bevat, dan kunt u het aan het eind van dit hoofdstuk afgedrukte programma invoeren. Indien u geen assembler-programma heeft, voert u het met de BASIC-lader in en slaat u het extern op. Daarna start u de lader. Nu kunt u het machinetaalprogramma van het aangegeven startadres tot en met het BASIC-RAM-einde opslaan (zie hoofdstuk 6). Het zo opgeslagen

machinetaalprogramma kunt u nu indien nodig met LOAD laden en met SYS (startadres) starten om een uitgewist programma weer tot leven te wekken of om een verkeerd geladen programma tenminste gedeeltelijk te redden.

Nadat u met de UNNEW-instructie de programmavectoren heeft gezet, slaat u het op een andere cassette op. Nu moet u vaststellen waar het programma is beschadigd. Dat gebeurt op dezelfde manier als waarop u een ander BASIC-programma op fouten onderzoekt.

De meest voorkomende oorzaak voor een "?LOAD ERROR" is een verkeerd ingestelde kop. Daarom moet u altijd eerst proberen de toonkop beter op de band in te stellen en zo nodig de hierboven beschreven procedure met verschillende toonkop-instellingen herhalen tot u een zo weinig mogelijk beschadigd programma kunt laden. Natuurlijk moet u dit advies niet opvolgen als u niet volstrekt deskundig bent. Laat het afregelen van de toonkop dan liever aan de vakman over. De kop is een uiterst gevoelig onderdeel van uw recorder.

```
100 REM UNNEW 64
110 PS=0:E=256*PEEK(56)+PEEK(55)-1:A=E-5
1
120 FORI=ATOE:READX:PS=PS+X:POKEI,X:NEXT
130 IFPS<>5274THENPRINT"FOUT IN DATA":EN
D
140 H=INT(A/256):POKE56,H:POKE55,A-256*H
150 PRINT"STARTADRESSE"A:NEW
160 DATA165,43,164,44,133,34,132,35,160,
3,200,177,34,208,251,200,152,24,101
170 DATA34,160,0,145,43,165,35,105,0,200
,145,43,32,51,165,165,34,105,2,133
180 DATA45,165,35,105,0,133,46,32,99,166
,76,123,227
```

```
100 REM UNNEW 20
110 PS=0:E=256*PEEK(56)+PEEK(55)-1:A=E-5
1
120 FORI=ATOE:READX:PS=PS+X:POKEI,X:NEXT
130 IFPS<>5319THENPRINT"DATA ERROR":END
140 H=INT(A/256):POKE56,H:POKE55,A-256*H
150 PRINT"STARTADRES"A:NEW
160 DATA165,43,164,44,133,34,132,35,160,
3,200,177,34,208,251,200,152,24,101
170 DATA34,160,0,145,43,165,35,105,0,200
,145,43,32,51,197,165,34,105,2,133
```

```
180 DATA45,165,35,105,0,133,46,32,99,198
,76,103,228
```

```
unnew 64      Profi-ass 64 v2.0    seite 1

120:                  ;
130:                  ; unnew-instructie
140:                  ;
150:                  ; oproep met sys startadres
160:                  ;
170:                  ;
172:   7000                       .opt p5,o2
174:   7000                       .pag 38,5
180:   7000                       .tit "unnew 64"
182:   c663           clr         =    $c663    ;($c663)
184:   e467           basic       =    $e467    ;($e467)
186:   c533           binden      =    $c533    ;($c533)
200:   7000                       *=   $7000
210:   7000 a5 2b                 lda  $2b      ;basicstart in buffer
220:   7002 a4 2c                 ldy  $2c
230:   7004 85 22                 sta  $22      ;schrijven
240:   7006 84 23                 sty  $23
250:   7008 a0 03                 ldy  #3       ;nul-byte zoeken
260:   700a c8           nul      iny
270:   700b b1 22                 lda  ($22),y
280:   700d d0 fb                 bne  nul
290:   700f c8                    iny           ;regellengte
292:   7010 98                    tya
294:   7011 18                    clc
300:   7012 65 22                 adc  $22      ; + basicstart geeft koppel-
310:   7014 a0 00                 ldy  #0       ;adres v/d eerste regel
320:   7016 91 2b                 sta  ($2b),y  ;koppeladr op basicstart
330:   7018 a5 23                 lda  $22 + 1  ;schrijven
340:   701a 69 00                 adc  #0
350:   701c c8                    iny
360:   701d 91 2b                 sta  ($2b),y
370:   701f 20 33 c5              jsr  binden   ;koppeladressen nieuw bereken
460:   7022 a5 22                 lda  $22      ;in $22/$23 staat eindadres
470:   7024 69 02                 adc  #2       ; + 2 geeft basiceindvector
480:   7026 85 2d                 sta  $2d
490:   7028 a5 23                 lda  $22 + 1

unnew 64      Profi-ass 64 v2.0    seite 2

500:   702a 69 00                 adc  #0
510:   702c 85 2e                 sta  $2e
520:   702e 20 63 c6              jsr  clr      ;zet overige pointers
530:   7031 4c 67 e4              jmp  basic    ;terug naar basic
17000-7034
```

5 LADEN EN OPSLAAN VANUIT HET PROGRAMMA

Bij de behandeling van de commando's SAVE en LOAD in hoofdstuk 2 is beschreven hoe u programma's direkt opslaat en laadt. Deze commando's kunnen echter ook in de programma's zelf zitten.

Het SAVE-commando werkt binnen het programma net zo als op de direkte manier. Daarom kan een programma zichzelf opslaan. Maar in plaats van het hele programma zult u waarschijnlijk veel vaker maar een deel van een BASIC-programma of van een machinetaalprogramma willen opslaan.

Stel dat u een programma heeft geschreven dat vanaf regel 10000 dataregels voortbrengt. Dit programma-onderdeel wilt u alleen opslaan om het later aan andere programma's te kunnen koppelen. U moet eerst vaststellen op welke geheugenplaats regel 10000 in het geheugen staat omdat u immers slechts vanaf regel 10000 wilt opslaan. Met de volgende programmaregels lost u dit probleem op:

```
1000 S = PEEK(43) + 256 * PEEK(44) : BS = S
1010 ZN = PEEK(S + 2) + 256 * PEEK(S + 3)
1020 IF ZN = 10000 THEN 1050
1030 S = PEEK(S) + 256 * PEEK(S + 1)
1040 GOTO 1010
```

Om deze regels te begrijpen moet u eerst weten hoe BASIC-regels in het geheugen staan.

De eerste beide bytes van een BASIC-regel vormen het zogenaamde koppeladres dat het startadres van de volgende BASIC-regel aangeeft en dus op het volgende koppeladres wijst. In de twee volgende bytes is het regelnummer opgeslagen. Dan volgt de inhoud van de BASIC-regel, afgesloten door een nulbyte. Het BASIC-programma staat dus als volgt in het geheugen:

```
KAL KAH RNL RNH BASIC-tekst 00 KAL KLH RNL RNH ...
```

Hier betekenen:

KAL	koppeladres	LSB
KAH	koppeladres	MSB
RNL	regelnummer	LSB
RNH	regelnummer	MSB

In ons kleine programma wordt in regel 1000 het programma-startadres uit de vectoren in S en BS ingelezen. In

regel 1010 wordt het regelnummer in de variabel RN overgedragen. De vraag in regel 1020 toetst of het gewenste regelnummer werd bereikt. Als dit is gebeurd, wordt dit in het verdere programma verwerkt. Is dit niet het geval, dan wordt in regel 1030 uit het koppeladres de startpositie van de volgende BASIC-regel in S geschreven en wordt er via regel 1040 weer naar 1010 gesprongen.

Na dit programma heeft u alle informatie om het laatste deel van uw programma op te slaan. Belangrijk is nu dat u na het opslaan weer de oude waarde in de BASIC-startvector schrijft. De dataregels slaat u zo op:

```
1050 POKE 43,S AND 255: POKE 44,INT(S/256) :REM WIJZER OP
     REGEL 10000 ZETTEN
1060 SAVE"DATAREGELS" :REM DATAREGELS OPSLAAN
1070 POKE 43,BS AND 255: POKE 44,INT(BS/256) :REM WIJZER
     TERUGZETTEN
```

Het opslaan van machinetaalprogramma's is ingewikkelder. Zie daarvoor hoofdstuk 6.

5.1 OVERLAY-techniek

Het LOAD-commando werkt binnen de programma's iets anders dan op de direkte wijze. In hoofdstuk 2 staat dat na een LOAD de BASIC-wijzers op het nieuwe programma worden ingesteld. Als u echter de LOAD-opdracht binnen het programma geeft, blijven de vectoren bewaard, net als alle tot dan toe gedefinieerde variabelen. Daarom is het mogelijk vanuit een hoofdprogramma verschillende subprogramma's op te roepen, die alle met dezelfde variabele werken. Deze methode heet de OVERLAY-techniek.

Belangrijk bij deze techniek is:

1. Omdat de BASIC-vectoren niet worden veranderd, moet het programma dat oproept minstens zo lang zijn als het programma dat wordt opgeroepen.

Direkt na een BASIC-programma worden de variabelen opgeslagen en de vector 45, 46 is daarmee ook de wijzer voor het begin van de variabelentabel. Is het later geladen programma echter langer, dan wordt er over de variabelen heengeschreven en wijst de wijzer in het nieuwe programma. Zodra een variabelenproces plaatsvindt, wordt de variabele binnen dat programma gezocht wat tot de "val"

van de computer kan leiden. Maar met een trucje kunt u met een korter programma een langer oproepen. Allereerst laadt u het langste nog te laden programma op de direkte manier in en u bepaalt door het uitlezen van de geheugenplaatsen 45,46 de lengte. De gevonden waarden kunt u het beste noteren. Vervolgens laadt u het programma dat oproept en plaatst u de volgende BASIC-regel aan het begin:

```
10 POKE 45,W1: POKE 46,W2: CLR
```

Het regelnummer is facultatief. Het moet echter wel het kleinste in uw programma zijn. W1 en W2 staan voor de door u genoteerde waarden. Door de POKE-commando's verlengt u uw programma kunstmatig. CLR zorgt ervoor dat ook de andere wijzers worden veranderd.

2. U moet erop letten dat na een LOAD het programma weer vanaf het begin wordt uitgevoerd. Dat is alleen zinvol, als u BASIC-programma's achteraf laadt. Wilt u echter een machinetaalprogramma achteraf laden, dan verandert het in het geheugen aanwezige BASIC-programma niet en start steeds weer opnieuw. Dit kunt u met een springtabel voorkomen. Bij het volgende programma worden aan het begin drie machinetaalprogramma's geladen voordat het hoofdprogramma begint.

```
10 REM START
20 IF A = 0 THEN A=1: LOAD"MPR1",1,1
30 IF A = 1 THEN A=2: LOAD"MPR2",1,1
40 IF A = 2 THEN A=3: LOAD"MPR3",1,1
50 REM HOOFDPROGRAMMA
```

3. Tenslotte een bijzonderheid van het opslaan van strings. Het bedrijfssysteem van uw Commodore is zo geconstrueerd, dat er zo weinig mogelijk geheugenruimte voor strings wordt gebruikt. Iedere keer als u aan een stringvariabele een string toekent, wordt er in de variabelentabel onder de variabelennaam een wijzer geplaatst die naar de betreffende string verwijst. Als u door een INPUT-opdracht aan een variabele een string toekent, wordt deze in een speciale tabel aan het eind van het BASIC-RAM geplaatst. Vindt de toekenning binnen het programma plaats, bijvoorbeeld door

```
100 A$="STRING"
```

dan wordt de string niet nog een keer in de springtabel geplaatst, maar wordt de wijzer op de positie van de string in het programma gezet.

Laadt u achteraf een programma, dan staat er op de overeenkomstige plaats iets anders en print de computer bij het commando PRINT A$

alleen maar onzin. Dit verhelpt u door een schijnbaar nieuwe string te definieren doordat u de in het programma gedefinieerde string met een lege string verbindt.

```
100 A$="STRING": A$=A$+""
```

Hierdoor laat u het bedrijfssysteem de string in de stringtabel plaatsen.

6 HET OPSLAAN VAN MACHINETAALPROGRAMMA'S

Als u al wat langer met uw Commodore werkt, wilt u wel eens bepaalde geheugenbereiken opslaan. Dit kan bijvoorbeeld de informatie op het beeldscherm of een machinetaalprogramma zijn. Weliswaar vervult Commodore BASIC deze wens niet, maar met een paar handigheidjes, zoals die beschreven werden in hoofdstuk 5, lukt dit toch wel.

De computer slaat bij een SAVE-commando steeds het bereik dat ligt tussen de wijzers in 43 en 44 of 45 en 46 op. Nu ligt het voor de hand de wijzers zo te veranderen dat ze de start en het einde van het door u gewenste geheugenbereik aangeven. Als u daarna echter nog "normaal" met uw computer wilt werken, mag u het zich niet zo gemakkelijk maken. Om de computer na het opslaan net zo te laten werken als daarvoor, moeten in de vectoren weer de oude waarden staan. U moet deze waarden echter van te voren "redden" en later weer in de wijzers 43-46 plaatsen.

Om de inhoud van de wijzers te redden, mag u ze niet aan de variabelentabel toevertrouwen omdat, zoals we al gezien hebben, de programma-eindvector tegelijkertijd de variabelen-startvector is. Zodra u deze wijzer verandert, vindt de computer geen variabelen meer. Daarom moet u de wijzerinhoud in die geheugenplaatsen POKEn die niet door het proces van opslaan of door het bedrijfssysteem kunnen worden veranderd. In het volgende voorbeeld gebruikt u de geheugenplaatsen 251 tot en met 254 die door het bedrijfssysteem niet worden gebruikt. Met deze regels schrijft u de informatie van het beeldscherm op cassette:

```
10 POKE 251, PEEK(43): POKE 252, PEEK(44
):REM START ONTHOUDEN
20 POKE 253, PEEK(45): POKE 254, PEEK(46
):REM EINDE ONTHOUDEN
30 POKE 43,0:POKE 44,4:REM VIDEOSTART IS
 BEGIN
40 POKE 45,232: POKE 46,3: REM VIDEORAME
INDE
50 SAVE"BEELDSCHERM",1,1: REM VIDEORAM O
PSLAAN
60 POKE 43, PEEK(251): POKE 44, PEEK(252
):REM START HERSTELLEN
70 POKE 45, PEEK(253): POKE 46, PEEK(254
): REM EINDE HERSTELLEN
```

Met een klein machinetaalprogramma gaat dit eenvoudiger. U voert in

```
SYS adr"NAAM",AA,sadr,eadr
```

met de volgende betekenis:

adr	startadres van het machinetaalprogramma
NAAM	naam van het bereik dat wordt opgeslagen
AA	apparaatadres (cassette = 1)
sadr	startadres van het bereik dat wordt opgeslagen
eadr	eindadres

en het bereik wordt opgeslagen zonder dat u ook maar een verandering van de vectoren hoeft toe te passen. De afgedrukte BASIC-laders slaan het kleine machinetaalprogramma aan het eind van het BASIC-RAM op en geven het startadres waarmee u het kunt oproepen. Voordat u echter de lader start, slaat u hem op omdat hij zichzelf automatisch uitwist.

Voor de liefhebbers volgt nu nog een uitleg van het machinetaalprogramma. In regel 310 wordt een routine van een bedrijfssysteem opgeroepen dat de naam van het beeldscherm leest en deze in de desbetreffende geheugenplaatsen schrijft. De routine die in regel 320 wordt opgeroepen, toetst of het eerstvolgende teken een komma is en plaatst de volgende term in het X-register. In regel 330 wordt het secundaire adres in het Y-register geladen en in regel 340 het logische filenummer in de accu. Met deze parameters wordt een routine opgeroepen die de file-parameters zet. Deze routine moet voor elk invoer- en opslagproces worden opgeroepen. In de regels 350-380 wordt weer op een komma getoetst. Vervolgens wordt het startadres ingelezen en verwerkt. Dit startadres wordt in de geheugenplaatsen $14 en $15 geplaatst. In de nu volgende regels worden de inhouden van $14/$15 via $FB/$FC gered en daarna wordt het eindadres ingelezen. De overdracht van deze parameters naar de SAVE-routine vindt plaats, doordat in de X- en Y-registers de eindadressen absoluut worden overgedragen en de accu de zeropage-positie bevat, waarin het startadres staat.

```
100 REM SAVE ADRES 64
110 E=256*PEEK(56)+PEEK(55)-1:A=E-47
120 FORI=ATOE:READX:POKEI,X:NEXT
130 H=INT(A/256):POKE56,H:POKE55,A-256*H
140 PRINT"STARTADRES"A:NEW
150 DATA32,87,226,32,0,226,160,1,169,0,3
2,186,255,32,14,226,32,138,173,32,247
```

```
160 DATA183,166,20,164,21,134,251,132,25
2,32,14,226,32,138,173,32,247,183,169
170 DATA251,166,20,164,21,76,216,255


100 REM SAVE ADRES VC-20
110 E=256*PEEK(56)+PEEK(55)-1:A=E-47
120 FORI=ATOE:READX:POKEI,X:NEXT
130 H=INT(A/256):POKE56,H:POKE55,A-256*H
140 PRINT"STARTADRES"A:NEW
150 DATA32,84,226,32,253,225,160,1,169,0
,32,186,255,32,11,226,32,138,205,32
160 DATA247,215,166,20,164,21,134,251,13
2,252,32,11,226,32,138,205,32,247,215
170 DATA169,251,166,20,164,21,76,216,255
```

```
80:     c000                    .Pag 61
100:    c000                    .oPt P1,o2
102:    c000
102:    c000
111:    c000
111:    c000
112:                    ; save adres
114:                    ; oProoeP met
115:                    ; sys adres "naam",dn,startadr.,eindadr.
119:                    ; waardes tussen haakjes
120:                    ; 9elden voor  vc20
125:    c000
125:    c000                    *=    $c000
130:    e257            setfname =    $e257    ; (e254)
140:    e200            hbyte    =    $e200    ; (e1fd) kommatest en byte => x
150:    ffba            setlfs   =    $ffba
160:    e20b            chrin    =    $e20b    ; (e20b) kommatest
170:    ad8a            frmnum   =    $ad8a     ; (cd8a)
180:    b7f7            facint   =    $b7f7    ; (d7f7) fac naar $14,$15
190:    ffd8            save     =    $ffd8
200:    00fb            buffer   =    $fb
300:    c000
300:    c000
310:    c000 20 57 e2           jsr  setfname  ; fnaam halen en Param. zetten
320:    c003 20 00 e2           jsr  hbyte     ; dn halen
330:    c006 a0 01              ldy  #1        ; sa
340:    c008 a9 00              lda  #0        ; lfn
350:    c00a 20 ba ff           jsr  setlfs    ;fileParameter zetten
360:    c00d 20 0b e2           jsr  chrin     ;startadres halen
370:    c010 20 8a ad           jsr  frmnum    ;en 9ebruiken
380:    c013 20 f7 b7           jsr  facint
390:    c016 a6 14              ldx  $14
400:    c018 a4 15              ldy  $15
410:    c01a 86 fb              stx  buffer    ;in buffer oPslaan
420:    c01c 84 fc              sty  buffer+1
430:    c01e 20 0b e2           jsr  chrin     ;eindadres halen
440:    c021 20 8a ad           jsr  frmnum    ;en 9ebruiken
450:    c024 20 f7 b7           jsr  facint
```

```
460:    c027 a9 fb          lda  #buffer   ;overdracht startadres
470:    c029 a6 14          ldx  $14       ;overdracht eindadres
480:    c02b a4 15          ldy  $15
490:    c02d 4c d8 ff       jmP  save
]c000~c030
```

7 DE CASSETTEBUFFER

In dit hoofdstuk gaan we nader in op het onderwerp cassettebuffer. Deze bezet de geheugenlocaties 828 ($033C) tot en met 1019 ($03FB) en heeft twee hoofdtaken. Ten eerste wordt in dit gebied de filekop, de zogenaamde header, geplaatst. Deze bevat de volgende parameters:

1. filetype,	byte 0	=828
2. startadres,	byte 1, 2	=829, 830
3. eindadres,	byte 3, 4	=831, 832
4. filenaam vanaf	byte 5	=833

De andere bytes bevatten spaties.

Als tweede functie werkt de cassettebuffer als tussengeheugen bij het SAVEn van een bestand. De data die moeten worden opgeslagen, worden eerst in deze geheugenruimte genoteerd en pas als de hele buffer is volgeschreven, wordt deze op de band gezet. Als u data laadt worden deze bloksgewijs in de buffer geladen en vervolgens door INPUT# of GET# uit de buffer gelezen, zoals in hoofdstuk 2 is beschreven.

Het filetype is als volgt gecodeerd:

1 = BASIC-programma, relatief laden
2 = datablok
3 = machinetaalprogramma, absoluut laden
4 = data header
5 = EOT-blok

De header of kop bevat alle belangrijke gegevens die voor het herkennen worden gebruikt. Het ligt voor de hand met behulp van deze data de programma- en datacassettes in een archief op te nemen.

7.1 De inhoudsopgave van een cassette maken

Het programma in deze paragraaf toont een inhoudsopgave van een cassette. Als u een printer heeft, kunt u deze inhoudsopgave ook afdrukken.

Programmabeschrijving

De regels 100-160 zijn voor de initialisatie. Hier worden de afzonderlijke tabulatorstops in TX$ en de verschillende filetypes in het veld TN$ ingelezen. Als het resultaat ook naar een printer moet worden gestuurd, wordt in regel 180 een printkanaal geopend en in 190-220 de titel aan de printer meegedeeld. In regel 230 wordt dan een naamloos databestand voor het lezen van cassettes geopend. Hierdoor wordt het eerstvolgende bestand van de cassette ingelezen.

Door PEEK worden in de regels 240-270 typ-byte, start- en eindadressen en filenamen uit de buffer gelezen en geformateerd. In de regels 280-290 wordt uit de start- en eindadressen de programmalengte in Kbyte berekend. In de regels 300-320 worden de data naar printer en beeldscherm doorgegeven. In regel 330 wordt met behulp van de programmalengte de tellerstand berekend. Nadat het databestand is gesloten springt het programma weer naar het OPEN-commando.

```
*******************"
100 REM    ***    CASSETTE-INHOUD    ***
110 PA=828: REM STARTADRES IN CASSETTEBU
FFER
120 PRINT"■■PRINTEN ? (J/N) ";D$
130 GET D$:IF D$="" THEN 130
140 T1$=CHR$(16)+"05":T2$=CHR$(16)+"10"
145 T3$=CHR$(16)+"20":T4$=CHR$(16)+"30"
150 T5$=CHR$(16)+"38":T6$=CHR$(16)+"50"
155 T7$=CHR$(16)+"63":REM TABULATORSTOPS
160 TN$(0)="RELATIEF":TN$(2)="ABSOLUUT":
TN$(3)="BESTAND"
170 IF D$<>"J" THEN 230
180 OPEN 2,4
190 PRINT# 2,T1$"LFN"T2$"TELLER"T3$"TYPE
"T4$"K-BYTE";
200 PRINT# 2,T5$"    BEGIN"T6$"    EINDE"T
7$"    NAAM"
220 PRINT# 2
230 OPEN 1,1
240 TY=PEEK(PA):TY$=TN$(TY-1)
```

```
250 A$=" :"+RIGHT$("       "+STR$(PEEK(PA
+1);(256*PEEK(PA+2))-1),6)
260 B$=" -"+RIGHT$("       "+STR$(PEEK(PA
+3);(256*PEEK(PA+4))-1),6)
270 C$=: ":FOR I=5 TO 20:C$=C$+CHR$(PEEK
(PA+1)):NEXT
280 T=VAL(RIGHT$(B$,6))-VAL(RIGHT$(A$,6)
)
290 K$=RIGHT$("      "+STR$(INT(T/1024*10
0)/100),6)
300 PRINT Z;TY$;K$;A$B$:PRINT C$:PRINT
310 N=N+1:N$=RIGHT$("       "+STR$(N),3)
315 IF D$<>"J" THEN 330
320 PRINT#2,T1$;N$; T2$;Z; T3$;TY$; T4$;
K$; T5$;A$; T6$;B$; T7$;C$
330 Z=INT(Z+(T/212)+4+2*(N/100))
340 CLOSE 1
350 GOTO 230
```

7.2 Vermelding van de gevonden databestanden

Het uitzoeken van de filenaam door middel van een PEEK-commando kan ook bij een stuurprogramma van een databestand nuttig zijn. Het is vervelend als de computer enige tijd zoekt, wanneer u een bepaald databestand van een cassette wilt inlezen, om tenslotte "?FILE NOT FOUND" te melden. Dit kan komen doordat u de band iets te ver heeft doorgespoeld. Indien de computer u evenwel vertelt welke databestanden hij aantreft, kunt u vaststellen of u te ver heeft gespoeld of niet. Vooropgesteld natuurlijk dat u de volgorde van de databestanden op de band ongeveer kent. De volgende BASIC-regels laten de computer elk databestand dat hij tegenkomt, aangeven:

```
100 INPUT"FILENAAM";F$
110 L=LEN(F$)
120 FE$=""
130 OPEN4,1,0,FE$
140 FORI=0TO15:FE$=FE$+CHR$(PEEK(833+I))
:NEXT
150 PRINT:PRINT"FOUND ";FE$
160 IFLEFT$(FE$,L)<>LEFT$(F$,L)THENCLOSE
4:GOSUB200:GOTO120
170 PRINT:PRINT"FILE WORDT GELADEN"
200 POKE198,0:WAIT198,1:RETURN
```

Door het naamloze OPEN-commando in regel 130 laat u de computer elke data-filekop in de buffer opslaan. In de regels 140-160 wordt dan de filenaam door middel van PEEK uit de buffer in FE$ overgebracht en in regel 170 aangegeven. Levert de vergelijking van FE$ met de verlangde filenamen in regel 180 geen overeenstemming op, dan wordt het zoeken voortgezet. In het andere geval wordt het databestand geladen.

7.3 Zelf-startende programma's en programmabescherming

Tot nu toe heeft u alleen data uit de cassettebuffer gelezen. Deze buffer is echter ook geschikt om data of programma's op te nemen. Zo kan de buffer met een bepaald programma ook nog andere data of een ander programma opslaan.

De buffer bezet de geheugenruimte van 828 tot en met 1019. Hij is dus 191 bytes lang. Voor de informatie-overdracht zijn, wanneer hij als header wordt opgeslagen, maximaal 1 + 2 + 2 + 16 (filetype, startadres, eindadres, filenaam) = 21 bytes nodig. Er blijven dus nog 170 bytes over. Dit is voldoende ruimte om kleine machinetaalprogramma's onder te brengen.

Als voorbeeld volgt een klein programma dat eigenlijk een voorprogramma is. Hiermee kunt u bijvoorbeeld op een speciale manier het volgende programma automatisch invoeren en starten. Zulke programma's kunnen uw programma beschermen en ook voor het laden van het hoofdprogramma het adresbereik van het BASIC-RAM verschuiven. Op deze manier kunnen ook nog een of meer grafische pagina's voor het BASIC-programma geplaatst worden. Het gekozen voorbeeld toont de techniek waar het om gaat bij speciale toepassingen en kan ook door een gebruiker die niet zelf programmeert, algemeen worden toegepast.

De programmabeschrijving:

In regel 10 wordt de filenaam gevraagd waaronder het hoofdprogramma moet worden opgeslagen. In de regels 20 en 30 wordt de filenaam door het opvullen met spaties op een lengte van 16 karakters gebracht. Door de regels 40 tot en met 70 wordt aan de filenaam een machinetaalprogramma gekoppeld dat in de

dataregels vanaf regel 180 vermeld staat. Het laadprogramma voor de VC 20 en Commodore 64 zijn op de data na gelijk. Daarom zijn alleen de dataregels voor de Commodore 64 afgedrukt. Door middel van de regels 80 en 90 kunt u nu BASIC-commando's invoeren die later, na het invoeren van de voorprogramma's, automatisch worden uitgevoerd. Als u dus een volgend BASIC-programma wilt invoeren en starten, voert u met korte commando's LOAD RETURN RUN RETURN RETURN in:

Vanwege programma-technische redenen die later zullen worden verklaard, is het bij dit programma niet mogelijk meer dan 10 karakters over te dragen. Deze karakters worden dan gekoppeld aan de filenaam achter het machinetaalprogramma (regels 100-150). In regel 160 wordt de wijzer voor een systeemroutine, namelijk die van de invoer- en wachtlus, op het machinetaalprogramma in de cassettebuffer omgezet. Regel 170 slaat vervolgens met behulp van het in het laatste hoofdstuk beschreven machinetaalprogramma deze vector op. U moet dus achter de SYS-instructie het startadres van uw programma SAVE ADRESS schrijven. Door dit opslag-commando wordt de filenaam met een machinetaalprogramma en BASIC-commando's in de buffer en op de band geschreven. Na het programma-einde springt de computer via de vector in 770 en 771, naar de locatie waar gewoonlijk de invoer-wachtlus staat. Omdat deze echter in ons machinetaalprogramma in regel 160 van de cassettebuffer werd geplaatst, springt hij daar heen en werkt dit af. Aangezien we in de uitgiftestring de computer het commando LOAD hebben gegeven, probeert hij het eerstvolgende programma te laden. Door het indrukken van de RUN/STOP-toets kunt u dit proces onderbreken. Hierdoor wordt het hoofdprogramma achter het invoerprogramma opgeslagen. Dit invoerprogramma moet door het voorprogramma worden geladen. Als u een programma achter de lader heeft geschreven, spoelt u de band terug naar het begin van het invoerprogramma en toetst LOAD en RETURN in. Uw computer vindt nu het invoerprogramma, voert dit uit, laadt direkt daarna het volgende programma en start dit.

De basisbeginselen van deze techniek zijn:

1. Overdracht van een machinetaalprogramma in de cassettebuffer door een toevoeging aan de filenaam.

2. Verandering van vectoren (bijvoorbeeld van vectoren voor de invoerlus op een eigen machinetaalprogramma) en het opslaan van deze veranderde vectoren. In uw machinetaalprogramma moet dan, voordat u dit verlaat met de sprong naar het geladen programma, de op het eigen machinetaalprogramma wijzende vector weer op de oude waarde worden teruggebracht.

3. De inhoud van geheugenruimte die als machinetaalprogramma opgeslagen werd, wordt steeds op dezelfde plaats in het geheugen geladen vanwaaraf ze op cassette werd opgeslagen met SAVE.

Uw computer bezit in de geheugenplaatsen 768-819 een reeks vectoren waar hij in bepaalde subprogramma's overheen springt. Door verandering van deze wijzer kunt u er een eigen programma invoegen. De invoer-wachtlus is het subprogramma dat uw Commodore 64 geduldig op een willekeurige invoer laat wachten. Voert u gegevens in, dan beoordeelt hij deze en voert de opdracht uit, om vervolgens weer in de oorspronkelijke invoer-wachtlus terug te keren.

Door ons programma wordt een voorprogramma voortgebracht dat het startadres van een eigen programma over de vector voor de invoer-wachtlus heenschrijft. Dit eigen programma wordt met de filenaam in het geheugen geschreven. Als u zelf verder wilt experimenteren, kunt u natuurlijk ook uw machinetaalprogramma in bijvoorbeeld het bereik $2C0-$2FF (704-766) stoppen en als voorprogramma het bereik $2C0-$303 opslaan.

Laten we nu het machinetaalprogramma eens bekijken. In de regels 200-230 wordt de vector voor de invoer-wachtlus hersteld. Vanaf de geheugenplaats $036C is uw invoer opgeslagen. Deze wordt in de regels 250 tot en met 290 aan de toetsenbordbuffer overgedragen. Door de sprong aan het eind van het programma in de invoer-wachtlus, wordt de computer ertoe gebracht, de commando's die in de toetsenbordbuffer staan uit te voeren. Dit voorbeeld heeft echter als nadeel dat het als een gevolg van de beperkte toetsenbordbuffer maar weinig commando's kan doorgeven. U kunt echter, als u het principe heeft begrepen, gemakkelijk dit programma aan uw eigen wensen aanpassen. U schrijft dan in het machinetaalprogramma vanaf regel 240 uw eigen routine en sluit deze met een sprong in de invoer-wachtlus af. Het is u zeker al duidelijk geworden, dat u deze techniek ook zeer goed voor bescherming van het programma kunt gebruiken omdat het programma immers zichzelf start. Als u bovendien nog een programmastop via de RUN/STOP-toets inbouwt en het hoofdprogramma zo verandert dat het alleen in verbinding met de lader kan lopen, heeft een vreemde gebruiker geen mogelijkheid om inzicht in uw programma te krijgen. De hele bescherming zou echter nutteloos zijn, als hier een precieze handleiding gegeven zou worden. U bent met wat kennis van programmeren in machinetaal in staat zelf eigen automatische startroutines te ontwikkelen.

```
110:   0351                    .opt p1,o
120:   0351                    *=   $0351
130:   0351
130:   0351
140:                   ;laad programma
150:   0351
150:   0351
160:   a483            vector   =    $a483    ; ($c483) vector
165:                   ;  in de invoerlus
170:   00c6            pa       =    $c6
180:   0277            buffer   =    $277     ; toetsenbordbuffer
185:   036c            tabel    =    $36c     ; tabel v/d basic-instr.
190:   0351
190:   0351
200:   0351 a9 83              lda  #<vector  ; zetten v/d oude
210:   0353 8d 02 03           sta  $302     ; vector
220:   0356 a9 a4              lda  #>vector
230:   0358 8d 03 03           sta  $303
234:                   ;
235:                   ;  waarden uit tabel
237:                   ;in de toetsenbordbuffer
238:                   ;    overdragen
239:                   ;
240:   035b ae 6c 03           ldx  tabel    ; aantal tekens
250:   035e 86 c6              stx  pa
260:   0360 bd 6c 03 loop      lda  tabel,x
270:   0363 9d 77 02           sta  buffer,x
280:   0366 ca                 dex
290:   0367 d0 f7              bne  loop
294:                   ;
295:                   ;in de invoer-wachtlus
297:                   ;    terugspringen
299:                   ;
300:   0369 4c 83 a4           jmp  vector
]0351-036c
```

```
5 REM LOADER VOOR C-64
10 INPUT"FILENAAM ";F$
20 SP$="                "
30 F$=LEFT$(F$+SP$,16):S=833
40 FORI=0TO26
50 :  READF
60 F$=F$+CHR$(F)
70 NEXT
80 PRINT"UITVOERSTRING MAX. 10 TEKENS "
90 INPUT"↑ = RETURN, EINDE 2 MAAL";A$
100 LA=LEN(A$):IFLA>10THEN90
110 F$=F$+CHR$(LA)
120 FORI=1TOLA
130 :  W$=MID$(A$,I,1):IFW$="↑"THENW$=CH
R$(13)
140 :  F$=F$+W$
150 NEXT
```

```
160 POKE770,81:POKE771,3
170 SYS12*4096F$,1,768,772
180 DATA169,131,141,2,3,169,196,141,3,3,
174,108,3,134,198,189,108,3,157,119
190 DATA 2,202,208,247,76,131,164


500 REM LOADER VC-20
510 REM VOOR VC-20,DE REGELS 180 EN 190
VERWIJDEREN
520 REM DE VOLGENDE REGELS KOMEN DAARVOO
R IN DE PLAATS
600 DATA 169,131,141,2,3,169,196,141,3,3
,174,108,3,134,198,189,108,3,157,119
610 DATA 2,202,208,247,76,196,131
```

8 HET FORMAT VAN DE OPSLAG OP CASSETTE

Om data of programma's op band vast te leggen, worden de afzonderlijke bytes en bits als golven geregistreerd. Voor de codering zijn drie verschillende tijden gedefinieerd:

K = kort (176 microseconden)
M = middellang (256 microseconden)
L = lang (336 microseconden)

Met deze golven worden drie verschillende combinaties gevormd die de volgende betekenis hebben:

LLMM = byte, deze combinatie gaat vooraf aan elke byte
MMKK = bit gezet = 1
KKMM = bit niet gezet = 0

De waarde 65 wordt dus als volgt op band vastgelegd:

LLMM	MMKK	KKMM	KKMM	KKMM	KKMM	KKMM	MMKK	KKMM	MMKK
byte	1	0	0	0	0	0	1	0	1
bit	0	1	2	3	4	5	6	7	PARITY ODD

Dit geeft 8.96 milliseconde voor een karakter.

Bij het leesproces telt een teller net zolang vanaf een bepaalde waarde terug, als er signaal staat op de verbindingspen waarmee de computer gegevens inleest vanaf de cassetterecorder. Aan de geconstateerde waarden herkent de computer dan of het om een geplaatst bit, een niet geplaatst bit of een bytemerk gaat. Om de teller steeds op het juiste tijdstip te laten starten, is een nauwkeurige synchronisatie tussen band en teller noodzakelijk. Daarom gaat aan elk blok een synchronisatie en een aftelling vooraf.

Een blok is als volgt opgebouwd:

1. Synchronisatiebytes en het aftellen
   - Door dit deel worden de leesbouwstenen op de band gesynchroniseerd. Verder vertelt de synchronisatie ook om wat voor blok het gaat.

2. Data

3. Toetsbyte
   - Tijdens het opslaan en laden vormt de computer uit alle bytes een EXOR-toetsbyte. Dit gebeurt door steeds de laatste waarde via een logische "exclusive-or"-functie met de volgende te verbinden en op te slaan. Wanneer zich leesfouten voordoen, komt deze toetsbyte niet met de opgeslagen toetsbyte overeen.

4. Herhaling van de data
   - Deże herhaling wordt bij het leesproces met de al ingelezen bytes vergeleken om eventuele leesfouten vast te stellen.

5. Toets- of proefsombyte (als bij 3.)

6. Blok-einde-markering

De complete files worden als volgt opgeslagen:

programmafiles	datafiles
1. header	header
2. programma	datablok eventueel volgende datablokken
3. eventueel EOT-blok	eventueel EOT-blok

9 HET KOPPELEN VAN BASIC-PROGRAMMA'S

U zult al wel gemerkt hebben dat het Commodore-BASIC 2.0 geen commando's ter beschikking stelt om verschillende programma's in het geheugen met elkaar te verbinden. Weer komen ons de BASIC-wijzers 43/44 en 45/46 te hulp. Ze "verbuigen" de BASIC-startwijzer naar het einde van het in het geheugen aanwezige programma en laden het te koppelen programma aansluitend daar achter. Daarna moet de wijzer 43 en 44 weer op de oude waarde worden ingesteld. Hierbij gaat u als volgt te werk:

1. U laadt het eerste programma. Alle regelnummers moeten kleiner zijn dan die van het te koppelen programma. Is dat niet het geval, dan werkt het samengestelde programma niet zoals zou moeten.

2. U bepaalt door

```
PRINT PEEK(43),PEEK(44)
```

het BASIC-startadres en u noteert dit.

3. U toetst nu in:

```
POKE43,(PEEK(45)+256*PEEK(46)-2)AND255
POKE44,(PEEK(45)+256*PEEK(46)-2)/256
```

Omdat aan het eind van een BASIC-programma steeds als eindmarkering twee nul-bytes staan, moet u de eindvector met twee bytes verminderen (zie ook paragraaf 2.1). Nu heeft u ogenschijnlijk geen programma meer in het geheugen. Dit kunt u nagaan door LIST in te toetsen.

4. U laadt nu het tweede programma. Wanneer u vervolgens LIST intoetst, verschijnt alleen het tweede programma.

5. U schrijft daarna de oude waarden weer in de geheugenplaatsen 43 en 44.

Door deze procedure heeft u twee programma's aan elkaar gekoppeld. Met een machinetaalprogramma gaat dit veel gemakkelijker. Het volgende kleine programma ziet aan de apestaart, als eerste karakter van een filenaam, dat het te laden programma aan het programma in het geheugen moet worden gekoppeld.

Programma beschrijving

Door de initialisering wordt de laadvector naar dit programma verschoven. Daardoor wordt bij elke laadoperatie naar dit programma verwezen. In het programma wordt dan het eerste karakter van de filenaam op een apestaart getest. Vindt het programma dit apestaartje niet, dan springt het in de oude routine. Vindt het dit wel, dan wordt de programmavector min 2 in de laad-start-adressenvector $C3/$C4 overgebracht en wordt de apestaart in de filenaam gewist. Daarna springt het programma weer in de oude laadroutine.

```
25:    c000                    *=   $c000
30:    c000                    .opt p1,o
32:    c000
32:    c000
33:                    ;******** merge **********
34:    c000
34:    c000                                    ;waarden tussen haakjes
35:                    ;   voor    vc 20
36:    c000
36:    c000
40:    0330            vector  =    $330      ;laadvector
50:    f4a5            laden   =    $f4a5     ;($f549) laadroutine
60:    00bb            fileadr =    $bb       ;pointer voor filenaam
70:    00b7            filelng =    $b7       ;filenaamlengte
80:    00c3            startadr =   $c3       ;pointer op prg. start
90:    002d            progend =    $2d       ;pointer op prg. einde
92:    c000
92:    c000
93:    c000
93:    c000
95:                    ;initialiseren v/h
96:                    ;    programma
97:    c000
97:    c000
100:   c000 a9 0b      init    lda  #<start
110:   c002 a2 c0              ldx  #>start
120:   c004 8d 30 03           sta  vector
130:   c007 8e 31 03           stx  vector+1
140:   c00a 60                 rts
142:   c00b
142:   c00b
144:                   ; hoofdprogramma
146:   c00b
146:   c00b
150:   c00b 48         start   pha
160:   c00c a2 00              ldx  #0
170:   c00e a1 bb              lda  (fileadr,x)
180:   c010 c9 40              cmp  #"@"      ;eerste teken '@'
190:   c012 d0 12              bne  einde     ;nee => normaal laden
195:   c014 38                 sec
200:   c015 a5 2d              lda  progend
210:   c017 e9 02              sbc  #2        ;pointer 2 verlagen
220:   c019 85 c3              sta  startadr  ;in laadstartadr.
230:   c01b a5 2e              lda  progend+1
240:   c01d e9 00              sbc  #0
```

```
250:    c01f 85 c4              sta  startadr+1
260:    c021 18                 clc
270:    c022 e6 bb              inc  fileadr  ; '@' wissen
280:    c024 c6 b7              dec  filelng
290:    c026 68         einde   pla
300:    c027 4c a5 f4           jmp  laden
]c000-c02a
```

```
10 REM MERGE C64
20 E=256*PEEK(56)+PEEK(55)-1:A=E-41
30 FORI=ATOE:READX:POKEI,X:NEXT
40 READX,Y,Z:X=A+X:Y=A+Y:Z=A+Z:H=INT(Z/2
56)
45 POKEX,Z-256*H:POKEY,H
46 POKEE-1,PEEK(816):POKEE,PEEK(817)
50 H=INT(A/256):POKE56,H:POKE55,A-256*H
65 SYSA:NEW
32000 DATA169,224,162,127,141,48,3,142,4
9,3,96,72,162,0,161,187,201,64,208,18
32001 DATA56,165,45,233,2,133,195,165,46
,233,0,133,196,24,230,187,198,183,104
32002 DATA76,165,244,1,3,11
```

```
10 REM MERGE VC20
20 E=256*PEEK(56)+PEEK(55)-1:A=E-41
30 FORI=ATOE:READX:POKEI,X:NEXT
40 READX,Y,Z:X=A+X:Y=A+Y:Z=A+Z:H=INT(Z/2
56)
45 POKEX,Z-256*H:POKEY,H
46 POKEE-1,PEEK(816):POKEE,PEEK(817)
50 H=INT(A/256):POKE56,H:POKE55,A-256*H
65 SYSA:NEW
32000 DATA169,224,162,127,141,48,3,142,4
9,3,96,72,162,0,161,187,201,64,208,18
32001 DATA56,165,45,233,2,133,195,165,46
,233,0,133,196,24,230,187,198,183,104
32002 DATA76,73,245,1,3,11
```

10 EEN PROGRAMMA BEDIENT DE CASSETTERECORDER

Tot nu toe hebben we alleen nog maar over het laad- en opslagproces van uw cassetterecorder gesproken. U heeft gezien dat de band automatisch kan worden gestart en gestopt. De computer kan dus de motor bedienen. Onder het motto "Wat de computer kan, kan ik allang" kunt u deze bedieningsmogelijkheid in uw programma benutten. Met de volgende geheugenplaatsen kunt u de motor sturen of vragen een toets van de recorder in te drukken:

VIC 20		Commodore 64		
adres	inhoud	adres	inhoud	functie
37148	252	1	AND 223	motor aan
37148	0	1	OR 32	motor uit
192	0	192	0	motor aan
192	1	192	1	motor uit
37151	64	1	16	toets gedrukt

Wachten op een toets van de recorder :

VIC 20	Commodore 64
WAIT 37151,64,64	WAIT 1,16,16

Als voorbeeld voor deze bedieningsmogelijkheden volgt nu een programma dat voor u een programma op cassette opzoekt. Het stelt een catalogus van alle op uw cassette aanwezige programma's samen en zoekt zelfstandig een programma op. U slaat een of meer programma's op cassette achter het voorbeeldprogramma op. Daarna stelt u met het programma en de variabele TI de tijd vast die de cassetterecorder nodig heeft om van het eind van het catalogusprogramma naar het te laden programma te spoelen. Deze periode wordt met de filenaam van het programma in dataregels opgeslagen. Als u vervolgens een bepaald programma zoekt, blijft de cassette-recorder zolang aan tot de actuele tijdsperiode met de opgeslagen notering overeenkomt. Als u tenslotte de PLAY-toets indrukt, wordt de verlangde file geladen.

Zo bedient u het programma:

Dit programma wordt aan het begin van de cassette opgeslagen. Op een afstand van circa 10 seconden kunt u daarachter de andere programma's opslaan. Als u deze in de catalogus wilt opnemen, noteert u de start-tellernummers en filenamen van de afzonderlijke programma's. Daarna spoelt u de cassette naar het begin terug en laadt het catalogusprogramma. Na het starten van het catalogusprogramma kiest u het punt "invoeren" uit het menu. Allereerst wordt u gevraagd de desbetreffende filenamen in de juiste volgorde in te toetsen. Het programma vraagt u vervolgens de forward-knop van de recorder in te drukken. De recorder start daarna door een druk op de SPACE-toets. U kunt de recorder net zo lang bij de genoteerde tellerstanden laten starten en stoppen tot u daardoor alle ingevoerde programma's een tijdsperiode heeft toegekend. Daarna slaat u het catalogusprogramma weer aan het begin van de cassette op. Nu kunt u - nadat u de opdracht "LADEN" heeft gegeven - een in de catalogus opgeslagen programma kiezen via een menu op het beeldscherm. Als u FFWD (Fast Forward - snel vooruit) indrukt, spoelt de computer naar het gewenste programma. Na het indrukken van de PLAY- en de SPACE-toets wordt het programma geladen.

Programmabeschrijving

In de regels 120 tot en met 170 zitten de subprogramma's voor het starten en stoppen van de motor. Moet het programma op een VIC 20 lopen, dan moeten de REM-opdrachten in de regels 130 en 160 worden weggelaten. Verder moet u in de regels 120 en 150 achter POKE 192,0: een REM toevoegen. In de regels 180 tot en met 340 wordt het programma geïnitialiseerd en wordt het menu op het beeldscherm geschreven. Bij de initialisering worden de namen- en tijdvelden gedimensioneerd en wordt de repeat-functie voor alle toetsen ingeschakeld. Door de bedrijfssysteem-routine SCREEN (65517) wordt het computertype bepaald en overeenkomstig de start van het video- en kleur-RAM gezet. Tenslotte worden de variabelen voor de motorbediening in overeenstemming met het type computer vastgelegd (regels 240 en 260). Naar uw keuze wordt naar "invoeren" of "laden" verwezen.

In de regels 350 tot en met 660 bevindt zich het blok "invoeren". In regel 390 wordt het aantal al ingevoerde files via READ uit de dataregels ingelezen. In de regels 400-420 worden de nieuwe filenamen ingelezen en in veld N$(n) opgeslagen. In de regels 530 tot en met 580 wordt de spoeltijd tussen de startpunten van de

afzonderlijke programma's bepaald en overeenkomstig de toegekende filenamen in het veld ZS(n) opgeslagen. Tevoren wordt in regel 580 een offset van 10 afgetrokken om de uitlooptijd van de motor op te vangen. Om dezelfde reden wordt bij de som van de afzonderlijke intervallen de tijdsperiode 10 opgeteld. Om fouten bij de bediening te vermijden wordt in regel 420 getest of een toets van de recorder is ingedrukt. Zo nodig wordt de gebruiker gevraagd de STOP-toets in te drukken.

In de regels 600-660 worden filenamen en tijden in dataregels geschreven en wordt het programma beëindigd. In regel 670 begint het programmapunt "laden". Het data-getal en de filedata worden ingelezen. In de regels 720-820 worden de files, steeds 8 stuks per pagina, via een 'kleiner dan'-teken, dat via de regels 770-810 wordt gestuurd, op het beeldscherm afgedrukt. Wordt de vraag naar de toets F7 in regel 790 positief beantwoord, dan wordt naar regel 830 (zoek-en laadroutine) verwezen.

Na het testen of er een toets van de recorder is ingedrukt (840), wordt in regel 900 met het WAIT-commando op het indrukken van de FFWD-toets gewacht. Hierna wordt de cassettemotor overeenkomstig de opgeslagen tijd gestart (regel 920-940). In de regels 950-1010 wordt het verlangde programma geladen. Als het programma vanuit de programma-modus zou worden geladen, zouden alleen programma's kunnen worden geladen die kleiner zijn dan het catalogusprogramma. Om dat te vermijden wordt het LOAD-commando, respectievelijk het fasttape LOAD-commando L op het beeldscherm, en RETURN in de toetsenbordbuffer geschreven. Zo wordt het programma vanuit de direkte modus geladen en worden de BASIC-vectoren gezet overeenkomstig het programma dat wordt geladen.

```
5 PRINT"**********************************
**********************************************"
10 GOTO190
20 ****************************************
30 *                                      *
40 *       C A T A L O G U S              *
50 *                                      *
60 *         VOOR CASSETTE                *
70 *                                      *
80 ****************************************
90 :REM
100 REM          WACHTLUS
110 POKE 198,0: WAIT 198,1: GET A$: RETU
RN
120 POKE 192,0: POKE1,PEEK(1) AND 223: R
EM MOTOR AAN C-64
```

```
130 :REM POKE 37148,252 :REM MOTOR AAN V
C20
140 RETURN
150 POKE 192,1: POKE1,PEEK(1) OR 32: REM
 MOTOR UIT C-64
160 :REM POKE 37148,0 :REM MOTOR UIT VC2
0
170 RETURN
180 REM **** INITIALISEREN ****
190 HD$="* * C A T A L O G U S * *"
200 SYS65517: SP=PEEK(781): ZE=PEEK(782)
:REM BEELDSCHERMGEGEVENS HALEN
210 BS=1024:BF=55296:FW=1:IF SP>25 THEN
260
220 BS=4*(PEEK(36866) AND 128)+64*(PEEK(
36869) AND 120)
230 BF=37888+4*(PEEK(36866) AND 128)
240 WZ=37151: WW=64: REM VC20 'WACHTEN O
P RECORDERTOETS
250 FW=6: GOTO 270
260 WZ=1: WW=16: REM C-64 'WACHTEN OP RE
CORDERTOETS
270 DIM N$(50),ZS(50): REM NAAM EN START
POSITIE
280 P1=3*SP+1: P2=2*SP
290 POKE 650,128: REM TOETSEN HERHALEN
300 PRINT"■";: PRINTTAB(SP/2-11)HD$: PRI
NT"■■■■■■■■■"
310 PRINTTAB(SP/2-6)"■I■NVOEREN":PRINT
320 PRINTTAB(SP/2-6)"■L■ADEN"
330 GOSUB 110: IF A$="L" THEN 680
340 IF A$<>"I" THEN 330
350 REM ********* INVOEREN ************
360 PRINT"■■■VOER DE FILENAMEN ":PRINT
370 PRINT"EEN VOOR EEN IN  ";:IF SP<40 T
HEN PRINT:PRINT
380 PRINT"@@ = EINDE"
390 RESTORE:READA:ZA=A
400 INPUT"■FILENAMEN";N$
410 IF N$<>"@@" THEN N$(A)=N$: A=A+1: GO
TO 400
420 ZE=A: A=ZA: IF(PEEK(WZ)AND WW)= WW T
HEN 450
430 PRINT"■DRUK OP DE":PRINT"■■STOP■ TOE
TS"
440 PRINT"■VAN DE RECORDER":WAIT WZ,WW,2
55-WW
450 PRINT"■DRUK DE ";:IF SP<40 THEN PRIN
T:PRINT
```

```
460 PRINT"■F.FWD■ TOETS!"
470 PRINT:PRINT"START DE RECORDER ";:IF
SP<40 THEN PRINT:PRINT;
480 PRINT"■■ DOOR HET DRUKKEN"
490 PRINT:PRINT"VAN EEN TOETS ";:IFSP<40
THENPRINT:PRINT
500 PRINT"EN STOP OP DE "
510 PRINT:PRINT"GEWENSTE PLAATS";:IF SP<
40 THEN PRINT:PRINT:PRINT" ";
520 PRINT" DOOR HET DRUKKEN":PRINT:PRINT
"VAN EEN TOETS"
530 WAITWZ,WW,WW: GOSUB 150: DI=0
540 GOSUB 110:GOSUB 120: R=TI: REM CC AA
N, TIJD MERKEN
550 PRINT"■■■■■■■STOPPEN BIJ START:"
560 PRINT"■"N$(A)
570 GOSUB 110: GOSUB 150: DI=TI-R+DI: RE
M CC UIT, VERSCHIL BEREKENEN
580 ZS(A)=DI-10:DI=DI+10:A=A+1
590 IF A<ZE THEN 540: REM LAATSTE FILENA
ME ?
600 REM * OPSLAAN IN DATA REGELS *
610 PRINT"■■■■2000DATA"A: REM AANTAL IN
DATAREGELS
620 FOR I=ZA TO ZE-1: PRINT2100+I"DATA"N
$(I)","ZS(I)
630 NEXT: PRINT"GOTO650"
640 POKE 198,11:FOR I=0 TO 10: POKE631+I
,13: NEXT: PRINT"■": END
650 PRINT"■SPOEL NU TERUG EN ":PRINT"■ZE
T HET PROGRAMMA ":PRINT"■AAN HET BEGIN"
660 PRINT:PRINT"VAN DE BAND":GOSUB120:EN
D
670 REM ******** LADEN VAN FILES ********
680 RESTORE:READ A$:FOR I=0 TO A-1:READ
N$(I),ZS(I):NEXT
690 PRINT"■";:PRINTTAB(SP/2-11)HD$
700 PRINT"■■■  F1 = NAAR BOVEN":PRINT"■■
  F2 = NAAR BENE.":PRINT"■■  F7 = LADEN"
710 PRINT"■■■■■":PRINTTAB(SP/2-6)"■  S P
A C E  ■":GOSUB 110: ZA=0
720 PRINT"■";:PRINTTAB(SP/2-11)HD$:Z=0:P
RINT
730 FOR I=ZA TO ZA+8:IF N$(I)<>"" THEN P
RINT: PRINTTAB(3)N$(I)
740 NEXT
750 POKE BS+P1+Z*P2,62: POKE BF+P1+Z*P2,
FW
760 GOSUB 110: REM TOETSENBORD UITLEZEN
```

```
770 IF(ASC(A$)=133) AND(Z>0) THEN POKE B
S+P1+Z*P2,32: Z=Z-1: GOTO750
780 IF(ASC(A$)=134)AND(Z<8)AND(N$(ZA+Z+1
)<>"")THENPOKEBS+P1+Z*P2,32:Z=Z+1:GOTO750
790 IFASC(A$)=136THEN830
800 IF ASC(A$)=133 AND Z=0 AND ZA THEN Z
A=ZA-9: GOTO720
810 IF(ASC(A$)=134) AND(N$(I))<>""AND(Z=
8) THEN ZA=ZA+9: GOTO720
820 GOTO 750
830 REM                LADEN
840 IF(PEEK(WZ)ANDWW)= WW THEN 870
850 PRINT"■DRUK DE ":PRINT"■■STOP■ TOETS
"
860 PRINT"■IN ":WAITWZ,WW,255-WW
870 PRINT"■";:PRINTTAB(SP/2-11)HD$:PRINT
:PRINT
880 PRINT"DRUK DE":PRINT"■    ■ F.FWD ■"
890 PRINT:PRINT"TOETS"
900 WAITWZ,WW,WW:REM WACHTEN OP RECORDER
TOETS
910 PRINT" OK "
920 R=TI:R2=R+ZS(Z+ZA)
930 IF R2<TI THEN 950
940 GOTO 930
950 GOSUB 150
960 PRINT"■";:PRINTTAB(SP/2-11)HD$
970 PRINT:PRINT"DRUK DE":PRINT"■      PL
AY    "
980 PRINT:PRINT"TOETS"
990 PRINT"■■■■■":PRINTTAB(SP/2-6)"■  S P
A C E  ■":GOSUB110
1000 PRINT"■■■■←L";CHR$(34)(N$(Z+ZA))CHR
$(34)
1010 POKE198,1:POKE631,13:POKE37148,252:
PRINT"■":END
1020 REM ****** AANTAL PRG'S *********
2000 DATA 3
2005 REM ****** PRG DATA'S **********
2100 DATA, 192
2101 DATA, 55
2102 DATA, 36
```

11 DE HARDWARE VAN DE CASSETTERECORDER

Nu de software afdoende is behandeld, kunnen we ons gaan wijden aan de hardware.

11.1 De verzorging van de cassetterecorder

Iedere cassetterecorder moet verzorgd worden. Het regelmatig voorkomen van "?LOAD ERROR" is meestal het gevolg van een vuile, verkeerd ingestelde of magnetisch geworden kop.

1. Het reinigen van de koppen en het aandrijfwieltje

Hiervoor zijn wattestaafjes zeer geschikt. Ook kunt u een lucifer met een watje gebruiken. Als reinigingsmiddel is alcohol, wasbenzine of spiritus zeer geschikt.

Om de koppen te kunnen reinigen, opent u de cassetterecorder en drukt u op de PLAY-toets. Nu kunt u met het in het reinigingsmiddel gedoopte wattestaafje de koppen (lees/schrijfkop en de uitwiskop) reinigen door voorzichtig over de voorkant van de koppen te wrijven tot ze niet meer bruin afgeven. Hierna wrijft u de koppen met een nieuw staafje droog.

Het verdient aanbeveling tegelijkertijd het gummi aandrijfwieltje schoon te maken, want na intensief gebruik zet zich hier bandslijpsel vast. Hierdoor wordt de band niet meer gelijkmatig langs de opname/weergave-kop geleid. Hiertoe maakt u een niet-pluizende stof (bijvoorbeeld een zakdoek) vochtig met een van de genoemde reinigingsmiddelen. Daarna houdt u deze met een vinger rechts tegen het wieltje, terwijl de cassetterecorder loopt.

2 Demagnetiseren

Om te demagnetiseren gebruikt u een demagnetiseur of een demagnetiserende cassette zoals die voor audio-apparatuur

verkrijgbaar is. De gebruiksaanwijzing vertelt u wat u moet doen.

11.2 De keuze van en de zorg voor de cassettes

Een cassette voor data- en programma-opslag hoeft niet aan de eisen voor een audio-cassette te voldoen. Het is echter belangrijk dat de cassette geen "DROP OUTS" heeft. Dit zijn plaatsen op de band waar de laag van magnetiserende deeltjes gebreken vertoont. Omdat op deze plaatsen niet goed opgenomen kan worden, kan hier informatie verloren gaan. Dit gevaar bestaat vooral bij goedkope cassettes en daarom kunt u beter de wat duurdere merken aanschaffen. Om een te lange speelduur te vermijden is het raadzaam C60-cassettes te gebruiken.

Aan het begin en het eind van een band bevinden zich gedeeltes, die de kracht, die aan het begin of het eind van een band hierop wordt uitgeoefend, op moet vangen. Niettemin kan toch de magneetband aan het begin en einde door deze spanning iets worden uitgerekt. Daarom kunt u de eerste en laatste 10-20 seconden van een bandje beter niet voor data-opslag gebruiken.

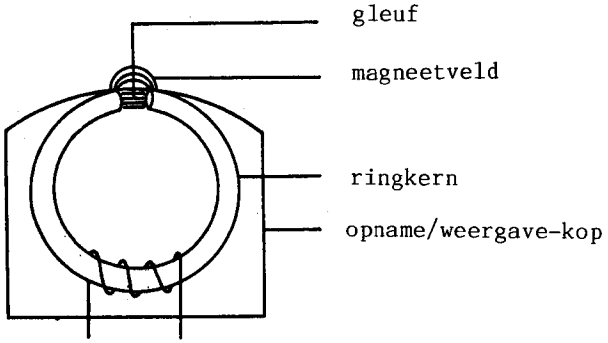
Verder gelden voor de cassetterecorders dezelfde regels als voor recorders voor audio- en videocassettes. Stel ze niet aan magnetische velden bloot en bewaar ze stofvrij. Ook kunt u ze beter niet te lang bewaren (enkele maanden) zonder ze te gebruiken. De magnetische velden van een band kunnen elkaar dan namelijk gaan beïnvloeden. Om programma's tegen invloed van buiten, maar ook tegen uw eigen fouten te beschermen, kunt u het beste van alle belangrijke programma's en data een kopie maken waar u niet mee werkt. Deze kopie legt u vast op een 'schrijfveilige' cassette. Zo kunt u steeds wanneer uw werkkopie wordt vernietigd op een moederband terug vallen.

11.3 De werkwijze van de cassetterecorder

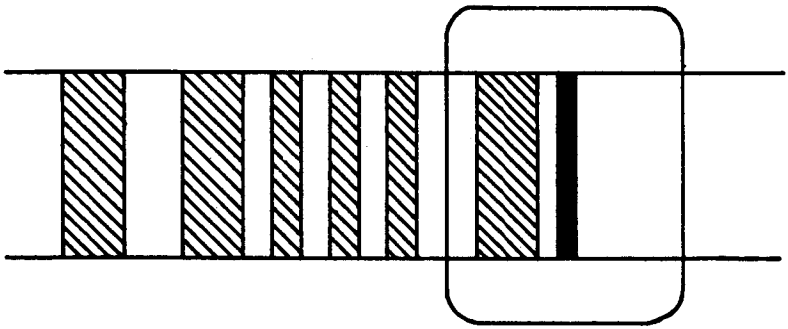
We hebben al gezien dat de informatie in de vorm van golven op de banden wordt vastgelegd. Bij de Commodore 64 gebeurt dat doordat er een spanning van circa 5 V = 1 of 0 V = 0 staat op portbit 3 van de processor (geheugenplaats 1). Bij de VIC 20 is het bit 3 van geheugenplaats 37152. Bij beide is deze aansluiting direkt met de cassetteport-aansluitingen E en 5 verbonden. Het signaal ziet er dan zo uit:



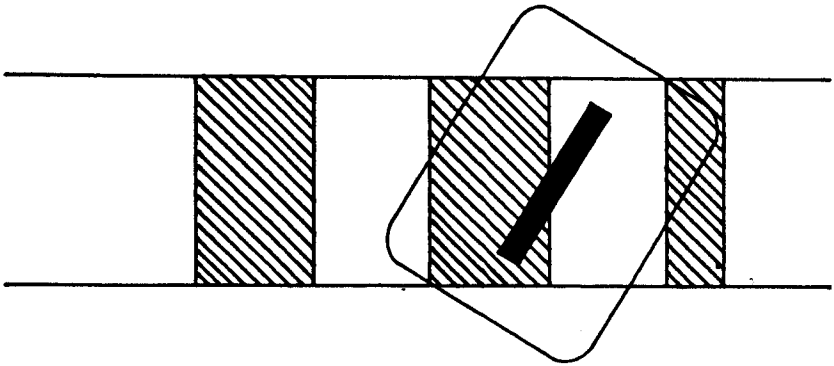
Via een aantal electronische stappen komt het signaal bij de opname/weergave-kop. Deze kop is een spoel met een ringkern die aan de zijde van de band een kleine gleuf heeft.



Overeenkomstig de op de spoelen-aansluitingen aangebrachte spanning ontstaat er een magneetveld bij de gleuf. Dit magneetveld zorgt ervoor dat de magnetiseerbare deeltjes in een bepaalde richting op de band komen te staan. Een volmaakt magnetiseringsvoorbeeld op de band ziet er dan zo uit:



Als u nu de band leest, verloopt het hele proces in omgekeerde richting. Door de gemagnetiseerde deeltjes wordt via de gleuf van de opname/weergave-kop een spanning in de spoel opgewekt die via enige schakelingen weer tot hetzelfde spanningsverloop leidt als bij het opslaan. Deze spanning ligt op de cassettepoort-aansluitingen D en 4. Als nu om een of andere reden de plaats van de opname/weergave-kop op de band bij het lezen anders is dan bij het opslaan, dan krijgt u het volgende beeld:

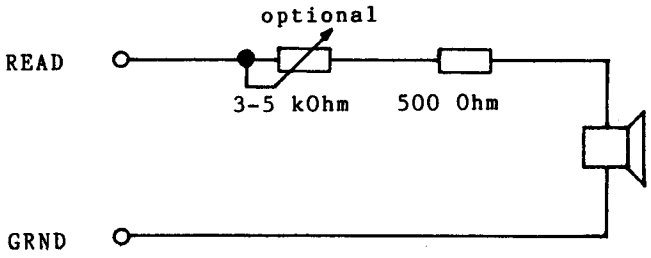


In het bovenste gedeelte van de gleuf is de magnetiseringsrichting anders dan in het onderste gedeelte waardoor de data foutief worden ingelezen.

11.4 Een luidspreker voor de cassetterecorder

Tot nu toe kent u alleen het onbetrouwbare telwerk om de positie van uw band te bepalen. Het zou prettig zijn vast te kunnen stellen of u binnen een programma een ander programma zoekt of dat u precies aan het begin van een programma staat. Zo kunt u sneller een bepaald programma vinden. Dit kunt u doen door een luidspreker op uw cassetterecorder aan te sluiten. Nu kunt u horen waar een programma begint. Door afwisselend, kortstondig vooruitspoelen en afspelen van de band kunt u nu veel beter de start van het volgende programma vinden. Verder is de akoestische "feed-back" bij de instelling van de opname/weergave-kop (zie paragraaf 11.5) een belangrijk hulpmiddel.

Voor aansluiting op een data-cassetterecorder is elke luidspreker geschikt. Omdat het klankgenot niet belangrijk is, kunt u ook van een oude miniradio een luidspreker bouwen. De luidspreker wordt aan de READ-leiding aangesloten. Om uw aansluiting niet teveel te belasten, moet u ook nog een weerstand van circa 500 Ohm in serie schakelen. Verder kunt u nog een potentiometer van 2-5 KOhm in serie schakelen om de geluidssterkte te kunnen regelen.



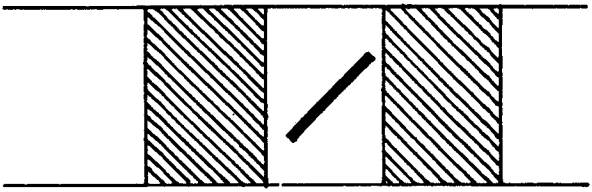
Aansluiting van de luidspreker

De luidspreker kunt u het gemakkelijkst aansluiten op de stekker van de cassetterecorder. Hiervoor opent u de stekker die aan de computer verbonden wordt en klemt u de ene aansluiting aan de GROUND-leiding (A/1) en de andere aan de READ-leiding (D/4).

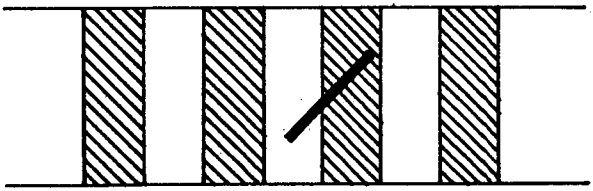
Daarna sluit u de stekker weer en steekt u hem in de cassettepoort van uw computer. Schakel nu de computer in, stop er een bespeelde cassette in en druk op de PLAY-toets. Nu hoort u een eentonige pieptoon of een tsjirpend geruis. De pieptoon betreft de synchronisatie en kenmerkt een blokbegin. Het tsjirpend geruis betreft de opgeslagen data.

11.5 Kopinstelling

Er kunnen verschillende oorzaken zijn, waardoor de opname/weergave-kop niet meer in de juiste positie tot de band staat. Dit kan tot foutief laden leiden. Dat gebeurt bijvoorbeeld als u een programma van een cassette wilt inlezen die niet door uw cassetterecorder werd geladen. De kop kan echter ook zelf na enige tijd verschuiven. Met een luidspreker is deze verschuiving goed te horen. Als de kop ten opzichte van de band ook maar de kleinste hoek maakt, gaat dit ten koste van de hoge frequenties. Aan de hand van twee tekeningen wordt dit duidelijk.

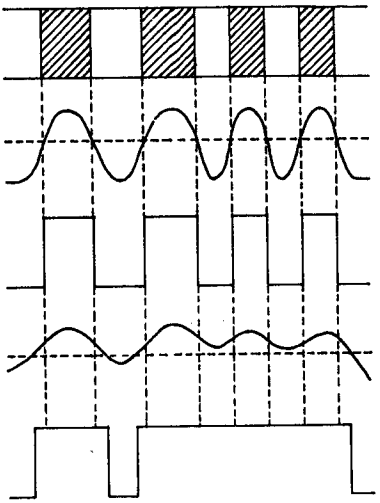


lage frequentie



hoge frequentie

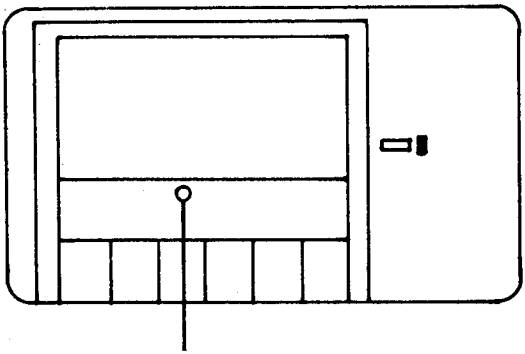
Zoals uit de tekeningen blijkt, zijn er bij lage frequenties nog steeds bandposities waarbij over de hele spleetruimte dezelfde magnetisering werkt. Bij hoge frequenties is er bij de spleet altijd een vermenging van verschillende gemagnetiseerde bereiken. Dus worden dergelijke frequenties onderdrukt.



uitgangssignalen bij verschillende opname/weergave-kopinstellingen

Bij het datalezen krijgt u het beeld uit afbeelding 2. Het signaal dat van de opname/weergave-kop afkomt, is een beetje sinusvormig. Dit signaal wordt binnen de cassetterecorder via een Smith-Trigger-schakeling in voor computers leesbare blokpulsen omgezet. Een Smith-Trigger-schakeling is een schakeling die zich bij een bepaalde spanning, de zogenaamde "golfspanning", in- en uitschakelt en zo een rechthoekspanning bij de uitgang opwekt. Uit afbeelding 2 wordt duidelijk dat door een verkeerd ingestelde kop de tijdsverhoudingen tussen de afzonderlijke golven, die immers de informatie bevatten, niet meer kloppen. Daardoor wordt de band foutief gelezen.

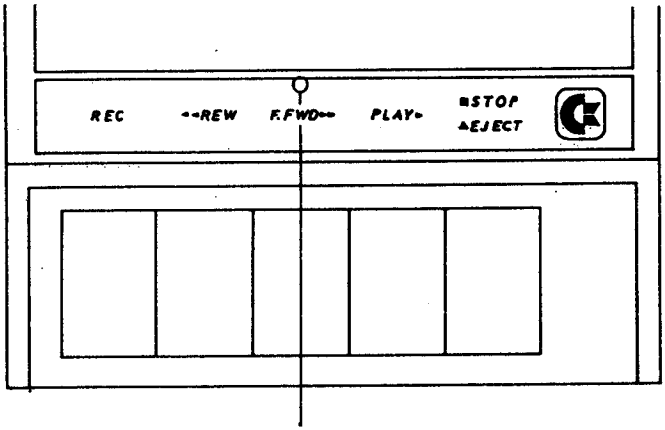
Na deze theoretische inleiding komen we bij het praktische gedeelte. Als u de nieuwste uitvoering van de cassetterecorder heeft, vindt u ongeveer in het midden voor het cassettevak een klein gat.



gat voor de schroef
voor de instelling van
de opname/weergave-kop

De hierna volgende beschrijving en alles in dit boek wat gezegd wordt over het zelf bijregelen van de opname/weergave-kop van uw recorder is niet zonder risico's. U kunt een eventueel van kracht zijnde garantie op uw apparaat teniet doen en onherstelbare schade aanrichten aan uw recorder. De lees/schrijfkop van uw recorder is een vitaal en gevoelig onderdeel van uw installatie, dat u beter met rust kunt laten en voor service in handen van uw dealer geven als u niet ruim voldoende kennis van zaken heeft.

Wie nog durft kan door dit gat met een kleine kruiskopschroevendraaier, als u de PLAY-toets heeft ingedrukt, de schroef voor de instelling van de opname/weergave-kop bereiken. De oude cassetterecorders hebben dit gat niet. Daarom moet u bij deze apparaten zelf een gat in het plastic-omhulsel boren. De precieze plaats vindt u op onderstaande tekening.



Hier moet u het gat boren

Voordat u de kop instelt nog twee waarschuwingen:

1. U kunt de kop zo verkeerd instellen dat het haast niet meer mogelijk is, hem zonder uitgebreide hulpmiddelen (meetapparatuur) weer op de juiste plaats te krijgen. U mag de instellingsschroef niet helemaal losdraaien. Normaal zijn twee slagen in de juiste richting voldoende om de kop correct in te stellen. Meestal is een halve slag ook al voldoende.

2. Nadat u de kop opnieuw geplaatst heeft, is het mogelijk dat u het programma, dat u met de oude instelling heeft opgeslagen niet meer kunt laden.

Om de oorspronkelijke kopinstelling te kunnen terugvinden en een verkeerd ingestelde kop beter te krijgen, gebruikt u het volgende kleine programma. Het schrijft een getsjirp met een groot aantal hoge frequenties op cassette. Omdat juist deze hoge frequenties het eerst worden onderdrukt, kunt u met dit getsjirp de kop-instelling vergemakkelijken en de kop ook preciezer instellen. U voert nu het programma in en slaat het op. Daarna schrijft u vervolgens op een lege cassette circa twee minuten het door dit programma voortgebrachte geluid.

```
10 REM KOPAFSTELLING HULPPROGRAMMA C-64
20 E=256*PEEK(56)+PEEK(55)-1:A=E-17
30 FORI=ATOE:READX:POKEI,X:NEXT
40 PRINT"STARTADRES"A
50 PRINT:PRINT"LEG EEN LEGE CASSETTE IN
EN"
60 PRINT:PRINT"DRUK OP RECORD & PLAY"
70 WAIT1,16,16
80 PRINT:PRINT"EINDE DOOR RUN/STOP TOETS
"
90 SYSA:END
1000 DATA160,255,169,255,162,255,32,177,
251,136,208,246,32,225,255,208,239,96
```

```
10 REM KOPAFTELLING HULPPROGRAMMA VC-20
20 E=256*PEEK(56)+PEEK(55)-1:A=E-17
30 FORI=ATOE:READX:POKEI,X:NEXT
40 PRINT"STARTADRES"A
50 PRINT:PRINT"LEG EEN LEGE CASSETTE IN
EN"
60 PRINT:PRINT"DRUK OP RECORD & PLAY"
70 WAIT37151,64,64
80 PRINT:PRINT"EINDE DOOR RUN/STOP TOETS
"
90 SYSA:END
1000 DATA160,255,169,255,162,255,32,177,
251,136,208,246,32,225,255,208,239,96
```

U spoelt daarna de band terug, pakt de kruiskopschroevendraaier en drukt op de PLAY-toets. U stopt de schroevendraaier recht in het gaatje en draait deze een halve slag naar rechts en naar links vanuit de uitgangspositie en u let hierbij op de klankverschillen. Alleen in de uitgangspositie hoort u de meeste hoge frequenties. Stop hierna een lege cassette in de recorder en herhaal het proces. U zult nu binnen een groot bereik geen klankverschillen meer kunnen constateren. De beste positie tussen de hoorbare veranderingen is hier de middenpositie. Op dezelfde wijze kunt u ook de kop plaatsen op de programma's die u niet kon laden. Probeert u ook hier op het gehoor de beste kopinstelling te vinden. Verandert u de kopinstelling net zolang tot u het programma kunt laden. Als het niettemin toch niet lukt, doe dan wat in paragraaf 4.1 vermeld is.

Voordat u echter het UNNEW-programma laadt of een programmafragment opslaat moet u de opname/weergave-kop met behulp van uw instellingscassette weer in de uitgangspositie brengen.

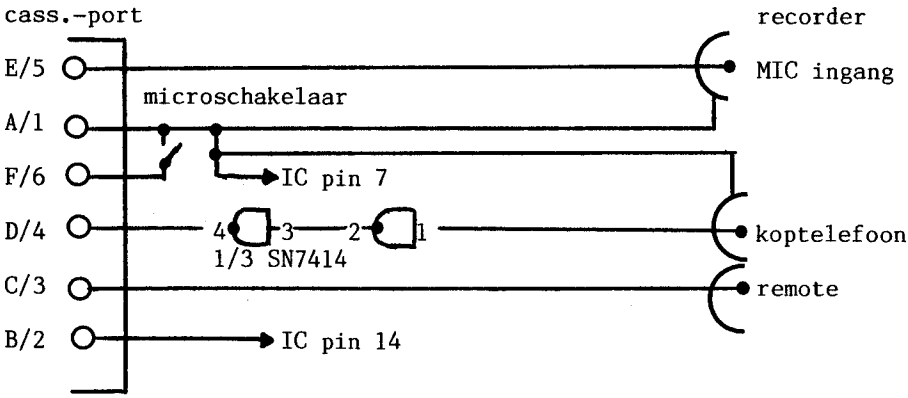
11.6 Andere cassetterecorders voor het opslaan van data

In plaats van de cassetterecorder van de Commodore 64 kunnen we ook andere cassetterecorders gebruiken. Een interface zorgt er hier voor dat van de band komende signalen in voor de computer begrijpelijke rechthoekimpulsen worden omgezet. Dergelijke interface-bouwonderdelen zijn niet duur. Voor de volledigheid volgt hier een beschrijving van een kleine schakeling voor de zelfbouw. De belangrijkste bouwonderdelen zijn:

Bouwonderdelen bestemd voor een interface voor een cassetterecorder:

1 printconnector voor de cassetteport van de VIC 20 of Commodore 64
1 stekker voor de koptelefoon-uitgang van uw cassetterecorder
1 stekker voor de microfoon-ingang van uw cassetterecorder
1 IC SN 7414 (6 inverterende Smith-triggers)
1 microschakelaar (aan/uit)
eventueel 1 stekker voor de REMOTE-ingang van uw recorder

Alleen die recorders zijn geschikt, die met een bedrijfsspanning van 4-6 volt werken en de minpool aan massa hebben. De schakeling vindt u op de volgende pagina.



Beschrijving van de aansluitingen op de cassettepoort:

A/1 = massa
B/2 = +5V
C/3 = cassettemotor (+5 V = een)
D/4 = leesleiding
E/5 = schakelaarleiding voor recordertoets

Het IC SN 7414 bevat zes inverterende Smith-Triggers. Smith-Triggers zijn schakelaars die uit elk willekeurig spanningsverloop een rechthoekimpuls kunnen maken. Hiervan heeft u er twee nodig omdat het hier om omkerende schakelingen gaat. De bedrijfsspanning van deze onderdelen ontlenen we aan de cassettepoort (pin 7-A/1, pin 14-B/2). Als u de cassetterecorder alleen voor uw computer gebruikt, kunt u de schakeling in de batterijruimte bouwen en de microfoon- en koptelefoonleiding vastsolderen. Handig is het, wanneer u een plaats in het mechanisme van de recorder vindt om de microschakelaar zo aan te brengen dat het circuit bij het indrukken van een toets gesloten wordt. Kan dit niet, dan moet u de schakelaar zelf bedienen voordat u een recordertoets indrukt.

12 EEN NIEUW GEBRUIKSSYSTEEM VOOR DE CASSETTERECORDER: "FAST-TAPE"

Tot nog toe is de bediening van de cassetterecorder met het in de computer ingebouwde gebruikssysteem verklaard. U hebt daarbij enige programma's gekregen die het werken met uw datarecorder vereenvoudigen. Nu wordt een nieuw gebruikssysteem geïntroduceerd, dat niet alleen de al beschreven vereenvoudigingen inhoudt, maar ook nog circa 10-20 keer sneller werkt en daarmee zelfs een diskdrive overvleugelt. Vanwege de lengte kost het enige moeite het programma in te toetsen, maar het resultaat mag er zijn. Het programma heeft de volgende kenmerken:

1. Het laadt en slaat de programma's tien keer sneller op dan andere programma's.

2. Het ondersteunt het datageheugen. Data kunnen tot 20 keer sneller worden geladen en opgeslagen dan met het normale gebruikssysteem.

3. Met slechts een commando maakt het een koppeling van BASIC-programma's mogelijk.

4. Ook het laden in of het opslaan van bepaalde geheugenbereiken is met een commando mogelijk.

5. Programma's en data die met FAST-TAPE worden opgeslagen, kunnen zowel door de VIC 20 als door de Commodore 64 worden gelezen.

6. Alle FAST-TAPE commando's werken zowel in direktmodus als in programmamodus.

De bediening van het programma

Na het laden van het programma initialiseert u met SYS7*4096 voor de VIC 20 en met SYS12*4096 voor de Commodore 64. Via de BASIC-lader wordt het programma automatisch geïnitialiseerd. Na een RESET moet u het programma echter opnieuw initialiseren.

Ieder FAST-TAPE commando wordt door een pijl naar links (<-) voorafgegaan. De volgende commando's staan u hiertoe ter beschikking. De tussen haakjes gezette termen zijn niet verplicht en hoeven dus niet te worden gegeven.

SA	Secundair Adres
Sadr, Eadr	Start- en eindadres
"naam"	kan ook als stringvariabele worden gegeven

S("NAAM",SA,SADR, EADR) FAST-TAPE SAVE

S("naam"): BASIC-programma wordt als verschuifbare file in het programma opgeslagen en met L vanaf het BASIC-startadres geladen.

S"naam",1: BASIC-programma wordt als absolute file opgeslagen en met L op dat adres geladen vanwaar het werd weggeschreven

S"naam", 1,Sadr,Eadr:
het geheugenbereik tussen Sadr en Eadr wordt als absolute file opgeslagen

L("NAAM, SA, SADR) FAST-TAPE LOAD

Als dit commando binnen een programma wordt gegeven, start het BASIC-programma analoog aan het LOAD-commando, weer bij de eerste regel.

L("naam"): Een programma of geheugenbereik dat met S wordt opgeslagen, wordt overeenkomstig het bij S gegeven SA absoluut of verschuifbaar geladen.

Wordt dit commando vanuit de direktmodus gegeven, dan worden de BASIC-vectoren overeenkomstig het geladen programma gezet.

L"naam",1: Programma's en geheugenbereiken worden absoluut geladen. Ook vanuit de direktmodus worden door dit commando de BASIC-vectoren niet veranderd. Daarom is een NEW-commando na het laden van een machineprogramma niet noodzakelijk.

L"naam",1,Sadr: Een geheugenbereik wordt op Sadr geladen, onafhankelijk van het adres, vanwaar het werd weggeschreven. Daardoor kunt u eenmaal opgeslagen beeldscherm-informatie ook weer in het video-RAM inlezen, als u het video-RAM heeft verschoven.

V("NAAM",SA,SADR) FAST-TAPE VERIFY

Werkt analoog aan L, vergelijkt echter alleen de geheugeninhoud met de op band aanwezige bytes.

M("NAAM") FAST-TAPE MERGE

Met dit commando kunt u een FAST-TAPE programma koppelen aan een in het geheugen aanwezig programma.

DS"NAAM" DATA-SAVE

Dit commando slaat alle tot nu toe gedefinieerde variabelen, velden en strings op. Met dit commando worden alleen de in het geheugenbereik aanwezige strings opgeslagen. Als uw programma stringopdrachten zoals

```
100 A$="TOETS"
```

bevat, kunt u de desbetreffende strings alleen met hetzelfde programma weer correct laden. Uitkomst biedt hier een verbinding met een lege string. Hiervoor geldt hetzelfde als in paragraaf 5.1 beschreven.

DL"NAAM" DATA-LOAD

Met dit commando laadt u de met DS opgeslagen variabelen, velden en strings. Alle tot aan dit commando gedefinieerde variabelen, velden en strings worden gewist.

Belangrijk!

DS en DL moeten steeds in verbinding met een file-naam worden gebruikt. Door dit commando worden de variabelen, velden en strings in drie blokken opgeslagen of geladen. De file-naam helpt de computer bij de identificatie van de eerste blokken.

Ook belangrijk!

Het BASIC-RAM-eind moet bij DS en de overeenkomstige DL hetzelfde zijn. De stringtabel wordt als absoluut blok opgeslagen en geladen. Als u het einde van de BASIC-RAM naar beneden toe verplaatst nadat u DS heeft gegeven en daarna met DL data laadt, kan eventueel een achter het nieuwe BASIC-RAM-eind geschreven machineprogramma worden vernietigd.

Om u te laten zien hoe gemakkelijk u met deze commando's data kunt verwerken, vindt u in het volgende hoofdstuk een uitvoerig beschreven data-verwerkingsprogramma. Door de BASIC-lader FT64 wordt het programma in de Commodore 64 opgeslagen vanaf geheugenplaats $C000. De BASIC-lader FT20 legt het programma in de VIC 20 vast vanaf geheugenplaats $7000. Dit betekent dat het programma alleen in een VIC 20 werkt als deze volledig is uitgebouwd. Het is echter zonder meer mogelijk met een assembler het FAST-TAPE-programma op een ander geheugenbereik om te schrijven.

```
82:   C000                    .PAG 61,5
90:   C000                    .OPT P2
92:   C000
92:   C000
94:                   ; FASTTAPE VOOR VC 20 EN C-64
95:   C000
95:   C000
95:   C000
96:                   ;WAARDEN TUSSEN HAAKJES VOOR
97:   C000
97:   C000
98:                   ;         VC 20
100:  C000                      *=    $C000
110:  0000            COMPU     =     $00       ; (2)
120:  0001            PORT      =     $1        ; ($9120)
130:  0002            ABSFLG    =     $2        ;
140:  00C0            MOFLAG    =     $C0
150:  00A3            BITC      =     $A3
153:  0004            IPOINT    =     4
155:  005F            CODE      =     "←"
157:  0073            CHRGET    =     $73
160:  0308            VECTOR    =     $308      ; INSTR.ADR.HALEN
165:  02A0            IRQFLG    =     $2A0      ; IRQ FLAG
170:  0090            STATUS    =     $90       ; STATUSBYTE
175:  00C3            STARTV    =     $C3       ; LAADSTARTVECTOR
180:  00AE            ENDVEC    =     $AE       ; LAADEINDVECTOR
185:  00D7            PSOM      =     $D7       ; PROEFSOMBYTE
186:  033C            CASBUF    =     $033C     ; STARTADR. VAN CASSETTEBUFFER
187:  C000
187:  DD00            CIA1      =     $DD00     ; C-64 = STARTADRES VAN CIA1
188:  DC00            CIA2      =     $DC00     ; C-64 = STARTADRES VAN CIA2
189:  C000
189:  C000
190:  F68F            SAVING    =     $F68F     ;($F728) GEEFT SAVING 'NAAM'
195:  F5D2            LOADING   =     $F5D2     ;($F66A) GEEFT 'LOADING' UIT
200:  A82C            STOP      =     $A82C     ; ($C82C) TEST STOPTOETS
220:  FC93            ENDEN     =     $FC93     ; ($FCCF) LADEN BEEINDIGEN
230:  E17A            INSEND    =     $E17A     ; ($E177) TERUGSPRONG NA LADEN
240:  F750            FOUND     =     $F750     ; ($F7D3) GEEFT FOUND 'NAAM'
260:  F5A9            STOEND    =     $F5A9     ; ($F641) EINDADR. NAAR X/Y
300:  A7E7            OLD       =     $A7E7     ; ($C7E7) INSTR. UITWERKEN
310:  A7AE            INTER     =     $A7AE     ; ($A7AE) INTERPRETERLUS
320:  F838            RPTOETS   =     $F838     ; ($F8B7) WACHT REC+PLAY-TOETS
330:  F817            PTOETS    =     $F817     ; ($F894) WACHT PLAY-TOETS
335:  B526            GABCOL    =     $B526     ; ($D526) GARBAGE COLLECTION
340:  E206            VOLGTE    =     $E206     ; ($E203) VOLGENDE TEKEN PRINT
350:  E257            HFNAM     =     $E257     ; ($E254) FILENAAM HALEN
360:  E200            HPARAM    =     $E200     ; ($E1FD) FILEPARAMETER HALEN
365:  E20E            CHRIN     =     $E20E     ; ($E20B) TEKEN HALEN
370:  AF08            SYNTAX    =     $AF08     ; ($CF08) SYNT.ERROR UITVOEREN
380:  AD8A            FRMNUM    =     $AD8A     ; ($CD8A) TERM UITWERKEN
390:  B7F7            FACINT    =     $B7F7     ; ($D7F7) IN INTEGER OMZETTEN
395:  A660            CLEAR     =     $A660     ; ($C660) CLR
400:  A437            FOUTUIT   =     $A437     ; ($C437) FOUT UITGEVEN
410:  A677            STOPPEN   =     $A677     ; ($C677)
420:  C000
420:  C000
430:  FFB7            HSTATU    =     $FFB7     ; STATUS HALEN
440:  FFBD            SETNAM    =     $FFBD     ; FILENAAMPARAMETER INSTELLEN
```

```
450:  FFBA           SETLFS   =    $FFBA   ; FILEPARAMETER INSTELLEN
460:  FFE4           GET      =    $FFE4   ; EEN TEKEN INLEZEN
500:  C000
500:  C000
510:  C000
510:  C000
520:                 ;PROGAMMA INITIALISEREN
530:  C000
530:  C000
560:  C000 A9 0B     INIT     LDA  #<START
570:  C002 8D 08 03           STA  VECTOR
580:  C005 A9 C0              LDA  #>START
590:  C007 8D 09 03           STA  VECTOR+1
600:  C00A 60                 RTS
602:  C00B
602:  C00B
603:  C00B
603:  C00B
604:                 ; PROGRAMMASTART
606:  C00B
606:  C00B
610:  C00B 20 73 00 START    JSR  CHRGET  ; TEKEN HALEN
620:  C00E F0 04             BEQ  EIND
630:  C010 C9 5F             CMP  #CODE   ; MET FASTTAPECODE
640:  C012 F0 03             BEQ  FTAPE   ; VERGELIJKEN
650:  C014 4C E7 A7 EIND     JMP  OLD     ; ONGELIJK => OUDE ROUTINE
654:  C017
654:  C017
655:                 ;FASTTAPE INSTR. UITWERKEN
657:  C017
657:  C017
660:  C017 20 73 00 FTAPE    JSR  CHRGET
670:  C01A 20 20 C0          JSR  ZOEK
680:  C01D 4C AE A7          JMP  INTER
690:  C020 A2 00     ZOEK    LDX  #0
700:  C022 DD 54 C0 ZOEKL    CMP  TAB,X   ; MET INSTRUCTIE-LETTERS
710:  C025 F0 08             BEQ  INSTRN  ; UIT TABEL VERGELIJKEN
720:  C027 E8                INX
730:  C028 E4 04             CPX  IPOINT
740:  C02A D0 F6             BNE  ZOEKL
750:  C02C 4C 08 AF          JMP  SYNTAX  ; GEEN FASTTAPE INSTR.
755:  C02F
755:  C02F
760:  C02F 8A        INSTRN  TXA
770:  C030 0A                ASL
770:  C031 AA                TAX          ; INSTRUCTIENUMMER * 2
780:  C032 BD 5A C0          LDA  INSADR+1,X  ; INSTRUCTIEADRES OP
790:  C035 48                PHA          ; STACK
800:  C036 BD 59 C0          LDA  INSADR,X
800:  C039 48                PHA
810:  C03A 4C 73 00          JMP  CHRGET  ; EN INSTR.UITVOEREN
815:  C03D
815:  C03D
820:  C03D C9 53     DATA    CMP  #"S"    ; GEGEV. OPSLAAN
830:  C03F F0 07             BEQ  DS
840:  C041 C9 4C             CMP  #"L"    ; GEGEV. LADEN
850:  C043 F0 09             BEQ  DL
860:  C045 4C 08 AF          JMP  SYNTAX
865:  C048
```

```
C048
C048 20 73 00 DS        JSR  CHRGET
C04B 4C 11 C3           JMP  DASAV1
C04E 20 73 00 DL        JSR  CHRGET
C051 4C 61 C3           JMP  DATLOD
C054
C054
                ;TABEL INSTRUCTIELETTERS
C054
C054
C054 53 4C 56 TAB       .ASC "SLVMD"
C059
C059
                ;TABEL INSTRUCTIE-LETTERS
C059
C059
C059 62 C0    INSADR    .WORDSAVE-1
C05B 6F C1              .WORDLOAD-1
C05D 72 C1              .WORDVERIFY-1
C05F 61 C1              .WORDMERGE1-1
C061 3C C0              .WORDDATA-1
C063
C063
C063
C063
C063
C063
                ;SAVE ROUTINE
C063
C063
C063 A2 05    SAVE      LDX  #$05     ; AANTAL SYNCHR.
C065 86 AB              STX  $AB      ; HERHALINGEN
C067 A2 00              LDX  #$00     ; FLAG WISSEN
C069 86 02              STX  ABSFLG
C06B 20 B0 C2           JSR  GETPARA  ; PARAMETER HALEN
C06E A5 02              LDA  ABSFLG
C070 29 02              AND  #2       ; TEST OP BIT 2
C072 D0 0B              BNE  ABSOLUT  ; GEZET => ABSOLUUT LADEN
C074 A2 04              LDX  #$04
C076 B5 2A    LOOP1     LDA  $2A,X    ; BASICSTART VERPLAATSEN
C078 95 AB              STA  $AB,X
C07A 95 A6              STA  $A6,X
C07C CA                 DEX
C07D D0 F7              BNE  LOOP1
C07F 20 38 F8 ABSOLUT   JSR  RPTOETS
C082 20 8F F6           JSR  SAVING   ; 'SAVING NAAM' UITVOEREN
C085 20 FC C0           JSR  MOTOR    ; MOTOR INSCHAKELEN
C088
C088
                ;HEADER SCHRIJVEN
C088
C088
C088 20 13 C1           JSR  SYNCH    ; SYNCHRONISATIE 1 SCHRIJVEN
C08B A5 B9              LDA  $B9      ; SECUNDAIRADR. INCREMENTEREN
C08D 18                 CLC
C08E 69 01              ADC  #$01
C090 CA                 DEX
C091 20 33 C1           JSR  WBYTE    ; SCHRIJFT SA
C094 A2 08              LDX  #$08+COMPU
```

```
1850:  C096 B9 A7 00 LOOP2    LDA  $A7,Y   ; SCHRIJFT START- EN EINDADRES
1860:  C099 20 33 C1          JSR  WBYTE
1870:  C09C A2 06             LDX  #$06+COMPU
1880:  C09E C8                INY
1890:  C09F C0 05             CPY  #$05
1900:  C0A1 EA                NOP
1910:  C0A2 D0 F2             BNE  LOOP2
1920:  C0A4 A0 00             LDY  #$00    ;SCHRIJFT FILENAAM EN 'SPATIES'
1930:  C0A6 A2 04             LDX  #$04+COMPU
1940:  C0A8 B1 BB     LOOP3   LDA  ($BB),Y
1950:  C0AA C4 B7             CPY  $B7
1960:  C0AC 90 03             BCC  TEXT
1970:  C0AE A9 20             LDA  #$20
1980:  C0B0 CA                DEX
1990:  C0B1 20 33 C1 TEXT     JSR  WBYTE
2000:  C0B4 A2 05             LDX  #$05
2010:  C0B6 C8                INY
2020:  C0B7 C0 BB             CPY  #$BB
2030:  C0B9 D0 ED             BNE  LOOP3
2034:  C0BB
2034:  C0BB
2035:                ;SYNCHRONISATIE 2 SCHRIJVEN
2036:  C0BB
2036:  C0BB
2040:  C0BB A9 02             LDA  #$02
2050:  C0BD 85 AB             STA  $AB
2060:  C0BF 20 13 C1          JSR  SYNCH   ; SCHRIJFT SYNCHRONISATIE
2070:  C0C2 98                TYA
2080:  C0C3 20 33 C1          JSR  WBYTE   ; 0 BYTE => EINDE HEADER
2090:  C0C6 84 D7             STY  PSOM    ; PROEFSOMBYTE WISSEN
2100:  C0C8 A2 07             LDX  #$07+COMPU
2104:  C0CA
2104:  C0CA
2105:                ;PROGRAMMA SCHRIJVEN
2106:  C0CA
2106:  C0CA
2110:  C0CA EA                NOP
2120:  C0CB B1 AC     PRGLOOP LDA  ($AC),Y  ; PROGRAMMA OP BAND SCHRIJVEN
2130:  C0CD 20 33 C1          JSR  WBYTE
2140:  C0D0 A2 03             LDX  #$03+COMPU
2150:  C0D2 E6 AC             INC  $AC      ; PROGRAMMA POINTER VERHOGEN
2160:  C0D4 D0 04             BNE  NOHI
2170:  C0D6 E6 AD             INC  $AD
2180:  C0D8 CA                DEX
2190:  C0D9 CA                DEX
2200:  C0DA A5 AC     NOHI    LDA  $AC
2210:  C0DC C5 AE             CMP  ENDVEC  ; PROGRAMMEINDE BEREIKT
2220:  C0DE A5 AD             LDA  $AD
2230:  C0E0 E5 AF             SBC  ENDVEC+1
2240:  C0E2 90 E7             BCC  PRGLOOP  ; NEE => VERDER
2250:  C0E4 EA                NOP
2260:  C0E5 A5 D7     LOOP4   LDA  PSOM    ; PROEFSOM SCHRIJVEN
2270:  C0E7 20 33 C1          JSR  WBYTE
2280:  C0EA A2 07             LDX  #$07+COMPU
2290:  C0EC 88                DEY
2300:  C0ED D0 F6             BNE  LOOP4
2310:  C0EF C8                INY
2320:  C0F0 84 C0             STY  MOFLAG  ; MOTOR UIT VOORBEREIDEN
2330:  C0F2 58                CLI
```

```
2340:  C0F3 18                 CLC
2350:  C0F4 A9 00              LDA  #$00
2360:  C0F6 8D A0 02           STA  IRQFLG
2370:  C0F9 4C 93 FC           JMP  ENDEN
2372:  C0FC
2372:  C0FC
2373:                  ;MOTOR STARTEN
2374:  C0FC
2374:  C0FC
2375:  C0FC 20 2C A8 MOTOR     JSR  STOP
2380:  C0FF A0 00              LDY  #$00
2390:  C101 84 C0              STY  MOFLAG
2400:  C103 AD 11 D0           LDA  $D011     ; BEELDSCHERM UIT
2410:  C106 29 EF              AND  #$EF      ; VERVALT VOOR
2420:  C108 8D 11 D0           STA  $D011     ; VC 20
2430:  C10B CA         ANLOOP  DEX            ; WACHTEN TOT MOTOR OP
2440:  C10C D0 FD              BNE  ANLOOP    ; TOEREN IS
2450:  C10E 88                 DEY
2460:  C10F D0 FA              BNE  ANLOOP
2470:  C111 78                 SEI
2480:  C112 60                 RTS
2481:  C113
2481:  C113
2482:                  ; SYNCHRONISATIE GENEREREN
2484:  C113
2484:  C113
2490:  C113 A0 00      SYNCH   LDY  #$00
2500:  C115 A9 02      LOOP5   LDA  #$02      ; STARTBYTE 255-9 * SCHRIJVEN
2510:  C117 20 33 C1           JSR  WBYTE
2520:  C11A A2 07              LDX  #$07+COMPU  ; BYTEWAARDE
2530:  C11C 88                 DEY
2540:  C11D C0 09              CPY  #$09
2550:  C11F D0 F4              BNE  LOOP5
2560:  C121 A2 05              LDX  #$05+COMPU
2570:  C123 C6 AB              DEC  $AB
2580:  C125 D0 EE              BNE  LOOP5
2581:  C127
2581:  C127
2582:                  ; COUNTDOWN SCHRIJVEN
2584:  C127
2584:  C127
2590:  C127 98         COUNTS  TYA            ; COUNTDOWN
2600:  C128 20 33 C1           JSR  WBYTE
2610:  C12B A2 07              LDX  #$07+COMPU
2620:  C12D 88                 DEY
2630:  C12E D0 F7              BNE  COUNTS
2640:  C130 CA                 DEX
2650:  C131 CA                 DEX
2660:  C132 60                 RTS
2662:  C133
2662:  C133
2664:                  ;BYTE OP BAND SCHRIJVEN
2666:  C133
2666:  C133
2670:  C133 85 BD      WBYTE   STA  $BD
2680:  C135 45 D7              EOR  PSOM
2690:  C137 85 D7              STA  PSOM
2700:  C139 A9 08              LDA  #$08
2710:  C13B 85 A3              STA  BITC
```

```
2720:  C13D 06 BD     SHIFT     ASL  $BD
2730:  C13F A5 01               LDA  PORT
2740:  C141 29 F7               AND  #$F7
2750:  C143 20 55 C1            JSR  T1LOOP
2760:  C146 A2 11               LDX  #$11+COMPU
2770:  C148 EA                  NOP
2780:  C149 09 08               ORA  #$08
2790:  C14B 20 55 C1            JSR  T1LOOP
2800:  C14E A2 0E               LDX  #$0E+COMPU
2810:  C150 C6 A3               DEC  BITC
2820:  C152 D0 E9               BNE  SHIFT
2830:  C154 60                  RTS
2840:  C155 CA        T1LOOP    DEX
2850:  C156 D0 FD               BNE  T1LOOP
2860:  C158 90 05               BCC  BIT0     ; CARRY GEZET, DAN
2870:  C15A A2 0B               LDX  #$0B     ; WACHTLUS
2880:  C15C CA        T2LOOP    DEX           ; VERLENGEN
2890:  C15D D0 FD               BNE  T2LOOP
2900:  C15F 85 01     BIT0      STA  PORT
2902:  C161 60                  RTS
2903:  C162
2903:  C162
2904:  C162
2904:  C162
2905:                 ;MERGE
2906:  C162
2906:  C162
2909:                 ;MERGE
2910:  C162 A5 2D     MERGE1    LDA  $2D
2911:  C164 38                  SEC
2912:  C165 E9 02               SBC  #2
2912:  C167 A8                  TAY
2912:  C168 A5 2E               LDA  $2E
2912:  C16A E9 00               SBC  #0
2913:  C16C A2 00               LDX  #0
2914:  C16E F0 09               BEQ  MERGE2
2915:  C170
2915:  C170
2916:  C170
2916:  C170
2917:                 ;LADEN
2918:  C170
2918:  C170
2919:  C170
2919:  C170
2920:  C170 A2 00     LOAD      LDX  #$00
2925:  C172 2C                  .BYTE$2C
2930:  C173 A2 01     VERIFY    LDX  #$01
2940:  C175 A4 2B               LDY  $2B      ; BASICSTARTVECTOREN IN
2950:  C177 A5 2C               LDA  $2C      ; STARTGEHEUGENPLAATSEN
2960:  C179 86 0A     MERGE2    STX  $0A      ; LOAD FLAG (0=LOAD/1=VERIFY)
2970:  C17B 86 93               STX  $93      ; ZETTEN
2975:  C17D A2 00               LDX  #0
2980:  C17F 86 02               STX  ABSFLG   ; ABSOLUUTFLAG WISSEN
2985:  C181 84 C3               STY  STARTV
2990:  C183 85 C4               STA  STARTV+1
3000:  C185 20 B0 C2            JSR  GETPARA
3010:  C188 20 8E C1            JSR  LOADR    ; LOAD EN VERIFY
3020:  C18B 4C 7A E1            JMP  INSEND
```

```
3022:  C18E
3022:  C18E
3024:                    ;LAADROUTINE
3026:  C18E
3026:  C18E
3040:  C18E 20 0A C2 LOADR    JSR  SSYNCH
3050:  C191 A5 AB             LDA  $AB
3060:  C193 C9 02             CMP  #$02      ; IS PRG ABS. OPGESLAGEN
3070:  C195 F0 08             BEQ  ABSOL     ; => ABSOLUUT LADEN
3080:  C197 C9 01             CMP  #$01      ; BYTE NIET 1 => VERDER ZOEKEN
3090:  C199 D0 F3             BNE  LOADR
3100:  C19B A5 B9             LDA  $B9       ; SECUNDAIRADRES = 0
3110:  C19D F0 10             BEQ  RELLOD    ; => VERPLAATSBAAR LADEN
3112:  C19F A9 02     ABSOL   LDA  #2        ; ABSOLUUTFLAG ZETTEN
3113:  C1A1 05 02             ORA  ABSFLG
3114:  C1A3 85 02             STA  ABSFLG
3120:  C1A5 AD 3C 03          LDA  CASBUF    ; STARTADRES UIT BUFFER
3130:  C1A8 85 C3             STA  STARTV    ; IN STARTVECTOR
3140:  C1AA AD 3D 03          LDA  CASBUF+1
3150:  C1AD 85 C4             STA  STARTV+1
3155:  C1AF 20 02 C3 RELLOD   JSR  ADTEST
3156:  C1B2 A5 02             LDA  ABSFLG    ; TEST OP 'WACHTEN'
3157:  C1B4 29 04             AND  #4
3158:  C1B6 D0 08             BNE  NOWAIT
3160:  C1B8 20 50 F7          JSR  FOUND     ; FOUND 'NAAM' UITVOEREN
3170:  C1BB 20 E4 FF WAIT     JSR  GET       ; VERVALT BIJ
3180:  C1BE F0 FB             BEQ  WAIT      ; VC 20
3190:  C1C0 20 2C A8 NOWAIT   JSR  STOP
3200:  C1C3 A4 B7             LDY  $B7       ; FILENAAMLENGTE
3210:  C1C5 F0 0B             BEQ  NONAME
3220:  C1C7 88        TESTNA  DEY            ; FILENAAMTEST
3230:  C1C8 B1 BB             LDA  ($BB),Y
3240:  C1CA D9 41 03          CMP  CASBUF+5,Y
3250:  C1CD D0 BF             BNE  LOADR
3260:  C1CF 98                TYA
3270:  C1D0 D0 F5             BNE  TESTNA
3280:  C1D2 84 90     NONAME  STY  STATUS    ; STATUS WISSEN
3290:  C1D4 20 D2 F5          JSR  LOADING
3300:  C1D7 AD 3E 03          LDA  CASBUF+2  ; EINDADRES BEREKENEN
3310:  C1DA 38                SEC            ; UIT VERSCHIL START+EINDADR.
3320:  C1DB ED 3C 03          SBC  CASBUF    ; UIT HEADER
3330:  C1DE 08                PHP            ; PLUS VASTGESTELDE
3340:  C1DF 18                CLC            ; STARTADRES
3350:  C1E0 65 C3             ADC  STARTV
3360:  C1E2 85 AE             STA  ENDVEC
3370:  C1E4 AD 3F 03          LDA  CASBUF+3
3380:  C1E7 65 C4             ADC  STARTV+1
3390:  C1E9 28                PLP
3400:  C1EA ED 3D 03          SBC  CASBUF+1
3410:  C1ED 85 AF             STA  ENDVEC+1
3420:  C1EF 20 1F C2          JSR  PLOAD
3430:  C1F2 A5 BD             LDA  $BD
3440:  C1F4 45 D7             EOR  PSOM      ; PROEFSOM TESTEN
3450:  C1F6 05 90             ORA  STATUS
3460:  C1F8 F0 04             BEQ  OK
3470:  C1FA A9 FF             LDA  #$FF
3475:  C1FC 85 90             STA  STATUS
3480:  C1FE A5 02      OK     LDA  ABSFLG    ; FLAG TESTEN
3484:  C200 D0 03             BNE  ALTADR    ; GEZET=> OUDE EINDVECTOR
```

```
3486:  C202 4C A9 F5          JMP  STOEND  ; BEWAREN
3488:  C205 A6 2D     ALTADR  LDX  $2D
3490:  C207 A4 2E             LDY  $2E
3492:  C209 60                RTS
3500:  C20A 20 58 C2 SSYNCH   JSR  SCOUNT  ; SYNCHRONISATIE ZOEKEN
3510:  C20D C9 00             CMP  #$00    ; SYNCH VOOR HEADER =>
3520:  C20F F0 F9             BEQ  SSYNCH  ; SYNCH NA HEADER ZOEKEN
3530:  C211 85 AB             STA  $AB     ; SECUNDAIRADRES + 1
3540:  C213 20 86 C2 SPSTART  JSR  HOLBYT  ; START V/H PROGRAMMA
3550:  C216 91 B2             STA  ($B2),Y  ; ZOEKEN
3560:  C218 C8                INY
3570:  C219 C0 C0             CPY  #$C0
3580:  C21B D0 F6             BNE  SPSTART
3590:  C21D F0 2D             BEQ  GEVON   ; EINDE HEADER,PRG LADEN
3600:  C21F 20 58 C2 PLOAD    JSR  SCOUNT  ; PROGRAMMA LADEN
3610:  C222 20 86 C2 LLOOP    JSR  HOLBYT
3620:  C225 C4 93             CPY  $93
3630:  C227 D0 02             BNE  VERGL   ; VERIFY DAN VERGELIJKEN
3640:  C229 91 C3             STA  (STARTV),Y
3650:  C22B D1 C3     VERGL   CMP  (STARTV),Y
3660:  C22D F0 02             BEQ  GELIJK
3670:  C22F 86 90             STX  STATUS
3680:  C231 45 D7     GELIJK  EOR  PSOM    ; PROEFSOM BEREKENEN
3690:  C233 85 D7             STA  PSOM
3700:  C235 E6 C3             INC  STARTV  ; ADRES VERHOGEN
3710:  C237 D0 02             BNE  NOTHI
3720:  C239 E6 C4             INC  STARTV+1
3730:  C23B A5 C3     NOTHI   LDA  STARTV
3740:  C23D C5 AE             CMP  ENDVEC  ; EINDADRES BEREIKT
3750:  C23F A5 C4             LDA  STARTV+1
3760:  C241 E5 AF             SBC  ENDVEC+1
3770:  C243 90 DD             BCC  LLOOP   ; NEE => VERDER
3780:  C245 20 86 C2          JSR  HOLBYT
3790:  C248 20 FC C0          JSR  MOTOR
3800:  C24B C8                INY
3810:  C24C 84 C0     GEVON   STY  MOFLAG
3820:  C24E 58                CLI
3830:  C24F 18                CLC
3840:  C250 A9 00             LDA  #$00
3850:  C252 8D A0 02          STA  IRQFLG
3860:  C255 4C 93 FC          JMP  ENDEN
3870:  C258 20 17 F8 SCOUNT   JSR  PTOETS  ; COUNTDOWN ZOEKEN
3880:  C25B 20 FC C0          JSR  MOTOR
3890:  C25E 84 D7             STY  PSOM
3900:  C260 A9 07             LDA  #$07    ; ($27) WAARDE VOOR TIMER LO
3910:  C262 8D 06 DD          STA  CIA1+6  ; (PORT+8) IN TIMER LO
3920:  C265 A2 01             LDX  #$01    ; WAARDE VOOR TIMER HI
3930:  C267 20 99 C2 SSTART   JSR  HOLBIT  ; START ZOEKEN
3940:  C26A 26 BD             ROL  $BD
3950:  C26C A5 BD             LDA  $BD
3960:  C26E C9 02             CMP  #$02    ; START GEVONDEN
3670:  C22F 86 90             STX  STATUS
3680:  C231 45 D7     GELIJK  EOR  PSOM    ; PROEFSOM BEREKENEN
3690:  C233 85 D7             STA  PSOM
3700:  C235 E6 C3             INC  STARTV  ; ADRES VERHOGEN
3710:  C237 D0 02             BNE  NOTHI
3720:  C239 E6 C4             INC  STARTV+1
3730:  C23B A5 C3     NOTHI   LDA  STARTV
3740:  C23D C5 AE             CMP  ENDVEC  ; EINDADRES BEREIKT
3750:  C23F A5 C4             LDA  STARTV+1
3760:  C241 E5 AF             SBC  ENDVEC+1
```

```
3770: C243 90 DD              BCC  LLOOP   ; NEE => VERDER
3780: C245 20 86 C2           JSR  HOLBYT
3790: C248 20 FC C0           JSR  MOTOR
3800: C24B C8                 INY
3810: C24C 84 C0     GEVON    STY  MOFLAG
3820: C24E 58                 CLI
3830: C24F 18                 CLC
3840: C250 A9 00              LDA  #$00
3850: C252 8D A0 02           STA  IRQFLG
3860: C255 4C 93 FC           JMP  ENDEN
3870: C258 20 17 F8 SCOUNT    JSR  PTOETS  ; COUNTDOWN ZOEKEN
3880: C25B 20 FC C0           JSR  MOTOR
3890: C25E 84 D7              STY  PSOM
3900: C260 A9 07              LDA  #$07    ; ($27) WAARDE VOOR TIMER LO
3910: C262 8D 06 DD           STA  CIA1+6  ; (PORT+8) IN TIMER LO
3920: C265 A2 01              LDX  #$01    ; WAARDE VOOR TIMER HI
3930: C267 20 99 C2 SSTART    JSR  HOLBIT  ; START ZOEKEN
3940: C26A 26 BD              ROL  $BD
3950: C26C A5 BD              LDA  $BD
3960: C26E C9 02              CMP  #$02    ; START GEVONDEN
3970: C270 D0 F5              BNE  SSTART
3980: C272 A0 09              LDY  #$09    ; => COUNTDOWN ZOEKEN
3990: C274 20 86 C2 EIND2     JSR  HOLBYT  ; EINDE V/D '2' - BYTES
4000: C277 C9 02              CMP  #$02    ; ZOEKEN
4010: C279 F0 F9              BEQ  EIND2
4020: C27B C4 BD     COUNTL   CPY  $BD     ; TEST DE COUNTDOWN
4030: C27D D0 E8              BNE  SSTART
4040: C27F 20 86 C2           JSR  HOLBYT
4050: C282 88                 DEY
4060: C283 D0 F6              BNE  COUNTL
4070: C285 60                 RTS
4072: C286
4072: C286
4075:                ; BYTE VAN BAND HALEN
4076: C286
4076: C286
4080: C286 A9 08     HOLBYT   LDA  #$08    ; 8 BIT
4090: C288 85 A3              STA  BITC
4100: C28A 20 99 C2 SHIFT7    JSR  HOLBIT
4110: C28D 26 BD              ROL  $BD     ; SCHUIFT CARRY IN INVOERBUF.
4120: C28F EA                 NOP
4130: C290 EA                 NOP
4140: C291 EA                 NOP
4150: C292 C6 A3              DEC  BITC
4160: C294 D0 F4              BNE  SHIFT7
4170: C296 A5 BD              LDA  $BD
4180: C298 60                 RTS
4182: C299
4182: C299
4184:                ; BIT VAN BAND HALEN VOOR C-64
4185: C299
4185: C299
4190: C299 A9 10     HOLBIT   LDA  #$10
4200: C29B 2C 0D DC WAITDA    BIT  CIA2+13  ;WACHT OP SIGNAAL AAN 'FLAG'
4210: C29E F0 FB              BEQ  WAITDA
4220: C2A0 AD 0D DD           LDA  CIA1+13  ; ICR LADEN. IS TB AL 0
4225:                ; => BIT 1 IS GEZET
4230: C2A3 8E 07 DD           STX  CIA1+7   ; WAARDE IN TB HI
4240: C2A6 48                 PHA           ; ICR OP STACK
4250: C2A7 A9 19              LDA  #$19     ; TB STARTEN, EENMALIG TELLEN
4260: C2A9 8D 0F DD           STA  CIA1+15
4270: C2AC 68                 PLA           ; ICR HALEN
4280: C2AD 4A                 LSR  A        ; BIT 1 IN CARRY VERSCHUIVEN
4290: C2AE 4A                 LSR  A
```

```
4300:  C2AF 60                RTS
4310:  C2B0
4310:  C2B0
4320:                 ; 'HOLBIT' VOOR VC-20
4330:  C2B0
4330:  C2B0
4340:                 ;HOLBIT LDA #$02
4350:                 ;WAITDA BIT PORT+13 ; IFR BIT 1 TESTEN
4360:                 ;       BEQ WAITDA
4370:                 ;       LDA PORT+13
4380:                 ;       STX PORT+9 ; TIMER HI ZETTEN
4390:                 ;       BIT PORT+1
4400:                 ;       ASL
4410:                 ;       ASL
4420:                 ;       ASL
4430:                 ;       RTS
4432:  C2B0
4432:  C2B0
4435:                 ;PARAMETER HALEN
4436:  C2B0
4436:  C2B0
4450:  C2B0 A9 00     GETPARA  LDA  #0
4460:  C2B2 20 BD FF           JSR  SETNAM

PROFI-ASS 64 V2.0     SEITE 10

4470:  C2B5 A2 01              LDX  #1       ; DEFAULT VOOR DN
4480:  C2B7 A0 00              LDY  #00      ; DEFAULT VOOR SA
4490:  C2B9 20 BA FF           JSR  SETLFS
4510:  C2BC 20 06 E2           JSR  VOLGTE   ; NOG MEER TEKENS
4520:  C2BF 20 57 E2           JSR  HFNAM
4540:  C2C2 20 06 E2           JSR  VOLGTE
4550:  C2C5 20 00 E2           JSR  HPARAM
4560:  C2C8 8A                 TXA
4570:  C2C9 A8                 TAY
4580:  C2CA 20 BA FF           JSR  SETLFS
4600:  C2CD 20 06 E2           JSR  VOLGTE
4610:  C2D0 20 F4 C2           JSR  HAALWAA
4620:  C2D3 86 AC              STX  $AC
4622:  C2D5 86 A7              STX  $A7
4624:  C2D7 84 A8              STY  $A8
4630:  C2D9 84 AD              STY  $AD
4632:  C2DB A2 01              LDX  #1       ; FLAG VOOR LADEN EN OPSLAAN
4634:  C2DD 86 02              STX  ABSFLG   ; OP BEPAALDE ADRES REDDEN
4650:  C2DF 20 06 E2           JSR  VOLGTE
4660:  C2E2 20 F4 C2           JSR  HAALWAA
4670:  C2E5 86 AE              STX  $AE
4672:  C2E7 86 A9              STX  $A9
4674:  C2E9 84 AA              STY  $AA
4680:  C2EB 84 AF              STY  $AF
4685:  C2ED A5 02              LDA  ABSFLG
4690:  C2EF 09 02              ORA  #2
4695:  C2F1 85 02              STA  ABSFLG
4700:  C2F3 60                 RTS
4710:  C2F4 20 0E E2 HAALWAA   JSR  CHRIN    ; VOLGENDE WAARDE HALEN
4720:  C2F7 20 8A AD           JSR  FRMNUM   ; GEBRUIKEN EN IN
4730:  C2FA 20 F7 B7           JSR  FACINT   ; 2 BYTE WAARDE OMREKENEN
4740:  C2FD A6 14              LDX  $14
4750:  C2FF A4 15              LDY  $15
4760:  C301 60                 RTS
4770:  C302 A5 02     ADTEST   LDA  ABSFLG   ; FLAG LADEN
4780:  C304 29 01              AND  #1
4790:  C306 F0 08              BEQ  NORM     ; ONGELIJK => NORMAAL LADEN
```

```
4800: C308 A5 AC            LDA  $AC
4810: C30A 85 C3            STA  STARTV
4820: C30C A5 AD            LDA  $AD
4830: C30E 85 C4            STA  STARTV+1
4840: C310 60       NORM    RTS
4940: C311
4940: C311
4945: C311
4945: C311
4950:               ; DATASAVE
4970: C311
4970: C311
5000: C311 20 26 B5 DASAV1  JSR  GABCOL
5010: C314 A2 05            LDX  #5
5020: C316 86 AB            STX  $AB
5030: C318 20 B0 C2         JSR  GETPARA
5040: C31B A9 01            LDA  #1       ; SA=1
5050: C31D 85 B9            STA  $B9
5060: C31F A2 04            LDX  #4
5070: C321 B5 2C    ULOOP   LDA  $2C,X    ; VARIABELE START- EN
5080: C323 95 AB            STA  $AB,X    ; EINDADRESSEN BUFFEREN
5085: C325 95 A6            STA  $A6,X
5090: C327 CA               DEX
5100: C328 D0 F7            BNE  ULOOP
5110: C32A 20 7F C0         JSR  ABSOLUT  ; OPSLAAN
5112: C32D A2 00            LDX  #0
5114: C32F 86 B7            STX  $B7      ; GEEN FILENAAM
5115: C331 A2 05            LDX  #5
5117: C333 86 AB            STX  $AB
5119: C335 CA               DEX
5130: C336 B5 2E    U2LOOP  LDA  $2E,X    ; ARRAY START- EN
5140: C338 95 AB            STA  $AB,X    ; EINDADRESSEN BUFFEREN
5145: C33A 95 A6            STA  $A6,X    ; EINDADRESSEN BUFFEREN
5150: C33C CA               DEX
5160: C33D D0 F7            BNE  U2LOOP
5170: C33F 20 7F C0         JSR  ABSOLUT  ; OPSLAAN
5175: C342 A2 05            LDX  #5
5177: C344 86 AB            STX  $AB
5190: C346 A5 33            LDA  $33      ; STRING START- EN
5200: C348 A6 34            LDX  $34      ;
5210: C34A 85 AC            STA  $AC
5215: C34C 85 A7            STA  $A7
5220: C34E 86 AD            STX  $AD
5225: C350 86 A8            STX  $A8
5230: C352 A5 37            LDA  $37      ; EINDADRESSEN BUFFEREN
5240: C354 A6 38            LDX  $38      ;
5250: C356 85 AE            STA  $AE
5255: C358 85 A9            STA  $A9
5260: C35A 86 AF            STX  $AF
5265: C35C 86 AA            STX  $AA
5270: C35E 4C 7F C0         JMP  ABSOLUT  ; OPSLAAN
5300: C361
5300: C361
5305: C361
5305: C361
5310:               ; DATALOAD
5320: C361
5320: C361
5410: C361 20 B0 C2 DATLOD  JSR  GETPARA
5420: C364 A9 01            LDA  #1       ; OP BESTEMDE ADRES LADEN
5430: C366 85 02            STA  ABSFLG
5440: C368 A5 2D            LDA  $2D      ; VARIABELE STARTADRES ALS
5450: C36A A6 2E            LDX  $2E      ; STARTADRES
```

```
5460:  C36C 85 AC            STA  $AC      ; BUFFEREN
5470:  C36E 86 AD            STX  $AD
5480:  C370 20 8E C1         JSR  LOADR    ; LADEN
5482:  C373 A9 05            LDA  #4+1     ; NIET WACHTEN, OP BESTEMDE
5486:  C375 85 02            STA  ABSFLG   ; ADRES LADEN
5490:  C377 A9 00            LDA  #0
5500:  C379 85 B7            STA  $B7      ; GEEN FILENAAM
5510:  C37B A6 AE            LDX  $AE      ; VARIABELEN EINDE
5520:  C37D A4 AF            LDY  $AF
5530:  C37F 86 2F            STX  $2F      ; IN VECTOR
5540:  C381 84 30            STY  $30
5550:  C383 86 AC            STX  $AC      ; EN LAADSTARTVECTOR
5560:  C385 84 AD            STY  $AD
5570:  C387 20 8E C1         JSR  LOADR    ; LADEN
5580:  C38A A6 AE            LDX  $AE      ; ARRAYEINDE
5590:  C38C A4 AF            LDY  $AF
5600:  C38E 86 31            STX  $31      ; IN VECTOR
5610:  C390 84 32            STY  $32
5620:  C392 A9 04            LDA  #4       ; VAN ADRES UIT HEADER
5630:  C394 85 02            STA  ABSFLG   ; LADEN
5640:  C396 20 8E C1         JSR  LOADR    ; LADEN
5642:  C399 AD 3C 03         LDA  CASBUF
5642:  C39C 85 33            STA  $33
5644:  C39E AD 3D 03         LDA  CASBUF+1
5644:  C3A1 85 34            STA  $34
5650:  C3A3 20 B7 FF         JSR  HSTATU   ; STATUS HALEN
5660:  C3A6 25 BF            AND  $BF      ; EOF-BIT WISSEN
5670:  C3A8 F0 05            BEQ  NOFOUT   ; GEEN FOUT
5680:  C3AA A2 1D            LDX  #$1D     ; OFFSET VOOR 'LOAD ERROR'
5690:  C3AC 4C 37 A4         JMP  FOUTUIT  ; FOUT UITVOEREN
5700:  C3AF A2 19     NOFOUT  LDX  #$19     ; STRING-DESCRIPTOR
5710:  C3B1 86 16            STX  $16      ; INDEX TERUGZETTEN
5720:  C3B3 A9 00            LDA  #00
5730:  C3B5 85 3A            STA  $3A
5730:  C3B7 85 10            STA  $10      ; ASLUITEN EN
5740:  C3B9 60               RTS           ; TERUG NAAR HOOFDPROGRAMMA
]C000-C3BA
```

```
100 REM BASICLOADER FT VC-20
110 E=29611:A=28672:PS=0
120 FORI=ATOE:READX:POKEI,X:PS=PS+X:NEXT
130 IFPS<>114760THENPRINT"FOUT IN DATA":
END
140 POKE55,0:POKE56,112
150 SYSA:NEW
160 DATA169,11,141,8,3,169,112,141,9,3,9
6,32,115,0,240,4,201,95,240,3,76,231
170 DATA199,32,115,0,32,32,112,76,174,19
9,162,0,221,84,112,240,8,232,228,4
180 DATA208,246,76,8,207,138,10,170,189,
90,112,72,189,89,112,72,76,115,0,201
190 DATA83,240,7,201,76,240,9,76,8,207,3
2,115,0,76,3,115,32,115,0,76,83,115
200 DATA83,76,86,77,68,98,112,105,113,10
8,113,91,113,60,112,162,5,134,171,162
210 DATA0,134,2,32,162,114,165,2,41,2,20
8,11,162,4,181,42,149,171,149,166,202
220 DATA208,247,32,183,248,32,40,247,32,
252,112,32,11,113,165,185,24,105,1
230 DATA202,32,43,113,162,10,185,167,0,3
2,43,113,162,8,200,192,5,234,208,242
240 DATA160,0,162,6,177,187,196,183,144,
3,169,32,202,32,43,113,162,5,200,192
250 DATA187,208,237,169,2,133,171,32,11,
113,152,32,43,113,132,215,162,9,234
260 DATA177,172,32,43,113,162,5,230,172,
208,4,230,173,202,202,165,172,197,174
270 DATA165,173,229,175,144,231,234,165,
215,32,43,113,162,9,136,208,246,200
280 DATA132,192,88,24,169,0,141,160,2,76
,207,252,32,44,200,160,0,132,192,202
290 DATA208,253,136,208,250,120,96,160,0
,169,2,32,43,113,162,9,136,192,9,208
300 DATA244,162,7,198,171,208,238,152,32
,43,113,162,9,136,208,247,202,202,96
310 DATA133,189,69,215,133,215,169,8,133
,163,6,189,173,32,145,41,247,32,78
320 DATA113,162,19,234,9,8,32,78,113,162
,16,198,163,208,232,96,202,208,253
330 DATA144,5,162,11,202,208,253,141,32,
145,96,165,45,56,233,2,168,165,46,233
340 DATA0,162,0,240,9,162,0,44,162,1,164
,43,165,44,134,10,134,147,162,0,134
350 DATA2,132,195,133,196,32,162,114,32,
136,113,76,119,225,32,255,113,165,171
360 DATA201,2,240,8,201,1,208,243,165,18
5,240,16,169,2,5,2,133,2,173,60,3,133
```

```
370 DATA195,173,61,3,133,196,32,244,114,
165,2,41,4,208,3,32,211,247,32,44,200
380 DATA164,183,240,11,136,177,187,217,6
5,3,208,196,152,208,245,132,144,32
390 DATA106,246,173,62,3,56,237,60,3,8,2
4,101,195,133,174,173,63,3,101,196
400 DATA40,237,61,3,133,175,32,20,114,16
5,189,69,215,5,144,240,4,169,255,133
410 DATA144,165,2,208,3,76,65,246,166,45
,164,46,96,32,77,114,201,0,240,249
420 DATA133,171,32,123,114,145,178,200,1
92,192,208,246,240,45,32,77,114,32
430 DATA123,114,196,147,208,2,145,195,20
9,195,240,2,134,144,69,215,133,215
440 DATA230,195,208,2,230,196,165,195,19
7,174,165,196,229,175,144,221,32,123
450 DATA114,32,252,112,200,132,192,88,24
,169,0,141,160,2,76,207,252,32,148
460 DATA248,32,252,112,132,215,169,39,14
1,40,145,162,1,32,142,114,38,189,165
470 DATA189,201,2,208,245,160,9,32,123,1
14,201,2,240,249,196,189,208,232,32
480 DATA123,114,136,208,246,96,169,8,133
,163,32,142,114,38,189,234,234,234
490 DATA198,163,208,244,165,189,96,169,2
,44,45,145,240,251,173,45,145,142,41
500 DATA145,44,33,145,10,10,10,96,169,0,
32,189,255,162,1,160,0,32,186,255,32
510 DATA3,226,32,84,226,32,3,226,32,253,
225,138,168,32,186,255,32,3,226,32
520 DATA230,114,134,172,134,167,132,168,
132,173,162,1,134,2,32,3,226,32,230
530 DATA114,134,174,134,169,132,170,132,
175,165,2,9,2,133,2,96,32,11,226,32
540 DATA138,205,32,247,215,166,20,164,21
,96,165,2,41,1,240,8,165,172,133,195
550 DATA165,173,133,196,96,32,38,213,162
,5,134,171,32,162,114,169,1,133,185
560 DATA162,4,181,44,149,171,149,166,202
,208,247,32,127,112,162,0,134,183,162
570 DATA5,134,171,202,181,46,149,171,149
,166,202,208,247,32,127,112,162,5,134
580 DATA171,165,51,166,52,133,172,133,16
7,134,173,134,168,165,55,166,56,133
590 DATA174,133,169,134,175,134,170,76,1
27,112,32,162,114,169,1,133,2,165,45
600 DATA166,46,133,172,134,173,32,136,11
3,169,5,133,2,169,0,133,183,166,174
610 DATA164,175,134,47,132,48,134,172,13
```

```
2,173,32,136,113,166,174,164,175,134
620 DATA49,132,50,169,4,133,2,32,136,113
,173,60,3,133,51,173,61,3,133,52,32
630 DATA183,255,37,191,240,5,162,29,76,5
5,196,162,25,134,22,169,0,133,58,133
640 DATA16,96



1000 REM LOADER FT64
1010 E=50105:A=49152:PS=0
1020 FORI=ATOE:READX:POKEI,X:PS=PS+X:NEX
T
1030 IFPS<>120504THENPRINT"FOUT IN DATA"
:END
1040 SYSA
10000 DATA169,11,141,8,3,169,192,141,9,3
,96,32,115,0,240,4,201,95,240,3,76,231
10010 DATA167,32,115,0,32,32,192,76,174,
167,162,0,221,84,192,240,8,232,228,4
10020 DATA208,246,76,8,175,138,10,170,18
9,90,192,72,189,89,192,72,76,115,0,201
10030 DATA83,240,7,201,76,240,9,76,8,175
,32,115,0,76,17,195,32,115,0,76,97,195
10040 DATA83,76,86,77,68,98,192,111,193,
114,193,97,193,60,192,162,5,134,171,162
10050 DATA0,134,2,32,176,194,165,2,41,2,
208,11,162,4,181,42,149,171,149,166,202
10060 DATA208,247,32,56,248,32,143,246,3
2,252,192,32,19,193,165,185,24,105,1
10070 DATA202,32,51,193,162,8,185,167,0,
32,51,193,162,6,200,192,5,234,208,242
10080 DATA160,0,162,4,177,187,196,183,14
4,3,169,32,202,32,51,193,162,5,200,192
10090 DATA187,208,237,169,2,133,171,32,1
9,193,152,32,51,193,132,215,162,7,234
10100 DATA177,172,32,51,193,162,3,230,17
2,208,4,230,173,202,202,165,172,197,174
10110 DATA165,173,229,175,144,231,234,16
5,215,32,51,193,162,7,136,208,246,200
10120 DATA132,192,88,24,169,0,141,160,2,
76,147,252,32,44,168,160,0,132,192,173
10130 DATA17,208,41,239,141,17,208,202,2
08,253,136,208,250,120,96,160,0,169,2
10140 DATA32,51,193,162,7,136,192,9,208,
244,162,5,198,171,208,238,152,32,51,193
10150 DATA162,7,136,208,247,202,202,96,1
33,189,69,215,133,215,169,8,133,163,6
```

```
10160 DATA189,165,1,41,247,32,85,193,162
,17,234,9,8,32,85,193,162,14,198,163
10170 DATA208,233,96,202,208,253,144,5,1
62,11,202,208,253,133,1,96,165,45,56
10180 DATA233,2,168,165,46,233,0,162,0,2
40,9,162,0,44,162,1,164,43,165,44,134
10190 DATA10,134,147,162,0,134,2,132,195
,133,196,32,176,194,32,142,193,76,122
10200 DATA225,32,10,194,165,171,201,2,24
0,8,201,1,208,243,165,185,240,16,169
10210 DATA2,5,2,133,2,173,60,3,133,195,1
73,61,3,133,196,32,2,195,165,2,41,4,208
10220 DATA8,32,80,247,32,228,255,240,251
,32,44,168,164,183,240,11,136,177,187
10230 DATA217,65,3,208,191,152,208,245,1
32,144,32,210,245,173,62,3,56,237,60
10240 DATA3,8,24,101,195,133,174,173,63,
3,101,196,40,237,61,3,133,175,32,31,194
10250 DATA165,189,69,215,5,144,240,4,169
,255,133,144,165,2,208,3,76,169,245,166
10260 DATA45,164,46,96,32,88,194,201,0,2
40,249,133,171,32,134,194,145,178,200
10270 DATA192,192,208,246,240,45,32,88,1
94,32,134,194,196,147,208,2,145,195,209
10280 DATA195,240,2,134,144,69,215,133,2
15,230,195,208,2,230,196,165,195,197
10290 DATA174,165,196,229,175,144,221,32
,134,194,32,252,192,200,132,192,88,24
10300 DATA169,0,141,160,2,76,147,252,32,
23,248,32,252,192,132,215,169,7,141,6
10310 DATA221,162,1,32,153,194,38,189,16
5,189,201,2,208,245,160,9,32,134,194
10320 DATA201,2,240,249,196,189,208,232,
32,134,194,136,208,246,96,169,8,133,163
10330 DATA32,153,194,38,189,234,234,234,
198,163,208,244,165,189,96,169,16,44
10340 DATA13,220,240,251,173,13,221,142,
7,221,72,169,25,141,15,221,104,74,74
10350 DATA96,169,0,32,189,255,162,1,160,
0,32,186,255,32,6,226,32,87,226,32,6
10360 DATA226,32,0,226,138,168,32,186,25
5,32,6,226,32,244,194,134,172,134,167
10370 DATA132,168,132,173,162,1,134,2,32
,6,226,32,244,194,134,174,134,169,132
10380 DATA170,132,175,165,2,9,2,133,2,96
,32,14,226,32,138,173,32,247,183,166
10390 DATA20,164,21,96,165,2,41,1,240,8,
165,172,133,195,165,173,133,196,96,32
10400 DATA38,181,162,5,134,171,32,176,19
```

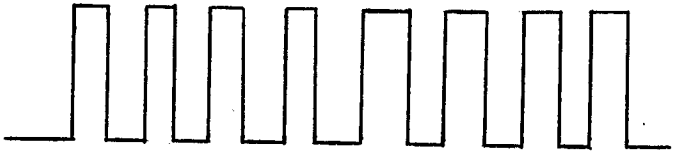
```
4,169,1,133,185,162,4,181,44,149,171
10410 DATA149,166,202,208,247,32,127,192
,162,0,134,183,162,5,134,171,202,181
10420 DATA46,149,171,149,166,202,208,247
,32,127,192,162,5,134,171,165,51,166
10430 DATA52,133,172,133,167,134,173,134
,168,165,55,166,56,133,174,133,169,134
10440 DATA175,134,170,76,127,192,32,176,
194,169,1,133,2,165,45,166,46,133,172
10450 DATA134,173,32,142,193,169,5,133,2
,169,0,133,183,166,174,164,175,134,47
10460 DATA132,48,134,172,132,173,32,142,
193,166,174,164,175,134,49,132,50,169
10470 DATA4,133,2,32,142,193,173,60,3,13
3,51,173,61,3,133,52,32,183,255,37,191
10480 DATA240,5,162,29,76,55,164,162,25,
134,22,169,0,133,58,133,16,96
```

12.1 Programmabeschrijving

Voor die lezers die enige basiskennis van het programmeren in assembler hebben, zal ik nader op het programma FAST-TAPE ingaan. Misschien verzint u nog enige verbeteringen die u in het programma wilt aanbrengen. Voor verdere literatuur verwijs ik u naar de desbetreffende hoofdstukken over de I/O-bouwstenen in de boeken van DATA BECKER/NEDERLANDS* zoals "COMMODORE 64 INTERN", "HET MACHINETAALBOEK 64" en "HET MACHINETAALBOEK 64 VOOR GEVORDERDEN". Bijzondere aandacht verdient daarbij het assembler en monitorprogramma in het softwarepakket "PROFIMAT 64" dat ook in een Nederlandse uitvoering verkrijgbaar is.

De principiële werkwijze

Afzonderlijke bits worden met een blokgolf op band geschreven. Byte 20, binair %00010100 ziet er als volgt uit:



waarde	0	0	0	1	0	1	0	0
bit nr.	7	6	5	4	3	2	1	0

In het volgende opslag-format wordt een programma opgeslagen:

1. Synchronisatie 1
   - 5 keer(246 keer byte 2). Hierdoor wordt bij het laadproces op het begin van een byte gesynchroniseerd.
   - Door de aftel- bytes 9, 8, 7, 6, 5, 4, 3, 2 en 1 wordt het einde van de synchronisatie aangegeven.

   Aanloop-record bestaande uit:
   - secundair adres-1
   - start- en eindadres
   - filenamen
   - Het aanloop-record wordt telkens tot 192 karakters met "SPACE" opgevuld en bij het laden in de cassette-buffer opgeslagen. Daarom kunt u ook hier met de filenaam machineprogramma's overdragen zoals in paragraaf 7.3 werd beschreven.

2. Synchronisatie 2
   - zoals synchronisatie 1, echter slechts 2 keer

3. Nul-byte
   - duidt aan dat er nu data volgen

4. Data

5. Toets- of proefsombyte, 255 keer

In het sub-programma INIT wordt de vector naar de interpretatie-lus van het FAST-TAPE programma omgebogen. Hierdoor wordt steeds als er een commando moet worden verwerkt, naar FAST-TAPE gesprongen. In het sub-programma START wordt getest of het eerste karakter de FAST-TAPE-code (pijl naar links) is.

Zoniét, dan wordt naar de interpretatie-lus teruggesprongen. Wordt de FAST-TAPE-code gevonden, dan vergelijkt het sub-programma ZOEK deze met de letter-tabel 'TAB' en bepaalt het desbetreffende startadres. Wordt het karakter niet gevonden, dan volgt er een "? SYNTAX ERROR". Anders wordt het desbetreffende commando-start-adres op de STACK geschoven. Na de RTS van de CHRGET-routine springt het programma naar het adres+1 en keert na de uitvoering naar het adres $C01D ($701D) terug.

Voor de afzonderlijke hoofdprogramma's worden enkele vaak gebruikte sub-programma's beschreven.

Byte op de band schrijven: WBYTE $C133

De overdracht van de byte vindt plaats via de accumulator. Deze byte wordt op de geheugenplaats $BD geschreven en daarna met de toetsbyte door middel van 'Exclusief of' gekoppeld. In $C139 wordt 8 (een byte = 8 bit) in de bitcounter geschreven. Door SHIFT begint de lus die 8 bits op de band schrijft. Eerst wordt de bit met de hoogste waarde in de carry-flag geschoven. De PORT-byte wordt nu in de accu geladen en de schrijf-byte (bit 3) gewist. Bij T1LOOP (bij gezette carry-flag bovendien T2LOOP) is de schrijflijn nu "high". Aan het eind van de tijdlus wordt de accu-inhoud in de port overgedragen met als gevolg dat de schrijflijn "low" wordt. In regel $C149 wordt dan in de accu weer bit 3 gezet en opnieuw naar de tijdloop T1LOOP vertakt. Aan het eind wordt door de overdracht van de accu-inhoud in de port de schrijflijn weer "high". In $C150 wordt de bitcounter gedecrementeerd en getest. Na de overdracht van alle 8 bits wordt teruggesprongen.

Byte van de band lezen: HALBYT $C286

Allereerst wordt vanaf regel $C286 de bitcounter op 8 gezet. Daarna worden 8 bits na elkaar via HALBIT geladen en via de carry-flag in $BD geschoven. Aan het eind (regel $C269) wordt via de accu de ingelezen byte overgedragen.

Bit van de band lezen: HALBIT $C299

De lus $C299-$C29E wacht tot bit 4 in de ICR2 van de CIA2 "high" wordt. Deze bit wordt gezet als er op de READ-leiding (D/4) van de cassette-port een signaal is. Daarna wordt de ICR1 geladen, het X-register als highbyte in de teller B van CIA2 geschreven, en ICR1 op de STACK gered. In $C2A9 worden bit 1,3 en 4 in het controleregister B gezet, wat tot gevolg heeft dat de teller B een keer terug telt en wordt gestart. Daarna wordt de geredde ICR1-inhoud teruggehaald en door twee keer naar rechts te shiften wordt bit 1 in de carry-flag geschoven. Deze bit wordt "high" als de teller tot nul heeft geteld voordat hij opnieuw wordt gestart. Op deze manier wordt vastgesteld of het lang duurt voordat de leeslijn weer "high" is, wat met een gezette bit overeenkomt.

Parameter halen: GETPARA $C2B0

Deze routine verzamelt de door de commando's overgedragen parameters en schrijft ze op de desbetreffende geheugenplaats. Verder wordt in overeenstemming met de parameter de flag ABSFLG gezet. Deze flag is een binaire flag en stuurt de volgende functies:

bit	waarde	werking
0	1	1 = file in een bij L-instructie overgedragen adres laden
1	2	1 = file absoluut laden of opslaan
2	4	1 = bij C64 niet wachten, maar direkt laden

Synchronisatie schrijven: SYNCH $C113

Deze subroutine zet net zo vaak byte 2 op de band als in $AB staat aangegeven (246 maal). Als afsluiting wordt een countdown uit de bytes 9, 8, 7, 6, 5, 4, 3, 2, 1 op de band gezet.

Synchronisatie zoeken: SCOUNT $C258

Eerst wordt de RECORD-toets gevraagd. Daarna wordt de motor gestart, het beeldscherm uitgeschakeld (alleen bij de Commodore-64) en de toetsbyte op nul gezet. Vanaf regel $C260 wordt 7 in de lowbyte geschreven en het X-register voor de highbyte van de timer met lgeladen. Vanaf SSTART worden dan net zolang bits gelezen tot de byte-waarde van de ingelezen bits precies 2 wordt. Daarna worden alle volgende bytes met de waarde 2 van de synchronisatie ingelezen en wordt bepaald of er een aftellen vanaf 9 volgt. Bij foutloos aftellen wordt naar het hoofdprogramma teruggesprongen.

Aanloop-record inlezen: SSYNCH $C20A

Dit sub-programma zoekt met SCOUNT naar de eerstvolgende synchronisatie. Met behulp van de laatste door SCOUNT ingelezen bytes wordt bepaald of het hier om synchronisatie 1 of 2 gaat. Is de gevonden synchronisatie niet synchronisatie 1, dan wordt verder gezocht. Anders wordt de laatste byte als secundair adres in $AB gered. Vervolgens wordt het aanloop-record ingelezen en in de cassettebuffer opgeslagen. Daarna wordt de motor via de gebruikssysteem-routine vanaf adres $FC93 uitgeschakeld en bij de Commodore 64 het beeldscherm weer ingeschakeld.

Programma laden: PLOAD $C21F

Door dit sub-programma wordt het programma ingelezen. Allereerst wordt met SCOUNT synchronisatie 2 gezocht en daarmee de leesroutine op de band gesynchroniseerd.

LLOOP is de eigenlijke laadroutine. Met HALBYT wordt een byte ingelezen en via de start-vector op indirekt geïndexeerde wijze in het geheugen geschreven en vergeleken. Bij V (VERIFY) wordt slechts vergeleken. Treedt hierbij een fout op, dan wordt de status-byte gezet. Vervolgens wordt de EXOR-toetsbyte bijgewerkt en de startvector geïncrementeerd. De lus LLOOP wordt net zolang doorlopen tot het verschil tussen start- en eindvector gelijk is aan nul. Daarna wordt de opgeslagen toetsbyte ingelezen, de motor uitgeschakeld en bij de Commodore 64 het beeldscherm ingeschakeld. Via de gebruikssysteem-routine $FC93 wordt naar het

hoofdprogramma teruggesprongen.

En de hoofdroutines tenslotte:

SAVE routine $C063

In $AB wordt eerst het herhalingsgetal van synchronisatie 1 opgeslagen en flag ABSFLG gewist. Daarna worden via GETPARA de afzonderlijke file-parameters ingelezen. Overeenkomstig ABSFLG worden dan de BASIC-vectoren in de laad-, start- en eindvectoren overgedragen. In $C07F wordt de RECORD-toets gevraagd, "SAVING NAAM" gegeven, de motor gestart en bij de Commodore 64 het beeldscherm uitgeschakeld. Via het sub-programma SYNCH wordt de synchronisatie op band geschreven. Daarna wordt het met 1 verhoogde secundaire adres op de band overgebracht. Vervolgens wordt vanaf $C094 het aanloop-record bestaande uit start- en eindadres, filenaam en vulbytes("SPACE"), op de band geschreven. Tenslotte volgt synchronisatie 2 die met een nul-byte wordt afgesloten. Vanaf $C0CB (PRGLOOP) volgt de opslagroutine van het eigenlijke programma. Het programma wordt via de programma-startvector $AC/$AD indirekt geïndexeerd uit het geheugen gelezen en aan de cassetterecorder overgedragen. Daarna wordt de vector $AC/AD geïncrementeerd en met de eindvector van het programma vergeleken. Deze lus wordt net zolang doorlopen tot de startvector gelijk is aan de eindvector. Ter afsluiting wordt dan 255 keer de EXOR-toetsbyte op de band geschreven. Vervolgens wordt 1 in de motor-flag geschreven om het uitschakelen van de motor voor te bereiden en het opslaan via de gebruikssysteem-routine in $FC93 wordt beëindigd.

Laadroutine $C1BE

Eerst wordt de synchronisatie door het sub-programma SSYNCH gezocht. Is de laatste byte van de gevonden synchronisatie die in $AB werd geschreven nul, dan gaat het om synchronisatie 2 en wordt er weer naar de start van deze routine teruggesprongen. Is $AB niet gelijk aan nul, dan wordt synchronisatie 1 gevonden en is de waarde in $AB het secundair adres. Is de file met het secundair adres eenmaal opgeslagen, of is het bij het laadcommando overgedragen secundair adres gelijk aan 1, dan wordt in ABSFLG bit 1 gezet en wordt het startadres uit de cassettebuffer in de laadstart-vector geschreven. Daarna wordt vanaf RELLOD "found name" gegeven.

Het sub-programma ADTOETS test aan de hand van bit nul van ABSFLG of er in plaats van het in de cassettebuffer aanwezige startadres, een ander, door het L-commando voortgebracht, adres als laadstart-vector moet worden benut. Is bit nul van ABSFLG gezet, dan wordt het met L voortgebrachte adres dat in $AC/$AD staat, in de startvector geschreven. Overeenkomstig bit 2 van ABSFLG wacht de computer daarna tot u een toets indrukt. Na het testen of de STOP-toets werd ingedrukt, wordt de filenaam van de gevonden file met die van de gevraagde file vergeleken, als u een naam opgaf. Stemmen de namen overeen, dan kan het laadproces beginnen en wordt de statusbyte gewist. De programma-lengte wordt gevormd door het verschil van start- en eindadres in het aanlooprecord. Door de programma-lengte met de startvector te vermeerderen, wordt het eindadres berekend en in de eindvector geschreven. Het sub-programma PLOAD laadt vervolgens het programma. Daarna worden de toetsbytes vergeleken en zo nodig wordt de status gezet. Als in ABSFLG geen bit is gezet, worden via de gebruikssysteem-routine STOEND de BASIC-vectoren op het programma-einde gezet.

De routines MERGE, LOAD en VERIFY lopen alle via de LOAD-routine. Bij MERGE wordt de startvector overeenkomstig de BASIC-eindvector gezet. Daarna wordt relatief geladen. Via het X-register wordt bij VERIFY de VERIFY-flag op 1 en bij LOAD op 0 gezet. Voor de LOAD-routine worden via GETPARA nog de resterende parameters ingelezen en in ABSFLG van de desbetreffende waarde voorzien.

Data opslaan en data lezen

De data worden in 3 blokken opgeslagen:

1. blok : variabelenbereik tussen de wijzers 45, 46/47, 48
2. blok : veldenbereik tussen de wijzers 47, 48/49, 50
3. blok : stringsbereik tussen de wijzers 51, 52/55, 56

De lengte van de afzonderlijke blokken wordt in het aanlooprecord door de start- en eindadressen gegeven.

DATASAVE $C311

Voordat de data kunnen worden opgeslagen, moeten ze op een zo klein mogelijke plaats bij elkaar worden gezet. Daarom wordt aan het begin de garbage-collection opgeroepen. Met behulp van

GETPARA wordt de filenaam gezet. Omdat een bepaald geheugenbereik moet worden opgeslagen, wordt daarna als secundair adres 1 gegeven. Door ULOOP wordt de wijzer op het begin of eindadres van het variabelenbereik in de start- en eindvectoren overgebracht. Tenslotte wordt dit bereik door het sub-programma ABSOLUT absoluut opgeslagen. De velden en strings worden daarna analoog aan de variabelen zonder filenaam via ABSOLUT opgeslagen.

DATALOAD $C361

Eerst wordt via GETPARA de filenaam gevormd. Door het zetten van bit 0 in ABSFLG wordt bereikt dat het variabelenbereik op een bepaald adres wordt geladen en de adressen in het aanlooprecord worden genegeerd.

Als laadadres wordt het startadres van de variabelen in de startvector overgedragen. Via LOADR wordt het variabelenbereik ingelezen. Daarna wordt bovendien bit twee in ABSFLG gezet, zodat de computer na het vinden van de eerstvolgende synchronisatie 1 niet hoeft te wachten op het indrukken van een toets door de gebruiker. Verder wordt geen filenaam opgegeven. De inhoud van $AE, $AF zet de wijzer op het eind van het beschreven bereik. Deze wijzer wordt tegelijk als startvector voor de velden in de laad-startvector overgenomen, zodat bij een nieuwe oproep van LOADR de velden worden ingelezen. Daarna wordt $AE, $AF in de wijzer op het eind van de velden geschreven.

Omdat de stringtabel vanaf het BASIC-RAM-einde naar beneden toeneemt, wordt het einde ervan niet door de lengte van het BASIC-programma beïnvloed. Daarom wordt de inhoud van dit geheugenbereik steeds op die plaats teruggeladen vanwaar het werd weggeschreven. Nu kan bit nul in ABSFLG worden gewist. Vervolgens wordt de tabel door de routine LOADR geladen. De wijzer op het eind van de stringtabel wordt dan uit de cassettebuffer gelezen en in 51, 52 opgeslagen.

Tenslotte wordt de status getest en zonodig wordt een foutmelding gegeven. Is er geen fout, dan wordt de stringdescriptor teruggezet en wordt naar het hoofdprogramma teruggesprongen.

13 DATAVERWERKING MET FAST-TAPE

In dit hoofdstuk wordt getoond dat ook met een cassetterecorder en geschikte software een efficiënte dataverwerking mogelijk is. Het in dit hoofdstuk afgedrukte data-verwerkingsprogramma heeft de volgende specificaties:

- Het opslaan en laden van data met fast-tape. Afhankelijk van de grootte van het databestand duurt het opslag- en laadproces tussen de 20 seconden en de 1,5 minuut.

- U kunt zelf het aantal datareeksen kiezen. Het maximum is afhankelijk van de vrije geheugenruimte van de computer.

- Een datareeks kan in een willekeurig aantal velden zijn onderverdeeld.

- Een veld mag maximaal 80 tekens bevatten.

- De aanduiding van de afzonderlijke velden is vrij en kan ook nadat deze heeft plaatsgevonden, worden veranderd.

- Er kan in een of meer velden tegelijk worden gezocht. U kunt hierbij kiezen of aan 1 dan wel aan alle zoekcriteria moet worden voldaan.

- De zoeksnelheid is in alle velden gelijk.

- Er kan naar grote en kleine overeenkomsten en verschillen worden gezocht. Indien u een BASIC-uitbreiding met INSTRING-instructie gebruikt, kunt u ook naar deelbereiken binnen de velden zoeken.

- Als er in meer velden wordt gezocht, kan er voor elk veld een andere zoekfunctie worden bepaald.

- Er kan steeds een HELP-PAGINA worden aangelegd die de afzonderlijke commando's verklaart.

- U kunt met de gevonden data listings aanleggen die u geformateerd kunt uitgeven.

- U kunt vrij bepalen welke velden moeten worden uitgegeven.

- De afzonderlijke velden kunnen met of zonder veldaanduiding worden uitgegeven.

- Listings kunnen volgens een willekeurig veld worden gesorteerd.

Dit programma is dusdanig van algemene aanpassingen voòrzien dat het zonder veranderingen zowel op de VIC 20 als op de Commodore 64 loopt. Deze algemene aanpassingen hebben een bepaalde geheugenplaats nodig en kunnen worden veranderd of gewist zodat er weer meer geheugenruimte beschikbaar komt. Aan het eind van dit hoofdstuk vindt u een gedetailleerde programmabeschrijving waarmee u het programma aan uw eigen wensen kunt aanpassen.

Nu volgt de bediening van het programma. Met dit programma kunt een databestand op willekeurige wijze inrichten en verwerken. U kunt een literatuur- of receptencatalogus aanleggen, uw platenverzameling registreren of een adressenbestand van uw vrienden inrichten. Om u de functie en bediening van dit programma duidelijk te maken, kunt u het beste aan de hand van deze handleiding voor de bediening een oefen-databestand aanleggen. Aangezien een receptencatalogus of iets dergelijks hiervoor te omvangrijk is, kunt u het beste een adressenbestand maken. Met een adressenbestand kunt u met weinig schrijfwerk een databestand van meer datarecords schrijven. Na de start met RUN verschijnt het volgende menu op uw beeldscherm.



Omdat u nog geen databestand heeft aangelegd, kiest u de eerste keer punt 8 van het menu.

Bestand inrichten

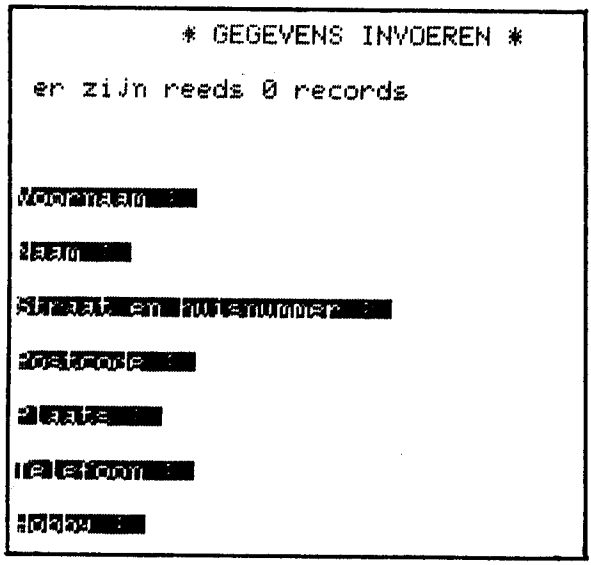
In dit menu-onderdeel moet u de basiskenmerken van uw bestand invoeren. U wordt verzocht het aantal velden en het maximale aantal datareeksen in te voeren. Deze waarden kunnen niet meer worden veranderd. Daarna wordt u gevraagd de veldaanduidingen voor de afzonderlijke velden te geven. Voor ons oefen-adressenbestand voert u als maximaal aantal datareeksen 10 of meer in en als aantal velden 7. De afzonderlijke velden kenmerkt u het beste als volgt:

1. Voornaam
2. Naam
3. Straat en huisnummer
4. Postcode
5. Woonplaats
6. Telefoon
7. Hobby

Als u deze gegevens heeft ingevoerd, keert u weer naar het menu terug. Hierna kunt u dit menu-onderdeel niet meer kiezen. Omdat u het programma nog niet echt heeft getest, er kunnen bij het intikken altijd fouten binnensluipen, slaat u het databestand uit voorzorg eerst zonder data op. Kiest u daarvoor onderdeel 7 "bestand opslaan". Er wordt u naar de filenaam gevraagd. U voert "adressen" in en drukt op RETURN. U wordt gevraagd een datacassette in uw recorder te stoppen. Let u erop dat de lees/schrijf-kop van de recorder niet helemaal aan het begin van de cassetteband staat, maar ongeveer vijf seconden looptijd verderop, zodat het databestand zeker wordt opgeslagen. Druk op RETURN en voldoe aan de uitnodiging "PRESSRECORD & PLAY ON TAPE". Nadat alle variabelen zijn opgeslagen,keert de computer naar het menu terug en kunnen we enige datareeksen invoeren.

Gegevens invoeren

Nadat u dit menu-onderdeel heeft gekozen, verschijnt het volgende beeld:



De cursor staat onder de veldaanduiding "Voornaam". U kunt nu in elk veld ten hoogste 80 karakters schrijven. Zolang u binnen een veld blijft, staan u alle mogelijkheden van de beeldschermeditor ter beschikking. U vult de velden als volgt in:



Nadat u de invoer in het laatste veld met RETURN heeft afgesloten, verschijnt "RETURN" in de laatste regel van het beeldscherm. U heeft vervolgens als invoermogelijkheden:

RETURN	De voorafgaande datareeks wordt in het databestand opgenomen en u kunt de volgende datareeks invoeren.
Q	De datareeks wordt niet opgenomen en het programma keert naar het menu terug.
K	De cursor verschijnt weer in het eerste veld en u kunt de invoer corrigeren.
andere toets	De datareeks wordt overgenomen en het programma keert naar het menu terug.

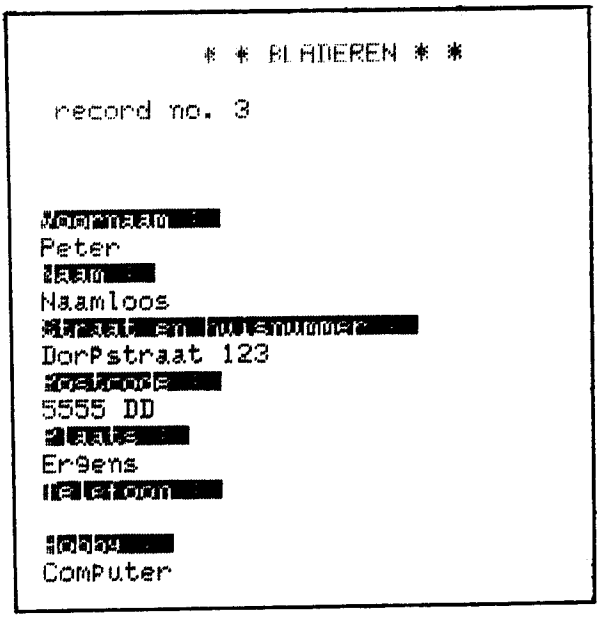
Bij een foutloze invoer drukt u op RETURN. In de kop wordt aangegeven dat er een datareeks bestaat en de cursor verschijnt weer in het eerste invoerveld. In de velden staat nog de laatste invoer die u desgewenst kunt overschrijven. Indien enige velden niet veranderen, kunt u door het indrukken van RETURN deze datareeks overnemen. U voert op deze wijze vier adressen in. Indien u over een uitbreiding met "INSTRING"-commando beschikt, moet u ook de volgende datareeks invoeren:



U keert nu door het indrukken van een willekeurige toets naar het menu terug.

Bladeren

U kiest onderdeel 2. In deze modus kunt u de data in de volgorde waarin u ze heeft ingevoerd, bekijken en eventueel veranderen. Door een druk op de RETURN-toets kunt u steeds een datareeks verder bladeren. Met de "-"-toets kunt u terugbladeren. Als u op de "K"-toets drukt, verschijnt de cursor in het eerste veld en kunt u de aangegeven datareeks veranderen. Met de "Q"-toets komt u weer terug in het menu.



Bestand onderhouden

Door menu-onderdeel 1 komt u in dit gedeelte van het programma. U kunt hier de data selecteren, wissen, veranderen en lijsten maken. Eerst zoekt u in het veld "Naam" alle datareeksen waarin de naam met "M" begint. U voert dus op de vraag naar welke velden moet worden gezocht, een 2 in. Daarop verschijnt de betreffende veldaanduiding revers op het beeldscherm. In de voorlaatste beeldschermregel verschijnt:



Dit zijn de keuzemogelijkheden die u heeft.

1. Deelstring (alleen met INSTRING-commando). U zoekt naar een karakterreeks die op een willekeurige plaats staat binnen het veld.

2. Gelijk U zoekt naar een significante karakter-reeks die precies gelijk is aan uw gezochte string. Er worden steeds net zo veel karakters vergeleken, als u bij de gezochte string heeft gevraagd. Voert u bijvoorbeeld "Ma" in, dan vindt het

programma man, mast, mama enzovoort. Dit geldt op dezelfde manier ook voor de volgende keuze-mogelijkheden.

3. Ongelijk Hier worden de strings gevonden die niet met de gezochte strings overeenkomen.

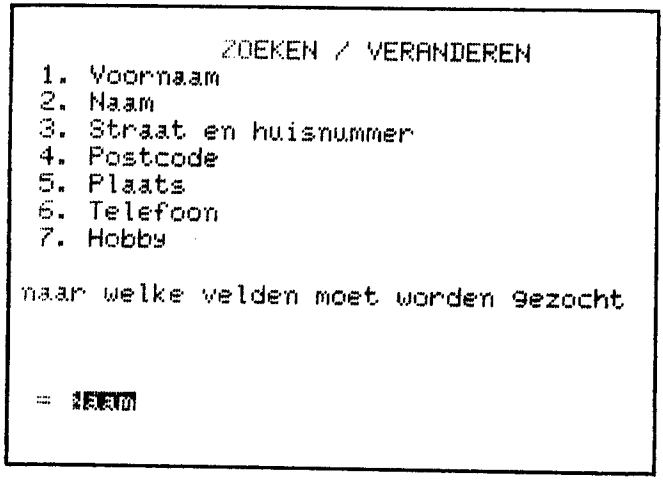
4. Groter of gelijk

Hier vindt het programma alle strings die groter of gelijk zijn. Voert u bijvoorbeeld M in bij de naam, dan worden alle namen gevonden, die met M of een in het alfabet daarop volgende letter (N, O, P, Q,...) beginnen.

5. Kleiner of gelijk

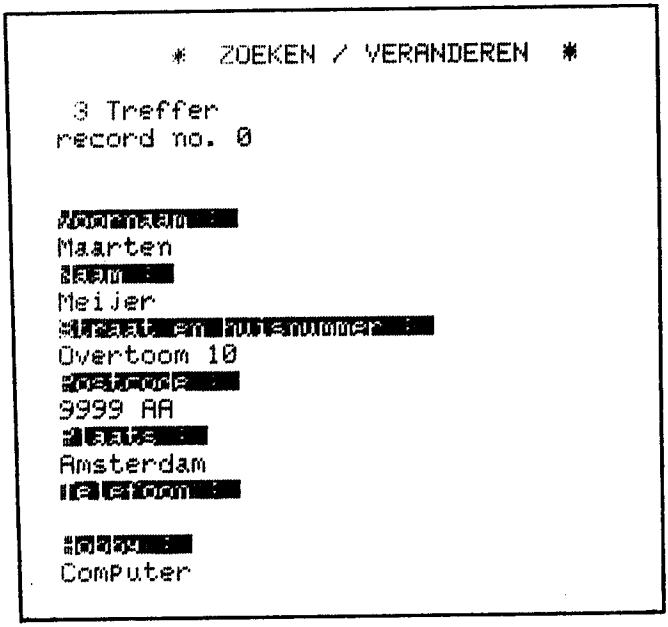
Voert u als gezochte string M in, dan worden alle namen die met A-M beginnen gevonden.

Drukt u op "2", dan krijgt u het volgende beeld:



Als u binnen andere velden wilt zoeken, hoeft u alleen maar het betreffende getal in te voeren. We zullen echter eerst naar de naam zoeken. Door een druk op de RETURN-toets sluit u de invoer af. Vervolgens wordt er gevraagd of er een datareeks met een of met alle overeenkomsten moet worden gevonden. Omdat u maar een

keuzecriterium heeft ingetoetst, drukt u op RETURN. De vastgestelde veldaanduiding en de cursor verschijnen. Omdat u alle namen wilt zoeken die met "M" beginnen, voert u M in en RETURN. Daarop verschijnt in de laatste beeldschermregel weer "RETURN". Als u nu op "C" drukt, kunt u de invoer corrigeren. Drukt u op RETURN, dan begint de computer met het zoeken en toont dan het trefgetal, het nummer van de aangegeven datareeks en de datareeks zelf.



U kunt net als onder "Bladeren" in de gevonden datareeksen bladeren. Verder kunt u ze door "C" corrigeren en door "W" wissen. Met "Q" komt u weer terug in het menu. Maakt u zich met de menu-onderdelen 1 en 2 vertrouwd, zodat u ziet hoe en hoe snel de gevraagde datareeksen kunnen worden gezocht en gecorrigeerd. Zoekt u daarna eens naar een criterium dat op meer dan een datareeks van toepassing is. Als u zo een lijst van datareeksen heeft gemaakt, kunt u ze via menu-onderdeel 4 naar een willekeurig veld, bijvoorbeeld "Naam", sorteren. Via het menu-onderdeel 3 kunt u de lijst op het beeldscherm laten verschijnen of met een printer afdrukken.

Lijst uitvoeren

In dit menu-onderdeel wordt u gevraagd welke velden moeten worden afgedrukt. Door het invoeren van de veldindexen kunt u de gevraagde velden kiezen en met RETURN de invoer beëindigen. Als u in plaats daarvan "C" invoert, worden de aangegeven velden gewist en kunt u de invoer herhalen. Nadat u de gewenste velden heeft gekozen, drukt u op RETURN. Daarna wordt u naar een titel gevraagd. Deze wordt dan bovenaan uw lijst afgedrukt. Bij de printeruitvoer wordt de titel in hoofdletters gedrukt. Daarna kunt u kiezen of de veldaanduidingen mee moeten worden uitgevoerd. Tenslotte wordt gevraagd of de uitvoer op het beeldscherm of via een printer moet gebeuren. Bij de printeruitdraai kunt u het volgende beeld krijgen:

```
ComPuter-Freaks

Peter
Naamloos
Dorpstraat 123
5555 DD
Ergens

ComPuter


Maarten
Meijer
Overtoom 10
9999 AA
Amsterdam

ComPuter


Marlies
Berendsen
Keizerstraat 56
1111 ZZ
Groningen

ComPuter/Koken
```

Het programma is zo gemaakt, dat elk veld in een eigen regel wordt geschreven. Door programmaveranderingen in de regels 2870-2930 kunt u het uitvoerformat aan uw wensen aanpassen. Als u de lijsten op het beeldscherm uitvoert, kunt u de uitvoer door een druk op de "SPACE"-toets stoppen en weer starten. Ik hoop dat de bediening van dit programma na enig oefenen geen problemen meer geeft.

```
10 goto 1060
90 rem zoek-routine
100 if(f>fe)or(n>=as) then return
110 a=sf(f)
120 on a gosub 150,190,250,280,220
130 f=f+1: goto 100
140 rem bereik
150 t=instr(u$(f),s$(f))
160 if t<>0 then n=n+1
170 return
180 rem helemaal '='
190 if s$(f)=left$(u$(f),len(s$(f)))then
 n=n+1
200 return
210 rem kleiner/gelijk '<='
220 if s$(f)>=left$(u$(f),len(s$(f)))the
n n=n+1
230 return
240 rem ongelijk '<>'
250 if s$(f)<>left$(u$(f),len(s$(f)))the
n n=n+1
260 return
270 rem groter/gelijk '=>'
280 if s$(f)<=left$(u$(f),len(s$(f)))the
n n=n+1
290 return
1000 rem wachtlus
1010 poke198,0:wait198,1:geta$:a=asc(a$)
:rem  wachtlus
1020 if a=133 then gosub 3750
1030 return
1050 rem  initialiseren
1060 dw$="[illegible]"
1070 sys65517:sp=peek(781):ze=peek(782):
rem  beeldschermgegevens halen
1080 c1=53281:c2=1:if sp=22 then c1=3687
9:c2=8
1090 dim z$(4):z$(0)=" T":z$(1)=" =":z$(
```

```
2)="<>":z$(3)=">=":z$(4)="<="
1100 rem   hoofdPro9ramma
1110 Print chr$(14): Poke c1,6*c2
1120 9osub 3560:Print:Print:ta=int(sP/5)
-2
1130 Printtab(ta)"1 Bestand onderhoud
en"
1140 Printtab(ta)"2 bladeren "
1150 Printtab(ta)"3 Lijst uitvoeren "
1160 Printtab(ta)"4 Lijst sorteren"
1170 Printtab(ta)"5 Ge9evens invoeren"
1180 Printtab(ta)"6 Bestand laden"
1190 Printtab(ta)"7 Bestand oPslaan"
1200 Printtab(ta)"8 Bestand be9innen"
1210 Printtab(ta)"9 Veldind. verandere
n"
1220 Printtab(ta)"E     EINDE"
1230 9osub 1010:a=val(a$): if a$="e"then
 1270
1250 on a 9osub 2130,2000,2700,2990,1860
,1570,1460,1700,1780
1260 9oto 1110
1265 rem  Pro9ramma beeindi9en
1270 if(fl and 2)=2 or(fl and 4)=0 then
end
1280 Print"                        ":Printt
ab(sP/2-8)"OP9elet !! ":9osub 1310
1290 if fl<>255 then 1110
1300 end
1305 rem     fout
1310 Poke c1,5*c2
1320 Print"":Print"       "
1330 Print:Printtab(sP/2-3)"bestand"
1340 Print:Printtab(sP/2-6)"is niet"
1350 Print:Printtab(sP/2-7)"oP9esla9en"
1360 for i=0 to 15:9et a$:if a$<>"" then
 i=15: next: 9oto 1440
1370 next
1380 Print"":Print"       "
1390 Print:Printtab(sP/2-3)"bestand"
1400 Print:Printtab(sP/2-6)"is niet"
1410 Print:Printtab(sP/2-7)"oP9esla9en"
1420 for i=0 to 15:9et a$:if a$="" then
i=15:next: 9oto 1310
1430 next
1440 if a$="J" then fl=255
1450 return
1460 rem   oPslaan
1470 if(fl and 1)=0 then return
```

```
1480 poke c1,7*c2:print"■": gosub 3590
1490 print:print:print"  "f$"■":input"Fi
lenaam";f$
1500 print:print"cassette inleggen"
1510 print:print:print"   << return >>"
1520 gosub1010
1530 if a$="k"then print"■":goto 1490
1540 if a$="q"then return
1550 if a<>13 then 1520
1560 ←dsf$:fl=flor2:return
1570 rem  laden
1580 if(fl and 6)=4 then gosub 1680:if f
l<>255 then return
1590 poke c1,7*c2:print"■":gosub 3600
1600 print:print:print"  "f$"■":input"fi
lenaam";f$
1610 print:print"cassette inleggen "
1620 print:print:print"   << return >>"
1630 gosub 1010
1640 ifa$="k"thenprint"■":goto1600
1650 ifa$="q"thenreturn
1660 if a<>13 then 1630
1670 ←dlf$:fl=fl and 251: return
1680 print"■■■■■■■■■■■■■■■■■■■■■■■":printt
ab(sp/2-8)"■■opgelet !!"
1690 gosub 1310:return
1700 rem  bestand beginnen
1710 poke c1,8*c2: if fl then return
1720 gosub3550:rem  menu op beeldscherm
1730 print:print:input"Veldaantal";va
1740 print:input"■Max aantal Records ";r
a
1750 print:print"<< return >> ":gosub 10
10:if a<>13 then print"■":goto 1730
1760 dim m$(va),e$(va,ra),sf(va),s$(va),
u$(va),gf%(ra)
1770 fl=1:rem "  flag zetten
1775 rem  veldindeling invoeren
1780 if(fl and 1)=0 then return
1790 gosub 3550:rem  menu op beeldscherm
1800 print:print"Geef de ";:if sp<40 the
n print:print
1810 print"veldindeling in"
1820 print:for i=0 to va-1:print:print"■
veld "i+1"■"
1830 print"  "m$(i)"■":inputm$(i):next
1840 printleft$(dw$,ze)"<< return >>";:g
osub 1010:if a$="k"thenprint"■":goto1800
1850 fl=fl or 4:return
```

```
1860 rem   9e9evens invoeren
1865 if fl and 1=0 then return
1870 Poke c1,3*c2: Print"■"
1880 close 1:oPen 1,0
1890 9osub 3580:Print:Print"Er zijn reed
s"rt;:if sP<40 then Print
1900 Print"Records "
1910 r1=rt:r2=rt:if(rt>0)and(a$<>"k")the
n r1=rt-1
1920 9osub 3650:9osub 3670:rem  velden o
P beeldscherm bren9en
1930 Printleft$(dw$,ze)"■■   << return >>
 ■";:9osub1010
1940 if a$="9" then return
1950 if a$="k" then 1890
1960 ifa<>13thenfl=flor12:rt=rt+1:return
1970 rt=rt+1:ifrt<rathen1890
1980 9osub3580:Print:Print:Print"Max. aa
ntal ":if sP<40 then Print
1990 Print:Print"Records bereikt":9osub
1010:return
2000 rem   bladeren
2005 if(fl and 8)=0 then return
2010 Poke c1,6*c2:Print"■"
2020 r1=0
2025 if r1<0 then r1=r1+rt
2030 9osub 3610:Print:Print"Record nr."r
1
2040 9osub 3650
2050 Printleft$(dw$,ze)"■■   << return >>
 ■";:9osub1010
2060 if a$="-" then r1=r1-1: 9oto 2025
2070 if a$="9" then return
2080 if a$="a" then 9osub 2560:9oto 2060
2090 if a<>13then 2050
2100 if r1<rt-1 then r1=r1+1: 9oto 2030
2110 9osub3610:Print"■■■■■■■■■■ Niet mee
r 9e9evens "
2120 r1=0:9oto2050
2130 rem   bestand verzor9en
2135 if(fl and 8)=0 then return
2140 Poke c1,4*c2
2150 for i=0 to va-1: sf(i)=0: next: as=
0
2160 9osub 3620:for i=0 to va-1:Print i+
1"■■. "m$(i):next
2170 Print:Print"Naar welke velden ";:if
 sP<40 then Print:Print
```

```
2180 Print"moet worden 9ezocht":Print:Pr
int
2190 9osub 1010:if a=13 then 2262
2200 if a$="9" then return
2210 if a$="k" then 2150
2220 i=val(a$):if i<1 or i>va then 2190
2230 if sf(i-1) then 2190
2240 d6=Peek(214):rem cursorre9el redden
2250 sf(i-1)=1:Printtab(3)"■"m$(i-1):9os
ub 3710
2260 Printleft$(dw$,d6+1)z$(sf(i-1)-1):a
s=as+1:9oto 2190
2262 9osub 3620:Print:Print"Vinden bij"
2264 Print:Printtab(4)"1 = een"
2265 Print:Printtab(4)"0 = alle"
2266 Print:inPut"overeenkomst(en)";ov
2267 if ov then as=ov
2270 9osub 3620: close 1: oPen 1,0
2280 Print:Print"Geef de zoekstrin9 ";:i
f sP<40 then Print:Print
2290 Print"in":Print:Print
2300 for f=0 to va-1:if sf(f)=0 then 233
0
2310 Print"■"m$(f)" : "
2320 Print s$(f)"■":inPut#1,s$(f):Print
2330 next
2340 Printleft$(dw$,ze)"<< return >>";:9
osub 1010:if a$="k"thenPrint"■":9oto2280
2350 if a$="9" then return
2360 if a<>13 9oto 2340
2370 rem   zoeken
2372 for f=0 to va-1:if sf(f) then fb=f:
f=va
2374 next:rem  laa9ste en hoo9ste
2376 for f=va-1 to 0 steP-1:if sf(f)then
fe=f:f=0
2378 next:rem  zoekveld bePalen
2380 z=0:for r =0 to rt-1
2390 for f=fb to fe-1:if sf(f) then u$(f
)=e$(f,r)
2400 next:
2410 n=0:f=fb:9osub 100:if n>=as then 9f
%(z)=r:z=z+1
2420 next:if z=0 then 2130
2430 rem   records oP beeldscherm bren9e
n
2440 i=0
2450 9osub 3620:Print:Print" Gevonden":P
rint"Record nr."9f%(i)
```

```
2460 r1=9f%(i):9osub3650
2470 Printleft$(dw$,ze)"■■   << return >>
 ■";:9osub1010
2480 if a$="-" and i>0 theni=i-1: 9oto 2
450
2490 if a$="9" then return
2500 if a$="v" then 9osub 2560:9oto2480
2510 if a$="w" then2640
2520 if a<>13 then 2470
2530 if i<z-1 then i=i+1:9oto 2450
2540 9osub 3620:Print"■■■■■■■■■■■■ Niet
meer 9e9evens "
2550 Printleft$(dw$,ze)"■■   << return >>
 ■";:9osub 1010:9oto 2450
2560 rem   veranderen
2570 close 1:oPen 1,0:r2=r1: 9osub 3670
2580 Printleft$(dw$,ze)"■■   << return >>
 ■";:9osub1010
2600 if a$="k" then 2570
2610 fl=fl or 4:fl=fl and 253:return
2620 return
2630 rem   wissen
2640 for i1=9f%(i)+1 to rt-1:rem " Ge9ev
ens
2650 for j=0 to va-1:rem  bufferen
2660 e$(j,i1-1)=e$(j,i1)
2670 nextj,i1
2680 z=z-1:rt=rt-1:fl=flor4:9oto2450
2690 rem "      lijst uitvoeren
2695 if z=0 then return
2700 Poke c1,7*c2:Print"■"
2710 for i=0 to va-1:sf(i)=0:next
2720 9osub 3630
2725 for i=0 to va-1:Print i+1"■■. "m$(i)
:next
2730 Print:Print"Welke velden moeten ";:
ifsP<40thenPrint:Print
2740 Print"uit9ePrint worden ":Print:Pri
nt
2750 9osub 1010:if a=13 then 2810
2760 if a$="9" then return
2770 if a$="k" then 2710
2780 i=val(a$):if i<1 or i>va then 2750
2790 if sf(i-1) then 2750
2800 sf(i-1)=1:Print"■"m$(i-1):Print:9ot
o 2750
2810 Print:inPut"Titel";ub$
2820 Print:Print"Veldindelin9en":Print:i
nPut"uitPrinten [1/0]";fm$
```

```
2830 fm=val(fm$)
2840 print:input"Printer=4/beelds.=0";dv
2850 if dv then open4,dv,7:cmd4
2860 print chr$(14)ub$chr$(15):print:pri
nt
2870 for r=0 to z-1
2880 for f=0 to va-1:if sf(f)=0 then 291
0
2890 if fm then print:print"■"m$(f)" : ■
"
2900 print e$(f,9f%(r))
2910 next:if dv then 2930
2920 get a$:if a$<>""then poke 198,0:wai
t198,1:get a$
2930 print:print:next
2940 if dv then print#4:close4:return
2950 print:print"     << return >>"
2960 gosub 1010:if a<>13 then 2960
2970 return
2980 rem  sorteren
2990 if z=0 then return
3000 gosub 3640
3010 for i=0 to va-1:printi+1"■. "m$(i):
next
3020 print"■Naar welk veld ";:if sp<40 t
hen print:print
3030 input"moet worden gesorteerd ";sf:s
f=sf-1
3040 f1=0:fh=0:fh=peek(49)+peek(50)*256:
rem bovengrens redden
3050 f1=peek(47)+peek(48)*256:rem onderg
rens redden
3060 dim n$(z),z(z+1)
3070 q=1:for i=0 to z-1:rem  sorteerveld
3080 n$(q)=e$(sf,9f%(i)):rem  opslaan
3090 z(q)=q:q=q+1:next
3100 gosub3240:rem   sorteerroutine
3110 for x=0 to va-1
3130 for y=0 to z-1
3140 n$(y+1)=e$(x,9f%(y))
3160 next
3170 for y=0 to z-1
3180 e$(x,9f%(y))=n$(z(y+1))
3200 next y,x
3210 for i=0 to z-1:n$(i)="":next:rem st
rings van n$() wissen
3220 gosub3520:rem herstellen van oude p
ointer
3230 return
```

```
3235 rem sorteertijd
3240 ti$="000000":Print"■"
3250 l=int(z/2)+1
3260 r=z:Print"■"r"■■   "
3270 if l>1 then 3340
3280 if r<=1 then 3330
3290 h2$=n$(l):h1=z(l)
3300 n$(l)=n$(r):z(l)=z(r)
3310 n$(r)=h2$:z(r)=h1
3320 r=r-1
3330 goto 3350
3340 l=l-1
3350 j=l
3360 i=2*j
3370 h2$=n$(j):h1=z(j)
3380 if i>r then 3480
3390 if i>=r then 3420
3400 if n$(i)>=n$(i+1) then 3420
3410 i=i+1:Print"■"i"■■   "
3420 if i>r then 3480
3430 if h2$>=n$(i) then 3480
3440 n$(j)=n$(i):z(j)=z(i)
3450 j=i
3460 i=2*j
3470 goto 3390
3480 n$(j)=h2$:z(j)=h1
3490 if r<>1 then 3270
3500 Print"■■■Sorteertijd";ti$
3510 return
3515 rem Pointer herstellen
3520 j=Peek(47)+Peek(48)*256+(fh-f1)
3530 Poke 49,j and 255:Poke 50,int(j/256
)
3540 return
3545 rem       menu
3550 Print"■";:Printtab(sp/2-10)"* BESTA
ND BEGINNEN *":return
3560 Print"■";:Printtab(sp/2-10)"■* * *
M E N U * * *":return
3570 Print"■";:Printtab(sp/2-9)"* BESTAN
D ONDERHOUDEN *":return
3580 Print"■■";:Printtab(sp/2-10)"* GEGE
VENS INVOEREN *":return
3590 Print"■";:Printtab(sp/2-8)"* GEGEVE
NS OPSLAAN *":return
3600 Print"■";:Printtab(sp/2-6)"* GEGEVE
NS LADEN *":return
3610 Print"■";:Printtab(sp/2-8)"* * BLAD
EREN * *":return
```

```
3620 Print"■";:Printtab(sP/2-8)"* ZOEKEN
 / VERANDEREN *":return
3630 Print"■";:Printtab(sP/2-7)"* LIJST
UITVOEREN *":return
3640 Print"■";:Printtab(sP/2-7)"* LIJST
SORTEREN *":return
3645 rem velden uitvoeren
3650 Print"■■■■■■":forf=0 to va-1:Print"
■"m$(f)" : "
3660 Print e$(f,r1):next:return
3670 Print"■■■■■■":for f=0 to va-1:Print
"■"m$(f)" : "
3680 Print e$(f,r1)"■":a=len(e$(f,r1))
3690 if a>sP then for x=0 to a/sP:Print"
■";:next
3700 inPut#1,e$(f,r1):Print:next:return
3705 rem zoekvorm kiezen
3710 Printleft$(dw$,ze-1)"■1:T/2:=/3:<>/
4:>/5:<■";
3720 9osub1010:j=val(a$):if J<1 or J>5 t
hen 3720
3730 sf(i-1)=j
3740 Printleft$(dw$,ze-1)"
     ";:return
3750 rem hulPPa9ina
3760 Print"■■ * * * * * * H u l P P a 9 i
n a * * * * * *"
3770 Print:Print"na << return >> verschi
jnen er een "
3780 Print:Print"aantal oPties :":Print:
Print
3790 Print"return  = door9aan":Print
3800 Print"k       = laatste invoer herh
alen ":Print
3810 Print"9       = Pro9ramma onderdeel
 stoPPen ":Print
3820 Print"-       = een zin teru9 ":Pri
nt
3830 Print"a       = een zin vooruit ":P
rint
3840 Print"**************************"
3850 9et a$:if a$="" then 3850
3860 return
5000 "**********************************
5010 "*  Bestandsbeheer met FasttaPe   *
5020 "*       (c) Dirk Paulissen       *
5030 "*                                *
5040 "* Gebruikte variabelen :         *
5050 "* va    = veldaantal             *
```

```
5060 "* ra     = recordaantal(max)       *
5070 "* rt     =   - '' - (actueel)      *
5080 "* f      = actueel veld            *
5090 "* r      = actueel record          *
5100 "* sp     = kolom nummer            *
5110 "* ze     = regel nummer            *
5120 "* a$     = vraagstring             *
5130 "* fl     = flag                    *
5140 "* e$(f,r) = gegevensstring         *
5150 "* sf(f)   = zoekveldflag           *
5160 "*           1 : deelstring zoeken  *
5170 "*           2 : helemaal '='       *
5180 "*           3 : ongelijk '<>'      *
5190 "*           4 : groter/gelijk '>=' *
5200 "*           5 : kleiner/gelijk'<=' *
5220 "* s$(f)   = zoekstring             *
5230 "* u$(f)   = onderzoekstring        *
5240 "* gf%()   = nummer v/d gevonden    *
5250 "*           records                *
5300 "*************************************
```

13.1 Programmabeschijving

De programma-opbouw is als volgt:

90-1030	snel subprogramma
1050-1090	initialisering
1100-1260	hoofdprogramma (menu)
1270-3540	subprogramma dat door het menu wordt opgeroepen
3550-3640	titels
3650-3850	subprogramma dat door andere subprogramma's wordt opgeroepen
5000-einde	copyright-aanduiding en opsomming van de gebruikte variabelen

Werkwijze

Het aanpassen aan de verschillende computers gebeurt in regel 1070 door het oproepen van de bedrijfssysteemroutine SCREEN (65517). Deze routine legt het aantal kolommen en regels vast in de geheugenplaatsen 781 en 782. Overeenkomstig deze waarden worden de geheugenplaatsen en de waarden voor de kleur POKE-commando's gezet (C1, C2). Het hoofdprogramma bestaat uit het menu en de verwijzing naar het gekozen sub-programma. Voor de databeveiliging wordt binnen dit sub-programma een flag FL gezet die steeds aan het begin wordt getest. De afzonderlijke bits van FL hebben, als ze zijn gezet, de volgende betekenis:

bit	waarde	betekenis
0	1	bestand ingericht
1	2	bestand opgeslagen
2	4	bestand veranderd
3	8	bestand bevat gegevensreeksen

Probeert u een databestand te laden of het programma te beëindigen voordat een veranderd databestand werd opgeslagen, dan wordt er naar de foutenroutine 1305 - 1450 overgegaan. De programma-onderdelen "opslaan" en "laden" spreken voor zich.

Bestand inrichten(1700-1850)

Nadat de titel door regel 1720 werd geschreven, wordt via INPUT het aantal velden (VA) en het maximale aantal records (MA) ingelezen. Door de vraag in 1750 kan een foutieve invoer worden gecorrigeerd. Volgens VA en MA worden in 1760 de velden gedimensioneerd en in 1770 bit 0 in FL gezet. In regel 1780 wordt bit 0 getest omdat de routine voor de invoer en verandering van de veldaanduidingen ook vanuit het menu kan worden opgeroepen. In de regels 1800 en 1810 wordt de uit te voeren string overeenkomstig het aantal kolommen van de computer in 1 of 2 regels uitgedrukt. Binnen een lus (regel 1820-1830) worden de afzonderlijke veldaanduidingen in M$(VA) ingelezen. Bestaande aanduidingen worden op het beeldscherm uitgevoerd en kunnen door RETURN worden overgenomen. Door de vraag in regel 1840 is weer een correctie mogelijk.

Gegevens invoeren (1860-1990)

In regel 1865 wordt aan de waarde van FL gezien of er al een bestand werd samengesteld. In 1880 wordt door OPEN1,0 het toetsenbordkanaal geopend. Hierdoor wordt bij een volgende INPUT#1 geen vraagteken uitgevoerd. Ter bescherming wordt voor elk OPEN-commando het desbetreffende databestand gesloten. Via de regels 1890-1900 wordt de titel en overeenkomstig het aantal kolommen een string uitgevoerd. In het sub-programma dat de afzonderlijke velden van een reeks aangeeft (3650-3660) of nieuwe velden inleest (3670-3700), wordt R1 of R2 als recordwijzer gebruikt. Daarom wordt in regel 1910 de wijzer op de in te voeren reeks RT in R1 en R2 overgedragen. Werd er al een reeks ingevoerd (RT groter dan 0), dan wordt R1 gedecrementeerd, zodat de velden van de laatst ingevoerde reeks worden aangegeven. Vanaf regel 1930 wordt gevraagd of er afgebroken (A$ = "Q"), gecorrigeerd (A$ = "K"), geeindigd (A <> 13) of doorgegaan moet worden. Is A=13, dan wordt in regel 1970 getest of het maximale aantal reeksen werd bereikt.

Bladeren (2000)

Met de lusvariabele R1 worden via het sub-programma "VELDEN UITVOEREN" de afzonderlijke datareeksen aangegeven. Daarna wordt een nieuwe keuze via de "wachtlus" (1010) gevraagd. Overeenkomstig de invoer wordt er doorgebladerd (2100), naar het menu teruggekeerd (2070) of in het sub-programma "VERANDEREN" (2080) overgegaan.

Bestand verzorgen (2130)

Na onderzoek van FL en een beeldschermPOKE worden de zoekflags (SF(I)) op 0 gezet (2150). Via de "wachtlus" wordt vanaf regel 2190 gevraagd vanaf welk veld moet worden gezocht. Wordt een geldig, nog niet gevraagd veldnummer ingevoerd (de test hiervoor staat in regel 2220-2230), dan wordt in regel 2240 de actuele cursorregel in D6 gered en in regel 2250 de veldaanduiding van het gewenste veld uitgedrukt. Via het sub-programma "ZOEKVORM KIEZEN" wordt vanaf regel 3710 de index van de verlangde zoekvorm in de desbetreffende zoekflag ingelezen. Met de geredde cursorregel (D6) wordt dan in regel 2260 het desbetreffende zoekvormkarakter voor de veldaanduiding gezet en in afwachting van andere zoekvelden teruggesprongen. In de regels 2262-2267 wordt het

gevraagde aantal van de overkomsten in OV geschreven. Via de regels 2280-2360 worden de zoekstrings in het veld (S$(F) ingelezen. Omdat de eigenlijke zoekroutine in elke datareeks de velden onderzoekt vanaf een startveld tot aan een eindveld, wordt in de regels 2372-2378 de laagste en hoogste veldindex bepaald en in FB en FE gered om zo de zoektijd te verbeteren. De hoofdzoeklus via de afzonderlijke datareeksen begint op regel 2380. In 2390 worden de te onderzoeken strings in het veld U$(F) overgedragen. In 2410 wordt de index voor gevonden overkomsten N op 0 en F op het startveld gezet voordat in de eigenlijke zoekroutine vanaf regel 100 wordt gesprongen. De zoekroutine ligt aan het begin van het programma omdat het beoogde adres wordt gezocht na een sprong (GOTO, GOSUB), achter de vragende regel of vanaf het programmabegin. Zou de routine aan het programma-einde liggen dan moest bij een sprong in een vorige regel steeds het hele programma na het beoogde regelnummer worden doorzocht, wat bij langere bestanden de zoektijd aanzienlijk zou verlengen. Aan het begin van de zoekroutine wordt getest of alle gevraagde velden al werden doorzocht, of dat het gevraagde aantal overeenkomsten werd gevonden. Is dit niet het geval, dan wordt via het veld SF(F) naar de gevraagde vergelijking gesprongen. Is de desbetreffende vergelijking logisch correct, dan wordt de overeenkomstenteller verhoogd. Nadat de veldindex F werd verhoogd, wordt er naar de start teruggesprongen. Na de sprong terug naar 2410 wordt, als het gevraagde overeenkomstengetal werd bereikt, het recordnummer in het veld GF%(Z) geschreven en het trefgetal Z geïncrementeerd. Na de beëindiging van het zoeken, worden de treffers analoog aan het sub-programma "BLADEREN" aangeduid (2430-2550).

Lijst uitvoeren (2690-2970)

Deze routine voert de door het veld GF%(I) bepaalde datareeksen op het beeldscherm of met de printer uit. De vraag naar de af te drukken velden is analoog aan de veldbepalingen in het sub-programma "ZOEKEN".

Sorteren (2980-3510)

Na de vraag of een lijst werd samengesteld en na de uitvoer van de titel op het beeldscherm (2990-3000), wordt het veld waarnaar gesorteerd moet worden, gevraagd en de desbetreffende index in SF geschreven. Voor de volgende sorteerroutine zijn enige hulpvelden

nodig waarvan de grootte van de omvang van de lijst afhangt. Omdat velden echter normaal gesproken slechts een keer mogen worden gedimensioneerd, hadden we bij de initialisering de hulpvelden N$(I) en Z(I) met het maximale aantal datareeksen moeten dimensioneren. Dat zou enerzijds veel geheugenruimte vragen en anderzijds de laad- en opslagtijd van en naar de band verlengen. Met een kunstgreep kunt u fouten selectief wissen en zo herdimensioneerbaar maken. U moet daarvoor voor de dimensionering de boven- en ondergrens van het bereik waarin de computer velden opslaat, redden. De daarna gedimensioneerde velden kunnen dan door het terugzetten van de wijzer op de oude lengte van de veldtabel worden gewist. Hierdoor worden in de regels 3040 en 3050 de wijzers op de start en het einde van de veldtabel in F1 en FH gered. In 3060 worden de hulpvelden overeenkomstig de lijstomvang gedimensioneerd. Door de lus in de regels 3070 tot en met 3090 wordt het desbetreffende recordveld van elke datareeks op de lijst in N$(I) overgedragen. Dit stringveld wordt later gesorteerd. Om daarna de betreffende andere velden van een datareeks aan het gesorteerde veld te kunnen toevoegen, wordt in Z(I) een "pointer" aangelegd die de positie van de desbetreffende datareeks op de lijst bevat. De sorteerroutine verwerkt velden met indexen vanaf 1. In het dataveld E$(F,R) zijn de datavelden echter vanaf index 0 opgeslagen. Daarom moeten de data en de pointer steeds in een veldvariabele met een index die 1 hoger is dan die in het dataveld, worden opgeslagen. Bij de opslagroutine is de verhoogde index drie keer nodig (index van N$, index van Z, pointer in Z). Om binnen de opslagloop niet drie keer "I+1" te moeten berekenen, heb ik een hulpvariabele Q gedefinieerd die binnen de lus slechts een keer moet worden geïncrementeerd. In regel 3100 wordt naar de sorteerroutine (3240-3510) verwezen. Deze routine sorteert het een-dimensionale veld N$(I) door verwisselen van de inhouden van N$(I). Bij elke verwisseling wordt ook de inhoud van de pointer verwisseld. Door een voorbeeld wordt dit duidelijk. Nemen we aan dat er 5 namen moeten worden gesorteerd die in deze volgorde in de velden staan:

```
n$(1) = Jan      z(1) = 1
n$(2) = Alfred   z(2) = 2
n$(3) = Peter    z(3) = 3
n$(4) = Dirk     z(4) = 4
```

Na het sorteren krijgen we:

```
n$(1) = Alfred   z(1) = 2
n$(2) = Dirk     z(2) = 4
n$(3) = Jan      z(3) = 1
n$(4) = Peter    z(4) = 3
```

De index van Z geeft nu de oplopende volgorde van de gesorteerde strings aan en de inhoud van elke Z geeft het recordnummer dat op de desbetreffende plaats moet staan. In de lusregel 3110-3200 worden de datavelden overeenkomstig de pointer verwisseld. De buitenste lus telt over de velden. In de eerste binnenste lus wordt het desbetreffende veld (bepaald door X) van alle datareeksen in de oude volgorde in het veld N$(I) overgedragen. In de tweede binnenste lus wordt de inhoud van N$(I) overeenkomstig de pointer Z(I) in het dataveld gesorteerd teruggeschreven. Vanaf regel 3210 worden de hulpvelden weer gewist. Eerst worden de strings gewist om de stringvector in het notitieschrift van de computer terug te zetten. Via het sub-programma in de regels 3520-3540 wordt bij de actuele start van de veldtabel de geredde lengte opgeteld en als eind van de veldtabel in de vector 49, 50 geschreven.

Ik hoop dat het programma u nu duidelijk is en dat u het programma naar uw eigen wensen weet aan te passen. Bijvoorbeeld door:

- printuitvoer; volgens het geformateerde databestand

- hulppagina; deze beeldschermuitvoer zou u op een aparte beeldschermpagina kunnen schrijven die dan door de POKE-commando's met het actuele beeldscherm wordt verwisseld. Dit is vooral voor de CBM 64 gebruikers interessant. Hier kunt u de extra beeldschermpagina onder het systeem-ROM (vanaf $A000) schrijven zodat geen BASIC-geheugenplaats verloren gaat.

14 CASSETTE-INHOUD EN CATALOGUS VOOR FAST-TAPE CASSETTES

In hoofdstuk 7 is een programma beschreven dat een inhoudsopgave voor uw cassettes samenstelt. Zo'n inhoudsopgave wilt u natuurlijk ook voor uw fast-tape cassettes maken. Daarom treft u hier een listing aan van een programma dat precies zo werkt als dat in hoofdstuk 7.

```
90 REM ************************************
95 REM *                                  *
100 REM*          INHOUD                  *
101 REM*                                  *
103 REM*  VAN FASTTAPE CASSETTES           *
104 REM*                                  *
105 REM*     (C) DIRK PAULISSEN           *
106 REM*                                  *
107 REM************************************
108 REM V=.7:KL=.54 :REM CONSTANTEN VOOR
 TELLERBEREKENINGEN  C60
109 V=.5:KL=.482:REM CONSTANTEN VOOR TEL
LERBEREKENINGEN  C90
110 PA=828:REM STARTADRES IN CASSETTEBUF
FER
115 SA=171 :REM SECUNDAIRADRES +1
116 PRINT"□          ****    INHOUD    **
**"
117 PRINT:PRINT"    **** VOOR FASTTAPE
CASSETTES  ****       "
120 PRINT"■■       PRINTEN ? [J/N]  ";D$
130 GET D$:IF D$="" THEN 130
136 TA$=CHR$(16)
140 T1$=TA$+"05":T2$=TA$+"10":T3$=TA$+"1
9":T4$=TA$+"27":T5$=TA$+"35"
142 T6$=TA$+"43":T7$=TA$+"61":REM TAB'S
145 IF D$ <> "J" THEN 200
150 OPEN2,4 :REM PRINTERKANAAL OPENEN
160 PRINT#2,T1$"LFN";T2$"TELLER";T3$"K-B
YTE";T4$" AANVANG"T5$"   EINDE";
162 PRINT#2,T6$"     NAAM"T7$"SECADR."
165 PRINT#2
190 SYS49674:REM VC20 = 29183
200 A$=":"+RIGHT$("     "+STR$(PEEK(PA)+
(256*PEEK(PA+1))-1),6)
210 B$="-"+RIGHT$("    "+STR$(PEEK(PA+2)
```

```
+(256*PEEK(PA+3))-1),6)
220 C$=": ":FOR I=5 TO 20:C$=C$+CHR$(PEE
K(PA+I)):NEXT:C$=C$+":"
230 T=VAL(RIGHT$(B$,5))-VAL(RIGHT$(A$,5)
):K$=STR$(INT(T/1024*1000+.5)/1000)
235 K$=RIGHT$("        "+STR$(INT(T/1024*
1000+.5)/1000),6)
240 SA$=STR$(PEEK(SA)-1)
250 PRINT Z;K$A$B$:PRINT C$:PRINT
252 FA=1+2*π*(Z/4400):REM AFHANKELIJKE F
ACTOR VOOR TELLERBEREKENING
255 Z1=Z:Z=INT(Z+V*FA+(T/1024)*(KL*FA)):
REM BANDTELLER BEREKENEN
256 PRINT"VOLGENDE TELLERSTAND CA."Z
260 N=N+1:N$=RIGHT$("        "+STR$(N),3)
270 IF D$="J" THEN PRINT#2,T1$;N$;T2$;Z1
;T3$;K$;T4$;A$;T5$;B$;T6$;C$;T7$SA$
300 GOTO 190
```

Het in paragraaf 10.1 beschreven catalogusprogramma werkt na de benodigde verandering eveneens met fast-tape. U moet het normale LOAD-commando door het fast-tape commando vervangen. Omdat de synchronisatie bij fast-tape programma's aanzienlijk korter is dan bij normaal opgeslagen programma's, is het raadzaam de programma's niet direkt achter elkaar op te slaan. Beter is steeds een tussenruimte van circa 5 seconden te nemen.

15 BACK-UP VAN DISK OP CASSETTE EN OMGEKEERD

Dit hoofdstuk is voor diegenen die zowel een diskdrive als een cassetterecorder bezitten. Een cassette is aanzienlijk steviger en minder storingsgevoelig dan een diskette. Daarom is het verstandig, ook voor diskette-bezitters, belangrijke programma's op cassette op te slaan. Met het volgende programma kunt u als u zelf alleen een cassetterecorder bezit, op een eenvoudige manier programma's uitwisselen met vrienden die een diskdrive hebben.

15.1 Back-up van cassette op disk

Met dit back up-programma kunt u verschillende fast-tapeprogramma's die achter elkaar op een cassette zijn opgeslagen, op een diskette opslaan. Voor dit programma is het niet van belang of het om BASIC-programma's of om machinetaalprogramma's gaat. De programma's worden zo gekopieerd, dat ze onafhankelijk lopen van het geheugenmedium van waar uit u ze laadt. Belangrijk is alleen dat alle programma's die u wilt kopieren een naam hebben, omdat de diskdrive in tegenstelling tot de datarecorder naamloze files niet accepteert. Omdat het back up-programma met fast-tape werkt, moet u het fast-tape programma laden voordat u het back up-programma start. De bediening is zeer eenvoudig. Voer het programma in uw computer in. Sla het voor het starten op, want het wist zichzelf na het starten uit. Als u het programma heeft gestart, wordt eerst het machinetaalprogramma in het geheugen geschreven achter het fast-tapeprogramma vanaf $C3C0 ($73C0). Daarna vraagt de computer u naar het aantal te kopieren programma's. Deze waarde wordt tijdelijk opgeslagen in de geheugenplaats 767 voor het machinetaalprogramma. Vervolgens moet u een geformateerde diskette in de diskdrive doen en de cassette met de te kopieren programma's in de datarecorder stoppen. Na een druk op de RETURN-toets vertakt het BASIC-programma zich in het machinetaalprogramma. Dit leest nu het eerste programma in het BASIC-RAM (het complete BASIC-RAM wordt als buffer gebruikt). Daarna wordt het programma uit het geheugen op diskette geschreven. Dit herhaalt zich net zo vaak als het aantal programma's. Aan het eind keert de computer weer in de direkte modus terug. Wilt u nog meer programma's kopieren, dan moet u het aantal programma's in de geheugenplaats 767 POKEn en het machinetaalprogramma opnieuw door SYS 50112 (SYS 28672 bij VIC 20) starten.

```
c3c0                    .pag 61,5
c3c0                    .tit "backup t=>d"
c3c0                    .opt p1
c3c0                    *=   $c3c0        ;vc20 = $73c0
c3c0
c3c0
                    ; ***********************************
                    ; *                                 *
                    ; *     backup van cassette naar    *
                    ; *                                 *
                    ; *     diskette met fasttape       *
                    ; *                                 *
                    ; ***********************************
c3c0
c3c0
c3c0
c3c0
                    ; waarden tussen haakjes
                    ;      voor  vc 20
c3c0
c3c0
000d                cr       =    13       ;
0002                absfl9   =    2
00ac                startv   =    $ac
00ae                endvec   =    $ae
033c                casbuf   =    $33c
02ff                aantal   =    $2ff
0803                cstart   =    $803     ; ($1203) bufferstart
c18e                loadr    =    $c18e    ; ($7188) fasttape laden
a642                new      =    $a642    ; ($c642) basic-instr. new
a437                error    =    $a437    ; ($c437) basic-warmstart
f693                meld     =    $f693    ; ($f72c) melding uitvoeren
c3c0
c3c0
ffb7                status   =    $ffb7
ffba                setlfs   =    $ffba
ffbd                setnam   =    $ffbd
ffc0                open     =    $ffc0
ffc3                close    =    $ffc3
ffc9                chkout   =    $ffc9
ffcc                clrch    =    $ffcc
ffd2                bsout    =    $ffd2
c3c0
c3c0
c3c0
c3c0
                    ; hoofdprogramma
c3c0
c3c0
c3c0 a9 00      start    lda  #$00
c3c2 20 42 a6            jsr  new
c3c5 20 70 c4            jsr  cload
c3c8 20 d4 c3            jsr  dsave
c3cb ce ff 02            dec  aantal
c3ce f0 03               beq  einde
c3d0 4c c0 c3            jmp  start
c3d3 60         einde    rts
c3d4
c3d4
c3d4
                    ; opslaan op disk
c3d4
c3d4
```

```
1089:  c3d4 a9 10     dsave     lda  #$10
1090:  c3d6 a2 41               ldx  #$41     ; naamparameter
1092:  c3d8 a0 03               ldy  #$03     ; zetten
1093:  c3da 20 bd ff            jsr  setnam
1095:  c3dd 20 93 f6            jsr  meld     ; ' saving naam ' uitvoeren
1100:  c3e0 a9 0d               lda  #cr
1110:  c3e2 20 d2 ff            jsr  bsout    ; nieuwe regel
1120:  c3e5 a2 10               ldx  #$10
1130:  c3e7 ca        elooP     dex
1140:  c3e8 bd 41 03            lda  casbuf+5,x
1150:  c3eb c9 20               cmP  #$20
1160:  c3ed d0 04               bne  endnam
1170:  c3ef e0 00               cPx  #$00
1180:  c3f1 d0 f4               bne  elooP
1190:  c3f3 e8        endnam    inx           ; einde v/d filenaam
1200:  c3f4 a9 2c               lda  #","     ; zoeken en ' ,P,w '
1210:  c3f6 9d 41 03            sta  casbuf+5,x
1220:  c3f9 e8                  inx           ; toevoegen
1230:  c3fa a9 50               lda  #"P"
1240:  c3fc 9d 41 03            sta  casbuf+5,x
1250:  c3ff e8                  inx
1260:  c400 a9 2c               lda  #","
1270:  c402 9d 41 03            sta  casbuf+5,x
1280:  c405 e8                  inx
1290:  c406 a9 57               lda  #"w"
1300:  c408 9d 41 03            sta  casbuf+5,x
1310:  c40b e8                  inx
1320:  c40c 8a                  txa           ; lengte v/d filenaam
1330:  c40d a2 41               ldx  #$41     ; startadr        ' '
1340:  c40f a0 03               ldy  #$03
1350:  c411 20 bd ff            jsr  setnam
1360:  c414 a9 08               lda  #$08     ; filenummer
1370:  c416 a2 08               ldx  #$08     ; device nummer
1380:  c418 a0 01               ldy  #$01     ; secundair adr.
1390:  c41a 20 ba ff            jsr  setlfs
1400:  c41d 20 c0 ff            jsr  oPen
1410:  c420 a2 08               ldx  #$08
1420:  c422 20 c9 ff            jsr  chkout
1430:  c425 a9 03               lda  #<cstart
1440:  c427 a2 08               ldx  #>cstart
1450:  c429 85 ac               sta  startv
1460:  c42b 86 ad               stx  startv+1
1470:  c42d ad 3e 03            lda  casbuf+2 ; eindadres berekenen
1480:  c430 38                  sec           ; uit verschil start- + eindadr.
1490:  c431 ed 3c 03            sbc  casbuf   ; uit de header Plus bePaalde
1500:  c434 08                  PhP           ; startadres
1510:  c435 18                  clc
1520:  c436 65 ac               adc  startv
1530:  c438 85 ae               sta  endvec
1540:  c43a ad 3f 03            lda  casbuf+3
1550:  c43d 65 ad               adc  startv+1
1560:  c43f 28                  PlP
1570:  c440 ed 3d 03            sbc  casbuf+1
1580:  c443 85 af               sta  endvec+1
1590:  c445 ad 3c 03            lda  casbuf   ; Programma startadr.
1600:  c448 20 d2 ff            jsr  bsout    ; naar disk zenden
1610:  c44b ad 3d 03            lda  casbuf+1
1620:  c44e 20 d2 ff            jsr  bsout
1622:  c451
1622:  c451
1623:                 ; save lus
```

```
1624:  c451
1624:  c451
1630:  c451 a0 00     psave     ldy  ##$00
1640:  c453 b1 ac               lda  (startv),y
1650:  c455 20 d2 ff            jsr  bsout
1660:  c458 e6 ac               inc  startv  ; adres verhogen
1670:  c45a d0 02               bne  nothi
1680:  c45c e6 ad               inc  startv+1
1690:  c45e a5 ac     nothi     lda  startv
1700:  c460 c5 ae               cmp  endvec  ; eindadres bereikt
1710:  c462 a5 ad               lda  startv+1
1720:  c464 e5 af               sbc  endvec+1
1730:  c466 90 e9               bcc  psave
1740:  c468 20 cc ff            jsr  clrch
1750:  c46b a9 08               lda  ##$08
1760:  c46d 4c c3 ff            jmp  close
1762:  c470
1762:  c470
1764:  c470
1764:  c470
1766:                 ; van cass. laden
1768:  c470
1768:  c470
1770:  c470 a2 00     cload     ldx  ##$00
1780:  c472 a0 03               ldy  #<cstart
1790:  c474 a9 08               lda  #>cstart
1800:  c476 86 0a               stx  $0a      ; load/verify flag
1810:  c478 86 93               stx  $93
1820:  c47a 84 ac               sty  $ac
1830:  c47c 85 ad               sta  $ad
1840:  c47e a9 05               lda  #1+4     ; op beschalde adres
1850:  c480 85 02               sta  absfl9   ; niet wachten
1860:  c482 a9 00               lda  ##$00    ; geen filenaam
1870:  c484 20 bd ff            jsr  setnam
1880:  c487 a2 01               ldx  ##$01    ; ga
1890:  c489 a0 00               ldy  ##$00    ; sa
1900:  c48b 20 ba ff            jsr  setlfs
1910:  c48e 20 8e c1            jsr  loadr
1922:  c491
1922:  c491
1924:                 ; status testen
1926:  c491
1926:  c491
1930:  c491 20 b7 ff holsta   jsr  status
1940:  c494 29 bf             and  ##$bf
1950:  c496 f0 05             beq  ok
1960:  c498 a2 1d             ldx  ##$1d
1970:  c49a 4c 37 a4          jmp  error
1980:  c49d 60        ok      rts
lc3c0-c49e
```

```
100 GOSUB240
110 PRINT"██████████████████████****************
*****"
120 PRINT"█         ** BACKUP MET FASTTA
PE **"
130 PRINT"█         ** VAN CASSETTE > DI
SK **"
140 PRINT"█         ** (C) DIRK PAULISSE
N  **"
150 PRINT"█         ********************
*****"
160 PRINT:PRINT:PRINT"          HOEVEEL P
ROGRAMMA'S WILT U"
170 PRINT:INPUT"          COPIEREN ";AZ
180 POKE 767,AZ
190 PRINT:PRINT:PRINT"          LEG DE OR
IGINEELE BAND IN DE"
200 PRINT"█         RECORDER EN DRUK OP
< RETURN > "
210 GET A$:IF A$<>CHR$(13) THEN 210
220 SYS29632
230 END
240 REM LAADPROGRAMMA FASTCOPY T-D 20
250 E=29853:A=29632:PS=0
260 FOR I=A TO E:READ X:POKE I,X:PS=PS+X
:NEXT
280 IF PS <> 27173 THEN PRINT"FOUT IN DA
TA":END
290 RETURN
300 DATA169,0,32,66,198,32,112,116,32,21
2,115,206,255,2,240,3,76,192,115,96
310 DATA169,16,162,65,160,3,32,189,255,3
2,44,247,169,13,32,210,255,162,16,202
320 DATA189,65,3,201,32,208,4,224,0,208,
244,232,169,44,157,65,3,232,169,80
330 DATA157,65,3,232,169,44,157,65,3,232
,169,87,157,65,3,232,138,162,65,160
340 DATA3,32,189,255,169,8,162,8,160,1,3
2,186,255,32,192,255,162,8,32,201,255
350 DATA169,3,162,18,133,172,134,173,173
,62,3,56,237,60,3,8,24,101,172,133
360 DATA174,173,63,3,101,173,40,237,61,3
,133,175,173,60,3,32,210,255,173,61
370 DATA3,32,210,255,160,0,177,172,32,21
0,255,230,172,208,2,230,173,165,172
380 DATA197,174,165,173,229,175,144,233,
32,204,255,169,8,76,195,255,162,0,160
390 DATA3,169,18,134,10,134,147,132,172,
133,173,169,5,133,2,169,0,32,189,255
```

```
400 DATA162,1,160,0,32,186,255,32,136,11
3,32,183,255,41,191,240,5,162,29,76
410 DATA55,196,96



5 PRINT"**********************************
**********"
100 GOSUB240
110 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
*****"
120 PRINT"▒          ** BACKUP MET FASTTA
PE **"
130 PRINT"▒          ** VAN CASSETTE > DI
SK **"
140 PRINT"▒          ** (C) DIRK PAULISSE
N  **"
150 PRINT"▒          **********************
*****"
160 PRINT:PRINT:PRINT"          HOEVEEL P
ROGRAMMA'S WILT U"
170 PRINT:INPUT"          KOPIEREN ";AZ
180 POKE 767,AZ
190 PRINT:PRINT:PRINT"          LEG DE OR
IGINELE BAND IN DE"
200 PRINT"▒          RECORDER EN DRUK OP
< RETURN >"
210 GET A$:IF A$<>CHR$(13) THEN 210
220 SYS50112
230 END
240 REM LAADPROGRAMMA FASTCOPY T-D 64
250 E=50333:A=50112:PS=0
260 FOR I=A TO E:READ X:POKE I,X:PS=PS+X
:NEXT
270 IF PS<>27517 THEN PRINT"FOUT IN DATA
":END
280 RETURN
290 DATA169,0,32,66,166,32,112,196,32,21
2,195,206,255,2,240,3,76,192,195,96
300 DATA169,16,162,65,160,3,32,189,255,3
2,147,246,169,13,32,210,255,162,16
310 DATA202,189,65,3,201,32,208,4,224,0,
208,244,232,169,44,157,65,3,232,169
320 DATA80,157,65,3,232,169,44,157,65,3,
232,169,87,157,65,3,232,138,162,65
330 DATA160,3,32,189,255,169,8,162,8,160
,1,32,186,255,32,192,255,162,8,32,201
```

```
340 DATA255,169,3,162,8,133,172,134,173,
173,62,3,56,237,60,3,8,24,101,172,133
350 DATA174,173,63,3,101,173,40,237,61,3
,133,175,173,60,3,32,210,255,173,61
360 DATA3,32,210,255,160,0,177,172,32,21
0,255,230,172,208,2,230,173,165,172
370 DATA197,174,165,173,229,175,144,233,
32,204,255,169,8,76,195,255,162,0,160
380 DATA3,169,8,134,10,134,147,132,172,1
33,173,169,5,133,2,169,0,32,189,255
390 DATA162,1,160,0,32,186,255,32,142,19
3,32,183,255,41,191,240,5,162,29,76
400 DATA55,164,96
```

15.2 Programmabeschrijving back-up cassette-disk

Voor diegenen die enige basiskennis van de assemblerprogrammering hebben, volgt hier de uitleg van de assemblerlisting. De hoofdlus van het programma staat in de geheugenplaatsen $C3C0 tot en met $C3D3. Eerst worden door het NEW-commando alle BASIC-vectoren teruggezet. Daarna wordt naar de cassette-load-routine vertakt. Na de uitvoering van dit sub-programma, wordt het programma door DSAVE op disk opgeslagen. Nadat AANTAL werd gedecrementeerd en getest, wordt er naar start teruggesprongen.

CLOAD ($C469)

De bufferstart CSTART wordt in de geheugenplaatsen $AC, $AD overgebracht. Uit deze geheugenplaatsen haalt fast-tape het gevraagde laadadres. Verder worden de LOAD/VERIFY-flags op 0=LOAD gezet. Door het zetten van bit 0 en 2 in ABSFLG laat u het fast-tapeprogramma op dat adres laden dat in $AC en $AD werd overgebracht. Na het vinden van het programma laat u het programma niet stoppen. Vanaf $C47B worden de noodzakelijke voorbereidingsroutines van het gebruikssysteem ZETNAM en ZETLFS opgeroepen. Daarna wordt naar het fast-tape subprogramma LOADR vertakt. Na het laadproces wordt de status getest en wordt in de hoofdlus teruggesprongen.

DSAVE

Het opslaan op diskette is iets moeilijker. Eerst wordt via de bedrijfsroutines "SAVING naam" uitgevoerd. Van $C3DE tot en met $C3EA wordt het einde van de filenaam die in de cassettebuffer staat, bepaald. Dit gebeurt doordat vanaf het einde net zolang op "SPACE" wordt getest tot er een ander teken wordt gevonden. In de volgende regels tot en met $C401 wordt ",P,W" (Programm, Write) aan de filenaam gekoppeld. Via het X-register wordt de lengte van de complete filenaam bepaald. Vanaf $C465 beginnen de noodzakelijke voorbereidingsroutines ZETNAM, ZETLFS, OPEN, CHKOUT voor het opslaan van de diskettes. Hierna wordt de startvector weer op de bufferstart gezet. Het eindadres van het geladen programma wordt berekend met behulp van het verschil tussen start en eindadres uit de cassettebuffer + startadres van de buffer ($C426-$C43C). Bij de diskette-opslag geven de eerste beide bytes het adres aan, waarvandaan werd opgeslagen. Dit adres staat als startadres in de cassettebuffer en wordt in $CH3E-$C447 aan de diskette doorgegeven. Vanaf PSAVE ($C44A) begint de eigenlijke programma-opslag. In het X-register wordt 0 geladen, terwijl de programmabytes indirekt geïndexeerd in de accumulator worden geladen en door BASOUT naar de diskdrive worden gestuurd. Daarna wordt de startvector geïncrementeerd en met de eindvector vergeleken. Zijn ze niet gelijk, dan wordt weer naar PSAVE gesprongen. Zijn de vectoren gelijk, dan wordt het kanaal van de diskette-uitvoer gesloten en wordt naar de hoofdlus teruggesprongen.

15.3 Back-up van diskette naar cassette

Met dit programma kunt u gemakkelijk een of meer programma's van een diskette op een of meer cassettes overbrengen. De invoer vindt plaats door een BASIC-lader die u na de programmabeschrijving vindt. Vergeet u echter niet deze BASIC-lader voor het starten op te slaan. Dit programma lukt alleen, als u fast-tape al heeft geladen.

Bediening van het programma

Het programma wordt gestart door SYS 50176 (SYS 29696 bij VIC 20). Direkt verschijnt de titel op het beeldscherm en wordt u gevraagd

de te kopiëren diskette in de diskdrive te stoppen en een willekeurige toets in te drukken. Na korte tijd verschijnt de titel van de diskette. Ook wordt de eerste file-invoer van de inhoudsopgave van de diskette met de vraag JA/NEE zichtbaar. Wanneer u het programma wilt kopiëren, drukt u op "J" en verschijnt er JA achter de file-invoer. Vervolgens wordt de volgende file-invoer getoond. Wilt u het programma niet kopiëren, dan drukt u op "N" wat met NEE op het beeldscherm wordt aangeduid. Op deze manier kunt u maximaal 48 programma's met JA merken. Indien u binnen het programma een tikfout maakt, kunt u door een druk op de RUN/STOP-toets het programma afbreken en naar de titelpagina terugkeren. Nadat u de hele inhoudsopgave heeft afgewerkt, wordt u gevraagd of u de programma's apart of achter elkaar wilt kopiëren. Wilt u alle programma's op een diskette opslaan, dan voert u "1" in. Als u "2" invoert, geeft de computer steeds aan welke cassette hij gaat inlezen en wacht op het indrukken van een toets. Zo kunt u voor elk programma een andere cassette gebruiken. Aan het eind van het kopiëren keert de computer weer naar de titelpagina terug. U kunt nu een nieuwe diskette kopiëren of door het indrukken van "E" in BASIC terugkeren.

```
60:    C400                    .OPT P1
95:                    ;**********************************
96:                    ;*                                *
100:                   ;*          BACKUP VAN            *
110:                   ;*                                *
115:                   ;*      DISKETTE NAAR CASSETTE    *
117:                   ;*                                *
120:                   ;*         MET FASTTAPE           *
122:                   ;*                                *
125:                   ;**********************************
130:                   ;
132:                   ; START MET SYS 50176 (SYS 29696)
134:   C400
134:                   ; WAARDES EN TOELICHTINGEN TUSSEN
136:   C400
136:                   ; HAAKJES GELDEN VOOR VC-20
140:                   ;
142:   C400                        *=   $C400    ; ($7400)
145:   C400
145:   C400            START       =    *
150:   E544            CLSCRN      =    $E544    ;($E55F) BEELDSCHERM WISSEN
152:   BDCD            NUMOUT      =    $BDCD    ;($DDCD) POS. INTEGER GETAL UITV.
154:   FCCA            MOTUIT      =    $FCCA    ;($FD08) MOTOR UIT
156:   F838            PTOETS      =    $F838    ;($F8B7) RECORDERTOETS UITLEZEN
158:   F30F            SFINUM      =    $F30F    ;($F3CF) ZOEKT FILENUMMER
160:   AB1E            STROUT      =    $AB1E    ;($CB1E) STRING UITVOEREN
162:   F31F            SETPAR      =    $F31F    ;($F3DF) ZET FILEPARAMETER
164:   C085            ABSOLUT     =    $C085    ;($707F) SPRONG IN FASTTAPE
169:   C400
169:   C400
```

```
169:  C400
170:  FFE4          GETBYT   =    $FFE4   ;BYTE VAN INVOERAPPARAAT HALEN
180:  FFBA          SETLFS   =    $FFBA   ;FILEPARAMETER ZETTEN
190:  FFBD          SETNAM   =    $FFBD   ;FILENAAMPARAMETER ZETTEN
200:  FFC0          OPEN     =    $FFC0
210:  FFC3          CLOSE    =    $FFC3
220:  FFB4          TALK     =    $FFB4
230:  FF96          SETALK   =    $FF96   ;SECUNDAIR ADR. NAAR TALK
240:  FFA5          IECIN    =    $FFA5   ;BYTE VAN IEC-BUS HALEN
250:  FFE7          CLALL    =    $FFE7
260:  FFAB          UNTALK   =    $FFAB   ;UNTALK UITVOEREN
280:  FFE1          STOP     =    $FFE1   ;STOPTOETS UITLEZEN
290:  FFD2          BASOUT   =    $FFD2   ;BYTE UITVOEREN
300:  C400
300:  C400
300:  C400
320:  00BA          DN       =    $BA     ;DEVICE NUMMER
330:  00B9          SA       =    $B9     ;SECUNDAIRADRES
340:  0098          MAXBLK   =    $98     ;($5D) MAX LAADBARE BLOKKEN
350:  000D          CR       =    $0D     ;CARRIAGE RETURN
360:  00FB          COPANZ   =    $FB     ;ACTUELE LENGTE
370:  00FC          TEMP     =    $FC     ;HULPBUFFER
380:  00FE          FLAG     =    $FE     ;ENKELE FLAG
390:  0041          LISTPT   =    $41     ;POINTER OP NAMENLIJST
400:  0340          FILBUF   =    $340    ;FILEBUFFER
410:  CA40          LNGTAB   =    START+$640  ;TAB. V/D NAAMLENGTE
420:  CB00          NAMTAB   =    START+$700  ;TAB. V/D FILENAAM
430:  0030          LSTMAX   =    $30     ;MAXIMALE LIJSTLENGTE
440:  0009          TLSADR   =    $09     ;($13) HIBYTE V/H LAADADRES
450:  0022          TEMP2    =    $22     ;HULPPOINTER
460:  00AC          WPGSTA   =    $AC     ;PTR VOOR WERKEL. PRGSTART
470:  00AE          WPGEND   =    $AE     ;PTR VOOR WERKEL. PRGENDE
480:  00A7          PRGSTA   =    $A7     ;PTR VOOR EIGENL. PRGSTART
490:  00A9          PRGEND   =    $A9     ;PTR VOOR EIGENL. PRGENDE
495:  00C0          MOFLAG   =    $C0     ;MOTORFLAG
520:                ;
530:                ;
560:  C400 A9 01          LDA  #1      ;(LDA #25)
570:  C402 8D 20 D0       STA  $D020   ;(STA $900F)
580:  C405 8D 21 D0       STA  $D021   ;(VERVALT)
590:  C408 A9 06          LDA  #6
600:  C40A 8D 86 02       STA  $286
610:  C40D A9 67          LDA  #<TITEL
620:  C40F A0 C8          LDY  #>TITEL
630:  C411 20 1E AB       JSR  STROUT
640:  C414 A9 34          LDA  #<ORGIN   ;"ORIGINEEL INLEGGEN "
650:  C416 A0 C8          LDY  #>ORGIN
660:  C418 20 1E AB       JSR  STROUT
670:  C41B 20 99 C7       JSR  WAIT
680:  C41E C9 45          CMP  #"E"    ;PROGRAMMA BEEINDIGEN
690:  C420 D0 01          BNE  OKCOPY
700:  C422 60             RTS
701:  C423
701:  C423
702:                ; INHOUD INLEZEN
703:  C423
703:  C423
710:  C423 20 B6 C7 OKCOPY  JSR  CARET
720:  C426 20 B6 C7         JSR  CARET
730:  C429 20 9F C7         JSR  INIT    ;DISK INITIALISEREN
740:  C42C A9 08            LDA  #$08
750:  C42E AA               TAX
760:  C42F A0 00            LDY  #$00
```

```
770:   C431 20 BA FF          JSR  SETLFS
780:   C434 A9 01             LDA  #$01
790:   C436 A2 DA             LDX  #<CATALO
800:   C438 A0 C9             LDY  #>CATALO
810:   C43A 20 BD FF          JSR  SETNAM
820:   C43D 20 C0 FF          JSR  OPEN
830:   C440 A9 08             LDA  #$08
840:   C442 20 B4 FF          JSR  TALK
850:   C445 A9 00             LDA  #$00
860:   C447 20 96 FF          JSR  SETALK
870:   C44A A0 04             LDY  #$04
880:   C44C 20 A5 FF LOOP1    JSR  IECIN   ;DE EERSTE 4 BYTES INLEZEN
890:   C44F 88                DEY
900:   C450 D0 FA             BNE  LOOP1
910:   C452 20 A5 FF          JSR  IECIN   ;BLOKGETAL INLEZEN
920:   C455 85 22             STA  TEMP2
930:   C457 20 A5 FF          JSR  IECIN
940:   C45A A6 22             LDX  TEMP2
950:   C45C 20 CD BD          JSR  NUMOUT  ;EN UITVOEREN
960:   C45F 20 B3 C7          JSR  SPACE
970:   C462 20 A5 FF LOOP2    JSR  IECIN   ;DISKNAAM ETC UITVOEREN
980:   C465 F0 06             BEQ  NUL
990:   C467 20 D2 FF          JSR  BASOUT
1000:  C46A 18                CLC
1010:  C46B 90 F5             BCC  LOOP2
1020:  C46D 20 B6 C7 NUL      JSR  CARET
1030:  C470 20 B6 C7          JSR  CARET
1040:  C473 20 A5 FF          JSR  IECIN
1050:  C476 20 A5 FF          JSR  IECIN
1060:  C479 A0 00             LDY  #$00    ;AANTAL TE KOPIEREN
1070:  C47B 84 FB             STY  COPANZ  ;FILES OP NUL ZETTEN
1071:  C47D
1071:  C47D
1072:                  ; INHOUD UITVOEREN EN VRAGEN
1073:  C47D
1073:  C47D
1080:  C47D 20 A5 FF LSTART   JSR  IECIN
1090:  C480 85 FC             STA  TEMP
1100:  C482 20 A5 FF          JSR  IECIN
1110:  C485 85 FD             STA  TEMP+1
1120:  C487 A6 FC             LDX  TEMP
1130:  C489 20 CD BD          JSR  NUMOUT  ;BLOKGETAL UITVOEREN
1140:  C48C 20 B3 C7          JSR  SPACE
1150:  C48F A0 00             LDY  #$00    ;FILENAAM
1160:  C491 20 A5 FF WRFILE   JSR  IECIN   ;INLEZEN
1170:  C494 48                PHA
1180:  C495 20 BB C7          JSR  UITVOR  ;UITVOEREN
1190:  C498 68                PLA          ;EN IN
1200:  C499 99 40 03          STA  FILBUF,Y  ;BUFFER SCHRIJVEN
1210:  C49C F0 03             BEQ  NAMEND
1220:  C49E C8                INY
1230:  C49F D0 F0             BNE  WRFILE
1240:  C4A1 20 A5 FF NAMEND   JSR  IECIN
1250:  C4A4 20 A5 FF          JSR  IECIN
1260:  C4A7 A5 90             LDA  $90     ;STATUS UITLEZEN
1270:  C4A9 F0 03             BEQ  VRAGEN
1280:  C4AB 4C 47 C5          JMP  INHEND
1290:  C4AE A5 FD     VRAGEN  LDA  TEMP+1
1300:  C4B0 D0 06             BNE  TOLONG
1310:  C4B2 A5 FC             LDA  TEMP
1320:  C4B4 C9 98             CMP  #MAXBLK  ;OP MAXIMALE LENGTE
1330:  C4B6 90 0A             BCC  LNGEOK  ;TESTEN
1340:  C4B8 A9 FA     TOLONG  LDA  #<SLONG  ;"TE LANG OM
```

```
1350:  C4BA A0 C8              LDY  #>SLONG  ;TE KOPIEREN"
1360:  C4BC 20 1E AB           JSR  STROUT  ;UITVOEREN
1370:  C4BF 4C 41 C5 FINISH    JMP  NXFILE
1380:  C4C2 A5 FB    LNGEOK   LDA  COPANZ
1390:  C4C4 C9 30              CMP  #LSTMAX
1400:  C4C6 90 0A              BCC  FANZOK
1410:  C4C8 A9 1A              LDA  #<LILONG  ;"LIJST TE LANG"
1420:  C4CA A0 C9              LDY  #>LILONG
1430:  C4CC 20 1E AB           JSR  STROUT  ;UITVOEREN
1440:  C4CF 18                 CLC
1450:  C4D0 90 ED              BCC  FINISH
1460:  C4D2 A9 00    FANZOK   LDA  #$00    ;"JA/NEE" UITVOEREN
1470:  C4D4 85 08              STA  $08
1480:  C4D6 A9 1F              LDA  #$1F    ;(VERVALT) TABULERING OVER
1490:  C4D8 85 D3              STA  $D3     ;(VERVALT)  KOLOMPOINTER
1500:  C4DA A9 30              LDA  #<JANEE
1510:  C4DC A0 C9              LDY  #>JANEE
1520:  C4DE 20 1E AB           JSR  STROUT
1530:  C4E1 20 E1 C7 LOOP3     JSR  INPUT
1540:  C4E4 C9 4E              CMP  #"N"
1550:  C4E6 F0 52              BEQ  NEE
1560:  C4E8 C9 4A              CMP  #"J"
1570:  C4EA D0 F5              BNE  LOOP3
1580:  C4EC A9 41              LDA  #<STRJA  ;JA UITVOEREN
1590:  C4EE A0 C9              LDY  #>STRJA
1600:  C4F0 20 1E AB           JSR  STROUT
1610:  C4F3 A5 FB              LDA  COPANZ
1620:  C4F5 20 F1 C6           JSR  SETPTR
1630:  C4F8 A2 00              LDX  #$00
1640:  C4FA E8        LOOP4    INX           ;FILENAAMSTART ZOEKEN
1650:  C4FB BD 40 03           LDA  FILBUF,X
1660:  C4FE C9 22              CMP  #$22    ;1.APOSTROF = START
1670:  C500 D0 F8              BNE  LOOP4
1680:  C502 86 FD              STX  TEMP+1  ;START REDDEN
1690:  C504 E8                 INX
1700:  C505 BD 40 03 OMZETT    LDA  FILBUF,X  ;NAMEN IN LIJST SCHRIJVEN
1710:  C508 C9 22              CMP  #$22    ;2. APOSTROF = EINDE
1720:  C50A F0 06              BEQ  VERSTU
1730:  C50C 91 41              STA  (LISTPT),Y
1740:  C50E E8                 INX
1750:  C50F C8                 INY
1760:  C510 D0 F3              BNE  OMZETT
1770:  C512 8A        VERSTU   TXA           ;EINDPUNT IN ACCU
1780:  C513 A4 FB              LDY  COPANZ  ;INDEX IN Y-REG.
1790:  C515 18                 CLC
1800:  C516 E5 FD              SBC  TEMP+1  ;LENGTE BEREKENEN
1810:  C518 99 40 CA           STA  LNGTAB,Y  ;IN TABEL SCHRIJVEN
1820:  C51B BD 40 03 WETEST    LDA  FILBUF,X
1830:  C51E D0 0A              BNE  TYPTST
1840:  C520 A9 4D     ILL      LDA  #<STRILL  ;"VERKEERDE FILETYPE"
1850:  C522 A0 C9              LDY  #>STRILL
1860:  C524 20 1E AB           JSR  STROUT  ;UITVOEREN
1870:  C527 4C 41 C5           JMP  NXFILE
1880:  C52A C9 53     TYPTST   CMP  #"S"    ;TEST OP PRG FILE
1890:  C52C F0 F2              BEQ  ILL
1900:  C52E C9 50              CMP  #"P"
1910:  C530 F0 03              BEQ  TYPOK
1920:  C532 E8                 INX
1930:  C533 D0 E6              BNE  WETEST
1940:  C535 E6 FB     TYPOK    INC  COPANZ  ;AANT. TE KOPIEREN
1950:  C537 18                 CLC           ;FILES VERHOGEN
1960:  C538 90 07              BCC  NXFILE
1970:  C53A A9 67     NEE      LDA  #<STRNEE ;"NEE" UITVOEREN
```

```
1980:  C53C A0 C9          LDY  #>STRNEE
1990:  C53E 20 1E AB       JSR  STROUT
2000:  C541 20 B6 C7 NXFILE JSR  CARET
2010:  C544 4C 7D C4       JMP  LSTART
2020:  C547 A9 08    INHEND LDA  #$08     ;FILE SLUITEN
2030:  C549 20 C3 FF       JSR  CLOSE
2040:  C54C A5 FB          LDA  COPANZ   ;MOETEN FILES GEKOPIEERD
2050:  C54E D0 03          BNE  COPYST   ;WORDEN/NEE => START
2060:  C550 4C 00 C4       JMP  START
2062:  C553
2062:  C553
2062:  C553
2064:                ;KOPIEREN VOORBEREIDEN
2066:  C553
2066:  C553
2066:  C553
2070:  C553 A9 72    COPYST LDA  #<VRAAG  ;"CONTINU OF EEN
2080:  C555 A0 C9          LDY  #>VRAAG  ; VOOR EEN"
2090:  C557 20 1E AB       JSR  STROUT   ;UITVOEREN
2100:  C55A 20 E1 C7 UITL2 JSR  INPUT
2110:  C55D C9 31          CMP  #"1"
2120:  C55F F0 07          BEQ  ENFLG
2130:  C561 C9 32          CMP  #"2"
2140:  C563 D0 F5          BNE  UITL2
2150:  C565 A9 00          LDA  #$00     ;OVEREENKOMSTIGE FLAG
2160:  C567 2C             .BYT $2C
2170:  C568 A9 FF    ENFLG LDA  #$FF
2180:  C56A 85 FE          STA  FLAG
2190:  C56C A9 0F          LDA  #$0F     ;FOUTKANAAL SLUITEN
2200:  C56E 20 C3 FF       JSR  CLOSE
2202:  C571
2202:  C571
2202:  C571
2204:                ;KOPIEER - ROUTINE
2206:  C571
2206:  C571
2206:  C571
2208:                ;INLEZEN VAN DISK
2209:                ;   VOORBEREIDEN
2210:  C571 A2 00          LDX  #$00     ;TEMP WISSEN
2220:  C573 86 FC          STX  TEMP     ;TEMP => PRG-TELLER
2230:  C575 A9 C0    LOLOOP LDA  #<SREAD  ;"READING"
2240:  C577 A0 C9          LDY  #>SREAD
2250:  C579 20 1E AB       JSR  STROUT   ; UITVOEREN
2260:  C57C A4 FC          LDY  TEMP     ;POS. IN TABEL
2270:  C57E BE 40 CA       LDX  LNGTAB,Y  ;LENGTE NAAR X-REG.
2280:  C581 A5 FC          LDA  TEMP
2290:  C583 20 F1 C6       JSR  SETPTR
2300:  C586 B1 41    WRTNAM LDA  (LISTPT),Y  ;FILENAAM UITVOEREN
2310:  C588 20 D2 FF       JSR  BASOUT
2320:  C58B C8             INY
2330:  C58C CA             DEX
2340:  C58D D0 F7          BNE  WRTNAM
2350:  C58F A9 02          LDA  #$02     ;FILENUMMER
2360:  C591 A2 08          LDX  #$08     ;GA
2370:  C593 A8             TAY           ;SA
2380:  C594 20 BA FF       JSR  SETLFS
2390:  C597 A6 FC          LDX  TEMP
2400:  C599 BD 40 CA       LDA  LNGTAB,X
2410:  C59C 85 22          STA  TEMP2
2420:  C59E A5 FC          LDA  TEMP
2430:  C5A0 20 F1 C6       JSR  SETPTR
2440:  C5A3 A2 00          LDX  #$00
```

```
2450:  C5A5 B1 41    LOOP6    LDA  (LISTPT),Y  ;FILENAAM IN BUFFER
2460:  C5A7 9D 40 03          STA  FILBUF,X  ;SCHRIJVEN
2470:  C5AA C8                INY
2480:  C5AB E8                INX
2490:  C5AC C6 22             DEC  TEMP2
2500:  C5AE D0 F5             BNE  LOOP6
2510:  C5B0 A0 00             LDY  #$00
2520:  C5B2 B9 CB C9 LOOP7    LDA  PRGRE,Y  ; ",P,R " TOEVOEGEN"
2530:  C5B5 9D 40 03          STA  FILBUF,X
2540:  C5B8 C8                INY
2550:  C5B9 E8                INX
2560:  C5BA C0 04             CPY  #$04
2570:  C5BC 90 F4             BCC  LOOP7
2580:  C5BE 8A                TXA
2590:  C5BF A2 40             LDX  #<FILBUF
2600:  C5C1 A0 03             LDY  #>FILBUF
2610:  C5C3 20 BD FF          JSR  SETNAM
2620:  C5C6 20 C0 FF          JSR  OPEN
2630:  C5C9 A9 09             LDA  #TLSADR
2640:  C5CB A0 00             LDY  #$00
2650:  C5CD 84 22             STY  TEMP2
2660:  C5CF 85 23             STA  TEMP2+1
2670:  C5D1 AD 11 D0          LDA  $D011    ;(REGELS 2670-2690 VERVALT)
2680:  C5D4 29 EF             AND  #$EF     ;BEELDSCHERM UIT
2690:  C5D6 8D 11 D0          STA  $D011
2700:  C5D9 A9 0F             LDA  #$0F     ;KANAAL 15 OPENEN
2710:  C5DB A2 08             LDX  #$08
2720:  C5DD A8                TAY
2730:  C5DE 20 BA FF          JSR  SETLFS
2740:  C5E1 A9 03             LDA  #$03     ; DISKETTE
2750:  C5E3 A2 D0             LDX  #<UIMIN  ;SNEL SCHAKELEN
2760:  C5E5 A0 C9             LDY  #>UIMIN
2770:  C5E7 20 BD FF          JSR  SETNAM
2780:  C5EA 20 C0 FF          JSR  OPEN
2790:  C5ED A2 02             LDX  #$02     ;LAADKANAAL OPENEN
2800:  C5EF 20 0F F3          JSR  SFINUM
2810:  C5F2 20 1F F3          JSR  SETPAR
2820:  C5F5 A5 BA             LDA  DN
2830:  C5F7 20 B4 FF          JSR  TALK
2840:  C5FA A5 B9             LDA  SA
2850:  C5FC 20 96 FF          JSR  SETALK
2860:  C5FF A0 00             LDY  #$00
2862:  C601
2862:  C601
2864:                 ;PROGRAMMA IN BUFFER LADEN
2866:  C601
2866:  C601
2870:  C601 20 A5 FF LDLOOP   JSR  IECIN    ;PRG LADEN
2880:  C604 20 09 C7          JSR  STOBYT
2890:  C607 A6 90             LDX  $90
2900:  C609 F0 F6             BEQ  LDLOOP
2910:  C60B 20 EB C7          JSR  FOUT
2920:  C60E 08                PHP
2930:  C60F A9 02             LDA  #$02
2940:  C611 20 C3 FF          JSR  CLOSE
2950:  C614 AD 11 D0          LDA  $D011    ;(VERVALT) BEELDSCHERM AAN
2960:  C617 09 10             ORA  #$10     ;(VERVALT)
2970:  C619 8D 11 D0          STA  $D011    ;(VERVALT)
2980:  C61C 28                PLP
2990:  C61D 90 03             BCC  GEENFT
3000:  C61F 20 37 C7          JSR  FOUTML
3002:  C622
3002:  C622
```

```
3004:                 ;OPSLAAN OP CAS. VOORBEREIDEN
3006:  C622
3010:  C622 A6 FC     GEENFT  LDX  TEMP
3020:  C624 A5 22             LDA  TEMP2
3030:  C626 85 AE             STA  WPGEND
3040:  C628 A5 23             LDA  TEMP2+1
3050:  C62A 85 AF             STA  WPGEND+1
3060:  C62C E8                INX
3070:  C62D A5 FC             LDA  TEMP
3075:  C62F D0 0D             BNE  GEENML
3080:  C631 A9 4E             LDA  #<ZICASS  ;"CASSETTE INLEGGEN "
3090:  C633 A0 C8             LDY  #>ZICASS
3100:  C635 20 1E AB          JSR  STROUT
3105:  C638 20 E1 C7          JSR  INPUT
3110:  C63B 20 25 C8          JSR  TOETS    ;RECORDERTOETS UITLEZEN
3115:  C63E A9 DC     GEENML  LDA  #<WRITI
3120:  C640 A0 C9             LDY  #>WRITI
3125:  C642 20 1E AB          JSR  STROUT
3130:  C645 A4 FC             LDY  TEMP
3135:  C647 BE 40 CA          LDX  LNGTAB,Y
3140:  C64A A5 FC             LDA  TEMP
3145:  C64C 20 F1 C6          JSR  SETPTR
3150:  C64F B1 41     WRNAM2  LDA  (LISTPT),Y
3155:  C651 20 D2 FF          JSR  BASOUT
3160:  C654 C8                INY
3160:  C655 CA                DEX
3165:  C656 D0 F7             BNE  WRNAM2
3210:  C658 A2 01             LDX  #$01     ;FILEPARAMETER
3220:  C65A A9 00             LDA  #0       ;VOOR OPSLAG OP CAS-
3230:  C65C A8                TAY           ;SETTE
3240:  C65D 20 BA FF          JSR  SETLFS
3250:  C660 A6 FC             LDX  TEMP
3260:  C662 BD 40 CA          LDA  LNGTAB,X
3270:  C665 85 22             STA  TEMP2
3280:  C667 A5 FC             LDA  TEMP
3290:  C669 20 F1 C6          JSR  SETPTR
3300:  C66C A2 00             LDX  #$00
3310:  C66E B1 41     ULOOP   LDA  (LISTPT),Y  ;FILENAAM IN
3320:  C670 9D 40 03          STA  FILBUF,X  ;CASSETTEBUFFER
3330:  C673 E8                INX           ;SCHRIJVEN
3340:  C674 C8                INY
3350:  C675 C6 22             DEC  TEMP2
3360:  C677 D0 F5             BNE  ULOOP
3370:  C679 8A                TXA           ;FILENAAMPARAMETER
3380:  C67A A2 40             LDX  #<FILBUF
3390:  C67C A0 03             LDY  #>FILBUF
3400:  C67E 20 BD FF          JSR  SETNAM
3410:  C681 A0 00             LDY  #$00
3420:  C683 A9 09             LDA  #TLSADR
3425:  C685 84 AC             STY  WPGSTA
3430:  C687 85 AD             STA  WPGSTA+1
3440:  C689 B1 AC             LDA  (WPGSTA),Y  ;PRGSTARTADR.
3450:  C68B 85 A7             STA  PRGSTA   ;IN PRGSTARTVECTOR
3460:  C68D C8                INY
3470:  C68E B1 AC             LDA  (WPGSTA),Y
3480:  C690 85 A8             STA  PRGSTA+1
3490:  C692 C8                INY
3500:  C693 84 AC             STY  WPGSTA
3520:  C695 A5 AE             LDA  WPGEND   ;EINDADRES BEREKENEN
```

```
3530:  C697 38                SEC
3540:  C698 E5 AC             SBC  WPGSTA
3550:  C69A 08                PHP
3555:  C69B 18                CLC
3560:  C69C 65 A7             ADC  PRGSTA
3570:  C69E 85 A9             STA  PRGEND
3580:  C6A0 A5 AF             LDA  WPGEND+1
3590:  C6A2 65 A8             ADC  PRGSTA+1
3600:  C6A4 28                PLP
3610:  C6A5 E5 AD             SBC  WPGSTA+1
3620:  C6A7 85 AA             STA  PRGEND+1
3622:  C6A9
3622:  C6A9
3624:                  ;PRG OP CAS. OPSLAAN
3626:  C6A9
3626:  C6A9
3630:  C6A9 A2 05             LDX  #5       ;SYNCHRONISATIE
3640:  C6AB 86 AB             STX  $AB      ;HERHALING
3650:  C6AD 20 85 C0          JSR  ABSOLUT   ;FASTTAPE OPSLAAN
3690:  C6B0 20 E5 C6          JSR  SCRON    ;(VERVALT)
3700:  C6B3 E6 FC             INC  TEMP
3710:  C6B5 A6 FC             LDX  TEMP
3720:  C6B7 E4 FB             CPX  COPANZ
3730:  C6B9 B0 07             BCS  ZENDEN
3732:  C6BB 24 FE             BIT  FLAG
3733:  C6BD 10 10             BPL  ENFLG2
3740:  C6BF 4C DF C6          JMP  NXTFIL
3755:  C6C2 A9 E6     ZENDEN  LDA  #<EOCOP   ;"EINDE V/H KOPIEREN"
3760:  C6C4 A0 C9             LDY  #>EOCOP
3770:  C6C6 20 1E AB          JSR  STROUT   ;UITVOEREN
3780:  C6C9 20 E1 C7          JSR  INPUT
3790:  C6CC 4C 00 C4          JMP  START
3810:  C6CF 20 55 C7 ENFLG2   JSR  NAAMUI
3820:  C6D2 A9 4E             LDA  #<ZICASS  ;"CASSETTE INLEGGEN "
3830:  C6D4 A0 C8             LDY  #>ZICASS
3840:  C6D6 20 1E AB          JSR  STROUT   ;UITVOEREN
3845:  C6D9 20 E1 C7          JSR  INPUT    ;WACHT OP EEN TOETS
3850:  C6DC 20 25 C8          JSR  TOETS
3870:  C6DF 20 B6 C7 NXTFIL   JSR  CARET
3880:  C6E2 4C 75 C5          JMP  LOLOOP   ;VOLGENDE FILE LADEN
3882:  C6E5
3882:  C6E5
3882:  C6E5
3884:                  ;      SUBPROGRAMMA
3886:  C6E5
3886:  C6E5
3886:  C6E5
3887:                  ;BEELDSCHERM AAN
3888:  C6E5
3888:  C6E5
3890:  C6E5 20 73 C7 SCRON    JSR  SEUIP    ;(REGELS 3890-3930 VERVALLEN)
3900:  C6E8 AD 11 D0          LDA  $D011
3910:  C6EB 09 10             ORA  #$10
3920:  C6ED 8D 11 D0          STA  $D011
3930:  C6F0 60                RTS
3932:  C6F1
3932:  C6F1
3934:                  ; LISTPT OP OVEREENKOMSTIGE
```

```
3935:                   ; FILENAAM ZETTEN
3936:  C6F1
3936:  C6F1
3940:  C6F1 A0 00       SETPTR  LDY  #$00
3950:  C6F3 0A                  ASL  A
3960:  C6F4 0A                  ASL  A
3970:  C6F5 84 42               STY  LISTPT+1
3980:  C6F7 0A                  ASL  A
3990:  C6F8 26 42               ROL  LISTPT+1
4000:  C6FA 0A                  ASL  A
4010:  C6FB 26 42               ROL  LISTPT+1
4020:  C6FD 85 41               STA  LISTPT
4030:  C6FF A5 42               LDA  LISTPT+1
4040:  C701 18                  CLC
4050:  C702 69 CB               ADC  #>NAMTAB
4060:  C704 85 42               STA  LISTPT+1
4070:  C706 A0 00               LDY  #$00
4080:  C708 60                  RTS
4082:  C709
4082:  C709
4084:                   ;BYTE IN HET GEHEUGEN SCHRIJVEN
4086:  C709
4086:  C709
4090:  C709 91 22       STOBYT  STA  (TEMP2),Y
4100:  C70B E6 22               INC  TEMP2    ;TELLER VERHOGEN
4110:  C70D D0 02               BNE  NOINC
4120:  C70F E6 23               INC  TEMP2+1
4130:  C711 58          NOINC   CLI
4140:  C712 60                  RTS
4142:  C713
4142:  C713
4144:                   ;ASCII-BYTE HALEN EN IN HEXGETAL
4145:                   ;       OMZETTEN
4146:  C713
4146:  C713
4150:  C713 20 A5 FF ASCHEX     JSR  IECIN
4160:  C716 29 0F               AND  #$0F
4170:  C718 0A                  ASL  A
4180:  C719 0A                  ASL  A
4190:  C71A 0A                  ASL  A
4200:  C71B 0A                  ASL  A
4210:  C71C 85 57               STA  $57
4220:  C71E 20 A5 FF            JSR  IECIN
4230:  C721 29 0F               AND  #$0F
4240:  C723 05 57               ORA  $57
4250:  C725 60                  RTS
4252:  C726
4252:  C726
4254:                   ;HEXGETAL IN ASCII-BYTE OMZETTEN
4255:                   ;     EN UITVOEREN
4256:  C726
4256:  C726
4260:  C726 48          HEXASC  PHA
4270:  C727 4A                  LSR  A
4280:  C728 4A                  LSR  A
4290:  C729 4A                  LSR  A
4300:  C72A 4A                  LSR  A
4310:  C72B 20 2F C7            JSR  HALBBT
4320:  C72E 68                  PLA
4330:  C72F 29 0F       HALBBT  AND  #$0F
4340:  C731 18                  CLC
4350:  C732 69 30               ADC  #$30
4360:  C734 4C D2 FF            JMP  BASOUT   ;UITVOEREN
```

```
4362:  C737
4362:  C737
4364:                    ;FOUTMELDING UITVOEREN
4366:  C737
4366:  C737
4370:  C737 AD 11 D0 FOUTML   LDA  $D011    ;(REGELS 4370-4390 VERVALLEN)
4380:  C73A 09 10             ORA  #$10     ;(MELD KOMT IN REGEL 4400)
4390:  C73C 8D 11 D0          STA  $D011
4400:  C73F A9 00             LDA  #<DRGAN  ;"DOORGAAN ? "
4410:  C741 A0 CA             LDY  #>DRGAN
4420:  C743 20 1E AB          JSR  STROUT   ;UITVOEREN
4430:  C746 20 E1 C7 FLOOP    JSR  INPUT    ;WACHT EEN TOETS
4440:  C749 C9 59             CMP  #$59
4450:  C74B D0 01             BNE  NEE2
4460:  C74D 60                RTS
4470:  C74E C9 4E    NEE2     CMP  #$4E
4480:  C750 D0 F4             BNE  FLOOP
4490:  C752 4C D0 C7          JMP  EINDE
4492:  C755
4492:  C755
4494:                    ;VOGENDE FILE UITVOEREN
4496:  C755
4496:  C755
4500:  C755 A9 0E    NAAMUI   LDA  #<SNFILE  ;"VOLGENDE FILE "
4510:  C757 A0 CA             LDY  #>SNFILE
4520:  C759 20 1E AB          JSR  STROUT   ;UITVOEREN
4530:  C75C A5 FC             LDA  TEMP     ;POS IN TABEL
4540:  C75E 0A                ASL  A        ;BEREKENEN
4550:  C75F 0A                ASL  A
4560:  C760 0A                ASL  A
4570:  C761 0A                ASL  A
4580:  C762 A6 FC             LDX  TEMP
4590:  C764 BC 40 CA          LDY  LNGTAB,X
4600:  C767 AA                TAX
4610:  C768 BD 00 CB WRNAM3   LDA  NAMTAB,X  ;FILENAAM UITVOEREN
4620:  C76B 20 D2 FF          JSR  BASOUT
4630:  C76E E8                INX
4640:  C76F 88                DEY
4650:  C770 D0 F6             BNE  WRNAM3
4660:  C772 60                RTS
4662:  C773
4662:  C773
4664:                    ;DISKETTE OP 'LANGZAAM' SCHAKELEN
4665:                    ;  EN BEELDSCHERM AAN
4666:  C773
4666:  C773
4670:  C773 AD 11 D0 SEUIP    LDA  $D011    ;(REGEL 4670-4820 VERVALT)
4680:  C776 29 EF             AND  #$EF     ;BEELDSCHERM AAN
4690:  C778 8D 11 D0          STA  $D011
4700:  C77B A9 0F             LDA  #$0F
4710:  C77D 20 C3 FF          JSR  CLOSE
4720:  C780 A9 0F             LDA  #$0F
4730:  C782 A2 08             LDX  #$08
4740:  C784 A8                TAY
4750:  C785 20 BA FF          JSR  SETLFS
4760:  C788 A9 03             LDA  #$03
4770:  C78A A2 D4             LDX  #<UIPLU
4780:  C78C A0 C9             LDY  #>UIPLU
4790:  C78E 20 BD FF          JSR  SETNAM
4800:  C791 20 C0 FF          JSR  OPEN
4810:  C794 A9 0F             LDA  #$0F
4820:  C796 4C C3 FF          JMP  CLOSE
4822:  C799
```

```
4822:  C799
4824:                    ;  WACHTLUS
4826:  C799
4826:  C799
4830:  C799 20 E4 FF WAIT      JSR  GETBYT
4840:  C79C F0 FB              BEQ  WAIT
4850:  C79E 60                 RTS
4852:  C79F
4852:  C79F
4854:                    ;DISK INITIALISEREN
4856:  C79F
4856:  C79F
4860:  C79F A9 0F     INIT     LDA  #$0F     ;COMMANDOKANAAL
4870:  C7A1 A2 08              LDX  #$08     ;OPENEN
4880:  C7A3 A8                 TAY
4890:  C7A4 20 BA FF           JSR  SETLFS
4900:  C7A7 A9 01              LDA  #$01     ; "I"
4910:  C7A9 A2 D8              LDX  #<STRI
4920:  C7AB A0 C9              LDY  #>STRI
4930:  C7AD 20 BD FF           JSR  SETNAM
4940:  C7B0 4C C0 FF           JMP  OPEN
4942:  C7B3
4942:  C7B3
4944:                    ;SPACE RESP. CARRIAGE RETURN ZENDEN
4946:  C7B3
4946:  C7B3
4950:  C7B3 A9 20     SPACE    LDA  #$20     ;SPACE
4960:  C7B5 2C                 .BYT $2C
4970:  C7B6 A9 0D     CARET    LDA  #CR      ;CARRIAGE RETURN
4980:  C7B8 4C D2 FF           JMP  BASOUT   ;ZENDEN
4982:  C7BB
4982:  C7BB
4984:                    ;UITVOERROUTINE
4986:  C7BB
4986:  C7BB
4990:  C7BB 20 D2 FF UITVOR    JSR  BASOUT
5000:  C7BE 8A       STOETS    TXA           ;A,X,Y REGISTER REDDEN
5010:  C7BF 48                 PHA
5020:  C7C0 98                 TYA
5030:  C7C1 48                 PHA
5040:  C7C2 20 E1 FF           JSR  STOP     ;STOPTOETS UITLEZEN
5050:  C7C5 18                 CLC
5060:  C7C6 D0 01              BNE  VERDER
5070:  C7C8 38                 SEC
5080:  C7C9 68       VERDER    PLA           ;REGISTER HERSTELLEN
5090:  C7CA A8                 TAY
5100:  C7CB 68                 PLA
5110:  C7CC AA                 TAX
5120:  C7CD B0 01     STOP2    BCS  EINDE
5130:  C7CF 60        RETURN   RTS
5132:  C7D0
5132:  C7D0
5134:                    ;        AFBREKEN
5136:  C7D0
5136:  C7D0
5140:  C7D0 A2 F6     EINDE    LDX  #$F6
5150:  C7D2 9A                 TXS
5160:  C7D3 20 E5 C6           JSR  SCRON    ;(VERVALT)
5170:  C7D6 A9 0F              LDA  #$0F
5180:  C7D8 20 C3 FF           JSR  CLOSE
5190:  C7DB 20 E7 FF           JSR  CLALL
```

```
5200:  C7DE 4C 00 C4          JMP  START
5202:  C7E1
5202:  C7E1
5204:                  ;  INVOERROUTINE
5206:  C7E1
5206:  C7E1
5210:  C7E1 20 E4 FF INPUT    JSR  GETBYT
5220:  C7E4 D0 E9             BNE  RETURN
5230:  C7E6 20 BE C7          JSR  STOETS
5240:  C7E9 90 F6             BCC  INPUT
5250:  C7EB A2 0F     FOUT    LDX  #$0F     ;STATUS UITLEZEN
5260:  C7ED 20 0F F3          JSR  SFINUM
5270:  C7F0 20 1F F3          JSR  SETPAR
5280:  C7F3 A5 BA             LDA  DN
5290:  C7F5 20 B4 FF          JSR  TALK
5300:  C7F8 A5 B9             LDA  SA
5310:  C7FA 20 96 FF          JSR  SETALK
5320:  C7FD 20 13 C7          JSR  ASCHEX
5330:  C800 C9 20             CMP  #$20
5340:  C802 08                PHP
5350:  C803 90 0B             BCC  MELUIT   ;EN BIJ FOUT
5360:  C805 48                PHA           ;UITVOEREN
5370:  C806 20 B6 C7          JSR  CARET
5380:  C809 20 B6 C7          JSR  CARET
5390:  C80C 68                PLA
5400:  C80D 20 26 C7          JSR  HEXASC
5410:  C810 20 A5 FF MELUIT   JSR  IECIN
5420:  C813 C9 0D             CMP  #CR
5430:  C815 F0 09             BEQ  MELEND
5440:  C817 28                PLP
5450:  C818 08                PHP
5460:  C819 90 F5             BCC  MELUIT
5470:  C81B 20 D2 FF          JSR  BASOUT
5480:  C81E 90 F0             BCC  MELUIT
5490:  C820 20 AB FF MELEND   JSR  UNTALK
5500:  C823 28                PLP
5500:  C824 60                RTS
5502:  C825 20 38 F8 TOETS    JSR  PTOETS
5502:  C828 20 CD C7          JSR  STOP2
5502:  C82B D0 F8             BNE  TOETS
5504:  C82D A9 07             LDA  #7
5504:  C82F 85 C0             STA  MOFLAG
5506:  C831 4C CA FC          JMP  MOTUIT
5511:  C834
5511:  C834
5511:  C834
5512:                  ; UITVOER STRINGS
5513:  C834
5513:  C834
5513:  C834
5520:  C834 0D 0D 0D ORGIN    .BYT 13,13,13
5530:  C837 12 4F 52          .ASC "■ORIGINEEL■ INLEGGEN !"
5540:  C84D 00                .BYT 0
5550:  C84E 0D 0D     ZICASS  .BYT 13,13
5560:  C850 12 43 41          .ASC "■CASSETTE■ INLEGGEN !"
5570:  C865 0D 00             .BYT 13,0
5580:  C867 93 11 1C TITEL    .ASC "■■■■       BACKUP VAN DISKETTE NAAR"
5590:  C889 0D 0D             .BYT 13,13
5600:  C88B 1F 20 20          .ASC "■              CASSETTE"
5602:  C8A2 0D 0D             .BYT 13,13
5604:  C8A4 20 20 20          .ASC "         MET ■■F A S T T A P E ■■"
5606:  C8C3 0D 0D 0D          .BYT 13,13,13
5608:  C8C6 20 20 20          .ASC "    (C)    DIRK PAULISSEN  "
```

```
5610:  C8E0 0D 0D 0D          .BYT 13,13,13
5620:  C8E3 1E 20 20          .ASC "■           E = EINDE■"
5630:  C8F9 00                .BYT 0
5640:  C8FA 0D        SLONG   .BYT 13
5650:  C8FB 1C 12 21          .ASC "■■!!■ TE LANG OM TE KOPIEREN ■"
5660:  C919 00                .BYT 0
5670:  C91A 0D        LILONG  .BYT 13
5680:  C91B 1C 12 21          .ASC "■■!!■ LIJST IS VOL ■"
5690:  C92F 00                .BYT 0
5700:  6930 12 4A 41 JANEE    .ASC "■JA/NEE■■■■■■■■■■■■"
5710:  C940 00                .BYT 0
5720:  C941 12 1E 20 STRJA    .ASC "■■  JA  ■■■"
5730:  C94C 00                .BYT 0
5740:  C94D 0D        STRILL  .BYT 13
5750:  C94E 1C 12 21          .ASC "■■!!■ VERKEERD FILETYPE■"
5760:  C966 00                .BYT 0
5770:  C967 1C 12 20 STRNEE   .ASC "■■ NEE ■■■"
5780:  C971 00                .BYT 0
5790:  C972 0D 0D 0D VRAAG    .BYT 13,13,13
5800:  C975 57 49 4C          .ASC "WILT U DE PROGRAMMA'S"
5810:  C98A 0D 0D             .BYT 13,13
5820:  C98C 20 20 20          .ASC "    1    CONTINU"
5830:  C99C 0D 0D             .BYT 13,13
5840:  C99E 20 20 20          .ASC "    2    EEN VOOR EEN"
5850:  C9B3 0D 0D             .BYT 13,13
5860:  C9B5 4B 4F 50          .ASC "KOPIEREN ?"
5870:  C9BF 00                .BYT 0
5880:  C9C0 0D 0D    SREAD    .BYT 13,13
5890:  C9C2 52 45 41          .ASC "READING "
5900:  C9CA 00                .BYT 0
5910:  C9CB 2C 50 2C PRGRE    .ASC ",P,R"
5920:  C9CF 00                .BYT 0
5930:  C9D0 55 49 2D UIMIN    .ASC "UI-"
5940:  C9D3 00                .BYT 0
5950:  C9D4 55 49 2B UIPLU    .ASC "UI+"
5960:  C9D7 00                .BYT 0
5970:  C9D8 49        STRI    .ASC "I"
5980:  C9D9 00                .BYT 0
5985:  C9DA 24        CATALO  .ASC "$"
5986:  C9DB 00                .BYT 0
5990 : C9DD 0D 0D     WRITI   .BYT 13,13
6000 : C9DF 53 41 56          .ASC "SAVING"
6010 : C9E6 00                .BYT 0
6020 : C9E7 0D D0     EOCOP   .BYT 13,13
6030 : C9E9 1C 12 45          .ASC"■■EINDE V/H KOPIEREN ■■"
6040 : CA00 00                .BYT 0
6050 : CA01 0D        DRGAN   .BYT 13
6060 : CA02 20 44 4F          .ASC" DOORGAAN ?"
6070 : CA0F 00                .BYT 0
6080 : CA10 0D 0D     SNFILE  .BYT 13,13
6090 : CA12 56 4F 47          .ASC"VOLGENDE FILE:"
6100 : CA21 00                .BYT 0
```

```
100 REM BACKUP DISKETTE => CASSETTE VOOR
 C-64
110 E=51757:A=50176:PS=0
120 FORI=ATOE:READX:POKEI,X:PS=PS+X:NEXT
130 IFPS<>171785THENPRINT"FOUT IN DATA":
END
140 SYSA:NEW
150 DATA169,1,141,32,208,141,33,208,169,
6,141,134,2,169,107,160,200,32,30,171
160 DATA169,52,160,200,32,30,171,32,153,
199,201,69,208,1,96,32,182,199,32,182
170 DATA199,32,159,199,169,8,170,160,0,3
2,186,255,169,1,162,225,160,201,32
180 DATA189,255,32,192,255,169,8,32,180,
255,169,0,32,150,255,160,4,32,165,255
190 DATA136,208,250,32,165,255,133,34,32
,165,255,166,34,32,205,189,32,179,199
200 DATA32,165,255,240,6,32,210,255,24,1
44,245,32,182,199,32,182,199,32,165
210 DATA255,32,165,255,160,0,132,251,32,
165,255,133,252,32,165,255,133,253
220 DATA166,252,32,205,189,32,179,199,16
0,0,32,165,255,72,32,187,199,104,153
230 DATA64,3,240,3,200,208,240,32,165,25
5,32,165,255,165,144,240,3,76,71,197
240 DATA165,253,208,6,165,252,201,152,14
4,10,169,252,160,200,32,30,171,76,65
250 DATA197,165,251,201,48,144,10,169,27
,160,201,32,30,171,24,144,237,169,0
260 DATA133,8,169,31,133,211,169,51,160,
201,32,30,171,32,225,199,201,78,240
270 DATA82,201,74,208,245,169,69,160,201
,32,30,171,165,251,32,241,198,162,0
280 DATA232,189,64,3,201,34,208,248,134,
253,232,189,64,3,201,34,240,6,145,65
290 DATA232,200,208,243,138,164,251,24,2
29,253,153,64,202,189,64,3,208,10,169
300 DATA81,160,201,32,30,171,76,65,197,2
01,83,240,242,201,80,240,3,232,208
310 DATA230,230,251,24,144,7,169,107,160
,201,32,30,171,32,182,199,76,125,196
320 DATA169,8,32,195,255,165,251,208,3,7
6,0,196,169,119,160,201,32,30,171,32
330 DATA225,199,201,49,240,7,201,50,208,
245,169,0,44,169,255,133,254,169,15
340 DATA32,195,255,162,0,134,252,169,199
,160,201,32,30,171,164,252,190,64,202
350 DATA165,252,32,241,198,177,65,32,210
,255,200,202,208,247,169,2,162,8,168
```

```
360 DATA32,186,255,166,252,189,64,202,13
3,34,165,252,32,241,198,162,0,177,65
370 DATA157,64,3,200,232,198,34,208,245,
160,0,185,210,201,157,64,3,200,232
380 DATA192,4,144,244,138,162,64,160,3,3
2,189,255,32,192,255,169,9,160,0,132
390 DATA34,133,35,173,17,208,41,239,141,
17,208,169,15,162,8,168,32,186,255
400 DATA169,3,162,215,160,201,32,189,255
,32,192,255,162,2,32,15,243,32,31,243
410 DATA165,186,32,180,255,165,185,32,15
0,255,160,0,32,165,255,32,9,199,166
420 DATA144,240,246,32,235,199,8,169,2,3
2,195,255,173,17,208,9,16,141,17,208
430 DATA40,144,3,32,55,199,166,252,165,3
4,133,174,165,35,133,175,232,165,252
440 DATA208,13,169,78,160,200,32,30,171,
32,225,199,32,37,200,169,227,160,201
450 DATA32,30,171,164,252,190,64,202,165
,252,32,241,198,177,65,32,210,255,200
460 DATA202,208,247,162,1,169,0,168,32,1
86,255,166,252,189,64,202,133,34,165
470 DATA252,32,241,198,162,0,177,65,157,
64,3,232,200,198,34,208,245,138,162
480 DATA64,160,3,32,189,255,160,0,169,9,
132,172,133,173,177,172,133,167,200
490 DATA177,172,133,168,200,132,172,165,
174,56,229,172,8,24,101,167,133,169
500 DATA165,175,101,168,40,229,173,133,1
70,162,5,134,171,32,133,192,32,229
510 DATA198,230,252,166,252,228,251,176,
7,36,254,16,16,76,223,198,169,237,160
520 DATA201,32,30,171,32,225,199,76,0,19
6,32,85,199,169,78,160,200,32,30,171
530 DATA32,225,199,32,37,200,32,182,199,
76,117,197,32,115,199,173,17,208,9
540 DATA16,141,17,208,96,160,0,10,10,132
,66,10,38,66,10,38,66,133,65,165,66
550 DATA24,105,203,133,66,160,0,96,145,3
4,230,34,208,2,230,35,88,96,32,165
560 DATA255,41,15,10,10,10,10,133,87,32,
165,255,41,15,5,87,96,72,74,74,74,74
570 DATA32,47,199,104,41,15,24,105,48,76
,210,255,173,17,208,9,16,141,17,208
580 DATA169,7,160,202,32,30,171,32,225,1
99,201,89,208,1,96,201,78,208,244,76
590 DATA208,199,169,26,160,202,32,30,171
,165,252,10,10,10,10,166,252,188,64
600 DATA202,170,189,0,203,32,210,255,232
```

```
,136,208,246,96,173,17,208,41,239,141
610 DATA17,208,169,15,32,195,255,169,15,
162,8,168,32,186,255,169,3,162,219
620 DATA160,201,32,189,255,32,192,255,16
9,15,76,195,255,32,228,255,240,251
630 DATA96,169,15,162,8,168,32,186,255,1
69,1,162,223,160,201,32,189,255,76
640 DATA192,255,169,32,44,169,13,76,210,
255,32,210,255,138,72,152,72,32,225
650 DATA255,24,208,1,56,104,168,104,170,
176,1,96,162,246,154,32,229,198,169
660 DATA15,32,195,255,32,231,255,76,0,19
6,32,228,255,208,233,32,190,199,144
670 DATA246,162,15,32,15,243,32,31,243,1
65,186,32,180,255,165,185,32,150,255
680 DATA32,19,199,201,32,8,144,11,72,32,
182,199,32,182,199,104,32,38,199,32
690 DATA165,255,201,13,240,9,40,8,144,24
5,32,210,255,144,240,32,171,255,40
700 DATA96,32,56,248,32,205,199,208,248,
169,7,133,192,76,202,252,13,13,13,18
710 DATA81,85,69,76,76,68,73,83,75,146,3
2,69,73,78,76,69,71,69,78,32,33,0,13
720 DATA13,18,90,73,69,76,67,65,83,83,69
,84,84,69,146,32,69,73,78,76,69,71
730 DATA69,78,32,33,13,0,147,17,28,32,32
,32,32,32,32,32,32,66,65,67,75,85,80,32
740 DATA86,79,78,32,68,73,83,75,69,84,84
,69,32,65,85,70,13,13,31,32,32,32,32
750 DATA32,32,32,32,32,32,32,32,32,32,67
,65,83,83,69,84,84,69,13,13,32,32,32
760 DATA32,32,32,32,32,77,73,84,32,18,12
9,70,32,65,32,83,32,84,32,84,32,65
770 DATA32,80,32,69,32,146,13,13,13,32,3
2,32,40,67,41,32,32,32,32,68,73,82
780 DATA75,32,80,65,85,76,73,83,83,69,78
,32,32,13,13,13,30,32,32,32,32,32,32
790 DATA32,32,32,32,32,69,32,61,32,69,78
,68,69,31,0,13,28,18,33,33,18,32,90
800 DATA85,32,76,65,78,71,69,32,90,85,77
,32,67,79,80,73,69,82,69,78,32,31,0
810 DATA13,28,18,33,33,18,32,76,73,83,84
,69,32,73,83,84,32,86,79,76,76,32,31
820 DATA0,18,74,65,47,78,69,73,78,157,15
7,157,157,157,157,157,146,31,0,18,30
830 DATA32,32,74,65,32,32,146,32,31,0,13
,28,18,33,33,18,32,73,76,76,69,71,65
840 DATA76,69,82,32,70,73,76,69,84,89,80
,31,0,28,18,32,78,69,73,78,32,146,32
```

```
850 DATA31,0,13,13,13,87,79,76,76,69,78,
32,83,73,69,32,68,73,69,32,80,82,79
860 DATA71,82,65,77,77,69,13,13,32,32,32
,32,49,32,32,32,32,68,85,82,67,72,71
870 DATA69,72,69,78,68,13,13,32,32,32,32
,50,32,32,32,32,69,73,78,90,69,76,78
880 DATA13,13,67,79,80,73,69,82,69,78,32
,63,0,13,13,82,69,65,68,73,78,71,32
890 DATA0,44,80,44,82,0,85,73,45,0,85,73
,43,0,73,0,36,0,13,13,83,65,86,73,78
900 DATA71,32,0,13,13,28,18,69,78,68,69,
32,68,69,83,32,67,79,80,73,69,82,69
910 DATA78,83,32,146,31,0,13,32,87,69,73
,84,69,82,32,77,65,67,72,69,78,32,63
920 DATA32,0,13,13,78,65,69,67,72,83,84,
69,83,32,70,73,76,69,32,58,32,0
```

```
100 REM BACKUP DISKETTE => CASSETTE VOOR
 VC20
110 E=31156:A=29696:PS=0
120 FORI=ATOE:READX:POKEI,X:PS=PS+X:NEXT
130 IFPS<>145927THENPRINT"FOUT IN DATA":
END
140 SYSA:NEW
150 DATA169,25,141,15,144,169,6,141,134,
2,169,19,160,120,32,30,203,169,221
160 DATA160,119,32,30,203,32,69,119,201,
69,208,1,96,32,98,119,32,98,119,32
170 DATA75,119,169,8,170,160,0,32,186,25
5,169,1,162,103,160,121,32,189,255
180 DATA32,192,255,169,8,32,180,255,169,
0,32,150,255,160,4,32,165,255,136,208
190 DATA250,32,165,255,133,34,32,165,255
,166,34,32,205,221,32,95,119,32,165
200 DATA255,240,6,32,210,255,24,144,245,
32,98,119,32,98,119,32,165,255,32,165
210 DATA255,160,0,132,3,32,165,255,133,4
,32,165,255,133,5,166,4,32,205,221
220 DATA32,95,119,160,0,32,165,255,72,32
,103,119,104,153,64,3,240,3,200,208
230 DATA240,32,165,255,32,165,255,165,14
4,240,3,76,64,117,165,5,208,6,165,4
240 DATA201,93,144,10,169,133,160,120,32
```

```
,30,203,76,58,117,165,3,201,48,144
250 DATA10,169,164,160,120,32,30,203,24,
144,237,169,0,133,8,169,188,160,120
260 DATA32,30,203,32,138,119,201,78,240,
82,201,74,208,245,169,206,160,120,32
270 DATA30,203,165,3,32,203,118,162,0,23
2,189,64,3,201,34,208,248,134,5,232
280 DATA189,64,3,201,34,240,6,145,65,232
,200,208,243,138,164,3,24,229,5,153
290 DATA64,122,189,64,3,208,10,169,218,1
60,120,32,30,203,76,58,117,201,83,240
300 DATA242,201,80,240,3,232,208,230,230
,3,24,144,7,169,244,160,120,32,30,203
310 DATA32,98,119,76,122,116,169,8,32,19
5,255,165,3,208,3,76,0,116,169,0,160
320 DATA121,32,30,203,32,138,119,201,49,
240,7,201,50,208,245,169,0,44,169,255
330 DATA133,5,169,15,32,195,255,162,0,13
4,4,169,77,160,121,32,30,203,164,4
340 DATA190,64,122,165,4,32,203,118,177,
65,32,210,255,200,202,208,247,169,2
350 DATA162,8,168,32,186,255,166,4,189,6
4,122,133,34,165,4,32,203,118,162,0
360 DATA177,65,157,64,3,200,232,198,34,2
08,245,160,0,185,88,121,157,64,3,200
370 DATA232,192,4,144,244,138,162,64,160
,3,32,189,255,32,192,255,169,19,160
380 DATA0,132,34,133,35,169,15,162,8,168
,32,186,255,169,3,162,93,160,121,32
390 DATA189,255,32,192,255,162,2,32,207,
243,32,223,243,165,186,32,180,255,165
400 DATA185,32,150,255,160,0,32,165,255,
32,227,118,166,144,240,246,32,148,119
410 DATA8,169,2,32,195,255,40,144,3,32,1
7,119,166,4,165,34,133,174,165,35,133
420 DATA175,232,165,4,208,13,169,247,160
,119,32,30,203,32,138,119,32,206,119
430 DATA169,105,160,121,32,30,203,164,4,
190,64,122,165,4,32,203,118,177,65
440 DATA32,210,255,200,202,208,247,162,1
,169,0,168,32,186,255,166,4,189,64
450 DATA122,133,34,165,4,32,203,118,162,
0,177,65,157,64,3,232,200,198,34,208
460 DATA245,138,162,64,160,3,32,189,255,
160,0,169,19,132,172,133,173,177,172
470 DATA133,167,200,177,172,133,168,200,
132,172,165,174,56,229,172,8,24,101
480 DATA167,133,169,165,175,101,168,40,2
29,173,133,170,162,5,134,171,32,127
```

```
490 DATA112,230,4,166,4,228,3,176,7,36,5
,16,16,76,197,118,169,115,160,121,32
500 DATA30,203,32,138,119,76,0,116,32,39
,119,169,247,160,119,32,30,203,32,138
510 DATA119,32,206,119,32,98,119,76,110,
117,160,0,10,10,132,66,10,38,66,10
520 DATA38,66,133,65,165,66,24,105,123,1
33,66,160,0,96,145,34,230,34,208,2
530 DATA230,35,88,96,32,165,255,41,15,10
,10,10,10,133,87,32,165,255,41,15,5
540 DATA87,96,72,74,74,74,74,32,9,119,10
4,41,15,24,105,48,76,210,255,169,141
550 DATA160,121,32,30,203,32,138,119,201
,89,208,1,96,201,78,208,244,76,124
560 DATA119,169,160,160,121,32,30,203,16
5,4,10,10,10,10,166,4,188,64,122,170
570 DATA189,0,123,32,210,255,232,136,208
,246,96,32,228,255,240,251,96,169,15
580 DATA162,8,168,32,186,255,169,1,162,1
01,160,121,32,189,255,76,192,255,169
590 DATA32,44,169,13,76,210,255,32,210,2
55,138,72,152,72,32,225,255,24,208
600 DATA1,56,104,168,104,170,176,1,96,16
2,246,154,169,15,32,195,255,32,231
610 DATA255,76,0,116,32,228,255,208,236,
32,106,119,144,246,162,15,32,207,243
620 DATA32,223,243,165,186,32,180,255,16
5,185,32,150,255,32,237,118,201,32
630 DATA8,144,11,72,32,98,119,32,98,119,
104,32,0,119,32,165,255,201,13,240
640 DATA9,40,8,144,245,32,210,255,144,24
0,32,171,255,40,96,32,183,248,32,121
650 DATA119,208,248,169,7,133,192,76,8,2
53,13,13,13,18,81,85,69,76,76,68,73
660 DATA83,75,146,32,69,73,78,76,69,71,6
9,78,32,33,0,13,13,18,90,73,69,76,67
670 DATA65,83,83,69,84,84,69,146,32,69,7
3,78,76,69,71,69,78,33,13,0,147,17
680 DATA28,32,32,66,65,67,75,85,80,32,86
,79,78,32,68,73,83,75,69,84,84,69,13
690 DATA13,31,32,32,32,32,32,32,65,85,70
,32,67,65,83,83,69,84,84,69,13,13,77
700 DATA73,84,32,18,129,70,32,65,32,83,3
2,84,32,84,32,65,32,80,32,69,32,146
710 DATA13,13,13,32,40,67,41,32,32,32,32
,68,73,82,75,32,80,65,85,76,73,83,83
720 DATA69,78,13,13,13,30,32,32,32,32,32
,69,32,61,32,69,78,68,69,31,0,13,28
730 DATA18,33,33,18,32,90,85,32,76,65,78
```

```
,71,69,32,90,85,77,32,67,79,80,73,69
740 DATA82,69,78,32,31,0,13,28,18,33,33,
18,32,76,73,83,84,69,32,73,83,84,32
750 DATA86,79,76,76,32,31,0,18,74,65,47,
78,69,73,78,157,157,157,157,157,157
760 DATA157,146,31,0,18,30,32,32,74,65,3
2,32,146,32,31,0,13,28,18,33,33,18
770 DATA32,73,76,76,69,71,65,76,69,82,32
,70,73,76,69,84,89,80,31,0,28,18,32
780 DATA78,69,73,78,32,146,32,31,0,13,13
,13,87,79,76,76,69,78,32,83,73,69,32
790 DATA68,73,69,32,13,80,82,79,71,82,65
,77,77,69,13,13,32,32,49,32,32,32,32
800 DATA68,85,82,67,72,71,69,72,69,78,68
,13,13,32,32,50,32,32,32,32,69,73,78
810 DATA90,69,76,78,13,13,67,79,80,73,69
,82,69,78,32,63,0,13,13,82,69,65,68
820 DATA73,78,71,32,0,44,80,44,82,0,85,7
3,45,0,85,73,43,0,73,0,36,0,13,13,83
830 DATA65,86,73,78,71,32,0,13,13,28,18,
69,78,68,69,32,68,69,83,32,67,79,80
840 DATA73,69,82,69,78,83,32,146,31,0,13
,32,87,69,73,84,69,82,32,77,65,67,72
850 DATA69,78,32,63,32,0,13,13,78,65,69,
67,72,83,84,69,83,32,70,73,76,69,32
860 DATA58,32,13,00
```

15.4 Programmabeschrijving back-up van disk naar cassette

Nu volgt nog een beschrijving van dit lange programma.

Opmerking vooraf:

Dit programma werkt met twee verschillende startadressen voor laden of opslaan. Het bufferstartadres is het adres van de buffer waarin het programma tussentijds werd opgeslagen om het te kunnen kopiëren. Het werkadres is het startadres vanwaar het programma gewoonlijk wordt geladen en werkt.

Het principe van het programma is als volgt:

Na de diskette-initialisering worden eerst de filenamen ingelezen en in de cassettebuffer tussentijds opgeslagen vanaf $0340. Daarna

volgt de vraag of het programma moet worden gekopieerd. Antwoordt u met "J", dan wordt getest of de maximale bloklengte, respectievelijk de lengte van de file-namenlijst wordt overschreden. Bovendien wordt, indien nodig, een foutmelding uitgevoerd. Wordt er geen overschrijding geconstateerd, dan wordt de filenaam en zijn lengte in twee tabellen (NAMTAB, LNGTAB) achter het back up-programma aan overgedragen. Verder test het programma ook het filetype van de te kopieren file. Gaat het hierbij niet om een programmafile, dan wordt een foutmelding gegeven. Als op deze manier de inhoudsopgave is verwerkt, wordt naar de kopieerroutine teruggesprongen. Na uitvoer van "READING naam", wordt de desbetreffende filenaam weer in de cassettebuffer overgebracht en aan ",P,R" gekoppeld. Met de zo uitgebreide filenaam wordt een leesbestand geopend. Het betreffende programma wordt na $0900 (VC 20 = $1300), het laadadres van het buffer-programma (BUFADR), geladen. Wanneer bij het lezen geen fouten optreden, worden het programma-werkadres (dat in de programmakop moet worden geschreven) en het programma-bufferadres (waaronder het programma voorlopig werd opgeslagen) aan de geheugenroutine van de fast-tape doorgegeven. Daarna wordt de fast-tape routine "ABSOLUT" vertakt. Vervolgens wordt weer naar de start teruggesprongen als er nog meer files moeten worden gekopieerd. De ASSEMBLER-listing is van veel commentaar voorzien om het programmaverloop te verduidelijken.

Tenslotte een laatste opmerking.

De diskdrive kan met twee overdrachtssnelheden werken. Een snelle voor de VIC 20 en een langzame voor de Commodore 64, omdat deze door de andere beeldschermsturing langzamer werkt. Wanneer u het beeldscherm bij het laden uitschakelt, werkt de Commodore 64 net zo snel als de VIC 20. Daarom vervallen bij de VIC 20 de betreffende omschakelingen in de listings.

BIJLAGE

A Belangrijke geheugenplaatsen

hex/decimaal	betekenis
+0001/1	uitvoerregister van de CPU
	bit3 schrijflijn voor de cassetterecorder
	bit4 CASS-SENS, status recordertoetsen
	bit5 motorsturing
000A/10	LOAD/VERIFY-flag; 0=LOAD 1=VERIFY
002B-002C/43-44	wijzer BASIC-RAM-start
002D-002E/45-46	wijzer einde BASIC-programma /start variabelen
002F-0030/47-48	wijzer begin van de velden
0031-0032/49-50	wijzer eind van de velden
0033-0034/51-52	wijzer eind van de strings
0037-0038/55-56	wijzer eind van het BASIC-RAM
0090/144	statusbyte
0093/147	LOAD/VERIFY; 0=LOAD/1=VERIFY
0098/152	aantal open files
0099/153	invoerapparaat
009A/154	uitvoerapparaat
00A6/166	wijzer in cassettebuffer
00AC-00AD/172-173	wijzer op actuele byte bij het schrijven en lezen
00AE-00AF/174-175	wijzer op programma-einde bij het lezen/schrijven

00B2-00B3/178-179
wijzer op start van de cassettebuffer

00B7/183 lengte filenaam

00B8/184 actuele logische filenummer

00B9/185 actueel secundair adres

00BA/186 actueel apparaatadres

00BE/190 pass-teller bij het lezen en schrijven

00BD/191 in-/uitvoerbyte

00C0/192 cassettemotor flag

00C1-00C2/193-194
in-/uitvoer startadres

00C3-00C4/195-196
in-/uitvoer eindadres

033C-03FB/828-1019
cassettebuffer

033C/828 filetype

033D-033E/829-830
programma startadres

033F-0340/831-832
programma eindadres

0341-/833- filenaam

*911C/37148 bit5 schakelaar cassettemotor

*911F/37151 bit6 CASS-SENS; status recordertoetsen

*9120/37152 bit3 schrijflijn

+ : geldt alleen voor Commodore 64
* : geldt alleen voor VIC 20